Laureate Online Education   Internet and Multimedia Technology

Module: MSC MD (INTMMT)   Seminar 6

Hypermedia, XML, Scripting

Introduction

In the last seminar we laid the groundwork for the production of a simple document that we could use to contain our multimedia production. This seminar we will look at how we can add value to the document by providing links to other documents and other files in general. We will look at the concept of hypertext; text with a hidden purpose, which most commonly provides links to other files, and how we can manipulate our document and its environment via scripts. Given that we accept the idea of a jump to another location using a link embedded in the text then it follows that we must be able to answer some questions about our navigation in this new world. Consider a traditional document; it can allow us to navigate from one page to the next in a linear fashion like reading a novel or we can jump from one section to another like some early self-teaching books required. (You would read a section then answer a question and, depending on your answer, you would be sent to a particular page). In either case, each move depended on the content of the previous move. In these documents you are constrained to a particular world; that of the document itself. The navigation tools are those of an index and possibly a cross reference table. Now we turn to navigation through a hypertext link. Here the world is open ended, one link leads to a page possibly in another document (in fact, perhaps not even in a textual document) located anywhere in the world which, in turn, can lead to another page and so on. In this world, even the starting point may be relatively random. (When you look up an item on the web, you do not always start with the first item found but you would start a novel at the beginning).This is a fundamental change in the rules. How are we to know where we are, where we were and where we can meaningfully go next? These issues are not satisfactorily resolved today but efforts are being made under the broad goal of producing a semantic web (Web that understands our needs and purpose). To do this we require further abstraction of our documentation rules and we need standards and structures which all can understand. We need not stop at hypertext providing further text. Why not have it start a programme or control an outcome derived from another programme? We could use a traditional language to provide this flexibility but we would like to stay within our web based world and be as platform independent as possible. This is achieved by the use of scripting languages and the scripts they produce. These scripts can manipulate content or allow interactivity with another application and so change the nature of our baseline document again. How can we control this interactivity, how can we make it secure and how can we ensure ethical use of these tools?

Most of this seminar is taken up with looking at these relatively new approaches to interaction with our media. A shorter final section looks at the implications for the networks and the protocols needed to enable our hypermedia world.

Hypertext

We should differentiate between the WWW and its use of hypermedia content and the concept of hypermedia. Hypermedia is often credited to Vannevar Bush and his paper entitled 'As we may think' (Bush, V. 1945.) written in 1945 long before the web became a reality. His paper suggests a device for augmenting a human mind and is immediately recognisable in its modern incarnation as the personal computer. However, he made an important comment on how this mass of information should be indexed and controlled; not by traditional indexing but by association of elements. To quote:

> "It is exactly as though the physical items had been gathered… from widely separated sources and bound together to form a new book. It is more than this, for any item can be joined into numerous trails" (Bush 1945).

He suggested a mechanical device he christened Memex, but his system of annotation and cross-referencing was never implemented.

The first hypertext system was that of Engelbart known as NLS (oN Line System) in the 1960s at Stanford Research Institute but it was Nelson who first coined the word hypertext in 1965. Soon scientific discussion was no longer about documents but about nodes and links between them. As we now take for granted, it is the links, which define the richness of our hypertext (and hypermedia) experience. Conklin (1987) proposed that we could define two types of links; referential links and organizational links. Referential links connect a source in a current node to a reference in a destination node whereas organisational links connect a superior node to its subordinates in a tree. A much-quoted early system is the 1981 Xanadu project of Nelson (Wolf 1995). Again, this project has not materialised in its original intent, which was much more than contemporaries envisaged for the internet. The concept was the sale of documents across a network with automatic royalties on every byte. To achieve this would mean the cross referencing of all parts of all literature! This might sound far fetched but already standards are being developed which will allow this for all aspects of hypermedia. (You are referred to the MPEG 7 (Cover 2002) and MPEG21 (Bormans & Hill 2002) standards for more information should you be interested in studying this further). Bill Atkinson (Apple Computers) developed HyperCard (Timeline n.d.) in 1987. The application was free with all Macintosh machines. HyperCard, for a while, became the most widely used hypertext system and many HyperCard applications were developed. All this occurred before the birth of the WWW in 1989.

