Introduction to Website Design and Development: HTML, CSS, and JavaScript

Don Colton
Brigham Young University–Hawai‘i

August 10, 2013

Quick Overview

What does it really take to make a website? We are here to take you "behind the scenes" into the inner workings of websites.

This book is for people who are new to website design. It is pretty basic, but it also covers a lot of ground and introduces other resources. You can get the PDF for free at `http://iwdd.tk/` or `http://iwdd.doncolton.com/`.

Quick Start: We make a simple webpage, about five lines long, right on the computer you are using. Then we talk about domain names and webhosting. We follow that with control panels, to get your website up, and copyright, to avoid getting your website taken down. Finally, we publish.

Web Development Tools: Face it: you do not always have to reinvent the wheel. Many websites are built using cookie-cutter technologies. In minutes you can have your idea (or your friend's) up and running. We focus on WordPress, the leading CMS in the world.

Media: Text is great, but what is a webpage without pictures? We give you basic image editing skills. And we look into audio and video.

Content: HTML: Here we pop open the hood and look at the engine that runs the web: HTML. Good news: it is a pretty simple language.

Presentation: CSS Style: This is where the real power lies. Adding CSS to HTML is like turning a typewriter into a word processor.

Action: JavaScript: You learn what it takes to put JavaScript into a webpage, and why you might want to.

Appendix: Sometimes we go deep on things, like browsers, text editors, passwords, and more. For those that want deep, enjoy!

Ready to learn? Skip ahead to Chapter 4 (page 20).

Otherwise, stick around to see the table of contents. It is coming right up.

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 stylesheets 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 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 Systems.)

Acknowledgements

I am pleased to acknowledge and express my thanks for several 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.

Professor Jay Merryweather (http://jaymerryweather.com/, MWXDesign) of the Brigham Young University–Hawaii Fine Arts department provided a detailed list of learning outcomes expected from this course in its role as prerequisite to the Web Design courses taught in the Fine Arts department.

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

Contents

Unit I

Quick Start

Web design is a mix of fun and details. The details are not really interesting until you have had some fun. So we will start with some fun. Then we will follow up with some details, once you know why they are important to you.

Chapter 1

HTML Tags

Here is a webpage. Key it in. Then we will discuss it and customize it.

```
<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.
<img src=don.jpg>
```

`<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 the 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.

Exam Question 1 (p.297)**:**
In HTML what tag is used to identify a level-1 header?

Required Answer:
h1

Exam Question 2 (p.297)**:**
In HTML what words does the h1 tag stand for?

Required Answer:
header one

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.

Exam Question 3 (p.297)**:**
In HTML what tag is used to present bold content?

Required Answer:
b

Exam Question 4 (p.297)**:**
In HTML what word does the b tag stand for?

Required Answer:
bold

`<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.

Exam Question 5 (p.297)**:**
In HTML what tag is used to present oblique (italic) content?

Required Answer:
i

Exam Question 6 (p.297)**:**
In HTML what word does the i tag stand for?

Required Answer:
italics

`<img...>` is the image tag. It designates 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 have 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.

Exam Question 7 (p.297)**:**
In HTML what tag is used to insert a picture?

Required Answer:
img

Exam Question 8 (p.298)**:**
In HTML what word does the img tag stand for?

Required Answer:
image

img

1.1 Seeing Your Webpage

To share your webpage, you must put it on a web server. Your webpage must have a "name" by which people can request it. That name is called its URL.

Exam Question 9 (p.298)**:**
What words does URL stand for?

Required Answer:
uniform resource locator

The language used for writing webpages is called HTML.

Exam Question 10 (p.298)**:**
What words does HTML stand for?

Required Answer:
hyper-text markup language

Chapter 2

Valid HTML

We were playing fast and loose with our first webpage. Strictly speaking, it was not valid HTML. There are formal rules for HTML, and when we follow those rules, our webpage should work with all browsers in pretty much the same way. When we do not follow those rules, the browsers are free to guess what we meant, and they may guess differently from each other.

It is a good idea to write valid HTML code.

2.1 Head and Body

We will put our webpage in context by wrapping it in body tags and putting a head in front of it. Like this:

```
<!DOCTYPE html>
<head lang=en>
<meta charset=utf-8 />
<title>Our Title</title>
</head>
<body>

your webpage goes here

</body>
```

The `<!DOCTYPE html>` line tells the browser that we are writting according to the HTML5 standard.

The `<head...>` and `</head>` tags surround the head of our webpage.

The `<head lang=en>` line tells the browser that the following lines are the head of the document, until we encounter `</head>` which ends the head. It also says that this document is presented in English.

The `<meta charset=utf-8 />` line tells the browser that the character set used with this document is UTF-8. **UTF-8** stands for UCS Transformation Format, 8 bit. It is the currently accepted standard.

The `<title>` and `</title>` tags surround the title of our webpage.

The `<body>` and `</body>` tags surround the body of our webpage.

2.2 Validation

We will add some lines for validation purposes. Put these right before the `</body>` tag.

```
<p>Validate:
<a href="http://validator.w3.org/check?uri=referer">HTML</a>
<a href="http://jigsaw.w3.org/css-validator/check/referer">CSS</a>
</p>
```

After saving your webpage, you should be able to load it, view it, and see that it is almost the same as before. When you click on the HTML link, the HTML validator will examing your code and tell you whether it is correct. (We will validate the CSS in our next activity.)

Exam Question 11 (p.298)**:**
In HTML what tag is used to identify a paragraph?

Required Answer:
p

Exam Question 12 (p.298)**:**
In HTML what word does the p tag stand for?

Required Answer:
paragraph

Chapter 3

Classes and Borders

Plain text is boring. Let's dress it up. We will create a style sheet, and we will apply the styles to our content.

3.1 Internal Style Sheet

Inside the head section of your webpage, add a style section, as follows:

```
<style type=text/css>
body { background-color: yellow; }
.c1 { border: solid blue; }
.c2 { border: double red; }
</style>
```

The `<style...>` and `</style>` tags identify the style section. Put it someplace between your `<head...>` and `</head>` tags.

The material inside the style section is called a style sheet. Specifically, this is 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.

Exam Question 13 (p.298)**:**
What words does CSS stand for?

Required Answer:
cascading style sheet

In the style sheet, the body line specifies that for that part of your webpage surrounded by body tags (which is the whole webpage), a certain style is designated. This style is to make the background color yellow. The styling is surrounded by curly braces.

There are many things we can control with CSS. Background color is just one of them. We can also put borders around things.

The .c1 line specifies that for a class named c1, there will be a solid blue border line around that content.

The .c2 line specifies that for a class named c2, there will be a double red border line around that content.

3.2 Apply The Styles

In our document, we will put a couple of paragraphs. We will designate a class for each paragraph. These lines go in the body of our webpage.

```
<p>This is a normal paragraph.</p>
<p class=c1>This is a class c1 paragraph.</p>
<p>This is a normal paragraph.</p>
<p class=c2>This is a class c2 paragraph.</p>
<p>This is a normal paragraph.</p>
```

Include the CSS validator link in your webpage. Validate your webpage.

Exam Question 14 (p.298)**:**
In CSS what is the one-character symbol for class?

Required Answer:
.

Exam Question 15 (p.298)**:**
What HTML attribute specifies class?

Required Answer:
class=

3.3 Here Are Some Styles

Exam Question 16 (p.298)**:**
What CSS attribute sets the size of your text font?

Required Answer:
font-size:

Exam Question 17 (p.298)**:**
What CSS attribute sets the foreground color of your text font?

Required Answer:
color:

Exam Question 18 (p.298)**:**
What CSS attribute sets the color of the background behind your text?

Required Answer:
background-color:

Exam Question 19 (p.298)**:**
What CSS attribute places a picture in the background?

Required Answer:
background-image:

Exam Question 20 (p.298)**:**
If you want to draw a box around some content, what CSS attribute would you use?

Required Answer:
border:

Exam Question 21 (p.298)**:**
If you draw a box around some content, what CSS attribute puts space between your content and the box?

Required Answer:
padding:

Exam Question 22 (p.298)**:**
If you draw a box around some content, what CSS attribute puts space between the box and neighboring content?

Required Answer:
margin:

Unit II

Fundamentals

Chapter 4

Local Webpages

Contents

In this unit, we give you the bare fundamentals necessary to create a webpage with a few simple elements. (We also define a few terms that will be used throughout the book.) Then, based on that foundation, we will step back and take a broader view of our subject.

4.1 Local Webpages

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.

But first we are going 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 23 (p.299)**:**
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 24 (p.299)**:**
What word means somewhere out on the Internet instead of right at your computer?

Acceptable Answer:
online

Exam Questions appear throughout the book. They identify things that I might ask on an exam. At the back of the book, Appendix P (page 297) is a Test Bank of these questions. For more about the test bank, see Section D (page 251).

We will start by building a simple webpage, offline.

4.2 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 N (page 287) at the end of this book, right before the index.

4.3 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 E (page 253).

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.

For serious testing of your webpages, you should view them in all five of the major browsers.

4.4 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.

4.5 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 Microsoft, **Notepad++** (free) is our recommendation.

For Apple OS X, **TextWrangler** (free) is our recommendation.

For Linux, **gedit** (free) is our recommendation.

For a more detailed look with additional options, including links for downloading, please see Appendix F (page 255).

4.6 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`

4.7 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.

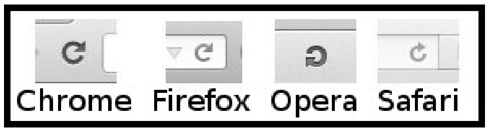
Edit your file. Add some markup. Then save your file. Follow this example, but feel free to use your own words.

```
<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.

4.8 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 25 (p.299)**:**
Where does your browser store files it might need later?

Required Answer:
cache

Exam Question 26 (p.299)**:**
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 27 (p.299)**:**
If you change a webpage, but when you visit it, it has not changed, what should you do?

Acceptable Answer:
reload or refresh

4.9 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 **leaching**. See section I.10.4 (page 273) 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.)

4.10 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 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 />`.

4.11 Centering

We will go much deeper into HTML in Unit V (page 121), but there are a few questions that often come up right about now, and this is a good time to answer them.

Centering is a popular request. How do I center my webpage?

There are a couple of ways to center your webpage.

You can add this line at the start of your file.

```
<center>
```

For balance, you should also add the matching end tag at the end of your file.

```
</center>
```

That is the old way to do it. That is not really the best way to do it, for reasons we will get into later, but it is commonly done.

The better way is to use CSS. (We get more deeply into CSS in Unit VI, page 173.) Put this at the top of your file.

```
<style type="text/css">
body { text-align: center; }
</style>
```

4.12 Background Color

Background coloring is another popular request. How do I color my webpage?

We can modify the style to include both centering and background coloration.

```
body { text-align: center; background-color: yellow; }
```

You can put an entire background image behind your webpage. Say something like this in your style section.

```
body { background-image: url(myphoto.jpg); }
```

4.13 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 5

Web Hosting

Contents

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.

These days you can buy webhosting for under $5 per month, depending on the length of the contract. You don't always get more by paying more. It pays to compare what different companies offer.

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.

5.1 Web Hosting Providers

You can Google search "web hosting" to find many hosting providers.

1&1: http://www.1and1.com/

Bluehost: http://www.bluehost.com/

Hostgator: `http://www.hostgator.com/shared`

Fatcow: `http://www.fatcow.com/`

5.2 Your Free Domain Name

As part of getting your web hosting set up, the web hosting provider will probably let you pick a free domain name. In fact, they will probably require it.

Pay attention here.

Pick a name that you can walk away from. If the time ever comes that you decide to use a different web hosting provider, you will be able to reinstall your content somewhere else.

However, you may not be able to keep your free domain name. (This actually happened to me.)

We will talk more about domain names in Chapter 7 (page 37).

The purpose of this free domain name is to give you a way to access their control panel.

The other purpose is to lock you into using their service. The free domain name they give you may not be portable, or may only be portable after you pay an additional and surprisingly high fee.

My advice is to keep them separate. Use the free domain name to access their service and manage your accounts. Get another domain name to operate your business. Keep them separate.

Chapter 6

Control Panel

Contents

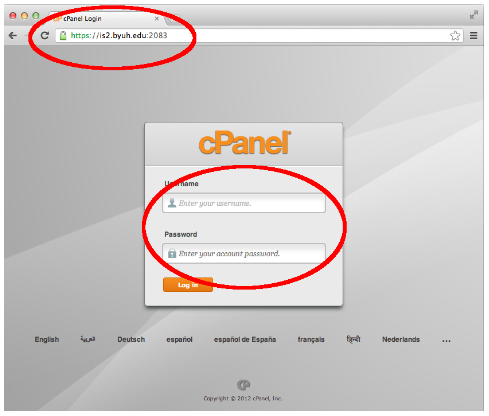
Once you have hosting, you need to set up your website and add some content. But how do you set it up? How do you add content?

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.

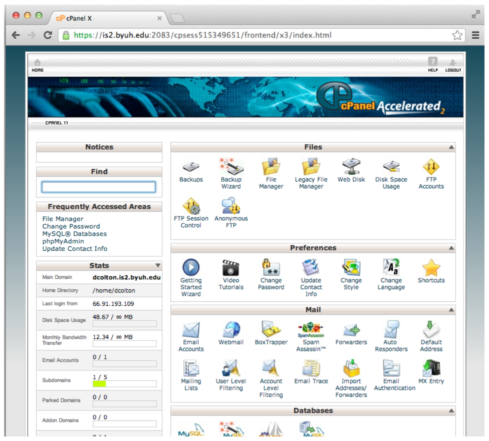
This front end runs the scripts and programs that, in olden days, would have been done by hand by said guru (a systems administrator).

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.



6.1 Domain Management

If you only have one domain, and you are using an Apache-based webserver, your **document root** will 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.

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`

6.2 File Manager

To 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 Appendix J (page 275).

WordPress and Drupal are very popular packaged solutions.

`http://wordpress.org/` has more on WordPress.

`http://drupal.org/` has more on Drupal.

6.2.1 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 is the clear leader, and is so important, in Appendix I (page 265) we provide details for several widely-used things that you can configure. Most of these would be considered to be advanced training.

Public html is part of Apache. It qualifies as basic training.

6.2.2 Public HTML

The file system for Apache normally puts all webpages in a directory (also known as a folder) named `public_html`. Sub-folders can be made and they can have content you provide.

So, start up cPanel, then enter the File Manager, then navigate to the `public_html` directory. This is where your home page will go.

6.2.3 The 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 28 (p.299)**:**
Is the domain name within a URL case-sensitive?

Required Answer:
no

Exam Question 29 (p.299)**:**
Is the path within a URL case-sensitive?

Required Answer:
yes

6.2.4 Uploading

You can upload files by using the `upload` button in cPanel. Normally this is how you would import things like pictures, but you can also upload style sheets and webpages and almost anything else.

6.2.5 Editing

You can create and edit files by using the appropriate buttons in cPanel. You might edit your webpages directly in cPanel. Or you might choose to edit those on your local computer and upload them.

6.2.6 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.

For more on copying between your personal computer and your webhost server, please see Appendix J (page 275).

6.3 Passwords

You should create and use a good password for cPanel. Anyone who knows your password can steal your identity and take over your website.

How long should a password be?

What characters should I use?

How do hackers get people's passwords?

How long does it take a hacker to crack a password?

What passwords do hackers guess first? Is 123456 a good password?

What if I have a really good password. Can I use the same really good password on more than one website?

How often should I change my password?

See Appendix K (page 278) for more discussion and answers to these and other questions.

Here are some short answers:

Passwords for important web accounts should be 12 characters long, maybe more. This includes email and online banking.

Using a variety of characters (letters, digits, punctuation) is helpful, but length is more important. Some password systems require the variety.

Using the same password on several important web accounts is dangerous. If one site gets hacked, all your accounts are in danger.

You should change your password if you think it has been discovered. Otherwise it is probably safe to leave it alone for several years.

Chapter 7

Domain Names

Contents

To put your webpages online, you will need a domain name and web hosting.

Every website needs a domain name. Otherwise, people would have to refer to it by number, which may work, but which is a bit of a pain, and also makes you look unprofessional.

7.1 Your First Domain Name

Your webhosting provider will probably give you a free domain name as part of your hosting purchase.

We recommend that you do **NOT** use that free domain name for your websites. Instead, use it only to set up and manage your websites.

The domain names you publicize and use for your main content should probably be add-on domain names, registered separately from your webhost.

Partly this is because you do not want 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.

7.2 Your Registrar

You get domain names from registrars. Normally you pay them money. They register your domain name and give you full control over it.

They also tell everyone the actual network address where your website is located. This service is called DNS, Domain Name Service.

Hosting is not the same as registration. Hosting provides the storage space and network access for your website. See chapter 5 (page 29) for details.

7.3 Rented vs Owned

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 30 (p.299)**:**
Can you really own a domain name?

Required Answer:
no

`http://en.wikipedia.org/wiki/Domain_registration` has more details.

You can Google search "domain registration" to find lots of domain registrars.

In Chapter 8 (page 42) below, we recommend a free registrar for .tk domain names.

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 31 (p.299)**:**

What is the typical cost for a .com domain name, in dollars per year?

Acceptable Answer:
10

7.4 Picking A Domain Name

You will probably want a domain name that is easy to remember. And easy to spell. Short names are also better than long ones.

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 dollars, to get a "premium" domain name.

Collecting domain names is fun, but it can be an expensive hobby. You have to keep paying for them. I bought a few that I eventually gave up. I still have four or five.

7.5 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. 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.

Your registrar provides the basic Domain Name Service 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 converts a domain name into a document root. The document root tells which folder (directory) holds your webpages.

7.6 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 32 (p.300)**:**
What words does TLD stand for?

Required Answer:
top level domain

7.7 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.

`http://byuh.doncolton.com` - my campus homepage.

`http://hangman.doncolton.com` - a hangman puzzle solver.

`http://quizgen.doncolton.com` - a quiz generator.

`http://iwdd.doncolton.com` - homepage for this book.

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.

Exam Question 33 (p.300)**:**
Who owns byuh.doncolton.com?

Required Answer:
doncolton.com

7.8 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.

Chapter 8

Dot TK: Domains for Free

Contents

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.

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.

8.1 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 referals collect money from Google for providing traffic to the eventual website.

8.2 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.

8.3 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.

8.4 Provide for DNS

This step happens at the dot.tk website.

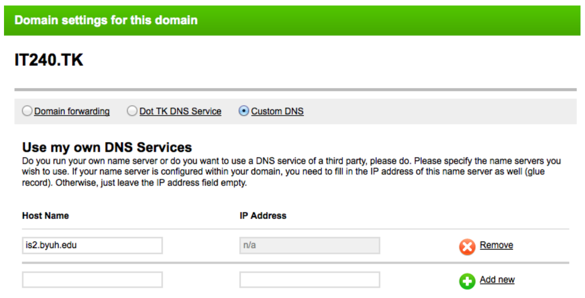
DNS is the Domain Name Service. When people come to visit your website, all they will know is your domain name. Your registrar (in this case, dot.tk) will tell them where to find your website.

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.



IT240.TK

Domain forwarding [✓ Custom DNS] Dot TK DNS Service

Use my own DNS Services

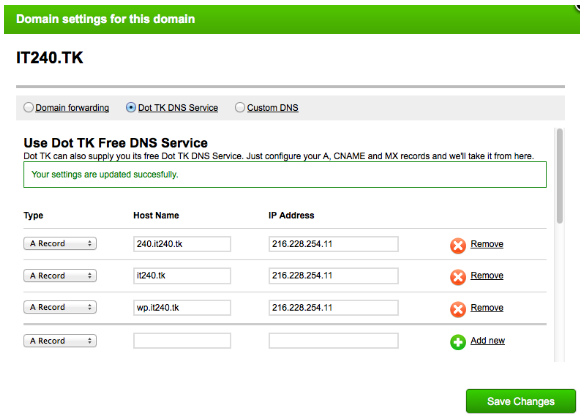
Do you run your own name server or do you want to use a DNS service of a third party, please do. Please specify the name servers you wish to use. If your name server is configured within your domain, you need to fill in the IP address of this name server as well (glue record). Otherwise, just leave the IP address field empty.

Host Name	IP Address	
is2.byuh.edu	n/a	Remove
		Add new

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).



IT240.TK

Domain forwarding [✓ Dot TK DNS Service] Custom DNS

Use Dot TK Free DNS Service

Dot TK can also supply you its free Dot TK DNS Service. Just configure your A, CNAME and MX records and we'll take it from here.

Your settings are updated succesfully.

Type	Host Name	IP Address	
A Record	240.it240.tk	216.228.254.11	Remove
A Record	it240.tk	216.228.254.11	Remove
A Record	wp.it240.tk	216.228.254.11	Remove
A Record			Add new

Redirecting makes your .tk name a shortcut to get to your original website with its original name.

8.5 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.

8.6 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.

8.7 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.

8.8 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.

8.9 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.

8.10 If Things Stop 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. 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 9

Troubleshooting Your New Domain

Contents

You think you did everything right, but you still cannot get to your website. What can you do? This chapter lists some troubleshooting steps that can be followed by yourself or someone helping you.

9.1 Time Causes Problems

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.

The reason we have propagation delays is that it costs too much to keep checking on configuration details. Instead, computers are programmed to remember those details, and only recheck them occasionally, like once a day.

The configuration settings are cached to remember them. Domain information in the **cache** will be remembered for 24 hours (or more or less), 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.

9.2 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?

Example: I created `it240.tk` and set it with the IPv4 address of `216.228.254.11`.

You can issue a "dig" command to find what the DNS system knows about your domain name.

