Chapter 4: HTTP and the Web Services

Web Server

- Refer to either the hardware or the software that helps to deliver Web content that can be accessed through the Internet.
- The primary function of a web server is to deliver web pages on the request to clients using the HTTP.
- Each time a client request arrives, the corresponding document is sent to the client.
- To improve efficiency, servers normally store requested files in the cache in memory.

Web Access/ Web Client/ Web Browser

- The WWW today is a distributed client/server service, in which a client using a browser can access a service using a server.
- Web Browser contains three parts:
    - A controller
        - receives inputs from the keyboard or the mouse
        - use the client programs to access the documents
        - use interpreter to display the document on the screen
    - Client protocol
        - e.g. FTP, TELNET, HTTP, etc.
    - Interpreters
        - e.g. HTTP, Java, Javascript, etc.

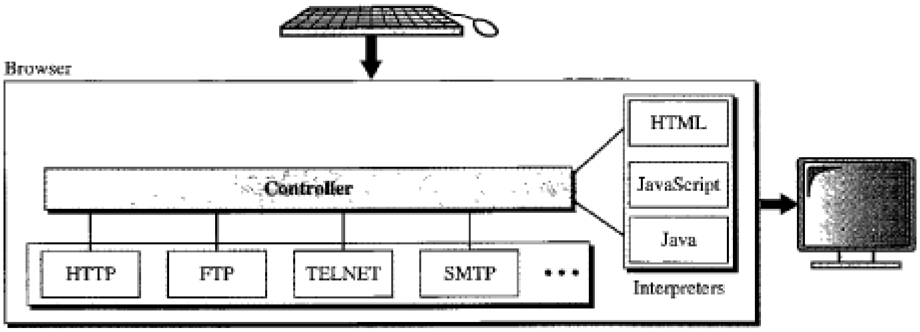


Fig. Browser

Uniform Resource Locator (URL)

- Facilitate the access of document distributed throughout the world.
- Standard for specifying any kind of information on the internet.
- Defines four things:
    - protocol
        - A client/server program used to retrieve the document
    - Host computer

        - Computer on which the information is located
        - Alias for the web document location
    - Port
        - optional field
        - port number of the server
    - Path
        - pathname of the file where the information is located
- e.g., http://www.google.com/index.html

Protocol://	Hostname:	Port/	Directory Path

Fig. URL

HTML

- **Example**

```
<HTML>
<HEAD>
<TITLE>A Sample Web Page</TITLE>
</HEAD>
<HR>
<BODY>
<center>
<H1>My Home Page</H1>
<IMG SRC="/images/myPhoto.gif">
<b>Welcome to My page!</b>
<p>
<! A list of hyperlinks follows.>
<a href="/doc/myResume.html"> My resume</a>.
<p>
<a href="http://www.someUniversity.edu/">My university<a>
</center>
<HR>
</BODY>
```

- http://www.w3schools.com/html/default.asp
- ---left for students

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.

- DHTML allows scripting languages to change variables in a web page's definition language, which in turn affects the look and function of otherwise "static" HTML page content, *after* the page has been fully loaded and during the viewing process. Thus the dynamic characteristic of DHTML is the way it functions while a page is viewed, not in its ability to generate a unique page with each page load.
- By contrast, a dynamic web page is a broader concept, covering any web page generated differently for each user, load occurrence, or specific variable values. This includes pages created by client-side scripting, and ones created by server-side scripting (such as PHP, Perl, JSP or ASP.NET) where the web server generates content before sending it to the client
- DHTML is the art of combining HTML, Javascript, DOM. *(The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents.)*
- DHTML stands for Dynamic HTML.
- DHTML is NOT a language or a web standard.
- DHTML is a TERM describing the art of making dynamic and interactive web pages.
- According to the World Wide Web Consortium (W3C):
    - *"Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."*

Uses of DHTML

DHTML allows authors to add effects to their pages that are otherwise difficult to achieve. For example, DHTML allows the page author to:

- 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 and respond to that data without having to send data back to the server.
- Include rollover buttons or drop-down menus.

A less common use is to create browser-based action games. During the late 1990s and early 2000s, a number of games were created using DHTML, but differences between browsers made this difficult: many techniques had to be implemented in code to enable the games to work on multiple platforms. Recently browsers have been converging towards the web standards, which has made the design of DHTML games more viable. Those games can be played on all major browsers and they can also be ported to Widgets for Mac OS X and Gadgets for Windows Vista, which are based on DHTML code.

The term "DHTML" has fallen out of use in recent years as it was associated with practices and conventions that tended to not work well between various web browsers. DHTML may now be referred to as unobtrusive JavaScript coding (DOM Scripting), in an effort to place an emphasis on agreed-upon best practices while allowing similar effects in an accessible, standards-compliant way.

Basic DHTML support was introduced with Internet Explorer 4.0, although there was a basic dynamic system with Netscape Navigator 4.0. When it originally became widespread, varying degrees of support among web browsers for the technologies involved made DHTML-style techniques difficult to develop and debug. Development became easier when Internet Explorer 5.0+, Mozilla Firefox 2.0+, and Opera 7.0+ adopted a shared DOM.

Structure of a web page

Typically a web page using DHTML is set up in the following way:

```
<!doctype html>
<html lang="en">
     <head>
          <meta charset="utf-8">
          <title>DHTML example</title>
     </head>
     <body>
          <div id="navigation"></div>

          <script>
               var init = function () {
                    myObj = document.getElementById("navigation");
                    // ... manipulate myObj
               };
               window.onload = init;
          </script>

          <!--
          Often the code is stored in an external file; this is done
          by linking the file that contains the JavaScript.
          This is helpful when several pages use the same script:
          -->
          <script src="myjavascript.js"></script>
     </body>
</html>
```

Example: Displaying an additional block of text

The following code illustrates an often-used function. An additional part of a web page will only be displayed if the user requests it.

```
<!doctype html>
<html lang="en">
     <head>
          <meta charset="utf-8">
          <title>Using a DOM function</title>
          <style>
               a {background-color:#eee;}
               a:hover {background:#ff0;}
               #toggleMe {background:#cfc; display:none; margin:30px 0;
padding:1em;}
          </style>
     </head>
     <body>
          <h1>Using a DOM function</h1>

          <h2><a id="showhide" href="#">Show paragraph</a></h2>

          <p id="toggleMe">This is the paragraph that is only displayed on
request.</p>

          <p>The general flow of the document continues.</p>

          <script>
               changeDisplayState = function (id) {
                    var d = document.getElementById('showhide'),
                         e = document.getElementById(id);
                    if (e.style.display === 'none' || e.style.display === '')
{
                         e.style.display = 'block';
                         d.innerHTML = 'Hide paragraph';
                    }
                    else {
                         e.style.display = 'none';
                         d.innerHTML = 'Show paragraph';
                    }
               };
               document.getElementById('showhide').onclick = function () {
                    changeDisplayState('toggleMe');
                    return false;
               };
          </script>
     </body>
</html>
```

- **DHTML Example**

```
<!DOCTYPE html>
<html>
<body>
<p id="p1">This is a text. This is a text. This is a text. This is a text. This is a text.
This is a text. This is a text.</p>
<input type="button" value="Hide text"
onclick="document.getElementById('p1').style.visibility='hidden'" />
<input type="button" value="Show text"
onclick="document.getElementById('p1').style.visibility='visible'" />
</body>
</html>
```

- For more info: http://www.w3schools.com/dhtml/default.asp

Wireless Markup Language (WML)

- New web language for making sites on mobile phones.
- Markup language intended for devices that implement the Wireless Application Protocol (WAP) specification, such as mobile phones.
- WAP provides a method to communicate across wireless networks quickly, securely and efficiently.
- It provides navigational support, data input, hyperlinks, text and image presentation, and forms, much like HTML.
- WML is a 'markup' language. This means that WML pages are written and saved as text files, using 'tags' like those found in HTML. Anyone familiar with HTML should find learning WML fairly easy.
- Usually have '.wml' extension.
- WML pages stored on the web server.
- Client access pages using WAP.
- WML pages use scripting similar to javascript.
- Can be used for tasks such as user input validation, error messages, etc.
- WML is an HTML type formatting language that has been defined as XML document type and which is optimized for small screens and limited memory capacity.
- A major difference between the implementations of WML and HTML languages is that the original ASCII based WML code is converted to binary code in the gateway server, before being forwarded to WAP device.
- In WAP development, a screen is known as a card. A collection of cards is known as a deck.

