Introduction to Website Design and Development: HTML5, CSS3, and JavaScript Fourth Edition

Don Colton
Brigham Young University–Hawai‘i

February 27, 2016

Contents

Unit I

Web Design 101

Foreword

The first part of this book is divided into chapters, each of which is roughly intended as the topic for one class period. The appendix contains additional material and provides a comprehensive index.

Many of the chapters include questions that students should be ready to answer in class. The questions are designed to let the teacher assess the student's reading effectiveness. With each question an answer is given. Sometimes, like with a yes/no question, a specific answer is required, and you must give the answer that is shown in the book. No other answer will be accepted. In most cases an acceptable (but not required) answer is given, by which we seek to balance between brevity and completeness. For these questions students are welcome to answer in their own words.

The back part of the book is an appendix. It contains material that goes beyond the 101 level. It also contains a list of the test questions that appear throughout the book. And it has a glossary of terms. The final part of the appendix is a comprehensive index of the important topics covered in the book.

Learning Objectives

This textbook provides support for the following learning objectives. By the conclusion of this course, students should be able to do these things.

- Learning objective. (How we achieve it.)
- Properly use HTML markup. (We cover h1, p, links, div, span, head, body, tables, lists, and forms.)
- Properly use CSS to style a webpage. (We cover box model, font

families, inline style, positioning, internal and external style sheets based on tag, ID, class, and pseudo-class.)

- Properly segregate HTML and CSS.
- Create valid HTML and CSS. (We validate our HTML and CSS against W3C standards.)
- Edit images. (We use Gimp to crop, resize, use transparency, and create icons.)
- Understand JavaScript. (We use it to dynamically alter the appearance of a webpage.)
- Understand Webhosting and DNS. (We establish a domain name and subdomains and populate them with content.)
- Understand Apache. (We use public html and index.html to create websites.)
- Understand CMS. (We install and customize WordPress, a popular Content Management System.)

Acknowledgements

I am pleased to acknowledge and express my thanks for the following important contributions.

W3C, the World Wide Web Consortium (http://www.w3.org/), in its role as an international community for developing open standards to ensure the long-term growth of the Web, provides clearly documented standards that are the basis of the modern Web.

You, the students who will read this book and apply its teachings, provide motivation and context in which my efforts have meaning.

Chapter 1

First Webpage

Contents

The language used for writing webpages is called HTML.

In this chapter we introduce HTML by having you build a simple webpage. We introduce CSS by having you add a small amount of styling to your HTML.

Exam Question 1 (p.336)**:** What does HTML stand for?
Acceptable Answer: hyper text markup language

Appendix 21 (page 221) tells how to make and see webpages "offline," without using any web hosting.

Appendix 22 (page 228) tells about the major browsers, and which ones to use in testing your webpages.

Appendix 23 (page 230) talks about text editors, for when you are editing right at your computer, and not by way of your web host.

We presume that you already have hosting, and that someone, maybe your instructor, will show you how to use it.

If that is not true, you can look in the appendix of this book for more help.

Appendix 19.1.2 (page 199) discusses web hosting. It tells you how to acquire it, and how to put your webpage on a web server.

Appendix 19.4 (page 204) shows how to use cPanel, a popular web hosting control panel.

1.1 Version One

Here is a webpage. Key it in. Keep the essential elements, but feel free to customize it. Use your own name instead of mine, for instance.

```
<h1>Don's Web Page</h1>
This is some normal text.
<b>This is some bold text.</b>
This is some normal text.
<i>This is some italic text.</i>
This is some normal text.
<br><img src=don.jpg width=500>
```

`<h1>` is the h1 tag. It identifies a heading at level 1, which is the most significant level. Heading levels range from h1 to h6. `</h1>` marks the end of that heading.

Notice that there is some content, "Don's Web Page", that is surrounded by the tags `<h1>` and `</h1>`. The tags are called markup. The content is called, well, content.

Next we have more content. Normal text. Bold text. Italic text.

`<b>` is the bold tag. It specifies that the content should be presented in **bold font.** `</b>` marks the end of the bold content.

`<i>` is the italic tag. It specifies that the content should be presented in *italic (slanted) font.* `</i>` marks the end of the italic content.

Incidentally, `<s>` is the strikethrough tag. It specifies that the content should be presented ~~with a horizontal line through the middle of it~~. `</s>` marks the end of the strikethrough content.

Incidentally, `<u>` is the underline tag. It specifies that the content should be presented <u>with a horizontal line right below it</u>. `</u>` marks the end of the

underline content.

`<br>` is the line break tag. It causes the following material to be on a new line, not on the end of the current line.

`<img...>` is the image tag. It specifies an image to be included in the webpage. There is no content. There is no `</img>` tag. The image tag is called a **void tag** because it does not "mark up" any content. I know, it seems like the picture itself is the content. But we count as content the things that appear between the start tag and the end tag.

In our case, `<img...>` has two parameters: src= and width=. The img src (source) gives a URL for the image source. The img width specifies that the image should be shown a certain number of pixels wide, in this case 500.

Exam Question 2 (p.336)**:** What does URL stand for?
Acceptable Answer: uniform resource locator

Width: We specify width= because often pictures are so large that they cannot be shown on the screen. By specifying width (or height, or both), we tell the browser how large to display the image.

Exam Question 3 (p.336)**:** When we specify image width, does it affect displayed size?
Required Answer: yes

Exam Question 4 (p.336)**:** When we specify image width, does it affect download time?
Required Answer: no

1.2 A Word About Pixels

We mentioned pixels. We should explain a bit more about them. For our purposes, the pixel is a unit of measure that tells the browser how big we want things to be. Section 12.4 (page 133) talks more about the measurement units that are available to us.

The Shrinking Real Pixel. Normally a pixel is one **pic**ture **el**ement. That is where it gets its name. It is one dot on the computer screen, or one blob of ink from an ink jet printer.

Over time, real pixels have been getting smaller and smaller. Printed things like the letters of words are getting smoother and smoother. Screens have gone beyond the point where the human retina can distinguish pixels without

a magnifying lens. Things measured in real pixels keep getting smaller.

This was a problem for webpages because images and other things got smaller on the screen, making them harder to see.

The Non-Shrinking CSS Pixel. To solve that problem, the CSS community came up with a revised definition of pixel. The official CSS definition is now the width of a "standard resolution" pixel (which means 96 dots per inch) when viewed at a distance of 28 inches (arms length). So, an item that is 96px wide should be as wide as an inch on a ruler held at arm's length. Scientifically speaking, there are about 47 pixels per degree of parallax, looking straight on. A CSS pixel on a billboard would be a lot bigger than a CSS pixel on an Oculus Rift virtual reality headset.

The bottom line here is that a CSS pixel could actually be many real pixels, but it will always appear to be the same size.

Exam Question 5 (p.337)**:** How many CSS pixels are in one degree of direct vision?
Acceptable Answer: 47

That gets rid of the problem of shrinking pixels. The moon viewed from earth is 24.3 pixels wide.

`http://inamidst.com/stuff/notes/csspx` by Sean Palmer explains more.

1.3 Seeing Your Webpage

You can probably see your webpage by simply saving it with .html at the end of the filename, and then double-clicking it on your desktop. Your default browser will display it.

To share your webpage it must be on a web server. Appendix 19.1.2 (page 199) explains how to put your webpage on a web server.

Your webpage must have a "name" by which people can request it. That name is called its URL.

1.4 Version Two

We will add some very simple styling.

Insert these lines at the end of your webpage.

```
This is a line
<span style=background:yellow>
with colored background</span>
for some of the words.
```

The **style=** thing is an HTML attribute. It is part of the span thing, which is an HTML tag.

Save your changes and reload your webpage.

Let's put a border around your picture. Inside your img tag, add the following style attribute:

```
style="border: thick black double"
```

Save your changes and reload your webpage. A thick black double line should appear around your picture. We will talk more about borders later.

1.5 Summary

You have created your first webpage. You have done the following things, possibly on your desktop:

- Used h1, b, and i tags.
- Displayed a picture.
- Controlled the displayed size of the picture.

If you put your webpage online, you have probably also done the following things:

- Shown you can cope with a control panel.
- Created a web page in the proper directory.
- Uploaded a picture.

Chapter 2

Content vs Markup

Contents

Every webpage has two main sections, the head and the body. The body contains all of our webpage's content. We will talk about that here. We will talk about the head in Chapter 3 (page 24).

2.1 Content

What is the purpose of a webpage? To communicate. Communicate what? Content. If we do not have something to say, why are we making a webpage?

2.1.1 Say Something

Maybe our content is artistic. Maybe it is words. Maybe it is interactive, like a game.

Decide what your content is.

If you don't have any content now, you can fake it using lorem ipsum. See section 25.2.1 (page 243), for details.

The easiest thing to work with is words, so we will start there.

Develop your content. Write your essay. Compose your poetry. Express your message. Create your content.

With your content in hand, we can begin the second step: markup.

As you develop your skills, you will probably find that you can do the markup as you create the content.

But for now, generate some content.

2.1.2 Look For Its Structure

With your content (a document of some kind) in hand, the next step is to look for the structure within.

Normally the structure consists of sentences arranged into paragraphs.

The paragraphs may fall into sections that have headings.

Headings and paragraphs are the major elements of structure.

You might also find that you have illustrations with titles or captions.

You might also find that you have tables of information.

Among your headings, you might find that you have major headings and minor headings.

Structural Markup is the process of telling the browser about the structure of your document.

Presentational Markup is the process of telling the browser more about how you want your document presented.

Marking some words as Bold or Italic is presentational markup.

Most presentational commands have been moved out of HTML and into

CSS, including centering and font specification.

Semantic Markup is the process of telling the browser what you are trying to accomplish rather than how to accomplish it. Within paragraphs, there may be words or phrases that you want to emphasize. Adding emphasis is semantic markup.

Webpage composition has been an evolving thing. Over time, best practices have been recognized. One of these is to separate the three main components: content (HTML), presentation and styling (CSS), and action (JavaScript). But in the early days of the web, these were often co-mingled.

Older presentational markup included things like `<center>`, which was used to center content on the screen, and `<font>`, which was used to specify fonts. Much of this older markup has been deprecated, by which we mean it is being phased out and should never be used again. However, you will probably see old examples that show how to do things that are still supported, but are currently discouraged.

Exam Question 6 (p.337)**:** What does deprecated mean?
Acceptable Answer: on its way out

Things that are deprecated can still be used at this time, but someday they may be forbidden. New development should always avoid deprecated capabilities.

Exam Question 7 (p.337)**:** Is the center tag deprecated?
Required Answer: yes

Exam Question 8 (p.337)**:** Is the font tag deprecated?
Required Answer: yes

2.1.3 Mark The Structure

Heading One: Put `<h1>` before and `</h1>` after each of your primary headings.

Paragraph: Put `<p>` before each paragraph. You can put `</p>` after each paragraph, but it is not required.

Bold: Put `<b>` before and `</b>` after each word or phrase you want to present in **bold font**. Bold is presentational markup.

Italics: Put `<i>` before and `</i>` after each word or phrase you want to present in *italic font*. Italic is also called slanted or oblique. Italic is presen-

tational markup.

As you do these things throughout your document, you are marking up your document.

2.2 Markup

There are many HTML markup tags that are recognized by web browsers but some are clearly more popular and more useful than others. Here is my list of common markups.

Best Practices: Tags should be written with lower-case letters. Write h1, not H1.

2.2.1 h1: Headings

`<h1>` is the structural markup that introduces a level-one heading. It does not force the heading to be bigger and bolder, although that normally happens. Instead, it just identifies it as a heading. The exact effect is specified elsewhere (in the CSS).

h1 means something is a major heading. h2 means something is a subheading.

Exam Question 9 (p.337)**:** The HTML tag "h1" stands for what word(s)?
Acceptable Answer: heading one

Exam Question 10 (p.337)**:** The HTML tag "h2" stands for what word(s)?
Acceptable Answer: heading two

Headings provide a fast way for your reader to understand your web page. They provide sign posts to your content. They identify the structure that is present in your webpage.

A level-one heading is a primary heading. It is the most important heading. Below a level-one heading there may be a level-two heading, which would be less important, and would identify a sub-part of the content related to the previous level-one heading.

Heading levels range from 1 to 6, marked as `<h1>` through `<h6>`.

For h1 the start tag would be `<h1>` and the end tag would be `</h1>`.

Having `</h1>` at the end is required. Each heading needs to be closed with

a matching slash tag.

The content that appears between the start and end tags is the heading. The start and end tags are the markup.

```
<h1>This is a level-one heading.</h1>
```

Warning: Do not use h1 h2 h3 simply for their presentational value to create different sizes of text. Use css for presentational things. Use h1 h2 h3 to describe the structure of your document.

2.2.2 p: Paragraphs

Paragraphs are the foundational unit of text content. As in regular writing, each paragraph is focused. And each paragraph is short enough to look inviting. But we are not here to teach you good writing skills. That is beyond our scope. Instead, we acknowledge that paragraphs are important and we identify them by using the proper markup.

Exam Question 11 (p.337)**:** The HTML tag "p" stands for what word(s)?
Acceptable Answer: paragraph

`<p>` is the structural markup that introduces a paragraph. `</p>` is optional at the end of the paragraph.

```
<p>This is a paragraph.</p>
```

```
<p>This is a paragraph.
```

You can use the **p** tag to identify paragraphs within your document. Typically this creates space between the paragraphs.

Paragraphs cannot be nested. (You cannot have one paragraph within another paragraph.)

2.2.3 br: Break

Each paragraph starts on a new line. But sometimes we want to start a new line while remaining in the same paragraph.

Exam Question 12 (p.337)**:** The HTML tag "br" stands for what word(s)?
Acceptable Answer: break

One example of this might be writing poetry or verse. For example, here is part of a poem I remember from my childhood (by Robert Frost).

Whose woods these are I think I know.
His house is in the village though.
He will not see me stopping here
to watch his woods fill up with snow.

http://www.ketzle.com/frost/snowyeve.htm has the rest of it.

We use `<br>` to insert a break into a line.

Break causes the next content to appear on a different line than the previous content.

Break is written `<br>` or `<br/>` or `<br />`.

There is no matching `</br>` tag. Break is a **void** tag, meaning it is a tag with no content. Having `</br>` at the end is forbidden.

2.2.4 Micro Markups

There are several markups that are used on small pieces of wording. I call them micro markups.

b: Bold

Bold specifies that the content should be rendered in a font that has thicker characters. Put `<b>` in front of the content and put `</b>` after the content.

Exam Question 13 (p.337)**:** The HTML tag "b" stands for what word(s)?
Acceptable Answer: bold

Having `</b>` at the end is required.

strong: Strong

Put `<strong>` before and `</strong>` after each word or phrase you want to say strongly. Strong is a kind of semantic markup and is typically rendered the same as `<b>` (bold).

Exam Question 14 (p.337)**:** The HTML tag "strong" stands for what word(s)?
Acceptable Answer: strong

i: Italic

Italic specifies that the content should be rendered in a font that is slightly slanted. Put `<i>` in front of the content and put `</i>` after the content.

Exam Question 15 (p.337)**:** The HTML tag "i" stands for what word(s)?
Acceptable Answer: italics

Having `</i>` at the end is required.

em: Emphasis

Put `<em>` before and `</em>` after each word or phrase you want to emphasize. Emphasis is semantic markup and is typically rendered the same as `<i>` (italics).

Exam Question 16 (p.337)**:** The HTML tag "em" stands for what word(s)?
Acceptable Answer: emphasize

s: Strike-through

Put `<s>` before and `</s>` after each word or phrase you want to present with **strike-through**. This style is often used to show content that was present before but has been ~~deleted~~, to show that the deletion is intentional.

Exam Question 17 (p.337)**:** The HTML tag "s" stands for what word(s)?
Acceptable Answer: strikethrough

u: Underline

Put `<u>` before and `</u>` after each word or phrase you want to present underlined.

Exam Question 18 (p.337)**:** The HTML tag "u" stands for what word(s)?
Acceptable Answer: underline

sub: Subscript

Put `<sub>` before and `</sub>` after each word or phrase you want to present as a lowered **subscript**, like the 2 in H_2O.

Exam Question 19 (p.337)**:** The HTML tag "sub" stands for what word(s)?
Acceptable Answer: subscript

We can also do subscripting by using the relative positioning command. See chapter 13 (page 138) for more information on positioning.

sup: Superscript

Put `<sup>` before and `</sup>` after each word or phrase you want to present as a raised **superscript**, like the 2 in x^2.

Exam Question 20 (p.337)**:** The HTML tag "sup" stands for what word(s)?
Acceptable Answer: superscript

We can also do superscripting by using the relative positioning command. See chapter 13 (page 138) for more information on positioning.

2.2.5 img: Image

Exam Question 21 (p.337)**:** The HTML tag "img" stands for what word(s)?
Acceptable Answer: image

Chapter 5 (page 49) will cover images in more depth. Here we introduce the basic points.

There is no `</img>` tag. Image is a **void** tag, meaning a tag with no content. Having `</img>` at the end is forbidden.

The img tag requests that an image be displayed. It has several attributes that are commonly specified.

src= To be useful, the image must have a source. That is specified using the src= attribute. The value of this attribute is a URL that identifies the image.

width= It is helpful to specify the size of the image. This can be done using the width= attribute. Width is measured in pixels. If you specify width but no height, the browser will calculate the height for you. Also, when you say width= you simply put a number of pixels, like `width=500`

and **not** like `width=500px` (where px is the CSS abbreviation for pixels).

height= It is helpful to specify the size of the image. This can be done using the height= attribute. Height are measured in pixels. If you specify height but no width, the browser will calculate the width for you.

If you specify both the width and the height, you can distort the image.

(We can also use CSS to specify max width, min width, max height, and min height, depending on what we are trying to achieve.)

alt= The img tag has a required attribute: alt= is used to provide a description of the image. This is primarily for disabled web users who might not be able to see the image, and must have a web reader say aloud the words you put in the alt= attribute.

Normally the alt text includes spaces. If so, the text must be surrounded by quote marks. If it is just one simple word, the quotes can be left off.

title= All tags have this optional attribute: title= is used to provide additional information that is only visible when the user hovers their mouse over the item. (It is not clear to me how you hover a mouse on a mobile device like a smart phone or tablet.)

Normally the title text includes spaces. If so, the text must be surrounded by quote marks.

2.2.6 a: Link

Exam Question 22 (p.337)**:** The HTML tag "a" stands for what word(s)?
Acceptable Answer: anchor

The a tag uses an href parameter to specify the destination your user will go to when they click on the link.

The a tag uses a target parameter to specify the destination is to be opened in a new tab in the browser.

Example: `<a href=somepage.html target=_blank>Click Here</a>`

The content between `<a>` and `</a>` is the clickable link. Other things can be placed there, such as images.

Having `</a>` at the end is required.

2.2.7 Comments

It is sometimes handy to put notes into a webpage, notes that are intended for future authors (including yourself) to explain this or that. These notes are not intended to be rendered as part of the webpage seen by your ultimate readers. They are for the authors.

`<!--` and `-->` are used to surround an HTML **comment**.

Those tags and everything between them will not be rendered (displayed) by the browser.

Comments cannot be nested. When a comment starts, it continues until the first end tag is found. Here is an example of what I mean.

Here is a comment.

```
aaa <!-- bbb ddd --> eee
```

Here is another comment.

```
<!-- ccc -->
```

If we insert the second comment into the middle of the first one, we get this:

```
aaa <!-- bbb <!-- ccc --> ddd --> eee
```

In this example, the comment that will be recognized by the browser is `<!-- bbb <!-- ccc -->` because the first `-->` marks its end. After the comment is ignored, this is what will be rendered.

```
aaa  ddd --> eee
```

Oops.

Exam Question 23 (p.337)**:** What character sequence marks the start of a comment in HTML?
Required Answer: `<!--`

Exam Question 24 (p.337)**:** What character sequence marks the end of a comment in HTML?
Required Answer: `-->`

Exam Question 25 (p.337)**:** What character sequence is not allowed within a comment in HTML5?
Required Answer: `--`

Exam Question 26 (p.338)**:** In HTML, can comments be nested?
Required Answer: no

Chapter 3

Head Sets The Stage

Contents

The body portion of a webpage, which we discussed in Chapter 2 (page 13), specifies the content to be presented.

The head portion of a webpage, which we discuss in this chapter, sets the stage for the body portion.

As web designers, we basically care about two things.

(1) Does our webpage look the way we want on all browsers and all devices?

(2) Do the search engines push traffic to our website.

The way we write our webpages will determine how things go. There is a very precise language for writing webpages. We will want to do our best to follow these standards.

3.1 Template

To help you out, I am going to jump the gun here just a bit, and give you a very short template that you can use on all your webpages. **You should memorize this.** It has the key elements that you will need. We will discuss them below.

```
<!DOCTYPE html><head lang=en><meta charset=utf-8>
<title>Your Title Goes Here</title>
<style>Your CSS Goes Here</style>
</head><body>
Your Marked-Up Content Goes here
</body>
```

3.2 Achieving Trust

The web is old. People have been making webpages for many, many years, and there are many, many webpages that do not follow the currently accepted standards.

Browsers are (and need to be) incredibly resilient. They want to do the right thing. They will do their best to display anything that shows up purporting to be a webpage. Sometimes webpages do not follow the rules of the HTML

language, but each browser will still do its best. They will not always do the same thing, though.

XHTML: After HTML4, many in the web community moved to XHTML, which tried to tighten standards but was frustrating and not widely accepted.

HTML5: After that failure, HTML5 went the other way and did not require things that could be reasonably inferred or implied by context. We find this with some older features of the HTML language, such as head, body, paragraph, list items, and table rows, where the endings (and sometimes beginnings) can be accurately inferred.

Trust: As web designers, if we want our content to be displayed in a particular way, we need to convince the browser that we know what we are doing. Do not break any important rules of the language. Then the browser will trust us. But if we do break rules, the browser will do what it thinks we want instead of what we actually requested. And different browsers may think differently.

We want search engines to trust us, too.

We should provide web content that follows the official rules of the HTML language. That way we increase our chances that all browsers will display our content in the same way, the way that we want and intend.

To do this, we must properly introduce our webpage, and provide it with a designated head and body.

3.3 Validation

Writing in a precise language can be tricky, and it is helpful to have some way to find out whether the things we are writing follow the rules.

Fortunately, there are some tools called validators that will look at our webpage and tell us whether it needs to be fixed.

W3C is the standards body that develops standards for the World Wide Web. http://www.w3.org/ is their homepage. Their materials are official and authoritative.

W3C does not try to provide tutorials and guides for novices, but instead caters to the detailed and precise needs of experts.

They provide two incredibly useful validators that can be used to find errors in our HTML code or CSS code.

HTML Validator

http://validator.w3.org/ is their validator for HTML code.

We can put this somewhere in our webpage to link to an HTML validator.

```
<a href='http://validator.w3.org/check?uri=referer'>html5</a>
```

CSS Validator

We can put this somewhere in our webpage to link to a CSS validator.

```
<a href=http://jigsaw.w3.org/css-validator/check/referer>css3</a>
```

JavaScript Validator

JavaScript can be very complicated. It is a full-fledged programming language, and because it is the main language on webpages, it is probably the most widely used programming language in the world.

Writing complicated pieces of JavaScript is far beyond the scope of this book. But it is good to know that there are validators out there that can look at our code and give us advice.

http://www.jslint.com/ is provided by Douglas Crockford, one of the original developers of JavaScript.

http://jshint.com/ is a competing validator provided by Anton Kovalyov, a computer programmer from San Francisco.

3.4 Document Type

Getting back to the actual writing of our webpages, the page starts by telling what kind of document it is.

Required: DOCTYPE. The first piece of a webpage is the doctype line. In HTML5 it looks like this:

```
<!DOCTYPE html>
```

This specifies that the webpage is written in HTML, and more particularly in the HTML5 dialect. If we do not specify it, the browser will just guess what version of HTML we are writing in.

Exam Question 27 (p.338)**:** What is the first markup element of every HTML5 webpage?
Required Answer: `<!DOCTYPE html>`

3.5 Head

Every HTML document is required to have exactly one **head** section. This is where we tell things "about" our webpage, such as its title, its description, its internal style sheet, and the external resources it needs.

Optional: start of head. The next piece of a webpage after the doctype introduces the head section. It looks like this:

```
<head lang=en>
```

This specifies the start of the head, and also that the webpage is written in the "en" (English) language.

Exam Question 28 (p.338)**:** What is the second markup element of every HTML5 webpage?
Acceptable Answer: `<head lang=...>...</head>`

It is optional because if we provide header material such as the character set or title, but without explicitly beginning our head, the browser will pretend that we typed `<head>`. I recommend you have an explicit head.

3.5.1 Character Set

One of the first things the browser wants to know about our page is what character set we are using. We will use UTF-8, but there are other character sets that are desirable for languages like Chinese, Japanese, and Korean, for instance.

Required: charset. Inside the head, we must specify the character set. Charset is a required element in the head. It is specified like this:

```
<meta charset=utf-8>
```

Utf-8 stands for UCS Transformation Format, 8 bit. There are other charac-

ter sets that might also be used instead of utf-8, but it is the most popular.

Exam Question 29 (p.338)**:** What HTML markup is used to specify our character set?
Acceptable Answer: `<meta charset=...>`

3.5.2 Title

Required: title. Inside the head, we also need to specify the webpage title. This is a required element in the head. It is specified like this:

```
<title>our title goes here</title>
```

Replace "our title goes here" with whatever we want our title to be.

Exam Question 30 (p.338)**:** What HTML markup is used to specify the title of our webpage?
Acceptable Answer: `<title>...</title>`

3.5.3 Style

Optional: style. Inside the head, we can specify styles. This is called the internal style sheet. It is done like this:

```
<style>
  body { text-align: center; background-color: #ffffcc; }
</style>
```

Exam Question 31 (p.338)**:** What HTML markup is used to create an internal style sheet?
Acceptable Answer: `<style>...</style>`

3.5.4 Scripts

If we are including any JavaScript in our webpage, part or all of it might be included in the head.

Exam Question 32 (p.338)**:** What HTML markup is used to identify JavaScript?
Acceptable Answer: `<script>...</script>`

3.5.5 Head to Body Transition

Optional: end of head. When we are done with the head, we should say so.

If we begin our body without explicitly ending our head, the browser will pretend that we typed `</head>`.

Every HTML document is required to have exactly one **body** section. This is where all the content appears. If we do not label our content with body tags, the browser will assume it for us.

Exam Question 33 (p.338)**:** What HTML markup is used to mark off the content of the document?
Acceptable Answer: `<body>...</body>`

Optional: start of body. Immediately after finishing the head, we introduce the body, like this:

```
<body>
```

Inside the body, we put all our content with its HTML markup.

Optional: end of body. When we are done with the body, we say so like this:

```
</body>
```

That ends our document.

If our webpage simply ends, the browser will pretend that we typed `</body>` as the page ended.

3.6 SEO: Search Engine Optimization

How do I drive traffic to my website? How do I get noticed by the search engines?

These questions are the key elements of the topic of "Search Engine Optimization," or **SEO**.

The easy answer is just to do a Google search on "Search Engine Optimization." I found 92 million results just now (March 2013).

http://en.wikipedia.org/wiki/Search_engine_optimization has Wikipedia's summary which is much better than my own.

But here is my own, anyway. Enjoy.

3.6.1 The Basics

Search engines are trying to help their users find good content. Users come with a goal. They want to learn about something. They type in some words. The search engine tries to read their mind based on those words, and tries to show them relevant websites.

What if our website is truly relevant, perhaps the most relevant of all websites? Do the users get to see our website listed on the first page?

Just as the search engine tries to understand the user, they also try to understand every website they visit. Including ours.

Ideally they want to understand our website, but that is very difficult. So, instead, they do the best they can.

(1) They look at our URL. Our domain name is important. Our choice of folder names and page names is important. Ideally they will be meaningful to our eventual customer.

(2) They look at our title. This is the small phrase we put between the `<title>` and `</title>` tags in our head section. Our title should be short, up to about 50 characters, and should be meaningful to our eventual customer.

Exam Question 34 (p.338)**:** For SEO, is title very important?
Required Answer: yes

Exam Question 35 (p.338)**:** For SEO, about how many characters of our title are used?
Acceptable Answer: 50

(3) They look at our meta description. This is a longer phrase we can put in our head section. It provides the wording that the search engine will show to customers. They are likely to read it when they try to decide whether to visit our page or not. It looks like this:

```
<meta name=description content="tell about our webpage here">
```

Exam Question 36 (p.338)**:** For SEO, is meta description very important?

Required Answer: yes

Exam Question 37 (p.338)**:** Give the prototype (pattern) for meta description.
Acceptable Answer: `<meta name=description content="...">`

Exam Question 38 (p.338)**:** For SEO, about how many characters of meta description are used?
Acceptable Answer: 150

Meta keywords used to be heavily used for SEO. It was abused by many websites that pretended to be something they were not. Now it is largely ignored by search engines.

Exam Question 39 (p.338)**:** For SEO, are meta keywords very important?
Required Answer: no

(4) They look at our content. We should have good content, the kind of content that our customer is looking for.

3.6.2 Scammers

Fake Websites: I will define **fake websites** to be ones that do not offer content. Their purpose is not to help users. Instead, their purpose is simply to make money as easily as possible. I will call the people that provide these websites **scammers**.

Why does this happen?

Google and other sources can place advertisements on pages. Every time an ad is displayed, a small amount of money goes to the owner of the webpage. Or every time an ad is clicked on, a larger amount of money goes to the owner of the webpage.

If we get lots of people to view our page, we can make money from the advertising that is displayed on our page.

If we can generate lots of pages, each of which gets some viewers, it can still add up to a lot of traffic.

Advertisers do not like fake websites. Google does not like fake websites. But there are billions of websites. They use computer-based rules for identifying such websites, and then they avoid putting them on the first page of the search results.

In short, we do not want to look like a fake website. We want to look (and be) sincere.

So, in the battle of Search Engine Optimization, we need to look good, and also avoid looking bad.

3.6.3 The Rules Are Very Secret

Places like Google are very secretive about their rules for giving priority to the best search results. They have good reason. If people knew how they calculated their rankings, people would scam the system and do things to fake out the computers.

As it is, people still figure out what is going on and try to scam the system. So places like Google are always on the lookout for scammers and when they find them, they move them to the bottom of their rankings. Permanently if possible.

Avoid looking like a bad guy.

3.6.4 Getting Noticed

Getting noticed is actually not very hard. The search engines have their armies of web crawlers (also called spiders) that are constantly reviewing the whole World Wide Web for new content.

My own websites get hit about once a week or once every two weeks by each of the major web crawlers: Google, Bing (including Yahoo), and others.

So, as long as we can be found, we will be indexed.

When we create a new domain, the crawlers will find the homepage of that domain. They will not find any of the other pages or subdomains unless they are linked.

3.6.5 We Can Look Permanent

If we paid for our domain name, like it is a dot-com or a dot-org, and we bought several years worth of time, then we look more permanent than someone that gets, oh, I don't know, maybe a dot-tk domain name and might just disappear tomorrow.

But that's just our entry into the game. We can buy 99 years of domain name and still not achieve a high page rank.

3.6.6 We Can Pay Money

There are lots of sites that will offer to get us noticed by the major search engines, and even the minor search engines, for a few USD a month.

My sense is, don't bother. We will get noticed anyway. They are just taking money from desperate web designers. Ignore them.

3.6.7 We Can Swap Links

The more sites that link to us, the more we look like a hub or touch point or crossroads or nexus. And that is good. If sites link to us, presumably it is because we are worth linking to. We have valuable content.

This nexus idea is actually at the core of the Google PageRank system. The more places that point to us, the more important we become.

Say we have 100 friends, and each of them adds a link to their homepage, or a nearby page, mentioning us. Our PageRank goes up. We are popular. Of course, we would be willing to reciprocate and put a link on our homepage to each of our friends, right?

If we want to scam the system, we can invent 100 friends and make a real rat's nest of links back and forth. It's a bit of work, but we can be sure someone has tried it. Lots of someones, actually.

However, if the links are not relevant to our topic, we could be penalized. Garbage links make us look like a non-content website.

3.6.8 Page Design

Design your content to make it easy for web crawlers to understand. Use the latest HTML5 tags, like article and header and footer.

Include lots of keywords and phrases that you think people will be looking for. Think like a fish to catch a fish.

3.6.9 The Bottom Line

Your best chance to get good PageRank is to have quality content. Original, quality content. Content that others link to. Content that people select when they are looking at Google's search results.

Now, go back and read that Wikipedia article.

3.7 Review of Previous Chapters

Exam Question 40 (p.338)**:** What HTML markup is used to specify a top level header?
Acceptable Answer: `<h1>...</h1>`

Exam Question 41 (p.338)**:** What HTML markup is used to specify a second-level heading?
Acceptable Answer: `<h2>...</h2>`

Exam Question 42 (p.338)**:** What HTML markup is used to start a new paragraph?
Required Answer: `<p>`

Exam Question 43 (p.338)**:** What HTML markup is used to present bold content?
Acceptable Answer: `<b>...</b>`

Exam Question 44 (p.338)**:** What HTML markup is used to present oblique (italic) content?
Acceptable Answer: `<i>...</i>`

Exam Question 45 (p.339)**:** What HTML markup is used to emphasize some content?
Acceptable Answer: `<em>...</em>`

Exam Question 46 (p.339)**:** What HTML markup is used to draw a line through some content?
Acceptable Answer: `<s>...</s>`

Exam Question 47 (p.339)**:** What HTML markup is used to draw a line under some content?
Acceptable Answer: `<u>...</u>`

Exam Question 48 (p.339)**:** What HTML markup is used to present lowered content (Like the 2 in H2O)?

Acceptable Answer: `<sub>...</sub>`

Exam Question 49 (p.339)**:** What HTML markup is used to present raised content (Like the 2 in x squared)?
Acceptable Answer: `<sup>...</sup>`

Exam Question 50 (p.339)**:** What HTML markup is used to go to a new line?
Required Answer: `<br>`

Exam Question 51 (p.339)**:** What HTML markup is used to insert a picture?
Acceptable Answer: `<img src=...>`

Exam Question 52 (p.339)**:** What HTML markup is used to specify a link to another page?
Acceptable Answer: `<a href=...>...</a>`

Chapter 4

HTML Tags and Attributes

Contents

In this chapter we will provide more information about HTML markup.

4.1 Markup: Presentational, Structural, Semantic

Content is more than meets the eye. It is the meaning.

Presentational Markup (style) is worried only about what meets the eye. It tells how you want certain things to be shown, how they look. It includes tags for line break, bold, italic, strike-through, underscore. It includes explicit coloring and sizing.

When you are first starting out with web design, the obvious first thing to do is work on presentation. It is what people see.

Do you like the ability to copy and paste words from a PDF file into some other document you are working on? If the PDF is simply a scan, you cannot copy and paste the words, just portions of the image. But if the PDF file came from a program, or if the scanner is smart enough, maybe the words can be selected and copied. The fact that they are words and not just smears of ink on a page, that fact goes beyond the presentation and into the semantics (meaning).

Structural Markup identifies the structure of your document. It includes tags for things like headings, paragraph, lists, and tables. It goes a step beyond presentation to begin revealing the meaning of the document. It shows the relationships between the parts of your webpage.

Structural markup is focused on the relationships between pieces of a webpage. Often this is incorrectly called "semantic," but they are actually different.

Semantic Markup (meaning) is a richer approach to telling what things are. Often this is reflected in how they are displayed. It includes tags for emphasis, title, header, footer, and navigation.

The long-term goal of structural and semantic markup is to help search engines and other programs understand your contents, so they can help everyday users like ourselves search for information and find it.

4.2 Tags

All HTML markup looks the same as this:

```
<tag att=val att=val att=val ...>
```

This is the HTML markup prototype. There is an opening angle bracket (less than sign), followed by the tag, followed by zero or more attributes. Then there is the closing angle bracket (greater than sign). Each attribute

must have a space in front of it.

Exam Question 53 (p.339)**:** Give the prototype (pattern) for HTML markup in general.
Acceptable Answer: `<tag att=val ...>`

In our study of HTML we will use the following vocabulary.

tag means the first piece of an HTML markup. It normally matches a word or phrase in English that has a similar meaning. For example, tag might be **h1** or **p** or **img**.

attribute= means one of the attributes (parameters) that modify the behavior of that HTML tag. For example, attribute might be **style=** or **class=** or **title=**.

value will appear in the context of attribute=value.

Later on, in our study of CSS we will find a similar vocabulary. However, for CSS **attribute:** with a colon will be used instead of **attribute=** with an equals sign.

All HTML markup is done using tags and attributes. Markup is always

(1) introduced by `<`,

(2) followed by a tag,

(3) followed by zero or more attributes,

(4) each attribute can have a value, and

(5) ended with `>` or `/>`.

4.2.1 Most Tags Come in Pairs

There are two main kinds of markup used in webpages.

The most common kind uses a **start tag** before the content to be marked, and a matching **end tag** after the content to be marked.

The end tag uses the same tag word as the start tag, but it has `/` right before the tag word.

Example: `<thisisatag>` goes with `</thisisatag>`.

4.2.2 Some Tags Stand Alone (Void)

The other kind uses a single tag and does not surround any text. These are called **void tags**.

Instead of marking up the existing content, void tags insert additional content, such as an image or a line break.

These are some commonly-used void tags: base, br, hr, img, input, link, meta.

```
Check out this picture: <img src=abc.jpg>
```

4.2.3 Starting and Ending a Tag

Exam Question 54 (p.339)**:** What character marks the start of each piece of HTML markup?
Required Answer: `<`

Exam Question 55 (p.339)**:** What character marks the end of each piece of HTML markup?
Required Answer: `>`

Exam Question 56 (p.339)**:** What two characters can mark the end of HTML markup for a void tag?
Required Answer: `/>`

Some authors strongly recommend using `/>` at the end of each void tag. I don't believe that any more.

Exam Question 57 (p.339)**:** When a tag is void, what does that mean?
Acceptable Answer: it cannot have a separate end tag

Exam Question 58 (p.339)**:** When do you need a space before the end of HTML markup?
Acceptable Answer: If the markup ends in /> and a non-quoted attribute value is right before it.

The reason you need a space between a non-quoted attribute value and /> is to avoid having the / wrongly accepted as part of the attribute value.

4.2.4 Nesting and Overlapping Markup

Let's define a couple of words: nesting and overlapping. Markup can usually be nested, but normally overlapping is not allowed.

By **nested tags** we mean:

```
<x> aaa <y> bbb </y> ccc </x>
```

In this case, the whole item, aaa bbb ccc, is tagged with the x tag, and a smaller part in the middle, bbb, is tagged with both the x tag and the y tag.

Exam Question 59 (p.339)**:** When one set of tags begins and ends totally inside another set of tags, what do we call that?
Acceptable Answer: nesting or nested

By **overlapping tags** we mean:

```
<x> aaa <y> bbb </x> ccc </y>
```

In this case, we apparently intend the first two-thirds, aaa bbb, to be tagged with the x tag, and the last two-thirds, bbb ccc, to be tagged with the y tag, with the middle third, bbb, tagged with both the x tag and the y tag.

Overlap is forbidden.

In most cases, the solution is to end the tag and then restart it, like this:

```
<x> aaa <y> bbb </y></x><y> ccc </y>
```

All you really need to know is that nesting is okay, and overlapping is not.

In case you are curious, here is why:

After your webpage is loaded into the browser, it is no longer stored as HTML. Instead, it is stored as a tree-like structure called the **DOM** (document object model). Trees cannot handle overlaps.

The browser may fix it for you automatically, but it is not required to. A validator will just say it is wrong.

Exam Question 60 (p.339)**:** What is wrong with this ordering: <a>.........?
Acceptable Answer: overlap

Exam Question 61 (p.339)**:** What is wrong with this ordering: <a>.........?
Acceptable Answer: nothing

4.3 Tag Attributes

We can use **attributes** to modify or customize tags. Attributes are also called **parameters** or **properties**.

Attributes are specified right after the tag. Here is an example.

```
<abc x=5 y=hello z=whatever>
```

In this example, there are three attribute/value pairs. The first is `x=5`. The second is `y=hello`. The third is `z=whatever`.

Normally the order does not make any difference. We could have said the following, and it would mean the same thing.

```
<abc z=whatever y=hello x=5>
```

Exam Question 62 (p.339)**:** In HTML does the order in which attributes are specified make any difference?
Required Answer: no

4.3.1 Blank Attribute Values

If the attribute's value is blank, you can specify it or leave it off. Here are three options.

`<... attribute="" ...>` (the value is explicitly the empty string)

`<... attribute= ...>` (the value is assumed to be the empty string)

`<... attribute ...>` (the value is assumed to be the empty string)

The second option, `<... attribute= ...>`, is legal but it is dangerous and should **not** be used. The following attribute, if any, could be interpreted to be its value if you are not careful.

Sometimes a missing attribute has a default value. The submit button, for example, has a default value that is different than specifying `value=''`.

4.3.2 Spacing

Although it is perfectly legal to have zero or more spaces before the `=`, and zero or more spaces after the `=`, we strongly recommend that spaces not be placed around the `=` (equals) sign. It makes things confusing.

`<... attribute="abc" ...>` (strongly recommended)

`<... attribute = "abc" ...>` (not recommended)

`<... attribute= "abc" ...>` (not recommended)

`<... attribute ="abc" ...>` (not recommended)

4.3.3 Quote Marks

For an attribute value, quote marks are normally not required.

To avoid confusion, if the value itself could be misunderstood to be HTML syntax, then the quote marks are required. These syntax characters are: space, the three kinds of quotes (single, double, and backquote), less than (<), greater than (>), and equals (=).

Having a space inside the value is the most common reason for needing quotes.

Specifically, when you are just writing letters, digits, and simple punctuation like dot (.) or slash (/), the quotes are not required. A URL does not usually require quote marks.

When quote marks are used, they can be either single quotes or double quotes, and must be the same before and after the value.

`<... attribute=abc ...>` (good)

`<... attribute=a'c ...>` (not allowed)

`<... attribute="a'c" ...>` (good)

Exam Questions: You should know whether certain characters make it necessary to enclose an attribute's value in quotes or not.

Space has special meaning in this part of HTML. It delimits things. Because it has special meaning, it must be quoted.

Exam Question 63 (p.340)**:** If an HTML attribute's value includes a space does that force it to be quote marked?
Required Answer: yes

Double-quote has special meaning in this part of HTML. It delimits a value. Because it has special meaning, it must be quoted (using single-quotes).

Exam Question 64 (p.340)**:** If an HTML attribute's value includes a

double-quote (") does that force it to be quote marked?
Required Answer: yes

Less-than has special meaning in this part of HTML. It starts a new tag. Because it has special meaning, it must be quoted.

Exam Question 65 (p.340)**:** If an HTML attribute's value includes a less-than (<) does that force it to be quote marked?
Required Answer: yes

Greater-than has special meaning in this part of HTML. It ends a tag. Because it has special meaning, it must be quoted.

Exam Question 66 (p.340)**:** If an HTML attribute's value includes a greater-than (>) does that force it to be quote marked?
Required Answer: yes

Single-quote has special meaning in this part of HTML. It delimits a value. Because it has special meaning, it must be quoted (using double-quotes).

Exam Question 67 (p.340)**:** If an HTML attribute's value includes a single-quote (apostrophe) (') does that force it to be quote marked?
Required Answer: yes

Back-quote has special meaning in this part of HTML. Because it has special meaning, it must be quoted.

Exam Question 68 (p.340)**:** If an HTML attribute's value includes a back-quote (`) does that force it to be quote marked?
Required Answer: yes

Equals has special meaning in this part of HTML. It goes between an attribute and its value. Because it has special meaning, it must be quoted.

Exam Question 69 (p.340)**:** If an HTML attribute's value includes an equals (=) does that force it to be quote marked?
Required Answer: yes

Letters do not have any special meaning in this part of HTML.

Exam Question 70 (p.340)**:** If an HTML attribute's value includes a letter (A a B b ...) does that force it to be quote marked?
Required Answer: no

Digits do not have any special meaning in this part of HTML.

Exam Question 71 (p.340)**:** If an HTML attribute's value includes a digit (1 2 3 ...) does that force it to be quote marked?

Required Answer: no

Colons do not have any special meaning in this part of HTML. They do have special meaning within a URL, and after a CSS attribute name, but not here.

Exam Question 72 (p.340)**:** If an HTML attribute's value includes a colon (`:`) does that force it to be quote marked?
Required Answer: no

Dots, also known as decimal points, periods, or full stops, do not have any special meaning in this part of HTML. They do have special meaning within a URL, but not here.

Exam Question 73 (p.340)**:** If an HTML attribute's value includes a dot (`.`) does that force it to be quote marked?
Required Answer: no

Exam Question 74 (p.340)**:** If an HTML attribute's value includes a dash (`-`) does that force it to be quote marked?
Required Answer: no

Slashes do not have any special meaning in this part of HTML. They do have special meaning within a URL, but not here.

Exam Question 75 (p.340)**:** If an HTML attribute's value includes a slash (`/`) does that force it to be quote marked?
Required Answer: no

Percents do not have any special meaning in this part of HTML. They do have special meaning within a URL, but not here.

Exam Question 76 (p.340)**:** If an HTML attribute's value includes a percent (`%`) does that force it to be quote marked?
Required Answer: no

Ampersands do not have any special meaning in this part of HTML. They do have special meaning in creating character references, but that does not affect us here.

Exam Question 77 (p.341)**:** If an HTML attribute's value includes an ampersand (`&`) does that force it to be quote marked?
Required Answer: no

If you are not quoting the value, you must leave a space after it, or you must end the tag immediately. Watch out for this situation:

`<img ... width=100 />` (right)

`<img ... width='100' />` (right)

`<img ... width=100/>` (wrong)

`<img ... width='100'/>` (okay but discouraged)

In the "wrong" case, it is wrong because the slash (/) becomes part of the value. That is, width is `100/` instead of being just `100`.

That is why some authors will tell you to leave a space before `/>` at the end of a tag.

Exam Question 78 (p.341)**:** In what is the value of width?
Required Answer: 100

Exam Question 79 (p.341)**:** In what is the value of width?
Required Answer: 100

Exam Question 80 (p.341)**:** In what is the value of width?
Required Answer: 100/

4.4 Global Attributes

Some attributes can be included on any HTML tag. These are called the global attributes. They are universally available. Here is a complete list: accesskey, class, contenteditable, contextmenu, dir, draggable, dropzone, hidden, id, lang, spellcheck, style, tabindex, title, and translate.

Exam Question 81 (p.341)**:** List the four commonly-used global attributes for HTML tags.
Required Answer: id, class, style, title

http://www.w3.org/TR/html-markup/global-attributes.html has more.

http://www.w3schools.com/tags/ref_standardattributes.asp has more.

4.4.1 style=

style= specifies any CSS styling attributes that may be desired. Example, `style='color:red'`. The style attribute provides immediate control of anything that can be done using a cascading style sheet. Because it is specified immediately at the tag, it takes precedence over anything the CSS style sheet may say.

4.4.2 id=

id= specifies the ID of the item. Example, `id=top`. The id attribute gives a unique name to the tag. Each id should only appear once on a web page. This name can be used by the web browser, to jump to that part of the webpage. It can also be used by CSS and JavaScript to control style and actions or animations relating to that item.

ID values must be unique within an html page, and must start with a letter. They can continue with letters, digits, underscores, dots (points, periods, full stops), dashes, and colons. (Specifically, they cannot include spaces.) There is no limit to length.

Exam Question 82 (p.341)**:** In HTML, what type of character can be first in an id?
Acceptable Answer: letter

Exam Question 83 (p.341)**:** In HTML, what six kinds of characters can appear after the first in an id?
Required Answer: letter, digit, dot, dash, colon, underscore

Exam Question 84 (p.341)**:** In HTML, is id=123 valid?
Required Answer: no

Exam Question 85 (p.341)**:** In HTML, is id=abc valid?
Required Answer: yes

Exam Question 86 (p.341)**:** In HTML, is id=a.b.c valid?
Acceptable Answer: yes

Exam Question 87 (p.341)**:** In HTML, is id=a-b-c valid?
Acceptable Answer: yes

Exam Question 88 (p.341)**:** In HTML, is id=”a b c” valid?
Acceptable Answer: no

Because the legal ID characters are so limited, it is true that you never need to put quote marks around an ID value.

4.4.3 class=

class= specifies the class of the item. Example, `class=highlight`. Like id, class can be used by CSS and JavaScript to control style and actions or animations relating to items that share that same class. Unlike id, the same class can be used on more than one item. If you have more than one class, separate them by spaces and enclose the list in quote marks.

4.4.4 title=

title= specifies text that will be displayed if the user hovers their mouse over the element for a second or so. Example, `title='more information'`.

It is a good way to provide hints or extra information that would otherwise clutter the screen.

I personally think that "title" is really a horrible name for this attribute because we already have something else called title. The whole document has a title which is specified in the head with the syntax `<title>...</title>`, and which is typically displayed in the tab of the browser. The two titles are totally different.

I think a better name would be "popup" but for historical reasons it is called title. This new thing called title can apply to any portion of a webpage.

Exam Question 89 (p.341)**:** What does the <title> tag do?
Acceptable Answer: gives a name to the whole webpage

Exam Question 90 (p.341)**:** What does the "title=" attribute do?
Acceptable Answer: provides information that is invisible until the user hovers over it

4.4.5 hidden= (Advanced)

hidden= specifies that the item will not be rendered. It is invisible and takes up no space.

It may be most useful in connection with JavaScript that can remove the hidden attribute when you want to make something appear.

JavaScript: `(something).removeAttribute("hidden");`

Chapter 5

Image Crop and Resize

Contents

One of the most popular things to include in a webpage is an image. In this chapter we show you how to do some basic manipulation of images: cropping and resizing.

First we will review the basic HTML options.

Exam Question 91 (p.341)**:** The HTML tag "img" stands for what word(s)?
Acceptable Answer: image

Exam Question 92 (p.341)**:** What HTML markup is used to insert a picture?
Acceptable Answer: `<img src=...>`

The image tag is `<img ...>`. It is a void tag.

