Introduction to HTML & CSS

Learn To Code Websites Like A Pro

By Danny Ajini
of Climb New Heights LLC.

CLIMB NEW HEIGHTS .com

Introduction To HTML & CSS:

Learn To Code Websites Like A Pro

By Danny Ajini

Of Climb New Heights LLC.

ClimbNewHeights.com

Table Of Contents

Chapter 1: Start Here
Chapter 2: Understanding HTML
Chapter 3: Understanding CSS
Chapter 4: Where To Write Your Code
Chapter 5: Browsers
Chapter 6: HTML Structure
Chapter 7: CSS Structure
Chapter 8: Common HTML Elements & Their Rules
Chapter 9: Common CSS Styles And Their Rules
Chapter 10: Getting Started
Chapter 11: Preparing Images For The Internet
Chapter 12: Manipulating Placement Of HTML Objects
Chapter 13: Margins And Padding
Chapter 14: Other Types Of Positioning Techniques
Chapter 15: Fonts, Fonts, Fonts!
Chapter 16: Semantic Code
Chapter 17: Using Color
Chapter 18: CSS Sprites
Chapter 19: Element States
Chapter 20: Handy Things To Consider
Chapter 21: Flash, Javascript & CSS Animations
Chapter 22: Validation & Troubleshooting
Chapter 23: Minified HTML/CSS
Chapter 24: Grid Systems
Chapter 25: Responsive Web Design
Chapter 26: The Favicon
Conclusion & Next Steps

Chapter 1: Start Here!

I know what you're thinking, "Oh, an introduction…SKIP!" However, I'd advise you not to skip this introduction because I'm not going to babble on about my qualifications or some heartfelt story about how my HTML/CSS book cured a dying man of his ailments.

Rather, I just wish to give this book a little bit of context.

As with most things in life, coding a website may seem extremely difficult from the outside looking in. However, once you know the tricks of the trade it'll be a piece of cake for you.

If you've taken a good look around on the internet then you may have noticed that there are many different styles of website. Everything from centered 1-page sites to websites with a sidebar navigation all the way to websites built entirely from Flash animations.

I mention this because this brings up the point that there is more than one way to code a website. Which we will get to later in the book.

Brief Intro To Website Design History

Disclaimer: This brief history is only based on my own observations. Since this isn't essential to the book I won't waste any time looking up dates, gathering references or fact checking. You'll simply have to take this at face value.

The advent of the internet brought about very ugly websites. Originally website designers would rely on a very outdated technique of laying out their websites using what are known as "tables". You may be familiar with tables because they are used in Microsoft Word.

Essentially the tables would serve to segment content on the page and allow the user to have 2 & 3 column layouts which are still very much used today.

Once this obsession with tables died down a bit, website designers then switched to laying out their websites using what are known as "divs". A div is another HTML "tag" which allows users to segment content as they see fit.

I should also mention that for a while there, Flash became extremely popular. Flash was loved by many because it allowed the designer to place objects wherever they wanted without code, to create amazing animations, and do things that simply weren't possible using HTML, CSS or Javascript.

This brings us to today. With the advent of HTML5 and CSS3 (the latest versions of HTML & CSS) website designers are able to create awesome animations and do things that were never possible in the past without either Flash animations or the usage of Javascript.

Today, with the rising popularity of mobile and tablet devices, mobile website design as well as adaptive web design are becoming the new standard which will take us into the websites of the future.

What I Hope To Accomplish With This E-book

It is my hope that someone who is unfamiliar with HTML/CSS can pick up this e-book and use it to get a solid understanding of coding a website with HTML & CSS.

I want to cover all the bases of HTML/CSS without going overboard. Please note that this subject is very vast and I will be covering a few of the things that are necessary to get started.

Please note however, that I am extremely long winded. That being said, I know that this is supposed to be an introductory book but along the way I will be giving you some of the seeds of knowledge necessary to help you towards your second phase of website building.

That's when you graduate from coding a static page to full fledged responsive website design.

There are many resources on the internet which can help you learn how to build a website. However, none of these resources will give you the edge that can only be gained by years of experience, hours of troubleshooting as well as yelling at your computer screen which I hope to impart on you.

That being said let's begin!

Chapter 2: Understanding HTML

So they say that the difference between a master and a novice is that the master knows the tricks of the trade. It's the tiny, subtle things that the master knows that gives him the edge over the novice.

You may not know this, but anyone who has ever used a computer (even one without internet) has probably used and manipulated HTML.

HTML means "HyperText Markup Language" FYI.

If you have ever used Microsoft Word, then you have most likely come in contact with a multitude of HTML elements and you didn't even realize it.

These include:

- Headings (Heading 1 through Heading 6)
- Tables
- Ordered Lists (A list with numbered bullet points)
- Unordered Lists (This list is an example)
- Links (Surely you've encountered a Word document with a link to a website)

I'm sure there are a few more examples but are the stand out elements which I knew most people would be familiar with.

And, just like a Word document, a website can be extremely bland with no styling…or you can use your imagination and jazz it up a bit using fancy fonts, colored text, images, backgrounds, etc!

Figure 2.1 – On the left is a website and on the right is the same website minus the CSS.

CSS stands for "Cascading Style Sheet" and when people refer to CSS they will often talk about the "stylesheet" which is the document containing the CSS code. As you can see above in Figure 2.1, on the left you have a website which consists of HTML & CSS. And on the right you have the EXACT SAME WEBSITE, however I deleted the stylesheet.

The HTML includes all of the elements that are on the page IE: the pictures, the words, the links, etc. and the CSS is the code that the website designer uses to denote the color, orientation, font, etc.

As you can see the website on the right (minus the CSS) looks very much like something

you could create in Microsoft Word.

And much like Microsoft word elements, each HTML element comes preset with styles.

For example if you were to take a word and link it to a website, it becomes blue and underlined. If you were to begin typing, the words would begin in the upper left portion of the document using a Times New Roman font and utilizing a 12 point font.

The reason why I mention the link and paragraph style attributes in Microsoft Word is because they are almost identical to the default appearances of “unstyled” HTML elements.

Chapter 3: Understanding CSS

So as I mentioned above CSS stands for Cascading Style Sheets. It's what you will be using as a beginning website designer/ developer to give your websites some flavor.

It's the tool that you will use to transform a page which resembles the ugliest of Word documents into a beautiful and interactive work of art!

As I mentioned before, HTML elements all have their default styles.

Links look like this.

Paragraphs and other unstyled text default to Times New Roman (or Macintosh equivalent) 16 point font.

- Unordered lists
- end up looking
- like this.

It will be your responsibility to take these elements and style them to your desire utilizing CSS.

Things You Can Do With CSS

- Change the background of the page (Many options including tiled image, gradient, image/ background color combo, solid color, etc)
- Change the orientation/ position of an element
- Change fonts, font size, font color
- Give elements background images

In all honesty, there is almost no limit to what you can do visually to a website using CSS. And with the advent of CSS3 there is all types of cool things that you can do such as creating animations to happen when the user hovers over an element, rounded corners on elements, text shadows, etc!

Chapter 4: Where to Write Your Code

Before we go any further on our journey we have to discuss what you will be using to code the websites that you design.

I personally use Dreamweaver to do all of my coding. It's not what I personally recommend. I mostly use it because it came with Adobe Creative Suite, something that I personally use because it's required for my profession.

However, you can edit almost any type of code using the standard text editor that comes with your computer.

With a Windows based computer you can use Notepad to write, view or edit almost any type of computer code you could ever wish to manipulate. On a Mac the equivalent is TextEdit.

It's crazy that the simplest program on your computer gives you the ability to code the most beautiful websites.

Essentially, the code editor that you use is irrelevant. If you use a simple code editor then you have the potential to become much more proficient in writing code because you won't have any crutches to rely on.

If you spend your time writing code with a simple text editor, you will most likely end up having to reference almost every bit of code that you write or commit the code to memory.

Free Code Editors

Although memorizing code will make you an exceptional website developer the purpose of this book isn't to deter you from getting your feet wet with HTML/CSS.

Therefore I'll give you a few options that you can use if you don't have the option to use a paid program.

Of course by the time you read this e-book there may be more options so don't be afraid to Google!

- Notepad (TextEdit on Mac) - No frills text editor. Be prepared to Google every step of the way.
- Notepad++ - This is a free, open source download that is essentially a simple text editor with a few features which will make your life easier.
- TextWrangler – Similiar to Notepad++ but is Mac only and not open source.

What Is The Deal With Code Editors???

If you're completely new to code. Then you may be asking yourself, "how is the code editor that I use important in anyway?"

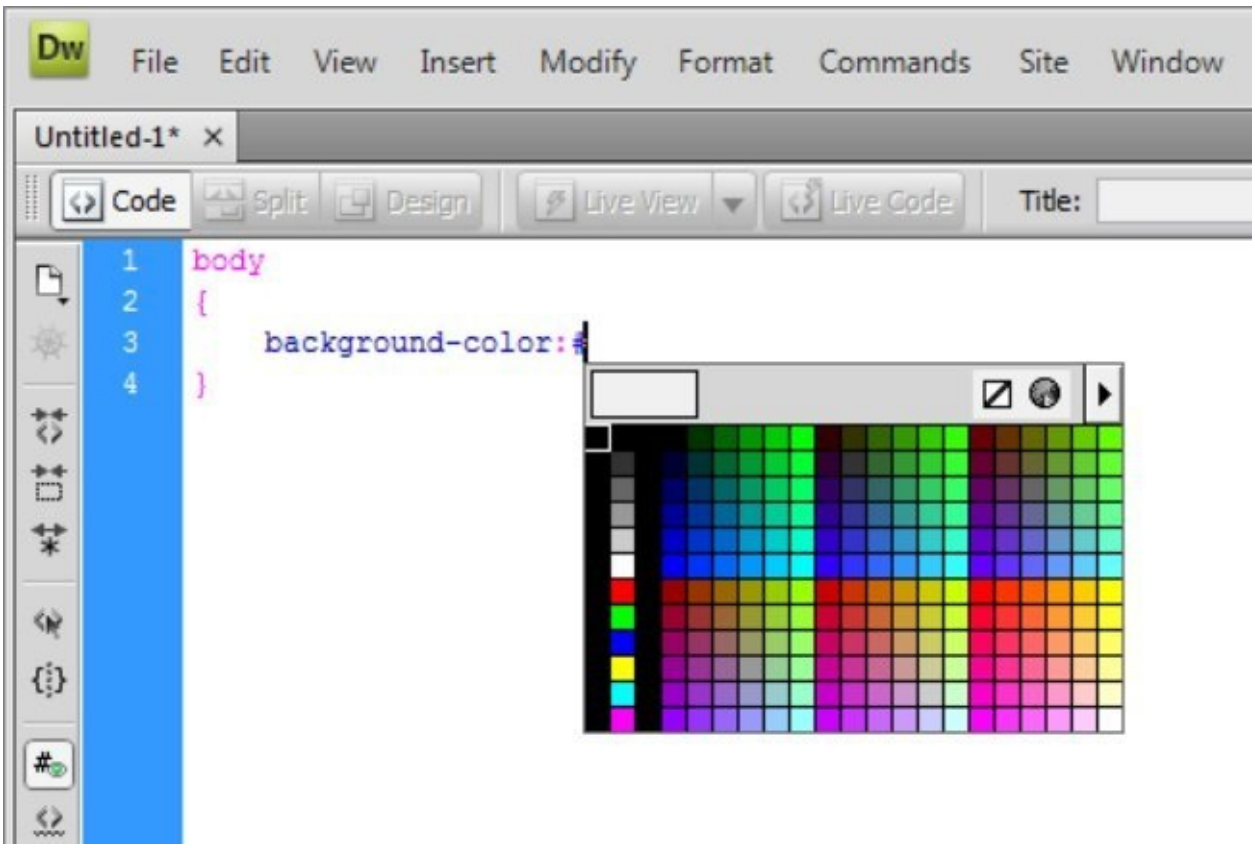
Basically, the code editor that you use can make your life easier. And you yourself can make your own life easier by paying attention to the tips that I'll give you in the chapter about "semantic" code.

Back to the topic at hand, the editor that you use can help you in a number of ways.

For instance, check out Figure 4-1. The language that you're looking at is CSS. You may not be able to tell, but I've decided to target the "body" of the HTML document. Then I decided to target the "background-color" of said body.

Colors on the internet are normally based on a hexidecimal system and each color designation begins with a hash marking. For instance "#ffffff" designates white. (We'll get into that more later.)

Figure 4.1 – I begin to specify a background color and a helper window pops up to assist me in choosing a color.



But as you can see, as soon as I began to type the hash mark, a helping hand popped up to help me choose a color.

This is just one example of how the text editor that you choose can help you get the job done.

Essentially, the best text editors can help you to remember tidbits of code, help you to keep your code organized and assist you when you forget essential pieces of code.

Chapter 5: Browsers

I know you're eager to start learning about website building however, the topic of browsers is one of the most important pieces to the puzzle.

An internet browser is simply the portal that you choose to help you view the abomination which has come to be known as the internet.

And just as the program you choose to write your code can help you get an edge, so can your browser.

As of today, the most popular internet browsers are Internet Explorer, Firefox, Opera, Chrome & Safari.

Why Do Browsers Matter?

If you are going to be coding websites it's important to have multiple browsers to test your websites in.

Not every browser displays the same font in the same way and some browsers have slightly different protocols for displaying certain objects.

Just as I mentioned before about how Microsoft Word has certain styles that they apply to different objects. Each browser also abides by certain styles for different HTML elements.

And while each browser has almost the same relative styles for each element, there are some subtle differences between browsers which can throw off your layout.

Often times the problem browser will be Internet Explorer. This is because Microsoft abides by a different set of rules pertaining to the way it's browser renders certain HTML elements.

You may think that this isn't an issue because you personally use Chrome, Firefox or Safari.

In reality, what you don't know is that a majority of people NEVER upgrade their browser from the default one their computer comes with. And since the majority of people use Windows, this means that a majority of people not only use Internet Explorer but a good chunk of those people have an OUTDATED version of Internet Explorer.

Only in recent years has Internet Explorer caught up with the modern browser world and begun to acknowledge modern website design techniques and render the new website design features in their own browsers.

What this means is that a good chunk of internet users world wide are viewing websites on a completely outdated and (for lack of a better word) non-compliant browser.

This makes coding HTML/CSS super fun!!! (sarcasm)

What Browsers To Download

The browsers that I personally recommend to download are:

- Internet Explorer
- Firefox
- Chrome

If you are a Windows user and are concerned about not having Safari, don't be. Google Chrome is using an open source web browsing system called "WebKit" which is what Safari uses. Keep that word WebKit in the back of your head because it will come up later when we start coding with CSS.

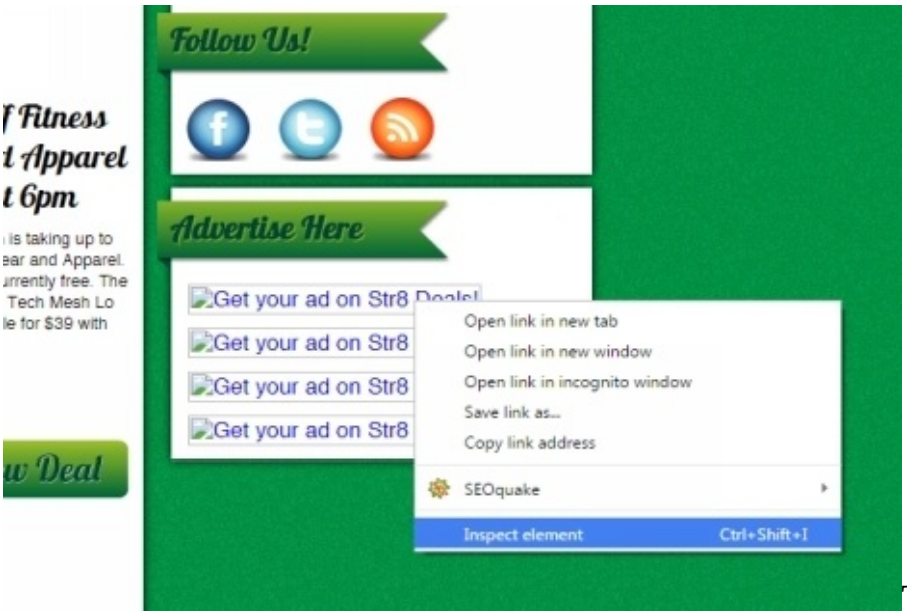
If you are a Mac user you should definitely download Internet Explorer so that you can make sure your websites perform as intended on Windows.

Between all of the browsers mentioned above you will have a pretty good idea of how your websites will perform for the myriad of internet users out there.

Advantages To Certain Internet Browsers

I personally crack open Google Chrome whenever I am going to begin a heavy session of coding.

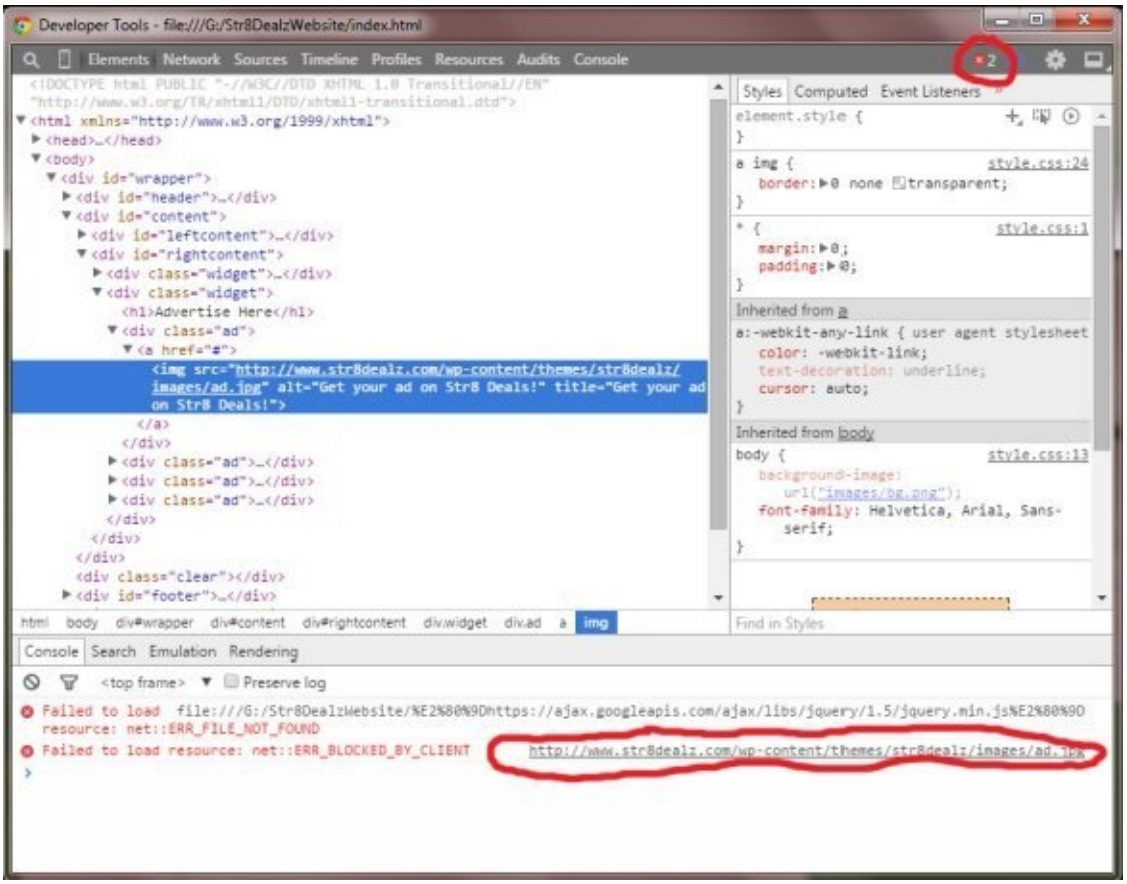
Figure 5.1 – I right clicked on the broken image and go to inspect element to see what's wrong.



This is because Google Chrome has more intuitive and user friendly code diagnostic capability.

In the image above you'll see that there are 4 broken images in the box labeled "Advertise Here". In Google Chrome, if I see an element that is behaving strange all I need to do is right click, followed by Inspect Element.

Figure 5.2 – Once I click inspect element this window pops up and gives me some helpful information.



Once I click “Inspect Element” then this extremely helpful “Developer Tools” window pops up.

The red “x” with the number in the corner, lets me know when files which the document says are supposed to accompany this file are missing.

I can see that the images which aren’t showing up are a result of a broken path.

A few things that I want to quickly mention:

1. Almost every modern browser has some sort of “Developer Tools” option however, I happen to think Google Chrome’s version is the most helpful and easiest to use. Google Chrome’s Developer Tools window is also very cool because it allows you to edit HTML & CSS directly in the Developer Tools window so you can experiment with your code before you actually put it to use on your website.
2. As you can see, when I “inspected the element” the panel on the right tells me every CSS style associated with the element in question. If there was a style which I happened to write incorrectly the Developer Tools window would let me know by putting a slash through the CSS rule. This lets me know that the CSS statement is not currently being used because it is written incorrectly.
3. These developer tools options are extremely helpful and have many more capabilities than I even know about or have the time to mention. If you end up becoming serious about coding, than you should make a conscious effort to learn more about what functionality the developer tools can offer you.

One thing that is pretty cool about Firefox is this 3D view option. I’m not exactly sure how useful it is. However, it can help you to understand the hierarchy of the page as well as show you where there may be errors in terms of elements placed incorrectly inside other elements.

In all honesty, when I tried to initialize the 3D view on my own computer a window popped up letting me know that 3D view has failed and to check the troubleshooting page on Firefox's website. I suspect it's because of my graphic card. However, here is a picture of what the Firefox 3D view looks like.

Figure 5.3 – The 3D view option in Firefox is super cool!

Nightly
Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Main_Page
Google
45pts
Main Page
Discussion
WIKIPEDIA
The Free Encyclopedia
Welcome to Wikipedia
Today's featured article
Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia
Inspect
tbody
tr
tbody
tr
3D (M)
HTML
Style

Chapter 6: HTML Structure

Before you begin to get acquainted with HTML you need to know a bit about the structure of HTML elements.

Beginning And Ending HTML Tags

Any element that can be wrapped around something has a beginning tag and an ending tag. The beginning tag will be encapsulated with using “less than” and “greater than” symbols (< >) the ending tag will be encapsulated with the same symbols with the addition of a forward slash (</ >).

