Client/Server paradigm

As we know, the World Wide Web is accessed thru the use of a Web Browser, more technically known as a **Web Client**.\(^1\) A Web Client makes requests of a **Web Server**\(^2\), which is software running on a host machine, also called a Server\(^3\), which responds to the requests.

Typically, the *request* made by the client consists of an **Universal Resource Locator (URL)** and, possibly, some additional data. The server’s *responses* consist of web pages containing HTML code which the client interprets and displays to the user.

The format and the way in which requests and responses are managed are defined by the **Hypertext Transport Protocol (HTTP)**. The format of the web pages is defined by the **Hypertext Markup Language (HTML)** which may be augmented by Cascading Style Sheets (CSS) and a scripting language such as JavaScript. The combination of HTML, CSS, and scripting is referred to as **Dynamic HTML (DHTML)**.

Addressing/Identifying a Web page

When we want to request a web page we type a **Uniform Resource Locator (URL)** into the address field of our browser. In order for the network to be able to route the HTTP request to the appropriate web server it needs to know the **IP Address** of the server’s host machine and the **port address** of the server. All publicly accessible Web Servers use the same port address, 80. The IP address is derived from the domain name portion of the URL using the Domain Name System (DNS). Consider the following URL:

```
```

Each part of the URL has a meaning and a purpose, as follows. Note that many parts of the URL are often not used or are optional.

- **http://** This defines which **protocol** is to be used for formatting the request. Other possible protocols which can be used in an URL

\(^1\) Internet Explore, Netscape, and Firefox are examples of web clients.

\(^2\) Microsoft IIS and Apache are examples of web servers.

\(^3\) The server host machine may contain other server software in addition to the web server, such as an FTP server, and E-Mail server, and so on. The term web server, E-Mail server, etc is sometimes used to refer to both the host machine as well as the software providing the service.
include ftp:// for accessing ftp servers and nntp:// for accessing news servers.

Alice.teaparty.wonderland.com

This is the complete (“fully qualified”) domain name. The domain name is hierarchical, so that each part of the name can be considered a ‘level’ in a domain name ‘tree’. The ‘root of the tree is the rightmost portion of the domain name while the leftmost portion of the domain tree is one of the ‘leaves’ of the hierarchy. This particular domain name consists of four parts, as follows:

com This is the top-level domain. Typical top-level domains include .com, .edu, .org, .gov, .mil, and .net. In addition, each country in the world has their own top-level domain name such as .us, .fr, .uk, .jp, and so on.

wonderland This is the 2\textsuperscript{nd} level domain within the ‘com’ high level domain. Notice the hierarchical nature of the URL: From right to left you have the high level domain, and than a 2\textsuperscript{nd} level domain within that high level domain.

teaparty Continuing down the hierarchy, this is a subdomain within the wonderland.com domain. Subdomains are not required and are created at the discretion of the organization owning the 2\textsuperscript{nd} level domain name. Each subdomain may contain subdomains of its own.

Alice The first part of a domain name is usually the hostname of the host machine (server) on which the server software resides, at the bottom of the hierarchical domain name tree. In the early days of the World Wide Web this was almost always ‘www’, but that is not a requirement.

Since each host on the Internet must have a unique IP Address, this domain name must correspond to a unique IP Address. DNS is used to determine the IP address from this domain name.

\footnote{Strictly speaking, the actual root of the domain name hierarchy is represented by a period ‘.’ and in a fully-qualified domain name it should appear to the right of the top-level domain name. We normally leave the period off, but it is important in some cases to know when to include and when to leave out the period.}
:23054 This is the **Port Address** of the server application which is running within the host alice.teaparty.wonderland.com. Since there may be several server software processes running on the same host, each one must have a unique port address. Since most servers have *well-known port addresses* this part of the URL is rarely used; as mentioned earlier http requests assume that the server’s port address is 80\(^5\). The port addresses must be specified as part of the URL only if the server’s port address is different from the accepted well-known address for that kind of server.

dormouse This is a **folder** on the server host. Each web site has a *document root*; this is the folder on the server machine in which the html web pages reside. Suppose, for instance, that the document root has the following **path** on the server:

C:\web\site\documentroot

Then the domain name alice.teaparty.wonderland.com actually points to this folder. If you create folders below the root, than they must be explicitly specified in the URL. Thus this portion of our URL,

Alice.teaparty.wonderland.com:23054/dormouse

actually points to the folder

C:\web\site\documentroot\dormouse

on the server host.

bio.htm This is the requested **web page**. This is not required. If the page is not specified (for instance, if the the specified URL were: http://alice.teaparty.wonderland.com) than the server will deliver a **default** web page. This is a page found in the document root folder and is usually called **index.htm**. The name of the default page is specified when the web server for the site is configured. In Windows IIS, for instance, the default page might be ‘default.asp’. If no default is specified for the folder an error message ‘Page Not Found’ is returned to the client, or, perhaps, al list of all the pages

---

\(^5\) Some other well-known port addresses you should be familiar with are: 25 for mail servers and 20 & 21 for ftp servers.
Hypertext Transport Protocol (HTTP)

The HTTP specifies exactly how messages are formatted when communicating between a web client and web server. The specification makes this communication independent of the Hardware or Operating Systems on which the client and server are running.

Only two formats are specified, one for requests (client to server) and one for responses (server to client). Both kinds of messages contain three parts. For requests the three parts are the Request Line, the Headers, and the Body. For responses they are the Status Line, the Headers, and the Body.

<table>
<thead>
<tr>
<th>Request Message</th>
<th>Response Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Line</td>
<td>Status Line</td>
</tr>
<tr>
<td>Blank Line</td>
<td>Blank Line</td>
</tr>
<tr>
<td>Headers</td>
<td>Headers</td>
</tr>
<tr>
<td>Body</td>
<td>Body</td>
</tr>
</tbody>
</table>

HTTP Message Formats

Request Message:

The Request Line or an HTTP request message consists of three fields:

- Method: This field specifies the action to be taken by the server, and can also be used to specify how data is passed to the server. There are several values this field may contain, but the two most common are POST (usually indicating that there is data being sent to the server, such as data from an HTML form) and GET (which simply says that the server is to send a page). Other allowed methods are
OPTIONS, HEAD, PUT, DELETE, TRACE, and CONNECT.  

Page
The web page being requested (e.g. bio.htm in our URL example).

Version
The HTTP version being used. There are two HTTP versions in common use, HTTP 1.0 and HTTP 1.1. HTTP 1.0 is non-persistent, which means that a new TCP session (3-way handshake) must be initiated for every element of a page. Version 1.1 is persistent, so the session (connection) persists until the entire page is delivered.

Request Headers contain information for the server. For instance, the ‘Host Header Name’ contains the domain name of the server and ‘Cookie’ contains any cookies on the client machine associated with this web site. Headers are optional.

The body of a request is often empty. If the request was generated by clicking on the submit button in an HTML form page, then the body will contain the form data.

Response Message:

The Status Line of an HTTP response message also consists of three fields

<table>
<thead>
<tr>
<th>Version</th>
<th>The same as for requests.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status Number</td>
<td>A number indicating the result of trying to answer the request. Typical numeric responses are 200 and 404 (see below).</td>
</tr>
<tr>
<td>Status Text</td>
<td>A brief text describing the status. If the status number is 200 then the text is simply OK, if the status number is 404 the text will say ‘Page Not Found’.</td>
</tr>
</tbody>
</table>

Response Headers contain information for the client. For instance, the ‘Set Cookie’

---


header will cause a cookie to be stored on the client machine.

The body of a request contains the DHTML to be sent to the client.

How HTTP Requests are Handled

For a garden-variety page, a request to the server simply results in the server getting the page from the appropriate directory and delivering it to the client. (See the first diagram below). The server does not treat differently those pages containing client-side scripts and those that contain no scripting.

If a page is not found on the server, than one of two things can happen
• An error message (code 404 - Not Found) is returned, or
• A directory listing of the directory which was the target of the URL is returned. This occurs if ‘browsing’ is enabled in the server configuration for the web site.

