

Chapter Three

Web Technologies¹

The most dynamic, but arguably also the most confusing and daunting aspect of e-Commerce, is the huge array of different technologies used to enable e-Commerce. Although a simple static website can be created in a matter of hours, it becomes a completely different ballgame if you are in charge of the website of, say, a national bank or retail chain.

Luckily, most e-businesses and website designers never have to investigate the more esoteric technologies; either because there are no complex “legacy” (i.e. old technology) information systems to interface with, or else because a lot of the ugly complexity is buried deep inside the innards of advanced website building tools such as DreamWeaver or Visual Studio.

The intention of this chapter, therefore, is not to explain all of the e-commerce technologies. That would be impossible, even in a book ten times its current size. Rather, this chapter gives a *bird’s eye* overview of the more common technologies. The particular emphasis is on the context by which these technologies were introduced. It is important to realize that technologies don’t appear out of nowhere. They evolve in response to deficiencies in existing technologies and are driven by the need for more sophisticated e-commerce capabilities. In the “trade”, all of these technologies are commonly referred to by their *acronyms*. This gives rise to the so-called “alphabet soup” of technology TLAs (Three-Letter Acronyms) which is often experienced as a barrier to outsiders and newcomers. Some of this techno-babble is actually unavoidable, just like anyone entering the medical profession needs to master its unique terminology (refer to Appendix 6).

The sheer number of E-commerce technologies means that only the more important ones can be discussed. Unfortunately this means that the very exciting technologies relating to hardware (e.g. networking “boxes”), telecommunications (e.g. the various wireless protocols) and M-commerce (such as WAP) have been left out (see chapter 11 for some information on these). Even for those that are included, relatively little space can be devoted to each technology. This may give a somewhat abstract feel to the discussion and the reader is encouraged to explore more detail on the numerous website resources devoted to each particular technology. Some starting points for exploration are given at the end of the chapter. To impart a more concrete understanding of some of the most critical technologies, a few sections include specific examples but the reader can skip these without loss of continuity.

This discussion is organized according to evolutionary successive generations of technologies.

¹ The author of this chapter is Dr. JP van Belle, Department of Information Systems, UCT. This is a chapter in: Cloete E. “E-Commerce: a Contemporary View. Pardus, Cape Town, 2004. pp. 61-78.

In this chapter you will find:

- **The different generations of e-commerce technology**
- **Internet technologies that enable e-Commerce.**
- **The early days of the World Wide Web**
- **Dynamic web pages**
- **Server side technologies**
- **True e-Commerce on the distributed Web**



The different generations of e-commerce technology

The phenomenal growth of the Internet is possibly without equal: what is arguably man's most complex and intricate structure on Earth has been built in just over a decade. Yet, the increasing sophistication of the Web can be divided into a number of distinct waves or generations, with each new set of technologies building on the foundation provided by the previous generation. Of course, a new generation does not replace the previous generation; "older" generation technologies prove to be perfectly adequate for a large number of websites.

The following gives a quick overview of the different generations². The technologies associated with each generation are discussed in much more detail under separate headings.

Generation 0: The Internet prior to the World Wide Web.

The Internet grew out of a 1960s military project to build a fail-safe network, designed to withstand a nuclear attack that would wipe out major portions of the network. This forced the designers to devise a decentralized and very flexible architecture whereby each computer on the Internet can communicate with any other computer on the Internet. This is made possible by giving each computer a unique "address" and defining universal standards ("**protocols**") on how data should be sent. Thus, the Internet became a huge network, consisting of many individual computer networks linked together whereby all computers are equally easily accessible. Also, a user or programmer does not need to worry about how information travels from the one computer to the computer on the other end. This is often visualized by substituting a *cloud* for the Internet to indicate that how the information actually travels across the Internet is invisible or immaterial.

Once these base standards were formulated, the computers on the Internet needed something worth sharing i.e. the key applications that shared information across networks

² Also refer to Appendix 2 for more information.

were identified and standards for each of those were formulated: file transmission, remote control of computers, e-mail, chat rooms, etc.

Generation 1: Static websites and client-server architecture.

It quickly became clear that the Internet was an ideal way to make information available to a world-wide audience. This required a standard in which textual information could be displayed in a consistent format, no matter what type of computer one used. In addition, the use of non-textual elements such as images and sound was also most desirable. Finally, it was important to let a document include direct links to any other document anywhere else on the internet: the concept of *hyperlinks* is now commonplace but a revolutionary idea in those days. These standards led to the birth of the World Wide Web (www) – or the Web, for short – which is, in essence, just another service available on the Internet.

As a reminder (see chapter 2): the web is built on the client/server architecture. On the web, computers that hold collections of web documents available to everyone on the web, are called *web servers*. The computer and, most confusingly, also the specific browser software requesting the document is called the *client* (another technical term is “user agent”). Finally, the process of requesting a document (or any other file) from the server is often referred to as *downloading*; the opposite process – sending a file to the server to make it available to others – is called uploading.

Generation 2: Dynamic websites and n-tier architecture.

Very quickly, people wanted to go beyond the mere presentation of static web pages. Requirements were more advanced visual effects, user interaction such as data entry and associated validation, simple calculations, and personalization of web pages depending on the user’s context. This required the invention of new programming languages, which are referred to as *scripting languages*.

Some scripting languages were designed to be included inside the web documents so that the client could process these instructions e.g. to display visual effects or user input validation. Not surprisingly, this is referred to as *client-side scripting*. Other scripting languages are executed by the web server e.g. to process and store user input or customize a web document with information from another database. This is referred to as *server-side scripting*.

Because server-side scripting usually involves the processing of customized user information, the server needs to interact with another application or a database. For performance or other reasons, the latter will normally be situated on yet another computer: the **application server** and/or **database server**. This 3-level architecture of client talking to web server that is talking to another server is referred to the **3-tier architecture**. In fact, there may be several application and database servers involved: the *n-tier* architecture.

Generation 3: Full e-Commerce and distributed computing.

As companies engaged more fully in E-commerce, their needs expanded quickly beyond that of creating dynamic web pages and processing simple user input. Scripting languages often proved inadequate for the creation and recording of very long and complex user sessions, processing payments, and liaising in real-time with the information systems of business partners (e.g. shipping companies, suppliers, agents) to determine when or where delivery is possible. These required a better way of handling complex transactions, more security and the real-time interaction of very disparate computer applications across multiple organizations. The latter is referred to as **distributed computing**.

The following section will discuss the technologies underlying each of these generations in greater detail. The more advanced or knowledgeable reader should note that, in order to make the following presentation more readable and manageable, many complexities and side issues have been glossed over. This has resulted in a number of simplifications and omissions that, hopefully, a future, longer exposition may rectify.

Core internet technologies.

Although the World Wide Web is only a decade-and-a-half old, it makes use of a number of internet technologies that have been around since the 1960s.

§ Uniquely identifying each Internet Computer: From IP address to domain name.

In order to allow any computer to exchange information with any other computer on the Internet, it was necessary to issue a unique identification to each connected computer. This functions as the computer's address to identify to or from which the information is sent. Because computers handle numbers very well, this ID is in numeric format and, because data is usually stored in "bytes", the numbers are of byte-size i.e. they can take values between 0 and 255 (2 to the power 8).

The decision was thus taken to use 4 byte-sized numbers (or for octets) as the Internet computer address – better known as the **IP-address** (IP=Internet Protocol – see below). For example, the computer that hosts the web pages on "How stuff works" has the IP address 216.183.103.150 (in "**dot**" notation) and the computer on which this chapter is written has the IP address 137.158.185.163.

Forty years ago, computers were expensive and relatively rare, so it was never envisaged that eventually there could be several billion computers around the world. As can be calculated, the maximum number of different addresses that can be allocated is about 4 billion ($2^{32} = 256^4 = 4\,298\,967\,296$; in practice less than that, because some addresses are reserved for testing and other purposes). Although it appears to be a large number, this is only about half of the world's total population and so not enough to meet the needs of the next decade. A number of temporary fixes have been devised such as *dynamically* allocating non-fixed, i.e. temporary, addresses to users that are not permanently connected (e.g. dial-up users); as well as other schemes such as *supernetting* (Classless

InterDomain Routing) to activate unused ranges of addresses and *overloading* (having multiple computers use one IP address). However, a new addressing scheme using 8 bytes, known as **IPv6**, is being rolled out and should meet all addressing needs for the foreseeable future.

Although computers are quite happy with the dot-notation IP-addresses, humans are less enamoured with long sequences of numbers. Thus a way was devised to also allow human-readable computer addresses, such as microsoft.com or uct.ac.za. These are known as *domain names* and are allocated on a country basis by *domain registrars* who keep a list of available and allocated names. There are two naming schemes in operation:

- American and international organizations do not have a top-level country code identifier but append a high-level domain code denoting the *type* of organization namely **.com** for commercial organizations, **.org** for non-profit organisations, **.edu** for educational institutions, **.gov** for governmental bodies, **.net** for networking companies and **.mil** for military organizations.
- All other countries append the two-letter country code (as agreed by the postal services) to the domain name e.g. .za (South Africa), .jp (Japan), .uk (United Kingdom), .to (Tonga) etc. In some small countries, no other domain type information is required e.g. both post.be (Post Office, Belgium) or wbag.at (Wiener Börse/Vienna Stock Exchange, Austria) or helanta.sh. However, most countries also use a second-level organization type domain code. Confusingly, some Asian and Latin American countries use the same organisation code convention as the US; whereas most European and African countries use **.co** instead of **.com** for commercial organizations and **.ac** for academic institutions (i.e. equivalent to **.edu**). Thus one encounters for instance company domain names such as americanair.com, malaysianairlines.com.my, saa.co.za, and easyjet.co.uk; as well as university domain names such as mit.edu, nus.edu.sg (National University of Singapore), and uct.ac.za.