Examples:

<body>This is the body tag, it is the body of the HTML document and all HTML elements which are to be featured on the page are between the beginning and ending of this element.**</body>**

** **This is a paragraph tag. All text that you wish to be included in this paragraph go here.** **

Self Contained HTML Tags

Elements that are self contained end themselves using a forward slash located right before the ending “>” symbol.

Below is a “break” tag. The break tag is used to cause a sentence to go to the next line, or free up some space in your document. The break tag can be used just like the enter key does when typing in a Microsoft Word document.

In the past using <br> used to be acceptable, valid HTML however in recent years you need to end the tag within itself (as seen below with a space and a forward slash).

Example:

**
**

Below is a “horizontal rule” tag. This HTML element creates a horizontal line on the page. Since this line cannot be wrapped around any other HTML elements, it is self contained. Below is an example of a horizontal rule.

Example:

<hr />

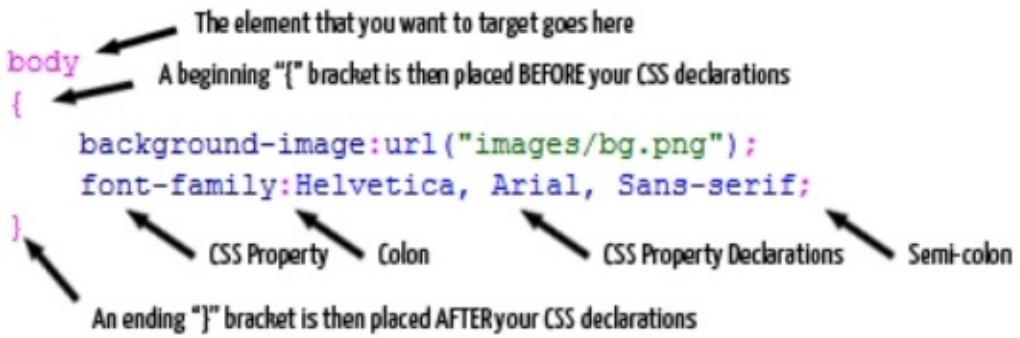
Chapter 7: CSS Structure

CSS can be a little bit more complicated because there are many CSS declarations that you can make and each declaration has it's own set of rules.

However, there is a few things that never change when it comes to CSS so I'll share them with you now so that you can have a better understanding before we start jumping head first into the subject.

CSS follows the structure that you see below. CSS is written like this when you declare your CSS in the head of the document or in a separate document.

Figure 7.1 – Below you can see the structure of CSS stylings.



Some people like to write their CSS directly in the HTML element itself it is known as "inline CSS". Below is an example.

```
<span style="color:#ff0000;">*</span>
```

With inline CSS styles there is no question as to the element being targeted so the targeting as well as the brackets get omitted. However the CSS property, the colon, the CSS property declarations and the ending semi-colon are all there.

Best practice with CSS is to keep all your CSS in a separate document and link the document to your HTML. However, as you can see there are a few different ways to write your CSS. To save time and to avoid confusion we will be just covering how to write your CSS in a separate document.

Inline styles are kind of frowned upon because they can increase load times in websites and are seen as being amateur-ish.

Targeting An HTML Element

In order to begin to write a CSS declaration about any element, you must first tell the CSS document which element to target.

Broad Example

If you wanted to target every instance of a particular HTML element you would begin your CSS declaration with the name of that element.

Examples:

- body (Targets the body of the HTML document and all elements contained within)
- h1 (Targets all h1 header elements in the document)
- a (Targets all links)

Targeting An Element by "Class"

Another way to target elements is to give them classes. Classes allow you to target elements based not on the type of HTML element but by the classes that you wish to give those elements.

Classes are great because they allow you to style objects with more precision.

Below is an example of giving an HTML element a class.

```
<span class="required">*</span>
```

As you can see, in order to give an element a class you must add "class=", followed by the class name in quotation marks.

In order to target a class, it's very similar to targeting an HTML element. Except you use the name of the class with a period in front of it.

Below is an example of targeting an HTML element by class.

```
.required
{
    color:#ff0000;
}
```

Targeting An Element By ID

Classes are cool because you can have as many elements on a page as you want with the same class.

However, there comes a time when maybe there is only one element on the page which you want to give a bit of significance to.

An ID is a way to target an HTML element. However, there can only be 1 element with the same ID per page.

Sure you can have multiple elements with the same ID on the same HTML page. However, when you go to validate the document it will show as invalid in an HTML validation program, as this is not a valid HTML technique.

To give an HTML element an ID it is very similar to the class declaration.

```
<span id="required">*</span>
```
Giving an HTML element an ID is almost identical to giving an HTML element a class. Except, instead of typing “class =‘class name’” you simply type “id=‘id name’”.

Targeting the element with CSS is almost identical as well. Except, instead of putting a period before the id name you use a hash mark (#).

```
#required
{
    color:#ff0000;
}
```

Nested CSS Targeting

To target elements inside other elements simply declare the elements from biggest to smallest separated by a space. For example to target an image wrapped in a link you would target those elements by using “a img”. Keep in mind you can get as deep as you want with nested CSS targeting.

As long as the element you are targeting is wrapped in the elements you specified your styles will effect the element in question.

Advanced CSS Targeting

With the advent of CSS3 there has become a few more ways to target HTML elements in order to style them.

For example, in HTML there are a few different elements that all use the same.

If you’re reading this book than chances are you’ve been around the block a few times and seen various things on the internet. One of the most common elements in websites is the web form.

You’ve most likely seen a few web forms in your day because there is always some advertiser or business that wants you to fill out and submit a form.

The “input” element is an extremely versatile element. This one element can be used as a text field in a form, as well as a date field, a phone number field AND a submit button!

This brings us to our next question, but how do you style a certain type of “input” element whithout disrupting other elements which you don’t want to tamper with?

```
input[type="text"]:not(.s), textarea , select.formStyle, input[type=date]
{
    font-family: 'Open Sans', sans-serif;
    border-radius:10px;
    padding:10px;
    font-size:14px;
    display:block;
    margin:10px 0;
    width:90%;
}
```

What you see above is a little bit of an advanced CSS declaration.

As you can see I’m targeting multiple HTML elements with the same CSS declaration by

separating each individual HTML element with a comma.

I'm also targeting two different types of inputs (type="text" as well as type="date") using brackets.

And you may not fully grasp this yet. However, I'm targeting EVERY input with the type equal to "text" EXCEPT ones where the class is "s". And I'm doing this using ":not(element in question)".

There are a few other advanced techniques to target HTML elements however, I don't want to overload you. We'll just cover some of the basic building blocks to get you going.

CSS Hierarchy

CSS is called Cascading Style Sheets because the rules set for CSS "cascade" down the document into other elements. For example, CSS styles that are lower in the document will over power styles that are higher in the document. A style on line 1 will be overpowered by a style on line 365 that targets the same element for instance. That means if you were to declare the default font-family for the document to be be Arial, then later on in the document you specify Tacoma, than the latter style would take effect.

CSS styles that are more specific in their CSS targeting will over power styles that are more vague in their targeting. For example a CSS style directed at all paragraph tags will be overpowered by a CSS style directed at all paragraph tags with a class or id of your choice.

If you are targeting one HTML object and you give the object two conflicting styles the browser will render the style that is lower in the document. For example if you give all headings a margin of 10 pixels and then later say that all headings should have a margin of 15 pixels, the browser will render the latter style.

If the HTML object itself has an inline style than in most cases the inline-style will be rendered. An example of an inline style would be as follows:

In most cases all of these conflicting styles can be bypassed with a declaration of "!important" before the closing semi-colon of the CSS declaration.

Chapter 8: Common HTML Elements & Their Rules

Divs

A div is essentially an HTML element which is used to hold other objects. I believe that div is short for division. Divs are important for laying out websites. Since the word div is not very descriptive, HTML5 has spawned new kinds of divs with different names. These new “div-like” elements are called: header, nav, section, aside, article, details, footer, and summary.

To use a div simply use this beginning and ending tag to wrap around the elements you wish to organize. Div’s are block level elements, which you will learn about later on in the book.

** **Div content goes in here.** **

The format is the same with the new HTML5 elements.

<header>Header content goes in here.**</header>**

Heading

Headings are just as they sound. They are text headings which range in size from H1-H6. H1 being the largest and most important in the eyes of search engines to H6 being the smallest and least important.

To use a heading simply wrap the heading text with beginning and ending header tags. Headings are block level elements. This may be obvious but don’t mismatch the tags for the heading.

<h1>This is an H1 heading.**</h1>**

<h2>This is an H2 heading.**</h2>**

…

<h6>This is an H6 heading.**</h6>**

Images

Images are also pretty self explanatory. Images are “inline block” level elements. To keep your code valid, make sure every image you use in your document has an “alt” specified. The “alt” attribute is important for when your image fails to load for any number of reasons. The alt attribute is also useful for visually impaired website visitors. In both cases the images would show the alternative text which describes the image so they can get a better feel for what the website is about.

To use an image simply use the following format.

****

Obviously the path should point to the place where the image is located and the alternative text should be replaced with actual alternative text. Since images can’t wrap around other elements the image tag ends itself, as you can see with the forward slash before the ending “greater than” symbol (>).

Quick note: your code will still be valid if you leave the alt attribute blank. However, this isn't the best practice. Essentially it is the alt attribute itself that makes the image valid and not the text inside.

Paragraphs, Spans & Blockquotes

A paragraph is exactly that…a paragraph. I will note that some people use unorthodox practices such as using a paragraph tag as a div. A paragraph is a block level element. To use a paragraph simply wrap the text in the following HTML tags.

** **Your paragraph text goes here.** **

A span is what you use when you have a piece of text which you wish to make special in some way. Spans are inline elements and are normally wrapped in block level elements, such as paragraphs for example.

To use a span, wrap your span text in the following HTML tags.

****Span text goes here.****

A blockquote is normally used when quoting something from another source or a person. A blockquote is a block level element and can be wrapped around other block level elements.

To use a blockquote simply wrap your blockquote in the following HTML tags.

<blockquote>Blockquote text goes here.**</blockquote>**

Unordered and Ordered Lists

If you've used Microsoft Word than surely you've encountered unordered and ordered lists. Unordered lists are bulleted lists and ordered lists are numbered. Both of which are block level elements.

This may seem weird to you, however most people use an extremely styled unordered list for their navigation bars. This is because an unordered list is specified for use with a list and a navigation bar is essentially a list of links.

To use an unordered list use the following format.

<ul>

**
List Item Goes Here
**

**
List Item Goes Here
**

**
List Item Goes Here
**

**
List Item Goes Here
**

**
**

UL means unordered list while LI means list item.

To use an ordered list use the following format.

<ol>

**
List Item 1 Goes Here
**

**
List Item 2 Goes Here
**

**
List Item 3 Goes Here
**

**
List Item 4 Goes Here
**

**
**

OL means ordered list while LI means list item.

Links

Links make the internet go around. Links are inline-elements. To use a link use the following format.

****Text, image, or whatever you wish to link goes here.****

Obviously keep in mind that the word path should be replaced with the url which you wish to link to.

Tables

In the past people would lay out entire websites using tables. Today however, that technique is frowned upon. Tables are great for keeping information neat and tidy. Tables are block level elements.

To use a table use the following format.

```
<table>
<th>
  <td>Jill</td>
  <td>Smith</td>
  <td>50</td>
</th>
<tr>
  <td>Eve</td>
  <td>Jackson</td>
  <td>94</td>
 </tr>
</table>
```

TH stands for table header, TR stands for table row and TD stands for table data. Essentially listing the top row as a table header isn't necessary as all rows can just be labeled as table rows. However, if you wish to style the top row using CSS you may want to utilize the TH tag as it will make it easier to target the top row.

Forms

Forms are a little beyond the scope of this book. Not because forms are difficult to use but because they normally require a bit of Javascript or other type of code to work. I won't go on to mention much about this. However, I will just give you a quick tip and say that in order for a text area to be valid code you must specify a cols and rows attribute. This

essentially gives the text area a width and height however you can later manipulate these dimensions with CSS if you wish.

Chapter 9: Common CSS Styles And Their Rules

I will go into a few different CSS techniques in greater detail later in the book so for now I'll just list a couple of the important CSS declarations and their rules.

Color

The color attribute always denotes the color of the text. To use the color attribute simply use the following format. Black is default when it comes to color, unless you're talking about a link.

color: #000000;

Later on in the book I'll explain how HTML color works and how to use the hexadecimal system for specifying colors. You can specify color using the hexadecimal system, rgb or even list them by name.

Fonts

There are a ton of CSS declarations which deal with fonts. We're going to briefly touch upon a few.

font-family: 1st choice, 2nd choice, last resort;

Fonts are specified by family. More on this later.

font-size: 12px;

Font size can be specified by pixel size, em's, you can even specify by using adjectives (large, small, etc.)

font-weight: bold;

Font weight can be specified by hundreds (100,200,300, up to 900). 100 being the thinnest and 900 being the thickest. You can also specify font weight by using bold, bolder, lighter & normal.

text-transform: uppercase;

Using text-transform you can specify that you want a particular block of text to be uppercase or lowercase. The text will then be rendered in that specific case regardless of how it's written in the actual HTML document.

line-height: 14px;

Using line-height you can specify the height between lines of text. You can specify the amount using pixels, ems, etc.

Height & Width

You can specify height and width for various objects. You can use pixels, em's, percentages, etc. as the unit of measurement. To use height and width use these formats:

height: 100px;

width: 50%;

Borders

There may come a time when you want to give something a border. You can specify all 4 borders at once or you can specify one border at a time. (There may be alternative ways to specify the border however, we'll just use the basics for now.)

border: 1px solid #ffffff;

To specify a border, begin with the border width in pixels, followed by the style (options include: solid, dotted, dashed, thin, inset, etc.), followed by the color. To specify a border on a specific side use:

border-right: 1px solid #ffffff;

border-left: 1px solid #ffffff;

border-top: 1px solid #ffffff;

border-bottom: 1px solid #ffffff;

Margins & Padding

I go into more detail later on in the book. However, margins form invisible barriers between objects.

Padding is essentially like a margin however it works as an internal barrier for objects inside of an HTML element. For example, if you have a div with a padding of 10 pixels all the way around. Any object inside of the div won't be able to get within 10 pixels of the edge.

I list out how to use these in detail later in the book.

Floats

Some HTML elements naturally stack on top of each other because they use up all the width allotted to them and some do not. In order to alleviate this for times when you have two objects that stack and wish to have them side by side, you can use floats.

To float an object use the following format:

float: left;

You can either float an object left, right or none.

A float basically tells a stack-able HTML element that it should only take up as much horizontal space as it actually needs rather than hogging up all the horizontal space. Then it tells the element to begin stacking either left, right or center.

Display

You can use the display attribute to display a block level element as inline, and inline element as block or not at all.

To use this feature use the following format:

display: block;

Your options here include block, inline, inline-block, as well as none. “Display:none;” is for when you don’t want to display an object at all.

Overflow

There might come a time when you have a div which you have specified a width or height for, and you have placed an object inside of the div which extends past the boundaries of its containing element.

In this case you have an overflow. To style this use the following format:

overflow: scroll;

Your options here include: scroll (which places scroll bars inside the element), hidden (which hides the excess) as well as visible (this shows the excess).

Position

Later on I’ll give you my technique for laying out a website. However, there are a few different positioning declarations.

To use the position declaration use the following format:

position: relative;

Position has a few options, such as relative (relative to it’s current position), fixed (fixed in one position relative to the browser window and doesn’t move) and absolute (this is an absolute position based on any containing elements.)

When you use the position property you must then specify the position based on the distance from either the top, left, right or bottom.

You do so using the following format:

top: 10px;

You can use any combination of top, left, right or bottom and use any valid unit of measure you wish. IE: pixels, points, ems, etc.

Things To Remember

There are honestly too many CSS declarations to mention here. You'll have to learn more as you go along. Some other great CSS declarations include border-radius which gives you rounded corners as well as some of the CSS3 transitions and animations which are a little more advanced.

Some things to remember are:

- You can't have more than one background image on a particular object
- HTML objects inherit default styles
- Child elements inherit styles from parent elements
- Styles that are lower in the stylesheet override styles which are higher up in the stylesheet (unless !important is used directly before the ending semi-colon)
- You only have to specify as much CSS as you need to. For example if an object inherited a style that you like, you don't have to redefine the style. Only use as much CSS as needed.

Chapter 10: Getting Started

Ok, now that we've gotten a few of the lesser boring facts out of the way we can begin to get our hands dirty a bit.

Anyone can read a book about coding a website however, it's not until you've taken that knowledge and began to apply it that your journey actually begins.

The Mockup

I begin each website that I build with a mockup. A mockup is essentially a Photoshop document of what the home page of your website is going to look like.

Figure 10.1 – Below is a mockup I created for a friend's website.

As you can see in Figure 10.1 this is an example of a mockup. It's a website design that I was messing around with for a friend who happens to loves a good bargain.

It's the website that I will be using as an example throughout this e-book because it's just a simple 2 column website.

The mockup is extremely important because it serves as the blue print to your website.

Once you have a mockup, you are ready to begin coding!

Setting Up Your Website Folder

I’m going to be honest here. When it comes to building a website, the sky is the limit.

You don’t have to format the folder that holds your website EXACTLY as I say.

You don’t have to format your code the way that you see in my examples.

However, if you keep your HTML documents clean and easy to understand. If you keep your folders clean and organized. If you keep all your images in one folder. These are all practices which will help you to code in a professional manner.

That being said, we’re going to set up a folder for your first HTML/CSS experiments.

1. Create a folder name it “website”
2. In that folder create 2 folders

- 1 called “css”
- 1 called “images”

The CSS folder will store your CSS files and your Images folder will store your image files.

Setting Up Your HTML & CSS Documents

Once you're ready to begin, open up your code editor of your choice and create two documents.

The first one will be named index.html and the second one will be named style.css. When it comes to HTML, the "index" usually denotes the "home page" and when you upload HTML to a server the server will serve the index as the home page.

Save this index file in the root folder that you just created, called "website". The CSS stylesheet will be saved in the CSS folder.

Setting up your HTML Document

Since the advent of HTML5, HTML documents have become much more streamlined. Below you'll see the bare minimum that you need to have on your page to have a valid HTML document.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>Here Is Where You Put The Title Of Your Website</title>
   <!—[if lt IE 9]>
   <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
   <![endif]—>
</head>
<body>

</body>
</html>
```

Open up your index.html and type the above code into your document. This is a barebones HTML document and pretty much everything that you see above is required to have a valid HTML document. (In the past a valid HTML document would have included a lot more declarations.)

- DOCTYPE = The document type
- HTML LANG = "en" (English) Note that the HTML tag begins at the top of the document and ends at the bottom to wrap up the document.
- Head = The place where your page title, links to CSS style sheets, Javascript code, and other miscellaneous code that is important for your website but isn't necessary for the aesthetics of your website.
- <!—[if lt IE 9]> = The code that follows this is to allow for HTML5 elements to work in older Internet Explorer browsers. I'm not sure that this line of code is necessary to have a valid document, however it doesn't hurt to add it. "if lt IE 9" means if later than Internet Explorer 9. The entire block of code basically means, if the browser viewing this document is Internet Explorer and if it is, if it's older than Internet Explorer than use this document linked here.
- Body = In between the opening and closing body tag is where your HTML goes.

Linking Your CSS Stylesheet

Unlike the HTML document the CSS document doesn't need as many declarations or requirements. The style sheet is basically up to you to style your HTML however you wish. (That's not to say that you don't have to use valid CSS.)

In order for your CSS styles to have any effect on your HTML document you need to link the stylesheet to it.

Go to your "index.html" and add the line of code which is featured below in bold.

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="utf-8">
    <title>Here Is Where You Put The Title Of Your Website</title>
    <link href="css/style.css" rel="stylesheet" type="text/css" />
    <!—[if lt IE 9]>
    <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]—>
</head>
<body>

</body>
</html>
```

This line of code links the stylesheet to your HTML document. The href is the path to get to the document you are trying to link.

Tying It All Together

Now that you have a blank HTML document and a stylesheet linked to it, you're ready to rock!

The best practice when coding a website is to have the HTML & CSS documents open in your text editor while you have the website opened in a browser.

Having the website open in a browser is essential because you want to be able to refresh the document to see the changes you make take effect. You can open the HTML document in a browser by right-clicking the file and going to "Open With" and choose the browser of your choice.

One thing that I like to do to make sure the CSS document really is linked to my HTML document is to denote a color for the background to see if it takes effect.

Go to your stylesheet and type the following.

```
body
{
    background-color:#000000;
}
```

Then once you've saved the stylesheet document you can refresh your browser and check to see that the background of the document has indeed changed.

The color code #000000 should denote black. Essentially you can put any 6 digit combination of letters and numbers there. However, if you make it #ffffff then the background will be white and you won't know if your changed actually took effect because that is already the default background color.

Backgrounds

I wasn’t sure initially how to structure this e-book. However, I’ve realized that the best way to go about teaching my knowledge of HTML/CSS was to go through some of the most useful properties and drop tiny morsels of wisdom along the way.

I know I mentioned that you should start each website off with a mockup. However, in this case since you’re just learning it’s not necessary.

If you’ve been following so far you should have a blank HTML document with a black (or possibly other color) background.

Go to your CSS document and make it blank again.

When it comes to doing anything HTML/CSS related, there are many ways to do every task.

That being said, we’re going to cover a few different ways to accomplish the same goal, giving our blank website a super cool background!

Tiled Backgrounds

One of the most popular background options is your standard tiled background.

One of my favorite websites for getting cool backgrounds for my projects is www.subtletepatterns.com, go there and download a background of your choice and save it in your images folder.

I choose the very first pattern that I saw, called “paisley” and you can see the pattern over to your right.

