Introducing HTML

Less than twenty years after its invention, the World Wide Web has become ubiquitous. Today, there are tens of billions of Web sites worldwide, and the number continues to grow at an almost exponential rate. There are many factors that have contributed to the growth of the Web, but one of the most important is that Hypertext Markup Language (HTML), the language in which Web pages are written, is easy to learn, and best of all, free.

History of HTML

The World Wide Web was invented in 1990 by Tim Berners-Lee. Berners-Lee worked at the European Particle Physics Laboratory (CERN) in Geneva, and at the time, no one operating system had come to dominate the market. Therefore, the visiting scientists at CERN were bringing in their own computer systems, and despite the fact that they were working together on cutting-edge technologies, they had no real way to share their data with one another except on paper.

In the 1960s, scientists at IBM developed the Standard Generalized Markup Language (SGML), which is a language for creating other languages. Also in the 1960s, researchers at Brown and Stanford Universities developed hypertext, a method of relating similar information.

Berners-Lee developed HTML by combining these two concepts: He took the idea of hypertext and applied it via SGML. He also developed the Hypertext Transfer Protocol (HTTP), which allows networked computers to exchange HTML documents, and the first browser/editor, which he called WorldWideWeb, a name later adopted for the entire system.

The first time Berners-Lee used these new technologies was on Christmas Day, 1990. It took several years to really catch on, but by 1995, new companies such as Yahoo! and Amazon.com were founded, and along with the Netscape browser, the Web quickly took off from there.

Versions of HTML

HTML went through several iterations in the early days when only a few people, mostly at universities, were using it. Berners-Lee first publically promoted the concept of HTML online in 1991, but it was not until 1993 that the Internet Engineering Task Force formally adopted an HTML specification as a "working draft." Two years later, they published the HTML 2.0 specification (note that, technically, there was never an HTML version 1.0).

HTML 3.0 was proposed in 1995, but was deemed too complex for implementation by browsers. It was fairly quickly supplanted by HTML 3.2, the first truly popular version of the language. Unfortunately, version 3.2 introduced a host of browser-specific elements and attributes that had been adopted by browser manufacturers such as Netscape and Microsoft, creating challenges for Web designers who wanted to create pages that would correctly display across multiple browsers.

Realizing that what had begun as a fairly simple text markup language was fast becoming too complex to manage, the World Wide Web Consortium (W3C), the body that had been established to manage the development of future versions of HTML and other Web-related technologies, released HTML 4 in 1997. This marked an attempt to find a common ground between the needs of future developers to have an easy-to-use and easy-to-learn yet robust language while preventing the then-millions of existing pages from breaking in newer browsers. The goal of this new version was to encourage Web designers to move away from the bloated, browser-specific code they had been forced to write and instead return their HTML documents to the original ideas of the language; they would then focus solely on the underlying structure of the document instead of visual formatting, which would now be handled by Cascading Style Sheets (CSS), the newly-developed formatting language of the Web.

The "Flavors" of HTML 4.01

HTML 4 (and the update version 4.01) has three versions. HTML 4.01 Transitional encourages, but does not require, designers to move from the presentation-heavy code of earlier HTML and begin using CSS for formatting. HTML 4.01 Strict requires that only structure-based code be used, with all formatting handled by CSS. The third, less commonly-used HTML 4.01 Frameset allows developers to create pages that utilize frames for page layout.

HTML Elements and Tags

HTML is made up of *elements*, which describe to the browser how a particular piece of text should be treated. Elements are expressed in HTML as tags, which consist of the element surrounded by angle brackets, or the greater than and less than symbols. An example of a tag that uses the `html` element is `<html>`.

Attributes

There are times when you need to expand on the information you are providing in a tag. For example, it is not enough to simply tell the browser that you want to display an image on the page; you need to tell it which image to display. This additional information for tags is given through *attributes*. Attributes appear after the element but before the final angle bracket in an opening tag. Attributes consist of the attribute name, an equal sign, and a value, which will be enclosed in quotation marks.

```
<img src="images/logo.gif">
```

Note that attributes are never repeated in the closing tag, and whenever you use more than one attribute in a single tag, they can be presented in any order.

Container and Empty Tags

Most HTML tags will surround a block of text, thus marking it up. These tags use elements that describe both when the markup needs to begin, and when it ends. For example, if you want to make some text bold, you need to tell the browser both when to begin making the text bold and when it should stop doing so. These *container tags* use an opening tag, which has the element and the angle brackets, and a corresponding closing tag, which includes a forward slash after the opening bracket and before the element.

```
<strong>Some text to be made bold-
faced</strong>
```

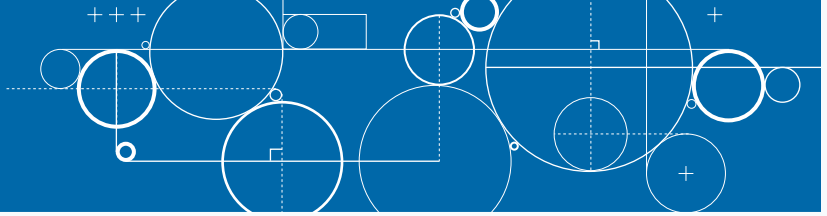
While the majority of tags are containers, there are a set of tags that essentially represent instructions to the browser, and thus have no matching closing tag. For example, while it makes sense that you need to tell the browser where to begin and end bold text, it likewise makes sense that you only need to tell the browser where to place an image — it has no logical end point, and therefore no closing tag. These are called *empty tags*, as they do not contain content.

