CS-21

T.Y.B.Sc. (CS)
Principles of Web Design & Web Technologies

Dr. Suhas Pednekar
Vice Chancellor
University of Mumbai
Mumbai

Dr. Dhaneswar Harichandan
Director Incharge
Institute of Distance & Open Learning,
University of Mumbai, Mumbai

Anil R. Bankar
Associate Prof. of History & Asst. Director &
Incharge Study Material Section,
University of Mumbai, Mumbai

Programme Co-ordinator **I/c Faculty of Science & Technology**	:	**Mandar Bhanushe** Assistant Professor Mathematics, IDOL.
Course Co-ordinator	:	**Manisha Divate** Asst. Prof. in Computer Science, MCA, MPhil (Computer), University of Mumbai
Course Writer	:	**Mrs. Pooja Manghirmalani** Asst. Professor Rizvi Collage of Education

March 2019

T.Y.B.Sc. (CS) Principles of Web Design & Web Technologies

Published by	:	Director Incharge Institute of Distance and Open Learning, University of Mumbai, Vidyanagari, Mumbai - 400 098.

DTP Composed	:	**7skills** Shop No.2, J. M. Nagar Complex Co. Hsg. Soc. Ltd., Pandit Dindayal Road, Anand Nagar, Dombivali (W), Thane - 421 202.
Printed by	:	

CONTENTS

1

WEB DESIGN: PRINCIPLES, PLANNING, NAVIGATION, CREATING PAGE TEMPLATES, WEB TYPOGRAPHY

Unit Structure

1.1 INTRODUCTION

Web design is a process of conceptualizing, planning, and building a collection of electronic files that determine the layout, colours, text styles, structure, graphics, images, and use of interactive features that deliver pages to ones site visitors.

Web design encompasses many different skills and disciplines in the production and maintenance of websites. The different areas of web design include web graphic design; interface design; authoring, including standardised code and proprietary software; user experience design; and search engine optimization.

It also encompasses several different aspects, including webpage layout, content production, and graphic design. While the terms web design and web development are often used interchangeably, web design is technically a subset of the broader category of web development.

1.2 PRINCIPLES OF WEB DESIGNING

Good website design needs a wide range of professionals having expertise in different areas. Their collective efforts need to put in when there is a critical decision to take place. Here in this article, we'll outline the 8 essential principles of good website design which must be pondered while developing a website. These design principles will definitely help web designers to develop awe-inspiring designs and to enhance the usability of a website.

Here is the list of 8-good design principles which will make your website aesthetic, user-friendly, effective, and engaging:

1.2.1. Simple is the best

Over-designed website may not work. Putting too many elements on the page may lead to distract visitors from the main purpose of your website. Simplicity always works in an effective web page design. Clean and fresh design of your website not only makes the website appealing, but also help user to navigate from one page to another seamlessly. Loading a website having design features which do not serve the purpose may be frustrating. Keep your design as simple as possible so that the visitors can feel it easy-to-use and can find their ways easily.

1.2.2. Consistency

Consistency in website design matter a lot. Give your attention to match design elements throughout each of the pages. It can be understood that your fonts, sizes, headings, sub-headings, and button styles must be the same throughout the website. Plan everything in advance. Finalize the fonts and the right colours for your texts, buttons etc, and stick to them throughout the development. CSS (Cascading Style Sheets) would come in handy to keep the complete information about design styles and elements.

1.2.3. Typography & Readability

No matter how good your design is text still rules the website as it provides users the desired information. You should keep your typography visually appealing and readable for visitors, along with tricky use of keywords and meta-data. Consider using fonts that are easier to read. The modern sans-serif fonts as Ariel, Helvetica etc. can be used for the body texts. Make proper combinations of typefaces for each and every design elements such as headlines, body texts, buttons etc.

1.2.4. Mobile compatibility

Keeping in mind the ever-growing usage of smartphones, tablets and phablets, web design must be effective for various screens. If your website design doesn't support all screen sizes, chance is that you'll lose the battle to your competitors. There are a number of web design studios or service points from where you can turn your desktop design into a responsive and adaptive one for all screen sizes.

1.2.5. Colour palette and imagery

A perfect colour combination attracts users while a poor combination can lead to distraction. This necessitates you to pick a perfect colour palette for your website which can create a pleasing atmosphere, thus leaving a good impact on visitors. Enhance users experience by selecting complementary colour palette to give a balanced-look to your website design. Remember to white space use as it avoids your website from visual clutter and mess. Also avoid using too many colours. 3 or 4 tones for the whole websites are ample to give appealing and clear design. Same is the case with images. Don't use multiple vibrant images

1.2.6. Easy loading

No one likes the website that takes too much time to load. So take care of it by optimizing image sizes, combing code into a central CSS or JavaScript file as it reduces HTTP requests. Also, compress HTML, JavaScript and CSS for enhanced loading speed.

1.2.7. Easy Navigation

Study shows that visitors stay more time on the websites having easy navigation. For effective navigation, you may consider creating a logical page hierarchy, using bread scrums, and designing clickable buttons. You should follow the "three-click-rule" so that visitors can get the required information within three clicks.

1.2.8. Communication

The ultimate purpose of the visitors is to get information, and if your website is able to communicate your visitors efficiently, most probably they would spend more time on your website. Tricks that may work to establish effortless communication with the visitors are – organizing information by

making good use of headlines and sub-headlines, cutting the waffle, and using bullet points, rather than long gusty sentences.

1.3 PLANNING WEBSITE DEVELOPMENT

Before getting started on any project, it's important to collaborate and make sure everyone has a clear vision before moving forward. Having a concrete plan and identifying potential red flags is the best way to prevent scope creep, which may affect budgets and deadlines down the road.

1.3.1 Determine Goals

What is one trying to accomplish? What's the main purpose of the website? It's important to understand the overall goal of the website when building one. One wants to make sure they are planning your "dream home" and that it's functional and built with the right intentions and audience in mind.

Whether one is trying to increase membership, convert more visitors into leads, or provide investors with valuable information, they need to establish these goals at the beginning so they know what the developer is aiming for.

1.3.2 Define the Target Audience

What are the demographics of the audience one is trying to target? Understanding the target market is vital to creating plans for a website that will appeal to them. Do the research, create buyer personas, and analyze competition. It's also a great idea to look at websites that the target market might visit. Make notes of likes and dislikes. By keeping the target audience in mind, it makes it much easier to design a website that resonates with them.

1.3.3 Search Engine Optimisation (SEO)

It's best practice to always keep SEO in mind as it directly impacts your online performance and success. There is no better time to focus on SEO than when one is building a website as it can help save a lot of time in the long run. Knowing what keywords they want to try and rank for makes it easier to incorporate into the site design and architecture. It's also extremely important these days to have a mobile-friendly or responsive website as Google ranks them higher than those that are not.

1.3.4 Plan for Content

Right content that speaks to the audience and is critical in creating a successful website. One needs to not only make sure the content is there, but also that it is educational and engaging to the audience and optimized for search engines. Developer can have a beautifully structured and designed website but if the content isn't there to back it up and people aren't seeing it, then it's a waste.

1.3.5 Develop Use Cases, Sitemap and Wireframes

Use cases help establish project requirements. They help determine how different users will behave on the site and a way of outlining the steps a user will take to accomplish their goal or task. The more its define and understand various use cases, the easier it will be moving forward. Building out a sitemap helps to organize the content one wants to have on your website. It builds the foundation of the pages developer wants included. When building out site map goal should be to keep it as intuitive and simple as possible. However, it's important that all of the features and functionality are outlined.

A wireframe is essentially a blueprint of your website that shows its skeletal framework. It's a way for the client to see the layout and overall navigation and functionality of the site before the building begins. Wireframes also help the designer get a better idea of where different components need to be and provides them an outline of how the website should function and where different features need to appear.

1.4 NAVIGATION

Most navigation types fall into three primary categories:

Structural

Connects one page to another based on the hierarchy of the site; on any page you'd expect to be able to move to the page above it and pages below it.

Associative

Connects pages with similar topics and content, regardless of their location in the site; links tend to cross structural boundaries.

Utility

Connects pages and features that help people use the site itself; these may lie outside the main hierarchy of the site, and their only relationship to one another is their function.

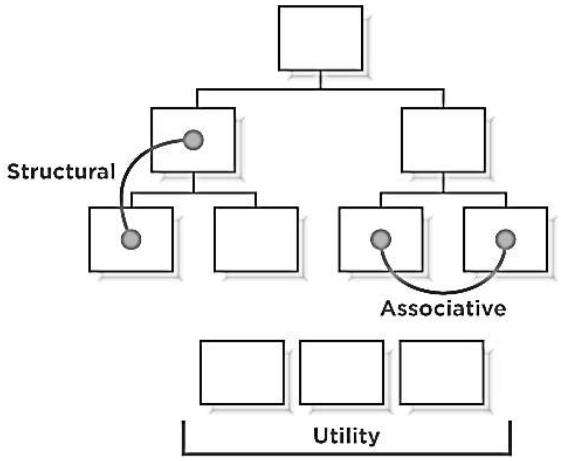


Figure 1.1- Types of Navigations

1.4.1 Structural Navigation

As its name implies, structural navigation follows the structure of a web site. It allows people to move up and down the different points of a site's hierarchy. Structural navigation can be further subdivided into two types: main navigation and local navigation.

Also called: global navigation, primary navigation, main nav.

The main navigation generally represents the top-level pages of a site's structure—or the pages just below the home page. The links in the main navigation are expected to lead to pages within the site and behave in a very consistent way. Users don't expect to land somewhere completely unrelated when using main navigation links. Changes in navigation from page to page are usually small when using the main navigation.

Overall, a main navigation supports a variety of user tasks and modes of information seeking, including known-item seeking, exploration, and even re-finding. From a user's standpoint, the main navigation plays a critical role in using the site:

- ✓ The main navigation provides an overview and answers important questions users may have when first coming to a site, such as "does this site have what I'm looking for?"
- ✓ The main navigation aids in orientation. It is comforting to have a persistent navigation mechanism across the site, particularly for large, information-rich sites.
- ✓ It allows people to switch topics. Visitors can get to other sections of a site efficiently, or they can reset their navigation path and start over using main navigation options.
- ✓ It helps when users get interrupted while navigating and reminds visitors where they are in a site.
- ✓ Main navigation gives shape to a site. In many ways, the main navigation defines the boundaries of the site itself.

The main navigation is often presented in a global navigation area, which generally includes the site logo and utility navigation. (See the following section for more on utility navigation). As the name "global" implies, these controls generally appear in an unchanged, consistent position on all or nearly all pages of a site.

Local Navigation

Also called: sub-navigation, page-level navigation.

Local navigation is used to access lower levels in a structure, below the main navigation pages. The term "local" implies "within a given category." On a given page, local navigation generally shows other options at the same level of a hierarchy, as well as the options below the current page.

Local navigation often works in conjunction with a global navigation system and is really an extension of the main navigation. Because local navigation varies more often than main navigation, it is often treated differently.

Common arrangements of local navigation and main navigation include:

Inverted-L

It is very common to place a global navigation along the top of the page and have local navigation as a vertical link list on the left in the shape of an inverted L.

Horizontal

Local navigation might also be represented by a second row of options under a horizontal global navigation or by dynamic menus.

Embedded vertical

When the main navigation is presented in a vertical menu on the left or right, it's common to embed the local navigation between the main navigation options in a tree-like structure.

1.4.2 Associative Navigation

Associative navigation makes important connections across levels of a hierarchy or site structure. While reading about one topic, the user can access to other topics. Three common types of associative navigation are: contextual navigation, quick links, and footer navigation.

a. Contextual Navigation

Also called: associative links, related links.

As the name implies, contextual navigation can vary. It's situational. Though links may transition to similar pages at the same level within the site, they quite frequently lead to new content areas, different page types, or even a new site.

Generally, contextual navigation is placed close to the content of a page. This creates a strong connection between the meaning of a text and the linked related pages.

b. Quick Links

Quick links provide access to important content or areas of the site that may not represented in a global navigation. Although similar to contextual navigation, quick links are contextual for the entire site, not a given page. They generally highlight frequently accessed content areas or tasks, but may also be used to promote areas deeper in the site. Marketers may see the value in quick links for an upsell effect.

Transitions from page to page using quick links may vary greatly. By definition, they tend to jump around. They may link to a related sub-site, online shop area, or even to a completely new web site.

Quick links often appear at the top or on the sides of pages. On the home page, they may be prominently positioned in component of their own, but on subsequent pages they may be reduced to a drop-down or dynamic menu.

c. Footer Navigation

Located at the bottom of the page, footer navigation is usually represented by text links. These often access a single page with no further levels of structure below them—a dead end, so to speak. Traditionally, footer navigation contains supplementary information not pertinent to main topic of the site, such as copyright information, terms and conditions, and site credits. In this sense, footer navigation doesn't address a specific user need, but addresses a legal requirement for site owners. Footer navigation is often used as a catch-all for various types of content and it can lack consistency in an organizational scheme.

1.4.3 Utility Navigation

Utility navigation connects tools and features that assist visitors in using the site. These pages are generally not part of the main topic hierarchy of the site. For example, a link to a search form or help pages aren't part of the main navigation or local navigation systems. Other options may not have a page associated with them at all. Instead, they are functions of the site, such as logging out or changing the font size.

Utility navigation may lead to varying page types or site functions. Transitions from page to page may be dramatic at times. For instance, from a single mechanism there may be links to a shopping cart, to a search form, and to a page about the site owner's organization—all quite different from one another, and potentially requiring significant reorientation on each new page.

Utility navigation is generally smaller than primary navigation mechanisms and appears on the top, sides, or bottom of the page. Global utility navigation quite often appears as simple text links. In some cases, the utility navigation is very closely related to the main navigation. As mentioned, utility navigation and main navigation often appear together in a global navigation area.

1.5 CREATING PAGE TEMPLATES

A website template (or web template) is a pre-designed webpage, or set of HTML webpages that anyone can use to "plug-in" their own text content and images into to create a website. Website templates are main attributes of a website. It is like a form of letter that is designed to justify separation of presentation and substance of the website. The main function of website template is to generate and produce web pages at a fast speed. The algorithm and the programming language of a website template remain pre-defined and work schematically once given instructions.

As e-commerce continues to change the way people become attracted to websites, it is wise to choose good and effective templates for the websites. If the template of a website is not attractive, people will not visit the site frequently. A good looking website template is eye catching and it shows the creativity of the web developer, designer or web master. Starting up a website requires both time and money. Free website templates solve both problems at the same time.

Flexible Website Templates

Creative web developers always try to create and deploy applications in templates which are flexible and easy to maintain so that users of the websites can work freely, without facing any problems in the template for more details go to www.impacts-audio.com. While creating website templates, web developers must keep in mind that the adulterate of infrastructure should be as low as possible. The website template developers sometimes face the problem of calculating combination of business logic and presentation logic. To overcome the problem, they take assistance from some software engineering concepts.

Free Website Templates

Everyone does not have willingness to hire web developers for building a website. They tend to do it themselves. Good news is, one does not need to be an expert on this field to attach a website template with his / her URL. The basic knowledge of internet and web developing is enough. It is rather easy to create a website these days because there are thousands of web templates available. For more details go to www.oversightsystem.com. Good thing is, there are a lot of free website templates available as well. One needs to select the best one from the list and use it. Nothing else is required. Moreover, most of the free website templates come with free licensing. It means, one does not have to worry about a thing. Additionally, website templates come with miniature advertisement and other presentations that make the website even more attractive.

Types of Website Templates

Free website templates can be found on fashion, sports, movies, business, corporate world, Blogging etc. One needs to search and find the best possible match. Free templates can be easily customized and used as a basis for anyone's website.

Copyright Issues

One has to be aware of the copyright laws regarding free website templates. Normally, the designers want the users to retain their names at the bottom of the website. Some designers however allow the users to attach the templates without any terms and conditions.

1.6 WEB TYPOGRAPHY

In layman terms, web typography can be explained as process of using different fonts on the Web. On the other way around, the web typography is a beautiful art of designing and arranging different types of web fonts, letters, words and paragraphs used to create great web design as per priorities. A good typography of web mainly helps a designer to establish effective visual hierarchy and provides good visual punctuation as well as graphic accents. Web typography helps online reader to connect text with images without any hassle.

It is a well known fact that TEXT plays very significant role in the web designing. Interestingly, in a well designed website, more and more online visitors are looking for text rather than color, images, graphics or sound. In fact, web fonts allow all the web designers to use different fonts which are not installed on the end user's computer system effectively. Web typography is an important aspect of web designing. From past few years lots of advancement has been noticed in the typography of web.

There are some prominent areas by focusing on which a person can easily improve his web typography. The long list of improvement tips includes font, size, scale and hierarchy, vertical rhythm of line spacing, measurement, well-planned grid and alignment, white space, color and contrast. Apart from that, right usage techniques like CSS image replacement, Scalable Inman Flash replacement, cufón, Facelift Image Replacement etc. Last but not the least, the most advanced feature known as CSS3@font-face rule.

A good web typography really holds significant place in web designing and helps to create perfect web portal by good web developer. However, there are some key principals regarding web typography should be kept in mind by a web design discussed below:

- ✓ Strong horizontal movement
- ✓ Simple but strong typography
- ✓ Good management of white space
- ✓ Upsizing of typography
- ✓ Fun fonts add spontaneity

SUMMARY

As we saw that Web design is a process of conceptualizing, planning, and building a collection of electronic files that determine the layout, colours, text styles, structure, graphics, images, and use of interactive features that deliver pages to ones site visitors.

Web design encompasses many different skills and disciplines in the production and maintenance of websites. It also encompasses several different aspects, including webpage layout, content production, and graphic design.

REFERENCES

1. Web Design The complete Reference, Thomas Powell, Tata McGrawHill
2. HTML and XHTML The complete Reference, Thomas Powell, Tata McGrawHill
3. JavaScript 2.0 : The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider
4. PHP : The Complete Reference By Steven Holzner, Tata McGrawHill
5. www.w3schools.com
6. www.github.com
7. www.w3professors.com
8. XML: A Beginner's Guide by Steven Holzner
9. AJAX For Beginners , Ivan Bayross and Sharanam Shah, SPD
10. Web Development with jQuery (WROX) by Richard York
11. Learning PHP, MySQL & JavaScript with j Query, CSS & HTML5 – by Robin Nixon ,SPD

2

WEB TERMINOLOGIES AND BASIC HTML

Unit Structure

Internet

ISP

browser

URL

WWW

http

hypertext

HTML

HTML BASIC TAGS

Nesting Elements

Empty Elements

Anatomy of an HTML

Images Section

Marking up text Section

Headings Section

Paragraphs Section

Lists Section

LinksSection

References

The WWW is the most widely used part of the Internet. It is based on URL's and http, and allows dynamic links to an incredible amount of information.