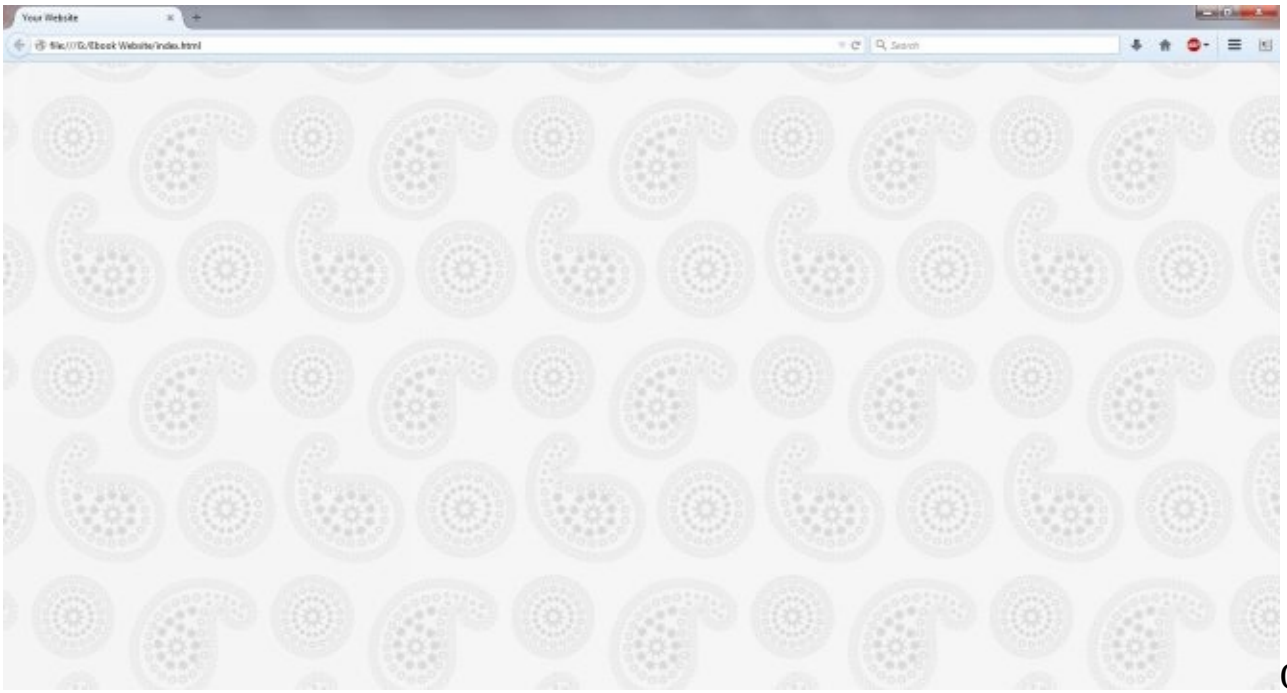
This pattern is seamless, meaning when you tile this pattern it will create a seamless background which is perfect for website backgrounds.

In order to tile this background we will have to edit our CSS stylesheet to reflect the changes that we wish to make.

Open up your stylesheet and then add the following code to add your tiled background.

```
body
{
    background-image:url(../images/paisley.png);
}
```

Figure 10.2 – Below is an HTML document with a tiled background.



Of course you should change “images” to the title of your images folder if you chose to name your folder differently, as well as change the name of your actual image file to match the file you have chosen.

Above is the result of my tiled background.

As I mentioned before with HTML elements all having default styles applied to them, CSS styles have default properties associated with them as well.

For instance having a declaration for a “background-image” always denotes that the background image will be tiled unless you declare otherwise.

You’ll see in a minute what other background option choices you have at your disposal.

A quick note about declaring paths.

Essentially, because our CSS file and our images are in different folders then we have to denote that in the path declaration.

```
body
{
    background-image:url(../images/paisley.png);
}
```

The “../” in front of the path is there in order to tell the stylesheet that the path to the image is not in the same folder but is one folder up relative to the CSS folder.

When working locally on your computer you will be dealing with what’s known as “relative” path declarations. This means that your paths will be based on where they are in relation to the current folder.

When you begin to upload your files to the internet you will be doing most of the same. However, there may come a time when you will need to use an absolute path declaration.

One example would be the one you witnessed above in the head of the HTML document.

<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>

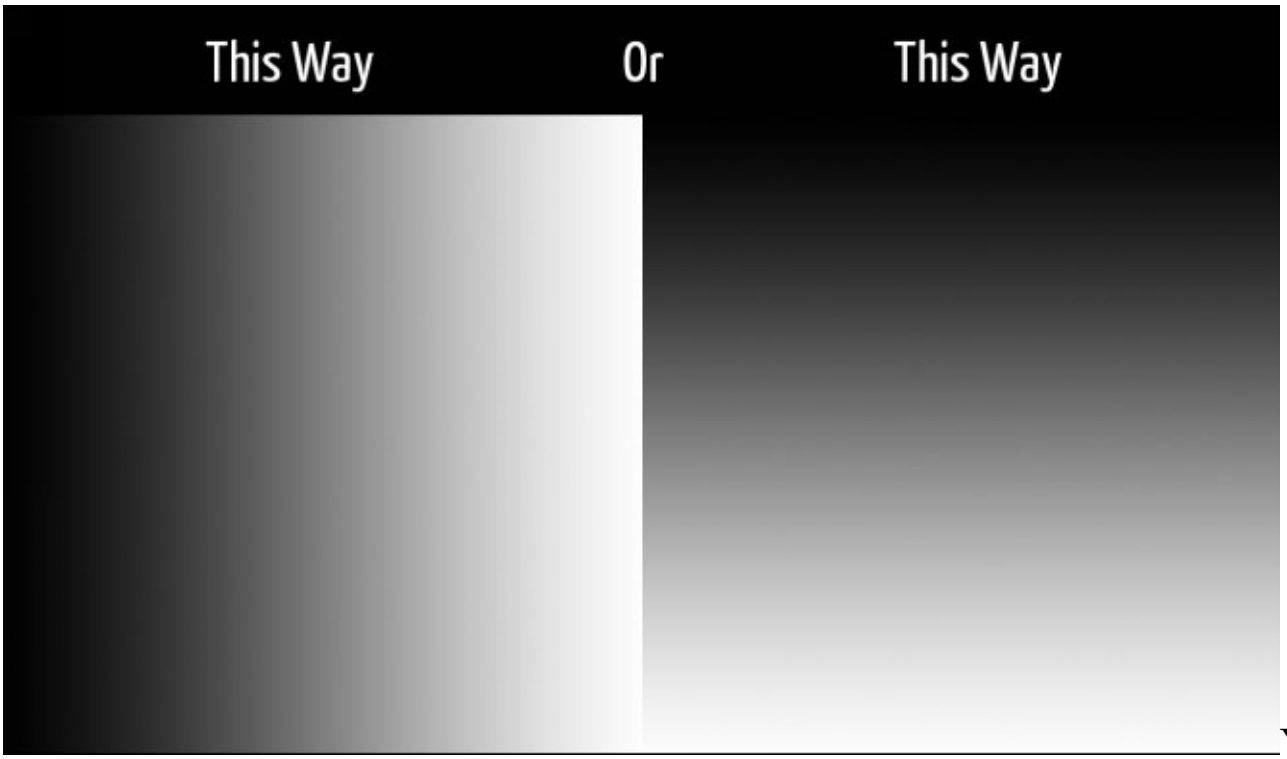
Because the Javascript file above is not relative to your website and is located on it's own url, you will need to declare the path beginning with the url.

Simply put, the path will need to be denoted beginning with "http://www.example.com/the-folder-in-question…"

Gradient Fun!

Ok, as I mentioned above, background images are tiled unless you specify otherwise.

Figure 10.3 – Below are two gradient examples. On the left is a gradient tiled on the y-axis and on the right is a gradient tiled on the x-axis.

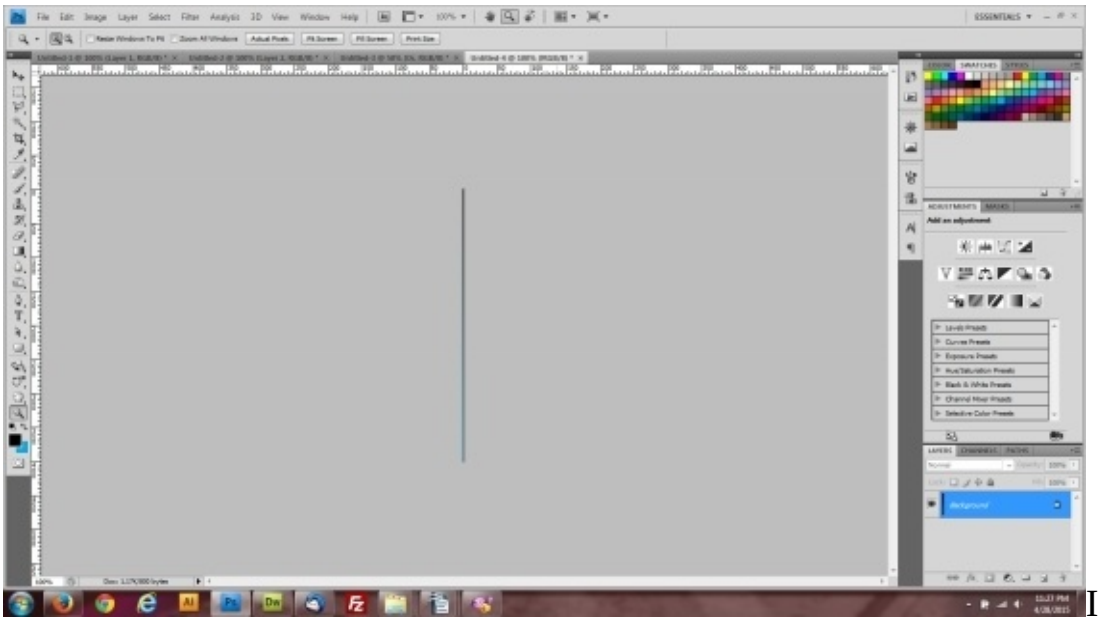


You can make a linear gradient in one of two ways. I used only a simple black to white gradient to illustrate my point. However, you can get as crazy as you wish with your gradient. You can even do circular gradients. (I'll show you how after this later)

In order to accomplish to top to bottom gradient you'll need an image to tile on it's x-axis (left and right). Since it repeats itself left and right the image only needs to be 1 pixel wide (you should always be concerned with using as little images as possible to save load time!)

Choose your two favorite colors to create a gradient. Create a file in Photoshop that is around 400 pixels tall and 1 pixel wide and fill the file with your gradient.

Figure 10.4 – Here I am creating a 1 pixel wide image which will serve as my gradient.



I decided to use black and blue for mine.

Save your image in your images folder and be sure to remember the name and the extension.

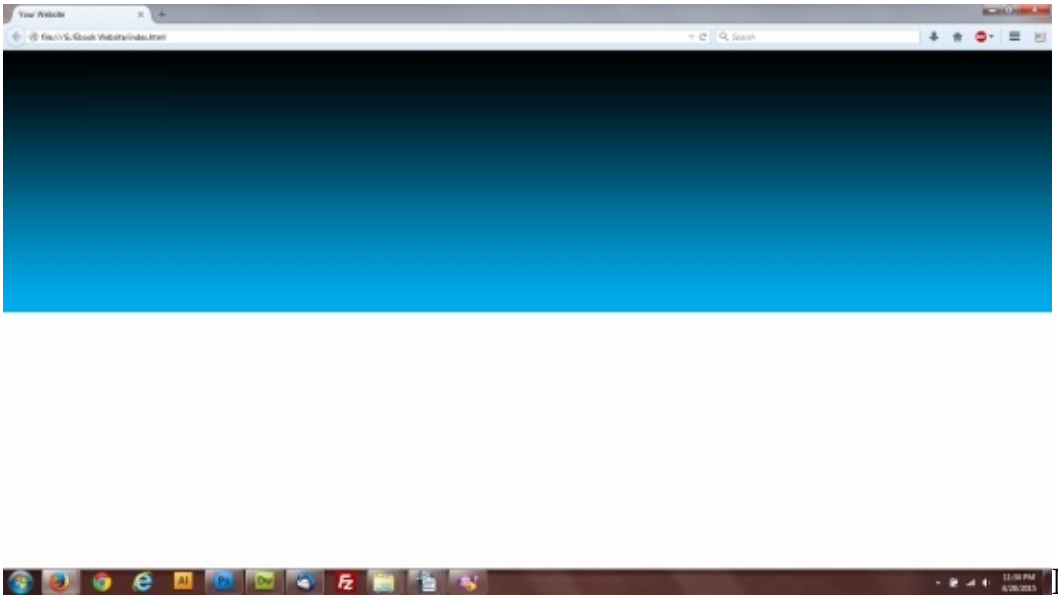
Go to your stylesheet and type:

```
body
{
    background-image:url(../images/gradient.png);
    background-repeat:repeat-x;
}
```

Obviously replace “gradient.png” with your filename and extension.

This is the result…

Figure 10.5 – Below is the result of my tiled gradient.

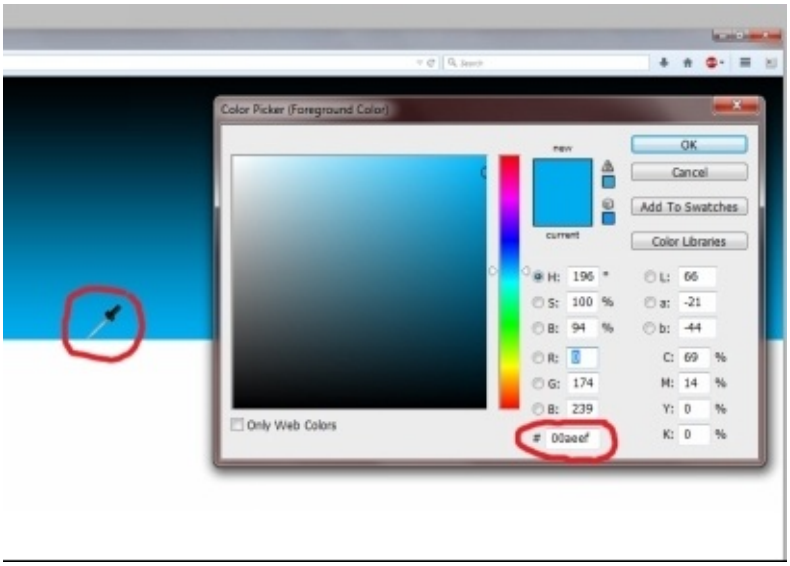


Now, I know what you’re thinking…I messed up. However, I wanted to show you this to illustrate my next point.

We declared a background image and told it to tile on the x-axis. It did, however the default for the background is still white and therefore there is white showing below where the image was tiled.

In order to fix this we'll have to also specify a background color along with the background image so the gradient will look fluid. Let's use our Photoshop eye-dropper tool in order to get the color code we need!

Figure 10.6 – Below I'm using the eye dropper tool to get to figure out the color at the very bottom of my gradient. This will help me to declare a background color and blend my gradient in with the page.



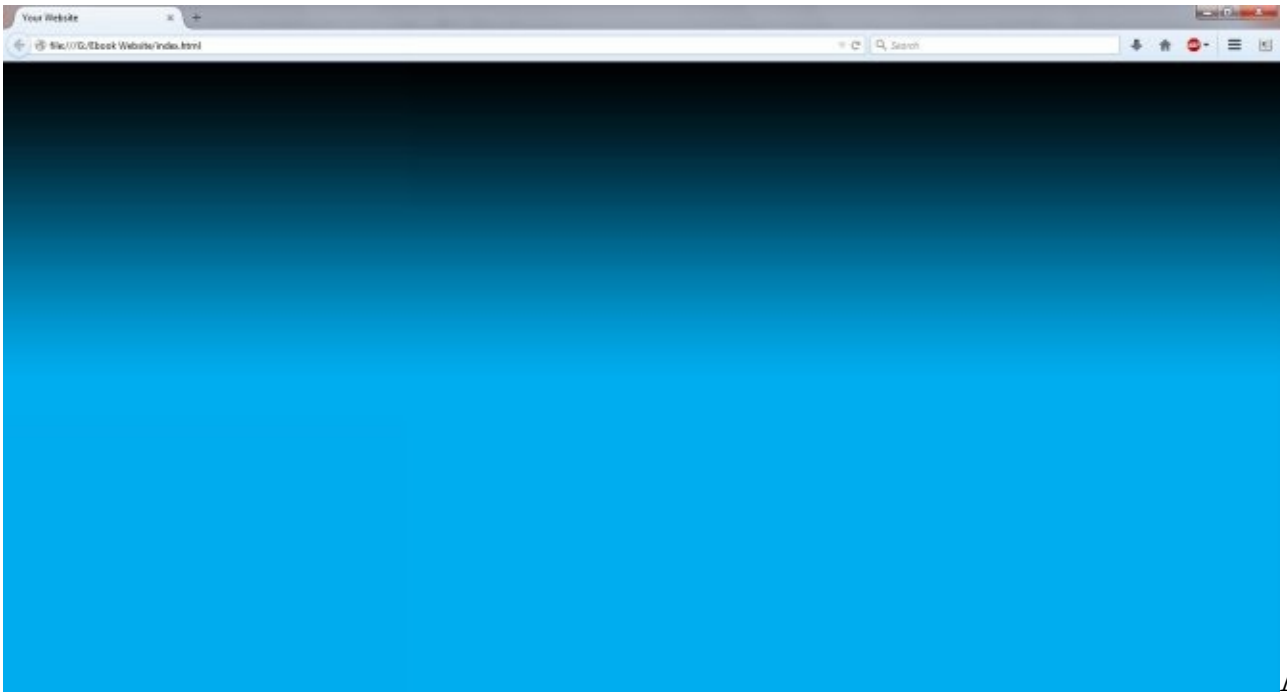
As you can see above we used the color picker tool to find the code for the color that we need. Now we can fix our gradient and make it continuous!

Go to your CSS and add:

```
body
{
    background-image:url(../images/gradient.png);
    background-repeat:repeat-x;
    background-color:#00aeef;
}
```

Obviously replace the hexcode number with the color which you need.

Figure 10.7 – Below is my new beautiful continuous gradient!

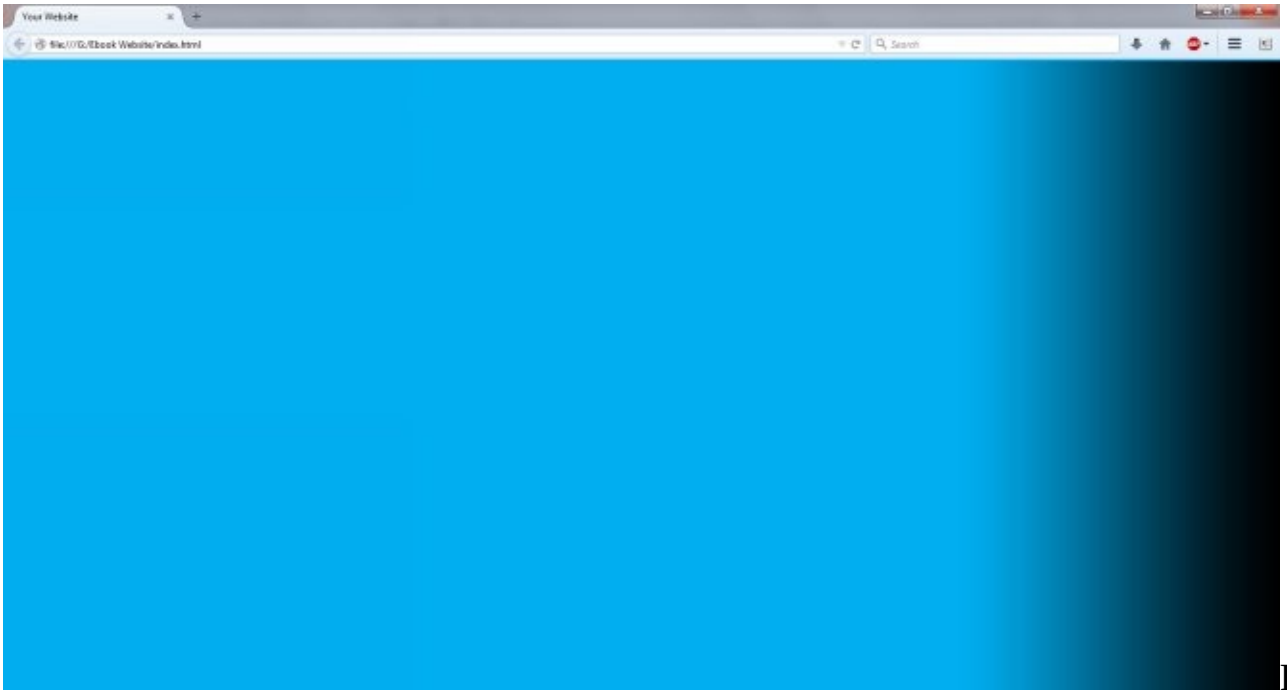

And our gradient is now fixed!

Using the above logic we can create a left to right gradient by following the same steps. Except in this case we use an image that is 1 pixel tall and “X” pixels wide. And instead of saying “background-repeat: repeat-x;” we change it to “background-repeat: repeat-y”.

Not only do we have control over how the background repeats but we also have control over the position. The default position for a background is always in the upper left corner.

This allows for a variety of different options.

Figure 10.8 – Getting funky with the gradients. This gradient runs along the right side of the browser window!


For example, the above gradient would be created with a 1 pixel tall by “X” pixels wide image which is repeating on the y-axis and the background-position is “right”.

The code would look like this.

```
body
{
    background-image:url(../images/gradientexample.png);
    background-repeat:repeat-y;
    background-color:#00aeef;
    background-position:right;
}
```

If you're worried about memorizing each CSS declaration, don't. Each of these things can be Googled at a later date if you forget. The most important thing is that you understand the underlying principals and see what is possible using HTML/CSS.