HTML Syntax Rules

While I have already noted that HTML uses a very loose syntax, there are still a few things about which you need to be aware.

- **HTML is case-insensitive**. You can use any case you want for elements, attributes, and attribute values. You can also freely mix and match cases, but being consistent will make your code easier to read.
- **HTML is whitespace-insensitive**. Within tags, the element must immediately follow the opening angle bracket. After that, however, you can put as much space as you wish, including carriage returns, between the element and any attributes, between attributes, and before the closing bracket. Outside of tags, you can use any whitespace characters, again including carriage returns, within your content without affecting the final display in your browser. Browsers always ignore more than one consecutive whitespace character, and always ignore tabs and carriage returns.
- **When using attributes, you can enclose the value of the attribute in single or double quotation marks**. Some attribute values need not have quotation marks at all, but it is easier to simply always use them rather than try to figure out which cases require them and which do not.

Introducing XHTML 1.0



While development of HTML continued, the W3C was also hard at work on another markup language. The Extensible Markup Language (XML) was designed to allow developers to create documents that focused solely on the actual data of their page, without regard to how such data might eventually be displayed. XML relies on a much stricter syntax than HTML. While beginners may find the loose syntax of HTML easy to learn, languages with stronger syntaxes are ultimately easier to debug and manage.

To provide Web designers with the tools they needed to be better able to troubleshoot pages, the W3C released Extensible HTML (XHTML) 1.0, ultimately intending that it replace HTML 4.01. XHTML uses the same set of elements and tags as HTML, and even includes the same three "flavors" — Transitional, Strict, and Frameset. However, it requires that pages be written in the much stricter XML syntax.

XHTML Syntax Rules

Although HTML and XHTML share a set of elements, there are several important differences between the syntaxes of each:

- **XHTML is case-sensitive**. All elements and attributes must be lowercase.
- **XHTML requires that attribute values be quoted**. You can use single or double quotation marks, but you must always use them.
- **XHTML requires that all attributes have values**. While rare, there are a few scattered examples of HTML attributes that have no value, and instead use a single word. In XHTML, these must have a value, which is always the name of the attribute repeated as the value. An example is the input element, used for forms (see Chapter 11 for details). When the input element is used to create a radio button, and you wish to have the button selected by default, HTML uses the checked attribute. In XHTML, it is written `checked="checked"`.
- **XHTML requires that tags be properly nested**. It is almost always possible to nest tags within other tags. For example, much of the content on your page will be in paragraphs, which use the `p` element. Within a paragraph, you may have text that you want to be both bold (using the `strong` element) and italic (using `em`). The code in this case would be `<p>Some <strong><em>bold, italic</em></strong> text.</p>`. Here, the `em` tag is nested within the `strong`, which is in turn nested within the `p`. This is the proper XHTML syntax, where the tags close in the opposite order from which they were opened, and each element is completely nested within the other. HTML would allow for `<p>Some <strong><em>bold, italic</strong></em> text.</p>`, where the order of the closing tags is reversed.
- **XHTML requires closing tags**. As was discussed previously, HTML has container tags that surround marked-up text and empty tags that merely give an instruction to the browser. In XHTML, even empty tags must be closed. You can either add the closing tag immediately after the opening, as in `<img src="images/picture.gif"></img>`, or simply add the forward slash before the closing angle bracket: `<img src="images/picture.gif" />`. Either is allowed, but the latter is preferred if for no other reason than it takes less code.

The one syntax rule that is the same between HTML and XHTML is that both are whitespace-insensitive.

Declaring the "Flavor" of XHTML

For your XHMTL document to be considered valid, you must begin the document with a *document type declaration.* This statement informs the browser which "flavor" of XHTML you are using to write your code. Although the declarations, also called *DOCTYPES*, look complex at first, they will always be exactly the same on every document that uses that type, so you can simply copy and paste them from one document to the next.

The XHTML Transitional DOCTYPE:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
```

The XHTML Strict DOCTYPE:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict //EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
```

The XHTML Frameset DOCTYPE:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
frameset.dtd">
```

Declaring the XHTML Namespace

Because XHTML documents are technically written in XML, it is necessary to also define the XHTML namespace for your document. This is a fairly technical XML detail, and fully understanding its purpose is not necessary, especially considering that, like the DOCTYPE, it will always be the same on every one of your documents (regardless of which DOCTYPE you are using). It is added as an attribute to the first tag in your page, which uses the `html` element:

```
<html xmlns=" http://www.w3.org/
1999/xhtml>.
```

Advantages of XHTML

Writing XHTML only requires a small amount of additional effort beyond writing HTML, and yet it provides several advantages. First, you can validate XHTML documents using one of several free online resources. Browsers will not throw errors or attempt to inform you in any way when they encounter mistakes in your Web page coding. Instead, they will simply ignore any elements or attributes they do not understand and continue trying to render the page as best they can. Unfortunately, this can make troubleshooting problem pages difficult. XHTML documents with a proper DOCTYPE declaration can be checked against a *validator*, which lets you know if there are problems and allows you to fix them.

Second, XHTML requires that you write better, cleaner code, which will prove much easier to edit and maintain later. Third, XHTML is "future-proof," meaning that because you declare the version of the language you are using right from the beginning, browsers into the distant future should be able to correctly render your pages.

Writing XHTML Strict documents also allows you to separate the content of your page, the XHTML, from the presentation of your page, the CSS, giving you much more flexibility in your designs. This topic is explored in greater detail throughout this book.

Introducing Cascading Style Sheets

The W3C developed CSS in an effort to provide a more robust formatting language for the Web. CSS can be used with any Web page, regardless of whether you are using HTML or XHTML, and regardless of the DOCTYPE you choose.

Versions of CSS

The original specification of CSS, version 1.0, was released at the end of 1996. It introduced text formatting, including bold and italic typefaces, indents, and spacing; the ability to add foreground and background colors to elements; alignment for most elements; and margins, padding, and borders on elements.

Version 2.0 was adopted about eighteen months later. Its most useful additions were most of the positioning properties used for page layout and media types that allow designers to target style sheets to specific uses.

Version 3.0 is still under development by the W3C, with no specific timeframe for adoption.

Browser Support for CSS

Perhaps the most commonly cited reason for not adopting CSS is concerns over browser support. While these concerns were at one time well justified, today few browsers lack significant support for CSS.

Of the major browsers on the market today, all fully support CSS 1.0, and all support most if not all of the properties in CSS 2.0. In fact, some browsers such as Mozilla Firefox even support some features in CSS 3.0, although they do so sporadically.

However, not all browsers implement CSS exactly according to standards, so there are still significant differences in the ways in which pages may render. It is absolutely imperative that, as a Web designer, you test each of your pages on a variety of browsers to ensure that it functions properly.

Basic CSS Syntax

CSS relies on a completely different syntax from XHTML. Instead of elements and tags, CSS uses *rules*, which are made up of a selector and declarations.

The *selector* is a reference to the portion of the XHTML document that should be formatted. The selector can be an XHTML element, or a special CSS selector such as an ID or class. ID and class selectors will be discussed in Chapter 9.

Declarations are made up of property-value pairs. Each property is separated from its value by a colon, and each declaration is separated from the next by a semicolon. The whole set of declarations is enclosed in curly braces:

```
p { color: #ff0000; font-weight:bold; }
```

CSS selectors are case-sensitive, while the declarations are case-insensitive. CSS is whitespace insensitive.

Properties that contain more than one word, such as font-weight, separate the words with hyphens. If a property value needs more than one word, such as in the name of a font like Times New Roman, that value needs to be in quotation marks.

Inheritance

XHTML documents are made up of nested sets of elements, creating parent-child relationships. Many, but not all, CSS properties inherit from parent-to-child elements. For example, if you have a paragraph that contains a span, the span will inherit most of the parent's properties, especially those regarding text and font styles.

Advantages of CSS

While it is more difficult to learn CSS formatting than HTML formatting, there are many advantages to using it:

- **CSS provides many formatting options not available in HTML**. Some examples include the ability to add borders and background colors to any element, control the spacing between lines, override default formatting options, and implement layout controls.
- **Pages that use CSS load and render more quickly**. Because your XHTML documents are smaller when you use CSS, the browser can download the pages more quickly; and because it contains only the code needed for the content, it renders more quickly as well. Subsequent pages in your site see an even bigger speed increase, because the CSS document has already been downloaded and cached by the browser.
- **You can reformat an entire site**. By separating your content and your presentation, you can change the formatting on your entire site from a central location — your CSS file. Therefore, it is possible to reformat your site without having to dig into individual content pages. You can find an example of this capability at www.csszengarden.com, which is a site that presents a single page with many different formats, all using CSS.
- **You can repurpose your site without rewriting it**. These days, more and more users are viewing pages on alternate devices such as cell phones. This creates both a unique challenge and a unique opportunity to designers. The challenge is in designing pages that can look good on a 19-inch widescreen computer display and a 3-inch cell phone screen. By having all of your presentational code in CSS, you can switch layouts dynamically as your users move from one device to the next, again without having to rewrite any of your underlying XHTML.

The Cascade

As you build your style sheets, you will encounter many instances where you have more than one style rule applying to the same element on your page. In these cases, the *cascade* specifies how conflicts are resolved. Style rules that are closer to the element in question take precedence over those farther away. Also, more specific rules override less specific rules, so an ID selector that targets one specific element will take precedence over a general element. Only declarations directly in conflict will be overridden, so a declaration from a less-specific selector will apply if the more-specific selector is silent on that property.

For example, take the following two rules:

```
p {color:#999999; font-weight:bold; }
p#heading {color:#000099; }
```

A paragraph with an ID of `heading` would be dark blue, using the `color:#000099` from the more-specific `p#heading` rule but would also be bold, as the more-specific rule does not state a `font-weight` property, so the less-specific rule using the element selector applies.

Linking and Importing

Keep your style sheet information in a separate document. This separates content from presentation, a key consideration behind the development of CSS. It also enables you to apply the same style sheet to multiple pages, so you can create one design for your entire Web site.

CSS provides for two ways to attach a CSS style sheet to an XHTML page. The more common is to use the XHTML `link` element:

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

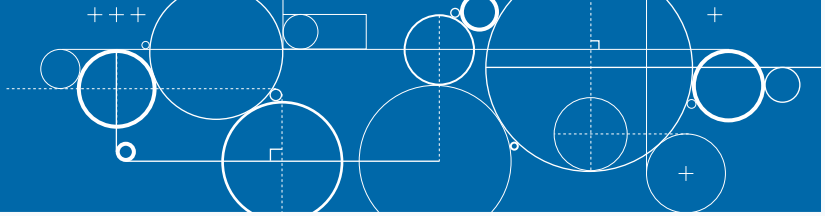
All three attributes are required. The value of the `rel` attribute is almost always `stylesheet`, and the value of `type` is always `text/css`.

The second method is to import your style sheet through the special CSS `@import` rule, which appears within an XHTML style tag:

```
<style type="text/css">
@import ("path_to_css_document");
</style>
```

The two methods are not mutually exclusive. If you use both, rules in the imported style sheet will take precedence over rules in the linked style sheet, which enables you to force certain properties to be overridden.

Introducing JavaScript



In the early days of the Web, a need arose to allow designers to provide some sort of interactivity for their users. Netscape, at the time the developer of the most popular browser, developed a scripting language for that purpose.

A Brief History of JavaScript

JavaScript was first introduced in Netscape version 2 in 1996. It was originally called LiveScript, but as Netscape was also adding support for Java to the browser, they decided to change the name. Several months later, Microsoft introduced their version of the language, called JScript, along with Internet Explorer (IE) version 3.0. JScript was intended to be compatible with JavaScript, with a different name simply to avoid trademark issues, although it did introduce several new features.

Later, Microsoft submitted JScript to Ecma International, an organization that develops and maintains standards for computer systems. Today, both JavaScript and JScript try to maintain compatibility with the standardized version, which is called ECMAScript, although both still retain certain variations from the standard.

Running JavaScript

In order to run, your JavaScript code must be interpreted by an application. In most cases, the application being used to run it will be a Web browser. However, many other applications support JavaScript in some form today, including Adobe Reader, which can interpret scripting contained in Portable Document Format (PDF) documents. Some commercial applications rely on the language to implement many aspects of the program itself. For example, many of the dialog boxes in Adobe Dreamweaver CS3 are written in HTML, with their functionality provided by JavaScript.

Writing JavaScript

JavaScript is text-based, and can be written in any text editor. Any editor designed to assist in writing Web pages is likely to provide help in the form of code hints and syntax highlighting for JavaScript.

Browser Support

Every major modern browser offers full support of JavaScript. Microsoft's IE officially supports ECMAScript, but this in effect means it supports JavaScript. Non-Microsoft browsers tend to claim to not support JScript, although that language's close relationship to ECMAScript means that other browsers effectively do support it.

JavaScript Is Not Java

Java is a very powerful object-oriented programming language from Sun Microsystems. JavaScript is a scripting language. Java is designed to allow developers to create full desktop applications, and in fact many of the applications that you run on your computer are likely written in Java. JavaScript is designed to enhance the capabilities of Web browsers. Except for the name, the two languages in fact have absolutely nothing in common. While it is common for beginning Web designers to confuse the two, care should be taken not to as there is no help available for Java that would be useful for JavaScript programming, and vice versa.

Advantages of JavaScript

As a scripting language, JavaScript is fairly easy to learn and implement. Unlike compiled languages such as Java or C#, JavaScript developers do not need any additional software to run their applications; they can use any modern browser instead.

JavaScript allows developers to achieve many effects not offered by HTML. For example, HTML form controls are extremely limited, and offer little in the way of validation controls to ensure that the data being entered is what is expected. JavaScript allows developers to write as complicated a validation scheme as they need on top of the form.

JavaScript can also work in conjunction with CSS to achieve advanced visual effects such as drop-down menus, accordian effects, and much more.

The Rise and Fall of Dynamic HTML

In the late 1990s, many Web sites began to want to expand the capabilities of the browser beyond merely presenting HTML, and saw JavaScript as a way of doing this. Dynamic HTML (DHTML) was an attempt at combining HTML, JavaScript, and CSS to produce many exciting visual effects. However, wide adoption of DHTML was hindered by important differences in browser implementation, at times requiring two completely different scripts to render in Netscape and Microsoft browsers. Many DHTML effects sacrificed usability and accessibility in the name of "cool" effects, and the language has been abandoned for the most part today.

Disadvantages of JavaScript

The biggest problem with using JavaScript is that all browsers allow users to disable it. Although few users actually take this step, there are enough people out there who sometimes or always surf the Internet with JavaScript disabled to cause problems for sites that rely on it for mission-critical site content.

Disabled users often encounter problems with JavaScript as well. Given much of the implementation of the language revolves around creating visual effects, assistive technologies such as screen readers for the blind often lack the capability to correctly interpret JavaScript effects.

For these reasons, you should always be sure to provide an alternative to JavaScript effects, and avoid using the language for anything that would cause the site to break entirely in non-script environments. For example, you should never use JavaScript to completely render your navigation.

Ajax: DHTML, Take Two

Ajax, or Asynchronous JavaScript and XML, was developed as a way to allow developers to extend on the concepts of DHTML while fixing many of its problems. Most Ajax development is done through pre-built JavaScript libraries, saving developers time in having to rewrite code. The better, more widely adopted libraries focus on good usability and accessibility, and also provide many features previously unavailable, such as the ability for JavaScript to refresh only a portion of a Web page. The extremely popular Google Maps application (http://maps.google.com) is an example of Ajax.

JavaScript Libraries

Today, many libraries of JavaScript functions exist that enable developers to implement complex scripting effects while requiring that they write little or no script themselves. These libraries free developers from having to spend time coding and debugging applications, and allow them instead to focus on the end-user experience.

Understanding Creation Tools

Because XHTML, JavaScript, and CSS are all written in plain text, you do not necessarily need a specialized tool to create even the most complicated Web pages. Notepad on Microsoft Windows and SimpleText on Apple Macintosh computers are both basic text editors that you can use for Web pages. However, an editor specifically designed for the purpose of creating Web pages will help in many ways.