In the last few years, new high-level domain names has been proposed (.info, .biz, .name, .aero and .museum), many of which have already taken effect.

To send messages, the computer needs to translate this domain name into a dot-format IP address. Several databases are kept which map or link each human-readable domain name to its IP address. These databases are known as *Domain Name Servers (DNS)* and the process of finding the IP address for a domain name is referred to as the DNS lookup. DNS are probably the most heavily used databases on the planet. Hence they also form the Achilles heel of the Internet. The domain names feature in e-mail addresses (president@whitehouse.gov) as well as website URLs (see under HTTP below).

§ The key to sending information across the Internet: TCP/IP

The Internet links an extremely diverse population of computers, ranging from very large supercomputers and IBM mainframes to small stand-alone PCs and Macs running a wide variety of operating systems. There are even coffee machines, vending machines, fridges, and photocopiers connected to the Internet. In order to enable them all to communicate with each other, they must agree on a basic standard for sending each other information.

The core common standard is known as the **TCP/IP protocol**, which is actually a set of two complementary protocols (a protocol is a communications standard).

The most basic of the pair is the **Internet Protocol (IP)**: it is concerned with how messages containing data are sent. Messages sent across the Internet are known as **traffic** and the underlying idea is to have a standard size message. Large messages, such as data files or pictures, are cut up by the sending computer into relatively small, uniformly-sized chunks known as **packets**. Each packet contains a source and destination IP address, a packet serial number and some other information. These packages are sent individually across the Internet, which means that they can take different routes. They are then reassembled at the destination to reconstitute the original message or data stream. A good analogy is to compare this with how a large amount of bulk goods can be shipped using containers.

As is the case with postal mail, individual packets can and sometimes do get lost. For most purposes, this is not acceptable and it is essential that some error checking and validation be performed to ensure that the entire message is sent and received correctly. This is handled by the **Transmission Control Protocol (TCP)**. (An alternative Internet protocol exists – **UDP** – where the loss of a few individual packets of data is less critical, as in some internet broadcasting and streaming applications.)

§ The various Internet services: FTP, telnet, SMTP, etc.

TCP/IP allows for the reliable transmission of data across the Internet between heterogeneous computers. However, it must be borne in mind that data does not exist on its own: it is generated and used by applications. For instance, there are many email packages available for a wide variety of computing platforms. Each of these has its own interface, different capabilities, and handles email messages differently. Yet these different packages need to be able to send email messages to each other; so a standard is needed to identify the various key elements of an email message such as the sender (field), the addressee, the subject, the date, the body, the priority etc.

The Internet has seen the rise of a whole range of *application-specific* standards, which are implemented on top of TCP/IP i.e. they make use of TCP/IP to send their data across the Internet. There are quite a few of these standards, but the ones you are most likely to still encounter are the following.

- **SMTP (Simple Mail Transfer Protocol)**: used for the sending of email text messages.
- **MIME (Multimedia Internet Mail Extensions)**: allows the inclusion of non-textual elements such as pictures in email messages, multiple message portions.
- **Telnet**: allows the remote control of a computer on the Internet. For instance, when a professor in Cape Town uses his PC to log into and run powerful simulations on a European supercomputer.

- **FTP (File Transfer Protocol)**: used for the reliable and quick transmission of files e.g. when you are downloading software or music files or uploading your website files to a web hosting server.

The early days of the World Wide Web

The scientist Tim Berners-Lee wanted to find a better way of sharing his research information with colleagues. In 1989, he proposed and implemented a set of three internet standards that formed the foundation of the web.

- First, he specified a way of embedding formatting and structuring information *inside* a document so that it would be displayed (more or less) the same, no matter what type of computer or operating system was used. This is the famous **HTML (Hypertext Markup Language)** that uses special keywords (called “**tags**”), to indicate (“**markup**”) the layout or meaning of each section of text.
- Second, he allowed any piece of text to link directly to any other text in the same or any other document anywhere on the Web. For instance, a reader can follow a **hypertext** link from a difficult word in a particular HTML document to a definition, explanation or example elsewhere. This was done by giving each web document its unique identifier: the **URL (Uniform Resource Locator)**.
- Finally, he introduced the client-server architecture whereby the information supplier would load a collection of web documents on an internet web server (computer) and the user (web client) would download and view a document using a browser program (such as Netscape or Internet Explorer). The communication between these computers is governed by the **HyperText Transfer Protocol (HTTP)**, which also supports the following of hyperlinks i.e. redirection to any other computer on the Internet by means of URLs.

Any of these three technologies can be used independently of each other, but it is the power of their combination that created the World-Wide Web.

