MIT 3107
ADVANCED INTERNET TECHNOLOGIES

Chapter 2

Internet Technologies

Learning Outcomes

- By the end of this chapter, the learner should be able to:
  - Describe functions of Markup languages.
  - Describe server side scripting and client side scripting.
  - Identify scripting languages.

HTML

- HTML (HyperText Mark-Up Language, sometimes also called HyperText Meta Language) is the language of the World Wide Web.
- All pages are encoded in some form of HTML, whether it be a version from 1994 or 2007, and Web browsers tend to maintain as much compatibility as possible.

HTML

- Part of the responsibility is shared with the producers of the Web content.
- If you use esoteric tags, which are not part of the standard, but which look as if they might follow it and which are only supported by a minority of browsers, you can expect your page to look a bit strange on some platforms.

HTML

- On the other hand, the simpler the use of HTML that is made, the fewer problems the page is likely to have in the longer term.
- So, there is a balance that you have to strike between elaborate presentation and maintainability of the code.

HTML

- This job is made easier by the fact that browser application providers tend to be somewhat looser in their adherence to standards than in the past, coupled with techniques such as server side scripting that generate much of the HTML at the same time as the content is presented.

HTML

- This is the key to HTML—it provides a standard definition for the presentation layer which is robust and well-defined by open standards maintained by the W3C.

HTML

- As a mark-up language, it contains information that is not designed to be interpreted as content (that is, the user never sees the HTML itself) but as presentation information that enriches the actual content.

HTML Tags

- The page content is enclosed in a mechanism known as tags, which tell the browser how the designer intended the content to be rendered in an abstract fashion.
- Each tag contains specific information relating to the way that the browser is to begin rendering the content that follows it.

HTML Tags

- Usually, there is a starting tag and ending tag (although there are exceptions).
- You might like to think of this as turning formatting on and off, because this makes it easier to conceptualize some of the principles.

HTML Tags

- So, a starting bold text tag tells the browser that, until the ending tag is reached, the content is to be rendered using a bold font.

HTML Tags

- However, the end result might not always be the same on all platforms, including that of the designer—it is only there to give details of the intention of the designer with respect to the content that is being presented.
- Bold text on one platform might be of a different size, weight, or font depending on the browser being used.

HTML Tags

- When the Web was in its infancy, these tags were all put into each document individually, because the pages were static i.e. once they were created and uploaded to the Web host, they did not change.

HTML Tags

- They might have been linked (and interlinked) to give the effect of navigating through a dynamic document set, but each page was a single document, edited by hand.

HTML Tags

- This meant that changing some of the elements required multiple changes on the page; if the designer wanted to insert new levels of heading, that would have knock-on effects through the rest of the page.
- Some headings would have to be promoted and others demoted.

HTML Tags

- For example, there might have been ten headings on a page, each denoting a piece of information of diminishing overall hierarchical importance.
- Heading 1 might have been the page title, heading 2 a section title, heading 3 a sub-section, and so on.

HTML Tags

- Each heading would have to be detailed separately in terms of size, color, and decoration, and HTML provides a set of relative hierarchical tags (H1, H2, and so on) that you can enclose the heading text in to provide decoration.
- This takes some of the pain away, but not all.

HTML Tags

- You still had to change all the H3 tags to H4 if you wanted to insert a new heading item.
- For menu management especially, this became a painful experience.
- It meant in some cases that you would have to go through a search and replace exercise whenever you wanted to insert a new menu item.

HTML Tags

- In addition, once HTML became more precise and offered the possibility to create very detailed tags to specify the exact font, size, and color, as well as discrete position of content on the page, designers became ever more ambitious.
- They were still manually editing the pages.

HTML Tags

- They had tools that made the management of the HTML itself easier, but few tools to maintain resemblance between multiple pages if the style of the content had to change.
- Let us assume, for example, that a company had a Web page where regular content was in green text and menus in black.

HTML Tags

- Each piece of text has to be marked up as being in either black or green.
- What happens when the company decides that green is no longer in vogue and wants to change it to red?
- The Web designer has to go back and change all the mark-up on the static pages so that the content is in red.

HTML Tags

- A far easier solution is to use a standard tag to describe the content in an abstract fashion (as either Content or Menu text) and then define the fact that Content text is in green and Menu text is in black in a single point.