```
dig it240.tk @8.8.8.8
```

The `@8.8.8.8` part means to use the DNS server that lives at **8.8.8.8**. Google is providing that as a free service to the users of the Internet. So, we are asking **dig** to use the Google DNS server and tell us what it knows.

The response should include your IP address. If it does not, then you know the problem is either with Google (possible but not likely), or with your domain registrar (possible but not likely), or with the way you did your configuration with your domain registrar (likely). But any of them **could** be the problem.

9.3 Is The Web Server Correct?

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.

Chapter 10

Copyright

Contents

I am not a lawyer, and this is not legal advice. But listen anyway.

Copyright can be your friend, or it can get you into trouble.

This is the general rule. (There are exceptions.)

- Do not publish pictures that are owned by someone else. This especially includes pictures you have copied from another website.
- Do not publish words that were written by someone else. This especially includes words you 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 your website.

To be safe, everything you post on the web should be your own work. That is not always possible. This chapter should help you know what you can legally do, and what trouble you 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.

`http://en.wikipedia.org/wiki/Copyright` has more about copyright.

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.

10.1 Terminology

Ideas cannot be protected by copyright. They may be partially protected by patent.

Exam Question 34 (p.300)**:**
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.

`http://en.wikipedia.org/wiki/Plagiarism` has more on plagiarism.

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 C.1, page 246.

Fair use is the concept (in the USA) 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 35 (p.300)**:**
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.

`http://en.wikipedia.org/wiki/Cover_version` has more.

Exam Question 36 (p.300)**:**
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. Appendix C (page 246) has more information.

Exam Question 37 (p.300)**:**
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 C.3 (page 248) for more information.

Appendix C (page 246) has more information about public domain sources of content.

10.2 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.

`http://en.wikipedia.org/wiki/Derivative_work` has more.

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 context.)

Exam Question 38 (p.300)**:**
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.

10.3 Finding Violations Is Easy

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 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 10.4 (page 54) 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.

10.4 DMCA: Digital Millenium Copyright Act

In the USA, the Digital Millenium 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 39 (p.300)**:**
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 40 (p.300)**:**
What is a Safe Harbor?

Acceptable Answer:
It is a set of rules that when followed protect you from legal liability.

Exam Question 41 (p.300)**:**
What words does DMCA stand for?

Acceptable Answer:
digital millenium copyright act

`http://en.wikipedia.org/wiki/Dmca` has more about the DMCA.

10.5 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 C.1, page 246).

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, record it yourself.

Chapter 11

Publish: Going Online

Finally we will put our simple webpage online.

Visit your control panel. Go into your file manager.

Get into your `public_html` directory. (Here we assume your webserver is running Apache, which is probably true.)

Upload your **index.html** and `myphoto.jpg` files that you made back in chapter 4 (page 20).

Visit your website. You can do this by typing your URL into the navigation bar of your browser. The URL will look something like this:

`http://fred.is2.byuh.edu/` or `http://mydomain.tk/`

Activity: First Webpage

This may be suitable for homework or an in-class activity.

Create a webpage.

Include your name as part of the page content.

Include some content that tells about you. Maybe include a quote you find inspiring or amusing.

Mark up your webpage the following tags: h1, b, i.

Include a picture that represents you or something about you, perhaps your own picture, or your family, or perhaps your hobby or a picture from a vacation.

Chapter 12

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.

12.1 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.

12.2 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.

12.3 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 the 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?

12.4 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 type="text/javascript">
  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>
```

12.5 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 13

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.

First, it has content, including markup. In Unit V (page 121) of this book, we will look at the most common kinds of markup that are used in building webpages.

Second, it has presentation, including style. In Unit VI (page 173) of this book, we will look at the most common kinds of styling that are used in building webpages.

Third, it has action, normally JavaScript. In Unit VII (page 218) of this book, we will look at the most common kinds of JavaScript that are used in building webpages.

13.1 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.

13.2 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.

13.2.1 Presentational Markup

Presentational markup is focused specifically on how things look. We have already seen `<b>` and `<i>` and `<br/>`.

`<b>` is the presentational markup that introduces a section of **bold** text.

`<i>` is the presentational markup that introduces a section of *italicized* text.

`<br/>` or less properly `<br>` is the presentational markup that causes a line break in your text.

`<hr/>` or less properly `<hr>` is the presentational markup that draws a horizontal line across the block it is in.

13.2.2 Structural Markup

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

`<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).

`<p>` is the structural markup that introduces a paragraph.

`<ol>` is the structural markup that introduces an ordered list.

`<table>` is the structural markup that introduces a table.

13.2.3 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.

13.3 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.

13.4 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.

13.5 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.)

13.6 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 stylesheet.

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 stylesheet. Just remember to do it.

Do worry about it. Later. (Soon.)

Skilled developers already have a good sense of how to design their stylesheets. They will not resort to quick and dirty techniques nearly so much as the newbies who lack the depth of experience.

13.7 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 in Unit VII (page 218). 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 36.9 (page 170). AJAX uses JavaScript to update just a part of a webpage with new content from the web.

Chapter 14

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.

14.1 W3.org

14.1.1 Validators

W3C is the standards body that develops standards for the World Wide Web. `http://www.w3.org/` is their homepage. Their materials are 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://validator.w3.org/` is their validator for HTML code.

`http://jigsaw.w3.org/css-validator/` is their validator for CSS code.

14.1.2 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.

14.2 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.

Unit III

Web Development Tools

Chapter 15

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).

15.1 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.

15.2 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 Chapter 16 (page 76) we look more at CMS.

15.3 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 42 (p.301)**:**
What words 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.

15.4 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 16

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 top CMS in the world is **WordPress**. It has about 55% of the CMS market (in 2011). Second place seems to be **Joomla**, and third place seems to be **Drupal**.

You can Google search “market share cms” for current information.

`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 Chapter 17 (page 78) 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 43 (p.301)**:**
What words does CMS stand for?

Required Answer:

content management system

Chapter 17

WordPress

Contents

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.

17.1 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 44 (p.301)**:**
What words does blog stand for?

Required Answer:
web log

17.2 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.

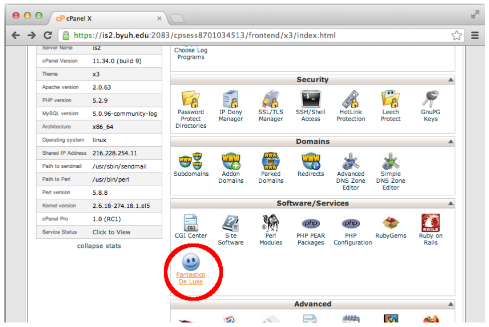
You can set up a blog named (yourchoice).wordpress.com for free. Of course, you can expect it to have outside advertisements and be limited, but it is free and that can be a good way to start.

17.3 Installing WordPress: cPanel and Fantastico

cPanel is commonly used and Fantastico is commonly available at web hosts.

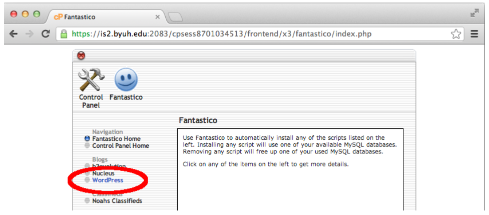
Using cPanel, you can quickly install WordPress for your work in this class.

After logging into cPanel, we select Fantastico. It will handle the installation of WordPress (or many other things).



Server Name	is2
cPanel Version	11.34.0 (build 9)
Theme	x3
Apache version	2.0.63
PHP version	5.2.9
MySQL version	5.0.96-community-log
Architecture	x86_64
Operating system	linux
Shared IP Address	216.228.254.11
Path to sendmail	/usr/sbin/sendmail
Path to Perl	/usr/bin/perl
Perl version	5.8.8
Kernel version	2.6.18-274.18.1.el5
cPanel Pro	1.0 (RC1)
Service Status	Click to View

Within Fantastico, we will select WordPress for installation.

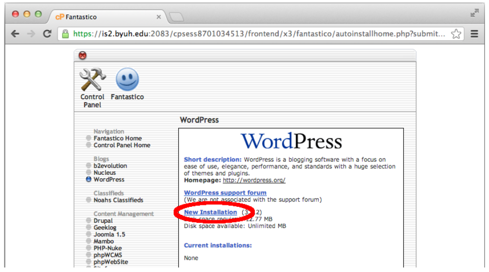


Fantastico

Use Fantastico to automatically install any of the scripts listed on the left. Installing any script will use one of your available MySQL databases. Removing any script will free up one of your used MySQL databases.

Click on any of the items on the left to get more details.

We will choose to do a New Installation.



WordPress

WordPress

Short description: WordPress is a blogging software with a focus on ease of use, elegance, performance, and standards with a huge selection of themes and plugins.
Homepage: http://wordpress.org/

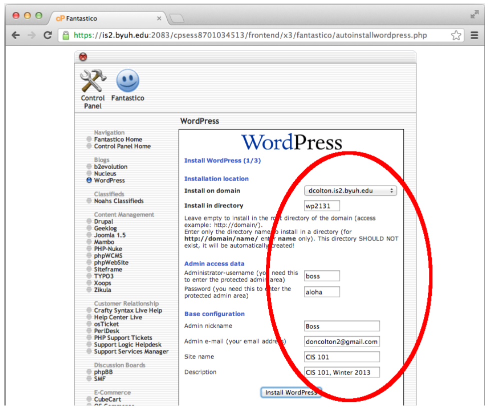
WordPress support forum
(We are not associated with the support forum)

New Installation (3.2)
Disk space required: 12.77 MB
Disk space available: Unlimited MB

Current installations:

None

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.



WordPress

WordPress

Install WordPress (1/3)

Installation location

Install on domain	dcolton.is2.byuh.edu
Install in directory	wp2131

Leave empty to install in the root directory of the domain (access example: http://domain/).
Enter only the directory name to install in a directory (for **http://domain/name/** enter **name** only). This directory SHOULD NOT exist, it will be automatically created!

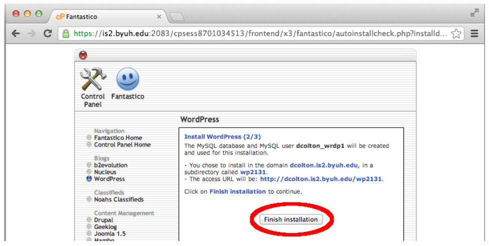
Admin access data

Administrator-username (you need this to enter the protected admin area)	boss
Password (you need this to enter the protected admin area)	aloha

Base configuration

Admin nickname	Boss
Admin e-mail (your email address)	doncolton2@gmail.com
Site name	CIS 101
Description	CIS 101, Winter 2013

Now we “finish” the installation.



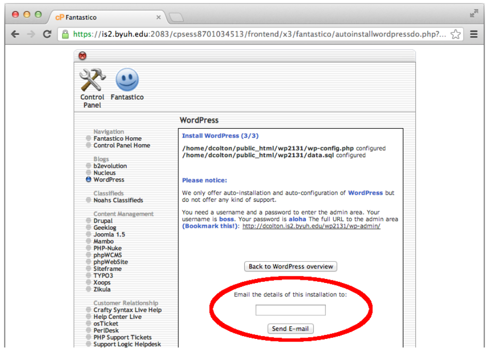
WordPress

Install WordPress (2/3)
The MySQL database and MySQL user **dcolton_wrdp1** will be created and used for this installation.

- You chose to install in the domain **dcolton.is2.byuh.edu**, in a subdirectory called **wp2131**.
- The access URL will be: **http://dcolton.is2.byuh.edu/wp2131**.

Click on **Finish installation** to continue.

And we email ourselves a copy of the results.



WordPress

Install WordPress (3/3)

/home/dcolton/public_html/wp2131/wp-config.php configured
/home/dcolton/public_html/wp2131/data.sql configured

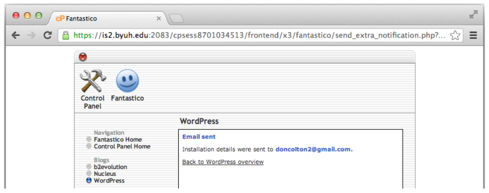
Please notice:

We only offer auto-installation and auto-configuration of **WordPress** but do not offer any kind of support.

You need a username and a password to enter the admin area. Your username is **boss**. Your password is **aloha** The full URL to the admin area **(Bookmark this!)**: http://dcolton.is2.byuh.edu/wp2131/wp-admin/

Email the details of this installation to:

It reports that the email was successfully sent.



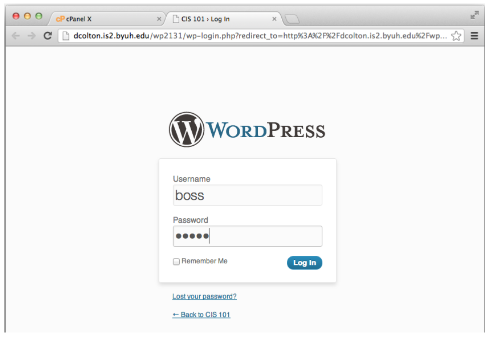
WordPress

Email sent

Installation details were sent to **doncolton2@gmail.com.**

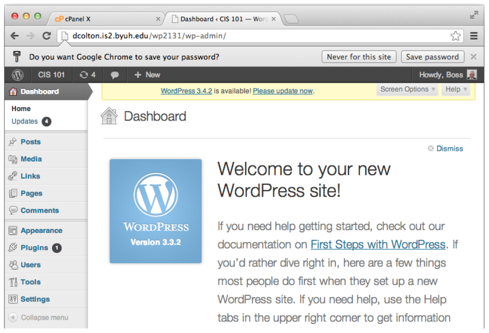
Back to WordPress overview

Here we log into our WordPress website.


cP cPanel X
CIS 101 › Log In
dcolton.is2.byuh.edu/wp2131/wp-login.php?redirect_to=http%3A%2F%2Fdcolton.is2.byuh.edu%2Fwp...
WordPress
Remember Me
Log In
Lost your password?
← Back to CIS 101

Username: boss
Password: ●●●●●

Welcome. And please upgrade to version 3.4.2. So, we follow the links and do the upgrade.



dcolton.is2.byuh.edu/wp2131/wp-admin/

Do you want Google Chrome to save your password? Never for this site | Save password

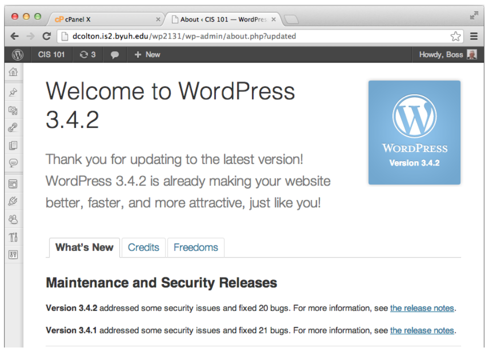
WordPress 3.4.2 is available! Please update now.

Dashboard

Welcome to your new WordPress site!

If you need help getting started, check out our documentation on First Steps with WordPress. If you'd rather dive right in, here are a few things most people do first when they set up a new WordPress site. If you need help, use the Help tabs in the upper right corner to get information

We have now upgraded.



dcolton.is2.byuh.edu/wp2131/wp-admin/about.php?updated

Welcome to WordPress 3.4.2

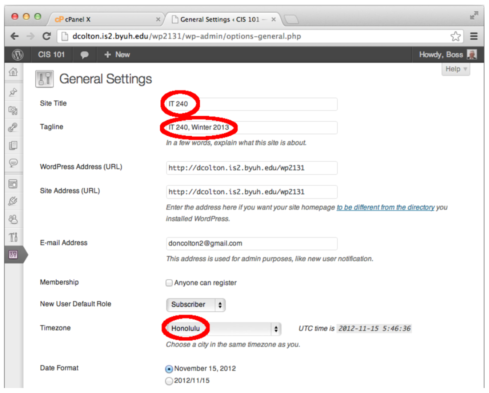
Thank you for updating to the latest version! WordPress 3.4.2 is already making your website better, faster, and more attractive, just like you!

Maintenance and Security Releases

Version 3.4.2 addressed some security issues and fixed 20 bugs. For more information, see the release notes.

Version 3.4.1 addressed some security issues and fixed 21 bugs. For more information, see the release notes.

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.



dcolton.is2.byuh.edu/wp2131/wp-admin/options-general.php

General Settings

Site Title	IT 240
Tagline	IT 240, Winter 2013 *In a few words, explain what this site is about.*
WordPress Address (URL)	http://dcolton.is2.byuh.edu/wp2131
Site Address (URL)	http://dcolton.is2.byuh.edu/wp2131 *Enter the address here if you want your site homepage to be different from the directory you installed WordPress.*
E-mail Address	doncolton2@gmail.com *This address is used for admin purposes, like new user notification.*
Membership	☐ Anyone can register
New User Default Role	Subscriber
Timezone	Honolulu *UTC time is* 2012-11-15 5:46:36 *Choose a city in the same timezone as you.*
Date Format	● November 15, 2012 ○ 2012/11/15

17.4 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.

17.5 WordPress Themes

todo: more to be added

using other people's themes

developing your own themes

17.6 WordPress Plugins

Plugins extend the functionality of your website.

todo: more to be added

blog / news feed

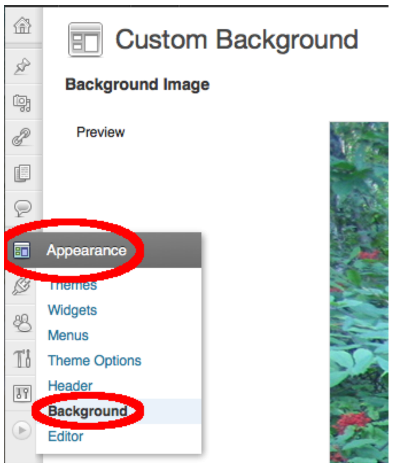
directory

making a Website Banner

17.7 WordPress HowTos

Question: How do you put a custom background image behind a WordPress page?

Answer: On the menu ribbon, select 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.



Unit IV

Media: Images, Audio, Video

Chapter 18

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.

18.1 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 managable. 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.

18.2 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.

18.3 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.

18.4 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.

18.5 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.

`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**.

`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.

18.6 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

18.7 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 19

Basic Image Processing

Contents

Besides text, **images** (**photographs**) are the most common content on webpages. In this chapter we will demonstrate the two most common modifications to images: **cropping** and **resizing**.