Wireless Markup Language (WML), based on XML, is a markup language intended 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).

WML Structure and Syntax

The following are three important areas in regard to WML syntax:

- WML document prologue
- WML character sets
- WML case sensitivity

WML Document Prologue

Every WML deck you write must contain a document prologue. Compilers on the device, WAP gateways, and remote servers all use document prologues to interpret your code. Developers must include the XML document prologue at the top of every WML deck:

```
1. <?xml version="1.0>
```

```
2. <!DOCTYPE wml PUBLIC “-//WAPFORUM//DTD WML 1.1//EN”
3. “http://www.wapforum.org/DTD/wml_1.1.xml”>
```

Note: Be aware of the use of spaces in the document prologue. A WML deck produces errors if the document prologue does not include correct use of spacing and punctuation.

The following is a line-by-line explanation of the document prologue:

The first line designates the XML version of the WAP server and WML compiler. WAP servers and WML compilers use XML to interpret your code. These servers and compilers then transform this information back into WML, so that a WAP device can display the information.

The second line defines the version of WML used. This states that you will use WML version 1.1 within your applications.

Note: In the document protocol the XML and WML versions are not the most current. The reason for this is that not all WAP enabled devices support the latest versions of these languages. Unless you are writing an application for specific device, use the language versions specified in this prologue.

The third line specifies the location of the WML document type definition (DTD). Any additional extensions or information for the WAP server or compiler are available from this site.

WML Character Sets

Character sets are the built-in methods for handling text characters. They represent letters and symbols specific to several languages. WML supports the character set of ISO-10646 (Unicode 2.0). Because this character set is the default for WML, you don’t need to declare it within your WML decks.

WML Case Sensitivity

WML is a precise language when it comes to case sensitivity. Character case does matter in WML. The basic rule of WML case sensitivity is that all tags and attributes must be lower case. For example, name1 and NAME1 are different variables.

Note: Most programming errors tend to deal with problems with character case. As a rule of thumb if your code doesn’t work check for the correct use of character case.

WML markup

WML documents are XML documents that validate against the WML DTD (Document Type Definition) . The W3C Markup Validation service (http://validator.w3.org/) can be used to validate WML documents (they are validated against their declared document type).

For example, the following WML page could be saved as "example.wml":

```
<?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>
```

A WML document is known as a “deck”. Data in the deck is structured into one or more “cards” (pages) – each of which represents a single interaction with the user.

WML decks are stored on an ordinary web server configured to serve the text/vnd.wap.wml MIME type in addition to plain HTML and variants. The WML cards when requested by a device are accessed by a bridge WAP gateway, which sits between mobile devices and the World Wide Web, passing pages from one to the other much like a proxy. The gateways send the WML pages on in a form suitable for mobile device reception (WAP Binary XML). This process is hidden from the phone, so it may access the page in the same way as a browser accesses HTML, using a URL (for example, http://example.com/foo.wml). (Provided the mobile phone operator has not specifically locked the phone to prevent access of user-specified URLs.)

WML has a scaled down set of procedural elements which can be used by the author to control navigation to other cards.

Consider a service that lets you enter a zip code, and obtain a list of clickable phone numbers of pizza parlors and taxicabs in your immediate location:

```
<card id="cM" title="MY_DOMAIN.com">
  <p>
    <b>Call A Taxi:</b><br />
    <a href="wtai://wp/mc;%2B19035551212">903-555-1212</a>
  </p>
</card>
```

Mobile devices are moving towards support for greater amounts of XHTML and even standard HTML as processing power in handsets increases. These standards are concerned with formatting and presentation. They do not however address cell-phone or mobile device hardware interfacing in the same way as WML.

WAP Design Considerations

When developing any type of WAP applications, keep in mind that WAP devices contain limited network bandwidth capabilities and memory compared to a PC.

The following are few tips to keep in mind when creating WAP applications:

- Keep WML decks and images to less than 1.5KB
- Avoid overuse of images
- Use simple and brief text within the applications if possible

WML Example

```
<?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>
```

XML

- XML stands for e**X**tensible **M**arkup **L**anguage.
- XML is designed to transport and store data.
- XML is important to know, and very easy to learn.
- XML is extensible: it does not contain a fixed set of tags.
- XML documents must be well-formed according to a defined syntax, and may be formally validated.
- XML focuses on the meaning of data, not its presentation.
- XML allows a document to be marked up for structured information.

XML document Example

```
<?xml version="1.0"?>
<note>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
</note>
```

For more info: http://www.w3schools.com/xml/

---more youself

WYSIWYG Authoring Tolls

- **WYSIWYG** is an acronym for *what you see is what you get*.
- The term is used in computing to describe a system in which content (text and graphics) displayed onscreen during editing appears in a form closely corresponding to its appearance when printed or displayed as a finished product, which might be a printed document, web page, or slide presentation.
- WYSIWYG implies a user interface that allows the user to view something very similar to the end result while the document is being created.
- In general WYSIWYG implies the ability to directly manipulate the layout of a document without having to type or remember names of layout commands.
- In web pages, WYSIWYG means the display precisely represents the appearance of the page displayed to the end-user.
- E.g., WYSIWYG XML Authoring (http://www.altova.com/authentic/wysiwyg-xml.html), WYSIWYG HTML Editor, WYSIWYG Web Builder, etc.

Browser

- A *Browser* is a software application that allows users to navigate through the web, display web pages that consist of links, text, pictures, and other media
- Browsers can be roughly classified into two types:
  - **Text Browsers**
    Render only text and ignore all images and multimedia elements
    Example: Lynx
  - **Graphical Browsers**
    Visual browsers-Render text, images, colors, graphs and other multimedia elements
    Example: Microsoft Internet Explorer, Netscape, Mozilla, Mosaic and Opera

Common Gateway Interface (CGI)

- The Common Gateway Interface (CGI) is a simple interface for running external programs, software or gateways under an information server in a platform-independent manner.
- An HTTP server is often used as a gateway to a legacy information system; for example, an existing body of documents or an existing database application. The Common Gateway Interface is an agreement between HTTP server implementers about how to integrate such gateway scripts and programs.
- CGI is a set of rules that describe how a web server communicates with another piece of software (CGI program).
- Common Gateway Interface (CGI) protocol is the first widely adopted protocol to augment HTTP in supporting run-time generated web contents.
- The Common Gateway Interface (CGI) is a standard for providing an interface, or a gateway, between an information server and an external process (that is, a process external to the server).
- Using the protocol, a web client may specify a program, known as a CGI script, as the target web object in an HTTP request.
- The web server fetches the CGI script, activates it as a process, passing to the process input data transmitted by the web client. The web script executes and transmits its output to the web server, which returns the web-script generated data as the body of a response to the web client.
- A CGI program can be written in:

  - Programming languages: C. Ada, C++, Fortran; such a program needs to be compiled to generate an executable.
  - Script languages such as Php, Perl, Tkl, cobra, such a program, referred to as a CGI script, requires the appropriate language interpreter to be present at the server host.
  - Commonly used for processing user input from HTML forms, and subsequently composing a web page sent as part of the server response.