INTERNET

A global network connecting millions computers. As of 1998, the Internet has more than 100 million users worldwide, and that number is growing rapidly. More than 100 countries are linked into exchanges of data, news and opinions. Unlike online services, which are centrally controlled, the Internet is decentralized by design. Each Internet computer, called a host, is independent. Its operators can choose which Internet services to use and which local services to make available to the global Internet community. Remarkably, this anarchy by design works exceedingly well. There are a

variety of ways to access the Internet. Most online services, such as America Online, offer access to some Internet services. It is also possible to gain access through a commercial Internet Service Provider (ISP). For most of its existence the Internet was primarily a research and academic network. More recently, commercial enterprises and a vast number of consumers have come to recognize the Internet's potential. Today people and businesses around the world can use the Internet to retrieve information, communicate and conduct business globally, and access a vast array of services and resources on-line.

ISP

Short for Internet Service Provider, a company that provides access to the Internet. For a monthly fee, the service provider gives you a software package, username, password and access phone number. Equipped with a modem, you can then log on to the Internet and browse the World Wide Web and USENET, and send and receive e-mail. In addition to serving individuals, ISPs also serve large companies, providing a direct connection from the company's networks to the Internet. ISPs themselves are connected to one another through Network Access Points (NAPs). ISPs are also called IAPs (Internet Access Providers).

browser

Short for Web browser, a software application used to locate and display Web pages. The two most popular browsers are Netscape Navigator and Microsoft Internet Explorer. Both of these are graphical browsers, which means that they can display graphics as well as text. In addition, most modern browsers can present multimedia information, including sound and video, though they require plug-ins for some formats.

URL

URLs make it possible to direct both people and software applications to a variety of information, available from a number of different Internet protocols. Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web. The first part of the address indicates what protocol to use, and the second part specifies the IP address or the domain name where the resource is located. protocol://domain_name.organization_type/full-path-of-file how://where/what As an exercise, let's look at this file's URL: http://www.netspace.org/users/dwb/url-guide.html The scheme for this URL is "http" for the HyperText Transfer Protocol. The Internet address of the machine is "www.netspace.org", and the path to the file is "users/dwb/www-authoring.html". When working with the WWW, most URLs will appear very similar to this one's overall structure. Note that when using FTP, HTTP, and Gopher URLs, the "full-path-of-file" will sometimes end in a slash. This indicates that the URL is pointing not to a specific file, but a directory. In this case, the server generally returns the "default index" of that directory. This

might be just a listing of the files available within that directory, or a default file that the server automatically looks for in the directory. With HTTP servers, this default index file is generally called "index.html", but is frequently seen as "homepage.html", "home.html", "welcome.html", or "default.html".

WWW

A system of Internet servers that support specially formatted documents. The documents are formatted in a language called HTML (HyperText Markup Language) that supports links to other documents, as well as graphics, audio, and video files. This means you can jump from one document to another simply by clicking on hot spots. Not all Internet servers are part of the World Wide Web. Short for World Wide Web Consortium, an international consortium of companies involved with the Internet and the Web. The W3C was founded in 1994 by Tim Berners-Lee, the original architect of the World Wide Web. The organization's purpose is to develop open standards so that the Web evolves in a single direction rather than being splintered among competing factions. The W3C is the chief standards body for HTTP and HTML.

http

Short for HyperText Transfer Protocol, the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page. HTTP is called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it. This is the main reason that it is difficult to implement Web sites that react intelligently to user input. This shortcoming of HTTP is being addressed in a number of new technologies, including ActiveX, Java, JavaScript and cookies. Currently, most Web browsers and servers support HTTP 1.1. One of the main features of HTTP 1.1 is that it supports persistent connections. This means that once a browser connects to a Web server, it can receive multiple files through the same connection. This should improve performance by as much as 20%.

hypertext

Hypertext simply means non linear text. A novel or magazine article is an example of linear text because it is meant to be read from beginning to end. Non linear communication is much harder to create because you must allow for the possibility of each reader accessing the material in a different order.

HTML

Short for HyperText Markup Language, the authoring language used to create documents on the World Wide Web. Hypertext, for easy navigation among resources (e.g. HyperText Markup Language or HTML, a standard

format for describing the structure of documents for transmission of hypermedia documents). HTML documents are ASCII files with embedded codes for logical markup, format (text styles, document titles, paragraphs, tables) and hyperlinks. Hypertext, for easy navigation among resources (e.g. HyperText Markup Language or HTML, a standard format for describing the structure of documents for transmission of hypermedia documents). HTML documents are ASCII files with embedded codes for logical markup, format (text styles, document titles, paragraphs, tables) and hyperlinks.

markup tags

The components of HTML.

web page

A document on the WWW. Every web page is identified by a unique URL (Uniform Resource Locator).

web site

A site (location) on the World Wide Web. Each Web site contains a home page, which is the first document users see when they enter the site. The site might also contain additional documents and files. Each site is owned and managed by an individual, company or organization.

home page

The main page of a Web site. Typically, the home page serves as an index or table of contents to other documents stored at the site.

HTML BASIC TAGS

HTML (Hypertext Markup Language) is the code that is used to structure a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables. As the title suggests, this article will give you a basic understanding of HTML and its functions.

HTML is not a programming language; it is a markup language that defines the structure of your content. HTML consists of a series of elements, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicize words, can make the font bigger or smaller, and so on. For example, take the following line of content:

My cat is very grumpy

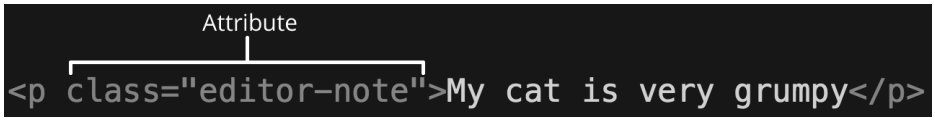
If we wanted the line to stand by itself, we could specify that it is a paragraph by enclosing it in paragraph tags:

My cat is very grumpy

The main parts of our element are as follows:

- ✓ The opening tag: This consists of the name of the element (in this case, p), wrapped in opening and closing angle brackets. This states where the element begins or starts to take effect — in this case where the paragraph begins.
- ✓ The closing tag: This is the same as the opening tag, except that it includes a forward slash before the element name. This states where the element ends — in this case where the paragraph ends. Failing to add a closing tag is one of the standard beginner errors and can lead to strange results.
- ✓ The content: This is the content of the element, which in this case is just text.
- ✓ The element: The opening tag, the closing tag, and the content together comprise the element.

Elements can also have attributes that look like the following:



Attributes contain extra information about the element that you don't want to appear in the actual content. Here, class is the attribute name, and editor-note is the attribute value. The class attribute allows you to give the element an identifier that can be used later to target the element with style information and other things.

An attribute should always have the following:

- ✓ A space between it and the element name (or the previous attribute, if the element already has one or more attributes).
- ✓ The attribute name, followed by an equals sign.
- ✓ Opening and closing quotation marks wrapped around the attribute value.

Nesting Elements

You can put elements inside other elements too — this is called nesting. If we wanted to state that our cat is very grumpy, we could wrap the word "very" in a element, which means that the word is to be strongly emphasized:

My cat is very grumpy.

You do however need to make sure that your elements are properly nested: in the example above, we opened the element first, then the element; therefore, we have to close the element first, then the element. The following is incorrect:

My cat is very grumpy.

The elements have to open and close correctly so that they are clearly inside or outside one another. If they overlap as shown above, then your web browser will try to make the best guess at what you were trying to say, which can lead to unexpected results. So don't do it!

Empty Elements

Some elements have no content and are called empty elements. Take the <img> element that we already have in our HTML page:

<img src="images/firefox-icon.png" alt="My test image">

This contains two attributes, but there is no closing </img> tag and no inner content. This is because an image element doesn't wrap content to affect it. Its purpose is to embed an image in the HTML page in the place it appears.

Anatomy of an HTML

That wraps up the basics of individual HTML elements, but they aren't handy on their own. Now we'll look at how individual elements are combined to form an entire HTML page. Let's revisit the code we put into our index.html example:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="My test image">
  </body>
</html>
```

Here, we have the following:

- ✓ <!DOCTYPE html> — The doctype. In the mists of time, when HTML was young (around 1991/92), doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML, which could mean automatic error checking and other useful things. However, these days no one cares about them, and they are just a historical artifact that needs to be included for everything to work right. For now, that's all you need to know.
- ✓ <html></html> — the <html> element. This element wraps all the content on the entire page and is sometimes known as the root element.

- ✓ <head></head> — the <head> element. This element acts as a container for all the stuff you want to include on the HTML page that isn't the content you are showing to your page's viewers. This includes things like keywords and a page description that you want to appear in search results, CSS to style our content, character set declarations, and more.
- ✓ <meta charset="utf-8"> — This element sets the character set your document should use to UTF-8, which includes most characters from the vast majority of written languages. Essentially, it can now handle any textual content you might put on it. There is no reason not to set this, and it can help avoid some problems later on.
- ✓ <title></title> — the <title> element. This sets the title of your page, which is the title that appears in the browser tab the page is loaded in. It is also used to describe the page when you bookmark/favourite it.
- ✓ <body></body> — the <body> element. This contains all the content that you want to show to web users when they visit your page, whether that's text, images, videos, games, playable audio tracks, or whatever else.

Images Section

Let's turn our attention to the <img> element again:

<img src="images/firefox-icon.png" alt="My test image">

This embeds an image into our page in the position it appears. It does this via the src (source) attribute, which contains the path to our image file. We have also included an alt (alternative) attribute. In this attribute, you specify descriptive text for users who cannot see the image, possibly because of the following reasons:

- ✓ They are visually impaired. Users with significant visual impairments often use tools called screen readers to read out the alt text to them.
- ✓ Something has gone wrong causing the image not to display. For example, try deliberately changing the path inside your src attribute to make it incorrect. If you save and reload the page, you should see something like this in place of the image:

The keywords for alt text are "descriptive text". The alt text you write should provide the reader with enough information to have a good idea of what the image conveys. In this example, our current text of "My test image" is no good at all. A much better alternative for our Firefox logo would be "The Firefox logo: a flaming fox surrounding the Earth."

Marking up text Section

This section will cover some of the essential HTML elements you'll use for marking up the text.

Headings Section

Heading elements allow you to specify that certain parts of your content are headings — or subheadings. In the same way that a book has the main title, chapter titles, and subtitles, an HTML document can too. HTML contains 6 heading levels, <h1>–<h6>, although you'll commonly only use 3 to 4 at most:

<h1>My main title</h1>
<h2>My top level heading</h2>
<h3>My subheading</h3>
<h4>My sub-subheading</h4>

Now try adding a suitable title to your HTML page just above your <img> element.

Paragraphs Section

As explained above, elements are for containing paragraphs of text; you'll use these frequently when marking up regular text content:

This is a single paragraph

Add your sample text (you should have it from What should your website look like?) into one or a few paragraphs, placed directly below your <img> element.

Lists Section

A lot of the web's content is lists, and HTML has special elements for these. Marking up lists always consist of at least 2 elements. The most common list types are ordered and unordered lists:

- ✓ Unordered lists are for lists where the order of the items doesn't matter, such as a shopping list. These are wrapped in a <ul> element.
- ✓ Ordered lists are for lists where the order of the items does matter, such as a recipe. These are wrapped in an <ol> element.
- ✓ Each item inside the lists is put inside an <li> (list item) element.

For example, if we wanted to turn the part of the following paragraph fragment into a list At Mozilla, we're a global community of technologists, thinkers, and builders working together ...

We could modify the markup to this

```
<p>At Mozilla, we're a global community of</p>
  <ul>
 <li>technologists</li>
 <li>thinkers</li>
 <li>builders</li>
</ul>
<p>working together ... </p>
```

LinksSection

Links are very important — they are what makes the web a web! To add a link, we need to use a simple element — <a> — "a" being the short form for "anchor". To make text within your paragraph into a link, follow these steps:

Choose some text. We chose the text "Mozilla Manifesto".

Wrap the text in an <a> element, as shown below:

<a>Mozilla Manifesto

Give the <a> element an href attribute, as shown below:

Mozilla Manifesto

Fill in the value of this attribute with the web address that you want the link to link to:

Mozilla Manifesto

You might get unexpected results if you omit the https:// or http:// part, called the protocol, at the beginning of the web address. After making a link, click it to make sure it is sending you where you wanted it to.

REFERENCES

1. Web Design The complete Reference, Thomas Powell, Tata McGrawHill
2. HTML and XHTML The complete Reference, Thomas Powell, Tata McGrawHill
3. JavaScript 2.0 : The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider
4. PHP : The Complete Reference By Steven Holzner, Tata McGrawHill
5. www.w3schools.com
6. www.github.com
7. www.w3professors.com
8. www.webster.edu
9. XML: A Beginner's Guide by Steven Holzner
10. AJAX For Beginners , Ivan Bayross and Sharanam Shah, SPD
11. Web Development with jQuery (WROX) by Richard York
12. Learning PHP, MySQL & JavaScript with j Query, CSS & HTML5 – by Robin Nixon, SPD

3

HTML, HTML FRAMES AND PUBLISHING AND MAINTAINING WEBSITES

Unit Structure

INTRODUCTION

HTML is the standard markup language for creating Web pages.

- ✓ HTML stands for Hyper Text Markup Language
- ✓ HTML describes the structure of Web pages using markup
- ✓ HTML elements are the building blocks of HTML pages
- ✓ HTML elements are represented by tags
- ✓ HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- ✓ Browsers do not display the HTML tags, but use them to render the content of the page

Sample of a simple HTML document

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Example Explained

- ✓ The <!DOCTYPE html> declaration defines this document to be HTML5
- ✓ The <html> element is the root element of an HTML page
- ✓ The <head> element contains meta information about the document
- ✓ The <title> element specifies a title for the document
- ✓ The <body> element contains the visible page content
- ✓ The <h1> element defines a large heading
- ✓ The <p> element defines a paragraph

At its heart, HTML is a fairly simple language made up of elements, which can be applied to pieces of text to give them different meaning in a document (Is it a paragraph? Is it a bulleted list? Is it part of a table?), structure a document into logical sections (Does it have a header? Three columns of content? A navigation menu?), and embed content such as images and videos into a page.`

Getting started with HTML

Covers the absolute basics of HTML, to get you started — we define elements, attributes, and other important terms, and show where they fit in the language. We also show how a typical HTML page is structured and how an HTML element is structured, and explain other important basic language features. Along the way, we'll play with some HTML to get you interested!

What's in the head? Metadata in HTML

The head of an HTML document is the part that is not displayed in the web browser when the page is loaded. It contains information such as the page <title>, links to CSS (if you want to style your HTML content with CSS), links to custom favicons, and metadata (data about the HTML, such as who wrote it, and important keywords that describe the document).

HTML text fundamentals

One of HTML's main jobs is to give text meaning (also known as semantics), so that the browser knows how to display it correctly. This article looks at how to use HTML to break up a block of text into a structure of headings and paragraphs, add emphasis/importance to words, create lists, and more.

Creating hyperlinks

Hyperlinks are really important — they are what makes the web a web. This article shows the syntax required to make a link, and discusses best practices for links.

Advanced text formatting

There are many other elements in HTML for formatting text that we didn't get to in the HTML text fundamentals article. The elements in here are less well-known, but still useful to know about. In this article you'll learn about marking up quotations, description lists, computer code and other related text, subscript and superscript, contact information, and more.

Document and website structure

As well as defining individual parts of your page (such as "a paragraph" or "an image"), HTML is also used to define areas of your website (such as "the header," "the navigation menu," or "the main content column.") This article looks into how to plan a basic website structure and how to write the HTML to represent this structure.

HTML FRAMES

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

Disadvantages of Frames

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages −

Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.

Sometimes your page will be displayed differently on different computers due to different screen resolution.

The browser's back button might not work as the user hopes.

There are still few browsers that do not support frame technology.

Creating Frames

To use frames on a page we use <frameset> tag instead of <body> tag. The <frameset> tag defines, how to divide the window into frames. The rows attribute of <frameset> tag defines horizontal frames and cols attribute defines vertical frames. Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

Example

Following is the example to create three horizontal frames −

```
<!DOCTYPE html>
<html>
```

```
  <head>
    <title>HTML Frames</title>
  </head>
  <frameset rows = "10%,80%,10%">
    <frame name = "top" src = "/html/top_frame.htm" />
    <frame name = "main" src = "/html/main_frame.htm" />
    <frame name = "bottom" src = "/html/bottom_frame.htm" />
    <noframes>
      <body>Your browser does not support frames.</body>
    </noframes>
  </frameset>
</html>
```

Example

Let's put the above example as follows, here we replaced rows attribute by cols and changed their width. This will create all the three frames vertically −

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Frames</title>
  </head>
  <frameset cols = "25%,50%,25%">
    <frame name = "left" src = "/html/top_frame.htm" />
    <frame name = "center" src = "/html/main_frame.htm" />
    <frame name = "right" src = "/html/bottom_frame.htm" />
    <noframes>
      <body>Your browser does not support frames.</body>
    </noframes>
  </frameset>
</html>
```

The <frameset> Tag Attributes

Following are important attributes of the **<frameset> tag −**

cols

- ✓ Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways −
- ✓ Absolute values in pixels. For example, to create three vertical frames, use cols = "100, 500, 100".
- ✓ A percentage of the browser window. For example, to create three vertical frames, use cols = "10%, 80%, 10%".
- ✓ Using a wildcard symbol. For example, to create three vertical frames, use cols = "10%, *, 10%". In this case wildcard takes remainder of the window.
- ✓ As relative widths of the browser window. For example, to create

three vertical frames, use cols = "3*, 2*, 1*". This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.

rows

- ✓ This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use rows = "10%, 90%". You can specify the height of each row in the same way as explained above for columns.

border

- ✓ This attribute specifies the width of the border of each frame in pixels. For example, border = "5". A value of zero means no border.

frameborder

- ✓ This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder = "0" specifies no border.

framespacing

- ✓ This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing = "10" means there should be 10 pixels spacing between each frames.