§ **HTML: a Universal language for formatting and structuring web pages.**

The most important standard is the way text inside a web page is annotated to ensure that it is formatted in a standard way, no matter what computer platform you are using. The fundamental idea is very simple yet extremely powerful and, although you are not likely to do much “**HTML coding**” from scratch, at least a superficial understanding of how HTML works is essential if you ever are involved with the development of a website. We therefore decided to go into a little more detail.

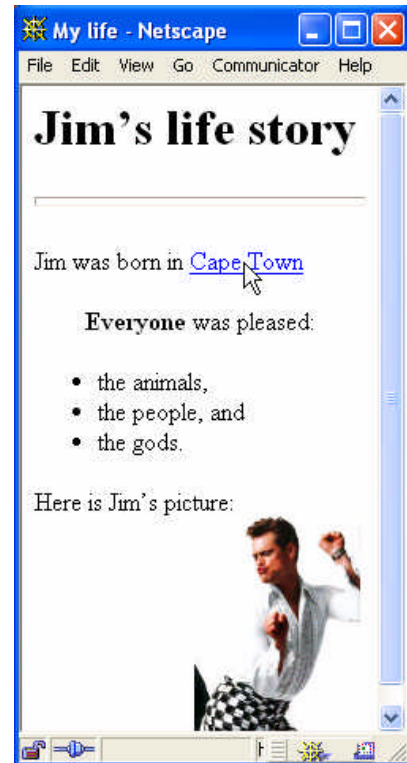
HTML is in essence an agreed set of code words – called **HTML tags** – which are included in the text of a document to indicate the layout or meaning of each section of text – this is referred to as the tag *marking up* the selected text.

- These tags are enclosed in square brackets, e.g. <HEAD> or <TITLE>.

- Tags usually come in pairs and enclose the applicable text, with the end tag identical to the opening tag but preceded by a slash symbol e.g. marking up the word “important” with the *bold* tag in “This is important.”
- There is a long list of defined tags, making up the “**markup language**”, and the language got more complex each time the HTML specification was updated and refined.

In practice, most HTML code will be generated automatically by your web page editor but it is often still necessary to understand the basic principles of HTML in order to understand why certain problems occur. The following example gives an excellent flavour of what HTML looks like.

```
<HTML>
  <HEAD>
    <TITLE>My life </TITLE>
    <META AUTHOR="Jim Bean">
  </HEAD>
  <BODY>
    <H1>Jim's life story</H1>
    <HR SIZE=5>
    <P>Jim was born in
    <A HREF="http://www.visitSA.coza/CT.html">
    Cape Town </A>
    <P><CENTER><B>Everyone</B> was
      pleased:</CENTER>
    <UL>
      <LI>the animals,
      <LI>the people, and
      <LI>the gods.
    </UL>
    <P>Here is Jim's picture:
    <IMG SRC=JimPic.JPEG ALIGN=RIGHT></P>
  </BODY>
</HTML>
```



Although this may look somewhat daunting at first, it is not all that complicated and you are strongly encouraged to spend an extra few minutes.

The following explains what is happening:

- The <HTML>-tag at the beginning indicates that this is an HTML document. Usually, the document will also have a file name ending in .HTML (or .HTM). Appropriately, the end of the document is signalled by the closing tag </HTML>.
- An HTML document usually consists of two parts: an introductory <HEAD> section describing what the document is about and the <BODY> section which contains the real information or textual content to be displayed.
- In this case, the HEAD contains the title of the document (this is displayed in the title bar of the browser window) as well as an indication of the document's author.
- The first line displayed when viewing the page through a web browser will be the text “Jim's life story”. The <H1> tags mark this up as a 1st (i.e. highest) level Heading which most browsers will display in a bold, large font.

- This is followed by a Horizontal Ruler (=line) `<HR>` with a thickness of 5 pixels as specified by the **SIZE** attribute.
- Then follows the first `<P>` paragraph of main text. An example of a hyperlink is shown for the word New York. The part of the document that is the clickable area which is to be hyperlinked to another document is denoted by the `<A>` tags; this usually includes text but can also include images. Its attribute is the **H**yperlink **RE**ference URL where the web user/surfer should be referred to, namely the CT.html document on the website found at “visitSA.co.za”.
- The text in the next paragraph is to be aligned in the `<CENTER>` inside the browser window. Note that the word “everyone” is emphasized in ``old.
- This paragraph is followed by a bullet list (an Unordered List ``) which consists of three List Items (``). Note that a List Item (or Paragraph) is usually followed by another, so there is no need to supply a closing tag `` or `</P>` respectively.
- The page also includes a photo (an image ``) of Jim, which is loaded separately by the browser. The image file name is JimPic.JPEG and is to be found on the same folder as the main HTML document.

Note that it doesn't matter whether tags are in upper case (capitals) or lower case, or mixed. Also, multiple spaces and line breaks in an HTML document are not normally displayed by the web surfer's browser. Generally, these are added to facilitate human readability of the document in an HTML editor.