Exam Question 93 (p.341)**:** For each <img> tag in valid HTML5, is a

separate closing tag required, optional, or forbidden?
Required Answer: forbidden

The commonly-used attributes of image are:

src= specifies the source of the image. This is a URL. See section 6.12 (page 68) for more about URLs. The source can be in the same folder as the webpage (relative addressing) or anywhere on the Internet (absolute addressing).

alt= specifies what should be used if the image is not usable. This is a required field because some users (especially blind users) cannot see images.

Exam Question 94 (p.341)**:** For each <img> tag in valid HTML5, is the "alt=" attribute required, optional, or forbidden?
Required Answer: required

Normally an image will be displayed in its natural size. This may be much smaller or larger than you would like. Width and height are two ways to control how the image is presented.

width= specifies how many pixels wide the displayed image should be. If the original image is narrower or wider, it will be stretched or squeezed to fit.

Exam Question 95 (p.342)**:** For each <img> tag in valid HTML5, is the "width=" attribute required, optional, or forbidden?
Required Answer: optional

height= specifies how many pixels tall the displayed image should be. If the original image is shorter or taller, it will be stretched or squeezed to fit.

With width and height, we are talking here about CSS pixels, not real pixels. See section 1.2 (page 10) for a reminder about the difference.

If only one of **width=** and **height=** is specified, the other will be automatically calculated to maintain the original **aspect ratio**, the ratio of width to height.

If both are specified, you can distort the image in fun ways.

Exam Question 96 (p.342)**:** List the four commonly-used attributes of the image tag.
Required Answer: src, alt, width, height

5.1 Basic Image Processing

Cropping removes part of the picture. It creates a smaller picture. It lets you direct your viewers' attention to the things you want them to see.

Exam Question 97 (p.342)**:** In image processing, what word describes removal of part of the picture, making it smaller?
Acceptable Answer: crop

Resizing presents the same image as before, but it changes the number of pixels in the picture. This can be done automatically by the browser, to make your image fit in the space you have allocated. It can also be done before the webpage is loaded to conserve on network bandwidth. It is sometimes called **scaling** or **resampling** or **downsampling** or **upsampling**.

We are talking here about real pixels, not CSS pixels. See section 1.2 (page 10) for a reminder about the difference.

Downsampling: When we take a larger number of pixels and generate a smaller number, that is called downsampling.

Exam Question 98 (p.342)**:** In image processing, what word describes combining pixels, making the image smaller?
Acceptable Answer: downsampling

You can use scaling to subdivide pixels, creating more of them. When you do it the effort is usually wasted because you do not get any more detail and your downloads take more time. When the browser does it, it is to match the pixels it received to the physical device on which the pixels will be shown.

Upsampling: When we take a smaller number of pixels and generate a larger number, that is called upsampling.

Exam Question 99 (p.342)**:** In image processing, what word describes dividing pixels, making the image larger?
Acceptable Answer: upsampling

Color modification and transparency are also commonly done. We will not demonstrate color modification, but in section 8.1 (page 82) and in appendix 27 (page 248) we will work with image transparency.

5.1.1 Size Affects Speed

Each of us can probably remember visiting a webpage that loaded, a, large, photo, so, slowly, we, almost, gave, up, waiting.

Modern digital cameras take big pictures. 4000 by 3000 is not uncommon, and works out to 12 megapixels. The finished images are in the range of 2 to 3 megabytes in size. (I saw a smart phone advertised with 41 megapixel resolution.) Images are getting bigger.

Google search "camera megapixel statistics" for current numbers.

On the other hand, even though modern computer screens are also getting bigger, they are still much smaller than camera images, perhaps 1600 by 1200 toward the high end and 1024 by 768 at the low end.

Google search "screen resolution statistics" for current numbers.

Within that space, only a fraction is taken up by any given image. The displayed size of an image may only be something like 400 by 300 for a large image, and they are often much smaller than that.

When we view a 4000 by 3000 image, but only show 400 by 300 pixels, the full image must be "resampled" by the browser. It also means that for every pixel shown, 100 pixels were downloaded.

Internet download speeds in the US are typically (2011) somewhere around 10 Mbps, ten megabits per second. A 2.5 megabyte image has 25 megabits, and would take 2.5 seconds to download. The reality is probably two to three times as long. 7 seconds feels like a long time to wait.

But 0.07 seconds, 1/100 of that original time, is hardly noticeable.

The bottom line is simple. When you are displaying pictures on web pages, and a lot of detail is not needed, it pays to downsample them in advance, and crop them to the exact part of the image you want to show. Sometimes image compression is done as well.

Pick your target size, and then work toward it. We will show you how.

5.1.2 Photo Editing Tools

Photoshop is legendary enough that people use photoshopping as a verb. And it is the most well-known photo editing software today. Most photo professionals use Photoshop.

Adobe Photoshop is a commercial product costing around $700. Many schools make the educational version available to their students. The educational price is typically half or less compared to the regular price.

Gimp (more properly, The Gimp) is a free alternative to Photoshop, and has many of the same features. Gimp is popular among open-source advocates and people with small budgets. (Free is a really good price.) Appendix 26 (page 246) has more.

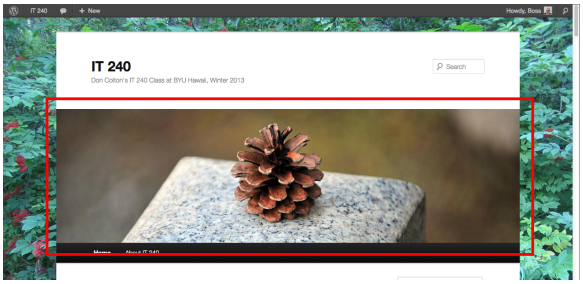
Since students are typically "poor, starving students," I have chosen to demonstrate photo editing using The Gimp. Also, it helps that I can install it on any machine I want for free. Because Gimp is free and has capabilities roughly equal to Photoshop, Gimp is the photo editor that I most frequently use.

Training: It is also said that the most expensive part of any computing system is the training. Nothing is really free. Training is usually required. Capabilities between the two editors are quite similar, but Photoshop probably has better support in terms of how-to books and documentation.

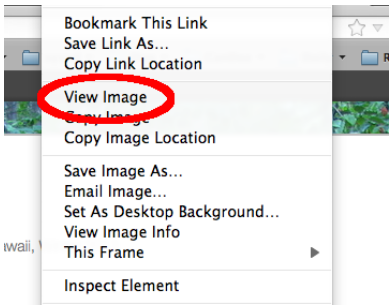
5.2 Cropping Removes Pixels

Cropping removes part of an image. In this example I use Gimp 2.8 and show the steps I took to create a 1000 by 288 banner for a WordPress website.

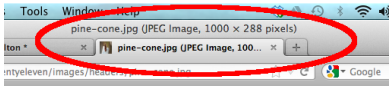
In this picture, I put a red box around the part of the webpage that is the header image. I intend to replace it with my own image. I want to find out its dimensions.



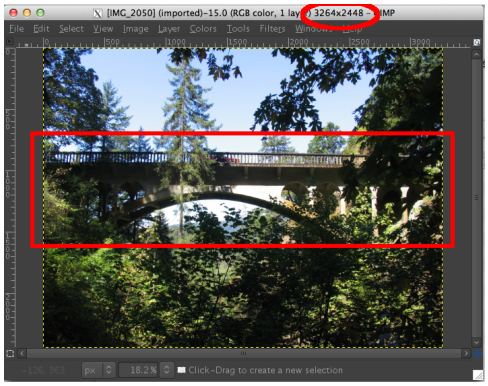
Using the Firefox browser, I did a right-click on the image. I selected "View Image" from the menu. I could also have selected "View Image Info" farther down the menu to learn what I wanted to know.



Viewing just the image, I can see that its dimensions are 1000 x 288 pixels. I repeated the process for another header image to make sure the dimensions were the same. They were.



Here I have started Gimp and I am looking at a full-sized original image from which I hope to extract a portion to become a new header image.

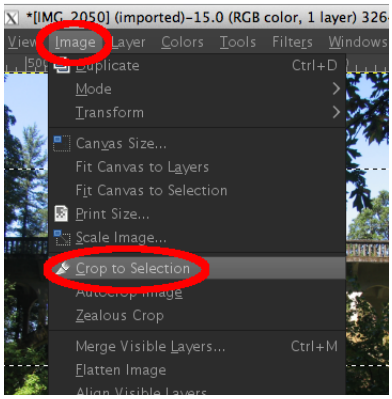


This is a picture of the bridge at Shepherd's Dell, a waterfall on the Oregon side of the Columbia River gorge, taken from the base of the waterfall and looking up towards the road.

You can see in the red circle at the top that the original size of the image is 3264 x 2448 pixels, and you can see in the red box the approximate portion I hope to extract.

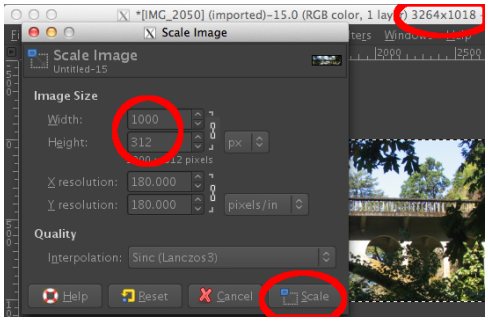
Using Gimp's rectangle select tool, I have drawn a line around the part to be kept. The rest will be cropped away.

From the menu, I select **Gimp Menu: Image / Crop to Selection**.



5.3 Scaling Combines Pixels

As you can see in the red circle at the top, the resulting image is now 3264 (the full original width) x 1018 (the new height) pixels.



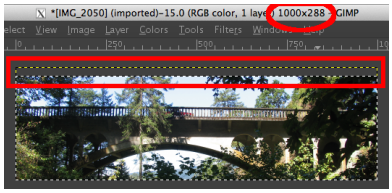
Our next task is to Scale the Image. From the menu, I select **Gimp Menu: Image / Scale Image**.

In the circle toward the middle, I change the width to 1000 and it shows me

the height would be 312. (If I change the height to 288, the width is less than 1000, and I do not want that.) Then I press the "Scale" button.

The result will be 1000 x 312 pixels.

Cropping by Canvas Size



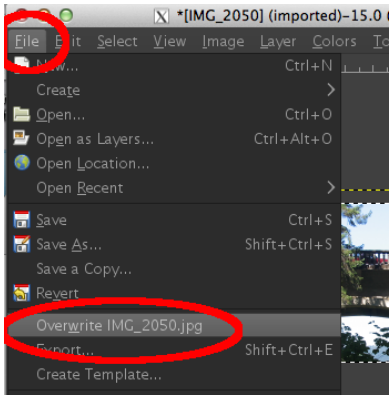
I need to make the image smaller, and exactly 1000 x 288 pixels. The easiest way to do that is by changing the Canvas Size. This is exactly like cropping, but instead of using the mouse to specify the size, I type in exact numeric values. Then I can use the mouse to position the image within that space.

From the menu, I select **Gimp Menu: Image / Canvas Size**.

I decide to keep the bottom part of the image. Within the red box you can see where the top part of the image has been removed. My image is now exactly the size I want.

Export the Resulting Image

From the menu, I select **Gimp Menu: File / Overwrite**. My new image is written in place of the original. Mine was just a copy. But if you wanted to keep the original, you would choose the next option, **Gimp Menu: File / Export**.



5.4 Advanced Tricks

There are a few things that commonly happen with images. We flow text around them. We put captions above or below them. We click on them to

see a more detailed view. In this section we show how to do these advanced tricks.

5.4.1 Flow Around

Sometimes we want to insert a picture next to some text. We do not want the picture to take up the whole row, but maybe be on the left or right edge of the row, and have the text flow around it.

This can be easily done using the css float: parameter. We will talk about float and clear later, but here is a short example.

```
<img src=abc.jpg width=200 style="float: left;">
```

The text that follows will appear to the right of the picture until the picture ends, and then will take up the whole column.

5.4.2 Captions

Sometimes we want to associate some text with a picture. We might think of this text as a caption.

This can be easily done using a table. We will talk about tables later, but here is a short example. (`tr` means table row. `td` means table data.)

```
<table style="float: left;">
<tr><td><img src=abc.jpg width=200>
<tr><td style="text-align: center;">
This is my caption.
</table>
```

5.4.3 Art Gallery

Sometimes we want a more-detailed image to be available for those that are interested, but we do not want to slow down the webpage for everybody else.

An example of this might be an art gallery page that shows many images at a moderate resolution, like maybe 500px in the larger dimension, so they will load quickly.

We can save two versions of each image: a smaller one for display to everyone, and a larger one for display to those users that are particularly interested. The larger one can be available by clicking on the smaller image.

We can achieve this by wrapping the smaller image inside a link to the larger image.

Exam Question 100 (p.342)**:** How can you show a small image that when clicked on shows a larger image?

Acceptable Answer: `<a href=big.jpg><img src=small.jpg></a>`

Chapter 6

Using External Resources

Contents

We have been talking about images, which are generally done up as separate files and then inserted into webpages.

Where do you put those files? There are three main answers.

Relative: Files that relate to just a particular webpage can easily be stored in the same folder as the webpage itself. They can be simply identified by name. For example:

```
src=mypic.jpg
```

Docroot: Files that relate to the whole website can best be stored in a special folder or group of folders. For example:

```
src=/images/logo.png
```

The slash components in front of the filename are called the path.

Absolute: Files that are borrowed from elsewhere can be identified by their full URL. For example:

```
https://commons.wikimedia.org/wiki/File:Stonehenge.jpg
```

6.1 Besides Images

Images are a good example of separate files, but there are a few other common examples of things that are separate.

External Style Sheets are often identified by a docroot-style name, as they typically apply to many pages in a website.

Internal Page Links are mostly relative or docroot. They link to separate webpages within the same website. Closely related pages are relative. General pages are docroot.

External Page Links are always absolute. They link to separate webpages in a different website.

Webfonts are normally absolute. They are typically brought in from someplace like Google as needed.

Scripts can be imported from absolute locations, but for security purposes they are often imported from within the same website as the page that uses them.

6.2 URL

Exam Question 101 (p.342)**:** Are spaces allowed in filenames?
Required Answer: no

Because spaces are not allowed in filenames, what can you do to get around that limitation? There are several commonly-used options.

(a) You can simply delete the spaces. thisisaspacedeletedexample. It can make long filenames difficult to understand.

(b) You can use a camel-case naming strategy. This is where each word of the filename is capitalized, and spaces are deleted. ThisIsACamelCaseFilenameExample. Readability is good.

(c) You can substitute underscores. this_is_a_filename_with_underscores. Readability is good.

(d) If you leave the spaces in the filename, then you must use `%20` in the URL.

Exam Question 102 (p.342)**:** What does %20 mean in a URL?
Required Answer: space

Exam Question 103 (p.342)**:** We identified three kinds of URLs. One is relative. List the other two.
Required Answer: docroot, absolute

6.3 Links

6.3.1 a: Anchor

The anchor tag is used to create hyper-text links, either within a page or between pages.

The most important attribute is the **href=** attribute. It tells where the browser should take you if you click on the link. Example:

```
<a href=/xyz.html>Go to XYZ</a>
```

The href= value can be specified as a relative address, a docroot-based address, or an absolute address.

6.3.2 Fragment in Same Page

Within your current page, you can use a Fragment ID to specify the place to which you want to jump.

```
<a href=#abc>Go to ID abc in this webpage</a>
```

```
<h2 id=abc>This is section abc</h2>
```

6.3.3 Fragment in Different Page

Within a different webpage, you can use a Fragment ID to specify the place to which you want to jump.

```
<a href=/xyz.html#abc>Go to ID abc in page xyz</a>
```

6.3.4 target=_blank

The **a tag** has an attribute **target** that can be used for several purposes, the most important of which may be causing a new tab to open in the browser.

```
target=_blank
```

```
<a href=/xyz.html target=_blank>Go to XYZ in a new tab</a>
```

BE CAREFUL with this as it can be very annoying to the user.

Historically, HTML4 walked away from this capability, believing that the choice of whether to open a new tab should be left to the user, and not under the control of the webpage designer.

HTML5 has reversed that decision, and `target=_blank` is now established as a standard part of the HTML language.

Exam Question 104 (p.342)**:** To open a link in a new tab, what HTML attribute=value should you use?
Acceptable Answer: `target=_blank`

6.3.5 Docroot and Path

As the website developer, you must put the content into the proper directories and files so it can be found when it is requested.

When a webpage is requested, the webhost splits the URL into a domain

name and a path. The domain name is used to find the document root. The path is used to find a file within that document tree.

If our URL is `http://example.com/a/b/c/d.html` the webhost will split that into two parts:

Domain: `example.com`

Path: `a/b/c/d.html`

For the domain, the webhost will remember the document root. Maybe it is something like this:

Document root: `/home/example/public_html`

The server replaces the domain name with the document root, resulting in this actual filename:

`/home/example/public_html/a/b/c/d.html`

Exam Question 105 (p.342)**:** Is the domain name within a URL case-sensitive?
Required Answer: no

Exam Question 106 (p.342)**:** Is the path within a URL case-sensitive?
Required Answer: yes

6.4 URLs

A fully-complete URL is absolute. It consists of a scheme, such as http, a domain name, such as example.com, and a path, such as folder/index.html. It would look like this:

`http://example.com/folder/index.html`

This is called an **absolute URL**, or an absolute address.

HTML allows you to leave out parts of the URL, thus making the URL less absolute and more relative.

Actually, a fully-specified URL has even more parts, including the query and the port number, but we will ignore them here.

6.5 Why Not Absolute?

Say we were writing instructions for installing a refrigerator. We might want to say "put it in the kitchen." But which kitchen?

Kitchen is relative to where we are. If I am standing at 123 Main Street, Anywhere, USA, it would mean the kitchen on that same property.

If we have to be absolute, we would have to write "put it in the kitchen of 123 Main Street, Anywhere, USA". And then the instructions would only work for that one house in the world.

On top of that, it is a bit tedious when talking to humans to add those extra layers of absoluteness.

Our absolute instructions are brittle, inflexible, and fragile. When we rewrite them for a different house, we have to be careful to make all the necessary corrections.

Webpages also refer to places and things. They use URLs to do it. We could make all our URLs absolute. But they would be brittle and fragile as well.

Sometimes we want absolute addressing. But when possible, we want relative addressing.

Things change. URLs change. Today your company website might be example.org, and next year it might be example.com.

Or you want to repurpose some work you did. And rather than go through your whole website looking for example.org and replacing it with example.com, you just wish it could be done once and for all.

6.6 Relative To Our Base

Our base is the webpage where we currently are. The other URL is the place we want to link to or pull a picture from. Here is an example. (1) is our base, where we are. (2) is what we want.

(1) `http://example.com/abc/def/ghi/index.html`

(2) `http://example.com/abc/def/ghi/photo.jpg`

The absolute way of talking about (2) would be this:

`http://example.com/abc/def/ghi/photo.jpg`

The relative way of talking about (2) would be this:

`photo.jpg`

With the relative way, we assume we are in the same folder on the same website. This is called a **relative URL**.

Both ways work, but if we ever have to move that webpage to a new location, absolute URLs need to be corrected. Relative URLs usually need no correction because they are still right.

6.7 Sibling, Cousin, Second Cousin

If we want a file in the same folder as the webpage, the relative URL is simple, as in the previous example. If (1) is our absolute address, and (2) is the relative address, then (3) is the matching absolute address. This is like being a sibling.

(1) `http://example.com/abc/def/ghi/index.html`

(2) `photo.jpg`

(3) `http://example.com/abc/def/ghi/photo.jpg`

We can also refer to the current directory by putting `./` in front of the file name. Thus, `photo.jpg` means the same thing as `./photo.jpg` does. Dot means where we are at right now. Dot dot means our parent directory. Sometimes the dot slash is added for clarity or to avoid ambiguity, but normally it is not written.

A file in the next higher (parent) directory has a relative address.

`../photo.jpg`

We can use `..` to go up a level from where we are now. If (1) is our absolute address, and (2) is the relative address, then (3) is the matching absolute address. This is like being an uncle or aunt.

(1) `http://example.com/abc/def/ghi/index.html`

(2) `../photo.jpg`

(3) `http://example.com/abc/def/photo.jpg`

We can use `..` as many times as we want. `../..` takes us up two levels from where we are now. If (1) is our absolute address, and (2) is the relative address, then (3) is the matching absolute address. This is like being a great

uncle or great aunt.

(1) `http://example.com/abc/def/ghi/index.html`

(2) `../../photo.jpg`

(3) `http://example.com/abc/photo.jpg`

We can also go the other way. If (1) is our absolute address, and (2) is the relative address, then (3) is the matching absolute address. This is like being a nephew or a niece.

(1) `http://example.com/abc/def/ghi/index.html`

(2) `jkl/photo.jpg`

(3) `http://example.com/abc/def/ghi/jkl/photo.jpg`

And we can combine the two directions. Go up a level, then turn around and come back down a level. If (1) is our absolute address, and (2) is the relative address, then (3) is the matching absolute address. This is like being a cousin.

(1) `http://example.com/abc/def/ghi/index.html`

(2) `../jkl/photo.jpg`

(3) `http://example.com/abc/def/jkl/photo.jpg`

If (1) is our absolute address, and (2) is the relative address, then (3) is the matching absolute address. This is like being a second cousin.

(1) `http://example.com/abc/def/ghi/index.html`

(2) `../../jkl/mno/photo.jpg`

(3) `http://example.com/abc/jkl/mno/photo.jpg`

6.8 DocRoot: Current Domain, Absolute Path

Relative addresses are relative to the domain name and path of the current webpage.

We can create addresses that are relative to the current domain name, but not the current path.

I will call these **DocRoot URLs**.

We do that by starting the address with a slash.

If (1) is our absolute address, and (2) is the docroot address, then (3) is the matching absolute address.

(1) `http://example.com/abc/def/ghi/index.html`

(2) `/pqr/photo.jpg`

(3) `http://example.com/pqr/photo.jpg`

In this case, because (2) starts with a slash, we throw away the entire path from (1), leaving us with just the domain name. Then we add the path from (2) giving us the result in (3).

6.9 img src=...

When creating an image (img) tag, the source (src) must be specified. It can be (R) relative, (D) docroot, or (A) absolute. (L) local is a form of absolute, but it is only visible on the original computer. The file is not online.

(R) `<img src=../pqr/photo.jpg>` - relative

(D) `<img src=/pqr/photo.jpg>` - docroot

(A) `<img src=http://example.com/pqr/photo.jpg>` - absolute

(L) `<img src=file:///whatever/photo.jpg>` - local

6.10 a href=...

When creating an anchor (a) tag, the http reference (href) must be specified. It can be (R) relative, (D) docroot, or (A) absolute. Again, (L) local is a form of absolute, but it is only visible on the original computer. The linked webpage is not online.

(R) `<a href=../pqr/index.html>` - relative

(D) `<a href=/pqr/index.html>` - docroot

(A) `<a href=http://example.com/pqr/index.html>` - absolute

(L) `<a href=file:///whatever/file.html>` - local

6.11 We Can Change Our Base

Your natural base for calculating relative URLs is the complete URL of your current webpage.

HTML allows you to specify a different base than your natural base. You can include one base declaration in the head of your webpage.

```
<base href=http://example.org/abc/>
```

That would mean "please act as though this webpage were actually located at http://example.org/abc/ even though you may have come to it by using a different URL. This can be handy if the webpage has one or more aliases.

All the relative addresses in the webpage would use the declared base instead of the natural base.

Exam Question 107 (p.342)**:** If our base URL is http://a.com/b/c/ and our stated URL is ../d/e/ what is our final URL?
Required Answer: http://a.com/b/d/e/

Exam Question 108 (p.342)**:** If our base URL is http://a.com/b/c/ and our stated URL is ./d/e/ what is our final URL?
Required Answer: http://a.com/b/c/d/e/

Exam Question 109 (p.342)**:** If our base URL is http://a.com/b/c/ and our stated URL is /d/e/ what is our final URL?
Required Answer: http://a.com/d/e/

Exam Question 110 (p.343)**:** If our base URL is http://a.com/b/c/ and our stated URL is d/e/ what is our final URL?
Required Answer: http://a.com/b/c/d/e/

If the stated URL starts with slash, we throw away all the path from the base URL and just add the stated URL.

For any other relative address, we combine the base URL and the stated URL. Then, every time we find ../ we delete it and the previous chunk of path. Every time we find ./ we just delete it.

6.12 Connecting With External Resources

A webpage is seldom complete in itself. Most often it refers to images and other webpages. Also it refers to external style sheets.

These external resources are identified by **URL**: Uniform Resource Locator.

6.12.1 img src=

"img" stands for image. The HTML **img tag** provides a way to retrieve an image for display in the current webpage. The src attribute specifies the URL where the image can be found.

6.12.2 a href=

In the **a** tag, a stands for anchor. The HTML **a tag** provides a link to another webpage. By clicking on the link, the user causes the other webpage to be loaded. The typical usage of the **a tag** is as follows:

```
<a href=aaa>bbb</a>
```

The aaa part is a URL that identifies the page to be loaded next.

The bbb part is text (or an image, or both) that is clickable to cause that page load.

```
<a href=aaa><img src=ccc></a>
```

In this version, ccc is the URL of the image to be displayed.

Fragment ID

At the tail end of a URL, an ID can be specified. This is also called a **fragment id**.

It is introduced by a hashtag, and requests the browser to (a) load the page specified, and (b) position the page so that the requested ID is visible, usually at the top of the browser's viewing area.

```
<a href=xxx#abc>yyy</a>
```

This creates a clickable link that displays the word yyy. When clicked, the webpage xxx is loaded, and within that webpage, the ID abc is sought. If found, it will become the focus of the webpage.

```
<h1 id=abc>yadda</h1>
```

This creates a target within the webpage. The target happens to be an h1 tag. It has an ID of abc and displayed content of yadda. By specifying an

ID, we can jump directly to this h1 when the webpage is loaded.

(There are other, obsolete ways to create a fragment ID target, but they are not recommended and we do not reveal them here.)

6.12.3 script src=

The **script tag** has two flavors: external and immediate.

```
<script src=aaa></script>
```

When the src attribute is specified, a JavaScript script will be retrieved from the URL that is given (aaa in this example). This JavaScript will then be executed immediately before the rest of the page is rendered.

The URL of this src=URL method can refer to static (unchanging) code, but more usefully it can refer to a **CGI** program that generates the code dynamically, to display a pageview count, or to display a quote of the day.

```
<script>bbb</script>
```

When the src attribute is not specified, the JavaScript is required to appear immediately after the **script tag**. It is bbb in this example. It will also be executed immediately.

```
<script src=aaa>bbb</script>
```

If both src and an immediate script are provided, the src takes priority. The bbb is ignored.

6.12.4 link

The head portion of a webpage can include several items by reference, including CSS style sheets.

```
<link rel=stylesheet href=xxx>
```

In this example, a style sheet is sought at URL xxx, and if found is loaded.

Chapter 7

Colors

Contents

Black and white: boring. Color: exciting. Color can attract our attention. It can affect us emotionally. It can be memorable.

CSS allows us to apply color in several ways. We can color the lettering of our webpage. We can color the background of all or part of our webpage. We can color the borders of things. We can use gradients that blend smoothly from one color to another.

Exam Question 111 (p.343)**:** What CSS attribute: sets the color of your lettering?
Required Answer: color:

Exam Question 112 (p.343)**:** What CSS attribute: sets the color behind your lettering?
Required Answer: background-color:

7.1 How To Specify Colors

There are many ways to specify colors. For webpages these include predefined name, hex code, rgb code (red-green-blue), and hsl code (hue-saturation-lightness). CMYK (cyan-magenta-yellow-black) is popular for printing but not reliable for webpages.

These examples show different ways to set the background color to be orange.

```
body { background-color: orange; } /* named color */
body { background-color: #FFA500; } /* hex code */
body { background-color: rgb(255,165,0); }
body { background-color: rgba(255,165,0,1.0); }
body { background-color: rgba(255,165,0,100%); }
body { background-color: hsl(39,100%,50%); }
body { background-color: hsl(39,1,.5); }
body { background-color: hsla(39,100%,50%,1.0); }
body { background-color: hsla(39,100%,50%,100%); }
```

Names: Predefined color names are things like red, green, blue, white, black, and gray. There are 17 standard color names in CSS 2.1, and a total of 140 that are understood by all modern browsers. In section 7.8 (page 79) we list all 140 of them.

Hex codes are expressed as a hashtag followed by three or six "digits" in the hexadecimal numbering system (base 16). These numbers range from 0 (the absence of the color) through 9 and A through F, for a total of 16 different "digits." F is the highest intensity. Using a six-digit code, red, green, and blue range from 00 to FF for 256 possible intensities. If only three digits are specified, each digit is treated as though it was used twice. Thus, #123 means #112233.

RGB (red, green, blue) numbers are exactly equivalent to hex codes, but instead of being expressed in hexidecimal they are written as a list in base 10. The values range from 0 (no color) to 255 (total color).

RGBA (red, green, blue, alpha) is like RGB but has an additional parameter to control **transparency** using an **alpha** value. Alpha ranges from 0.0 (transparent) to 1.0 (**opaque**).

HSL and **HSLA** (hue, saturation, lightness, alpha) are also usable on webpages.

Hue refers to angle on a **color wheel**, where 0 (or 360) is red, 120 is green, and 240 is blue.

Saturation refers to color vs gray scale. 0% is gray, like an old black-and-white television. 100% is full color.

Lightness refers to tint and shade, as though you were adding paint to the color, white (tint) to lighten it, or black (shade) to darken it. 0% means completely black (maximum shade). 100% means completely white (maximum tint). 50% is natural color, without shade or tint.

Tint means to add white. Tone means to add gray (equal parts of white and black), which is accomplished by way of the saturation control. Shade means to add black.

7.2 The RGB Additive Model

Web colors are based on an additive model, using Red, Green, and Blue light as the nominal components. (I say "nominal" because the green is really more of a lime color.) The more colors you put in, the lighter the color becomes.

Regular painting is based on a subtractive model. The more colors you put in, the muddier the paint becomes.

Exam Question 113 (p.343)**:** In rgb, what does the r stand for?
Required Answer: red

Exam Question 114 (p.343)**:** In rgb, what does the g stand for?
Required Answer: green

Exam Question 115 (p.343)**:** In rgb, what does the b stand for?
Required Answer: blue

7.3 Layers and Transparency

Subtitles on television or movies are often provided for alternate languages or for hearing impaired viewers. Often these subtitles are overlaid right onto the movie itself.

HTML allows for layers of material. You can have a background image with content that overlays the image. You can have a picture that has

words overlaid onto it. You can have one picture overlaid on top of another. Maybe we can see through parts of that picture to the picture that is below.

In the RGB model there is also a component for **transparency** or **opacity**, called the **alpha** channel.

Exam Question 116 (p.343)**:** In rgba, what does the a stand for?
Required Answer: alpha

Exam Question 117 (p.343)**:** What does the alpha channel control?
Acceptable Answer: opacity

A value of 1 or 100% means totally opaque.

A value of 0 means totally transparent.

A background color of rgba(255,255,255,0.7) would be 70% opaque white. That is, you would be able to see through it like looking through a thin film of white paper.

A background color of rgba(0,0,0,0.5) would be 50% opaque black. That is, you would be able to see through it like looking through a thin film of black paper.

Contrast: Such rgba background colors can be very useful when the content you are showing is on top of a background image that is very busy. If the background image swallows up your content, you might want to put a layer of partly transparent white or black between the content and the ultimate background.

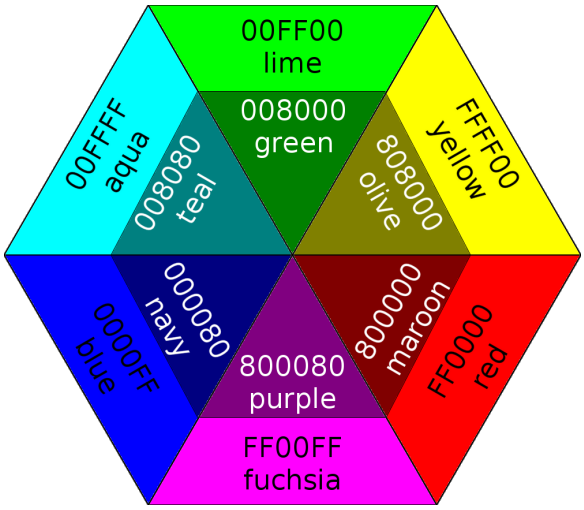
7.4 The Web Color Wheel

Because RGB is the most natural and commonly used way of specifying colors on the web, let us try to develop some intuition about how these numbers relate to the colors we might want.

The outer ring of the color wheel uses FF (full intensity) and 00 (absence) to create the primary and secondary web colors.

The inner ring of the color wheel uses 80 (half intensity) and 00 (absence) to create shaded versions of the primary and secondary web colors.

Absent from these 12 colors is orange.



CSS version 2.1 gives official names to 17 colors, including the 12 shown in this color wheel. It also includes the color orange and four desaturated colors: white, silver, gray, and black.

Here is the complete list of 17: **aqua**, **black**, **blue**, **fuchsia**, **gray**, **green**, **lime**, **maroon**, **navy**, **olive**, **orange**, **purple**, **red**, **silver**, **teal**, **white**, and **yellow**.

Fuchsia is named for German botanist Leonhart Fuchs. The incorrect spelling, fuschia, is pretty common, and may even work in some browsers, but is still incorrect.

Exam Question 118 (p.343)**:** What color is named for a German botanist?
Required Answer: fuchsia

Gray is also commonly spelled grey. Use gray.

Exam Question 119 (p.343)**:** What is color `#ffffff` officially named?
Required Answer: white

Exam Question 120 (p.343)**:** What is color `#c0c0c0` officially named?
Required Answer: silver

Exam Question 121 (p.343)**:** What is color `#808080` officially named?
Required Answer: gray

Exam Question 122 (p.343)**:** What is color `#000000` officially named?
Required Answer: black

Exam Question 123 (p.343)**:** What is color `#ff0000` officially named?

Required Answer: red

Exam Question 124 (p.343)**:** What is color `#800000` officially named?
Required Answer: maroon

Exam Question 125 (p.343)**:** What is color `#00ff00` officially named?
Required Answer: lime

Exam Question 126 (p.343)**:** What is color `#008000` officially named?
Required Answer: green

Exam Question 127 (p.343)**:** What is color `#0000ff` officially named?
Required Answer: blue

Exam Question 128 (p.343)**:** What is color `#000080` officially named?
Required Answer: navy

Exam Question 129 (p.343)**:** What is color `#ffff00` officially named?
Required Answer: yellow

Exam Question 130 (p.344)**:** What is color `#808000` officially named?
Required Answer: olive

Exam Question 131 (p.344)**:** What is color `#ffa500` officially named?
Required Answer: orange

Exam Question 132 (p.344)**:** What is color `#ff00ff` officially named?
Required Answer: fuchsia

Exam Question 133 (p.344)**:** What is color `#800080` officially named?
Required Answer: purple

Exam Question 134 (p.344)**:** What is color `#00ffff` officially named?
Required Answer: aqua

Exam Question 135 (p.344)**:** What is color `#008080` officially named?
Required Answer: teal

7.5 Gradients

A gradient is a background image that is automatically generated by the browser according to your directions. It smoothly blends two or more colors between color stops. There are two types of gradients currently available:

linear and radial.

Exam Question 136 (p.344)**:** Gradient is an example of what CSS attribute?
Acceptable Answer: background-image:

Exam Question 137 (p.344)**:** List the two types of gradients.
Required Answer: linear, radial

Color Stops

The background image is presented as bands of color. For linear gradients it starts at one edge of the image and moves to the other edge. For radial gradients it starts at the center and moves to the outside.

For each color stop you specify a color in one of the usual ways (by name, hex, rgb, etc.). The first color stop will be at 0% (the starting point) and the last will be at 100% (the ending point), with any other color stops evenly spaced between. You can override the spacing by giving specific locations (like 50%).

If you put two color stops at the same location, the blending will be instant, which means that you will get an immediate change of color instead of a gradual blend.

Linear Gradients

Linear gradients also have an angle: 0deg (zero degrees) goes from bottom to top. 90deg goes from left to right. 180deg goes from top to bottom and is the default. 270deg goes from right to left.

Here is an example of a linear rainbow gradient with red on the left:

```
h1 { background-image: linear-gradient( 90deg,
 red, orange, yellow, green, blue, indigo, violet ); }
```

Radial Gradients

Radial gradients have a starting point and a shape.

The starting point defaults to the center of the space. A different center

point can be specified as a percentage distance from the top or bottom and a percentage distance from left or right.

The shape defaults to ellipse with an aspect ratio that matches the space you are filling. You can specify circle if you want.

Here is an example of a radial rainbow gradient with red in the center:

```
h1 { background-image: radial-gradient( circle,
 red, orange, yellow, green, blue, indigo, violet ); }
```

Repeating Gradients

You can also have a gradient repeat, just like other background images can repeat. (We will not present the details here.)

7.6 Tools

Colorzilla (http://www.colorzilla.com/) has a wonderful toolbox for Firefox and for Chrome. It also has a very nice gradient generator.

7.7 For Further Study

http://www.w3.org/wiki/Colour_theory has an excellent, in-depth article on color theory written by Linda Goin.

You may want to read more about colors and color theory. We recommend doing a Google search for phrases like these.

- color theory
- color wheel
- analygous colors
- complementary colors
- color and emotion

7.8 The 140 Named Colors

http://www.w3schools.com/cssref/css_colornames.asp gives a list with examples of the 140 colors that are supported by all browsers. (Actually this list has 141 colors but I am not sure which one is extra.)

AliceBlue	#F0F8FF	AntiqueWhite	#FAEBD7
Aqua	#00FFFF	Aquamarine	#7FFFD4
Azure	#F0FFFF	Beige	#F5F5DC
Bisque	#FFE4C4	Black	#000000
BlanchedAlmond	#FFEBCD	Blue	#0000FF
BlueViolet	#8A2BE2	Brown	#A52A2A
BurlyWood	#DEB887	CadetBlue	#5F9EA0
Chartreuse	#7FFF00	Chocolate	#D2691E
Coral	#FF7F50	CornflowerBlue	#6495ED
Cornsilk	#FFF8DC	Crimson	#DC143C
Cyan	#00FFFF	DarkBlue	#00008B
DarkCyan	#008B8B	DarkGoldenRod	#B8860B
DarkGray	#A9A9A9	DarkGreen	#006400
DarkKhaki	#BDB76B	DarkMagenta	#8B008B
DarkOliveGreen	#556B2F	DarkOrange	#FF8C00
DarkOrchid	#9932CC	DarkRed	#8B0000
DarkSalmon	#E9967A	DarkSeaGreen	#8FBC8F
DarkSlateBlue	#483D8B	DarkSlateGray	#2F4F4F
DarkTurquoise	#00CED1	DarkViolet	#9400D3
DeepPink	#FF1493	DeepSkyBlue	#00BFFF
DimGray	#696969	DodgerBlue	#1E90FF
FireBrick	#B22222	FloralWhite	#FFFAF0
ForestGreen	#228B22	Fuchsia	#FF00FF
Gainsboro	#DCDCDC	GhostWhite	#F8F8FF
Gold	#FFD700	GoldenRod	#DAA520
Gray	#808080	Green	#008000
GreenYellow	#ADFF2F	HoneyDew	#F0FFF0
HotPink	#FF69B4	IndianRed	#CD5C5C
Indigo	#4B0082	Ivory	#FFFFF0
Khaki	#F0E68C	Lavender	#E6E6FA
LavenderBlush	#FFF0F5	LawnGreen	#7CFC00
LemonChiffon	#FFFACD	LightBlue	#ADD8E6
LightCoral	#F08080	LightCyan	#E0FFFF
LightGoldenRodYellow	#FAFAD2	LightGray	#D3D3D3

LightGreen	#90EE90	LightPink	#FFB6C1
LightSalmon	#FFA07A	LightSeaGreen	#20B2AA
LightSkyBlue	#87CEFA	LightSlateGray	#778899
LightSteelBlue	#B0C4DE	LightYellow	#FFFFE0
Lime	#00FF00	LimeGreen	#32CD32
Linen	#FAF0E6	Magenta	#FF00FF
Maroon	#800000	MediumAquaMarine	#66CDAA
MediumBlue	#0000CD	MediumOrchid	#BA55D3
MediumPurple	#9370DB	MediumSeaGreen	#3CB371
MediumSlateBlue	#7B68EE	MediumSpringGreen	#00FA9A
MediumTurquoise	#48D1CC	MediumVioletRed	#C71585
MidnightBlue	#191970	MintCream	#F5FFFA
MistyRose	#FFE4E1	Moccasin	#FFE4B5
NavajoWhite	#FFDEAD	Navy	#000080
OldLace	#FDF5E6	Olive	#808000
OliveDrab	#6B8E23	Orange	#FFA500
OrangeRed	#FF4500	Orchid	#DA70D6
PaleGoldenRod	#EEE8AA	PaleGreen	#98FB98
PaleTurquoise	#AFEEEE	PaleVioletRed	#DB7093
PapayaWhip	#FFEFD5	PeachPuff	#FFDAB9
Peru	#CD853F	Pink	#FFC0CB
Plum	#DDA0DD	PowderBlue	#B0E0E6
Purple	#800080	RebeccaPurple	#663399
Red	#FF0000	RosyBrown	#BC8F8F
RoyalBlue	#4169E1	SaddleBrown	#8B4513
Salmon	#FA8072	SandyBrown	#F4A460
SeaGreen	#2E8B57	SeaShell	#FFF5EE
Sienna	#A0522D	Silver	#C0C0C0
SkyBlue	#87CEEB	SlateBlue	#6A5ACD
SlateGray	#708090	Snow	#FFFAFA
SpringGreen	#00FF7F	SteelBlue	#4682B4
Tan	#D2B48C	Teal	#008080
Thistle	#D8BFD8	Tomato	#FF6347
Turquoise	#40E0D0	Violet	#EE82EE
Wheat	#F5DEB3	White	#FFFFFF
WhiteSmoke	#F5F5F5	Yellow	#FFFF00
YellowGreen	#9ACD32		

Chapter 8

Image Transparency

Contents

Let's say we want to put our organization's logo on many of our webpages. Maybe it has a fancy shape, and is not merely square. But on webpages all images are square (or, more accurately, rectangular).

No problem, right? We can simply color the space outside of the logo to match the background of the webpage. It will blend in perfectly.

And that works, kind of.

There are two big problems with this approach. (1) What if we decide to change the background color of our webpages? Do we need to make up a new logo with the new background color? (2) What if we want to use the logo on many different pages with many different background colors?

The solution to these and similar problems is transparency.

In chapter 5 (page 49) we introduced the basic skills of cropping and resizing images. In this chapter we build on that knowledge by showing how to make part of an image transparent.

8.1 Irregular Shapes and Transparency

You probably know that the human eye contains a retina, and the retina is composed of cones and rods. The rods detect black and white. The cones detect only three colors: red, green, and blue.

Computer images commonly use the RGB model that is made up of layers or channels: red, green, and blue. Color printers commonly use the CMYK model that uses four colors of ink: cyan, magenta, yellow, and black.

The red, green, and blue channels each tell how much of that color of light should be shown at each point of the image.

For transparency, we simply invent a new channel to tell how transparent or opaque the image should be at each point.

To do irregular shapes, the easiest method is to use the alpha channel to create transparency in parts of the image. Not all file formats support transparency. PNG is probably the best one to use.

Exam Question 138 (p.344)**:** Does the .gif image file format support transparency?
Required Answer: yes

Exam Question 139 (p.344)**:** Does the .jpg image file format support transparency?
Required Answer: no

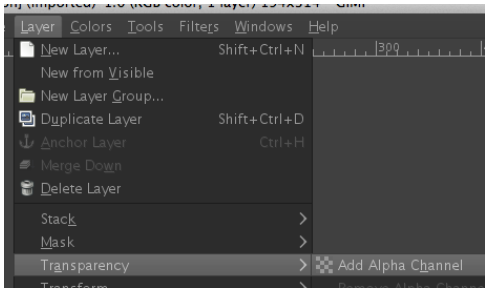
Exam Question 140 (p.344)**:** Does the .png image file format support transparency?
Required Answer: yes

In this section we show you how to make an irregularly-shaped image. Really, though, the image will be rectangular just like always, but parts of it will be transparent.

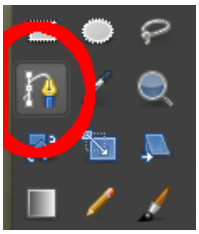
We will use a picture of myself. I have already cropped the picture to be very close to the material I want to keep. The original picture is much bigger. We will continue by deleting everything that is not me. We will make the deleted parts transparent. We will save the resulting image as a png file.

Here is the original picture we will use. It is 194x314 pixels, which will be a good size for our finished web graphic.

Using **Gimp Menu: Layer / Transparency / Add Alpha Channel**, we create the ability to have transparency in this image. (Red, green, and blue are also channels.)



We select the fountain-pen tool by double-clicking it. This will let me click my way around the important part of the picture, doing a selection that is irregular rather than rectangular or circular. This tool is actually much more powerful that what I am showing you. It creates a Bézier curve.



I started down at my neck, clicking with the fountain-pen tool along the edge of my head. You can see I have progressed up the left edge and am most of the way across the top of the image. The series of white circles indicate the locations where I clicked. It is useful to get as close as you can to the line between what you want and what you do not want. It can be helpful to view the image at a higher magnification to aid in following the boundary.

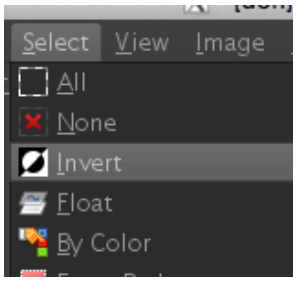
I have made it all the way around my picture. On the last click, instead of simply clicking, I hold down the control key while I click. This causes the loop to be closed, joining the last click with the first click. At this time, I can adjust any of the dots if I did not place them well. Just click on the dot and drag it. When the dots are all good enough, I press ENTER to select the region I have encircled.

If I delete now, I will delete my face because that is what is selected.

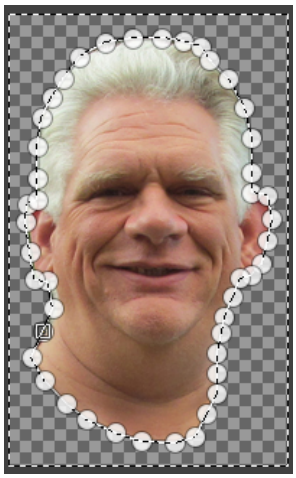
Notice the grey checkerboard pattern where my face was. This pattern is the commonly-used indication that the space is transparent.

Fortunately, there is **Gimp Menu: Edit / Undo**. I use it to get me back to where I was.

I use the Select / Invert command to invert my selection, thus selecting everything but my face. Now I am ready to delete again.



After deleting, just my face remains. The rest of the image has been replaced by transparency. I am ready to save. I will use **Gimp Menu: File / Export** to save the image as a **png** file. If I saved it as a jpg, the transparency would be replaced by white. After saving I can upload my png file to the web and use it as a graphic on a webpage.



Appendix 27 (page 248) gives a demonstration of how to create an image like a logo from scratch, without using a photograph.

8.2 Favicon

It is very popular to have a small image that represents your entire website, or specific pages of that website. The small image is called an **icon**.

In the old days, what we currently call bookmarks were called favorites. The icon used to represent the webpage or website was called the favorites icon, favicon.

Exam Question 141 (p.344)**:** What does favicon stand for?
Required Answer: favorite icon

In olden days, icons were created to create a brand identity for groups of webpages. There were special editors for creating them.

These days almost any image can be used for an icon. I have had success with .png files. I like that they allow transparency.

8.2.1 By Filename

The original method for displaying such icons was to create a specially formatted file called `favicon.ico` and put it in your document root folder.

This method will always work. Nobody would dare break all the websites that rely on it.

Your mileage may vary. Different browsers may act differently, but more and more they try to provide the same basic functions so web pages look and act the same across all major browsers.

Besides putting `favicon.ico` in the document root, it can sometimes also be placed in any directory of the website. Some browsers will use the favicon that is "closest" to the webpage that is being viewed. This is not totally reliable though.

Many browsers are flexible about the exact format of the favicon file, and will allow it to be a gif or jpg or png or something else. Just rename it to favicon.ico and things will normally work.

(The browser can internally examine the favicon.ico file to decide what format it is.)

8.2.2 By Link

Another method is available. This lets you have a different icon for each page of your website. You can specify a `link` tag in the head section of your html page.

```
<link rel=icon href=favicon.ico>
```

This lets you easily rename the favicon to something else, as in this example.

```
<link rel=icon href=myicon.png>
```

Although an icon is presumed to be an image, it could actually be a sound that is played. You should probably avoid that.

8.2.3 Many Sizes

Favicons can come in many sizes so the device can use the one that fits the best. The favicon.ico format actually supports a collection of images. And the link rel=icon method supports a sizes= parameter to tell which sizes are present in the file.

I would not bother having many sizes. Just go for a reasonably sized image, maybe 160x160, and let the browser resize it as needed.

There are a huge number of devices that use favicons, all the way from iPhones to big-screen TVs. You may want to create a dozen images or more of different sizes so the browser can use the one closest to its desired size. If you want exact control over how your favicon looks on a variety of platforms, you probably need to create all the common sizes: 16, 32, 48, 57, 72, 76, 114, 120, 144, 150, 152, 310.

Do a Google search on favicon to learn more about it.

8.3 Review of Tags and Attributes

In chapter 4 (page 36) we discussed HTML tags and attributes. Here we have some review questions for the chapter test.

Some HTML tags require closing tags. Others forbid them. And with some they are optional. These questions review your knowledge about closing tags.

div and span have not been fully introduced yet, but they appear among these questions anyway.