Code Editors

Code editors are applications that allow you to type your code directly. Unlike simple text editors, code editors will provide many tools to make design easier.

Most code editors share common features:

- **Line numbers**. Debuggers for JavaScript and other programming languages and validators for XHTML and CSS will all return error messages that reference line numbers, so having an editor that shows those numbers makes finding the errors easier.
- **Code hints**. Many editors will display a list of available elements or attributes as you type. Obviously, this saves you from having to have the exact names memorized, but it also helps in reducing your typing time as you can generally only type a few letters of the name before having the editor finish it for you; it also reduces spelling errors.
- **Tag completion**. Editors specifically designed for Web pages will add closing tags where needed, again saving you the time of having to type them.
- **Integrated reference or help functions**. Most editors will include in their help system a complete list of elements, attributes, and a description of each.
- **Color coding and syntax highlighting**. Almost every editor will color code elements and attributes in XHTML, properties and rules in CSS, and language constructs in JavaScript, making the code easier to read and errors easier to spot.

Eclipse

Eclipse is very widely used open source editor. It was originally built for Java developers, but has many plug-ins that extend its capabilities to include other languages, including XHTML, CSS, and JavaScript. It can be downloaded free of charge from www.eclipse.org. Eclipse is available for both Windows and Macintosh.

Macromedia HomeSite

HomeSite is a very feature-rich code editor. Given Macromedia's acquisition by Adobe, there has been little new development of the product, and Adobe has given no indication as to whether they will continue developing it in the future. A free 30-day trial of HomeSite is available at www.adobe.com/products/homesite. HomeSite is only available for Windows.

TopStyle

This editor was developed by the original creator of HomeSite, Nick Bradbury. It currently exists in both a commercial version and a free version with a smaller feature set. TopStyle's main focus is on CSS, and though it can be used to create XHTML, many developers use it solely for editing their style sheets. TopStyle is available only for Windows. Information about TopStyle can be found at www.newsgator.com/Individuals/TopStyle/Default.aspx.

BBEdit

A popular editor amongst Macintosh developers is BBEdit from Bare Bones Software. It features many of the standard features of text editors, along with powerful search tools that can search across documents and utilize regular expressions, as well as a built-in File Transfer Protocol (FTP) client. You can purchase BBEdit at www.barebones.com.

TextMate

TextMate was developed as an alternative to BBEdit for Macintosh users. It is an extremely configurable editor, allowing users to create their syntax highlighting schemes, create macros to extend the program's functionality, and save snippets to reuse code later. A free trial and purchase information for TextMate can be found at www.macromates.com.

Visual Editors

While code editors tend to give designers more direct control over their code, visual design editors tend to allow for faster workflows as they save time: The developer has to type little if any code. More importantly, they present the designer with a visual representation of the final page, allowing for a more natural design process. Visual editors are often referred to as WYSIWYG (What You See Is What You Get) editors, although none of them present an absolutely perfect rendering of how the page will appear in a browser.

Features that you will find in most visual editors include:

- A visual design view with a representation of a browser's display
- Toolbars or panels for inserting common elements and modifying their properties
- A visual CSS editor for simplified style sheet creation and modification
- An integrated FTP or other publishing tool for uploading completed pages
- Powerful site management features that allow for integrated file management
- A code view for manually entering or modifying code if needed

Adobe Dreamweaver CS3

Dreamweaver is a feature-rich, powerful design tool. Considered by many to be the industry standard amongst Web designers, Dreamweaver not only supports the features mentioned previously, but it also has tools for creating dynamic, database-driven Web sites with a minimal amount of code. Its focus is on designing standards-based XHTML pages. Dreamweaver CS3 is the first version of the program released by Adobe, and includes support for Adobe's Ajax library, called Spry. Dreamweaver also has tight integration with other Adobe products, including easy insertion of Flash movies and round-trip graphic editing from Fireworks and Photoshop. You can download a 30-day trial of Dreamweaver from www.adobe.com/products/dreamweaver. Dreamweaver is available for both Macintosh and Windows.

Microsoft Expression Web

Expression is Microsoft's latest entry in the visual design tool market, replacing the now-discontinued FrontPage. Unlike its predecessor, Expression is geared towards professional designers and is being placed as a direct competitor to Dreamweaver. It supports CSS and standards-based design, and also features tools that allow developers to create pages that use Microsoft's ASP.NET server-side technology. You can download a trial of Expression from www.microsoft.com/expression/products/overview.aspx. It is only available for Windows.

Adobe GoLive

Before their acquisition of Macromedia, Adobe's Web development tool was GoLive. Its best features focus on its integration with other Adobe products, in particular Photoshop, from which it supports round-trip editing of graphics, and InDesign, which allows designers to take complex print layouts and convert them to Web-friendly layouts. While GoLive is no longer a part of the Creative Suite product line, having been replaced by Dreamweaver, it is still in active development, with a new version, GoLive 9, that was released in 2007. Visit www.adobe.com/products/golive to download a trial version. Like other Adobe products, GoLive is available in both Macintosh and Windows versions.

Microsoft FrontPage

Even though it has been officially discontinued by Microsoft, FrontPage remains a popular editor. FrontPage was originally designed to make creating Web pages as easy as creating documents in a word processor such as Microsoft Word. Even though it had a very simple interface and required little or no learning, it tended to lack many features professionals desired, and its reliance on creating code that would only render correctly in IE made creating cross-browser and cross-platform pages difficult. Still, its ease of use more than compensated for that in many people's eyes.