HTML Tags

- This mechanism is known as defining a style, and the place that it is usually defined is in an external style sheet.

HTML Tags

- The separation of the positioning and flow information and the actual style details allowed designers to change the individual artifacts with a single change to the style sheet rather than making multiple changes to static pages.

HTML Tags

- The pages could still be static, but it was possible to change the look and feel by simply swapping the style sheets.
- This is a trend that has been further enhanced by server side programming, allowing for users (visitors or browsers) to select style sheets dynamically so that their experience is customized accordingly.

Style Sheets

- A style sheet is a concretization of style information referred to in a page of HTML; each tag can be customized by changing its style.

Style sheets

- Style sheets give guidelines for the style of all the classes of elements in a page, where the Web designer has determined that they should be different from the default.

Style Sheets

- Physically, a style sheet can be a separate document, downloaded along with the HTML, or it can be part of the HTML itself.
- The style information enhances the way that the standard HTML tags are displayed, and is downloaded to the client.

Style Sheets

- You might give styles for headings, general text, paragraphs, shading, tables, and so on that override the defaults that are chosen by the browser application manufacturer.

Style Sheets

- Remember that the W3C does not actually mandate any rendering information, but provides a comprehensive framework to support almost any rendering possibility.

Style Sheets

- In addition to enhancing standard tag styles, you can name element classes.
- A class is a specialization of an element, to which you give a name that is meaningful to you.

Style Sheets

- So, you might create a collection of named classes and have different kinds of tables, paragraphs, headings, and textual elements.

Style Sheets

- For example, you might decide to have a different colored background for certain types of paragraphs, and enclose these with the standard paragraph tag, but with a named class for each kind of paragraph.

Style Sheets

- These new classes then become specializations of the default.
- You can change the default independently of the new classes, making it a very powerful mechanism for altering the style of the pages from a central point.
- However, this is still static presentation.

Style Sheets

- Dynamic presentation layer generation is also possible, on two fronts:
  - You can generate style information and attach it to the page that is being downloaded. This requires server side scripting and is not overly complex.

Style Sheets

- Secondly, you can write a script that is part of the downloaded page, but that changes the style information dynamically within the page. The difference might not be apparent at first, so an example might help envisage the two mechanisms.

Style Sheets

- Assume that you want to allow visitors to change the layout of the page so that it matches more exactly their screen size. You do this by attaching a style sheet to the page that contains layout information.

Style Sheets

- This layout information will include discrete positioning for the various elements (menu bars, navigation and advertising sections, and so on) using named tags.

Style Sheets

- To keep the example simple, assume a vertical Menu Section that is 25% of the screen width, a middle Content Section that is 50%, and an Advertising Section that is 25% of the available screen width.
- The user has a small screen, and the Content Section is unreadable, so you want to change the proportions to 15%, 70%, and 25%.

Style Sheets

- You have two choices
  - you can generate a collection of styles that meets these proportions using a server side script, or
  - you can create a client side script that does that for you.
- Either mechanism is acceptable in this case, but each requires a different set of scripts.

Style Sheets

- In the first case, you dynamically generate the content specifically for each visitor; in the second, you manipulate the styles using a script that is written once for all users.
- In reality, you will most often use a combination of the two approaches.

Style Sheets

- This level of dynamic generation is also often a result of data exchange with the server, and again, it can be either the client or server that generates the resulting HTML.

Style Sheets

- So, you might write a script that can dynamically detect the resolution that the user is using and adjust the page accordingly.
- This is where the first implementation is weaker.

Style Sheets

- You would need to communicate with the server to tell it what kind of screen the user has, and only then download the correct layout information.
- In the second case, you could detect and update the layout locally.

Style Sheets

- You can also change styles dynamically, and asynchronously, as you communicate with the server, usually in reaction to something that the user has done.

Style Sheets

- So, if you have detected that the screen resolution is 640 X 480, and rendered the page accordingly, you could allow the user to change that by selecting a different rendering style.

Style Sheets

- If there were more items to change than just the layout (the color, for example), you could also handle that by regenerating the page or dynamically altering the styles.

Style Sheets

- The locally scripted solution is the only possible solution in cases where the Web host does not offer support for server side scripting.