Exam Question 142 (p.344)**:** For each <head> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: optional

Exam Question 143 (p.344)**:** For each <body> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: optional

Exam Question 144 (p.344)**:** For each <h1> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 145 (p.344)**:** For each <h2> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 146 (p.344)**:** For each <p> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: optional

Exam Question 147 (p.344)**:** For each <b> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 148 (p.345)**:** For each <i> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 149 (p.345)**:** For each tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 150 (p.345)**:** For each <s> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 151 (p.345)**:** For each <u> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 152 (p.345)**:** For each <sub> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

Required Answer: required

Exam Question 153 (p.345)**:** For each <sup> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 154 (p.345)**:** For each <br> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: forbidden

Exam Question 155 (p.345)**:** For each <img> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: forbidden

Exam Question 156 (p.345)**:** For each <img> tag in valid HTML5, is the "width=" attribute required, optional, or forbidden?
Required Answer: optional

Exam Question 157 (p.345)**:** For each <img> tag in valid HTML5, is the "alt=" attribute required, optional, or forbidden?
Required Answer: required

Exam Question 158 (p.345)**:** For each <a> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 159 (p.345)**:** For each tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 160 (p.345)**:** For each tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Chapter 9

CSS: The Style Sheet

Contents

In Chapter 1 (page 8) we used a small amount of styling. We styled a background color to be yellow, and we styled an image to have a border. As we further learned in section 4.4.1 (page 46), there is a globally available attribute, `style=`, that can be applied to any HTML tag. It allows you to apply styling directly to that element. It is called in-line styling.

Often we want to apply similar styling to many elements of the same kind. Styling each one individually is tedious, and making changes later is truly awful. We want a general solution.

Our more general solution is the style sheet.

The `<style>` and `</style>` tags identify a style sheet. (You can have more than one.) Put your main style sheet someplace between your `<head>` and `</head>` tags.

Web browsers will also accept style sheet material other places in your document, but the best place to put it is in the head.

The material inside the style section is called an internal style sheet. It is possible to have an external style sheet that is shared among many webpages, but this internal style sheet is used by just this webpage.

The language used in your webpage is HTML, but within the style sheet, the language is called CSS, or Cascading Style Sheet language. It has somewhat different linguistic rules than HTML does.

Exam Question 161 (p.346)**:** What does CSS stand for?
Acceptable Answer: cascading style sheet

Within a style sheet, all CSS looks the same as this prototype:

```
target { attribute: values; attribute: values; ... }
```

The attribute list for a target is enclosed in curly braces. Individual attributes are separated from their values a colon, and terminated (separated from the next attribute) by a semi-colon.

Exam Question 162 (p.346)**:** Give the prototype for CSS directives.
Acceptable Answer: target { attribute: value; ... }

In this chapter we will look at targets that are actual HTML tags. Later, in section 10.1 (page 99), we will look at IDs and classes as targets.

We are familiar with the `<body>` tag. We can use it in CSS to apply styling to the whole body of the webpage.

```
body { background-color: yellow; }
```

If we include that line in our style sheet, the background of the whole webpage will turn yellow.

We can use a similar approach to control the styling of `<h1>`, `<h2>`, `<p>`, `<a href=...>` and all other tags.

9.1 Why Bother With Style Sheets?

People that create content generally do the markup on that content.

People that design styles generally do NOT create the content. It requires a different skill set that is probably more artistic.

It is widely accepted that the styling (CSS) and the content (HTML) should be kept separate so that changes to styling do not require any changes to the content.

The same holds true for JavaScript. It can be hidden inside the content, or it can be kept separate. It is better to keep it separate.

9.2 Style Sheet vs Inline Styling

To give all paragraphs a yellow background, we could insert the following line into a style sheet.

```
p { background-color: yellow; }
```

Inline Styling: To give one specific paragraph a yellow background, we could use the following in place of the `<p>` tag.

```
<p style="background-color: yellow;">
```

or: `<p style=background-color:yellow>`

9.3 background-color:

As we have just seen, the **background-color:** attribute can be used to control the background color of a webpage. We talked about color in Chapter 7 (page 71). Feel free to experiment.

Exam Question 163 (p.346)**:** What CSS attribute: sets the color behind your lettering?
Required Answer: background-color:

9.4 color:

The **color:** attribute can be used to control the foreground color of the text on a webpage.

Exam Question 164 (p.346)**:** What CSS attribute: sets the color of your lettering?

Required Answer: color:

To style all `<h1>` headers with green lettering on a pink background, we could use this style directive.

```
h1 { color: green; background-color: pink; }
```

9.5 font-size:

We can control the size of our lettering by using the **font-size:** attribute.

Exam Question 165 (p.346)**:** What CSS attribute: sets the size of your lettering?
Required Answer: font-size:

Note that **text-size:** is **not** a CSS attribute. Use **font-size:** instead.

To make all level-two headings twice as big as ordinary text, we could use the following directive.

```
h2 { font-size: 200%; }
```

font-size: can be measured in many different ways, including points. To make all bold text appear in 24-point size, we could use the following directive.

```
b { font-size: 24pt; }
```

As you will remember, there are 47 points in one central degree of vision, and 96 points equals one inch held at arm's length. Commonly font sizes of 9pt or 10pt are used in books, with about 72pt per inch on the printed page.

Old-style HTML might show things like `<font size=3>`. The old style is deprecated. Do not use it.

9.6 font-weight: (normal, bold)

font-weight: can be normal, bold, or an x-hundred number like 100, 200, ..., 900. The smaller the number, the lighter the weight.

Exam Question 166 (p.346)**:** What CSS attribute: sets the thickness of your lettering?
Required Answer: font-weight:

If you want to, you can redefine `<b>` to have no boldness by using this directive.

```
b { font-weight: normal; }
```

9.7 font-style: (normal, italic)

font-style: can be normal, italic, or oblique (which means the same as italic).

Exam Question 167 (p.346)**:** What CSS attribute: sets the slant of your lettering?
Required Answer: font-style:

You could even redefine `<i>` to have no slant by using this directive.

```
i { font-style: normal; }
```

9.8 text-align: (left, right, center, justify)

Text can be centered, or it can be justified to the left, to the right, or to both sides of the column. The **text-align:** attribute can have a value of left, right, center, or justify.

Old-style HTML might show things like `<center>`. The old style is deprecated. Do not use it.

```
text-align: center; /* ragged left and right */
text-align: left; /* straight left, ragged right */
text-align: right; /* ragged left, straight right */
text-align: justify; /* straight left and right */
```

Exam Question 168 (p.346)**:** List the four "text-align:" options.
Required Answer: left, right, center, justify

To center the text of all paragraphs, we can do this.

```
p { text-align: center; }
```

9.9 text-shadow: x y blur color

This is kind of a fun one. The **text-shadow:** property can be used to display a shadow behind your lettering. This is almost like creating new fonts. This can look very cool. Or it can look awful. Use your artistic judgment.

Exam Question 169 (p.346)**:** What CSS attribute: creates a shadow behind your lettering?
Required Answer: text-shadow:

It has four parameters.

```
text-shadow: (x) (y) (blur) (color);
```

Here is an example.

```
h1 { text-shadow: 1px 2px 3px red; }
```

The first parameter, in this case 1px, is the (horizontal) x-offset for the shadow. Positive numbers move the shadow to the right. Negative numbers move the shadow to the left.

The second parameter, in this case 2px, is the (vertical) y-offset for the shadow. Positive numbers move the shadow down. Negative numbers move the shadow up.

The third parameter, in this case 3px, is the amount of blur. Small numbers result in a crisp shadow. Large numbers result in a blurry shadow.

The fourth parameter, in this case red, is the color of the shadow. You can use any method of specifying the color, including methods like **rgba** that allow **transparency**.

Neon Lights: You can create a neon lights effect by setting the x and y offsets to zero. Use a sans-serif font, maybe with rounded ends.

Multi Shadow: You can have more than one shadow. Separate the specifications with commas.

```
text-shadow: 1px 2px 3px red, 2px 4px 3px blue;
```

The second shadow will appear behind the first. The third shadow will appear behind the second. Etc.

Textless Shadow:

You can also change the color of the text itself, perhaps to match the back-

ground. Then the only thing visible is the shadow.

You can also do this with the **opacity** attribute.

Google search "css text effects" for more ideas.

9.10 CSS Comments

CSS is a whole different language than HTML, and it has its own way of identifying comments. Comments in the style sheet are surrounded by `/*` at the front, and `*/` at the end.

Exam Question 170 (p.346)**:** What marks the start of a comment in CSS?
Required Answer: `/*`

Exam Question 171 (p.346)**:** What marks the end of a comment in CSS?
Required Answer: `*/`

Exam Question 172 (p.346)**:** In CSS can comments be nested?
Required Answer: no

9.11 Review of Previous Chapters

The `<b>` tag starts a section of **bold** text, and the `</b>` tag identifies the end of that section.

```
<b>Here is some bold text.</b>
```

Bold text uses a font that is thicker and heavier and darker than normal. It is said to have more weight than normal text.

Strong text is (by default) just bold text. Bold is presentational markup. Strong is semantic markup.

Exam Question 173 (p.346)**:** In HTML the tag usually does the same as what other tag?
Required Answer: `<b>`

```
<i>Here is some italic text.</i>
```

Italic text uses a font that is slanted to make it look special.

```
<em>This is emphasized text.</em>
```

Emphasized text is similar to italicized text, but **em** tries to indicate your

purpose, and **i** indicates how you want it done. For practical purposes they are pretty much the same.

Exam Question 174 (p.346)**:** In HTML the tag usually does the same as what other tag?

Required Answer: `<i>`

Chapter 10

Font Families

Contents

Most of this chapter is about fonts, but we will start out with a brief discussion of classes because it provides a short-cut to help us use fonts more easily.

10.1 Style Sheet vs Inline Styling

We can put lots of styling information right inside our markup, but the smart money says we should keep them apart. The best-practices approach is to put virtually all of our CSS into a style sheet. From here on out, we will follow that approach in this book.

In chapter 9 (page 90) we learned about targets and attributes, and that this is the prototype for style sheet entries.

```
target { attribute: values; attribute: values; ... }
```

In this section we will learn about class= which give us another way to apply special styling to selected paragraphs.

10.1.1 class=

Within any HTML tag, we can say that it belongs to one or more classes. For example:

```
<img class=floatLeft src=... alt=...>
```

If you want to apply more than one class, separate them by spaces and put the list in quotes:

```
<img class="floatLeft framed" src=... alt=...>
```

Within the style sheet, you specify the attributes that belong to that class. Those attributes will be applied to the marked-up content as though they had been specified inline.

```
.floatLeft { float: left; margin-right: 5px; }
.floatRight { float: right; margin-left: 5px; }
.framed { border: thick black double; }
```

The immediate advantage of this approach is that you can provide the attributes one time and then use them many times, without retyping them.

The latent advantage of this approach is that you can change the attributes of any class very easily and have it apply to all the content that belongs to that class. You do not need to look for each item in your HTML and fix it separately.

Exam Question 175 (p.346)**:** In CSS what is the one-character symbol for class?
Required Answer: .

The one-character symbol for class is . (dot).

10.1.2 Class Names

If you get to choose your class names, I recommend you follow the simple rule: start with a letter and continue with letters or digits or dashes. Underscore counts as a letter. Case matters. (Big A and little a are recognized as being different from each other.)

Exam Question 176 (p.346)**:** What is the simple rule for creating a legal class name?
Acceptable Answer: letter first, then letters or digits or dashes

This is the same rule that many computer languages use for creating variable names (except for dashes).

The complex rule for class names is that you can use almost any character, but you have to properly escape some of them, which can make life difficult.

Having established a nice way to specify fonts through the use of classes, we will now turn to the fonts themselves.

10.2 What Is A Font?

A font is a typeface. The origin of the word font is related to fondue, like dipping bread into melted cheese. The letters used in printing were originally made from melted lead. (The distance between lines of text is still called leading, pronounced led-ing.)

Fonts can affect the emotional response that users have to your webpage. They can be heart-warming, amusing, or really annoying.

The use of interesting fonts can really spice things up. They are especially nice for headings. It can be a problem if the body text gets too fancy because

it makes text harder to read.

http://en.wikipedia.org/wiki/Font has a wonderful article giving historic background and more.

For our purposes, a font is a set of letter shapes used to create a message.

10.2.1 Word Processing Font Choices

When using a word processor like Microsoft Word, a variety of fonts, sometimes a huge variety, is available. And you can download and install even more. You can create your document and then render it as a printed document with full confidence that your letters will look the way you want.

But one seldom spoken nightmare of word processing is document sharing. A document may look beautiful on your own computer, but when you send it to a friend and they open it, if they do not have the same fonts installed, the document may look dramatically different, especially if you are using a carefully-chosen but obscure font.

That is because the fonts are not part of the document. Fonts are sold separately. The document says what fonts it wants, and the computer is expected to provide those fonts. Sadly this problem can also apply to PDF files.

The good news is that modern computers generally come with a large enough collection of fonts that the document-sharing problem is often minimal.

10.2.2 Web Page Font Choices

Word processing programs, such as Microsoft Word, provide a selection of fonts that can be used in creating documents. Web designers would love to have similar freedom in selecting fonts for their websites.

The document-sharing problem is much larger on the web. Your users may be running Microsoft Windows, Mac OS X, Linux Ubuntu, Apple IOS, Android, or who knows what else. They may be using Internet Explorer, Firefox, Chrome, Safari, Opera, or who knows what else.

Getting your page to display the way you want, with the fonts you want, depends on all your users having those same fonts, or something close enough. It is not as easy as we might like.

10.2.3 Raster vs Vector

Raster Fonts: Ultimately, fonts are rendered as pixels on the user's screen or on a printed page. Raster fonts were used originally and are defined in terms of these pixels. Basically, all screens are raster. Even the retinas of our eyes are raster, with their individual cones and rods. If you magnify a raster font, the curved edges eventually become pixelated or jagged.

Exam Question 177 (p.346)**:** What kind of font represents shapes as pixels?
Acceptable Answer: raster

Vector Fonts: To avoid jagged edges, it is desirable to define characters as mathematical curves instead of a raster of pixels. Vector fonts are defined in terms of the curves that describe each character, and those same curves can generate raster images in a variety of sizes. Vector fonts are widely used now. If you magnify a vector font, the edges always remain smooth.

Exam Question 178 (p.346)**:** What kind of font represents shapes as strokes?
Acceptable Answer: vector

Vector fonts are also called **scalable fonts** because they can be resized (scaled) to different sizes without getting rough edges.

10.2.4 Copyright and Knock-Offs

Copyright does not cover raster character shapes. If it did, you could copyright a font, and then print anything you want using that font, and nobody could photocopy it because the individual characters were copyrighted. Courts in the US have rejected that idea.

http://en.wikipedia.org/wiki/Intellectual_property_protection_of_typefaces has more.

Instead, copyright and licensing do cover mathematical instructions for drawing the shape of individual characters. This also includes kerning tables that control the distance between characters. If you want the original font in all its original glory, you need to get it from the original font designer or foundry.

Exam Question 179 (p.347)**:** What does kerning control?
Acceptable Answer: distance between adjacent letters

Kerning tables tell the browser that it can put the letters AV closer together than the letters VV. Each pair of letters is examined by the designer and their ideal distance is decided. This can be especially important for cursive fonts.

But what this also means is a font designer can release a font and someone else, even another professional font designer, can copy its letter shapes by coming up with a new set of mathematical instructions that looks about the same. After all, the original digital fonts were copied from printing-press fonts. Thus, knock-offs of fonts are commonly available.

Due to trademark restrictions, knock-offs cannot use the same name as the original font. So alternate names are given. The font you want to use may be available under several different names, depending on which platform the user is running.

10.2.5 Good News, Bad News

Good News: For the commonly-used fonts, something identical or similar is probably installed on your user's platform.

Bad News: Even commonly-used fonts may be installed under a different name than they are on your own platform.

Font Stacks: The current solution is to let you, the webpage designer, specify a list of acceptable fonts. The browser will look at the list and use the first font it can. This lets you list the font name as found on your platform, plus the font names of similar fonts found on other platforms. See section 10.4 (page 106) for more information.

10.3 Five Generic Font Families

CSS recognizes five generic (broad) families of fonts. They are **serif**, **sans-serif**, **monospace**, **cursive**, and **fantasy**. Within each family, there are thousands of individual fonts that have been created by font designers.

http://en.wikipedia.org/wiki/Font_family_(HTML) has more on font families.

Exam Question 180 (p.347)**:** List the five generic font families.
Required Answer: serif, sans-serif, cursive, fantasy, monospace

10.3.1 Serif

Serifs are those spikey things that poke out of the corners of letters. Originally they were used as reinforcement to keep the letters from breaking in the printing press, but they continue to be used. People are used to seeing it, and it is easy to read. Serif fonts are commonly used in body text.

Exam Question 181 (p.347)**:** What generic font family includes Times?
Required Answer: serif

The most common serif typeface is Times Roman.

Other common examples of serif typefaces include Baskerville, Bookman, Garamond, Georgia, and Palatino. Also, any font with the word serif in its name, but not the word sans, is probably a serif font.

http://en.wikipedia.org/wiki/Samples_of_serif_typefaces shows some examples.

10.3.2 Sans-serif

Sans-serif means without serifs. Sans is from the Latin word (sine) that means "without." Sans-serif fonts are commonly used in headings.

Popular wisdom has long held that serif fonts are best for printed work and sans-serif fonts are best for online work. That is because online work has long been rendered using a low number of pixels per inch, and serifs often look bad in that setting. More recently, displays have reached "retina quality" and the printed experience can be achieved online just fine. You can ignore those earlier warnings. Serif is wonderful for online work these days.

Exam Question 182 (p.347)**:** What generic font family includes Helvetica?
Required Answer: sans-serif

The most common sans-serif typeface is Helvetica.

Other common examples of sans-serif typefaces include Arial, Geneva, Impact, Lucida Sans, Tahoma, and Verdana. Also, any font with the word sans in its name is probably a sans-serif font.

http://en.wikipedia.org/wiki/Samples_of_sans_serif_typefaces shows some examples.

10.3.3 Monospace

Monospace means one spacing (size), and hearkens back to the day of typewriters. Each letter has the same width. Wide letters like m and w take up the same amount of space as the narrow letters like i and j. Monospace fonts are commonly used to represent data entry and computer code.

Exam Question 183 (p.347)**:** What generic font family has letters all the same width?
Required Answer: monospace

Monospace: `iiiii jjjjj mmmmm wwwww`

Proportional: iiiii jjjjj mmmmm wwwww

Exam Question 184 (p.347)**:** What generic font family includes Courier?
Required Answer: monospace

The most common monospace typeface is Courier.

Other common examples of monospace include Lucida Console and Monaco. Also, any font with the word mono in its name is probably a monospace font.

Because the whole point of monospace is to line things up with the left margin, it does not make any sense to use it with text-align right, center, or justify.

Exam Question 185 (p.347)**:** Which generic font should almost always be text aligned left?
Required Answer: monospace

10.3.4 Cursive

Cursive fonts try to emulate human handwriting, as with calligraphy. Often they connect adjacent letters in a flowing manner. They are used in headings because they are beautiful, but they are avoided in body text and at small sizes because they are more difficult to read. It can work well when there are only a few words to be shown.

Exam Question 186 (p.347)**:** What generic font family emulates human handwriting?

Required Answer: cursive

To **really** customize your webpages, you can create your own cursive font based on your own personal handwriting. Okay, it is a bit more work than most people would attempt, but wouldn't it be cool?

Common examples of cursive include: Apple Chancery, Zapf Chancery, Comic Sans (even though it has the word Sans in it). Also, any font with the word cursive or script or manuscript in its name is probably a cursive font.

10.3.5 Fantasy

Fantasy fonts make up a "none of the above" category. They may be fun in headings but are generally not suitable for body text. The **fantasy** font tends to be highly decorative. It might be good for monograms or fancy initial capital letters, like with the **:first-letter** pseudo-class.

http://cssfontstack.com/ has a list of fonts.

10.4 The Font Stack

The **font-family:** CSS attribute lets you specify the font in which your text will be presented. There are many thousands of fonts that have been created.

The bad news is that (in 2012) only a small fraction of those thousands of fonts that exist are installed on any one personal computer. If the designer (you) wants the user to see the page rendered using a particular font, and the user does not have that font installed, then the browser will need to choose another font and it may not be what the designer wanted.

The current solution to this problem is the "CSS font stack." For each font specification, the webpage designer gives a list (or stack) of fonts that would be acceptable. The browser will use the first font if it is available to the user. If that font is not available, it checks for the next font (the **fallback font**), and the next. Hopefully at least one font is available.

Specify the best font first, and then a list of fallback choices, and finally a generic font last.

Font Family Examples: Here are a few examples.

```
font-family: "Times New Roman", times, serif;
font-family: helvetica, arial, sans-serif;
font-family: courier, monospace;
font-family: papyrus, fantasy;
```

Each font family specification starts with `font-family:` and is followed by a list of one or more fonts.

Best First: The best font choice should be listed first, and then list the fall-back fonts to be used in case the best font is not available on your user's platform.

Generic Last: The last choice you list should always be one of the five generic font families: serif, sans-serif, monospace, cursive, or fantasy. These are guaranteed to be available on every platform.

Exam Question 187 (p.347)**:** Tell what is wrong with this font spec: font-family: Times Roman, serif;
Acceptable Answer: quotes needed on Times Roman

Exam Question 188 (p.347)**:** Tell what is wrong with this font spec: font-family: ARiaL, sans-serif;
Acceptable Answer: nothing

Exam Question 189 (p.347)**:** Tell what is wrong with this font spec: font-family: arial, sans-serif;
Acceptable Answer: nothing

Exam Question 190 (p.347)**:** Tell what is wrong with this font spec: font-family: sans-serif, arial;
Acceptable Answer: generic must be last

Exam Question 191 (p.347)**:** Tell what is wrong with this font spec: font-family: arial, sans-serif, serif;
Acceptable Answer: too many generics

Cross-Platform Awareness: If you are trying to get a particular look, you should try to include an acceptable font for each major platform your users might be using: Microsoft Windows, Mac OS X, Mac iOS (iPhone or tablet), Android (phone or tablet), Linux.

Web Fonts: You may be able to select a downloadable font that works across many platforms. But do not assume every platform will accept it. See section 10.6 (page 110).

Google search "commonly available fonts" to see what your users probably have available.

Commonly available fonts include Arial, Verdana, Georgia, and Times New Roman, but even these tend to vary by browser and platform.

Here are some resources you can look at to find the commonly available fonts for various platforms.

`http://www.ampsoft.net/webdesign-l/WindowsMacFonts.html` gives a table of fonts showing their names under Windows, under Mac, and under their generic family.

`http://www.w3schools.com/cssref/css_websafe_fonts.asp` gives some examples of font stacks for serif, sans-serif, and monospace characters.

`http://webdesign.about.com/od/fonts/a/aa080204.htm` has advice about choosing fonts.

Exam Question 192 (p.347)**:** In font names, does capitalization matter?
Required Answer: no

Spelling: Font names must be spelled exactly correctly. However, capitalization does not matter in font names. (It frequently matters in other places.)

Spaces: If a font name has any spaces in it, the whole name must be given in quotes.

Exam Question 193 (p.347)**:** In font family specs, what font should always be included?
Required Answer: generic

Exam Question 194 (p.347)**:** In font family specs, what font should always be listed last?
Required Answer: generic

Generic: The last font in each list should be one of the five generic fonts: serif, sans-serif, cursive, fantasy, or monospace.

Like all CSS attributes, the font family specification ends with **;** (a semicolon).

Avoid The Font Tag

The `<font>` HTML tag was introduced in 1995 by Netscape, long before CSS was invented.

Old-style HTML would use `<font face="times,serif">` (for example) to specify a font stack.

The old style is **deprecated**, having been replaced by CSS Font Properties. Do not use it. Still, you will probably run into it from time to time, and older tutorials may rely on using it.

10.5 Defending Against Missing Fonts

How do you handle missing fonts? The font stack is your first line of defense. But what if you really really want a specific font for use in a top-of-page banner of something?

One solution is to render your special text in advance and save it as an image. This is guaranteed to work on any browser that displays images. This is commonly done for page headings and logos.

Another solution is to use a web font which will be downloaded to your user's computer if they don't already have it. This also works on every modern browser. See section 10.6 (page 110).

Font by Image

If you seriously want a particular font for the heading of your page, one solution is to use the font locally to create an image file that can be used inside your page.

You can Google search "free fonts" to find thousands of very nice fonts.

You might choose to use a .png file with a transparent background so that your image can overlay any background you might choose in the future.

There are several problems with the "image" solution.

(a) Images, especially headings, can be large. If you are using the same font but different wording for several headings, and each heading has its own image, you have to download all of them as needed.

(b) Images do not scale well. Raster graphics (gif, jpg, png, etc.,) become pixelated when they are enlarged.

(c) Image creation must be integrated with content development. But image is style, and ideally should be done separately.

However, despite these small problems, creating images is very popular, especially for things like image-based submit buttons.

10.6 Web Fonts

A web font can be automatically downloaded to your user's computer if they don't already have it. This works on every modern browser.

Google provides a large but limited collection of fonts you can use freely.

Fonts.com provides a gigantic collection of fonts you can pay to use.

10.6.1 Google Web Fonts (Free)

http://www.google.com/fonts is a source for webpage fonts. (I like using their "poster" tab to view their fonts.)

They have hundreds of font families available (613 in Dec 2012, 617 in Feb 2013, 632 in Feb 2014, 671 in Feb 2015). The fonts they provide are open source and free of charge.

In your HTML, you "source" the font like this, but replace the XXX with the name of the font.

```
<link rel=stylesheet type=text/css
 href='http://fonts.googleapis.com/css?family=XXX'>
```

This line should come very early in your `<head>` section. By putting it early (or even first), it gives the browser more time to fetch the font in case the user does not already have it.

Once the font has been downloaded and cached by the browser, it does not have to download again, making future page loads faster.

In your CSS, you specify the font stack like this, but replace the XXX with the name of the font.

```
font-family: 'XXX', sans-serif;
```

Here, XXX would be the requested font, and sans-serif would be the fallback font, which is one of the five fonts guaranteed to be available.

Here are a few fonts from Google that caught my eye: UnifrakturMaguntia, UnifrakturCook, New Rocker, Sancreek, Seaweed Script, Miltonian, Monoton, Smokum, Nosifer.

10.6.2 Web Fonts: fonts.com (Paid)

If there is a particular font that you just have your heart set on, it is likely to be available at fonts.com.

http://www.fonts.com/web-fonts has over 20,000 web fonts available for your use. Their free plan allows you to use any of 3000 fonts for up to 25,000 pageviews per month. They also have paid plans that offer more fonts and more pageviews.

10.7 Special Glyphs

If you are quote-marking the value of an attribute, and it needs to include the same kind of quote mark in the middle, unless you do something special that interior quote mark will be treated as an ending quote mark, and then things will get confusing.

In some computer languages, you can escape the normal meaning of a special character by putting \ right before it. This is not supported by the HTML standard. Instead, you can use a "character reference" which is an alternate way to specify a character.

`<... attribute="a"c" ...>` (bad)

`<... attribute="a&quot;c" ...>` (good)

Many unusual glyphs (characters) can be inserted into your webpage by using character references.

There are character references for things like copyright and registered trademark and special symbols. There are thousands of these character references supported in HTML.

`&` – Character references always start with an ampersand.

; – They should always end with a semi-colon. The ending semi-colon is optional if things are not ambiguous, but future character references could make a once-unambiguous thing ambiguous. Play it safe.

Here are a few handy glyphs.

&quot;	"
&apos;	’
&amp;	&
&copy;	©
&reg;	®
&trade;	TM

&gt;	>
&lt;	<
&geq;	≥
&nequiv;	≠
&leq;	≤

&male;	♂
&female;	♀
&spades;	♠
&diamonds;	♢
&hearts;	♡
&clubs;	♣

http://shapecatcher.com/ is a wonderful tool that lets you draw a picture of the glyph you are looking for. It will search its database and give you a list of glyphs that it thinks match most closely. I use it often.

There are hundreds (maybe thousands) more of these character references supported in HTML. Google search for “html special characters” to see more of them.

Exam Question 195 (p.347)**:** What HTML character entity reference is for a space without letting the line split?
Required Answer:

Exam Question 196 (p.347)**:** What HTML character entity reference is for the ampersand (and) symbol?
Required Answer: &

Exam Question 197 (p.347)**:** What HTML character entity reference is for the less-than symbol?
Required Answer: <

Exam Question 198 (p.348)**:** What HTML character entity reference is for the greater-than symbol?
Required Answer: >

Exam Question 199 (p.348)**:** What HTML character entity reference is for a double-quote?
Required Answer: "

Exam Question 200 (p.348)**:** What HTML character entity reference is for a single-quote?
Required Answer: '

Exam Question 201 (p.348)**:** What HTML character entity reference is

for the (C) copyright symbol?
Required Answer: ©

Exam Question 202 (p.348)**:** What HTML character entity reference is for the (R) registered-trademark symbol?
Required Answer: ®

Exam Question 203 (p.348)**:** What HTML character entity reference is for the TM nonregistered-trademark symbol?
Required Answer: ™

10.8 Font and Text Attributes

We have talked a lot about fonts themselves. There are some other properties you can use. For more information than is presented here, you can do a Google search. It will turn up lots of examples.

text-shadow: was introduced in section 9.9 (page 95) and provides a way to put a shadow behind your wording.

font-variant: can be normal or small-caps. You can cause your lower-case lettering to be rendered as upper-case but smaller than full capital letters.

clip: can be used to hide part of the text. This is useful when the text might be very long and you only want to display the first few inches of it. See also overflow.

letter-spacing: lets you increase or decrease the distance between letters. See also word-spacing.

line-height: controls the distance between baselines on successive rows of text. A line height between 1.0 and 1.2 is considered normal. Sometimes called **leading** (pronounced led-ing).

overflow: controls what happens when content is too big to fit in the space you have allocated for it. See also clip.

text-decoration: can be none, underline, overline, line-through, or blink. (Please avoid blink. Users complain about it. Some browsers have actually disabled it.)

text-indent: affects the first line of text in a paragraph.

text-transform: can be none, capitalize, uppercase, or lowercase.

word-spacing: You can add or subtract from the amount of space normally used between words. See also letter-spacing.

Chapter 11

Hover and Pseudo Classes

Contents

In this chapter we will talk about pseudo classes, pseudo elements, and animated transitions.

We have used explicit HTML markup to apply styling to the elements of our webpages. We can control the background color, font, padding, border, margin, and many other aspects of the style. We have used classes to help us target our stylings.

But think about links. They do something unusual. When we first view a link, it is colored in such a way that we can tell we have never been there. (Or, at least, it is not in our history.) But after we visit it, the link changes color. And yet the HTML of the webpage has not changed.

Pseudo classes and pseudo elements allow us to style things even without special markup. In this chapter we explore some of them. The list of pseudo classes seems to be constantly growing, and they let you do some really fun stuff, so there are lots of tutorials out there.

Exam Question 204 (p.348)**:** What does pseudo mean?
Acceptable Answer: fake

Pseudo means fake or false. It is fake in the sense that there is no explicit, static markup that it targets. Instead it is based on things that are implicit or dynamic.

11.1 Link Styling

We can style links differently depending on whether they have been visited or not. Let's make unvisited links bold and green, and visited links italic and red.

```
:link { color: green; font-weight: bold; }
:visited { color: red; font-style: italic; }
```

Put this into the style sheet of a webpage, and then create a few links and see how their styling changes as the links are visited.

Exam Question 205 (p.348)**:** What is the prefix for pseudo-class?
Required Answer: :

The prefix symbol for pseudo-class is : (colon).

Exam Question 206 (p.348)**:** What is the pseudo-class for a visited link?
Required Answer: :visited

:visited - change the style of an a-link element when we can tell it has been visited.

Exam Question 207 (p.348)**:** What is the pseudo-class for an unvisited link?
Required Answer: :link

:link - change the style of an a-link element when we can tell it has not been visited.

There is also a pseudo-class for a link that is active, which really only takes effect during the moment that the link is being clicked (from mouse down

to mouse up). It can create a nice and immediate sense of feedback to the user, letting them know that you know they have clicked on a link.

```
:active { background-color: yellow; }
```

:active - change the style of an a-link element when we are actively clicking on it.

11.2 Text Styling

We can style bits of unmarked text depending on whether it is the first letter or part of the first line of the paragraph.

```
p:first-letter { font-size: 200%; }
p:first-line { font-weight: bold; }
```

Technically, these are called pseudo elements (as distinguished from pseudo classes). I don't see the big distinction, but some people do. And technically they are introduced by two colons instead of one, like this.

```
p::first-letter { font-size: 200%; }
p::first-line { font-weight: bold; }
```

Exam Question 208 (p.348)**:** What is the pseudo-element for the first character of something?
Required Answer: ::first-letter

Exam Question 209 (p.348)**:** What is the pseudo-element for the first row of text of something?
Required Answer: ::first-line

As should be obvious, this styling will cause the first line of each paragraph to be bold, even without any `<b>` tags. And the first letter of the paragraph will be twice as big as normal, which is a fun effect I have seen used in books.

Exam Question 210 (p.348)**:** What is the prefix for pseudo-element?
Required Answer: ::

The prefix symbol for pseudo-element is `::` (double colon).

Most browsers seem to be okay with these being specified either way (one colon or two colons). It was originally one colon, so browsers will probably continue to accept that. CSS validators may be more picky.

::first-line - style the first line (not sentence) of a paragraph differently than the other lines.

::first-letter - style the first letter of a paragraph differently than the other letters. Make it twice as big, for example, or use a different font.

Drop Caps: One fun thing to do with the first letter is to make it a lot bigger and then to drop it down so that it takes up two lines. This is called a "drop capital." We can do this by adding a float left to the letter, which will allow the rest of the paragraph to slide up a bit.

```
p::first-letter { font-size: 200%; float: left; }
```

Of course, once you are styling the size and float, you might as well think about picking some fancy font to go with it.

11.3 First Child

Sometimes we want to style the first or last paragraph differently than the others. Perhaps we want different padding, or we only want the first letter of the first paragraph to be special, not the first character of every paragraph. We can target specific paragraphs by number.

```
p:first-child { ... } /* first paragraph */
p:nth-child(2) { ... } /* second paragraph */
p:nth-child(2n) { ... } /* even paragraphs */
p:nth-child(2n+1) { ... } /* odd paragraphs */
p:nth-child(3n) { ... } /* paragraph 3, 6, 9, ... */
p:last-child { ... } /* last paragraph */
p:nth-last-child(2) { ... } /* paragraph before last */
p:only-child { ... }
```

First, last, and nth are all counted from the parent element. If the paragraphs are part of a div, then it would be the parent.

The even and odd choices, (2n) and (2n+1), are helpful in tables when we might want alternating rows to be distinct, with a slighly different background color for example.

:nth-child - Use `:nth-child(2n)` to style alternating list items or table rows to have different background colors.

The "child" pseudo classes do their counting based on the immediate parent of the item, no matter what type the other children are. Sometimes we want to count only items of a certain type. Then we can use the "of type" pseudo classes. Here for illustration we will use the bold tag within a paragraph. (Not that you would ever want to do this, but just for illustration.)

```
b:first-of-type { ... } /* first bold */
b:nth-of-type(2) { ... } /* second */
b:nth-of-type(2n) { ... } /* even bolds */
b:nth-of-type(2n+1) { ... } /* odd bolds */
b:last-of-type { ... } /* last bold */
b:nth-last-of-type(2) { ... } /* bold before last */
b:only-of-type { ... }
```

If more than one of these pseudo-classes matches, the latest one specified will take effect. Thus, if you specify only-child and then specify first-child, first-child will win. Normally you should specify the most general things first and the most specific things last.

Exam Question 211 (p.348)**:** If two CSS specs apply to the same item, which takes precedence, the first specified or the last specified?
Required Answer: last

11.4 Hover and Focus

The :hover pseudo-class targets objects that have the mouse hovering over them. The mouse often indicates the portion of the webpage that the user is most interested in.

Note that not all devices have mice. This pseudo-class works great for normal computers but not so great for iPads or smart phones.

Exam Question 212 (p.348)**:** What pseudo-class targets an item when you move the mouse across it?

Required Answer: :hover

The following css will cause a red outline to appear around any paragraph as you pass your mouse over it.

```
p:hover { outline: red 2px dotted; }
```

:hover - change the style of an element while the mouse rolls across it.

Data entry happens in input fields which we will discuss in a future chapter. We may want to help the user keep track of where their keystrokes will be recorded as they tab from field to field. The :focus pseudo-class lets us do that.

Exam Question 213 (p.348)**:** What pseudo-class targets the input field where typed text would be entered?
Required Answer: :focus

The following css will cause a red outline to appear around each input field as you tab to it.

```
:focus { outline: red 2px dotted; }
```

:focus - change the style of the element that would receive any characters that are typed.

11.5 Transitions and Animations

As you hover or do other things that cause changes on your webpage, the changes normally take effect immediately.

It can be fun and helpful to have a delayed reaction. For example, when you use title= the title appears after a brief delay. We do not want to interrupt the user as they mouse around. We only want to do it when we think they are interested.

Transitions give us a way to delay the start of an action, and also to let the action progress slowly and smoothly.

Exam Question 214 (p.349)**:** What attribute lets you make styling changes smoothly instead of instantly?
Required Answer: transition:

Transitions and animations work in the current versions of all major browsers. Internet Explorer before version 10 does not work.

Some changes cannot be transitioned smoothly. They are immediate. You can find lists but it is perfectly reasonable to just experiment.

Numeric Attributes: The general rule is this: If the attribute is represented by a number, then it can be transitioned gradually. If not, then the transition is always immediate. So, colors are numeric. Opacity is numeric. Font size is numeric. Font shadowing is numeric. Position and size are numeric. But pictures are not numeric. Font names are not numeric.

Let's illustrate this with a simple example.

```
<style>
* { transition: all 2s; }
p { background-color: white; }
p:hover { background-color: red; }
</style>
<body>
<p>This text has a changing background color.
</body>
```

Make that a webpage. Load it. Then mouse over the text.

11.5.1 Transitions

Among things that make webpages dynamic, the **:hover** attribute is probably the most simple and interesting example. It lets you change the style of an element when the mouse rolls across it.

transition: lets you control the timing with which things change. Timing includes the delay before the transition starts, and the speed curve by which the transition makes progress.

If all transitions should happen with the same timing, you can specify the keyword "all" as the property, like this:

```
transition: all 2s;
```

11.5.2 Animation

Animation is like transition, but it does not have a trigger. It just starts running when the webpage loads. You can have it repeat the same animation over and over, either forever or up to a limit.

11.5.3 Transition Attributes

Transition and animation have four attributes: the property to be animated, the duration of the transition, the shape of the timing curve, and the delay before transition starts. You express it like this as part of your CSS.

```
transition: property duration timing delay;
```

Property: You cannot omit this, but you can specify the keyword "all" to refer to all properties. The property is the thing that will be changed gradually. Common examples include width, height, background-color, color, opacity, font-size, and padding.

Duration: You should not omit this. If you do omit duration, it defaults to zero, and the transition is instantaneous, which kind of defeats the purpose of having a transition. It is expressed as a number followed by the letter s (meaning seconds). For example, 2s means two seconds.

Delay: You can safely omit this. If you do, it defaults to zero (seconds). It will wait this long before starting the transition. It is expressed as a number followed by the letter s (meaning seconds). For example, 2s means two seconds.

Timing: You can safely omit this. If you do, it defaults to "ease" which means it starts out slow, increases in speed, and then ends up slow. The list of pre-defined timing functions includes: ease, ease-out, linear, step-end, and steps(n,end). You can also specify your curve exactly using Bézier curve parameters.

11.5.4 Animating Several Properties Differently

You can manage several properties at the same time. The syntax is to make a comma-separated list of attributes and their timings. For example:

```
transition: width 2s, height 2s, opacity 4s;
```

11.6 Swapping Out an Image

It is fun to be able to replace one image with another as the user moves their mouse across the image. There are many ways to do this. Ideally the images should be exactly the same size.

Let's pretend we have two images, 1.jpg and 2.jpg, that are the same size. We want to see 1.jpg normally, but when the mouse is over 1.jpg, we want to see 2.jpg instead. We will demonstrate four ways this could be done.

CSS-controlled Content Replacement (Instant)

Here is a solution that uses CSS to change the content of the image. The first line goes in the style section. The second line goes in the body of the webpage. We identify the image with a unique ID, in this case, id=x.

```
#x:hover { content: url(2.jpg); }

<img id=x src=1.jpg width=500 alt="alternating images">
```

When you hover over the image, the content is immediately changed. What's behind the door? Hover and see.

JavaScript-controlled Content Replacement (Instant)

Here is a solution that uses JavaScript to change the source URL. The entirety goes in the body of the webpage. On mouseover, the src is replaced with the URL of the second image. On mouseout, the src is replaced with the URL of the original image.

```
<img src=1.jpg width=500
onmouseover='this.src="2.jpg"'
onmouseout='this.src="1.jpg"'
alt="alternating images">
```

The JavaScript onmouseover and onmouseout commands let us replace part of the HTML markup, in this case the src, with a new value.

Imagine bubble wrap that pops as you mouse over it, permanently replacing the image of an unpopped bubble with that of a popped bubble.

Scrolling a Combined Image (Animated)

Here is a solution that uses CSS and transitions to scroll the image from the visible part to the invisible part. With this approach, you create one larger image that has each of the smaller images embedded in it. The first three lines are CSS. The last line is HTML.

One big advantage of this approach is that there is only one image to be loaded. When two separate images are involved, there can be a delay while the second image is retrieved from the server.

```
div#x { background-image: url(1.jpg);
 transition: background-position 2s; }
div#x:hover { background-position: -400px 0; }

<div id=x></div>
```

The scrolling image approach is very powerful but also takes more work in preparation. It can make a nice slide show.

Cross Fading Images (Animated)

Here is a solution that uses opacity to fade one image out and fade the other image in. We wrap the first img inside a div, and set the background image of the div to be the second image. Then when we hover over the image, we transition the opacity of the first image slowly to zero, revealing the second image.

Because both images are loaded initially, there is no delay to load the second image when the user causes the cross fading to begin.

```
img { transition: opacity 2s; }
img:hover { opacity: 0; }

<div style=background-image:url(2.jpg)>
<img src=1.jpg alt=""></div>
```

The `alt=""` is needed because img tags are required to have an alt attribute, whether it is blank or not.

11.7 For Further Study

Google search “pseudo classes” to get lists of options that are available.

Google search "css transition" for lots of tutorials.

http://dev.w3.org/csswg/css-transitions/ has authoritative information about transitions.

https://developer.mozilla.org/en-US/docs/CSS/CSS_animated_properties gives a list of the properties that can be animated.

Chapter 12

Box Model

Contents

We often desire a bit of space between the text and pictures that make up our webpage. And we would like to control that space.

Beginning webpage authors often create space by using `<p>` or `<br>` repeatedly, like this:

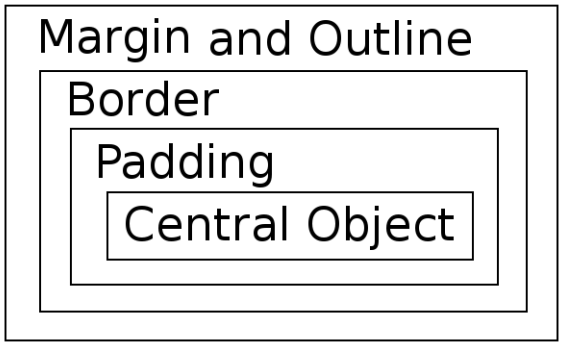
```
<br><br><br><br><br><br><br><br><br>
```

That does create a certain amount of vertical space on your webpage. But it is fragile and identifies you as unskilled.

CSS provides a simple but powerful system for telling the browser just how much space to leave, and how to fill it. Every web designer must understand the box model.

12.1 The Basic Box

The CSS system is called the **box model**. It provides for an inner box that is tightly fitted to the central object, the core content you are displaying. Around that are padding, border, and margin.



Exam Question 215 (p.349)**:** List the three major components of the box model.
Required Answer: padding, border, margin

The five components are content, padding, border, margin, and outline, but when people talk about the box model, normally they are just talking about padding, border, and margin.

12.1.1 Content

The inner-most part of the box is the content itself. It could be a picture or a paragraph or a table or something more complex. Its dimensions may be limited, or it may fill all the available space.

The other elements, padding, border, margin, and outline, are mostly empty. Their sizes can be measured in actual pixels or as a percentage of the size of the content, or in comparison to the size of the characters in the content.

We have already discovered that we can style the content by controlling its color, background, and font.

You can set the width and/or height of the content by specifying one or more width: or height: attributes. They are as follows:

width: You can specify the width of the content. If your content would naturally be larger or smaller than this, the browser will scale or clip or pad it to match this value.

max-width: You can specify the maximum width of the content. If your content would naturally be larger than this, the browser will scale or clip it down to this maximum value.

min-width: You can specify the minimum width of the content. If your content would naturally be smaller than this, the browser will scale or pad it up to this minimum value.

height: and **max-height:** and **min-height:** are defined similarly.

12.1.2 Padding

Closest to the content is the padding. It is transparent, so it uses the same background color and background image as the central object.

Exam Question 216 (p.349)**:** Does padding share the same background as the content?
Required Answer: yes

Exam Question 217 (p.349)**:** Can padding style be different on each of the four sides?
Required Answer: yes

Padding has thickness but no other attribute. The thickness can be positive or zero but cannot be negative. The thickness can be different on each of the four sides: top, right, bottom, and left.

Exam Question 218 (p.349)**:** Can "padding:" be negative?
Required Answer: no

12.1.3 Border

Next out from the padding is the border. It has thickness, color, and style. It can have an actual image, like a picture frame.

Exam Question 219 (p.349)**:** Does border share the same background as the content?
Required Answer: yes

Exam Question 220 (p.349)**:** Can border style be different on each of the four sides?
Required Answer: yes

If part of the border is transparent, as with double, dotted, and dashed, it shares the same background as the content and padding.

If the border thickness is zero, the border is invisible.

Border has several standard style options: none (invisible, zero-width border), hidden (similar to none), dotted, dashed, solid (a single line), double, groove (the border appears to be carved inward), ridge (the border appears to be poking outward), inset (the content appears to be pushed inward), and outset (the content appears to be pushed outward).

Exam Question 221 (p.349)**:** Besides none and hidden, list the other eight "border-style:" options.
Required Answer: dotted, dashed, solid, double, groove, ridge, inset, outset

Thickness, color, and style can be different on each of the four sides. If you want, you can have four different thicknesses, four different colors, and four different styles.

Border Image: It is possible to construct custom picture frames, with corners and edges exactly as you might like. The discussion is beyond the scope of this chapter, but it is very entertaining, and the interested student is invited to do a web search for border-image.

12.1.4 Margin

Next out from the border is the margin.

Margin has thickness but not color or style. The margin is always transparent. Margin can be positive, zero, or negative. Adjacent margins may

overlap.

Exam Question 222 (p.349)**:** Does margin share the same background as the content?
Required Answer: no

Exam Question 223 (p.349)**:** Can margin style be different on each of the four sides?
Required Answer: yes

Adjacent vertical (top and bottom) margins may collapse in some cases. This means they share the same space. In that case, the larger of the two margins will be used.

Exam Question 224 (p.349)**:** Can "margin:" be negative?
Required Answer: yes

Negative margin can move items closer together than they would normally be. You can use negative values to cause things to overlap.

12.1.5 Outline

Outline is rarely used. It is probably most used for temporary highlighting because turning it off and on does not cause the page to reflow.

It begins just outside the border and overlaps the margin and maybe beyond.

Like the border, the outline has thickness, color, and style.

Unlike the border, the outline does not take up any space. It simply overlaps whatever is in its way.

Exam Question 225 (p.349)**:** Which box model parameter is used for temporary highlighting?
Required Answer: outline

Exam Question 226 (p.349)**:** Which box model parameter has invert as a color option?
Required Answer: outline

Exam Question 227 (p.349)**:** Can outline style be different on each of the four sides?
Required Answer: no

Outline has the same color options that the border does, plus one more: invert.

Outline has the same style options that the border does, but these things can NOT vary by side. You can only have one color, one thickness, and one style.

12.1.6 Rounded Corners: Border-Radius

It is popular to be able to round the corners of a box. This can be easily done with the `border-radius` attribute. The radii are specified in this order: nw ne se sw. For example:

```
border-radius: 2px 4px 6px 8px;
```

That is, the northwest (top left) corner is specified first. In clock terminology, this is the 10:30 position. Next is the northeast (top right) corner, the 1:30 position. Next the southeast (bottom right) corner, the 4:30 position, and last the southwest (bottom left) position, the 7:30 position.

Exam Question 228 (p.349)**:** Which box model parameter creates rounded corners?
Required Answer: border-radius

The rounding can be elliptical. You can specify two parameters for each corner. Consult the web for examples.

12.2 Best Practices

Invest: Spend the necessary time to make your webpage easy to read. Otherwise your readers will go away. Try hard to serve their needs.

Padding: Use it. If your words go smack up against pictures or screen edges it reduces comprehension for your readers. Give some white space around your paragraphs. I mean on all four sides. `tl;dr`? Big blocks of text get skipped. Small blocks of text are not as daunting and are more likely to be read, or at least started.

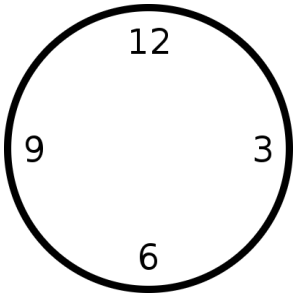
Width: Avoid wide columns. If you are presenting text for people to read, your column should be narrow enough that their eyes can scan left to right easily without losing their place. About 50em (70 characters) is a pretty comfortable width. Think about newspaper columns, for example.

Length: Avoid long paragraphs. (`tl;dr` again.) Studies show that website visitors tend to read the first line or two of the paragraph and then just skim

or skip the rest, unless it is really interesting. The longer the paragraph, the less likely they will read the last word.

Long Pages: These are okay, so long as the content is broken up into reasonable paragraphs. Recognize that your readers may be skimming, not reading every word. Use bold and headings to throw up hooks that will grab reader attention by identifying answers to their questions.

12.3 Clockwise Order: top, right, bottom, left



For padding, border, and margin, each side of the box can be specified separately, by name. It can also be specified as part of a list of up to four measurements, given in **clockwise** order. Clockwise order is (12 o'clock or 0deg) top, (3 o'clock or 90deg) right, (6 o'clock or 180deg) bottom, (9 o'clock or 270deg) left. If (9) is not given, it copies from (3). If (6) is not given, it copies from (12). If (3) is not given, everything copies from (12).