Educationalists are quick to point out that hypertext links facilitate memory and understanding and this has lead to encouragement of WWW use in the classroom. (You will pursue this open question in a DQ). Yet they often ignore the associated problems. How often have you started at one point with a specific purpose in mind and found after several links that you have shifted the purpose of your search (Barabasi 2002). You have become disoriented by the number of links and overloaded by the shear quantity of the information available and its very fluid scope. This is cognitive overload brought about by an inability to see all the information in one place (one link replaces anther). We can recognize this in our own research. There is no author to guide us through the maze, but compare this to a structured text; only the required links are made available or they are pre-selected for relevance. Hypermedia has added variety and additional information (we can argue on that one) but it has an added cost for navigational complexity.

Earlier we mentioned nodes and links and it is this way of thinking about hypertext that allows us to divorce ourselves from the idea of fixed content and brought the idea of browsing into being. Browsing means we can take whatever path we wish through a sea of information but it also demands new concepts if we are to return to our starting point. To find the way graphical approaches have been used showing the links and the nodes but this soon becomes clumsy and cluttered; particularly if you loop back on yourself. Today, we are firmly fixed on the use of searching, indexing and history as the means to move successfully around the web. You may feel that there is nothing much new in searching and indexing but the use of history files to gain easy access to known areas is. Actually, they have all seen considerable improvement with the advent of the web. Browsing is the human approach to searching by using association but computers search through databases using indexes. We can consider our nodes to be URL’s (URI’s, if we wish to be up to date) which can be organized according to certain keys (keywords) for later retrieval from the database. A further refinement is the weighting of results according to the number of links pointing to a URL. The assumption being that more important documents will be referenced more often. Unfortunately, this has lead to abuse and to the predominance on the first page of a search result of other search lists. The search for the semantic web is a response to the need for a better understanding of what a user wants to retrieve. This is currently not available so we are using descriptors to tell the search engines what the file contains. These are meta-data tags which provide a description (a title) and a set of keywords. Look for something similar to the following on a web page:

```
<meta name=”description” content=”Lack of Brains hinders research”/>
<meta name=”keywords” content=”missing, anatomy, Columbus Dispatch, silly
headlines”/>
```

The above also demonstrates the unlikeliness of finding what you want given the first two keywords chosen to describe the article. ☺

We have discussed the links but what about the nodes? To what are we linking? Currently we link to a Uniform Resource Identifier (URI) although we are more used to the special subset known as a Uniform Resource Locator (URL). Immediately we are made aware that this is a resource and much more inclusive than a document. Web pages are described by the protocol, http, the domain name of a server running the protocol and the path to a page or script which can generate the page. One such script designed as a standard for interfacing between resources and the web is the common gateway interface (CGI). The process is shown in figure 1. In figure 1 we can see that a URL is sent to the browser that, in turn, generates an HTTP protocol 'Get' request to the server. If the server finds an extension which it recognizes as a script reference, e.g. .cgi, then it hands over to a 'gateway programme' which generates the required page. This new page is sent to the browser for display. Microsoft use the Active Server Pages (ASP) scripts to achieve the same end.