Understanding Web Browsers

Regardless of which tool you use to design your pages, you should keep in mind that, ultimately, you are designing for the browser. That is the tool on which your users will view your page. Unfortunately, from the early days of Web development, browsers have posed numerous problems for designers.

The Browser Wars

In 1994, Marc Andreessen and Jim Clark, both of whom had been working for several years on the Web in its earliest implementations, founded the Netscape Corporation and published Netscape Navigator, the first widely adopted Web browser. Netscape Navigator quickly became the most popular browser in the world, but in 1995 Microsoft released IE 1.0 as an add-on feature to Windows 95.

The two companies would fiercely compete for the browser market for the next several years in what became known as the "browser wars." Unfortunately, both for designers and HTML, the competition mostly revolved around each browser releasing a new version with new features — proprietary HTML elements or attributes. This was ultimately the reason behind the development of HTML 4.01, XHTML 1.0, and CSS, so in the end, the Web community benefited more from the browser wars than they were hurt by them.

The browser wars eventually ended as IE benefited from its tight integration with Windows while the operating system gained dominance. In 1999, Netscape was purchased by AOL, which uses the brand to this day but officially disbanded the company in 2003.

Internet Explorer

By far the most dominant browser on the market today is Microsoft Internet Explorer, or IE. The fact that it is installed by default on every copy of Windows ensures that it will continue to dominate the market for the foreseeable future.

From a Web designer's perspective, this is both good and bad. The good is that if your page displays correctly on IE, you can be sure that in most situations the vast majority of your users will see your page as you do. The bad is that IE has long been the least standards-compliant browser on the market, so getting your page to display correctly on it, while still maintaining the appearance of the page on other browsers, can be challenging.

In 2006, Microsoft released IE 7. They introduced several new features in this version, but for designers, the most important aspect was the news that IE 7 would adhere much closer to standards than past versions, and correctly implement several CSS rules that before had been either ignored by IE or incorrectly rendered.

All users of Windows Vista have IE 7 by default. At this time, it is the only version of IE that will work on Vista. Most Windows XP users have by now also upgraded to IE 7, although significant numbers still use IE 6.

For many years, Microsoft developed a version of IE for Macintosh computers. Oddly, this version of IE often displayed pages differently from the Windows versions, at times being more standards-compliant, and at times less. In 2006, Microsoft announced the end of development of the Mac version of IE.

Mozilla Firefox

In 1998, Netscape created the Mozilla Organization, which would continue the development of their browser as an open source project. Mozilla is the name of the underlying rendering engine used by many browsers today, and is a combination of Mosaic, the name of the browser Andreessen and Clark built before founding Netscape, and Godzilla.

Today, Mozilla supports several open source projects, the most relevant being the Firefox browser. Firefox is designed as a standards-compliant browser, which means that if you create your page following the rules of XHTML and CSS set forth by the W3C, your page should display correctly in Firefox. Due mostly to this, Firefox is often the primary browser on computers used by Web developers.

Firefox is currently in version 2.0, and is available for Windows, Macintosh, and most Unix machines. You can download the browser for free from www.getfirefox.com.

Apple Safari

Apple announced in 2003 that they had developed their own browser, to replace Internet Explorer for Macintosh as the default browser on Mac OS X. Beginning with the release of OS X version 10.3, Safari was bundled with the operating system; as of the release of version 10.4, it is the only browser included.

Because Safari is based on the WebKit rendering engine, rather than Mozilla, early versions had some rendering oddities that made designing pages for it more challenging. Today, however, it is mostly standards compliant, so pages that render correctly on Firefox will almost certainly render correctly on Safari.

For many years, the biggest challenge for the majority of designers in dealing with Safari was the fact that it was only available for the Mac. This changed in 2008 when Apple released Safari 3 for both Mac and Windows. Safari is available as a free download from www.apple.com/safari/download.

Opera

The Opera browser, from Opera Software in Norway, was for a very long time the only browser for which you had to pay. Therefore, it did not develop as strong a following as its feature set should have allowed. When Opera 8 was released and it was announced that it would be free, more people decided to take a look.

Opera has long been standards-based, but its most recent version, 9.2, features tools that allow designers to see what their pages may look like on very small screens, so those designers who wish to develop pages for mobile devices will often use Opera for this purpose.

Opera is available for Windows and Macintosh from www.opera.com/download.

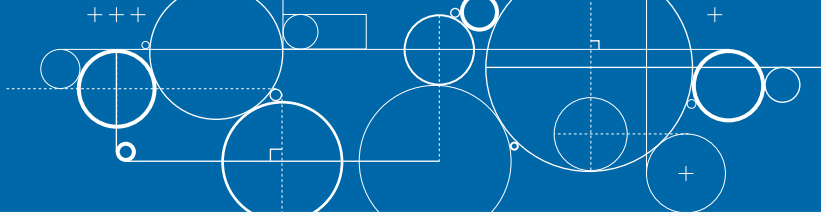
Which Browser Should You Use?

Which browser you use for general Web use is a completely personal decision. However, for Web design, you need to have as many of the browsers listed previously as you can. In general, you will want to design initially with the more standards-based browsers, such as Firefox, in mind, and then modify your page as needed to fix problems in the less standards-based browsers, such as Internet Explorer.

Worrying About Old Browsers

A lot of clients and bosses may have read or heard something saying that their pages should be built to support older versions of browsers, particularly older versions of Internet Explorer such as 5 or 5.5, or Netscape 4.7. These browsers offer varying levels of support for modern design principals, especially CSS, and can pose a significant challenge to designers. The obvious answer to the question about whether or not to support them is to determine if anyone is actually still using them. Unfortunately, accurate browser usage statistics for the Web as a whole are hard to come by and will vary greatly. The good news, though, is that you do not need statistics for the Web as a whole — you only need to care about which browsers *your* users have, and that is a statistic that is easy to come by, because your Web server will have logs of that. So before you start rebuilding your page for Netscape 4.7, check your server logs. Odds are good that it will show that you have little or no users still on old browsers, and you are free to stop worrying about them.