TAG <frame>

Following are the important attributes of <frame> tag –

src

- ✓ This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top_frame.htm" will load an HTML file available in html directory.

name

- ✓ This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.

frameborder

- ✓ This attribute specifies whether or not the borders of that frame

are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).

marginwidth

- ✓ This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth = "10".

marginheight

- ✓ This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight = "10".

noresize

- ✓ By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize = "noresize".

scrolling

- ✓ This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars.

longdesc

- ✓ This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc = "framedescription.htm"

Browser Support for Frames

If a user is using any old browser or any browser, which does not support frames then <noframes> element should be displayed to the user.

So you must place a <body> element inside the <noframes> element because the <frameset> element is supposed to replace the <body> element, but if a browser does not understand <frameset> element then it should understand what is inside the <body> element which is contained in a <noframes> element.

You can put some nice message for your user having old browsers. For example, Sorry!! your browser does not support frames. as shown in the above example.

Frame's name and target attributes

One of the most popular uses of frames is to place navigation bars in one frame and then load main pages into a separate frame.

Let's see following example where a test.htm file has following code –

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Target Frames</title>
  </head>
  <frameset cols = "200, *">
    <frame src = "/html/menu.htm" name = "menu_page" />
    <frame src = "/html/main.htm" name = "main_page" />
    <noframes>
      <body>Your browser does not support frames.</body>
    </noframes>
  </frameset>
</html>
```

Here, we have created two columns to fill with two frames. The first frame is 200 pixels wide and will contain the navigation menu bar implemented by menu.htm file. The second column fills in remaining space and will contain the main part of the page and it is implemented by main.htm file. For all the three links available in menu bar, we have mentioned target frame as main_page, so whenever you click any of the links in menu bar, available link will open in main page.

Following is the content of menu.htm file

```
<!DOCTYPE html>
<html>
  <body bgcolor = "#4a7d49">
    <a href = "http://www.google.com" target =
"main_page">Google</a>
    <br />
    <br />
    <a href = "http://www.microsoft.com" target =
"main_page">Microsoft</a>
    <br />
    <br />
    <a href = "http://news.bbc.co.uk" target = "main_page">BBC
News</a>
  </body>
</html>
```

Following is the content of main.htm file –

```
<!DOCTYPE html>
<html>
  <body bgcolor = "#b5dcb3">
    <h3>This is main page and content from any link will be displayed
here.</h3>
    <p>So now click any link and see the result.</p>
  </body>
</html>
```

The targetattribute can also take one of the following values −

_self

Loads the page into the current frame.

_blank

Loads a page into a new browser window. Opening a new window.

_parent

Loads the page into the parent window, which in the case of a single frameset is the main browser window.

_top

Loads the page into the browser window, replacing any current frames.

targetframe

Loads the page into a named targetframe.

PUBLISHING A WEBSITE

The process of publishing a website can be broken down into three simple steps: finding a Web hosting provider and registering your domain, setting up your website and publishing content.

Step 1: Finding a Web Hosting Plan

The kind of Web Hosting your site will need depends on certain considerations, such as its functionality, the amount of visitors you expect to receive, and how many pages it will comprise of.

Unless you expect your website to receive thousands of visitors each day or plan to create a website with high demands, you can most likely get by with a Web Hosting provider that offers either Shared Hosting, also known as Cheap Web Hosting.

With Shared Hosting, your site will share server resources with other websites on the same server as you. Because modern servers are incredibly powerful, multiple websites can reside on one server without creating performance issues.

Hosting providers monitor Shared Servers extensively. If a website is using too much resource from the server, providers take the necessary steps to ensure the stability of service. All providers aim to offer as close to 100% uptime as possible.

Using WordPress

A WordPress hosting plan – allows you to easily install the WordPress Content Management System (CMS) with a single click. Depending

on the complexity of your website idea, you may be able to simply select a theme defining the appearance of the website and start publishing content immediately.

WordPress is, for good reason, the most popular CMS in the world - it is flexible, user-friendly, and responsible for running over half of active websites on the Internet today.

Often, the Web Hosting company will even take responsibility for keeping the CMS, plugins and themes up to date - ensuring constant stability and security for your website.

Domain Registration

The Domain Name defines the address that people use to find your website - for example, the Domain Name of this site is webhostingsearch.com. Most Web Hosting companies offer to register a domain on your behalf when you buy a hosting package with them. It is usually simplest to register a Domain with your hosting company, as they will then automatically link the Domain Name to your web server.

If you prefer however, you can also purchase a Domain Name from a 'Domain Registrar'. To register your Domain this way, you will need to log in to your account on the Domain Registrar's website and link the Domain to your website's DNS address(es). These can usually be found in the email you get Our Domain Search tool can help you to decide on the best Domain Name for your website, check its availability, and help you to register it. Simply specify a few keywords, and it will provide suggestions based on those and similar words, including synonyms and related expressions.

Automatic Application Install

After creating your Web Hosting account and registering your Domain Name, you can log in to your hosting account. Most Web Hosting providers utilize control panels with intuitive graphical front-ends such as cPanel or Plesk.

Here you will be able to customize your hosting plan and install software applications, including Content Management Systems such as WordPress, Joomla or Drupal.

Step 2: Setting Up Your Website

Before you start, consider whether you want to create a dynamic website by using a Content Management System (CMS), or a static website by using a website builder or your own handwritten code.

In general, maintaining a large static website requires more effort than maintaining a dynamic website powered by a CMS. However, if you just want to create a small website with a few simple pages, a static

website may be worth considering. Although they lack the functionality provided by CMS plugins, they are generally quick and cheap to develop.

Web Design Software: Creating a 'Static' Website

If you know how to write in HTML, CSS and JavaScript, you can design your website using a text editor such as Notepad++ and preview your pages by opening the files in your browser. When you create a website in this manner, you are creating a 'static' website; each page is a unique file on your web server and is displayed exactly as you create it when someone visits its URL.

If creating a website this way is not an option for you, there are numerous WYSIWYG (What You See Is What You Get) Website Builders which enable you to create a website with all the necessary functions, even if you have no technical knowledge.

Content Management Systems (CMS)

Creating a website with a CMS frees you from designing each page from scratch. Content Management Systems store your website's information in a database and rely on a template to define its appearance and functionality. These templates use HTML and JavaScript with CSS, and usually every page keeps the same basic design and structure.

Installing a CMS onto your Web Hosting plan is usually a simple process, thanks to one-click installers built into modern hosting control panels. Once it is installed, you can easily add and modify the content, design and functionality of your website by logging into the CMS from any Internet browser.

If you want to change the appearance of your website, you can simply change to a new template or theme. Templates dictate the different types of pages available to you. These usually share some elements but differ in a crucial ways – for example the contact page, home page and article page will look different from each other due to their conflicting intentions. You can change templates and themes whenever you like without altering your website’s content or taking it offline.

Setting Up a Content Management System

Installing and configuring a CMS is a very simple process if your Web Hosting company offers automatic CMS installation.

After you install the CMS through your hosting company’s control panel, you will receive instructions via email explaining how to log in to the CMS. For example, with WordPress this is accomplished by adding “/wp-admin” to the end of your website’s URL.

After entering your password, you will see the main menu of your CMS. You will most likely want to install a template first, as the design and functionality of the site may dictate the content you place within it. On WordPress, you can choose a template by selecting the 'Themes' option under the 'Appearance' menu, and adapt it to your preferences by using the 'Customize' feature.

Once the theme is sorted, we recommend adding some useful plugins. All-in-one SEO for example, is very useful for analyzing your content and suggesting how to rank higher in search results. You can also find plugins which increase your website's security. When installing plugins, be careful to not add too many at once – occasionally plugins do not work well together, or significantly slow down your site.

Step 3: Publishing Content

Now you've got the Web Hosting plan set up and the design and functionality of your website decided upon, it's time for arguably the most important part; adding content. Without quality content, your site will not be found by your target audience, regardless of how good it looks.

Static Websites

If you opted to create your website using a WYSIWYG editor or your own handwritten code, your website will be available to the public as soon as you upload the files to the server. Although your Web Hosting provider likely has a file management solution for this purpose, these systems are often slow and unreliable, even sometimes disconnecting unexpectedly in the middle of large transfers.

Instead, we suggest using an open-source File Transfer Protocol (FTP) tool such as WinSCP or FileZilla, which connects directly to your server. If the connection is terminated, it will automatically try to resume it and continue your file transfer, saving you time and frustration.

Dynamic Websites

If you opted to create your website using a dynamic Content Management System such as WordPress, you can add new content by clicking the 'Add New' link under the 'Posts' or 'Pages' menu section on the Dashboard. Pages are for content that rarely changes, such as 'About Us' or 'Company History'. New pages will usually appear in your website's navigation bar, which can be further customized in the 'Menus' section.

If your website is a blog, most of your content will be Posts. Each Post that you create will be dated, and the front page of your website can be configured to display your most recent posts. Remember that blogs are not exclusively personal; many business websites have blog sections for news updates and other time-sensitive information, and

for letting search engines know that the site is being actively maintained – which can contribute to improving your ranking.

MAINTAINING A WEBSITE

Why is Website Maintenance Important?

With regular website maintenance your site will run smoothly. No disgruntled visitors because something on the site didn't work or a link you provided is broken.

Regular visitors are looking for what is new, so provide them with new and exciting information, products or features.

Website are subject to being hacked. Using a proper website maintenance program you can try and avoid being hacked by keeping everything up to date.

There are all kinds of things that need to be done when maintaining a website. Whether you decide to do these yourself or hire out the work, it still needs to be done.

Website Updates

Think about it, if you visit a website that is not updated regularly will you continue to visit it? Why should your own website or blog be like that then?

Website Content

Website content can include written text, images, free downloads, anything that is going to draw new visitors in and keep your existing visitors coming back. Here are some ideas:

- **Product Updates**

Product updates are really important if your site has an ecommerce element to it.

1. New products added announced.

   Don't forget to update your navigation, to add a page for the new product and add it to your site map.

   If you have a design that has a side navigation (called a sidebar in a blog) the updates can be added to it so no matter what page a visitor arrives on, they see the new product announcement.

2. Discontinuation of a product. Like the announcement of new products, if you are discontinuing a product, including an announcement in the side navigation will make the information available on all your pages.

3. Upcoming price changes would be another thing to announce. It might even spur some rush purchases too.

- **Company News**

Did you get a mention in the newpaper? Someone did a review of your site or product? This is all what is called social proof and you need to show visitors what others are saying about you.

Growing so much you need new staff? Announce and introduce your new staff members.

- **Giveaways**

Having a giveaway or a contest periodically is a good way to create some buzz about your site.

Updating old content should be on your website maintenance list also. Information becomes out dated so keep your content up to date to show your visitors you are on top of the subject.

Feature Addition

If at the time you created the website there was something that got left out because of budget constraints, maybe the budget can afford it now?

Take a look at your website/blog every once in a while and see if there is some kind of improvement that can be made. Something that you didn't think of previously.

Maybe you have some feed back from visitors that needs to be implemented?

- Fix a usability issue that has been mentioned.
- Time to add a blog if you have a regular website and do not have one already?
- Did you join a social networking site or two? Add the appropriate button(s) and links to your social profiles.
- Add a frequently asked question section to cut down on emails and phone calls aske these regular questions.

Regular Website Maintenance Tasks

There are some regular website maintenance tasks you should perform on a scheduled basis. Scheduling at least monthly would be the timeline to start with.

Backing Up Your Website

Backing up your website is something you should do all the time, especially if you are the type that uses the online interface of your store or blog to make changes. Things happen. Even though the web hosting company says they backup the sites on their servers, their last backup could have been before your last edit. If the server crashes for some reason or your site gets hacked, your edits will be gone if the web hosting company restores what they had backed up. Image

loosing a whole day's work, just because you didn't take a few minutes to backup the site.

Monitor Website Outages

If your site goes down, you want to be the first to know and not receive an email from someone else they can not access your site.

SiteUp is a small program that runs on your computer in the background checking your site on a regular basis. It will notify you when the site is down with a popup. Obviously though, your computer has to be on for it work.

Check Domain Registration Information

Look up in the WHOIS records what information is recorded for your domain name. Make sure it is correct. Sometimes when you initially sign up for your domain you would have used an email address that is no longer vaild. This needs to be updated as when there is a problem with your domain or an expiry notice is sent out you won't get the emails. They are sent to the email address on record.

Test Website Speed

Testing the download speed of your site regularly is important. Especially if you have added a new feature. Web surfer have a very short attention span. If your site is slow to load, they are not going to wait. You need to do everything you can to improve the download speed of your site or blog so visitors stay to read your content and hopefully provide you with organic incoming links by spreading the word for you what a wonderful site you have.

Link Check

Links become broken over time. With changes within the site and if you referenced someone in one of your articles or somewhere else within the site links could have changed or are broken.

The task to find broken links isn't too hard. Just use a link checker to test your external links and internal links at least once a month.

Software Updates

Third party software, like your ecommerce software, WordPress and Joomla for example, are always updating their software. You need to keep on top these updates and install them as soon as they come out. The updates won't just be new features, they will include security updates too.

Analyze Your Stats

Analyze not just your sales stats but your website stats too.

Traffic Stats

Look at your web server stats to determine your website traffic. If your web hosting account doesn't have website stats then get one installed. Something like Awstats that provides:

- Pages entered on and left on
- Time spent on the site
- Bounce rate
- Referring sites
- Countries your visitors are from
- Keywords/phrase that were used to find you

Google Analytics will provide some of this information. It may not be as complete as a website stats program that is run from your actual server.

One thing a website stats program installed on your server will do that Google Analytics doesn't is show you who is hotlinking (linking directly to your images on your site) . e.g. your images, PDFs, reports, etc.. These people are stealing your content and your bandwidth if they do not have your permission to do so. With this information your can stop the hotlink.

Search Engine Results

Are you showing up on the first page for the keywords/phrase you want to? If you have given it some time, e.g. a few months, to get onto the first page of the search results naturally then maybe it is time to look at your content and revise it.

Reputation Management

Using Google Alerts, you can monitor your website name, your name, your brand and your content on the web.

You will know who is talking about you. This gives you an opportunity to jump into the conversation. Thank those who are praising you. Fix a problem that is being discussed related to your business.

Tracking your website address with Google Alerts is 2 fold.

1. You see who is linking to you and can pop over there and say thanks.
2. You can catch the use of your content without your permission.

REFERENCES

1. Web Design The complete Reference, Thomas Powell, Tata McGrawHill
2. HTML and XHTML The complete Reference, Thomas Powell, Tata McGrawHill
3. JavaScript 2.0 : The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider
4. PHP : The Complete Reference By Steven Holzner, Tata McGrawHill
5. www.w3schools.com
6. www.github.com
7. www.w3professors.com
8. XML: A Beginner's Guide by Steven Holzner
9. AJAX For Beginners , Ivan Bayross and Sharanam Shah, SPD
10. Web Development with jQuery (WROX) by Richard York
11. Learning PHP, MySQL & JavaScript with j Query, CSS & HTML5 – by Robin Nixon, SPD

INTRODUCTION TO J.SCRIPT AND CLIENT SIDE REFERENCE

Unit Structure

INTRODUCTION TO JAVASCRIPT

JavaScript is a very powerful client-side scripting language. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage more lively and interactive, with the help of JavaScript. JavaScript is also being used widely in game development and Mobile application development.

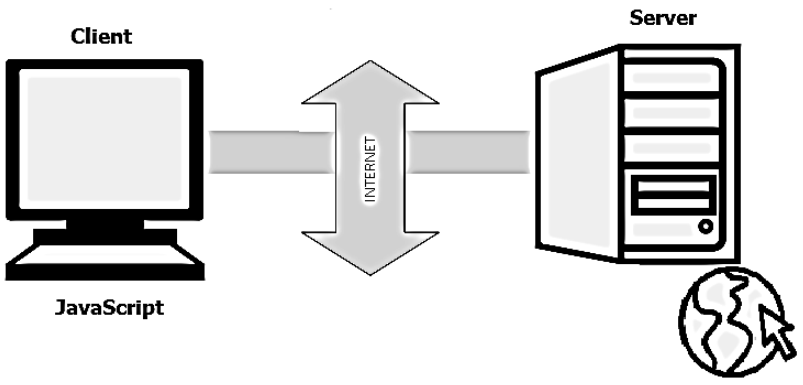


Figure 4.1- JaveScript and the WorldWideWeb

Javascript History

JavaScript was developed by Brendan Eich in 1995, which appeared in Netscape, a popular browser of that time.

The language was initially called LiveScript and was later renamed JavaScript. There are many programmers who think that JavaScript and Java are the same. In fact, JavaScript and Java are very much unrelated. Java is a very complex programming language whereas JavaScript is only a scripting language. The syntax of JavaScript is mostly influenced by the programming language C.

How to Run JavaScript?

Being a scripting language, JavaScript cannot run on its own. In fact, the browser is responsible for running JavaScript code. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it. The main advantage of JavaScript is that all modern web browsers support JavaScript. So, you do not have to worry about whether your site visitor uses Internet Explorer, Google Chrome, Firefox or any other browser. JavaScript will be supported. Also, JavaScript runs on any operating system including Windows, Linux or Mac. Thus, JavaScript overcomes the main disadvantages of VBScript (Now deprecated) which is limited to just IE and Windows.

Tools You Need

To start with, you need a text editor to write your code and a browser to display the web pages you develop. You can use a text editor of your choice including Notepad++, Visual Studio Code, Sublime Text, Atom or any other text editor you are comfortable with. You can use any web browser including Google Chrome, Firefox, Microsoft Edge, Internet Explorer etc.

A Simple JavaScript Program

You should place all your JavaScript code within <script> tags (<script> and </script>) if you are keeping your JavaScript code within the HTML document itself. This helps your browser distinguish your JavaScript code from the rest of the code. As there are other client-side scripting languages (Example: VBScript), it is highly recommended that you specify the scripting language you use. You have to use the type attribute within the <script> tag and set its value to text/javascript like this:

<script type="text/javascript">

Hello World Example:

```
<html>
<head>
        <title>My First JavaScript code!!!</title>
        <script type="text/javascript">
                alert("Hello World!");
        </script>
</head>
```

<body>
</body>
</html>

What can in-browser JavaScript do?

Modern JavaScript is a “safe” programming language. It does not provide low-level access to memory or CPU, because it was initially created for browsers which do not require it.

Javascript’s capabilities greatly depend on the environment it’s running in. For instance, Node.JS supports functions that allow JavaScript to read/write arbitrary files, perform network requests, etc.

In-browser JavaScript can do everything related to webpage manipulation, interaction with the user, and the webserver.

For instance, in-browser JavaScript is able to:

- ✓ Add new HTML to the page, change the existing content, modify styles.
- ✓ React to user actions, run on mouse clicks, pointer movements, key presses.
- ✓ Send requests over the network to remote servers, download and upload files (so-called AJAX and COMET technologies).
- ✓ Get and set cookies, ask questions to the visitor, show messages.
- ✓ Remember the data on the client-side (“local storage”).
- ✓ What CAN’T in-browser JavaScript do?
- ✓ JavaScript’s abilities in the browser are limited for the sake of the user’s safety. The aim is to prevent an evil webpage from accessing private information or harming the user’s data.

Examples of such restrictions include:

- ✓ JavaScript on a webpage may not read/write arbitrary files on the hard disk, copy them or execute programs. It has no direct access to OS system functions.
- ✓ Modern browsers allow it to work with files, but the access is limited and only provided if the user does certain actions, like “dropping” a file into a browser window or selecting it via an <input> tag.
- ✓ There are ways to interact with camera/microphone and other devices, but they require a user’s explicit permission. So a JavaScript-enabled page may not sneakily enable a web-camera, observe the surroundings and send the information to the NSA.
- ✓ Different tabs/windows generally do not know about each other. Sometimes they do, for example when one window uses JavaScript to open the other one. But even in this case, JavaScript from one page may not access the other if they come from different sites (from a different domain, protocol or port).

- ✓ This is called the “Same Origin Policy”. To work around that, both pages must contain a special JavaScript code that handles data exchange.
- ✓ This limitation is, again, for the user’s safety. A page from http://anysite.com which a user has opened must not be able to access another browser tab with the URL http://gmail.com and steal information from there.
- ✓ JavaScript can easily communicate over the net to the server where the current page came from. But its ability to receive data from other sites/domains is crippled. Though possible, it requires explicit agreement (expressed in HTTP headers) from the remote side. Once again, that’s a safety limitation.
- ✓ Such limits do not exist if JavaScript is used outside of the browser, for example on a server. Modern browsers also allow plugin/extensions which may ask for extended permissions.

Languages “over” JavaScript

- ✓ The syntax of JavaScript does not suit everyone’s needs. Different people want different features.
- ✓ That’s to be expected, because projects and requirements are different for everyone.
- ✓ So recently a plethora of new languages appeared, which are transpiled (converted) to JavaScript before they run in the browser.
- ✓ Modern tools make the transpilation very fast and transparent, actually allowing developers to code in another language and auto-converting it “under the hood”.

Examples of such languages:

- ✓ CoffeeScript is a “syntactic sugar” for JavaScript. It introduces shorter syntax, allowing us to write clearer and more precise code. Usually, Ruby devs like it.
- ✓ TypeScript is concentrated on adding “strict data typing” to simplify the development and support of complex systems. It is developed by Microsoft.
- ✓ Dart is a standalone language that has its own engine that runs in non-browser environments (like mobile apps). It was initially offered by Google as a replacement for JavaScript, but as of now, browsers require it to be transpiled to JavaScript just like the ones above.

There are more. Of course, even if we use one of these languages, we should also know JavaScript to really understand what we’re doing.

CLIENT SIDE REFERENCE (of JavaScript)

Client-Side JavaScript (CSJS) -- an extended version of JavaScript that enables the enhancement and manipulation of web pages and client browsers

Another possible use of JavaScript is for writing programs to perform arbitrary computations. You can write simple scripts, for example, that compute Fibonacci numbers, or search for primes. In the context of the Web and web browsers, however, a more interesting application of the language might be a program that computed the sales tax on an online order, based on information supplied by the user in an HTML form. As mentioned earlier, the real power of JavaScript lies in the browser and document-based objects that the language supports. To give you an idea of JavaScript's potential, the following sections list and explain the important capabilities of client-side JavaScript and the objects it supports.

Control Document Appearance and Content

The JavaScript Document object, through its write( ) method, which we have already seen, allows you to write arbitrary HTML into a document as the document is being parsed by the browser. For example, you can include the current date and time in a document or display different content on different platforms.

You can also use the Document object to generate documents entirely from scratch. Properties of the Document object allow you to specify colours for the document background, the text, and the hypertext links within it. This amounts to the ability to generate dynamic and conditional HTML documents, a technique that works particularly well in multi frame documents. Indeed, in some cases dynamic generation of frame content allows a JavaScript program to replace a traditional server-side script entirely.

Google Chrome and Internet Explorer 5 support proprietary techniques for producing Dynamic HTML effects that allow document content to be dynamically generated, moved, and altered. IE 4 also supports a complete DOM that gives JavaScript access to every single HTML element within a document. And IE 5.5 and Netscape 6 support the W3C DOM standard (or at least key portions of it), which defines a standard, portable way to access all of the elements and text within an HTML document and to position them and modify their appearance by manipulating their Cascading Style Sheets (CSS) style attributes. In these browsers, client-side JavaScript has complete power over document content, which opens an unlimited world of scripting possibilities.

Control the Browser

Several JavaScript objects allow control over the behavior of the browser. The Window object supports methods to pop up dialog boxes to display simple messages to the user and get simple input from the user. This object also defines a method to create and open (and close) entirely new browser windows, which can have any specified size and any combination of user controls. This allows you, for example, to open up multiple windows to give the user multiple views of your web site. New browser windows are also useful for temporary display of generated HTML, and, when created without

the menu bar and other user controls, these windows can serve as dialog boxes for more complex messages or user input.

JavaScript does not define methods that allow you to create and manipulate frames directly within a browser window. However, the ability to generate HTML dynamically allows you to programmatically write the HTML tags that create any desired frame layout.

JavaScript also allows control over which web pages are displayed in the browser. The Location object allows you to download and display the contents of any URL in any window or frame of the browser. The History object allows you to move forward and back within the user's browsing history, simulating the action of the browser's Forward and Back buttons.

Yet another method of the Window object allows JavaScript to display arbitrary messages to the user in the status line of any browser window.

Interact with HTML Forms

Another important aspect of client-side JavaScript is its ability to interact with HTML forms. This capability is provided by the Form object and the form element objects it can contain: Button, Checkbox, Hidden, Password, Radio, Reset, Select, Submit, Text, and Textarea objects. These element objects allow you to read and write the values of the input elements in the forms in a document. For example, an online catalog might use an HTML form to allow the user to enter his order and could use JavaScript to read the input from that form in order to compute the cost of the order, the sales tax, and the shipping charge. JavaScript programs like this are, in fact, very common on the Web. We'll see a program shortly that uses an HTML form and JavaScript to allow the user to compute monthly payments on a home mortgage or other loan. JavaScript has an obvious advantage over server-based scripts for applications like these: JavaScript code is executed on the client, so the form's contents don't have to be sent to the server in order for relatively simple computations to be performed.

Another common use of client-side JavaScript with forms is for validating form data before it is submitted. If client-side JavaScript is able to perform all necessary error checking of a user's input, no round trip to the server is required to detect and inform the user of trivial input errors. Client-side JavaScript can also perform preprocessing of input data, which can reduce the amount of data that must be transmitted to the server. In some cases, client-side JavaScript can eliminate the need for scripts on the server altogether! (On the other hand, JavaScript and server-side scripting do work well together. For example, a server-side program can dynamically create JavaScript code on the fly, just as it dynamically creates HTML.)

Interact with the User

An important feature of JavaScript is the ability to define event handlers -- arbitrary pieces of code to be executed when a particular event occurs. Usually, these events are initiated by the user, when, for example, she moves the mouse over a hypertext link, enters a value in a form, or clicks

the Submit button in a form. This event-handling capability is a crucial one, because programming with graphical interfaces, such as HTML forms, inherently requires an event-driven model. JavaScript can trigger any kind of action in response to user events. Typical examples might be to display a special message in the status line when the user positions the mouse over a hypertext link or to pop up a confirmation dialog box when the user submits an important form.

Read and Write Client State with Cookies

A cookie is a small amount of state data stored permanently or temporarily by the client. Cookies may be transmitted along with a web page by the server to the client, which stores them locally. When the client later requests the same or a related web page, it passes the relevant cookies back to the server, which can use their values to alter the content it sends back to the client. Cookies allow a web page or web site to remember things about the client -- for example, that the user has previously visited the site, has already registered and obtained a password, or has expressed a preference about the color and layout of web pages. Cookies help you provide the state information that is missing from the stateless HTTP protocol of the Web.

When cookies were invented, they were intended for use exclusively by server-side scripts; although stored on the client, they could be read or written only by the server. JavaScript changed this, because JavaScript programs can read and write cookie values and can dynamically generate document content based on the value of cookies.

Still More Features

In addition to the features I have already discussed, JavaScript has many other capabilities, including the following:

JavaScript can change the image displayed by an <img> tag to produce image rollover and animation effects.

JavaScript can interact with Java applets and other embedded objects that appear in the browser. JavaScript code can read and write the properties of these applets and objects and can also invoke any methods they define. This feature truly allows JavaScript to script Java.

As mentioned at the start of this section, JavaScript can perform arbitrary computation. JavaScript has a floating-point data type, arithmetic operators that work with it, and a full complement of standard floating-point mathematical functions.

The JavaScript Date object simplifies the process of computing and working with dates and times.

The Document object supports a property that specifies the last-modified date for the current document. You can use it to automatically display a timestamp on any document.

JavaScript has a window.setTimeout( ) method that allows a block of arbitrary JavaScript code to be executed some number of milliseconds in

the future. This is useful for building delays or repetitive actions into a JavaScript program. In JavaScript 1.2, setTimeout( ) is augmented by another useful method called setInterval( ) .

The Navigator object (named after the Netscape web browser, of course) has variables that specify the name and version of the browser that is running, as well as variables that identify the platform on which it is running. These variables allow scripts to customize their behavior based on browser or platform, so that they can take advantage of extra capabilities supported by some versions or work around bugs that exist on some platforms.

In client-side JavaScript 1.2, the Screen object provides information about the size and color-depth of the monitor on which the web browser is being displayed.

As of JavaScript 1.1, the scroll( ) method of the Window object allows JavaScript programs to scroll windows in the X and Y dimensions. In JavaScript 1.2, this method is augmented by a host of others that allow browser windows to be moved and resized.

What JavaScript Can't Do

Client-side JavaScript has an impressive list of capabilities. Note, however, that they are confined to browser- and document-related tasks. Since client-side JavaScript is used in a limited context, it does not have features that would be required for standalone languages:

JavaScript does not have any graphics capabilities, except for the powerful ability to dynamically generate HTML (including images, tables, frames, forms, fonts, etc.) for the browser to display.

For security reasons, client-side JavaScript does not allow the reading or writing of files. Obviously, you wouldn't want to allow an untrusted program from any random web site to run on your computer and rearrange your files!

JavaScript does not support networking of any kind, except that it can cause the browser to download arbitrary URLs and it can send the contents of HTML forms across the network to server-side scripts and email addresses.

SUMMARY

JavaScript was initially created as a browser-only language, but is now used in many other environments as well.

Today, JavaScript has a unique position as the most widely-adopted browser language with full integration with HTML/CSS.

There are many languages that get "transpired" to JavaScript and provide certain features. It is recommended to take a look at them, at least briefly, after mastering JavaScript.

REFERENCES

1. Web Design The complete Reference, Thomas Powell, Tata McGrawHill
2. HTML and XHTML The complete Reference, Thomas Powell, Tata McGrawHill
3. JavaScript 2.0 : The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider
4. PHP : The Complete Reference By Steven Holzner, Tata McGrawHill
5. www.w3schools.com
6. www.github.com
7. www.w3professors.com
8. XML: A Beginner's Guide by Steven Holzner
9. AJAX For Beginners , Ivan Bayross and Sharanam Shah, SPD
10. Web Development with jQuery (WROX) by Richard York
11. Learning PHP, MySQL & JavaScript with j Query, CSS & HTML5 – by Robin Nixon, SPD

5

INTRODUCTION TO XML

Unit Structure

INTRODUCTION

- ✓ XML stands for EXtensible Markup Language
- ✓ XML is a markup language much like HTML.
- ✓ XML was designed to describe data.
- ✓ XML tags are not predefined in XML. You must define your own tags.
- ✓ XML is self describing.
- ✓ XML uses a DTD (Document Type Definition) to formally describe the data.

The main difference between XML and HTML

XML is not a replacement for HTML. XML and HTML were designed with different goals:

XML was designed to describe data and to focus on what data is. HTML was designed to display data and to focus on how data looks.

HTML is about displaying information, XML is about describing information.

XML is extensible

The tags used to markup HTML documents and the structure of HTML documents are predefined. The author of HTML documents can only use tags that are defined in the HTML standard.

XML allows the author to define his own tags and his own document structure.

XML is a complement to HTML

It is important to understand that XML is not a replacement for HTML. In the future development of the Web it is most likely that XML will be used to structure and describe the Web data, while HTML will be used to format and display the same data.

XML in future Web development

We have been participating in XML development since its creation. It has been amazing to see how quickly the XML standard has been developed, and how quickly a large number of software vendors have adopted the standard.

We strongly believe that XML will be as important to the future of the Web as HTML has been to the foundation of the Web. XML is the future for all data transmission and data manipulation over the Web.

What is XML?

XML is a general purpose mechanism for describing hierarchical data. Data items are contained within elements and attributes. Elements can contain textual data, attributes and other elements, attributes can just contain textual data.

Whenever there is a need to store complex data, whether it is for passing between systems or storing in a file, the data must be marked up in some way. When it is read back in, it needs to be possible to tell were one record ends and another begins, and which record is contained within another. It was this gap that XML filled.

Whenever there is a need to store complex data, whether it is for passing between systems or storing in a file, the data must be marked up in some way. When it is read back in, it needs to be possible to tell were one record ends and another begins, and which record is contained within another. It was this gap that XML filled.

Prior to XML, developers would produce their own proprietary formats for storing data, with varying levels of success. It's not difficult to conceive of your own system for representing data, but you would also have to write your own custom parser and serializer which may be a complex and error prone task. XML provides a general mechanism for marking up and representing data. As XML is now a wide spread standard, various other tools and technologies are available on most platforms. Using XML will make it possible for you to take advantage of may other XML aware technologies such as XPATH, DTD, XSLT, XSD, XQUERY, and XML Data Binding.

One complaint about XML is that its verbose, and consumes to much space.

XML EXAMPLE