§ The URL: A standardized way of locating information anywhere on the Web.

The URL is nothing but a unique address to identify a document on a website. A document is identified by the name of the website, the directory (or folder) in which it is located and its file name. The following is an example of a longer URL:

<http://www.howstuffworks.com/computers/programming/java.html#history>

This identifies the protocol (usually **http** or **https**); the domain (howstuffworks.com) with the prefix “www.” indicating that it can be found on the domain's web server (not always necessary); the sub-directory on the webserver (computers/programming/); the name of the document (java.html) and a specifically named target location inside the document: the start of the section named “history”.

There are many alternative forms of URLs. The most common one is the use of **relative URLs** that refer to documents or files on the same server e.g. “./images/happy.gif” would look for a JPEG picture named “happy.gif” inside the folder “images” in the parent directory (indicated by “..”) of where the current (HTML) file is located. Many larger organisations have several servers, each with their own name as a prefix to the domain: commerce.uct.ac.za, its.uct.ac.za, bremner.uct.ac.za... You may also encounter URLs where the domain is appended with a port code or the domain itself is substituted with the actual IP-address in dot-notation, obviating the need to do a DNS lookup.

§ HTTP for the transmission of web pages between the web site and browser.

We will not discuss HTTP in detail, although it is actually a very simple protocol. The client – the browser of the user, also called the **user agent** – sends a “**request**” to a web server asking for a specific web page and the latter “**sends**” the requested document back. This is referred to as a **connection-less protocol**, since no permanent connection or channel between the server and client is set up, as is typical for most Internet communications. The user agent is usually an internet browser such as Internet Explorer, Netscape, Opera or Mozilla but it could also be a **spider** from a search engine which is trying to catalogue the page in its database of web documents. Note that the server also runs a specific type of software to respond to the HTTP requests: the **server software**, such as Apache or IIS.

Certain websites store a limited amount of customized user information on the client computer; this is called the **cookie**. When the browser requests a web page from that website, HTTP will automatically send the cookie along.

Often things do not go as planned. A common error is that no web page can be found at the location indicated by an outdated or misspelled URL, which results in HTTP’s most common error code: the infamous “**404: Page not found.**” A more recent specification, **HTTP-S (HTTP-Secure)**, encrypts the web page to allow for more secure data communication between server and client.

Second generation programming on the Web to create dynamic web pages

The Web soon became popular beyond anyone’s expectations. Naturally, people wanted to use it for more than just the sharing of static web pages: they wanted customized pages, interactivity, processing capabilities, precise lay-out control and advanced multi-media capabilities. The initial response was to extend the HTML language to make it more and more powerful, but this approach had only limited success. What was needed was a way to include custom programming capabilities in web sites. Depending on where the code is executed, one can distinguish between server-side and client-side technologies.

Client-side technologies: Scripting and applets.

The quickest way to create **dynamic web pages** is by using client-side technologies. It is possible to include programming code in the form of a **script language** inside the HTML page (usually in the HEAD-section) although it can put in a separate file. The latest browser software is capable of interpreting this script and executing it accordingly. The most popular scripting language is **JavaScript**, (technically two versions exist:

ECMAScript, which is the official standard version, and **JScript** which is Microsoft's implementation). An example of client-side JavaScript is given below.

```
<script type = "text/javascript">
<!--
alert("hello world");
//-->
</script
```

When incorporated inside an HTML page, this tiny script will pop-up a window with the message "hello world".

Note that JavaScript has an **object-oriented** feel to it, although it is by no means very pure. The language itself has a very short learning curve because of its similarity to other programming languages. The biggest obstacle is learning what and how the various document and application objects can be accessed and controlled. The more advanced dynamic control of the various HTML elements and its browser by means of scripts is referred to as **DHTML (Dynamic HTML)** although this also includes the much richer model of the HTML document specification (and its object specification) to allow advanced formatting control.

Microsoft has also tried to promote its proprietary script language, **VBScript**, which is based on its popular Visual Basic language. However, this can only be executed inside Microsoft's own browser, Internet Explorer, and thus restricts the portability of web pages that have adopted this.

The main **advantage** of client-side scripting is that all processing is done on the computer of the user (client), thus placing hardly any extra load on the web server and making for very responsive effects. The drawbacks are that the added code increases the size of the files, increasing download times. This is exacerbated by the fact that many incompatibilities exist between different browsers. Usually, the programmer has to check the browser type and include different scripts for the various browsers, increasing the complexity and making the process much more error-prone. Also, the scripting code is plainly visible and accessible to any user by selecting the View Source command of one's browser. Most importantly, the type of effects that can be achieved with client-side scripting are limited to the information available on the client end.

The most **popular uses** for client-side scripting are therefore the creation of special visual effects, more flexible layout, user input data validation and limited calculations.

This discussion would be incomplete if two other frequently used approaches were not mentioned. One is the use of **helper applications** and **browser plug-ins**. This refers to the use of proprietary software applications to extend the capabilities of the browser. A well known plug-in is **FLASH**, which enables sophisticated graphics effects. There are many other plug-ins to support various sound capabilities, 3-D graphics or motion video. Many of the more common plug-ins now come standard with the more recent versions of web browsers since most users appear to be reluctant to install additional plug-ins manually.

Finally, it is possible to embed true programming code in the form of **JAVA applets**. Unlike Java/VBScript, Java is a widely used, fully-fledged, pure OO programming language. Apart from its power, it does not suffer (as much) from the many incompatibilities between different platforms, giving rise to the slogan “write once, run everywhere”. It has been specifically designed with various security issues in mind. However, compatibility and performance issues as well as the need to install a Java interpreter have so far limited its success.

Server-side technologies

Much more powerful are the various **server-side technologies**. This refers to software (code) that is executed on the server instead of the client. It receives input from HTML form data, cookies, HTTP headers, or parameters passed as part of the URL (e.g. whenever the URL contains a “?”). The server code then dynamically builds a custom HTML page for the client that invoked the code. The advantages of using server-side scripting are many:

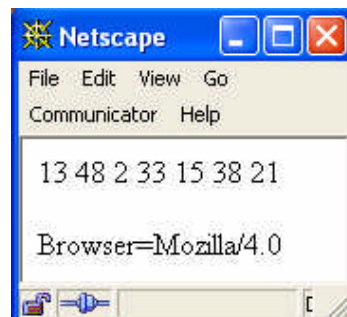
- It can access server-side databases, e-mail servers, mainframe computers etc. (leading to a 3- or n-tier architecture)
- Because the platform (computer) on which the code is executed is known and under full control of the web designer, there are no compatibility problems.
- The code remains hidden from users and is thus secure and virtually tamper-proof.
- The browser does not have to do any processing, which is ideal for performance-lacking Internet devices such as cell phones and other mobile devices.

One major disadvantage of client side technologies is that the server can quickly become overloaded by processing requests. This can be overcome by the scalability of a more distributed network with multiple load-balanced servers and n-tier architecture. The other problem is that these technologies are usually much more complex to learn and implement. For this reason, the discussion that follows will limit itself to a brief overview of the available technologies.

§ CGI-scripting

The oldest form of server-side scripting made use of the **Common Gateway Interface (CGI)** standard and a wide variety of scripting languages were developed. To give you a flavour of a CGI script, consider the following minimal example.

```
#!/usr/bin/perl
# part1: normally processes input
#       and interacts with back-end
$b=$ENV{'HTTP_USER_AGENT'};
for ($q=0; $q<6; $q++) {
  $seq[$q]=int(rand(49))+1;
}
# part2: this section generates the output as HTML
print "Content-type: text/html\n\n";
print "<html><body>";
print "<p>Sequence is @seq \n";
print "<p>Browser=$b";
```



```
print "</body></html> \n";
```

The freely available **PHP** (Personal Home Page), **Perl** (Program Extraction and Reporting Language) and **Python** are among the more popular, server-side scripting languages. They are easy-to-learn yet powerful, although they can often be quite cryptic to debug.

Note that, in a normal scenario, the user would, say, post some user input data using an HTML form to the web server who would then invoke the Perl script. This script would parse and process the input and use it to query, say, a back-end database. In the example above, *part1* merely checks what browser posted the data (and stores it in the variable \$b) and uses a loop to calculate a sequence of 7 random numbers between 1 and 50. The results are then be formatted in HTML as illustrated in *part2* which creates a minimal HTML file with the sequence and browser information. This HTML page is then returned back to the web server that sends the page to the client.

§ Active Server Pages (ASP).

Although very powerful, the CGI scripting languages are very difficult to debug and no user-friendly environments for developing and managing complex websites existed. This led to the development of the **ASP** technology. Although based on the same idea, the technology is much more mature, easier to use and more scalable. ASP pages contain static HTML code, interspersed with server-side code written in a scripting language where customized content is required. Again, the ASP script will typically parse input data and access or update data in a database. The most commonly used ASP scripting languages are **VBScript** and **Jscript**.

True e-Commerce on the Distributed Web.

Truly enabled complex and secure transactions on the web require even more powerful technologies, marking the move towards the **distributed object web**. This marks the development of technologies and standards to enable information to be passed between powerful objects that can be located across many organisational boundaries across the entire web. This enables, for example, the heterogeneous information systems of suppliers, buyers, banks, and shipping companies to interact in real-time and without human intervention across the Internet. The array of available technologies is truly mind-boggling in diversity and complexity – in fact, the many acronyms in use often are referred to as the alphabet soup.

