4.1. HTTP, Web Servers and Web Access
4.2. Universal naming with URLs
4.3. WWW Technology: HTML, DHTML, WML, XML
4.4. Tools: WYSIWYG Authoring Tools
4.5. Helper applications: CGI; PERL, JAVA, JAVA SRIPTS, PHP, ASP, .NET Applications
4.6. Introduction to AJAX (Programming)
4.7. Browser as a rendering engine: text, HTML, gif and jpeg Introduction

Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.

Based on the web service architecture, we create the following two components as a part of web services implementation −

i. **Service Provider or Publisher:** This is the provider of the web service. The service provider implements the service and makes it available on the Internet or intranet.

We will write and publish a simple web service using .NET SDK.

ii. **Service Requestor or Consumer:** This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

We will also write two web service requestors − one **web-based consumer** (ASP.NET application) and another **Windows application**-based consumer.

How Does a Web Service Work?

A web service enables communication among various applications by using mostly open standards such as HTML, XML, and SOAP. A web service takes the help of −

- XML to tag the data
- SOAP(Simple Object Access Protocol) to transfer a message

4.1. HTTP, Web Servers and Web Access

***HTTP :** The Hypertext Transfer Protocol (HTTP) is an *application-level protocol for distributed, collaborative, hypermedia information systems*. HTTP has been in use by the World-Wide Web global information initiative since 1990.

- HTTP/0.9, was a simple protocol for *raw data transfer* across the Internet.

- HTTP/1.0, as defined by *RFC 1945* , *improved the protocol by allowing messages* to be in the format of MIME-like messages, containing meta information about the data transferred and modifiers on the request/response semantics.

- "HTTP/1.1". This protocol includes more stringent requirements than HTTP/1.0 in order to ensure reliable implementation of its features.

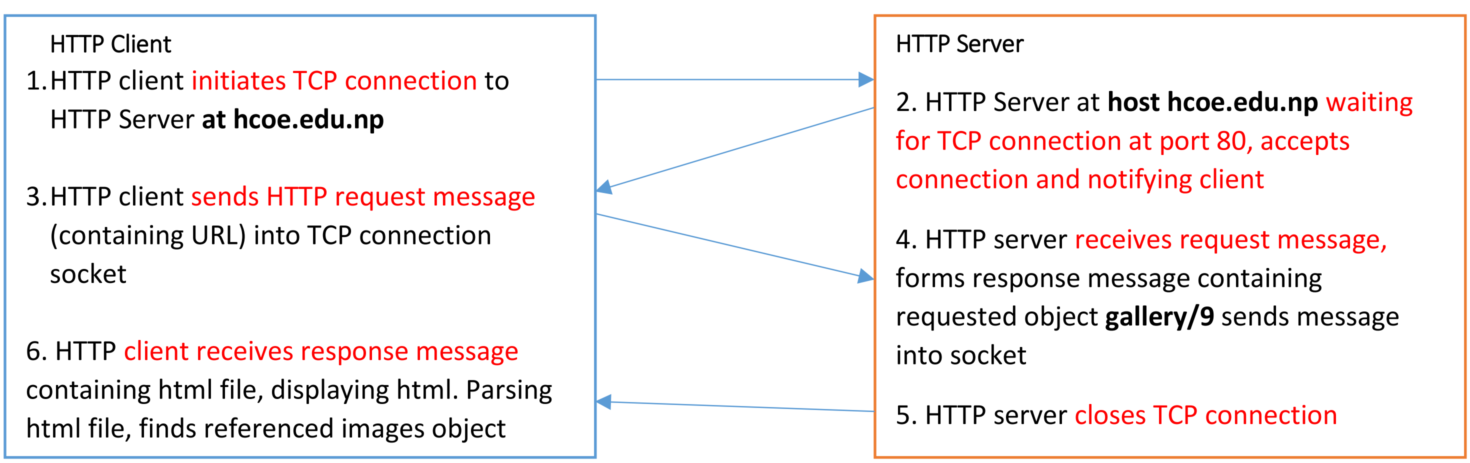
. Messages are passed in a format similar to that used by Internet mail as defined by the Multipurpose Internet Mail Extensions (MIME).

How HTTP Works ?

- Client initiates TCP connection (creates socket) to server , typically port 80
- Server accepts TCP connection from client
- HTTP messages (Application layer protocol message) exchanged between browser(HTTP client) and Web Server(HTTP Server)
- TCP Connection Closed

Example

Suppose user enters http://hcoe.edu.np/gallery/9 contains some text and images



Difference between HTTP and HTTPS

1) In case of HTTP URL begins with **"HTTP://"** and for HTTPS connection it is **"HTTPS://"**
2) HTTP is unsecured on other hand HTTPS is secured.
3) HTTP uses port 80 for communication unlike HTTPS which uses port 443
4) No certificates required for validation in case of HTTP. HTTPS requires SSL Digital Certificate
5) No encryption in HTTP; Data encrypted before sending and receiving in HTTPS.

*** Web Server or** Internet server**:** A Web server is a system that *delivers content or services to end users* over the Internet. A Web server consists of a *physical server, server operating system (OS) and software* used to facilitate HTTP communication.

- Web server *runs a website by returning HTML files* over an HTTP connection.

- Web server is any Internet server that *responds to HTTP requests to deliver content and services*. Depending on context, the term can refer to the hardware or Web server software on the server. In terms of software, there have been literally hundreds of Web servers over the years, but Apache and Microsoft's IIS have emerged as two of the most popular systems.

- **General Server Characteristics** - Web servers have two separate directories

  - The *document root* is the root directory of all servable documents (well, not really all)
  - The *server root* is the root directory for all of the code that implements the server

How Web Servers Work ?

- ✓ The browser breaks the URL into three parts: The protocol ("http") The server name ("www.website.com") The file name ("webpage.html")
- ✓ The browser communicates with a **name server(DNS),** which translates the server name, into an IP address
- ✓ The browser then forms a connection to the **Web server** at that IP address on port 80.
- ✓ Following the HTTP protocol, the browser sends a **GET** request to the server, asking for the file
- ✓ The server sends the **HTML text** for the Web page to the browser.
- ✓ The browser reads the HTML tags and formats the page onto the screen.

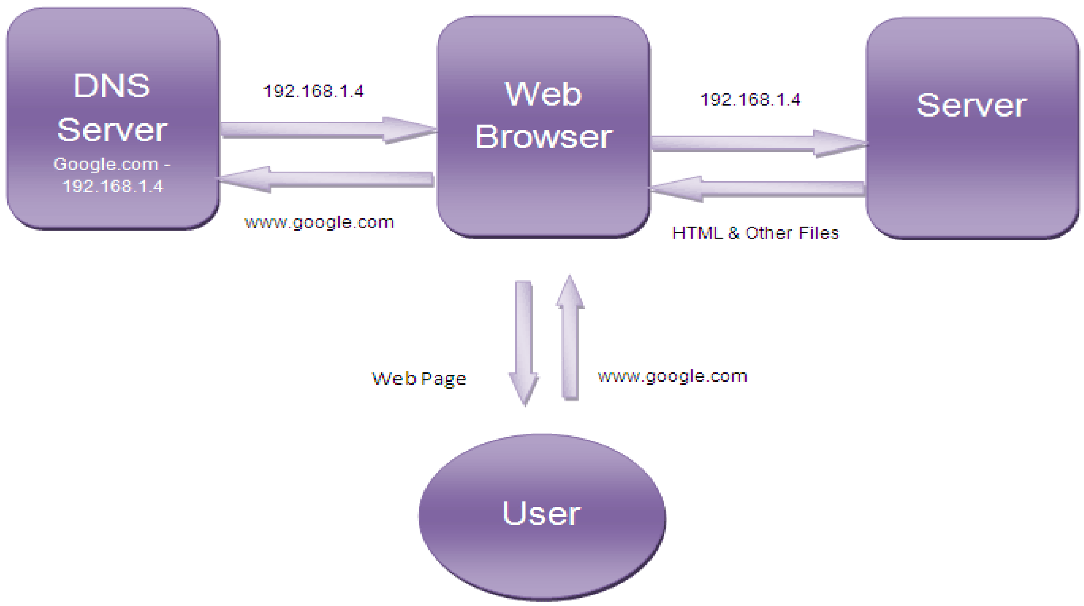


Fig. Web server operation

Popular Web Servers and Common Security Threats

- ✓ Apache Web Server
- ✓ IIS Web Server
- ✓ Sun ONE Web Server

Nature of Security Threats in a Web Server Environment.

- ✓ Bugs or Web Server Misconfiguration: due to lack of knowledge, Human error
- ✓ Browser-Side or Client Side Risks: Cross site scripting
- ✓ Sniffing Denial of Service Attack: Service or Message Overloading

***Web access that includes access to WWW.**

Web access management (WAM) is a process for identity authentication for Web access. It is a form of access and identity management which controls access to Web resources like Web servers and secure servers by providing authentication management through policy-based authorizations as well as audit and report services. It is often used in Web-based applications to regulate external user access through the use of username and password key pairs.

Web Access Management Principles

- A single means of access from intranet to internet
- Your applications become available over the web
- Protect your web applications

Web Access Management Architecture

- **Plugins (Web Agent)** are programs that are installed on every web/application server, register with those servers, and are called at every request for a web page. They intercept the request and communicate with an external policy server to make policy decisions. One of the benefits of a plugin (or agent) based architecture is that they can be highly customized for unique needs of a particular web server. One of the drawbacks *is that a different plugin is required for every web server on every platform (and potentially for every version of every server). Further, as technology evolves, upgrades to agents must be distributed and compatible with evolving host software.*

- **Proxy**-based architectures differ in that all web requests are routed through the proxy server to the back-end web/application servers. This can provide a more universal integration with web servers since the common standard protocol, HTTP, is used instead of vendor-specific application programming interfaces (APIs). One of the drawbacks is that additional hardware is usually required to run the proxy servers.
- **Tokenization** differs in that a user receives a token which can be used to directly access the back-end web/application servers. In this architecture, the authentication occurs through the web access management tool but all data flows around it. This removes the network bottlenecks caused by proxy-based architectures. One of the drawbacks *is that the back-end web/application server must be able to accept the token or otherwise the web access management tool must be designed to use common standard protocols.*

4.2. Universal naming with URLs

URL (Uniform Resource Locator, previously Universal Resource Locator) is the *unique address* for a file that is accessible on the Internet. URL is *the global address of documents and other resources on the World Wide Web*. A *common way to get to a Web site* is to enter the URL of its home page file in your Web browser's address line. However, any file within that Web site can also be specified with a URL. Such a file might be any Web (HTML) page other than the home page, an image file, or a program such as a common gateway interface application or Java applet.

A URL components:

- **Protocol identifier**: For the URL http://example.com, the protocol identifier is http, used to access the file resource
- **Domain Name or Resource name:** For the URL http://example.com, the resource name is example.com, that identifies address of resource on the Internet
- **Path Name**, a hierarchical description that specifies the location of a file in that computer.

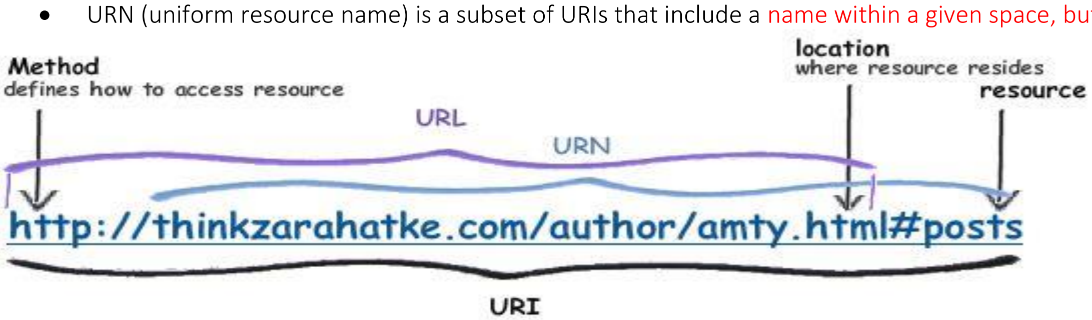
Note that the protocol identifier and the resource name are separated by a colon and two forward slashes.

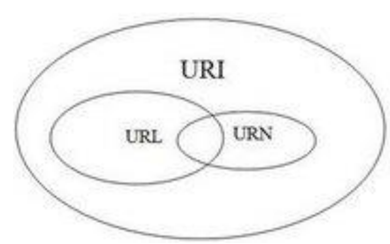
- The protocol identifier indicates the name of the protocol to be used to fetch the resource. E.g. HTTP, FTP etc
- The resource name is the complete address to the resource. The format of the resource name depends entirely on the protocol used, but for many protocols, including HTTP,
- The resource name contains one or more of the following components:

- **Host Name :** The name of the machine on which the resource lives.
- **Filename :** The pathname to the file on the machine.
- **Port Number :** The port number to which to connect (typically optional).
- **Reference :** A reference to a named anchor within a resource that usually identifies a specific location within a file (optional).

e.g. On the Web (which uses the Hypertext Transfer Protocol, or HTTP), an example of a URL is:
http://www.ietf.org/rfc/rfc2396.txt => *which specifies the(i)protocol - use of a HTTP (Web browser) application, (ii) domain name - a unique computer named www.ietf.org, and (iii) pathname - the location of a text file or page to be accessed on that computer whose pathname is /rfc/rfc2396.txt.*

- URI (uniform resource identifier) identifies a resource (text document, image file, etc)
- URL (uniform resource locator) is a subset of the URIs that include a network location
- URN (uniform resource name) is a subset of URIs that include a name within a given space, but no location





4.3. WWW Technology: HTML, DHTML, WML, XML

The **World Wide Web** (abbreviated **WWW** or **the Web**) is an information space where documents and other web resources are identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and can be accessed via the Internet.

"The World Wide Web is the universe of network-accessible information, an **embodiment** of human knowledge." – W3C

Function of WWW

I. Linking : Most web pages contain hyperlinks to other related pages and perhaps to downloadable files, source documents, definitions and other web resources. In the underlying HTML, e.g. Example.org Homepage
II. Dynamic update of web pages : To make web pages more interactive, some web applications also use Stylesheet, JavaScript techniques, which is dynamically updates to viewer.
III. www prefix : When a user submits an incomplete domain name to a web browser in its address bar input field, some web browsers automatically try adding the prefix "www" to the beginning of it and possibly ".com", ".org" and ".net" at the end,
IV. Protocol identifier : The protocol identifiers *http://* and *https://* at the start of a web URI refer to Hypertext Transfer Protocol or HTTP Secure, respectively. They specify the communication protocol to use for the request and response. The HTTP protocol is fundamental to the operation of the World Wide Web, and the added encryption layer in HTTPS is essential when browsers send or retrieve confidential data, such as passwords or banking information. Web browsers usually automatically prepend http:// to user-entered URIs, if omitted.

***HTML** : **Hypertext Markup Language**, commonly referred to as **HTML**, is the *standard markup language used to create web pages*. Along with CSS, and JavaScript, HTML is a *foundation of technology,* used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications. *Web browsers* can read HTML files and render them into visible or audible web pages. HTML describes *the structure of a website semantically* along with signs for presentation, *making it a markup language, rather than a programming language.*

File extention : .html,.htm ; Internet media type : text/html ; Type code : TEXT ; Developed by : W3C ; Initial Release 1993 ; type of format : Document format; Extebted from : SGML ; Extended to : XHTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

DHTML** : **Dynamic HTML**, or **DHTML**, is *an umbrella term for a collection of technologies used together to create interactive and animated web sites by using a* ***combination *of a static markup language (such as HTML), a client-side scripting language (such as JavaScript), a presentation definition language (such as CSS), and the Document Object Model.* The application of DHTML was introduced by Microsoft with the release of Internet Explorer 4 in 1997.

DHTML allows authors to *add effects* to their pages by using scripting language is changing the DOM and page style.

DHTML is the combination of HTML, CSS and JavaScript.

- *Animate text and images* in their document, independently moving each element from any starting point to any ending point, following a predetermined path or one chosen by the user.
- *Embed a ticker that automatically refreshes* its content with the latest news, stock quotes, or other data.
- Use a form to capture *user input, and then process, verify and respond to that data without having to send* data back to the server.
- Include *rollover buttons or drop-down menus.*

How DHTML Works : Rollover style changes using DHTML

DHTML is a combination of HTML, Cascading Style Sheets, JavaScript, and the Document Object Model. The web page shown here uses simple DHTML to change the style of links to be red and underlined when the mouse is rolled over them. You can use this basic format to tie CSS styles to common events like onMouseOver or OnClick, so you can change the styles of most elements on the fly.

Example: Rollover style changes using DHTML

```
<html>    (A)
<head>
<title>Rollover Style Changes</title>
<style>    (B)
<!--
a { text-decoration: none; }
-->
</style>

<script>    (C)
```

```
<!--
function turnOn(currentLink) {
    currentLink.style.color = "#990000";    (D)
    currentLink.style.textDecoration = "underline";
}

function turnOff(currentLink) {
    currentLink.style.color = "#0000FF";
    currentLink.style.textDecoration = "none";
}
//-->
</script>
</head>

<body bgcolor="#FFFFFF">
<a href="#home"    (E)
   onMouseOver="turnOn(this);" onMouseOut="turnOff(this);">Home</a>
<a href="#contact"
   onMouseOver="turnOn(this);" onMouseOut="turnOff(this);">Contact</a>
<a href="#links"
   onMouseOver="turnOn(this);" onMouseOut="turnOff(this);">Links</a>
</body>
</html>
```

(A) This page is an HTML file, so it starts with normal <html> and <head> HTML tags.
(B) In the <head>, we have a CSS style sheet, defined using the <style> tag, that removes any text decorations from all the links in the document. In this case, the point is to remove the default underlines from links.
(C) Inside the <script> tag, there are two JavaScript functions, turnOn( ) and turnOff( ), that change the style of a link when the user moves the mouse over and back out of the link. When the mouse enter a link, the text is underlined and turned red. When the mouse exits, these effects are removed.
(D) The script uses the DOM to reference the link's style attribute and change the color and textDecoration properties, which are the DOM equivalents of the CSS properties color and text-decoration.
(E) In this <a> tag, the onMouseOver and onMouseOut event handlers are used to set up the calls to turnOn( ) and turnOff( ).

***WML : Wireless Markup Language** (WML), *based on XML, is a markup language proposed for devices that implement the Wireless Application Protocol (WAP) specification, such as mobile phones*. It provides navigational support, data input, hyperlinks, text and image presentation, and forms, much like HTML (Hypertext Markup Language). It preceded the use of other markup languages now used with WAP, such as HTML itself, and XHTML (which are gaining in popularity as processing power in mobile devices increases).

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
  <card id="main" title="First Card">
    <p mode="wrap">This is a sample WML page.</p>
  </card>
</wml>
```

The following are some key features of WML as compared to HTML:

- WML is a markup language *for small, wireless computing devices.*
- In WML, *variables can be defined that store data in string format*. In HTML, variables cannot be stored.
- WML uses *WML script for client-side scripting*, which is stored in a separate file. HTML uses JavaScript.
- The supported *image format for WML is WBMP*. HTML supports JPEG, GIF and BMP.
- A micro-browser is used to run WML markup. A regular browser, such as Internet Explorer, Firefox or Chrome, is used to run HTML markup.
- WML follows XHTML specification and is therefore case sensitive. HTML is not case sensitive.
- WML has fewer tags compared to HTML.
- A deck is a set of WML cards. In HTML, a site is a set of HTML pages.

WML-equipped devices have the following characteristics:

- *Display Size:* Devices *have a small screen size* and low resolution; therefore, WML has to be capable of rendering content regardless of display size.

- *Input:* Small computing devices *do not have a mouse or pointer-based navigation devices*. They may have a small numeric keypad or a QWERTY keypad based on whether the device is simple or sophisticated. WML has to be capable of obtaining necessary user input regardless of the limitations of the device.
- *Processing:* They have *limited-capacity rechargeable batteries with a low-power CPU and low memory.* WML browsers should act like thin clients and perform minimal processing on the device.
- *Network Capabilities:* Small computing devices have a *low bandwidth and high network latency*. WML has to ensure maximum efficiency in fetching requested Web pages from the server.

***XML : Extensible Markup Language (XML)**

Extensible : XML is extensible. It lets you define your own tags, the order in which they occur, and how they should be processed or displayed. Another way to think about extensibility is to consider that XML allows all of us to extend our concept of what a document is: it can be a file that lives on a file server, or it can be a transient piece of data that flows between two computer systems (as in the case of Web Services).

Markup : The most recognizable feature of XML is its tags, or elements (to be more accurate). However, XML allows you to define your own set of tags.

Language : XML is a language that's very similar to HTML. It's much more flexible than HTML because it allows you to create your own custom tags. However, it's important to realize that XML is not just a language. XML is a meta-language: a language that allows us to create or define other languages. For example, with XML we can create other languages, such as RSS (**Rich Site Summary**; or **Really Simple Syndication**) is a type of web feed which allows users to access updates to online content in a standardized, computer-readable format). Write `<rss version="2.0"> after <?xml version="1.0"?>`

XML is a markup language, user defined language that *defines a set of rules for encoding documents in a format which is both human-readable and machine-readable.* It is defined by the W3C's XML 1.0 Specification and by several other related specifications, all of which are free open standards.

The design goals of XML emphasize *simplicity, generality and usability across the Internet*. It is a textual data format with strong *support via Unicode* for different human languages. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures such as those used in web services.

The basic building block of an XML document is an element, defined by tags. An element has a beginning and an ending tag. All elements in an XML document are contained in an outermost element known as the root element. XML can also support nested elements, or elements within elements. This ability allows XML to support hierarchical structures. Element names describe the content of the element, and the structure describes the relationship between the elements.

XML Does Not Use Predefined Tags

The XML language has no predefined tags.

The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

HTML works with predefined tags like <p>, <h1>, <table>, etc.

With XML, the author must define both the tags and the document structure.

XML is Extensible

Most XML applications will work as expected even if new data is added (or removed).

Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <data>).

Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.

The way XML is constructed, older version of the application can still work:

```
<note>
 <date>2015-09-01</date>
 <hour>08:30</hour>
 <to>Tove</to>
 <from>Jani</from>
 <body>Don't forget me this weekend!</body>
</note>
```

Old Version	New Version
Note To: Tove From: Jani **Head: (none)** Don't forget me this weekend!	Note To: Tove From: Jani Date: 2015-09-01 08:30 Don't forget me this weekend!

XML Simplifies Things

- It simplifies data sharing
- It simplifies data transport
- It simplifies platform changes

- It simplifies data availability

Many computer systems contain data in incompatible formats. Exchanging data between incompatible systems (or upgraded systems) is a time-consuming task for web developers. Large amounts of data must be converted, and incompatible data is often lost.
XML stores data in plain text format. This provides a software- and hardware-independent way of storing, transporting, and sharing data.
XML also makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.
With XML, data can be available to all kinds of "reading machines" like people, computers, voice machines, news feeds, etc.

HTML Vs XML

- XML stands for Extensible Markup Language, HTML - short for Hypertext Markup Language
- In XML, *Data is stored* in separate XML files but in HTML, it is stored inside the files.
- HTML is *predefined Language. XML is user defined language*.
- To display HTML, *we used CSS. In XML, we used XSL.*
- XML was designed to describe data and to focus on what data is. HTML was designed to display data and to focus on how data looks.
- *HTML is about displaying information, XML is about describing information*
- HTML *is static and XML is dynamic.*
- HTML is presentation language where as XML is not either a programing language or a presentation language. It is used to transfer data between applications and databases.
- HTML is not case-sensitive where as XML is case-sensitive.
- In XML we can define our own tags as it is not possible in HTML.
- In XML it is mandatory to close each and every tag where as in HTML it is not required.
- XML describes the data where as HTML only defines the data.

HTML Vs DHTML

- HTML is a mark-up language, while DHTML is a collection of technology.
- *DHTML creates dynamic web pages, whereas HTML creates static web pages.*
- DHTML allows including small animations and dynamic menus in Web pages.
- DHML used events, methods, properties to insulate dynamism in HTML Pages.
- DHML is basically using JavaScript and style sheets in an HTML page.
- HTML sites will be slow upon client-side technologies, while DHTML sites will be fast enough upon client-side technologies.
- HTML creates a plain page without any styles and Scripts called as HTML. Whereas, DHTML creates a page with HTML, CSS, DOM and Scripts called as DHTML.
- HTML cannot have any server side code but DHTML may contain server side code.
- In HTML, there is no need for database connectivity, but DHTML may require connecting to a database as it interacts with user.
- HTML files are stored with .htm or .html extension, while DHTML files are stored with .dhtm extension.
- HTML does not require any processing from browser, while DHTML requires processing from browser which changes its look and feel.

WML Vs XML

WAP/WML	HTML
Markup language for wireless communication	Markup language for wired communication
Makes use of variables	Does not use variables
WML script stored in a separate file	Javascript is embeded in the same HTML file
Images stored as WBMP	Images are stored as GIF, JPEG or PNG
WBMP is a 2 bit image	Size of the images are much larger in HTML
Case sensitive	Not case sensitive
WML has fewer tags than HTML	HTML has more tags than WML
A set of 'WML Cards' make a 'DECK'	A set of 'HTML pages' make a 'SITE'

4.4. Tools: WYSIWYG Authoring Tools :*Drag and Drop features*

A computer screen display which appears on screen as it will be seen when printed on paper.

The WYSIWYG "***What You See Is What You Get***" is a system, in such editors we edit not directly the source code of your documents, but its **presentation** as it (hopefully) will appear in the final document. So instead of writing blocks of code manually (as you e.g. would do it in Word or Latex), you manipulate with design components using an editor window. This means *that you view something very similar to the end result while the document or image is being created.*

It is easier than ever to create a Web site with an HTML editor, as software developers continue to add tools that let you develop advanced features with style. Today's Web authoring tools can provide the power to build an interactive, animated, state-of-the-art Web site suitable for anything from a personal Web page to a midsize business site. New Web designers don't need to know HTML to create discussion groups, *pop-up windows, navigation bars, animated page transitions, Dynamic HTML, or a dozen other advanced features in order to integrate them into a site with an elegant and consistent design.*

Web authoring tools : Web authoring tools are used to create Web content, and cover a wide range of software programs you can download to your computer or access online. The World Wide Web Consortium, or W3, issues guidelines for web authoring tools that create a basic industry standard for web accessibility. The guidelines encourage web-authoring tool manufacturers to include specific features in their products that will aid Internet users with disabilities. All of the major web-authoring tool manufacturers follow the W3 guidelines.

- Word Processors : Word processors like Microsoft Word, WordPerfect or OpenOffice Writer are some of the most popular web authoring tools available. Users can create a Web page just as they would a printable document and then save it in HTML format, creating a quick and easy web page. Because users are usually familiar with the word processor on their computers, creating HTML pages with the same program represents a low learning curve. These usually present content in a what you see is what you get format, or WYSIWYG, meaning how the page appears on the screen is how it will appear when it's online.
- Desktop Publishing Programs : Desktop publishing programs, like Adobe InDesign and Scribus are designed for producing material like newspapers, magazines, books and Web pages. Like word processors, desktop publishing programs provide a WYSIWYG interface. Their advanced Web authoring options, such as page layout and style elements, give users more control over the page's appearance. These programs also support multimedia objects, like images, graphics or audio files. Completed pages can be converted to both HTML and CSS files.
- Online Web Page Builders : Website hosting sites usually offer their customers many web-authoring tool options to create and maintain their web pages online. Tools can include Web page builders, shopping systems, audio/visual editors and domain options. The builders incorporate many web authoring tools, including word processing, graphic editing, templates and layout schemes. Webpage builders have two main editing options: HTML or a non-HTML interface. Users who have limited HTML knowledge can use the non-HTML interface to drop and drag items to create layouts and use the text option to type in content.
- HTML Editors : HTML editing programs like Adobe Dreamweaver are some of the most powerful web authoring tools available. They are generally used by professional Web designers to create commercial websites. Most HTML editors are similar to web-page builders in offering users HTML or non-HTML interfaces. The non-HTML interface allows the user to see how the web page will look when it is uploaded to the Internet. HTML editors are used to type raw code, much as one would in a plain text document like a word processor, including HTML, CSS, JavaScript or XML. Most of the work is performed using a built-in text editor. HTML editors feature HTML validation checkers that will run through a web page and check for markup errors and accessibility validation issues.
- Plain Text Editors : Basic text editors like Notepad are also a useful Web authoring tool for those familiar with the code. Unlike word processors or desktop publishing programs, plain text editors do not apply additional code to what appears in the document. Plain text editors are also useful for quickly making edits to completed pages that require updates.

Styles and formats in the WYSIWYG

Formats allow you to cleanly format text via the WYSIWYG. The formats available are Paragraph, Address, Preformatted and Headings 1-6.

- Paragraph: Normally, text for general content uses the paragraph format by default.
- Address: Used to wrap content that provides contact information for a document or a major part of a document.
- Preformatted: Text is displayed in a fixed-width font preserves both spaces and line breaks. Often used to display computer code.
- Headings: Provide semantic and structure information about the hierarchy of the page content much like outline headings. H1 is the most important heading and H6 is least important.

Typically, the design **goals of a WYSIWYG** application may include the following:

- Provide high-quality printed output on a particular printer
- Provide high-quality printed output on a variety of printers
- Provide high-quality on-screen output
- Allow the user to visualize what the document will look like when printed
- Allow the user to visualize what the website will look like when published

4.5. Helper applications: CGI; PERL, JAVA, JAVA SRIPTS, PHP, ASP, .NET Applications

Client-side Environment

The client-side environment used to run scripts is usually a browser. The processing takes place on the end users computer. The source code is transferred from the web server to the user's computer over the internet and run directly in the browser.
The scripting language needs to be **enabled** on the client computer. Sometimes if a user is conscious of **security risks** they may switch the scripting facility off. When this is the case a message usually pops up to alert the user when script is attempting to run.

Server-side Environment

The **server-side environment** that runs a scripting language is a web server. A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser. It is usually used to provide interactive web sites that interface to databases or other data stores on the server.
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.

* CGI (Common Gateway Interface)

- CGI, is *a set of standards* that define *how information is exchanged between the web server and a custom script.*
- The Common Gateway Interface (CGI) is a method used for server programming in such a way that, web applications can be equipped with scripting languages like python as their back end, for processing the client requests. CGI are external gateway programs to interface with information servers such as HTTP servers.
- CGI defines a way for a web server to interact with external 'content generating' programs, which are often referred to as CGI programs or CGI scripts. It is the simplest, and most common, way to put dynamic content on your web site.

To understand the concept of CGI, let's see what happens when we click a hyper link to browse a particular web page or URL.

- Your browser contacts the HTTP web server and demand for the URL ie. filename.
- Web Server will parse the URL and will look for the filename in if it finds that file then sends back to the browser otherwise sends an error message indicating that you have requested a wrong file.
- Web browser takes response from web server and displays either the received file or error message.

These CGI programs can be a PERL Script, Shell Script, C or C++ program etc.

CGI Architecture Diagram

You might be quite familiar with the html-php web pages, where the php processor manages the dynamic content generation, while html & css gives the presentation of this content as a webpage. In CGI programming we can use scripting languages such as python, instead of php, for processing data coming from the html pages.

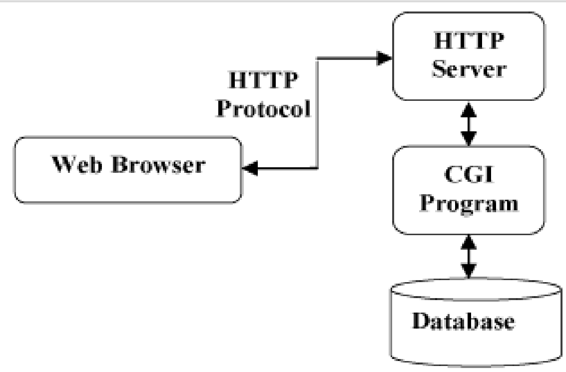


Fig. CGI Architecture

First CGI Program

Here is a simple link which is linked to a CGI script called hello.cgi. This file is being kept in /cgi-bin/ directory and it has following content. Before running your CGI program make sure you have chage mode of file using **chmod 755 hello.cgi** UNIX command.

```
#!/usr/bin/perl
print "Content-type:text/html\r\n\r\n";
print '<html>';
print '<head>';
print '<title>Hello Word - First CGI Program</title>';
print '</head>';
print '<body>';
print '<h2>Hello Word! This is my first CGI program</h2>';
print '</body>';
print '</html>';
```

If you click hello.cgi then this produces following output:
Hello Word! This is my first CGI program

*PERL – www.perl.org

PERL is a *high-level, general-purpose, interpreted, dynamic programming languages.* The languages in this family include *Perl 5 and Perl 6*.
The Perl languages *borrow features* from other programming languages *including C, shell script (sh), AWK, and sed*
In addition to CGI, Perl 5 is used for *system administration, network programming, finance, bioinformatics*, and other applications, such as for GUIs.

How Perl Works

When the Perl compiler is fed a Perl program, the first task it performs is *lexical analysis*: breaking down the program into its basic syntactic elements (often called *tokens*). If the program is:

print "Hello, world!\n";

the lexical analyzer breaks it down into three tokens: print, "Hello, world!\n", and the final semicolon. The token sequence is then *parsed*, fixing the relationship between the tokens. In Perl, the boundary between lexical analysis and parsing is blurred more than in other languages. (Other computer languages, that is. If you think about all the different meanings new Critter might have depending on whether there's a Critter package or a subroutine named new, you'll understand why. On the other hand, we disambiguate these kinds of things all the time in English.)

Once a program has been parsed and (presumably) understood, it is compiled into a tree of *opcodes* representing low-level operations, and finally that tree of operations is executed--unless you invoked Perl with the -c ("check syntax") switch, which exits upon completing the compilation phase. It is during compilation, not execution, that BEGIN blocks, CHECK blocks, and use statements are executed.

***JAVA –** www.java.com, www.oracle.com/java/

-Java is a general-purpose computer programming language that is *concurrent, class-based, object-oriented and also platform independent.* It is intended to let application developers **"Write Once, Run Anywhere" (WORA),** meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

JIT is one of the java compilers (Just-In-Time compiler).

- Java applications are typically *compiled to bytecode that can run on any Java virtual machine (JVM)* regardless of computer architecture. As of 2016, Java is one of the *most popular programming languages in use*, particularly for client-server web applications, with a reported 9 million developers.

- Java was originally developed by *James Gosling at Sun Microsystems* (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. *The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.*

- *One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support.* This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to architecture-specific machine code. *Java bytecode instructions are similar to machine code*, but they are intended to be *executed by a virtual machine (VM)* written specifically for the host hardware. *End users commonly use a Java Runtime Environment (JRE)* installed on their own machine for standalone Java applications, or in a web browser for Java applets.

How Java works

- First, we should have a java source code which must be saved with Program.java extension.
- Then we use a JAVA Compiler to compile the source code to get java bytecode which must have a Program.class extension. We can say that java bytecode is a modified version od java source code.
- Now we pass the java bytecode through an interpreter called JVM(JAVA Virtual Machine) which will read every single statement at a time from java bytecode and convert it to machine level code and then will execute the code. We get the output only after JVM converts and execute the code.

Note: JAVA has platform specified JVM interpreter such specified JVM for Linux, Windows, Macintosh which allow us to execute java programs at various platform easily.

***JavaScript :** www.javascript.com/

In contrast to server-side code, client-side scripts are embedded on the client's web page and processed on the client's internet browser. Client-side scripts are written in some type of scripting language like JavaScript and interact directly with the page's HTML elements like text boxes, buttons, list-boxes and tables. HTML and CSS (cascading style sheets) are also used in the client. In order for client-side code to work, the client's internet browser must support these languages.

There are many **advantages** to client-side scripting including faster response times, a more interactive application, and less overhead on the web server. Client-side code is ideal for when the page elements need to be changed without the need to contact the database. A good

example would be to dynamically show and hide elements based on user inputs e.g. input validation However, disadvantages of client-side scripting are that scripting languages require more time and effort, while the client's browser must support that scripting language.
JavaScript is a *high-level, dynamic, untyped, and interpreted programming language.* It has been standardized in the ECMAScript language specification. Alongside *HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production;* the majority of websites employ it, and all modern Web browsers support it without the need for plug-ins. *It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.*
Features :-

- *Universal support :* All modern Web browsers support JavaScript with built-in interpreters.
- *Imperative and structured :* supports much of the structured programming syntax from C
- *Dynamic :* a variable that is at one time bound to a number may later be re-bound to a string
- *Prototype-based (Object-oriented) :* JavaScript has a small number of built-in objects, including Function and Date
- *Functional*
- *Delegative* : JavaScript supports implicit and explicit delegation.
- *Miscellaneous* : run-time environment, arrays and objects, regular expressions
- *Vendor-specific extensions* : officially managed by Mozilla Foundation

```
<script>
    document.getElementById('hellobutton').onclick = function() {
      alert('Hello world!');            // Show a dialog
      var myTextNode = document.createTextNode('Some new words.');
      document.body.appendChild(myTextNode);    // Append "Some new words" to the page
    };
  </script>
```

***PHP :** php.net
PHP (recursive acronym for PHP: Hypertext Preprocessor*)* is a widely-used *open source , server side scripting language* that is especially suited for web development and can be embedded into HTML. *File extension are .php, .phtml, .php3, .php4, .php5, .php7, .phps supporting OS are Unix and Windows. Influenced by C, C++, Java, Perl.*

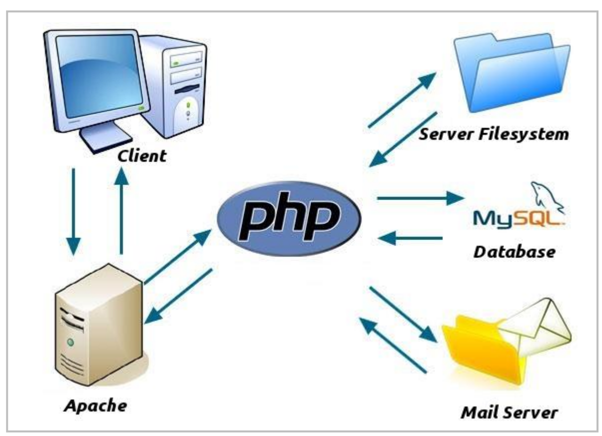
PHP is an interpreted language. It can be compiled to bytecode by third party-tools, though

```
<?php
    echo "Hi, I'm a PHP script!";
  ?>
```

- PHP is a *server-side scripting language* designed primarily for web development but also used as a general-purpose programming language. Originally created by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced by The PHP Development Team.
- PHP originally stood for *Personal Home Page*, but it now stands for the recursive acronym *PHP: Hypertext Preprocessor*.
- PHP code *may be embedded into HTML or HTML5 code*, or it can be used in combination with various web template systems, web content management systems and web frameworks.
- PHP code is usually *processed by a PHP interpreter implemented as a module* in the web server or as a Common Gateway Interface (CGI) executable. The *web server combines the results of the interpreted and executed PHP code,* which may be any type of data, including images, with the generated web page.

How does PHP works ?
PHP is a Server side Scripting Language Which means code will be executed in Server and then server sends only Plain HTML Code to Browser. Browser can understand only HTML, CSS, JAVASCRIPT which are Client-Side-Scripting Languages.

Here is the Step by Step Process:
• Client will send a HTTP/HTTPS request to Apache Server.
• PHP engine executes that Commands.
• And finally server sends HTML output to Client.
• In Client System HTML will executed by the Client Application (EX: Browser) and shows output.



PHP used in most popular websites:

1. Google
2. Facebook
3. Wikipedia
4. Yahoo
5. WordPress

***ASP:**

i. It has limited oops support and not having built in support for xml.
ii. Very less development and debugging tool available. Meaning that difficult to debug the code.
iii. ASP you can only do scripting using visual basic scripting and java scripting.
iv. Error handling is very poor.
v. It has no high level programming structure. Mixed of html and server side scripting.
vi. You must be entering first line as -

```
<%LANGUAGE="VBSCRIPT" CODEPAGE="960"%>
```

vii. It has no in built validation control. Meaning that validating page is difficult for developers.
viii. In the classic ASP if you need to update code on the existing page then it is mandatory to restart the server to get reflect.

ASP.NET

i. ASP.NET is full featured object oriented programming.
ii. It has full support of xml. Which helps easy data exchange.
iii. Various tools and compiler available. Microsoft Visual studio makes your debugging job easier.
iv. ASP.NET we can use either C# or VB.NET as server side programming language.
v. ASP.NET gives you three tire architecture. It allow you to keep your business logic, views everything separate. Meaning that easy to enhance applications.
vi. Error handling is very good.
vii. You are required to make language directive with page as below.

```
<%@Page Language="VB" CodePage="960"%>
<%@QutputCache Duration="60" VaryByParam="none" %>
```

viii. It has state management support.
ix. In built validation controls. It has rich validation set - custom validator, range validator, regular expression, compare and require field validation control which makes your job easier.

***.NET**

There are several server-side technologies that can be used when developing desktop or web applications. Server-side code uses the .NET Framework and is written in languages like C# and VB.NET. Server-side processing is used to interact with permanent storage like databases or files. The server will also render pages to the client and process user input. Server-side processing happens when a page is first requested and when pages are posted back to the server. Examples of server-side processing are user validation, saving and retrieving data, and navigating to other pages.

The disadvantage of server-side processing is the page **postback**: it can introduce processing overhead that can decrease performance and force the user to wait for the page to be processed and recreated. Once the page is posted back to the server, the client must wait for the server to process the request and send the page back to the client.

The .NET Framework is a technology that supports building and running the next generation of applications and XML Web services. The .NET

The .NET platform was designed to provide:

- The ability to make the entire range of computing devices work together and to have user information automatically updated and synchronized on all of them
- Increased interactive capability for Web sites, enabled by greater use of XML(Extensible Markup Language) rather than HTML
- A premium online subscription service, that will feature customized access and delivery of products and services to the user from a central starting point for the management of various applications, such as e-mail, for example, or software, such as Office .NET
- Centralized data storage, which will increase efficiency and ease of access to information, as well as synchronization of information among users and devices
- The ability to integrate various communications media, such as e-mail, faxes, and telephones
- For developers, the ability to create reusable modules, which should increase productivity and reduce the number of programming errors

.NET Framework to develop the following types of **applications and services**:

- Console applications. Building Console Applications.
- Windows GUI applications (Windows Forms). Windows Forms.
- Windows Presentation Foundation (WPF) applications. : create attractive and effective user interfaces
- ASP.NET applications. Web Applications with ASP.NET.
- Windows services. create long-running executable applications that run in their own Windows sessions. These services can be automatically started when the computer boots, can be paused and restarted, and do not show any user interface.
- Service-oriented applications using Windows Communication Foundation (WCF). to build secure, reliable, transacted solutions that integrate across platforms and interoperate with existing investments.

4.6 Introduction to AJAX(programming)

There are mainly four methods for **Sending request to server**

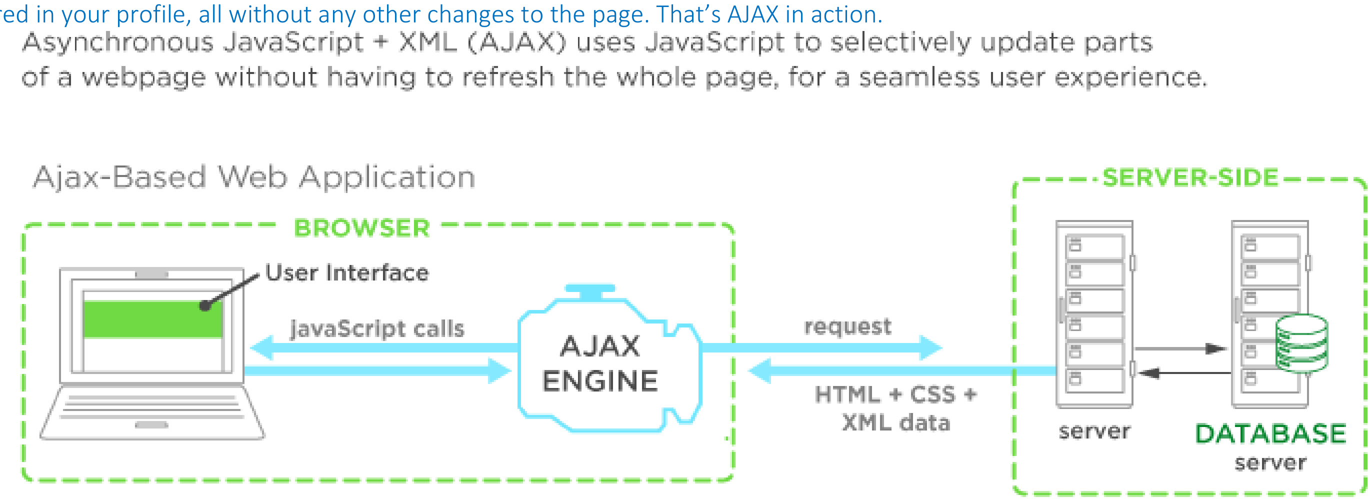
- **i)** type URL in browser
- **ii)** using hyperlink/ anchor
- **iii)** using form submit/ form action and
- **iv)** AJAX

- AJAX (**Asynchronous JavaScript And XML**) is a set of *Web development techniques using many Web technologies on the client side to create **asynchronous (Parallelly or Different process runs one at a time)** Web applications*. With Ajax, Web applications *can send data to and retrieve from a server asynchronously (in the background) without interfering or reload the entire page with the display and behavior of the existing page.*
- AJAX makes **features** like more details on scrolling page, drop-down menus, predictive text, auto-filled text, and more possible, all without clicking Refresh. What it means for servers is less stress on the network and faster operations. That's the core benefit of AJAX-enabled sites: overall, they're more responsive and very efficient on both sides.
- Ajax is not a single technology, but rather a group of technologies. *HTML and CSS* can be used in combination to mark up and style information. The *DOM* is accessed with JavaScript to dynamically display – and allow the user to interact with – the information presented. *JavaScript and the XMLHttpRequest object* provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.
- AJAX is a group of interrelated client- and server-side development technologies that allows parts of a webpage to be updated without having to reload the entire page—think of sites like YouTube, Google Maps, Gmail, and tabs within Facebook. It changed usability and the speed of web applications with its innovative concept: asynchronously exchanging small amounts of data with the server behind the scenes, without affecting the rest of the page.

How does AJAX work?

AJAX calls are *asynchronous,* meaning they're made behind the scenes, independently from each other and the site as a whole. When a browser makes an AJAX call to the server, it isn't stuck waiting for a response, halting all of the site's functionality. Instead, the web service on the server will send the data back to the browser once the task is completed, where client-side scripts will process the response and deliver it to the user.

For example: If you want to give a movie on Netflix a star rating, you can click the rating, it will show up immediately, and the rating gets stored in your profile, all without any other changes to the page. That's AJAX in action.



AJAX uses the following technologies:

- XML or JSON: the text-only format used to transfer data from server to browser script. Developers are increasingly using JSON over XML because of its native JavaScript compatibility.
- CSS: the language used to style how the data will look onscreen
- JavaScript: displays the data in the browser and processes user requests/interactions like clicks
- XMLHttpRequest **objects**: the keystone is case sensitive of AJAX, they actually retrieve the data with the server behind the scenes. All modern browsers support XMLHttpRequests.

e.g.
```
obj = new XMLHttpRequest();
    .......................use of GET method .....................
    obj.open("GET","login.php?id=5",true);  // prepare for request (true or false = asynchronous or synchronous)
    obj.send();
.......................use of POST method .....................
    obj.open("POST","login.php",true);
    obj.setRequestHeader("Content-type","application/x-www-form-urlencoded");
    obj.send("id=5");
```

Callbacks vs. Postbacks.

- A **Postback** occurs when the data (the whole page) on the page is posted from the client to the server..ie the **data is posted-back to the server**, and thus the page is refreshed (redrawn)...think of it as '**sending the server the whole page (asp.net) full of data**'.
- **A Callback is also a special kind of postback**, but it is just a quick round-trip to the server to get a small set of data (normally), and thus the page is not refreshed, unlike with the postback...think of it as '**calling the server, and receiving *some* data back**'.
- With Asp.Net, **the ViewState is not refreshed when a callback is invoked**, unlike with a postback

WHEN TO USE AJAX

The following are some of the idea situations when you should use Ajax

- **Auto complete**– if you have worked with any search engine, chances are suggestions were made to you before you could complete typing. Ajax can be used to provide such functionality
- **Auto save**– when you are composing an email and something to your internet and you lose the connection to the server, chances are you will find your work in drafts directory. This is just one example. If you develop a content management system, you can also provide similar functionality where you save the users work periodic to provide auto restore functionality.
- **Pagination**– you can only see so much at a time. Pagination allows you to display a limited number of items at a time and provide links that allow users to view the next segment. Google Search lists 10 items per page and provides you with links to other result pages. Ajax can greatly improve the user experience when displaying content that has been paginated.
- **Blog comments**– what's a blog without a comments section? The experience is improved when it is implemented using Ajax. Users can submit comments without reloading the entire page which can be very frustrating.
- **Real time validation**– you can use Ajax to provide feedback to users in real time. Its say a user is filling in a registration form on website. You can use Ajax to validate the submitted email address and let the user know if it is available before the user even submits the form.
- **…and many more**– Ajax can also be used to implement functionality such as surveys, online voting, filtering and sorting data etc.

ADVANTAGES OF AJAX

The following are some of the advantages of Ajax when developing web applications.

- **Improved User Experience**– Ajax makes it possible to create interactive applications that are fast and do not require reloading the whole page. The user can continue using the application whilst Ajax operations are going on in the background
- **Reduced bandwidth usage**– Bandwidth costs money and Ajax enables you to save. Traditional applications require reloading all of the assets even if you are only interested in a small section of the application. This results in using up a lot of bandwidth. Ajax only lets you pull the required data from the server.
- **Improved system performance**– Ajax only retrieves the required data from the server. This greatly improves the system performance and response time.
- **Promotes separation of data, business logic and presentation**– Ajax calls usually retrieve data from the server and if necessary business logic is applied. Data is displayed after these actives have successfully been completed.

DISADVANTAGES OF AJAX

- **Requires JavaScript**– JavaScript is a client-side technology and you have no control over it. If the user disables JavaScript on their web browser then Ajax will not work.
- **Web browser compatibility**– not all web browsers especially very old ones have support for all of the technologies that Ajax uses. These days most browsers support these technologies so you should worry much about this.
- **Hard/impossible to bookmark content**– users usually bookmark content so that they can easily go back to it later. With Ajax content, this is impossible or at a minimum requires extra effort to implement.
- **JavaScript Content generally isn't SEO friendly**– it is easier for Search Engines to crawl classic content compared to content that is generated via JavaScript. Developing SEO content with JavaScript requires extra efforts.
- In conclusion, in most cases, the advantages outweigh the disadvantages and you will have to work with Ajax in one way or another. The next section looks at when you should use Ajax

4.7. Browser as a rendering engine: text, HTML, gif and jpeg

It's *the bit of the web browser that's responsible displaying content.* Most commonly, that involves parsing **HTML** (the language that describes the structure of web pages) and **CSS** (the language that describes how HTML should be styled), and then rendering the web page the HTML & CSS describes.

The browser's main components are

1. **The user interface**: this includes the address bar, back/forward button, bookmarking menu, etc. Every part of the browser display except the window where you see the requested page.
2. **The browser engine**: marshals actions between the UI and the rendering engine.
3. **The rendering engine** : responsible for displaying requested content. For example if the requested content is HTML, the rendering engine parses HTML and CSS, and displays the parsed content on the screen.
4. **Networking**: for network calls such as HTTP requests, using different implementations for different platform behind a platform-independent interface.
5. **UI backend**: used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform specific. Underneath it uses operating system user interface methods.
6. **JavaScript interpreter**. Used to parse and execute JavaScript code.
7. **Data storage**. This is a persistence layer. The browser may need to save all sorts of data locally, such as cookies. Browsers also support storage mechanisms such as localStorage, IndexedDB, WebSQL and FileSystem

The rendering engine

The responsibility of the rendering engine is well... Rendering, that is display of the requested contents on the browser screen. *By default the rendering engine can display HTML and XML documents and images. It can display other types of data via plug-ins or extension; for example, displaying PDF documents using a PDF viewer plug-in.* However, rendering focus on the main use case: displaying HTML and images that are formatted using CSS.

The Main Flow

The rendering engine will *start getting the contents of the requested document* from the networking layer. This will usually be done in **8kB** chunks. After that, this is the basic flow of the rendering engine:

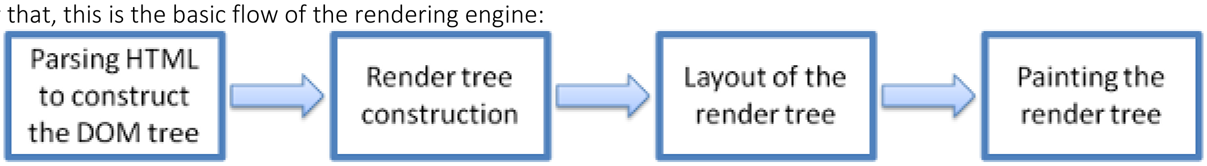


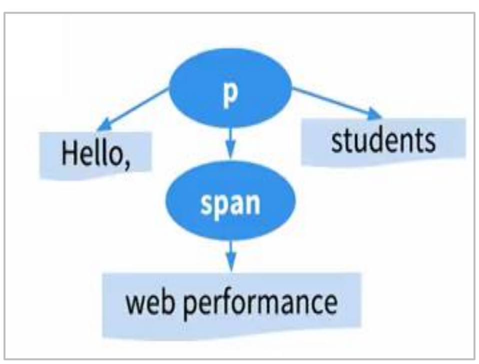
Figure : Rendering engine basic flow

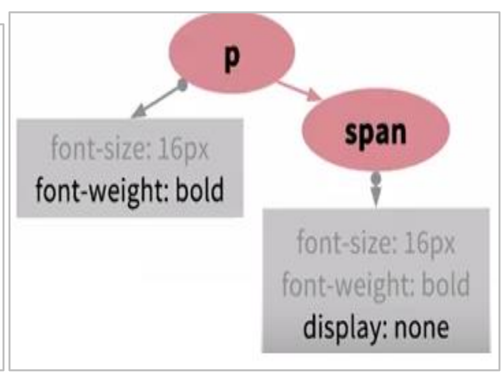
DOM (Document Object Model): is a cross-platform and language-independent application programming interface that treats an HTML, XHTML, or XML document as a tree structure wherein each node is an object representing a part of the document

- The rendering engine *will start parsing the HTML document and convert elements to DOM nodes* in a tree called the "***content tree***". The engine will *parse the style data, both in external CSS files and in style element*s. Styling information together with visual instructions in the HTML will be used to create another tree: the ***render tree.***

Example :