Significance of CGI

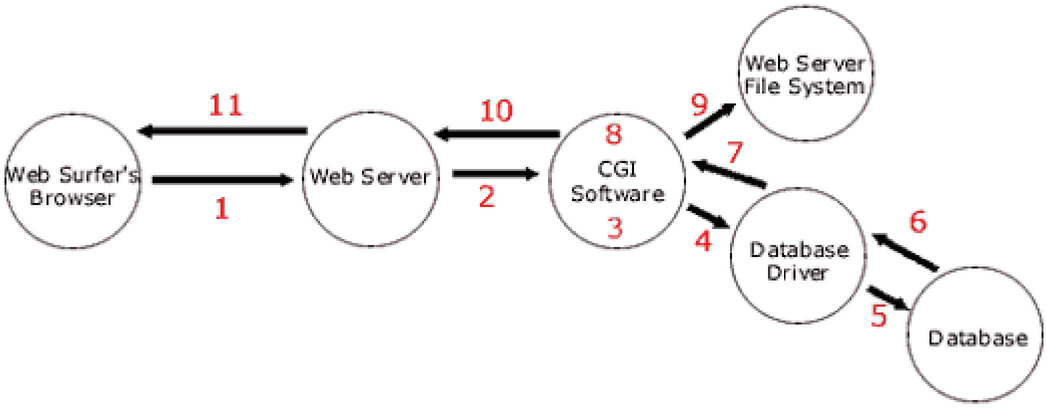
The importance of CGI lies in the fact that its flexibility has made it a standard for running executable files from Web servers. This standard allows for true interactivity, in endless ways, on Web sites. For example, CGI scripts can implement the following kinds of features

- **Access Counters**
  - Display the number of visitors to your site in a text-based or graphical manner
- **Advertisements**
  - Set up banner rotations on your Web page and track their statistics
- **Audio Management**
  - Provide and manage audio files in different formats for users to listen to
- **Bulletin Board Message Systems**
  - Provide on-line message forums for threaded discussions
- **Calendars**
  - Schedule events and/or allow users to post dated information
- **Chat**
  - Provide for real-time chats on the Web
- **Classified Ads**
  - Allow users to post information on buying, selling, and/or trading possessions
- **Clocks**
  - Display the current time on your Web page in an image- or text-based format
- **Commerce and Finance**
  - Allow users to use calculators, credit cards, etc.
- **Content Retrieval**
  - Retrieve and integrate content of all kinds into your Web site (e.g., news and headlines, stock quotes, weather)
- **Cookies**
  - Track visitors and store user information
- **Countdowns**
  - Display on your Web page the length of time until a specified event
- **Customer Support**
  - Maintain knowledge bases; provide for customer support e-mail ticketing, real-time customer support, and FAQ maintenance
- **Database Manipulation**
  - Create, edit, and manipulate databases; allow users to search your databases
- **Development Environments**
  - Aid in the development of programs and promote collaboration
- **Editing Web Pages**
  - Allow users or administrators to create and edit Web pages
- **File Management**
  - Manage your files and directories via the Web (e.g., file downloading/uploading, link protection)
- **Form Processing**
  - Process basic forms and send results via email
- **Games**

- ➢ Allow users to play games on your Web site
- **Guestbooks**
  - ➢ Allow visitors to sign in and leave a message on your Web site
- **HTML Manipulation**
  - ➢ Create and insert tables, frames, lists, headers, and footers
- **Homepage Communities**
  - ➢ Allow visitors to create their own customized Web pages on your site
- **Image Display**
  - ➢ Display images in different ways and formats; index images; allow for picture posting; timed rotation of images
- **Imagemaps**
  - ➢ Create clickable images that redirect the user to another page
- **Instant Messaging**
  - ➢ Allow users to participate in instant messaging or to use various features of instant messaging systems
- **Interactive Stories**
  - ➢ Allow visitors to add to an existing story
- **Internet Utilities**
  - ➢ Provide for a wide range of standard Internet programs (e.g., finger, telnet, traceroute, whois)
- **Link Indexing Scripts**
  - ➢ Allow visitors to add links to your Web site
- **Link Verification**
  - ➢ Test the links on your page; evaluate your server and its performance
- **Logging Accesses and Statistics**
  - ➢ Track the number of visitors to your site; log useful statistics regarding your site and visitors
- **Mailing Lists**
  - ➢ Allow for mailing lists and management of existing mailing lists
- **News Posting**
  - ➢ Post and manage news and updates to your site
- **Password Protection**
  - ➢ Password protect your Web site
- **Postcards**
  - ➢ Allow visitors to your Web page to send an Internet postcard to someone
- **Random Items**
  - ➢ Include random links, text, images, pages, etc. on your Web site
- **Redirection**
  - ➢ Redirect users in various ways (e.g., jump boxes, browser-based, error-based)
- **Reservations and Scheduling**
  - ➢ Make reservations and usage schedules for items, people, etc.
- **Searching**
  - ➢ Allow users to perform searches on your web site using keywords and phrases
- **Shopping Carts**
  - ➢ Set up an on-line store using shopping cart and catalog features

- **Spam Prevention**
  - ➢ Randomly generate fake (but real-looking) e-mail addresses for spammer robots to pick up, resulting in a bunch of bounced spam for the spammer and less spam for you and your users
- **Surveys and Voting**
  - ➢ Conduct surveys, take votes, allow users to post ratings and reviews
- **Tests and Quizzes**
  - ➢ Create and grade tests and quizzes automatically over the Web
- **Web Server Maintenance**
  - ➢ Maintain and monitor your server
- **Web-Based E-Mail**
  - ➢ Supply e-mail access to users through their Web browser
- **Website Promotion**
  - ➢ Set up affiliate programs for sites referring visitors to you; set up contests/awards to attract visitors to your site; set up a link exchange between sites with banner ads; allow users to recommend your Web site to a friend; submit your URL to search engines; track referrers; create your own ring of Web sites

How CGI Script Works



1. The Web surfer fills out a form and clicks, “Submit.” The information in the form is sent over the Internet to the Web server.
2. The Web server “grabs” the information from the form and passes it to the CGI software.
3. The CGI software performs whatever validation of this information that is required. For instance, it might check to see if an e-mail address is valid. If this is a database program, the CGI software prepares a database statement to either add, edit, or delete information from the database.
4. The CGI software then executes the prepared database statement, which is passed to the database driver.
5. The database driver acts as a middleman and performs the requested action on the database itself.
6. The results of the database action are then passed back to the database driver.
7. The database driver sends the information from the database to the CGI software.

8. The CGI software takes the information from the database and manipulates it into the format that is desired.
9. If any static HTML pages need to be created, the CGI program accesses the Web server computer’s file system and reads, writes, and/or edits files.
10. The CGI software then sends the result it wants the Web surfer’s browser to see back to the Web server.
11. The Web server sends the result it got from the CGI software back to the Web surfer’s browser.

PERL

- PERL stands for *Practical Extraction Report Language*.
- PERL
  - Is an interpreted language.
  - Runs on multiple platforms: Windows, Unix, Mac, etc.
  - Is a scripting language.
  - Is a typeless language.
  - Has some aspects similar to C language.
  - Could be as procedural as you want it to be.
  - Could be as object-oriented as you want it to be (PERL 5).
- PERL can be downloaded from: http://www.perl.com/pub/a/language/info/software.html
- PERL is a case-sensitive language just like C or Java.
- "#"sign is used for comments in PERL
  - ***Example***:

    ```
    #!/usr/bin/perl
    # This program is to . . .
    ```

Perl Programming Standards

- The following coding standards should be followed:

  - Start the program with a header comment giving info about the PERL path, programmer, copyrights, last modified date, and description.

    Example:

    ```
    #!/usr/bin/perl
    ########################################
    # Ashok K. Pant
    # Copyrights © 2012
    # All rights reserved.
    #
    # Last modified 07/04/2012
    # This program is to . . .
    ########################################
    ```

  - Use three-line-comment style.

    Example:

    ```
    #
    # Incrementing variable $count
    #
    $count++;
    ```

- Use meaningful names for variables or subroutines. Capitalize every word except the first one.

  Example:

  ```
  $myBuffer=1;
  sub printThankYouMessage(){
  …}
  ```

- For file handlers such as STDIN, STDOUT use all-cap identifiers.
- Use three-space indentation. ***Example***:

  ```
  #
  # comment here…
  #
  $buffer = ";
  foreach $field (@array1){
     foreach $value (@array2){
        $buffer .= "$field: $value\n";
     }
  }
  ```

PERL Data types

- PERL has three built-in data types: scalars, arrays of scalars, and associative arrays of scalars, known as "hashes".
- Scalar Variables
  - A scalar may contain one single value in any of three different flavors: a number, a string, or a reference.
  - Scalar values are always named with '$‘ at the beginning, even when referring to a scalar that is part of an array or a hash.