§ A clash of standards: DCOM+, CORBA, EJB & J2EE

The concept of a distributed object platform where enterprise applications were anticipated to become a huge complex cloud of interacting business objects, was conceived and pioneered by the “open standards” **Object Management Group (OMG)** who formulated the basic architecture and many associated standards.

- Their first contribution was the specification of an overall architecture whereby objects would discover each other using a sort of middleman – the Object Request Broker (**ORB**). This architecture specification became known as the **CORBA** (Common Object Request Broker Architecture).
- This required the specification of a number of related standards. In order to allow objects to communicate with each other, they need to present a uniform interface to each other and also allow for a number a standard methods, e.g. to discover the type of services they are able to provide. This resulted in the specification of a Interface Definition Language (**IDL**) for objects implemented in various different programming languages.
- Finally, they introduced a version of their standard which allowed different ORBs to locate and communicate with each other to work across a TCP/IP platform: the **IOP** (Internet InterORB Protocol).

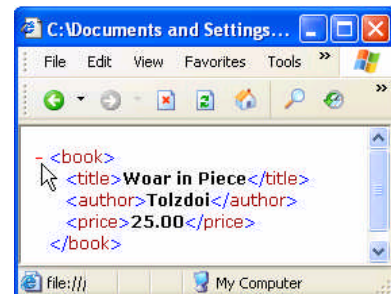
Microsoft had a set of proprietary products and specifications which competed with products adhering to the above standard, evolving and culminating in its **DCOM+** offering. This has now been superseded by its **.NET framework**.

In the Java camp – the “write-once, run anywhere” language of the web – a new look was taken at the original Java language. A standard was developed to define an interface specification for web-based objects programmed in Java – the **Enterprise Java Beans (EJBs)**. A comprehensive development environment and set of applications for building industry-strength EJBs was released as **J2EE** (Java 2 Enterprise Edition), which included a standard way for these EJBs to communicate with each other and object databases using a much simpler protocol than CORBA or DCOM.

§ XML and related technologies

A relatively recent standard that is becoming extremely important is the **eXtensible Markup Language (XML)** specification. It is a standard way of annotating data inside the document itself. At a superficial glance, it may look similar to HTML except that the tags used in XML must be defined by the user. For instance, the record of a book could be marked up as follows in XML.

```
<book>
  <title>Woar in Piece</title>
  <author>Tolzdoi</author>
  <price currency="USD">25.00</price>
</book>
```



Its flexibility has prompted its adoption in databases, for communication between applications across heterogeneous platforms and as a universal format for defining documents. Of course, if several different people or organisations wish to share XML information, they need to agree on a standard set of tags and their meanings (a so-called *namespace*). A number of these standards have already been proposed for various industries and communities.

Although XML is a very simple technology to understand in principle, a large number of associated standards have added significant complexity: there are standards for enforcing additional constraints on an XML document (its *schema*), for defining a conversion mechanism between different XML formats and for displaying or processing XML documents.

§ Web services

Web services expose the functionality of organisational information systems across the Internet. This allows the building of real-time, flexible, distributed inter-organisational systems using a variety of components. The underlying complexity of each component is fully hidden; all that is available is the external interface that takes the form of a programmable URL. For instance, you could interface your personal scheduling application with an airline reservation system that could interface your organisational purchasing system which would interface with the bank's funds transfer system, etc.

If you are offering or needing a particular web service, you would use a publicly accessible directory of web services of which **UDDI** (Universal Description, Discovery and Integration registry) is an example. These web services are described using a standard template or format, known as the **Web Services Description Language (WSDL)**. Web services communicate with each other using **SOAP (Simple Object Access Protocol)** that is basically a standardized type of envelope containing a message in XML format. The most developed and best-known technical framework for implementing web services is currently **Microsoft's .NET framework**.

Aspects on research: the semantic web

Most of the technologies discussed in this chapter focus on standards and protocols. These tend to concern themselves with structural issues i.e. how data should be formatted. However, the user ultimately seeks *information*. Therefore, he/she is not interested in the structure of the data but the actual meaning of this data. Information on the web can currently only be found by literal searches for keywords. This leads to lots of false hits, e.g. when a word has multiple meanings or the context cannot be specified easily, as well as lots of "missed" pages e.g. those using synonyms.

The inventor of the web, Berners-Lee, has now proposed to augment the information on the web by using an XML specification by the name of RDF (Resource Description Framework) to mark up the contents of the web page with semantic information. This would allow computers to infer the meaning of the text on a web page; hence the moniker "semantic web". The proposal is still in its early stages and lots of research is currently underway. Some of it involves defining taxonomies of knowledge ("ontologies") and others deal with the problem of how to embed artificial logic and reasoning in the associated technologies. Although the path to a truly semantically annotated web is likely to be a long one, it may well be worth the wait. Opponents lament the underlying complexity; still others think that advances in natural language processing may make the

semantic web obsolete. The debate is remarkably hot for what is, currently, mainly an academic issue.