```
<p>
    Hello, <span> web performance </span> students
</p>
```

- 



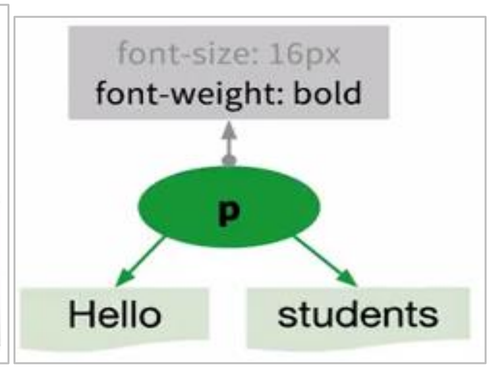
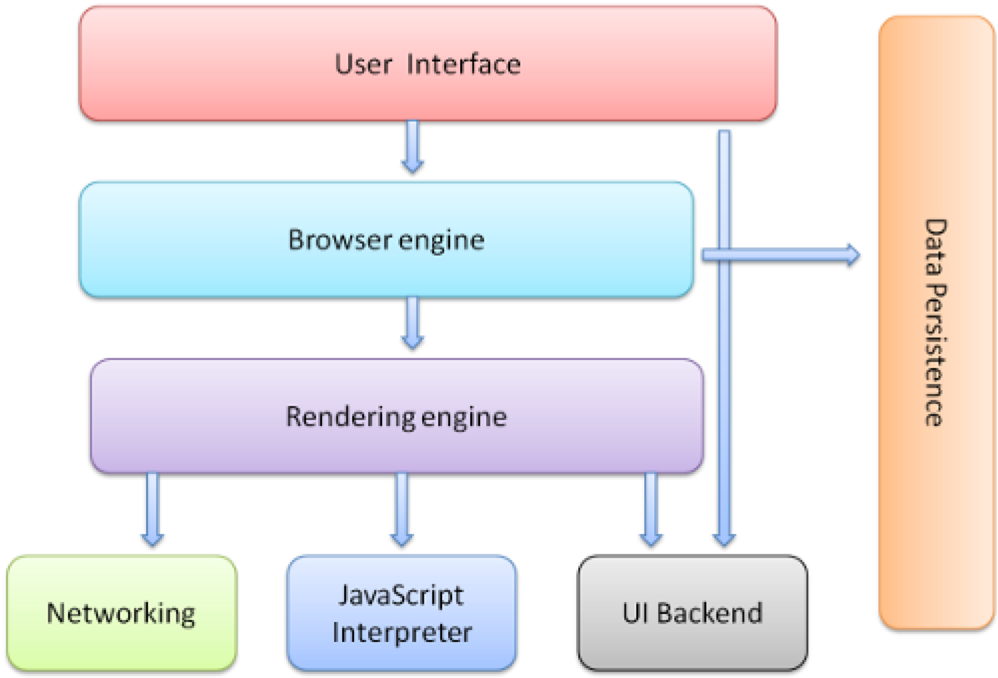


Fig 1 : DOM Tree or Content Tree *Fig 2 : CSSOM Tree* *Fig 3 : Render Tree*

- *The render tree contains rectangles with visual attributes like color and dimensions. The rectangles are in the right order to be displayed on the screen.*
- *After the construction of the render tree it goes through a "layout" process. This means giving each node the exact coordinates where it should appear on the screen. The next stage is painting–the render tree will be traversed and each node will be painted using the UI backend layer.*

It's important to understand that this is a gradual process. For better user experience, the rendering engine will try to display contents on the screen as soon as possible. It will not wait until all HTML is parsed before starting to build and layout the render tree. Parts of the content will be parsed and displayed, while the process continues with the rest of the contents that keeps coming from the network.

.

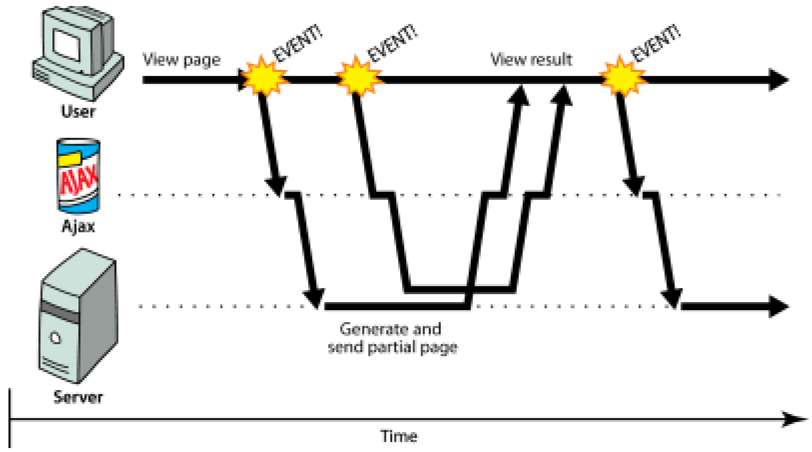


Browser	**Rendering Engine**	**Source**
Internet Explorer	Trident	Microsoft
AOL Explorer	Trident	Microsoft
Firefox	Gecko	Mozilla
Netscape	Gecko	Mozilla
Safari	WebKit	WebKit
Chrome	WebKit	WebKit
Opera	Presto	Opera
Konqueror	KHTML	KHTML

Synchronous Request: A request is called synchronous when it waits for the response of that particular request before executing the other one. i.e when a client makes a call synchronously then it blocks the client browser to ensure that client couldn't make another call before getting the server response for that previous call.

Asynchronous Request: An asynchronous call works independently i.e it doesn't wait for the server response before executing another call or request. so u can simply make different calls at the same time without waiting for the server response.

Now get to the point **i.e "How does it works in AJAX?."**



In diagram, there're three events fired at the different time. As we can see first event is fired independently n third event is fired after getting the response of first n second event, hence third event is also independent from first and second event.

lets talk about the second event, which is fired before getting the response of first event.

If first event was fired synchronously: If first event was fired synchronously then second event will not work. n client will only be able to get the response of first event n then after third event will process independently.

If first event was fired asynchronously(As shown in diagram): If first event was fired asynchronously then second event will work independently without waiting for the response of first event n client will get the response of both events accordingly n then third event will process...