Style Sheets

- So, style sheets allow more control over the rendering, with the possibility to group rendering information into collections, which can then be dynamically altered.
- Again, this is via an open standard mechanism that is maintained by the W3C.

Style Sheets

- In the same way that you need a standard way to describe the documents, you also need a standard way to exchange the data that they contain.

XML

- HTML is based on a standard called XML, which allows you to define your own content delivery specifications.
- In short, HTML is an application of XML, but formally, the new standard of the Web is XHTML.
- XHTML is a formal redefinition of HTML under XML.

XML

- This makes the HTML code easier to validate, as it adheres more closely to the XML standard for data exchange.
- XML is a mechanism by which data can be exchanged with a server.

XML

- This data can be used in a variety of different application areas, and is not just used for Web programming.
- The widest use, until recently, was as a way for bloggers and other information, content, and product providers to inform their fans when new items were available.

XML

- For example, RSS (Really Simple Syndication) is based on XML; these are feeds of data that are designed to be read and displayed.
- If you open an RSS feed in a Web browser, the structured XML is laid bare.

RSS XML

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Liftoff News</title>
    <link>http://liftoff.msfc.nasa.gov/</link>
    <description>Liftoff to Space Exploration.</description>
    <language>en-us</language>
    <pubDate>Tue, 10 Jun 2003 04:00:00 GMT</pubDate>
    <lastBuildDate>Tue, 10 Jun 2003 09:41:01 GMT</lastBuildDate>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <generator>Weblog Editor 2.0</generator>
    <managingEditor>editor@example.com</managingEditor>
    <webMaster>webmaster@example.com</webMaster>