Understanding Web Servers



While the browser is the palette for which you design your pages, Web servers play an important role as well. As a Web designer, you do not need to understand servers to the point that you would feel comfortable administering a server — hopefully, that will be someone else's job. However, a basic understanding of servers and how they work will enable you to better appreciate their role in getting your pages to your users.

The Client-Server Model

The Internet and all of its associated applications, including the Web, use a client-server networking approach. Early client-server networking relied on so-called "dumb terminals" on the client side, which were little more than monitors connected to the network that relied on the server to do all of their processing. Today, of course, your users will be using full-fledged computers that are much more powerful than the servers of old. However, these clients must still connect to a server in order to receive Web documents.

HTTP

Berners-Lee, in creating the Web, established a new protocol that would allow for the transmission of hypertext documents over a TCP/IP connection. He dubbed this the Hypertext Transmission Protocol, or HTTP. While few users of the Internet have any knowledge of what it is or why it exists, one cannot escape being aware of HTTP as it, of course, begins most Web addresses.

TCP/IP

Networking relies on established protocols to ensure that the computers on both sides of the connection are able to communicate and exchange their data as needed. The most important protocols for the Internet are the Transmission Control Protocol (TCP) and the Internet Protocol (IP). TCP allows for the delivery of that data, while IP establishes a consistent addresses format that allows machines anywhere in the world to find one another. TCP/IP dates back to the 1960s, and in fact the co-developers of the protocols are often referred to as the "fathers of the Internet." Every computer that wishes to directly connect to the Internet needs to have a unique IP address. Currently, the most widely adopted form of IP is version 4, in which addresses are expressed as four numbers between 0 and 255, separated by periods or dots. IP version 4 yields just over 4 billion unique addresses. However, the ever-increasing popularity of the Web has created a shortage of addresses, so a new version of IP, version 6, is beginning to gain use. IP version 6 uses a new address scheme that will yield a seemingly endless supply of address (3.4×10^{38} total).

Web Servers

A *Web server* is merely a piece of software that runs on a computer that is connected to the Internet and allows its files to be shared. While many people use the term to refer to the physical machine on which the Web server software is running, it is important to remember that a Web server is actually software. While many Web servers exist, by far the two most popular and widely used are Internet Information Services by Microsoft and the Apache Web Server from the Apache Software Foundation.

Regardless of the manufacturer, Web servers all do essentially the same thing. They listen for requests to be made via HTTP, and respond to those requests, usually by finding the document the user has asked for and sending it to them. For basic HTML documents, the server will not open or read the document, but will instead merely send it off to the browser.

Internet Information Services

Microsoft's entry in the Web server market, Internet Information Services (IIS), was introduced in 1995 on the Windows NT Server operating system. Since then, the company has released a new version of IIS with each new iteration of its server-based operating system, so today versions exist to run on Windows 2000, Windows XP, Windows 2003, and Windows Vista. While each new version has sought to improve security, the basic feature set of the server has remained mostly unchanged. One of the nicest features of IIS is its graphic user interface, which makes configuring the server much easier than its competitors.

IIS unfortunately only runs on Windows, and only on the version of Windows for which is was designed. It is free, and is included along with any of the versions of Windows mentioned previously with the exception of Windows XP Home Edition, which does not include or support IIS.

Apache Web Server

The main competitor for IIS is the open source Apache Web Server. Maintained by the non-profit Apache Software Foundation, this lightweight server will run on practically any operating system, including Macintosh, various Unix installations, and Windows. Macintosh OS X ships with Apache preinstalled.

When it was originally developed, software engineers took a series of existing software components and pieced them together to create the server. Thus, it was literally "a patchy" server, and when it was released, Apache decided to keep the Apache variation of that name for the server itself as well as the foundation.

Officially, Apache lacks any user interface for configuration. Instead, users must modify a text file, http.config, in which all of the server's settings are kept. This file can be daunting to the uninitiated, but with some explanation and a little practice, it becomes fairly simple to maintain the server. There are several third-party implementations that provide some sort of graphical interface for configuration as well.

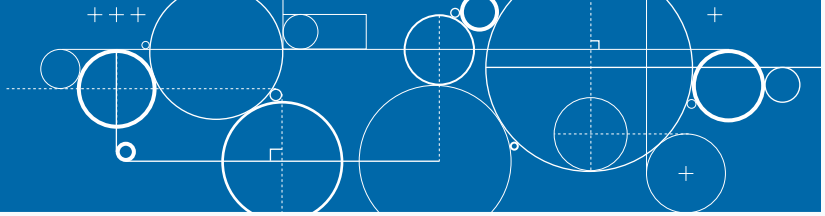
Default Documents

Often, users will make a request to a Web server without asking for a specific document. This always occurs when a user merely enters the domain name of a site, as typing in **www.wiley.com**. In these cases, the browser is essentially coming to the server and merely asking to connect instead of requesting a document. The server, however, still needs to send a document back to the user, so every server has a setting that allows the administrator to determine a set of special filenames that will be served by default. Apache has long set the list to index.html and index.htm, while Microsoft has always insisted on preconfiguring IIS to look for Default.htm. Most of the time, the server administrator will add index.htm and index.html to the IIS settings, as these have become well established as default filenames for Web site's home pages.

Root Folders