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.

Resizing changes the number of pixels in the picture. It is sometimes called **downsampling** or **resampling**.

Color modification and transparency are also commonly done. We will not demonstrate color modification, but in Chapter 21 (page 99) we will work with transparency.

19.1 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 at 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, 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.

19.2 Photo Editing Tools

Probably the two most commonly used high-end photo editing tools are Adobe Photoshop and The Gimp.

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.

The Gimp is a free, open-source product. Free is a really good price. Chapter 20 (page 97) has more.

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.

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.

19.3 The Gimp

In the examples in this chapter, I am using **Gimp** version 2.8.

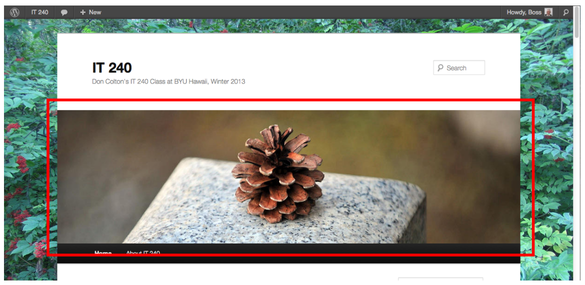
Chapter 20 (page 97) has more information on The Gimp.

Because Gimp is free and has capabilities roughly equal to Photoshop, Gimp is the photo editor that I most frequently use.

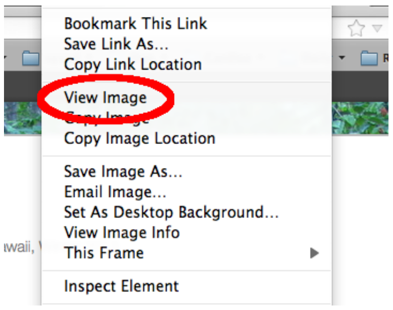
In the example that follows, I will show the steps I took to create a 1000 by 288 banner for my WordPress website.

19.4 Sample Image

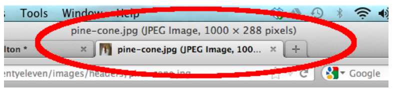
In this picture, I put a red box around the part of the webpage that is the header image. I intend to replace it with my own image. I want to find out its dimensions.



Using the Firefox browser, I did a right-click on the image. I selected "View Image" from the menu.

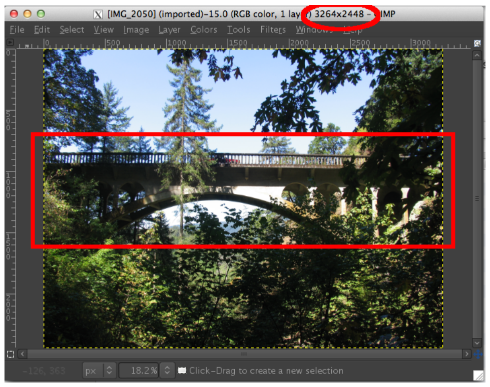


Viewing just the image, I can see that its dimensions are 1000 x 288 pixels. I repeated the process for another header image to make sure the dimensions were the same. They were.


pine-cone.jpg (JPEG Image, 1000 × 288 pixels)
pine-cone.jpg (JPEG Image, 100...

19.5 Crop My Image

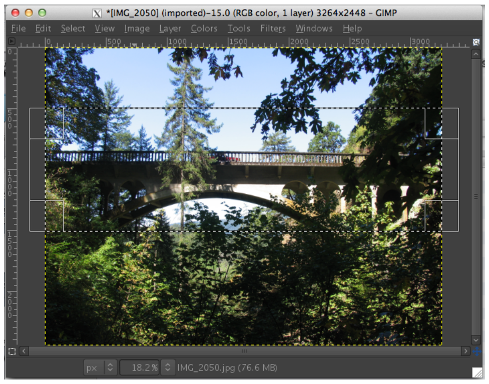
Here I have started Gimp and I am looking at a full-sized original image from which I hope to extract a portion to become a new header image.


[IMG_2050] (imported)-15.0 (RGB color, 1 layer) 3264x2448 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Click–Drag to create a new selection

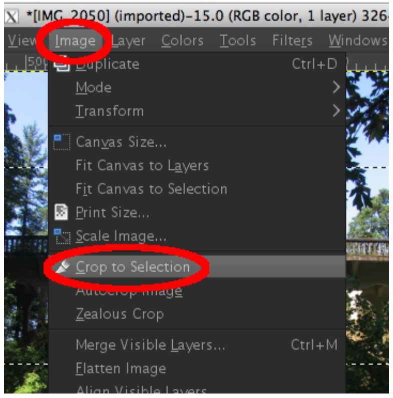
This is a picture of the bridge at Shepherd's Dell, a waterfall on the Oregon side of the Columbia River gorge, taken from the base of the waterfall and looking up towards the road.

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.


*[IMG_2050] (imported)-15.0 (RGB color, 1 layer) 3264x2448 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
px 18.2% IMG_2050.jpg (76.6 MB)

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



19.6 Scale My Image

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.

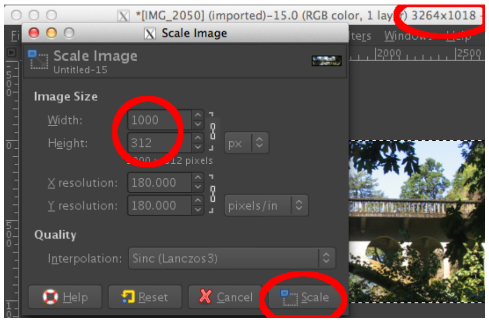


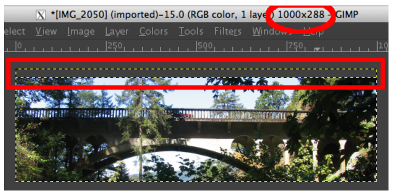
Image Size
Width: 1000
Height: 312 px
1000 x 312 pixels
X resolution: 180.000
Y resolution: 180.000 pixels/in
Quality
Interpolation: Sinc (Lanczos3)

Our next task is to Scale the Image. From the menu, I select **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.

19.7 Change the Canvas Size

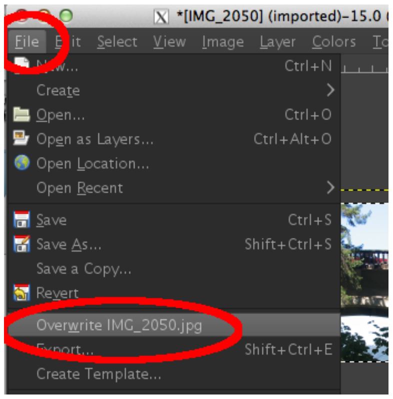

*[IMG_2050] (imported)-15.0 (RGB color, 1 laye 1000x288 – GIMP
elect View Image Layer Colors Tools Filters Windows Help
0 250 500 750

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 **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.

19.8 Export the Resulting Image



From the menu, I select **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, **Menu: File / Export**.

Chapter 20

The Gimp

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.

Menu: Edit / Cut deletes whatever is currently selected.

Menu: Edit / Undo reverses the effects of the previous command. Can be done repeatedly.

Menu: File / Export writes the current image in whatever image format is desired.

Menu: File / New creates a new image that is blank.

Menu: File / Overwrite writes the current image by replacing an existing image file.

Menu: Filters / Map / Make Seamless modifies the current image so it can be used for seamlessly tiling the background of a webpage.

Menu: Filters / Map / Small Tile shows what the current image would look like if it were used to tile the background of a webpage.

Menu: Image / Canvas Size changes the size of the image.

Menu: Image / Crop to Selection deletes everything that is not currently selected, and changes the size of the image.

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.

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.

Menu: Layer / Transparency / Add Alpha Channel adds the capability for transparency to the current layer of the image.

Chapter 21

Image Transparency

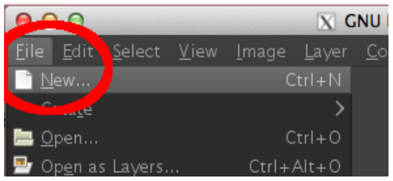
In the examples in this chapter, I am using Gimp version 2.8.

In this chapter, we demonstrate how to make a portion of an image transparent.

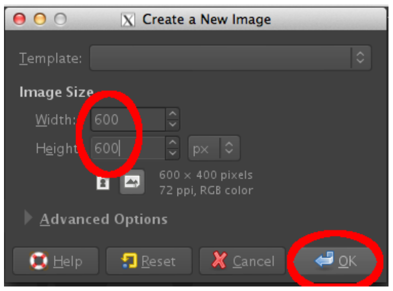
When such an image is displayed on a webpage, the part that is transparent will show the colors behind the image, whatever would have been there if the image were not.

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.

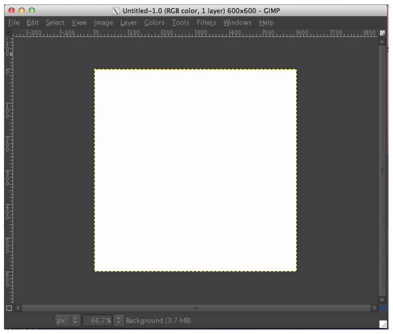
From the menu, I select **Menu: File / New**.


GNU
File Edit Select View Image Layer Co
New... Ctrl+N
Create
Open... Ctrl+O
Open as Layers... Ctrl+Alt+O

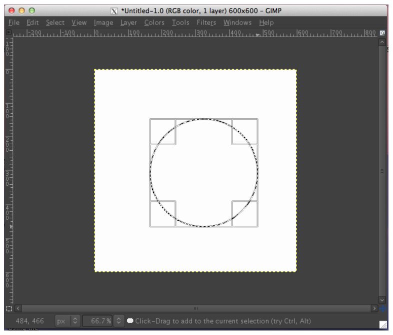
I specify the size, here 600 by 600 pixels, and click on "OK."



This creates a canvas that is 600 by 600, with a white background.

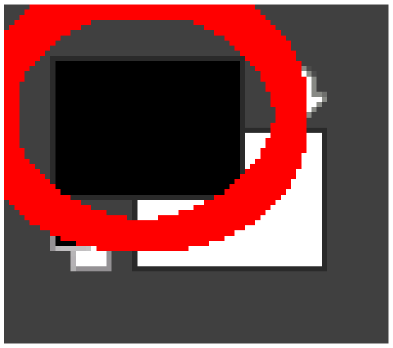

Untitled-1.0 (RGB color, 1 layer) 600x600 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
px
66.7 %
Background (3.7 MB)

I double-click on the circle icon, and then click and drag a circle on the canvas.

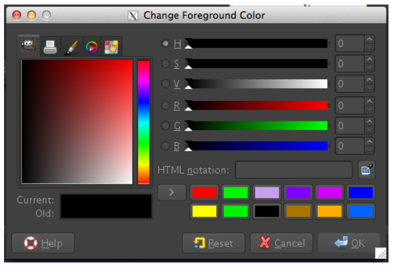

*Untitled-1.0 (RGB color, 1 layer) 600x600 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
484, 466
px
66.7 %

Click–Drag to add to the current selection (try Ctrl, Alt)

The color scheme is black (foreground) on white (background). I want to change the foreground color so I double-click on the black square.

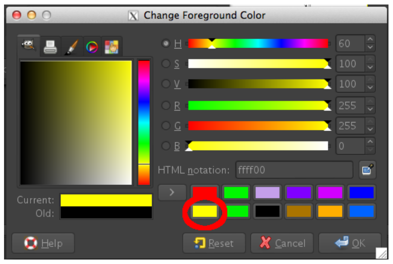


I am presented with a color selection screen.



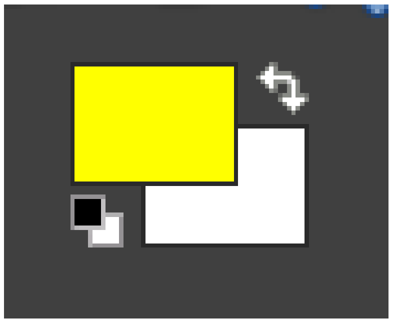
H 0, S 0, V 0, R 0, G 0, B 0

I want yellow, so I double-click the yellow cell.

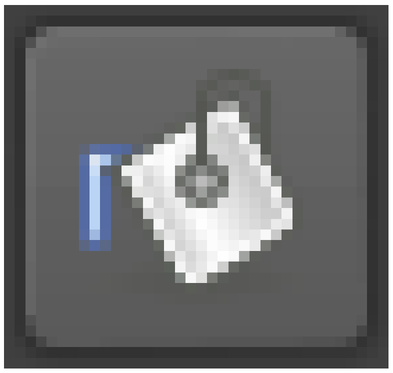


H 60
S 100
V 100
R 255
G 255
B 0
HTML notation: ffff00

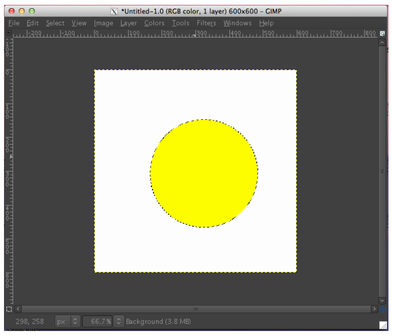
The color scheme changes to yellow (foreground) on white (background).



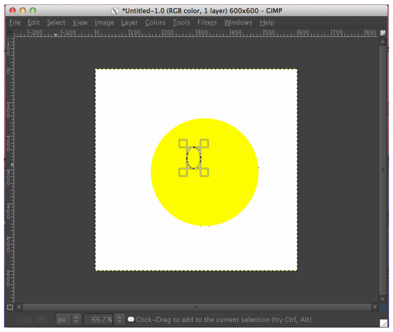
I double-click on the paint bucket. I will use it to dump yellow paint into my circle.



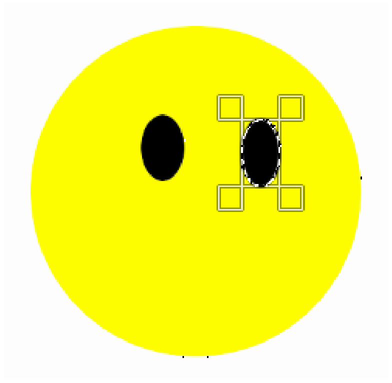
I move the cursor into my circle and double-click. The inside of the circle is now yellow.


*Untitled-1.0 (RGB color, 1 layer) 600x600 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
298, 258
px
66.7%
Background (3.8 MB)

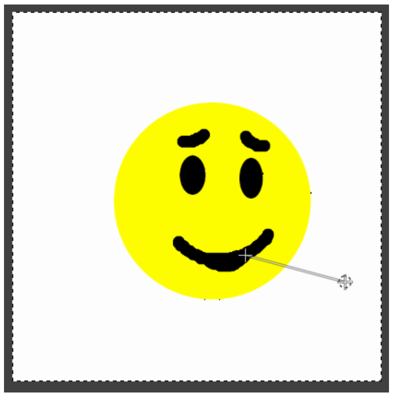
I change the color scheme back to black on white. Then I double-click the ellipse tool, and I draw an ellipse for an eye on my smiley face.


*Untitled-1.0 (RGB color, 1 layer) 600x600 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
px
66.7%
Click–Drag to add to the current selection (try Ctrl, Alt)

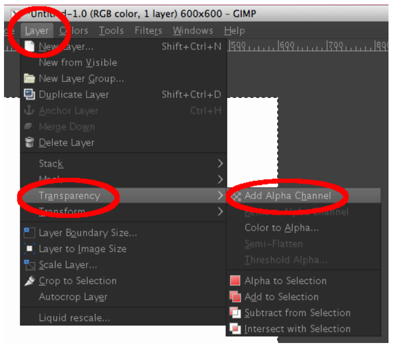
Using the bucket, I fill the eye with black. Then I repeat the process to create another eye.



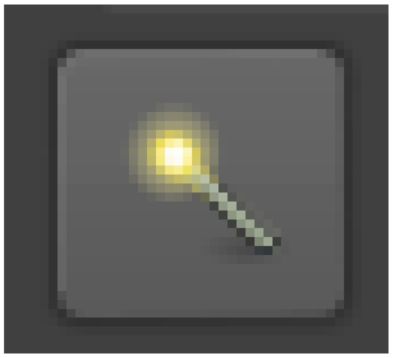
I double-click the pencil tool and use it to draw eye brows and a mouth. My graphic is complete. (And actually kind of cute.)



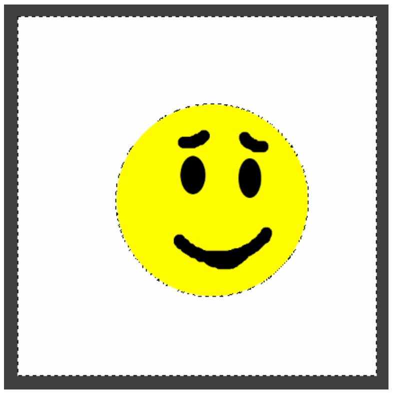
I am ready to add transparency (although I could have done it earlier). I want the white parts to go away. I select **Menu: Layer / Transparency / Add Alpha Channel**.



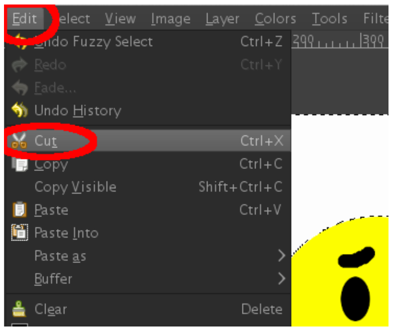
Next, I select the magic wand tool and click anywhere in the white part of my canvas.



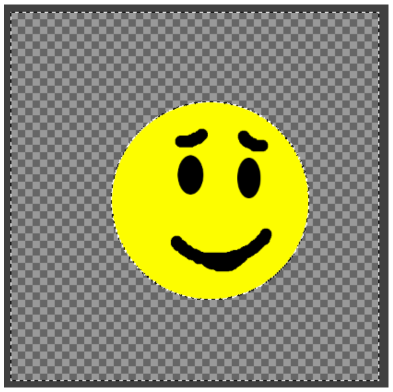
The entire white area gets a dotted line around it, indicating that it has been selected.



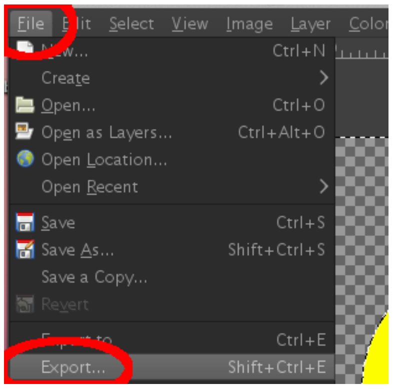
I select **Menu: Edit / Cut**, and press enter.



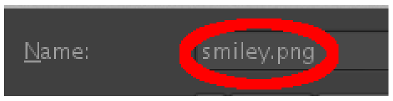
The white parts of my canvas have been replaced by a gray checkerboard, which is commonly used as the symbol for transparency.



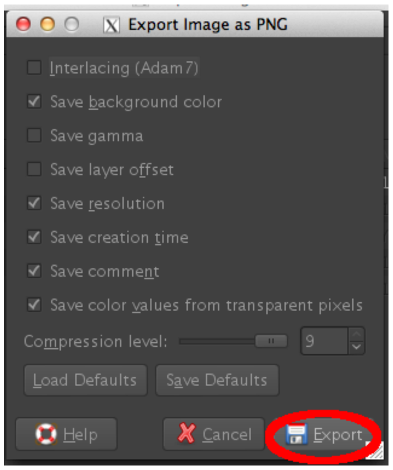
Next, I select **Menu: File / Export**.



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.


Export Image as PNG

- [ ] Interlacing (Adam7)
- [✓] Save background color
- [ ] Save gamma
- [ ] Save layer offset
- [✓] Save resolution
- [✓] Save creation time
- [✓] Save comment
- [✓] Save color values from transparent pixels

Compression level: 9

Chapter 22

Favicon

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 good luck with .png files. I like that they allow transparency.