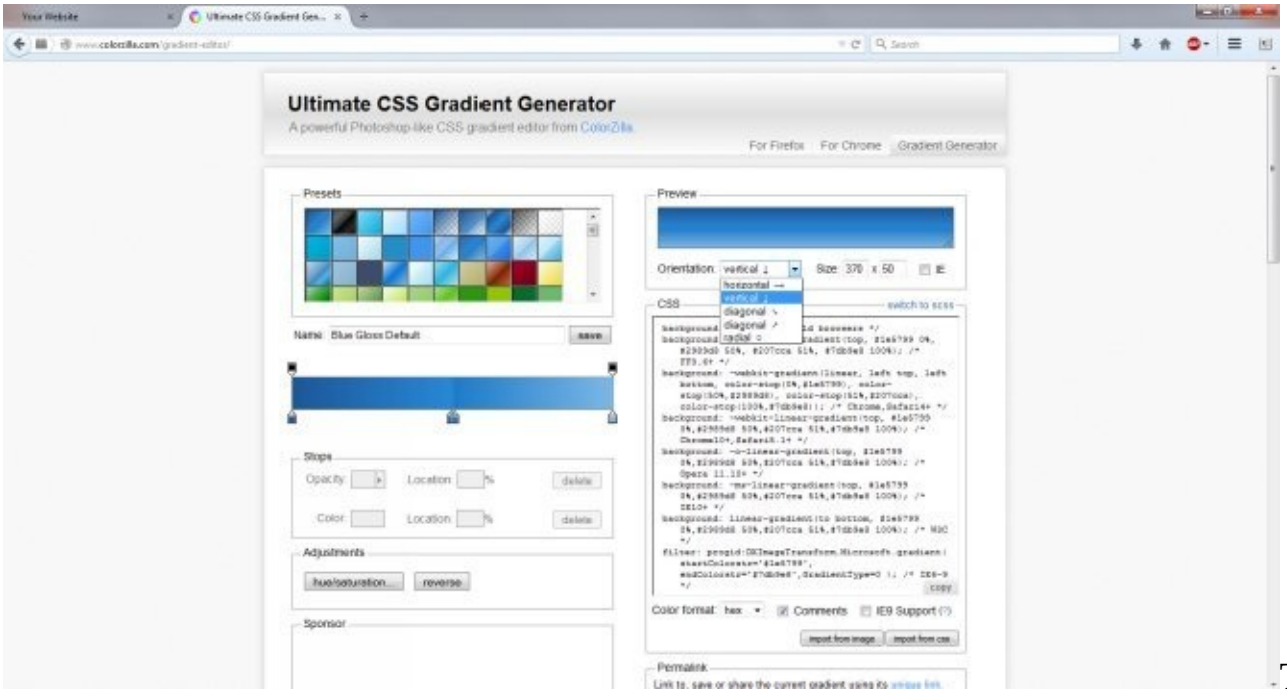
Radial Gradients & Image-less Gradients

The following gradient options have only been available recently. HTML5 and CSS3 have brought along many, extremely cool features that were once only available using Javascript or extremely image-heavy solutions.

However, now we have the ability to create radial gradients and image-less gradients using only CSS! The only downside to this solution is that it isn't compatible with older versions of Internet Explorer. However, as time progresses, the older versions of IE will die off and worrying about cross browser compatibility will be a thing of the past.

I personally use this website http://www.colorzilla.com/gradient-editor/ to create my gradients. I found it by Googling "gradient generator" and it's the first organic search result.

Figure 10.9 – This "Ultimate CSS Gradient Generator" is awesome for creating image-less gradients of all types!.



This website is awesome because:

- It allows you to edit the gradient just like you would using image editing software
- You can change the orientation of the gradient from vertical, horizontal, diagonal as well as radial
- The CSS is generated for you and they included CSS declarations for the most common browsers as well as a backwards compatible solution for older IE browsers
- They have pre-made gradients
- Best of all it saves you a ton of time

Once you've created a gradient you're happy with, simply copy the CSS and paste it into your style sheet.

Non-repeating Background Images

Above I've mentioned a few techniques for creating background effects. However, there may come a time when you want an element to have just one picture as it's background.

You may want the picture to be featured in the bottom right corner. Or you may want a heading to have a background image just to the left of the text.

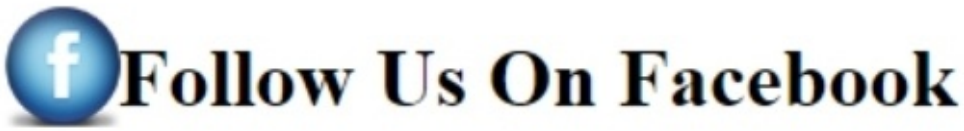
In order to do this you need to give the element a background image as stated above. However, we need to state that the background should not repeat.

Finally we can specify a position for the background. Options for placing the background image include top, bottom or center followed by left or right. You can use a combination of two positions when it makes sense. For example, "top right" or "bottom left".

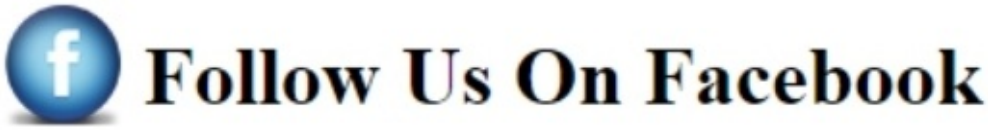
Keep in mind that if you're placing a background image in a text element and you want the image to be to the left of your text, you will have to use padding in order to make sure that your text doesn't cover the background image. Also, if your text isn't as tall as your image you will have to use padding to make sure the background image doesn't get clipped.

```
h1
{
    background-image:url(../images/fb.png);
    background-repeat:no-repeat;
    background-position:left;
    padding:10px 0 0 45px;
}
```

Above is the CSS that I started with and below is the result.



As you can see my text is a little too close to the picture so to alleviate that I have to give it a little bit more padding on the left side.



Above is the result. Obviously we probably don't want to use Times New Roman however, we'll learn about fonts a little later in the book.

Chapter 11: Preparing Images For The Internet

So hopefully if you're reading this book you have a little bit of knowledge of graphic design. And hopefully you have some type of image editing software.

I'll be using Photoshop to show you how to prepare images for the internet.

This is extremely handy to know and is normally something that you would learn through trial and error.

72 DPI

If you're a graphic designer or come from a print background than you may be familiar with image resolution.

Typically, when working with images that are being prepared for print the standard resolution is 300dpi. However, when it comes to the internet most images are 72 dpi. In some rare instances you may encounter pictures with resolutions which are upwards of around 90 something dpi, 72 being the standard.

You also want to make sure that the images are saved using the correct file extension and that you are optimized for the file extension you've chosen.

Most Common Image Extensions

The most common image extensions that you will deal with when it comes to creating (pixel based) images for the web are .jpg, .gif and .png. Of course you also have .svg's which are "Scalable Vector Graphics" however we won't be dealing with those for the time being.

Each of these extensions have their advantages and disadvantages when it comes to saving certain types of images.

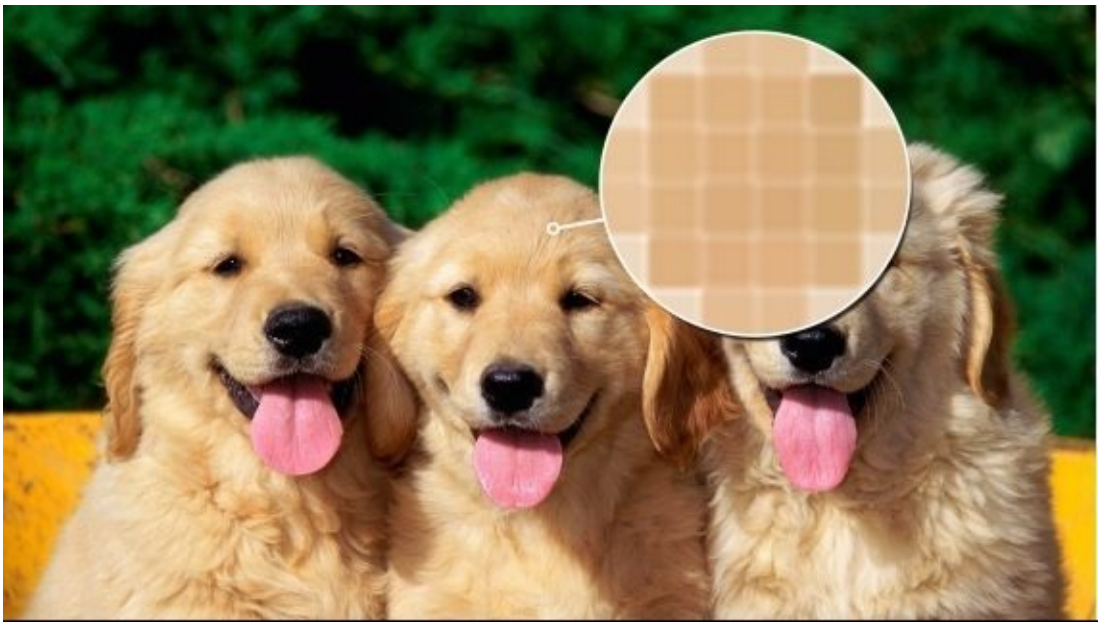
When it comes to the internet, how you save your images matters. If you save your file in a format that isn't optimal for the image you're saving then it will take longer to load.

When To Use A .JPG

A .jpg is best for when the image you are trying to save is photographic in nature.

What I mean by that, is if the image is made from thousands of tiny pixels of varying color and shade than saving it as a .jpg will be the best option for you.

Figure 11.1 – As you can see this image is made up of thousands of tiny squares of color.



As you can see from the image above, that tiny spot on that dog's forehead alone has so much variance in the color from pixel to pixel that saving this image as a .png or .gif will cause your website to load extremely slow.

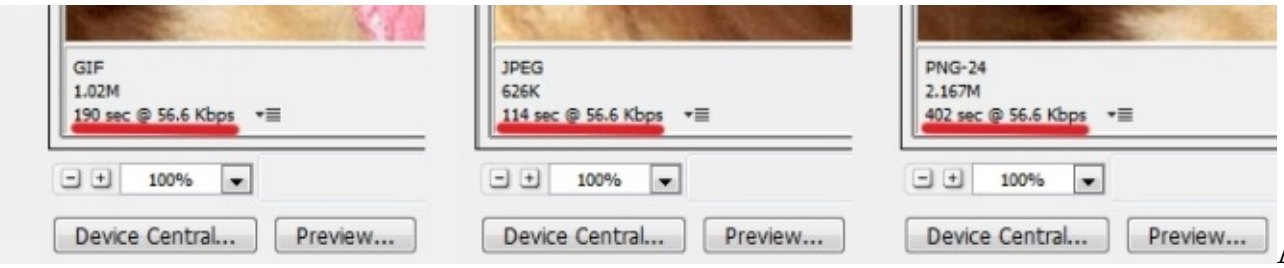
Saving your photographic images as a .jpg is great because in photo editing programs such as Photoshop, it gives you the option to lower the quality of the image so that the file will be smaller.

You are able to lower the quality of the image and see how the final image will be effected, as well as how long the image will take to load BEFORE you actually save the file.

What I like to do is lower the image quality, while watching the estimated load time of the image. This allows me to get the image to a point where the image quality is acceptable, while the image load time is not excessive.

I took this same image above (it is 1920x1080 pixels this is very large for internet pictures) and using the “Save For Web And Devices” menu in Photoshop figured out how long this same image would take to load using the most common file extensions.

Figure 11.2 – Below you can see a preview of the load time on the puppy picture using various image extensions.



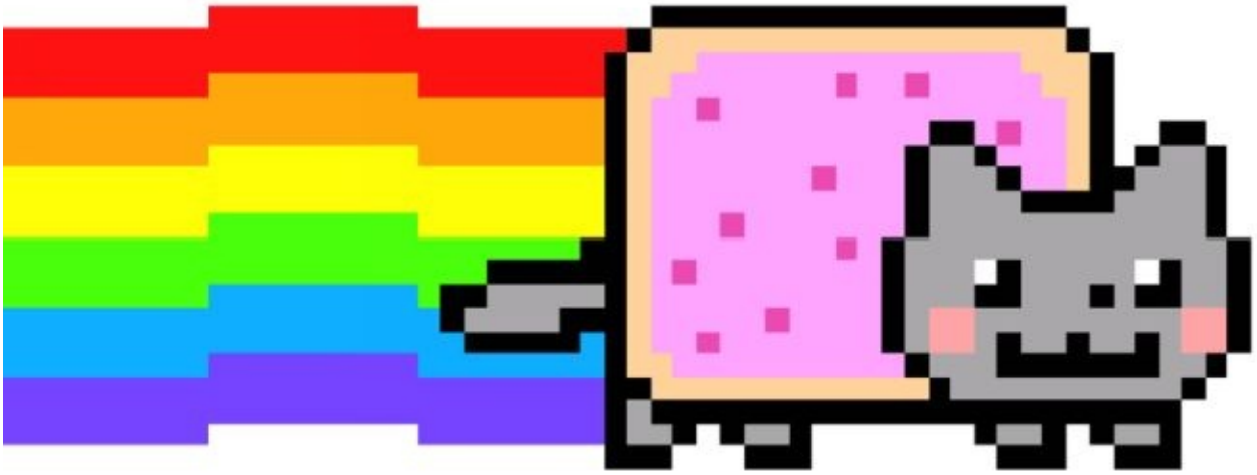
As you can see, for the same exact image, saving this image as a .png would cause this image to take 402 seconds @ 56.6 Kbps, a .gif would take 190 seconds and a .jpg at 100% quality would take 114 seconds.

Keep in mind the .png and .gif could be altered to load in less time however the image quality would drastically suffer but, the .jpg could be lowered in quality with minimal difference in the final product and still load extremely fast.

.PNG’s And .GIF’s

.PNG’s and .GIF’s are best used when the image in question is based on areas of solid color rather than thousands of pixels of various color.

Figure 11.3 – Since the image below is based off blocks of solid color a .png or .gif file extension would work best.

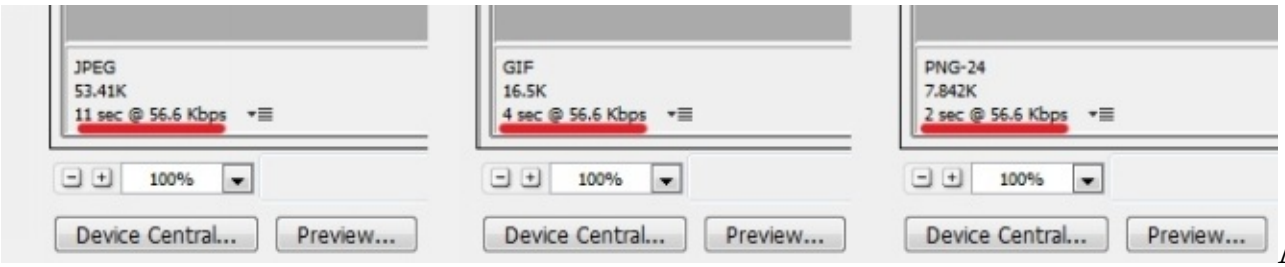


The picture above would be better suited to be saved as either a .gif or a .png. If the image were to be animated it would most certainly be saved as a .gif because .gif’s support animation.

Both .gif and .png support transparent objects, however .png is much better at transparency.

You probably can’t tell by looking at the above picture however it has a transparent background. When you try to save an object with a transparent background as a .jpg the transparent part will be turned white.

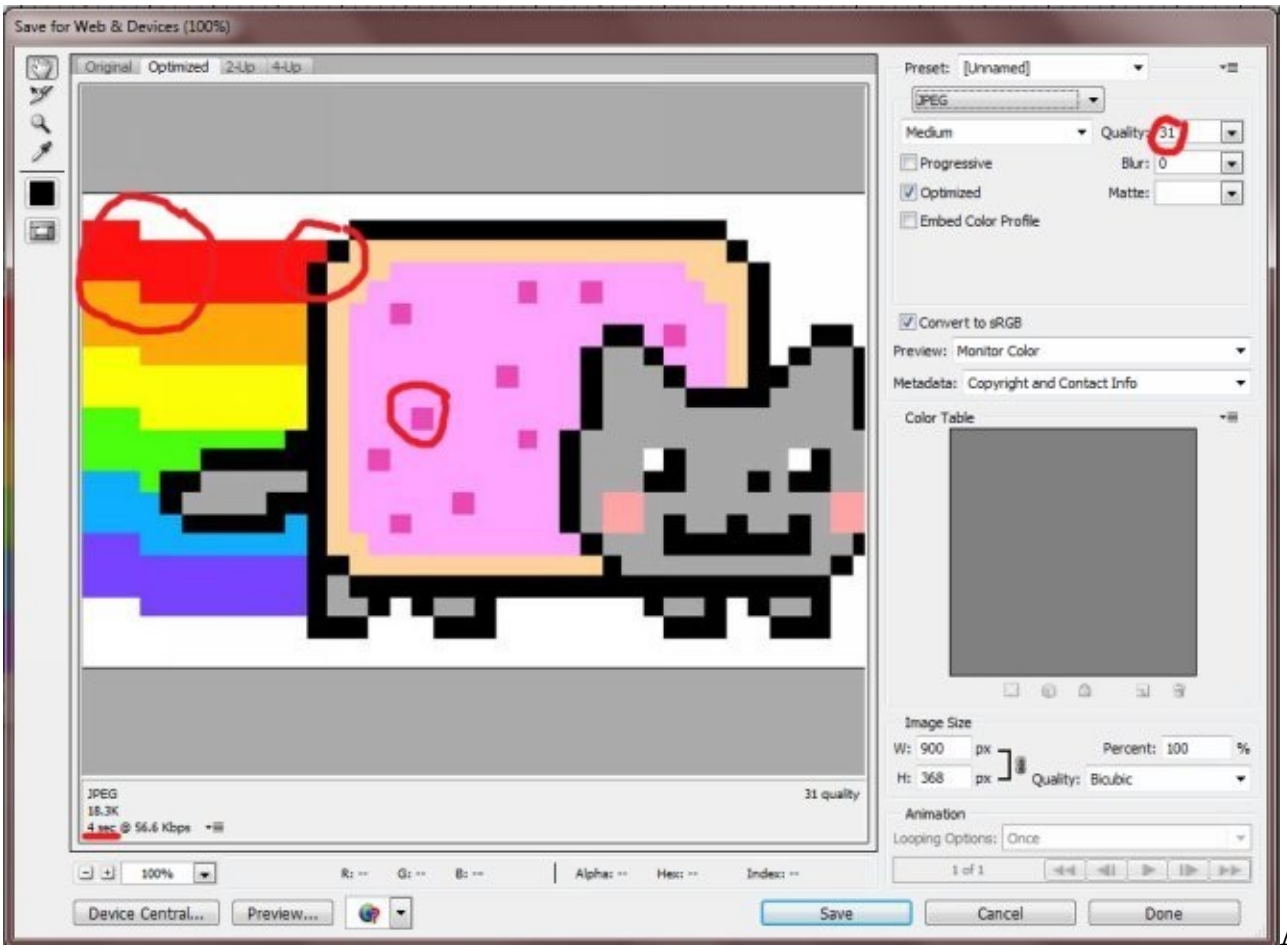
Let’s do the same experiment as we did with the dog pictures.



As you can see, when trying to save the above image as a .jpg at 100% image quality the image will take more than twice the amount of time it would have taken the .gif and .png to load. Not to mention the fact that the .gif and the .png will have transparency.

Also not to mention the fact that if if you did try to save the above image as a .jpg and you tried to lower the quality in order to save load time, this would happen.

Figure 11.4 – When attempting to save this image as a .jpg some mild "compression artifacts" beings to appear.



As you can see, I lowered the quality of the image down to 31%. At 31% quality the image takes 4 seconds to load. And as you can see, lowering the quality of the image has caused the image to suffer greatly.

If you'll look at the areas on the image that I have circled you can see subtle variances in the color of the individual pixels causing an extremely ugly effect to take place.

This is known as a .jpg "artifact".

A Word About Transparency

When I was going to college to learn about website design. I created a website for a school project. The website had an image with a transparent background featured in front

of a colored background.

When I looked closer I noticed that there were tiny pixels around the edge of the transparent image which weren't quite right.

In order to illustrate the problem, I'll create a web page with a red background, save the image above as a .png as well as a .gif and place the image on the page to show you the issue I was having.

Figure 11.5 – The small amount of white pixels surrounding the picture below is an issue of "matting".

This is what happens when you save the image as a .gif. As you can see there are white pixels surrounding the image in certain places.

Now, let's try the same thing with a .png.

Figure 11.6 – Same image saved as a .png. The .png offers superior transparency without having to mess around with "matting".

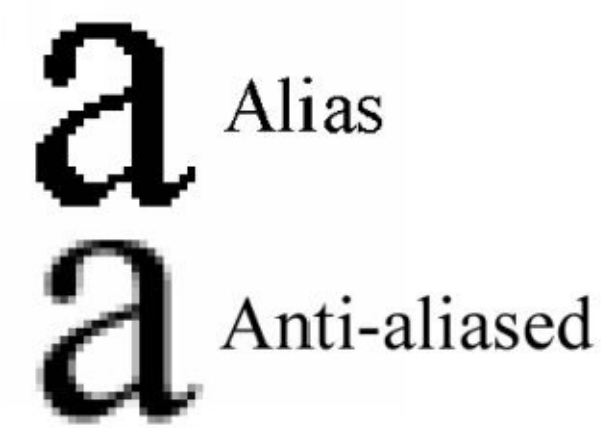
As you

can see the image now doesn't have those pesky pixels there anymore!

But what caused this?

In all honesty, this was a problem which effected me for a while when I first started web design. And even when I asked my teacher at the time he seemed to be of no help.

What I've found is that this effect is caused because of what is known as "anti-aliased" pixels. When it comes to transparent pixel based pictures. The pixels around the outside of the image can't be a solid color. If that were the case then the outer edge of the image would look "blocky".



The image to the right illustrates this fact. It's a little hard to create rounded edges and beautiful, picturesque photographs using only tiny squares of color. So in order to alleviate this problem, anti-aliasing is done and it essentially creates half filled in pixels around the edge of the image to create the illusion of a smooth edge.

This causes problems when it comes to saving transparent images because if you try to save this image as a .gif these anti-aliased pixels will have a white background by default.

This causes the effect that you saw above with the white pixels around our cat image. However, when saving your transparent images in .gif as well as .png format you have the option to choose a "matte" that is a background color for those anti-aliased pixels.

That way, if you want to use a transparent image on a red background you can specify red as the matte color so the anti-aliased pixels will be less noticeable.

Of course you can skip all this by saving the image as a .png-24 which has superior transparency abilities. However, high quality, transparent .png's can sometimes take a long time to load.

For this reason, I use a website called www.tinypng.com to lower the file size on my transparent .png's as well as all my .png's in general.

Figure 11.7 – The car image below had an extremely large file size before I used tinypng.com to compress it.

Tiny PNG is great for when you have a .jpg quality image that you still want to have superior transparency. The car in the above website is fully transparent, including the shadow the car is casting as well as the light rays coming from behind the car. Saving a transparent image like this will cause the image to take a long time to load. However, with image compression software it's a piece of cake!

Chapter 12: Manipulating Placement Of HTML Objects

Knowing how to place objects where you want them is extremely important when it comes to designing a website the way you want it to look.