Exam Question 229 (p.349)**:** When we say margin: 9px 8px 7px 6px; what is the value of margin-top?
Required Answer: 9px

Exam Question 230 (p.349)**:** When we say margin: 9px 8px 7px 6px; what is the value of margin-right?
Required Answer: 8px

Exam Question 231 (p.349)**:** When we say margin: 9px 8px 7px 6px; what is the value of margin-bottom?
Required Answer: 7px

Exam Question 232 (p.350)**:** When we say margin: 9px 8px 7px 6px; what is the value of margin-left?
Required Answer: 6px

Exam Question 233 (p.350)**:** When we say margin: 9px 8px 7px; what is the implied fourth value?
Required Answer: 8px

Exam Question 234 (p.350)**:** When we say margin: 9px 8px; what is the implied third value?

Required Answer: 9px

Exam Question 235 (p.350)**:** When we say margin: 9px 8px; what is the implied fourth value?
Required Answer: 8px

12.4 Measurement Units

The length measurement (size) of padding, borders, margins, and outlines can be specified in any of several ways. This variety is in response to the many ways web designers want to control the appearance of their webpages.

12.4.1 Percentages

Distance can be measured by percentage, in which case, by default, it is based on the size of the central object (interior content). This is called **box-sizing: content-box**.

If we specify **box-sizing: border-box**, then distance percentages are based on the size of outer border, which includes the size of the border, padding, and content. Although this is not the default, it is a very popular alternative to content-box sizing.

Exam Question 236 (p.350)**:** List the two "box-sizing:" options.
Required Answer: border-box, content-box

Exam Question 237 (p.350)**:** What "box-sizing:" option is the default?
Required Answer: content-box

12.4.2 Pixels

Some, like **px** (**pixel**), are designed to be displayed on a screen. As previously stated in 1.2 (page 10), there are 47 CSS pixels in one degree of direct vision width. Originally a pixel was one dot on the computer screen. This was the natural measurement the computer used to render things. For many purposes I believe it is the best measurement today.

Exam Question 238 (p.350)**:** How many CSS pixels are in one degree of direct vision?

Required Answer: 47

Exam Question 239 (p.350)**:** Typically how many px are there per inch?
Required Answer: 96

12.4.3 Absolute Lengths: cm, mm, in, pt, pc

Some, like **in** (**inch**), **cm**, and **mm**, are designed to work well with physically printed media or PDF files. The browser will adopt some conversion factor between inches and pixels for displaying things on the screen, depending on how far from the viewer's eyes the screen is typically located.

cm: Distance can be measured in centimeters.

mm: Distance can be measured in millimeters.

in: Distance can be measured in inches.

pt: Distance can be measured in points. This is a traditional printer's measure. A point is 1/72 of an inch. It is also the way we measure font sizes.

Exam Question 240 (p.350)**:** How many pt are there per inch?
Required Answer: 72

Exam Question 241 (p.350)**:** What does pt stand for?
Required Answer: point

pc: Distance can be measured in picas. This is a traditional printer's measure. A pica is 12 points, 1/6 of an inch. Typically this is the distance between baselines of consecutive rows of single-spaced print (text in paragraph form).

Exam Question 242 (p.350)**:** How many pc are there per inch?
Acceptable Answer: 6

Exam Question 243 (p.350)**:** What does pc stand for?
Acceptable Answer: pica

12.4.4 Font-Relative Lengths: em, ex, ch

Some, like **em**, **ex**, and **ch** are designed to scale up or down according to the font of the nearby content.

em: Horizontal distance can be measured in "em"s, where one em is the width of the letter m in the current font.

The size of the em defines the size of the font. If the em is 12pt wide, then you have a 12pt font.

ex: Vertical distance can sometimes be measured in "ex"s, where one ex is the height of the letter x in the current font. An ex defaults to the size of half of an em if it cannot be determined from the font.

ch: Horizontal distance between the centers of adjacent digits in numbers. Technically this is the distance between the centers of adjacent zeros, but as all numbers are typically the same distance center to center, it also describes the distance between any two digits.

12.4.5 Viewport Percentages: vw, vh, vmin, vmax

Some, like **vw**, **vh**, **vmin**, and **vmax** are designed to scale up or down according to the size of the **viewport**, which is the browser window.

vw: Viewport Width. There are 100 units in the total width of the viewport. 50vw is the horizontal middle of the browser window.

Exam Question 244 (p.350)**:** What CSS measurement unit is based on how wide the browser window is?
Required Answer: vw

vh: Viewport Height. There are 100 units in the total height of the viewport. 50vh is the vertical middle of the browser window.

Exam Question 245 (p.350)**:** What CSS measurement unit is based on how tall the browser window is?
Required Answer: vh

vmin: Viewport Minimum. This is the smaller of vw and vh. In portrait mode, vw is smaller. In landscape mode, vh is smaller.

vmax: Viewport Maximum. This is the larger of vw and vh. In portrait mode, vh is larger. In landscape mode, vw is larger.

12.5 Box Math: Ultimate Size

The ultimate width and height used by a paragraph or picture depends on (a) the basic size of the central object, plus (b) the thickness of the padding, plus (c) the thickness of the border, plus (d) the thickness of the margin. Outline thickness does not have any effect. Collapsing vertical margins may allow adjacent margins to overlap each other.

Example: The central object is an image that is 200px wide and 100px tall. We have the following style sheet. What is the final size rendered?

```
img { padding: 10px 20px 30px 40px;
 border: 15px 25px 5px double black;
 margin: 9px 6px;
 outline: 13px red dotted; }
```

Vertically, we have 100px in the image. On top we have 10px padding, 15px border, and 9px margin. On the bottom we have 30px padding, 5px border, and 9px margin. Outline does not count. The total is 100 + 10 + 15 + 9 + 30 + 5 + 9 = 178 pixels.

Horizontally, we have 200px in the image. On the right we have 20px padding, 25px border, and 6px margin. On the left we have 40px padding, 25px border, and 6px margin. Outline does not count. The total is 200 + 20 + 25 + 6 + 40 + 25 + 6 = 322 pixels.

Because box math is painful, the `box-sizing: border-box;` model is increasingly popular. It just seems to make lots of things easier.

12.6 Float and Clear

Everything on a webpage is organized into boxes, and is surrounded by padding, border, (outline,) and margin. Normally the larger boxes are just stacked vertically on the page. (Characters and words stack horizontally within the paragraph.)

Float lets the big things stack differently.

For example, say you want to have an image displayed next to some text. You actually want the text to flow around the image. How can you do that? In section 10.1.1 (page 99) we introduced this idea. We showed that you

can float an image left or right, and flow the following content around that image like a rock in a stream of water.

Exam Question 246 (p.350)**:** List the two "float:" options.
Required Answer: left, right

As you float things, they will go left or right as far as they can, and as high on the webpage as they can. If you float lots of things, it may not look natural, as they might leave lots of empty space below the outermost objects.

Exam Question 247 (p.350)**:** List the three "clear:" options.
Required Answer: left, right, both

Clear cancels out the floating that is pending and forces the following items to appear below all that were above. You can clear left, clear right, or clear both.

12.7 For Further Study

http://www.w3.org/TR/css3-values/ is intended to officially list and define things like px, pt, in, mm, and more.

http://www.w3schools.com/css/css_boxmodel.asp explains more about the box model in general.

http://www.w3schools.com/css/css_border.asp explains more about borders.

http://www.w3schools.com/css/css_margin.asp explains more about margins.

Chapter 13

Positioning: Relative and Absolute

Contents

Normally the content of a webpage simply flows from top to bottom. We do get a bit fancy with float left and float right, but everything pretty much just flows down the page.

Sometimes we want to get really fancy. We want to have a menu bar that stays in the same place, at the top, bottom, left, or right side of the viewport, no matter how the rest of the content scrolls.

Or we may want to do drop-down menus that only appear when the mouse hovers over the menu heading. In section 15.6 (page 159) we illustrate this.

As we play with positioning, we often want things to overlap (or underlap), one on top of the other, like menus on top of content, or one piece of content on top of another.

Positioning helps us do all those things. Positioning can take an element out of the normal flow and anchor it to the viewport or to some movable element of the page.

There are four basic position types: **static**, **relative**, **absolute**, and **fixed**.

Exam Question 248 (p.350)**:** List the four CSS position: options.
Required Answer: static, relative, absolute, fixed

Why these four words? I have no clue. These words do not have any clear natural meaning. They are confusing to the point of being meaningless. You just have to memorize them.

13.1 Directional Adjustments

Static is the default, and should never be used except when trying to cancel out one of the other options.

Except for static, we can adjust the position of an element by giving an offset to one or more of its edges.

There are four directional adjustments: **left:**, **right:**, **top:**, and **bottom:**.

Exam Question 249 (p.350)**:** List the four CSS positioning directions.
Required Answer: left, right, top, bottom

Positive distances move toward the center of your "anchor" (the thing on which your position is based), and negative distances move away from center.

Thus, for left: a positive distance moves you right. For right: a positive distance moves you left. For top: a positive distance moves you down. For bottom: a positive distance moves you up.

Exam Question 250 (p.351)**:** If we say left: 5px; which direction will things move? (left or right)
Required Answer: right

Exam Question 251 (p.351)**:** If we say left: -5px; which direction will things move? (left or right)

Required Answer: left

Exam Question 252 (p.351)**:** If we say right: 5px; which direction will things move? (left or right)
Required Answer: left

Exam Question 253 (p.351)**:** If we say right: -5px; which direction will things move? (left or right)
Required Answer: right

Exam Question 254 (p.351)**:** If we say top: 5px; which direction will things move? (up or down)
Required Answer: down

Exam Question 255 (p.351)**:** If we say top: -5px; which direction will things move? (up or down)
Required Answer: up

Exam Question 256 (p.351)**:** If we say bottom: 5px; which direction will things move? (up or down)
Required Answer: up

Exam Question 257 (p.351)**:** If we say bottom: -5px; which direction will things move? (up or down)
Required Answer: down

13.2 Elements Retained In Flow

In two cases, the positioned element is retained in the flow. That is, it continues to take up space between the things that come before and after it in the HTML, and it inherits its width from its surroundings.

Exam Question 258 (p.351)**:** List the two CSS positions that retain the element in the flow.
Required Answer: static, relative

13.2.1 position: static;

Exam Question 259 (p.351)**:** Which CSS position is the default?
Required Answer: static

Static is the normal way of things. Elements take up space in the flow. This is the default.

Static is never used as a basis for other positioning.

13.2.2 position: relative;

Relative is like static in that it takes up space in the flow.

Relative is different from static in that it can be used as a basis for other positioning, and it also supports directional adjustments.

Directions are relative to its natural place in the flow.

13.3 Elements Removed From Flow

In two cases, the positioned element is removed from the flow. That is, it takes up no space between the things that come before and after it, and it does not inherit its width from its surroundings. Instead, it overlaps or hides beneath the other content.

Exam Question 260 (p.351)**:** List the two CSS positions that remove an element from the flow.
Required Answer: absolute, fixed

When an element is removed from the flow, the rest of the document flows as though it were not even there. It takes up no space in the flow. It does not affect the position of static or relative things.

When an element is removed from the flow, it is usually necessary to specify a width for the element, since it cannot be logically deduced from its surroundings. Optionally you can also set a height.

13.3.1 position: absolute;

Exam Question 261 (p.351)**:** Which CSS position depends on its closest positioned ancestor?
Required Answer: absolute

With absolute, the element is removed from the flow. It takes its position based on the closest ancestor that is not positioned static. Its position will follow that ancestor as the webpage is scrolled up or down, left or right.

If there is no ancestor that is positioned, it uses body. It is the same as if you said `body { position: relative; }` in your CSS.

Directions are relative to the corresponding edge of the ancestor. Positive numbers move toward the center of the ancestor, and negative numbers move away.

left: 5px; This will move the left edge of the element 5px to the right compared to the left edge of its ancestor. Negative numbers move it to the left.

right: 5px; This will move the right edge of the element 5px to the left compared to the right edge of its ancestor. Negative numbers move it to the left.

top: 5px; This will move the top edge of the element 5px lower compared to the top edge of its ancestor. Negative numbers move it up.

bottom: 5px; This will move the bottom edge of the element 5px higher compared to the bottom edge of its ancestor. Negative numbers move it lower.

If both top and bottom are specified, top takes precedence and bottom is ignored.

13.3.2 position: fixed;

Exam Question 262 (p.351)**:** Which CSS position depends on the viewport?
Required Answer: fixed

With fixed, the element is removed from the flow. It takes its position relative to the edges of the viewport. Its position will not change even if the rest of the webpage is scrolled up or down, left or right.

This can be handy for headers or footers or navigation that you want to always be visible. But because they are removed from the flow, you may need to add margins to the rest of your content so it is not covered by the header or footer.

Directions are relative to the corresponding edge of the viewport. Positive numbers move toward the center of the viewport, and negative numbers move away.

left: 5px; This will move the left edge of the element 5px to the right compared to the left edge of the viewport. Negative numbers move it to the left (off screen).

right: 5px; This will move the right edge of the element 5px to the left compared to the right edge of the viewport. Negative numbers move it to the right (off screen).

top: 5px; This will move the top edge of the element 5px lower compared to the top edge of the viewport. Negative numbers move it up (off screen).

bottom: 5px; This will move the bottom edge of the element 5px higher compared to the bottom edge of the viewport. Negative numbers move it down (off screen).

If both top and bottom are specified, top takes precedence and bottom is ignored.

13.4 Elements that Overlap / Underlap (z-index)

When elements overlap, the one with the higher **z-index** covers the one with the lower z-index. Z comes from the z axis in three-dimensional geometry, where the x axis is left-right, the y axis is up-down, and the z axis is closer or farther away. The higher the z-index, the closer the element is to the person viewing the webpage.

Example: `.over { z-index: 1; }`

Exam Question 263 (p.351)**:** What attribute specifies stacking order for overlap?
Required Answer: z-index

Currently there are no three-dimensional special effects. With 3d-glasses that might happen in the future. Right now, everything is flat with higher z-indexes covering lower ones.

With hover and z-index adjustment, you could pull a card out of a deck of playing cards, or show a picture out of a pile of pictures.

The default z-index is zero.

If two elements have the same z-index (maybe the default of zero), the one that appears later in the HTML will go on top.

13.5 Centering

Left, right, top, and bottom work great for positioning relative to the edges. Corners are easy. But what if you want to put something in the center?

Percentages help us. We can say `left: 50%;` to put the left edge in the center of the screen. We can say `top: 50%;` to put the top edge in the center of the screen.

But what if we want the middle of the element to be in the middle of the screen?

We can also use `transform: translateX(-50%);` to move the element left by half of its width, and we can use `transform: translateY(-50%);` to move the element up by half of its height.

Obviously, if we can center this way, we can position something anyplace on the screen by using the same percentage-based approach.

13.6 Dynamic Repositioning: Hover Effects

http://designshack.net/?p=26842 has a fascinating article that answered a nagging question for me. I wanted to create a sliding-door or opening-window effect. Unfortunately, those terms seem to be already in use to identify something other than what I wanted, so it was difficult to find my objective.

I decided to call it Peek-A-Boo.

Basically, I wanted to show an image that was covered up by another image. Then, when I hovered my mouse over the stacked images, the hidden picture would be revealed.

I tried several approaches but nothing really worked. Finally I came across on the article mentioned above.

This is how he made it work.

Basically you will have a multi-layer background image in an otherwise empty div. The last layer will be the hidden image. The other layers will move out of the way when you hover over them.

Decide exactly how big the images are going to be. (There may be a way to do this automatically, but short of using JavaScript I did not figure anything

out.)

Then make the images the right size and specify the div (or whatever) to also be the right size.

Finally, use :hover to adjust the background position of the front images so they slide out of the way, possibly with a transition time.

Here is an example that I worked up. It uses two images, barndoor.png (400x450) and don.jpg (400x475) set into a div that is specified to be 400x450. (The bottom 25ps of don.jpg will not be shown.)

The barndoor image is used twice, once offset to the left by 200px, and once offset to the right by 200px. When :hover comes into effect, the offset is increased to 400px to push the barndoor completely out of the way, revealing the bottom image, a picture of myself.

To further dress things up a bit, I used border-radius to round the corners of the overall image.

```
<!DOCTYPE html><head lang=en><meta charset=utf-8>
<title>Peek A Boo</title>
<style>
/* http://designshack.net/?p=26842 */
* { transition: 2s; }
body { text-align: center; }
div#picture { width: 400px; height: 450px; border-radius: 20px;
 background-image: url(barndoor.png), url(barndoor.png), url(don.jpg);
 background-repeat: no-repeat;
 background-position: -200px 0, 200px 0, 0 0; }
div#picture:hover { background-position: -400px 0, 400px 0, 0 0; }
</style>
</head><body>
<div id=picture></div>
</body>
```

Chapter 14

JavaScript and AJAX

Contents

Markup, styling, and action are the three main aspects of web design.

Markup (html) is pretty easy. It is the content creation part of web design.

Styling (css) is way fun. It is the artistic part of web design.

Action is what we will talk about now. And that means JavaScript and AJAX.

JavaScript gives you the ability to modify a page dynamically, right on the user's device, without reloading the page from a server somewhere.

AJAX gives you the ability to send and receive information without waiting for the submit button to be pressed.

14.1 Yikes! Programming?

JavaScript is a programming language, but most web designers let somebody else do all the programming. They just insert some "canned" routines into their webpages.

That is the level of skill you must attain.

To do that, you must understand enough JavaScript to read it at an introductory level, and to make simple changes to it when needed.

If you happen to also be a programmer, yay. But this book will barely skim the surface of JavaScript programming. This chapter is JavaScript for Web Designers, not JavaScript for Programmers.

There is a very large body of free JavaScript code available on the Internet. You can find online various snippets of JavaScript that you can include in your webpages. Others have written them so you do not need to be a programmer. But you need to know how to insert them into your webpage properly. Google search "free javascript code" for many examples.

For now, just breathe deeply and relax.

14.2 What can JavaScript Do?

You probably use JavaScript-based websites everyday. Web-based email is all about JavaScript. News websites are literally infested with JavaScript. Facebook and other websites use JavaScript to continually load new content as you scroll your screen.

More? One of the most common uses of JavaScript is the (more) option you commonly see after the first few lines of some content. We know that users only read the first line or two and then skip on to the next item. Once in a while they are interested and want to see more. JavaScript lets them click on a (more) link that reveals the rest of the content, and a (less) link that hides it again.

Variations: Show the artwork with various frames. Show the clothing with various print patterns. Show the room with various wall colors. JavaScript can make this instantaneous.

Filtering: Reduce and clear away clutter, showing the user just those pieces of content that match their interests.

Validation: See whether the credit card number is legal. (Credit card numbers include a check-digit to quickly find typing errors.)

See whether the email address they typed twice is the same both times.

See whether the password they typed twice is the same both times. And see whether the password is strong or weak.

Calculators: Fill in a few blanks and see a result. Shipping costs? Mortgage or car payments?

Widgets: Paid advertisements, trending hashtags, latest comments, newsfeeds, weather, quote of the day, joke of the day: these use AJAX to update your screen with new information.

But Wait! When users want to leave your webpage, JavaScript can put up a message with a special offer. This can be really annoying, but it might be just what your boss wants.

14.3 Try It: Changing Colors

Let's get started. Add this HTML which includes a bit of JavaScript to any webpage you care to mess up. This code will allow the user to change the color of the text and the color of the background on a webpage, simply by moving the mouse.

```
<p>Text:
<span onmouseover="body.style.color='red'">[Red]</span>
<span onmouseover="body.style.color='white'">[White]</span>
<span onmouseover="body.style.color='blue'">[Blue]</span>
<span onmouseover="body.style.color='black'">[Black]</span>
Background:
<span onmouseover="body.style.backgroundColor='red'">[Red]</span>
<span onmouseover="body.style.backgroundColor='white'">[White]</span>
<span onmouseover="body.style.backgroundColor='blue'">[Blue]</span>
<span onmouseover="body.style.backgroundColor='black'">[Black]</span>
```

The color names must have quote marks around them. Otherwise, they will be treated as JavaScript variables that are probably undefined.

Reload the page. Then run your mouse over any of the new color words that appeared. Depending on the word, your page should change color.

Details:

The **span** tag simply identifies a portion of your webpage so styles and other things can be applied to it. Here we use it as a vehicle to let us apply a JavaScript event handler.

The **onmouseover** attribute for each span specifies a bit of JavaScript that will be carried out when the mouse passes over the contents of the span.

The **body** object is the body of the webpage. It is the same as the `<body>` tag that is used to surround the webpage's content. More properly we should say document.body because the body is a sub-part of the document, but body works here.

The **style** attribute is attached to body. It gives us access to every style characteristic of the body, including colors, fonts, margins, paddings, you name it.

The **color** attribute is attached to style. It represents the foreground color of the font used.

By changing any attribute, we immediately change that attribute for everything in body, unless something else with a higher priority intervenes.

The **backgroundColor** attribute is attached to style. It represents the background color of the font used.

14.4 Things To Know

Anything you can do in CSS, you can also do in JavaScript. Anything you can specify in CSS you can specify or change using JavaScript.

The two ways are intentionally very similar.

In **CSS**, we would say this:
```
body { background-color: white; }
```

In **JavaScript**, we would say this:
```
body.style.backgroundColor='white';
```

But there are a couple of things that you need to learn. You need to learn the **DOM**, and you need to learn Camel Case.

14.4.1 The DOM

The **DOM** is the Document Object Model. It is a way of identifying parts of the webpage so they can be used or changed by a JavaScript program.

Exam Question 264 (p.351)**:** What does DOM stand for?
Acceptable Answer: document object model

When a webpage is going to be displayed, it must first be loaded into memory inside the browser.

This loading process converts the text of the webpage into a memory-only representation called the DOM. The HTML exists merely to let the browser use it to create the DOM. The DOM is the real webpage.

The DOM is sometimes referred to as a tree, or as the document tree.

Once the tree is built, the browser uses it to render the page. To render means to display the page.

The DOM can be modified "on the fly" by using JavaScript. When the DOM is modified, the browser is required to immediately update the displayed page so it correctly matches the DOM.

None of these changes affect the original webpage that was presented by the server. They only affect the webpage that is seen at the browser.

14.4.2 Camel Case

One thing we wish to point out here is the use of Camel Case for naming the background color attribute.

You may have noticed that with JavaScript backgroundColor was used instead of the CSS version, background-color.

CSS is happy to have dashes inside attribute names. In JavaScript, dashes mean subtraction.

So, instead of having dashes, the attribute names are pushed together, and any letter that was after a dash gets changed to uppercase. The rest are lowercase.

When we do this, we call the result **camel case** to emphasize that the profile of the word looks like the back of a camel.

Exam Question 265 (p.352)**:** Convert border-top-right-radius to camel case.
Required Answer: borderTopRightRadius

14.4.3 Where Should JavaScript Go?

On your webpage, where should you put the JavaScript?

Normally the `<script>` elements are included as part of the head, somewhere between the `<head>` and `</head>` tags.

Sometimes they can be placed elsewhere inside the body of the document.

They can also be imported from another file by including special wording in the head section of the webpage.

`<script src=...></script>` is one way.

14.4.4 On... Triggers

JavaScript programs are usually triggered by external events. Specifically, they are most often triggered by the actions of the user, such as moving or clicking the mouse or pressing a key on the keyboard. They can also be triggered by events that happen as a page is loaded.

http://www.w3schools.com/tags/ref_eventattributes.asp has a list of event attributes. These are the triggers. They go within other markup. They all start with the word "on".

onclick and **click** mark code that is triggered when the user clicks on something.

Exam Question 266 (p.352)**:** What HTML attribute= is used to run a JavaScript action when the user single-clicks on the specified item?
Required Answer: `onclick=`

Exam Question 267 (p.352)**:** What HTML attribute= is used to run a JavaScript action when the user double-clicks on the specified item?
Required Answer: `ondblclick=`

onmouseover and **mouseover** mark code that is triggered when the user moves the mouse onto something.

Exam Question 268 (p.352)**:** What HTML attribute= is used to run a JavaScript action when the mouse enters the specified item?
Required Answer: `onmouseover=`

onmouseout and **mouseout** mark code that is triggered when the user moves the mouse away from something.

Exam Question 269 (p.352)**:** What HTML attribute= is used to run a JavaScript action when the mouse leaves the specified item?
Required Answer: `onmouseout=`

A full treatment of events and event handlers is beyond the scope of this book. But to give you a sense of what is possible, here is a list of event handlers from the w3.org website (2012-12).

Event Handlers: onabort, onafterprint, onbeforeprint, onbeforeunload, onblur, oncancel, oncanplay, oncanplaythrough, onchange, onclick, onclose, oncontextmenu, oncuechange, ondblclick, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, ondurationchange, onemptied, onended, onerror, onfocus, onfullscreenchange, onfullscreenerror, onhashchange, oninput, oninvalid, onkeydown, onkeypress, onkeyup, onload, onloadeddata, onloadedmetadata, onloadstart, onmessage, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onmousewheel, onoffline, ononline, onpagehide, onpageshow, onpause, onplay, onplaying, onpopstate, onprogress, onratechange, onreset, onresize, onscroll, onseeked, onseeking, onselect, onshow, onstalled, onstorage, onsubmit, onsuspend, ontimeupdate, onunload, onvolumechange, onwaiting.

If one looks interesting to you, Google search it.

14.4.5 Is Java the Same Thing?

People sometimes think Java and JavaScript are the same. They are not. They share the same first four letters in their name. That's about it. You can blame Marketing.

Exam Question 270 (p.352)**:** Are Java and JavaScript the same thing?
Acceptable Answer: no

Exam Question 271 (p.352)**:** Are Java and JavaScript related?
Required Answer: no

14.4.6 Indexed Content

JavaScript can create content, but such content is not indexed by search engines. Use static HTML for any content that you want to be seen by search engines.

14.5 Try It: More or Less (Hide and Reveal)

To extend your skills, here is some simple JavaScript you can use to hide or reveal content.

This is useful when you have lengthy content available, but you only want to show the first few lines in case the user is not interested. They can click on `[more]` for the rest of the content.

It could also be useful for sample test questions, where the question is shown but the answer is hidden until the user is ready to see it.

Here we use it to show a well-worn joke, where the punch line is revealed on demand.

```
<p>Why did the chicken cross the road?
<u onclick='document.getElementById("abc").hidden=false'>
[answer]</u>
<p id=abc hidden>To get to the other side!
<u onclick='document.getElementById("abc").hidden=true'>
[hide]</u>
```

HTML Details:

The **p** tag identifies a paragraph.

The **u** tag causes `[answer]` and `[hide]` to be underlined. You could use **span** or any other tag.

The **id** attribute designates an ID for the paragraph. JavaScript will use that ID to find and modify that paragraph.

The **hidden** attribute designates the paragraph as initially hidden. JavaScript can be used to reveal it.

The **onclick** attribute specifies a bit of JavaScript that will be carried out when the mouse is clicked on the target.

JavaScript Details:

The **document** object is the entire document.

The **getElementById** function searches the document for an object that has the specified ID, which in this case is `abc`.

The **hidden** DOM attribute specifies whether the object is to be hidden or revealed.

14.6 Try It: Make a Function

Functions let us simplify our webpage.

The onclick= html can get rather verbose. You can shorten it considerably by creating a JavaScript function.

In the body.style.color example, we used JavaScript to immediately modify an attribute of body. We are doing exactly one thing.

Sometimes we want to do several things all at once. We can put those things into a JavaScript function and give it a name. Then, in the onclick= html we can simply mention the name of the function and all those things will happen. This can greatly simplify your html, plus it collects all the JavaScript into one place.

Warning: The name you give to your JavaScript function must be different than any of the IDs you have on your webpage.

In this example, we create a function named `colors` that does several steps, and we call it with two parameters, the foreground (text) color, and the background color. Inside the function, we change both colors.

```
<p>
<span onmouseover="colors('white','red')">[White/Red]</span>
<span onmouseover="colors('red','white')">[Red/White]</span>
<span onmouseover="colors('red','blue')">[Red/Blue]</span>

<script>
function colors(fg,bg) {
  document.body.style.color=fg;
  document.body.style.backgroundColor=bg; }
</script>
```

14.7 Try It: Change the Class of an Element

If you want to change a whole bunch of styling all at once, one of the easiest ways is to assign the styling to a **class** and then just change the class of the element.

Here is a JavaScript function we have named **reclass** that uses **getElementById** and **setAttribute** to change the class of an element. Place it in your `<head>`.

```
<script>
function reclass(e,c){
 document.getElementById(e).setAttribute("class",c) }
</script>
```

Here is a style sheet that defines two classes to be used in our example. Place it in your `<head>`.

```
<style>
.x { padding: 10px; border: red solid 5px;
     margin: 5px; background-color: yellow; }
.y { padding: 20px; border: blue double 20px;
     margin: 10px; background-color: pink; }
</style>
```

Here is some `<body>` code to change the class of an item. We click on a `<button>` to call our JavaScript function **reclass** to find the item whose ID is "abc" and change its class to something new.

```
<button onmousedown="reclass('abc','x')">Red</button>
<button onmousedown="reclass('abc','y')">Blue</button>
<button onmousedown="reclass('abc','z')">None</button>
```

Here is an example of how to mark the paragraphs or images. Establish an ID for the item that will be changed.

```
<p id=abc>This paragraph changes style.
```

Chapter 15

Lists and Menus

Contents

Just as tables have been around forever, lists have too. But instead of presenting tabular (two dimensional) material, lists are linear (one dimensional). They can be numbered (or lettered) or just have bullets (or nothing at all). And they can have sub-lists.

15.1 Basic Markup

If the list is to be numbered, we begin it with `<ol>` (for "ordered list") and end it with `</ol>` (required). We start each item with `<li>` and end it with `</li>` (optional).

If the list is to have bullets (un-numbered), we begin it with `<ul>` (for "unordered list") and end it with `</ul>` (required). We start each item with `<li>` and end it with `</li>` (optional).

To some extent, by styling we can make an `<ol>` to look like a `<ul>` and vice versa.

15.2 Vertical Indented Lists

By default, lists are naturally vertical, from top to bottom. In section 15.6 (page 159) we will tell how to make horizontal lists. Sub-lists are indented within the overall list.

List items generally begin with a number or a **bullet**. The content of the list item is indented and the bullet hangs out to the left.

Sub-lists are very common, and are indented another level beyond the outer list. We can have lists of lists of lists, to as many levels as we want.

Exam Question 272 (p.352)**:** The HTML tag "ol" stands for what word(s)?
Acceptable Answer: ordered list

Exam Question 273 (p.352)**:** For each <ol> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 274 (p.352)**:** The HTML tag "ul" stands for what word(s)?
Acceptable Answer: unordered list

Exam Question 275 (p.352)**:** For each <ul> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 276 (p.352)**:** The HTML tag "li" stands for what word(s)?
Acceptable Answer: list item

Exam Question 277 (p.352)**:** For each <li> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: optional

Exam Question 278 (p.352)**:** Can lists be nested (a list inside a list)?
Required Answer: yes

15.3 ol: Ordered Lists

Ordered lists can be numbered in any of several ways. This is controlled by the list-style-type attribute.

```
ol.a { list-style-type: upper-roman; }
ol.b { list-style-type: lower-alpha; }

<ol class=a><li>...</ol>
<ol class=b><li>...</ol>
```

A variety of numbering systems are available, including armenian, cjk-ideographic, decimal, decimal-leading-zero, georgian, hebrew, hiragana, hiragana-iroha, katakana, katakana-iroha, lower-alpha, lower-greek, lower-latin, lower-roman, upper-alpha, upper-latin, and upper-roman.

Exam Question 279 (p.352)**:** For numbered lists, are there more than 10 numbering systems available?
Required Answer: yes

Exam Question 280 (p.352)**:** What CSS list attribute controls the numbering system?
Acceptable Answer: list-style-type:

We can make the numbering start someplace other than one.

```
<ol start=50>
```

Exam Question 281 (p.353)**:** For numbered lists, can we specify the starting value?
Required Answer: yes

We can make the numbering count down instead of counting up. By default, it will end at 1.

```
<ol reversed>
```

Exam Question 282 (p.353)**:** For numbered lists, can we count down instead of up?
Required Answer: yes

15.4 ul: Unordered Lists

Unordered lists normally have bullets. We can choose which kind. This is controlled by the list-style-type attribute.

```
ul.a { list-style-type: disc; }
ul.b { list-style-type: circle; }
```

```
ul.c { list-style-type: square; }
ul.d { list-style-type: none; }
ul.e { list-style-image: url(...); }

<ul class=a><li>...</ul>
<ul class=b><li>...</ul>
```

15.5 li: List Items

Normally all we need is `<li>` at the start of a list item. We can over-ride the numeric value of any item by specifying a value, like this:

```
<li value=20>
```

Exam Question 283 (p.353)**:** What attribute overrides the number of an li element?
Acceptable Answer: value=

15.6 Horizontal Lists (Dropdown Menus)

List are often menu items, and we might like to position them across the top of our webpage, using hover to reveal the sub-lists.

Phase One: We start with our menu system, unstyled. The main thing to notice is that we have a main list with four menu items, and each menu item has several sub-menu items. Also, we have placed an ID on the whole list. We can place this into a webpage and see how it looks.

```
<ul id=menu>
<li>menu1
  <ul>
  <li>item1a
  <li>item1b
  </ul>
<li>menu2
  <ul>
  <li>item2a
  <li>item2b
  <li>item2c
```

```
  <li>item2d
  </ul>
<li>menu3
  <ul>
  <li>item3a
  <li>item3b
  <li>item3c
  </ul>
<li>menu4
  <ul>
  <li>item4a
  <li>item4b
  <li>item4c
  <li>item4d
  <li>item4e
  </ul>
</ul>
```

When we insert this into a webpage, we will get a regular list of lists.

Phase Two: To convert this into a horizontal menu, our first step is to modify the style sheet so the outer list items are "display: inline-block". We will also assign "position: relative" to the outer menu items and "position: absolute" to the inner lists. See chapter 13 (page 138) for more information on positioning.

```
#menu li { display: inline-block; position: relative; }
#menu li ul { position: absolute; }
```

When we refresh our webpage, we will see a horizontal list of menus, but they are all scrunched up together.

Exam Question 284 (p.353)**:** What styling attribute makes a list horizontal?

Acceptable Answer: display: inline-block;

The default display attribute for lists is `display: block;`

Phase Three: We need some box model help. We can add these lines to our style sheet.

```
#menu li { width: 150px; border: 1px black solid;
```

```
 border-radius: 15px 15px 0 0; padding: 5px 10px;
 color: black; background-color: silver; }
#menu li:hover { color: white; background-color: black; }
#menu li li { text-align: left; border-radius: 0; width: 200px; }
#menu li ul { left: -6px; padding: 5px; }
```

When we refresh our webpage, we will see the contents are pretty much as they were before, but greatly beautified. Also, the hover attribute has been activated to highlight the menu item under the mouse.

Phase Four: The only thing left is to make the drop-down menus disappear until we want to see them. We do this by manipulating the display: attribute. We do "display: none" by default, and "display: block" when we hover.

```
#menu li ul { display: none; }
#menu li:hover ul { display: block; }
```

And we are done! We have a drop-down menu system.

Exam Question 285 (p.353)**:** What styling attribute makes an element disappear?
Acceptable Answer: display: none;

`display: block;` creates a vertical column of blocks.

`display: inline-block;` creates a horizontal row of blocks, similar to what `float: left;` would do.

Exam Question 286 (p.353)**:** What styling attribute makes an element (re)appear?
Acceptable Answer: display: block;

15.7 For Further Study

http://www.w3schools.com/tags/tag_ol.asp has more on ordered lists.

http://www.w3schools.com/tags/tag_ul.asp has more on unordered lists.

http://www.w3schools.com/cssref/pr_list-style.asp has list styling options.

Chapter 16

Tables

Contents

The table is a very old HTML structure. It has been around forever. It is pretty simple, but it has a few nice options.

16.1 Basic Markup

You begin it with `<table>` and end it with `</table>` (required). You start each row with `<tr>` and end it with `</tr>` (optional). You start each heading cell with `<th>` and end it with `</th>` (optional). You start each data cell with `<td>` and end it with `</td>` (optional).

The size of each cell is determined automatically by the browser so the web designer does not have to worry about it. You can specify the width if you want. More on that below.

The `</tr>` closing tag is optional because it is implied whenever a new row starts or when the table ends.

The `</th>` and `</td>` closing tags are optional because they are implied whenever a new cell starts or when the row ends.

Exam Question 287 (p.353)**:** For each <table> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 288 (p.353)**:** The HTML tag "tr" stands for what word(s)?
Acceptable Answer: table row

Exam Question 289 (p.353)**:** For each <tr> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: optional

Exam Question 290 (p.353)**:** The HTML tag "th" stands for what word(s)?
Acceptable Answer: table heading

Exam Question 291 (p.353)**:** For each <th> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: optional

Exam Question 292 (p.353)**:** The HTML tag "td" stands for what word(s)?
Acceptable Answer: table data

Exam Question 293 (p.353)**:** For each <td> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: optional

Exam Question 294 (p.353)**:** Can tables be nested (a table inside a table)?
Required Answer: yes

Here is an example of a simple table:

```
<table>
<tr><th>Left Header<th>Middle Header<th>Right Header
<tr><td>Upper Left<td>Upper Middle<td>Upper Right
<tr><td>Middle Left<td>Middle Middle<td>Middle Right
<tr><td>Lower Left<td>Lower Middle<td>Lower Right
</table>
```

The result would be rendered something like this:

Left Header	Middle Header	Right Header
Upper Left	Upper Middle	Upper Right
Middle Left	Middle Middle	Middle Right
Lower Left	Lower Middle	Lower Right

16.2 Past Abuses

You can put almost anything into a cell, including another entire table. The cells are lined up nicely which makes it tempting to use them for the layout of non-tabular material.

In fact, tables have been grossly abused in the past as a mechanism for controlling the **layout** of webpages. If we go back to some old-timey tutorials, we will learn more than we should about using tables to position the graphical elements of webpages.

Exam Question 295 (p.353)**:** Should tables be used for the layout of non-tabular material?
Required Answer: no

The problem is that such layout is brittle. It does not slide gently into the future as screen resolutions and dimensions change.

The solution is to use things like float and clear for the positioning of webpage elements, and to restrict the use of tables to things that really are tabular.

Avoid using tables as devices for positioning non-tabular content.

16.3 Cell Styling

Most of the styling is pretty obvious.

text-align: is used to modify the natural alignment of table elements. `<td>` elements are naturally left aligned. `<th>` elements are naturally centered and bolded.

color: is used to color the lettering that appears in a cell.

background-color: is used to color the background that appears behind the lettering in a cell.

background-image: is used to place an image in a cell. We can also use a foreground image with the `<img>` tag.

border: is used to draw lines between cells. There is an old markup, `<table border=1>`, that is deprecated but was used to automatically add lines around and between cells. The border: mechanism is much more powerful.

Exam Question 296 (p.353)**:** Is the table border= attribute deprecated?
Required Answer: yes

16.4 Merging Cells

Sometimes we want to merge cells. This is especially true for headings that span more than one column. It can also be true for left-edge headings that span more than one row.

Exam Question 297 (p.353)**:** What HTML attribute= is used to merge table cells horizontally?
Required Answer: `colspan=`

Exam Question 298 (p.353)**:** What HTML attribute= is used to merge table cells vertically?
Required Answer: `rowspan=`

Vertical Text: If we are spanning (merging) lots of rows, especially on the left-hand edge of the table, we might want to use vertically oriented lettering. This can be done using the transform: rotate() property. Transform has an amazingly wide variety of options, but that is beyond the scope of what we want to cover in this book.

The following CSS will rotate text -90 degrees (so the text reads from bottom to top) if it is within a td cell that is the first child of its row. (Sadly it does not also rotate your cursor, so copy-paste can be confusing.) 270deg means the same as -90deg.

```
td:first-child { transform: rotate(-90deg); }
```

Exam Question 299 (p.354)**:** What CSS attribute lets us rotate something?
Acceptable Answer: transform: rotate();

We may also be able to use the writing-mode: property that allows us to specify whether writing is top to bottom and left to right or some other sequence. This might be handy for arabic (right to left) or chinese (top to bottom).

If we want horizontal letters, one letter per line, we can try something like this, which acts like putting `<br>` after every letter.

```
{ width: 1px; word-wrap: break-word; text-align: center; }
```

Another option is to create an image, which would give us complete control, but that just seems wrong because we freeze everything and destroy copy-paste. It is much better to use actual text instead of an image if we can.

16.5 Controlling Width

Cell widths are usually decided by the browser depending on the content that is present. Usually the browser tries to minimize the overall length of the table.

Because the browser automatically takes control of sizing, things like **min-width:** and **max-width:** and **min-height:** and **max-height:** apply to many parts of a webpage, but they **normally** do not work on table cells.

Exam Question 300 (p.354)**:** Does max-width: normally work with table cells?
Required Answer: no

This can be frustrating. But there is a solution. Simply take away the browser's right to adjust cell widths by using the table-layout: fixed attribute.

```
table { table-layout: fixed; }
```

Exam Question 301 (p.354)**:** List the two "table-layout:" options.
Required Answer: auto, fixed

With auto, which is the default, we get the normal behavior with the browser setting widths by the widest unbreakable content it finds. With fixed the width is determined by the first row of the table and is not changed by later content. This makes layout faster and more consistent for settings where data may change from time to time.

16.6 Row Styling

One common form of row styling is to alternate the background color or border thickness on every other line, or every fifth line, to make it easier for the user's eye to follow across the table through a sea of information.

```
tr:nth-child(2n) { background-color: white; }
tr:nth-child(2n+1) { background-color: yellow; }
```

Exam Question 302 (p.354)**:** What CSS selector targets even numbered rows?
Required Answer: tr:nth-child(2n)

Exam Question 303 (p.354)**:** What CSS selector targets odd numbered rows?
Required Answer: tr:nth-child(2n+1)

We can also help the user clearly see one whole line by styling it differently using hover, perhaps with an outline or a background color.

```
tr:hover { outline: thin red solid; }
tr:hover { background-color: #9999ff; }
```

Remember that when two stylings conflict, the later one usually wins. Therefore, the more general styling should be applied first and the exception should be applied last. In this case, nth-child should be applied first and hover last.

16.7 Column Styling

When using table-layout: fixed, we can carefully assign column widths by specifying the width: attribute for each of the `<th>` elements.

The class Approach: Our first approach assigns a class to each column. Of course, we can use meaningful names instead of .th1 to help us remember what we are trying to style. Then we simply add a class attribute to each heading element.

```
.th1 { width:  20px; }
```

```
.th2 { width:  50px; }
.th3 { width: 120px; }
.th4 { width:  20px; }
.th5 { width:  40px; }

<th class=th1>First Heading
<th class=th2>Second Heading
<th class=th3>Third Heading
<th class=th4>Fourth Heading
<th class=th5>Fifth Heading
```

The nth-child Approach: Assiging class attributes to each heading can get a bit cumbersome. We can use nth-child to control width.

```
th:nth-child(1) { width:  20px; }
th:nth-child(2) { width:  50px; }
th:nth-child(3) { width: 120px; }
th:nth-child(4) { width:  20px; }
th:nth-child(5) { width:  40px; }

<th>First Heading
<th>Second Heading
<th>Third Heading
<th>Fourth Heading
<th>Fifth Heading
```

The no "th" Approach: Styling by way of th can be limiting because we may want to assign the text-align property differently on a per-column basis like this:

```
td:nth-child(1) { text-align: right; }
td:nth-child(2) { text-align: center; }
td:nth-child(3) { text-align: left; }
td:nth-child(4) { text-align: right; }
td:nth-child(5) { text-align: right; }
```

It can be useful to merge all this information into one set of `<td>` specifications. With this approach, the `<th>` element can be avoided and the heading row can be identified by `<tr class=th>` which is then filled with `<td>` elements instead of `<th>` elements. But they look like `<th>`.

```
tr.th { font-weight: bold; text-align: center; }
td:nth-child(1) { width:  20px; text-align: right; }
td:nth-child(2) { width:  50px; text-align: center; }
td:nth-child(3) { width: 120px; text-align: left; }
td:nth-child(4) { width:  20px; text-align: right; }
td:nth-child(5) { width:  40px; text-align: right; }

<tr class=th><td>head1<td>head2<td>head3...
<tr><td>content1<td>content2<td>content3...
```

16.8 Automatic Row Numbering

Sometimes it is desirable to number the rows of a table even though the HTML does not include numbers. We can add them using CSS.

Content: The **content:** tag can be used in connection with the **::before** and **::after** pseudo-elements. This lets you create "content" that is not true content but is styling by adding words.

Counters: You can invent counters, initialize them, increment them, and display them. Here is an example.

```
table { counter-reset: trNum; }
tr::before { content: counter(trNum) ".";
 counter-increment: trNum; }
```

The first line creates a counter named trNum (for table row number) and resets its value to zero at the start of every table. You can override the initial value. You can name your counter almost anything but avoid reserved words.

The other lines say that for each tr element in the table, before the actual HTML content, some new content should be displayed. This new content consists of the value of the trNum counter, followed by a dot. The trNum counter is also incremented by one. You can override the size of the increment.

Our new content mostly acts like a new `<td>` but not totally.

Position: Your row numbers will naturally be in the top left corner of their cell, and you may want them to line up better with the table content. Relative positioning can adjust where the numbers appear.

Skipping Headings: You may want to avoid numbering the headings and just number the data rows. Here is an example that treats the first row differently than the rest.

```
tr:first-child::before { ... }
tr:not(:first-child)::before { ... }
```

This allows you to increment and display the counter on just those rows that are not the first child.

16.9 Overfull Cells

What should happen when a cell has more content than will fit on one line? Normally that content will be displayed on as many lines as needed, keeping within the width boundaries that are in effect.

Sometimes we want to just display one line of information and hide the rest. A common use case might be showing subject lines of emails. We might want each email to have just one line in the table.

white-space: nowrap can be used to keep the cell contents all on one line. White-space: normal is the default.

Exam Question 304 (p.354)**:** What CSS "attribute: value;" keeps cell contents on one line?
Required Answer: white-space: nowrap;

What if the subject is longer than will fit in the cell? We can clip it and simply not display the remainder that would not fit.

overflow: hidden can be used to hide any content that would not fit in the cell. This is also called clipping. Overflow: visible is the default, and causes the extra content to be displayed beyond the space reserved for the cell.

Exam Question 305 (p.354)**:** What CSS "attribute: value;" clips cell content that does not fit inside the cell?
Required Answer: overflow: hidden;

```
table { white-space: nowrap; overflow: hidden; }
```

overflow: scroll is another option. It adds a horizontal scroll bar to let the user see the hidden content.

We may want to clip the content normally, but display it when the user expresses interest by hovering the mouse over the cell. This can be achieved in at least two ways. One is to use a title= attribute and repeat the content. That seems like a lot of work but it solves the problem. Another way is to use the :hover pseudo-class to restyle the cell, removing the nowrap attribute. This allows the content to use as many lines as needed, but only when the user hovers over the content. If more lines are used, then the page is reflowed, which can be annoying.

```
th:hover, td:hover { white-space: normal; }
```

16.10 For Further Study

http://www.w3.org/wiki/Styling_tables tells how to style tables.

http://www.w3schools.com/cssref/css3_pr_transform.asp explains more about the transform attribute.

Chapter 17

Forms and Inputs

Contents

Simple links will take you from one webpage to another. But web pages can do so much more. Often we want two-way communication, customized communication, with the computers that send us webpages.

Static things, be they webpages, images, or embedded media, are the same every time and for every customer. We store the pages as html files on the server.

Dynamic things, typically webpages, are made to order when they are requested. They are typically different every time and for every customer. Examples include web-based email, Facebook, and any website where you log in.

This customization depends on that two-way communication. Two-way communication uses **CGI** programs. We talk briefly about CGI programs in section 17.7.2 (page 182) below.

Two-way communication is the subject of this chapter.

How do you communicate with the server that is the source of the web page you receive?

The simple answer is "forms."

We use forms to let the user provide information to the server. The browser lets the user fill out the form. Then the browser sends the data from the form to the server. The server responds by sending a new, customized webpage, back to the browser.

17.1 The form Tag

Forms are used to submit information from the user, by way of the browser, to a server somewhere.

`<form>` is the tag to start a form.

`</form>` is the tag to end a form.

Exam Question 306 (p.354)**:** For each <form> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: optional

Without the form tag, you can still make something that looks like a form, but it will not communicate with a server.

The form has several commonly used attributes.

method= can be `post` or `get`.

action= specifies the URL of a program that will process the inputs if the

form is submitted.

17.2 Input Fields In General

Within the form, there can be any number of input fields. In order to be transmitted, each has a name and a value.

There are three major types of input fields: input, textarea, and select. Each type is covered in a section below.

Exam Question 307 (p.354)**:** List the three form input tags.
Required Answer: input, textarea, select

No matter which input type is used, it must have a name or data cannot be sent back from the form.

It can also be designated by **autofocus** to receive the cursor when the page is loaded. If more than one field has autofocus, the browser is free to choose which field to use. You cannot rely on it being the first autofocus. It might be the last or it might vary randomly.

And it can (probably) participate in **taborder**, which tells which field you go to next when you press the tab key.

17.2.1 name=value

When your form is submitted, each input field is either transmitted as part of the form or not transmitted.

We do not transmit the entire form. We do not transmit what you see. Instead we transmit the bare minimum of information. For each field, we transmit its name, which is part of the webpage itself, and we transmit its value, which might be provided by the user.

The only thing that is transmitted is **name=value**.

17.2.2 The name= Parameter

The **name** parameter is used in every field.

The name parameter is not displayed to the user, but they can see it with a “show page source” command.

17.3 The input Tag

input covers most of the cases. It is a void tag. (No end tag is allowed.)

Exam Question 308 (p.354)**:** For each <input> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: forbidden

Within the input tag category, there are many types. They are identified by including a type attribute within the input tag.

`<input type=xxx ...>` is the tag for an input field.

The default type is `text`. That is also the type that is used if what you ask for is not recognized by the browser.

We use ... to indicate that more attributes will be included in the input tag.