    <item>
```

Figure 2.2
RSS XML example

XML

- On top of which, the HTML standard is well defined, and does not allow you to casually start inventing tags that can contain the data that we need.
- Instead, you need to have the data returned in a way that allows you to extend the basic definition by adding your own definitions.

XML

- One such mechanism for data exchange is called XML, which is an open standard for data exchange that allows for the creation of data elements and attributes.
- Others include JSON, which requires special handling in most browsers and is not usually decodable by the platform itself.

XML

- A mixture of client and server side technology is needed to allow a user to display feeds in a browser.
- For example, the server can offer the feeds, but the client needs to be able to display the elements that are contained in each entry to the user in a way that enables them to look at them in a meaningful manner and possibly click on links that are included as part of the feed information.

XML

- Part of the browser scripting language has to allow one to manipulate the XML data that is returned—building something that has meaning from the structured data stream.
  - This requires some clever client side programming, because XML manipulation is not an innate part of the HTML specification.

XML

- However, it is a part of some standard scripting language implementations that allow the Web programmer to make a request using a special kind of HTTP request—XHTTP Request—and process the data that is returned.

XML

- Like all areas of Web programming, the client side programming languages (scripting) have to be reasonably standardized to allow producers to be reasonably certain that consumers all experience the same effect regardless of platform.

Client Side Scripting

- Client side scripting is a way to extend the functionality of a Web page beyond static data by requesting that the user agent (the Web browser, for example) do some additional processing on behalf of the Web programmer.

Client Side Scripting

- This can be for the purpose of making the Web page interactive, dynamic, or just animated in some way.
- The content, layout, and other features used in the rendering of the page can all be manipulated using client side scripts.

Client Side Scripting

- This makes them very useful for displaying items that are platform-, browser-, or user-dependent.

Client Side Scripting

- Client side scripting allows you to validate forms, change the document with respect to interaction with the user, and manipulate style information dynamically, as well as generating new content.

Client Side Scripting

- Like all programming languages, the client side scripting mechanism usually allows for selective execution (decision making), flow control (loops), and other programming constructs.

Client Side Scripting

- The script interpreted is either built into the browser (JavaScript, for example) or is interpreted via a plug-in that has to be downloaded from the manufacturer (Flash, for example).

Client Side Scripting

- A plug-in is an external application that is automatically invoked by the browser when needed.
- Because it is an external application, the plug-in is operating-system-specific.
- The plug-in is associated with a data object—generally using the file extension—to allow the Web server to properly handle data that are not originally supported.

Client Side Scripting

- For example, if one of the page components is a PDF document, the Web server will receive the data, recognize it as a “Portable Document Format” object, and launch Adobe Acrobat Reader to present the document on the client computer.

Client Side Scripting

- The XHTTP mechanism that allows you to do Asynchronous JavaScript and XML (AJAX) is an example of such a case; Microsoft Silverlight is another.
- In both of these, standard JavaScript and XML programming can be used, and the relevant plug-ins are either freely available or built into the browser.

Client Side Scripting

- The implementation of the scripting language lets you access the content and layout information through the document model.
- This lets you communicate with objects, be they built-in (like HTML tags) or external (like third-party plug-ins).

Client Side Scripting

- This document model also lets you find out information, programmatically, about the client environment (browser, platform, user, and so on) that can help you to generate relevant content on the server side.

Client Side Scripting

- For example, you might detect that the user is running Windows, and therefore only display advertising that is relevant to Windows users.
- Or, you might use a cookie to store information about the user's last visit to the site and use that to try and help the user navigate through the site when he or she returns.

Client Side Scripting

- Both of these examples will require a certain amount of server side programming, which allows you to generate content selectively to be sent to the browser.
- Server side programming powers the bulk of Websites on the Internet today.

Server Side Programming

- From the actual HTTP server (such as Apache or IIS) which handles requests and delivers data, to the various Webmail, interactive applications (such as Facebook, MySpace, YouTube, and so on) without some logic on the server none of these services would be possible.

Server Side Programming

- There is a difference between a server side application, such as the Web server, and a server side script.
- Server side scripts are interpreted by a server side application.

Server Side Programming

- In order to make Websites more interactive, dynamic, and customized, some form of server side programming was required.
- The first pages were static text, and this gave way to style sheets and client side scripting.
- However, as the technology expanded, Web programmers saw the need to go a step further.

Server Side Programming

- Programmers started to make little programs that were designed to run in the background and generate content.
- They ran natively as applications capable of interfacing with the Web server and provided functionality from databases to content presentation.

Server Side Programming

- However, this not a good solution as all operating systems are different, so there was a need to find a standard scripting language which would work everywhere a Web server was present.

Server Side Programming

- The reasons for this were many, but the chief features are that:
  1. The scripting language can be mixed with HTML
  2. The scripts run on, and have access to services provided by, the Web server.

Server Side Programming

- A scripting language is also interpreted i.e. the manufacturer can distribute an interpreter built for a given platform (a Windows version, a Linux version, a MacOS version, and so on) and all users can exchange scripts without having to build them for each platform.

Server Side Programming

- This is another advantage of using a scripting language rather than building little applications that run on the server i.e. code can be written once and run everywhere.

Server Side Programming

- Over time, the community has settled on three possible contenders for server side scripting:
  - ASP (for Microsoft, mainly),
  - Perl, and
  - PHP, which are multiplatform.

Server Side Programming

- The difference between a client side script, such as a JavaScript which is also embedded in the page, is that the Web server will interpret the PHP that it encounters and send the result of that processing to the client rather than the text of the script itself.

Server Side Programming

- So, if you wanted, you could have PHP code to generate JavaScript (or HTML, or style information) which is then delivered to the client.

Server Side Programming

- So, server side scripts have the advantage of being invisible to the user as the server interprets them at the same time as it serves the Web page.
- Because they are interpreted on the Web page, they can also be quite lengthy and complex; assume that there is more processing power available to the server than the client.

Server Side Programming

- Being able to build complex applications, coupled with the Open Source philosophy of the PHP community, means that many frameworks have been implemented.
- Some of these are application frameworks and extensions, and some of them are complete Web content delivery systems, made available to the general public.

Server Side Programming

- The key advantage of using them is that it gives a good starting point for code of your own.
- Provided that it is Open Source, and provided that the license permits it, you can customize the PHP scripts to your own needs.
- There are many frameworks that actively encourage this.

Server Side Programming

- In the main, these are CMS (or Content Management Systems), which allow you to create complex document delivery systems.
- Blogger is an example of a fairly simple CMS implementation.

Server Side Programming

- More complex ones allow for categorization of documents, advertising placement, dynamic menus, and other features.


JOMO KENYATTA UNIVERSITY
OF AGRICULTURE AND TECHNOLOGY
TECHNOLOGY FOR DEVELOPMENT

END