    Examples:

    ```
    $day  #A simple scalar value "day"
    $day[28] #the 29th element of @day
    $day{'Feb'} #The 'Feb' value from %day
    ```
- Array Variables
  - An array is basically a way to store a whole bunch of scalar values under one name.
  - An array name begins with an "@" symbol.

    Examples:

    ```
    @arrayName = ("element1", "element2");
    @browser = ("NS", "IE", "Opera");
    ```

    @one_two_thre = (1, 2, 3);
  - To access a single element is an array, use a $+array name+[+ index+]. Array indexing starts form zero. *Examples*:
    $arrayName[0] # the first element of the array
    $browser[1]   # This will return ‘IE’.
- Associative Arrays “Hashes”
  - Associative arrays are created with a set of key/value pairs. A key is a text string of your choice that will help you remember the value later.
  - A hash name begins with % sign.
    Examples:
    %hashName = ('key1', 'value1', 'key2', 'value2');
    %ourFriends = ('best', 'Don', 'good', 'Robert', 'worst', 'Joe');
  - To access an element, use $+hash name+{+key+}. *Examples*:
    $hashName{‘key1’}   #This will return value1
    $ourFriends{'good'} #This will return ‘Robert’

PERL Operators

- PERL uses:
  - Arithmetic operations: +, -, *, /, %,**.
  - Relational operations: <, >, <=, >=, ==.
  - String Operations: ., x, eq, ne, lt, gt, le, ge.
  - Assignment Operators: =, +=, -+, *=, /=, .=.
  - Increment/decrement operators: ++, --.
  - Boolean operations: &&, ||, !.
  - Quotation marks:
    - ”” character string with variable interpolation.
    - ’’ character string without variable interpolation.
    - ``   system commands stings with variable interpolation.
    - qq|…| character string with variable interpolation.

PERL Control Structures

- If /else control structure looks like:

```
if(condition){
        If body
}
else{
         Else body
}
```

***Example*:**

```
if ($gas_money < 10) {
        print "You don’t have enough money!";
}
else {
            print "You have enough money.";
}
```

- For loop has the following C-based formula:

  ```
  for (initialize;condition; increment){
              code to repeat
            }
  ```

  ***Example*:**

  ```
  for ($count=1; $count<11; $count++){
            print "cool\n";
  }
  ```

- While loop has also the following C-based formula:

  ```
   while (test condition) {
             code to repeat
  }
  ```

  ***Example*:**

  ```
  $count=1;
  while ($count < 11){
              print "$count\n";   $count++;
  }
  ```

- foreach has the following formula:

  ```
  foreach variable_name (array_name){
  code to repeat
  }
  ```

  ***Example*:**

  ```
  foreach $item(@inventory){
              print “$item\n”;
  }
  ```

Dealing with Files in Perl

- Reading Files:
  - STEP 1: Open the file for reading:
    open(FILE, “data.txt") || die (“Can’t Open file”);
  - STEP 2: Read the data from the file:
    @rawData = <FILE>;
  - STEP 3: Close the file:
    close(FILE);

- Writing to files:
  - ➢ STEP 1: Open the file for writing:
    open(FILE, “>data.txt") || die (“Cannot Open file”);
  - ➢ STEP 2: Write data to the file:
    print FILE “New Data”;
  - ➢ STEP 3: Close the file:
    close(FILE);

- Appending Files:
  - ➢ STEP 1: Open the file for appending:
    open(DFH, “>>data.txt") || die (“Cannot Open file”);
  - ➢ STEP 2: Write data at the end of the file:
    print DFH “Another data line”;
  - – STEP 3: Close the file:
    close(DFH);

Perl Subroutines

- A subroutine is a named group of statements that does a specific job and can be called over and over. Subroutines are very important for software reuse.
- The formula of creating subroutines in PERL is

```
sub subName(){
        group of statements
}
```

- TO call a subroutines as follows:
  subName(arg1, arg2, arg3, …);
- PERL pass the arguments to subroutines in an array named @_

Useful build in commands

- The chop function is used to "chop off" the last character of a string variable
  Example: chop(“Hii”); #return “Hi”
- The length function simply gives you back the number of characters in a string variable.
  Example: length(“Hello”); #return 5
- The substring function is a way to get a portion of a string value, rather than using the entire value.
  Example:
  $str = substr(“Cold”, 1, 3); #return “old”

- The split function separates a string expression into a list.
  ***Example*:**

my (@pairs) = split(/&/, $buffer);

- The translate(tr) function exchanges each occurrence of a character in the search string with its matching character in the replacement string.
  Example: $value =~ tr/+/ /;
- The substitute(s///) function searched the search string for the input pattern and replaces it with the replacement pattern.
  Example:
  $searchString =~ s/OPattern/NPattern/g;

Regular Expressions

- PERL has three primary functions that use regular expressions: split, substitute(s///) and pattern-match(m//).
- These functions are used to extract the data sent by the client before processing it on the server side.
  Example:
  ($name, $value) = split(/=/,$pair);
  $value =~ tr/+/ /;
  $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex($1))/eg;

Environment variables

- Associative array %ENV is used to carry on the information coming from the client. PERL gives you an access to this variable. Therefore you start your CGI program by extracting this information.

Name	What it Gets
CONTENT_LENGTH	Number of characters submitted from a POST command
HTTP_USER_AGENT	Type of Browser viewer is using.
QUERY_STRING	String in URL after the ? character
REMOTE_ADDR	Remote user's IP address

Database Connection in Perl

- You can use two ways to access a database from a PERL program.
  - Using DBI
  - Using ODBC
- Using ODBC is currently supported by every major software company in the world.

Simple CGI Example using PERL

Here is a simple CGI script. Save it as ‘hello.cgi’. This file is being kept in /cgi-bin/ directory. Before running your CGI program make sure you have change mode of file using **chmod 755 hello.cgi** UNIX command.

```
#!/usr/bin/perl


 print "Content-type: text/html\n\n";
 print <<HTML;
            <html>
            <head>
            <title>A Simple Perl CGI</title>
            </head>
            <body>
            <h1>A Simple Perl CGI</h1>
            <p>Hello World</p>
            </body>
 HTML
 exit;
```

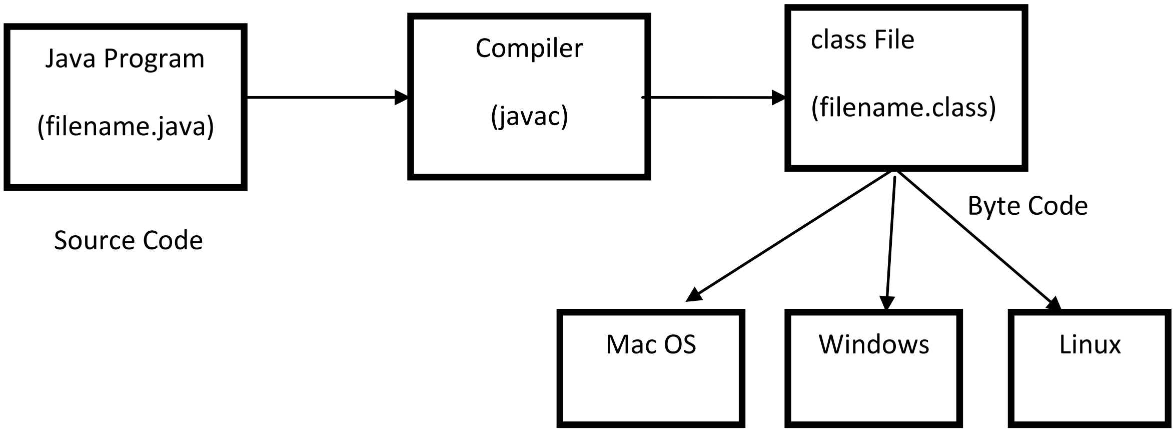
```
#!/usr/local/bin/perl
#   A simple Perl CGI script
print "Content-type: text/html\n\n";
print "<head>\n";
print "<title>Hello, World</title>\n";
print "</head>\n";
print "<body>\n";
print "<font color = blue>\n";
print "<h1>Hello, World</h1>\n";
print "</font>\n";
print "</body>\n";
```

```
#!/usr/bin/perl
$title = "My first Script";
$greeting = "Welcome to my first
script.";
print "Content-type: text/html\n\n";
print <<EOF;
<html>
<head>
```

```
<title>$title</title>
</head>
<body>
<h1>$title</h1>
<p>$greeting</p>
</body>
</html>
EOF
```

- Line 1: #!/usr/bin/perl
  Every Perl program starts with #!/usr/bin/perl.
  It tells the program where to find the Perl interpreter so that the script can run
- Line 2: $title = "My first Script";
- Line 3: $greeting = "Welcome to my first script.";
  Lines 2 and Line 3 assigns values to the variables. Variables in Perl are always preceded by the $ sign
- Line 4: print "Content-type: text/html\n\n";
  Line 4 tells the browser that the content will be in text/html and skips a line
- Line 5: print <<EOF;
  …
  EOF
  This print command tells the program to print the HTML code until it finds the word EOF. Line
  6 to Line 14 act in the same way as in PHP

Java



- Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithread, and dynamic language.
- Java virtual machine (JVM)
  A Java virtual machine is software that is implemented on virtual and non-virtual hardware and on standard operating systems. A JVM provides an environment in which Java bytecode can be executed, enabling such features as automated exception handling, which provides *root-cause* debugging information for every software error (exception), independent of the source code. A JVM is distributed along with a set of standard class libraries that implement the Java application programming interface (API). Appropriate APIs bundled together with JVM form the Java Runtime Environment (JRE).
- Java data types
  - java is a strongly typed language
  - byte, short, int, long, float, double, boolean, char

A sample java code

```
Class Example {
/* This is a simple Java program
Example. java
*/
public static void main (String args [])
{
//Prints My first java program
System.out.println("My First Java
Program");
}
}
```

Java operators and control structures

- **Arithmetic operators**
  +, -, *, /, %, ++, --, +=, -=
- **Logical operators**
  ~, &, ||
- **Relational operators**
  =, >, <,!=
- **Bitwise operators**
  <<, >>
- **Control statements in java are similar to the control statements in C/C++**
  if, switch, for, while, do, break, return, continue

Java and the Internet

- Java has a technology that allows small programs known as “applets” to be embedded in web pages.

  An Applet Example

  ```
  import java.awt.*;
  import java.applet.*;
  public class FirstApplet extends Applet{
          public void paint(Graphics g){
                  g.drawString(“My First Applet”,20,20):
          }
  }
  ```
- Line 1 and Line 2 imports AWT (Abstract Window Toolkit) package and Applet package.
- Line 4 *paint()*- applet redisplays its output.
- Line 6 *drawString()* **Void drawstring**(**String** Message, **int** x ,**int** y)In a Java window 0,0 refers to the upper left corner location.

Java Applet Embedded to Internet

```
<HTML>
<HEAD>My First Applet</HEAD>
<BODY>
<appletcode=”First Applet” width=200 height=60>
</applet>
</BODY>
</HTML>
```

- Execution of applet: Web browser and HTML file with appropriate applet tag
- Width and height specify the dimensions of the display area used by the applet

Java script

- **What is JavaScript?**
    - Language developed by Netscape
    - Primary purpose is for "client-end" processing of HTML documents
        - JavaScript code is embedded within the html of a document
        - An interpreter in the browser interprets the JavaScript code when appropriate
        - Code typically allows for "preprocessing" of forms and can add "dynamic content" to a Web page

- **How to include JavaScript in html?**
    - JavaScript programs require the <SCRIPT> tag in .html files

      <script type = "text/javascript">

      ACTUAL JavaScript code here

      </script>
    - These can appear in either the <HEAD> or <BODY> section of an html document
        - Functions and code that may execute multiple times is typically placed in the <HEAD>
            - These are only interpreted when the relevant function or event-handler are called
        - Code that needs to be executed only once, when the document is first loaded is placed in the <BODY>
    - Note that, unlike PHP scripts, JavaScripts are visible in the client browser
        - Since they are typically acting only on the client, this is not a problem

    - However, if we want to prevent the script itself from being (easily) seen, we can upload our JavaScript from a file
        - This will show only the upload tag in our final document, not the JavaScript from the file
        - Use the src option in the tag

          <script type = "text/javascript" src = "bogus.js"></script>
            - However, the source is likely still not really hidden
            - In a temp folder somewhere on computer and can be seen with a utility like FireBug
            - Thus you should not "hardcode" into Javascript anything that you don't want the client to see

    - JavaScript variables have no types
        - Type is determined dynamically, based on the value stored
            - This is becoming familiar!
            - The **typeof** operator can be used to check type of a variable
    - Declarations are made using the **var** keyword
        - Can be implicitly declared, but not advisable
        - Declarations outside of any function are global

    - Declarations within a function are local to that function
    - Variables declared but not initialized have the value **undefined**
  - Variable identifiers are similar to those in other languages (ex: Java)
    - Cannot use a keyword
    - Must begin with a letter, $, or _
      - Followed by any sequence of letters, $, _ or digits
    - Case sensitive

  - Numeric operators in JavaScript are similar to those in most languages

    +, –, *, /, %, ++, --

    - Precedence and associativity are also fairly standard
  - Strings
    - Have the + operator for concatenation
    - Have a number of methods to do typical string operations
      - charAt, indexOf, toLowerCase, substring
    - Looks kind of like Java – intentionally
  - Similar to PHP, with mixed number/string type expressions, JavaScript will coerce if it can
    - If operator is + and an operand is string, it will always coerce other to string
    - If operator is arithmetic, and string value can be coerced to a number it will do so
      - If string is non-numeric, result is NaN (NotaNumber)
    - We can also explicitly convert the string to a number using parseInt and parseFloat
    - Again looks like Java

- **Relational operators:**

  ==, !=, <, >, <=, >=
    - The above allow for type coercion. To prevent coercion there is also ===, !==
      - Similar to PHP
    - Boolean operators

      &&, ||, !
    - &&, || are short-circuited (as in Java and PHP)
      - Discuss
- **Control statements similar to Java**
  - if, while, do, for, switch
    - Variables declared in for loop header are global to the rest of the script

- **Functions**
  - Similar to Java functions, but
    - Header is somewhat different

      function name(param_list)

        - Return type not specified (like PHP, since JS has dynamic typing)
        - Param types also not specified
    - Functions execute when they are called, just as in any language
    - To allow this, function code should be in the <HEAD> section of the .html file
    - Variables declared in a function are local to the function
    - Parameters are all value
        - No parameter type-checking
        - Numbers of formal and actual parameters do not have to correspond
            - » Extra actual parameters are ignored
            - » Extra formal parameters are undefined
        - All actual parameters can be accessed regardless of formal parameters by using the **arguments** array

- More relaxed version of Java arrays
    - Size can be changed and data can be mixed
    - Cannot use arbitrary keys as with PHP arrays
- Creating arrays
    - Using the new operator and a constructor with multiple arguments
      var A = new Array("hello", 2, "you");
    - Using the new operator and a constructor with a single numeric argument
      var B = new Array(50);
    - Using square brackets to make a literal
      var C = ["we", "can", 50, "mix", 3.5, "types"];

- Array Length
    - Like in Java, length is an attribute of all array objects
    - However, in Javascript it does not necessarily represent the number of items or even mem. locations in the array
    - Unlike Java, length can be changed by the programmer
    - Actual memory allocation is dynamic and occurs when necessary
        - An array with length = 1234 may in fact have memory allocated for only a few elements
        - When accessed, empty elements are undefined

- **Array Methods**

    - There are a number of predefined operations that you can do with arrays
        - concat two arrays into one
        - join array items into a single string (commas between)
        - push, pop, shift, unshift
            - Push and pop are a "right stack"
            - Shift and unshift are a "left stack"
        - sort
            - Sort by default compares using alphabetical order

        - To sort using numbers we pass in a comparison function defining how the numbers will be compared
      - reverse
        - Reverse the items in an array
    - These operations are invoked via method calls, in an object-based way
      - Also many, such as sort and reverse are mutators, affecting the array itself
  - JavaScript also has 2-dimensional arrays
    - Created as arrays of arrays, but references are not needed

- **JavaScript is an object-based language**

  - It is NOT object-oriented
  - It has and uses objects, but does not support some features necessary for object-oriented languages
    - Class inheritance and polymorphism not supported
      - They can be "faked" but are not really there
      - http://www.webreference.com/js/column79/
      - http://www.webreference.com/js/column80/

  - JavaScript objects are actually represented as property-value pairs
    - Actually similar to keyed arrays in PHP
    - The object is analogous to the array, and the properties are analogous to the keys
      - However, the property values can be data or functions (methods)
    - Ex:

```
var my_tv = new Object();
my_tv.brand = "Sony";
my_tv.size = 34;
 my_tv.jacks = new Object();
my_tv.jacks.input = 4;
 my_tv.jacks.output = 2;
```

    - Note that the objects can be created and their properties can be changed dynamically
    - Also, objects all have the same data type – object
    - We can write constructor functions for objects if we'd like, but these do not create new data types – just easy ways of uniformly initializing objects

```
function TV(brand, size, injacks, outjacks)
{
   this.brand = brand;
   this.size = size;
   this.jacks = new Object();
   this.jacks.input = injacks;
   this.jacks.output = outjacks;
}
…
var my_tv = new TV("Sony", 34, 4, 2);
```

- Once an object is constructed, I can change its properties if I want to
  - Let’s say I want to add a method to my TV called "show_properties"

```
function show_properties()
{
   document.write("Here is your TV: <BR/>");
   document.write("Brand: ", this.brand,"<BR/>");
   document.write("Size: ", this.size, "<BR/>");
   document.write("Input Jacks: ");
   document.write(this.jacks.input, "<BR/>");
   document.write("Output Jacks: ");
   document.write(this.jacks.output, "<BR/>");
}
…
my_tv.show = show_properties;
```

What can Javascript do?

- **JavaScript gives HTML designers a programming tool -** HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- **JavaScript can react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can manipulate HTML elements -** A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form input
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

See more on: http://www.w3schools.com/js/default.asp

Example

```
<HTML>
<HEAD>
<TITLE>First JavaScript Example</TITLE>
</HEAD>
```

```
<BODY>
<H2>This line is straight HTML</H2>
<H3>
<SCRIPT type = "text/javascript">
   document.write("These lines are produced by<br/>");
   document.write("the JavaScript program<br/>");
   alert("Hey, JavaScript is fun!");
</SCRIPT>
</H3>
<H2>More straight HTML</H2>
<SCRIPT type = "text/javascript" src="bogus.js"></script>
</BODY>
</HTML>
```

```
<html>
<body>
<script type="text/javascript">
document.write("My First JavaScript!")
</script>
</body>
</html>
```

- Line 1:<html>
- Line 2:<body> HTML tags
- Line 3:<script type=”text/javascript”> Indicates the scripting language used in the HTML page. Code follows is a JavaScript code.
- Line 4:document.write (“My First JavaScript”) Write an output to the HTML page.
- Line 5:</script> To close script tag
- Line 6:</body>
- Line 7:</html> To close HTML tags

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
</head>
<body>
```

```
<h1>My First Web Page</h1>
<p id="demo">This is a paragraph.</p>

<button type="button" onclick="displayDate()">Display Date</button>

</body>
</html>
```

Server side scripting

- Server side scripting languages run on the machine that is serving the website
- PHP, ASP, JSP and PERL are some of the server side scripting languages
- Advantage - you can hide the code from the users who are accessing the web page

PHP (Hypertext Preprocessor)

- **What is PHP?**
    - Language developed by Rasmus Lerdorf from the Apache Group
    - Its primary use is for server-side scripting
        - Ex: To process HTML forms
        - Ex: To dynamically generate HTML
    - PHP scripts are often embedded within HTML documents
        - The server processes the HTML document, executing the PHP segments and substituting the output within the HTML document
    - The modified document is then sent to the client
    - As mentioned previously, **the client never sees the PHP code**
        - The only reason the client even knows PHP is involved is due to the file extension → .php
            - But even this is not required if the server is configured correctly
- **PHP is a HUGE language**
    - It is a fully functional language
    - It has an incredible amount of built-in features
        - Form processing
        - Output / generate various types of data (not just text)
        - Database access
            - Allows for various DBs and DB formats
        - Object-oriented features
            - Somewhat of a loose hybrid of C++ and Java
            - We will look at PHP OOP later
- **We will look at only a TINY part of PHP**
    - There are also many tools that are already pre-written in / for PHP

    - If you are building a substantial project you may want to use some of these
        - Ex: http://pear.php.net/
        - Also see “links” on php.net home page
    - There are also content management systems written in PHP
        - Ex: http://drupal.org/
        - Ex: http://wordpress.org/

- **PHP Program Structure**
    - PHP, as with many scripting languages, does not have nearly the same structural requirements as a language like Java
    - A script can be just a few lines of code or a very large, structured program with classes and objects
        - The complexity depends on the task at hand
    - However, there are some guidelines for incorporating PHP scripts into HTML files

    - When a PHP file is requested, the PHP interpreter parses the entire file
        - Any content within PHP delimiter tags is interpreted, and the output substituted
        - Any other content (i.e. not within PHP delimiter tags) is simply passed on unchanged
        - This allows us to easily mix PHP and other content (ex: HTML)
        - See:
            - http://us3.php.net/manual/en/language.basic-syntax.phptags.php
            - http://us3.php.net/manual/en/language.basic-syntax.phpmode.php

- **Consider the following PHP file**

```
<!DOCTYPE html>
<html>

 <head>
  <title>Simple PHP Example</title>
 </head>
 <body>
   <?php echo "<p><h1>Output</h1>";
        echo "<h2>Output</h2>";
        echo "<h3>Output</h3></p>";
   ?>
   <script language="PHP">
        echo "\n<b>More PHP Output</b>\n";
        echo "New line in source but not rendered";
        echo "<br/>";
        echo "New line rendered but not in source";
```

```
    </script>
 </body>
</html>
```

- **Now consider the resulting HTML**

```
<!DOCTYPE html>
<html>
 <head>
  <title>Simple PHP Example</title>
 </head>
 <body>
    <p><h1>Output</h1><h2>Output</h2><h3>Output</h3></p>
<b>More PHP Output</b>
New line in source but not rendered<br/>New line rendered but not in source </body>
</html>
```

    - How will it look in the browser?
        - Look at it in the browser!

- If we prefer to separate a PHP code segment from the rest of our script, we can write it in another file and include it
    - Sometimes it is easier if we "separate" the PHP code from the straight html
        - We also may be using several different files, esp. if we are using classes
    - But we must tag it even if it is already within a PHP tagged segment
        - Included files are not interpreted by default
            - Don’t necessarily have to be PHP
            - If we want PHP, include PHP tags within the included file
        - See http://us3.php.net/manual/en/function.include.php

- **Example**