17.3.1 The value= Parameter

The **value** parameter is used in every field.

The value parameter may or may not be openly displayed to the user, but they can see it with a "show page source" command.

The value parameter may or may not be easily changed by your user.

Hackers always have the ability to change any parameter.

17.3.2 Button Inputs

Exam Question 309 (p.354)**:** List the four button type= attributes for the input tag.
Required Answer: submit, reset, button, image

submit is the most typical input.

reset does not submit, but instead resets all input fields to their original values.

button gives you more capability to specify what should appear on the face of the button.

image lets you present an image. The exact place that the user clicks on the image will be transmitted (x and y coordinates). The image acts as an

image map.

type=submit

`<input type=submit ...>` is the tag for a submit button.

In some ways, this is the most important input field. Without it, some browsers will not ever allow your form to be submitted.

Exactly one submit button will be transmitted to your **CGI** program.

The name=value from the submit button will be transmitted, along with all the other fields in the form, when you click on the submit button.

JavaScript can be used to intercept the submit button and do something else.

The most important parameters are: name, value.

The **name=** parameter is normal.

The **value=** parameter will be visually displayed on the submit button.

17.3.3 Non-Button Inputs

Exam Question 310 (p.354)**:** List the six old-school non-button type= attributes for the input tag.
Required Answer: text, password, checkbox, radio, file, hidden

Besides these attributes, there are several new (HTML5) type= attributes that we mention below.

type=text

The text field is almost equal to the submit button in importance.

`<input type=text ...>` is the tag for a text field. It provides one line of data entry space for the user to type in something.

If no type is found, the default is text. If a type is found but cannot be understood by the browser, it is presumed to be text.

The most important parameters are **name** and **value**.

The name=value will be transmitted when your form is submitted.

The **name** parameter is normal.

The **value** parameter will be displayed to the user and can be easily changed by them.

Other useful parameters are: size, maxlength, and pattern.

The **size** parameter specifies how big the text field should be. It is the number of characters that should be visible. In my experience, you can usually see more than that. For size=1, you can probably see two or three characters. This may vary by browser.

CSS can be used to make the text field the particular size you want.

The **maxlength** parameter specifies how many characters will be accepted. If you set maxlength=5 and they try typing in 6 characters, only the first 5 will appear.

The **pattern** parameter specifies a regular expression that must be obeyed. See Section 17.7.1 (page 182) for more on regular expressions.

type=password

`<input type=password ...>` is the tag for a password field. It acts almost exactly like a text field, but it obscures the thing typed in, replacing each character with a dot or other symbol. It provides one line of space for the user to type in something.

The most important parameters are: name, size.

The **name** parameter is normal.

The **size** parameter specifies how big the text field should be. It is the number of characters that should be visible. In my experience, you can usually see more than that. For size=1, you can probably see two or three characters. This may vary by browser.

Other useful parameters are: value.

The **value** parameter will be displayed to the user as a series of dots and can be easily changed by them. If your user is a hacker, they can see the value in that field.

type=checkbox

`<input type=checkbox ...>` is the tag for a checkbox. This is generally a square space in which there is either a checkmark or not.

Normally each checkbox has a different name.

The name and value are transmitted to your **CGI** program if and only if the box is checked. Checked? Transmitted. Not checked? Not transmitted.

You can pre-check one or more the spots by using the checked parameter. `checked=checked` or simply `checked` with no equals sign.

Note that `checked=anything` probably works.

Note that `checked=false` does not do what you might think. It acts just the same as `checked=true` because the browser just cares whether the checked attribute is present.

type=radio

`<input type=radio ...>` is the tag for a radio button. This is generally a round space in which there is either a dot or not.

Radio buttons are for selecting among mutually exclusive alternatives, like yes/no, male/female, or ford/honda/toyota/chevy.

Several related radio buttons are grouped together by having the same name. If one is clicked, the dot moves to that space and is removed from any other space it may have been in before.

You can pre-select one of the spots by using the checked parameter. `checked=checked` or simply `checked` with no equals sign.

This is pretty much just like the checkbox option above.

Note that `checked=anything` probably works.

Note that `checked=false` does not do what you might think. It acts just the same as `checked=true` because the browser just cares whether the checked attribute is present.

type=file

`<input type=file ...>` is the tag for uploading a file.

The entire file will be uploaded when the form is submitted.

type=hidden

`<input type=hidden ...>` is the tag for a hidden field. The information in this field is generally not seen or modified by the user. It is put there by the server when the webpage is created, and it is sent back to the server when the form is submitted. Hidden fields and cookies are the main ways that the server can keep track of many conversations (sessions) going on with many browsers all at the same time.

The only useful parameters are: name, value.

The name=value will be transmitted when your form is submitted.

The **name** parameter is normal.

The **value** parameter is not displayed to the user. It is hidden. That's the whole point. But it is transmitted when the form is submitted.

17.3.4 HTML5 Input Types

There are several new HTML5 type= attributes. These may become more useful in the future. They may not work in any special way, but they are always safe to use because when a browser does not understand one of these, it will use type=text instead.

For dates and times: type=datetime, type=datetime-local, type=date, type=month, type=time, and type=week.

For numbers: type=number and type=range.

For colors: type=color.

For special types of data: type=email, type=url, type=tel, and type=search.

17.4 The Textarea Tag

textarea provides a rectangular space where several lines of information can be entered. It has a required end tag.

The **textarea** tag has a name but the value is not specified inside the tag. The value is whatever appears between the **textarea** tag and its ending tag,

or whatever is entered into that space by the user.

Exam Question 311 (p.354)**:** For each <textarea> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

CSS can be used to make the textarea the particular size you want. You can also use the rows= and cols= parameters to get a particular size.

The user can often resize the textarea box by grabbing (mouse-down) the lower right corner and stretching the box or shrinking it.

17.5 The Select and Option Tags

select and **option** work together to create a drop-down list. The select tag has a name attribute but no value. The option tag has a value attribute but no name.

Exam Question 312 (p.354)**:** For each <select> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 313 (p.354)**:** For each <option> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: optional

Here is an example.

```
<select name=abc>
  <option value=123>First Three</option>
  <option value=456>Next Three</option>
  <option value=789>Last Three</option>
</select>
```

17.6 Auto Focus and Tab Index

We may want to assist the user by pre-loading the cursor into a certain input field, and by controlling the order in which the input fields are visited when the user presses the tab key.

Exam Question 314 (p.354)**:** What HTML attribute= is used to start the cursor in a certain input field?

Required Answer: `autofocus=`

The **autofocus=** attribute causes the focus to begin on a certain element. Normally it would be used with an `<input ...>` element. This causes the cursor to rest inside that field when the page is first loaded.

The value of autofocus does not matter, so it can be left off. In that case, the equals sign can also be left off.

When the user pressed the **tab** key, the **tabindex=** attribute tells the browser where to send the cursor next. After the highest-numbered field, tabbing returns to the lowest-numbered field. If tabindex is not specified, the browser normally sends the cursor to the next input field it can find in the HTML.

Exam Question 315 (p.355)**:** What HTML attribute= is used to specify the order in which fields are reached by tabbing?
Required Answer: `tabindex=`

Normally a user provides bits of information as they fill out a form. At the conclusion of providing each bit of information, the user can press the TAB key to move to the next field.

The normal sequence of fields is exactly the same as the order in which the appear in the HTML of the webpage.

Sometimes we want that order to be different.

We can control tab order using the **tabindex** attribute.

```
<input ... tabindex=1 ...>
```

When you press the TAB key, your browser will move to the input field with the next higher tabindex. After it reaches the highest tabindex, it will start over with the lowest one.

If several fields have the same tabindex, they go in the order that they appear in the HTML.

http://www.w3schools.com/tags/att_global_tabindex.asp has more.

17.7 Advanced Concepts

17.7.1 Input Patterns

It is possible to specify a pattern for what the user is allowed to type into an input field.

For example, if you wanted to force the user to type a five-digit number, like for a US zip code, you could say:

```
<input ... pattern=[0-9]{5} ...>
```

The `[0-9]` part means a character in the range of zero through nine.

The `{5}` part means five of them.

For a ten-digit telephone number using the xxx-xxx-xxxx format you could say:

```
<input ... pattern=[0-9]{3}-[0-9]{3}-[0-9]{4} ...>
```

Patterns use something called a "regular expression." The rules for patterns are not difficult to learn, and are very powerful, but we will not cover them here. You can find more on the web. Just do a Google search for "regular expression tutorial" or "html5 pattern tutorial".

Using a pattern does not guarantee that the browser will enforce it, but most modern browsers do.

17.7.2 CGI: The Common Gateway Interface

HTML is the language that browsers understand. It tells them the content of the webpage that they will present to the user.

Static HTML is the same every time it is presented. To change it, someone must change the page that is saved on the server.

But servers can do more than simply serve up the same content every time. Servers can run programs.

CGI, the Common Gateway Interface, is based on that idea. Instead of finding a file to send back to the browser, we find a program to run. That program is called a CGI program. When it runs, it creates the webpage for the user. The server receives the webpage from the CGI program. The server hands it back to the browser. The browser renders it for the user.

In web design, CGI stands for common gateway interface. In other fields, it could stand for computer generated imagery.

The CGI program receives all the name=value pairs provided by the browser. It separates them and considers them to decide what the user wants. Then it builds a webpage, made to order.

CGI can also build images or PDF files on the fly.

17.7.3 Special Encoding

When the browser sends information to the server, it collects together the names and values from the web form. Mostly it sends it exactly as you typed it.

Characters like A-Z and 0-9 are transmitted as written.

Spaces are changed into plus signs before being transmitted.

Special characters, like =, are converted into hex codes before being transmitted. Your **CGI** program on the server side will have to change them back into their original values.

17.7.4 AJAX: Asynchronous JavaScript And XML

Beyond CGI, which refreshes a whole page at a time, we have **AJAX**.

AJAX is is closely related to CGI, but it does not involve the sending of whole pages.

AJAX uses JavaScript to update a small (or large) part of a webpage with new content from the web.

It might add new comments as they are posted to a blog.

It might display current news headlines or weather alerts.

It might cause new email messages to appear in your web mail account.

All this can happen without resending the whole page from the server.

The details of AJAX are well beyond the scope of this book. We mention it so you will know it exists and what word to search on when you want to learn more. It is a commonly-used advanced feature of website design.

17.8 For Further Study

http://www.w3schools.com/tags/att_input_pattern.asp has more on input patterns.

Chapter 18

Responsive Web Design (RWD)

Contents

Ancient websites did not worry about cell phones. (They barely worried about color and images.) Today's websites do. Today's devices include smart phones, phablets (phone tablets), tablets, laptops, game consoles, televisions, print, 3d-glasses, and wrist watches.

Up until this point in this book we have been developing for a laptop or larger format. Today we will swing to the other end of the spectrum.

Exam Question 316 (p.355)**:** What does RWD stand for?
Acceptable Answer: responsive web design

18.1 Try It Out

Before we spend much time talking about it, let's just take it for a test drive. Key in this webpage.

```
<!DOCTYPE html><head lang=en><meta charset=utf-8>
<meta name=viewport content="width=device-width, initial-scale=1">
<title>MQ Test</title>
<style>
body { background-color: orange; }
p:after { content: "default" }
@media ( min-width: 300px ) {
  body { background-color: lime; }
  p:after { content: "min-width: 300px" } }
@media ( min-width: 400px ) {
  body { background-color: aqua; }
  p:after { content: "min-width: 400px" } }
@media ( min-width: 800px ) {
  body { background-color: pink; }
  p:after { content: "min-width: 800px" } }
</style>
</head><body>
<h1>Media Query Test</h1>
<p>Matched Query:
</body>
```

Next, look at the website on a browser and try resizing the screen. You should notice that at certain sizes the background color and :after content should change. Or pull it up on your cell phone and then shift between portrait and landscape. Or use the Chrome browser "inspect element" capability to resize the screen.

The whole point of RWD is that you can change the styling based on the amount of screen space that is available to you.

Obviously you could change the "breakpoints" that were used above by adding more or changing their values.

Try it. Then keep reading.

18.2 Best Practices

Scale: Tell the browser how to scale your content for the viewport. This should go near the top of your HTML. Otherwise the browser will make its best guess for scaling, which could result in a tiny webpage that needs to be zoomed up, or a huge webpage that needs to be zoomed down or scrolled.

Viewport is the technical name for the area of the browser where your website can be displayed. As your viewport gets wider, you may want to take advantage of the extra space to arrange your presentation differently. The media query min-width attribute is very helpful.

```
<meta name=viewport
 content="width=device-width, initial-scale=1">
```

Mobile First: Mobile is the fastest growing segment among web clients. The best-practices popular wisdom is to start all new projects by designing and building them for mobile clients, like smart phones. Then scale them up to larger form factors and make adjustments in your CSS to take advantage of the larger real estate. (Maybe someday we will design for wristwatch first or for eyeglasses first.)

Exam Question 317 (p.355)**:** Explain the "mobile first" philosophy.
Acceptable Answer: first design your website to run well on mobile devices, then modify it for other platforms

Also remember that smaller devices typically have less computing power. Over time this will probably change, but for today it is best on small devices to avoid all the cool stuff like shadows, gradients, and transformations. Use media queries to add them into the CSS for the larger devices.

Exam Question 318 (p.355)**:** Shadows, gradients, and transformations should be limited by media queries. (yes/no)
Required Answer: yes

Relative Units: I personally like to specify things in pixels, which I consider to be a relative unit because it always looks the same size. But for

RWD the common wisdom says that pixels are not relative. Instead, RWD assumes the reliable measure is text size. I should be using things like 50em instead of 400px. That way, if the user zooms in or out, the webpage can respond appropriately.

Image Sizing: Images naturally have certain dimensions that are measured in pixels. If you display a large image on a small device, will it scale down or will it force the user to scroll? Use a max-width: 100% attribute to keep the image from being too wide to fit on the device's screen.

Worst Practice: No Customization

The worst practice is to ignore the variation that is out there. This is a very common practice.

Worst Practice: Separate Website

Historically, one popular approach has been to create separate webpages, or whole websites, devoted to mobile platforms (meaning smart phones), etc. Example.com might be some organization's main website, and m.example.com might be that same organization's website that is optimized for mobile devices. This is regarded as a bad approach.

SEO is important. Having the same content on more than one webpage hurts your **SEO** rankings because it splits your page referrals. Google specifically advises webmasters to present their content only once, and then style it differently depending on the platform.

18.3 Media Queries (MQ)

Through CSS we can specify different styling for different device geometries.

Exam Question 319 (p.355)**:** What does MQ stand for?
Acceptable Answer: media query

```
/* default (tiny screen) styling goes first */
@media screen and (min-width: 400px) { /* small */ }
@media screen and (min-width: 800px) { /* medium */ }
@media screen and (min-width: 2000px) { /* large */ }
@media print { /* paper */ }
```

Exam Question 320 (p.355)**:** List the two main media types that can be specified in media queries.

Required Answer: screen, print

Media types include media=all, media=screen, media=print, media=tv, and apparently even media=3d-glasses. But the two main media types are screen and print.

CSS lets us specify "breakpoints" where we change over to different styling. In the list above, the breakpoints are at 400px, 800px, and 2000px.

Following are some current display sizes, just so you can get a feeling for the variety of what is out there. Be impressed. Then read the section on breakpoints below.

Smart Phone Viewport Sizes: 213x320 320x480 320x533 320x568 320x569 360x640 375x667 384x640 400x640 414x736 424x753 480x854 540x960 720x1280

Tablet Viewport Sizes: 960x600 966x604 1024x600 1024x768 1280x800 1920x1200 2560x1600

Notebook and Laptop Screen Sizes: 1280x800 1280x950 1440x900

CSS provides the capability to specify your styling based on media attributes. A very popular attribute is min-width. This lets you revise your styling if the screen is at least a certain number of pixels wide.

Exam Question 321 (p.355)**:** List the four most important media query viewport attributes.
Required Answer: min-width, max-width, min-height, max-height

You should specify your styling for a tiny device first, and then use successive media queries to handle larger and larger devices. Each larger size can be triggered by using the min-width attribute. That way, if a small device does not understand media queries, it gets the small styles.

Exam Question 322 (p.355)**:** List the two media query orientations.
Required Answer: portrait, landscape

orientation=portrait applies if the viewport is taller than it is wide. orientation=landscape applies if the viewport is wider than it is tall. You can use these to adjust your styling depending on how the device is being held.

18.4 Visual Breakpoints

As you work your way up through a variety of screen sizes, there are points where it makes sense to rearrange your content to take advantage of the

larger screen. Those points are called **visual breakpoints**, or simply **breakpoints**. Sometimes these are described as the points at which your design seems to break because your content does not look its best on the screen any more.

In the past, designers tried to pick breakpoints based on the phones that were most popular, and to design their content to look great on that newest device. This approach is now seen as a mistake. You cannot predict the next popular size. Instead, work with your content.

Your content should dictate your breakpoints. Not any specific device. Start stretching the viewport and see where things stop looking good. That should be a breakpoint.

18.5 A Five-Section Layout

To illustrate the media query breakpoint concept, we will go in depth with a fairly common layout. Imagine a page that consists of five sections: header, footer, content, navigation (nav), and advertisements (ads).

For **SEO** we want the most important things first. We may (or may not) want to have our content come before our navigation and advertisements because things that come higher in the HTML are seen as being more relevant by the search engines. Therefore the order of elements in this HTML webpage will be as follows:

```
header
content
navigation
advertisements
footer
```

We will call the content section C, the navigation section N, and the advertisements section A.

On a wide display, such as a desktop, the core of our webpage will implement an NCA (nav content ad) design with an overall header and footer. The main section will have a 25-50-25 percent split.

The NCA design is based on the idea that our content appears in the middle of the webpage but goes at the start of our html. Our navigation appears to the left of it and our advertising appears to the right.

On a tablet display our webpage will implement a CN-A design. Content and navigation will appear side by side, but the advertising will be deferred until later. Within the CN section there will be a 66-33 split.

On a cell phone display, our webpage will show the divisions one after another, vertically, implement a C-N-A design.

We will use media queries to select the styling that should be in effect. For phones, we will assume the width is less than 360px. For desktops, we will assume the width is 800px or more. The media queries could look something like this.

```
/* default: phone CSS */
@media (min-width: 360px) { /* tablet overrides */ }
@media (min-width: 800px) { /* desktop overrides */ }
```

18.5.1 HTML: Most Important First

For SEO, we want to have our HTML include the most important things first. We will use an html layout something like this.

```
<h1>Headers</h1><p>lorem ipsum ...
<h1>Content</h1><p>lorem ipsum ...
<h1>Navigation</h1><p>lorem ipsum ...
<h1>Advertisements</h1><p>lorem ipsum ...
<h1>Footers</h1><p>lorem ipsum ...
```

We will add divs so we can properly float things later.

```
<div id=hdr><!-- page header -->
<h1>Header</h1><p>lorem ipsum ...
</div><!-- end of hdr -->

<div id=cnt><!-- content -->
<h1>Content</h1><p>lorem ipsum ...
</div><!-- end of cnt -->

<div id=nav><!-- navigation -->
<h1>Nav</h1><p>lorem ipsum ...
```

```
</div><!-- end of nav -->

<div id=adv><!-- advertisements -->
<h1>Ads</h1><p>lorem ipsum ...
</div><!-- end of adv -->

<div id=ftr><!-- page footer -->
<h1>Footer</h1><p>lorem ipsum ...
</div><!-- end of ftr -->
```

18.5.2 C-N-A Narrow Layout

On a smart phone, we want to keep our margins and padding to a minimum, and we want to make each of our sections as wide as the device. We use width: 100% and clear: both to achieve this.

```
* { box-sizing: border-box; margin: 0; padding: 0; }
/* default: phone CSS */
#hdr { width: 100%; clear: both; }
#cnt { width: 100%; clear: both; }
#nav { width: 100%; clear: both; }
#adv { width: 100%; clear: both; }
#ftr { width: 100%; clear: both; }
```

18.5.3 CN-A Tablet Layout

On a tablet, we want the heading to span the whole device, but the content and navigation to share the width of the screen.

```
@media (min-width: 360px) { /* tablet overrides */
  #hdr { width: 100%; clear: both; }
  #cnt { width:  67%; float: left; }
  #nav { width:  33%; float: right; }
  #adv { width: 100%; clear: both; }
  #ftr { width: 100%; clear: both; }
}
```

We have included several of these lines because we are covering all five divs out of an abundance of caution. However, since they just repeat the CSS

that is already in effect, we could slim down the section to just these lines:

```
@media (min-width: 360px) { /* tablet overrides */
  #cnt { width:  67%; float: left; }
  #nav { width:  33%; float: right; }
}
```

18.5.4 NCA Laptop Layout

On a desktop or large laptop, we want the heading to span the whole device, and the content, navigation, and advertisements to share the width of the screen. We also want the content to be in the middle of the screen, with the navigation to the left and the advertisements to the right.

This is a bit more tricky because the navigation appears first on the screen, but the content appears first in the HTML. We can handle this by bundling them together in a new div that we will call CN. The C part will float right in the CN container, and the N part will take the remaining space.

We must go back to the HTML and add a div with id=CN that contains the other two divs, id=cnt and id=nav. Then we can style it as follows:

```
@media (min-width: 800px) { /* desktop overrides */
  #CN  { width:  75%; float: left; }
  #cnt { width:  67%; float: right; }
  #nav { width:  33%; float: left; }
  #adv { width:  25%; float: right; }
}
```

18.5.5 Try it!

Build this model. Then put it in your browser and try resizing the browser window (the viewport). When it goes below 800px does it change to the other layout? When it goes below 360px, does it change again?

Granted this is not the most elegant layout. It's not too bad, but you can probably come up with something better. The point is to show you how it is done, to get you thinking.

18.6 Supporting Markup

To make RWD work well, we need to mark up our content wisely. Then the CSS can do the right thing. Without the supporting markup, we are very limited.

The main markup we will talk about here is div and span.

Many tags, like h1 or p, have specific intended meanings that relate to the structure of the webpage.

The **div** (`<div>`) and **span** (`<span>`) tags, on the other hand, are generic. They do not imply any semantic cohesiveness (although it may exist), but they do group things together. They enclose a portion of the webpage so you can easily apply styling to it.

`<div>` meaning division, encloses a block, a vertical (rectangular) section of a webpage, such as one or more paragraphs. A div cannot legally appear within a paragraph. It can, however, appear within another div.

18.6.1 Divisions

Think of **div** as a super paragraph.

You can use the **div** tag to identify sections of your document for special treatment. Typically you might do this to control styling.

`<div>`s can be nested.

Divs cannot appear within a paragraph, but paragraphs can appear within a div.

Exam Question 323 (p.355)**:** What HTML markup is used to specify a vertical block?
Acceptable Answer: `<div>...</div>`

`<section>` is like div, but is less generic. It does imply semantic cohesiveness: the parts enclosed go together because they are about the same thing.

18.6.2 Spans

You can use the **span** tag to identify phrases of your document for special treatment. Typically you might do this to control styling.

Think of span as a generic form of bold or italic. Often it marks just a word or two within a line or paragraph.

`<span>` generally encloses a horizontal (linear) section of a webpage, such as a few words. A span cannot include a div or a paragraph. It can, however, include another span.

`<span>`s can be nested.

Paragraphs cannot appear within a span, but spans can appear with a paragraph, or within each other.

Exam Question 324 (p.355)**:** What HTML markup is used to mark off a horizontal block of characters?
Acceptable Answer: `<span>...</span>`

Exam Question 325 (p.355)**:** In HTML what does the "span" tag stand for?
Acceptable Answer: span

18.6.3 Nesting

By nesting divs, and varying the margin, border, and padding, you can achieve a variety of stylistic effects.

One can insert divs and spans that are totally empty, and use them as anchors for styling, such as the insertion of graphical elements for beauty or branding. The graphics can be handled entirely within the CSS parts of the webpage once the div or span is placed within the HTML.

Similarly, divs and spans, even empty ones, can be the target of JavaScript actions, replacing them with new content as needed.

Exam Question 326 (p.355)**:** Can a div legally appear within another div?
Required Answer: yes

Exam Question 327 (p.355)**:** Can a div legally appear within a p?
Required Answer: no

Exam Question 328 (p.355)**:** Can a div legally appear within a span?
Required Answer: no

Exam Question 329 (p.355)**:** Can a p legally appear within a div?

Required Answer: yes

Exam Question 330 (p.355)**:** Can a p legally appear within another p?
Required Answer: no

Exam Question 331 (p.355)**:** Can a p legally appear within a span?
Required Answer: no

Exam Question 332 (p.355)**:** Can a span legally appear within a div?
Required Answer: yes

Exam Question 333 (p.356)**:** Can a span legally appear within a p?
Required Answer: yes

Exam Question 334 (p.356)**:** Can a span legally appear within another span?
Required Answer: yes

18.7 For Further Study

Do a Google search on "Responsive Web Design."

https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/ has a helpful five-minute video.

Google's advice (and just about everybody's) is to start by designing for the smallest device, and then expand slowly, looking for "break points" where the content could be laid out better due to the larger space available. This avoids getting caught up in the myriad details of each new device that comes out.

Chapter 19

First Website

Contents

Things are about to get bigger. Until this point we have mainly worried about webpages and smaller things. In this chapter, we "go large" and look at the whole website. We will consider things like domain names, hosting, file structure, and backing up.

19.1 Steps to Setting Up a Website

There are many turn-key solutions to making a website. Run a wizard. Answer a few questions. Fill in some templates. Pretty soon you are up and running. Wordpress is one of the most popular solutions. There is Drupal. There is Wix. There is Weebly. There is Joomla. There is Blogger. There is Squarespace. It is okay to go with these canned solutions. They are fast and easy.

In this chapter, we assume instead that you want complete control. We assume that you want to have things your way, and not the way of some canned solution. We will teach you what you need to know.

To put your webpages online, you will need a domain name and web hosting.

Registrar: Registrars sell you a domain name and provide DNS services. They will also want to be your webhost, which has advantages and disadvantages.

Webhost: Webhosts sell you an IP address, hosting space, and network access. They will also want to be your registrar, which has advantages and disadvantages.

I recommend that you keep them separate. This lets you shop around and change providers later, in case things do not work out perfectly the first time around. (I am on my third webhost.)

19.1.1 Registrar

Domain Name: You will need a domain name, which you get from a registrar. Normally you pay them money. They register your domain name and give you full control over it. We talk about domain names more in section 19.3 (page 201) below.

DNS: They also tell everyone the actual network address where your website is located. This service is called DNS, Domain Name Service. It connects your domain name with your webhosting location.

The network address, also called an IP address, is assigned by your webhosting provider.

19.1.2 Webhosting Provider

Hosting is not the same as registration. Hosting provides storage space and network access for your website.

The two main options are shared hosting and dedicated hosting. Shared is lots cheaper. That is what we will talk about here. Dedicated hosting gives you more responsibility and costs a lot more money.

You need a place for your webpages. The easiest thing is to pay someone for shared hosting space. It may cost around $5 or $10 per month. It gives you a control panel or shell access so you can create and update your content.

Webhosting gives you space in the **cloud** to store your webpages. It also pays for the network traffic that goes to and from your website.

`http://ask-leo.com/how_can_i_find_a_good_web_host.html` is a nice article by a friend of mine that covers important aspects of this question.

`http://www.w3schools.com/hosting/` provides an unbiased discussion and background information.

`http://en.wikipedia.org/wiki/Web_hosting_service` gives a Wikipedia treatment of the subject.

You can do a Google search for "web hosting" to see alternatives that exist.

You can expect to spend about $5 per month (as of 2013), in advance, with a two-year contract, for pretty much unlimited web hosting. You don't always get more by paying more. It pays to compare what different companies offer.

1&1: http://www.1and1.com/

Bluehost: http://www.bluehost.com/

Hostgator: http://www.hostgator.com/shared

Fatcow: http://www.fatcow.com/

For dedicated hosting, the shared hosting providers can sell you an upgrade or you can find a place like Rackspace.

Rackspace: http://www.rackspace.com/

19.1.3 Be Cautious with Your Free Domain Name

Your web host will probably give you a free domain name. Take it. That domain name will be the way you gain access to the control panel. The control panel is how you will manage and add content to your website.

Be aware that the free domain name is one way webhosting providers stop you from shopping around for better deals later. They would love to lock you into a relationship. You should assume that if you ever change webhosting providers, the free domain name will be lost. You will not be able to bring it with you, or only after payment of a substantial extra fee.

Register your really cool domain name in some other way. It is risky to have all your eggs in one basket, in case your webhost goes out of business, or decides to change their terms of service in ways that you find to be abusive.

19.2 Passwords

You will have a lot of passwords. Your registrar will have a password. Your webhost will have a password. If you use a turn-key solution like Wordpress, you will have a password. You probably already have a ton of passwords.

Appendix 24 (page 232) has more discussion about passwords. If you have never thought much about it, you owe it to yourself to do so now. But here we will take a quick look at passwords.

19.2.1 Common Mistakes

There are two common mistakes that people make with their passwords.

First, they pick something easy. Hackers actually have lists of passwords that people have used in the past, and some of them are pretty common. Like the word password itself. Or the digits 123456. You do not want anything that is on one of those lists. (Everything that is short is on those lists.)

Second, they use the same password in a lot of places. This is not so bad if those are not places you care about, but important places deserve their own password.

Many places enforce password policies that steer people away from picking something easy, but not from reusing the same password.

19.2.2 Password Advice

How long should a password be?

Passwords for important things should be 12 characters or longer. This includes email and online banking.

What characters should I use?

Using a variety of characters (letters, digits, punctuation) is helpful, but length is more important. Many password systems require the variety.

How often should I change my password?

You should change your password if you think it has been discovered. Otherwise it is probably safe to leave it alone for several years.

What About Password Managers?

Add-on password managers, like LastPass, are wonderful. Built-in password managers are helpful but generally not as secure.

19.3 Your Domain Name

Every website needs a domain name. In the early days, people could refer to websites by number (IP address), which may work, but which is a bit of a pain, and also makes you look unprofessional. In today's Internet, a domain

name is pretty much mandatory. You will probably want a domain name that is easy to remember. And easy to spell. Short names are also better than long ones.

Dot-com is the most well-known top-level domain, but there are many other top-level domains, including national domains. Within one of those top-level domains, you will rent a name. Then you will advertise it to your friends and neighbors and search engines so people can find your website.

Domain names are normally not actually owned. Instead, technically, they are rented on a year-to-year basis. They are acquired by paying money to a domain registrar. Even so, it is commonly acceptable to say you "own" your domain name. But we will not be confused about that.

Exam Question 335 (p.356)**:** Can you really own a domain name?
Required Answer: no

It is typical to pay around $10 per year for a .com domain name. Or $12. But 10 is a nice, round number that is pretty accurate.

Exam Question 336 (p.356)**:** What is the typical cost for a .com domain name, in USD per year?
Acceptable Answer: 10

After your domain name is registered, you will probably spend a bit of effort establishing it as your "brand," your identity on the web. So it is worth some extra effort at the start of the process to pick something that you will find comfortable for a long time.

Your chosen registrar will tell you whether the domain name is available. If you are like me, the first name you try will be already taken by someone else. But keep looking.

If you are really REALLY serious, you could even pay lots of money, like thousands of USD, to get a "premium" domain name.

http://en.wikipedia.org/wiki/Domain_registration has more details.

You can Google search "domain registration" to find lots of domain registrars.

19.3.1 Top Level Domains

Those endings, .com, .org, .net, and .gov, are called Top Level Domains, abbreviated TLD. Besides those just mentioned, each country has its own,

like .us and .tk. Several industries have their own TLDs.

http://en.wikipedia.org/wiki/Top-level_domain has more.

Exam Question 337 (p.356)**:** What does TLD stand for?
Acceptable Answer: top level domain

eTLD or Public Suffix

Normally people register a domain name directly under a TLD, like .com or .org or .edu or .us. But there are other options.

An example is `.co.uk`, which is the UK version of .com.

This is called an **Effective Top Level Domain** (**eTLD**), or a **Public Suffix**. A public suffix is anything under which you can directly register a domain name.

http://en.wikipedia.org/wiki/Public_Suffix_List has more.

19.3.2 Sub-domains Are Free

With your domain name established, you can create your own sub-domains. For example, I have:

http://doncolton.com - my main domain and personal homepage, for which I pay a yearly rental fee.

http://byuh.doncolton.com - my campus homepage. The subdomain costs me nothing.

http://hangman.doncolton.com - a hangman puzzle solver. The subdomain costs me nothing.

http://quizgen.doncolton.com - a quiz generator. The subdomain costs me nothing.

http://iwdd.doncolton.com - homepage for this book. The subdomain costs me nothing.

The sub-domain name goes in front of your domain name. There is a dot that goes between them.

Sub-domains are typically free (depending on your webhosting provider), and you can name them whatever you want, including something deceptive,

although that is not recommended. Each one is "owned" by the next-shorter domain name, by dropping the front element.

Do not be confused about who owns the subdomain.

Exam Question 338 (p.356)**:** Who or what owns byuh.doncolton.com?
Acceptable Answer: doncolton.com

19.3.3 DNS: Domain to IP Address

The domain name serves two purposes in the actual mechanics of retrieving webpages.

The first purpose is to identify the computer that has the webpage we want. This computer is called a server. It is operated by your webhost. And the only way to reach this computer is by knowing its address.

Addresses on the Internet are called IP addresses. IP stands for Internet Protocol. The most common type of **IP address** is a version-4 address, also called **IPv4**. An IPv4 address consists of four numbers separated by dots. For example, 12.34.56.78 is an IP address. There is a newer type of IP address gaining popularity. It is called **IPv6** (version six).

Your registrar provides the basic Domain Name Service (**DNS**) that converts a domain name into an IP address. Your account with the registrar gives you the ability to specify that IP address.

Once we have the IP address, we can send a request to the correct computer.

The second purpose is to identify the document root of the website within that computer. It is common for one computer to provide storage for dozens or even hundreds of websites. Frequently they all share the same IP address, but they each have their own **document root**.

Your webhost provides a **control panel**, such as **cPanel**, that helps you attach a domain name to a document root. The document root tells which folder (directory) holds your webpages.

19.4 The cPanel Control Panel

Once you have hosting, you need to set up your website and add some content. But how do you set it up? How do you add content?

These days, webhosting providers almost always give you a web-based "front end" or control panel that allows you to create and maintain your website. That way you don't have to be a computer guru.

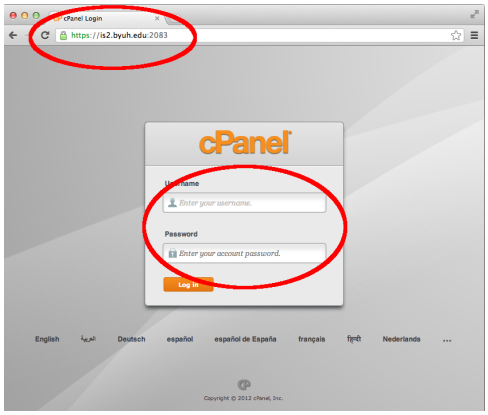
This front end runs the scripts and programs that, in olden days, would have been done by hand by said guru (a systems administrator).

Log into your control panel. You should find an option called File Manager, or something like that. It will let you create files and folders. It will let you upload content, such as photographs.

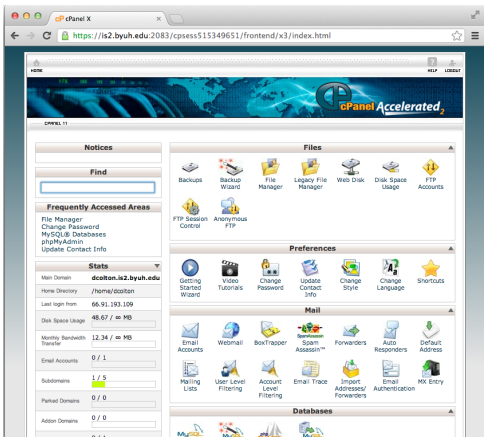
The control panel also lets you do things like AddOn Domains, or update your contact information (email address), or change your password, or create email addresses. (Yes, you can invent your own email addresses, like `guru@doncolton.com`, or whatever you like.)

cPanel is a widely used and full-featured front end. We will use it for illustration here.

The cPanel login screen looks something like this. The circle at the top shows the URL. The circle in the middle shows the place for your username and password.



After logging in, you will see a menu. It may look something like this.



19.4.1 Domain Management

If you only have one domain, and you are using an Apache-based webserver, your **document root** will probably be `public_html`. The document root is the folder (directory) in which the webpages for that website are stored. Those files are called the **document tree**.

If you have addon domains or subdomains, each of those will have its own document root. It is possible for domains to share the same document root. In that case, each webpage might have an alias.

The webhost keeps a list of all the domains it is hosting, and for each domain (or subdomain) it knows the document root.

When a webpage request arrives at the server, the server will break the request into two main parts: the domain name and the path.

Using the domain name, the server will identify the document root.

Using the path, the server will identify the exact file (or CGI program) inside that document tree.

For example, we may have the following:

Domain name: `doncolton.com`

Document root: `/home/doncolton/public_html`

URL: `http://doncolton.com/books/iwdd/iwdd.pdf`

In this case, the user presents the URL and expects to receive a pdf file in response. The browser sends the whole URL to the server.

The server splits the URL as follows:

Domain name: `doncolton.com`

Path: `books/iwdd/iwdd.pdf`.

The server replaces the domain name with the document root, resulting in this actual filename:

`/home/doncolton/public_html/books/iwdd/iwdd.pdf`

Exam Question 339 (p.356)**:** What is the default folder name for webpages on an Apache server?
Required Answer: public_html

Remember: none of this is seen by the end user. It is only seen by the developer (you).

19.4.2 File Manager

To "publish" (add content), you will probably use a file manager. (You might use **rsync** or something based on it, or you might use a packaged solution.)

We will use the file manager.

Rsync is a method for copying between your personal computer and your webhost server. You can read more about it in section 19.7 (page 210).

WordPress and Drupal are very popular packaged solutions.

`http://wordpress.org/` has more on WordPress.

`http://drupal.org/` has more on Drupal.

In using the file manager, you need to understand how your web server organizes things. You should understand the Apache approach.

19.4.3 Add-On Domains

You can attach an add-on domain to your account. This will require that (a) you have registered the add-on domain, (b) you told the registrar what DNS to use, as specified by your webhost, and (c) you decide on a document root for that domain.

Each domain can (and probably should) have its own document root.

19.5 Troubleshooting Your New Domain

You think you did everything right, but you still cannot get to your website. What can you do? Here are some troubleshooting steps that can be followed by yourself or someone helping you.

19.5.1 Caching Causes Time Delay Problems

Caching is an important efficiency "hack" used extensively on the Internet. By "hack" I mean it is not perfect, but it really helps speed things up.

The main idea behind caching is that things do not change very often. The picture you retrieved an hour ago is probably identical to the one you would find if you retrieved it now.

This holds true for domain names and their IP addresses. They rarely change, so the computers that make up the Internet assume that information is valid for a day or two, or maybe a week.

You need to be aware that getting things configured properly suffers from **propagation delay** problems. Propagation is the process of sharing information from its authoritative source out to the ultimate users.

Domain information is cached so any changes you make will not show up immediately. In fact, it is hard to tell when they will show up. You need to be patient.

Specifically, if you have something configured wrong, and you fix it, the rest of the Internet still may be remembering the way it was before. Your fix may take time to propagate.

19.5.2 Is the Outside DNS Correct?

First thing to check: Is the world-wide DNS system reporting the correct IP address for each of your websites?

Do a Google search on "dns lookup" to find websites that will tell you the current dns information without caching.

`http://mxtoolbox.com/DNSLookup.aspx` is one example.

`http://ping.eu/nslookup/` is another example.

`https://www.ultratools.com/tools/dnsLookup` is another example.

The response should include your correct IP address. If it does not, you need to visit your registrar's website.

Normally the problem is with the way you did your configuration.

19.5.3 Is the Web Server Correct?

If the registrar information is correct, log into your webhost and check your domain names.

Each domain name has an assigned document root. Check them all.

The document root should have some content, like an index.html file, with permissions like 644 to allow viewing.

19.6 Apache Web Server

Apache is a very commonly used web server. Apache has about two thirds (2/3) of the market (2011). Its closest competitor is Microsoft's IIS (Internet Information Services or Server).

Because Apache has long been the clear leader, and is so important, many people treat it as though it were **the** way to do things. We provide details for several widely-used things that you can configure.

We already mentioned some important details when we discussed cPanel, like the role of public html and document roots.

19.6.1 Directory Browsing

Many users know that they can take a URL and shorten it by removing the last part of the path. Then they submit it and see what comes back.

This is called directory browsing.

If the final component of the path is a directory, and not a file, the Apache webserver will automatically produce an index, on the fly, unless you tell it not to.

This method can frequently be used to gain access to webpages that have not yet been linked, and are perhaps not yet meant to be visible.

Directory browsing can be handy. You can have a directory full of files, and

when you visit that directory you can see a list of all those files. Normally they are linked for easy access.

19.6.2 index.html

There is a hugely important exception to directory browsing.

Exam Question 340 (p.356)**:** What is the default file name for webpages on an Apache server?
Required Answer: index.html

If one of the files in the directory is named `index`, as in **index.html** or **index.htm** or **index.cgi** or **index.php**, Apache will assume that file will do the work of showing the index. Apache will simply show that file instead of creating an index.

Frequently the index file does not show an "index" at all. Instead, it is the main html file to be shown, or is the cgi script to be run. The other files in that directory are images or style sheets that support the main file.

This is the way we recommend you use it. For each webpage you have, create a directory. Then call the html file **index.html** and include with it the css files and media files that it will use.

There is a priority order that is followed if more than one index file exists. This is specified in the Apache configuration file. Normally **index.html** has precedence. After that we get `index.htm`, and still later **index.cgi**. The one with the highest priority gets displayed. The others get ignored.

There may be other special filenames, like **default.html**, that act like index.html does.

Be aware that `index` and `Index` are not the same. Exact spelling is important. You must use lower-case letters.

19.7 Publishing and Backing Up

Publishing is the act of copying content from your local master copy (offline) to your webhosting server (online).

Backing up is the act of copying content from your webhosting server (online) to a local backup copy (offline).

In case of emergency, you can use an offline copy to replace the online copy.

If you value your website, you should never, ever, find yourself with only one copy of it. You should always have a backup. Many people rotate among several backups, in case something becomes broken and they do not notice it right away.

19.7.1 Upload v Download

We consider our webhost to be part of the Internet Cloud, that great, shapeless mass of things we do not need to understand.

Our websites are part of the cloud.

Our laptop or desktop computer are here on the ground, right in front of us.

When we say **upload**, we mean to take files (content) and copy them up from the ground (offline) to the cloud (online).

Exam Question 341 (p.356)**:** What word describes the copying of files from our personal computer to the server that is hosting our website?
Acceptable Answer: upload

When we say **download**, we mean to take files (content) and copy them down from the cloud (online) to the ground (offline).

Exam Question 342 (p.356)**:** What word describes the copying of files from the server that is hosting our website to our personal computer?
Acceptable Answer: download

19.7.2 Uploading (Publishing)

Normally when we upload, we are publishing something that was developed locally.

Some people recommend that all your development be done locally, and that your online copy is just that, a copy.

But there are times that you might want to do things online first, and then make a backup. In that case, your online version is the original and your local is a backup copy. Having that backup means that you can recreate your website using another hosting provider, or you can reinstate your website if it has been defaced by hackers.

19.7.3 Downloading (Backing Up)

It is good to keep a copy of your website right on your personal computer, in case anything bad happens to your webhost. This is called keeping a backup.

What could happen?

Your webhost could go out of business.

Your website could be hacked by a bad guy, and your content could be deleted or defaced.

It's just a good idea to keep a backup. Once your website begins to have real value to you, make sure you have a copy of it.

Normally when we download, we are making a back-up copy of something that has already been published on the web.

19.7.4 Incremental Backup: Rsync

Rsync is an amazing program that uses an algorithm invented by Andrew Tridgell, an Australian Ph.D. student, around 1999. It has since become a very popular way to back up and publish.

The idea of incremental copying is based on the hope that the source copy and the target copy are already very similar. Therefore, all that remains is to find the differences and make corrections.

http://en.wikipedia.org/wiki/Rsync has more on rsync.

We can run rsync directly from the command line in either Apple OS X or in Unix or Linux.

To download the `public_html` directory from `abc.com`, and to save it on the local machine at `/backups/abc.com`, you could use the following command.

```
rsync -auvz abc.com:public_html /backups/abc.com/
```

19.7.5 Microsoft Windows: DeltaCopy

DeltaCopy is a free program based directly on rsync. It is a "wrapper" around rsync, providing a Windows-based user interface.

http://www.aboutmyip.com/AboutMyXApp/DeltaCopy.jsp tells more about

it and provides a link for free download.

19.8 Advanced Material

We could put this material in an appendix, but it really belongs here. It is, however, more advanced than you probably need at first, but in the long run these are concepts that you should be aware of.

19.8.1 MIME Types v File Extensions

Browsers rely on webservers to tell them what kind of content is being delivered. This is done by sending a MIME Type first, and then the content.

Many servers decide the MIME Type based on the filename extension. If the extension is .jpg, they send a MIME Type indicating the content to be an image of the JPEG type.

When you host an unusual file type, you may need to configure the server so it knows about that file type.

19.8.2 robots.txt and Search Engines

Search engines like Google, Yahoo, and Bing, traverse the web using web crawlers. Web crawlers are programs that pretend to be browsers, but in reality they look at each page they can find and store it in an index for later use.

Search engines are capable of using any webpage they can find, including ones that you might not want them to notice. Mainly they should index content that is stable for the long term. Things that change daily are not really good for indexing.

You can tell these web crawlers what to index and what to ignore by putting that information in a special file, `robots.txt`.

The good news is that most web crawlers respect your wishes as expressed in your robots.txt file.

The bad news is they get to decide whether to respect your wishes or not. There is no guarantee they will pay any attention to your request.

http://www.robotstxt.org/ has more information, or you can Google "robots.txt".

19.8.3 .htaccess

The `.htaccess` file provides you the opportunity to customize things related to your website or individual webpages.

Passwords can be used to restrict access to authorized users.

URLs can be rewritten to change what the user typed into what you wish they had typed.

You can put `Options -Indexes` in your .htaccess file to explicitly turn off index generation.

Custom Error Pages

Add this line to your .htaccess file.

```
ErrorDocument 404 http://whatever/404.html
```

Then create a webpage named 404.html with the message you want to provide.

Some websites automatically specify **404.html** as the default error document.

You can add this line to your .htaccess file to handle type-500 errors (cgi program crashes).

```
ErrorDocument 500 http://whatever/whatever.html
```

http://en.wikipedia.org/wiki/List_of_HTTP_status_codes has a list of HTTP status codes. Some are usable for error documents. But 404 is really the only one that comes up very often.

Password Protection

One popular thing to customize is password protection. This can be used to restrict access to authorized users.

You can password-protect a webpage or group of pages. Tutorials are available online. Google search for "apache htaccess password" for some of them.

Two files are involved. The `.htaccess` file specifies that passwords are being used. Another file of your choice, maybe `.htpasswd`, gives the actual details.

http://www.thesitewizard.com/apache/password-protect-directory.shtml has a very helpful article.

Rewrite

Another popular thing to customize is the rewriting of URLs. For example, the user might type `www.example.com` and you want it to be treated as though they had typed `example.com`, or vice versa. This can be done with rewrite rules.

Rewrite is particularly handy if you decide to rename a section of your website. Visitors using the old URL can be directed to the new location. It also works for renaming a whole website. You might have two domains, `abc.com` and `alwaysbecorrect.com` where the short one is the one you want people to use, and the long one is just another way of saying the name of your business. You can rewrite the long name as though the user had typed in the short name.

Rewrite can also control how users view certain directories. Say you have a directory `public_html/wp` where you have installed WordPress. The URL would be `example.com/wp`. But maybe you are using a subdomain, and you want the URL to be `wp.example.com`. You can use rewrite rules to change the incorrect typing into the correct URL.

Here is an example I am using.

```
RewriteEngine on
RewriteCond %{HTTP_HOST} ^quizgen.doncolton.com$
RewriteRule ^(.*)$ http://quizgen.tk/$1 [R=301,L]
```

The first line tells Apache to turn on the Rewrite Engine.

The second line says "if" the `HTTP_HOST` is exactly quizgen.doncolton.com, keep going.

The third line says if you get here, change the name to quizgen.tk.

This example is simply to give you a quick idea of what is possible. You can Google search for "apache rewrite" to get more detailed tutorials.

Leech Protection

Many websites block outside linking to media on their website, or sometimes even to webpages on their website. Such deep linking is called **leeching** (as in the blood-sucking leech). Using **.htaccess** rewrite rules, we can make it so the picture is only viewable from our own website.

One reason to prevent leeching is the desire to display advertising near their images. Deep linking retrieves just the image and not the advertising.

Another reason to prevent leeching is the desire to reduce bandwidth usage. Say I have a webpage, pretty small, but it has links to large images that are stored elsewhere. My webhost only charges me for the small webpage, and not for the large images. The webhost of the large images gets to pay for the bandwidth each time someone looks at my webpage.

On the other hand, some websites request you to link to their images. It gives them a way to measure how often the images are retrieved. It goes in their log files. They want that.

We will show you how to prevent leeching.

In this example, we check the referrer. If it is doncolton.com, we allow the request to continue. Otherwise, we apply the rewrite rule and replace the URL with noleech.jpg. Presumably noleech.jpg would be an image telling the user that the requested image is not available.