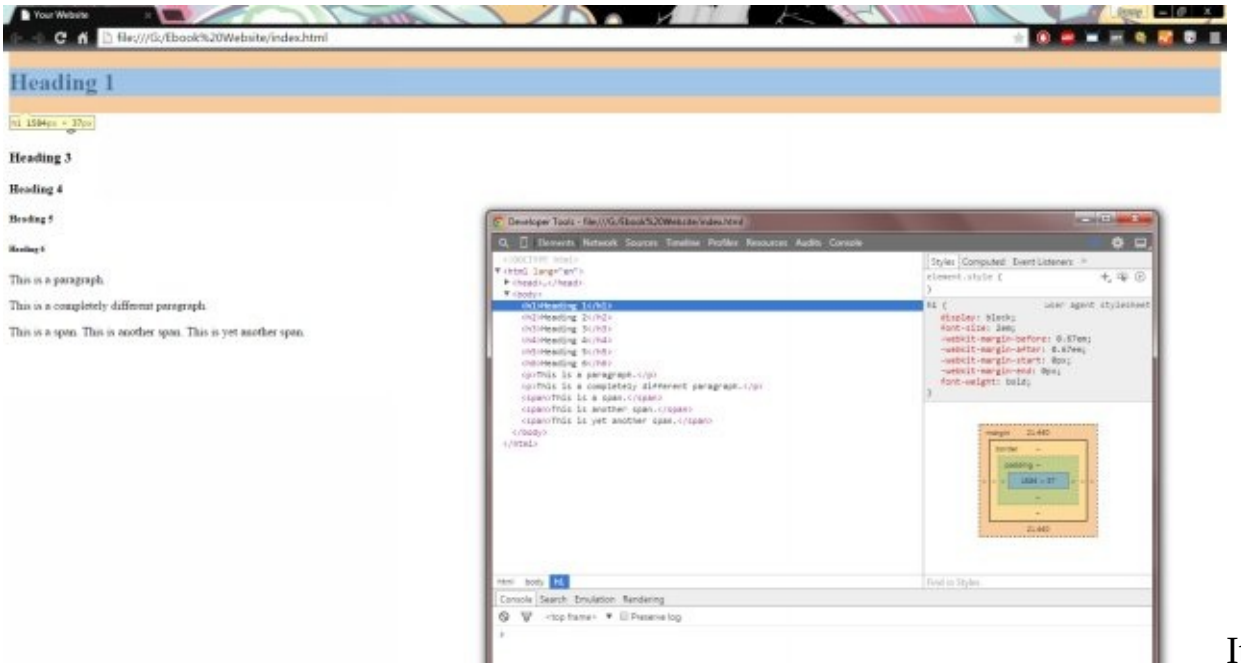
In order to do this however, you must first know a little bit about how HTML objects behave.

HTML objects are either considered to be “inline” or “block” level elements.

Inline vs. Block Level Elements

In order to illustrate the difference between an inline element and a block level element I'll be creating different HTML elements on a blank page and then inspecting them using Google Chrome's element inspect option.

Figure 12.1 – I clicked "inspect element" on the Heading 1 tag at the top of the page to show you that block level elements take up as much horizontal space as possible. Even though the text itself only utilizes about 1/16th of the page it is still causing elements below it to be placed on the next line.



It is a little hard to tell, but I filled the page with a bunch of block level elements and a few inline elements.

The difference between a block level element and an inline element is that a block level element will take up as much space as it **needs** vertically and as much space as it **can** horizontally.

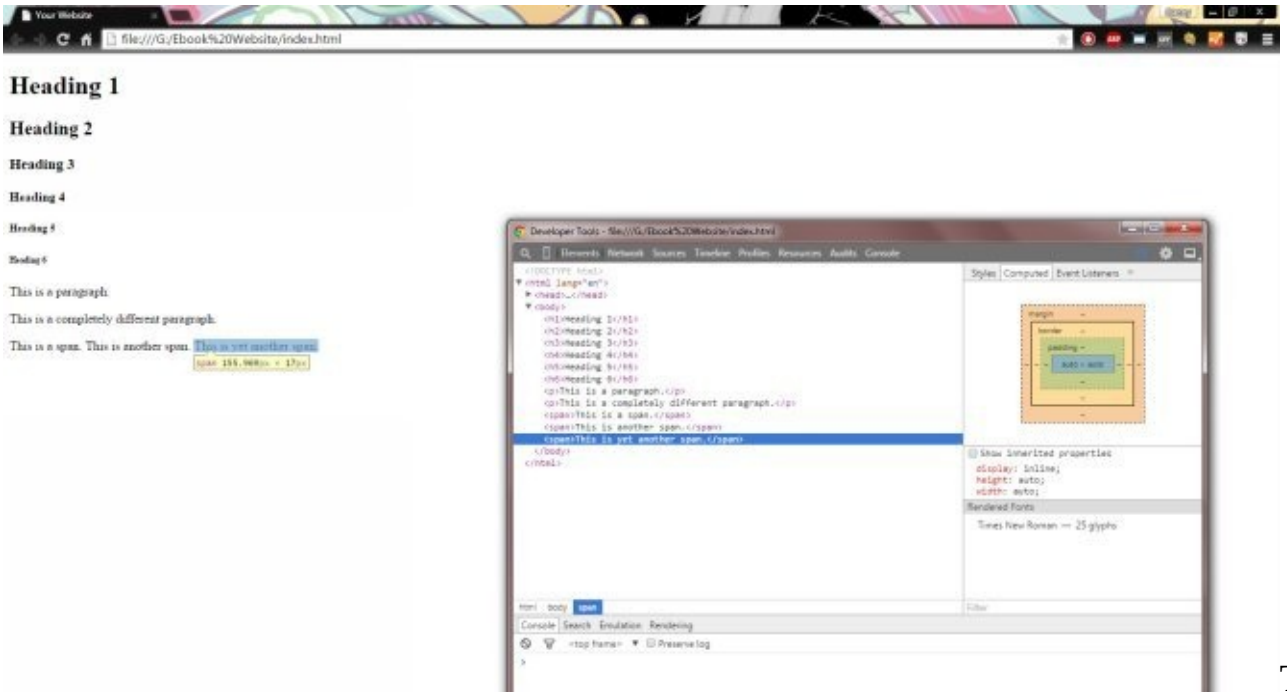
As you can see I have inspected the H1 tag (that's whats known as a heading 1) and the blue and pink region illustrate the amount of space that the object takes up. Clearly, the "Heading 1" element is taking up the entire width of the page.

If I were to put the same H1 tag in a container which spanned only half the width of the browser window the H1 tag would still take up as much room as it could horizontally. Except in that case it would be limited by it's container which only allows it to take up half the width of the browser window.

The page above has mostly block level elements and as you can see they stack on top of each other.

The very last line however, has 3 different inline elements. Let's choose the "inspect element" option to learn more about the nature of those inline elements.

Figure 12.2 – I clicked "inspect element" on a span element. The line with the span element contains 3 different span elements. Notice that they are all on the same line. This is because they are inline elements.



The very last line of the document includes 3 “span” tags. A span is a type based element which holds less significance/ importance than a paragraph which could be considered the span’s older brother.

As you can see as I inspected the span element, the element itself takes up only as much space as it needs to. This is the same with all inline elements.

The relationship between block level and inline level elements is important to know because there may come a time where you want to have two block level elements directly next to each other.

Or there may come a time where you want to display an inline element as though it were a block level element.

Inline-block Elements

Inline-block elements are inline elements that have a width and a height, rather than only taking up as much space as necessary. This is something that is encountered rarely but is still great to know and another tool in your arsenal.

“Floating” Elements

Before we get too deep into this I’ll show you basically how people normally set up websites.

I will do so by taking a mockup which I’ve already done and using a yellow outlins to show the different dividing boxes which I used to create the layout.

In modern website development people use what is known as a “div” to separate and layout their content.

A div is a block level HTML element and is very handy for laying out websites if you know how to manipulate them.

Figure 12.3 – I wrapped every div this page contains in a yellow box to illustrate laying out a website.

If you’re a little confused than that’s great! It means you understand how a block level element works.

If you’re not confused than basically each of these boxes, based on the way they’re supposed to behave should be stacked on top of each other. In fact they would be, if it weren’t for CSS.

Let’s revisit the picture we saw earlier in the book.

As you can see the image on the right is the exact same code with the CSS removed.

So this tells us is that with CSS we are able to lay out objects the way we want them.

If you want two block level elements to sit side by side then they need to:

1. **Each element needs to have a specified width. Block level elements usually take up as much room as they can horizontally so without a specified width they will just stack on top of each other.**
2. **The combined width of all the elements to be placed side by side can not be greater than the width of the containing element. Let's say you're making a website that is 1,000 pixels wide. You can't create two 600 pixel wide div's and expect them to stack side by side. Also, you can't have two 500 pixel wide divs with 10 pixel margins and expect them to stack side by side because the combined width plus the 10 pixel margins will make the total width of the objects 520 pixels.**
3. **The objects need to be "floated".** A float, is a CSS style declaration which tells the object to basically stack horizontally to either the right or left. This is based on whether you specify the object float either right or left. The CSS declaration looks like this.

```
Div
{
   float:left;
}
```

You can also specify that there be no float by typing (float:none;). This practice of floating objects is extremely important for modern website design. In the past people would have to use tables to get the same type of effect.

“Clearing” Elements

Floats and clears go hand in hand. Let’s say that you have two floating objects inside of a container object.

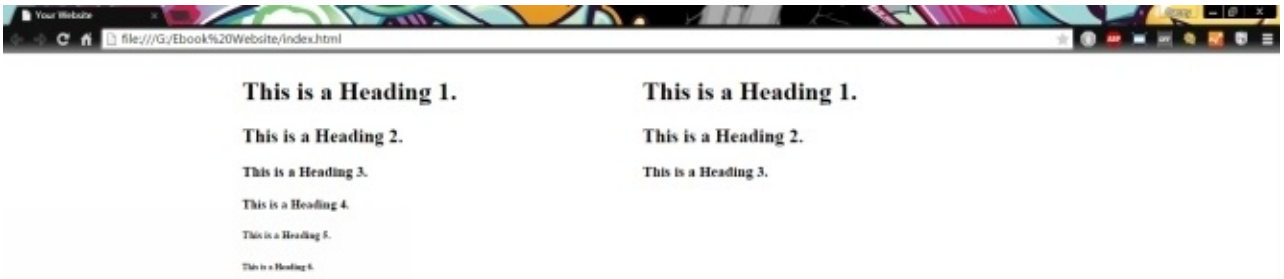
Using CSS you have specified a red background for the container element and the two floating objects don’t have any kind of background.

The container object is wrapped around both objects so what will the background of the entire area be? If you’re a logical person than you would assume that since the container has a red background that the background of the entire area would be red.

However, since floating objects don’t behave quite like block level elements anymore they don’t trigger the container element to wrap completely around both floating elements.

I know it sounds a little confusing so I’ll just illustrate it.

Figure 12.3 – Below are two divs side by side inside of a container div. I specified for the container div to have a red background however there is no background color present…what gives???



Ok, so above you’ll see the result of what I was talking about.

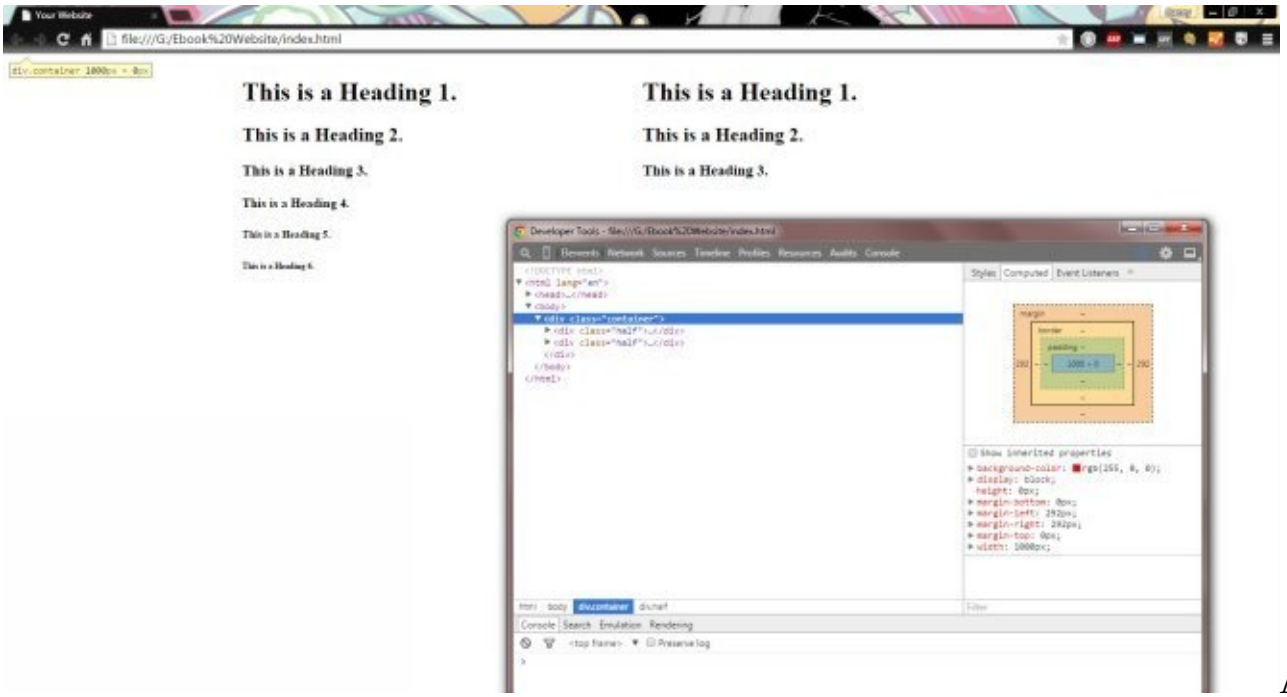
I created a div with a class of container. I said for the div to be 1,000 pixels wide and be centered in the middle of the page.

Within that container div I created two divs which are supposed to have a width of 50% and are to be floated left.

They are stacked side by side however the container isn’t wrapped around them. We know this because the container div is supposed to have a red background.

Let’s use the “inspect element” option to see what happened to our container div.

Figure 12.4 – I clicked “inspect element” on the container div only to see that it isn’t wrapped around the elements inside it.

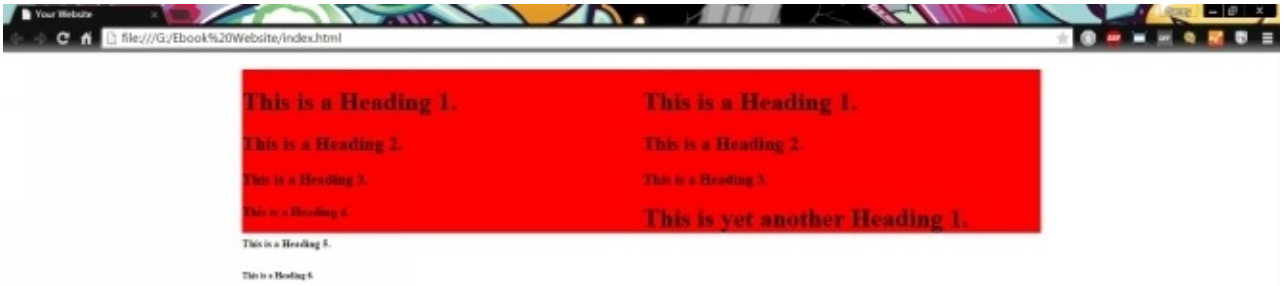


As you can see the container is not wrapping around the two div's which are inside of it.

The reason for this is because the HTML document doesn't really see floating objects *as if they were there*.

It's a little hard to explain so let's put a block level element after our two floating divs and see what happens.

Figure 12.4 – It's beginning to look a little better, we can see the red background but it isn't completely covering the elements inside it.



I placed a H1 tag after both divs and as you can see the container now sees that there is a block level element inside it and wraps around it.

You may be wondering why the H1 tag didn't go below both of the floating divs.

This is because, just like we write from left to right. HTML goes from left to right. And if you don't specify otherwise each element will try to go directly into the upper left corner of the screen.

In order for the H1 tag we just placed in the document to go beneath the two floating objects, we have to use another CSS property called a "clear".

I purposely put half as many objects into the right column div as the left column div so it would be shorter. And because there is nothing there to tell the H1 tag to clear both of those floating elements, it goes to the nearest upper left corner that is available.

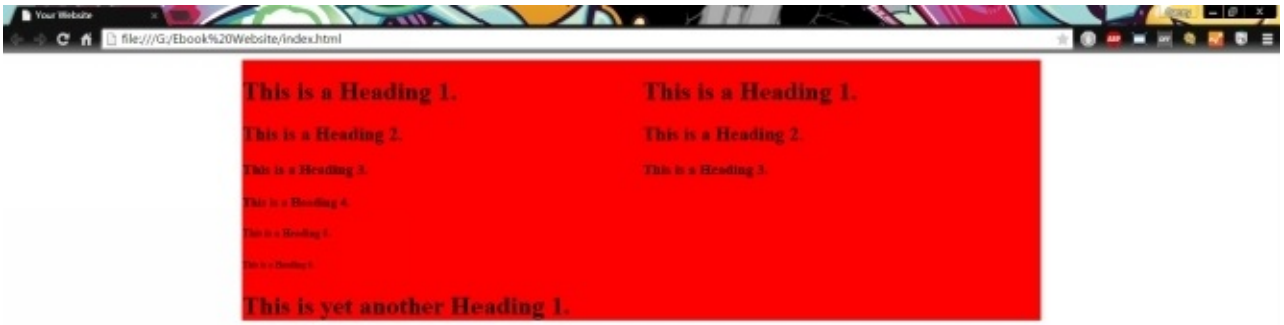
The clear property has a few different options. You can either clear objects to the left, to the right, or both.

Now, let me give that H1 tag a class of clear. And then I can use CSS to specify that any element with a class of clear should clear “both”.

```
<body>
<div class="container">
    <div class="half">
        <h1>This is a Heading 1.</h1>
        <h2>This is a Heading 2.</h2>
        <h3>This is a Heading 3.</h3>
        <h4>This is a Heading 4.</h4>
        <h5>This is a Heading 5.</h5>
        <h6>This is a Heading 6.</h6>
    </div>
    <div class="half">
        <h1>This is a Heading 1.</h1>
        <h2>This is a Heading 2.</h2>
        <h3>This is a Heading 3.</h3>
    </div>
    <h1 class ="clear">This is yet another Heading 1.</h1>
</div>
</body>
```

```
.container
{
    width:1000px;
    margin:0 auto;
    background-color:#f00;
}

.half
{
    width:50%;
    float:left;
}

.clear
{
    clear:both;
}
```

Above is the HTML as well as the CSS that’s involved. And below is the result.



As you can see, floats and clears are extremely helpful when it comes to laying out websites.

Next we will learn about margins and padding which will also be helpful when it comes to lay outs as well as styling objects the way you want them to look.

Chapter 13: Margins And Padding

Margins and padding are both extremely powerful tools. Both of which do almost the same thing with a few minor differences.

Margins

A margin works to basically serve as a barrier, pushing objects away from other objects.

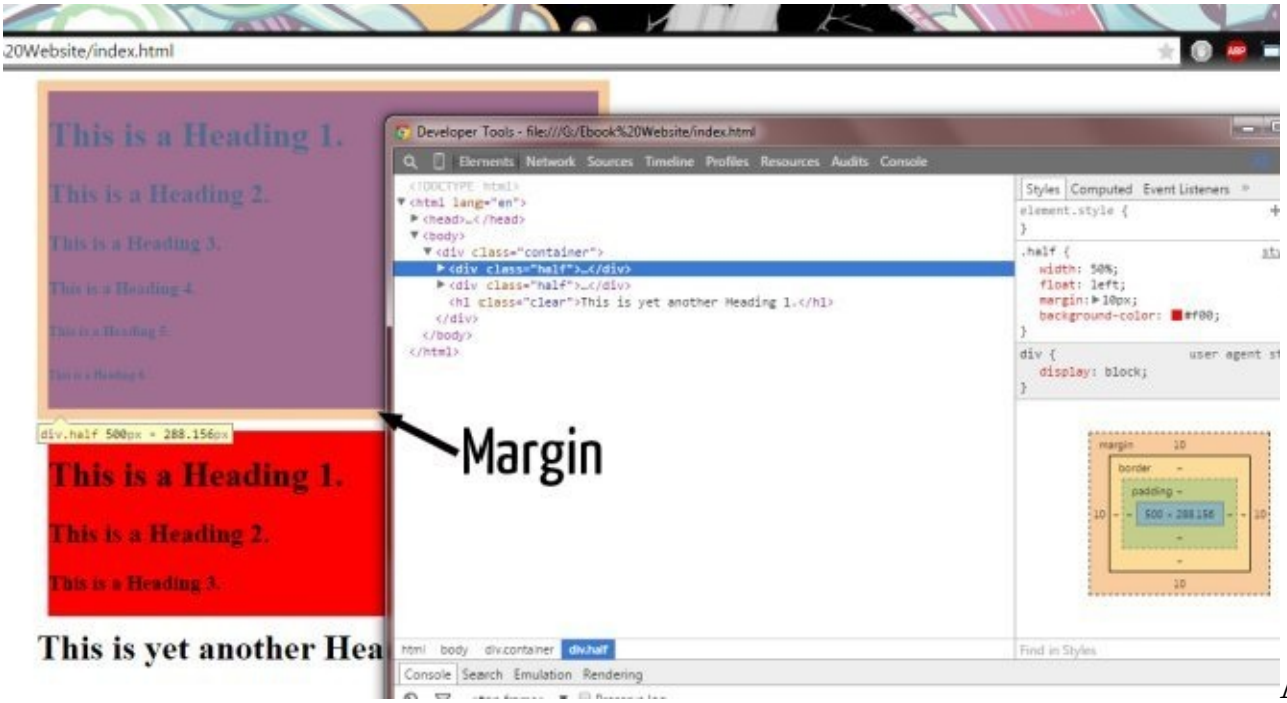
If you set a margin of 10 pixels to an object that means that no other HTML element can get within 10 pixels of that element.

While margins don't add to the actual size of the element, they do sort of add size to the coverage of the element.

For example, if you were to have a containing div which you specified to be 1,000 pixels wide. Then you placed two floating divs inside the container. Each of the two floating divs were said to have a width of 500 pixels. Then both divs would fit directly side by side with each other in the container.

However, if you were to give both of the floating divs a margin of 10 pixels, they would stack on top of each other because they wouldn't have enough room to fit side by side.

Figure 13.1 – These two divs are supposed to sit side by side inside of a 1,000 pixel wide container. Each div is 500 pixels wide. However, now because each div has a margin of 10 pixels around, they don't fit next to each other anymore.



Above is the same website from earlier. However instead of the containing div having a red background I gave each of the floating divs a red background, as well as a margin of 10 pixels around the entire border.

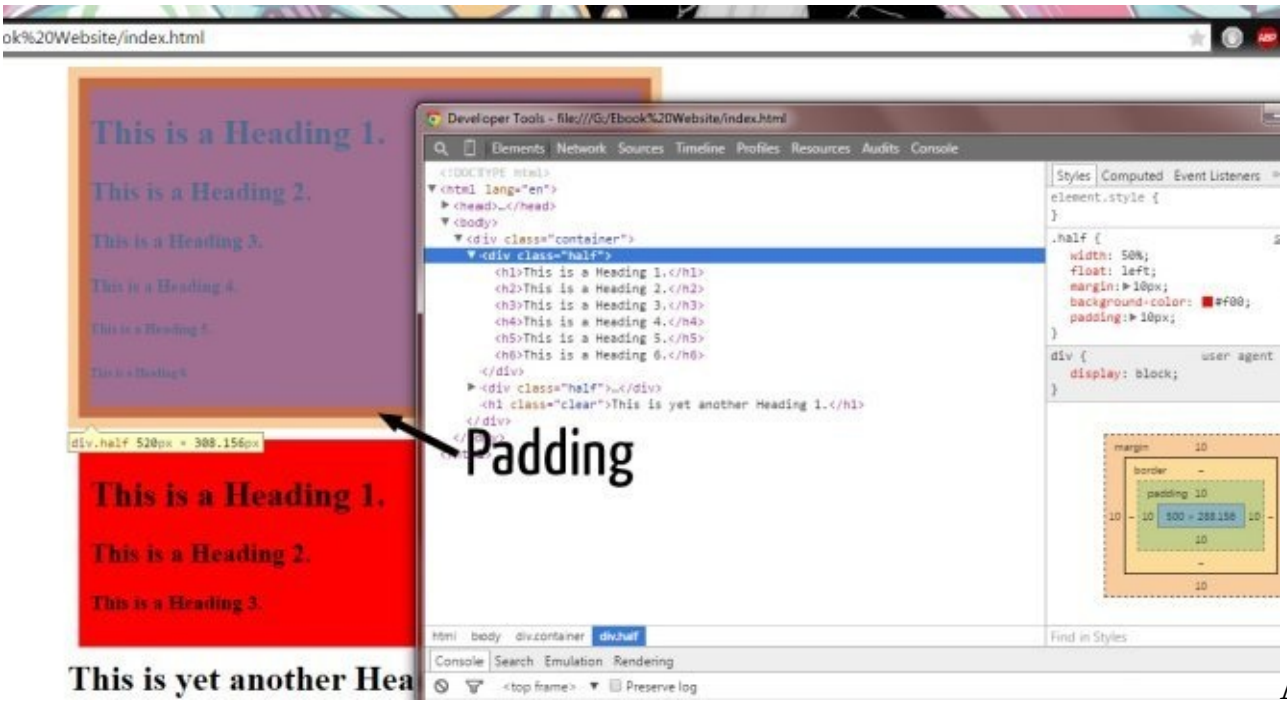
As you can see, even though the width I specified was 50% they don't fit side by side anymore because of the margins. And as you can see by the "inspect element" tool in Google Chrome, it illustrates the margin using that peach type color.

Padding

Padding is almost identical to margins, except padding works inside the element. If you were to specify a 10 pixel padding to an element, the element would grow 10 pixels on each side. Also, no element would be able to get within 10 pixels of the edge of the element.

In the above example you can see that the text inside each div actually touches the edge of the div itself. This isn't very attractive, so let's give those div's a padding of 10 pixels so we can see what happens when we do!

Figure 13.2 – I gave this div a padding of 10 pixels all around and clicked "inspect element".



As you can see the text is not touching the edge anymore. Google Chrome uses a green color to denote padding however you aren't able to tell in this example because it's being overpowered out by the red background of the divs.

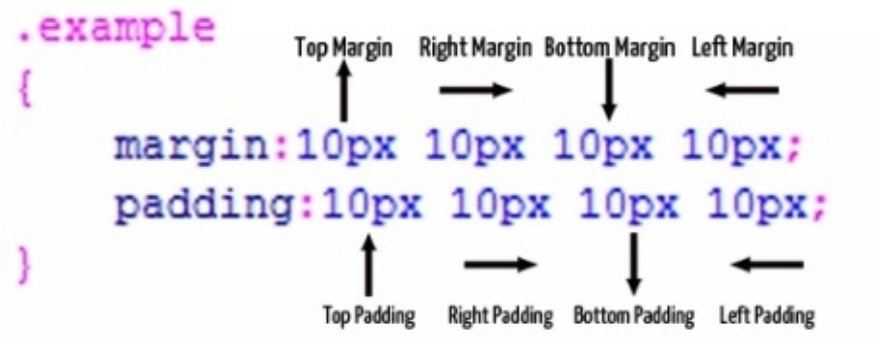
Specifying Margin & Padding

Specifying margins and padding both work the same way. Keep in mind you can specify your margin size in a variety of increments including px, pt, cm, etc.

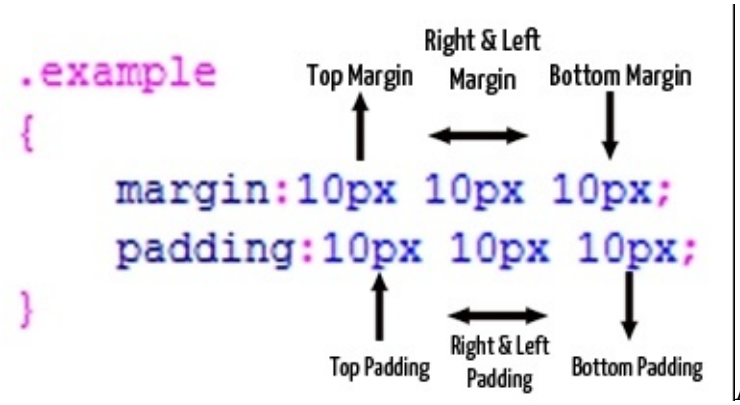
The simplest ways to specify margin and padding is as follows.

```
.example
{
    margin-top: 10px;
    margin-right: 10px;
    margin-bottom: 10px;
    margin-left: 10px;
    padding-top: 10px;
    padding-right: 10px;
    padding-bottom: 10px;
    padding-left: 10px;
}
```

You can go a little further and simplify it.



You can go even further to simplify it.



And of course you have the furthest simplification.

Negative Margins

There may come a time when you want to position an object and you find no better alternative. This is when you can use negative margins.

Before I even mention how or why you would want to use them I'll say that this isn't the most professional option. However, I have been known to use this "HTML cheat code" from time to time.

In all honesty, you probably won't find an excuse to pull this trick out of your hat until you become a little bit more advanced as a website designer.

And I can't think of an example that would be understandable to the person who is just getting acquainted with HTML/ CSS.

I've used a negative margin in the past when dealing with code that I'm using in my website that I haven't written. And for some reason I can't get a particular object to be where I want it to be using conventional means. That's when I'll begin to start messing with a negative margin in order to push an object to go where I want it to go.

Centering Objects With Margin

This is one of the most widely used HTML/CSS techniques. I personally use this in just about EVERY website I've ever done.

Website designers use this in order to center the website content in the middle of the page.

```
.container
{
    width:1000px;
    margin:0 auto;
}
```

As you can see, I've created a class called "container" and specified a margin of "0 auto". What this means is that on the top and bottom of this "container" there should be a margin of 0 pixels and on the left and right the margin should auto regulate.

This is a great technique however, don't go thinking that you can center ANY object using this technique.

In order for this technique to work:

- The object needs to be a block level element (or at least specified to display block using CSS)
- The object needs to have a width specified
- The object can't be floated
- The object can't have a fixed or absolute position (we'll cover that next)

Chapter 14: Other Types Of Positioning Techniques

In all honesty, I try to use floats to layout most of my websites. However, there are some exceptions that pop up from time to time.

Relative Positioning

This type of positioning is relative to the object's original position.

Basically after you've laid out some objects on the page. You may find that a particular object is not exactly where it needs to be.

Using relative positioning, you simply specify that the position of that element is "position: relative;" and then you are able to move it based on it's original position. You can say "top: 10px" or whatever the case may be using "top, bottom, right & left".

Absolute Positioning

Absolute positioning is absolute, however it is based on any parent elements. For instance if you want to position an object “absolutely” and that object was inside of a div. The position you specify will be relative to that div.

If the object is not wrapped in other elements it will be relative to the HTML page itself.

Absolutely positioned elements don’t have any effect on other elements. They won’t push any other elements out of the way and are basically taken out of the general “flow” of the document.

Fixed Positioning

Fixed positioning is relative to the browser window itself. In other words when you open your web browser, depending on your screen size, if the browser is minimized or full screen, etc.

As you scroll a fixed position element won't move. I've used this in the past for times when I wanted the background of the page to be a fixed image and I didn't want the image to scroll as the page scrolled.

I've also used this for a social media sharing box which was affixed to the side of the browser window.

Other uses you might have seen before include the arrow at the bottom right of the screen that pops up when you've scrolled a significant amount. (The button that allows you to quickly scroll up to the top of the page.)

Using Positioning

In order to use this type of positioning you simply have to specify the positioning type using CSS.

One line of CSS will be dedicated to:

position: (relative, fixed or absolute);

Followed by lines of code which will tell WHERE to position the element based on: top, right, left, and bottom. You are able to use pixels, points, Em's, etc. as your unit of measure.

Chapter 15: Fonts, Fonts, Fonts!

Alright, so if you have a background in graphic design you would know that a serif font is one with tiny pointy things at the tips of the letter. Serif fonts are great for print because the serifs seem to blend the words together and are said to make it easier for the reader.

However, on the internet, where everything is made up of pixels, serif fonts are not ideal. Not saying you shouldn't use them, however if you're going to use a serif font it's better to use them for only headers or special type elements rather than body text.

The reason for this is because pixels are square and when translating tiny squares into a serif font, the font's often get lost in translation. This is where the term "web safe fonts" comes into play.

How Declaring Fonts Works

So, keep in mind that Times New Roman (or equivalent) is the default font unless specified otherwise. This means that unless you want to use Times New Roman you'll definitely need to specify which font you'd like to use.

Fonts get specified by family. To specify which fonts you want to use you start off with the CSS declaration "font-family".

```
h1
{
    font-family:
}
```

- Verdana, Geneva, sans-serif
- Georgia, Times New Roman, Times, serif
- Courier New, Courier, monospace
- Arial, Helvetica, sans-serif
- Tahoma, Geneva, sans-serif
- Trebuchet MS, Arial, Helvetica, sans-serif
- Arial Black, Gadget, sans-serif
- Times New Roman, Times, serif
- Palatino Linotype, Book Antiqua, Palatino, serif
- Lucida Sans Unicode, Lucida Grande, sans-serif

As you can see my text editor is trying to provide me with a list of suggestions.

```
h1
{
    font-family: Arial, Helvetica, sans-serif;
}
```

First Choice — Second Choice — Last Resort

Above is the typical font declaration. You basically list your options separated by a colon. The first on the list is the prioritized font, followed by all of your second choices and finally either serif, sans-serif or cursive depending on the font family.

What this does is, no matter what fonts the person has on their computer you will have a greater chance of having the type be rendered the way you want it.

In the above example, if the person has Arial than the computer will render the font as Arial. If the person is using a Mac, then they probably don't have Arial and in that case the computer would render the font as Helvetica. And if all else fails, the computer would just use the default sans-serif font.

Most of the default/classic declarations only have 3 options, however you can specify as many "fall back" fonts as you'd like.

Other Font Methods

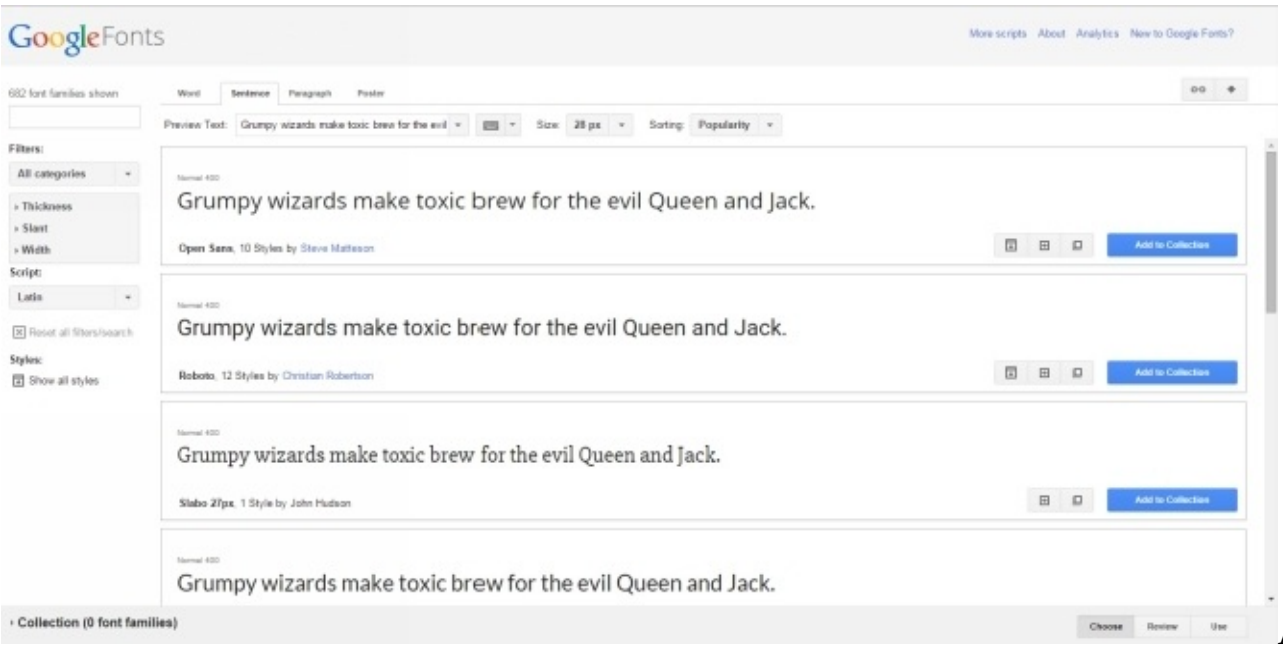
Obviously, not everyone will want to use “normal” fonts for their web design projects. Some people will want to use special fonts because it fits in with their design, or company branding, etc.

There are a few other options for creating websites using special fonts. Of which I’ll be covering the one that I use most often.

Google Fonts

Google has a library of web safe fonts listed at www.google.com/fonts where you can find a font that you like and use it on your web page.

Figure 15.1 – Below is the Google web font archive located at www.google.com/fonts.

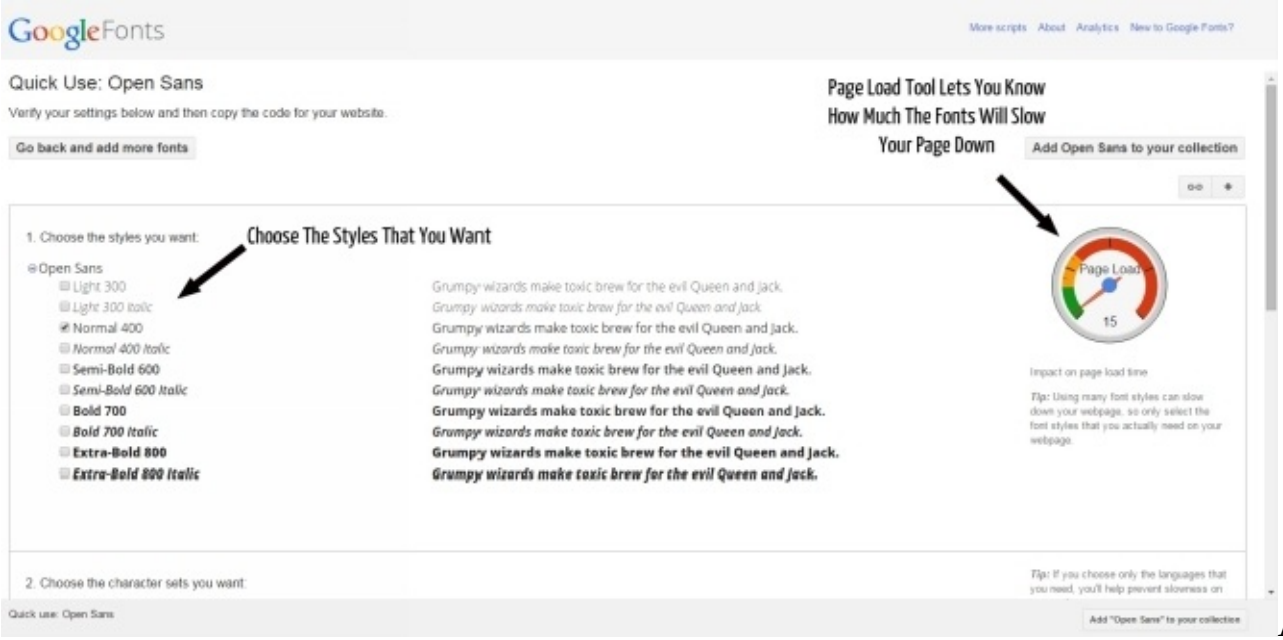


Above is a screen shot of Google’s font website. On here you can choose the font you want to use, as well as the weights of the font that you want to use. (The weight is the thickness of the font.)

I will warn you however, the more Google fonts as well as the more weights that you choose for your project the longer it will take your website to load.

In order to use Google fonts, simply choose the font you wish to use. Google will give you the link to a CSS stylesheet that you will have to link to in the header of your document. Then you will need to use the font-declaration that they give you whenever you want to use the font.

Figure 15.2 – I am able to use as many font weights as I wish. However on the right, the page load graphic will tell me how much load time the fonts will add to my site.



As you can see above once you've chosen the font, you can choose the font weights you wish to use. Then Google updates the Page Load Tool to let you know how much the fonts will effect page load time.

Once you're done, scroll down to see…

I'm pretty sure I've mentioned this but different browsers render the EXACT same font DIFFERENTLY. This is important to know because what looks great in one browser may look totally weird in another if the font is of a slightly different size.

Let's check out how the same font renders in Firefox, Google Chrome and Internet Explorer. Keep in mind that these fonts are supposedly "Google" fonts. I think you'll be surprised!

Google Chrome

This Is Open Sans, A Google Font.

Mozilla Firefox

This Is Open Sans, A Google Font.

Internet Explorer

This Is Open Sans, A Google Font.

Above you'll see Open Sans, rendered in Google Chrome, Firefox as well as Internet Explorer. As you can see each font appears slightly different. Each line of text is a different length even though each line contains the EXACT same characters.

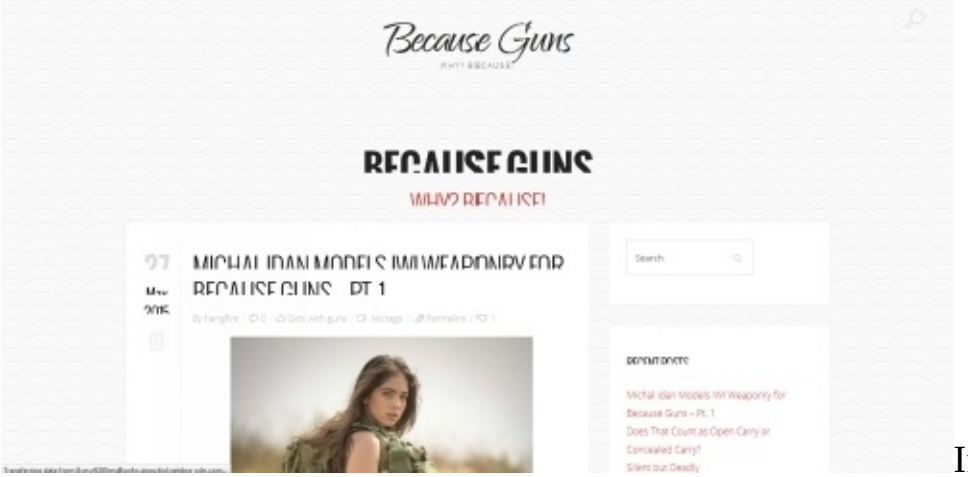
And the funniest thing…Google Chrome isn't the best at displaying it's own font. Internet Explorer, the worst browser of all time, is arguably the best at displaying fonts.

Google Font Disclaimer

This is another tip that I've learned through experience. Some Google fonts such as Oswald will display oddly if you have the font downloaded on your computer.

What I've found is that some Google fonts, Oswald in particular will display only the top half of the font when you are using a medium weight font.

Figure 15.3 – Below is a firearms blog which utilizes Oswald in a medium weight. Only the top half of the text is displayed.



If you happen to notice this, you can remedy this by using a different font weight. However, just realize that you are noticing this glitch because you have the font downloaded on your own computer and that most internet users will not have this problem…unless they are also web designers. As of today, I believe this issue happens mostly with Firefox.

Chapter 16: Semantic Code

Semantic code is basically the act of making sure that the code that you write is easily understandable to anyone who wants to work with it.

In the past (some people including myself still use this today), people would lay out their websites using divs and give the divs classes according to their function. The divs would be named header, nav, sidebar, footer, etc.

However, with the advent of HTML5 new div-“like” elements have emerged. These new HTML5 elements basically work exactly the same as a div however they are more semantic, as they are named after their function.

The new elements are named header, nav, aside, section, article, summary, details and footer. Because older versions of Internet Explorer don’t recognize these elements people often add the code directly beneath this sentence to the head of their document to alleviate this problem.

```
<!—[if lt IE 9]>

<script src=“http://html5shiv.googlecode.com/svn/trunk/html5.js”></script>

<![endif]—>
```

You may remember the above code from the HTML5 document I had you set up earlier.

Writing Semantic Code

When coding websites, I recommend that you always keep the concept of semantic code in the back of your mind.

You never know when a website that you have worked on in the past may need to be worked on by someone else.

In order to make sure that your websites are easily understandable make sure to:

- Use logical names when giving classes and id's to HTML elements
- Use the above mentioned HTML5 elements to mark the important sections of your document
- Use HTML as well as CSS comments whenever necessary in order to help people understand what's going on in the document

HTML & CSS Comments

HTML & CSS comments can help you to break up the clutter of messy HTML & CSS documents and help you to better understand the code.

```
<table width="800" border="0" cellpadding="0">
  <tr>
    <!--cell 1: navigational links-->
    <td width="100"> </td>
    <!--cell 2: main page content-->
    <td width="600">
        <table width="100%" border="0" cellpadding="0">
            <tr>
                <td> </td>
            </tr>
        </table>
    </td>
    <!--cell 3: side bar cell-->
    <td width="100"> </td>
  </tr>
</table>
```

Examples of comments in HTML

The format for an HTML comment is located in the picture below.Above you'll see an extremely old fashioned layout. It's an example of a website that was laid out using tables.

HTML techniques have come a long way since these days. However, the format for writing an HTML comment is still the same.

To write a comment in CSS the format is illustrated in the picture below.

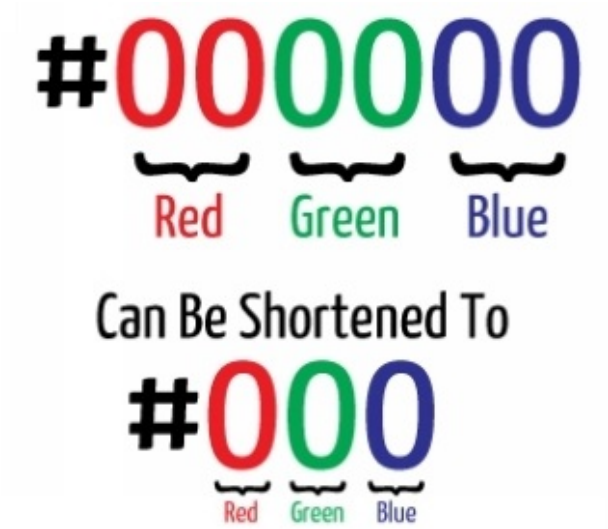
File contents for Styles/styles.css

```
/* Global Reset */
* {
    margin: 0;
    padding: 0;
}

/* Here is a comment that will not be visible in the browser */
/* This template css file was created by John Smith on 2/3/2008, etc. */

body {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 0.8em;
    height: 100%;
}
input, select {
    font-size: 12px;
```

The CSS comments are highlighted in the above picture. I think CSS comments are great when you have a specific section of your website that a large chunk of CSS styles correspond to. You can place a CSS comment at the beginning and end of the chunk of code to let others know what the styles were meant for.

Chapter 17: Using Color

In most cases when you need to specify a color, you will be using a hexadecimal system. You can see the format for specifying these colors below.



Hexadecimal colors, are based off of 6 numbers/ letters as the name suggests. The characters used in hexadecimal colors range from a-z and 0-9. Smaller numbers indicate darker colors while larger numbers indicate brighter colors.

This is because computer colors are based off an additive color system. That is, a system where color is made by adding light. The more light added of each color the brighter the color will be.

In the image above you can see the color code for black. As you can see there is 0 red, 0 green and 0 blue which results in a black screen. Whereas full red, full green and full blue would result in white.

Also in the image above you can see that a 6 character color code can be converted to a 3 character color code if it is made up of 3 groups of matching characters. For example #330011 can be shortened to #301, #999999 can be shortened to #999 and #FFFFFF can be shortened to #FFF. However, #030303 or #996500 cannot be shortened.

Essentially numbers 0-9 indicate levels of color from low to high. Equal amounts of red, green and blue create either white, black or a shade of gray. #000000 indicates a color of black because it uses equal parts (zero) red, green and blue whereas #FFFFFF indicates white because it is equal parts (F possibly stands for “full” in this case) red, green and blue.

Also, you can get pure red, green or blue by specifying them to be full while the other two colors are specified to be zero. For example, to specify red you could say #FF0000 (or #F00) which means full red, 0 green and 0 blue. Obviously, there are many more hues of red, green and blue available however, this is for when you want to use the purest form of each.

Hexadecimal Color Usage

One of the most common uses of this hexadecimal color code is when you are specifying a font color. You can do this by using the CSS declaration (color: #000000;) where #000000 is the color of your choice.

Other uses for specifying color can come in handy when specifying background color (background-color: #000000). As well as when specifying the color of a border (border: 1px solid #000000;)

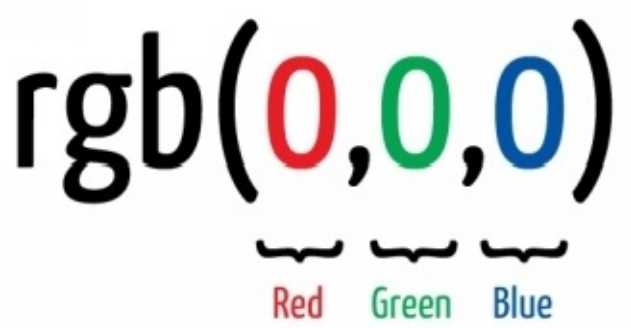
RGB Color Code

Sometimes, you will need to specify color using a RGB color system. These instances occur much less frequently than the hexadecimal system.

Instances where you would use this happen mostly with new CSS3 declarations such as when specifying opacity, you need to use the RGBa code followed by the opacity using a decimal system. Or when specifying a CSS3 gradient.

I won’t get into specifics on when to use them because you can easily find out how to use them when the time arises using a quick Google search.

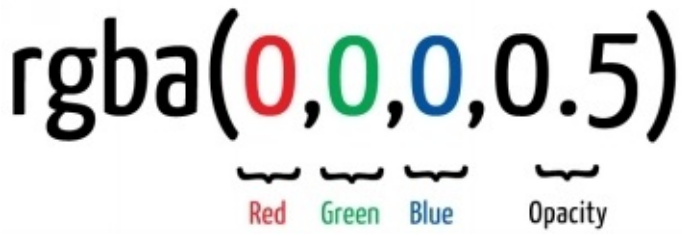
The way to specify RGB colors are as follows:



Each number can be any value from 0-255. 0 being none and 255 being full. Same rules apply as the hexadecimal. For example rgb(255,0,0) is pure red and rgb(25,25,25) is some light shade of gray.

Opacity: RGBA Color Code

With HTML5 and CSS3 we now have options for declaring opacity. RGBA is essentially the same declaration as RBG however there is one more object that gets declared and that's the opacity. The opacity level goes from 0.0 – 0.1. 0.0 being 0 percent and 1.0 being 100 percent. The format looks like this.



Specifying Color By Name

Recently, you have been able to specify colors by name. However, this is fairly new to CSS3 and I haven't gotten used to using it, nor have I used it.

Correction, I use it only when specifying a color as "transparent" (AKA no-color).

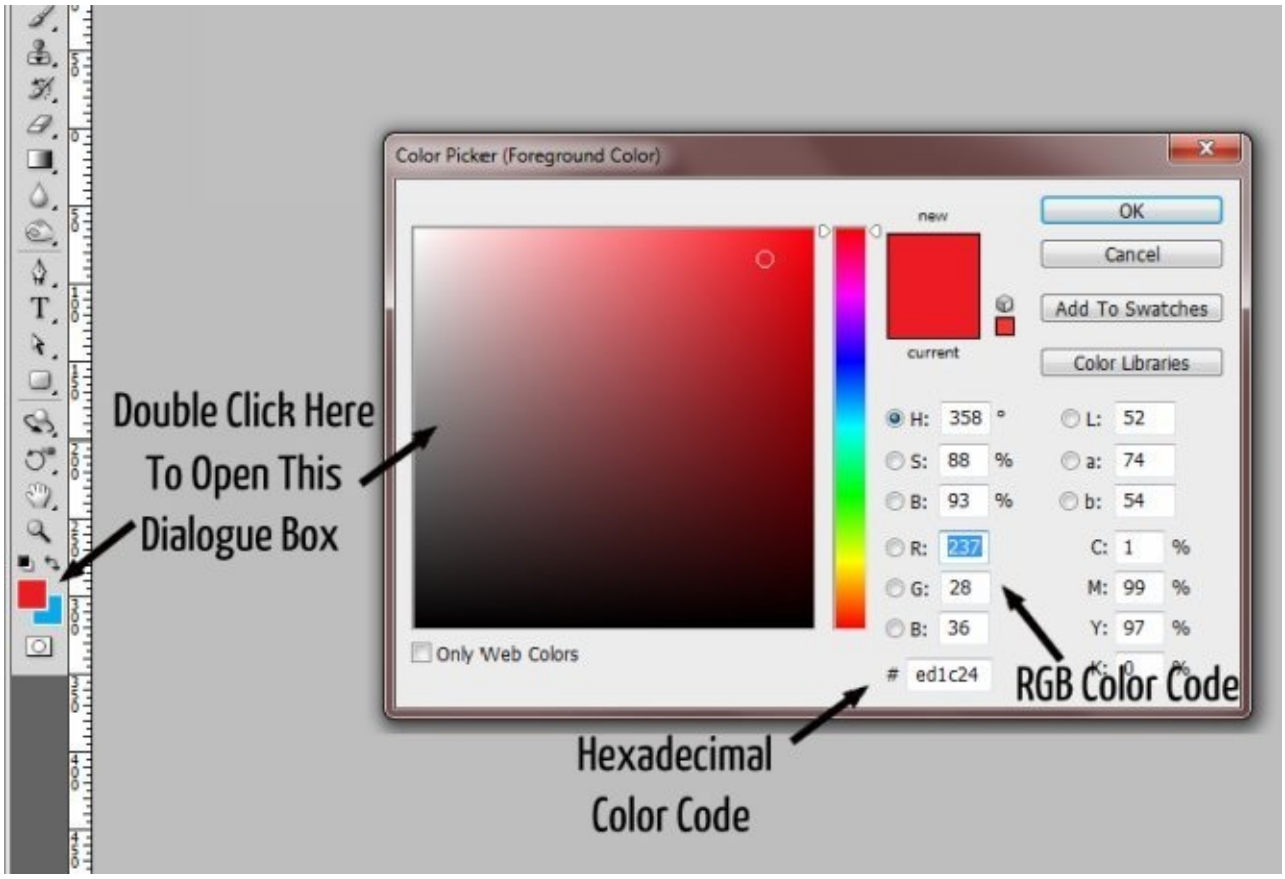
Check it out and get comfortable with it if you wish! As I mentioned earlier, there's more than one way to skin a cat when it comes to HTML/CSS!

Using The Correct Color From Your Mockup

After you've created your mockup in Photoshop, you're going to want to get the correct colors to specify in your CSS. Luckily, this is very easy to do.

Simply use the eye-dropper tool to grab the color then double click on the foreground color which pops up (this should be the color you chose). The resulting dialogue window should have both the hexadecimal color code as well as the levels of red, green & blue to use for rgb color codes.

Figure 17.1 – In Photoshop, you can use the Color Picker window to figure out what hexidecimal or RGB color codes to specify in your CSS document.


Color Picker (Foreground Color)
new
current
OK
Cancel
Add To Swatches
Color Libraries
H: 358 °
S: 88 %
B: 93 %
R: 237
G: 28
B: 36
L: 52
a: 74
b: 54
C: 1 %
M: 99 %
Y: 97 %
K: 0 %
ed1c24
Only Web Colors
Double Click Here
To Open This
Dialogue Box
RGB Color Code
Hexadecimal
Color Code

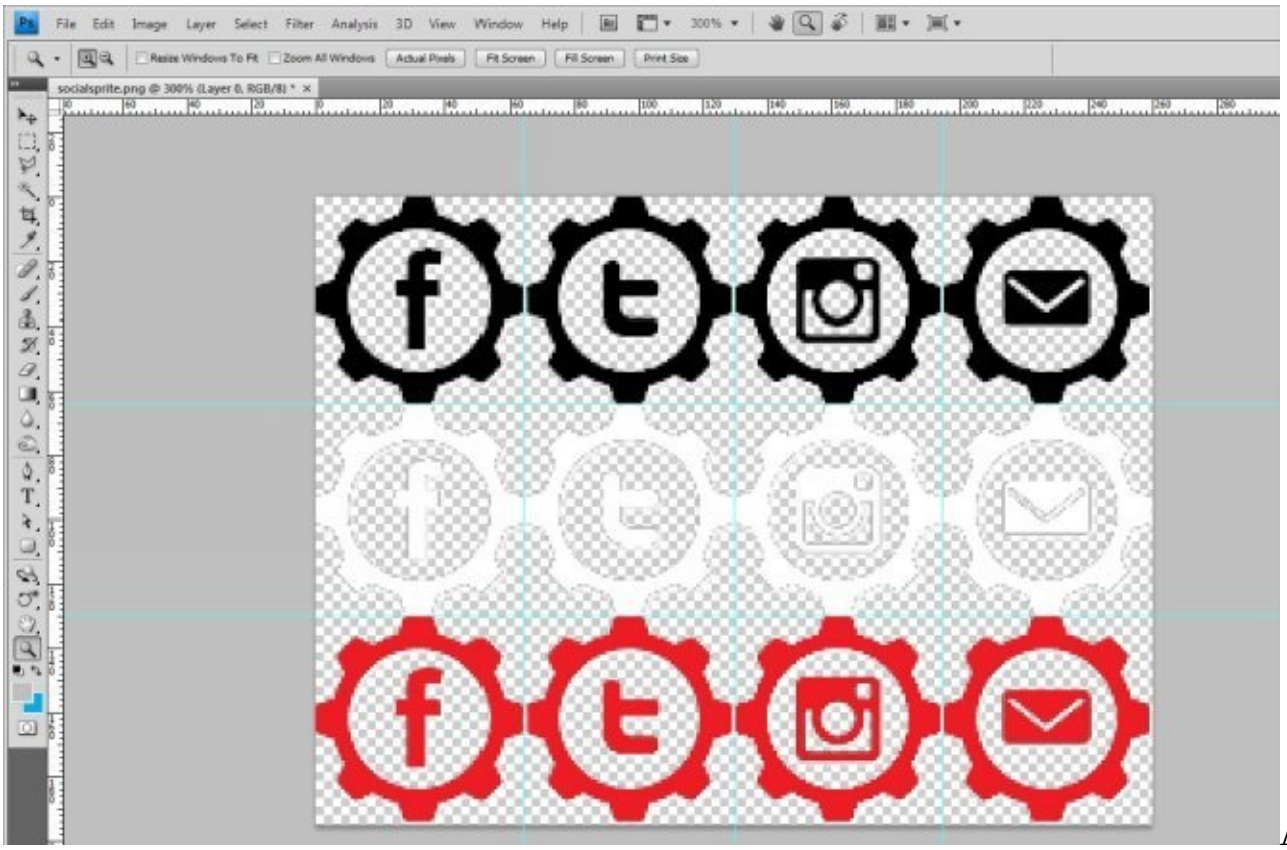
Chapter 18: CSS Sprites

The more images you use as well as the size of the images makes for a slow loading website.

In order to combat this phenomena, some people utilize a concept known as CSS sprites. A CSS sprite is when you have multiple images in one image. So rather than loading a bunch of tiny images the browser has to load 1 slightly larger image which results in faster load time. Normally these sprites are PNG format because PNG's support superior transparency capabilities.

I love to use CSS sprites when dealing with social media icons.

Figure 18.1 – Below is an image that I setup to be used as a CSS sprite image. Notice the rulers are visable and the guides are being used. This makes it easier for me to use actual coordinates in my CSS declarations.



Above you'll see an example of a CSS sprite image that I created. I created black and white versions of the same social media icons as well as red versions which would be used when the user hovers over the icons.

As you'll notice I made use of the Photoshop guides and the ruler is set to pixels. This is important because these coordinates will be of use to us later.

Using CSS Sprites

To use a CSS sprite you need to specify how tall and wide the image will be then you must specify the background image plus the coordinates (from the left as well as top of the image).

```
a.social
{
    height:65px;
    width:65px;
    display:block;
    float:left;
    margin:0 5px 10px 0;
}
```

As you can see above, I gave the links which will be my social media CSS sprites a class of “social”. Then I proceeded to give each a height and width of 65 pixels. I said for the links to display block because otherwise a link would be an inline element.

Each link will float left so that they will line up next to each other left to right, and then I specified a margin.

Then each individual symbol needed to specify the background as well as the coordinates.

```
.fbblack
{
    background:url(../images/socialsprite.png) 0 0;
}
```

Above you’ll see the CSS for the black Facebook icon. The “0 0” means that the image begins 0 pixels from the left as well as 0 pixels from the top.

```
.twblack
{
    background:url(../images/socialsprite.png) -65px 0;
}
```

Above you’ll see the CSS for the black Twitter icon. The “-65px 0” means that the image begins 65 pixels from the left and 0 from the top.

And finally I specified a “hover state” for each of the icons. I’ll show you how I did that and we’ll talk about element states next!

```
.fbblack:hover, .fbwhite:hover
{
    background:url(../images/socialsprite.png) 0 -130px;
```

}

Above is the hover state for the black Facebook icon as well as the white Facebook icon. I made the graphic so that both white and black social media icons would have a red hover state, which is why both of these use the same background image and CSS sprite coordinates.

As you can see above, the red hover state is 0 pixels from the left and 130 pixels from the top.

This topic may be confusing to some so don’t be afraid to do a little bit of your own research as well as experimentation to learn more. I mostly wanted to tell you about this technique so you’d have it as part of your arsenal.

Chapter 19: Element States

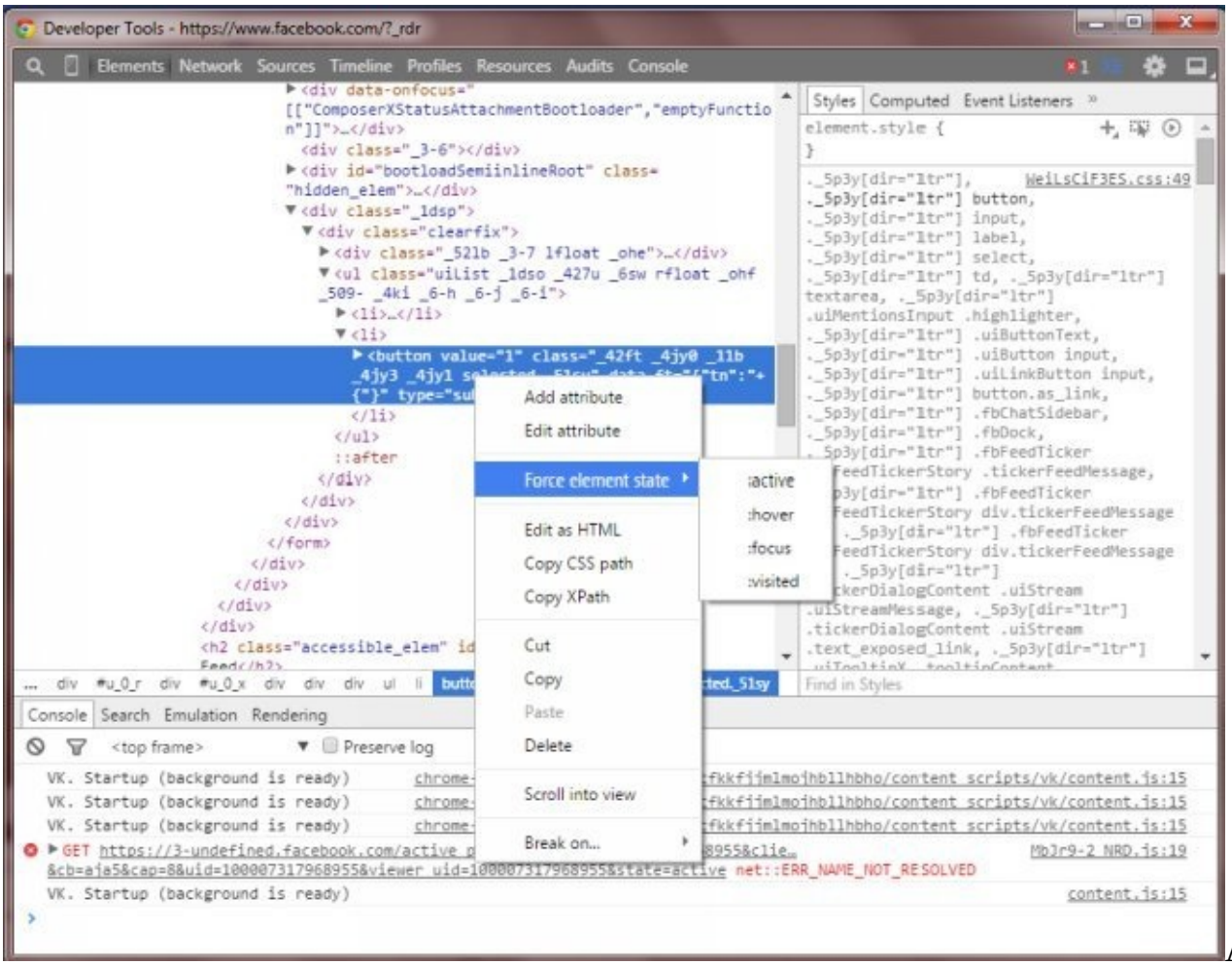
Interactive elements such as links and inputs (search bars, text boxes, etc) have what are known as different “states”.

Because of this, you are able to style the “state” of each element using CSS.

Before I confuse you any further the states I’m referring to are:

- Active
- Hover
- Focus
- Visited

Figure 19.1 – When you write click an element inside the Developer Tools window in Google Chrome you can “force an element state” that is, cause an element to behave as if it is displaying a state of your choice. This is useful when trying to style a specific element state.



Above is a screenshot of the Google Chrome developer tools and this is another reason why I love Chrome so much for coding websites. After you right click and “inspect element” on an object. You can right click in the developer tools window and go to “force element state” followed by the state you want to see.

This is great for testing your CSS, because as I mentioned before you can style all of these element states using CSS.

Active

Active refers to an active link.

Hover

Hover refers to a link or object that is being hovered over with the cursor.

Focus

Focus refers to an input element which is currently active. For example when you're filling out a form and you hit the tab button. The next form field to get the blinking cursor in it is the field that has "focus".

Visited

Visited refers to a link that you previously visited.

Styling Element States

In order to style these element states you simply need to target the HTML element using either a class, id or other technique followed by a colon and the element state.

An example of this would be from the CSS sprite chapter where I specified the portion of the image to display for a regular social media icon versus a social media icon the user is hovered over.