```
<?php
$title = "My First Script";
$greeting = "Welcome to my first script.";
?>
<html>
<head>
<title><?php echo($title) ?></title>
</head>
<body>
<h1><?php echo($title) ?></h1>
<p><?php echo($greeting) ?></p>
</body>
</html>
```

- Line 1: <?php
  A PHP scripting block starts with **<?php** and ends with **?>**
- Line 2: $title = "My First Script";
- Line 3: $greeting = "Welcome to my first script.”;
  PHP variables start with a special character $.
- Line 4: ?>
  This line is used to close the PHP tag
- Line 5: <html>
- Line 6: <head>
  Basic HTML tags.
- Line 7: <title><?php echo($title) ?></title>
  Displays the string “My First Script “ in the title. echo is a basic output statement in PHP
- Line 8: </head>
- Line 9: <body>
  Basic HTML tags
- Line 10: <h1><?php echo($title) ?></h1>
- Line 11: <p><?php echo($greeting) ?></p>
  The above lines show how echo is used in different tags
- Line 12: </body> and Line 13: </html> close the body and html tags respectively.

See more on: http://www.w3schools.com/php/default.asp

ASP

Microsoft® Active Server Pages (ASP) is a server-side scripting technology that can be used to create dynamic and interactive Web applications and was introduced in 1998 as Microsoft's first server side scripting engine.. An ASP page is an HTML page that contains server-side scripts that are processed by the Web server before being sent to the user's browser. You can combine ASP with Extensible Markup Language (XML), Component Object Model (COM), and Hypertext Markup Language (HTML) to create powerful interactive Web sites.

Server-side scripts run when a browser requests an .asp file from the Web server. ASP is called by the Web server, which processes the requested file from top to bottom and executes any script commands. It then formats a standard Web page and sends it to the browser.

It is possible to extend your ASP scripts using COM components and XML. COM extends your scripting capabilities by providing a compact, reusable, and secure means of gaining access to information. You can call components from any script or programming language that supports Automation. XML is a meta-markup language that provides a format to describe structured data by using a set of tags.

What can ASP do?

- Dynamically edit, change, or add any content of a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the results to a browser
- Customize a Web page to make it more useful for individual users
- The advantages of using ASP instead of CGI and Perl, are those of simplicity and speed
- Provide security - since ASP code cannot be viewed from the browser
- Clever ASP programming can minimize the network traffic

See more on: http://www.w3schools.com/asp/default.asp

Example

```
<!DOCTYPE html>
<html>
<body>
<%
response.write("Hello World!")
%>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
<%
response.write("<h2>You can use HTML tags to format the text!</h2>")
%>
<%
response.write("<p style='color:#0000ff'>This text is styled with the style
attribute!</p>")
%>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
<%
dim name
name="Donald Duck"
response.write("My name is: " & name)
%>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
<%
Dim famname(5),i
famname(0) = "Jan Egil"
famname(1) = "Tove"
famname(2) = "Hege"
famname(3) = "Stale"
famname(4) = "Kai Jim"
famname(5) = "Borge"
For i = 0 to 5
    response.write(famname(i) & "<br />")
Next
%>
</body>
</html>
```

ASP.NET

- ASP.NET is a new ASP generation. It is not compatible with Classic ASP, but ASP.NET may include Classic ASP.
- ASP.NET pages are compiled, which makes them faster than Classic ASP.
- ASP.NET has better language support, a large set of user controls, XML-based components, and integrated user authentication.
- ASP.NET pages have the extension .aspx, and are normally written in VB (Visual Basic) or C# (C sharp).
- User controls in ASP.NET can be written in different languages, including C++ and Java.
- When a browser requests an ASP.NET file, the ASP.NET engine reads the file, compiles and executes the scripts in the file, and returns the result to the browser as plain HTML.
- ASP.NET is a development framework for building web pages and web sites with HTML, CSS, JavaScript and server scripting.
- ASP.NET supports three different development models:
    - Web Pages
        - Simplest ASP.NET model
        - Similar to PHP and classic ASP.
        - Built-in templates and helpers for database, video, graphics, social media and more.
    - MVC (Model View Controller)
        - MVC separates web applications into 3 different components:
            - Models for data
                - Parts of the application that handles the logic for the application data.

        - Often model objects retrieve data (and store data) from a database.
    - Views for display
        - Parts of the application that handles the display of the data.
        - Most often the views are created from the model data.
    - Controllers for input
        - Part of the application that handles user interaction.
        - Typically controllers read data from a view, control user input, and send input data to the model.
- Web Forms
    - The traditional ASP.NET event driven development model
    - Web pages with added server controls, server events, and server code.

Example

```
<html>
<body>
    <h1>Hello Web Pages</h1>
    <p>The time is @DateTime.Now</p>
</body>
</html>
```

```
<script  runat="server">
Sub Page_Load
  link1.HRef="http://www.w3schools.com"
  link1.Target="_blank"
  link1.Title="W3Schools"

  link2.HRef="http://www.microsoft.com"
  link2.Target="_blank"
  link2.Title="Microsoft"
End Sub
</script>

<!DOCTYPE html>
<html>
<body>
```

```
<form runat="server">
<a id="link1" runat="server">Visit W3Schools!</a>
<br />
<a id="link2" runat="server">Visit Microsoft!</a>
</form>

</body>
</html>
```

Optputs:Visit W3Schools!

Visit Microsoft!

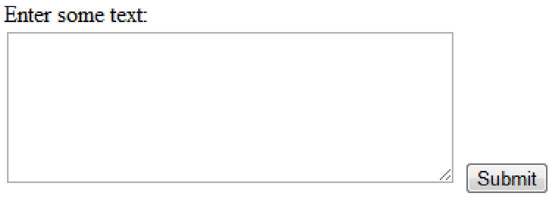
```
<script  runat="server">
Sub submit(sender As Object, e As EventArgs)
  p1.InnerHtml = "<b>You wrote:</b> " & textarea1.Value
End Sub
</script>

<!DOCTYPE html>
<html>
<body>

<form runat="server">
Enter some text:<br />
<textarea id="textarea1" cols="35" rows="6" runat="server" />
<input type="submit" value="Submit" OnServerClick="submit" runat="server" />
<p id="p1" runat="server" />
</form>