```
<?xml version="1.0" encoding="UTF-8" ?>
<book isbn="0123-456-789">
   <title>The Lion The Witch and the Wardrobe</title>
</book>
```

In the example "book" and "title" are elements, and "isbn" is an attribute. The first line is the XML header which defines the character encoding.

Well Formed XML (Valid XML)

Well formed means that all the tags match up. In the example above we opened a "book" element on line 1, and closed it on line 3. Without line 3, the "book" element is not closed, and thus the document is not well formed. Also an attribute must have a closing quote (note the standard allows either single or double quotes to be using for attributes, as long as they are the same at the start and the end of the attribute, i.e. isbn="...." or isbn='....' but no isbn="....'

Escaping control characters

If the textual data within and element needs to contain a < or > character, then this must be escaped in order to prevent it being confused with an element marker. This is done by replacing it with the literal < or > respectively. This introduces a problem when trying to use the & character, so this is escaped using the & literal. Similarly in an attribute, the quote " character must be escaped using " and ' using '

& (& or "ampersand")
< (< or "less than")
> (> or "greater than")
' (' or "apostrophe")
" (" or "quotation mark")

International Support

The first line of an XML Schema is the XML header. This has an optional "encoding" field which describes how the rest of the document should be interpreted (i.e. how to map the data in the file into characters). The standard formats when dealing with international characters are utf-8 and utf-16,

character encoding is a whole separate topic, but if you are using an XML Editor (like Liquid Studio, then this is all taken care of automatically).

Reading XML Data Programmatically

DOM (Document Object Model)

A DOM parser is an XML parser that reads XML data and stores it in a set of objects, these objects can then be examined and the data extracted from them. The structure of the DOM objects is standardized by the W3C, so the code for reading XML is more or less standard across multiple platforms.

One of the drawbacks of a DOM parser is that the whole XML document has to be read into its object form (and thus into system memory) meaning it is not possible to deal with very large XML files.

DOM parsers exist for all the main platforms and languages, and are typically built into the platforms core framework.

SAX (Simple API for XML)

A SAX parser is more primitive XML reader. For every entity it reads in the XML data, it fires an event (or callback) which the consuming application must deal with or ignore. This basic interface makes it possible to cope with arbitrarily large XML files, as the consuming application need only store its current state, discarding information that has already been read, when it's no longer needed. It does however complicate the handling of the data, as the application must keep track of its state (i.e. position in the XML document).

Typically this technique is combined with a DOM parser. The application keeps track of its position in the XML until it comes across a chunk of data it needs to deal with, it then constructs a DOM tree based on a small section of the whole XML document, processes the DOM tree and discards it before moving on.

XML Data Binding

XML Data Binding is similar to the DOM mechanism. The XML document is read into a set of objects, however instead of being read into a set of general purpose DOM objects, its read into a set of classes generated specifically to deal with the type of XML document being read. These classes are generated using an XML Schema which knows about the shape of valid XML documents.

This mechanism makes it much easier for developers to work with XML data, as they are dealing with strongly typed objects (i.e. they have names and properties that reflect the elements and attributes in the XML data).

This technique must also read the entire XML Document into memory (as with DOM), but this restriction can be worked around, see Liquid XML Data Binding - dealing with large files.

There are a number of tools for XML Data Binding, Liquid Studio provides a solution that generates classes for C#, C++, Java, VB.Net & Visual Basic.

XML Schemas

An XML Schema formally specifies the structure of an XML document. This has a number of uses:

Validation - The XML Schema can be used to validate an XML document, to ensure that it contains all the correct data in the correct places.

Interoperability - Because the shape of the XML Document is described formally there is no ambiguity, meaning that each team working with a given XML Schema know what the resulting XML document must look like, there are no ambiguous specifications to work from.

Code Generation - The XML Schema can be used to generate code that will allow developers to read and write XML data using strongly typed classes. Meaning developers just have to work with simple objects with strongly typed properties. This technique is known as XML Data Binding. Liquid Studio provides XML Data Binding for C#, C++, Java, VB.Net & Visual Basic.

Visualizations - It is possible to show the structure of an XML Schema Graphically making it easier for developers to understand.

Documentation - An XML Schema can contain documentation, which can be generated into a convenient human readable form, see XML Schema Standards Library.

There are a number of mechanisms for describing an XML Schema.

DTD (Document Type Definition) - the original standard, defined within the W3C's XML standard. The DTD standard is all but obsolete now, replaced by the W3C's XSD standard. DTD's have their own format, can define substitutions internally within themselves, requiring multiple parses to extract the normalised document. They were also quite limited, allowing course validation, and minimal re-use.

XDR (XML-Data Reduced) - a standard developed by Microsoft that bridged the gap between DTD, and XSD schemas. A parser was implemented in MSXML up to version 6 when it was dropped. It was also used to describe data in older versions of Biz Talk. The document was described in terms of XML, and was very simplistic, offering minimal validation or reuse, but was simple to parse and extensible.

XSD (XML Schema Definition) - ratified by the W3C, it is now the de facto mechanism of describing XML documents. It allows for complex validation, re-use via inheritance and type creation, is described in terms of XML, so is easy to parse, and has support on most platforms. Almost all major data standards are now described in terms of XSDs.

RELAX NG (REgular LAnguage for XML Next Generation) - RELAX NG is relatively simple structure, and shares many features with the W3C XSD standard, data typing, regular expression support, namespace support,

ability to reference complex definitions. Open source parsers exist on most platforms, but it is not widely used.

XML SYNTAX

For XML to be well-formed it must obey the following syntax rules.

- ✓ XML is case sensitive so start and end tag names must match exactly.
- ✓ For example, the following start and end tags have a mismatch in case
- ✓ and so are not well-formed,
  <CARS>
  </Cars>

The end-tag name needs to have all capital letters, i.e. </CARS>, in order for it to match the start tag.

- ✓ No spaces are allowed between the < and the tag name.
- ✓ Tag names must begin with an alpha character, and contain only alphanumeric characters.
- ✓ An element must have an open and closing tag unless it is empty.
- ✓ An empty element that does not have a closing tag must be of the
- ✓ form < . . . / >. For example, <nan/>.
- ✓ Tags must nest properly. That is, when one element contains another
- ✓ element then the start and end tags of the inner element must be
- ✓ between the start and end tags of the parent element. For example,
  <CARS>
  <CYL> 6 </CYL>
  </CARS>
- ✓ Here the CYL tag is nested within the CARS tag. Note the use of indentation makes it easier to see the nesting.
- ✓ All attribute values must appear in quotes in a name = "value" format.

  <dim size="2"/>. This example shows an empty tag with a size attribute of “2”.
- ✓ Isolated markup characters are not allowed in text. However, they may be specified via entity references. For example, the < is specified by the entity reference < and the > symbol is >.
- ✓ In addition to element tags, XML has markup for comments, which is information not shown to the user; processing instructions, which is similar to code meant for the processor; and character data that is not to be processed but simply passed straight through to the user. Comments must appear between <!−− and −− >.

  For example,
  <!-- This is a comment which is so long
  that it apears on three lines of the

document before it ends with two - followed by >. -->

- ✓ It is possible to include in a document character data that is not processed and so the > is ignored. The character data must appear between

  <![CDATA[and]]>.

  <![CDATA[This is character data that can have any special character in it such as < or > or & and not have to worry about it being interpreted as a special character by the processor.]]>

- ✓ Finally, processing instructions must appear between <? and ?>. For example, an XML document must start with the processing instruction that identifies it as an xml document and provides the XML version number as an attribute,

  <?xml version = "1.0" ?>

VALID XML

The rules provided in the previous section are just syntax rules for insuring that an XML document is well-formed. But we typically want to have documents that are more than well-formed; we want to include application specific structure in the markup. For example, with geographic data we may want tags for locations, x and y coordinates, city names, etc. Tags for these entities are specified through a set of Document Type Definitions (DTD for short) or schema. With a DTD we can: provide the name of a valid element; limit the content of an element to character data, specific other elements, or to be empty; and specify the attributes that are required or allowed in the tag.

Well-formed XML obeys XML syntax rules described in the previous section, but valid XML, in addition to being well-formed, obeys a specified DTD. The DTD may appear within the document itself or be provided via reference,

<!DOCTYPE dataset SYSTEM "../DataSetByRecord.dtd">

Here we specify a DTD to use via a document type declaration. The dataset gives the root element name that the DTD will be applied to in the verification process. At times we may want to use more than one DTD. We can do this through name spaces. That is, each DTD is given a name and the DTD name is prepended to the appropriate tag names and attribute names. For example, the object element below lists three name spaces, r, c, and bioc, and the URL's where their respective DTDs can be found.

<object xmlns:r="http://www.r-project.org"
xmlns:c="http://www.c.org"
xmlns:bioc="http://www.bioconductor.org"
type="R-pop-environment" hidden="true">

We specify which name space a tag uses within the object element as follows:
<r:code>
x <- rep(23, 2)

```
</r:code>
<c:code>
x+
</c:code>
```

XML PARSER

A parser has the job of reading the XML, checking it for errors, and passing it on to the intended application. If no DTD or schema is provided, the parser simply checks that the XML is well-formed. If a DTD is provided then the parser also determines whether the XML is valid, i.e. that the tags, attributes, and content meet the specifications found in the DTD, before passing it on to the application. Models for parsing XML are described in greater detail in sections ??.

XHTML

Some readers will have thought of HTML when we mentioned markup and meta-information. After all, what is the difference between HTML and XML? We can add meta information to HTML documents using the META tag, but this is not exactly what we mean by meta-information. A little thought and familiarity with HTML will quickly bring us to problems. HTML has a fixed set of markup elements, e.g. H1, H2, a, img, B, and so on. It doesn't even have a NUMBER or REAL markup for representing real numbers.

Hopefully it is evident at this point that an important role of XML is to separate out information (content) from the structure and format. The markup provides the structure of the content, and the the format determines how the content is to be rendered for viewing by the user. As a simple example, an array of numbers that corresponds to the miles per gallon of various makes of cars may be provided via XML as follows:

```
<array name="MPG" size="7" type="numeric">
<e>21.0</e> <e>21.0</e> <e>22.8</e> <e>21.4</e>
<e>18.7</e> <e>18.1</e> <e>14.3</e>
</array>
```

The content consists of the set of values 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, and 14.3. The structure provided via the markup tells us that the content forms an array of numbers of length 7, and the array is named MPG. HTML does not make the same division between content, structure, and format. Many tags describe how to format content, but provide no information about the type of content. For example, the HTML tags B for bold face, br for line break, and hr for horizontal rule are all instructions for the visual rendering of content. HTML has been extended to XHTML by requiring all tags to be lower case, all elements to be properly closed with end tags, and attribute values to appear between quotes. Although these rules mean that we can require XHTML documents to be well-formed and valid, XHTML is not up to the job of describing complex structures such as factors, data frames, S objects, and so on. Clearly XHTML is lacking in this regard. With XML we can define much richer application-specific markup.

To drive home the difference between format and content, consider the numbers in the above example: 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, and 14.3. They can be represented as a text list, (21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3), as a summary statistic: 19.6, as a stem-and-leaf plot,

14 | 3
16 |
18 | 17
20 | 004
22 | 8

or in a graphic such as a histogram. The conversion of the content into one of these formats occurs when the XML is processed for viewing. The eXtensible Stylesheet Language (XSL) contains instructions for formatting the XML. (XSL is itself an XML document that provides a template describing how to view the document.) The XML document together with the XSL stylesheet are processed by an XSLT processor to create the data display. The content remains the same as shown in the XML document, but with a processor the format is changed for the data view.

REFERENCES

1. Web Design The complete Reference, Thomas Powell, Tata McGrawHill
2. HTML and XHTML The complete Reference, Thomas Powell, Tata McGrawHill
3. JavaScript 2.0 : The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider
4. PHP : The Complete Reference By Steven Holzner, Tata McGrawHill
5. www.w3schools.com
6. www.github.com
7. www.w3professors.com
8. XML: A Beginner's Guide by Steven Holzner
9. AJAX For Beginners , Ivan Bayross and Sharanam Shah, SPD
10. Web Development with jQuery (WROX) by Richard York
11. Learning PHP, MySQL & JavaScript with j Query, CSS & HTML5 – by Robin Nixon, SPD

6

INTRODUCING AJAX AND USING AJAX AND PHP

Unit Structure

INTRODUCTION

AJAX is the acronym for Asynchronous JavaScript & XML. It is a technology that reduces

the interactions between the server and client. It does this by updating only part of a web page rather than the whole page. The asynchronous interactions are initiated by JavaScript.

JavaScript is a client side scripting language. It is executed on the client side by the web browsers that support JavaScript.JavaScript code only works in browsers that have JavaScript enabled.

XML is the acronym for Extensible Markup Language. It is used to encode messages in both human and machine-readable formats. It's like HTML but allows you to create your custom tags. For more details on XML, see the article on XML

AJAX is based on internet standards, and uses a combination of :

- ✓ XMLHttpRequest object (to exchange data asynchronously with a server)
- ✓ JavaScript/DOM (to display/interact with the information)
- ✓ CSS (to style the data)
- ✓ XML (often used as the format for transferring data)

Using AJAX

It allows developing rich interactive web applications just like desktop applications. Validation can be performed done as the user fills in a form without submitting it. This can be achieved using auto completion. The words that the user types in are submitted to the server for processing. The server responds with keywords that match what the user entered. It can be used to populate a dropdown box depending on the value of another dropdown box. Data can be retrieved from the server and only a certain part of a page updated without loading the whole page. This is very useful for web page parts that load things like

- ✓ Tweets
- ✓ Comments
- ✓ Users visiting the site etc.

PHP

- ✓ PHP stands for HyperText Preprocessor
- ✓ PHP is an interpreted language, i.e. there is no need for compilation.
- ✓ PHP is a server side scripting language.
- ✓ PHP is faster than other scripting language e.g. asp and jsp.

  PHP is an open source, interpreted and object-oriented scripting language i.e. executed at server side. It is used to develop web applications (an application i.e. executed at server side and generates dynamic page).
- ✓ PHP is a server side scripting language.
- ✓ PHP is an interpreted language, i.e. there is no need for compilation.
- ✓ PHP is an object-oriented language.
- ✓ PHP is an open-source scripting language.
- ✓ PHP is simple and easy to learn language.

PHP Features

There are given many features of PHP.

- ✓ Performance: Script written in PHP executes much faster then those scripts written in other languages such as JSP & ASP.
- ✓ Open Source Software: PHP source code is free available on the web, you can developed all the version of PHP according to your requirement without paying any cost.
- ✓ Platform Independent: PHP are available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
- ✓ Compatibility: PHP is compatible with almost all local servers used today like Apache, IIS etc.

- ✓ Embedded: PHP code can be easily embedded within HTML tags and script.

How To Install PHP

To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:

- ✓ WAMP for Windows
- ✓ LAMP for Linux
- ✓ MAMP for Mac
- ✓ SAMP for Solaris
- ✓ FAMP for FreeBSD
- ✓ XAMPP (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, OpenSSL, Mercury Mail etc.

If you are on Windows and don't want Perl and other features of XAMPP, you should go for WAMP. In a similar way, you may use LAMP for Linux and MAMP for Macintosh.

Example

```
<!DOCTYPE>
<html>
<body>
  <?php
    echo "<h2>Welcome to W3professors.com</h2>";
  ?>
</body>
</html>
```

PHP is widely used in web development now a days. Dynamic websites can be easily developed by PHP. But you must have the basic the knowledge of following technologies for web development as well.

- ✓ HTML
- ✓ CSS
- ✓ BOOTSTRAP
- ✓ JavaScript
- ✓ AJAX
- ✓ XML and JSON
- ✓ JQuery

CREATING AN PHP AJAX APPLICATION

We will create a simple application that allows users to search for popular PHP MVC frameworks. Our application will have a text box that users will

type in the names of the framework. We will then use mvc AJAX to search for a match then display the framework's complete name just below the search form.

Step 1) Creating the index page

Index.php

```
<html>
   <head>
      <title>PHP MVC Frameworks - Search Engine</title>
      <script type="text/javascript" src="/auto_complete.js"></script>
   </head>
   <body>
      <h2>PHP MVC Frameworks - Search Engine</h2>
      <p><b>Type the first letter of the PHP MVC Framework</b></p>
      <form method="POST" action="index.php">
         <p><input type="text" size="40" id="txtHint"
onkeyup="showName(this.value)"></p>
      </form>
      <p>Matches: <span id="txtName"></span></p>
   </body>
</html>
```

Here,

"onkeyup="showName(this.value)"" executes the JavaScript function showName everytime a key is typed in the textbox.

This feature is called auto complete

Step 2) Creating the frameworks page

frameworks.php

```
<?php
$frameworks = array("CodeIgniter","Zend Framework","Cake
PHP","Kohana") ;
$name = $_GET["name"];
if (strlen($name) > 0) {
   $match = "";
   for ($i = 0; $i < count($frameworks); $i++) {
      if (strtolower($name) == strtolower(substr($frameworks[$i], 0,
strlen($name)))) {
         if ($match == "") {
            $match = $frameworks[$i];
         } else {
            $match = $match . " , " . $frameworks[$i];
         }
      }
   }
}
echo ($match == "") ? 'no match found' : $match;
?>
```

Step 3) Creating the JS script

auto_complete.js

```
<script>
function showName(str){
   if (str.length == 0){ //exit function if nothing has been typed in the textbox
      document.getElementById("txtName").innerHTML=""; //clear previous
results
      return;
   }
   if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera,
Safari
      xmlhttp=new XMLHttpRequest();
   } else {// code for IE6, IE5
      xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
   xmlhttp.onreadystatechange=function() {
      if (xmlhttp.readyState == 4 && xmlhttp.status == 200){
document.getElementById("txtName").innerHTML=xmlhttp.responseText;
      }
   }
   xmlhttp.open("GET","frameworks.php?name="+str,true);
   xmlhttp.send();
}
</script>
```

Here,

“if (str.length == 0)” check the length of the string. If it is 0, then the rest of the script is not executed.

“if (window.XMLHttpRequest)…” Internet Explorer versions 5 and 6 use ActiveXObject for AJAX implementation. Other versions and browsers such as Chrome, FireFox use XMLHttpRequest. This code will ensure that our application works in both IE 5 & 6 and other high versions of IE and browsers.

“xmlhttp.onreadystatechange=function…” checks if the AJAX interaction is complete and the status is 200 then updates the txtName span with the returned results.

Step 4) Testing our PHP Ajax application

Assuming you have saved the file index.php In phututs/ajax, browse to the URL http://localhost/phptuts/ajax/index.php

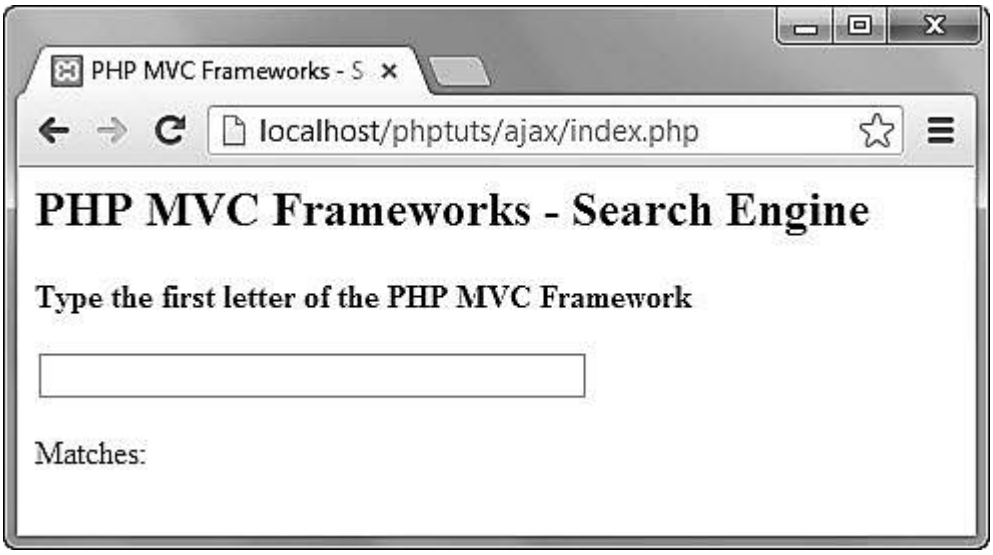


Figure 6.1

Type the letter C in the text box You will get the following results.

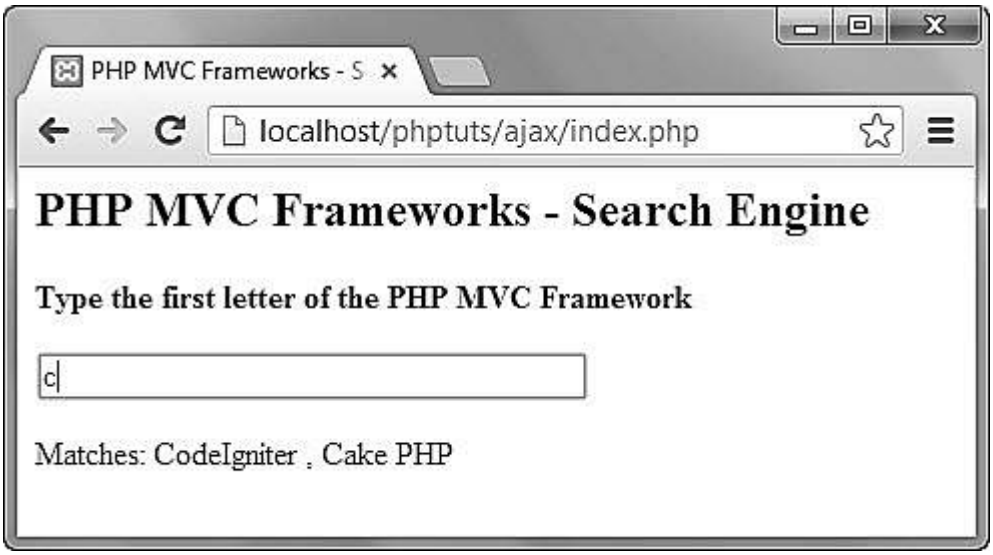


Figure 6.2

The above example demonstrates the concept of AJAX and how it can help us create rich interaction applications.

SUMMARY

- ✓ AJAX is the acronym for Asynchronous JavaScript and XML
- ✓ AJAX is a technology used to create rich interaction applications that reduce the interactions between the client and the server by updating only parts of the web page.
- ✓ Internet Explorer version 5 and 6 use ActiveXObject to implement AJAX operations.

- ✓ Internet explorer version 7 and above and browsers Chrome, Firefox, Opera, and Safari use XMLHttpRequest.

REFERENCES

1. Web Design The complete Reference, Thomas Powell, Tata McGrawHill
2. HTML and XHTML The complete Reference, Thomas Powell, Tata McGrawHill
3. JavaScript 2.0 : The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider
4. PHP : The Complete Reference By Steven Holzner, Tata McGrawHill
5. www.w3schools.com
6. www.github.com
7. www.w3professors.com
8. XML: A Beginner's Guide by Steven Holzner
9. AJAX For Beginners , Ivan Bayross and Sharanam Shah, SPD
10. Web Development with jQuery (WROX) by Richard York
11. Learning PHP, MySQL & JavaScript with j Query, CSS & HTML5 – by Robin Nixon, SPD

7

XML AND AJAX

Unit Structure

INTRODUCTION

Ajax (Asynchronous JavaScript and XML) is a method of building interactive applications for the Web that process user requests immediately. Ajax combines several programming tools including JavaScript, dynamic HTML (DHTML), Extensible Markup Language (XML), cascading style sheets (CSS), the Document Object Model (DOM), and the Microsoft object, XMLHttpRequest.

Ajax allows content on Web pages to update immediately when a user performs an action, unlike an HTTP request, during which users must wait for a whole new page to load. For example, a weather forecasting site could display local conditions on one side of the page without delay after a user types in a zip code.

Google Maps is one well-known application that uses Ajax. The interface allows the user to change views and manipulate the map in real time. Ajax applications do not require installation of a plug-in, but work directly with a Web browser. Because of the technique's reliance on XMLHttpRequest, early applications worked only with Microsoft's Internet Explorer browser, but most other browsers now support Ajax.

Applications created with Ajax use an engine that acts as an intermediary between a user's browser and the server from which it is requesting

information. Instead of loading a traditional Web page, the user's browser loads the Ajax engine, which displays the page the user sees. The engine continues to run in the background, using JavaScript to communicate with the Web browser. User input or clicking on the page sends a JavaScript call to the Ajax engine, which can respond instantly in many cases. If the engine needs additional data, it requests it from the server, usually using XML, while it is simultaneously updating the page.

Ajax is not a proprietary technology or a packaged product. Web developers have been using JavaScript and XML in combination for several years. Jesse James Garrett of the consultancy firm Adaptive Path is credited with coining the name "Ajax" as a shorthand way to refer to the specific technologies involved in a current approach.

XML is a special language used to structure and store .xml files. It is a syntax based on tags.

In some cases, the answer received from Ajax (the data transmitted from the server of the script) can be the content of an XML document. This working method is used by API applications to transfer data from one server to another (from an external server of the site that solicited the transfer). Using Ajax, you can read an XML document directly.

HOW AJAX WORKS

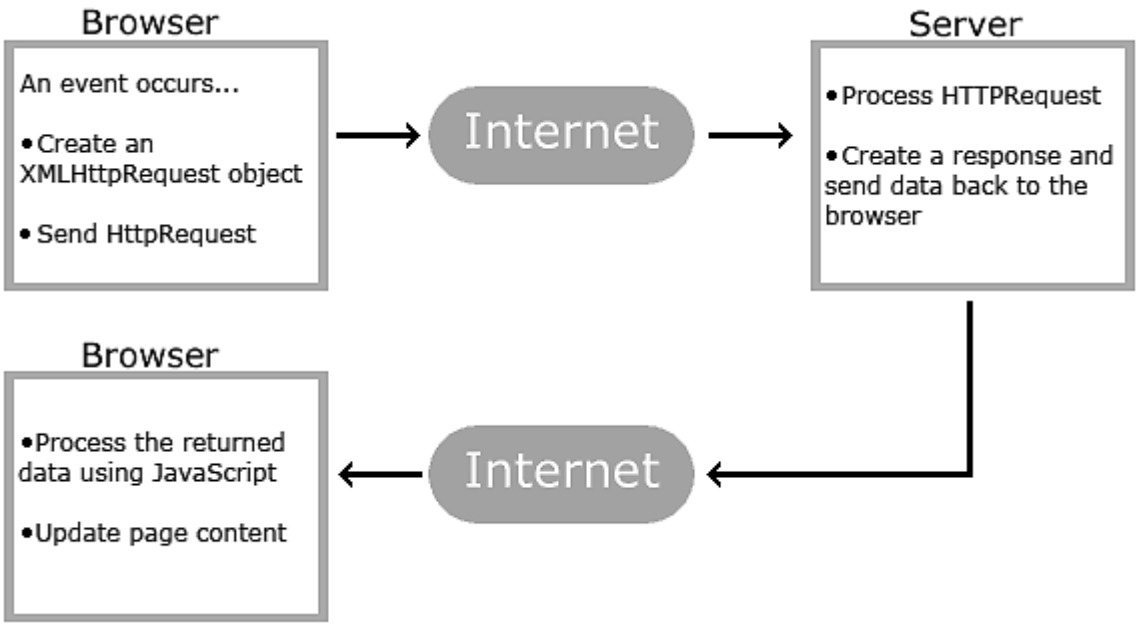


Figure 7.1- Working of AJAX

AJAX is based on internet standards, and uses a combination of:

- ✓ XMLHttpRequest object (to exchange data asynchronously with a server)
- ✓ JavaScript/DOM (to display/interact with the information)
- ✓ CSS (to style the data)
- ✓ XML (often used as the format for transferring data)

The XMLHttpRequest object

Allows to interact with the servers, thanks to its methods and attributes.

Attributes

readyState	the code successively changes value from 0 to 4 that means for "ready".
status	200 is OK 404 if the page is not found.
responseText	holds loaded data as a string of characters.
responseXml	holds an XML loaded file, DOM's method allows to extract data.
onreadystatechange	property that takes a function as value that is invoked when the readystatechange event is dispatched.

Methods

open(mode, url, boolean)	mode: type of request, GET or POST url: the location of the file, with a path. boolean: true (asynchronous) / false (synchronous). optionally, a login and a password may be added to arguments.
send("string")	null for a GET command.

Building a request, step by step

First step: create an instance

This is just a classical instance of class, but two options must be tried, for browser compatibility.

```
1  if (window.XMLHttpRequest)     // Standard object
2  {
3      xhr = new XMLHttpRequest();     // Firefox, Safari, ...
4  }
5  else if (window.ActiveXObject)   // Internet Explorer
6  {
7      xhr = new ActiveXObject("Microsoft.XMLHTTP");
8  }
```

Or exceptions may be used instead:

```
1  try {
2     xhr = new ActiveXObject("Microsoft.XMLHTTP");   // Trying IE
3  }
4  catch(e)    // Failed, use standard object
5  {
6    xhr = new XMLHttpRequest();
7  }
```

Second step: wait for the response

The response and further processing are included in a function and the return of the function will be assigned to the **onreadystatechange** attribute of the object previously created.

```
1  xhr.onreadystatechange = function() { <span class="Style1">//
   instructions to process the response</span> };
```

```
1  if (xhr.readyState == 4)
2  {
3     // Received, OK
4  } else
5  {
6    // Wait...
7  }
```

Third step: make the request itself

Two methods of XMLHttpRequest are used:
- **open**: command GET or POST, URL of the document, true for asynchronous.
- **send**: with POST only, the data to send to the server.

The request below read a document on the server.

```
1  xhr.open('GET', 'https://www.xul.fr/somefile.xml', true);
2  xhr.send(null);
```

Examples

Get a text

```
1  <html>
2  <head>
3  <script>
1  function submitForm()
2  {
3      var xhr;
4      try {  xhr = new ActiveXObject('Msxml2.XMLHTTP');   }
5      catch (e)
6      {
7          try {   xhr = new ActiveXObject('Microsoft.XMLHTTP');    }
8          catch (e2)
9          {
10           try {  xhr = new XMLHttpRequest();     }
11           catch (e3) {  xhr = false;   }
12         }
13     }
14
15   xhr.onreadystatechange  = function()
16   {
17        if(xhr.readyState  == 4)
18        {
19            if(xhr.status  == 200)
```

```
20                document.ajax.dyn="Received:"  + xhr.responseText;
21              else
22                document.ajax.dyn="Error code " + xhr.status;
23          }
24     };
25
26    xhr.open("GET", "data.txt",  true);
27    xhr.send(null);
28 }
1   </script>
2   </head>
3
4   <body>
5       <FORM method="POST" name="ajax" action="">
6           <INPUT type="BUTTON"
    value="Submit"  ONCLICK="submitForm()">
7           <INPUT type="text" name="dyn"  value="">
8       </FORM>
9    </body>
10  </html>
```

Comments on the code:

new ActiveXObject(Microsoft.XMLHTTP)

This constructor is for Internet Explorer.

new XMLHttpRequest()

This constructor is for any other browser including Firefox.

http.onreadystatechange

An anonymous function is assigned to the event indicator.

http.readyState == 4

The 4 state means for the response is ready and sent by the server.

http.status == 200

This status means ok, otherwise some error code is returned, 404 for example.

http.open("POST", "data.xml", true);

POST or GET

URL of the script to execute.

true for asynchronous (false for synchronous).

http.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

This is for POST only.

http.send(document.getElementById("TYPEDTEXT").value);

Send data to the server. Data comes from the "TYPEDTEXT" variable filled throught the form by the user.

Figure 7.2- Demonstration of the above code

Get from XML

To get data from an XML file, we have just to replace this line:

```
1  document.ajax.dyn="Received:" + xhr.responseText;
```

by this code:

```
1  // Assign the XML file to a var
2  var doc = xhr.responseXML;
3  // Read the first element
4  var element = doc.getElementsByTagName('root').item(0);
5   // Assign the content to the form
6  document.ajax.dyn.value= element.firstChild.data;
```

Figure 7.3- Demonstration of the above code

Write to body

The text read is put into the body of the page, and not into a textfield. The code below replaces the textfield form object and the second part replaces the assignment into the JavaScript function.

```
1  <div id="zone">
2       <span class="Style1">... some text to replace ...</span>
3   </div>
```

```
1  document.getElementById("zone").innerHTML = "Received:" +
   xhr.responseText;
```

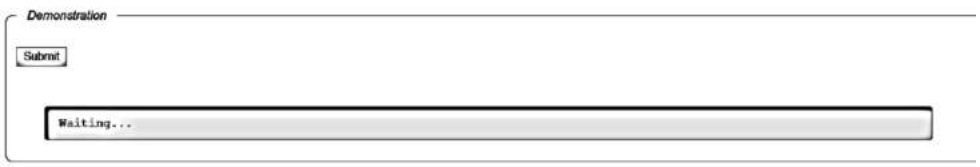


Figure 7.4- Demonstration of the above code

Post a text

A text is sent to the server and is written into a file. The call to the "open" method changes, the argument is POST, the url is the name of a file or script that receives the data sent, and that must process it. And the "send" method has now a value as argument that is a string of parameters.

```
1  xhr.open("POST", "ajax-post-text.php", true);
2  xhr.setRequestHeader("Content-Type", "application/x-www-form-
   urlencoded");
3  xhr.send(data);
```

The parameter of the send method is in the format of the HTML POST method. When several values are sent, they are separated by the ampersand symbol:

```
1  var data = "file=" + url + "&content=" + content;
```

The "file" parameter is the name of a file created to store the content. The filename must be checked by the server to prevent any other file to be modified.

The demonstration is included in the archive.

Using an external file

It is simpler to include a JavaScript file. This line will be included into the head section of the HTML page:

```
1  <script src="ajax.js" type="text/javascript"></script>
```

And the function is called with this statement:

```
1  var xhr = createXHR();
```

The script in the ajax.js file:

```
1   function createXHR() {
2      var request = false;
3      try    {
4        request = new ActiveXObject('Msxml2.XMLHTTP');
5      }
6      catch (err2)    {
7      try {
8        request = new ActiveXObject('Microsoft.XMLHTTP');
9      }
10     catch (err3) {
11       try {  request = new XMLHttpRequest();}
12       catch (err1) { request = false;}
13     }
14    }
15   return request;
16  }
```

Loading the XML data

The first stage is obtaining the XML data. They can be received from PHP (or other server application) as a string, or they can be taken directly from the .xml.

Once the data has been received, they must be loaded in a DOM Java Script object.

a) If XML data are received from Ajax from a server script, in a string row, the loading codes of the XML format are:

CODE:

```
// Function to load the XML details into a DOM Javascript object
Function getXML_string(txt_xml) {
// If you are not using Internet Explorer as a browser
if(window.DOMParser) {
// create the DOM object and save the “tree”
//   using data from the “xmlDoc”
get xml = new DOM Parser();
xml.Doc = getxml.parse (txt_xml,”sourcecode/xml”);
}
else {
// create the XML document in an “xmlDoc”
xmlDoc=new ActiveXObject(“Microsoft.XMLDOM”);
xmlDoc =”false”;         // You can use it for
//   forcing the program to load an XML document
xmlDoc.getXML_string(txt_xml);
}
return xmlDoc;
}
// the element with the data received from the server
var string_xml = ‘the string text where the XML is stored
Received from the server using Ajax (called using GET or POST)’;
// You are calling the function “getXML_row()” with “row_xml”
xml_dom = getXML_row(row_xml);
```

Internet Explorer uses the function “getXML_row()” to load the string of data in a DOM object, and the other browsers are using the DOMParser() function.

– The “row_xml” variable retains data that Ajax receives from the php script and sometimes, even the string data from the xml doc.

b) If the XML data must be taken directly from Ajax from the “file.xml”, you must use the object “XMLHttpRequest”:

EXAMPLE

```
// The function that takes the XML DOM variables from the code
// the content of an XML (the address from the “file” parameter)
function getXML_file {
// If the web browser supports XMLRequest protocol with HTTP
if (window.XMLHttpRequest) {
```

```
// Create the variable that contains the instance from the XMLHttpRequest
xhttp = new XMLHttpRequest();
}
else {
// For Internet explorer 5 or newer
xhttp = new ActiveXObject(“Microsoft.XMLHTTP”);
}
// Define and execute the functions that calls for “file”
xhttp.open(“GET”, file ,false);

xhttp.send(null);
// Retain and resend the answer, transformed in XML DOM
xmlDoc = xhttp.response for XML object;
return xmlDoc;
}
// The variable with the address of the XML file
var file_xml = ‘file.xml’;
// If calls for the function “getXML_file()” with “file_xml”
var xml_dom = getXML_(file_xml);
```

– the function “responseXML” transforms the received data directly into XML DOM.

EXAMPLE 2

AJAX can be used for interactive communication with an XML file.

The following example will demonstrate how a web page can fetch information from an XML file with AJAX:

Figure 7.5

On clicking:

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T'Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti
The dock of the bay	Otis Redding
Picture book	Simply Red

Figure 7.6

When a user clicks on the "Get CD info" button above, the loadDoc() function is executed. The loadDoc() function creates an XMLHttpRequest object, adds the function to be executed when the server response is ready, and sends the request off to the server.

When the server response is ready, an HTML table is built, nodes (elements) are extracted from the XML file, and it finally updates the element "demo" with the HTML table filled with XML data:

LoadXMLDoc()

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
    myFunction(this);
    }
  };
  xhttp.open("GET", "cd_catalog.xml", true);
  xhttp.send();
}
function myFunction(xml) {
  var i;
  var xmlDoc = xml.responseXML;
  var table="<tr><th>Title</th><th>Artist</th></tr>";
  var x = xmlDoc.getElementsByTagName("CD");
  for (i = 0; i <x.length; i++) {
    table += "<tr><td>" +
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
    "</td><td>" +
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
    "</td></tr>";
  }
  document.getElementById("demo").innerHTML = table;
}
```