The trick is to name your image file **favicon.ico** and put it in your root directory.

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.

Chapter 23

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.

23.1 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); }
```

23.2 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.

23.3 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.

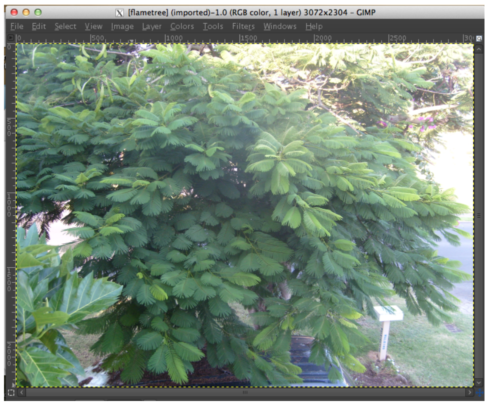
dirt, as in a newly-turned garden plot.

grass, as on a lawn. This may suffer from the obvious discernible pattern of the individual blades of grass.

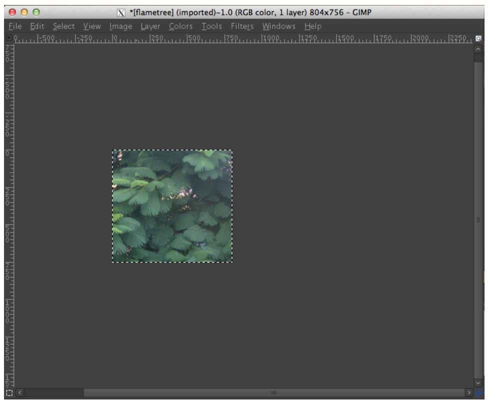
23.4 Example 1

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.

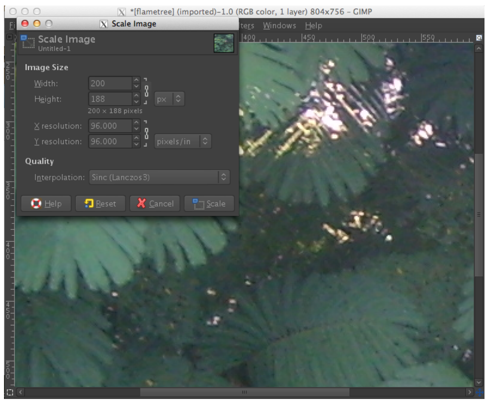
Here is our original picture. It is a Flame Tree located in my own front yard. The leaves look as though they might make a nice background.


[flametree] (imported)-1.0 (RGB color, 1 layer) 3072x2304 - GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help

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.


*[flametree] (imported)-1.0 (RGB color, 1 layer) 804x756 - GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help

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.

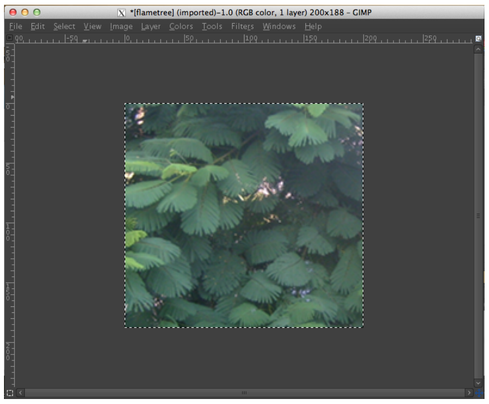

*[flametree] (imported)-1.0 (RGB color, 1 layer) 804x756 - GIMP
Scale Image

Scale Image
Untitled-1

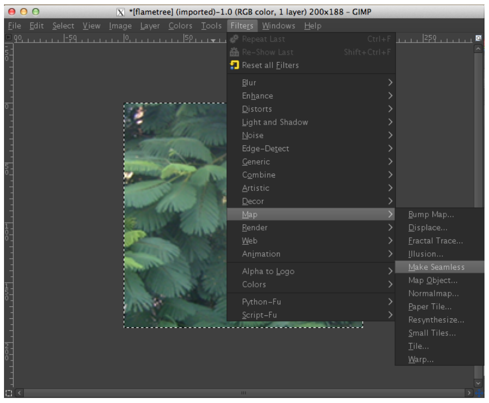
Image Size
Width: 200
Height: 188 px
200 × 188 pixels
X resolution: 96.000
Y resolution: 96.000 pixels/in

Quality
Interpolation: Sinc (Lanczos3)

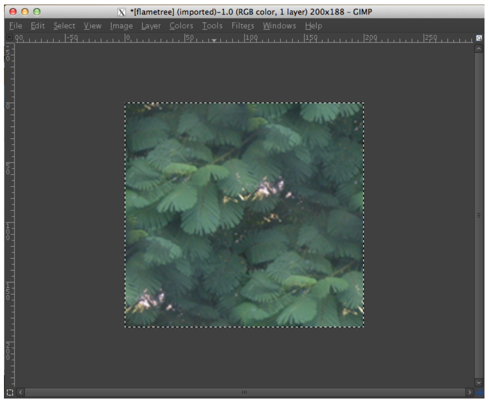
We resize the view. Here we see the resulting image, displayed at 200%.


*[flametree] (imported)-1.0 (RGB color, 1 layer) 200x188 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help

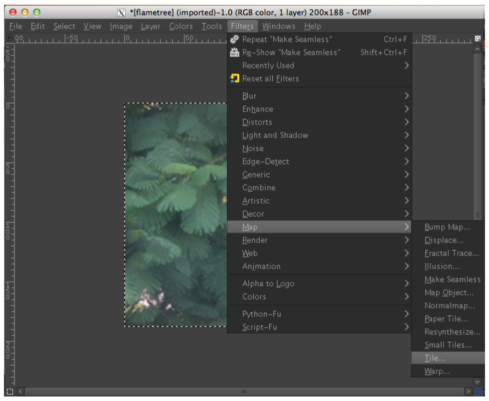
Here is the trick. We use **Menu: Filters / Map / Make Seamless**. This will do all the work for us.



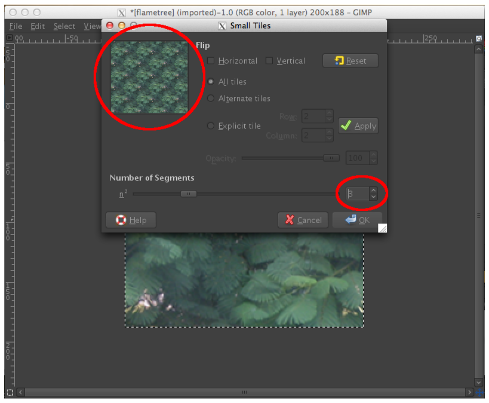
You can see that the resulting image is different than it was before. It tiles perfectly, by which we mean that the top and bottom edges match perfectly, as to the left and right edges.


*[flametree] (imported)-1.0 (RGB color, 1 layer) 200x188 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help

We will use **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.

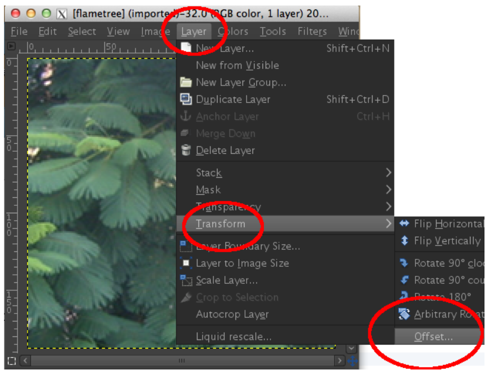


Bottom line: easy to do, but creates an obvious pattern. In any given instance, such a pattern may not be a problem. But at other times it may be annoying. You decide.

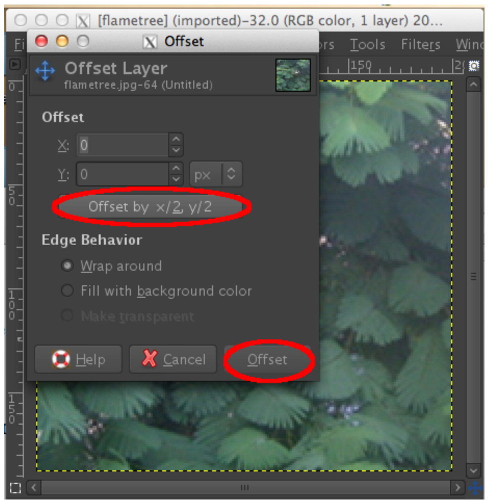
23.5 Example 2

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 **Menu: Layer / Transform / Offset**.

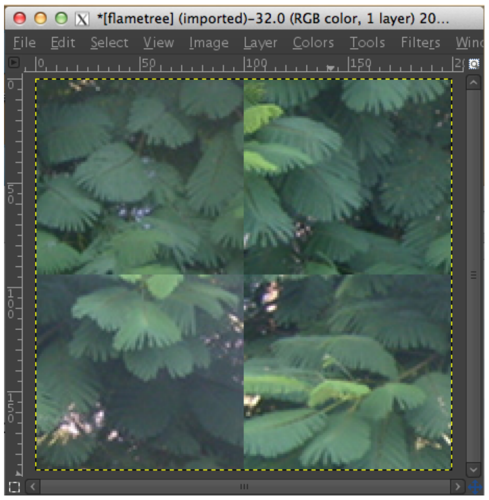


We choose to offset by x/2 and y/2, which means we move the image half way in the left-right dimension, and half way in the up-down dimension.

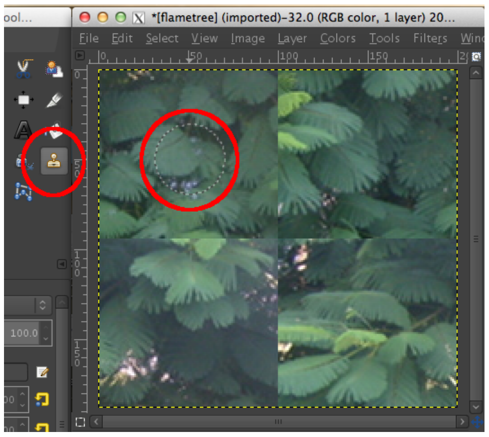

[flametree] (imported)-32.0 (RGB color, 1 layer) 20…
Offset
Offset Layer
flametree.jpg-64 (Untitled)
Offset by x/2, y/2
Help
Cancel
Offset

Offset
X: 0
Y: 0 px
Edge Behavior
[✓ Wrap around]
Fill with background color
Make transparent

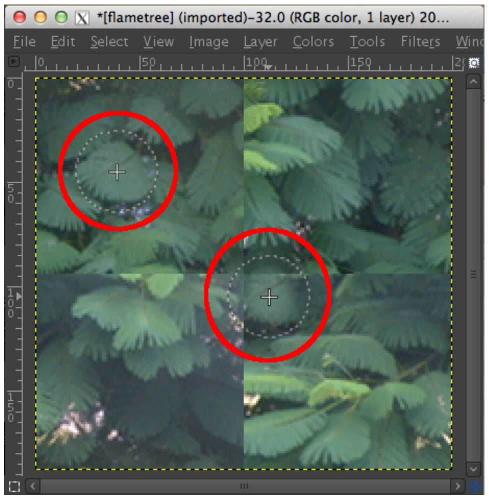
The four corners are now in the middle. The upper left and lower right corners have swapped places. The upper right and lower left have swapped places. All the tiling problems have been moved right to the middle where we will fix them manually.


*[flametree] (imported)-32.0 (RGB color, 1 layer) 20…
File Edit Select View Image Layer Colors Tools Filters Wind

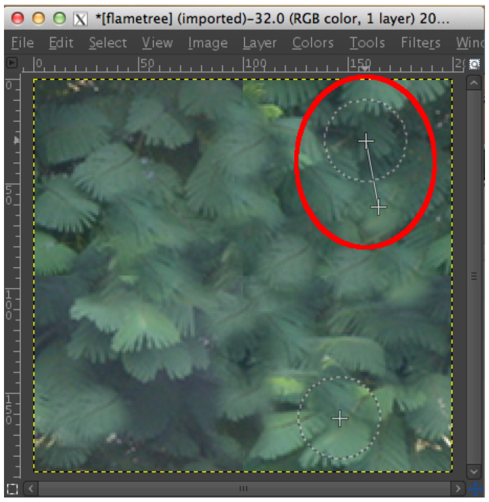
We select the clone tool. It looks like a rubber stamp. Double-click it to select it. Our cursor turns into a circle. You can adjust its size.


*[flametree] (imported)-32.0 (RGB color, 1 layer) 20...
File Edit Select View Image Layer Colors Tools Filters Wind

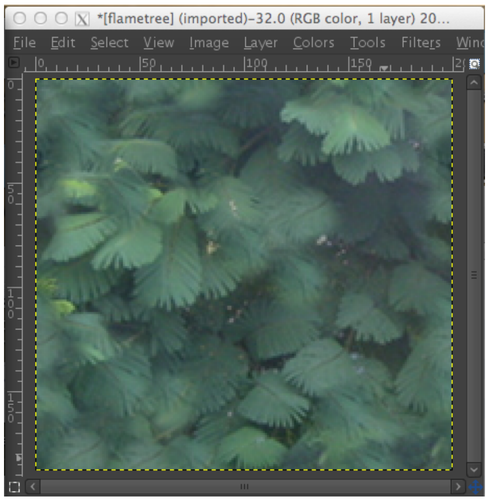
Hold down the CTRL (control) key and click on a section of the image that you would like to copy. We are copying from the upper left corner. Then move over something you want to hide and click (without using the control key). The image will be copied, but the edges will be faded so it looks smooth.


*[flametree] (imported)-32.0 (RGB color, 1 layer) 20...
File Edit Select View Image Layer Colors Tools Filters Wind

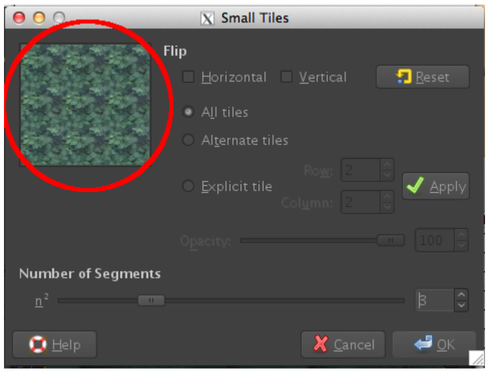
We repeat a bunch of times. CTRL click to select. Then click various places to paste. Then find something new to select. Then paste some more. Overwrite anything that looks like it would be distracting. Especially overwrite the line where the four corners come together.


*[flametree] (imported)-32.0 (RGB color, 1 layer) 20...
File Edit Select View Image Layer Colors Tools Filters Wind

Repeat the **Menu: Layer / Transform / Offset** step we did above. This will show us whether the edges still look good together. If necessary, do more cloning.


*[flametree] (imported)-32.0 (RGB color, 1 layer) 20...
File Edit Select View Image Layer Colors Tools Filters Wind

Use **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.



Small Tiles

Flip

Horizontal Vertical Reset

● All tiles

Alternate tiles

Explicit tile Row: 2 Column: 2 Apply

Opacity: 100

Number of Segments

n² 3

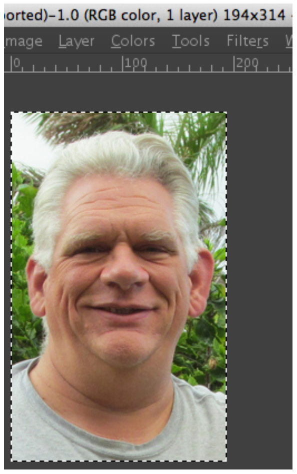
Using **Menu: Layer / Transform / Offset** and the clone tool gives you excellent control over your finished product, but takes longer. Your mileage may vary.

Chapter 24

Irregular Image Shapes

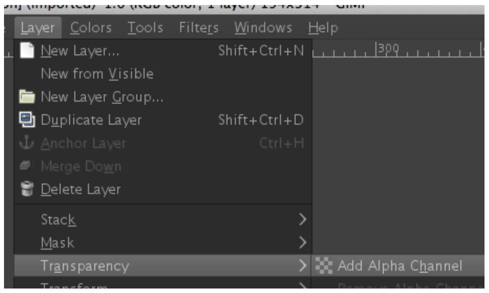
In this chapter we show you how to make an irregularly-shaped image. Really, though, the image will be rectangular just like always, but parts of it will be transparent.

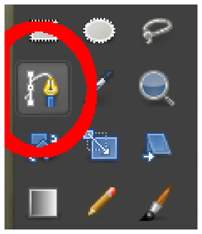
We will use a picture of myself. I have already cropped the picture to be very close to the material I want to keep. The original picture is much bigger. We will continue by deleting everything that is not me. We will make the deleted parts transparent. We will save the resulting image as a png file.


orted)–1.0 (RGB color, 1 layer) 194x314
mage Layer Colors Tools Filters
0 100 200

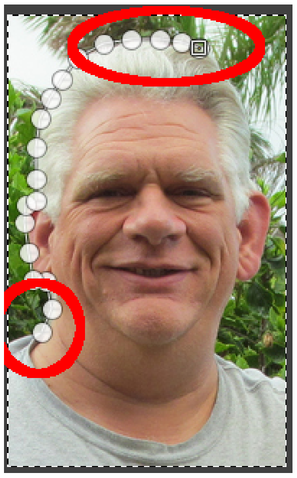
Here is the original picture we will use. It is 194x314 pixels, which will be a good size for our finished web graphic.

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

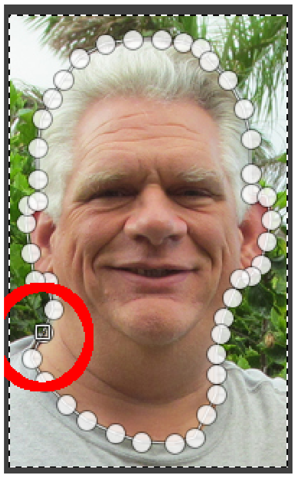




We select the fountain-pen tool by double-clicking it. This will let me click my way around the important part of the picture, doing a selection that is irregular rather than rectangular or circular. This tool is actually much more powerful that what I am showing you. It creates a Bézier curve.



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.

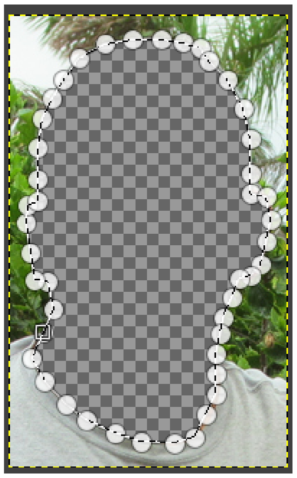


I have made it all the way around my picture. On the last click, instead of simply clicking, I hold down the control key while I click. This causes the loop to be closed, joining the last click with the first click. At this time, I can adjust any of the dots if I did not place them well. Just click on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

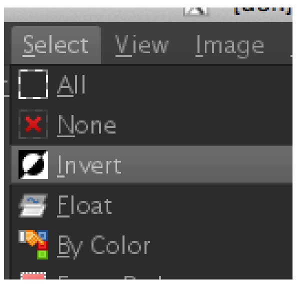
If I delete now, I will delete my face because that is what is selected.

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

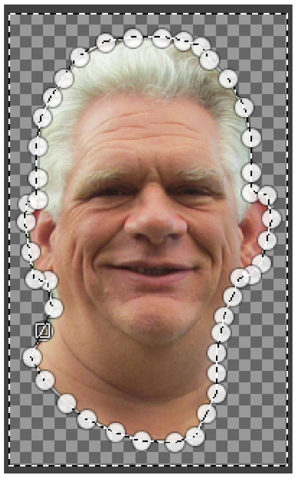
Fortunately, there is **Menu: Edit / Undo**. I use it to get me back to where I was.



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 **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.



Unit V

Content: HTML Markup

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 25

Popular Markup

In this unit, we look at the basic markup used to specify the structure and the presentation of a webpage.

In this chapter, we simply list the most popular markups that are currently in use.

`<h1>` and `</h1>` to surround a major heading.

`<p>` and `</p>` to surround a paragraph.

`<a href="URL">label</a>` to insert a **link**.

`<img src="filename.ext" />` to insert an **image**.

`<b>` and `</b>` to surround **bold** text.

`<i>` and `</i>` to surround *italic* text.

`<!--` and `-->` to surround a **comment**.

Chapter 26

Content in General

Contents

26.1 Say Something

What is a webpage for? To communicate. Communicate what? Content. If you do not have something to say, why are you making a webpage?

(If you don't have any content now, you can fake it using lorem ipsum. See section C.1, page 246, for details.)

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

Decide what your content is.

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.

And if you are really good, you can do the markup as you create the content.

But for now, generate some content.

26.2 Look For The 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.

Within paragraphs, there may be words or phrases that you want to emphasize. Adding emphasis is structural markup.

Structural Markup is the process of telling the browser about the structure of your document. Markup is discussed more in chapter 27 (page 129).

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.

Older presentational markup included things like `<center>`, which was used to center content on the screen. Much of this older markup has been deprecated, by which we mean it is being phased out and should never be used again. Bottom line, you may see old examples that show how to do things that are discouraged or not allowed currently.

Exam Question 45 (p.302)**:**
What does deprecated mean?

Acceptable Answer:
being phased out

26.3 Mark The Structure

Put `<h1>` before and `</h1>` after each of your major headings.

Put `<h2>` before and `</h2>` after each of your minor headings.

Put `<p>` before and `</p>` after each paragraph.

Put `<em>` before and `</em>` after each word or phrase you want to emphasize.

Exam Question 46 (p.302)**:**
In HTML what tag is used to present emphasized content?

Required Answer:
em

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

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

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 47 (p.302)**:**
In HTML what tag is used to present strike-through content?

Required Answer:
s

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

Exam Question 48 (p.303)**:**
In HTML what tag is used to present underlined content?

Required Answer:
u

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 49 (p.303)**:**
In HTML what tag is used to present lowered (subscripted) content?

Required Answer:
sub

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 50 (p.303)**:**
In HTML what tag is used to present raised (superscripted) content?

Required Answer:
sup

Do these things throughout your document. You are marking up your document.

26.4 Comments in HTML

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 a **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 end tag is found. Here is an example of what I mean.

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

In this example, the comment is `<!-- bbb <!-- ccc -->` because the first `-->` marks its end. The leftover part is:

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

Oops.

Exam Question 51 (p.303)**:**
What marks the start of a comment in HTML?

Required Answer:
`<!--`

Exam Question 52 (p.303)**:**
What marks the end of a comment in HTML?

Required Answer:

```
-->
```

Exam Question 53 (p.303)**:**
In HTML, can comments be nested?

Required Answer:
no

26.5 Doctype, Head, Body

Browsers are incredibly resilient. 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.

As a web designer, you should hold up your end of the contract, and provided web content that follows the official rules of the HTML language. That way you increase your chances that all browsers will display your content in the same way, the way that you want and intend.

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

The first piece of a webpage is the doctype line. It looks like this:

```
<!DOCTYPE html>
```

This specifies that the webpage is written in the HTML5 dialect of the HTML language.

The next piece of a webpage 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.

Inside the head, we need to specify the character set. It is done like this:

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

As mentioned previously, utf-8 stands for UCS Transformation Format, 8 bit. Charset is a required element in the head.

Inside the head, we need to specify the webpage title. It is done like this:

```
<title>your title goes here</title>
```

Replace “your title goes here” with whatever you want your title to be. This is a required element in the head.

Inside the head, we can also specify styles. It is done like this:

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

When you are done with the head, you say so like this:

```
</head>
```

Immediately after finishing the head, you introduce the body, like this:

```
<body>
```

Inside the body, you put all your content with its HTML markup.

When you are done with the body, you say so like this:

```
</body>
```

That ends your document.

Chapter 27

Markup in General

Contents

All HTML markup looks the same as this prototype:

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

Exam Question 54 (p.303)**:**
Give the HTML Prototype.

Acceptable Answer:
```
<tag attribute=value ...  >
```

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

tag means the first element 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 `/>`.

27.1 Paired Tags

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>`.

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 just bold text.

Exam Question 55 (p.303)**:**
In HTML the strong tag 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 56 (p.303)**:**
In HTML what word does the em tag stand for?

Acceptable Answer:
emphasis or emphasize

27.2 Tags That Stand Alone

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" />
```

27.3 Starting and Ending Markup

Exam Question 57 (p.303)**:**
What character(s) mark the start of HTML markup?

Required Answer:
`<`

Exam Question 58 (p.303)**:**
What character(s) mark the end of HTML markup, assuming that it is NOT a void tag?

Required Answer:

`>`

Exam Question 59 (p.303):
What character(s) mark the end of HTML markup, assuming that it IS a void tag?

Required Answer:
`/>`

Exam Question 60 (p.303):
When a tag is void, what does that mean?

Acceptable Answer:
It cannot have a separate end tag.

27.4 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 61 (p.303):
In HTML does the order of attributes matter?

Required Answer:
no

27.4.1 Blank 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 `""`)

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

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

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.

27.4.2 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.

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

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

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

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

27.4.3 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 62 (p.303)**:**
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 63 (p.303)**:**
If an HTML attribute's value includes a double-quote (`"`) does that force it to be quote marked?

er:
yes

63 1 If an HTML attribute's value includes a double-quote (") does that force it to be quote marked? 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 64 (p.303)**:**
If an HTML attribute's value includes a less-than (`<`) does that force it to be quote marked?

er:
yes

64 1 If an HTML attribute's value includes a less-than (¡) does that force it to be quote marked? 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 65 (p.304)**:**
If an HTML attribute's value includes a greater-than (`>`) does that force it to be quote marked?

er:
yes

65 1 If an HTML attribute's value includes a greater-than (¿) does that force it to be quote marked? 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 66 (p.304):
If an HTML attribute's value includes a single-quote (`'`) does that force it to be quote marked?

er:
yes

66 1 If an HTML attribute's value includes a single-quote (') does that force it to be quote marked? yes

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

Exam Question 67 (p.304):
If an HTML attribute's value includes a back-quote (`` ` ``) does that force it to be quote marked?

er:
yes

67 1 If an HTML attribute's value includes a back-quote (‘) does that force it to be quote marked? 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 68 (p.304):
If an HTML attribute's value includes an equals (`=`) does that force it to be quote marked?

er:
yes

68 1 If an HTML attribute's value includes an equals (=) does that force it to be quote marked? yes

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

Exam Question 69 (p.304):
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 70 (p.304)**:**
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 71 (p.304)**:**
If an HTML attribute's value includes a colon (`:`) does that force it to be quote marked?

er:
no

71 1 If an HTML attribute's value includes a colon (:) does that force it to be quote marked? 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 72 (p.304)**:**
If an HTML attribute's value includes a dot (`.`) does that force it to be quote marked?

er:
no

72 1 If an HTML attribute's value includes a dot (.) does that force it to be quote marked? no

Exam Question 73 (p.304)**:**
If an HTML attribute's value includes a dash (`-`) does that force it to be quote marked?

er:
no

73 1 If an HTML attribute's value includes a dash (-) does that force it to

be quote marked? 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 74 (p.304)**:**
If an HTML attribute's value includes a slash (`/`) does that force it to be quote marked?

er:
no

74 1 If an HTML attribute's value includes a slash (/) does that force it to be quote marked? 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 75 (p.304)**:**
If an HTML attribute's value includes a percent (`%`) does that force it to be quote marked?

er:
no

75 1 If an HTML attribute's value includes a percent (%) does that force it to be quote marked? 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 76 (p.304)**:**
If an HTML attribute's value includes an ampersand (`&`) does that force it to be quote marked?

er:
no

76 1 If an HTML attribute's value includes an ampersand (&) does that force it to be quote marked? 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.

27.4.4 Character References

If you are quote-marking something, 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)

Appendix H (page 263) has more character references for things like copyright and registered trademark and special symbols. There are thousands of these character references supported in HTML.

27.5 Commonly Used Attributes

Each tag can have certain common attributes. We list them here.

style= : For 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.

id= : For 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.

class= : For 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.

title= : For example, `title='more information'`. This gives you a way to provide some hidden text. When the mouse hovers over the item for long enough, maybe a half a second, the title will appear. This is a good way to provide hints or extra information that would otherwise clutter the screen.

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

27.6 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 77 (p.304)**:**
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. Trees cannot handle overlaps.

We talk more about the DOM in Chapter 56 (page 238).

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

Exam Question 78 (p.304)**:**
What is wrong with this HTML tag order: a b /a /b?

Acceptable Answer:
overlap

Exam Question 79 (p.304)**:**
What is wrong with this HTML tag order: a b /b /a?

Acceptable Answer:
nothing

Chapter 28

Headings and Paragraphs

Contents

28.1 Headings

Headings provide a fast way for your reader to understand your web page. They provide sign posts to your content.

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

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>`.

Exam Question 80 (p.305)**:**
In HTML what tag is used to identify a second-level heading?

Required Answer:
h2

28.2 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.

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

28.3 Line Breaks

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

One example of this might be writing poetry or verse. For example, here is part of a poem 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 more.

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

Exam Question 81 (p.305)**:**
In HTML what tag is used to insert a line-break?

Required Answer:
br

Chapter 29

Global HTML 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 82 (p.305)**:**
List the five commonly-used global attributes for HTML tags.

Required Answer:
style, id, class, title, hidden

class= specifies the class of the item. If you have more than one class, separate them by spaces and enclose the list in quote marks. Classes can be shared by several items.

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

id= specifies the ID of the item. The ID must be unique. It cannot be shared by another item.

style= specifies any CSS styling attributes that may be desired. This styling will take priority over every other source of styling directions.

title= specifies text that will be displayed if the user hovers their mouse over the image for a second or more.

Chapter 30

Images

Besides text, the most popular thing to include in a webpage is an image.

The image tag is `<img ... />`.

The commonly-used attributes of image are:

src= specifies the source of the image. This is a URL. See Chapter 31 (page 145) for more.

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

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.

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.

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.

Exam Question 83 (p.305)**:**
List the four commonly-used attributes of the image tag.

Required Answer:
src, alt, width, height

Here is another handy attribute, but not as commonly used.

Chapter 31

Connecting With External Resources

Contents

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.

The following chapter, Chapter 32 (page 148), looks into the issues of relative, docroot, and absolute addressing for URLs. This current chapter is focused on where these links occur and how they are expressed.

31.1 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.

31.2 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.

31.2.1 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.)

31.2.2 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"
```

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 established as a standard part of the HTML language.

31.3 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.

31.4 link

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

```
<link rel="stylesheet" type="text/css" href="xxx" />
```

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

Chapter 32

Relative, DocRoot, and Absolute URLs

Contents

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.

32.1 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.

32.2 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 probably need no correction because they are still right.

32.3 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`

A file in the next larger 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`

32.4 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).

32.5 img src="..."

When creating an image (img) tag, the source (src) must be specified. It can be (R) relative, (D) docroot, or (A) absolute.

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

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

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

32.6 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.

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

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

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

32.7 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 a 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 it might actually be someplace else.

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

Exam Question 84 (p.305)**:**
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 85 (p.305)**:**
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 86 (p.305)**:**
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 87 (p.305)**:**
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/

Chapter 33

Lists, Ordered and Unordered

Lists can be ordered or unordered. If the list is ordered, each item is introduced by a number or similar thing that counts its position in the list. If the list is unordered, each item is introduced by a symbol called a **bullet**.

Exam Question 88 (p.306)**:**
In HTML what words does the ol tag stand for?

Required Answer:
ordered list

Exam Question 89 (p.306)**:**
In HTML what tag is used to introduce a numbered list?

Required Answer:
ol

The numbering scheme for an ordered list is controlled by the **start**, **type**, and **reversed** attributes.

`http://www.w3schools.com/tags/tag_ol.asp` has more on ordered lists.

Unordered lists simply have bullets. They are styled using CSS **list-style** rather than direct attributes.

`http://www.w3schools.com/cssref/pr_list-style.asp` has list styling options.

`http://www.w3schools.com/tags/tag_ul.asp` has more on unordered lists.

Exam Question 90 (p.306)**:**
In HTML what words does the ul tag stand for?

Required Answer:
unordered list

Exam Question 91 (p.306)**:**
In HTML what tag is used to introduce an un-numbered list?

Required Answer:
ul

Exam Question 92 (p.306)**:**
In HTML what words does the li tag stand for?

Required Answer:
list item

Exam Question 93 (p.306)**:**
In HTML what tag is used to introduce an entry in a list?

Required Answer:
li

Chapter 34

Tables

Contents

Tables have been very popular in the past, and have been grossly abused as a mechanism for controlling the **layout** of webpages. If you go back to some old-timey tutorials, you will learn more than you should about using tables to position the graphical elements of webpages.

The problem with that is it is brittle. It does not slide gently into the future.

Still, tables have a useful role for reporting things that really are tabular.

table: A table is introduced by the `<table>` tag. It is concluded by the `</table>` tag.

tr: Between the table tags, the table is made up of zero or more table rows. Each table row starts with a `<tr>` tag and ends with a `</tr>` tag.

td: Between the tr tags, the table row is made up of zero or more table data elements. Each table data element starts with a `<td>` tag and ends with a `</td>` tag.

th: Table header tags are like td tags. They can appear in a table row. Each table header element starts with a `<th>` tag and ends with a `</th>` tag.

Here is an example of a simple table:

```
<table>
```

```
<tr><th>Left Header</th><th>Right Header</th></tr>
<tr><td>Upper Left</td><td>Upper Right</td></tr>
<tr><td>Middle Left</td><td>Middle Right</td></tr>
<tr><td>Lower Left</td><td>Lower Right</td></tr>
</table>
```

The result would look something like this:

Left Header	**Right Header**
Upper Left	Upper Right
Middle Left	Middle Right
Lower Left	Lower Right

Browsers will normally allow you to leave out the `</tr>`, `</td>`, and `</th>` tags.

Exam Question 94 (p.306)**:**
In HTML what tag is used to introduce a table?

Required Answer:
table

Exam Question 95 (p.306)**:**
In HTML what tag is used to introduce a row within a table?

Required Answer:
tr

Exam Question 96 (p.306)**:**
In HTML what tag is used to introduce a cell within a table row?

Required Answer:
td

Exam Question 97 (p.306)**:**
In HTML what table attribute is used to merge cells horizontally?

Required Answer:
colspan=

Exam Question 98 (p.306)**:**
In HTML what table attribute is used to merge cells vertically?

Required Answer:
rowspan=

`http://www.w3.org/wiki/Styling_tables` tells how to style tables.

34.1 Cell Width

Things like **min-width:** and **max-width:** and **min-height:** and **max-height:** apply to many parts of a webpage, but they do not work on table cells.

You can specify the minimum size of a cell by specifying its CSS **width:** attribute.

You can use **white-space: nowrap** to keep the cell contents all on one line, which can make the line wider.

Chapter 35

Divs and Spans

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 do group things together. They just 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.

`<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.

`<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.

Exam Question 99 (p.306)**:**
Can a div legally appear within another div?

Required Answer:
yes

Exam Question 100 (p.306)**:**
Can a p legally appear within a div?

Required Answer:

yes

Exam Question 101 (p.306)**:**
Can a div legally appear within a p?

Required Answer:
no

Exam Question 102 (p.306)**:**
Can a p legally appear within another p?

Required Answer:
no

Exam Question 103 (p.307)**:**
Can a span legally appear within another span?

Required Answer:
yes

Exam Question 104 (p.307)**:**
Can a span legally appear within a p?

Required Answer:
yes

Exam Question 105 (p.307)**:**
Can a p legally appear within a span?

Required Answer:
no

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.

Chapter 36

Forms

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. This customization depends on that two-way communication. Two-way communication uses **CGI** programs. We talk briefly about CGI programs in section 36.8 (page 170) 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.

36.1 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.

The form has several commonly used attributes.

method= can be `post` or `get`.

action= specifies the program that will process the inputs if the form is submitted.

Within the form, there can be any number if input fields.

`<input/>` is the tag for an input field. More on this below.

One special input field is the text area. It is a rectangular space, with a certain number of rows and columns.

`<textarea>` is the tag to start a text area.

`</textarea>` is the tag to end a text area.

The **textarea** tag has a name but no value. The value is whatever appears between the **textarea** tag and its ending tag, or whatever is entered into that space by the user.

The **select** tag provides for a drop-down list of options. Each is specified by an **option** tag. They must be used together. The select tag has a name attribute but no value. The option tag has a value attribute but no name.

Exam Question 106 (p.307)**:**
List the three form input tags.

Required Answer:
input, textarea, select

36.2 Every Input Field

No matter which input type is used, it must have a name.

It can also be designated by **autofocus** to receive the cursor when the page is loaded.

And it can (probably) participate in **taborder**, which tells which field you go to next when you press the tab key.

36.2.1 name=value

In every case, 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 may be provided by the user.

The only thing that is transmitted is **name=value**.

Characters like A-Z and 0-9 are transmitted as written.

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.

36.2.2 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.

The name parameter is not easily changed by your user, unless they are a hacker. Probably even then they would not want to change it.

36.3 The input Tag

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.

We use ... to indicate that more attributes will be included in the input tag.

36.3.1 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 the value parameter.

36.3.2 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.

Exam Question 107 (p.307)**:**
List the six non-button type= attributes for the input tag.

Required Answer:
text, password, checkbox, radio, file, hidden

Exam Question 108 (p.307)**:**
List the four button type= attributes for the input tag.

Required Answer:
submit, reset, button, image

There are several new (HTML5) type= attributes. These are discussed in the next section.

36.3.3 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 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, value, size.

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.

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: maxlength, pattern.

The **maxlength** parameter specifies how many characters will be accepted. If you set maxlength=5 and they try typing in 6 characters, on the first 5 will appear.

The **pattern** parameter specifies a regular expression that must be obeyed. See Appendix L (page 284) for more on regular expressions.

36.3.4 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.

36.3.5 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"`

Note that `checked="anything"` probably works.

Note that `checked="false"` does not do what you might think.

36.3.6 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.

Several such spaces are usually associated 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"`

Note that `checked="anything"` probably works.

Note that `checked="false"` does not do what you might think.

36.3.7 type=file

`<input type=file ... />` is the tag for uploading a file.

The entire file will be uploaded when the form is submitted.

36.3.8 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.

36.4 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 109 (p.307)**:**
What HTML attribute causes the cursor to begin 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 110 (p.307)**:**
What HTML attribute specifies the order in which fields are reached by tabbing?

Required Answer:
tabindex=

36.5 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, type=week.

For numbers: type=number, type=range.

For special types of data: type=email, type=url, type=tel, type=search.

For colors: type=color.

`http://www.w3schools.com/html/html5_form_input_types.asp` has more.

36.6 Tab Order

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.

`http://www.w3schools.com/tags/att_global_tabindex.asp` has more.

36.7 Input Patterns

Appendix L (page 284) has more on patterns, but here is a brief overview.

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}" ... />
```

For a ten-digit telephone number, xxx-xxx-xxxx, you could say:

```
<input ... pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}" ... />
```

Using a pattern does not guarantee that the browser will enforce it, but most modern browsers do.

36.8 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.

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.

36.9 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 or weather.

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 that it is an advanced feature of website design.

Unit VI

Presentation: CSS Style

Sources:

http://www.w3.org/TR/CSS2/ provides the most authoritative information about version 2.1 of CSS.

There does not seem to currently be an authoritative standard for CSS3.

http://www.w3.org/TR/#tr_CSS provides a history of W3C documents about CSS, and is the place where one could find whether a newer specification has been published.

Chapter 37

Styling Quick Start

Contents

As part of the head of your webpage, somewhere between `<head>` and `</head>`, you should have a style section. We have seen this before. Here is an example using colors.

37.1 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 type="text/css">
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.

Comments in the style section are delimited by `/*` at the front, and `*/` at the end.