</body>
</html>
```

Output:



More on: http://www.w3schools.com/aspnet/default.asp

Asynchronous JavaScript and XML (AJAX)

- This is technique that was coined in Feb. 2005 but that has been used (more or less) for quite a while before that
- AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.
- It allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
- Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.
- It allows the client and the server to communicate without requiring a "hard" submit and page refresh
    - ➢ In particular, the page can be updated without reloading it in its entirety
- Makes the user interface for a Web app. appear more like that of a desktop app
- AJAX is not really a technology by itself
    - ➢ combination of Javascript, XML and some server-side scripting to create the XML
        - server-side scripting could be done in PHP, .NET, Java Servlet or Java Server Page (JSP)
- It is a technique that combines a set of known technologies in order to create faster and more user friendly web pages.
- It is a client side technology.
- Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.
- **Simple idea:**
    - ➢ Issue an HTTP request to the server from Javascript without replacing the current page.
    - ➢ Process the response with Javascript code in the browser.

Purpose of AJAX

- Prevents unnecessary reloading of a page.

- When we submit a form, although most of the page remains the same, whole page is reloaded from the server.
- This causes very long waiting times and waste of bandwidth.
- AJAX aims at loading only the necessary information, and making only the necessary changes on the current page without reloading the whole page.

Simple Processing

- AJAX is based on Javascript, and the main functionality is to access the web server inside the Javascript code.
- We access to the server using special objects; we send data and retrieve data.
- When user initiates an event, a javascript function is called which accesses server using the objects.
- The received information is shown to the user by means of the Javascript's functions.

History of AJAX

- Starts with web pages
- Static web pages
    - Static html page is loaded
    - No interaction with user
- Dynamic web pages
    - Html page is generated dynamically
    - Interaction with user
    - Becomes slower as functionality increases
    - Speed becomes untolerable, so AJAX has been born

Structure of AJAX System

- Client/Server architecture
- XMLHTTP object is used to make request and get response in Client side
- Request can be done via "GET" or "POST" methods
    - "GET": parameters are attached to the url used to connect.
    - "POST": parameters are sent as parameters to a function
- Not many changes in Server side
    - Response is a combination of xml tags

C/S Process

- Most of the time client requires two files
    - Html page: handles GUI and calls a function of JavaScript.
    - JavaScript: handles the business logic of the system
- In JavaScript, a request is sent to client and response is taken from server via XMLHTTP object
- Response of server should be returned in xml file structure
- Only requested data is returned

XMLHTTP object

- The object that is used to connect to the remote server is called XMLHTTP.
- It resembles the Java's URL object that is used to access a specific URL and get the contents.

Creating the Object

- For IE 5.5:
  objXmlHttp=new ActiveXObject("Microsoft.XMLHTTP")
- For Mozilla:
  objXmlHttp=new XMLHttpRequest()

Sending information

- After creating the object, we can send information to the web server and get the answer using this object's functions:
- GET METHOD: *xmlhttp.open("GET", url, true)*
  xmlhttp.send()
- POST METHOD: *xmlhttp.open("POST", url, true)*
  xmlhttp.send("date=11-11-2006&name=Ali")
- Third argument tells that data will be processes asynchronously. When server responds, the "OnReadyStateChange" event handler will be called.

Retrieving Information

- We get the returned value with the property "xmlHttp.responseText".

Pros/Cons

- Advantages:
  - Independent of server technology.
  - Apart from obtaining the XMLHTTP object, all processing is same for all browser types, because Javascript is used.
  - Permits the development of faster and more interactive web applications.
- Disadvantages:
  - The back button problem. People think that when they press back button, they will return to the last change they made, but in AJAX this doesn not hold.
  - Possible network latency problems. People should be given feedback about the processing.
  - Does not run on all browsers.

See more about AJAX on: http://en.wikipedia.org/wiki/AJAX , http://developer.mozilla.org/en/docs/AJAX , See more about AJAX on: http://www.w3schools.com/ajax/default.asp

=====AJAX======

- AJAX centers around the XMLHttpRequest object
- This object allows the client to connect to a server and to respond to the reply once it has become available
- It is a bit tricky to use, and is not consistent across all browsers (esp. older versions)
- However, once you get the hang of it, it can be a very useful tool
- Basic Idea:
  - Programmer creates an instance of an XMLHttpRequest object
  - Several useful attributes and methods are associated with this object:
    - see: http://www.w3.org/TR/XMLHttpRequest/
    - Let's look at a few of them to see how they work together
  - **onreadystatechange**
    - Attribute to which we assign an EventListener (which is a function callback)
    - This will associate the function with the occurrence of the readystatechange event
      - This event fires in several places, throughout the the execution (each time the state changes)
      - We can check the readyState to see what, if anything, we will do in response
  - **open(*method, url, async*)**
      - Where "method" is an HTTP method
      - Where "url" is the location of the server
      - Where "async" is a boolean to determine if the transfer is to be done asynchronously or not
    - Method to switch the object to the open state – i.e. get ready to send data to the server
  - **send(data)**
      - Where "data" is a the information to be sent to the server
      - Can be formatted in various ways, with different encodings
      - Ex: var=value pair query string (like what you see in the URL of a form submitted via GET)
    - Sends the data to the server, where (maybe) a script may run and the response is sent back
  - **readyState**
    - Attribute that stores the current state of the object
    - Changes throughout the execution:
      - 0 → uninitialized
      - 1 → loading
      - 2 → loaded
      - 3 → interactive
      - 4 → complete

- **status**
    - – Did everything work correctly?
        - » 200 – yes it did
        - » 404 – Not found
        - » 500 – internal server error
        - » For more codes, see http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

- **responseType**
    - – What type of data did the server send back to the client?
- **response, responseText, responseXML**
    - – Get the data that was returned
    - – We can parse this to utilize the data to update our page

- **Big Picture:**
    - The XMLHttpRequest object enables us to interact with the server "behind the scenes" and update our web page accordingly
    - Server can still execute scripts as before (ex: PHP), but now it will send updates to the CURRENT page rather than an entirely new page

Example:

```
xhr = new XMLHttpRequest();
xhr.onreadystatechange = xhrHandler;
xhr.open("POST", url);
xhr.send(postData);
...
function xhrHandler() {
  if (this.readyState != 4) {
     return;
  }
  if (this.status != 200) {
     // Handle error ...
     return;
  }
  ...
  var text = this.responseText;
  var document = this.responseXML;
}
```

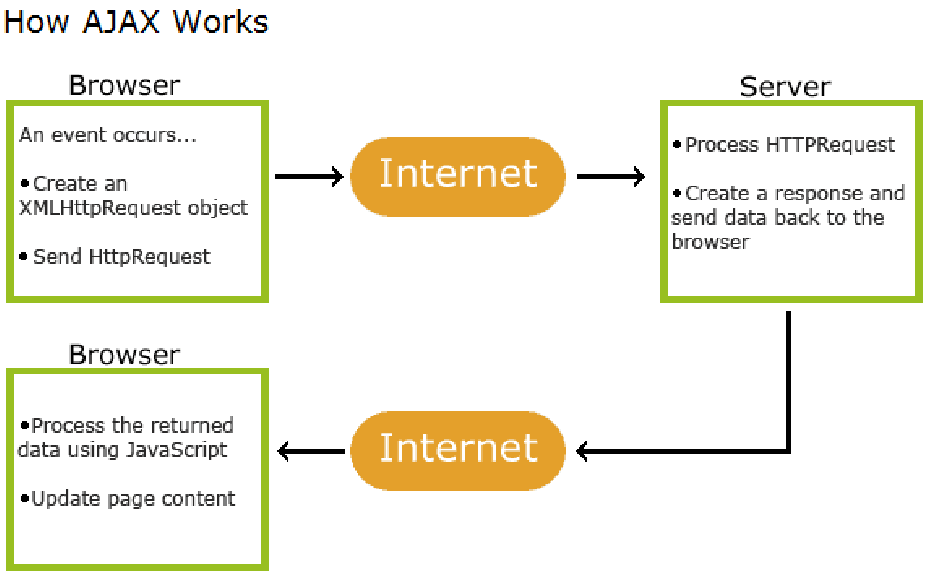
```
<!DOCTYPE html>
<html>
```

```
<head>
<script type="text/javascript">
function loadXMLDoc()
{
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
}
</script>
</head>
<body>

<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>

</body>
</html>
```

- Either GET or POST request (but usually POST).
- Arbitrary request message (typically name-value pairs just like other POSTs).
- Event-based: sending code returns immediately after calling send.
- Event handler gets called at various stages in the processing of the request (usually only care about state 4: done).
- Response available as either raw text or an XML document.
- Can set request headers and read response headers.


How AJAX Works
Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

Note:- AJAX applications are browser- and platform-independent!

Google Suggest
AJAX was made popular in 2005 by Google, with Google Suggest.

Google Suggest is using AJAX to create a very dynamic web interface: When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a list of suggestions.

Browser as a rendering engine

A **web browser engine**, (sometimes called **layout engine** or **rendering engine**), is a software component that takes marked up content (such as HTML, XML, image files, etc.) and formatting information (such as CSS, XSL, etc.) and displays the formatted content on the screen. It "paints" on the content area of a window, which is displayed on a monitor or a printer. A web browser engine is typically embedded in web browsers, e-mail clients, on-line help systems or other applications that require the displaying (and editing) of web content.

Rendering is formatting and presenting information to human readers.

Software that renders HTML pages (Web pages) turns the HTML layout tags in the page into the appropriate commands for the operating system, which causes the formation of the text characters and images for screen and printer. Also called a "layout engine," a rendering engine is used by a Web browser to render HTML pages, by mail programs that render HTML e-mail messages, as well as any other application that needs to render Web page content. For example, Trident is the rendering engine for Microsoft's Internet Explorer, and Gecko is the engine in Firefox. Trident and Gecko are also incorporated into other browsers and applications. Following is a sampling of browsers and rendering engines.

More about how browsers work: http://taligarsiel.com/Projects/howbrowserswork1.htm

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

Webpage Development Tools

- Allows users to develop WebPages without knowledge of HTML
- Web authoring software generate the markup languages in the background while the novice user can concentrate on the layout of the page

Some development tools….

- Netscape composer
- Microsoft Frontage
- Macromedia Dreamweaver

- Netscape composer
  - Tool that is used to create HTML based documents
  - Uses fonts, styles, paragraphs, and lists, and includes an integrated spelling checker
- Microsoft FrontPage
  - A product of Microsoft that helps in designing, building and maintaining websites
  - A user-friendly and relatively powerful WYSIWYG webpage editor
- Macromedia Dreamweaver
  - Fully-featured professional web development tool
  - HTML code editing, advanced table editing, site management tools, built in FTP client, support for animation, DHTML and third-party tags, including ASP, Apache, Cold Fusion, Tango, and many others