Installing a Web server on your computer opens an obvious security hole in your machine — you are running software that invites others to connect to you and retrieve files. To counteract this issue, servers are configured to retrieve documents from one specific folder on the machine's hard drive. It can see and send files from subfolders of the root, but it cannot go above the root to any other directory. The root is the directory in which all of your Web pages, and any supporting files such as CSS documents, images, Flash movies, and anything else must be located. The server will be unable to locate any files stored outside of the root directory. When setting up your computer on which you plan to develop Web pages, you should mimic this functionality of the server and make sure that your pages will continue to work after you transfer them to the server.

Plan Your Web Site



When you begin work on your Web site, there will be an urge to simply sit down at the computer, open up your editor, and start creating. However, you will soon find that you start running into roadblocks, as you continually need to tweak your navigation, find new content, or even reach a point where you are simply unsure of what to do next.

Instead of this haphazard approach, you should force yourself to sit down and carefully plan your site before you ever begin typing anything. You can do this planning on the computer or on paper — whichever is easier for you.

Why Do You Want a Web Site?

The first thing to determine is exactly why you are creating a Web site. What exactly do you hope your users will get out of it? With billions of sites on the Web, the odds are extremely good that there are already sites out there that provide the same or similar information or goods or services that your site will. So ask yourself, why would someone want to use your site instead of someone else's? Occasionally, you may even talk yourself out of building the Web site at all if you cannot satisfactorily answer this question.

Who Are Your Users?

No traditional brick-and-mortar company would consider opening a new location without performing thorough market research, and yet few Web sites do the same. You need to know who your users are, at least in a general sense, before you begin any development. Are they more likely to be young or old? How much education are they likely to have? How much income? Are they likely to be computer and Web savvy or not? All of these questions should influence your approach to your pages. Computer professionals expect a much different experience when they visit a site than computer neophytes. The former group is more likely to be using higher-end hardware and software and high-speed connections, so you can feel more comfortable using multimedia on your site if that is your target. If, on the other hand, you expect a lot of users from rural areas where high-speed connections are less common and you expect your users to be much less technically inclined, then you need to adjust your site contents accordingly.

Where Will You Get Your Content?

Content is king on the Web. People surf the Web for a lot of different reasons, but in the end they are all looking for one thing: content. The content of your site will primarily be the text and images on your pages. If you are building a personal site, then obviously the content will come from you. Corporate sites may have a team of developers to help create the content. Content may already exist, either in printed form, which will require that it be input into a computer, or it may already exist electronically and can possibly be copied and pasted from some other application.

If you plan to offer multimedia on your site, you will possibly have an even bigger hurdle to overcome, as the creation of music and video requires much more time and energy and will certainly involve other, likely expensive equipment and software. Either way, you will want to have a good idea of what your content will be and where you will get it before you start work on the site, and especially before you attempt to give an employer or client a proposed deadline for completion of work on the site.

What Will the Visual Design Look Like?

While you should focus on content first and foremost, the Web is without a doubt a primarily visual medium, and what your site looks like is important. Most professional designers will sketch out many different designs before they settle on the one they want. Some use computer graphics programs such as Adobe Photoshop or Adobe Fireworks for this purpose, while others simply draw rough sketches on paper. Once you settle on a design, you can begin to figure out how the content will fit and make changes to the design as needed. Note that good Web sites modify the design to fit the desired content, not the other way around, as content is by far the more important of the two. Ask yourself this: Would you be more likely to return to an ugly site that gave you the information you need, or a gorgeous site that had little to offer beyond the looks? Hopefully, of course, you will not have to make this choice on your site, and will instead have something that is both functional and pretty; just remember that functional should take precedence.

Organize Your Files

As was already mentioned in the section on Web servers, all of your Web site's assets — the HTML documents, images, style sheets, multimedia, and so on — need to be placed within a single directory on your computer as you develop your site, as this is how they will work once on the server. However, it is not considered a best practice to place all of these files loose in one directory. It is not uncommon for even small sites to have many hundreds of files, and larger sites can reach into the tens of thousands. From the very beginning of the design process, you should have a plan for the organization of the files on your site. You can change this organization later, but unless you are using a tool with site management features such as Adobe Dreamweaver, moving a file from one directory to another late in the development process can mean having to edit hundreds of your Web pages to fix now-broken links.

The only file that absolutely must reside directly in the root folder is the home page. Some designers choose to put all of the top-level documents — the documents that represent the main sections of the site — in the root as well. At the very least, you will want a folder to store your images, and if you are using multimedia, you probably want a folder for that as well. If any of the main sections of the site will have multiple files, placing them in a subdirectory of the root will help keep you organized.

The most important thing to keep in mind is that there is not one "correct" way to organize the files. If you are working alone on the site, then you should use whatever organizational scheme makes the most sense to you. If you are working in a team, then you all need to sit down and discuss how you will stay organized.