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} !^http://.*doncolton.com/.*$ [NC]
RewriteRule .*[.](jpg|png)$ http://doncolton.com/noleech.jpg [R,NC]
```

This stops all requests for jpg and png files that are part of the doncolton.com domain, unless the referrer is also in the doncolton.com domain.

Unit II

Appendix

Chapter 20

Glossary of Terms

We present here an alphabetical listing of some of the vocabulary words and acronyms used throughout this book.

AJAX: Asynchronous JavaScript And XML. It is a method for updating a small part of a webpage instead of replacing the whole thing. See section 17.7.4 (page 183).

bandwidth: The amount of network traffic being used. High-resolution photographs use more bandwidth than low-resolution photographs. HDTV uses more than standard TV.

browser: a computer application that displays your webpage to the user.

cache: something kept locally so that it does not need to be retransmitted each time it is used. Static elements can be cached. Normally this includes images, but sometimes it includes whole web pages.

case-insensitive: A situation where the difference between capital letters and lowercase letters does not make any difference. Many aspects of HTML are case-insensitive.

case-sensitive: A situation where the difference between capital letters and lowercase letters makes a big difference.

CGI: Common Gateway Interface. It is a method for creating a webpage dynamically, so that it might be different every time and for every user. See section 17.7.2 (page 182).

cursive: One of the five generic font families. Its main feature is that adjacent letters try to connect with each other, resulting in a flowing style

of text. Cursive is not good for body text or small text because it is hard to read. Cursive can be nice for headings.

deprecated: something that was once approved and commonly used, but is now viewed as the wrong way to do things. It has been replaced by a better way, but the old way is still allowed for now. Someday the old way may be removed. It is being phased out.

dynamic: something that changes automatically, and therefore can be different every time we use it.

fantasy: One of the five generic font families. It captures everything that did not fall into the other families: serif, sans-serif, monospace, and cursive. Fantasy is usually not good for body text or small text because it is hard to read. Fantasy can be nice for major headings.

homepage: The root or main webpage of a domain, but sometimes another special page. It is incorrect to refer to all webpages as homepages. Can be written as "homepage" or "home page" (with a space).

JavaScript: The scripting language most commonly used with webpages. It provides the ability to modify pages without involving a server or the use of bandwidth.

local: right at your computer instead of somewhere out on the Internet. It means the same thing as "offline."

lowercase: Non-capital letters. Letters like "ABC" are called uppercase. Letters like "abc" are called lowercase. They are also called small letters.

monospace: One of the five generic font families. It means uniformly-spaced. Narrow letters like i and j take the same amount of space as wide letters like m and w. Monospace is commonly used for code examples.

offline: right at your computer instead of somewhere out on the Internet. It means the same thing as "local." Sometimes it is written with a hyphen: off-line.

online: somewhere out on the Internet, not local. Sometimes it is written with a hyphen: on-line.

raster fonts: Ultimately, fonts are rendered as pixels on the user's screen or on a printed page. Raster fonts were used originally and are defined in terms of these pixels. Basically, all screens are raster. Even the retinas of our eyes are raster. If you magnify a raster font, the curved edges eventually become jagged.

render: the process used by the browser to convert HTML into a usable image. Also the process used by the browser to convert vector fonts into raster images displayed on a webpage.

sans-serif: One of the five generic font families. It literally means "without serifs". Sans-serif fonts have plain ends on their strokes. Sans fonts are commonly used for headings in web pages and printed pages.

serif: The most popular of the five generic font families. Its characters have spikes, hooks, or barbs at the ends of the strokes. The spikes were originally used to keep the letters from cracking when used in a printing press. Serif fonts are commonly used for body text in web pages and printed pages.

server: the computer that is hosting your webpage. It receives requests from the browser and sends webpages for the browser to display. Also called a webserver.

static: something that does not change automatically, and therefore is normally the same every time we use it.

uppercase: Capital letters. Letters like "ABC" are called uppercase. Letters like "abc" are called lowercase. They are also called big letters.

user: the human that is looking at your webpage.

vector fonts: To avoid the jagged edges of raster fonts, it is desirable to define characters as mathematical curves instead of a raster of pixels. Vector fonts are defined in terms of the curves that describe each character, and those same curves can generate raster images in a variety of sizes. Vector fonts are widely used now. If you magnify a vector font, the edges always remain smooth.

webpage: A page designed for use on the web, whether or not it is actually on the web. Can be written as "webpage" or "web page" (with a space).

webserver: A computer that makes webpages available to browsers. Can be written as "webserver" or "web server" (with a space).

Chapter 21

Working Offline

Contents

Webpages should be on the web, right? That means being hosted "on line" and having a domain name.

We are going to get to that shortly.

In this chapter we show you how to build a webpage locally, right on the computer you are using to read this PDF file.

The word **local** means right at your computer, instead of somewhere out on the Internet. The word **offline** means the same thing. Sometimes it is written off-line with a hyphen.

Exam Question 343 (p.356)**:** What word means right at your computer instead of somewhere out on the Internet?

Acceptable Answer: local or offline

The word **online** means somewhere out on the Internet, not local. Sometimes it is written on-line with a hyphen.

Exam Question 344 (p.357)**:** What word means somewhere out on the Internet instead of right at your computer?
Acceptable Answer: online

We will start by building a simple webpage, offline.

21.1 More Vocabulary

We have defined online and offline. We need a few more words.

By **user** we mean the human that is looking at your webpage.

By **browser** we mean the computer application that displays your webpage to the user.

By **server** we mean the computer that is hosting your web page. It receives requests from the browser and sends webpages for the browser to display.

By **render** we mean the process used by the browser to convert HTML into a usable image.

These and other terms are defined in the glossary in appendix 20 (page 218) at the end of this book, right before the index.

21.2 You Have A Browser

We assume you have a browser. The "big five browsers" are Microsoft's **Internet Explorer** (**IE**), Mozilla's **Firefox** (FF), Google's **Chrome**, Apple's **Safari**, and Opera Software's **Opera**. Each is free. Just download it, install it, and you can run it.

For more on browsers, including links to download them, please see Appendix 22 (page 228).

For this course, we recommend that you use Chrome, Firefox, Opera, or Safari for development and testing of your webpages.

Internet Explorer has a bad track record for being far less compliant with official Internet standards than the other four browsers. However, recent

versions of Internet Explorer have improved greatly, and Microsoft is creating a whole new browser that will replace Internet Explorer.

For serious testing of your webpages, you should view them in all five of the major browsers.

21.3 An Empty Folder

On your desktop, create an empty folder. Name it "iwdd" (which stands for Introduction to Website Design and Development, the name of this book).

(You can actually put the folder somewhere else, assuming you know how to get to it. And you can call it something else.)

Inside that folder we are going to create a webpage.

21.4 Text Editor

We will be working directly with webpage code. You will benefit greatly from using a text editor.

Text editors are different from word processors. Word processors are designed to make documents. Text editors only edit text. They do not handle margins and fonts and bold and italic and underline.

Text editors just handle simple characters. But they also do very helpful things like syntax highlighting and checking.

For all platforms (Microsoft, Apple OS X, and Linux), **Komodo Edit** (free) is our current recommendation.

For Microsoft, **Notepad++** (free) is our alternate recommendation.

For a more detailed look with additional options, including links for downloading, please see Appendix 23 (page 230).

21.5 Text Plain

Create a file named `index.html`.

Use your text editor to create it and type it in. Here it is.

```
This is Don's webpage.
This is more of Don's webpage.
```

Feel free to type whatever words you want, but make two lines.

Now visit it. Use "open with" and select your browser.

When you arrive at your webpage, the words you wrote are presented.

Also notice the "navigation bar" where the browser tells you the URL of the webpage you are viewing. Because you are local it will say something ...

like this: `file:///xx/xx/xx/iwdd/index.html`

or this: `file:///C:/Users/.../Desktop/iwdd/index.html`

If you were online, it would look more like this:

`http://xx/xx/xx/iwdd/index.html`

21.6 Simple Markup

In HTML mode, we can start adding hyper-text markup.

Markup looks like `<something>` where the `<` (open bracket, or "less than") identifies the start of the markup, and the `>` (close bracket, or "greater than") identifies the end of the markup, and the things in between specify what the markup is supposed to do.

Edit your file. Add some markup. Then save your file. Follow this example, but feel free to use your own words.

```
<h1>This is Don's webpage.</h1>
This is normal text.
<b>This is bold text.</b>
This is normal text.
<i>This is italic text.</i>
This is normal text.
```

That's an h-one on the first line, not an h-el.

21.7 Reloading Your Webpage

On your browser find the reload button. For Chrome, Firefox, Opera, and Safari, they (currently) look like this.

After saving your changes, reload your webpage.

Click on the "circle arrow." It will instruct your browser to get the webpage content from whatever source it used before.

When you reload your page, you should see a heading, some normal text, some **bold text**, some normal text, some *italic text*, and some normal text.

Instead of seeing six lines, you probably only see two lines, one heading, and one with all the rest. HTML ignores your line breaks. If you want line breaks, you have to ask for them explicitly. The markup for line break is `<br>`.

The `<>` things are called **tag**s. We write `<tag>` for the **start tag**, and `</tag>` for the **end tag**.

The tags in this example are h1, b, and i.

More about Reloading

Normally when you view a page, it will show the most recent copy it remembers. The recent copy is stored in the browser's **cache** to avoid using your Internet connection. (Downloading can be slow, and might use up your data cap for the month.)

The browser's cache is stored right on your local computer. Usually it is in a hidden part of your file system. It is hidden to prevent you from messing with it directly.

When you do a reload, the browser ignores its cache and goes back to the original source.

If the original source has changed, you may need to do a reload so you can see those changes.

Exam Question 345 (p.357)**:** Where does your browser store files it might need later?
Required Answer: cache

Exam Question 346 (p.357)**:** If you change a webpage, but when you visit

it, it has not changed, what is probably the cause?
Acceptable Answer: cache or caching

Exam Question 347 (p.357)**:** If you change a webpage, but when you visit it, it has not changed, what should you do?
Acceptable Answer: reload or refresh

21.8 Adding a Photo

Place a photograph into the same folder. We will assume you called it `myphoto.jpg` for this example. If you called it something else, hopefully you can figure out what to do.

Modify your webpage by adding a line at the end, as follows:

```
<h1>This is Don's webpage.</h1>
This is normal text.
<b>This is bold text.</b>
This is normal text.
<i>This is italic text.</i>
This is normal text.
<img src=myphoto.jpg>
```

Save it and view your page again. The picture should now appear after the text of your webpage.

The source can be any URL that resolves to a picture. However, some websites intentionally block outside linking to their media. They consider it to be **leeching**. See section 19.8.3 (page 216) for details.

Depending on where your photo came from, it might be normally sized or it might be really big. We can fix that. Add a width parameter to your img line as follows:

```
<img src=myphoto.jpg width=500>
```

The width parameter specifies the display width of your picture in pixels. It says that no matter what your picture's natural size is, make it look like it is 500 pixels wide. (There is also a height parameter.)

21.9 Start and End Tags

The h1, b, and i tags have separate start and end tags.

The img markup does not have a separate **end tag**. It has only one tag. Tags like this are called **void tags**.

When there is no end tag, we may choose to indicate our awareness of that fact by ending with a `/` right before the closing bracket, like this:

```
<tag .... />
```

The two times this most often comes up are `<img ... />` and `<br />`.

21.10 Congratulations

Congratulations. With any luck, you have just created a webpage. Maybe it is your first webpage ever. Congratulations.

And we are just getting started!

Chapter 22

Browser Recommendations

The "big five browsers" are Microsoft's Internet Explorer (IE), Mozilla's Firefox (FF), Google's Chrome (Chr), Apple's Safari (Saf), and Opera Software's Opera (Op). Each is free. Just download it, install it, and you can run it.

Historically, Internet Explorer has had the largest market share, by which we mean it is installed on more computers and used by more people than any other browser. This has changed. It is still widely used, but not dominant.

Web designers have hated Internet Explorer because it does not support the official standards for webpages. It has been a nightmare. (I am not exaggerating.) But the good news is that Microsoft seems to be making real progress toward being "standards compliant."

Being standards compliant is a bit of a moving target. Probably none of the major browsers (or minor ones, for that matter) are fully compliant. But Chrome, Firefox, Opera, and Safari tend to be very close, and much closer that Internet Explorer (historically).

Microsoft's Internet Explorer comes pre-installed with Windows, and is not available for Apple OS X or Linux.

Chrome: https://www.google.com/intl/en/chrome/browser/ is the homepage for Google's Chrome Browser. It is free.

Firefox: http://www.mozilla.org/en-US/firefox/ is the homepage for Mozilla Firefox. It is free.

Opera: http://www.opera.com/download/ is the download page for Opera.

It is free.

Safari: http://www.apple.com/safari/ is the homepage for Apple Safari. It is free. It is also the dominant browser choice for Apple products such as the iPad, the iPhone, and the iPod touch.

Chapter 23

Text Editor Recommendations

Contents

For several of the classes I teach, you need to create text files. This includes programming files, HTML files, CSS files, and configuration files.

To edit such files, we expect you to use a text editor, not a WYSIWYG editor.

Text editors are different from word processors. Word processors are designed to make documents. Text editors only edit text. They do not handle margins and fonts and bold and italic and underline.

Text editors just handle simple characters. But they also do very helpful things like syntax highlighting and checking.

23.1 All Around

Komodo Edit has been strongly recommended to me, and from my readings it is well regarded by many people. Komodo Edit is free and Open-Source

software.

Komodo has a huge advantage in platform freedom compared to other editors I have used. You can use it on Microsoft Windows, on Apple OS X, and on Linux.

http://komodoide.com/komodo-edit/ is the official place to download the latest version of Komodo Edit.

I don't have a lot of experience with Komodo yet.

23.2 Microsoft Windows

If you are using a Microsoft Windows operating system, another good alternative is the text editor named **Notepad++** which you can find free on the Internet. I have recommended it in the past.

http://notepad-plus-plus.org/ is the homepage for Notepad Plus Plus. It is free. It runs on Microsoft Windows.

23.3 Just Use A Word Processor

You can actually use a word processor, but if you do, be careful to save your work as plain text. Something like Libre Office (Open Office) should work fine. It's not as bad as I make it sound.

http://www.libreoffice.org/ is the homepage for Libre Office.

23.4 What Do I Use?

For many years I have used EMACS, and it is still my go-to editor. It works well for the kinds of things that I do, and I have memorized many of the keyboard shortcuts (cursor movement, search, cut and paste, etc.). I see no reason to change. But if I were starting over, I would definitely consider other choices.

Chapter 24

Password Recommendations

Contents

You would be amazed at how many lame passwords are in use. Sometimes it does not matter, but sometimes it leads to serious heartache.

Don't become a victim.

You should create a good password, or at least know how.

Anyone who knows your password can steal your identity and take over your website. Maybe you don't care. Maybe you will later.

One of the first things you will want to do after logging in for the first time is change your password. At first, your website will not be very valuable, and the password you use will not matter much. As your website gains value, you should make sure your password is appropriately difficult to guess.

Current wisdom on passwords is simple: make them long. Long means 12 to 16 characters.

Old-time wisdom harkens back to the day when passwords were only allowed to be 8 characters or less. To make passwords difficult to guess, it was

recommended that they include a mix of UPPER- and lower-case letters, plus digits, plus special characters, and that they be changed frequently.

These days, passwords are almost never stored in plain text by your webhost. Almost. Be careful who you trust.

http://xkcd.com/792/ has a cute comic about this.

Instead, they store a **hash** of your password. The hash is also called **cypher text**. The hash is created by mixing up your password in a complicated but repeatable way. The mixing is so thorough that it cannot be undone. When you log in, they take the password you just entered, mix in the same way, and check to see if the result matches the hash that was stored. If so, you are granted permission to manage your website.

Hackers sometimes steal copies of these hash tables. Then they share them. Once they have your hash, since the mixing cannot be undone, they try lots of different passwords to see if they can find one that hashes to the same value. For a short password, this takes **very little time.**

24.1 Online Password Cracking

Online means across the Internet. All work is done remotely and delays are common.

This is the most difficult path to password cracking. Each attempt must pass across the network and be processed by the webhost. Delays make this take a long time. It is generally not feasible to use brute-force guessing in an online setting.

Instead, dictionary attacks are used, based on information about you, maybe learned from your Facebook account. Who are your best friends? Your pets? Your dates (birthday, anniversary)? Your phone numbers? Your favorite entertainers?

24.2 Offline Brute Force Guessing

Offline means without using the Internet. All work can be done on a local computer without any delays.

When they are just guessing, they start with one-letter “words” (including

single-digit numbers). Then they move on to two-letter words, and so on. This is called brute force.

http://en.wikipedia.org/wiki/Password_cracking mentions (in 2012) that common desktop computers can try over 100 million passwords per second. Every year that number goes up. **Moore's Law** says that, on average, computing speed doubles every 18 months, so in 2015 the speed will probably be 400 million passwords per second.

Exam Question 348 (p.357)**:** What is Moore's Law?
Acceptable Answer: Computing speed doubles every 18 months.

Of course, this assumes the hacker has your hash, and can do the cracking offline. If not, each guess takes much longer.

Let's assume we have a computer capable of 100 million guesses per second.

Using lower-case letters, we have 26 choices per character.

lower-case letters	time to crack
1	260 billionths of a second
2	6.7 millionths of a second
3	175 millionths of a second
4	4.5 thousandths of a second
5	118 thousandths of a second
6	3 seconds
7	80 seconds
8	35 minutes
9	15 hours
10	16 days
11	1.1 years
12	30 years

And remember that every 18 months, those times are cut in half due to Moore's Law.

What if we use a bigger variety of characters in our password? It really helps.

Using letters (26 lower, 26 upper), digits (10), and special characters (maybe around 30), we have about 100 choices per character.

totally-random characters	time to crack
1	1 millionth of a second
2	1 / 10,000 of a second
3	1 / 100 of a second
4	1 second
5	1.6 minutes
6	2.7 hours
7	3.7 months
8	30 years

And that is just with a single desktop computer. Imagine if they had a bot-net of zombie computers all working together. Of course, you and I are not worth the effort, but cracking an administrator password to a major website could be.

http://en.wikipedia.org/wiki/Botnet shows that in 2009 there were millions of computers in some bot nets.

totally-random characters	botnet time to crack
8	14 minutes
9	1 day
10	3 years
11	300 years

How many characters do you want in your password? 12 is really considered to be a minimum for anything you really want to protect.

24.3 Common Passwords

Over time hackers have developed lists of **common passwords**. Hackers will try these first before going to brute force. This is called a **dictionary attack**. Type "common passwords" into a web search engine for an eye-opening experience.

The dictionary attack is probably the best approach for a hacker that does not have your hash, since there are many fewer words in the dictionary than there are random letter combinations.

http://blog.eset.com/2012/06/07/passwords-and-pins-the-worst-choices lists these passwords as its top ten: password, 123456, 12345678, 1234, qwerty, 12345, dragon, pussy, baseball, football.

http://mashable.com/2011/11/17/worst-internet-passwords/ lists these

passwords as its top ten: password, 123456, 12345678, qwerty, abc123, monkey, 1234567, letmein, trustno1, dragon.

24.4 Account Chaining

If a hacker discovers your password on site xyz, they can try the same username and password on other sites, like email or banking (PayPal) or shopping (iTunes, Amazon) or social (Facebook, LinkedIn) or gaming (Sony, Blizzard).

It is good to vary your passwords, at least for accounts that you consider to be valuable.

If anyone gets your email password, you are in a world of hurt. Normally they can change **any** of your passwords because they may all be linked to your same email address.

24.5 How Often To Change Your Password?

The old-time wisdom says you should change your password often. You want to change it faster than your enemy can guess it.

In the days of eight-character passwords, it makes some sense. Not much, but some.

The big problem with frequent changes is memorization. Who can memorize a new password and remember it reliably? When we are forced to change our password often, one of several solutions typically emerges.

(a) The password gets written down. It's on the yellow sticky-note under the desk phone, or on the wall.

(b) The password is the same as before, but just part of it changed. Maybe it is "alohaFeb2000" in February of 2000, and in March, it will be changed to ...

If you have a good, secure password, there is no need to change it, ever. By good and secure, we typically mean long, like 12 to 16 characters, or maybe more, and hard to guess.

But if you ever think that it has been revealed, compromised, leaked, or broken, then you should change it, everywhere it is used.

24.6 Password Management Tools

There are nice web-based tools, free and paid, that make it easy for you to have a different password for every website, and make your passwords long.

I recommend **lastpass**. It is a free password management tool. It facilitates having a different password for each website you join, and sharing those passwords among several computers that you might commonly use. It also fills in the blanks for you, reducing errors due to keying something in wrongly.

http://lastpass.com/ provides free downloads.

You can google "password management tools" for other other options.

Chapter 25

Copyright and Intellectual Property

Contents

Webpages and websites exist in an even broader context of intellectual property. It is important to know what rules are currently used by society so that we do not get into trouble.

25.1 Overview

I am not a lawyer, so I am not authorized to give legal advice. Therefore, this is not legal advice. But listen anyway. It's on the test.

Copyright can be our friend, or it can get us into trouble.

This is the general rule. (There are exceptions.)

- Do not publish pictures that are owned by someone else. This especially includes pictures we have copied from another website.
- Do not publish words that were written by someone else. This especially includes words we have taken from another website.
- Do not link to illegal content, such as hacked games or pirated movies.

The penalty can be monetary damages and permanent loss of our website.

To be safe, everything we post on the web should be our own work.

Obviously that is not always practical or possible. This section should help us know what we can legally do, and what trouble we can get into.

Generally, copyright is the right to prevent people from making copies of your work. Copyright is generally automatic. If someone copies your work, you can force them to stop doing it. If you have formally filed your copyright, you can also sue for damages.

Copyright applies to words, images, audio recordings, and motion pictures. And it applies to other things, but those four things are the ones we most often use on webpages.

25.1.1 Terminology

Ideas cannot be protected by copyright. They may be partially protected by patent.

Exam Question 349 (p.357)**:** Can ideas be copyrighted?
Required Answer: no

Expression is protected by copyright. Copyright covers the expression of ideas, and not the ideas themselves. You must be careful in using words written by others, but you are free to rewrite, to express even the very same ideas, using your own words.

Plagiarism is related to copyright. Plagiarism is when you present the intellectual work of other people as though it were your own. This may happen by cut-and-paste from a website, or by group work on homework.

In some cases, plagiarism may also create a violation of copyright law. If you borrow wording from someone else, you should identify the source.

Lorem Ipsum looks like Latin. It is generic text that is used for filler on webpages when the actual content is not yet available. See section 25.2.1 (page 243).

Fair use is the concept (in the USA, with similar concepts elsewhere) that even though something is copyrighted, there are times when copies can still be legally made. Personal (non-commercial) use may qualify. News reporting may qualify. Scholarly use may qualify. Handicapped accessible use may qualify.

Exam Question 350 (p.357)**:** What is Fair Use?
Acceptable Answer: It is the concept in US law that sometimes things can be copied legally without permission even if they are copyrighted.

A **Cover Version** is a re-recording of a song that has already been released by someone else. It is allowed by copyright law.

Exam Question 351 (p.357)**:** What is a Cover Version?
Acceptable Answer: It is re-recording of a song that has already been released by someone else.

Public Domain is the category of things that are not under copyright restrictions any more. It applies to everything when it gets old enough. It can apply to newer things if the author agrees. Section 25.2 (page 242) has more information.

Exam Question 352 (p.357)**:** What is a Public Domain?
Acceptable Answer: It is things that are not under copyright restriction.

Creative Commons is another place where content, especially images, can be found under liberal terms. See Appendix 25.2.3 (page 244) for more information.

25.1.2 Derivative Works

Derivative works are often subject to the copyright of the original work. That means you are normally not allowed to publish derivative works.

When you take major elements of someone else's copyrighted work and use them, the result is a derivative work. (Ideas are not elements, in this con-

text.)

Exam Question 353 (p.357)**:** What is a Derivative Work?
Acceptable Answer: It is taking major elements of someone else's copyrighted work and using them in a new work.

For example, cropping a picture, or recoloring a picture, or inserting a new face into a picture, would all be examples of creating a derivative work.

If you rewrite part of another author's text, you have created a derivative work.

If you read their text, figure out the main idea, and then rewrite it totally in your own words, you are safe. It is not a derivative work.

25.1.3 It Is Easy To Catch Violations

With search engines like Google, you know that computers are constantly "crawling" the World-Wide Web looking at content including text and media.

Duplicated content is quickly noticed. Content owners can subscribe to services that watch for unexpected copies of their work.

Content owners can (and do) use services that automatically send take-down notices.

The bottom line here is that if you copy someone's webpage, you might have your whole website vanish because of a take-down request. See section 25.1.4 (page 241) for more.

Or you might just get a nice email message containing a thinly veiled threat of a take-down notice, but giving you time to fix it yourself first.

25.1.4 DMCA: Digital Millennium Copyright Act

In the USA, the Digital Millennium Copyright Act (DMCA) contains an important provision called the Safe Harbor. This is to protect web content providers. If a copyright holder claims that a website violates their copyright, they can file a take-down notice with the provider.

If the provider takes down the material that is claimed to violate copyright, they are safe from legal threats. If not, they may face legal action.

The person posting the material can respond that the take-down notice is wrong, and possibly get the material reinstated.

Webhost providers that follow the take-down rules are granted immunity from copyright violations that occur on their websites.

Exam Question 354 (p.358)**:** What is a Take-Down notice?
Acceptable Answer: It is formal request to a web provider to remove (take down) content claimed to be protected by copyright.

Exam Question 355 (p.358)**:** What is a Safe Harbor?
Acceptable Answer: It is a set of rules that when followed protect you from legal liability.

Exam Question 356 (p.358)**:** What does DMCA stand for?
Acceptable Answer: digital millennium copyright act

25.1.5 Copyright Summary

The safest course is to create your own content, or have it created for you. If you get content from any other source, make sure you have the right to use it.

Text: If you need text, write it yourself. Or rewrite it in your own words. If you just need filler, use **lorem ipsum** (see section 25.2.1, page 243).

Images: If you need an image, take it yourself or find it on Creative Commons. Simply using images you find on Google is **not** safe enough. You need to verify that the image can be used or you risk being shut down.

Audio: If you need sound, create and record it yourself.

25.2 Public Domain Content

When creating a webpage, you might want to work with some content that you did not have to write.

Lorem Ipsum is a good fallback for unreadable content that is still easy on the eyes. You can generate some random words in Latin and use it as your sample content.

Project Gutenberg is a good fallback for readable (and interesting) content that is in the public domain. You can download a book and use it as your

sample content.

Creative Commons is a good fallback for images.

25.2.1 Lorem Ipsum

When you are trying to create a webpage layout, you often need filler text (fake content) to show how things will generally look.

You could cut and paste something from another website, but that can result in copyright infringement problems. Not the best plan.

You could type in blah blah blah blah blah blah until you have filled the space you want.

You could type in yadda yadda yadda yadda yadda yadda yadda until you have filled the space you want.

These things work but look unnatural. Typographers typically use some fake Latin-looking text that is called **lorem ipsum**.

How about this next paragraph? Looks pretty real. It is almost totally fake.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed et urna ut sem malesuada varius. Nullam sit amet risus elit. Aenean dapibus, nisl ut vestibulum euismod, sem metus rutrum est, at auctor turpis sem ac dolor. Integer interdum, turpis eget volutpat venenatis, lectus diam commodo neque, cursus suscipit orci neque placerat turpis. Curabitur nisi velit, malesuada vitae porta sit amet, consectetur ut magna. Nulla facilisi.

`http://www.lipsum.com/` is a source for random lorem ipsum text that you can use as filler for your own web designs.

Use any web search engine and look for "lorem ipsum" to find other good filler text generators that are available.

Exam Question 357 (p.358)**:** What is lorem ipsum?
Acceptable Answer: filler text

Exam Question 358 (p.358)**:** What fancy name do we use to describe fake text that is inserted into a webpage just to give it some body?
Acceptable Answer: lorem ipsum

25.2.2 Project Gutenberg

Project Gutenberg has collected thousands of writings (mainly older books) that are in the public domain (at least in the USA) because their USA copyrights have expired.

To give you a sense of what is available, here is a list of the top ten books downloaded from Project Gutenberg on 2012-12-23, with the one-day download count in each case.

1. A Christmas Carol by Charles Dickens (1228)
2. Les Misérables by Victor Hugo (852)
3. Grimm's Fairy Tales by Jacob Grimm and Wilhelm Grimm (677)
4. The Kama Sutra of Vatsyayana by Vatsyayana (583)
5. Darkey Ways in Dixie by Margaret A. Richard (553)
6. Twas the Night before Christmas by Clement Clarke Moore (542)
7. Adventures of Huckleberry Finn by Mark Twain (514)
8. How to Analyze People on Sight by Elsie Lincoln Benedict and Ralph Paine Benedict (511)
9. Eighteenth Brumaire of Louis Bonaparte by Karl Marx (498)
10. Pride and Prejudice by Jane Austen (494)

25.2.3 Creative Commons

Creative Commons is a concept, and also an organization. It is a place where content can be found under liberal terms. It is a good source for things like photographs that you cannot easily make yourself, for example, famous people or landmarks, but that someone else has made and donated to public use.

Works that are available at Creative Commons may be restricted in minor ways. For example, they may require you to identify the original author.

25.3 For Further Study

http://en.wikipedia.org/wiki/Copyright has more about copyright.

http://en.wikipedia.org/wiki/Plagiarism has more on plagiarism.

http://en.wikipedia.org/wiki/Cover_version has more about cover versions of music.

http://en.wikipedia.org/wiki/Derivative_work has more about derivative works.

http://en.wikipedia.org/wiki/Dmca has more about the DMCA.

http://en.wikipedia.org/wiki/Lorem_ipsum has more about lorem ipsum.

http://www.gutenberg.org/ is the homepage for Project Gutenberg.

http://en.wikipedia.org/wiki/Creative_commons has more about the Creative Commons organization.

Chapter 26

The Gimp Image Editor

Since I am talking so much about The Gimp, it seems good to include a chapter with some of the main points.

Gimp is free software. Do a Google search for Gimp to find many resources.

http://www.gimp.org/downloads/ is a source for free downloads. Gimp is available for PC, Mac, and Linux.

Gimp Menu Commands

Here are some Gimp commands that I use in this book.

Gimp Menu: Edit / Cut deletes whatever is currently selected.

Gimp Menu: Edit / Undo reverses the effects of the previous command. Can be done repeatedly.

Gimp Menu: File / Export writes the current image in whatever image format is desired.

Gimp Menu: File / New creates a new image that is blank.

Gimp Menu: File / Overwrite writes the current image by replacing an existing image file.

Gimp Menu: Filters / Map / Make Seamless modifies the current image so it can be used for seamlessly tiling the background of a webpage.

Gimp Menu: Filters / Map / Small Tile shows what the current image would look like if it were used to tile the background of a webpage.

Gimp Menu: Image / Canvas Size changes the size of the image.

Gimp Menu: Image / Crop to Selection deletes everything that is not currently selected, and changes the size of the image.

Gimp Menu: Image / Scale Image changes the number of pixels in the image but does not change the general appearance of the image. Mostly it is used to shrink an image to a smaller size so it can be more quickly displayed on a webpage.

Gimp Menu: Layer / Transform / Offset shifts the image left or right, and up or down, with wrap-around so the pixels shifted off on the left get added back on the right, and so forth.

Gimp Menu: Layer / Transparency / Add Alpha Channel adds the capability for transparency to the current layer of the image.

Chapter 27

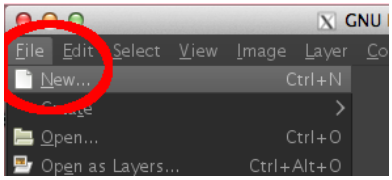
Gimp Makes A Logo

In this appendix, we use Gimp 2.8 to demonstrate how to create an image like a logo from scratch, without using a photograph. A portion of the image will be transparent.

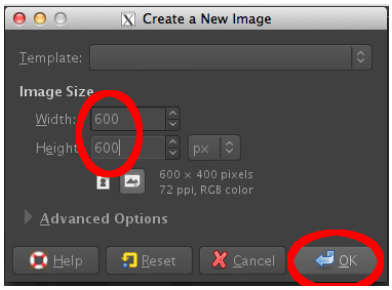
When such an image is displayed on a webpage, the part that is transparent will show the colors behind the image, whatever would have been there if the image were not.

This can be particularly helpful for logos or trademarks because they can be displayed on any page without worrying about the background color or texture of that page.

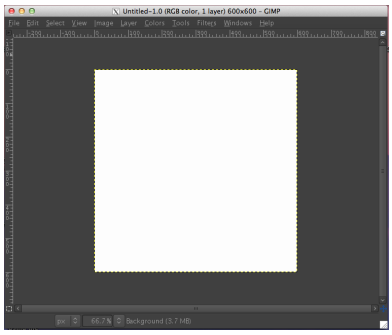
From the menu, I select **Gimp Menu: File / New**.



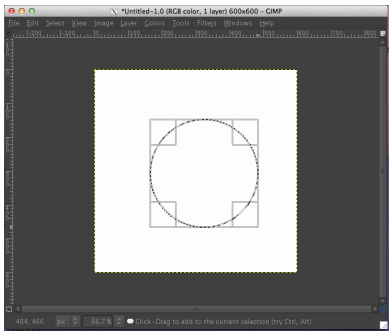
I specify the size, here 600 by 600 pixels, and click on "OK."



This creates a canvas that is 600 by 600, with a white background.

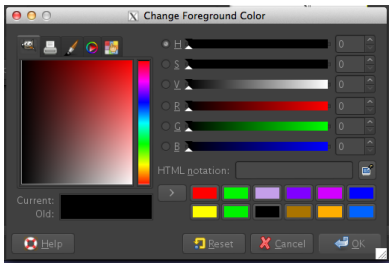


I double-click on the circle icon, and then click and drag a circle on the canvas.

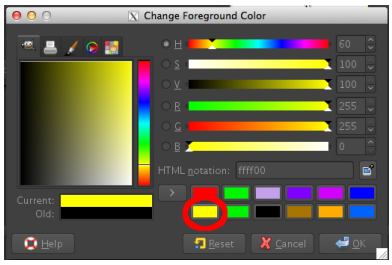


The color scheme is black (foreground) on white (background). I want to change the foreground color so I double-click on the black square.

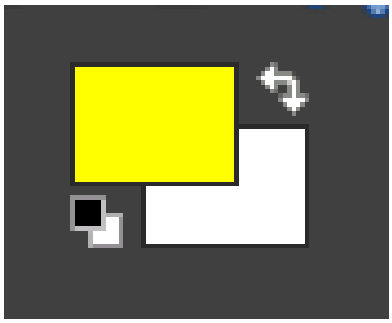
I am presented with a color selection screen.



I want yellow, so I double-click the yellow cell.

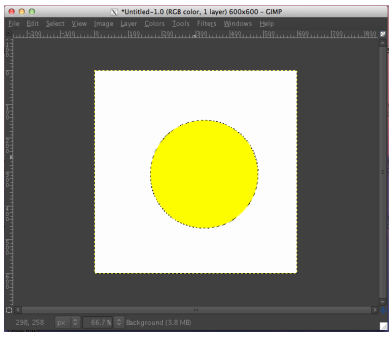


The color scheme changes to yellow (foreground) on white (background).

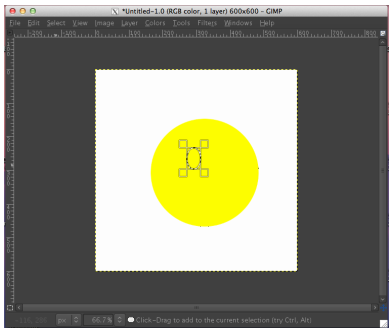


I double-click on the paint bucket. I will use it to dump yellow paint into my circle.

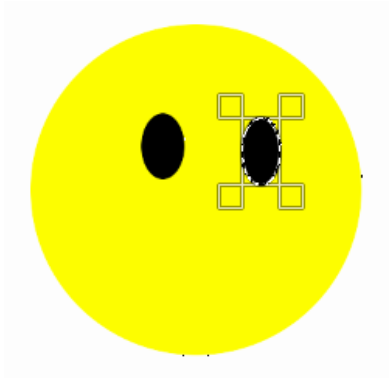
I move the cursor into my circle and double-click. The inside of the circle is now yellow.



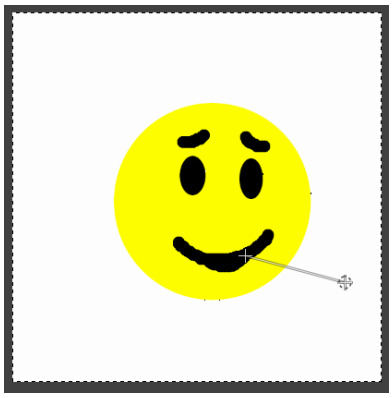
I change the color scheme back to black on white. Then I double-click the ellipse tool, and I draw an ellipse for an eye on my smiley face.



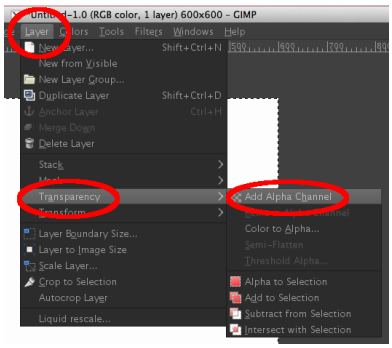
Using the bucket, I fill the eye with black. Then I repeat the process to create another eye.



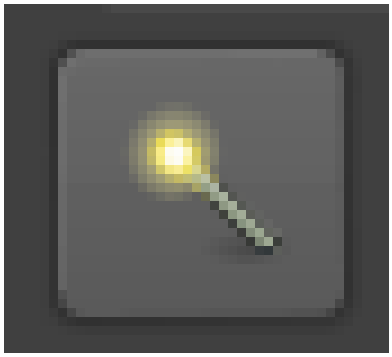
I double-click the pencil tool and use it to draw eye brows and a mouth. My graphic is complete. (And actually kind of cute.)



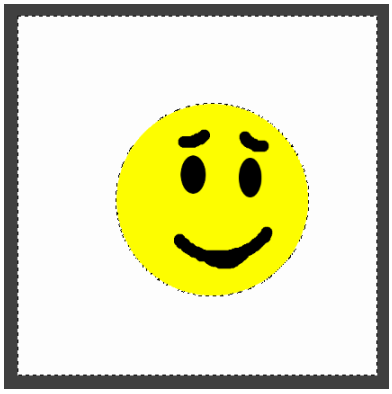
I am ready to add transparency (although I could have done it earlier). I want the white parts to go away. I select **Gimp Menu: Layer / Transparency / Add Alpha Channel**.



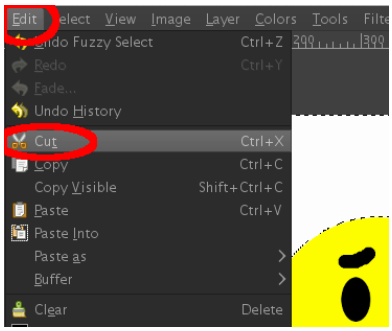
Next, I select the magic wand tool and click anywhere in the white part of my canvas.



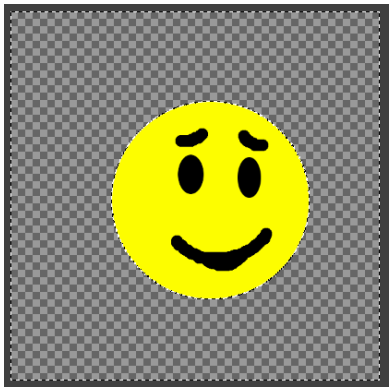
The entire white area gets a dotted line around it, indicating that it has been selected.



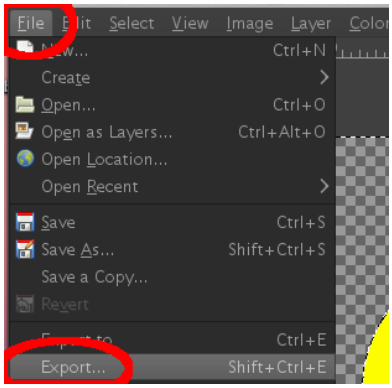
I select **Gimp Menu: Edit / Cut**, and press enter.



The white parts of my canvas have been replaced by a gray checkerboard, which is commonly used as the symbol for transparency.

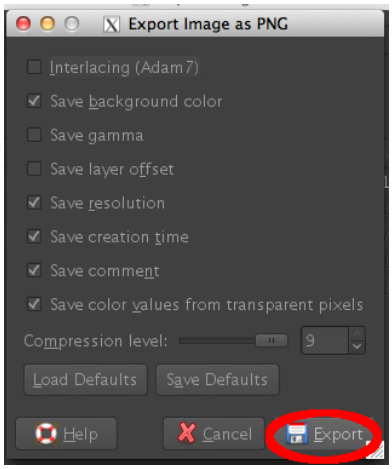


Next, I select **Gimp Menu: File / Export**.



png supports transparency. jpg does not.

I complete the export process. I now have a file named smiley.png that consists of a smiley face surrounded by transparent background. Later I will be able to use it on a web page.



Unit III

Under Development

Chapter 28

Styling With CSS

Contents

Best Practices: Write valid HTML code.

Exam Question 359 (p.358)**:** For each <meta> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: forbidden

Exam Question 360 (p.358)**:** What HTML markup is used to specify the name of the webpage?
Acceptable Answer: `<title>...</title>`

Exam Question 361 (p.358)**:** For each <title> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?
Required Answer: required

Exam Question 362 (p.358)**:** What CSS attribute: places a picture in the background?

Required Answer: background-image:

Exam Question 363 (p.358)**:** What CSS attribute: draws a box around some content?
Required Answer: border:

Exam Question 364 (p.358)**:** What CSS attribute: puts space between any box you draw around content, and neighboring content?
Required Answer: margin:

Exam Question 365 (p.359)**:** In CSS what is the one-character symbol for class?
Acceptable Answer: .

Exam Question 366 (p.359)**:** What HTML attribute= is used to specify class?
Required Answer: `class=`

Exam Question 367 (p.359)**:** What CSS attribute places a picture in the background?
Acceptable Answer: background-image:

Exam Question 368 (p.359)**:** If you want to draw a box around some content, what CSS attribute would you use?
Acceptable Answer: border:

Exam Question 369 (p.359)**:** If you draw a box around some content, what CSS attribute puts space between your content and the box?
Acceptable Answer: padding:

Exam Question 370 (p.359)**:** In CSS, can padding be negative?
Acceptable Answer: no

Background images and colors extend through both the content, the padding, and the borders (if the border has gaps, like with double, dotted, or dashed). Backgrounds do not extend into the margin.

Exam Question 371 (p.359)**:** In CSS, does background extend into padding?
Acceptable Answer: yes

Exam Question 372 (p.359)**:** In CSS, does background extend into borders?
Acceptable Answer: yes

Exam Question 373 (p.359)**:** In CSS, does background extend into margins?

Acceptable Answer: no

Exam Question 374 (p.359)**:** If you draw a box around some content, what CSS attribute puts space between the box and neighboring content?
Acceptable Answer: margin:

Exam Question 375 (p.359)**:** In CSS, can margin be negative?
Acceptable Answer: yes

Where To Specify Styling

Styling can be done on a per-element basis, a per-page basis, or a multi-page basis. Appendix 34 (page 288) talks more about this.

The best practice is to do the styling at the level that matches how broadly the style happens. Is it everywhere or just in exactly one spot?

If you want certain styles to apply across your whole website, or several similar pages, they you should create an **external style sheet** and link to it from each webpage. This is the most-recommended approach.

If your webpage is more personalized than the whole website, you would use an **internal style sheet** (or, stylesheet, written without the space) specified in the head of your webpage. This would take precedence over anything stated in the external style sheet.

For small areas of a webpage that need special styling, it is normal to do in-line styling, using the **style=** attribute of HTML. But be careful. If you are styling many things the same way, you should move the styling to your style sheet, either the internal one or an external one, and then use **class=** to apply your styling.

28.1 In-line Style

When you use style= as an HTML parameter, it is called in-line styling, and in case of conflict, it takes precedence over all other styling that might have been requested.

If we want a particular h1 to use red text, we could do it like this:

```
<h1 style="color: red;">This is a heading</h1>
```

Every HTML tag allows style= as a parameter.

Although inline styling is permitted, using inline style is regarded as a bad thing. The best practice is to separate the styling into a style sheet, such as the internal style sheet we will mention next. The HTML part of the document can then hook onto the styles by using class= and id= as needed.

28.2 Internal Style Sheet

When you use `<style>` and `</style>` in the head of an HTML document, it is called an internal style sheet.

If we want all h1 headings to use red text, we could do it like this:

```
<head>...
<style>
h1 { color: red; }
<style>
<body>...
<h1>This is a heading</h1>
```

If we want one particular h1 heading to use red text, we could use an ID to target that h1, like this:

```
<head>...
<style>
#red { color: red; }
<style>
<body>...
<h1 id=red>This is a heading</h1>
```

If we want several h1 headings to use red text, we could use a class to target them, like this:

```
<head>...
<style>
.red { color: red; }
<style>
<body>...
<h1 class=red>This is a heading</h1>
```

28.3 External Style Sheet

External style sheets (or, stylesheets, without the space) can be shared among many webpages.

```
<head>...
<link rel=stylesheet type=text/css href=/style.css>
```

Then, in a totally separate file, in this case, in (docroot)/style.css, we would have this content:

```
.red { color: red; }
```

The external style sheet gets "included" into your webpage by reference. It is pretty much the same as if you had pasted it into your webpage between a `<style>` and `</style>` tag set.

Notice that the external style sheet does not have any html tags. Specifically, it does not start with `<style>` or end with `</style>` as an internal style sheet would.

The advantage of using an external style sheet is that the same style sheet can be used by many webpages, and corrections to the style sheet can be made in a single place, and have immediate effect everywhere that style sheet is used.

28.4 Box Model: Padding, Border, Margin

The Box Model has its own chapter: 12 (page 126). We summarize here briefly, and give a few examples, but see the chapter for details.

```
padding: 5px;
border: solid black 3px;
margin: 5px auto;
```

Webpage content is naturally rectangular. Someday that might change, but for now it is rectangular. It is possible to flow content around these boxes. See the float: and clear: styles elsewhere.

padding: Around the content, you can have padding, which is empty space that has the same background as the content.

border: Around the padding you can have a border, which is colored and patterned and has thickness. Empty parts of the border are also colored the same as the background of the content.

margin: Around the border you can have margin, which is always transparent.

28.5 Fonts

Fonts are the letter shapes used by the text. Letters can be plain or fancy. This is controlled by the font family you select, or by the default font family if you do not select anything.

Chapter 10 (page 99) went in depth with fonts. In particular, see section 10.4 (page 106) for more information about the font stack. For right now, we can get you started with these:

```
font-family: serif; /* normal printing */
font-family: sans-serif; /* plain printing */
font-family: cursive; /* flowing script */
font-family: fantasy; /* strange things */
font-family: monospace; /* even letter sizing */
```

28.6 CSS Validator

W3C is the standards body that develops standards for the World Wide Web. http://www.w3.org/ is their homepage. Their materials are authoritative.

W3C does not try to provide tutorials and guides for novices, but instead caters to the detailed and precise needs of experts.

They provide two incredibly useful validators that can be used to find errors in your HTML code or CSS code.

http://jigsaw.w3.org/css-validator/ is their validator for CSS code.

28.7 CSS

todo: Target practice: using css selectors.

Exam Question 376 (p.359)**:** What CSS attribute: makes an element partially transparent?
Required Answer: opacity:

Exam Question 377 (p.359)**:** What CSS attribute: sets the width of a block item?
Required Answer: width:

Exam Question 378 (p.359)**:** What CSS attribute: sets the largest allowed width of a block item?
Required Answer: max-width:

Exam Question 379 (p.359)**:** What CSS attribute: sets the smallest allowed width of a block item?
Required Answer: min-width:

Exam Question 380 (p.359)**:** What CSS attribute: sets the height of a block item?
Required Answer: height:

Exam Question 381 (p.359)**:** What CSS attribute: sets the largest allowed height of a block item?
Required Answer: max-height:

Exam Question 382 (p.359)**:** What CSS attribute: sets the smallest allowed height of a block item?
Required Answer: min-height:

Exam Question 383 (p.359)**:** What CSS attribute: starts flow-around (of text around pictures)?
Required Answer: float:

Exam Question 384 (p.360)**:** What CSS attribute: stops the flow-around action?
Required Answer: clear:

28.8 Example: Colors

This sets the overall background color to be yellow, and the text colors as follows: h1 is blue, h2 is red, and p is green. For h2, the background color is not yellow, but instead white.

```
<style>
body { background-color: yellow; }
h1 { color: blue; }
h2 { color: red; background-color: white; }
p { color: green; }
</style>
```

Each "line" consists of a selector, then an opening brace, then the style commands, and then a closing brace. I say "line" in quotes because it can actually spread over several lines with no problem.

The selectors in our example above are body, h1, h2, and p. Each of these matches to a tag in the webpage.

The styling attributes we show are background-color and color. There are many styling attributes.

We talked about color and backgrounds in chapter 7 (page 71).

28.9 Borders

Images, paragraphs, and other things, can have borders added to them. The following styling creates a solid border that has a thickness of 1em (the width of the letter m in the current font) with a color of pink.