Exam Question 111 (p.307)**:**
What marks the start of a comment in CSS?

Required Answer:
`/*`

Exam Question 112 (p.307)**:**
What marks the end of a comment in CSS?

Required Answer:
`*/`

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 talk more about color and backgrounds in chapter 39 (page 181).

37.2 Centering

Text can be centered, or it can be justified to the left, to the right, or to both sides of the column. We use the text-align attribute.

```
text-align: center;
text-align: left;
text-align: right;
text-align: justify;
```

Exam Question 113 (p.307)**:**
List the four text-align: options.

Required Answer:
left, right, center, justify

37.3 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 41.6 (page 192) talks about the different kinds of dimensions you can use, including px (pixel), in (inch), cm, and mm.

Pattern: Section 41.2 (page 189) talks about the different kinds of borders you can use, including solid, dotted, dashed, and double.

Color: Chapter 39 (page 181) talks 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 41 (page 188) provides coverage.

37.4 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 42 (page 194) goes in depth with fonts. In particular, see section 42.3 (page 199) 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 */
```

We can also control the font size.

```
font-size: 24pt; /* 24 points */
font-size: 150%; /* 1.5 x normal */
```

Chapter 38

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.

38.1 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;
```

38.2 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%
```

38.3 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.

38.4 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(`255255,255,0.5);,

Shade (darken) the background, and make it partly transparent.

`background-color: rgba(`00,0,0.5);,

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="background color: rgba(255,255,255,0.5);">
Put your content here.
</div></body>
```

38.5 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

38.6 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.

Chapter H (page 263) 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>`

Chapter 39

Colors and Backgrounds

Contents

Plain backgrounds are boring. We like color. We like pictures. CSS allows us to specify backgrounds that are automatically generated, such as colors or gradients, or that are based on images.

Backgrounds can be specified for the whole page or for individual elements of the page.

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

39.1 Color in General

It is very common to specify colors by hex value or by color name.

Section O.2 (page 291) has a color wheel and hex codes.

It is also possible to specify them using **RGB** or **RGBA** (red, green, blue, alpha).

Alpha refers to **transparency**, and ranges from 0.0 (transparent) to 1.0 (**opaque**).

It is also possible to specify them using **HSL** or **HSLA** (hue, saturation, lightness, alpha).

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. 0% is gray. 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. Shade means to add black.

See section O.2 (page 291) for a color wheel and hex codes.

`http://www.w3schools.com/cssref/css_colors_legal.asp` has more on specifying colors in general.

39.2 Color of Text

You can specify the color of text by writing `color: xxx;` where the xxx is replaced by a legal color value.

39.3 Color Behind Text (Background)

You can specify the background color of text by writing `background-color: xxx;` where the xxx is replaced by a legal color value.

39.4 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.

39.5 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.

39.6 Gradients

Gradients are a way to create a background that shades gradually from one color to another.

todo: more to be added

The ColorZilla website has a gradient generator.

39.7 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 40

Untangling HTML and Style

You will have noticed that in our examples, we placed `style="..."` directly into the HTML of the webpage.

This is called in-line styling.

In-line styling is bad because it mixes HTML and styling together. 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?

40.1 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.

40.2 The Cascade

In-line beats internal. Internal beats external.

The most general solution to styling a website is to create a single stylesheet and apply it to all the webpages in the website.

This is called an external stylesheet.

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 stylesheet. 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 stylesheet.

When a style is specified in both an external stylesheet and and internal stylesheet, the internal stylesheet 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 stylesheet, the in-line style takes priority.

40.3 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

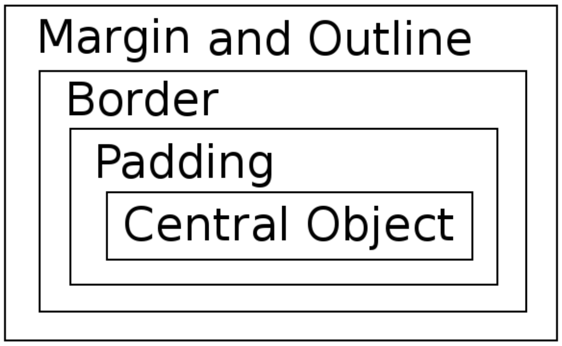
Chapter 41

The 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. CSS provides a simple but powerful system for telling the browser just how much space to leave, and what to fill it with.

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.



`http://www.w3schools.com/css/css_boxmodel.asp` has more on the box model.

Exam Question 114 (p.308)**:**
List the four major attributes of the box model.

Required Answer:
padding, border, margin, outline

41.1 Padding

Closest to the object is the padding. It uses the same background color and background image as the central object.

Padding has thickness. Changing the thickness will cause the page to reflow.

41.2 Border

Outside the padding is the border. It has thickness, color, and style.

If the border thickness is zero, the border is invisible.

The border has style options: none (invisible, zero-width border), hidden (similar to none), dotted, dashed, solid, double, groove (border carved inward), ridge (border poking outward), inset (content inward), and outset (content outward).

Exam Question 115 (p.308)**:**
List the ten border-style: options.

Required Answer:
none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset

Thickness, color, and style can vary by side. If you want, you can have four different thicknesses, four different colors, and four different styles.

Changing the thickness will cause the page to reflow. Changing the color or style will not cause the page to reflow.

`http://www.w3schools.com/css/css_border.asp` has more.

41.3 Margin

Next out from the border is the margin.

Margin has thickness but not color or style. The margin is always transparent.

Adjacent vertical (top and bottom) margins may collapse in some cases. In that case, the larger of the two margins will be used.

Changing the thickness will cause the page to reflow.

You can use negative values to make things overlap.

`http://www.w3schools.com/css/css_margin.asp` has more.

41.4 Outline

Outline is rarely used. It begins just outside the border and overlaps the margin and maybe beyond. It is useful for highlighting something temporarily.

Like the border, the outline has thickness, color, and style.

Unlike the border, the outline does not take up any space. Changing the thickness does NOT cause the page to reflow. It simply overlaps whatever is in its way. This supports its use in temporary highlighting.

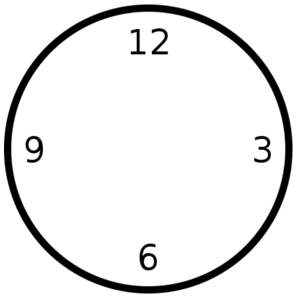
Outline has the same color options that the border does, plus one more: invert.

Outline has the same style options that the border does.

These things can NOT vary by side. You can only have one color, one thickness, and one style.

41.5 Clock Order

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 clock order. Clock order is (12) top, (3) right, (6) bottom, (9) 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 116 (p.308)**:**
When we say margin: 9px 8px 7px 6px; which margin is 9px?

Required Answer:
top

Exam Question 117 (p.308)**:**
When we say margin: 9px 8px 7px 6px; which margin is 8px?

Required Answer:
right

Exam Question 118 (p.308)**:**
When we say margin: 9px 8px 7px 6px; which margin is 7px?

Required Answer:
bottom

Exam Question 119 (p.308)**:**
When we say margin: 9px 8px 7px 6px; which margin is 6px?

Required Answer:
left

Exam Question 120 (p.308)**:**
When we say margin: 9px 8px 7px; what is the implied fourth value?

Required Answer:
8px

Exam Question 121 (p.308)**:**
When we say margin: 9px 8px; what is the implied third value?

Required Answer:

9px

Exam Question 122 (p.308)**:**
When we say margin: 9px 8px; what is the implied fourth value?

Required Answer:
8px

41.6 Dimension Units

Distance is measured in any of several ways.

Some, like **px** (**pixel**), are designed to be displayed on a screen.

Some, like **in** (**inch**), **cm**, and **mm**, are designed to work well with physically printed media or PDF files.

Some, like **em**, **ex**, **pt** (**point**), **pc** (**pica**), and percentage, are designed to scale up or down as appropriate for the nearby content.

px: Distance can be measured in pixels. A pixel is one dot on the screen. This is the natural measurement the computer must eventually use to render things.

pt: Distance can be measured in points. A point is about 1/72 of an inch.

pc: Distance can be measured in picas. A pica is 12 points, 1/6 of an inch.

in: Distance can be measured in inches. The browser will adopt some conversion factor between inches and pixels for displaying things on the screen.

cm: Distance can be measured in centimeters.

mm: Distance can be measured in millimeters.

em: Distance can be measured in "em"s, where one em is the width of the letter m in the current font.

ex: Distance can sometimes be measured in "ex"s, where one ex is the height of the letter x in the current font. An ex is often about half of an em.

percentage: Distance can be measured by percentage, in which case it is based on the size of the central object (interior content).

`http://www.w3schools.com/cssref/css_units.asp` has more.

41.7 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 items 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.

Chapter 42

Fonts and Text

Contents

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 slows down people when they try to read it.

We also talk about text, and some ways it can be modified.

42.1 What Is A Font?

A font is a typeface. The origin of the word font is related to fondue. The letters used in printing were originally made from melted lead. (The distance between lines of text is still called leading, pronounced led-ing.)

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.

42.1.1 Word Processing Font Choices

When using a word processor like Microsoft Word, a variety, sometimes a huge variety, of fonts 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 unspoken nightmare of word processors 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.

42.1.2 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.

42.1.3 Raster and 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.

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.

42.1.4 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 notion.

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.

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.

42.1.5 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 42.3 (page 199) for more information.

42.2 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 123 (p.308)**:**
List the five generic font families.

Required Answer:
serif, sans-serif, cursive, fantasy, monospace

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.

Common examples of serif typefaces: Baskerville, Bookman, Garamond, Georgia, Palatino, Times New Roman, Times.

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.

Sans-serif means without serifs. Sans is the latin word for "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.

Common examples of sans-serif typefaces: Arial, Geneva, Helvetica, Impact, Lucida Sans, Tahoma, 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.

Monospace means one 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.

Monospace: `iiiii jjjjj mmmmm wwwww`

Proportional: iiiii jjjjj mmmmm wwwww

Examples of monospace: Courier, Lucida Console, Monaco.

Also, any font with the word mono in its name is probably a monospace font.

Cursive fonts try to connect adjacent letters in a flowing manner, like human handwriting. 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.

Examples of cursive: Apple Chancery, Zapf Chancery, Comic Sans.

Also, any font with the word cursive or script or manuscript in its name is probably a cursive font.

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.

Examples of fantasy: Papyrus.

`http://cssfontstack.com/` has a list of fonts.

42.3 Font-Family: 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.

Cross-Platform Awareness: Try to include an acceptable font for each platform your users might be using: Microsoft Windows, Mac OS X, Mac iOS (iPhone or tablet), Android (phone or tablet), Linux.

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.

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, it must be in quotes.

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 semi-colon).

42.3.1 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.

42.4 Text Shadowing

CSS3 provides the ability to display a shadow behind a font. This is almost like creating new fonts. This can look very cool. Or it can look awful. Use your artistic judgment.

Exam Question 124 (p.308)**:**
What CSS attribute does shadowing of your text font?

Required Answer:
text-shadow:

The **text-shadow:** property can be used. Here is an example.

```
text-shadow: 1px 2px 3px red;
```

Even though this is officially a new feature in CSS3, it is already supported by most modern browsers.

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:

Set 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 background. Then the only thing visible is the shadow.

You can also do this with the **opacity** parameter.

Google search "css text effects" for more ideas.

42.5 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.

42.5.1 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.

42.6 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.

42.6.1 Google Web Fonts (Free)

`http://www.google.com/webfonts` 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). 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.

42.6.2 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.

42.7 Other Font and Text Properties

We have spent a lot of time talking about fonts themselves. There are some other properties you can use.

42.7.1 Font Adjustments

You can specify the font size, style, variant, and weight.

```
font-size: 14pt;
```

font-size: can be measured in many different ways, including points.

Old-style HTML might show this as `<font size=3>`. The old style is deprecated. Do not use it.

text-size: is not an attribute. Use **font-size:** instead.

font-style: can be normal, italic, or oblique (which means the same as italic).

font-variant: can be normal or small-caps.

font-weight: can be normal, bold, or an x-hundred number like 100, 200, ..., 900.

42.7.2 Text Adjustments

The normal settings are probably fine in most cases. But there are a few attributes you can mess with to change how your text appears.

For more information than is presented here, you can do a Google search. It will turn up lots of examples.

background-color: changes the color of field behind the text.

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.

color: changes the color of the text.

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-align: can be left, right, center, or justify.

text-decoration: can be none, underline, overline, line-through, or blink. (Please avoid blink.)

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 43

Positioning

Contents

(todo: Write a lot more in this chapter.)

Positioning is mostly about relative and absolute positioning. But there are a few important other things to think about, and they also fall under this heading.

43.1 Horizontal Centering

To center some text within a paragraph or a div or the whole body, we can do this.

```
text-align: center;
```

To center a block of whatever, we can use margin: auto to assist us.

```
margin: auto;
```

43.2 Vertical Centering

http://blog.themeforest.net/tutorials/vertical-centering-with-css/ is a very good blog post about vertical centering.

43.3 Float and Clear

The box model applies here. We start with the understanding that everything on a webpage fits into a box, and is surrounded by padding, border, (outline,) and margin. Normally these boxes are just stacked horizontally or vertically on the page.

Float lets them be someplace else.

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?

(todo: write more)

43.4 Absolute and Relative

(todo: write more)

Chapter 44

ID

Contents

The one-character symbol for ID is # (hash).

Exam Question 125 (p.309)**:**
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.

44.1 id=

44.2 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">`

Chapter 45

Class

Contents

The one-character symbol for class is `.` (dot).

It is used in CSS to indicated that a particular selector is a class selector.

45.1 class=

Chapter 46

Pseudo-Classes

The one-character symbol for pseudo-class is : (colon).

Exam Question 126 (p.309)**:**
What is the one-character symbol for pseudo-class?

Required Answer:
:

Exam Question 127 (p.309)**:**
What does pseudo mean?

Acceptable Answer:
fake

:hover - change the style of an element when the mouse rolls across it.

:visited - change the style of an a-link element when we can tell it has been visited.

:active - change the style of an a-link element when we are actively clicking on it.

:first-line - style the first line 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.

:nth-child - Use `:nth-child(2n)` to style alternating list items or table rows to have different background colors.

Chapter 47

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 48

Transitions and Animations

Contents

In Chapter 46 (page 212) we introduced pseudo classes. Some of these are dynamic, and let your webpage respond to user behavior. (Others of these are static, and let you style based on things like being the first line or the first character.)

Transitions and animations work in the current versions of all major browsers. Internet Explorer before version 10 does not work.

48.1 Simple Example

Let's start with a simple example.

```
<style>
div { transition: all 2s; background-color: white; }
div:hover { background-color: red; }
</style>
<body>
<div>This text has a changing background color.</div>
```

```
</body>
```

Make that a webpage. Load it. Then mouse over the text.

48.2 Transitions

For things that are dynamic, the **:hover** attribute is probably the greatest single 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;

Google search “css transition” for lots of tutorials.

`http://dev.w3.org/csswg/css-transitions/` has authoritative information about transitions.

48.3 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;`

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.

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.

Property: You cannot omit this. The property is the thing that will be changed gradually. Common examples include width, height, background-color, color, opacity, font-size, and padding.

`https://developer.mozilla.org/en-US/docs/CSS/CSS_animated_properties` gives a list of the properties that can be animated.

You can simply specify the property as "all" and all changed properties will follow the transition timeline.

48.4 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, background-color 4s;
```

Unit VII

Action: JavaScript Programs

This is probably the shortest unit of the book, mostly because JavaScript can get beyond "beginner" very quickly. We will provide a basic overview and give you some basic skills. Then, when you are curious, you can wade into the deeper waters and search the Internet or elsewhere for more advanced information.

When we refer to JavaScript, we are specifically interested in Client-Side JavaScript, by which we mean the JavaScript that is run within a web browser itself. There are other settings in which JavaScript can be used, but we will not be interested in them here.

Exam Question 128 (p.309)**:**
In DHTML, what does the D stand for?

Required Answer:
dynamic

The original method of getting "action" (changing what you see in your browser) is to follow a link or submit a form (press a Submit button).

Off-page links and forms get a server involved. This uses bandwidth, which eats into your data cap, if you have one. Reducing your bandwidth consumption can be important especially for mobile platforms that can have very tight data caps.

JavaScript can make things happen without using bandwidth or involving the server. Or if bandwidth is used, it can be kept to a minimum.

JavaScript is used to add action or animation to a webpage. It can modify all or part of the current page without reloading it.

JavaScript is powerful. It is actually a Turing-complete programming language, which means it is just as powerful as C++ or any other programming language.

Because it is so powerful and this is an introductory textbook, we will only barely scratch the surface on how to use it and what is possible with it.

Note: JavaScript and Java are very much unrelated. People sometimes think they are the same. They are not. They share the same first four letters in their name. That's about it.

Exam Question 129 (p.309)**:**
Are Java and JavaScript the same thing?

Acceptable Answer:

no

Exam Question 130 (p.309)**:**

Are Java and JavaScript related?

Acceptable Answer:

no

Chapter 49

A Little JavaScript Fun

The purpose of this chapter is to help you see how JavaScript can make changes to your webpage dynamically.

This is not a programming class. We do not expect you to invent your own JavaScript code. We **do** hope you will be able to take (cut and paste) code others have written and make simple customizations that suit your needs.

49.1 Body Style Colors

Let's start out by adding 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>
</p>
```

The colors must have quote marks around them. Otherwise, they will be treated as JavaScript variables that are 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 an 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.

49.2 Camel Case

In **CSS**, we say things like this.

```
body { background-color: white; }
```

In the **DOM**, we say things like this.

```
document.body.style.backgroundColor='white';
```

We talk more about the DOM in Chapter 56 (page 238).

The two ways are intentionally very similar.

Anything you can do in CSS, you can also do in JavaScript. Anything you can specify in CSS you can change using JavaScript.

One thing we wish to point out here is the use of Camel Case for naming the background color attribute.

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.

49.3 Any Function

In the body.style.color section, we used JavaScript to immediately modify an attribute of body. We are doing exactly one thing.

Instead of doing it immediately, we can **call** a JavaScript **function** that does anything we tell it to do. It can include many steps. It can be a complete and very complicated program.

We will keep things simple. In this example, we create a function named `colors` and 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>
</p>

<script type="text/javascript">
function colors(fore,back) {
  document.body.style.color=fore;
  document.body.style.backgroundColor=back; }
</script>
```

49.4 Where Does JavaScript Go?

On your webpage, where should you put the JavaScript?

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.

Chapter 50

JavaScript to Change Class

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.

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 type="text/css">
.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.</p>
```

Chapter 51

JavaScript to 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.

In this chapter, we will 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>
<p id="abc" hidden>To get to the other side!
<u onclick='document.getElementById("abc").hidden=true'>
[hide]</u></p>
```

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.

Chapter 52

JavaScript for External Content

Contents

An html webpage can pull in content from other sources. One common way to do this is by using `<script src=...>` to bring in JavaScript. (Only bring in JavaScript you trust.)

In this chapter we show the basics of how to use JavaScript to pull in externally-generated content. In this case, we will pull in the current date. We will insert the date directly into our webpage.

52.1 The /date/ Script Request

Reminder: Only insert scripts you trust. Scripts are enormously powerful and can do amazing things, including changing your links or content. Do you want to turn your webpage into an advertisement for who knows what?

Add this to your webpage.

```
<p>The date is <script src="/date/"></script> today.</p>
```

This will cause your webpage to include the following HTML.

`<p>The date is ... today.</p>`

The ... part will be created based on the JavaScript that is provided by the script at `/date/`.

`/date/` is a program right on your own server, and totally under your control.

52.2 The /date/ CGI Program

Teaching you to write CGI is really beyond the scope of this book. However, we will go so far as to provide you with a simple CGI program that you can key in and use. We want you to see what is involved. We do not expect you to write your own programs.

The following CGI program is needed at `/date/` to provide the current date.

Create the following directory.

`public_html/date`

In that directory, create the file `index.cgi` and change its permissions to be 755. (755 makes it an executible program instead of a webpage.)

Note that index ends in `.cgi` rather than the `.html` that we have normally used. That is because this is a CGI program instead of an HTML webpage.

For your **index.cgi** file put in the following contents. Below the program, we explain what each line means.