Chapter 20: Handy Things To Consider

There are a few CSS styles which make it into every stylesheet that I write for a number of reasons. We'll go over those now.

Handy Internet Explorer Style

When you wrap an image with a link, Firefox and Google Chrome don't have an issue with this. However, for some reason, older versions of Internet Explorer absolutely LOVE to place an ugly blue border around all images wrapped in links.

Figure 20.1 – Older versions of Internet Explorer used to place an ugly blue border around any images that are wrapped in links. Using developer tools in Internet Explorer I was able to display the page as if it was rendered in an older version of Internet Explorer. The red circle is around the area in the developer tools window in Internet Explorer where you can manipulate the display to reflect older browser versions.

Apparently, they fixed this issue in newer versions of Internet Explorer. However when you hit F12 you can open the developer tools window. I used the window to show what the website would look like using Internet Explorer 8.

As you can see there is a blue border around the image. In order to combat this I add the following style to EVERY stylesheet I write.

```
a img
{
    border:0 none transparent;
}
```

It targets all img (images) wrapped in an a (link). In case you don't understand the CSS declaration, the standard protocol for defining a border is "border: (width in pixels) (style:solid, dotted, dashed) (color);". So I specified a border 0 pixels wide, no style and the color to be "transparent".

Global Resets

So as I mentioned earlier, each HTML element has a default style attached to it. Different browsers have a bit of leeway when deciding how each element should be displayed.

For this reason many people link to a CSS stylesheet known as a “reset”. This stylesheet basically styles each element in a way which is uniform across all modern browsers.

I don’t feel like this is necessary. However I do use the following style in all my stylesheets.

```
html, body
{
    margin:0;
    padding:0;
}
```

Some people like to add “*” along with html and body. The asterisk is what’s known as a “wild-cat” and basically targets EVERY element. However, I feel that just denoting html and the body element is good enough.

Default Link Hover Styles

When you begin to get fancy with your link styling. Don’t forget to style the “hover” state.

Oftentimes you’ll finish making a link look like a super cool button and then you’ll go to test it out and say…”oh crap, when I hover over this button the text turns blue and get’s an underline”.

To combat this make sure you specify a font color for the hover state as well as specify “text-decoration: none;” if you don’t want there to be an underline.

Default Font

By default, the font will be Times New Roman (or Mac equivalent). To get around this you can specify the default font of your choice at the top of your stylesheet by targeting the “body” with the font-family of your choice.

```
body
{
    font-family: Arial, Helvetica, Sans-serif;
}
```

Chapter 21: Flash, Javascript & CSS Animations

In the past, the only way to utilize simple and even some complex animations was to use either Flash or Javascript. However, with the advent of CSS3, now animations can be done using CSS alone!

This is great news because Apple products hate flash and some browsers (especially Internet Explorer) tend to block Javascript.

Keep in mind however that these animations may not have support in older Internet Explorer browsers. However, what else is new. Internet Explorer tends to not work with a lot of cool HTML/CSS features.

I won’t go into extreme details on how to use CSS animations. However, just keep it in the back of your mind that it is possible to animate the color, position, etc of elements using CSS exclusively.

You can even set time delays, animation speed, and other aspects of the animation which were never possible using CSS!

Chapter 22: Validation & Troubleshooting

There comes a time in every web designers life when they have absolutely no idea why their code isn’t working.

Luckily I have a few different solutions for you.

Developer Tools

One of the first things that I do when faced with problem is to use developer tools in the browser that I'm viewing the website in.

Pretty much every modern browser is equipped with developer tools to help you figure out your code.

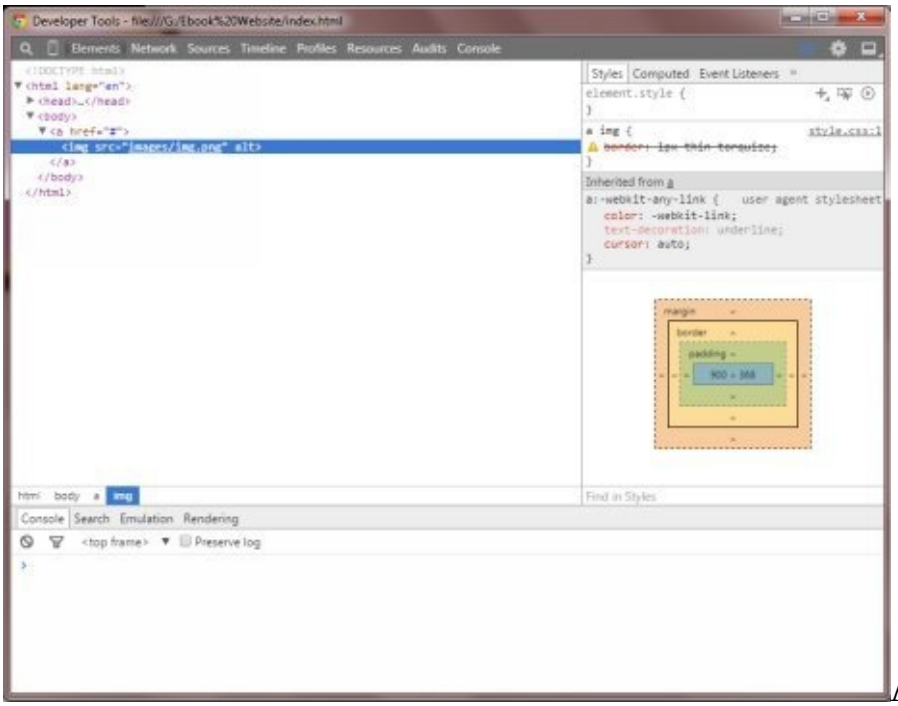
Google Chrome happens to be my favorite because you are able to edit the HTML and CSS directly in the developer tools window and preview the changes before you commit them to your document.

If you notice that a certain CSS style isn't working correctly, check your developer tools and see if there is something wrong.

Google Chrome's developer tools window will put a slash through any CSS style that either:

- Is written incorrectly
- Is meant for a different browser (some CSS styles aren't global yet so they have different prefixes for Mozilla and WebKit based browsers) Google Chrome will put a slash through all Mozilla based styles

Figure 22.1 – The Developer Tools window is telling me that the style is not taking effect for some reason, the caution symbol next to the style lets me know it's invalid.



Above you can see that there is a little caution symbol and a slash through the incorrectly defined style. I tried to define a style for links wrapped in images. I gave them a border 1 pixel wide, with a thin style then I purposely spelled turquoise wrong so the style would be invalid.

Developer tools will also let you know if you forgot to link a file, if the file path is invalid, etc.

W3C Validator

The second way to diagnose your HTML/CSS is to validate it. W3C stands for World Wide Web Consortium and all HTML has standards.

When you are having a particularly difficult time with your code it helps to validate the code and see if it throws any errors. Sometimes your code isn't behaving properly because it is invalid.

Simply go to http://validator.w3.org (or Google W3C validator) and there are a few different ways to submit your code.
You can either submit the entire url, you can submit by file upload or you can simply copy your entire HTML document and paste it into the textbox.

Once W3C validator checks your code they'll let you know if you have a missing tag, if you have a mismatched tag or if you did anything else that is against protocol.

Chapter 23: Minified HTML/CSS

Up until this point, I've given you a ton of information on writing code as well as the ability to inspect other peoples code using web browser developer tools. I felt that it may be necessary to include a little tid-bit on minified HTML/CSS.

As I mentioned earlier, the formatting of your HTML/CSS is mostly for you and other developers that may touch your code. Neatly formatting your code makes it easier to read, understand and edit.

However, this neat formatting is for humans only. The browser is a machine and that properly formatted code amounts for little to the browser except pesky white space surrounding the code that it is trying to read.

For this reason, many people tend to minify their HTML and CSS. What this means is they format their code without unnecessary white space.

Figure 23.1 – Below is minified CSS. Minified CSS takes less time for the browser to load.

```
1  body{text-align:left;margin:0;padding:0;color:#555;line-height:1.4em;letter-spacing:.2px;bac
   ,arial,helvetica,sans-serif}.limit{margin:0 auto;max-width:1000px}#wrap{width:1000px;clear:b
   }#header{clear:both;margin:0 auto;padding:0;background:url(http://johneengle.com/images/back
   }.space{margin-bottom:10px}.more-space{margin-bottom:17px}#logo img{max-width:100%;height:au
   !important;font-family:arial,sans-serif!important;line-height:18px!important;outline-width:0
   relative!important;padding:5px 0 0 0!important;height:33px!important;overflow:hidden!importa
   text-align:center!important;text-shadow:none!important;color:#fff;background:url(http://john
   display:inline!important;margin:0!important}.emailbar-form input{text-transform:none!importa
   !important;font-size:15px!important;font-weight:normal!important;width:auto!important;font-f
   ;text-shadow:none!important;-moz-border-radius:5px!important;-webkit-border-radius:5px!impor
   background:#fff!important;color:#999!important;border:1px solid #999!important;outline-style
   -moz-border-radius:5px!important;padding:4x 7px!important;width:auto!important}#emailbar-sub
   url(http://johneengle.com/images/emailbar-submit-button.png) repeat-x;border:0!important;bor
   !important;display:inline!important;height:auto!important;color:#fff;font-weight:bold}.fixed
   margin:0 auto;max-width:980px;clear:both;border:1px solid #e5e5e5;border-width:0 0 1px 0;bac
   list-style:none;line-height:1;margin:0}#catnav ul a{position:relative;display:block;z-index:
   border-right:1px solid #e3e3e3;float:left;width:auto}#catnav ul li.search{float:right;paddin
   ul,#catnav ul li.sfhover ul,#catnav ul li li:hover ul,#catnav ul li li.sfhover ul{left:auto}
   100%;float:none;padding:0;margin:0}.singlepost .entry p a{font-weight:bold}.sitemap .entry{m
   }#contentright{font-size:14px;width:30.33%;float:right;padding:0;margin:0}#sidebar{width:100
   block;overflow:hidden;padding:0}#sidebar-bottom-left{width:54.375%;float:left;padding:0;marg
   overflow:hidden}#footer{clear:both;margin:0 auto;padding:30px 0;background:#000;font-size:14
   display:block;height:0;clear:both;visibility:hidden}.clearfix,blockquote{display:inline-bloc
   h1,h2,h3,h4,h5,h6,h7,#sitetitle .title{letter-spacing:0;margin:0;padding:0;text-align:left;f
   font-size:19pt}h4{font-size:17pt}.entry h1,.entry h2,.entry h3,.entry h4{margin-bottom:15px}
```

Above you'll see an image of some minified CSS. As you can tell it's very difficult to read for human eyes however, for an internet browser this minified CSS can save load time over a non-minified stylesheet.

As you become more advanced in your coding skills you may choose to minify your code. There are plenty of CSS and HTML minifying code generators that make it easy by doing all the minifying for you.

I personally don't minify my HTML however I use http://www.cssminifier.com to minify my CSS stylesheets.

When minifying code, make sure to keep a non-minified version on file just in case any mishaps occur.

Chapter 24: Grid Systems

I wanted to briefly touch on grid systems because when I first started out I used to love this grid system known as the 960 grid system.

My very first websites were all 1,000 pixels wide and I used to have to use my best judgment to come up with widths for items in my layouts. For example if I was doing a two column layout, I would probably use an 700 pixel wide content area and a 300 pixel wide side bar.

Then I found the 960 grid system. The idea behind it was that 960 pixels was the perfect balance between older computer monitors which normally maxed out at around 1,000 pixels wide and newer computers with full resolution screens.

Having a container of 960 pixels showed up well on almost all monitors and the number 960 is divisible by more numbers than say 1,000 for instance. That way you could have, up to 12 equally sized columns.

Why Grid Systems?

The reason why I'm mentioning grid systems is for 2 reasons:

1. Grid systems make it easier for the website designer for a number of reasons.
- They normally come with a Photoshop document based on the dimensions of the grid so that you can lay out websites with ease and not have to worry as much about the dimensions because they're already accounted for.
- They save time because the website designer doesn't have to spend as much time thinking of how wide particular objects are going to be, thinking of class names to name divs of varying sizes, or doing math to get correct dimensions.
2. Grid systems paved the way for Bootstrap. Bootstrap is a responsive grid system that is used by many web designers today, including me!

I will briefly mention that after the popularity of the 960 grid system died off, a new grid system emerged for a moment. This new grid system was based off a 1,200 pixel wide container. I believe the reasoning behind this was the fact that computer monitors were becoming wider and older computer monitors were being phased out.

If you want to check out the 960 grid system simply visit http://960.gs.

Chapter 25: Responsive Web Design

Once you have a solid grasp on general website design, responsive website design will be easy for you to pick up.

I recommend messing around with just pages which don’t respond to the width of the browser window before you try to incorporate any responsive web design techniques in your arsenal however.

But, I want you to get the most out of this e-book so we’ll touch on a few of the cornerstones of responsive web design.

The Viewport

One difference between static and responsive website design is the fact that with responsive design you need to designate that the width of the screen should be the width of the website.

You do this by adding a declaration to the head of the document. Also in this declaration you can specify if the user can zoom in as well as how much they can zoom.

<meta name="viewport" content="width=device-width, initial-scale=1">

Above is an example of one of these declarations.

Specifying Widths

In static website design, when we create a 2 column layout, we think of how wide we want our columns to be in pixels and create them.

However, when it comes to responsive design, static widths don’t cut it.

When you want the widths of objects to change with the width of the browser window it’s imperative to designate widths in percentages.

This way your website stays in proportion whether the device is 1,200 pixels wide or 350 pixels wide.

Responsive Images

The width of your images is also important because as the screen gets smaller your images obviously can't be full resolution or else your end user will have to scroll on their phone to see the entire picture.

To alleviate this you need to specify a width or maximum width of 100% or less. The height can be auto regulating by simply specifying that the width be "auto". That is, an automatic height relative to the current width.

Responsive Type

Obviously when your screen is wide all Heading 1 elements can be large, however when you view the same website on a smaller screen that heading will most likely take up the entire screen.

Since this isn’t the most aesthetic way to build a website, some people like to specify their font sizes in em’s rather than pixels.

I for one don’t like to do this so I use the next topic to alleviate this.

Responsive Navigation

Obviously when your screen reaches a point below 1,000 pixels you can’t have a wide navigation bar anymore. At some point the width of the browser will be too narrow to support your navigation bar.

However, don’t fear because there are a variety of responsive navigation solutions to choose from. All you have to do is Google it!

Break Points & Media Queries

These next two topics go hand in hand.

I will also mention that if you're going to be doing responsive web design you NEED to download a resolution plugin. That is, a plugin that tells you the width of the browser window at any given time.

The reason for this is when you're dealing with a responsive website, as you make the browser window smaller you need to see exactly WHEN the website begins to look weird.

Then you can say with authority, "ok when the website is less than 1,000 pixels wide this object get's knocked out of place." This is what's known as a "break point".

You can then specify a "media query" that says, when the browser window is less than 1,000 pixels width than do *this*. Obviously *this* can be anything you want it to be because you will be specifying your will using CSS.

The way media queries work is you can either say: when the browser is below this width do *this*, when the browser is above this width do *this* or when the browser is between this width and this width do *this.*

When the browser is below this width do this.

```
@media only screen and (max-width : 320px) {
/* Styles Go In Here */
}
```

When the browser is above this width do this.

```
@media only screen and (min-width : 321px) {
/* Styles Go In Here */
}
```

When the browser is below this width and above this width do this.

```
@media only screen and (min-device-width : 768px) and (max-device-width : 1024px){
/* Styles Go In Heret */
}
```

Something To Remember

Remember that media queries should go at the end of your CSS document because when it comes to CSS the things at the bottom hold more precedence than the things at the top of the document.

As the browser window get's smaller the media queries that pertain to that browser width

will begin to take effect and the CSS styles that pertain to wider screens fall back.

If you encounter a particularly difficult CSS style that doesn’t take effect and you’ve tried everything and done your “developer tools” research, than you can add “**!important**” just before the semi-colon of the CSS style declaration. “**!important**” gives the CSS style precedence over any other inherited styles.

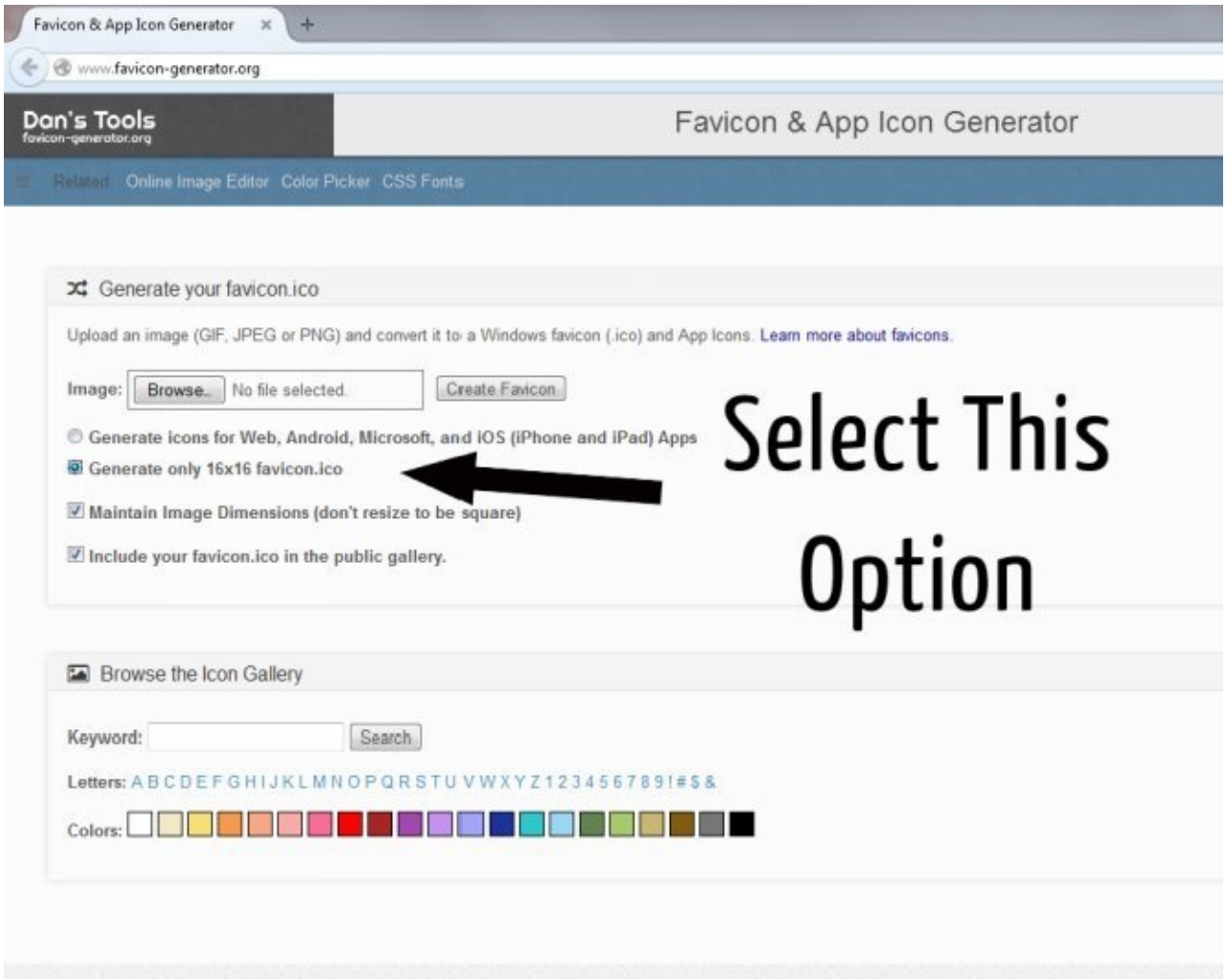
Chapter 26: The Favicon

The icing on the cake to your website is the favicon. That is the little 16x16 pixel image that is associated with your website and gets saved with the title of your website when someone adds your site to their favorites.

To create a favicon, simply create a 16x16 pixel document in Photoshop or the image editing software of your choice. Create an icon to represent your website. Since you will need to save the file as a .png, your favicon can be transparent.

I then use http://favicon-generator.org/ to generate my favicon. I save the resulting file in the root folder of my website. The favicon extension is .ico and the favicon generator website also gives you the code that you need to attach your favicon to your site. That code goes into the head of your documents.

Figure 26.1 – This website changed since I first started using it. However, when creating your favicon, make sure to select the "Generate only 16x16 favicon.ico". This saves you from downloading unnecessary files that you don't need.


Favicon & App Icon Generator
www.favicon-generator.org
Don's Tools
favicon-generator.org
Favicon & App Icon Generator
Related: Online Image Editor Color Picker CSS Fonts
Generate your favicon.ico
Upload an image (GIF, JPEG or PNG) and convert it to a Windows favicon (.ico) and App Icons. Learn more about favicons.
Image: Browse... No file selected. Create Favicon
Generate icons for Web, Android, Microsoft, and iOS (iPhone and iPad) Apps
Generate only 16x16 favicon.ico
Maintain Image Dimensions (don't resize to be square)
Include your favicon.ico in the public gallery.
Select This Option
Browse the Icon Gallery
Keyword: Search
Letters: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 1 2 3 4 5 6 7 8 9 ! # $ &
Colors:

Conclusion & Next Steps

I hope you've enjoyed my e-book. I sincerely hope that I've given you a solid foundation of knowledge to build upon.

Going forward, there are some amazing resources online for learning about HTML, CSS as well as Javascript.

One of my favorite websites, where I learned a majority of my coding knowledge was a website called www.w3cschools.com. There you'll find easy tutorials about pretty much anything you could want to know related to code. Including your next steps which will be to get further acquainted with HTML & CSS as well as learn about Bootstrap (widely used responsive grid system) and responsive web design.

When you are in a bind, don't forget to use your Google skills.

There you'll find resources from a few of my other favorite websites. One of which is called www.stackoverflow.com. This website is basically a forum of experienced code ninjas answering each others questions pertaining to code.

Another one of my favorite websites is www.css-tricks.com. There you'll find tons of great resources related to HTML & CSS.

Before I conclude this book. I'm going to give you one of the most important tips in this entire book. This is one of the tips which makes me such a fast coder.

Every time you Google an HTML element or CSS style, if you find yourself looking up the same stuff over and over. After a while, try to commit it to memory. The more stuff you commit to memory the faster you will be because you can spend less time on the internet trying to find out how to write code and more time actually writing code.

Thank you so much for reading!