```
border: 1em solid pink;
border: thin double black;
```

Thickness: Section 12.4 (page 133) talks about the different kinds of dimensions you can use, including px (pixel), in (inch), cm, and mm.

Pattern: Section 12.1.3 (page 129) talks about the different kinds of borders you can use, including solid, dotted, dashed, and double.

Color: Chapter 7 (page 71) talked about the different colors you can use, and how you can specify them. You can specify them by name or by their mix of red, green, and blue, or other ways.

Borders are part of a larger discussion about the box model, which includes borders as well as margins and padding. Chapter 12 (page 126) provides coverage.

Chapter 29

Dot TK: Domains for Free

Contents

It is good to go through the process of setting up your own domain name, even if it is just for the experience.

The first time you set up a domain name, the process can be daunting. Take a deep breath. The end result is really worth it.

There are places that will give you a domain name for free. One of those is Tokelau, a territory of New Zealand. They seem to be doing it partly for fame and glory.

I have used them and I recommend them. This chapter will tell you how to use their free service.

There is a downside. For **SEO**, having a free domains is a negative.

You will be doing three things at once.

(a) You will be using the **dot.tk** website to secure a domain name.

(b) You will be using your webhost's control panel to (b1) create a document root, (b2) create a homepage, and (b3) activate the addon domain.

(c) You will be using email to receive a message from **dot.tk** that lets you prove that your email address works so they can get back in touch with you later.

You will bounce back and forth between **dot.tk**, control panel, and email during the process of setting up your new domain name.

http://dot.tk/ is the place to start.

29.1 Terms of Service

Because **dot.tk** is giving away domain names for free, they need to be careful. They do not want to be abused. They worry about cyber-squatting. They worry about copyright violations. Their terms of service require that you actually create a website, and that you do not host illegal content such as porn or warez or things that violate copyright, and that you not simply park your domain name. They actively scan all their free domain names to make sure you are not abusing their free service.

Porn means pornography, like pictures of nude people. It is illegal in some places. Avoid posting it.

Warez means computer programs, often games, that have been illegally modified (hacked) to avoid the anti-theft features. These features are called **DRM**, for Digital Rights Management.

Copyright means images (normally) or other media content that is owned by someone else and is already posted on the web. It is easy to notice when the same thing gets posted a second time. Avoid posting your favorite cartoon or anime character.

Parking means a webpage that is not intended to provide content, but only to refer visitors to another website. Often these referrals collect money from Google for providing traffic to the eventual website.

29.2 Create Some Content

This step happens at your webhost, by way of your control panel.

You will be creating an addon domain.

Decide on a document root for your addon domain. We recommend this:

`(home)/public_html/tk/`

Using your control panel, create a `tk` folder within your `public_html` folder.

Within your `tk` folder, create a webpage. Call it `index.html`.

This is the webpage that the **dot.tk** web crawler will see when they come to check up on you. Your page needs to exist. It will be the homepage for your website.

Very soon, after you have activated your account with the **dot.tk** people, you will come back to your control panel and attach your new domain name as an addon domain, and you will mention the document root you have prepared here.

29.3 Pick A Domain Name

This step happens at the **dot.tk** website.

Pick a domain name that you like. Keep trying until you find one that is available.

WARNING: Once you find a name you like, and is available, you apparently need to complete the registration using the same computer at that time. If you move to a different computer to continue, the new computer will not be able to claim the domain name.

29.4 Provide for DNS

This step happens at the **dot.tk** website.

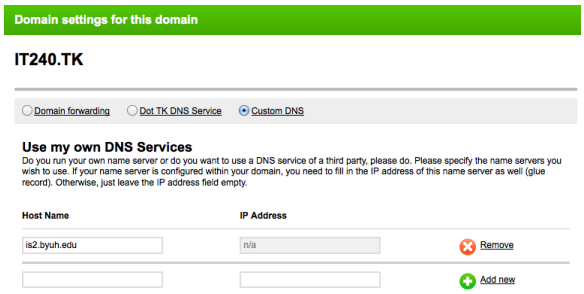
DNS is the Domain Name Service. When people come to visit your website, all they will know is your domain name. Your registrar (in this case, **dot.tk**) will tell them where to find your website.

dot.tk provides three options for DNS. (a) Redirection. (b) Use the **dot.tk** DNS system. (c) Provide your own DNS system.

We strongly recommend (c) you provide your own DNS system. This is already being done for you by your webhosting provider. Use it.

If using your own DNS, what is the fully-qualified domain name of your DNS server?

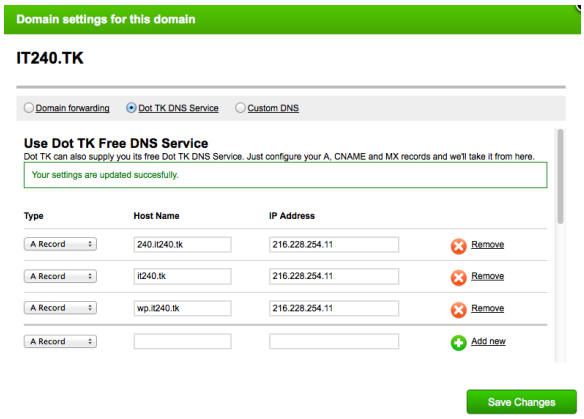
In this example, I am directing .tk to use my DNS (is2.byuh.edu) to handle everything.



You would use (b) if you want to host subdomains of your main domain on more than one webhosting provider.

If using their DNS, what is the IP address of your web server?

In this example, I am directing my .tk domain and two subdomains to point to the same IP address (216.228.254.11).



Redirecting makes your .tk name a shortcut to get to your original website with its original name.

29.5 Registration Length

This step happens at the **dot.tk** website.

Pick a registration length. The default is 3 months, but you can pick any number between 1 and 12. Later you can renew for free.

dot.tk will send you an email about 15 days before your domain name

expires. At that time you can extend your registration for up to 12 additional months.

One thing this means is that your email address must be working. If they send an email to you and you do not respond, maybe because you never got their email, then they will simply let your domain name expire.

29.6 Captcha

This step happens at the **dot.tk** website.

dot.tk wants to know that they are dealing with a real human, and not some automated robot that is setting up lots of domains. So they use a technique called **Captcha** to have you prove you are a human.

http://en.wikipedia.org/wiki/CAPTCHA has more on the Captcha technology.

29.7 Register Yourself

This step happens at the **dot.tk** website, and in your email inbox.

Now they have your domain name ready to go. They know how to direct people to your website. And they know you are a human.

Next they want to be able to get in touch with you later. You need to register.

WARNING: There is an option for getting the domain name without registration. If you pick this option, you will never be able to correct any mistakes you might have made setting things up. You really, really want to register.

dot.tk offers quite a few different ways you can identify yourself, including by email or through your Facebook account. Pick your method. I strongly recommend doing it by email.

When I used the email method, they sent me an email message with instructions I had to follow. Basically, they emailed me a code, and I had to visit a certain webpage and paste in that code within some number of hours. Then they immediately activated my account.

29.8 Activate Your Addon Domain

This step happens at your webhost, by way of your control panel.

Go back to your control panel on your webhost. Go into the section for addon domains. Type in your new domain name and type in the document root. When you save those changes, your website should start working.

29.9 Verify Your Domain Is Working

If you registered `whatever.tk`, visit `http://whatever.tk/` with your favorite browser to make sure you can see your new homepage.

Hopefully everything will work and you can do the happy dance. If not, go back over the previous instructions carefully.

If Things WERE Working But Suddenly Stop

Sometimes you will have everything working and then a day or two later it stops working.

As mentioned above, **dot.tk** uses a robotic web crawler to make sure you are following their terms of service.

If their web crawler finds any violation of their terms of service, they will shut you down and notify you.

If their web crawler cannot access your website, they will shut you down and notify you. They will give you a few hours to get something set up, but if too much time passes and you still do not have anything, they will shut you down.

If their web crawler finds copyrighted materials on your website, they will shut you down and notify you.

If their web crawler finds you have parked your domain name, they will shut you down and notify you.

If they shut you down, people visiting your website will probably see a page of advertisements hosted by **dot.tk**. This is called **parking** your website. People will not see your content. You may think that your website has been hijacked. It is just the **dot.tk** people trying to get your attention because you did not respond to their email.

You may be tempted to start over, create a new domain name, and cross

your fingers for good luck. That is a bad idea. Whatever caused the problem in the first place will cause another problem for your next domain name. It is better to solve the problem.

Look in your email for a message from **dot.tk**. Maybe it is in your spam folder. Read it carefully. Keep calm and do not freak out.

Follow the directions in their email to you. This may include making sure your website is actually serving a legitimate webpage and that you are not displaying copyrighted materials. Once you have complied with their terms of service, they may have a link you can press to reactivate your website.

If all else fails, send an email to `support@dot.tk` and humbly tell them that your website is messed up somehow, and you are not sure what to do to fix it. Remember that they are providing this as a free service so be nice to them. In my experience, they respond quickly, often within an hour, and either reactivate your website or tell you exactly what violation they found so you can fix it.

The **dot.tk** organization seems to be located in Amsterdam, in the Netherlands.

Chapter 30

ID

Contents

The one-character symbol for ID is `#` (hash), both in CSS and in the URL.

Exam Question 385 (p.360)**:** In CSS what is the one-character symbol for ID?
Required Answer: #

It is used in the fragment ID, as the last part of a URL, to specify that part of the webpage that should receive the focus.

It is used in CSS to indicated that a particular selector is an ID selector.

It is used in JavaScript to target specific elements for special treatment.

The ID itself must contain one or more characters, and none of them can be a space character (space, cr, lf, ff).

The ID must be unique within the webpage.

30.1 Fragment ID

The fragment ID is the last part of a URL. It specifies that part of the webpage that should receive the focus. Within the page, that part is identified by having `id=something`.

Example: We want to use "abc" as the ID. We want to identify a certain h2 heading with that ID. When we go to the webpage, we want that h2 heading to be in view, even if it is far down the page.

The URL would be `http://(something).html#abc`

The HTML would include `<h2 id=abc>`

30.2 Cascading

The C in CSS stands for Cascading.

At any point on the webpage, the style depends on the tags that are in effect, and the classes that are in effect, and the IDs that are in effect.

CSS has special rules for resolving any differences that might occur.

For example, if the body tag requests a yellow background, and within that, the paragraph tag requests a green background, and with that, the span tag requests a white background, the span tag will win because it is closest to the actual words being colored.

Styles requested right inside the HTML take priority over styles requested in the style sheet.

Styles requested in the style sheet have various priorities depending on how specific the request was, and whether it is based on ID or class or tag.

The most specific request wins.

Chapter 31

Backgrounds

Contents

transparency fixed repeat-x tiled 100% auto background-image image attachment

31.1 Background Image (URL)

You can specify the background image behind text by writing `background-image: url('xxx');` where the xxx is replaced by the URL for your picture.

The image will extend through the padding and border areas, but not into the margin.

More than one image can be specified. (Separate the URLs by commas.) They will be layered.

The images can be partly transparent. This can be accomplished by using an alpha channel on the image itself. It can also be accomplished by using an opacity setting.

http://www.w3schools.com/cssref/pr_background-image.asp has more.

The initial image can be positioned vertically (top, center, or bottom), and horizontally (left, center, or right). Position can also be specified using percentages or lengths (like pixel or inch).

The image can repeat from its initial position, in either the x direction (horizontally) or the y direction (vertically) or both.

http://www.w3schools.com/cssref/pr_background-position.asp has more.

31.2 Opacity

Opacity is a measure of how transparent something is. An opacity of zero means it is totally transparent. An opacity of one means it is totally opaque, or solid, with nothing showing through. Opacity can be any numeric value between zero and one.

In CSS, `opacity: 0.5;` would set opacity to half transparent, half not, almost as though it were a ghost.

When seeking transparency, the two major choices are to use opacity, as above, and to use a background color that is partly transparent, via rgba. See **rgba** in the index for several discussions.

31.3 Gradients

Gradients are a way to create a background that shades gradually from one color to another.

Example: `body { background: linear-gradient(90deg, #aabbcc, #887766);}`

http://css3gen.com/gradient-generator/ has a nice linear generator.

todo: more to be added

The ColorZilla website has a gradient generator.

31.4 Overlapping Backgrounds

It is possible, and sometimes fun, to specify several backgrounds for the same object. This can work well if the layers in front are partly transparent.

```
body { background-image: url(a.png), url(b.png), url(c.png); }
```

The first image will be displayed on top. The second will be displayed beneath it. If the first has some parts that are transparent, then the second will show through in those places. The bottom layer should not have any transparency.

By default the background repeats both x (horizontally) and y (vertically). If you use different image sizes, the resulting background can appear to be non-repeating.

Chapter 32

WordPress and CMS

Contents

32.1 Content Management Systems

You **can** develop your own website from scratch, but many people do not. Instead, they use **CMS**, a **Content Management System**.

About 1/3 of the top million websites (by traffic count in 2011) use a CMS.

The most commonly used CMS in the world is **WordPress**. It has about 55% of the CMS market in 2011, and 60% in 2015. Second place seems to be **Joomla**, and third place seems to be **Drupal**.

You can Google search "market share cms" for current information.

http://w3techs.com/technologies/history_overview/content_management

http://wordpress.org/

http://www.joomla.org/

http://drupal.org/

Because CMS is so useful as a way to get a website up and running, we will spend some time looking into it.

Specifically, in Appendix 32.2 (page 278) we will look into the leading CMS, WordPress.

Then we will return to looking at the fundamentals that underlie every CMS and the many other websites that exist: HTML, CSS, and JavaScript.

Exam Question 386 (p.360)**:** What does CMS stand for?
Required Answer: content management system

32.2 WordPress

About 1/3 of the top million websites (by traffic count in 2011) use a CMS. The top CMS in the world is WordPress. It has about 55% of the CMS market (in 2011). They are "proudly powered by WordPress."

http://wordpress.org/ is their homepage.

WordPress was invented to, well, press words. Think of it as a printing press for the web. It is great for sharing news and blogging. Those are its roots. But it has grown beyond that.

So, expect great blogging capabilities, and then other things too.

32.2.1 What is a Blog?

I know this probably sounds like a stupid question, but let's be fair. People may not know.

The word "blog" comes from "web log" which is like a log book, but published on the web. If you move the space, "web log" becomes "we blog".

http://en.wikipedia.org/wiki/Blog has more.

People who publish blogs are called bloggers.

Blogs consist of articles. These articles can include pictures or other media.

Blogs generally have a way for people to respond. Readers can post comments. Readers can reply to comments.

We will require you to become a blogger, at least for a few weeks. And we will require you to become a reader of the blogs written by your classmates. At least for a few weeks. See the course study guide for more information.

Exam Question 387 (p.360)**:** What does blog stand for? (two words)
Required Answer: web log

32.2.2 Installing WordPress

The fastest way to install WordPress is to just take a free blog right at WordPress itself.

http://wordpress.com/

Notice that this is a dot-com address, and WordPress itself is a dot-org address.

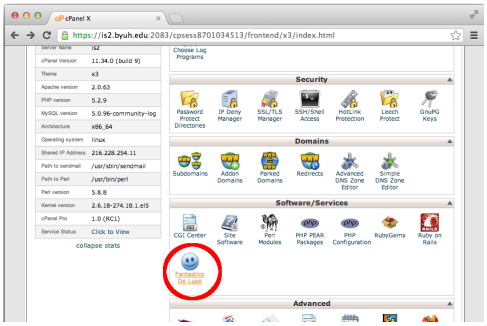
You can set up a blog named (yourchoice).wordpress.com for free. Of course, you can expect it to have outside advertisements and be limited, but it is free and that can be a good way to start.

32.2.3 Installing WordPress: cPanel and Fantastico

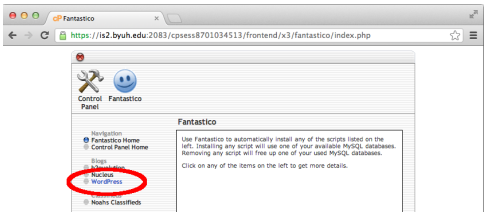
cPanel is commonly used and Fantastico is commonly available at web hosts.

Using cPanel, you can quickly install WordPress for your work in this class.

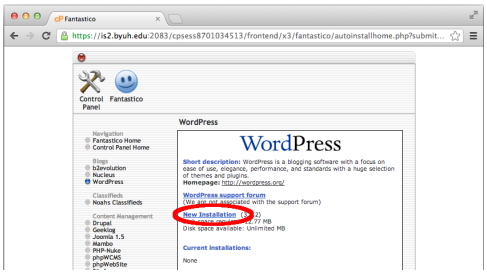
After logging into cPanel, we select Fantastico. It will handle the installation of WordPress (or many other things).



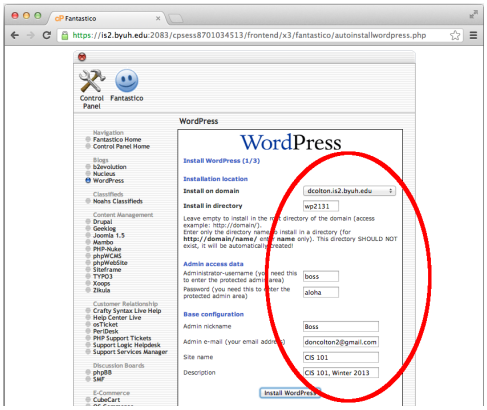
Within Fantastico, we will select WordPress for installation.



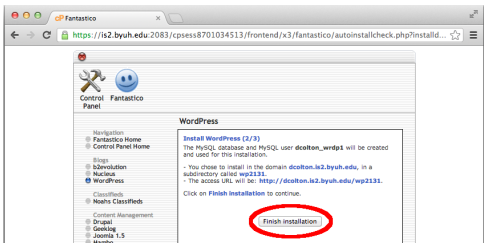
We will choose to do a New Installation.



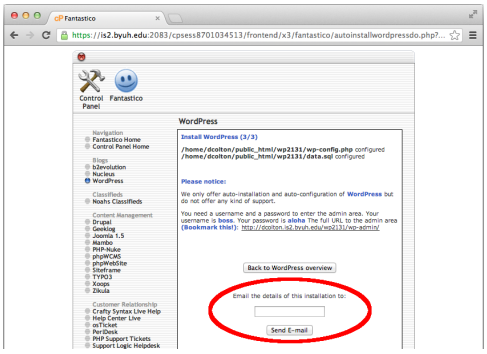
Fill in the blanks as you feel appropriate. Assume that you cannot change the domain, directory, or user name later, but the remaining items (password, nickname, email, site name, and description) can be changed.



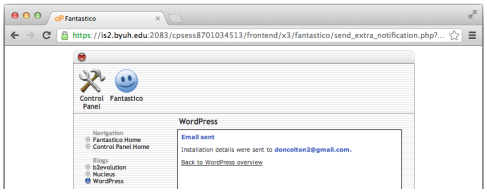
Now we "finish" the installation.



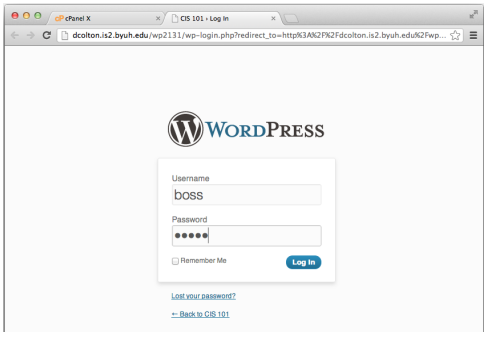
And we email ourselves a copy of the results.



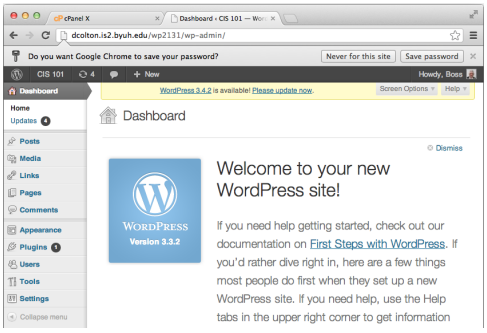
It reports that the email was successfully sent.



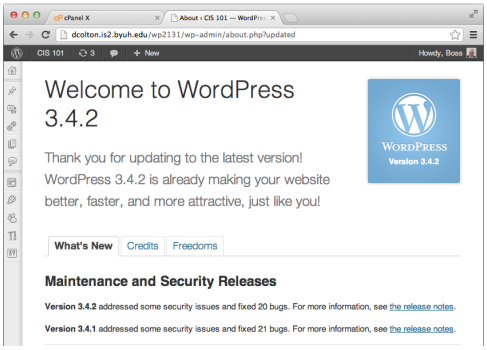
Here we log into our WordPress website.



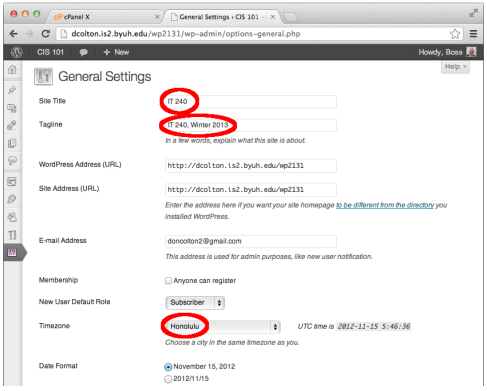
Welcome. And please upgrade to version 3.4.2. So, we follow the links and do the upgrade.



We have now upgraded.



There is a "ribbon" on the left edge of the screen. We can select the settings icon. (It looks like a couple of sliders for controlling trebble or base in music.) We can revise some of our earlier settings.



32.2.4 WordPress Tutorials

http://www.netchunks.com/a-list-of-legally-free-wordpress-ebooks/ A List Of Legally Free WordPress Ebooks, by Kori, posted Nov 2011. This is a blog entry. It lists six ebooks and four PDFs.

http://easywpguide.com/download/ Easy WP Guide WordPress Manual, by Anthony Hortin. 118 page PDF (2012). Free download.

32.2.5 WordPress Themes

todo: more to be added

using other people's themes

developing your own themes

32.2.6 WordPress Plugins

Plugins extend the functionality of your website.

todo: more to be added

blog / news feed

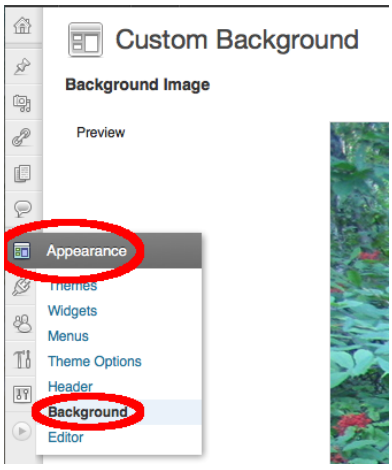
directory

making a Website Banner

32.2.7 WordPress HowTos

Question: How do you put a custom background image behind a WordPress page?

Answer: On the menu ribbon, select WP Menu: Appearance / Background, to establish a custom background for your entire website. You can upload a new image or select one that is already in your media library.



32.3 WordPress: Advanced Concepts

In Appendix 32.2 (page 278) we discuss **WordPress** and give some basic assistance in getting started with it. WordPress continues to grow and its popularity has probably passed the "tipping point" so that it will eventually represent 80% of the **CMS** market.

So, how do you get in on all this WordPress goodness? You can write your own themes. That way you can really take control of your websites, but at the same time ride on the success coattails of WordPress.

http://codex.wordpress.org/ is the perfect place to start your training.

WordPress uses the **PHP** scripting language.

WordPress uses the **mySQL** database language.

32.3.1 Developing Your Own Themes

You can start out easy by just developing your own theme. Mostly this involves creating CSS and images. You end up with your own private version of what WordPress should look like. (Or you can share it.) You can do as little or as much as you want, and WordPress will make up the difference.

http://codex.wordpress.org/Theme_Development

This page gives a simple, one-page introduction to developing WordPress themes. It covers all the major topics, giving a paragraph or two about each. Read it to get a sense for whether you want to take the plunge, and how much air to gulp in before diving.

32.3.2 The WordPress Loop

Maybe a theme is too under-powered for your vision. Maybe you want a full-blown application, but you like the fact that WordPress is widely available and provides a myriad of small things, like logging in and password recovery. You can use WordPress as your basis and build almost anything within that framework.

http://codex.wordpress.org/The_Loop

WordPress builds webpages using a database and custom programming that is part of each theme. The loop is the way you can totally customize what

is going up on the user's screen. This is where you capability equivalent to Drupal or any other high-end Content Management System. This is where WordPress escapes from being merely a blogging platform and turns into a full-blown application development platform.

32.3.3 Developing Your Own PlugIns

PlugIns can be shared across many themes. Each is basically a mini-webpage, a page within a page.

https://codex.wordpress.org/Writing_a_Plugin

This page gives a simple, one-page overview to developing WordPress plug-ins.

Chapter 33

Audio and Video

Contents

33.1 Encoding Audio

As with images, there are many encoding schemes for audio. We will briefly introduce mulaw, wav, mp3, Ogg Vorbis, and WebM.

MuLaw

Wav

MP3

Ogg Vorbis

WebM

33.2 Encoding Video

Video seems to have more experimental and vendor-connected encodings than audio or image. These seem to be connected to specific video players, such as Adobe's Flash, Apple's QuickTime, or Microsoft's Widows Media Video. Some vendor-neutral standards are emerging.

Google's YouTube (https://www.youtube.com/) has been massively successful, and its influence is helping the Internet's video world converge to a few standards.

MPEG

MP4

Ogg Theora

WebM

Chapter 34

Untangling HTML and Style

Contents

In some of our examples we placed `style="..."` directly into the HTML of the webpage.

This is called in-line styling.

It is usually considered to be a bad thing. Why so?

For uniformity and consistency, we generally like to apply one style across our entire webpage or website. When we embed the styling commands directly into the content, it makes it very difficult to stay consistent.

Ideally we want to specify all our styling commands in one place, and then have them apply throughout our website.

So, avoid in-line styling.

In an ideal world, we would put as little styling into the HTML as possible. That is because it makes things more complicated.

Why untangle? Writing content (HTML) is a skill. Styling content (CSS) is a skill. It is easier to find two persons who enjoy and are each expert in one area than it is to find one person who enjoys and is expert in both areas.

(We run into a similar problem when we use JavaScript like `onclick="..."`

directly in the HTML markup. It is better to separate it out.)

How can we best keep the CSS separate from the HTML?

34.1 Rules and Exceptions

The concept of CSS is to allow style to be established at many levels. We start with a general rule. Then we make exceptions. Then we make exceptions to the exceptions.

Our modern calendar (the Gregorian) works that way. The general rule is that each year has 365 days. The exception is leap years, which happen every 4 years. Then the year has 366 days. BUT, every 100 years, we skip a leap year. We are back to 365 days. BUT BUT, every 400 years, it is leap year anyway.

With our calendar, we have a rule, and an exception, and an exception to the exception, and an exception to the exception to the exception. Why not?

Styles also have rules and exceptions.

You can have an overall style, say maybe black text, at the document level. That would apply to all text in the entire document.

But you might want to break your own rule on a certain paragraph.

You can have a style, say maybe red text, at the paragraph level. It would apply to all the text in that paragraph, even if the document specified black text.

CSS is built on the idea that the same attribute, like text color, may be specified several different ways. CSS looks at all the requests and decides which request gets priority.

34.2 The Cascade

In-line beats internal. Internal beats external.

The most general solution to styling a website is to create a single style sheet and apply it to all the webpages in the website.

This is called an external style sheet.

This works wonderfully well in many cases.

It is especially wonderful because a new style can be introduced throughout your website by simply replacing the style sheet. All the webpages will refer to the new sheet.

The next most general solution is to style an individual webpage. Instead of having a file that is shared, the styling information can be embedded into the webpage, usually as part of the `<head>`.

This is called an internal style sheet.

When a style is specified in both an external style sheet and an internal style sheet, the internal style sheet takes priority.

The most specific solution is to style an individual tag within a webpage. This is done by using the `style="..."` attribute of the tag.

This is called an in-line style.

When a style is specified by in-line style as well as external or internal style sheet, the in-line style takes priority.

34.3 Specificity

When a style is specified in several different places, the most specific place takes priority.

Thus, an in-line style always takes precedence over everything.

But within style sheets, both internal and external, specificity can be calculated to resolve which style will be applied.

ID is more specific than class. Only in-line is more specific than ID.

Class is more specific than tag.

todo: add more content

Chapter 35

ch31 CSS Selectors

Contents

35.1 Tags

Each of the HTML tags can be used as a selector. body, div, h1, p, etc.

JavaScript can select some elements based on tag.

document.body

35.2 ID

Only one item can belong to a specific ID. If more items have the same ID, the browser cannot be expected to do what you want. Consider using a class.

JavaScript can select a particular item based on its ID.

document.getElementById('abc')

35.3 Class

In HTML markup, each tag can have a class= attribute. The attribute can have a list of classes to which this particular tag belongs. If there is more than one class, they must be surrounded by quote marks and separated by spaces.

Several items can belong to the same class.

35.4 Pseudo Class

active, visited

first child, nth child, first line, first letter

35.5 Transitions

todo: as we move from field to field, we can identify the field that is currently active by using a transition.

Chapter 36

ch32 Untangling HTML and Scripting

Contents

You will have noticed that in our examples, we placed "onmouseover" directly into the HTML of the webpage.

This is called in-line scripting.

In-line scripting is not best because it mixes HTML and scripting together. In an ideal world, we would put as little JavaScript into the HTML as possible. That is because mixing makes things more complicated.

Why untangle? (We said this before in the CSS unit, but it bears repeating.) Writing content (HTML) is a skill. Styling content (CSS) is a skill. Scripting active content (JavaScript) is a skill. It is easier to find three persons who enjoy and are each expert in one area than it is to find one person who enjoys and is expert in all three areas.

We run into a similar problem when we use `style="..."` directly in the HTML markup. It is better to separate it out and keep it in a style sheet.

But how can we keep the JavaScript separate from the HTML?

36.1 Add Event Listener

The secret is to include some JavaScript that runs after your webpage has loaded. It can wander through the DOM and add event listeners where you want them.

Here is an example.

```
<script>
var elements = document.getElementsByClassName("abc");
for (var i=0; i<elements.length; i++) {
  elements[i].addEventListener("keypress",myABC,true) }
</script>
```

In this example, we use a JavaScript function, getElementsByClassName, to search through the document tree for elements that belong to the class `abc`. The variable `elements` is used to hold a list of those elements.

Next, we walk down the list and perform the `addEventListener` procedure to each such element. Specifically we add an event listener for the `keypress` event.

This is essentially the same as if we had written:

```
<tag onkeypress='myABC'>
```

But instead, we can write this in the HTML:

```
<tag class=abc>
```

By doing this, we have removed almost all of the JavaScript from the HTML, and isolated it to only a few places.

36.2 ezCalc

Here is a complete example.

```
<!DOCTYPE html><head lang=en><meta charset=utf-8>
<title>ezCalc</title>
<style>
  body { text-align: center; background-color: #ffffcc; }
  input { font-size: 150%; }
```

```
</style>
</head>
<body>
<h1>ezCalc: One-Line JavaScript Calculator</h1>

<p>Type a calculation in any ezCalc blank.
 Press = to evaluate it.<br>
You can use numbers, parentheses, and + (add), - (subtract),
 * (multiply), / (divide), and % (remainder,mod).
<br><input size=50 class=ezCalc placeholder=ezCalc>
<br><input size=50 class=ezCalc placeholder=ezCalc>
<br><input size=50 class=ezCalc placeholder=ezCalc>

<script>
function ezCalc(event) { // evaluate if keystroke is "="
  var e = event || window.event;
  var k = e.charCode || e.keyCode;
  if (k != 61) return true; // 61 is "="; browser acts normal
  var a = String(e.target.value); // save for later use
  var patt = new RegExp("^[0-9.()+*/%-]+$");
  if (patt.test(a) == false) { return false } // avoid errors
  e.target.value = eval(a);
  e.preventDefault(); // prevent = being added after new value
  return false; } // true vs false makes no difference?
// register function for keydown on elements with class=ezCalc
var ezs = document.getElementsByClassName("ezCalc");
for (var i=0; i<ezs.length; i++) {
  ezs[i].addEventListener("keypress",ezCalc,true) }
</script>
```

Chapter 37

ch34 Getting Feedback

Contents

Once your website has gone live and people are viewing it, you want feedback. You want to get inside the mind of the user, find the problems with your website, and fix them for the best possible visiting experience.

Unfortunately (or maybe fortunately for those of us with thin skin), you cannot get the true feedback you want. The closest you can come is focus groups or lab-based observation of paid users. Expensive.

37.1 Pageview Counters

The first approximation to the truth is a pageview counter. If it shows a high number, then the page is being viewed lots of times. If it shows a low number, you are just not getting noticed.

This is sometimes called a **hit counter**.

And the pageview counter can be displayed right on the page to reassure visitors that they are visiting a popular page. There is something here worth

seeing.

If your page is generated dynamically, using CGI or an equivalent technology, the page generation program can keep a count of visitors, and can display it as part of the page itself.

If your page is static, you can still insert dynamic elements into it using JavaScript. Those dynamic elements could keep a pageview count.

You can do a Google search for "page view counter" to find things that are free and easy to install.

Counters require someone to remember and provide those numbers to you. You will have to create an account on their server, or install their software on your server.

37.2 User Comments

Many webpages allow users to leave comments. This serves several useful purposes.

First, it allows users to build a sense of community. They can talk to fellow viewers. They can hear alternative interpretations to the significance of what was on the page.

Second, it allows the content creators to listen to their community, and possibly respond back. You might be surprised and delighted or dismayed at what your viewers are thinking and saying.

Third, much like using a pageview counter, it tells viewers that your page is popular.

There is a downside: Spam. Spammers can leave comments too. They can say things like:

Wonderful page. Amazing. I love your work. You really made me think. You should really check out this.

They can say the same thing, robotically, no matter what the page is about. And some people will probably click through.

I have seen comment lists that were more than 50% spam.

There are two common responses to spam. (a) Use **Captcha** or similar technology to have the user prove they are not a robot. (b) Use an approval

process whereby a human views each comment and either accepts it or not.

37.3 Logfile Analysis

The Apache webserver, and presumably most other webservers, keeps logfiles. These logs tell the IP address of the visitor, and the time of the visit, and the page visited.

They may also identify the source of the page request. Did the user click on a link, and their request has a "referrer"? That would be worth knowing.

By looking at the requests from the same IP address, you can start to form a picture of their journey through your website. Where did they come from? Where did they go first, second, and third? How much time passed between each page retrieval? Where were they when they lost interest and quit asking for more?

37.4 External Analytics

The next step up from logfile analysis is Analytics.

Analytics works from your webserver's log files, or from something equivalent. It does a more thorough job of analysis.

And it can be out-sourced. Add a bit of JavaScript to your page and someone can provide analytics to you.

You want to know how many people have visited. How popular are you? And where are your visitors coming from? And what path do your visitors follow through your website? How much time did they spend on each page? What was the last page they looked at before they walked away from your website?

These are the kinds of questions that are answered by analytics.

Google provides free analytics. Do a Google search on "analytics" and you will find them. Sign up for a free account. Then insert a few lines of code right before the end tag for your head section. Every pageload, their code runs and sends back information to their server.

It is free, but they get something out of it. They get to see how popular your page is, and they can use that information to affect their awareness

that your page exists, and your ratings when people search for the things on your page.

Here is a sample of the JavaScript that Google has you insert into the webpages you want to track.

```
<script>
  var _gaq = _gaq || [];
  _gaq.push(['_setAccount', 'your account number goes here']);
  _gaq.push(['_trackPageview']);
  (function() {
    var ga = document.createElement('script');
    ga.type = 'text/javascript'; ga.async = true;
    ga.src = ('https:' == document.location.protocol
      ? 'https://ssl' : 'http://www')
      + '.google-analytics.com/ga.js';
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(ga, s);
  })();
</script>
```

37.5 Tracking Cookies

Of course, you are not the only web developer that wants to know what the users are thinking and doing.

Advertisers spend good money to learn the same things, and to get their message in front of the eyes of potential customers.

Sometimes their methods frighten people. Are you paranoid about **cookies**? Can "they" piece together a history of the sites you have visited? Will the police arrest you for suspicious behavior?

The public response to tracking also works against you. Your analytics may not be everything you wanted.

Still, analytics are useful.

Chapter 38

ch35 Content, Presentation, Action

Contents

You have created your first webpage. It includes several kinds of markup including a picture. This is the first step. It is a big step.

We will look at a webpage as being a collection of three main ingredients.

38.1 Content

The most simple webpage consists of plain text content. It is very easy to create. It can also be very boring, especially when you compare it with the competition that is out there.

Still, content is the main reason that people visit most websites. Content is king.

38.2 Markup

When you write a report, in addition to pure content you probably also use headings, special fonts, margins, and pictures.

We can identify these things within our webpages. This is done by inserting **markup** to specify, directly or indirectly, how the content should appear.

There are three main kinds of markup: presentational, structural, and semantic.

38.2.1 Presentational Markup

38.2.2 Structural Markup

38.2.3 Semantic Markup

Warning: Often when people say Semantic Markup, they actually mean Structural Markup. This is a common mistake. Expect it in others. Avoid it in yourself.

Semantic markup is focused on what the human thinks they are looking at. Semantic markup is getting more popular. Semantic means "meaning" and focuses on what kind of content it is, not on its structural relationships or how it should look.

`<header>` is an example of semantic markup to identify the header (top) of a webpage.

`<footer>` is an example of semantic markup to identify the footer (bottom) of a webpage.

`<nav>` is an example of semantic markup to identify the navigation bar of a webpage. That would typically include links to other pages.

Do a web search on "html5 semantic tags" for more information.

Semantic markup is beyond the scope of this book. We mention it so you will know it exists, and that it is an advanced feature of web design.

38.3 Monolithic Pages

Style is the presentation of the content. It includes centering, margins, fonts, colors, borders, columns, and other aspects of layout.

The word "monolith" literally means "one rock" and refers to the lack of separate structure. If you pour some lava into a mold and let it cool, it will harden into a chunk of rock without any discernible structure. It's just all mixed together.

In the olden days (and you will still find webpages that do it this way), all style was implemented through directives **embedded** in the webpage. We already saw a couple of these: `<b>` for bold, and `<i>` for italic. But there were many others, including the use of tables for layout, and especially the use of special properties of markup elements.

The presentation was embedded deeply into each webpage. Changing the presentation could be very difficult, and could require modifying lots of separate webpages.

This is seen as a bad thing.

38.4 Cascading Style

Modern style is most often implemented through Cascading Style Sheets (**CSS**). This largely separates the presentation from the content, and that is a good thing.

Instead of marking a phrase as "yellow background," we put it into a class called, for example, highlight. Then we specify that highlight includes yellow background.

If we want to change from yellow to pink, it is an easy matter, with a single change to a single place in the website. All sections belonging the the class "highlight" will immediately change to pink background.

38.5 Separating Form and Content

There are many good reasons for getting away from the monolithic coding approaches of yesterday. The reason I like best is this:

Content experts are not always great at style.

Style experts are not always great at content.

Separating form, which is style, from content, is a straight-forward recognition that we may want to divide the labor of website design among experts, each of which can specialize in the area they love.

But there is another great reason: reuse. The style that is created for one page can often be reused on other pages. In fact, your website will look more cohesive if the style is shared.

And there is another great reason: mass updating. If you decide to change the style for your website, and if that style is all in one place, shared across all your webpages, then you only have to change that one place. After you change your style sheet, **all** of the pages that use it will be instantly restyled. (Unless caching temporarily gets in the way.)

38.6 Best Practices: Separation

It is well-established that the best-practice approach is to separate content from styling.

What that means to you is the HTML of your webpage should include as little styling as possible. Maybe none. All the styling should be moved external, to a style sheet.

The downside to this is it requires planning. It requires analysis and thinking. It promotes orderly thinking. And at the start of a project, it slows you down.

So, don't worry about it. At first. Use in-line or internal style.

While you are developing your webpages, it is okay to put styling right into the page. That is the "quick and dirty" development process. You can come back later and migrate it to a style sheet. Just remember to do it.

Do worry about it. Later. (Soon.)

Skilled developers already have a good sense of how to design their style sheets. They will not resort to quick and dirty techniques nearly so much as the newbies who lack the depth of experience.

38.7 Action

In olden days, webpages were static. You clicked on a link and got a whole new page. The burden was on the webserver to control everything. To change the webpage, we have the server revise it and send it back to us.

The modern approach is to move that control more and more into the webpage itself.

A good example is form validation. You might key in a credit card number and try to submit your form, but the webpage may do a preliminary check on the credit card number to make sure it is valid, or at least not clearly invalid.

Credit card numbers include something called a check digit, which generally comes last. That digit can be totally predicted by the digits that come before it. If you change the last digit, you don't get a different credit card. Instead you get an invalid number, one that can never be correct for any credit card.

By checking for a correct check digit, the webpage can save the server some work. Most errors result from one digit being changed, or from two digits being swapped. The check digit catches 100% of those kinds of errors.

This form validation is typically done using JavaScript.

We will talk about JavaScript lightly. It is more advanced than we really want to get into deeply. That is why we will just talk about it lightly. (Whole books have been written about JavaScript.)

AJAX: Even more powerful than simple JavaScript is something called AJAX, which we discuss briefly in section 17.7.4 (page 183). AJAX uses JavaScript to update just a part of a webpage with new content from the web.

Chapter 39

ch36 Notable Resources

Contents

Of course, search engines are amazing for getting three or four views of almost any subject, and in web design those views are very helpful, even if they turn out to be only 90% correct.

I want to call out two websites for special recognition. I really like both of them.

W3.org is an authoritative and official site. For the beginner, its single greatest contribution may be the HTML and CSS validators it hosts.

W3Schools is a privately developed website that is well organized and very easy to use. It has wonderful tutorial information.

39.1 W3.org

39.1.1 Validators

W3C is the standards body that develops standards for the World Wide Web. `http://www.w3.org/` is their homepage. Their materials are author-

itative.

W3C does not try to provide tutorials and guides for novices, but instead caters to the detailed and precise needs of experts.

They provide two incredibly useful validators that can be used to find errors in your HTML code or CSS code.

http://validator.w3.org/ is their validator for HTML code.

39.1.2 Tutorials

http://www.w3.org/wiki/Category:Tutorials has a list of educational materials provided by the W3 Consortium.

W3Schools, which we talk about next, gets right to the point and tends to answer my question immediately.

The W3 Consortium tutorials, on the other hand, try to be complete and informative. Like me, they tend to be wordy.

So, my advice is start with the W3Schools. If you get your answer, you are done. If you want more, try the W3 Consortium.

39.2 W3Schools is Amazing

By far the best online web development tutorial I have found is the one by W3Schools. Because of its consistent usefulness to me, I am pleased to provide links to many of the relevant webpages they provide.

W3Schools covers HTML, HTML5, CSS, CSS3, JavaScript, and many other web technologies.

Warning: W3Schools is not affiliated in any way with the W3C consortium. They are privately owned and operated. And they have been criticized for sometimes providing outdated or incorrect information. I think such criticism is a bit silly, but I should admit it exists.

By and large, though, I am very pleased with how easily they make information available, and their generally high level of accuracy.

http://en.wikipedia.org/wiki/W3Schools has more.

http://www.w3schools.com/default.asp is the W3Schools homepage.

Chapter 40

ch37 A General Overview

Contents

If your needs are common enough, you might be able to use a web development tool to create your webpage or whole website.

That could save you a lot of time, and make it possible to hand off the maintenance of the website to someone else (like the person who wanted you to set it up for them).

40.1 Common Needs

What is "common enough"? Out of all the many ways things could look, or be structured, there have emerged certain standard ways of doing things.

Blogging is probably the "poster child" for common. There are lots of blogs out there.

Photo albums is another good example.

Social media, including things like Facebook, let people create web pages where the emphasis is much more on content, to the extent that you actually

lose control over how the page is presented. You only control the content, and maybe not even that.

For-sale websites, like eBay and Craig's List, let people create web pages where the goal is to sell something. You are very limited in what you can do, but those limits do not usually matter because that enable you to easily do the main thing: advertise and sell something.

When any activity becomes common enough, someone will create a package deal that hides all the common details and lets you focus on the parts you want to control. Typically you just want to control the content.

40.2 Web Development Tools

What is a Web Development Tool? It is any solution where you do not have to learn HTML or CSS but you still end up with webpages.

Two important categories are Content Management Systems, such as WordPress, and HTML editors (page creation applications), such as Dreamweaver.

Content Management Systems, called CMS for short, use a cookie-cutter approach to web development. You pick your page layout from a list of already-created styles. You add an article or a page to your website. The CMS keeps track of everything.

Blogs are a good example of CMS.

In Appendix 32 (page 277) we look more at CMS.

40.3 HTML Editors

You should know that HTML editors exist and are used by many people.

http://en.wikipedia.org/wiki/HTML_editors has more, and can point you to lists of editors and comparisons between them.

Usually HTML editors will limit what you can do. They make many common tasks easy, but take away your ability to do some less common things. An HTML editor may be exactly right for you. Or not.

Word processors often have the capability to export documents in HTML format. It is not always pretty, but it is usually an option. It gets the job done quickly.

Typically these editors use a **WYSIWYG** (**wizzy-wig**) approach. That stands for What You See Is What You Get, and has become the standard way for Word Processors to edit documents.

Exam Question 388 (p.361)**:** What does WYSIWYG stand for?
Acceptable Answer: what you see is what you get

The alternative to WYSIWYG is Markup. We will learn Markup, but you should know that WYSIWYG exists and is very popular.

40.4 Uncommon: The Road Less Traveled

One danger of web development tools is that they limit you to doing things their way. Certain options may not be available to you.

The other danger with web development tools is that everything looks alike. While there are many customizations possible, still, when you see a WordPress page, you can often tell it is a WordPress page.

Maybe your creativity demands its own voice.

Maybe you do not want to be just like everyone else.

If so, then you need to go deeper, into HTML and CSS, and maybe even JavaScript. We cover those later.

But in this unit we cover things that are easy to do.

Why reinvent the wheel every time?

Sometimes using a tool is good enough.

Chapter 41

ch38 Codecs: Coding and Decoding

Contents

Text, images, sounds, and video: these are the media we most often encounter on the Internet. Each of these is a natural part of our life experience.

But for the Internet, they each have one important challenge: encoding.

Briefly put, computers work in a language called **binary**, where everything is represented by strings of ones and zeroes.

How do we represent things that are not ones and zeroes? How do we represent, for instance, the number five?

One method, binary integer, represents it as 101.

Another method, ascii character, represents it as 0110101.

Websites must encode all content in ways that the web browsers can correctly decode, or you just get random characters across your screen.

Theoretically, there are an infinite number of different ways that coding could be done. But from a practical point of view, we need to agree on one way, or at most a few ways, of doing it. Otherwise, communication is difficult.

A **codec** is a piece of software (usually) that codes (co...) and decodes (..dec). It is used by the browser to decode the audio or video back into a usable form.

41.1 With Codes, Shorter is Better

One of the earliest electronic codes is the one that was used with the original telegraph. Morse code was built out of dots and dashes with extra space between letters. It was a variable-length code, with "e", the most common letter in English, represented by the shortest code, a single dot. "q" and "j," much less common letters, were represented by longer codes.

http://en.wikipedia.org/wiki/Morse_code has more.

Basing the Morse code on letter frequency was a good way to keep message length short.

We do something similar with language. Words, which we use to encode our thoughts, have varying lengths. Simple pronouns like "you," "me," "he," "she," and "it" are very short. Words in everyday usage tend to be longer but still manageable. Names of chemicals and drugs tend to be huge.

The secret is frequency. Frequent words and phrases become short. Less frequent words are long. This is a good solution. It helps keep overall message length short. Human language seems to naturally do this.

The bad news is that frequency depends on the sender and receiver of the message. Russians might use "da" (yes) a lot more often than I do.

The result is that there are a lot of different coding schemes for just about everything you can imagine, each having a place where it would be the best.

41.2 Patents

Patents are a common method for protecting Intellectual Property. When inventors want the exclusive right to control how their inventions are used,

they seek a patent. After a patent is granted, those that use the intellectual property must pay a royalty (money) to its owner.

Coding schemes can be considered Intellectual Property. The inventor may believe his or her scheme is awesome and may seek to patent it.

Patents can prevent standardization as people try to find ways to avoid paying royalties. This can result in more coding schemes.

41.3 MIME Types

MIME stands for Multipurpose Internet Mail Extensions. **MIME type** originated as a way to send email attachments. It did its job very well. As a result, it has grown far beyond its original role, and has been adopted as the way for identifying the coding method for all kinds of content.

http://en.wikipedia.org/wiki/Mime_type has more.

There are standard MIME types for text, image, audio, video, and many more things.

MIME types show up in webpages as **content-type** declarations.

But before you can even specify a MIME type, you have to settle on the text encoding with which you will specify the MIME type.

41.4 Encoding Text (Characters)

The standard approach to encoding text is to encode the characters one by one, giving each its own binary string.

The good news is that increasingly the problem of encoding text seems to be solved. The world seems to have reached an agreement.

The "world" has largely agreed on a "Universal Character Set" (UCS).

http://en.wikipedia.org/wiki/Universal_Character_Set has more.

The web has largely agreed on UTF-8.

UTF-8 stands for UCS Transformation Format, 8 bit. It is the currently accepted standard.

http://en.wikipedia.org/wiki/UTF-8 has more.

ASCII is an old standard. It stands for American Standard Code for Information Interchange. It is a subset of UTF-8.

Other coding schemes are also used, particularly in places where UTF-8 results in long encodings. I am thinking here especially of China, Japan, and Korea.

http://en.wikipedia.org/wiki/CJK_characters has more.

41.5 Encoding Images

There are several popular encoding schemes for images. In this section we identify five of them: GIF, JPEG, PNG, SVG, and TIFF. There are many more but their usage is less common. There is nothing to prevent the creation of additional schemes in the future.

GIF: Graphics Interchange Format

GIF stands for Graphics Interchange Format. It is one of the oldest image standards on the Internet. It was originally protected by patents, which caused some developers to avoid using it.

GIF provides both animation and transparency. Transparency allows for images that do not appear to be rectangular.

http://en.wikipedia.org/wiki/GIF has more.

JEPG: Joint Photographic Experts Group

JPEG stands for Joint Photographic Experts Group. It is very commonly used both in digital cameras and on the Internet. It is commonly abbreviated **JPG**.

JPEG does not provide either animation or transparency. It does a good job of image compression.

http://en.wikipedia.org/wiki/JPEG has more.

PNG: Portable Network Graphics

PNG stands for Portable Network Graphics. It is increasingly popular and

was developed in part to overcome the patent issues with GIF encoding.

PNG provides **transparency**. Transparency allows for images that do not appear to be rectangular.

http://en.wikipedia.org/wiki/Portable_Network_Graphics has more.

SVG: Scalable Vector Graphics

SVG stands for Scalable Vector Graphics. Codings like GIF, JPEG, and PNG, are called **raster graphics** and reflect individual pixels in the image. SVG uses **vector graphics** reflects whole lines rather than individual pixels. As a result, the image does not degrade and pixelate when it is expanded (when you zoom in).

http://en.wikipedia.org/wiki/Svg has more.

TIFF: Tagged Image File Format

TIFF stands for Tagged Image File Format. It is popular among publishers and other high-end users because it retains more information than jpeg and other raster formats do. File sizes also tend to be much larger than for jpeg.

High-end cameras tend to record their images in "raw" formats which are similar to TIFF.

Because TIFF files are huge in comparison to JPG and other formats, TIFF is not commonly used on the webpages.

http://en.wikipedia.org/wiki/Tagged_Image_File_Format has more.

Chapter 42

ch39 Tiled Backgrounds

Contents

Face it. Backgrounds can have a dramatic effect on the feeling you get when you visit a webpage.

There are a couple of options. In this chapter we will talk about tiled backgrounds, but you could also use (a) a large photograph, (b) something you download from the web, or (c) a simple color.

(a) large photographs suffer, sometimes greatly, from slow download.

(b) something from the web suffers from copyright problems, and from lack of personal creativity.

(c) simple color suffers from being boring.

We want to achieve three things: (a) not boring, (b) no copyright problems, (c) fast download.

We can achieve our goals by making our own tiled background.

42.1 Tiling

By default, if a background image is not large enough to fill the viewable screen, it will be repeated, both in the x direction (horizontal) and in the y direction (vertical).

Our trick will be to take a small portion of an image. We call that small portion a tile. Then we will repeat it endlessly.

The CSS goes in the style section, and looks like this:

```
body { background-image: url(my_tile.jpg); }
```

42.2 Edge Effect

When we tile the image, the edges may not line up pleasingly. They may be rather abrupt. It may be obvious that two tiles are coming together, and it may create a visible lattice or grid. That would be distracting.

Using an artificial image, you may be able to crop it exactly right so there will be no edge effect.

Using a natural image, we will use the Gimp Map/Resynthesize tool to alter the edges so they tile nicely.

42.3 Textures

Take a picture of something. We will crop it to get our tile.

For our background image, we want to use a texture that will be pleasing and random enough that it does not appear to be repeating. We also want it to not be distracting.

Uniform lighting helps a lot. Avoiding the vanishing-point effect of perspective helps a lot. Try to get a good perpendicular shot with even lighting.

Here are some ideas of textures. They might help you think of something even better.

leather, as on a leather-bound book.

cork, as on a cork bulletin board.

paper, but not plain paper. I am thinking of something with more character, more like parchment, or linen, or vellum. Even blank newspaper, maybe near the edge of a page.

fabric, but without any printing on it. Emphasis would be on the weave of the threads.

carpet, like fabric, preferably with small features, or photographed from enough distance that we can shrink the features.

masonry, as in bricks, like a brick wall. However, the pattern may be much too large for effective tiling.

masonry, as in a close-up of an individual brick or CMU (cinder) block.

stucco, as on the side of a building.

cement, as on a sidewalk. We want to avoid cracks and anything else that will be noticeable when repeated.

asphalt, as on a road.

sand, as on a beach.

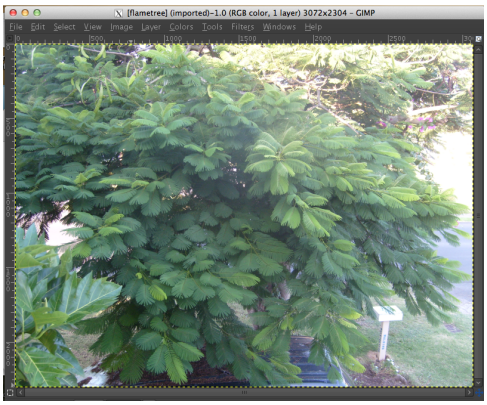
dirt, as in a newly-turned garden plot.

grass, as on a lawn. This may suffer from the obvious discernible pattern of the individual blades of grass.

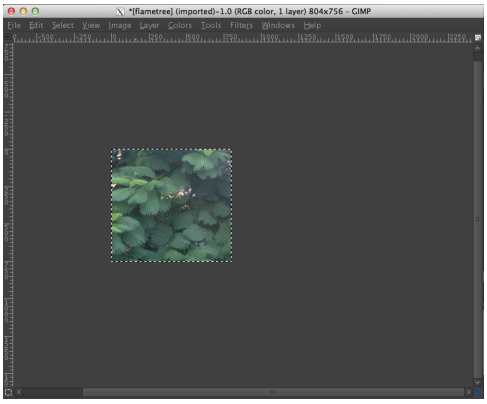
42.4 Example 1

There is an easy way to turn anything into a tile. We show it first. It has the advantage that it is very easy. It has the disadvantage that the repeated pattern is obvious and may be distracting.

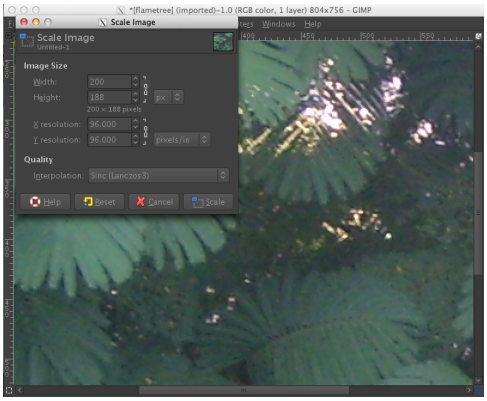
Here is our original picture. It is a Flame Tree located in my own front yard. The leaves look as though they might make a nice background.



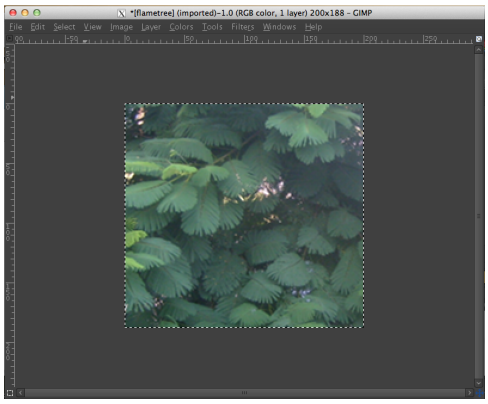
We select a small portion of the tree where the leaves seem to be fairly similar. It is good to have similar lighting across the entire image.



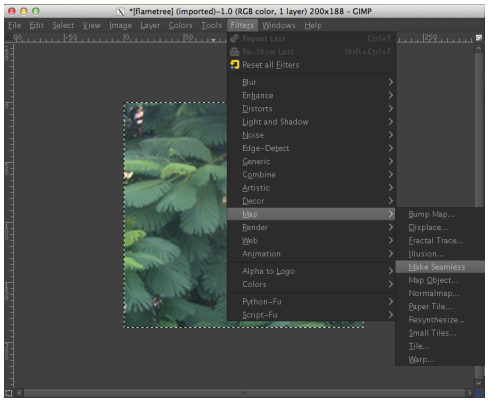
We scale the image so its size is reasonably small. Here we choose to make the image 200 pixels wide. Our goal is to make the file small enough to load quickly but still big enough to be interesting.



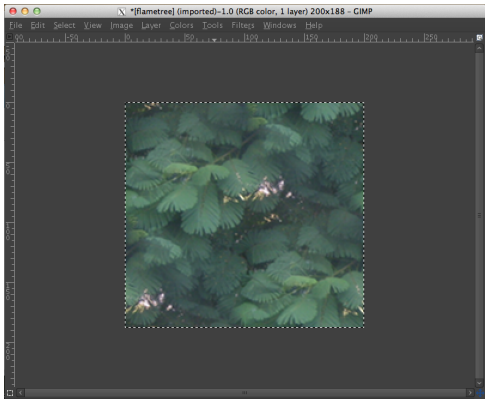
We resize the view. Here we see the resulting image, displayed at 200%.



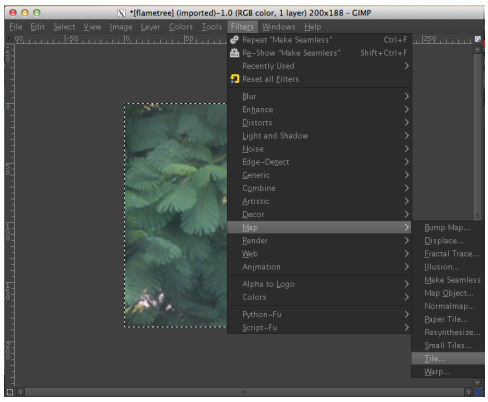
Here is the trick. We use **Gimp Menu: Filters / Map / Make Seamless**. This will do all the work for us.



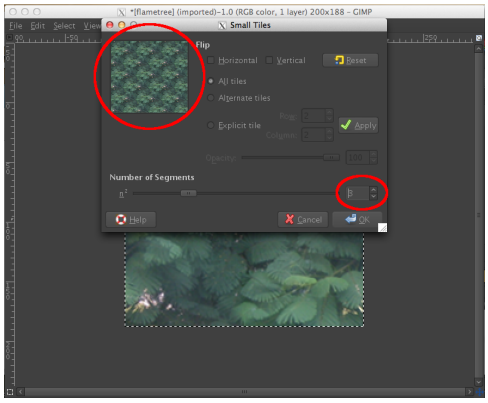
You can see that the resulting image is different than it was before. It tiles perfectly, by which we mean that the top and bottom edges match perfectly, as to the left and right edges.



We will use **Gimp Menu: Filters / Map / Small Tile** to see exactly how this might look as a background.



The number of segments defaults to 2. We bump it up to 3. We can see in the upper corner a view of how things would look. Although it tiles perfectly, it has the unfortunate result of creating a diamond-shaped repeating pattern.

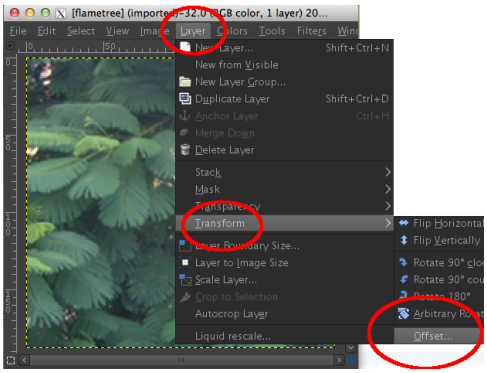


Bottom line: easy to do, but creates an obvious pattern. In any given instance, such a pattern may not be a problem. But at other times it may be annoying. You decide.

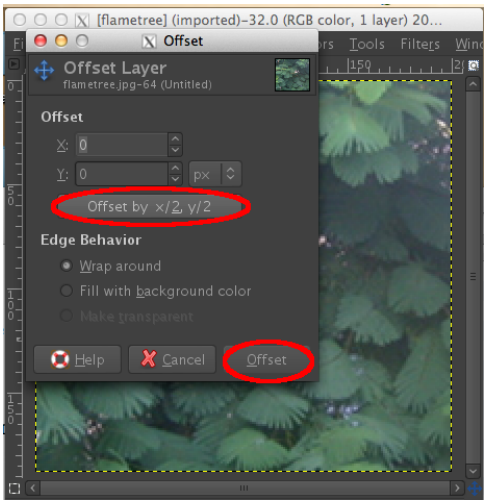
42.5 Example 2

We can also use the cloning tool to turn an image into a tile. It has the advantage that the repeated pattern is much less obvious. It has the disadvantage that it takes longer. It has the advantage that you have much more control over the finished product.

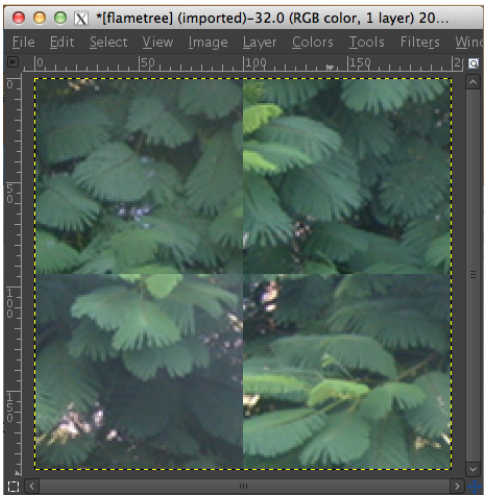
Starting over with our same cropped and scaled image from Example 1, we do **Gimp Menu: Layer / Transform / Offset**.



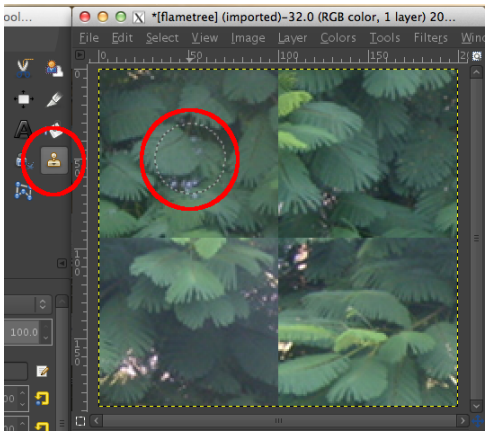
We choose to offset by x/2 and y/2, which means we move the image half way in the left-right dimension, and half way in the up-down dimension.



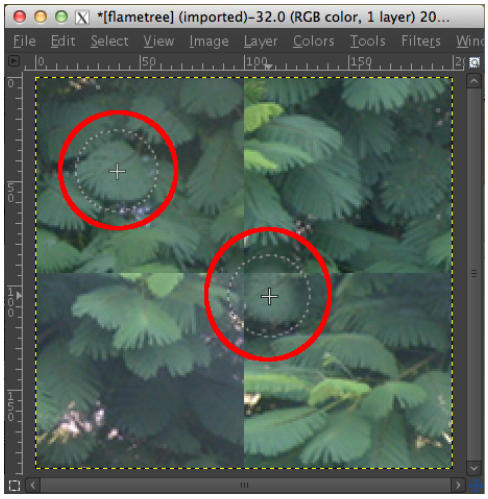
The four corners are now in the middle. The upper left and lower right corners have swapped places. The upper right and lower left have swapped places. All the tiling problems have been moved right to the middle where we will fix them manually.



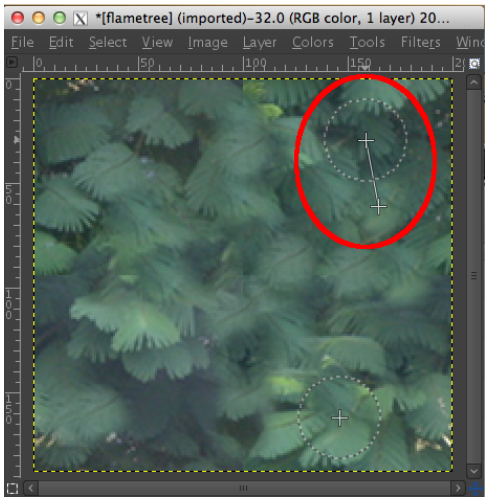
We select the clone tool. It looks like a rubber stamp. Double-click it to select it. Our cursor turns into a circle. You can adjust its size.



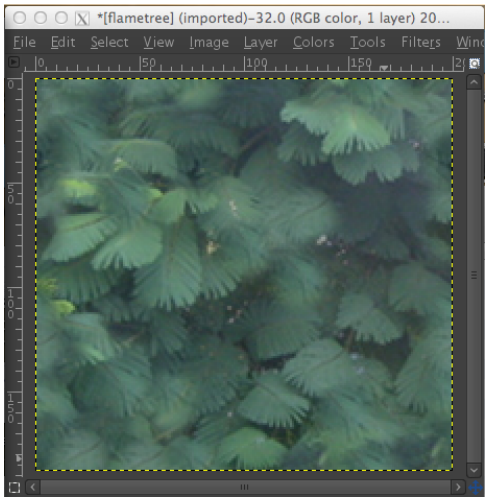
Hold down the CTRL (control) key and click on a section of the image that you would like to copy. We are copying from the upper left corner. Then move over something you want to hide and click (without using the control key). The image will be copied, but the edges will be faded so it looks smooth.



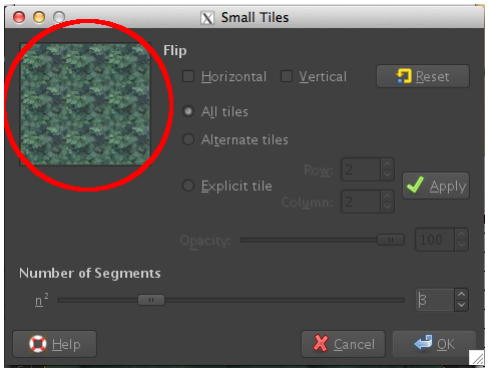
We repeat a bunch of times. CTRL click to select. Then click various places to paste. Then find something new to select. Then paste some more. Overwrite anything that looks like it would be distracting. Especially overwrite the line where the four corners come together.



Repeat the **Gimp Menu: Layer / Transform / Offset** step we did above. This will show us whether the edges still look good together. If necessary, do more cloning.



Use **Gimp Menu: Filters / Map / Small Tile** again to see exactly how this might look as a background. A pattern is still obvious. It is much less annoying to me than in the first example.



Using **Gimp Menu: Layer / Transform / Offset** and the clone tool gives you excellent control over your finished product, but takes longer. Your mileage may vary.

doctype

Sources:

http://www.w3.org/TR/2012/CR-html5-20121217/ provides the most authoritative information about version 5 of HTML as of 2012-12-17.

http://www.w3.org/TR/#tr_HTML provides a history of W3C documents about HTML5, and is the place where one could find whether a newer specification has been published.

Chapter 43

ch42 Elements of HTML

Contents

The **html** tag is the root element of an HTML document "tree." Within the html element, there will be a head element and a body element. You should have exactly one html element.

This is the overall structure of an HTML document.

```
<!DOCTYPE html>
<head> head things </head>
<body> body things </body>
```

43.1 The head Element

The **head** tag specifies the metadata about the webpage, and also includes the title element. You should have exactly one head element. The things that appear inside the head are called metadata, which means data about the webpage that follows.

This is the overall structure of the head element.

```
<head lang=en>
<title> this is the title </title>
<base href=something>
<meta ...>
<link ...>
<script ...></script>
</head>
```

The **title** tag comes within the head. You should have exactly one title element. If you bookmark a page, the browser will typically use the title (and favicon) to identify the bookmark, so the title should be helpfully descriptive when used as a bookmark.

```
<title>
IWDD Sample Webpage
</title>
```

The **base** tag comes within the head. It provides a basis for resolving relative URL references in the rest of the document. It defaults to the URL by which the page was originally loaded. You can have at most one base element.

```
<base href=http://example.com/2013/>
```

Base can be particularly handy if there is more than one URL that could lead to this same webpage. I have used it when I wanted to have a page temporarily become my homepage, but permanently be a different part of my website.

The **link** tag comes within the head. It provides linkage to resources that have their own URLs. Each link must specify an href (URL) and a rel (relationship).

Links includes things that will be used immediately like icons and style sheets. Links also include things like related webpages, such as author and license. (or next and back and up?)

The **style** tag can come within the head and provides CSS style information, or a link to such information. The style can be applied (selected) or not depending on the **media** attribute, and whether it matches the medium that is presenting the webpage, such as screen, printer, or audio.

The **script** tag can come within the head and provides an interactive element to the webpage. Typically scripting is done in JavaScript. Scripts respond to and process events, such as mouseover and keypress, as well as real-time webpage updating using things like AJAX.

Warning: Script can either specify a source file (src=) or immediately provide the JavaScript code. If it does both, the src= takes precedence and the inline code is ignored.

The **meta** tag comes within the head and provides metadata that is not otherwise available through title, base, link, style, script, and any other things that might have their own tags.

43.2 The body Element

Some tags are identified as semantic. The semantic tags normally have counterparts that are non-semantic. Semantic tags assert some additional, special, cohesive meaning: this content is not just a div, it is an article or a header or a footer.

The **body** tag (not semantic) identifies the body element, which is the webpage content. It also supports the event handlers (JavaScript) that affect the webpage content.

43.2.1 div and Friends

The **div** tag (not semantic) provides generic grouping of things. It does not imply the content within has a shared theme. When the content does have a shared theme, other tags like **article**, **section**, and **nav** may be more appropriate.

The **article** tag (semantic) is like **div**, but it identifies a composition that is perhaps like a blog post or a comment. The body may have any number of articles. Articles may have sub-articles within themselves.

The **section** tag (semantic) is like **div**, but it implies the content within shares the same theme. It is more general than **article**.

The **nav** tag (semantic) is like **div**, but it implies the content is generally there to provide links to related content. A screen-reader for the blind, for example, might skip over the nav section without reading it aloud, because

it is not the main content.

The **aside** tag (semantic) is like **div**, identifies "by the way" content that is a deeper view of something that is only tangential to the main flow of the webpage.

43.2.2 Headings

The **h1** tag (structural) identifies a top-level header.

The **h2** tag (structural) identifies a second-level header, and implies a child relationship to the previous **h1** tag.

Chapter 44

Target Skills, Basic Skills

Target Skills

So, what is typically expected of students who have passed an introductory course in website design and development? What would your friends and family expect of you?

(a) Help me set up a new website for a person or family or small business.

(b) Help me modify an existing website.

(c) Help me fix problems with an existing website. For example, why does it load so slowly?

(d) How do I buy a domain name?

(e) Do I need a security certificate for my website?

(f) How do I arrange for hosting?

(g) How do I back up my website?

(h) What about website "generators" for things like blogging?

These are the target skills, which we could also call exit skills. They are the skills that enable students to overcome their own web design problems and to solve website design problems for others.

Basic Skills

The target skills, mentioned above, are things you probably already intend to learn.

Some of the target skills can be memorized, but many cannot. Instead, they must be understood. Basic concepts come up like HTML, CSS, and JavaScript.

The basic skills and concepts are those other things you have to learn first before you can be truly proficient in the target skills. They are things that perhaps you didn't intend to learn. They may not be as glamorous. You may not know they even exist. But you don't score baskets in basketball (the target skill) unless you are also good at dribbling (the basic skill).

Chapter 45

Advanced Eye Candy

Contents

This chapter is based on student requests. How can I do this or that? Strictly speaking, the content is not your basic knowledge. It is a bit more advanced. Not hard, really. But not basic either.

CSS gives us great capabilities to do cool things. This chapter is a collection of fun things you can do with CSS.

45.1 Stop The Background From Scrolling

When a webpage is big enough, scroll bars appear and you can scroll up and down to see the hidden parts of the page.

Normally the background scrolls along with the page content. But if you want it to be fixed and not scroll, you can use this:

```
background-attachment: fixed;
```

The default is to scroll:

```
background-attachment: scroll;
```

45.2 Stretch The Background Image

Say you have a background image, but it is either too big or too small. You do not want it repeating. You just want it to take the full space available. The first number is width. The second number is height (length).

```
background-size: 100% auto;
```

If you are willing to have the aspect ratio go a bit crazy, you can do this:

```
background-size: 100% 100%;
```

There are a couple of special options for background size.

Cover: This scales the background image to be as large as possible without repeating. Some parts of the image will probably not be visible, depending on the shape of the image and the shape of the screen.

```
background-size: cover;
```

Contain: This scales the background image to be completely visible, but may involve some repeating, depending on the shape of the image and the shape of the screen.

```
background-size: contain;
```

45.3 Scroll Bar

If you are stretching the image to 100%, on pages with a scroll bar it will always look the same, and on pages without a scroll bar it will always look the same, but when you go from one to another, the image will change slightly.

```
overflow: scroll;
```

You can avoid this by using **overflow: scroll** to force the scroll bar to always be present.

45.4 Translucent Background

By "translucent" we mean partially transparent. You can see it, and you can also see through it.

You have a layer of text, and you want to be able to see through it to the background image. Opacity does not work because it changes background and text both. To change just the background, do this:

Tint (whiten) the background, and make it partly transparent.

```
background-color: rgba(255,255,255,0.5);
```

Shade (darken) the background, and make it partly transparent.

```
background-color: rgba(0,0,0,0.5);
```

The last number, in these examples 0.5, is the opacity. It ranges from 0.0 (totally transparent) to 1.0 (totally opaque).

This works well with a **div** that surrounds everything. For example:

```
<body style="background-attachment: fixed;
 background-image: url(abc.jpg);">
<div style="margin: 10px 20%; border: double;
 background color: rgba(255,255,255,0.5);">
Put your content here.
</div></body>
```

45.5 First Letter Different Style

Say you want the first letter of the first paragraph to be a fancy letter from a whole different font, like Old English. How can you do that?

We can use pseudo-classes to achieve that result.

todo: add more

45.6 Fancy Characters (Glyphs)

Besides the normal A to Z, 0 to 9, and punctuation, there are many other characters that can be used to dress up your webpage.

Section 10.7 (page 111) goes into more depth on these special character glyphs you can use.

These are not styling in the CSS way, but they are stylish. Here are a few. You can insert them directly into the HTML of your webpages. For example:

```
<h1>&spades;&clubs; Play Cards! &hearts;&diamonds;</h1>
```

Unit IV

Test Bank and Index

Chapter 46

Test Bank

This test bank includes all exam questions from throughout the textbook. They are collected together here as an aid to study. For answers, follow the link in each question to the place in the book where the question and answer are originally shown.

Test Bank

Unit I: Web Design 101

Unit I, Chapter 1: First Webpage

1: (p.8) What does HTML stand for?

2: (p.10) What does URL stand for?

3: (p.10) When we specify image width, does it affect displayed size?

4: (p.10) When we specify image width, does it affect download time?

5: (p.11) How many CSS pixels are in one degree of direct vision?

Unit I, Chapter 2: Content vs Markup

6: (p.15) What does deprecated mean?

7: (p.15) Is the center tag deprecated?

8: (p.15) Is the font tag deprecated?

9: (p.16) The HTML tag "h1" stands for what word(s)?

10: (p.16) The HTML tag "h2" stands for what word(s)?

11: (p.17) The HTML tag "p" stands for what word(s)?

12: (p.17) The HTML tag "br" stands for what word(s)?

13: (p.18) The HTML tag "b" stands for what word(s)?

14: (p.18) The HTML tag "strong" stands for what word(s)?

15: (p.19) The HTML tag "i" stands for what word(s)?

16: (p.19) The HTML tag "em" stands for what word(s)?

17: (p.19) The HTML tag "s" stands for what word(s)?

18: (p.19) The HTML tag "u" stands for what word(s)?

19: (p.20) The HTML tag "sub" stands for what word(s)?

20: (p.20) The HTML tag "sup" stands for what word(s)?

21: (p.20) The HTML tag "img" stands for what word(s)?

22: (p.21) The HTML tag "a" stands for what word(s)?

23: (p.22) What character sequence marks the start of a comment in HTML?

24: (p.22) What character sequence marks the end of a comment in HTML?

25: (p.22) What character sequence is not allowed within a comment in

HTML5?

26: (p.22) In HTML, can comments be nested?

Unit I, Chapter 3: Head Sets The Stage

27: (p.27) What is the first markup element of every HTML5 webpage?

28: (p.27) What is the second markup element of every HTML5 webpage?

29: (p.28) What HTML markup is used to specify our character set?

30: (p.28) What HTML markup is used to specify the title of our webpage?

31: (p.28) What HTML markup is used to create an internal style sheet?

32: (p.28) What HTML markup is used to identify JavaScript?

33: (p.29) What HTML markup is used to mark off the content of the document?

34: (p.30) For SEO, is title very important?

35: (p.30) For SEO, about how many characters of our title are used?

36: (p.30) For SEO, is meta description very important?

37: (p.31) Give the prototype (pattern) for meta description.

38: (p.31) For SEO, about how many characters of meta description are used?

39: (p.31) For SEO, are meta keywords very important?

40: (p.34) What HTML markup is used to specify a top level header?

41: (p.34) What HTML markup is used to specify a second-level heading?

42: (p.34) What HTML markup is used to start a new paragraph?

43: (p.34) What HTML markup is used to present bold content?

44: (p.34) What HTML markup is used to present oblique (italic) content?

45: (p.34) What HTML markup is used to emphasize some content?

46: (p.34) What HTML markup is used to draw a line through some content?

47: (p.34) What HTML markup is used to draw a line under some content?

48: (p.34) What HTML markup is used to present lowered content (Like the 2 in H2O)?

49: (p.35) What HTML markup is used to present raised content (Like the 2 in x squared)?

50: (p.35) What HTML markup is used to go to a new line?

51: (p.35) What HTML markup is used to insert a picture?

52: (p.35) What HTML markup is used to specify a link to another page?

Unit I, Chapter 4: HTML Tags and Attributes

53: (p.38) Give the prototype (pattern) for HTML markup in general.

54: (p.39) What character marks the start of each piece of HTML markup?

55: (p.39) What character marks the end of each piece of HTML markup?

56: (p.39) What two characters can mark the end of HTML markup for a void tag?

57: (p.39) When a tag is void, what does that mean?

58: (p.39) When do you need a space before the end of HTML markup?

59: (p.40) When one set of tags begins and ends totally inside another set of tags, what do we call that?

60: (p.40) What is wrong with this ordering: <a>.........?

61: (p.40) What is wrong with this ordering: <a>.........?

62: (p.41) In HTML does the order in which attributes are specified make

any difference?

63: (p.42) If an HTML attribute's value includes a space does that force it to be quote marked?

64: (p.42) If an HTML attribute's value includes a double-quote (") does that force it to be quote marked?

65: (p.43) If an HTML attribute's value includes a less-than (<) does that force it to be quote marked?

66: (p.43) If an HTML attribute's value includes a greater-than (>) does that force it to be quote marked?

67: (p.43) If an HTML attribute's value includes a single-quote (apostrophe) (') does that force it to be quote marked?

68: (p.43) If an HTML attribute's value includes a back-quote (`) does that force it to be quote marked?

69: (p.43) If an HTML attribute's value includes an equals (=) does that force it to be quote marked?

70: (p.43) If an HTML attribute's value includes a letter (A a B b ...) does that force it to be quote marked?

71: (p.43) If an HTML attribute's value includes a digit (1 2 3 ...) does that force it to be quote marked?

72: (p.44) If an HTML attribute's value includes a colon (:) does that force it to be quote marked?

73: (p.44) If an HTML attribute's value includes a dot (.) does that force it to be quote marked?

74: (p.44) If an HTML attribute's value includes a dash (-) does that force it to be quote marked?

75: (p.44) If an HTML attribute's value includes a slash (/) does that force it to be quote marked?

76: (p.44) If an HTML attribute's value includes a percent (%) does that force it to be quote marked?

77: (p.44) If an HTML attribute's value includes an ampersand (`&`) does that force it to be quote marked?

78: (p.45) In what is the value of width?

79: (p.45) In what is the value of width?

80: (p.45) In what is the value of width?

81: (p.45) List the four commonly-used global attributes for HTML tags.

82: (p.46) In HTML, what type of character can be first in an id?

83: (p.46) In HTML, what six kinds of characters can appear after the first in an id?

84: (p.46) In HTML, is id=123 valid?

85: (p.46) In HTML, is id=abc valid?

86: (p.46) In HTML, is id=a.b.c valid?

87: (p.46) In HTML, is id=a-b-c valid?

88: (p.46) In HTML, is id="a b c" valid?

89: (p.47) What does the <title> tag do?

90: (p.47) What does the "title=" attribute do?

Unit I, Chapter 5: Image Crop and Resize

91: (p.49) The HTML tag "img" stands for what word(s)?

92: (p.49) What HTML markup is used to insert a picture?

93: (p.49) For each <img> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

94: (p.50) For each <img> tag in valid HTML5, is the "alt=" attribute required, optional, or forbidden?

95: (p.50) For each <img> tag in valid HTML5, is the "width=" attribute required, optional, or forbidden?

96: (p.50) List the four commonly-used attributes of the image tag.

97: (p.51) In image processing, what word describes removal of part of the picture, making it smaller?

98: (p.51) In image processing, what word describes combining pixels, making the image smaller?

99: (p.51) In image processing, what word describes dividing pixels, making the image larger?

100: (p.58) How can you show a small image that when clicked on shows a larger image?

Unit I, Chapter 6: Using External Resources

101: (p.61) Are spaces allowed in filenames?

102: (p.61) What does %20 mean in a URL?

103: (p.61) We identified three kinds of URLs. One is relative. List the other two.

104: (p.62) To open a link in a new tab, what HTML attribute=value should you use?

105: (p.63) Is the domain name within a URL case-sensitive?

106: (p.63) Is the path within a URL case-sensitive?

107: (p.68) If our base URL is http://a.com/b/c/ and our stated URL is ../d/e/ what is our final URL?

108: (p.68) If our base URL is http://a.com/b/c/ and our stated URL is ./d/e/ what is our final URL?

109: (p.68) If our base URL is http://a.com/b/c/ and our stated URL is /d/e/ what is our final URL?

110: (p.68) If our base URL is http://a.com/b/c/ and our stated URL is d/e/ what is our final URL?

Unit I, Chapter 7: Colors

111: (p.71) What CSS attribute: sets the color of your lettering?

112: (p.71) What CSS attribute: sets the color behind your lettering?

113: (p.73) In rgb, what does the r stand for?

114: (p.73) In rgb, what does the g stand for?

115: (p.73) In rgb, what does the b stand for?

116: (p.74) In rgba, what does the a stand for?

117: (p.74) What does the alpha channel control?

118: (p.75) What color is named for a German botanist?

119: (p.75) What is color `#ffffff` officially named?

120: (p.75) What is color `#c0c0c0` officially named?

121: (p.75) What is color `#808080` officially named?

122: (p.75) What is color `#000000` officially named?

123: (p.75) What is color `#ff0000` officially named?

124: (p.76) What is color `#800000` officially named?

125: (p.76) What is color `#00ff00` officially named?

126: (p.76) What is color `#008000` officially named?

127: (p.76) What is color `#0000ff` officially named?

128: (p.76) What is color `#000080` officially named?

129: (p.76) What is color `#ffff00` officially named?

130: (p.76) What is color `#808000` officially named?

131: (p.76) What is color `#ffa500` officially named?

132: (p.76) What is color `#ff00ff` officially named?

133: (p.76) What is color `#800080` officially named?

134: (p.76) What is color `#00ffff` officially named?

135: (p.76) What is color `#008080` officially named?

136: (p.77) Gradient is an example of what CSS attribute?

137: (p.77) List the two types of gradients.

Unit I, Chapter 8: Image Transparency

138: (p.82) Does the .gif image file format support transparency?

139: (p.82) Does the .jpg image file format support transparency?

140: (p.82) Does the .png image file format support transparency?

141: (p.86) What does favicon stand for?

142: (p.88) For each <head> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

143: (p.88) For each <body> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

144: (p.88) For each <h1> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

145: (p.88) For each <h2> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

146: (p.88) For each <p> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

147: (p.88) For each <b> tag in valid HTML5, is a separate closing tag

required, optional, or forbidden?

148: (p.88) For each <i> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

149: (p.88) For each tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

150: (p.88) For each <s> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

151: (p.88) For each <u> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

152: (p.88) For each <sub> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

153: (p.89) For each <sup> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

154: (p.89) For each <br> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

155: (p.89) For each <img> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

156: (p.89) For each <img> tag in valid HTML5, is the "width=" attribute required, optional, or forbidden?

157: (p.89) For each <img> tag in valid HTML5, is the "alt=" attribute required, optional, or forbidden?

158: (p.89) For each <a> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

159: (p.89) For each tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

160: (p.89) For each tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

Unit I, Chapter 9: CSS: The Style Sheet

161: (p.91) What does CSS stand for?

162: (p.91) Give the prototype for CSS directives.

163: (p.92) What CSS attribute: sets the color behind your lettering?

164: (p.92) What CSS attribute: sets the color of your lettering?

165: (p.93) What CSS attribute: sets the size of your lettering?

166: (p.93) What CSS attribute: sets the thickness of your lettering?

167: (p.94) What CSS attribute: sets the slant of your lettering?

168: (p.94) List the four "text-align:" options.

169: (p.95) What CSS attribute: creates a shadow behind your lettering?

170: (p.96) What marks the start of a comment in CSS?

171: (p.96) What marks the end of a comment in CSS?

172: (p.96) In CSS can comments be nested?

173: (p.96) In HTML the tag usually does the same as what other tag?

174: (p.97) In HTML the tag usually does the same as what other tag?

Unit I, Chapter 10: Font Families

175: (p.100) In CSS what is the one-character symbol for class?

176: (p.100) What is the simple rule for creating a legal class name?

177: (p.102) What kind of font represents shapes as pixels?

178: (p.102) What kind of font represents shapes as strokes?

179: (p.102) What does kerning control?

180: (p.103) List the five generic font families.

181: (p.104) What generic font family includes Times?

182: (p.104) What generic font family includes Helvetica?

183: (p.105) What generic font family has letters all the same width?

184: (p.105) What generic font family includes Courier?

185: (p.105) Which generic font should almost always be text aligned left?

186: (p.105) What generic font family emulates human handwriting?

187: (p.107) Tell what is wrong with this font spec: font-family: Times Roman, serif;

188: (p.107) Tell what is wrong with this font spec: font-family: ARiaL, sans-serif;

189: (p.107) Tell what is wrong with this font spec: font-family: arial, sans-serif;

190: (p.107) Tell what is wrong with this font spec: font-family: sans-serif, arial;

191: (p.107) Tell what is wrong with this font spec: font-family: arial, sans-serif, serif;

192: (p.108) In font names, does capitalization matter?

193: (p.108) In font family specs, what font should always be included?

194: (p.108) In font family specs, what font should always be listed last?

195: (p.112) What HTML character entity reference is for a space without letting the line split?

196: (p.112) What HTML character entity reference is for the ampersand (and) symbol?

197: (p.112) What HTML character entity reference is for the less-than symbol?

198: (p.112) What HTML character entity reference is for the greater-than symbol?

199: (p.112) What HTML character entity reference is for a double-quote?

200: (p.112) What HTML character entity reference is for a single-quote?

201: (p.112) What HTML character entity reference is for the (C) copyright symbol?

202: (p.113) What HTML character entity reference is for the (R) registered-trademark symbol?

203: (p.113) What HTML character entity reference is for the TM nonregistered-trademark symbol?

Unit I, Chapter 11: Hover and Pseudo Classes

204: (p.116) What does pseudo mean?

205: (p.116) What is the prefix for pseudo-class?

206: (p.116) What is the pseudo-class for a visited link?

207: (p.116) What is the pseudo-class for an unvisited link?

208: (p.117) What is the pseudo-element for the first character of something?

209: (p.117) What is the pseudo-element for the first row of text of something?

210: (p.117) What is the prefix for pseudo-element?

211: (p.119) If two CSS specs apply to the same item, which takes precedence, the first specified or the last specified?

212: (p.119) What pseudo-class targets an item when you move the mouse across it?

213: (p.120) What pseudo-class targets the input field where typed text would be entered?

214: (p.120) What attribute lets you make styling changes smoothly instead of instantly?

Unit I, Chapter 12: Box Model

215: (p.127) List the three major components of the box model.

216: (p.128) Does padding share the same background as the content?

217: (p.128) Can padding style be different on each of the four sides?

218: (p.128) Can "padding:" be negative?

219: (p.129) Does border share the same background as the content?

220: (p.129) Can border style be different on each of the four sides?

221: (p.129) Besides none and hidden, list the other eight "border-style:" options.

222: (p.130) Does margin share the same background as the content?

223: (p.130) Can margin style be different on each of the four sides?

224: (p.130) Can "margin:" be negative?

225: (p.130) Which box model parameter is used for temporary highlighting?

226: (p.130) Which box model parameter has invert as a color option?

227: (p.130) Can outline style be different on each of the four sides?

228: (p.131) Which box model parameter creates rounded corners?

229: (p.132) When we say margin: 9px 8px 7px 6px; what is the value of margin-top?

230: (p.132) When we say margin: 9px 8px 7px 6px; what is the value of margin-right?

231: (p.132) When we say margin: 9px 8px 7px 6px; what is the value of margin-bottom?

232: (p.132) When we say margin: 9px 8px 7px 6px; what is the value of margin-left?

233: (p.132) When we say margin: 9px 8px 7px; what is the implied fourth value?

234: (p.132) When we say margin: 9px 8px; what is the implied third value?

235: (p.133) When we say margin: 9px 8px; what is the implied fourth value?

236: (p.133) List the two "box-sizing:" options.

237: (p.133) What "box-sizing:" option is the default?

238: (p.133) How many CSS pixels are in one degree of direct vision?

239: (p.134) Typically how many px are there per inch?

240: (p.134) How many pt are there per inch?

241: (p.134) What does pt stand for?

242: (p.134) How many pc are there per inch?

243: (p.134) What does pc stand for?

244: (p.135) What CSS measurement unit is based on how wide the browser window is?

245: (p.135) What CSS measurement unit is based on how tall the browser window is?

246: (p.137) List the two "float:" options.

247: (p.137) List the three "clear:" options.

Unit I, Chapter 13: Positioning: Relative and Absolute

248: (p.139) List the four CSS position: options.

249: (p.139) List the four CSS positioning directions.

250: (p.139) If we say left: 5px; which direction will things move? (left or right)

251: (p.139) If we say left: -5px; which direction will things move? (left or right)

252: (p.140) If we say right: 5px; which direction will things move? (left or right)

253: (p.140) If we say right: -5px; which direction will things move? (left or right)

254: (p.140) If we say top: 5px; which direction will things move? (up or down)

255: (p.140) If we say top: -5px; which direction will things move? (up or down)

256: (p.140) If we say bottom: 5px; which direction will things move? (up or down)

257: (p.140) If we say bottom: -5px; which direction will things move? (up or down)

258: (p.140) List the two CSS positions that retain the element in the flow.

259: (p.140) Which CSS position is the default?

260: (p.141) List the two CSS positions that remove an element from the flow.

261: (p.141) Which CSS position depends on its closest positioned ancestor?

262: (p.142) Which CSS position depends on the viewport?

263: (p.143) What attribute specifies stacking order for overlap?

Unit I, Chapter 14: JavaScript and AJAX

264: (p.150) What does DOM stand for?

265: (p.151) Convert border-top-right-radius to camel case.

266: (p.151) What HTML attribute= is used to run a JavaScript action when the user single-clicks on the specified item?

267: (p.151) What HTML attribute= is used to run a JavaScript action when the user double-clicks on the specified item?

268: (p.152) What HTML attribute= is used to run a JavaScript action when the mouse enters the specified item?

269: (p.152) What HTML attribute= is used to run a JavaScript action when the mouse leaves the specified item?

270: (p.152) Are Java and JavaScript the same thing?

271: (p.152) Are Java and JavaScript related?

Unit I, Chapter 15: Lists and Menus

272: (p.157) The HTML tag "ol" stands for what word(s)?

273: (p.157) For each <ol> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

274: (p.157) The HTML tag "ul" stands for what word(s)?

275: (p.157) For each <ul> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

276: (p.157) The HTML tag "li" stands for what word(s)?

277: (p.157) For each <li> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

278: (p.157) Can lists be nested (a list inside a list)?

279: (p.158) For numbered lists, are there more than 10 numbering systems available?

280: (p.158) What CSS list attribute controls the numbering system?

281: (p.158) For numbered lists, can we specify the starting value?

282: (p.158) For numbered lists, can we count down instead of up?

283: (p.159) What attribute overrides the number of an li element?

284: (p.160) What styling attribute makes a list horizontal?

285: (p.161) What styling attribute makes an element disappear?

286: (p.161) What styling attribute makes an element (re)appear?

Unit I, Chapter 16: Tables

287: (p.163) For each <table> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

288: (p.163) The HTML tag "tr" stands for what word(s)?

289: (p.163) For each <tr> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

290: (p.163) The HTML tag "th" stands for what word(s)?

291: (p.163) For each <th> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

292: (p.163) The HTML tag "td" stands for what word(s)?

293: (p.163) For each <td> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

294: (p.163) Can tables be nested (a table inside a table)?

295: (p.164) Should tables be used for the layout of non-tabular material?

296: (p.165) Is the table border= attribute deprecated?

297: (p.165) What HTML attribute= is used to merge table cells horizontally?

298: (p.165) What HTML attribute= is used to merge table cells vertically?

299: (p.165) What CSS attribute lets us rotate something?

300: (p.166) Does max-width: normally work with table cells?

301: (p.166) List the two "table-layout:" options.

302: (p.167) What CSS selector targets even numbered rows?

303: (p.167) What CSS selector targets odd numbered rows?

304: (p.170) What CSS "attribute: value;" keeps cell contents on one line?

305: (p.170) What CSS "attribute: value;" clips cell content that does not fit inside the cell?

Unit I, Chapter 17: Forms and Inputs

306: (p.173) For each <form> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

307: (p.174) List the three form input tags.

308: (p.175) For each <input> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

309: (p.175) List the four button type= attributes for the input tag.

310: (p.176) List the six old-school non-button type= attributes for the input tag.

311: (p.180) For each <textarea> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

312: (p.180) For each <select> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

313: (p.180) For each <option> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

314: (p.180) What HTML attribute= is used to start the cursor in a certain input field?

315: (p.181) What HTML attribute= is used to specify the order in which fields are reached by tabbing?

Unit I, Chapter 18: Responsive Web Design (RWD)

316: (p.186) What does RWD stand for?

317: (p.187) Explain the "mobile first" philosophy.

318: (p.187) Shadows, gradients, and transformations should be limited by media queries. (yes/no)

319: (p.188) What does MQ stand for?

320: (p.188) List the two main media types that can be specified in media queries.

321: (p.189) List the four most important media query viewport attributes.

322: (p.189) List the two media query orientations.

323: (p.194) What HTML markup is used to specify a vertical block?

324: (p.195) What HTML markup is used to mark off a horizontal block of characters?

325: (p.195) In HTML what does the "span" tag stand for?

326: (p.195) Can a div legally appear within another div?

327: (p.195) Can a div legally appear within a p?

328: (p.195) Can a div legally appear within a span?

329: (p.195) Can a p legally appear within a div?

330: (p.196) Can a p legally appear within another p?

331: (p.196) Can a p legally appear within a span?

332: (p.196) Can a span legally appear within a div?

333: (p.196) Can a span legally appear within a p?

334: (p.196) Can a span legally appear within another span?

Unit I, Chapter 19: First Website

335: (p.202) Can you really own a domain name?

336: (p.202) What is the typical cost for a .com domain name, in USD per year?

337: (p.203) What does TLD stand for?

338: (p.204) Who or what owns byuh.doncolton.com?

339: (p.207) What is the default folder name for webpages on an Apache server?

340: (p.210) What is the default file name for webpages on an Apache server?

341: (p.211) What word describes the copying of files from our personal computer to the server that is hosting our website?

342: (p.211) What word describes the copying of files from the server that is hosting our website to our personal computer?

Unit II: Appendix

Unit II, Chapter 20: Glossary of Terms

Unit II, Chapter 21: Working Offline

343: (p.221) What word means right at your computer instead of somewhere

out on the Internet?

344: (p.222) What word means somewhere out on the Internet instead of right at your computer?

345: (p.225) Where does your browser store files it might need later?

346: (p.225) If you change a webpage, but when you visit it, it has not changed, what is probably the cause?

347: (p.226) If you change a webpage, but when you visit it, it has not changed, what should you do?

Unit II, Chapter 22: Browser Recommendations

Unit II, Chapter 23: Text Editor Recommendations

Unit II, Chapter 24: Password Recommendations

348: (p.234) What is Moore's Law?

Unit II, Chapter 25: Copyright and Intellectual Property

349: (p.239) Can ideas be copyrighted?

350: (p.240) What is Fair Use?

351: (p.240) What is a Cover Version?

352: (p.240) What is a Public Domain?

353: (p.241) What is a Derivative Work?

354: (p.242) What is a Take-Down notice?

355: (p.242) What is a Safe Harbor?

356: (p.242) What does DMCA stand for?

357: (p.243) What is lorem ipsum?

358: (p.243) What fancy name do we use to describe fake text that is inserted into a webpage just to give it some body?

Unit II, Chapter 26: The Gimp Image Editor

Unit II, Chapter 27: Gimp Makes A Logo

Unit III: Under Development

Unit III, Chapter 28: Styling With CSS

359: (p.256) For each <meta> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

360: (p.256) What HTML markup is used to specify the name of the webpage?

361: (p.256) For each <title> tag in valid HTML5, is a separate closing tag required, optional, or forbidden?

362: (p.256) What CSS attribute: places a picture in the background?

363: (p.257) What CSS attribute: draws a box around some content?

364: (p.257) What CSS attribute: puts space between any box you draw around content, and neighboring content?

365: (p.257) In CSS what is the one-character symbol for class?

366: (p.257) What HTML attribute= is used to specify class?

367: (p.257) What CSS attribute places a picture in the background?

368: (p.257) If you want to draw a box around some content, what CSS attribute would you use?

369: (p.257) If you draw a box around some content, what CSS attribute puts space between your content and the box?

370: (p.257) In CSS, can padding be negative?

371: (p.257) In CSS, does background extend into padding?

372: (p.257) In CSS, does background extend into borders?

373: (p.257) In CSS, does background extend into margins?

374: (p.258) If you draw a box around some content, what CSS attribute puts space between the box and neighboring content?

375: (p.258) In CSS, can margin be negative?

376: (p.262) What CSS attribute: makes an element partially transparent?

377: (p.262) What CSS attribute: sets the width of a block item?

378: (p.262) What CSS attribute: sets the largest allowed width of a block item?

379: (p.262) What CSS attribute: sets the smallest allowed width of a block item?

380: (p.262) What CSS attribute: sets the height of a block item?

381: (p.262) What CSS attribute: sets the largest allowed height of a block item?

382: (p.262) What CSS attribute: sets the smallest allowed height of a block item?

383: (p.262) What CSS attribute: starts flow-around (of text around pictures)?

384: (p.262) What CSS attribute: stops the flow-around action?

Unit III, Chapter 29: Dot TK: Domains for Free

Unit III, Chapter 30: ID

385: (p.272) In CSS what is the one-character symbol for ID?

Unit III, Chapter 31: Backgrounds

Unit III, Chapter 32: WordPress and CMS

386: (p.278) What does CMS stand for?

387: (p.279) What does blog stand for? (two words)

Unit III, Chapter 33: Audio and Video

Unit III, Chapter 34: Untangling HTML and Style

Unit III, Chapter 35: ch31 CSS Selectors

Unit III, Chapter 36: ch32 Untangling HTML and Scripting

Unit III, Chapter 37: ch34 Getting Feedback

Unit III, Chapter 38: ch35 Content, Presentation, Action

Unit III, Chapter 39: ch36 Notable Resources

Unit III, Chapter 40: ch37 A General Overview

388: (p.309) What does WYSIWYG stand for?

Unit III, Chapter 41: ch38 Codecs: Coding and Decoding

Unit III, Chapter 42: ch39 Tiled Backgrounds

Unit III, Chapter 43: ch42 Elements of HTML

Unit III, Chapter 44: Target Skills, Basic Skills

Unit III, Chapter 45: Advanced Eye Candy

Unit IV: Test Bank and Index

Unit IV, Chapter 46: Test Bank

Index