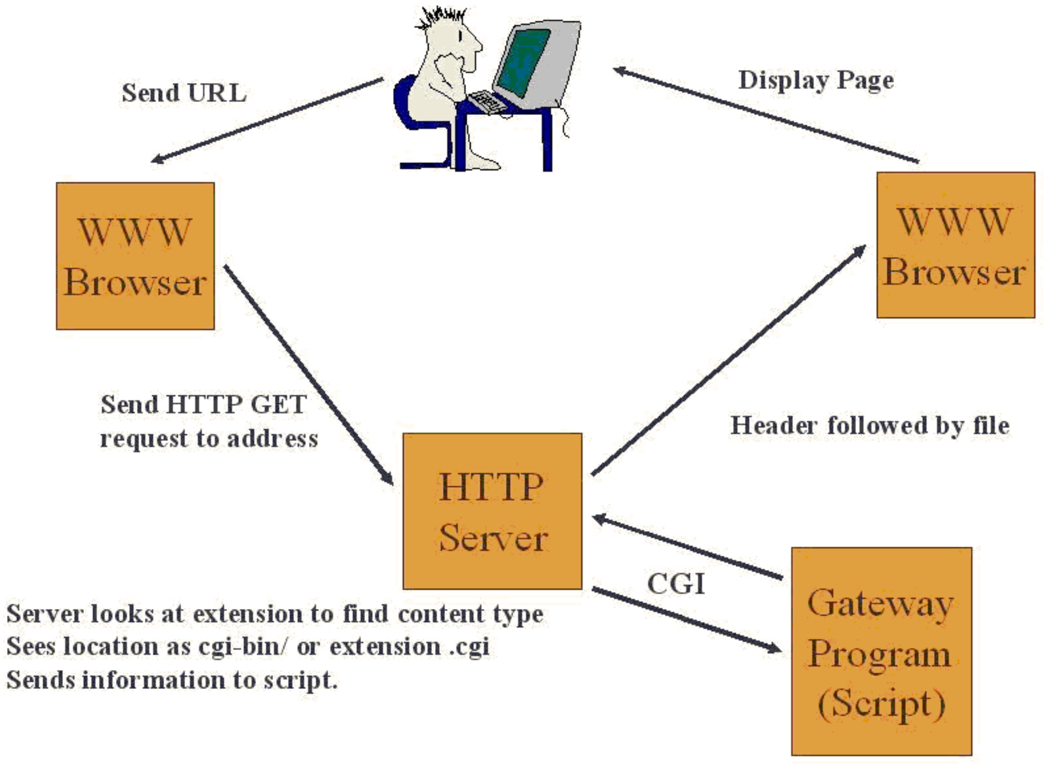
Send URL
Display Page
WWW Browser
WWW Browser
Send HTTP GET request to address
Header followed by file
HTTP Server
CGI
Gateway Program (Script)
Server looks at extension to find content type
Sees location as cgi-bin/ or extension .cgi
Sends information to script.

Figure 1. The web and interaction with CGI scripts

It is not the intention to go into more detail at this stage as you will meet this topic again in the IN – Programming the Internet module.

When using HTML our links use the href attribute with a URL pointing to the page we want. Actually, it does not matter whether the destination is a page of HTML or not since the server will return the requested item with the data type in the MIME header. The server maintains a list of correspondences between file extensions and Mime types so that it can build the header content. Then at the receiver end, a browser helper application will take the appropriate action. This may be to start up Cosmo for a .wrl type or Adobe for a .pdf file. Plug-ins are also used. These are software modules that are loaded by the browser when it starts up. Well known plug-ins are those that deal with video and proprietary graphics files such as Flash from Macromedia or Cosmo for VRML files from SGI. Basically, plug-ins are made available wherever it is realised by the browser manufacturer that the application developers have better skills when dealing with their own products.

Several times we have mentioned the idea that our resources are not always text based and that implies that our means of dealing with this diversity requires us to consider more than just the layout of the document. We are familiar with HTML as a common means to manipulate and separate content from structure, unfortunately its repertoire of tags is limited. You may think that simply extending HTML tag definitions is acceptable but unfortunately this would ultimately lead to a massive increase in tags with increasingly narrower usability. What is needed is a means to define your own tags yet simply allow the end user application to interpret them. The solution lies in the Standard Generalized Markup Language (SGML, ISO 8879) and more specifically in its derivative form XML (eXtensible Markup Language).

XML

It is the international standard for the definition of device independent means of representing texts in electronic form. You should note it is concerned with the ‘definition’ of methods and not the methods themselves. This means that SGML is a meta-language or a language describing a markup language. SGML introduced the idea of a document type and therefore a document type definition (DTD). This formalizes what constitutes a particular document type and the tags that are allowed within it. The definition of a seminar document, for example, could be that it consists of a copyright notice, a title and a module code, followed by an introduction and a sequence of paragraphs. Anything lacking these items, or part of them, or not having them in the specified order would not meet the DTD specification of a seminar. XML was specifically designed for the internet and became a W3C: World Wide Web Consortium (WWC 1998) recommendation in 1998 after just two years of development but it follows the same organization as its originator SGML. Thus, XML is itself a meta-language! Developers have used it to produce several well-known DTD’s. Amongst these, and of special interest to us in the next seminar, are SVG (Scalable Vector Graphics and SMIL (Synchronised Multimedia Integration Language). Another is XHTML which implements HTML using an XML DTD. This makes XHTML a fixed version of XML for use on the web. Its implementation typically involves a software application known as a ‘parser’ or XML processor that can read the document type definition and validate any document using that particular DTD. Parsers will also produce a new version of the document formatted according to application requirements. (This includes checking for file types that the application cannot understand or invalid URL’s). This new form can then be used by other pieces of software to provide additional functions, such as formatting and database management.

XML syntax is familiar since it uses tags which contain elements e.g.<tag> and </tag> and attributes which can be assigned values. It is also possible to pass instructions to the XML processor using a declaration such as <!...> e.g. <!Doctype MySeminar>. These declarations apply to all processors but you may require a specific target for your processing instructions and this is done using <? ?> e.g.

```
<p> Why not have a pagebreak now
<?AcommonWP (new page) ?>
<?Anotherone  DO:page –break?>
<pagebreak/>
</p>
```

Since the way browsers manage unknown tags is to ignore them, then the application is assumed to understand its own version of the command but no others.

You may feel that every document must have an attached DTD but it would be very wasteful of resources. Fortunately, the DTD can be stored in an external subset of the file so that other files can reference it. In fact, it is possible for a document to reference different DTD’s as needed.

To display XML satisfactorily requires the use of style sheets. This is because XML has nothing but rudimentary layout controls. Look at the following example.

```
<title> This is an example </title>
<intro><para> Style is a value judgment</para></intro>
<para> There are no <em>good</em> or <em>bad</em> styles</para>
<Beware><para> style is subjective </para></Beware>
```

If our XML processor can only remove the tags then we would have:

This is an example Style is a value judgment There are no good or bad styles style is subjective.

Barely readable and distinctly bad style ☺ . However if we can provide a style sheet to further interpret the tags, we could produce the following:

This is an example

Style is a value judgement

There are no ***good*** or ***bad*** styles

Beware: style is subjective

This time we have some bold script, some italic and some highlighted. However, it is still only an example and not something you would wish upon your readers. Notice even a style sheet would not remove a grammatical error. For those who are interested the above can be achieved by giving ID’s to the relevant style, e.g.

```
<style type =”text/css”>
#id-name1 {style definition}
#id-name2 {style definition}
</style>
```

Then you use the ID attribute of the tag to change the style used, e.g.

```
<H2 ID = “style –name1”>
```

There are a considerable number of properties associated with CSS, which we cannot cover here. You may like to look at [CSS] which contains the complete specification for CSS.

Scripting

So far we have introduced the concepts associated with finding and positioning data but we have not looked closely at interactivity with objects related to the document. As we have seen, these objects are not limited in type but they do rely on two associated ideas. We need to establish a connection between an action and an event. Current operating systems are built upon responses to events producing known re- actions. They are known as event –driven systems. We create different events every time we move the cursor or press a key. A simple thing like moving the mouse can create many different events, e.g. the mouse is in motion, the mouse has entered a particular area and the mouse has left the area. Some events are internal for example the ending of another programme or the emptying of a buffer. The operating system responds to each event according to the associated routine. In order for us to use these event handlers we must define a set of actions e.g. mouse button down. Scripts allow us to define what the outcome of a particular interaction will be. Scripting languages were designed to allow the user to control user interface elements and multimedia objects. This is achieved by associating the script with an event via a tag. You might ask what the difference between a script and a programming language is and, to be honest, it is hard to differentiate between modern scripts and standard languages such as C. The difference is partly historical and partly to do with complexity of functionality. Scripting languages are designed to augment the application they are associated with. They are specific to purpose and do not possess the generality of C++ or Java. This module is concerned with multimedia and so we confine ourselves to scripts that deal with sounds, text, film or timelines etc.

The most well known scripting language in multimedia for the Web is probably that used by Flash 5 called Action Script. It is a descendant of the first scripting product for the Web which was known as LiveScript from Netscape and it quickly became JavaScript.. No, it is not a copy of Java (in fact, nothing much like it at all) but it prompted Microsoft to introduce Jscript and began a proprietary war. In an attempt to gain acceptance as a standard JavaScript was offered to ECMA (European Computer Manufacturers Association) in 1996. Slowly the two browsers are becoming more compliant with the ECMA standard finally adopted in 1997. Meanwhile the World Wide Web Consortium were developing the Document Object Model (DOM) to allow browsers to interface to standard definitions of objects. ECMA interfaces with those definitions. ECMAscript became the basis for ActionScript although there are differences.

Before we leave the topic of scripts there are a few comments to be made on the programming environment for the web. There are two environments for Web scripting; one is with the client and the other on the server. This gives rise to the terms, client – side and server – side. For the moment let us look at client – side scripts for controlling media objects obtained from the server. Much is made of the security threat to the host from malicious scripts. Scripts running in a web browser are forbidden access to local resources or files and can only interact with the server to request other resources or post information gathered from web forms. The DOM allows ECMAscript to interface to an object called **document** which, unsurprisingly, allows the script to interact with a webpage. This is achieved by burying the script in the document (using the script element) so that when the browser reads the page it executes the script. Here is an example:

```
<html>
<head>
<title> Hello World </title>
</head>
<body>
<script type=”text/javascript”>
document.write(‘<h1>’ + document.title + ‘</h1>’);
</script>
</body>
</html>
```

If you try this, you can see that the write method of document has been used to insert ‘Hello World’ into the body header of a standard web page. Obviously we could use the same idea to include further text using variables to hold it.

It is not the purpose here to provide a comprehensive introduction to scripting language constructs and data types since you will find that in future modules. (As people with even some programming experience, you will find no big surprises even if you have never used a scripting language before).

Finally this week, I want to make some comments on the web, multimedia and the network

We have come to the end of a very mixed seminar on Hypermedia, XML, and Scripting. Its intent has been to introduce a collection of topics, which make the WWW a multimedia experience. The detail is covered in the IN and WA modules so I have tried not to pre-empt them. There is a considerable amount of background reading that could be attached to this seminar and I would ask you to be selective at this point in the programme. For that reason, I am only going to ask you to read key parts of the text this seminar. (You may of course scan or ‘skim’ the others.)

Links and References

Barabasi A.L. (2002), The new Science of Networks, Plume Books, ISBN: 0452284392
Interesting discussions on the subject of links in many branches of science and society in general and the Internet in particular (chapter 12 and 13) will be found in: Barabasi, A.-L.(2002) Linked: and in the Web Site:
http://www.nd.edu/~alb/
(Accessed: 23rd September 2010)

Bormans J. & Hill K. ed. (2002) Coding of Moving Pictures and Audio [Internet]
http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm
ISO MPEG21 overview
(Accessed: 23rd September 2010)

Bush V. (1945), As We May Think
The Atlantic Monthly, July 1945. Available from;
http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm,
A look at the future of meaningful human data retrieval
(Accessed: 23rd September 2010)

Conklin, J. (1987). Hypertext: An introduction and survey. Computer, 20(9), 17-41.

Cover R. ed (2002) Mption Picture Experts Group: MPEG-7 Standard [Internet]
http://xml.coverpages.org/**mpeg7**.html
An overview of MPEG 7
(Accessed: 23rd September 2010)

CSS The specifications for CSS can be found at
http://www.w3.org/Style/CSS/
http://www.w3.org/TR/CSS1/ (2008)
http://www.w3.org/TR/CSS21/ (2007)
(Accessed: 23rd September 2010)

Timeline Anon (n.d.) Timeline of Computer History [Internet]
http://www.computerhistory.org/timeline/
This is an interesting site for facts on the history of computing in general.
(Accessed: 23rd September 2010)

Wolf G. (1995), The Curse of Xanadu, Wired Issue 3.06 [Internet]
http://www.wired.com/wired/archive/3.06/xanadu.html
(Accessed: 23rd September 2010)

WW3 The World Wide Web Consortium will be found at: http://w3c.org
(Accessed: 23rd September 2010)

Tools:

There are many HTML editors (See http://www.tucows.com). A recommended one is the free web page creation utility AceHTML from Visicom Media. It can be found in Tucows (http://www.tucows.com/preview/194452.html) or in Download.com (http://www.download.com/3000-2048-10014527.html). This is a basic textual html /xhtml editor that does not automatically generate most code, though it will provide you with a starting point for a Web page.