Client-Side Scripting

Client-Side Scripting is accomplished by inserting lines of executable program code (a ‘script’) directly within an HTML web page. The programming language used to write the script may be JavaScript, VBscript or any other available scripting language which the web browser (web client) knows how to interpret. The script is interpreted and executed by the browser (web client) which displays the web page. The sequence of operations, then, is

1. The browser (Web Client) requests a web page from a web server.
2. The web server returns the page to the web client
3. The web client processes the html and any embedded scripts and displays the results on the users display.

Notice that

a. The extension of the pages containing scripts is htm or html; it does not
change to reflect the scripting language
b. The only resources available to the scripts are those on the client machine, none that are on the web server host machine.
c. The browser must be `enabled` for the scripting language

**Server-Side Scripting**

As you might guess, Server-Side Scripting consists of scripts that are executed on the web server host machine, not on the client machine. As with client-side scripts, the scripts themselves may be included directly among the text in an HTML document. When the page containing the script is requested, the web server passes the page containing the script to a script processor which performs the actions requested by the script and returns the page, which now contains only HTML, to the server who, in turn, forwards it to the requesting web client (browser). The full sequence of events is

1. The browser (Web Client) requests a web page from a web server.
2. The server forwards the page, if scripting is included, to the script processor.
3. The script processor runs the script and returns the results to the server.
4. The web server returns the page to the web client
5. The web client processes the html and any embedded client side scripts and displays the results on the users display.

Notice that

a. The extension of the pages containing scripts is NOT .htm or .html; The extension (.jsp, .vb, .php, etc) tells the server which script processor to

---

9 However, scripts may also be contained in separate files which are then 'included' in the HTML document. The effect is the same as if the scripts were written as part of the HTML document. The advantage is that these external files can be `reused` - the same script can then be included, if desired, in many HTML documents without having to type them over and over again. Also, if changes ever need to be made to the script, only one file needs to be modified, not every HTML page where the script is used.
use\(^\text{10}\). Also, the server can be configured to recognize that all pages in a particular directory contain scripts. An URL for such a configuration might look like \texttt{http://domain.com/cgi-bin/page.php}. In this example the fact that the page, \texttt{page.php}, is in the directory 'cgi-bin' indicates to the server that the page contains a script.

b. Many server features and parameters are available to server-side scripts, including
   \begin{itemize}
   \item Time and Date information
   \item The names and IP addresses of the server and client
   \item Server environment variables
   \end{itemize}

c. Server-side scripts can create, read and write files on the server
d. Server-side scripts can access data bases on the server

Accessing data bases is a very important feature of server-side scripts, for instance, for businesses on the web, where data bases are used to keep product information. Data Bases are accessed directly by the scripts, through a common data base access language called SQL (Structured Query Language). See the next diagram:

---

\(^{10}\) This is set up in the configuration of the web server. In fact, this configuration could be set up to send pages with .htm or .html extensions to a script processor, if that were desired.
Secure HTTP

Frequently we are asked to submit private/confidential information to a server via a web page form. For instance, a shopping site will want to know the Credit Card number you are using for your purchases. Clearly a need for a secure form of communication is required so this confidential data will not be compromised.

Such security is provided by another protocol: HTTPS (Secure HTTP). In actuality, HTTP is not modified, but it uses the services of another protocol, Secure Socket Layer (SSL), to provide security. Thus, HTTPS is referred to as ‘HTTP over SSL’

SSL has been superseded by Transport Layer Security (TLS) which provides essentially the same services and is still often referred to as SSL. SSL provides

• **Authentication** of the server (and sometimes the client as well) which guarantees to the client that the server to which the client is connected is, in fact, the correct server (and not a spoof), and
• **Encryption** of data for confidentiality.

Authentication is provided through the use of a Certificate Authority (CA) which provides a certificate with which the server establishes its bona fides, and a Key Distribution Infrastructure (KDI) which securely delivers Session Keys to both parties, the client and the server. Public Key Encryption is used to generate certificates and session keys. A session key is then used by the client and server for their exchange of data. When the session completes the session key is destroyed.

When secure web communication is desired, the URL protocol is https:// instead of http://. The port used is 443 instead of port 80