```
#! /usr/bin/perl --
( $sec, $min, $hr, $d, $m, $y ) = localtime();
$y += 1900; $m++;
print "content-type: text/javascript\n";
print "cache-control: no-cache\n" ;# prevent caching
print "\n" ;# end of headers
print "document.write(\"$m/$d/$y\");" ;# the javascript
```

If you used copy-and-paste, double-check all the quote marks to make sure they came through properly.

The lines have the following meanings.

`#! /usr/bin/perl --`

The `#!` tells the server that this file is a script. The `/usr/bin/perl` part says where the server can find a program that can understand the rest of the script. In this case, it says to use the Perl interpreter.

Note: You may need to use `/usr/local/bin/perl` or something else. Your webhosting provider should make that information easily available.

```
( $sec, $min, $hr, $d, $m, $y ) = localtime();
```

This tells Perl to grab the local time and to place the first six parameters into variables.

```
$y += 1900; $m++;
```

This tells Perl to add 1900 to the `$y` variable, and to add one to the `$m` variable. `$y` stands for year, and measures the number of years since 1900. `$m` stands for month and tells the month number, counting from zero.

```
print "content-type: text/javascript\n";
```

This tells Perl to start talking to the browser. The browser is expecting JavaScript. We announce that we are giving it JavaScript.

```
print "cache-control: no-cache\n" ;# prevent caching
```

This tells Perl to tell the browser to please not cache the contents of this file. If you need it again, ask for it again. The browser is free to listen to this line or ignore it, but most will listen to it.

```
print "\n" ;# end of headers
```

This tells Perl to tell the browser that we are done with the headers, and we will proceed with the JavaScript.

```
print "document.write(\"$m/$d/$y\");" ;# the javascript
```

This tells Perl to tell the browser that document.write(...) is the JavaScript that was requested. By the time it is sent to the browser, the `$m/$d/$y` part will have been replaced in Perl by the actual month, day, and year, as calculated in Perl.

The program ends at this point. This tells Perl to tell the browser that it has received everything that was being sent, and it can go on about its business.

The browser will then execute the JavaScript that was received, which will write the date into the document (the webpage) at the place the script was requested.

52.3 Writing Your Own CGI

We use the date program above mostly as a simple example. Once you have a CGI program of any kind, you can ask a programmer to modify and improve it to do whatever you want.

As mentioned above, teaching you to write CGI is really beyond the scope of this book.

`http://ipup.doncolton.com/` provides a free textbook (written by me) that teaches how to write CGI using Perl.

Here are a few ideas of things you could do:

Count the days until Christmas.

Count the times this page was requested. You could maintain a file that holds a count of how many times your page has been accessed. Or you could access a database that has that same information.

Provide an interesting quote or witty saying.

Show a recent news headline.

Advertise your latest blog post.

Chapter 53

Other JavaScript Ideas

There is a very large body of free JavaScript code available on the Internet. You do not need to be a programmer and write your own code. You do need to be a little bit brave to integrate code written by others. It is not rocket science, but it is not entirely trivial either.

Google search "free javascript code" for many examples.

JavaScript can be used to verify and/or reformat information being keyed in by a user.

Say you are looking for a phone number, and they type in `8005551212`, and you want it to appear as `(800) 555-1212`. You can do that with JavaScript.

Say you are looking for a credit card number, and they type in `1234-5678-9012-3456`, and you want it to appear without hyphens or spaces as `1234567890123456`. You can do that with JavaScript.

Say you want to make sure the check-digit on a credit card number is valid before sending it to the server. (This catches most credit card typing mistakes.) You can do that with JavaScript.

Say you want to make sure the new password, which was typed in twice, is the same both times. You can do that with JavaScript.

Say you want to check the strength of a new password to maybe encourage the user to pick something better. You can do that with JavaScript.

However, you cannot guarantee that the user is running JavaScript, so the webserver that receives the information may have to fix it again, just in case the browser did not.

Chapter 54

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.

onmouseover and **mouseover** mark code that is triggered when the user moves the mouse onto something.

onmouseout and **mouseout** mark code that is triggered when the user moves the mouse away from something.

We certainly cannot give a full treatment of events and event handlers. Because this is an introductory textbook, this topic 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.

Chapter 55

Untangling HTML and Scripting

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?

55.1 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 type='text/javascript'>
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.

55.2 ezCalc

Here is a complete example.

```
<!DOCTYPE html><head lang=en><meta charset=utf-8 />
<title>ezCalc</title>
<style type=text/css>
  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 />
</p>
<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 56

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 131 (p.310)**:**
What words does DOM stand for?

Required 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.

Unit VIII

Appendix

Appendix A

Elements of HTML

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>
<html>
<head> head things </head>
<body> body things </body>
</html>
```

A.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.

A.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.

A.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.

A.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.

Appendix B

CSS Attributes

Following is a mostly complete list of the CSS attributes mentioned on the W3 Consortium website. New attributes may become official. I intend to add them here. If you do not see it, please let me know so I can update my list.

`http://www.w3.org/TR/CSS2/` is the source of most of my information.

`http://www.w3.org/TR/CSS2/propidx.html` I especially used Appendix F, the Full Property Table.

azimuth: Aural (related to sound). From what direction should the sound seem to be coming?

background-attachment:

background-color: Specifies the color of the background. The attribute for foreground color (of text) is simply **color:**.

background-image: Specifies the image that is to be displayed in the background.

background-position: Where to position the background image.

background-repeat: How to repeat the background image. repeat. repeat-x. repeat-y. no-repeat.

background:

border-collapse: In a table, if two borders touch each other, should they merge or be kept separate?

border-color: What color should the border be?

border-style:

border-spacing: In a table.

border-style:

Appendix C

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.

C.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? Looks pretty real. It is 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://en.wikipedia.org/wiki/Lorem_ipsum` has more about lorem ipsum.

`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 132 (p.311)**:**
What is lorem ipsum?

Acceptable Answer:
filler text

Exam Question 133 (p.311)**:**
What fancy name do we use to describe fake text that is inserted into a webpage just to give it some body?

Required Answer:
lorem ipsum

C.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.

`http://www.gutenberg.org/` is the homepage for Project Gutenberg.

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 Misrables 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)

C.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 the public domain.

http://en.wikipedia.org/wiki/Creative_commons has more about the Creative Commons organization.

Appendix D

The Original Preface

This is the original preface. Sadly, it is boring, which is why it got sent to the back of the book.

Introduction to Website Design and Development (IWDD) is a college text book. It is intended to help students get started in the enormous and fascinating world of website design and development.

We assume you, our student, our reader, have no prior experience designing or creating websites. We just assume you have looked at many websites and you want to try your own hand at making one. And we assume you can type and use a web search engine.

There are probably thousands of books on website design and development. There are probably thousands of free websites and blogs that teach website design. It is a very popular subject.

We will not cover all of website design in this book. In fact, our goal is to cover just the very basics. Our goal is to give a basic understanding to the student, so they are ready to search out more detailed, specific knowledge they may need for the projects they have in mind.

The depth of coverage is suitable to an introductory one-semester course that meets for about forty hours.

Introduction to Website Design and Development (IWDD) really focuses on first things. What are the first things you need to know about designing and creating webpages? Answering that question and teaching those things to you is our goal.

We focus on target concepts, topics, and skills that are normally expected of students who have passed such a course. We also cover the basic skills and concepts that support these target objectives. And we do some preparation for more advanced courses in website design.

Organization

This book tries to be brief. It tries to go directly to the heart of each concept. As such, it may be more of a reference book than a story book. An extensive index is provided.

We assume, and expect, that you have used a web search engine before, and that you can do it again. Web search engines include (in 2012) Google (`http://google.com`), Yahoo (`http://yahoo.com`), and Microsoft's Bing (`http://bing.com`), among many others.

I will follow common practice and use "Google search (x)" or "Google (x)" to mean "use your favorite web search engine to search for (x)", as in "google this" or "google that". I apologize in advance to readers that prefer Yahoo or Bing or something else.

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).

Test Bank

At the back of the book, Appendix P (page 297) is a Test Bank.

As material is covered in the book, exam questions are inserted to show what the student should be learning and may be tested on. These exam questions appear throughout the book, together with acceptable answers. The test bank repeats these same questions that appeared throughout the book, but without their answers.

The Test Bank is a way for students to test themselves by reviewing the questions and making sure they know at least one acceptable answer.

(The acceptable answers provided are often just the most simple answer I have found that covers the question adequately. There are often much more complete and accurate answers that go far beyond the minimal answer shown as "acceptable." Please forgive me for that.)

The Test Bank is also a way for teachers to be reminded of specific things that I think students should be able to answer.

In many cases the questions and answers summarize material that is presented nearby in substantially greater detail.

In other cases the questions and answers are the actual presentation of that material. This is especially true when the specific material is something simple like vocabulary, and repetition would be tiresome and redundant.

Following is the format in which questions and answers are presented.

Exam Question 134 (p.311)**:**
What are target skills?

Acceptable Answer:
Target skills are the skills you intend to gain.

Exam Question 135 (p.311)**:**
What are basic skills?

Acceptable Answer:
Basic skills are the skills you must learn before you can be good at the target skills.

The questions are hot-linked to make it easy for the student to jump back and forth between the test bank and the content chapters.

Appendix E

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.

Appendix F

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.

F.1 Microsoft Windows

If you are using a Microsoft Windows operating system, we recommend the text editor named **Notepad++** which you can find free on the Internet.

`http://notepad-plus-plus.org/` is the homepage for Notepad Plus Plus. It is free. It runs on Microsoft Windows.

F.2 Apple Mac OS X

If you are using an Apple operating system, we recommend **TextWrangler**.

`http://www.barebones.com/products/textwrangler/` is the homepage for TextWrangler. It is free. It runs on Mac OS X.

F.3 Linux

If you are using a Linux operating system, maybe Ubuntu, we recommend Sublime Text.

SciTE: free for MS and Linux, paid for OS X.

gedit: for Linux.

geany: for MS and Linux.

kate:

textmate2: free

F.4 Spend Money

`http://www.sublimetext.com/` is the homepage for Sublime Text. Sublime Text is available for OS X, Windows and Linux. You can evaluate it for free, but it costs money ($59 the last I checked) for continued use.

F.5 Use A Word Processor Anyway

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.

Appendix G

Search Engine Optimization

Contents

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

These questions are the key elements of what is called "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.

G.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 your website is truly relevant, perhaps the most relevant of all websites? Do the users get to see your 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 yours.

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

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

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

(3) They look at your meta description. This is a longer phrase you put in your 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 your page or not. It looks like this:

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

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

G.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 you get get lots of people to view your page, you can make money from the advertising that is displayed on your page.

If you 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, you do not want to look like a fake website. You want to look (and be) sincere.

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

G.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.

G.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 you can be found, you will be indexed.

When you 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.

G.5 You Can Look Permanent

If you paid for your domain name, like it is a dot-com or a dot-org, and you bought several years worth of time, then you 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 your entry into the game. You can buy 99 years of domain name and still not achieve a high page rank.

G.6 You Can Pay Money

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

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

G.7 You Can Swap Links

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

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

Say you have 100 friends, and each of them adds a link to their homepage, or a nearby page, mentioning you. Your PageRank goes up. You are popular. Of course, you would be willing to reciprocate and put a link on your homepage to each of your friends, right?

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

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

G.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 that you think people will be looking for. Think like a fish to catch a fish.

G.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.

Appendix H

Special Glyphs

Many unusual glyphs can be inserted into your webpage by using character references.

`&` – Character references always start with an ampersand.

`;` – They frequently end with a semi-colon. The ending semi-colon is optional if things are not ambiguous.

Here are a few handy glyphs.

`&quot;`	"
`&apos;`	'
`&amp;`	&
`&copy;`	©
`&reg;`	®
`&trade;`	™

`&gt;`	$>$
`&lt;`	$<$
`&geq;`	$\geq$
`&nequiv;`	$\neq$
`&leq;`	$\leq$

`&male;`	♂
`&female;`	♀

`&spades;`	♠
`&diamonds;`	♢
`&hearts;`	♡
`&clubs;`	♣

There are hundreds (maybe thousands) more of these character references supported in HTML. Google "html special characters" for more.

Exam Question 136 (p.311)**:**
In HTML what is the character entity reference for a space without letting the line split?

Required Answer:
&nbsp;

Exam Question 137 (p.312)**:**
In HTML what is the character entity reference for the ampersand (and) symbol?

Required Answer:
&amp;

Exam Question 138 (p.312)**:**
In HTML what is the character entity reference for the less-than symbol?

Required Answer:
&lt;

Exam Question 139 (p.312)**:**
In HTML what is the character entity reference for the greater-than symbol?

Required Answer:
&gt;

Exam Question 140 (p.312)**:**
In HTML what is the character entity reference for a double-quote?

Required Answer:
&quot;

Exam Question 141 (p.312)**:**
In HTML what is the character entity reference for a single-quote?

Required Answer:
&apos;

Exam Question 142 (p.312)**:**
In HTML what is the character entity reference for the (C) copyright symbol?

Required Answer:
&copy;

Exam Question 143 (p.312)**:**
In HTML what is the character entity reference for the (R) registered-trademark symbol?

Required Answer:
&reg;

Exam Question 144 (p.312)**:**
In HTML what is the character entity reference for the TM nonregistered-trademark symbol?

Required Answer:
&trade;

Appendix I

The Apache Webserver

Contents

Apache is a very commonly used webserver. Apache has about 2/3 of the market (2011). Its closest competitor is Microsoft's IIS (Internet Information Services or Server).

Because Apache is the clear leader, and is so important, many people treat it as though it were **the** way to do things.

In this Appendix we provide details for several widely-used things that you should know about, and may be able to configure.

I.1 Understanding The URL

Webpages are retrieved by URL. URL stands for Uniform Resource Locator.

The URL for this book looks like this:

`http://iwdd.tk/iwdd.pdf`

Technically, the more correct term is URI. URI stands for Uniform Resource Identifier. Most people go with URL, and we will also.

A URL is composed of several distinct chunks. Here is a typical version.

`(scheme)://(host)/(path)`

Assuming the http scheme, and example.com as the domain name, we have the following:

`http://example.com/(path)`

There are other pieces that might be present in a URL. Here is a more complete version.

`(scheme)://(host:port)/(path)?(query)#(fragID)`

The **scheme** is almost always **http** or **https**. You may also see **file** or **ftp** or **mailto** or **javascript**. Sometimes scheme is called **protocol**.

The **host** is the network location, and is typically a domain name but may be an IP address. Occasionally it includes a port number, but usually not. The host is also called the **hostname** or the **hostport**.

The **path** tells which object we want from the website. It generally reflects the underlying file system at the server where the static content is stored. Static content generally includes web pages and media. The path is also called the **pathname**.

The **path** is the component in which we are most interested in this appendix. We look into it more deeply in the next section.

The **query** is used with dynamic content, such as that provided by a CGI program. It is composed of name=value pairs. They are usually separated by `&` but may be separated by `;` instead. The query is also called the **search**.

The **fragID**, or **fragment** ID, specifies a particular ID within the webpage. Within the webpage, the part that has that ID is supposed to receive the focus when the page is loaded. Generally that means that part of the page is positioned at the top of the browser's viewport (its visible area). The fragID is also called the **hash**.

I.2 The Path: Directories and Folders

The path component of the URL generally reflects the actual file structure present on the server. There is no requirement that this be the case, but it is most often true anyway.

The slash character, `/`, is typically used to divide the whole path into sections. The first section identifies a folder or directory within the document root. The second section identifies a folder or directory within the first section. The last section identifies a specific file within the directory specified by the preceeding sections.

In the following example, docs is a folder within the document root, 2013 is a folder within docs, 01 is a folder within 2013, and file.html is a file within the 01 folder.

```
docs/2013/01/file.html
```

I.3 The Document Root: public html

The user never knows exactly what is on the server, or where it is stored. All the user really knows is (a) I presented a particular URL, and (b) the server sent back this content.

In Apache, the content lives inside a tree-structured file system, with one part of it being designated as the **document root**.

In this example: `docs/2013/01/file.html`, the docs folder is located in the document root.

Almost always, the document root is `public_html` and it is located in the home directory of the website owner. Apache lets you change that, but nobody does.

For a Linux-based webserver (which is very common), the directory structure will be something like this:

```
/home/(username)/public_html
```

If your username is “fred” then your document root would probably be:

`/home/fred/public_html`

For a large hosting provider, home may be replaced by home2 or home3 or something else.

Sometimes the `/home/username` portion, which is called your home directory, is represented using a tilde, as follows:

`~/public_html`

Remember: none of this is seen by the end user. It is only seen by the developer.

I.4 index.html, index.htm, index.cgi

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 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.

There is a hugely important exception.

If one of the files is named `index`, as in **index.html** or **index.htm** or **index.cgi**, 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.

Note: `index` and `Index` are not the same. You must use lower-case letters.

I.5 Directory Browsing

Many users know about the automatic index generation. When they see a path, they may change the URL by deleting the last part of the path. Then they submit it and see what comes back.

This is called directory browsing.

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.

If you care about this for your own website, you should always provide an **index.html** file in every directory that users might discover.

You can put `Options -Indexes` in your .htaccess file to explicitly turn of index generation. (We talk more about .htaccess below in section I.10, page 271.)

I.6 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".

I.7 Icons

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**.

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 still works, and will probably always work.

Besides putting `favicon.ico` in the document root, it can 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.

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.)

Another method is available. 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.

I.8 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.

I.9 Public HTML

The file system for Apache normally puts all webpages in a directory (also known as a folder) named `public_html`. Sub-folders can be made and they can have content you provide.

Normally the path portion of the URL exactly matches the subdirectory structure in Apache.

Certain filenames are special. **index.html** is one of those. If the URL leads to a directory instead of a file, Apache will normally display a list of files in that directory. However, if one of the files is **index.html**, Apache will present that file instead.

Similarly, **index.cgi** is special. If the URL leads to a directory, and **index.html** is not found, but **index.cgi** is present, Apache will run it as a program, and will treat the output of that program as the webpage to be sent back to the browser.

I.10 .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.

I.10.1 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.

I.10.2 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.

I.10.3 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.

I.10.4 Leach Protection

Many websites block outside linking to media on their website, or sometimes even to webpages on their website. Such deep linking is called **leaching** (as in the blood-sucking leach). Using **.htaccess** rewrite rules, we can make it so the picture is only viewable from our own website.

One reason to prevent leaching is the desire to display advertising near their images. Deep linking retrieves just the image and not the advertising.

Another reason to prevent leaching 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 leaching.

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 noleach.jpg. Presumably noleach.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/noleach.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.

Appendix J

Publishing and Backing Up

Contents

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.

J.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 145 (p.312)**:**
What word describes the copying of files from our personal computer to the server that is hosting our website?

Acceptable Answer:
upload or uploading

When we say **download**, we mean to take files (content) and copy them down from the cloud (online) to the ground (offline).

Exam Question 146 (p.312)**:**
What word describes the copying of files from the server that is hosting our website to our personal computer?

Acceptable Answer:
download or downloading

J.2 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.

J.3 Mac OS X and Linux

The OS X operating system and the Linux operating system are both based on Unix. As such, they have a lot in common.

We can run rsync directly from the command line in either OS X or in Linux.

J.3.1 Downloading (Backing Up)

Normally when we download, we are making a back-up copy of something that has already been published on the web.

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/
```

J.3.2 Uploading (Publishing)

Normally when we upload, we are publishing something that was developed locally.

To upload everything in the `version1` directory on the local machine to the `public_html` directory at `abc.com`, you could use the following command.

```
rsync -auvz version1/* abc.com:public_html/
```

todo: add more here

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.

J.4 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.

Appendix K

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.**

K.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?

K.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 147 (p.313)**:**
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.

K.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.

K.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.

K.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.

K.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.

Appendix L

Patterns (Regular Expressions)

It is possible to specify a pattern for what the user is allowed to type into an input field.

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

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}" ... />
```

For a ten-digit telephone number, xxx-xxx-xxxx, 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 easy to learn, and are very powerful, but we will not cover them here. You can find more on the web. Just google "regular expression tutorial" or "html5 pattern tutorial".

Using a pattern does not guarantee that the browser will enforce it, but most modern browsers do.

Appendix M

WordPress: Advanced Concepts

In Chapter 17 (page 78) we introduced **WordPress** and gave 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.

M.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 glup in before diving.

M.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.

M.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.

Appendix N

Glossary

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 36.9 (page 170).

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 36.8 (page 170).

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).

Appendix O

Exam Questions

Contents

This chapter contains exam questions that have not yet been relocated to better locations throughout the book.

todo: Target practice: using css selectors.

O.1 URL

Exam Question 148 (p.313)**:**
What does %20 mean in a URL?

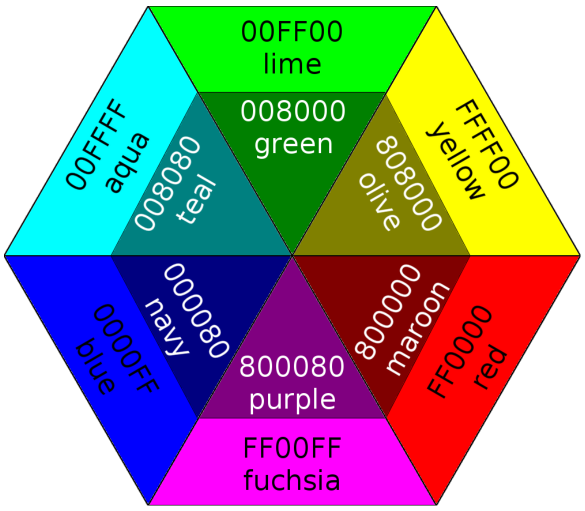
Required Answer:
space

Exam Question 149 (p.313)**:**
We identified three kinds of URLs. One is relative. List the other two.

Required Answer:
docroot, absolute

O.2 The Web Color Wheel

Web colors are based on an additive model, using Red, Green, and Blue as the nominal components; "nominal" because the green is really more of a lime color.



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.

Noticably 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 of white, silver, gray, and black.

Here is the complete list: **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.

Gray is also commonly spelled grey. Use gray.

Exam Question 150 (p.313)**:**
What is the official name of this color: `#ffffff`?

nswer:
white

white **Exam Question 151** (p.313)**:**
What is the official name of this color: `#c0c0c0`?

nswer:

silver

silver **Exam Question 152** (p.313)**:**
What is the official name of this color: `#808080`?

nswer:
gray

gray **Exam Question 153** (p.313)**:**
What is the official name of this color: `#000000`?

nswer:
black

black

Exam Question 154 (p.313)**:**
What is the official name of this color: `#ff0000`?

nswer:
red

red **Exam Question 155** (p.313)**:**
What is the official name of this color: `#800000`?

nswer:
maroon

maroon

Exam Question 156 (p.313)**:**
What is the official name of this color: `#00ff00`?

nswer:
lime

lime **Exam Question 157** (p.313)**:**
What is the official name of this color: `#008000`?

nswer:
green

green

Exam Question 158 (p.314)**:**
What is the official name of this color: `#0000ff`?

nswer:
blue

blue **Exam Question 159** (p.314)**:**
What is the official name of this color: `#000080`?

nswer:
navy

navy

Exam Question 160 (p.314)**:**
What is the official name of this color: `#ffff00`?

nswer:
yellow

yellow **Exam Question 161** (p.314)**:**
What is the official name of this color: `#808000`?

nswer:
olive

olive **Exam Question 162** (p.314)**:**
What is the official name of this color: `#ffA500`?

nswer:
orange

orange

Exam Question 163 (p.314)**:**
What is the official name of this color: `#ff00ff`?

nswer:
fuchsia

fuchsia **Exam Question 164** (p.314)**:**
What is the official name of this color: `#800080`?

nswer:
purple

purple

Exam Question 165 (p.314)**:**
What is the official name of this color: `#00ffff`?

nswer:
aqua

aqua **Exam Question 166** (p.314)**:**

What is the official name of this color: `#008080`?

nswer:
teal

teal

Exam Question 167 (p.314)**:**
What does alpha mean?

Acceptable Answer:
opacity or transparency

O.3 CSS

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

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

Targets are separated by commas. Attributes are separated by semi-colons.

Exam Question 168 (p.314)**:**
What CSS attribute makes an element partially transparent?

Required Answer:
opacity:

Exam Question 169 (p.314)**:**
What CSS attribute sets the width of a block item?

Required Answer:
width:

Exam Question 170 (p.314)**:**
What CSS attribute sets the largest allowed width of a block item?

Required Answer:
max-width:

Exam Question 171 (p.314)**:**
What CSS attribute sets the smallest allowed width of a block item?

Required Answer:
min-width:

Exam Question 172 (p.314)**:**
What CSS attribute sets the height of a block item?

Required Answer:
height:

Exam Question 173 (p.314)**:**
What CSS attribute sets the largest allowed height of a block item?

Required Answer:
max-height:

Exam Question 174 (p.314)**:**
What CSS attribute sets the smallest allowed height of a block item?

Required Answer:
min-height:

What pseudo-class targets an item when you tab to it? focus

What pseudo-class targets an item when you click inside it? focus

List some pseudo-classes. :hover, :link, :active, :visited, :first-child, :nth-child(), :first-letter, :first-line

O.4 JavaScript

What JavaScript command matches this CSS: body background-color: white document.body.style.backgroundColor="white"

Exam Question 175 (p.314)**:**
What HTML attribute runs a JavaScript action when the mouse enters the specified item?

Required Answer:
onmouseover=

Exam Question 176 (p.314)**:**
What HTML attribute runs a JavaScript action when the mouse leaves the specified item?

Required Answer:
onmouseout=

Exam Question 177 (p.314)**:**
What HTML attribute runs a JavaScript action when the user single-clicks on the specified item?

Required Answer:
onclick=

Exam Question 178 (p.315)**:**
What HTML attribute runs a JavaScript action when the user double-clicks on the specified item?

Required Answer:
ondblclick=

Appendix P

Test Bank

Test Bank

Unit I Questions

Chapter 1 Questions

1: (p.11) In HTML what tag is used to identify a level-1 header?

2: (p.11) In HTML what words does the h1 tag stand for?

3: (p.12) In HTML what tag is used to present bold content?

4: (p.12) In HTML what word does the b tag stand for?

5: (p.12) In HTML what tag is used to present oblique (italic) content?

6: (p.12) In HTML what word does the i tag stand for?

7: (p.12) In HTML what tag is used to insert a picture?

8: (p.13) In HTML what word does the img tag stand for?

9: (p.13) What words does URL stand for?

10: (p.13) What words does HTML stand for?

Chapter 2 Questions

11: (p.15) In HTML what tag is used to identify a paragraph?

12: (p.15) In HTML what word does the p tag stand for?

Chapter 3 Questions

13: (p.16) What words does CSS stand for?

14: (p.17) In CSS what is the one-character symbol for class?

15: (p.17) What HTML attribute specifies class?

16: (p.18) What CSS attribute sets the size of your text font?

17: (p.18) What CSS attribute sets the foreground color of your text font?

18: (p.18) What CSS attribute sets the color of the background behind your text?

19: (p.18) What CSS attribute places a picture in the background?

20: (p.18) If you want to draw a box around some content, what CSS attribute would you use?

21: (p.18) If you draw a box around some content, what CSS attribute puts space between your content and the box?

22: (p.18) If you draw a box around some content, what CSS attribute puts space between the box and neighboring content?

Unit II Questions

Chapter 4 Questions

23: (p.21) What word means right at your computer instead of somewhere out on the Internet?

24: (p.21) What word means somewhere out on the Internet instead of right at your computer?

25: (p.25) Where does your browser store files it might need later?

26: (p.25) If you change a webpage, but when you visit it, it has not changed, what is probably the cause?

27: (p.25) If you change a webpage, but when you visit it, it has not changed, what should you do?

Chapter 5 Questions

Chapter 6 Questions

28: (p.34) Is the domain name within a URL case-sensitive?

29: (p.35) Is the path within a URL case-sensitive?

Chapter 7 Questions

30: (p.38) Can you really own a domain name?

31: (p.38) What is the typical cost for a .com domain name, in dollars per year?

32: (p.40) What words does TLD stand for?

33: (p.41) Who owns byuh.doncolton.com?

Chapter 8 Questions

Chapter 9 Questions

Chapter 10 Questions

34: (p.52) Can ideas be copyrighted?

35: (p.53) What is Fair Use?

36: (p.53) What is a Cover Version?

37: (p.53) What is a Public Domain?

38: (p.54) What is a Derivative Work?

39: (p.55) What is a Take-Down notice?

40: (p.55) What is a Safe Harbor?

41: (p.55) What words does DMCA stand for?

Chapter 11 Questions

Chapter 12 Questions

Chapter 13 Questions

Chapter 14 Questions

Unit III Questions

Chapter 15 Questions

42: (p.75) What words does WYSIWYG stand for?

Chapter 16 Questions

43: (p.76) What words does CMS stand for?

Chapter 17 Questions

44: (p.79) What words does blog stand for?

Unit IV Questions

Chapter 18 Questions

Chapter 19 Questions

Chapter 20 Questions

Chapter 21 Questions

Chapter 22 Questions

Chapter 23 Questions

Chapter 24 Questions

Unit V Questions

Chapter 25 Questions

Chapter 26 Questions

45: (p.124) What does deprecated mean?

46: (p.125) In HTML what tag is used to present emphasized content?

47: (p.125) In HTML what tag is used to present strike-through content?

48: (p.125) In HTML what tag is used to present underlined content?

49: (p.125) In HTML what tag is used to present lowered (subscripted) content?

50: (p.126) In HTML what tag is used to present raised (superscripted) content?

51: (p.126) What marks the start of a comment in HTML?

52: (p.126) What marks the end of a comment in HTML?

53: (p.127) In HTML, can comments be nested?

Chapter 27 Questions

54: (p.129) Give the HTML Prototype.

55: (p.130) In HTML the strong tag does the same as what other tag?

56: (p.131) In HTML what word does the em tag stand for?

57: (p.131) What character(s) mark the start of HTML markup?

58: (p.131) What character(s) mark the end of HTML markup, assuming that it is NOT a void tag?

59: (p.132) What character(s) mark the end of HTML markup, assuming that it IS a void tag?

60: (p.132) When a tag is void, what does that mean?

61: (p.132) In HTML does the order of attributes matter?

62: (p.134) If an HTML attribute's value includes a space does that force it to be quote marked?

63: (p.134) If an HTML attribute's value includes a double-quote (`"`) does that force it to be quote marked?

64: (p.134) If an HTML attribute's value includes a less-than (`<`) does that force it to be quote marked?

65: (p.134) If an HTML attribute's value includes a greater-than (`>`) does that force it to be quote marked?

66: (p.135) If an HTML attribute's value includes a single-quote (`'`) does that force it to be quote marked?

67: (p.135) If an HTML attribute's value includes a back-quote (`` ` ``) does that force it to be quote marked?

68: (p.135) If an HTML attribute's value includes an equals (`=`) does that force it to be quote marked?

69: (p.135) If an HTML attribute's value includes a letter (A a B b ...) does that force it to be quote marked?

70: (p.136) If an HTML attribute's value includes a digit (1 2 3 ...) does that force it to be quote marked?

71: (p.136) If an HTML attribute's value includes a colon (`:`) does that force it to be quote marked?

72: (p.136) If an HTML attribute's value includes a dot (`.`) does that force it to be quote marked?

73: (p.136) If an HTML attribute's value includes a dash (`-`) does that force it to be quote marked?

74: (p.137) If an HTML attribute's value includes a slash (`/`) does that force it to be quote marked?

75: (p.137) If an HTML attribute's value includes a percent (`%`) does that force it to be quote marked?

76: (p.137) If an HTML attribute's value includes an ampersand (`&`) does that force it to be quote marked?

77: (p.139) When one set of tags begins and ends totally inside another set of tags, what do we call that?

78: (p.140) What is wrong with this HTML tag order: a b /a /b?

79: (p.140) What is wrong with this HTML tag order: a b /b /a?

Chapter 28 Questions

80: (p.141) In HTML what tag is used to identify a second-level heading?

81: (p.142) In HTML what tag is used to insert a line-break?

Chapter 29 Questions

82: (p.143) List the five commonly-used global attributes for HTML tags.

Chapter 30 Questions

83: (p.144) List the four commonly-used attributes of the image tag.

Chapter 31 Questions

Chapter 32 Questions

84: (p.153) If our base URL is http://a.com/b/c/ and our stated URL is ../d/e/ what is our final URL?

85: (p.153) If our base URL is http://a.com/b/c/ and our stated URL is ./d/e/ what is our final URL?

86: (p.153) If our base URL is http://a.com/b/c/ and our stated URL is /d/e/ what is our final URL?

87: (p.153) If our base URL is http://a.com/b/c/ and our stated URL is d/e/ what is our final URL?

Chapter 33 Questions

88: (p.154) In HTML what words does the ol tag stand for?

89: (p.154) In HTML what tag is used to introduce a numbered list?

90: (p.155) In HTML what words does the ul tag stand for?

91: (p.155) In HTML what tag is used to introduce an un-numbered list?

92: (p.155) In HTML what words does the li tag stand for?

93: (p.155) In HTML what tag is used to introduce an entry in a list?

Chapter 34 Questions

94: (p.157) In HTML what tag is used to introduce a table?

95: (p.157) In HTML what tag is used to introduce a row within a table?

96: (p.157) In HTML what tag is used to introduce a cell within a table row?

97: (p.157) In HTML what table attribute is used to merge cells horizontally?

98: (p.157) In HTML what table attribute is used to merge cells vertically?

Chapter 35 Questions

99: (p.159) Can a div legally appear within another div?

100: (p.159) Can a p legally appear within a div?

101: (p.160) Can a div legally appear within a p?

102: (p.160) Can a p legally appear within another p?

103: (p.160) Can a span legally appear within another span?

104: (p.160) Can a span legally appear within a p?

105: (p.160) Can a p legally appear within a span?

Chapter 36 Questions

106: (p.163) List the three form input tags.

107: (p.165) List the six non-button type= attributes for the input tag.

108: (p.165) List the four button type= attributes for the input tag.

109: (p.168) What HTML attribute causes the cursor to begin in a certain input field?

110: (p.168) What HTML attribute specifies the order in which fields are reached by tabbing?

Unit VI Questions

Chapter 37 Questions

111: (p.175) What marks the start of a comment in CSS?

112: (p.175) What marks the end of a comment in CSS?

113: (p.175) List the four text-align: options.

Chapter 38 Questions

Chapter 39 Questions

Chapter 40 Questions

Chapter 41 Questions

114: (p.189) List the four major attributes of the box model.

115: (p.189) List the ten border-style: options.

116: (p.191) When we say margin: 9px 8px 7px 6px; which margin is 9px?

117: (p.191) When we say margin: 9px 8px 7px 6px; which margin is 8px?

118: (p.191) When we say margin: 9px 8px 7px 6px; which margin is 7px?

119: (p.191) When we say margin: 9px 8px 7px 6px; which margin is 6px?

120: (p.191) When we say margin: 9px 8px 7px; what is the implied fourth value?

121: (p.191) When we say margin: 9px 8px; what is the implied third value?

122: (p.192) When we say margin: 9px 8px; what is the implied fourth value?

Chapter 42 Questions

123: (p.197) List the five generic font families.

124: (p.201) What CSS attribute does shadowing of your text font?

Chapter 43 Questions

Chapter 44 Questions

125: (p.209) What is the one-character symbol for ID?

Chapter 45 Questions

Chapter 46 Questions

126: (p.212) What is the one-character symbol for pseudo-class?

127: (p.212) What does pseudo mean?

Chapter 47 Questions

Chapter 48 Questions

Unit VII Questions

128: (p.218) In DHTML, what does the D stand for?

129: (p.218) Are Java and JavaScript the same thing?

130: (p.219) Are Java and JavaScript related?

Chapter 49 Questions

Chapter 50 Questions

Chapter 51 Questions

Chapter 52 Questions

Chapter 53 Questions

Chapter 54 Questions

Chapter 55 Questions

Chapter 56 Questions

131: (p.238) What words does DOM stand for?

Unit VIII Questions

Chapter A Questions

Chapter B Questions

Chapter C Questions

132: (p.247) What is lorem ipsum?

133: (p.247) What fancy name do we use to describe fake text that is inserted into a webpage just to give it some body?

Chapter D Questions

134: (p.252) What are target skills?

135: (p.252) What are basic skills?

Chapter E Questions

Chapter F Questions

Chapter G Questions

Chapter H Questions

136: (p.263) In HTML what is the character entity reference for a space without letting the line split?

137: (p.263) In HTML what is the character entity reference for the ampersand (and) symbol?

138: (p.264) In HTML what is the character entity reference for the less-than symbol?

139: (p.264) In HTML what is the character entity reference for the greater-than symbol?

140: (p.264) In HTML what is the character entity reference for a double-quote?

141: (p.264) In HTML what is the character entity reference for a single-quote?

142: (p.264) In HTML what is the character entity reference for the (C) copyright symbol?

143: (p.264) In HTML what is the character entity reference for the (R) registered-trademark symbol?

144: (p.264) In HTML what is the character entity reference for the TM nonregistered-trademark symbol?

Chapter I Questions

Chapter J Questions

145: (p.276) What word describes the copying of files from our personal computer to the server that is hosting our website?

146: (p.276) What word describes the copying of files from the server that is hosting our website to our personal computer?

Chapter K Questions

147: (p.280) What is Moore's Law?

Chapter L Questions

Chapter M Questions

Chapter N Questions

Chapter O Questions

148: (p.290) What does %20 mean in a URL?

149: (p.290) We identified three kinds of URLs. One is relative. List the other two.

150: (p.291) What is the official name of this color: `#ffffff`?

151: (p.291) What is the official name of this color: `#c0c0c0`?

152: (p.292) What is the official name of this color: `#808080`?

153: (p.292) What is the official name of this color: `#000000`?

154: (p.292) What is the official name of this color: `#ff0000`?

155: (p.292) What is the official name of this color: `#800000`?

156: (p.292) What is the official name of this color: `#00ff00`?

157: (p.292) What is the official name of this color: `#008000`?

158: (p.292) What is the official name of this color: `#0000ff`?

159: (p.293) What is the official name of this color: `#000080`?

160: (p.293) What is the official name of this color: `#ffff00`?

161: (p.293) What is the official name of this color: `#808000`?

162: (p.293) What is the official name of this color: `#ffA500`?

163: (p.293) What is the official name of this color: `#ff00ff`?

164: (p.293) What is the official name of this color: `#800080`?

165: (p.293) What is the official name of this color: `#00ffff`?

166: (p.293) What is the official name of this color: `#008080`?

167: (p.294) What does alpha mean?

168: (p.294) What CSS attribute makes an element partially transparent?

169: (p.294) What CSS attribute sets the width of a block item?

170: (p.294) What CSS attribute sets the largest allowed width of a block item?

171: (p.294) What CSS attribute sets the smallest allowed width of a block item?

172: (p.294) What CSS attribute sets the height of a block item?

173: (p.295) What CSS attribute sets the largest allowed height of a block item?

174: (p.295) What CSS attribute sets the smallest allowed height of a block item?

175: (p.295) What HTML attribute runs a JavaScript action when the mouse enters the specified item?

176: (p.295) What HTML attribute runs a JavaScript action when the mouse leaves the specified item?

177: (p.295) What HTML attribute runs a JavaScript action when the user single-clicks on the specified item?

178: (p.296) What HTML attribute runs a JavaScript action when the user double-clicks on the specified item?

Index