Example 3

Create a simple XMLHttpRequest and retrieve a .txt file

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
```

```
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>
</body>
</html>
```

How to build an Ajax website?

You need for some wrapper. A short list of frameworks is provided below. Your JavaScript program, integrated into a web page, sends request to the server to load files for rebuilding of pages. The received documents are processed with DOM's methods or XML parsers and the data are used to update the pages.

Drawbacks of Ajax

- If JavaScript is not activated, Ajax can't works. The user must be asked to set JavaScript from within options of the browser, with the "noscript" tag. - Since data to display are loaded dynamically, they are not part of the page, and are ignored by search engines and thus not indexed. - The asynchronous mode may change the page with delays (when the processing on the server take some times), this may be disturbing especially since the result can happens after the instructions that call the server. - The back button may be deactivated (this is not the case in examples provided here). This may be overcomed and it is easily with HTML 5.

REFERENCES

1. Web Design The complete Reference, Thomas Powell, Tata McGrawHill
2. HTML and XHTML The complete Reference, Thomas Powell, Tata McGrawHill
3. JavaScript 2.0 : The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider
4. PHP : The Complete Reference By Steven Holzner, Tata McGrawHill
5. www.w3schools.com
6. www.github.com
7. www.w3professors.com
8. XML: A Beginner's Guide by Steven Holzner
9. AJAX For Beginners, Ivan Bayross and Sharanam Shah, SPD
10. Web Development with jQuery (WROX) by Richard York
11. Learning PHP, MySQL & JavaScript with j Query, CSS & HTML5 – by Robin Nixon, SPD

8

HANDLING XML IN AJAX APPLICATIONS

Unit Structure

PREREQUISITES

We will use Tomcat to run the Ajax application. Tomcat is the servlet container that is used in the official reference implementation for the Java Servlet and JavaServer Pages technologies. Download jakarta-tomcat-5.0.28.exe and run it to install Tomcat to any location you'd like -- c:\tomcat5.0, for instance.

Download the source code and Web application (in wa-ajax-Library.war) for this tutorial.

INTRODUCTION TO AJAX

Ajax basics

Ajax enables a dynamic, asynchronous Web experience without the need for page refreshes. It incorporates the following technologies:

XHTML and CSS provide a standards-based presentation.

Document Object Model (DOM) provides dynamic display and interaction.

XML and XSLT provide data interchange and manipulation.

XMLHttpRequest provides asynchronous data retrieval.

JavaScript binds everything together.

The core of Ajax technology is a JavaScript object: XMLHttpRequest. This object is supplied through browser implementations -- first through Internet Explorer 5.0 and then through Mozilla-compatible browsers. Take a closer at this object.

XMLHttpRequest

With XMLHttpRequest, you can use JavaScript to make a request to the server and process the response without blocking the user. As you create your Web site and use XMLHttpRequest to perform screen updates on the client's browser without the need for refresh, it provides much flexibility and a rich user experience.

Examples of XMLHttpRequest applications include Google's Gmail service, Google's Suggest dynamic lookup interface, and the MapQuest dynamic map interface. In the next sections, we demonstrate how to use XMLHttpRequest object in detail as we demonstrate the design of a book order application and implement it.

Application design

- ✓ Elements of the application
- ✓ The sample Web-based book order application will contain the following client-side functions implemented in Ajax

Subscription ID validation

- ✓ A View Authors list
- ✓ A View Publishers list

The objective here is to show how real time validation and page refreshes in a Web page make user interaction smoother and more efficient.

Structure of the application

The diagram in Figure 8.1 depicts the design architecture of the sample book order application:

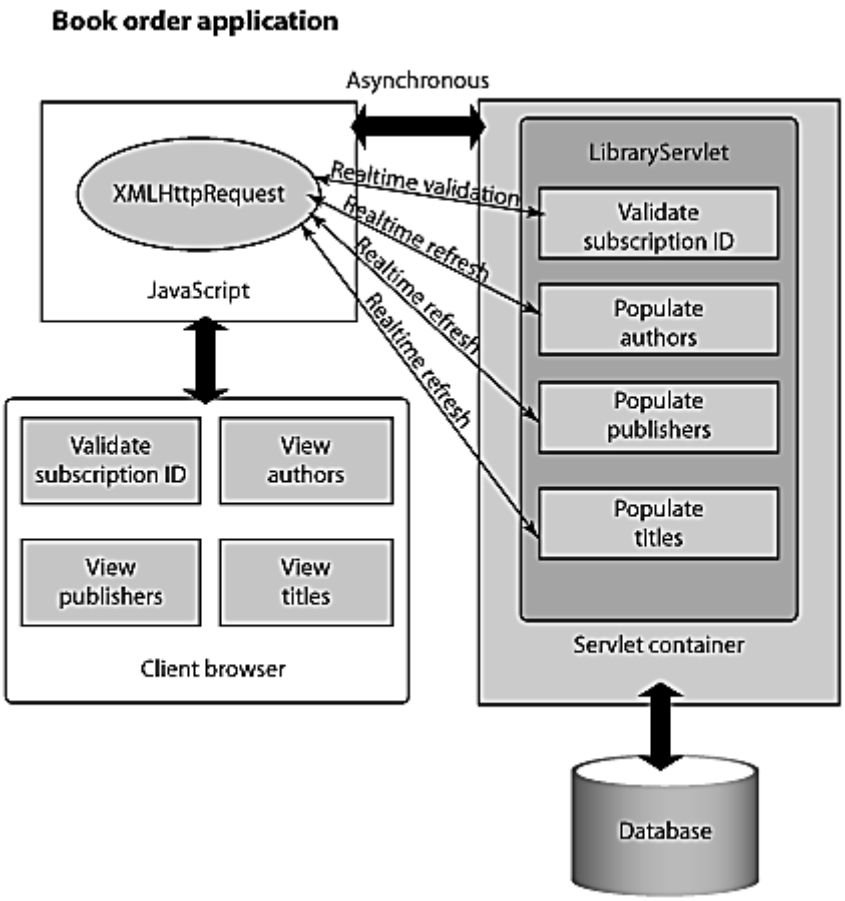


Figure 8.1- The Ajax architecture

The Ajax architecture

The application will be a single Web page developed with JavaServer Pages (JSP) technology. The user will be able to invoke the Web page using a Web browser (such as Microsoft® Internet Explorer) and enter the Subscription ID which the application validates in real time. As the ID is validated asynchronously, the user can input more information. The user can view the book titles either by Author or Publisher. The screen will populate the Authors or Publishers list based on user choice. Based on the selection, the Titles list is populated. All these lists will populate in real time -- in other words, the page is not refreshed, but still the data comes from the backend tier. We call this phenomenon real time refreshes.

As you can see in Figure 1, the XMLHttpRequest JavaScript object helps with the real time asynchronous processing. This object makes a request in the form of XML over HTTP to the LibraryServlet servlet residing in a Web container. The servlet then queries the database, fetches the data, and sends it back to the client, again in the form of XML over HTTP. The requests and responses occur in real time without refreshing the page.

This is what makes Ajax so powerful. The user does not wait for page reload to complete because there is no page reload.

Implementing the application

Application implementation with Ajax

In this section, we do a code walkthrough of the sample book order application and take a close look at each Ajax-based, Javascript component:

- ✓ Validate Subscription ID
- ✓ View Authors
- ✓ View Publishers
- ✓ View Titles

Code walkthrough: Validate the subscription ID

Let's start with the function Validate Subscription ID<input type="text" name="subscriptionID" onblur="validate(this.form)"/>. This code creates a text field where users can enter Subscription IDs. Once the user enters the ID and moves to the next field in the form, the onBlur event fires. This event calls a JavaScript function validate():

```
1  var req;
2  function validate(formObj) {
3      init();
4      req.onreadystatechange = subscriptionValidator;
5      req.send("subscriptionID=" + formObj.subscriptionID.value);
6  }
```

The validate() function takes formObj as a parameter. It first calls the init() function:

```
1   function init() {
2       if (window.XMLHttpRequest) {
3           req = new XMLHttpRequest();
4       } else if (window.ActiveXObject) {
5           req = new ActiveXObject("Microsoft.XMLHTTP");
6       }
7       var url = "/Library/LibraryServlet";
8       req.open("POST", url, true);
9       req.setRequestHeader("Content-Type", "application/x-www-form-
    urlencoded");
10 }
```

Code walkthrough: init()

Now look at the init() function does (we divide the code in parts):

```
1  if (window.XMLHttpRequest) {
2      req = new XMLHttpRequest();
3  } else if (window.ActiveXObject) {
4      req = new ActiveXObject("Microsoft.XMLHTTP");
5  }
```

The init() function first creates the XMLHttpRequest object. This request object is the core of Ajax. It sends and receives the request in XML form. This piece of code checks for browser support for the XMLHttpRequest

object (most browsers support it). If you are using Microsoft Internet Explorer 5.0 or above, then the second condition is executed.

```
1  req.open("POST", url, true);
2  req.setRequestHeader("Content-Type", "application/x-www-form-
   urlencoded");
```

Once your code creates the XMLHttpRequest object, you need to set certain request properties. In the preceding code, the first line sets the request method, request URL, and the type of request (whether it is asynchronous or not). It does so by calling the open() method on the XMLHttpRequest object.

Here we will use the POST method. Ideally, use POST when you need to change the state on the server. Our application is not going to change the state, but we still prefer to use POST. The url is the URL of the servlet to be executed. true indicates that we will process the request asynchronously.

For the POST method, we need to set the request header Content-Type. This is not required for the GET method.

```
1    function validate(formObj) {
2        init();
3        req.onreadystatechange = subscriptionValidator;
4        req.send("subscriptionID=" + formObj.subscriptionID.value);
5    }
```

Code walkthrough: Callback handler 1

To continue with the validation method, next you assign the subscriptionValidator callback handler to onreadystatechange which will fire at every state change on the request.

What is this callback handler all about? Since you are processing the request asynchronously, you need a callback handler which is invoked when the complete response is returned from the server -- the callback handler is where you will validate the subscription ID (that is, write your actual validation code).

The handler acts as a listener. It waits until the response is complete. (More on the handler code later.) To send the request, the last line calls the send() method. The request is sent as a name=value pair. For the GET method, the request is sent as part of the URL, so the send() method is passed a null parameter.

The request is sent to the servlet. The servlet processes the request and sends back the response in real time. This is how the servlet processes the request. The next code snippet illustrates the LibraryServlet -- doPost() method.

```
1    public void doPost(HttpServletRequest req, HttpServletResponse resp) throws
2    ServletException, IOException {
3        String ID = null;
4        ID = req.getParameter("subscriptionID");
```

```
5         if (ID != null) {
6             String status = "<message>" + this.validID(ID) + "</message>";
7         this.writeResponse(resp, status);
8         }
9     }
```

Code walkthrough: Callback handler 2

The doPost() method gets the subscriptionID from the request parameter. To validate the ID, it calls the validID() method. This method validates the ID and returns true if the ID is valid, otherwise it returns false. It constructs the return status in XML format and writes the response by calling the writeResponse() method. Now examine the writeResponse() method.

```
1  public void writeResponse(HttpServletResponse resp, String output)
   throws IOException {
2     resp.setContentType("text/xml");
3     resp.setHeader("Cache-Control", "no-cache");
4     resp.getWriter().write(output);
5  }
```

The response is sent in XML format. The first line sets the response content type, which is text/xml. The next line sets the header Cache-Control with the value of no-cache. This header is mandatory. Ajax requires that response output is not cached by the browser. To write the response, the last line calls the getWriter().write() method.

Code walkthrough: Callback handler 3

The request is processed by the servlet and the response is sent back to the client. Remember, this all happens in the background without a page refresh. Now the callback handler method that we discussed earlier, will handle and parse the response:

```
1  function subscriptionValidator() {
2     if (req.readystate == 4) {
3     if (req.status == 200) {
4         var messageObj =
   req.responseXML.getElementsByTagName("message")[0];
5         var message =
   messageObj.childNodes[0].nodeValue;
6         if (message == "true") {
7         msg.innerHTML = "Subscription is valid";
8         document.forms[0].order.disabled = false;
9         } else {
10        msg.innerHTML = "Subscription not valid";
11        document.forms[0].order.disabled = true;
12        }
13    }
14    }
15}
```

Code walkthrough: Revisiting XMLHttpRequest

As mentioned earlier, the XMLHttpRequest object is the core object which constructs and sends the request. It also reads and parses the response coming back from the server. Look at the code in parts.

```
1  if (req.readystate == 4) {
2      if (req.status == 200) {
```

The preceding code checks the state of the request. If the request is in a ready state, it will then read and parse the response.

What do we mean by ready state? When the request object attribute readystate has the value of 4, it means that the client received the response and is complete. Next we check the request status (whether the response was a normal page or an error page). To ensure that the response is normal, check for the status value of 200. If the status value is 200, then it will process the response.

```
1  var messageObj =
   req.responseXML.getElementsByTagName("message")[0];
2  var message = messageObj.childNodes[0].nodeValue;
3  if (message == "true") {
4      msg.innerHTML = "Subscription is valid";
5      document.forms[0].order.disabled = false;
6  } else {
7      msg.innerHTML = "Subscription not valid";
8      document.forms[0].order.disabled = true;
9  }   }
```

Next, the request object reads the response by calling responseXML property. Note the servlet sent back the response in XML so we use responseXML. If the response sent was in text, then you can use the responseText property.

In this example, we deal with XML. The servlet constructed the response in a <message> tag. To parse this XML tag, call the getElementsByTagName() method on the responseXML property of the XMLHttpRequest object. It gets the tag name and the child value of the tag. Based on the value parsed, the response is formatted and written in HTML.

You just finished validating the subscription ID, all without a page refresh.

Code walkthrough: View authors, publishers, titles

The other functionalities -- View Authors, View Publishers, and View Titles -- work along similar lines. You have to define separate handlers for each functionality:

```
1   function displayList(field) {
2       init();
3       titles.innerHTML = " ";
4       req.onreadystatechange = listHandler;
5       req.send("select=" + escape(field));
```

```
6  }
7
8  function displayTitles(formObj) {
9      init();
10     var index = formObj.list.selectedIndex;
11     var val = formObj.list.options[index].value;
12     req.onreadystatechange = titlesHandler;
13     req.send("list=" + val);
14 }
```

Remember, this sample application allows the user to view the titles by author or publisher. So either the Author list or the Publisher list displays. In such a scenario, the application only calls one callback handler based on the user selection -- in other words, for author and publisher list, you have only one listHandler callback handler.

To display the titles list, you will use titlesHandler. The remaining functionality stays the same with the servlet processing the request and writing back the response in XML fornat. The response is then read, parsed, formatted, and written in HTML. You can render the list in HTML as a select.....options tag. This sample code snippet shows the titlesHandler method.

```
1  var temp = "<select name=\"titles\" multiple\>";
2  for (var i=0; i<index; i++) {
3      var listObj = req.responseXML.getElementsByTagName("list")[i];
4      temp = temp + "<option value=" + i +">" +
   listObj.childNodes[0].nodeValue
5  + "</option>";
6  }
7  temp = temp + "</select>";
8  titles.innerHTML = temp;
```

So far, we've demonstrated how to implement real time validation and refreshes. WIth Ajax, you can choose among several ways to add spice and flair to user interactions on your Web sites. Next we'll run the application.

Running and testing the application

Run the application

Download the sample code wa-ajax-Library.war and copy it to your Tomcat Webapp directory (for example, c:\Tomcat 5.0\Webapps). To start the Tomcat server, type the following:

```
1  cd bin
2  C:\Tomcat 5.0\bin> catalina.bat start
```

Tomcat is now started with your Ajax Web application deployed.

Test the application

To test the application:

Open your Web browser. Point to http://localhost:tomcatport/Library/order.jsp where the variable tompcatport is the port that your Tomcat server runs on.

You will see the subscription screen.

In the Enter Subscription ID field, type any user ID except "John" and tab out of the field.

The subscription ID request that you made to the server asynchronously will be validated. You will see a message "Subscription not valid" as shown in Figure 8.2:

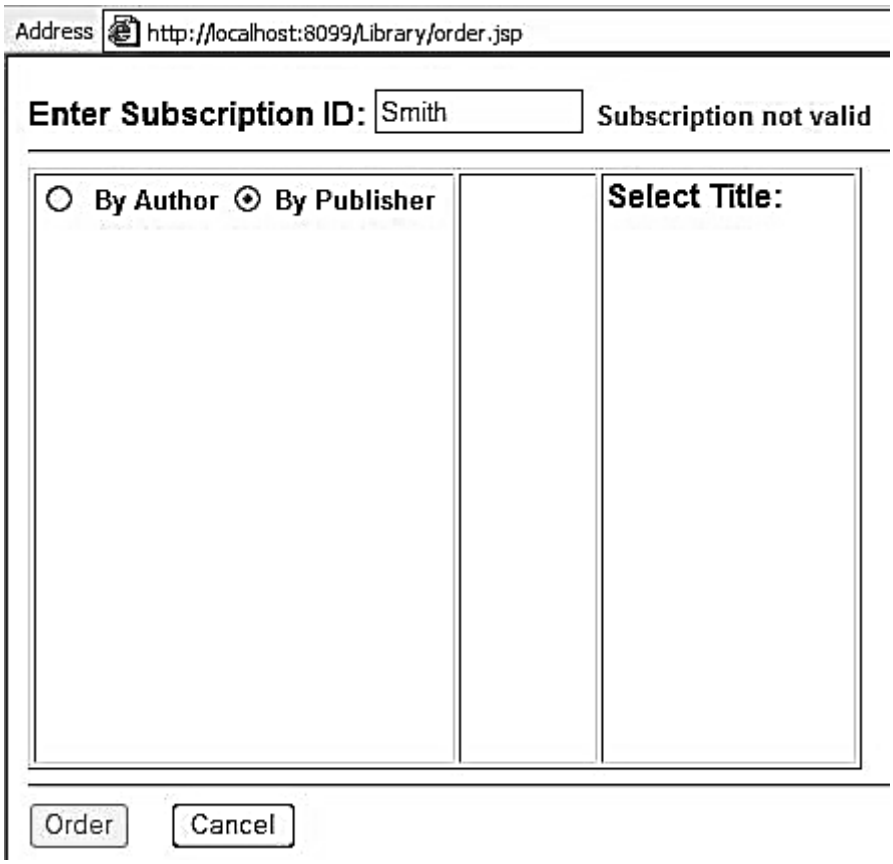


Figure 8.2- The "Subcription not valid" screen

The "Subcription not valid" screen

The application validated the user asynchronously and provided runtime validation without refreshing the browser.

Type in the user ID value, John.

You will see a message "Subscription is valid". Once the subscription is valid, the application enables Order button.

Select the By Author or By Publisher radio button to populate the author or publisher drop-down list, respectively.

Select an author or publisher from the drop-down list.

The title area is populated dynamically (as in Figure 8.3).

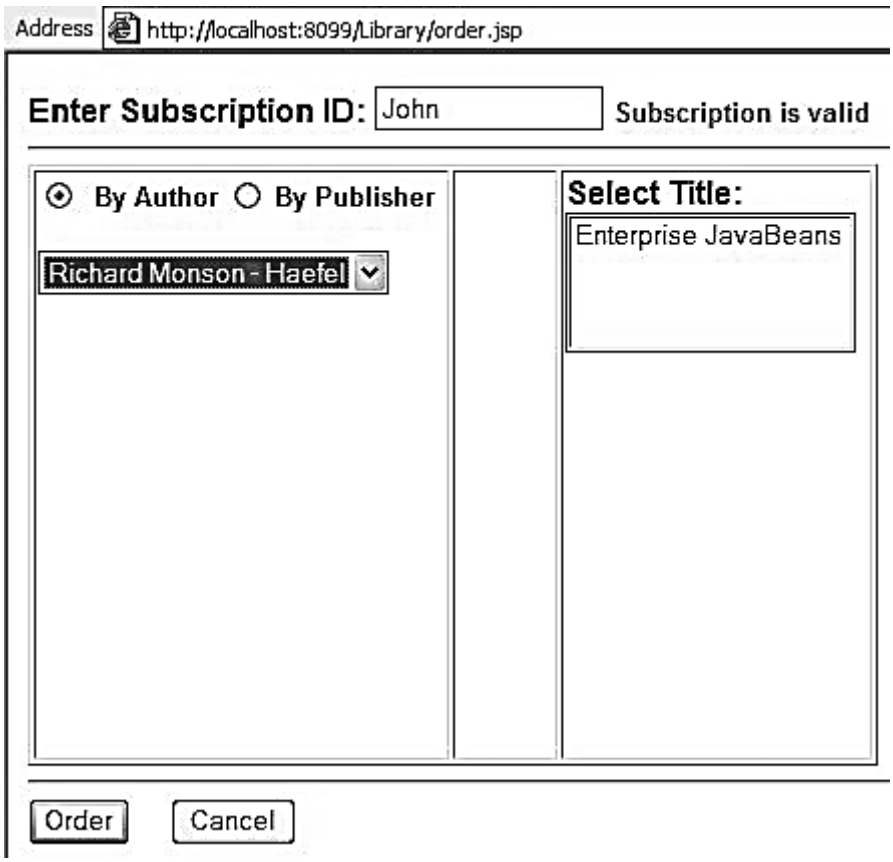


Figure 8.3 The "Subcription is valid" screen

The "Subcription is valid" screen

When you select the author or publisher, the application requests the server to provide the title information associated with the selected author or publisher at runtime from the server. The title information displays without refreshing the browser.

SUMMARY

Ajax has come a long way since its inception. We believe Ajax can be applied as more than just a design pattern, though Ajax still has some issues:

- ✓ Browser support for the XMLHttpRequest object can be constraining. Most browsers do support the XMLHttpRequest object, but a few do not (usually the older version of browsers).
- ✓ Ajax is best suited for displaying a small set of data. If you deal with large volumes of data for a real time display of lists, then Ajax might not be the right solution.
- ✓ Ajax is quite dependent on JavaScript. If a browser doesn't support JavaScript or if a user disables the scripting option, then you cannot leverage Ajax at all.
- ✓ The asynchronous nature of Ajax will not guarantee synchronous request processing for multiple requests. If you need to prioritize your

validation or refreshes, then design your application accordingly.

- ✓ Even with these potential hiccups, Ajax still stands as the best solution to enhance your Web pages and resolve page-reload issues

REFERENCES

1. Web Design The complete Reference, Thomas Powell, Tata McGrawHill
2. HTML and XHTML The complete Reference, Thomas Powell, Tata McGrawHill
3. JavaScript 2.0 : The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider
4. PHP : The Complete Reference By Steven Holzner, Tata McGrawHill
5. www.w3schools.com
6. www.github.com
7. www.w3professors.com
8. XML: A Beginner's Guide by Steven Holzner
9. AJAX For Beginners , Ivan Bayross and Sharanam Shah, SPD
10. Web Development with jQuery (WROX) by Richard York
11. Learning PHP, MySQL & JavaScript with j Query, CSS & HTML5 – by Robin Nixon, SPD

9

WORKING WITH CSS IN AJAX APPLICATION AND AJAX DESIGN ISSUES

Unit Structure

INTRODUCTION

CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External stylesheets are stored in CSS files. CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

Using CSS helps in:

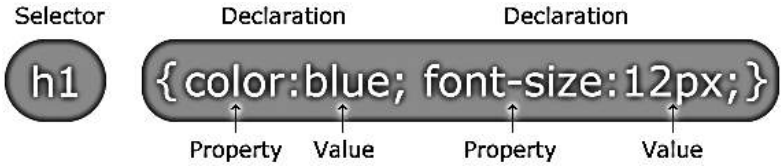
HTML was NEVER intended to contain tags for formatting a web page. HTML was created to describe the content of a web page, like:
<h1>This is a heading</h1>
This is a paragraph.

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS. CSS removed the style formatting from the HTML page. The style definitions are normally saved in external .css files. With an external stylesheet file, you can change the look of an entire website by changing just one file.

CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



- ✓ The selector points to the HTML element you want to style.
- ✓ The declaration block contains one or more declarations separated by semicolons.
- ✓ Each declaration includes a CSS property name and a value, separated by a colon.
- ✓ A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

  In the following example all <p> elements will be center-aligned, with a red text color:

```
p {
  color: red;
  text-align: center;
}
```

CSS Selectors

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

The element Selector

The element selector selects elements based on the element name.

You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

Example

```
p {
  text-align: center;
  color: red;
}
```

The id Selector

- ✓ The id selector uses the id attribute of an HTML element to select a specific element.
- ✓ The id of an element should be unique within a page, so the id selector is used to select one unique element!
- ✓ To select an element with a specific id, write a hash (#) character, followed by the id of the element.
- ✓ The style rule below will be applied to the HTML element with id="para1":

Example
```
#para1 {
  text-align: center;
  color: red;
}
```

The class Selector

- ✓ The class selector selects elements with a specific class attribute.
- ✓ To select elements with a specific class, write a period (.) character, followed by the name of the class.
- ✓ In the example below, all HTML elements with class="center" will be red and center-aligned:

  Example
  ```
  .center {
    text-align: center;
    color: red;
  }
  ```
- ✓ You can also specify that only specific HTML elements should be affected by a class.

  In the example below, only <p> elements with class="center" will be center-aligned:

  Example
  ```
  p.center {
    text-align: center;
    color: red;
  }
  ```
- ✓ HTML elements can also refer to more than one class.
- ✓ In the example below, the <p> element will be styled according to class="center" and to class="large":

  Example
  ```
  <p class="center large">This paragraph refers to two classes.</p>
  ```

Grouping Selectors

If you have elements with the same style definitions, like this:

```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}
```

```
p {
  text-align: center;
  color: red;
}
```

It will be better to group the selectors, to minimize the code. To group selectors, separate each selector with a comma. In the example below we have grouped the selectors from the code above:

Example

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

CSS IN AJAX

Using CSS in AJAX requires jQuery

jQuery is a lightweight, "write less, do more", JavaScript library.

The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

- ✓ HTML/DOM manipulation
- ✓ CSS manipulation
- ✓ HTML event methods
- ✓ Effects and animations
- ✓ AJAX
- ✓ Utilities

jQuery Syntax

The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).

Basic syntax is: $(selector).action()

- ✓ A $ sign to define/access jQuery
- ✓ A (selector) to "query (or find)" HTML elements
- ✓ A jQuery action() to be performed on the element(s)

Examples:

- ✓ $(this).hide() - hides the current element.

- ✓ $("p").hide() - hides all <p> elements.
- ✓ $(".test").hide() - hides all elements with class="test".
- ✓ $("#test").hide() - hides the element with id="test".

jQuery DOM Manipulation

One very important part of jQuery is the possibility to manipulate the DOM. jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes.

DOM = Document Object Model

The DOM defines a standard for accessing HTML and XML documents: "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

Get Content - text(), html(), and val()

We will use the same three methods from the previous page to set content:

- ✓ text() - Sets or returns the text content of selected elements
- ✓ html() - Sets or returns the content of selected elements (including HTML markup)
- ✓ val() - Sets or returns the value of form fields

The following example demonstrates how to set content with the jQuery text(), html(), and val() methods:

Example

```
$("#btn1").click(function(){
  $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
  $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
  $("#test3").val("Dolly Duck");
});
```

A Callback Function for text(), html(), and val()

All of the three jQuery methods above: text(), html(), and val(), also come with a callback function. The callback function has two parameters: the index of the current element in the list of elements selected and the original (old) value. You then return the string you wish to use as the new value from the function. The following example demonstrates text() and html() with a callback function:

Example

```
$("#btn1").click(function(){
  $("#test1").text(function(i, origText){
    return "Old text: " + origText + " New text: Hello world!
```

```
    (index: " + i + ")";
  });
});
$("#btn2").click(function(){
  $("#test2").html(function(i, origText){
    return "Old html: " + origText + " New html: Hello <b>world!</b>
    (index: " + i + ")";
  });
});
```

Set Attributes - attr()

The jQuery attr() method is also used to set/change attribute values. The following example demonstrates how to change (set) the value of the href attribute in a link:

Example

```
$("button").click(function(){
  $("#w3s").attr("href", "https://www.w3schools.com/jquery/");
});
```

The attr() method also allows you to set multiple attributes at the same time. The following example demonstrates how to set both the href and title attributes at the same time:

Example

```
$("button").click(function(){
  $("#w3s").attr({
    "href" : "https://www.w3schools.com/jquery/",
    "title" : "W3Schools jQuery Tutorial"
  });
});
```

A Callback Function for attr()

The jQuery method attr(), also comes with a callback function. The callback function has two parameters: the index of the current element in the list of elements selected and the original (old) attribute value. You then return the string you wish to use as the new attribute value from the function. The following example demonstrates attr() with a callback function:

Example

```
$("button").click(function(){
  $("#w3s").attr("href", function(i, origValue){
    return origValue + "/jquery/";
  });
});
```

Add New HTML Content

We will look at four jQuery methods that are used to add new content:

- ✓ append() - Inserts content at the end of the selected elements

- ✓ prepend() - Inserts content at the beginning of the selected elements
- ✓ after() - Inserts content after the selected elements
- ✓ before() - Inserts content before the selected elements

jQuery append() Method

The jQuery append() method inserts content AT THE END of the selected HTML elements.

Example
```
$("p").append("Some appended text.");
```

jQuery prepend() Method

The jQuery prepend() method inserts content AT THE BEGINNING of the selected HTML elements.

Example
```
$("p").prepend("Some prepended text.");
```

Add Several New Elements With append() and prepend()

In both examples above, we have only inserted some text/HTML at the beginning/end of the selected HTML elements. However, both the append() and prepend() methods can take an infinite number of new elements as parameters. The new elements can be generated with text/HTML (like we have done in the examples above), with jQuery, or with JavaScript code and DOM elements. In the following example, we create several new elements. The elements are created with text/HTML, jQuery, and JavaScript/DOM. Then we append the new elements to the text with the append() method (this would have worked for prepend() too) :

Example
```
function appendText() {
  var txt1 = "<p>Text.</p>";              // Create element with HTML
  var txt2 = $("<p></p>").text("Text.");   // Create with jQuery
  var txt3 = document.createElement("p");  // Create with DOM
  txt3.innerHTML = "Text.";
  $("body").append(txt1, txt2, txt3);      // Append the new elements
}
```

jQuery after() and before() Methods

The jQuery after() method inserts content AFTER the selected HTML elements. The jQuery before() method inserts content BEFORE the selected HTML elements.

Example
```
$("img").after("Some text after");
$("img").before("Some text before");
```

Add Several New Elements With after() and before()

Also, both the after() and before() methods can take an infinite number of new elements as parameters. The new elements can be generated with

text/HTML (like we have done in the example above), with jQuery, or with JavaScript code and DOM elements. In the following example, we create several new elements. The elements are created with text/HTML, jQuery, and JavaScript/DOM. Then we insert the new elements to the text with the after() method (this would have worked for before() too) :

Example

```
function afterText() {
  var txt1 = "<b>I </b>";                  // Create element with HTML
  var txt2 = $("<i></i>").text("love ");     // Create with jQuery
  var txt3 = document.createElement("b");    // Create with DOM
  txt3.innerHTML = "jQuery!";
  $("img").after(txt1, txt2, txt3);          // Insert new elements after <img>
}
```

jQuery - Remove Elements

With jQuery, it is easy to remove existing HTML elements. To remove elements and content, there are mainly two jQuery methods:

- ✓ remove() - Removes the selected element (and its child elements)
- ✓ empty() - Removes the child elements from the selected element

jQuery remove() Method

The jQuery remove() method removes the selected element(s) and its child elements.

Example

```
$("#div1").remove();
```

jQuery empty() Method

The jQuery empty() method removes the child elements of the selected element(s).

Example

```
$("#div1").empty();
```

Filter the Elements to be Removed

The jQuery remove() method also accepts one parameter, which allows you to filter the elements to be removed. The parameter can be any of the jQuery selector syntaxes. The following example removes all <p> elements with class="test":

Example

```
$("p").remove(".test");
```

This example removes all <p> elements with class="test" or class="demo":

Example

```
$("p").remove(".test, .demo");
```

jQuery - Get and Set CSS Classes

With jQuery, it is easy to manipulate the CSS of elements.

jQuery Manipulating CSS

jQuery has several methods for CSS manipulation. We will look at the following methods:

- ✓ addClass() - Adds one or more classes to the selected elements
- ✓ removeClass() - Removes one or more classes from the selected elements
- ✓ toggleClass() - Toggles between adding/removing classes from the selected elements
- ✓ css() - Sets or returns the style attribute

Example Stylesheet

The following stylesheet will be used for all the examples on this page:

```
.important {
  font-weight: bold;
  font-size: xx-large;
}
.blue {
  color: blue;
}
```

jQuery addClass() Method

The following example shows how to add class attributes to different elements. Of course you can select multiple elements, when adding classes:

Example

```
$("button").click(function(){
  $("h1, h2, p").addClass("blue");
  $("div").addClass("important");
});
```

You can also specify multiple classes within the addClass() method:

Example

```
$("button").click(function(){
  $("#div1").addClass("important blue");
});
```

jQuery removeClass() Method

The following example shows how to remove a specific class attribute from different elements:

Example

```
$("button").click(function(){
  $("h1, h2, p").removeClass("blue");
});
```

jQuery toggleClass() Method

The following example will show how to use the jQuery toggleClass()

method. This method toggles between adding/removing classes from the selected elements:

Example
```
$("button").click(function(){
  $("h1, h2, p").toggleClass("blue");
});
```

jQuery - css() Method

The css() method sets or returns one or more style properties for the selected elements.

Return a CSS Property

To return the value of a specified CSS property, use the following syntax:

css("propertyname");

The following example will return the background-color value of the FIRST matched element:

Example
```
$("p").css("background-color");
```

Set a CSS Property

To set a specified CSS property, use the following syntax:

css("propertyname","value");

The following example will set the background-color value for ALL matched elements:

Example
```
$("p").css("background-color", "yellow");
```

Set Multiple CSS Properties

To set multiple CSS properties, use the following syntax:

css({"propertyname":"value","propertyname":"value",...});

The following example will set a background-color and a font-size for ALL matched elements:

Example
```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

SUMMARY OF jQUERY

jQuery Properties

Property	Description
context	Removed in version 3.0. Contains the original context passed to jQuery()

jquery	Contains the jQuery version number
jQuery.fx.interval	Change the animation firing rate in milliseconds
jQuery.fx.off	Globally disable/enable all animations
jQuery.support	A collection of properties representing different browser features or bugs (Intended for jQuery's internal use)
length	Contains the number of elements in the jQuery object

AJAX DESIGN ISSUES

Ajax is a new when it comes to Web applications, and as such, new rules about how the interface should and shouldn't work are emerging. Those rules have not been formalized yet, but the Ajax community is discussing them. Before launching into creating your own Ajax applications, consider the following design issues.

Breaking the back button and bookmarks

When you have control over what's going on in a Web page and you're using JavaScript to make things turn on and off in a page — or even to alter the page's entire appearance — the browser's Back button won't work anymore. The Back button works from the browser's history object, which stores the successive pages that have been loaded into the browser. But if you aren't loading new pages — which is what Ajax is all about — the history object doesn't know about them.

This is one to keep in mind as you design your Ajax applications. If necessary, provide your own local Back button using JavaScript. If you want to let the user move backwards to previous window states, you have to keep track of what's been going on and let the user navigate as they want to.

Leaving the user in control

Ajax applications can seem to take on a life of their own because they operate behind the scenes. And they can communicate with the server even when the user doesn't want them to — as when the user makes a typing error. You can imagine how you'd feel if you'd just entered a typo and it was immediately stored in a database by an application that didn't ask you if you wanted to store anything.

So, to give your applications a good feel, here are a few tips for putting users in control:

Remember that, ideally, your application is supposed to respond to events caused only by the user. Users can find too much server-side validation disconcerting because it creates the impression that you're correcting them at every keystroke. Don't forget that one of the design principles of graphical

user interfaces (GUIs) is that the user should be in control, that they should direct the action.

Remembering all the different browsers

As with any Web application, it's worthwhile to keep in mind that there are many different browsers around, and your Ajax application should be tested in the ones you want to support.

As of this writing, Internet Explorer and Firefox make up about 96 percent of browser use, and the rest (Opera, Safari, and so on) are each in the 1 percent or less category. And don't forget that not all browser will support JavaScript, or will have JavaScript turned on — and for those users, you should have a backup plan.

Avoiding a sluggish browser

Ajax applications can be large, and when they start using up resources like memory and CPU speed, you've got to be careful. A large application can use up a huge amount of memory, especially if you're not careful about getting rid of large objects that have been created.

Sometimes, developers use Ajax just because it's a new thing. Be careful about that tendency, too. Ajax solves many problems, but if you don't have to use it, there's no reason to. And also, don't forget that your Ajax applications might not work in all browsers.

Handling sensitive data

With Ajax, it's easy to send data without the user knowing what's going on. In fact, that's part of the whole client/server connection thing that makes Ajax so popular. But it's also true that the user may not want to send the data you're sending.

It's best to be careful about sensitive data. The Internet is not necessarily a secure place for sensitive data, after all, and if you start sending social security numbers or credit card numbers without the user's permission, you could wind up in trouble. So give the users the benefit of the doubt — ask before you send sensitive data.

Creating a backup plan

Ajax relies on being connected to a server but don't forget that not everyone is online all the time. And your own server may go down, so your users may be working from cached pages. If you can't connect to a page online, you should have some kind of backup. And that goes for users who have browsers that don't support JavaScript, too.

Showing up in search engines

Google searches billions of Web pages for the text that its users search for — but if the text you display is loaded into a page based on user actions, not on browser refreshes, Google isn't able to see that text. So bear in mind that if you want to make your page searchable on search engines like Google,

you've got to give your page the search terms they need. (You can store your keywords in a <meta> tag in the browser's <head> section, for example, which is where search engines expect to find them. See this site for more information on that.)

AJAX is growing very fast and that is the reason that it contains many issues with it. We hope with the passes of time, they will be resolved and AJAX will become ideal for web applications. We are listing down a few issues that AJAX currently suffers from.

Complexity is increased

Server-side developers will need to understand that presentation logic will be required in the HTML client pages as well as in the server-side logic.

Page developers must have JavaScript technology skills.

AJAX-based applications can be difficult to debug, test, and maintain

JavaScript is hard to test - automatic testing is hard.

Weak modularity in JavaScript.

Lack of design patterns or best practice guidelines yet.

Toolkits/Frameworks are not mature yet

Most of them are in beta phase

No standardization of the XMLHttpRequest yet

No support of XMLHttpRequest in old browsers

Iframe will help

JavaScript technology dependency and incompatibility

Must be enabled for applications to function.

Still some browser incompatibilities exist.

JavaScript code is visible to a hacker

Poorly designed JavaScript code can invite security problems.

REFERENCES

1. Web Design The complete Reference, Thomas Powell, Tata McGrawHill
2. HTML and XHTML The complete Reference, Thomas Powell, Tata McGrawHill
3. JavaScript 2.0 : The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider
4. PHP : The Complete Reference By Steven Holzner, Tata McGrawHill
5. www.w3schools.com
6. www.github.com

7. www.w3professors.com

8. XML: A Beginner's Guide by Steven Holzner

9. AJAX For Beginners , Ivan Bayross and Sharanam Shah, SPD

10. Web Development with jQuery (WROX) by Richard York

11. Learning PHP, MySQL & JavaScript with j Query, CSS & HTML5 – by Robin Nixon, SPD