The last word

The Internet is based on a few fairly old but well-chosen standards. In 1989, the basic standards for the World-Wide Web were established. The quick adoption of the Web for commercial purposes required the development of more powerful technologies to create dynamic and customized pages. Finally, fully-fledged E-commerce websites require distributed information systems and make use of a wide variety of advanced complex technologies. It is unlikely that the current technologies are the last word since there are still many issues to be resolved. This will mean that current standards will be updated and newer technologies will keep appearing, especially those supporting inter-organisational communication, mobile applications and ubiquitous computing.

Questions

- 1 Describe the different generations or evolutionary waves that E-commerce technologies went through. Explain the needs that drove this evolution and how each successive technology addressed problematic issues from a previous generation.
- 2 Describe the process of how a static HTML page gets displayed inside your browser window from the point when you type in its URL.
- 3 Explain the differences between client-side and server-side scripting. What are the benefits of each and for what type of purposes would you use them.
- 4 What is the difference between Java, JavaScript and J2EE?
- 5 What is the difference between HTML and XML?

Discussion questions

- 1 Make a list of all the acronyms used in the chapter, together with their brief descriptions. Use at least half of them to build a crossword puzzle.
- 2 Select a client-side scripting language of your choice and write a small script that prompts the user for a number and return the square of it. As an extra challenge, try to create a small animation onscreen.
- 3 Research a client-side scripting language that would read information from a cookie and update a visitor's count as well as the page last visited on a database. (Challenging)
- 4 A large life insurance company wishes to build an Internet-based information system that would allow individuals to bypass the middlemen (insurance brokers) to obtain online quotes based on individuals' particulars as well as to interrogate the company's system for existing client's policy details. Discuss the relative merits of using J2EE as opposed to .NET.
- 5 Find two examples of commercial databases that use XML as a native storage format and investigate their architecture (Oracle is one example). Also find two proposed XML specification standards for communicating business documents

between organisations (ebXML is one example) and give your opinion about their expected future.

References

- 3.1 Bates, C. (2002) **Web Programming: Building Internet Applications**. Wiley.
- 3.2 Benatallah, B.; Rabhi, F.A. & Mehandjiev, N. (2003) **E-Commerce Enabling Technologies**. Pearson.
- 3.3 Coombs, J.; Coomb, T.; Crowder, D. & Crowder R. (1998) **Setting up an Internet Sit for Dummies**. IDG Books.
- 3.4 Goldman, J.E.; Rawles, P.T. & Mariga, J.R. (1999) **Client/Server Information Systems: A Business-Oriented Approach**. Wiley.
- 3.5 Knobloch, M. & Kopp, M. (2003) **Web Design with XML**. Wiley.
- 3.6 McGrath, M. (2001) **CGI & Perl in easy steps: Server-side scripting simplified**. Computer Step.
- 3.7 McGrath, M. (2001) **JavaScript in easy steps: Create dynamic interactive web pages**. Computer Step.
- 3.8 Norris, M. & West, S. (2001) **eBusiness Essentials: Technology and Network Requirements for Mobile and Online Markets**. Wiley.
- 3.9 Schneider, G.P. (2002) **Electronic Commerce**. Thomson.
- 3.10 (Anon.) (2003) **Perl, CGI, and JavaScript Complete**. Sybex.

Web resources

- § A very interesting historical perspective on how and why the various web standards came into being written by the inventor of the WWW himself, Tim Berners-Lee, can be found on the website of the **W3C**: the organization in charge of setting web standards: <http://www.w3.org/DesignIssues/Overview.html>. Their website not only contains the full specification of all the Internet related standards, but also much background material and more tutorial-like documentation.
- § An excellent introduction to the overall workings of the Internet, including the client/server architecture, the URL, IP addresses etc. is one of the many enlightening articles on the HowStuffWorks website: <http://www.howstuffworks.com/web-server1.htm>.
- § It is hard to beat the many other introductory but well-written Internet tutorials on www.HowStuffWorks.com e.g. their tutorial on ASP (<http://computer.howstuffworks.com/asp.htm>), CGI scripting (<http://computer.howstuffworks.com/cgi.htm>) and another 30 or so e-Commerce related technologies.
- § Learning a scripting language is simple – go to the WebMonkey on <http://hotwired.lycos.com/webmonkey/> and locate their various tutorials for Java, JavaScript, PHP, Perl etc.
- § There are many beginner's introductions to HTML; the Bare Bones Guide to HTML
- § <http://werbach.com/barebones/barebone.html> is just one of many.

- § If you are interested in XML, check out the XML FAQ on <http://www.ucc.ie:8080/cocoon/xmlfaq> and the more condensed <http://www.w3.org/XML/1999/XML-in-10-points>.
- § Finally, if you want to find out about J2EE and .NET, first go to the sites of their respective owners: <http://java.sun.com> and www.microsoft.com not forgetting that it will prove difficult to find an unbiased opinion on the respective merits of their technologies.