Web Documents

Web Documents are a collection of linked individual pages, each with optional scrolling and non-scrolling areas, text, and perhaps images and other media embedded.
A **web document** is similar in concept to a web page.
Every Web document has its own URI (a **uniform resource identifier** (**URI**) is a string of characters used to identify a name or a resource). Note that a Web document is not the same as a file: a single Web document can be available in many different formats and languages, and a single file, for example a PHP script, may be responsible for generating a large number of Web documents with different URIs
Examples of Web Documents would be:

- Normal Windows Help files
- Many educational CD-ROM's
- Much of the World Wide Web

Three basic types of web documents

Static

A static web document resides in a file that it is associated with a web server. The author of a static document determines the contents at the time the document is written. Because the contents do not change, *each request for a static document results in exactly the same response.*

Dynamic

A dynamic web document does not exist in a predefined form. When a request arrives the web server runs an application program that creates the document. The server returns the output of the program as a response to the browser that requested the document. Because a fresh document is created for each request, *the contents of a dynamic document can vary from one request to another*.

Active

An active web document consists of a computer program that the server sends to the browser and that the browser must run locally. When it runs, the active document program can interact with the user and change the display continously.

Web administration

This involves maintaining one or many websites.

Web administration tasks

– Collaborate with development teams to discuss, analyze, or resolve usability issues.
– Identify or address interoperability requirements.
  Track, compile, and analyze web site usage data.
  Document application and web site changes or change procedures.
  Develop or document style guidelines for web site content.
  Test new software packages for use in web operations or other applications.
– Ensuring that the web servers, hardware and software are operating correctly,
– Designing the website,
– Generating and revising web pages,
– Replying to user comments
– Examining traffic through the site

Client-side and server-side scripting

There are two main ways to customise Web pages and make them more interactive. The two are often used together because they do very different things.

Scripts
A script is a set of instructions. For Web pages they are instructions either to the Web browser (client-side scripting) or to the server (server-side scripting). These are explained more below.
Scripts provide change to a Web page. Think of some Web pages you have visited. Any page which changes each time you visit it (or during a visit) probably uses scripting.
All log on systems, some menus, almost all photograph slideshows and many other pages use scripts. Google uses scripts to fill in your search term for you, to place advertisements, to find the thing you are searching for and so on. Amazon uses scripting to list products and record what you have bought.
Client-side scripting
Client-side scripting generally refers to the class of computer programs on the web that are executed by the user's web browser (*client-side)*, instead of on the web server (*server-side*).
Client-side scripts have greater access to the information and functions available on the user's browser.
The client is the system on which the Web browser is running. JavaScript is the main client-side scripting language for the Web. Client-side scripts are interpreted by the browser.
By viewing the file that contains the script, users may be able to see its source code. Many web authors learn how to write client-side scripts partly by examining the source code for other authors' scripts.
The process with client-side scripting is:

1. the user requests a Web page from the server
2. the server finds the page and sends it to the user
3. the page is displayed on the browser with any scripts running during or after display

So client-side scripting is used to make Web pages change after they arrive at the browser. It is useful for making pages a bit more interesting and user-friendly. It can also provide useful gadgets such as calculators, clocks etc. but on the whole is used for appearance and interaction.
Client-side scripts rely on the user's computer. If that computer is slow they may run slowly. They may not run at all if the browser does not understand the scripting language. As they have to run on the user's system the code which makes up the script is there in the HTML for the user to look at (and copy or change).

Server-side scripting
Server-side scripting is a web server technology in which a user's request is verified by running a script directly on the web server to generate dynamic web pages. It is usually used to provide interactive web sites that interface to databases or other data stores. This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.
The server is where the Web page and other content lives. The server sends pages to the user/client on request. The process is:

1. the user requests a Web page from the server
2. the script in the page is interpreted by the server creating or changing the page content to suit the user and the occasion and/or passing data around
3. the page in its final form is sent to the user and then cannot be changed using server-side scripting

The use of HTML forms or clever links allow data to be sent to the server and processed. The results may come back as a second Web page.
Server-side scripting tends to be used for allowing users to have individual accounts and providing data from databases. It allows a level of privacy, personalisation and provision of information that

is very powerful. E-commerce, MMORPGs and social networking sites all rely heavily on server-side scripting.
Server-side scripts, written in languages such as Perl, PHP, ASP.NET, Java, and server-side VBScript, are executed by the web server when the user requests a document. They produce output in a format understandable by web browsers (usually HTML), which is then sent to the user's computer. The user cannot see the script's source code (unless the author publishes the code separately), and may not even be aware that a script was executed.
The script is interpreted by the server meaning that it will always work the same way. Server-side scripts are never seen by the user (so they can't copy your code). They run on the server and generate results which are sent to the user. Running all these scripts puts a lot of load onto a server but none on the user's system.
The combination
A site such as Google, Amazon or Facebook will use both types of scripting:

- server-side handles logging in, personal information and preferences and provides the specific data which the user wants (and allows new data to be stored)
- client-side makes the page interactive, displaying or sorting data in different ways if the user asks for that by clicking on elements with event triggers

Client-side scripts have greater access to the information and functions available on the user's browser, whereas server-side scripts have greater access to the information and functions available on the server

INTERFACE STANDARDS: COMMON GATEWAY INTERFACE (CGI)
The common gateway interface (CGI) is a standard way for a Web server to pass a Web user's request to an application program and to receive data back to forward to the user. When the user requests a Web page (for example, by clicking on a highlighted word or entering a Web site address), the server sends back the requested page. However, when a user fills out a form on a Web page and sends it in, it usually needs to be processed by an application program. The Web server typically passes the form information to a small application program that processes the data and may send back a confirmation message. This method or convention for passing data back and forth between the server and the application is called the common gateway interface (CGI). It is part of the Web's Hypertext Transfer Protocol (HTTP
If you are creating a Web site and want a CGI application to get control, you specify the name of the application in the uniform resource locator (URL) that you code in an HTML file. This URL can be specified as part of the FORMS tags if you are creating a form. For example, you might code:
<FORM METHOD=POST ACTION=http://www.mybiz.com/cgi-bin/formprog.pl>
and the server at "mybiz.com" would pass control to the CGI application called "formprog.pl" to record the entered data and return a confirmation message. (The ".pl" indicates a program written in PERL but other languages could have been used.)
The common gateway interface provides a consistent way for data to be passed from the user's request to the application program and back to the user. This means that the person who writes the application program can makes sure it gets used no matter which operating system the server uses (PC, Macintosh, UNIX, OS/390, or others). It's simply a basic way for information to be passed from the Web server about your request to the application program and back again.
Because the interface is consistent, a programmer can write a CGI application in a number of different languages. The most popular languages for CGI applications are: C, C++, Java, and PERL.
An alternative to a CGI application is Microsoft's Active Server Page (ASP), in which a script embedded in a Web page is executed at the server before the page is sent.

WEB SECURITY

The primary security controls are:

• Confidentiality
• Access control (autentication and authorization)
• Integrity
• Availability
• Nonrepudiation.

•**Confidentiality** refers to the protection of information from unauthorized disclosure to a person or computing entity. Cryptography enables confidentiality and protects data.

•**Access control** (autentication and authorization) autenticates the identity of the entity trying to access a computing resource, and controls the use of computing resource per predetermined levels of entitlement. Authentication by static password, token, one time password, biometrics autentication, public and private key autentication.

•**Integrity** controls protect the data/or computing resource from any intentional or unintentional tampering. It is ensured by cryptography, hash function and proper access control. (antivirus software, signing the mobile code).

•**Availability** refers to the continuity of IT processing and the availability of information. Adequate configuration of the system and controlled processes and procedures guards against denial of services attack.

•**Nonrepudiation** controls ensure that users cannot deny actions they undertook. Other organizations require their users to read and agree to business agreements that hold users aaccountable to their authentication tokens. Transactions entered through those passwords are thus a user‘s sole responsibility.

WEB SECURITY CONSIDERATIONS

Web presents new challenges (not generally appreciated) in the context of computer and network security:

- the Internet is **two way**
- the web is increasingly serving as a **highly visible outlet**
- although web browsers are very easy to use, web servers are relatively easy to configure and manage, the web content is increasingly easy to develop, **the underlying software is extraordinarily complex**
- a web server **can be exploited as a launching pad for attack** on corporation’s internal network
- **causual and untrained (in security matters) users** are common clients for web based services.

<u>WEB SECURITY THREATS</u>

- **Integrity**:

Threats	Consequences	Countermeasures
• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic	• Loss of information • Compromise of machine • Vulnerability to all other threats	Cryptographic checksums

in transit		

Confidentiality:

Threats	Consequences	Countermeasures
• Eavesdropping on the net • Theft of information from server • Theft of data from client • Information about network configuration • Information about which clients talks to server	• Loss of information • Loss of privacy	Encryption, web proxies

Denial of services:

Threats	Consequences	Countermeasures
• Killing of user threads • Flooding machine with bogus threads • Filling up disk or memory • Isolating machine by DNS attack	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent

Authentication:

Threats	Consequences	Countermeasures
• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

<u>CHALLENGES OF WEB SECURITY</u>

The web presents new challenges not generally appreciated in the context pf computer and network security:

- The Internet is two ways. Unlike traditional publishing environments, even electronic publishing systems involving teletext, voice response, or fax-back, the Web is vulnerable to attacks on the Web servers over the Internet.
- The web is increasingly serving a highly visible outlet for corporate and product information and as the platform for business transactions. Reputations can be damaged and money can be lost if the Web servers are subverted.
- A web server can be exploited as a launching pad into the corporation's or agency's entire computer complex. Once the web server is subverted, an attacker may be able to gain access to data and systems not part of the Web itself but connected to the server at the local site.
- Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge.