**KENYA TECHNICAL TRAINERS COLLEGE**
**COMPUTER STUDIES DEPARTMENT**

**KTTC/TDS/CS/F01**

## CS3104 – Web Technology Programming (Practical)

**Unit Description:**
This unit specifies competencies required to design a website. It involves gathering data required, determining website design tool, developing a functional website and host a website developed.

**Summary of Learning Outcomes**
By the end of this course, the learners should be able to:

(a) Gather data required to develop a functional website.
(b) Determine website design tools
(c) Develop a functional website
(d) Host website developed.

| WEEK | LEARNING OUTCOME | CONTENT | NO. OF HOURS |
|---|---|---|---|
| 1 | Gather data required to develop a website. | • Meaning of terms<br>• Importance of website<br>• Types of website<br>• Website requirements<br>• Web programming languages<br>Exercises on website requirements analysis. | 3 |
| 2 | Determine the website design tools | • Types of website authoring tools<br>• Criteria of choosing website authoring tools<br>• Installation and configuration of website authoring tools<br>• Use of website authoring tools<br>Exercises on identification and installation of authoring software tools. | 3 |
| 3-4 | Develop a functional Static website | • Development of static web pages is done using Hyper Text Markup Language (HTML)<br>o Formatting, hyperlink, tables, frames, forms and lists<br>• Standardization of formatting a website by use of Cascaded Style Sheets (CSS)<br>o Embedded, in-page, external CSS.<br>Develop a 3-page static website using HTML and CSS. (default, services, contact) | 6 |
| 5-6 | Develop a functional Dynamic Website 1 | • Adding Client-Side interactivity to the website using JavaScript<br>o JS statements, variables, operators, data types, functions, objects, events, strings, numbers and arrays. | 6 |

| WEEK | LEARNING OUTCOME | CONTENT | NO. OF HOURS |
|---|---|---|---|
| | | • Add interactivity to the website using JS. | |
| 7 - 8 | Develop a functional Dynamic Website 2 | • Adding Server-Side Interactivity to the website using Active Server Pages (ASP) and Internet Information Server (IIS)<br>○ Introduction to Active Server Pages (ASP)<br>○ Setting up IIS to Work in your Machine<br>○ ASP Basics<br>○ ASP Data Types and Variables<br>○ Control Structures<br>○ Control Structures<br>○ Form Input Data Processing<br>○ ASP procedures | 6 |
| | | Add interactivity to the website using ASP and IIS. | |
| 9 | Develop a functional Dynamic Website 3 | • Adding Database interactivity to the website using MS Access<br>○ Scope of variables<br>○ Redirecting the browser<br>○ Cookies and sessions<br>○ Database connectivity | 3 |
| | | Link database to a database to the website using ASP, Access and IIS. (Add registration form to add new members to the database) | |
| 10 | Host Website Developed | • Website hosting process<br>• Factors to consider when selecting a host<br>• Legal and regulatory requirements<br>• Domain name<br>• Uploading website<br>• Security measure. | 3 |
| | | Host the website. | |
| | | **TOTAL HOURS** | **30** |

**Resources:**

There is no required text for this course. Class handouts and a number of online resources, including tutorials on http://www.w3schools.com/ will be used.

**Suggested Methods of delivery**

- Presentations and practical demonstrations by trainer
- Guided learner activities
- Research project assignments
- Supervised activities and project in a workshop

The delivery may also be supplemented and enhanced by the following, if opportunity allows:

- Visit expert worker from ICT sector

- Industrial visits

**Mode of assignment:**

**Formative Assessments**

a) Class Exercises
b) Online CATs
c) Assignments
d) Observation
e) Oral

**Summative Assessment**

f) Final written examination

## INTRODUCTION

### Terms

- Internet: Wide area network connecting computers and other electronic devices globally. Also called the Net.
- Internet Service Provider (ISP): The ISP is commercial organization that provides internet connections, along with a set of support services for a fee. Bandwidth and cost are two considerations when deciding which commercial ISP service to use. Bandwidth refers to the amount of data that can be transferred through a communications medium in a fixed amount of time.
- Web browsers are software programs that allow users to access the web content.
- URL (Uniform Resource Locator) specifies the address (i.e., location) of the web page displayed in the browser window. Each web page on the Internet is associated with a unique URL. URLs usually begin with http://, which stands for Hypertext Transfer Protocol (HTTP), the standard protocol (or set of communication rules) for transferring web documents over the Internet. HTTP is a data communication protocol used to establish communication between client and server.
- Viewing Source Code, or the original code written to create the web page you are viewing. Source code allows the viewer to understand how the programmer created the page.
- Search Engines
- The File Transfer Protocol (FTP) is a set of rules used to transfer data, especially large files, over the Internet. An FTP site's URL begins with ftp:// rather than http://, and can also be accessed either with the web browser or software that supports FTP. Transferring a file from the local machine to another location on the Internet is called uploading and can be accomplished using the FTP protocol
- A **website** is a collection of related web pages containing images, videos or other digital assets. A website is hosted on at least one web server, accessible via a network such as the Internet or a private local area network through an Internet address also called **Uniform Resource Locator** (URL).
- **A Web Page:** a document, written in plain text with formatting instructions of Hypertext Markup Language (HTML, XHTML). A web page may incorporate elements from other websites with suitable markup anchors. Web pages are accessed and transported with the Hypertext Transfer Protocol (HTTP)
- **A Web Server:** A website is hosted on a computer system known as a web server. It retrieves and delivers the web pages in response to requests from the website users.

The basic objective of a web server is to store, process and deliver web pages to the users. This intercommunication is done using HTTP Protocol. A web server also supports SMTP and FTP protocols for Emailing and for file transfer and storage.
Apache Web Server, Microsoft's Internet Information Server (IIS), NGINX, Apache Tomcat, are examples of commonly used web server software

- Technologies such as HyperText Markup Language (HTML), JavaScript, CSS and Extensible Markup Language (XML) are used to build the portions of web-based applications that reside on the client side (i.e., the portions of applications that typically run on web browsers such as Firefox or Microsoft's Internet Explorer).
- Technologies such as web servers, databases, ASP, PHP, Ajax, Ruby on Rails and Java are used to build the server side of web-based applications. These parts of applications typically run on "heavy-duty" computer systems on which organizations' business-critical websites reside.
- **A Static Website**
- Static websites are ones that are fixed and display the same content for every user, usually written exclusively in HTML and CSS.
- The website can look beautiful with multimedia elements and videos but the page's source code won't change, no matter what actions a user takes on it.
- Visitors are not able to control what information they receive via a static website
- The only way a user can interact with a static page is by clicking hyperlinks and filling Forms.
- Static website are good for those who want to build informational websites like company's brochure site.
- They are edited using three broad categories of software:
    - ✓ Text editors, such as (Notepad, Sublime Text, Atom, TextEdit, Visual Studio Code), where content are manipulated directly within the editor program. With text based editors, you can't see live preview of the site.
    - ✓ WYSIWYG-What you see is what you get. The editors that provide and editing interface that shows how the code looks on a working web page.
        - WYSIWYG offline editors, such as Microsoft FrontPage and Adobe Dreamweaver, with which the site is edited using a GUI interface and the final HTML markup is generated automatically by the editor software
        - WYSIWYG online editors which create media rich online presentation like web pages, widgets, intro, blogs, and other documents.

✓ Template-based editors, such as Rapidweaver and iWeb,

HTML Editors have basic features like syntax highlighting, inserting common HTML elements, auto completion and intelliSense (Smart Completions based on your types, functions and modules)

- **Dynamic Website**
  A dynamic website is one that can display different content and provide user interaction, by making use of advanced programming and databases in addition to HTML.
- Compared to static websites, which are purely informational, a dynamic website is more interactive and functional.
- Dynamic websites rely on both client-side and server-side scripting languages e.g. JavaScript, PHP or ASP. Client side scripting refers to code that is executed by the browser, usually with JavaScript. Server-side Scripting refers to code that is executed by the server (before the content is sent to the user's browser).
- A special editor such as an Integrated Development Environment (IDE) is required to build dynamic websites, along with strong technical skills in server-side programming language.
- Dynamic websites are easier to maintain, they encourage efficient data management, and you can expand them with extra functionality in future. (The downside is that they take longer to build and the initial costs are higher)
- A network protocol/communication protocol is a set of rules governing the communication and transfer of data between two nodes on the network.
    - A protocol specifies the format and meaning of messages exchanged between computers across a network.
    - Format is sometimes called syntax. Meaning is sometimes called semantics
    - Protocols are implemented by a protocol software framework that is part of OS e.g. OSI and TCP/IP

## INTERNET APPLICATIONS

An *Internet application* does something for end users. It is generally not concerned with how data is actually transmitted between the hosts. Here are some distributed applications that require well-defined application level protocols:

- Sending and receiving email
- Searching and browsing information archives
- Copying files between computers
- Conducting financial transactions
- Navigating (in your car, smart scooter, smart bike, or other)
- Playing interactive games
- Video and music streaming
- Chat or voice communication (direct messaging, video conferencing)

In addition, there are a number of *network services* such as:

- Name servers
- Configuration servers
- Mail gateways, transfer agents, relays
- File and print servers

**All Internet applications work over the exact same transport layers**. The Internet says nothing about how these application should work. It provides IP and TCP & UDP and that's it. You can build *anything* on top of those.

- Applications pretty much just need to know:
  i.) the *IP address* of the other party (what *host* the other party is running on—a network layer concept), and
  ii.) The *port number* of the application running at the other end (because the other machine might be running multiple services—a transport layer concept).

  It passes those two pieces of information to the transport layer to make the communication happen.

| APPLICATION LAYER (HTTP, FTP, SMTP, ...) |
|---|
| TRANSPORT LAYER (TCP, UDP, ...) |
| NETWORK LAYER (IP) |
| LINK LAYER (Ethernet, Wifi, ...) |

- IP Addresses for Hosts
  A host address will be 32 bits for IPv4 and 128 bits for IPv6.
- Port Numbers
  Once packets get to the right machine, they have to get to the right program running on that machine. The abstraction here is the port number. Port numbers are in the range 0..65535.
  On the Internet, port numbers are partitioned as follows:
  i.) *Ports*, assigned by the IETF Review or IESG Approval procedures described in RFC 8126.
  ii.) 1024...49151 are the *User Ports*, assigned by IANA using the IETF Review process, the "IESG Approval" process, or the "Expert Review" process, as per RFC 6335.
  iii.) 49152...65535 are the *Dynamic Ports*, which are unrestricted, do with them whatever you want.
- Sockets

## WEB

A system of Internet Servers that support specially formatted documents in a markup language called HTML (HyperText Markup Language) that supports links to other documents, as well as graphics, audio, and video files.

This means you can jump from one document to another simply by clicking on hot spots.

The Web uses HTTP protocol to transmit data and share information. When you type a web address into your browser and hit enter:

- The browser goes to the DNS Server, and finds the real address of the server that the website lives on.
- The browser sends HTTP message to the server, asking it to send a copy of the website to the client. This message and all other data sent between the client and server, is sent across the internet using TCP/IP.
- If the server approves clients request, the server sends the client "200 OK" message and then starts sending the websites files to the browser as a series of packets
- The browser assembles the packets into a complete web age and displays it to you.

The Web is just one of the ways that information is shared over the Internet; others include email, instant messaging and File Transfer Protocol (FTP)

**QUALITIES OF AN EFFECTIVE WEBSITE**

A successful website meets clearly identified goals and provides compelling content that draws the audience to the site again and again. In addition, it is easy to navigate and is attractively designed. The most effective website reflects best practices across all of these elements:

**a) Appearance;** A site must be visually appealing, polished and professional. Your website may be the first, and only, impression a potential customer receives of who you are. An attractive site is far more likely to generate a positive impression and keep visitors on your site once they arrive. Guidelines:

- **Good use of color:** an appropriate color scheme will contain 2 or 3 primary colors that blend well and create a proper mood or tone for the organization. Don't overdo the color, as it can distract the written content.
- **Text that is easily read:** The most easily read combination is black text on a white background, but many other color combinations are acceptable if the contrast is within an
  appropriate range. Use fonts that are easy to read and are found on most of today's computer
  systems. Depending on your audience. Keep font size for paragraph text between 10 and 12 pts.
- **Meaningful graphics:** Graphics are important, as they lend visual variety and appeal to an
  otherwise boring page of text. However, don't over-use them, and make sure that they add meaning or context to your written content. Don't overload any one page with more than 3 or 4 images. **Quality photography:** A simple way to increase visual appeal is to use high quality photography. High quality product images are especially important for online retailers.
- **Simplicity:** Keep it simple and allow for adequate white space. Uncluttered layouts allow
  viewers to focus on your message. Don't overload your site with overly complex design, animation, or other effects just to impress your viewers.

**b) Content:** Along with style, your site must have substance. Remember that the audience is looking for information that will help them make a decision, so the website should be informative and relevant. Use this opportunity to increase visitor confidence in your organization's knowledge and competence. Guidelines:

- **Short and organized copy**: Clearly label topics and break your text up into small paragraphs. Don't bore your visitors with visually overwhelming text. You've got less than 10 seconds to hook your visitors, so grab their attention by being clear, concise and compelling.
- **Update your content regularly**: No one likes to read the same thing over and over again. Dead or static content will not bring visitors back to your site!
- **Speak to your visitors**: Use the word *you* as much as possible. Minimize the use of *I, we* and *us.*
- **Consider a pro**: Unless you're an especially good writer, consider using a professional to write or edit your text content.

**c) Functionality:** Every component of your site should work quickly and correctly. Broken or poorly constructed components will leave your visitors frustrated and disillusioned with your organization. Across the spectrum, everything should work as expected, including hyperlinks, contact forms, site search, event registration, and so on. Error-free copy: Remember the exposure your website will get. Double-check your facts and figures, as you don't know who may be quoting you tomorrow. Nor do you want to be recognized or remembered for typos, incorrect grammar and punctuation, or misspellings. Spelling mistakes and bad grammar are as unforgivable on a website as they are in other organization materials.

**d) Usability;** A critical, but often overlooked component of a successful website is its degree of usability. Your site must be easy to read, navigate, and understandable. Some key usability elements include:

- **Simplicity:** The best way to keep visitors glued to your site is through valuable content, good organization and attractive design. Keep your site simple and well organized.
- **Fast-loading pages:** A page should load in 20 seconds or less via dial-up; at more than that, you'll lose more than half of your potential visitors.
- **Minimal scroll:** This is particularly important on the first page. Create links from the main page to read more about a particular topic. Even the Search Engines will reward you for this behaviour.
- **Consistent layout:** Site layout is extremely important for usability. Use a consistent layout and repeat certain elements throughout the site.
- **Prominent, logical navigation:** Place your menu items at the top of your site, or above the fold on either side. Limit your menu items to 10 or fewer. Remember, your visitors are in a hurry -- don't make them hunt for information.
- **Descriptive link text:** Usability testing shows that long link text makes it much easier for visitors to find their way around a site. Long, descriptive link text is favored by Search Engines, too. Back links are important to give users a sense of direction and to keep them from feeling lost. Use a site map, and breadcrumbs, if necessary.
- **Cross-platform/browser compatibility:** Different browsers often have different rules for displaying content. At a minimum, you should test your site in the latest versions of Internet Explorer, as well as Firefox and Safari.
- **Screen Resolution:** Screen resolution for the typical computer monitor continues to increase. Today, the average web surfer uses a resolution of 1024 x 768 pixels. However, you need to make sure that what looks good at this setting will also work nicely for other resolutions.

**e) Search Engine Optimized (SEO):** Some of the search engine optimization rules includes:

- Include plenty of written content in HTML format. Don't use Flash, JavaScript or image-only objects for your navigational items.
- offering quality content, using proper metadata and effective keywords, and having inbound links from relevant high-quality pages
- Use important keywords frequently and appropriately.
- Minimize the use of tables and use Cascading Style Sheets for layout and positioning; keep
  your HTML code clutter-free.
- Leverage your links -- make them descriptive and use your keywords in the link text

**Web Components:**

Browser, Web Server, Hypermedia (Text, Images, Selectable pointers to other pages) Links, Document representation (HTML), Transfer protocol, etc

**WEB PROGRAMMING**

- Web Programming. Writing the necessary source code to create a website. Web development refers to building, creating and maintaining websites. It includes aspects such as
    - Web design
    - Web publishing\web programming and
    - Database management
  - A web Designer only designs website interfaces using HTML, CSS, etc
  - A web Developer maybe involved in designing a website, but also write a web script in languages such as PHP and ASP. Web Developer may help maintain and update a database used by dynamic website.
- Approaches to Web Programming
  - Server-Side Programming
  - Client-Side Programming
- Web Programming Languages
  - Static technologies like HTML, CSS, JavaScript, XML
  - Perl, PHP, Python, Ruby on Rails, Script

  Two Divisions of Web Development:

  - Front End Development (Client-Side Development). Constructing what a user sees when they load a web application-the content, design and how you interact with it. This is done using HTML, CSS and JavaScript codes
  - Back-End Development (Server-side Development) controls what goes on behind the scenes of a web application. A backend often uses a database to generate a front-end. The Scripts are written using coding languages and frameworks like PHP, Ruby on Rails, ASP.NET, Perl, Java, Node.js, Python.
- Web Programming Interfaces
  - Common Client Interface (CCI)

Inside a Browser. The main Controller:

i) Receives input from user and

ii) Invokes client and interpreter.

- Clients are built in the browser and can be HTTP, FTP, Email, etc. the client uses network to fetch items.
- Interpreter is used to display items.

- Common Gateway Interface (CGI)
  A standard way for a web server to pass a Web user's request to an application program and to receive data back to forward to the user

- Criteria for Choosing a Web Programming Language
  - Efficiency of the language itself;
  - Available platforms and frameworks;
  - Community support;
  - How difficult the language is to learn.

# HTML Coding
# Introduction

- HTML stands for Hyper Text Markup Language
- HTML is used to design web pages using markup language
- HTML describes the structure of a Web page
- HTML is platform independent
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content

.

```
<!DOCTYPE html>
<html>
   <head>
       <title>Page Title</title>
</head>
<body>
   <h1>This is a Heading</h1>
   <p>This is a paragraph.</p>
   <hr>
</body>
</html>
```

**Note:**

- There are four tags every HTML document should have. These tags define what type of document it is, and the major sections. These tags are `<HTML>`, `<HEAD>`, `<TITLE>`, and `<BODY>`.

## Explanation

The *<!DOCTYPE html>* declaration tells version of HTML

The *<html> </html>* Element

- The root element of an HTML page. Defines an HTML document
- All other HTML elements are encapsulated within the <html> element
- The tag lets the browser know it's a HTML web page. It tells the browser that the file contains HTML coded information. The file extension .html indicates that the file is a HTML document.

The *<head> </head>* element

- Contains Meta information about the HTML page. (Meta- data about data)
- Meta Data typically define the document *title, character set, styles, scripts, etc.*
- Statements (or tags) that give information to a person visiting your website, or information such as those needed for a Search Engine

- Provides information about the document. You do not see this information displayed on the browser screen. It can be seen by choosing View Page Source
- The HTML Elements that can be used inside the head element are *<title>*, *<base>, <link>, <style>, <script>, <meta>* and *<noscript>* element.
- The `<title> </title>` element is required.
  - ✓ The element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab).
  - ✓ Also displays a title for the page in search-engine results. Search engine web crawlers pay particular attention to the words used in the title.
  - ✓ It provides a title for the page when it is added to favorites or book marked.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>A simple HTML document</title>
</head>
<body>
    <p>Hello World!</p>
</body>
</html>
```

- The `<base> </base>` element

  Used to define a base URL for all relative links contained in the document.

  Set the base URL once at the top of your page, and then all subsequent relative links will use that as a starting point.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Defining a base URL</title>
    <base href="https://www.tutorialrepublic.com/">
</head>
<body>
    <p><a href="html-tutorial/html-head.php">HTML
    Head</a>.</p>
</body>
</html>
```

- The *<link> </link>* element defines the relationship between the current document and the external documents or resources. A common use of link element is to link to external style sheets.

```
<head>
    <title>Linking Style Sheets</title>
    <link rel="stylesheet" href="style.css">
</head>
```

An HTML document's <head> element may contain any number of <link> elements. The <link> element has attributes, but no contents.

- The *<style> </style>* element is used to define embedded style information for an HTML document. The style rules inside the <style> element specify how HTML elements render in a browser.

```
<head>
    <title>Embedding Style Sheets</title>
    <style>
        body { background-color: YellowGreen; }
        h1 { color: red; }
        p { color: green; }
    </style>
</head>
```

- The *<meta>  </meta>* element provides metadata about the HTML document. Metadata is a set of data that describes and gives information about other data.
- The *<script> </script>* element is used to define client-side script, such as JavaScript in HTML documents.

```
<head>
    <title>Adding JavaScript</title>
    <script>
        document.write("<h1>Hello World!</h1>")
    </script>
</head>
```

The *<body> </body>* element

- Defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, forms, etc.
- Only the content inside the <body> section will be displayed in a browser.

The *<h1>* element defines a large heading

The *<p>* element defines a paragraph

The above visualizes of an HTML page structure

**View HTML Source Code:**
Right-click in an HTML page and select "View Page Source" (in Chrome, Edge or other browsers). This will open a window containing the HTML source code of the page.

**Inspect an HTML Element:**

Right-click on an element (or a blank area), and choose "Inspect" or "Inspect Element" to see what elements are made up of (you will see both the HTML and the CSS). You can also edit the HTML or CSS on-the-fly in the Elements or Styles panel that opens.

**HTML is Not Case Sensitive**

HTML tags are not case sensitive: <P> means the same as <p>

## What is an HTML Element?

– An HTML element is defined by a start tag, some content, and an end tag:

```
<tagname>Content goes here...</tagname>
```

Tags are the instructions which are being directly embedded in the text of an HTML document

– Some HTML elements have no content (like the <br> element). These elements are called empty elements. Empty elements do not have an end tag!

– Nested Elements. E.g.

```
<!DOCTYPE html>
<html>
   <head>
      <title> Nested Elements Example</title>
   </head>
       <body>
      <h1>This is <i>italic</i> heading</h1>
      <p>This is <u>underlined</u> paragraph</p>
   </body>
</html>
```

– Types of tags:

i.) Paired Tags/ Container Tags. Consists of two tags e.g. `<i> </i>,`

ii.) Unpaired Tags/ Empty Tags/ Singular Tags. Single tags, does not need a companion tag. E.g. `<hr> or <hr />, <br> or <br />`

Also tags can be differentiated based on the purpose they are used. E.g. Formatting tags, Page structure tags, control tags, Character Tags

## HTML Attributes

– All HTML elements can have attributes
– Attributes provide additional information about elements
– Attributes are always specified in the start tag
– Attributes usually come in name/value pairs like:

**`name="value"`**

e.g HTML `title` Attribute

```
<p title="Free tutorials">The Easiest One</p>
```

- *The title attribute specifies extra information about an element.*
- *The information is most often shown as a tooltip text when the mouse moves over the element.*
- *The title attribute can be used on any HTML element*

– The **global attributes** are attributes that can be used with all HTML elements. E.g. ***class, style, lang, tabindex, title, id, hidden***

– **Event attributes** have the ability to let events trigger actions in a browser, like starting a JavaScript when a user clicks on an element. E.g. ononline,

onresize, onunload, onerror, onload, onstorage, onkeyup, onclick, ondrag, oncopy, onplay

## The HTML <body> tag

- defines the main content of the HTML document or the section of the HTML document that will be directly visible on your web page.
- In addition to the Global Attributes, some of the attributes that are specific to the **<body>** tag are:

`Bgcolor:` or **B**ack**G**round, of your page. You need to put the colour in as a **HEX code**, like the rest of these colours e.g

`bgcolor="#FFFFFF"` or `<body bgcolor="yellow">`

`<body text="red" >`: text attribute sets the text color of all text contained within the body tags. Sets the color of your text which you can later modify through additional tags inside of the body

`<body link="white" vlink="black" >:` Specifies base colors for visited or unvisited links.

`<body topmargin="50" leftmargin="50">:` sets pixel value margins for the left, right, top, or bottom of your website

`<body background= "myimage.png">:` Sets Image to be used a background.

`< background= "http://www.mite.com/med/BACKGROUND.gif ">`

*Most of these tags are deprecated in HTML 4.0 thus use CSS (HTML Style attribute)*

### HTML <hr> Tag

- The `<hr>` tag defines a thematic break in an HTML page (e.g. a shift of topic).
- The `<hr>` element is most often displayed as a horizontal rule that is used to separate content (or define a change) in an HTML page.
- `<hr width="50%" >`
- `<hr style="width:50%;">,`
- `<hr style="height:30px;">,`
- `<hr style="height:40px" noshade>,`
- `<hr style="height:200px; border-width:0; color:gray; background-color:gray" >`
- `<hr style="width:50%; margin-left:40;">`

### The HTML <marquee> tag

- Used for scrolling piece of text or image displayed either horizontally across or vertically down your web site page depending on the settings
- `<marquee>This is basic example of marquee</marquee>`
- `<marquee direction = "up">The direction of text will be from bottom to top.</marquee>`

| Attribute | Value | Description |
|---|---|---|
| Behavior | scroll<br>slide<br>alternate | Defines the type of scrolling. |
| Bgcolor | rgb(x,x,x) | *Deprecated* − Defines the direction of scrolling the content. |
| Direction | up<br>down<br>left<br>right | Defines the direction of scrolling the content. |
| Height | pixels or % | Defines the height of marquee. |
| Hspace | pixels | Specifies horizontal space around the marquee. |
| Loop | number | Specifies how many times to loop. The default value is INFINITE, which means that the marquee loops endlessly. |
| Scrolldelay | seconds | Defines how long to delay between each jump. |
| scrollamount | number | Defines how far to jump. |
| Width | pixels or % | Defines the width of marquee. |
| Vspace | pixels | Specifies vertical space around the marquee |

## TAGS THAT AFFECT THE APPEARANCE OF YOUR TEXT

### Header Tags

- Logical tags that make your headings larger (or smaller), they also bold the headings at the same time. There are only six HEADER tags and they range from H1 to H6. Some search engines give words appearing in headers more importance in their index. The headers are often used by these search engines to build an "outline" of the document which appears in the search results.

### Centering Text:

- Centering Text: e.g. `<H2 ALIGN="CENTER">`a heading goes here`</H2>`. The ALIGN attribute can also take, ALIGN="RIGHT" and ALIGN="LEFT".
- You can also use

```
<CENTER>
.
.
all lines or blocks of text will be centered between these two tags
.
.
</CENTER>
```

### Emphasizing Text:

- Emphasizing entails using ITALICS and BOLDFACE.
- `<EM>` is used to emphasize text. It is a **logical tag** and so describes the meaning of the text to be displayed rather than how the text is to be displayed. In most browsers, the meaning is italics. Logical tags allow the browser to render that information in the manner most appropriate for that browser.

- **<STRONG>** is also a logical tag. It is used to strongly emphasize text. <STRONG> is distinct from <EM>. In most browsers, STRONG is identical to boldface.
- Using <B> for Boldface or <I> for Italics. <STRONG> is accepted by all browsers as a way of strongly emphasizing text which in most cases is boldface. Instead of the <STRONG> and </STRONG> tags for strongly emphasizing text, you will sometimes see <B> and </B> for bolding text.

  While <STRONG> is a "**Logical Style Command**", <B> is a "**Physical Style Command**". A physical style command cannot be rendered differently.

  Therefore if a browser does not accept the <B> tag, then your text will simply not be bolded as the browser has no alternative way to display them on the screen. The same reasoning is also be applied for using the tag <I> to print in italics instead of the <EM> tag. Now if for some reason you want to ensure only italics or boldface and nothing else, then use the italic font <I> or the boldface font <B>.

```
<Html>
   <Head>
        <Title>OUTDOOR LIVING - ONTARIO</Title>
   </Head>
   <Body>
        <H2 Align="Center">OUTDOOR LIVING IN
        ONTARIO</h2>
        <H3>POINTS OF INTEREST</H3>
        <HR>
        <P>Northern Ontario:
        <P>    <strong>Five Mile Lake Provincial Park
        </strong>
        <P>    <EM>1.5 square miles.</EM> Camping,
        fishing (<EM>walleye, northern pike, brook
        trout</EM>), canoeing, self-guided nature
        trails.
        <P>
        <strong><em>Recommended</em></strong>
        <hr>
   </body>
</html>
```

### The <br>, <p> and <pre> tags

– <p> defines a paragraph. A container which marks a block of text as a paragraph in a webpage and the browser leaves a line between two lines. It has align attributes that specifies text alignment, align=”center”, align=”justify”

- <br> defines a line break. An empty tag used to break a line and display proceeding text from the next line, without giving space between two lines.
- e.g.  This is <br> a paragraph <br> with line breaks
- The <pre> tag defines preformatted text. The text inside <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks.
- E.g.

```
<pre>
My Bonnie lies over the Ocean
My Bonnie lies over the sea
Oh, bring back my Bonnie to me.
</pre>
```

**The HTML Quotation and Citation Elements**

**<Blockquote> tag** specifies a section that is quoted from another source. E.g.

```
<p>Here is a quote from WWF's website:</p>

<blockquote cite="http://www.wild.org/index.html">
For 50 years, WWF has been protecting the future of
nature. The world's leading conservation organization,
WWF works in 100 countries and is supported by 1.2
million members in the United States and close to 5
million globally.
</blockquote>
```

**The** <dd> element provides the description, definition, or value for the preceding term (<dt>) in a description list (<dl>).

```
<p>Beverages:</p>
<dl>
        <dt>Coffee</dt>
          <dd>Black hot drink</dd>
        <dt>Milk</dt>
         <dd>White cold drink</dd>
</dl>
```

```
<dl> A list of terms and their definitions/descriptions.
   <dt>HTML</dt>
      <dd>Hypertext Markup Language, the language for authoring
      web documents.</dd>
   <dt>PHP</dt>
      <dd>An acronym for "PHP: Hypertext Preprocessor". A server
      scripting language, and a powerful tool for making dynamic
      and interactive Web pages. PHP is an interpreted  widely-
      used, open source scripting language. </dd>
   <dt>CSS</dt>
```

```
    <dd>Cascading Style Sheets, the language for defining the
    styles and presentation of an HTML document. CSS describes
    how HTML elements should be displayed. There are three ways
    you can use to implement CSS: internal, external, and inline
    styles</dd>
<dt> JavaScript</dt>
    <dd>JavaScript ("JS" for short) is a full-fledged dynamic
    programming language that can add interactivity to a
    website. avaScript adds behavior to web pages.  A
    scripting language built-in to most browsers and designed
    to be used with web documents. </dd>
<dt> ASP and ASP.NET</dt>
   <dd>Active Server Pages. ASP is a development framework for
    building web pages. ASP supports many different
    development models:Classic ASP, ASP.NET Web Forms, ASP.NET
    MVC, ASP.NET Web Pages, ASP.NET API, ASP.NET Core. a
    server side Web building scripting engine. </dd>
   </dl>
```

**HTML Definition element** (**<*dfn*>**) is used to indicate the term being defined within the context of a definition phrase or sentence. E.g.

```
<p>A <dfn>validator</dfn> is a program that checks
for syntax errors in code or documents.</p>
```

| | |
|---|---|
| <abbr> | Defines an abbreviation or acronym |
| <address> | Defines contact information for the author/owner of a document |
| <blockquote> | Defines a section that is quoted from another source |
| <cite> | Defines the title of a work |
| <q> | Defines a short inline quotation |

**HTML Subscript and Superscript Tags**

The <sub> tag is used to add a subscript text to the HTML document.

The <sup> tag is used to add a superscript text to the HTML document

```
<p>Testing <sub>subscript text</sub></p>

<p>Testing <sup>superscript text</sup></p>
```

**HTML Symbols:** HTML Special Character Codes

Write their **entity names**. An entity name always starts with an **ampersand (&)** and ends with a **semicolon (;)**:

| Char | Entity | Number | Definition |
|---|---|---|---|
| |   |   | Non-breaking space |
| ¢ | ¢ | ¢ | cent |
| © | © | © | Copyright sign |
| ® | ® | ® | Registered sign |
| ° | ° | &#176 | Degree |
| " | " | " | double quotation mark |

| Char | Entity | Number | Definition |
|---|---|---|---|
| $ | &dollar; | &#36; | Dollar |
| % | &percnt; | &#37; | Percent |
| & | &amp; | &#38; | Ampersand |
| ' | &apos; | &#39; | single quotation mark (apostrophe) |
| * | &ast; | &#42; | Asterisk |
| < | &lt; | &#60; | less than |
| > | &gt; | &#62; | greater than |
| @ | &commat; | &#64; | At symbol |
| ™ | &trade; | &#8482; | Trademark |
| ∑ | &sum; | &#8721; | N-ary Summation |

### The HTML Style Attribute

The HTML `style` attribute is used to add styles to an element, such as color, font, size, and more

Syntax: `<tagname style="property:value;">`

```
<body style="background-color:powderblue;">
     <h1> This is a heading</h1>
     <p> This is a paragraph.</p>
</body>

<h1 style="color:blue;text-align:center">This is a
header</h1>
<p style="color:green;">This is a paragraph.</p>

<h1 style="color:blue;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

<h1 style="font-family:verdana;">This is a heading</h1>
<p style="font-family:courier;">This is a paragraph.</p>

<h1 style="font-size:300%;">This is a heading</h1>
<p style="font-size:160%;">This is a paragraph.</p>

<h1 style="text-align:center;">Centered Heading</h1>
<p style="text-align:center;">Centered
paragraph.</p>
```

### HTML Block and Inline Elements

- There are two display values: block and inline
- A block-level element always starts on a new line and takes up the full width available
- An inline element does not start on a new line and it only takes up as much width as necessary

- The  element is a block-level and is often used as a container for other HTML elements
- The  element is an inline container used to mark up a part of a text, or a part of a document
- A block level element has a top and a bottom margin, whereas an inline element does not.

– The ** element is a block-level element.** Other examples are:

| | | | |
|---|---|---|---|
| <address> | <dt> | <hr> | <section> |
| <article> | <fieldset> | <li> | <table> |
| <aside> | <figcaption> | <main> | <tfoot> |
| <blockquote> | <figure> | <nav> | <ul> |
| <canvas> | <footer> | <noscript> | <video> |
| <dd> | <form> | <ol> | |
|  | <h1>-<h6> | <p> | |
| <dl> | <header> | <pre> | |

– **Examples of inline element:**

– This is a  element inside a paragraph. Hello World

| | | | |
|---|---|---|---|
| <a> | <code> | <object> | <sub> |
| <abbr> | <dfn> | <output> | <sup> |
| <acronym> |  | <q> | <textarea> |
| <b> | <i> | <samp> | <time> |
| <bdo> | <img> | <script> | <tt> |
| <big> | <input> | <select> | <var> |
| <br> | <kbd> | <small> | |
| <button> | <label> |  | |
| <cite> | <map> |  | |

– **Note:** An inline element cannot contain a block-level element!

## CREATING LINKS IN HTML

– HTML links are hyperlinks.
– You can click on a link and jump to another document.
– When you move the mouse over a link, the mouse arrow will turn into a little hand.

**Note:** A link does not have to be text. A link can be an image or any other HTML element!

**1. Links are used to:**

a) Jump from section to section within the same web page (also called Page Jump)

b) Link to a different page within your own website (such as linking this lesson to the next lesson or to my home page). Also Called **Internal Links**

c) Link to another web page or website anywhere in the world. Also Called **External Links**

- There are different ways to provide these links. The three most common ones are:
  (a) Clicking on a word, phrase or sentence
  (b) Clicking on a button
  (c) Clicking on an image (that is, a picture or graphic).

- Building internal / external links is important for a few different reasons:
  a) Internal links are used to create navigation menus that help website visitors navigate our website.
  b) Internal links are used in the text of website content to help website visitors locate related content.
  c) Internal links are also used by search engine web crawlers to locate the pages of a website and to share authority (also known as link juice) with the other pages of a website.
  d) External links may be recommended, required, or just best-practice to provide proper attribution to the source of an idea or a resource.
  e) External links allow us to refer website visitors to useful related content.

**2. HTML Links - Syntax**

The HTML <a> tag defines a hyperlink. It has the following syntax:

```
<a href="url">link text</a>
```

- The most important attribute of the <a> element is the href attribute, which indicates the link's destination.
- The *link text* is the part that will be visible to the reader.
- Clicking on the link text, will send the reader to the specified URL address.

By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

**3. <a>…</a> Attributes of the anchor Element**

| Attribute name | Values | Notes |
|---|---|---|
| hreflang | | Specifies the language of the linked resource. |
| download | | Directs the browser to download the linked resource rather than opening it. |
| target | _blank<br>_parent<br>_self | Specifies the context in which the linked resource will open. |

| | | |
|---|---|---|
| | _top<br>frame name | |
| title | text | Defines the title of a link, which appears to the user as a tooltip. |
| href | url | Specifies the linked document, resource, or location. |

**4. HTML <a> href Attribute**

- Syntax: `<a href="URL">`
- If the href attribute is not present, the `<a>` tag WILL NOT BE A HYPERLINK.
- You can use `href="#top"` or `href="#"` to link to the top of the current page
- Attribute Values

| Value | Description |
|---|---|
| URL | The URL of the link.<br>Possible values:<br>• An absolute URL - points to another web site (like `href="http://www.example.com/default.htm"`)<br>• A relative URL - points to a file within a web site (like `href="default.html"`)<br>• Link to an element with a specified id within the page (like `href="#section2"`)<br>• Other protocols (`like https://, ftp://, mailto:, file:`, etc..)<br>• A script (like `href="javascript:alert('Hello');"`) |

**Examples**

1. How to create a link to W3Schools.com:
```
<a href="https://www.w3schools.com/">Visit W3Schools.com!</a>
```
2. How to use an image as a link:

   To use an image as a link, just put the `<img>` tag inside the <a> tag:
```
<a href="https://www.w3schools.com">
     <img border="0" alt="W3Schools"
     src="logo_w3s.gif" width="100" height="100">
</a>
```

```
<a href="default.asp">
     <img src="smiley.gif" alt="HTML tutorial"
     style="width:42px;height:42px;">
</a>
<!--We can also use an image as an anchor element-->
<a><img src="http://example.com/image-file.png" /></a>
```

3. How to link to an email address:

   Use `mailto:` inside the `href` attribute to create a link that opens the user's email program (to let them send a new email):

```
<a href="mailto:someone@example.com">Send email</a>

<a href="mailto:benesad@yahoomail.com">Send email</a>
```

4. How to link to a phone number:

```
<a href="tel:+254733781901">+2547 333 78 901</a>
```

5. How to link to another section on the same page:

```
<a href="#section2">Go to Section 2</a>
```

6. How to link to a JavaScript:

```
<a href="javascript:alert('Hello World!');">Execute
JavaScript</a>
```

7. Button as a Link

   To use an HTML button as a link, you have to add some JavaScript code.

   JavaScript allows you to specify what happens at certain events, such as a click of a button:

```
     <button onclick=
     "document.location='default.asp'"> HTML Tutorial
     </button>
```

   **Combined**

```
 <p>You can reach Michael at:</p>
 <ul>
   <li><a href="https://example.com">Website</a></li>
   <li><a
 href="mailto:m.bluth@example.com">Email</a></li>
   <li><a href="tel:+123456789">Phone</a></li>
 </ul>
```

## 5. The `target` Attribute

- By default, the linked page will be displayed in the current browser window. To change this, you must specify another target for the link.
- The target attribute specifies where to open the linked document.
- The target attribute can have one of the following values:

`_self` - Default. Opens the document in the same window/tab/ frame as it was clicked

`_blank` - Opens the document in a new window or tab

`_parent` - Opens the document in the parent frame

`_top` - Opens the document in the full body of the window

`framename` Opens the linked document in the named iframe

**Examples**

1. Use target="_blank" to open the linked document in a new browser window or tab:

   ```
   <a href="https://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
   ```

2. Download Example

   ```
   <a href=http://example.com/file.doc
   download="Example_File" > download a file </a>, use the
   <code>href</code> attribute to identify the file to be downloaded, and the
   <code>download</code> attribute to provide a name for the downloaded file
   ```

3. `To tell the browser that a <a href="http://example.com" hreflang="es">a link</a> points to a resource that is in a different langauge, we can use the <code>hreflang</code> attribute.`

## 6. Absolute URLs vs. Relative URLs

- An **absolute URL** (a full web address) in the `href` attribute.
- A local link (a link to a page within the same website) is specified with a **relative URL** (without the "https://www" part):
- **Examples**

  1. Linking to an absolute URL

     ```
     <a href="https://www.mozilla.com"> Mozilla </a>
     ```

     ```
     <h2>Absolute URLs</h2>
     <p><a href="https://www.w3.org/">W3C</a></p>
     <p><a href="https://www.google.com/">Google</a></p>
     ```

  2. Linking to relative URLs

     ```
     <a href="//example.com">Scheme-relative URL</a>
     ```

     ```
     <a href="/en-US/docs/Web/HTML">Origin-relative URL</a>
     ```

     ```
     <a href="./p">Directory-relative URL</a>
     ```

```
<h2>Relative URLs</h2>
<p><a href="html_images.asp">HTML Images</a></p>
<p><a href="/css/default.asp">CSS Tutorial</a></p>
```

## Note:

1. **A Web of Links**
   The World Wide Web is best described as a massive library of hyperlinked documents where anchor elements are used to create bridges between related documents. In this definition, anchor elements occupy their rightful place as the glue that ties the web together and the bridges that allow web users to move from one document to a related document.
2. **The <link> element** is used to define a relationship between an HTML document and an external resource. This element is most commonly used to define the relationship between a document and one or more external CSS stylesheets. Some of the Link Element attributes are href, rel, media, title, type
3. The <a> element, or anchor element, it used to create a hyperlink to another webpage or another location within the same webpage.
4. The <base> element is used to identify a base URL upon which to build all relative URLs that appear on a webpage.

### More on Links

1. **Placing Subject in Emails**

   You can also add the subject, cc, and bcc lines as part of the mailto: link. For example, to add a topic (a subject) in the subject line of your e-mail (the most popular option), the format is:

   ```
   <A HREF="mailto:dmwaura@nysei.go.ke?subject=topic">Please
   send more product information</A>
   ```

   ```
   <HREF="mailto:d.dmwaura@nysei.go.ke?subject=product
   information">Please send more product information</A>
   ```

2. If you **want** to include the e-mail address of the person you also want to receive a carbon copy of the same e-mail (that is, the person you want to be cc'd), the format is:

   ```
   <A HREF="dmwaura@nysei.ac.ke?cc=person@site.com">
   ```

   ```
   <A HREF="mailto:dmwaura@nysei.ac.ke?bcc=person@site.com">
   ```

3. **Placing a Message in the Body of Emails**

   You can also place a personal message in the body of an email using the? body attribute as in:

   ```
   <A HREF="mailto:?body=Don't forget also to send in your
   ```

```
financial statements as soon as possible.">Click here
when ready to send us your email.</A>
```

**4. Links in general**

```
<H4 ALIGN="CENTER">
    <A HREF="#links">| Links in general |</A>
    <A HREF="#page">| Links within a page - page jump |</A>
    <A HREF="another">| Linking to another page in your website|</A>
    <A HREF="world">| Linking to another page anywhere in the world
    |</A>
    <A HREF="button">| Using a link button |</A>
    <A HREF="mail">| MAILTO: (sending an e-mail) |</A>
</H4>
```

(a) This is used in page jumps.

```
<A HREF="#top">enclosed text</A>
```

(b) This statement is used in linking to another page in your directory (that is, to another one of your web pages).

```
<A HREF="index.htm">enclosed text</A>
```

(c) This is used in linking to a website anywhere in the WWW

```
<A HREF="http://www.press.com/perma/">enclosed text</A>
```

**5. Links within a page**

```
<a href ="#linkname"> word</a>
```

*linkname* is the name of the section that you are linking to
The # symbol instructs the browser to look through the HTML document for a named anchor.

A **named** anchor is a hidden reference marker for a particular section of the same page. It is also used to mark a section of another page. e.g.

```
<body>
<a name ="top"></a>
. .
<a href ="top">TOP</a>
</body>
Or
<a href ="#linkname"> word</a>
. .
<A name ="linkname">about some text here</a>
```

**6. Links between sections of different documents**

The HTML code for linking to a named anchor in another local HTML document is as follows:

Suppose you want to link from documentA.html to documentB.html In documentA.html:

```
<a href = "documentB.html#linkname"> Text to activate link</a>
```

In documentB.html:

```
<a name ="linkname">Text that responds to the link</a>
```

7. **Using a Link Button**

Using <form> … </form> Element:

(a)
```
<H3 ALIGN="CENTER">
  <FORM METHOD="GET" ACTION="index.htm">
      <INPUT TYPE="submit" VALUE="Return to Home
      Page">
  </FORM>
 </H3>
```

(b)
```
<H3 ALIGN="CENTER">
 <FORM METHOD="GET" ACTION="index.htm">
    <INPUT TYPE="submit" VALUE="Return to Home Page">
  </FORM>
  <FORM METHOD="GET" ACTION="readcv.htm">
    <INPUT TYPE="submit" VALUE="Read My Curriculum
    Vitae">
  </FORM></H3>
```

**Using <table> … </table> Element:**

```
<TABLE BORDER="0">
  <TR>
    <TD>
      <FORM METHOD="GET" ACTION="index.htm">
       <INPUT TYPE="submit" VALUE="Return to Home
       Page">
      </FORM>
    </TD>
    <TD>
      <FORM METHOD="GET" ACTION="readcv.htm">
       <INPUT TYPE="submit" VALUE="Read My Curriculum
       Vitae">
      </FORM>
    </TD>
  </TR>
</TABLE>
```

# HTML IMAGES

Images include Pictures, Graphics, Icons, Clip Art, Etc. Images can improve the design and the appearance of a web page.

While images can really add beauty to a web page, be careful that you do not overuse them. It takes time for a browser to display an image and so a lot of images can take a lot of time. Also, the bigger the image, the longer it will take to display. Many viewers do not want to wait a long time.

1. **HTML Images Syntax**

   The HTML <img> tag is used to embed an image in a web page.

   Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.

   The <img> tag is empty, it contains attributes only, and does not have a closing tag.

   The <img> tag has two required attributes:

   - src - Specifies the path to the image
   - alt - Specifies an alternate text for the image

   Syntax

   ```
   <img src="url" alt="alternatetext">
   ```

2. **The `src` Attribute**

   – The required `src` attribute specifies the path (URL) to the image.

   – **Note:** When a web page loads; it is the browser, at that moment that gets the image from a web server and inserts it into the page. Therefore, make sure that the image actually stays in the same spot in relation to the web page, otherwise your visitors will get a broken link icon. The broken link icon and the alt text are shown if the browser cannot find the image.

   – Example

   <img src="img_chania.jpg" alt="Flowers in Chania">

3. **The alt Attribute**

   – The required alt attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute).

   – The value of the alt attribute should describe the image:

   Example

   ```
   <img src="img_chania.jpg" alt="Flowers in Chania">
   <img src="wrongname.gif" alt="Flowers in Chania">
   ```

4. **Image Size - Width and Height**

   You can use the `style` attribute to specify the width and height of an image.

   – Example

```
<img src="img_girl.jpg" alt="Girl in a jacket"
style="width:500px;height:600px;">
```

- **Alternatively, you can use the width and height attributes:**

```
<img src="img_girl.jpg" alt="Girl in a jacket"
width="500" height="600">
```

- The width and height attributes always define the width and height of the image in pixels.

```
<body>

   <img src="html5.gif" alt="HTML5 Icon" width="128"
   height="128">

   <img src="html5.gif" alt="HTML5 Icon"
   style="width:128px;height:128px;">

</body>
```

**5. Border and Align Attributes**

- The BORDER attribute is used to place (or eliminate) a border around the image.
- BORDER="0" turns it off border - that is, no border.
- BORDER="0" a border of 10 pixels
- The ALIGN attribute is used in conjunction with placing text around the image. That is, ALIGN controls the alignment of the image with respect to text.

```
<HTML>
  <HEAD>     <TITLE> HALL RENTAL </TITLE> </HEAD>
<BODY>
  <H2 ALIGN="CENTER">ABOUT OUR HALL</H2>
  <HR>
  <H4 ALIGN="LEFT">
    <IMG SRC="whechair.gif" BORDER="0"
     ALIGN="MIDDLE"> We are wheelchair accessible.
  </H4>
  Please call for rates
  <HR>
</BODY>
</HTML>
```

- Change the ALIGN="MIDDLE" to ALIGN="BOTTOM". Change to ALIGN="TOP"
- The values TOP, MIDDLE and BOTTOM specify where any text following the image should be placed. Remember that if more than one line follows after the image, the additional lines will be placed below the image. You can also

force text below such an aligned image by using the BR tag with the CLEAR attribute as in <BR CLEAR>.

6. **HSPACE Attribute**
   - HSPACE and VSPACE indicate the number of pixels that should be left free around the image
   - HSPACE used to create horizontal space around the image
   - VSPACE used to create vertical space around the image

```
<BODY>
   <H2 ALIGN="CENTER">ABOUT OUR HALL</H2>
   <HR>
   <H4 ALIGN="LEFT">
     <IMG SRC="whechair.gif" BORDER="0"
     ALIGN="MIDDLE" WIDTH="60" HEIGHT="60" HSPACE="10"
     ALT="chair"> We are wheelchair accessible.
   </H4>
   Please call for rates
   <HR>
</BODY>
```

7. **Images in another Folder**
   - If you have your images in a sub-folder, you must include the folder name in the src attribute:
   - Example

```
<img src="/images/html5.gif" alt="HTML5 Icon"
style="width:128px;height:128px;">
```

8. **Images on another Server/Website**
   - To point to an image on another server, you must specify an absolute (full) URL in the src attribute:
   - Example

```
<img
src="https://www.w3schools.com/images/w3schools_gree
n.jpg" alt="W3Schools.com">
```

9. **Image as a Link**
   - To use an image as a link, put the <img> tag inside the <a> tag:

```
<a href="default.asp">
   <img src="smiley.gif" alt="HTML tutorial"
   style="width:42px;height:42px;">
</a>
<BODY>
   <H4 ALIGN="CENTER">
   <A HREF="index.htm">
```

```
      <IMG SRC="back.gif" WIDTH="40" HEIGHT="40"
      HSPACE="10" ALIGN="MIDDLE" ALT="home">
    </A>Click on the button to head home
    </H4>
    <HR>
</BODY>
```

**10. Common Image Formats**

– Here are the most common image file types, which are supported in all browsers (Chrome, Edge, Firefox, Safari, Opera):

| Abbreviation | File Format | File Extension |
|---|---|---|
| – APNG | – Animated Portable Network Graphics | – .apng |
| – GIF | – Graphics Interchange Format | – .gif |
| – ICO | – Microsoft Icon | – .ico, .cur |
| – JPEG | – Joint Photographic Expert Group image | – .jpg, .jpeg, .jfif, .pjpeg, .pjp |
| – PNG | – Portable Network Graphics | – .png |
| – SVG | – Scalable Vector Graphics | – .svg |

**11. Images not appearing in Your Browser?**

– If an image is not being displayed in your browser, then the browser simply can't locate it.

– Here are some suggestions to try and fix the situation.

(a) First make sure that the name of the image and the extension matches exactly the name and extension used in the coding.

For example, if the image is named `gardens.jpeg,` then the coding must be:

```
<IMG SRC="gardens.jpeg">.
```

If you accidentally write:

```
<IMG SRC="garden.jpeg">
```
*(name does not match)*
or

```
<IMG SRC="gardens.jpg">
```
*(extension does not match)*

Then the browser cannot display the image because there is not an exact match

(b) Place the images into the same folder as the web page. Often if they are not in the same folder, they do not get displayed. So your best results will occur if the images are in the same folder as the web page

(c) If your images appear in your browser when you are working off-line, but they do not appear in your browser after you have uploaded your web page to the internet (to your host server), then make sure that you have also uploaded each image to your host server. You not only have to upload your web pages to your host server, but you also need to upload each individual image.

## 12. Background Graphics

Using a feature called tiling, a browser takes the image and repeats it across and down to fill your browser window.
<BODY BACKGROUND = “filename.gif”>

We can also use the HTML style attribute and the CSS background-image property:

```
<!DOCTYPE html>
<html>
<head>
<title>Example</title>
    <!-- Styles -->
    <style>
    body {
      background-image: url("/pix/samples/bg1.gif");
      background-position: 50% 50%;
      background-repeat: repeat;
    }
    </style>
</head>
<body>
    <!-- HTML -->
    <h3>Whole Page</h3>
    <p>This example has a background image applied to the
    <code>body</code> element.</p>
</body>
</html>
```

# HTML Lists

– HTML lists allow web developers to group a set of related items in lists.

### 1. Unordered HTML List

– An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

– The list items will be marked with bullets (small black circles) by default:

– Example

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

### 2. Ordered HTML List

– An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

– The list items will be marked with numbers by default:

– Example

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

### 3. HTML Description Lists

– HTML also supports description lists.

– A description list is a list of terms, with a description of each term.

– The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

– Example

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

### 4. HTML List Tags

| Tag | Description |
|---|---|
| <ul> | Defines an unordered list |
| <ol> | Defines an ordered list |
| <li> | Defines a list item |
| <dl> | Defines a description list |

| | |
|---|---|
| <dt> | Defines a term in a description list |
| <dd> | Describes the term in a description list |

## 5. HTML Unordered Lists

- The HTML <ul> tag defines an unordered (bulleted) list. An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.
- The list items will be marked with bullets (small black circles) by default:
- Example

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

## 6. Unordered HTML List - Choose List Item Marker

- The CSS list-style-type property is used to define the style of the list item marker. It can have one of the following values:

| Value | Description |
|---|---|
| disc | Sets the list item marker to a bullet (default) |
| circle | Sets the list item marker to a circle |
| square | Sets the list item marker to a square |
| none | The list items will not be marked |

- Example - Disc

```
<ul style="list-style-type:disc;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

- Example - Circle

```
<ul style="list-style-type:circle;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

- Example - Square

```
<ul style="list-style-type:square;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

- Example - None

```
<ul style="list-style-type:none;">
  <li>Coffee</li>
```

```
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

### 7. Nested HTML Lists

- Lists can be nested (list inside list):
- Example

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

- **Note**: A list item (<li>) can contain a new list, and other HTML elements, like images and links, etc.

### 8. Horizontal List with CSS

- HTML lists can be styled in many different ways with CSS.
- One popular way is to style a list horizontally, to create a navigation menu:
- Example

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333333;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 16px;
  text-decoration: none;
}

li a:hover {
  background-color: #111111;
}
```

```
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

</body>
</html>
```

## 9. HTML Ordered Lists

- An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.
- The list items will be marked with numbers by default:
- Example

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

## 10. Ordered HTML List - The Type Attribute

- The type attribute of the <ol> tag, defines the type of the list item marker:

| Type | Description |
|---|---|
| type="1" | The list items will be numbered with numbers (default) |
| type="A" | The list items will be numbered with uppercase letters |
| type="a" | The list items will be numbered with lowercase letters |
| type="I" | The list items will be numbered with uppercase roman numbers |
| type="i" | The list items will be numbered with lowercase roman numbers |

- Numbers:

```
<ol type="1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

- Uppercase Letters:

```
<ol type="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

- Lowercase Letters:

```
<ol type="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

- Uppercase Roman Numbers:

```
<ol type="I">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

- Lowercase Roman Numbers:

```
<ol type="i">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

**11. Control List Counting**

- By default, an ordered list will start counting from 1. If you want to start counting from a specified number, you can use the start attribute:

- Example

```
<ol start="50">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

**12. Nested HTML Lists**

- Lists can be nested (list inside list):

- Example

```
<ol>
  <li>Coffee</li>
  <li>Tea
    <ol>
      <li>Black tea</li>
      <li>Green tea</li>
    </ol>
  </li>
  <li>Milk</li>
</ol>
```

- **Note:** A list item (<li>) can contain a new list, and other HTML elements, like images and links, etc.

**13. HTML Other Lists**

- HTML also supports description lists.

**14. HTML Description Lists**

– A description list is a list of terms, with a description of each term.

– The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

– Example

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

– Use the HTML <dl> element to define a description list

– Use the HTML <dt> element to define the description term

– Use the HTML <dd> element to describe the term in a description list

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Ordered List</title>
   </head>

   <body>
      <ol type = "i">
         <li>Beetroot</li>
         <li>Ginger</li>
         <li>Potato</li>
         <li>Radish</li>
      </ol>
   </body>

</html>
```

You can use **start** attribute for <ol> tag to specify the starting point of numbering you need. Following are the possible options −

```
<ol type = "1" start = "4">    - Numerals starts with 4.
<ol type = "I" start = "4">    - Numerals starts with IV.
<ol type = "i" start = "4">    - Numerals starts with iv.
<ol type = "a" start = "4">    - Letters starts with d.
<ol type = "A" start = "4">    - Letters starts with D.
```

Example where we used <ol type = "i" start = "4" >

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Ordered List</title>
   </head>

   <body>
      <ol type = "i" start = "4">
```

```
            <li>Beetroot</li>
            <li>Ginger</li>
            <li>Potato</li>
            <li>Radish</li>
        </ol>
    </body>

</html>
```

Example

```
<ol reversed>
    <li>Item 3</li>
    <li>Item 2</li>
    <li>Item 1</li>
</ol>
```

Example

```
<style>
    #nav {
        background: lightgray;
        overflow: auto;
        }
    #nav li {
        float: left;
        list-style-type: none;
        padding: 10px;
        }
</style>
<body>
<ul id="nav">
    <li><a href="#Using_Lists_for_Menus">Home</a></li>
    <li><a href="#Using_Lists_for_Menus">About Us</a></li>
    <li><a href="#Using_Lists_for_Menus">Contact Us</a></li>
</ul>
```

# HTML Tables

- HTML tables allow web developers to arrange data into rows and columns.
- Other than arranging data in rows and columns tables can also be used :
    1. To divide the page into various sections
    2. To create menus
    3. To add interactive form fields
    4. To create fast loading headers of page
    5. To achieve alignment of images

### 1. Define a HTML Table

- The <table> tag defines an HTML table.
- Each table row is defined with a <tr> tag.
    - Each table header is defined with a <th> tag.
        - By default, the text in <th> elements are bold and centered.
    - Each table data/cell is defined with a <td> tag.
        - By default, the text in <td> elements are regular and left-aligned.

Example

```
<table border="10" bordercolor="red" bgcolor="yellow" >
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

- **Note:** The <td> elements are the data containers of the table.
  They can contain all sorts of HTML elements; text, images, lists, other tables, etc.

- **WIDTH attribute**

  Allows designers to set the width of the table using two methods, either as an explicit value or a percentage value. e.g.

  a) `<table width=400>` gives the table which is fixed to 400 pixels.
  b) using a percentage of the available space which allows the table to expand or contract along with the browser if it is resized at any point.
  e.g.

`<Table Width="80%">` Forces The Table To Take 90% of The Available Horizontal Space.

Using CSS <table style="width:60%" >

**2. HTML Table - Add a Border**

– This attribute sets the thickness of the borders surrounding the table. e.g.

<Table Border =”2”>
<table border="10" bordercolor="red" bgcolor="yellow" >

– If no border is desired a value of BORDER=0 is given.

<table border="0" bordercolor="red" bgcolor="yellow" >

– To add a border to a table, use the CSS border property:

Example

```
table, th, td {
  border: 1px solid black;
}

<table style="width:60%; border:2px solid black;" >
```

**3. HTML Table - Add Cell Padding**

– Cell padding specifies the space between the cell content and its borders.

– If you do not specify a padding, the table cells will be displayed without padding.

```
<table cellpadding = "20" border ="2">
```

– To set the padding, use the CSS padding property:

Example

```
th, td {
  padding: 15px;
}
```

**4. HTML Table - align**

– To align table on the right of the browser <table border="2" align="right" >

– To center table on the browser <table border="2" align="center" >

By default, table headings are bold and centered.

– ALIGN

Specifies the horizontal alignment of cell data for a row. ALIGN can be either LEFT, RIGHT, or CENTER. <tr align="right" > <td>Jill</td> </tr>

– VALIGN

Specifies the vertical alignment of cell data for a row. It takes one of the values TOP, MIDDLE, or BOTTOM

**5. HTML Table – Add Cell Spacing Border Spacing**

– cell spacing specifies the space between the cells.

<table cellspacing = “20”>

<table border="20" align="center" cellspacing="40"

To set the border spacing for a table, use the CSS border-spacing property:

```
table {
  border-spacing: 5px;
}
```

## 6. HTML Table - Cell that Spans Many Columns

– To make a cell span more than one column, use the colspan attribute:

Example

```
<table style="width:100%">
  <tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
  </tr>
  <tr>
    <td>Bill Gates</td>
    <td>55577854</td>
    <td>55577855</td>
  </tr>
</table>
```

## 7. HTML Table - Cell that Spans Many Rows

– To make a cell span more than one row, use the rowspan attribute:

Example

```
<table style="width:100%">
  <tr>
    <th>Name:</th>
    <td>Bill Gates</td>
  </tr>
  <tr>
    <th rowspan="2">Telephone:</th>
    <td>55577854</td>
  </tr>
  <tr>
    <td>55577855</td>
  </tr>
</table>
```

## 8. Row Group Elements

– Table rows may be grouped into a table head, table foot and the table body sections.
This division enables the browser to support scrolling of table body independent of the table head and foot.

– Table head and foot information can be repeated each page that contains table data.

– **THEAD**

- This contains the header information about the columns. This element defines a group of header rows in a table.
- The <thead> element is used in conjunction with the <tbody> and <tfoot> elements to specify each part of a table (header, body, footer).
- The <thead> element must have one or more <tr> tags inside
- The <thead> tag must be used in the following context: As a child of a <table> element, after any <caption> and <colgroup> elements, and before any <tbody>, <tfoot>, and <tr> elements.

- **TFOOT**
- This contains the footer information about the columns. This element defines a group of footer rows in a table.

- **TBODY**
- This defines a group of data rows in a table. A table must have one or more TBODY element. It contains row groups

```
<html>
      <head>
            <style>
            thead {color: green;}
            tbody {color: blue;}
            tfoot {color: red;}
            table, th, td {
            border: 1px solid black;
            }
            </style>
      </head>
<body>
      <table>
            <thead>
                  <tr>
                        <th>Month</th>
                        <th>Savings</th>
                  </tr>
            </thead>
            <tbody>
                  <tr>
                        <td>January</td>
                        <td>$100</td>
                  </tr>
                  <tr>
                        <td>February</td>
                        <td>$80</td>
                  </tr>
            </tbody>
            <tfoot>
                  <tr>
                        <td>Sum</td>
                        <td>$180</td>
                  </tr>
            </tfoot>
      </table>
</body>
</html>
```

## 9. HTML Table - Add a Caption

- This gives the caption for the title of the table. The default position of the title is centered at the top of the table.
- It is only permitted after the TABLE tag.
- It has the following align attribute values: ALIGN=BOTTOM, TOP, LEFT, RIGHT
- To add a caption to a table, use the <caption> tag:

Example

```
<table style="width:100%">
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$50</td>
  </tr>
</table>
```

- **Note:** The <caption> tag must be inserted immediately after the <table> tag.

## 10. Summary

- Use the HTML <table> element to define a table
- Use the HTML <tr> element to define a table row
- Use the HTML <td> element to define a table data
- Use the HTML <th> element to define a table heading
- Use the HTML <caption> element to define a table caption
- Use the colspan attribute to make a cell span many columns
- Use the rowspan attribute to make a cell span many rows

## 11. HTML Table Tags

| Tag | Description |
|---|---|
| <table> | Defines a table |
| <th> | Defines a header cell in a table |
| <tr> | Defines a row in a table |
| <td> | Defines a cell in a table |
| <caption> | Defines a table caption |
| <colgroup> | Specifies a group of one or more columns in a table for formatting |
| <col> | Specifies column properties for each column within a <colgroup> element |
| <thead> | Groups the header content in a table |

| | |
|---|---|
| <tbody> | Groups the body content in a table |
| <tfoot> | Groups the footer content in a table |

**Sample Program 1**

```
<html>
<body text="blue" >
      <table border="1" cellspacing="10" >
            <tr>
                  <td bgcolor= "#888888">Column 1</td>
                  <td bgcolor="#ffff0">Column 2</td>
                  <td bgcolor="#aaaa">Column 3</td>
            </tr>
            <tr>
                  <td rowspan="2">Row 1 Cell 1</td>
                  <td bgcolor="yellow" >Row 1 Cell 2</td>
                  <td bgcolor="green" >Row 1 Cell 3</td>
            </tr>
            <tr>
                  <td bgcolor= "rgb(255,255,0)" >Row 2 Cell
                  2</td>
                  <td bgcolor= "rgb(0,255,0)" >Row 2 Cell
                  3</td>
            </tr>
            <tr>
                  <td colspan="3">Row 3 Cell 1</td>
            </tr>
      </table>
  </body>
</html>
```

**Sample Program 2**

```
<html >
  <head>
     <title>Tables</title>
  </head>
<body>
    <h1>Table Example Page</h1>
          <table border = "1">
                <caption>Here is a sample table.</caption>
                <thead>
                    <tr>
                        <!-- merge two rows -->
                        <th rowspan = "2">
                          <img src = "camel.gif" width = "205" height
                        =
                            "167" alt = "Picture of a camel" />
                        </th>
                        <!-- merge four columns -->
                        <th colspan =" 4" >
                           <h1> Camelid Comparison</h1>
                           <p> Approximate as of 6/2007</p>
                        </th>
                    </tr>
                    <tr>
```

```
            <th># of Humps</th>
            <th>Indigenous region</th>
            <th>Spits?</th>
            <th>Produces Wool?</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <th>Camels (bactrian)</th>
              <td>2</td>
              <td>Africa/Asia</td>
              <td>Yes</td>
              <td>Yes</td>
        </tr>
        <tr>
            <th>Llamas</th>
              <td>1</td>
              <td>Andes Mountains</td>
              <td>Yes</td>
              <td>Yes</td>
        </tr>
    </tbody>
</table>
<p>
<table border="1">
    <tr>
        <td width="50%">
            <ul>
                <li>List Item 1</li>
                <li>List Item 2</li>
                <li>List Item 3</li>
            </ul>
        </td>
        <td>
            <ul>
                <li>List Item 4</li>
                <li>List Item 5</li>
                <li>List Item 6</li>
            </ul>
        </td>
    </tr>
    <tr>
        <td>
            <p>Avoid losing floppy disks with
            important school...</p>
        </td>
        <td>
            <a href="http://www.espn.com"
            target="_blank" rel="nofollow">
                <img
                src="http://www.Facebook.com/pics
                /htmlT/ahman.gif"/>
            </a>
        </td>
    </tr>
</table>
```

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

**Sample Program 3**

```
<html>
    <head>
          <title>Lists</title>
    </head>
<body>
    <h1>The Best Features of the Internet</h1>
    <!-- create an unordered list -->
    <ul>
       <li>You can meet new people from countries around the
       world.</li>
       <li>You have access to new media as it becomes public:
          <ul>
             <li>New games</li>
             <li>Around the clock news</li>
             <li>Search engines</li>
             <li>Shopping</li>
          </ul> <!-- ends the nested list of line-->
       </li>
       <li>New applications
          <!-- nested ordered list -->
          <ol>
               <li>For business</li>
               <li>For pleasure</li>
          </ol>
       </li> <!-- ends line 27 new applications li -->
       <li>Programming <!-- another nested ordered list -->
          <ol>
               <li>XML</li>
               <li>Java</li>
               <li>XHTML</li>
               <li>Scripts</li>
               <li>New languages</li>
          </ol>
       </li> <!-- ends programming li of line  -->
       <li>Links</li>
       <li>Keeping in touch with old friends</li>
       <li>It is the technology of the future!</li>
    </ul> <!-- ends the unordered list of line -->
</body>
</html>
```

**Sample Program 4**

```
<html>
<head>
    <title> Navigation Bar</title>
</head>
<body>
    <p>
          <a href = "list.html">
                <img src = "buttons/list.jpg" width = "65" height = "50"
                alt = "List Example Page" />
          </a>
```

```
            <a href = "contact.html">
                  <img src = "buttons/contact.jpg" width = "65" height =
                  "50" alt = "Contact Page" />
            </a>
            <a href = "table1.html">
                  <img src = "buttons/table.jpg" width = "65" height =
                  "50" alt = "Table Page" />
            </a>
            <a href = "form.html">
                  <img src = "buttons/form.jpg" width = "65" height = "50"
                  alt = "Feedback Form" />
            </a>
      </p>
      <p>if x &lt; 10 then increment x by 1</p>
</body>
</html>
```

# HTML – Frames

## 1. Introduction to frames

- HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document.
- Multiple views offer designers a way to keep certain information visible, while other views are scrolled or replaced. For example, within the same window, one frame might display a static banner, a second a navigation menu, and a third the main document that can be scrolled through or replaced by navigating in the second frame.
- A collection of frames in the browser window is known as a frameset.
- The window is divided into frames in a similar way the tables are organized: into rows and columns.

## 2. Disadvantages of Frames

- There are few drawbacks with using frames, so it's never recommended to use frames in your webpages −

  i.) Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.

  ii.) Sometimes your page will be displayed differently on different computers due to different screen resolution.

  iii.) The browser's back button might not work as the user hopes thus causing navigational issues to users.

  iv.) There are still few browsers that do not support frame technology.

  v.) Using HTML frames is not good for SEO (the content of your frame is just invisible for major search engines);

  vi.) You cannot add internal links of the website built with HTML frames to your favorites

## 3. Creating Frames

- To use frames on a page we use <frameset> tag instead of <body> tag.
- The <frameset> tag defines, how to divide the window into frames.
- The rows attribute of <frameset> tag defines horizontal frames and cols attribute defines vertical frames.
- Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

  **Note** − The <frame> tag deprecated in HTML5.

  **Example**

  Following is the example to create three horizontal frames −

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Frames</title>
   </head>

   <frameset rows = "10%,80%,10%">
      <frame name = "top" src = "/html/top_frame.htm" />
      <frame name = "main" src = "/html/main_frame.htm" />
      <frame name = "bottom" src = "/html/bottom_frame.htm" />

      <noframes>
         <body>Your browser does not support frames.</body>
      </noframes>

   </frameset>
</html>
```

**Example**

This creates three frames vertically −

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Frames</title>
   </head>

   <frameset cols = "25%,50%,25%">
      <frame name = "left" src = "/html/top_frame.htm" />
      <frame name = "center" src = "/html/main_frame.htm" />
      <frame name = "right" src = "/html/bottom_frame.htm" />

      <noframes>
         <body>Your browser does not support frames.</body>
      </noframes>
   </frameset>

</html>
```

## 4. The <frameset> Tag Attributes

– Following are important attributes of the <frameset> tag −

| Attribute & Description |
|---|
| **cols**<br>– Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways −<br>i.) Absolute values in pixels. For example, to create three vertical frames, use *cols = "100, 500, 100"*.<br>ii.) A percentage of the browser window. For example, to create three vertical frames, use *cols = "10%, 80%, 10%"*.<br>iii.) Using a wildcard symbol. For example, to create three vertical frames, use *cols = "10%, *, 10%"*. In this case wildcard takes remainder of the window. |

| |
|---|
| iv.) As relative widths of the browser window. For example, to create three vertical frames, use *cols = "3*, 2*, 1*"*.<br>This is an alternative to percentages.<br>You can use relative widths of the browser window.<br>Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth. |
| **rows**<br>– This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset.<br>For example, to create two horizontal frames, use rows = "10%, 90%".<br>You can specify the height of each row in the same way as explained above for columns. |
| **border**<br>– This attribute specifies the width of the border of each frame in pixels.<br>For example, border = "5".<br>A value of zero means no border. |
| **frameborder**<br>– This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder = "0" specifies no border. |
| **framespacing**<br>– This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing = "10" means there should be 10 pixels spacing between each frames. |

## 5. The <frame> Tag Attributes

– Following are the important attributes of <frame> tag −

| Attribute & Description |
|---|
| **src**<br>– This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL.<br>– For example, src = "/html/top_frame.htm" will load an HTML file available in html directory. |
| **name**<br>– This attribute allows you to give a name to a frame.<br>– It is used to indicate which frame a document should be loaded into.<br>– This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link. |
| **frameborder**<br>– This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no). |
| **marginwidth** |

| |
|---|
| – This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content.<br>– The value is given in pixels. For example marginwidth = "10". |
| **marginheight**<br>– This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents.<br>The value is given in pixels. For example marginheight = "10". |
| **noresize**<br>– By default, you can resize any frame by clicking and dragging on the borders of a frame.<br>The noresize attribute prevents a user from being able to resize the frame.<br>For example noresize = "noresize". |
| **scrolling**<br>– This attribute controls the appearance of the scrollbars that appear on the frame.<br>This takes values either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars. |
| **longdesc**<br>– This attribute allows you to provide a link to another page containing a long description of the contents of the frame.<br>For example longdesc = "framedescription.htm" |

## 6. Browser Support for Frames

– If a user is using any old browser or any browser, which does not support frames then <noframes> element should be displayed to the user.
– So you must place a <body> element inside the <noframes> element because the <frameset> element is supposed to replace the <body> element, but if a browser does not understand <frameset> element then it should understand what is inside the <body> element which is contained in a <noframes> element.
– You can put some nice message for your user having old browsers. For example, *Sorry!! your browser does not support frames.*

## 7. Frame's name and target attributes

– One of the most popular uses of frames is to place navigation bars in one frame and then load main pages into a separate frame.
– Let's see following example where a test.htm file has following code −

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Target Frames</title>
   </head>

   <frameset cols = "200, *">
      <frame src = "html/menu.html" name = "menu_page" />
      <frame src = "html/main.html" name = "main_page" />
```

```
      <noframes>
         <body>Your browser does not support frames.</body>
      </noframes>
   </frameset>

</html>
```

- Here, we have created two columns to fill with two frames. The first frame is 200 pixels wide and will contain the navigation menu bar implemented by **menu.html** file.
- The second column fills in remaining space and will contain the main part of the page and it is implemented by **main.html** file.
- For all the three links available in menu bar, we have mentioned target frame as **main_page**, so whenever you click any of the links in menu bar, available link will open in main page.

Following is the content of menu.htm file

```
<!DOCTYPE html>
<html>

   <body bgcolor = "#4a7d49">
      <a href = "http://www.google.com" target = "main_page">Google</a>
      <br />
      <br />

      <a href = "http://www.microsoft.com" target = "main_page">Microsoft</a>
      <br />
      <br />

      <a href = "http://news.bbc.co.uk" target = "main_page">BBC News</a>
   </body>

</html>
```

Following is the content of main.htm file −

```
<!DOCTYPE html>
<html>

   <body bgcolor = "#b5dcb3">
      <h3>This is main page and content from any link will be
       displayed here.</h3>
      <p>So now click any link and see the result.</p>
   </body>

</html>
```

- Load **test.htm** file, see the result −
- Click links available in the left panel and see the result.

**The *target attribute* can also take one of the following values −**

| Option & Description |
|---|

| |
|---|
| **_self**<br>Loads the page into the current frame. |
| **_blank**<br>Loads a page into a new browser window. Opening a new window. |
| **_parent**<br>Loads the page into the parent window, which in the case of a single frameset is the main browser window. |
| **_top**<br>Loads the page into the browser window, replacing any current frames. |
| **targetframe**<br>Loads the page into a named targetframe. |

# HTML - Iframes

- You can define an inline frame with HTML tag **<iframe>**.
- The <iframe> tag is not somehow related to <frameset> tag, instead, it can appear anywhere in your document.
- The <iframe> tag defines a rectangular region within the document in which the browser can display a separate document, including scrollbars and borders.
- An inline frame is used to embed another document within the current HTML document.
- An HTML iframe is used to display a web page within a web page.
- The **src** attribute is used to specify the URL of the document that occupies the inline frame.

## Example

Following is the example to show how to use the <iframe> −

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Iframes</title>
   </head>

   <body>
      <p>Document content goes here...</p>

      <iframe src = "html/menu.html" width = "555" height = "200">
         Sorry your browser does not support inline frames.
      </iframe>

      <p>Document content also go here...</p>
   </body>

</html>
```

### 1. The <Iframe> Tag Attributes

- Most of the attributes of the <iframe> tag, including *name, class, frameborder, id, longdesc, marginheight, marginwidth, name, scrolling, style,* and *title* behave exactly like the corresponding attributes for the <frame> tag.
- **Note** − The *frameborder*, *marginwidth*, *longdesc*, *scrolling*, *marginheight* attributes deprecated in HTML5. Do not use these attributes.

| Attribute & Description |
|---|
| **src**<br>– This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL.<br>For example, src = "/html/top_frame.htm" will load an HTML file available in html directory. |
| **name**<br>– This attribute allows you to give a name to a frame.<br>It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link. |
| **frameborder**<br>– This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no). |
| **marginwidth**<br>– This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content.<br>The value is given in pixels. For example marginwidth = "10". |
| **marginheight**<br>– This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents.<br>The value is given in pixels. For example marginheight = "10". |
| **height**<br>– This attribute specifies the height of <iframe>. |
| **scrolling**<br>– This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto".<br>For example scrolling = "no" means it should not have scroll bars. |
| **longdesc**<br>– This attribute allows you to provide a link to another page containing a long description of the contents of the frame.<br>For example longdesc = "framedescription.htm" |
| **width**<br>– This attribute specifies the width of <iframe>. |

**Summary**

- The `<frameset>` and `</frameset>` attributes are used instead of the `<body>` tag in the HTML document that consists of frames.
- The `<frame />` attribute contains the name of a frame, the `name` and `id` attributes and the URL of the HTML page that is displayed in this frame.
- The `<iframe>` and `</iframe>` attributes allow to set a frame in a regular HTML document.
- The `<noframes>` and `</noframes>` attributes contain alternative information for the browsers that do not support frames.

**Sample Program 1**

```
<HTML>
<HEAD>
     <TITLE>A simple frameset document</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
        <FRAMESET rows="100, 200">
            <FRAME src="contents_of_frame1.html">
            <FRAME src="contents_of_frame2.gif">
        </FRAMESET>
     <FRAME src="contents_of_frame3.html">
  <NOFRAMES>
      <P>This frameset document contains:
      <UL>
         <LI><A href="contents_of_frame1.html">Some neat contents</A>
         <LI><IMG src="contents_of_frame2.gif" alt="A neat image">
       <LI><A href="contents_of_frame3.html">Some other neat
          contents</A>
      </UL>
  </NOFRAMES>
</FRAMESET>
</HTML>
```

**Sample Program 2**

```
<HTML>
<HEAD>
     <TITLE>A frameset document</TITLE>
</HEAD>
<FRAMESET cols="33%,33%,33%">
  <FRAMESET rows="*,200">
      <FRAME src="contents_of_frame1.html">
      <FRAME src="contents_of_frame2.gif">
  </FRAMESET>
  <FRAME src="contents_of_frame3.html">
  <FRAME src="contents_of_frame4.html">
</FRAMESET>
</HTML>
```

**Sample Program 3**

```
<HTML>
<HEAD>
```

```
<TITLE>A frameset document</TITLE>
</HEAD>
<FRAMESET cols="33%,33%,33%">
  <FRAMESET rows="*,200">
      <FRAME src="contents_of_frame1.html" scrolling="no">
      <FRAME src="contents_of_frame2.gif"
                marginwidth="10" marginheight="15" noresize>
  </FRAMESET>
  <FRAME src="contents_of_frame3.html" frameborder="0">
  <FRAME src="contents_of_frame4.html" frameborder="0">
</FRAMESET>
</HTML>
```

### Sample program 4

```
<HTML>
<HEAD>
     <TITLE>A frameset document</TITLE>
</HEAD>
      <FRAMESET rows="50%,50%">
         <FRAME name="fixed" src="init_fixed.html">
         <FRAME name="dynamic" src="init_dynamic.html">
      </FRAMESET>
      </HTML>
```

Then, in `init_dynamic.html`, we link to the frame named "dynamic".

```
<HTML>
<HEAD>
     <TITLE>A document with anchors with specific targets</TITLE>
</HEAD>
<BODY>
      ...beginning of the document...
      <P>Now you may advance to
          <A href="slide2.html" target="dynamic">slide 2.</A>
      ...more document...
      <P>You're doing great. Now on to
          <A href="slide3.html" target="dynamic">slide 3.</A>
</BODY>
</HTML>
```

Activating either link opens a new document in the frame named "dynamic" while the other frame, "fixed", maintains its initial contents.

***Note.*** *A frameset definition never changes, but the contents of one of its frames can. Once the initial contents of a frame change, the frameset definition no longer reflects the current state of its frames.*

*There is currently no way to encode the entire state of a frameset in a URI. Therefore, many user agents do not allow users to assign a bookmark to a frameset.*

*Framesets may make navigation forward and backward through your user agent's history more difficult for users.*

**Sample program 4**

Use the <iframe> tag to embed another document within the current HTML document:

```
<IFRAME src="foo.html" width="400" height="500"
             scrolling="auto" frameborder="1">
  [Your user agent does not support frames or is currently configured
  not to display frames. However, you may visit
  <A href="foo.html">the related document.</A>]
  </IFRAME>
```

**Sample Program 5**

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>HTML frames</title>
    <style type="text/css">
    </style>
</head>
<frameset rows="15%,*,10%">
  <frame src="header.html" name="top" id="top" />
    <frameset cols="20%,*,11%">
      <frame src="menu.html" name="left" id="left" />
      <frame src="home.html" name="home" id="home" />
      <frame src="right.html" name="right" id="right" />
    </frameset>
  <frame src="footer.html" name="bottom" id="bottom" />
    <noframes>
      <body>
        <p><a href="menu.html">Contents</a></p>
      </body>
    </noframes>
</frameset>
</html>
```

# HTML <meta> Tag

- The <meta> tag defines metadata about an HTML document. Metadata is data (information) about data.
- <meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.
- Metadata will not be displayed on the page, but is machine parsable.
- Metadata is used by browsers (how to display content or reload page), search engines (keywords), and other web services.
- It has a method that lets web designers take control over the viewport (the user's visible area of a web page)

Examples

1. Define keywords for search engines:

```
<meta name="keywords" content="HTML, CSS, JavaScript">
```

2. Define a description of your web page:

```
<meta name="description" content="Free Web tutorials for HTML and CSS">
```

3. Define the author of a page:

```
<meta name="author" content="Hudson Wanjau">
```

4. Refresh document every 30 seconds:

```
<meta http-equiv="refresh" content="30">
```

5. Setting the viewport to make your website look good on all devices:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

   - The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.
   - You should include the following <meta> element in all your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

   - This gives the browser instructions on how to control the page's dimensions and scaling.

   **Example:**

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <meta name="author" content="John Doe">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
</head>
```

# HTML Colors

**1. Introduction**

– Colors ca be specified on page level using <body> tag or they can be set colors for individual tags using **bgcolor** attribute.

– The <body> tag has following attributes which can be used to set different colors −
  - **bgcolor** − sets a color for the background of the page.
  - **text** − sets a color for the body text.
  - **alink** − sets a color for active links or selected links.
  - **link** − sets a color for linked text.
  - **vlink** − sets a color for *visited links* − that is, for linked text that you have already clicked on.

## 2. HTML Color Coding Methods

– There are following three different methods to set colors in your web page −
  i) **Color names** − You can specify color names directly like green, blue or red.
  ii) **Hex codes** − A six-digit code representing the amount of red, green, and blue that makes up the color.
  iii) **Color decimal or percentage values** − This value is specified using the rgb( ) property.

### HTML Colors - Color Names Example

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Colors by Name</title>
   </head>

   <body text = "blue" bgcolor = "green">
      <p>Use different color names for body and table and see the
       result. </p>

      <table bgcolor = "black">
         <tr>
            <td>
               <font color = "white">This text will appear white
                on black background.</font>
            </td>
         </tr>
      </table>
   </body>

</html>
```

### HTML Colors - Hex Codes Example

A hexadecimal is a 6-digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).
Each hexadecimal code will be preceded by a pound or hash sign #.

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Colors by Hex</title>
```

```
   </head>

   <body text = "#0000FF" bgcolor = "#00FF00">
      <p>Use different color hexa for for body and table and see
       the result.</p>

      <table bgcolor = "#000000">
         <tr>
            <td>
               <font color = "#FFFFFF">This text will appear white
               on black background.</font>
            </td>
         </tr>
      </table>
   </body>

</html>
```

**HTML Colors - RGB Values Example**

This color value is specified using the rgb( ) property. This property takes three values, one each for red, green, and blue. The value can be an integer between 0 and 255 or a percentage.

Note − All the browsers does not support rgb() property of color so it is recommended not to use it.

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML Colors by RGB code</title>
   </head>

   <body text = "rgb(0,0,255)" bgcolor = "rgb(0,255,0)">
     <p>Use different color code for for body and table and see
     the result.</p>

      <table bgcolor = "rgb(0,0,0)">
         <tr>
            <td>
              <font color = "rgb(255,255,255)">This text will
              appear white on black background.</font>
            </td>
         </tr>
      </table>
   </body>

</html>
```

## 3. HTML - Backgrounds

By default, your webpage background is white in color. You may not like it, but no worries. HTML provides you following two good ways to decorate your webpage background.

- **HTML Background with Colors**

The **bgcolor** attribute is used to control the background of an HTML element, specifically page body and table backgrounds.

```
<table bgcolor = "#6666FF" width = "100%">
<table bgcolor = "yellow" width = "100%">
<table bgcolor = "rgb(255,0,255)" width = "100%">
```

- **HTML Background with Images**

  You can specify an image to set background of your HTML page or table.

```
 <table background = "/images/html.gif" width = "100%"
height = "100">
<table background = "/images/pattern1.gif" width = "100%"
height = "100">
```

## 4. HTML - Fonts

- Fonts play a very important role in making a website more user friendly and increasing content readability. Font face and color depends entirely on the computer and browser that is being used to view your page
- HTML **<font>** tag can be used to add style, size, and color to the text.
- You can use a **<basefont>** tag to set all of your text to the same size, face, and color.
- The font tag is having three attributes called **size, color**, and **face** to customize your fonts.
- Note −The font and basefont tags are deprecated , it's suggested to use CSS styles to manipulate your fonts.

  - **Set Font Size Example**

```
<html>

   <head>
      <title>Setting Font Size</title>
   </head>

   <body>
      <font size = "1">Font size = "1"</font><br />
      <font size = "2">Font size = "2"</font><br />
      <font size = "3">Font size = "3"</font><br />
      <font size = "4">Font size = "4"</font><br />
      <font size = "5">Font size = "5"</font><br />
      <font size = "6">Font size = "6"</font><br />
      <font size = "7">Font size = "7"</font>
   </body>

</html>
```

  - **Setting Font Face Example**

```
<head>
      <title>Font Face</title>
   </head>
```

```
   <body>
      <font face = "Times New Roman" size = "5">Times New
      Roman</font><br />
      <font face = "Verdana" size = "5">Verdana</font><br />
      <font face = "Comic sans MS" size =" 5">Comic Sans
MS</font><br/>
      <font face = "WildWest" size = "5">WildWest</font><br
/>
      <font face = "Bedrock" size = "5">Bedrock</font><br />
   </body>
```

- **Setting Font Color Example**

```
<head>
      <title>Setting Font Color</title>
   </head>

   <body>
      <font color = "#FF00FF">This text is in
pink</font><br/>
      <font color = "red">This text is red</font>
   </body>
```

- **The <basefont> Element**

  The <basefont> element is supposed to set a default font size, color, and typeface for any parts of the document that are not otherwise contained within a <font> tag.

  You can use the <font> elements to override the <basefont> settings.

  The <basefont> tag also takes color, size and face attributes and it will support relative font setting by giving size a value of +1 for a size larger or −2 for two sizes smaller.

```
<html>

   <head>
      <title>Setting Basefont Color</title>
   </head>

   <body>
       <basefont face = "arial, verdana, sans-serif" size = "2"
       color = "#ff0000">
      <p>This is the page's default font.</p>
      <h2>Example of the &lt;basefont&gt; Element</h2>

      <p><font size = "+2" color = "darkgray">
            This is darkgray text with two sizes larger
         </font>
      </p>

      <p><font face = "courier" size = "-1" color = "#000000">
            It is a courier font, a size smaller and black in color.
         </font>
      </p>
   </body>
```

```
</html>
```

# HTML <form> Tag

## 1. Introduction

- The <form> tag is used to create an HTML form for user input.
- Form contains special elements called *controls* (checkboxes, radio buttons, menus, etc.), and labels on those controls.
- Users generally "complete" a form by modifying its controls (entering text, selecting menu items, etc.), before submitting the form to an agent for processing (e.g., to a Web server, to a mail server, etc.)
- A form then takes input from the user and posts it to a back-end application such as CGI, ASP Script, VB Script, Python Script or PHP script. The back-end application will perform required processing on the passed data based on defined business logic inside the application. Refer to CGI for details on how form data upload works.
- The FORM element acts as a container for controls and can contain one or more of the following form elements:
    - <input> <textarea> <button> <select> <option> <optgroup> <fieldset> <label> <output>

## 2. Attributes

| Attribute | Value | Description |
|---|---|---|
| action | *URL* | Specifies where to send the form-data when a form is submitted:<br>i) Sending to CGI Programs. action=”/cgi-bin/handler.cgi”<br>ii) Sending to JavaScript Functions. action=”javascript:SomeFunction()”<br>iii) Sending to mailto: Links. action=”mailto:name@domain.com”<br>iv) Sending to a Different Web Page. action=”http://newdomain.com/page.html” |
| autocomplete | on<br>off | Specifies whether a form should have autocomplete on or off |
| enctype | application/x-www-form-urlencoded<br>multipart/form-data<br>text/plain | Specifies how the form-data should be encoded when submitting it to the server (only for method="post") |
| method | get<br>post | Specifies the HTTP method to use when sending form-data. Get for is a request for a static resource e.g. HTML page. Post is a request for a dynamic resource |

| Attribute | Value | Description |
|---|---|---|
| | | with input parameters of the request contained within the body of the request |
| name | *text* | Specifies the name of a form |
| novalidate | novalidate | Specifies that the form should not be validated when submitted |
| rel | external<br>help<br>license<br>next<br>nofollow<br>noopener<br>noreferrer<br>opener<br>prev<br>search | Specifies the relationship between a linked resource and the current document |
| target | _blank<br>_self<br>_parent<br>_top | Specifies where to display the response that is received after submitting the form |

- *Other Attributes defined elsewhere*
  - id, class (document-wide identifiers)
  - lang (language information), dir (text direction)
  - style (inline style information)
  - title (element title)
  - target (target frame information)
  - onsubmit, onreset, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (intrinsic events)

## 3. Examples

- An HTML form with checkboxes:

```
<form action="/action_page.php" method="get">
          <input type="checkbox" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label>
  <br>
  <input type="checkbox" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label>
  <br>
  <input type="checkbox" name="vehicle3" value="Boat" checked>
  <label for="vehicle3"> I have a boat</label>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

- An HTML form with radio buttons:

```
<form action="/action_page.php" method="get">
  <input type="radio" id="male" name="gender" value="male">
  <label for="male">Male</label>
  <br>
  <input type="radio" id="female" name="gender" value="female"
      checked="checked">
  <label for="female">Female</label>
  <br>
  <input type="radio" id="other" name="gender" value="other">
  <label for="other">Other</label>
  <br><br>
  <input type="submit" value="Submit">
</form>
```

- a form that includes labels, radio buttons, email, file, & push buttons (reset the form or submit it):

```
<form action="http://somesite.com/prog/adduser" method="post">
    <p>
          <label for="firstname">First name: </label>
          <input type="text" id="firstname">
  <br>
          <label for="lastname">Last name: </label>
  <input type="text" id="lastname">
  <br>
          <label for="email">Email: </label>
          <input type="email" id="email" >
  <br>
  <label for="search">Search: </label>
  <input type="search" name="search" >
  <br>
  <label for="Url">Enter URL Address: </label>
  <input type="search" name="url"  autofocus >
  <br>

          <input type="radio" name="sex" value="male"> Male<br>
          <input type="radio" name="sex" value="female"> Female<br>
  <label for="img">Select image:</label>
  <br>
          <input type="file" id="img" name="img" accept="image/*">
  <br>
  <label for="file">Select File:</label><br>
          <input type = "file" /><br>
  <br>
  <label for="date">Date:</label>
  <input id="dob" name="dob" type="date">
  <br> <br>
  <label for="month">Month:</label>
  <input id="expiry" name="expiry" type="month" required>
  <br> <br>
  <input type="submit" value="Send">
  <input type="reset">
          </p>
</form>
```

- Example with JavaScript function:

```
<form method="POST" action="/cgi-bin/formupload/handler.cgi">
```

```
   <input type="hidden" name="something" value="A line of
content">
   <p>
   What is your home page URL?<br>
<input type="text" name="url" size="17" maxlength="44">
</p>
<p>
What is your password?<br>
<input type="password" name="P" size="9" maxlength="20">
</p>
<p>
   Select one or more colors that you like:<br>
<input type="checkbox" name="c_blue" value="yes">
<input type="checkbox" name="c_red" value="yes" CHECKED>
<input type="checkbox" name="c_pink" value="yes" CHECKED>
<input type="checkbox" name="c_yellow" value="yes">
</p>
<p>
What is your favorite color?<br>
<input type="radio" name="color" value="blue">
<input type="radio" name="color" value="red" CHECKED>
<input type="radio" name="color" value="pink">
</p>
<p>
Upload any type of file:<br>
<input type="file" name="upfile">
</p>
<p>
      <input type="button" value="Give me a message" onClick="return
   alert('a message')">
<!-- Note: onClick works only with JavaScript enabled browsers. -->
</p>
<p>
Tell me why you are here!<br>
<textarea name="message" cols="20" rows="5"></textarea>
</p>
<p>
What is your most <b><u>un</u></b>favorite color:<br>
<select name="s_unfav">
      <option value="green">Green</option>
      <option value="red">Red</option>
      <option value="blue" SELECTED>Blue</option>
      <option value="yellow">Yellow</option>
      <option value="pink">Pink</option>
</select>
</p>
<p>
Select one or more colors you do not like:<br>
<select name="s_like" size="4" MULTIPLE>
      <option value="green">Green</option>
      <option value="red" SELECTED>Red</option>
      <option value="blue" SELECTED>Blue</option>
      <option value="yellow">Yellow</option>
      <option value="pink">Pink</option>
</select>
</p>
<p>
```

```
   <input type="image" name="imageclick" src="myimage.gif" border="0">
   </p>
   <p>
   <input type="reset" value="Erase Everything!">
   </p>
   <p>
   <input type="submit" name="1" value="This is my first visit">
   </p>
   <p>
   <input type="submit" name="2"value="I've been here before">
   </p>
</form>
```

– **Another Example**

```
<HTML>
<HEAD>
   <TITLE>Land Of Boxes Form</TITLE>
</HEAD>
<BODY>
   <H1><FONT SIZE="5" COLOR="coral">Land Of Boxes Order Form</FONT></H1>
   <FORM METHOD="post" ACTION="mailto:name@xyz.co.za">
         <TABLE>
         <TR>
               <TD ALIGN="right">Name:</TD>
               <TD><INPUT TYPE="text" NAME="name" SIZE="35"></TD>
         </TR>
         <TR>
               <TD ALIGN="right">Email:</TD>
               <TD><INPUT TYPE="text" NAME="email" SIZE="35" VALUE=""></TD>
         </TR>
         <TR>
               <TD ALIGN="right">Address:</TD>
               <TD><TEXTAREA NAME="address" COLS="30" ROWS="5"></TEXTAREA></TD>
         </TR>
         <TR>
               <TD ALIGN="right">Postcode:</TD>
               <TD><INPUT TYPE="text" NAME="postcode" SIZE="20"></TD>
         </TR>
         </TABLE><BR><BR>
         <TABLE BORDER=1>
               <TR>
                     <TD><B>Product</B></TD>
                     <TD><B>Part Number</B></TD>
                     <TD><B>Quantity</B></TD>
                     <TD><B>Unit Price</B></TD>
                     <TD><B>Subtotal</B></TD>
               </TR>
               <TR>
                     <TD>EconoBox</TD><TD>LB100</TD>
                     <TD><INPUT type="text" name="Qlb100" size="8"></TD>
                        <TD>R5</TD>
                            <TD><INPUT type="text" name="Sublb100" size="15"
                        value="R"></TD>
               </TR>
               <TR>
                     <TD>Standard Box</TD><TD>LB200</TD>
```

```
            <TD><INPUT type="text" name="Qlb200" size="8"></TD>
               <TD>R10</TD>
               <TD><INPUT type="text" name="Sublb200" size="15"
               value="R"></TD>
      </TR>
      <TR>
            <TD>Premium Box</TD><TD>LB300</TD>
               <TD><INPUT type="text" name="Qlb300"
               size="8"></TD><TD>R15</TD>
               <TD><INPUT type="text" name="Sublb300" size="15"
               value="R"></TD>
      </TR>
      <TR>
            <TD>Deluxe Box</TD><TD>LB400</TD>
               <TD><INPUT type="text" name="Qlb400"
               size="8"></TD><TD>R20</TD>
               <TD><INPUT type="text" name="Sublb400" size="15"
               value="R"></TD>
      </TR>
      <TR>
            <TD>Super Deluxe Box</TD><TD>LB500</TD>
               <TD><INPUT type="text" name="Qlb500"
               size="8"></TD><TD>R30</TD>
               <TD><INPUT type="text" name="Sublb500" size="15"
               value="R"></TD>
      </TR>
      <TR>
            <TD COLSPAN=5 ALIGN="right">
               Total: <INPUT type="text" name="total" size="15"
               value="R"></TD>
      </TR>
</TABLE><BR><BR>
<B>Credit Card Details</B><BR>
<TABLE>
      <TR>
            <TD>Card Type:</TD>
      </TR>
      <TR>
               <TD><INPUT type="radio" name="cardmake"
               value="master">Mastercard</TD>
            <TD>Card Number:</TD>
               <TD><INPUT type="text" name="cardnumber"
               size="20"></TD>
      </TR>
      <TR>
               <TD><INPUT type="radio" name="cardmake"
               value="visa">Visa</TD>
            <TD>Expiry date:</TD>
               <TD><INPUT type="text" name="expiry" size="10"
               maxlength="5" value="mm/yy"></TD
      </TR>
      <TR>
               <TD><INPUT type="radio" name="cardmake"
               value="amex">American Express</TD>
      </TR>
      <TR>
```

```
                        <TD><INPUT type="radio" name="cardmake"
                         value="diners">Diners Club</TD>
                </TR>
                </TABLE><BR><BR>
                <TABLE>
                <TR>
                      <TD ALIGN="right"><B>Delivery Address:</B></TD>
                          <TD><TEXTAREA NAME="delivery" COLS="25"
                          ROWS="5"></TEXTAREA></TD>
                </TR>
          </TABLE><BR><BR>
          <INPUT type="submit" name="submit" value="Send order">
          <INPUT type="reset" name="reset" value="Clear form">
   </FORM>
</BODY>
</HTML>
```

## 4. Control types

- Users interact with forms through *controls*. HTML defines the following control types:

a) **Buttons**

– Authors may create three types of buttons:

i) Submit buttons: When activated, a submit button submits a form.

ii) Reset buttons: When activated, a reset button resets all controls to their initial values.

iii) Push buttons: Push buttons have no default behavior. Each push button may have client-side scripts associated with the element's event attributes. When an event occurs (e.g., the user presses the button, releases it, etc.), the associated script is triggered.

**b) Checkboxes**

– Checkboxes (and radio buttons) are on/off switches that may be toggled by the user. A switch is "on" when the control element's checked attribute is set.

– Several checkboxes in a form may share the same control name.

**c) Radio Buttons**

– Radio buttons are like checkboxes except that when several share the same control name, they are mutually exclusive: when one is switched "on", all others with the same name are switched "off". The INPUT element is used to create a radio button control.

– At all times, exactly one of the radio buttons in a set is checked.

**d) Menus**

– Menus offer users options from which to choose. The SELECT element creates a menu, in combination with the OPTGROUP and OPTION elements.

**e) Text Input**

- Authors may create two types of controls that allow users to input text.
  - The INPUT element creates a single-line input control and
  - The TEXTAREA element creates a multi-line input control.

**f) File select**

- This control type allows the user to select files so that their contents may be submitted with a form. The INPUT element is used to create a file select control.

**g) Hidden Controls**

- Authors may create controls that are not rendered but whose values are submitted with a form. Authors generally use this control type to store information between client/server exchanges that would otherwise be lost due to the stateless nature of HTTP. The INPUT element is used to create a hidden control.

## 5. Control types created with INPUT element

- The control type defined by the INPUT element depends on the value of the TYPE attribute:

| TYPE attribute: | Description |
|---|---|
| **Text** | – Creates a single-line text input control. |
| **Password:** | – Like "text", but the input text is rendered in such a way as to hide the characters (e.g., a series of asterisks). This control type is often used for sensitive input such as passwords.<br>– **Note.** *This mechanism affords only light security protection. Although the password is masked by user agents from casual observers, it is transmitted to the server in clear text, and may be read by anyone with low-level access to the network.* |
| **checkbox** | – Creates a checkbox. |
| **radio** | – Creates a radio button. |
| **submit** | – Creates a submit button. |
| **image** | – Creates a graphical submit button. |
| **reset** | – Creates a reset button. |
| **button** | – Creates a push button. User agents should use the value of the value attribute as the button's label. |
| **hidden** | – Creates a hidden control. |
| **file** | – Creates a file select control. User agents may use the value of the value attribute as the initial file name. |
| **search** | – Creates a simple search action |

| | |
|---|---|
| **url** | – For web addresses. You can use the multiple attribute to enter more than one URL |
| **Dates and Times** | – HTML 5 enables sate pickers.<br>`<input id="dob" name="dob" type="date">`<br>`<input id="expiry" name="expiry" type="month" required>`<br>`<input id="time" name="time" type="time">` |
| **Numbers as Spinboxes** | – HTML5 can create an input field that accepts minimum and maximum numbers and then you can be able to scroll through the numbers and pick one.<br>`<input type="number" min="0" max="10" step="1" value="0">`<br>type="number" means that this is a number field<br>min="0" specifies the minimum acceptable value for this field.<br>max="10" is the maximum acceptable value.<br>step="1", combined with the min value, defines the acceptable numbers in the range: 0, 1, 2, and so on, up to the maxvalue.<br>value="0" is the default value that displays before selecting another number. |
| Numbers **as Sliders** | – <input type="range" min="0" max="10" step="1" value="0"> |

## 6. The BUTTON element

– Buttons created with the BUTTON element, they function just like buttons created with the INPUT element, but they offer richer rendering possibilities: the BUTTON element may have content.

– Attribute definitions

name = *cdata*

This attribute assigns the control name.

value = *cdata*

This attribute assigns the initial value to the button.

type = submit |button |reset

This attribute declares the type of the button. Possible values:

- submit: Creates a submit button. This is the default value.
- reset: Creates a reset button.
- button: Creates a push button.

```
<FORM action="http://somesite.com/prog/adduser" method="post">
    <P>
    <BUTTON name="submit" value="submit" type="submit">
      Send <IMG src="/icons/wow.gif" alt="wow"></BUTTON>
    <BUTTON name="reset" type="reset">
      Reset <IMG src="/icons/oops.gif" alt="oops"></BUTTON>
    </P>
 </FORM>
```

## 7. The SELECT and OPTION elements

– The <select> tag is used to construct drop-down list boxes (sometimes called drop-down menus) and scrolling list boxes (sometimes called scrolling menus)

– SELECT Attribute definitions

*name* = cdata

*This attribute assigns the control name.*

*size* = number

*If a SELECT element is presented as a scrolled list box, this attribute specifies the number of rows in the list that should be visible at the same time.*

*multiple*

*If set, this boolean attribute allows multiple selections. If not set, the SELECT element only permits single selections.*

```
<form method="POST" action="/cgi-bin/handler.cgi">
    <select name="myselect" size="4" MULTIPLE>
        <option value="1">First</option>
        <option value="2" SELECTED>Second</option>
        <option value="3">Third</option>
        <option value="4" SELECTED>Fourth</option>
        <option value="5">Fifth</option>
    </select>
    <input type="submit">
</form>
```

– **OPTION Attribute definitions**

selected

When set, this boolean attribute specifies that this option is pre-selected.

value = *cdata*

This attribute specifies the initial value of the control. If this attribute is not set, the initial value is set to the contents of the OPTION element.

```
<FORM action="http://somesite.com/prog/component-select"
method="post">
   <P>
   <SELECT multiple size="4" name="component-select">
      <OPTION selected value="Component_1_a">Component_1</OPTION>
      <OPTION selected value="Component_1_b">Component_2</OPTION>
      <OPTION>Component_3</OPTION>
      <OPTION>Component_4</OPTION>
      <OPTION>Component_5</OPTION>
      <OPTION>Component_6</OPTION>
      <OPTION>Component_7</OPTION>
   </SELECT>
   <INPUT type="submit" value="Send"><INPUT type="reset">
```

```
   </P>
</FORM>
```

### 8. HTML <textarea> Tag

Represents a multi-line plain-text editing control, useful when you want to allow users to enter a sizeable amount of free-form text, for example a comment on a review or feedback form

The **size** of a text area is specified by the <cols> and <rows> attributes (or with CSS).

The **name** attribute is needed to reference the form data after the form is submitted (if you omit the name attribute, no data from the text area will be submitted).

The **id** attribute is needed to associate the text area with a label.

```
<label for="story">Tell us your story:</label>
<textarea id="story" name="story" rows="5" cols="33">
      It was a dark and stormy night...
</textarea>

<textarea name="textarea"  rows="5" cols="30" minlength="10"
maxlength="20">
      Write something here
</textarea>
<textarea name="textarea" rows="5" cols="30" placeholder="Comment text.">
</textarea>

<textarea name="textarea" rows="5" cols="30" disabled>
      I am a disabled textarea
</textarea>

<textarea name="textarea" rows="5" cols="30" readonly>
      I am a readonly textarea
</textarea>
```

### 9. Labels

Some form controls automatically have labels associated with them (press buttons) while most do not (text fields, checkboxes and radio buttons, and menus).

For those controls that have implicit labels, user agents should use the value of the **value** attribute as the label string.

The LABEL element is used to specify labels for controls that do not have implicit labels,

The for attribute associates a label with another control explicitly: the value of the for attribute must be the same as the value of the id attribute of the associated control element.

```
<FORM action="..." method="post">
<TABLE>
  <TR>
    <TD><LABEL for="fname">First Name</LABEL>
    <TD><INPUT type="text" name="firstname" id="fname">
  <TR>
    <TD><LABEL for="lname">Last Name</LABEL>
    <TD><INPUT type="text" name="lastname" id="lname">
</TABLE>
</FORM>
```

### 10. Adding structure to forms: the FIELDSET and LEGEND elements

The **FIELDSET** element allows authors to group thematically related controls and labels. Grouping controls makes it easier for users to understand their purpose while simultaneously facilitating tabbing navigation for visual user agents and speech navigation for speech-oriented user agents. The proper use of this element makes documents more accessible.

The **LEGEND** element allows authors to assign a caption to a FIELDSET. The legend improves accessibility when the FIELDSET is rendered non-visually.

The example below has a form that one might fill out at the doctor's office. It is divided into three sections: personal information, medical history, and current medication. Each section contains controls for inputting the appropriate information.

```
<FORM action="..." method="post">
 <P>
 <FIELDSET>
  <LEGEND>Personal Information</LEGEND>
  Last Name: <INPUT name="personal_lastname" type="text"
tabindex="1">
  First Name: <INPUT name="personal_firstname" type="text"
  tabindex="2">
  Address: <INPUT name="personal_address" type="text" tabindex="3">
  ...more personal information...
 </FIELDSET>
 <FIELDSET>
  <LEGEND>Medical History</LEGEND>
  <INPUT name="history_illness"
         type="checkbox"
         value="Smallpox" tabindex="20"> Smallpox
  <INPUT name="history_illness"
         type="checkbox"
         value="Mumps" tabindex="21"> Mumps
  <INPUT name="history_illness"
         type="checkbox"
         value="Dizziness" tabindex="22"> Dizziness
  <INPUT name="history_illness"
```

```
        type="checkbox"
        value="Sneezing" tabindex="23"> Sneezing
  ...more medical history...
 </FIELDSET>
 <FIELDSET>
  <LEGEND>Current Medication</LEGEND>
  Are you currently taking any medication?
  <INPUT name="medication_now"
        type="radio"
        value="Yes" tabindex="35">Yes
  <INPUT name="medication_now"
        type="radio"
        value="No" tabindex="35">No

  If you are currently taking medication, please indicate
  it in the space below:
  <TEXTAREA name="current_medication"
            rows="20" cols="50"
            tabindex="40">
  </TEXTAREA>
 </FIELDSET>
</FORM>
```

### 11. Disabled and read-only controls

For example, one may want to disable a form's submit button until the user has entered some required data.

Similarly, an author may want to include a piece of read-only text that must be submitted as a value along with the form.

```
<INPUT disabled name="fred" value="stone">
```

***Note.*** *The only way to modify dynamically the value of the* `disabled` *attribute is through a script.*

***Note.*** *The only way to modify dynamically the value of the* `readonly` *attribute is through a script.*

### 12. Form Submission Method

The method attribute of the FORM element specifies the HTTP method used to send the form to the processing agent. This attribute may take two values:

- get: With the HTTP "get" method, the form data set is appended to the URI specified by the action attribute (with a question-mark ("?") as separator) and this new URI is sent to the processing agent.
- post: With the HTTP "post" method, the form data set is included in the body of the form and sent to the processing agent.

The "get" method should be used when the form is idempotent (i.e., causes no side-effects). Many database searches have no visible side-effects and make ideal applications for the "get" method.

If the service associated with the processing of a form causes side effects (for example, if the form modifies a database or subscription to a service), the "post" method should be used.

### 13. Successful Controls

A *successful control* is "valid" for submission. Every successful control has its control name paired with its current value as part of the submitted form data set. A successful control must be defined within a FORM element and must have a control name. However:

- Controls that are disabled cannot be successful.
- All "on" checkboxes may be successful.
- For radio buttons that share the same value of the name attribute, only the "on" radio button may be successful.
- For menus, the control name is provided by a SELECT element and values are provided by OPTION elements. Only selected options may be successful. When no options are selected, the control is not successful and neither the name nor any values are submitted to the server when the form is submitted.
- The current value of a file select is a list of one or more file names. Upon submission of the form, the *contents* of each file are submitted with the rest of the form data. The file contents are packaged according to the form's content type.
- The current value of an object control is determined by the object's implementation.

Hidden controls and controls that are not rendered because of style sheet settings may still be successful.

### 14. Processing form data

When the user submits a form (e.g., by activating a submit button), the user agent processes it as follows.

***Step one: Identify the successful controls***

***Step two: Build a form data set***

A *form data set* is a sequence of control-name/current-value pairs constructed from successful controls

***Step three: Encode the form data set***

The form data set is then encoded according to the content type specified by the enctype attribute of the FORM element.

***Step four: Submit the encoded form data set***

Finally, the encoded data is sent to the processing agent designated by the action attribute using the protocol specified by the method attribute.

This specification does not specify all valid submission methods or content types that may be used with forms. However, HTML 4 user agents must support the established conventions in the following cases:

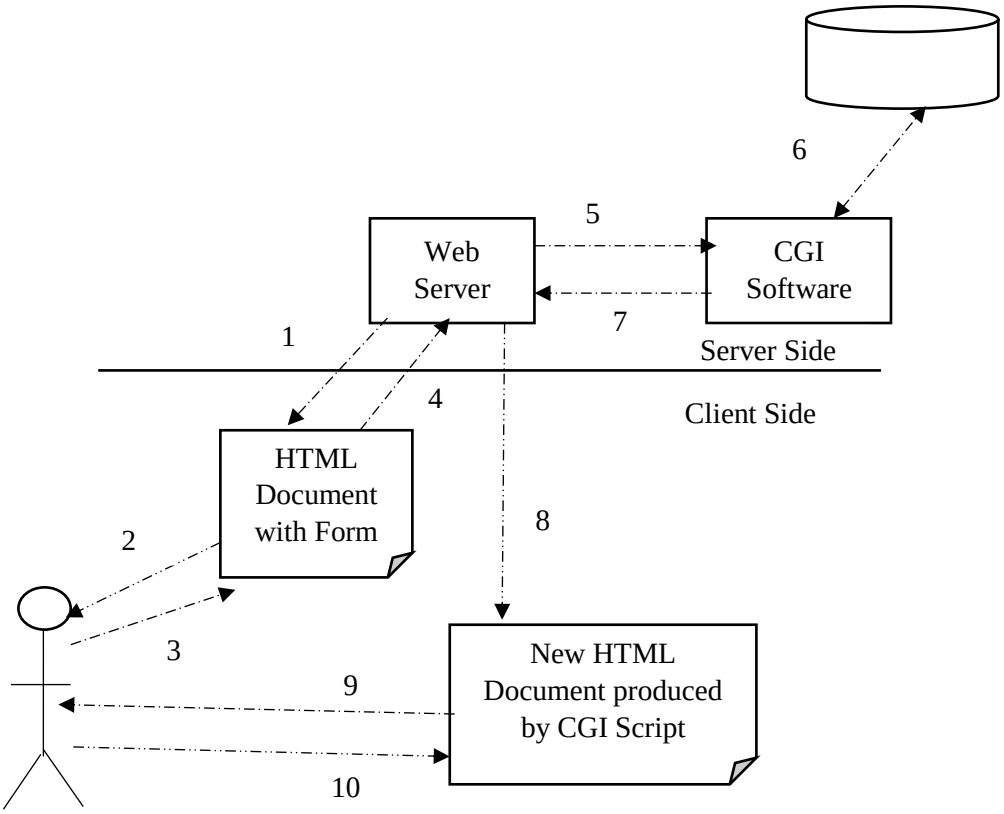
- If the method is "get" and the action is an HTTP URI, the user agent takes the value of action, appends a `?' to it, then appends the form data set, encoded using the "application/x-www-form-urlencoded" content type. The user agent then traverses the link to this URI. In this scenario, form data are restricted to ASCII codes.
- If the method is "post" and the action is an HTTP URI, the user agent conducts an HTTP "post" transaction using the value of the action attribute and a message created according to the content type specified by the enctype attribute.

For any other value of action or method, behavior is unspecified.

User agents should render the response from the HTTP "get" and "post" transactions.

## Processing Forms

HTML forms supply a way for the user to interact with a Web server. The most widely used method to process the data submitted through a form is to send it to server-side software typically written in a scripting language. The figure below outlines the kind of processing that takes place.



1. The user retrieves a document containing a form from a Web server.
2. The user reads the Web page and interacts with the form it contains.
3. Submitting the form sends the form data to the server for processing.
4. The Web server passes the data to a CGI program.
5. The CGI software may use database information or store data in a server-side database.
6. The CGI software may generate a new Web page for the server to return to the user.
7. The user reads the new Web document and may interact with it.

Note:

- The elements used to create controls generally appear inside a FORM element, but may also appear outside of a FORM element declaration when they are used to build user interfaces.
- Note that controls outside a form cannot be successful controls.
- When a form is submitted for processing, some controls have their name paired with their current value and these pairs are submitted with the form. Those controls for which name/value pairs are submitted are called **successful controls**.

**Assignment**

Job Application Form

Create an on-line job application form. The application form is for a computer company called SpeedyPC who are advertising for computer programmers. Your form's action should post the application to the email address: benardesadia@gmail.com.

Your application form must have the following elements:

– Position applied for (autofocus), name, nationality, date of birth (selected from an auto picker), address (in a text area), telephone number and email (required).

– Educational history and qualifications.

– Work experience/employment/training in terms of employer history and number of years of experience selected from a slider. Set maximum years of experience to 10 years.

– Personal statement.

– Two referees including names, occupation, relationship, address, telephone

# HTML – HEADER

## 1. Introduction

The <head> tag is a container of various important tags like <title>, <meta>, <link>, <base>, <style>, <script>, and <noscript> tags.

## 2. The HTML <title> Tag

The HTML <title> tag is used for specifying the title of the HTML document.

## 3. The HTML <meta> Tag

The HTML <meta> tag is used to provide metadata about the HTML document which includes information about page expiry, page author, list of keywords, page description etc.

## 4. The HTML <base> Tag

The HTML <base> tag is used for specifying the base URL for all relative URLs in a page, which means all the other URLs will be concatenated into base URL while locating for the given item. For example, all the given pages and images will be searched after prefixing the given URLs with base URL http://www.tutorialspoint.com/ directory:

```
<html>
<head>
      <title>HTML Base Tag Example</title>
      <base href="http://www.tutorialspoint.com/" />
</head>
<body>
      <img src="/images/logo.png" alt="Logo Image"/>
      <a href="/html/index.htm" title="HTML Tutorial"/>HTML Tutorial</a>
</body>
</html>
```

## 5. The HTML <link> Tag

The HTML <link> tag is used to specify relationships between the current document and external resource. Following is an example to link an external style sheet file available in css sub-directory within web root:

```
<html>
<head>
      <title>HTML link Tag Example</title>
      <base href="http://www.tutorialspoint.com/" />
      <link rel="stylesheet" type="text/css" href="/css/style.css">
</head>
<body>
<p>Hello, World!</p>
</body>
</html>
```

## 6. The HTML <style> Tag

The HTML <style> tag is used to specify style sheet for the current HTML document.

Following is an example to define few style sheet rules inside <style> tag:

```
<html>
      <head>
      <title>HTML style Tag Example</title>
      <base href="http://www.tutorialspoint.com/" />
      <style type="text/css">
            .myclass{
            background-color: #aaa;
            padding: 10px;
            }
      </style>
</head>
<body>
<p class="myclass">Hello, World!</p>
</body>
</html>
```

## 7. The HTML <script> Tag

The HTML <script> tag is used to include either external script file or to define internal script for the HTML document. Following is an example where we are using JavaScript to define a JavaScript function

```
<html>
<head>
      <title>HTML script Tag Example</title>
      <base href="http://www.tutorialspoint.com/" />
      <script type="text/JavaScript">
            function Hello(){
            alert("Hello, World");
            }
      </script>
</head>
<body>
      <input type="button" onclick="Hello();" name="ok" value="OK"
/>
</body>
</html>
```

## 8. HTML <noscript> Tag

The `<noscript>` tag defines an alternate content to be displayed to users that have disabled scripts in their browser or have a browser that doesn't support script.

```
<html>
   <head>
      <title>HTML noscript Tag</title>
   </head>
   <body>
      <script type = "text/JavaScript">
         <!--
            document.write("Hello JavaScript!")
         -->
      </script>
```

```
      <noscript>
         Your browser does not support JavaScript!
      </noscript>
   </body>
</html>
```

# HTML – STYLE SHEET

## 1. Introduction

– CSS stands for Cascading Style Sheets
– CSS describes how HTML elements are to be displayed on screen, paper, or in other media
– CSS saves a lot of work. It can control the layout of multiple web pages all at once
– External stylesheets are stored in CSS files

## 2. Why Use CSS?

– CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

## 3. Three ways of inserting a style sheet / How to Add CSS

i) External CSS
ii) Internal CSS
iii) Inline CSS

## 4. External CSS

– With an external style sheet, you can change the look of an entire website by changing just one file!
– External styles are defined within the <link> element, inside the <head> section of an HTML page:
– Example

### i) styles.html file

```
<html>
      <head>
            <link rel="stylesheet" href="mystyle.css">
      </head>
<body>
      <h1>This is a heading</h1>
      <p>This is a paragraph.</p>
</body>
</html>
```

### ii) mystyle.css file

An external style sheet can be written in any text editor, and must be saved with a .css extension.

```
body {
  background-color: lightblue;
}
```

```
h1 {
  color: navy;
  margin-left: 20px;
}
```

### 5. Internal CSS

– An internal style sheet may be used if one single HTML page has a unique style.
– Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

```
<html>
<head>
      <style>
            body {
              background-color: linen;
            }
            h1 {
              color: maroon;
              margin-left: 40px;
            }
      </style>
</head>
<body>
      <h1>This is a heading</h1>
      <p>This is a paragraph.</p>
</body>
</html>
```

### 6. Inline CSS

– An inline style may be used to apply a unique style for a single element.
– To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

```
<html>
<body>
      <h1 style="color:blue;text-align:center;">
            This is a heading
      </h1>
      <p style="color:red;">This is a paragraph.</p>
</body>
</html>
```

– An inline style loses many of the advantages of a style sheet (by mixing content with presentation).

### Example

### Index.html File

```
<html>
<head>
  <meta charset="UTF-8">
  <title>Pure CSS DropDown Menu</title>
  <link rel="stylesheet" href="style.css">
```

```
</head>
<body>
<div id="container">
    <nav>
        <ul>
            <li><a href="#">Home </a> </li>
            <li><a href="#">Web Design </a>
               <ul>
                     <li><a href="#">HTML </a> </li>
               <li><a href="#">CSS </a> </li>
                           <li><a href="#">JavaScript </a> </li>
              </ul>
            </li>
            <li><a href="#">Web Development</a>
            <ul>
                      <li><a href="#">PHP</a></li>
                <li><a href="#">Node JS</a></li>
                <li><a href="#">Advance JS</a>
               <ul>
                    <li><a href="#">Angular</a></li>
                    <li><a href="#">VUE</a></li>
                    <li><a href="#">React</a>
                        <ul>
                            <li><a href="#">React Native</a></li>
                            <li><a href="#">React JS</a></li>
                            <li><a href="#">Material Design</a></li>
                        </ul>
                    </li>
                     </ul>
                </li>
               </ul>
            </li>
            <li><a href="#">Graphic Design</a>
            <ul>
                <li><a href="#">Photoshop</a></li>
                <li><a href="#">Illustrator</a></li>
                <li><a href="#">Adobe XD</a></li>
               </ul>
            </li>
            <li><a href="#">Contact</a></li>
            <li><a href="#">About</a></li>
        </ul>
         </nav>
  <h1>Simple <span>CSS</span> DropDown Menu</h1>
 </div>
</body>
</html>
```

**Style.css File**

```
body {
   background: #212121;
   font-size:22px;
   line-height: 32px;
   color: #ffffff;
   word-wrap:break-word !important;
         font-family: 'Quicksand', sans-serif;
   margin: 0;
```

```
   padding: 0;
}
h1 {
   font-size: 60px;
   text-align: center;
   color: #FFF;
   margin-top: 150px;
         font-family: 'Russo One', sans-serif;
 }
h1 span {
   color: #FF4649;
}
#container {
   margin: 0 auto;
}
nav {
   margin: 35px 0;
   background-color: #FF4649;
}
nav ul {
   padding: 0;
      margin: 0;
   list-style: none;
   position: relative;
 }
nav ul li {
   display:inline-block;
   background-color: #FF4649;
 }
nav a {
   display:block;
   padding:0 10px;
   color:#FFF;
   font-size:20px;
   line-height: 60px;
   text-decoration:none;
}
nav a:hover {
   background-color: #333;
}
nav ul ul {
   display: none;
   position: absolute;
   top: 60px;
}
nav ul li:hover > ul {
   display:inherit;
}
nav ul ul li {
   width:230px;
   float:none;
   display:list-item;
   position: relative;
}
nav ul ul ul li {
   position: relative;
   top:-60px;
```

```
   left:230px;
}
nav ul ul li {
   border: 1px solid white;
}
li > a:after {
      content:  ' ▼';
}
li > a:only-child:after {
      content: '';
      }
```

# JAVASCRIPT

# Introduction to JavaScript

## 1. Introduction

- **JavaScript** is a lightweight, interpreted **programming** language with object-oriented capabilities. It is designed for creating network-centric applications.
- It is open and cross-platform.
- It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to **interact** with the user and make **dynamic** pages.
- Scripts offer authors a means to extend HTML documents in highly active and interactive ways.

## 2. Why Learn JavaScript

Some of the advantages of learning JavaScript:

1. JavaScript has the capability of being used to develop both front-end and back-end parts of a website.
2. JavaScript is everywhere, it comes installed on every modern web browser and so to learn JavaScript you really do not need any special environment setup. For example Chrome, Mozilla Firefox , Safari and every browser you know as of today, supports JavaScript.
3. JavaScript helps you create really beautiful and crazy fast websites with the best Graphical User Experience.
4. JavaScript usage has now extended to mobile app development, desktop app development, and game development. This opens many opportunities for you as JavaScript Programmer.
5. Great thing about JavaScript is that you will find tons of frameworks and Libraries already developed which can be used directly in your software development to reduce your time to market.

There are many useful JavaScript frameworks and libraries available:

- Angular
- React
- jQuery
- Vue.js
- Ext.js
- Ember.js
- Meteor
- Mithril
- Node.js
- Polymer
- Aurelia
- Backbone.js

## 3. Applications of JavaScript Programming

- **Client side validation** - Verifies any user input before submitting it to the server.
- **Manipulating HTML Pages** - JavaScript helps in manipulating HTML page on the fly. This helps in adding and deleting any HTML tag very easily using

JavaScript and modify your HTML to change its look and feel based on different devices and requirements.

- **User Notifications** - You can use JavaScript to raise dynamic pop-ups on the webpages to give different types of notifications to your website visitors.
- **Back-end Data Loading** - JavaScript provides Ajax library which helps in loading back-end data while you are doing some other processing. This really gives an amazing experience to your website visitors.
- **Presentations** - JavaScript also provides the facility of creating presentations which gives website look and feel. JavaScript provides RevealJS and BespokeJS libraries to build a web-based slide presentation.
- **Server Applications** - Node JS is built on Chrome's Javascript runtime for building fast and scalable network applications. This is an event based library which helps in developing very sophisticated server applications including Web Servers.

## 4. Client-Side JavaScript

- The script is included in or referenced by an HTML document for the code to be interpreted by the browser.
- It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.
- For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field. The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

## 5. Advantages of JavaScript

The merits of using JavaScript are −

- **Less Server Interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased Interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer Interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

## 6. JavaScript Development Tools

You can start with a simple text editor such as Notepad.

Other editing tools include −

- **Microsoft FrontPage** − FrontPage provides web developers with a number of JavaScript tools to assist in the creation of interactive websites.

- **Macromedia Dreamweaver MX** − It provides several handy prebuilt JavaScript components, integrates well with databases, and conforms to new standards such as XHTML and XML.

## 7. JavaScript - Syntax

- JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.
- You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.
  - Script in <head>...</head> section.
  - Script in <body>...</body> section.
  - Script in <body>...</body> and <head>...</head> sections.
  - Script in an external file and then include in <head>...</head> section.
- The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

```
<script ...>
  JavaScript code
</script>
```

- The script tag can take two attributes−

```
<script language = "javascript" type = "text/javascript">
         JavaScript code
</script>
```

- Type attribute is a must.

**Example:**

```
<html>
   <body>
      <script language = "javascript" type = "text/javascript">
         <!--
            document.write("Hello World!");
         //-->
      </script>
   </body>
</html>
```

## 8. Statements in JavaScript

- Statements in JavaScript are generally followed by a semicolon character,
- JavaScript can allow you to omit this semicolon if each of your statements are placed on a separate line:

**Example:**

```
<script language = "javascript" type = "text/javascript">
   <!--
      var1 = 10
      var2 = 20
   //-->
</script>
```

- **Note − It is a good programming practice to use semicolons.**
- But when formatted in a single line as follows, you must use semicolons −

```
<script language = "javascript" type = "text/javascript">
   <!--
      var1 = 10; var2 = 20;
   //-->
</script>
```

## 9. Case Sensitivity

- JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.
- So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.
- **NOTE** − Care should be taken while writing variable and function names in JavaScript.

## 10. Comments in JavaScript

- Single Line Comment: Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Multiple Line Comment: Any text between the characters /* and */ is treated as a comment.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

**Example**

```
<script language = "javascript" type = "text/javascript">
   <!--
      // This is a comment. It is similar to comments in C++

      /*
      * This is a multi-line comment in JavaScript
      * It is very similar to comments in C Programming
      */
   //-->
```

```
</script>
```

## 11.Non-JavaScript Browsers

– You can add a **noscript** block immediately after the script block as follows −

```
<html>
   <body>
      <script language = "javascript" type = "text/javascript">
         <!--
            document.write("Hello World!")
         //-->
      </script>

      <noscript>
         Sorry...JavaScript is needed to go ahead.
      </noscript>
   </body>
</html>
```

– Now, if the user's browser does not support JavaScript or JavaScript is not enabled, then the message from </noscript> will be displayed on the screen.

## 12. Examples

### Example 1

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function sayHello() {
               alert("Hello World")
            }
         //-->
      </script>
   </head>

   <body>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
   </body>
</html>
```

### Example 2

```
<html>
   <head>
   </head>

   <body>
      <script type = "text/javascript">
         <!--
```

```
                document.write("Hello World")
             //-->
          </script>

          <p>This is web page body </p>
       </body>
    </html>
```

**Example 3**

```
    <html>
       <head>
          <script type = "text/javascript">
             <!--
                function sayHello() {
                   alert("Hello World")
                }
             //-->
          </script>
       </head>

       <body>
          <script type = "text/javascript">
             <!--
                document.write("Hello World")
             //-->
          </script>

          <input type = "button" onclick = "sayHello()" value = "Say
    Hello" />
       </body>
    </html>
```

**Example 4**

**JavaScript in External File**

Allows reusing identical JavaScript code on multiple pages of a site.

Include the external JavaScript file in your HTML code using **script** tag and its **src** attribute.

```
<html>
   <head>
      <script type = "text/javascript" src = "filename.js" ></script>
   </head>

   <body>
      .......
   </body>
</html>
```

The content in **filename.js** file:

```
function sayHello() {
   alert("Hello World")
}
```

# JavaScript - Variables

## 1. Datatypes

JavaScript allows you to work with three primitive data types −

- **Numbers,** eg. 123, 120.50 etc.
- **Strings** of text e.g. "This text string" etc.
- **Boolean** e.g. true or false.

JavaScript also defines two trivial data types, **null** and **undefined,** each of which defines only a single value.

In addition to these primitive data types, JavaScript supports a composite data type known as **object**. We will cover objects in detail in a separate chapter.

## 2. Variables

Variables are declared with the **var** keyword as follows.

```
<script type = "text/javascript">
   <!--
      var money;
      var name;
   //-->
</script>
```

You can also declare multiple variables with the same **var** keyword as follows −

```
<script type = "text/javascript">
   <!--
      var money, name;
   //-->
</script>
```

Storing a value in a variable is called **variable initialization**. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

```
<script type = "text/javascript">
   <!--
      var name = "Ali";
      var money;
      money = 2000.50;
   //-->
```

```
</script>
```

**Note** − Use the **var** keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

JavaScript is **untyped** language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

## 3. JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

- **Global Variables** − A global variable has global scope which means it can be defined anywhere in your JavaScript code.
- **Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable with the same name. If you declare a local variable or function parameter with the same name as a global variable, you effectively hide the global variable. Take a look into the following example.

```
<html>
   <body onload = checkscope();>
      <script type = "text/javascript">
         <!--
            var myVar = "global";      // Declare a global variable
            function checkscope( ) {
               var myVar = "local";    // Declare a local variable
               document.write(myVar);
            }
         //-->
      </script>
   </body>
</html>
```

## 4. JavaScript Variable Names

While naming your variables in JavaScript, keep the following rules in mind.

- You should not use any of the JavaScript reserved keywords as a variable name. These keywords are mentioned in the next section. For example, **break** or **boolean** variable names are not valid.

- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, **123test** is an invalid variable name but **_123test** is a valid one.
- JavaScript variable names are case-sensitive. For example, **Name** and **name** are two different variables.

## 5. JavaScript Reserved Words

A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

| | | | | | | |
|---|---|---|---|---|---|---|
| abstract | else | instanceof | switch | continue | implements | short |
| boolean | enum | int | synchronized | debugger | import | static |
| break | export | interface | this | default | in | super |
| byte | extends | long | throw | delete | protected | while |
| case | false | native | throws | do | public | with |
| catch | final | new | transient | double | return | volatile |
| char | finally | null | true | function | var | |
| class | float | package | try | goto | void | |
| const | for | private | typeof | if | | |

## 6. JavaScript - Operators

### i) Arithmetic Operators

– JavaScript supports the following arithmetic operators −
– Assume variable A holds 10 and variable B holds 20, then −

| Sr.No. | Operator & Description |
|---|---|
| 1 | **+ (Addition) Adds** two operands. **Eg:** A + B will give 30 |
| 2 | **- (Subtraction)** Subtracts the second operand from the first. **Eg:** A - B will give -10 |
| 3 | *** (Multiplication)** Multiply both operands. **Eg:** A * B will give 200 |
| 4 | **/ (Division)** Divide the numerator by the denominator. **Ex:** B / A will give 2 |
| 5 | **% (Modulus)** Outputs the remainder of an integer division. **Ex:** B % A will give 0 |
| 6 | **++ (Increment).** Increases an integer value by one. **Ex:** A++ will give 11 |
| 7 | **-- (Decrement).** Decreases an integer value by one. **Ex:** A-- will give 9 |

**Note** − Addition operator (+) works for Numeric as well as Strings. e.g. "a" + 10 will give "a10".

**Example**

```
<html>
   <body>

      <script type = "text/javascript">
         <!--
            var a = 33;
            var b = 10;
            var c = "Test";
            var linebreak = "<br />";

            document.write("a + b = ");
            result = a + b;
            document.write(result);
            document.write(linebreak);

            document.write("a - b = ");
            result = a - b;
            document.write(result);
            document.write(linebreak);

            document.write("a / b = ");
            result = a / b;
            document.write(result);
            document.write(linebreak);

            document.write("a % b = ");
            result = a % b;
            document.write(result);
            document.write(linebreak);

            document.write("a + b + c = ");
            result = a + b + c;
            document.write(result);
            document.write(linebreak);

            a = ++a;
            document.write("++a = ");
            result = ++a;
            document.write(result);
            document.write(linebreak);

            b = --b;
            document.write("--b = ");
            result = --b;
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      Set the variables to different values and then try...
```

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

### ii) Comparison Operators

– Assume variable A holds 10 and variable B holds 20, then −

| Sr.No. | Operator & Description |
|---|---|
| 1 | **= = (Equal).** Checks if the value of two operands are equal or not, if yes, then the condition becomes true. **Eg:** (A == B) is not true. |
| 2 | **!= (Not Equal)** Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. **Eg:** (A != B) is true. |
| 3 | **> (Greater than).** Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. **Eg:** (A > B) is not true. |
| 4 | **< (Less than).** Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. **Eg:** (A < B) is true. |
| 5 | **>= (Greater than or Equal to).** Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. **Eg:** (A >= B) is not true. |
| 6 | **<= (Less than or Equal to).** Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. **Eg:** (A <= B) is true. |

## Example

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var a = 10;
            var b = 20;
            var linebreak = "<br />";

            document.write("(a == b) => ");
            result = (a == b);
            document.write(result);
            document.write(linebreak);

            document.write("(a < b) => ");
            result = (a < b);
            document.write(result);
            document.write(linebreak);

            document.write("(a > b) => ");
            result = (a > b);
            document.write(result);
```

```
            document.write(linebreak);

            document.write("(a != b) => ");
            result = (a != b);
            document.write(result);
            document.write(linebreak);

            document.write("(a >= b) => ");
            result = (a >= b);
            document.write(result);
            document.write(linebreak);

            document.write("(a <= b) => ");
            result = (a <= b);
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      Set the variables to different values and different
operators and then try...
   </body>
</html>
```

### iii) Logical Operators

– Assume variable A holds 10 and variable B holds 20, then −

| Sr.No. | Operator & Description |
|---|---|
| 1 | **&& (Logical AND).** If both the operands are non-zero, then the condition becomes true. **Eg:** (A && B) is true. |
| 2 | **|| (Logical OR).** If any of the two operands are non-zero, then the condition becomes true. **Eg:** (A || B) is true. |
| 3 | **! (Logical NOT).** Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. **Eg:** ! (A && B) is false. |

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var a = true;
            var b = false;
            var linebreak = "<br />";

            document.write("(a && b) => ");
            result = (a && b);
            document.write(result);
            document.write(linebreak);

            document.write("(a || b) => ");
            result = (a || b);
```

```
            document.write(result);
            document.write(linebreak);

            document.write("!(a && b) => ");
            result = (!(a && b));
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      <p>Set the variables to different values and
different operators and then try...</p>
   </body>
</html>
```

### iv) Bitwise Operators

- JavaScript supports the following bitwise operators −
- Assume variable A holds 2 and variable B holds 3, then −

| Sr.No. | Operator & Description |
|---|---|
| 1 | **& (Bitwise AND).** It performs a Boolean AND operation on each bit of its integer arguments. **Eg:** (A & B) is 2. |
| 2 | **\| (BitWise OR).** It performs a Boolean OR operation on each bit of its integer arguments. **Eg:** (A \| B) is 3. |
| 3 | **^ (Bitwise XOR).** It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both. **Eg:** (A ^ B) is 1. |
| 4 | **~ (Bitwise Not).** It is a unary operator and operates by reversing all the bits in the operand. **Eg:** (~B) is -4. |
| 5 | **<< (Left Shift). Eg:** (A << 1) is 4. |
| 6 | **>> (Right Shift)** Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand. **Eg:** (A >> 1) is 1. |
| 7 | **>>> (Right shift with Zero).** This operator is just like the >> operator, except that the bits shifted in on the left are always zero. **Ex:** (A >>> 1) is 1. |

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var a = 2; // Bit presentation 10
            var b = 3; // Bit presentation 11
            var linebreak = "<br />";

            document.write("(a & b) => ");
            result = (a & b);
            document.write(result);
            document.write(linebreak);
```

```
            document.write("(a | b) => ");
            result = (a | b);
            document.write(result);
            document.write(linebreak);

            document.write("(a ^ b) => ");
            result = (a ^ b);
            document.write(result);
            document.write(linebreak);

            document.write("(~b) => ");
            result = (~b);
            document.write(result);
            document.write(linebreak);

            document.write("(a << b) => ");
            result = (a << b);
            document.write(result);
            document.write(linebreak);

            document.write("(a >> b) => ");
            result = (a >> b);
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      <p>Set the variables to different values and
different operators and then try...</p>
   </body>
</html>
```

### v) Assignment Operators

– JavaScript supports the following assignment operators −

| Sr.No. | Operator & Description |
|---|---|
| 1 | **= (Simple Assignment)** Assigns values from the right side operand to the left side operand<br><br>**Eg:** C = A + B will assign the value of A + B into C |
| 2 | **+= (Add and Assignment)**. It adds the right operand to the left operand and assigns the result to the left operand.<br><br>**Eg:** C += A is equivalent to C = C + A |
| 3 | **−= (Subtract and Assignment)**. It subtracts the right operand from the left operand and assigns the result to the left operand.<br><br>**Eg:** C -= A is equivalent to C = C - A |

| | |
|---|---|
| 4 | ***= (Multiply and Assignment).** It multiplies the right operand with the left operand and assigns the result to the left operand.<br><br>**Ex:** C *= A is equivalent to C = C * A |
| 5 | **/= (Divide and Assignment).** It divides the left operand with the right operand and assigns the result to the left operand.<br><br>**Ex:** C /= A is equivalent to C = C / A |
| 6 | **%= (Modules and Assignment).** It takes modulus using two operands and assigns the result to the left operand.<br><br>**Ex:** C %= A is equivalent to C = C % A |

**Note** − Same logic applies to Bitwise operators so they will become like <<=, >>=, >>=, &=, |= and ^=.

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var a = 33;
            var b = 10;
            var linebreak = "<br />";

            document.write("Value of a => (a = b) => ");
            result = (a = b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a += b) => ");
            result = (a += b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a -= b) => ");
            result = (a -= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a *= b) => ");
            result = (a *= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a /= b) => ");
            result = (a /= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a %= b) => ");
```

```
            result = (a %= b);
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      <p>Set the variables to different values and different
operators and then try...</p>
   </body>
</html>
```

### vi) Miscellaneous Operator

– The **conditional operator** (? :) and the **typeof operator**.

**Conditional Operator (? :)**

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

| Sr.No. | Operator and Description |
|---|---|
| 1 | **? : (Conditional )**<br><br>If Condition is true? Then value X : Otherwise value Y |

**Example**

Try the following code to understand how the Conditional Operator works in JavaScript.

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var a = 10;
            var b = 20;
            var linebreak = "<br />";

            document.write ("((a > b) ? 100 : 200) => ");
            result = (a > b) ? 100 : 200;
            document.write(result);
            document.write(linebreak);

            document.write ("((a < b) ? 100 : 200) => ");
            result = (a < b) ? 100 : 200;
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      <p>Set the variables to different values and different
operators and then try...</p>
```

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

**typeof Operator**

- The **typeof** operator is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.
- The *typeof* operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.
- Here is a list of the return values for the **typeof** Operator.

| Type | String Returned by typeof |
|---|---|
| Number | "number" |
| String | "string" |
| Boolean | "boolean" |
| Object | "object" |
| Function | "function" |
| Undefined | "undefined" |
| Null | "object" |

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var a = 10;
            var b = "String";
            var linebreak = "<br />";

            result = (typeof b == "string" ? "B is String" : "B is
            Numeric");
            document.write("Result => ");
            document.write(result);
            document.write(linebreak);

            result = (typeof a == "string" ? "A is String" : "A is
            Numeric");
            document.write("Result => ");
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      <p>Set the variables to different values and different operators
and then try...</p>
   </body>
</html>
```

## 7. JavaScript Special Characters

In JavaScript you can add special characters to a text string by using the backslash sign.

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

```
var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);
document.write ("You \& I are singing!");
```

The table below lists other special characters that can be added to a text string with the backslash sign:

| Code | Outputs |
|---|---|
| \' | single quote |
| \" | double quote |
| \& | ampersand |
| \\ | backslash |
| \n | new line |
| \r | carriage return |
| \t | tab |
| \b | backspace |
| \f | form feed |

## 8. JavaScript - if...else Statement

A conditional statement that allow your program to make correct decisions and perform right actions.

JavaScript supports the following forms of **if..else** statement −

- if statement
- if...else statement
- if...else if... statement.

### i) if statement

Allows JavaScript to make decisions and execute statements conditionally.

**Syntax**

```
if (expression) {
  Statement(s) to be executed if expression is true
}
```

**Example**

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var age = 20;

        if( age > 18 ) {
          document.write("<b>Qualifies for driving</b>");
        }
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

**ii) if...else statement**

**Syntax**

```
if (expression) {
  Statement(s) to be executed if expression is true
} else {
  Statement(s) to be executed if expression is false
}
```

**Example**

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var age = 15;

        if( age > 18 ) {
          document.write("<b>Qualifies for driving</b>");
        } else {
          document.write("<b>Does not qualify for driving</b>");
        }
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

### iii) if...else if... statement

The **if...else if...** statement is an advanced form of **if…else** that allows JavaScript to make a correct decision out of several conditions.

**Syntax**

```
if (expression 1) {
  Statement(s) to be executed if expression 1 is true
} else if (expression 2) {
  Statement(s) to be executed if expression 2 is true
} else if (expression 3) {
  Statement(s) to be executed if expression 3 is true
} else {
  Statement(s) to be executed if no expression is true
}
```

**Example**

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var book = "maths";
        if( book == "history" ) {
          document.write("<b>History Book</b>");
        } else if( book == "maths" ) {
          document.write("<b>Maths Book</b>");
        } else if( book == "economics" ) {
          document.write("<b>Economics Book</b>");
        } else {
          document.write("<b>Unknown Book</b>");
        }
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
<html>
```

## 9. JavaScript - Switch Case

You can use multiple **if...else…if** statements, to perform a multiway branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable. The switch statement handles exactly the same situations handled by repeated **if...else if** statements.

**Syntax.**

```
switch (expression) {
   case condition 1: statement(s)
   break;

   case condition 2: statement(s)
   break;
   ...

   case condition n: statement(s)
   break;

   default: statement(s)
}
```

The **break** statements indicate the end of a particular case. If they were omitted, the interpreter would continue executing each statement in each of the following cases.

**Example**

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var grade = 'A';
            document.write("Entering switch block<br />");
            switch (grade) {
               case 'A': document.write("Good job<br />");
               break;

               case 'B': document.write("Pretty good<br />");
               break;

               case 'C': document.write("Passed<br />");
               break;

               case 'D': document.write("Not so good<br />");
               break;

               case 'F': document.write("Failed<br />");
               break;

               default:  document.write("Unknown grade<br />")
            }
```

```
            document.write("Exiting switch block");
         //-->
      </script>
      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

Break statements play a major role in switch-case statements. Try the following code that uses switch-case statement without any break statement.

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var grade = 'A';
            document.write("Entering switch block<br />");
            switch (grade) {
               case 'A': document.write("Good job<br />");
               case 'B': document.write("Pretty good<br />");
               case 'C': document.write("Passed<br />");
               case 'D': document.write("Not so good<br />");
               case 'F': document.write("Failed<br />");
               default: document.write("Unknown grade<br />")
            }
            document.write("Exiting switch block");
         //-->
      </script>
      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

**Example**

```
<script type="text/javascript">
       var score=prompt("enter the score");
       switch(true){
        case ((score>=80)&& (score<=99)):
              document.write("Grade = A Excellence Performance");
              break;
        case ((score>=60) &&(score<=79)):
              document.write("Grade = B Above Average");
              break;
```

```
      case ((score>=40) &&(score<=59)):
            document.write("Grade = C Average");
            break;
      case ((score>=30) &&(score<=39)):
            document.write("Grade = D Below Average");
            break;
      case ((score>=02) &&(score<=19)):
            document.write("Grade = E Poor");
            break;
      default:
            document.write(" Re-Enter the score" );
      }
</script>
```

## 10.JavaScript - While Loops

### i) The While loop

The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is **TRUE**. Once the expression becomes **FALSE,** the loop terminates.

The syntax

```
while (expression) {
   Statement(s) to be executed if expression is true
}
```

**Example**

```
<html>
   <body>

      <script type = "text/javascript">
         <!--
            var count = 0;
            document.write("Starting Loop ");

            while (count < 10) {
               document.write("Current Count : " + count + "<br />");
               count++;
            }

            document.write("Loop stopped!");
         //-->
      </script>

      <p>Set the variable to different value and then try...</p>
```

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

### ii) The do...while Loop

The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is **false**.

syntax

```
do {
   Statement(s) to be executed;
} while (expression);
```

**Note** − Don't miss the semicolon used at the end of the **do...while** loop.

**Example**

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var count = 0;
            document.write("Starting Loop" + "<br />");
            do {
               document.write("Current Count : " + count + "<br />");
               count++;
            }
            while (count < 5);
            document.write ("Loop stopped!");
         //-->
      </script>
      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

## 11.JavaScript - For Loop

### i) For Loop

The '**for**' loop is the most compact form of looping. It includes the following three important parts −

- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.

- The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

You can put all the three parts in a single line separated by semicolons.

The syntax of **for** loop is JavaScript is as follows −

```
for (initialization; test condition; iteration statement) {
   Statement(s) to be executed if test condition is true
}
```

**Example**

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var count;
            document.write("Starting Loop" + "<br />");

            for(count = 0; count < 10; count++) {
               document.write("Current Count : " + count );
               document.write("<br />");
            }
            document.write("Loop stopped!");
         //-->
      </script>
      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

## ii) JavaScript - Loop Control

JavaScript provides full control to handle loops and switch statements. There may be a situation when you need to come out of a loop without reaching its bottom. There may also be a situation when you want to skip a part of your code block and start the next iteration of the loop.

To handle all such situations, JavaScript provides **break** and **continue** statements. These statements are used to immediately come out of any loop or to start the next iteration of any loop respectively.

### a) The break Statement

The **break** statement, is used to exit a loop early, breaking out of the enclosing curly braces.

**Example**

The following example illustrates the use of a **break** statement with a while loop. Notice how the loop breaks out early once **x** reaches 5 and reaches to **document.write (..)** statement just below to the closing curly brace −

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
         var x = 1;
         document.write("Entering the loop<br /> ");

         while (x < 20) {
            if (x == 5) {
               break;   // breaks out of loop completely
            }
            x = x + 1;
            document.write( x + "<br />");
         }
         document.write("Exiting the loop!<br /> ");
         //-->
      </script>

      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

We already have seen the usage of **break** statement inside **a switch** statement.

### b) The continue Statement

The **continue** statement tells the interpreter to immediately start the next iteration of the loop and skip the remaining code block. When a **continue** statement is encountered, the program flow moves to the loop check expression immediately and if the condition remains true, then it starts the next iteration, otherwise the control comes out of the loop.

**Example**

```
   <html>
      <body>
         <script type = "text/javascript">
            <!--
               var x = 1;
               document.write("Entering the loop<br /> ");

               while (x < 10) {
                  x = x + 1;

                  if (x == 5) {
                     continue;   // skip rest of the loop body
                  }
                  document.write( x + "<br />");
```

```
            }
            document.write("Exiting the loop!<br /> ");
         //-->
      </script>
      <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

# JavaScript - Functions

## 1. Introduction

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

## 2. Function Definition

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

The basic syntax is shown here.

```
<script type = "text/javascript">
   <!--
      function functionname(parameter-list) {
         statements
      }
   //-->
</script>
```

**Example**

```
<script type = "text/javascript">
   <!--
      function sayHello() {
         alert("Hello there");
      }
   //-->
</script>
```

## 3. Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
   <head>
      <script type = "text/javascript">
         function sayHello() {
            document.write ("Hello there!");
         }
      </script>

   </head>
```

```
   <body>
      <p>Click the following button to call the function</p>
      <form>
         <input type = "button" onclick = "sayHello()" value = "Say Hello">
      </form>
      <p>Use different text in write method and then try...</p>
   </body>
</html>
```

## 4. Function Parameters

A function can take multiple parameters separated by comma.

**Example**

```
<html>
   <head>
      <script type = "text/javascript">
         function sayHello(name, age) {
            document.write (name + " is " + age + " years old.");
         }
      </script>
   </head>

   <body>
      <p>Click the following button to call the function</p>
      <form>
         <input type = "button" onclick = "sayHello('Zara', 7)" value =
"Say Hello">
      </form>
      <p>Use different parameters inside the function and then
try...</p>
   </body>
</html>
```

## 5. The return Statement

A JavaScript function can have an optional **return** statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.

**Example**

```
<html>
   <head>
      <script type = "text/javascript">
         function concatenate(first, last) {
            var full;
            full = first + last;
            return full;
         }
         function secondFunction() {
```

```
            var result;
            result = concatenate('Zara', 'Ali');
            document.write (result );
         }
      </script>
   </head>

   <body>
      <p>Click the following button to call the function</p>
      <form>
         <input type = "button" onclick = "secondFunction()" value =
"Call Function">
      </form>
      <p>Use different parameters inside the function and then
try...</p>
  </body>
</html>
```

# JavaScript - Events

## 1. Introduction

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

## 2. onclick Event Type

This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

### Example

```
<html>
   <head>
      <script type = "text/javascript">
```

```
         <!--
            function sayHello() {
               alert("Hello World")
            }
         //-->
      </script>
   </head>

   <body>
      <p>Click the following button and see result</p>
      <form>
         <input type = "button" onclick = "sayHello()" value = "Say
Hello" />
      </form>
   </body>
</html>
```

## Adding Event Handlers

## Example:

```
<form>
       <input type="button" value ="touch me" onClick="alert('Hellooo
       World');prompt('Enter Name');confirm('are you sure')">
</form>
```

**Example**

```
<a href = "http//google.com" onMouseOver="alert('Game Time')"> Hover over me </a>

<br><br>

<a href = "http//google.com" onMouseOut="alert('Come Back!!')"> Hover over me </a>
```

**Example**

```
<body onLoad="alert('Your Website has Loaded ')">
        <H1> Onload Event Handler</h1>
</body>
```

**Example**

```
<body onUnload="alert('Goodbye ');">
        <H1> Onload Event Handler</h1>
</body>
```

## 3. onsubmit Event Type

**onsubmit** is an event that occurs when you try to submit a form. You can put your form validation against this event type.

**Example**

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function validation() {
               all validation goes here
               .........
               return either true or false
            }
         //-->
      </script>
   </head>

   <body>
      <form method = "POST" action = "t.cgi" onsubmit = "return
validate()">
         .......
         <input type = "submit" value = "Submit" />
      </form>
   </body>
</html>
```

## 4. onmouseover and onmouseout

These two event types will help you create nice effects with images or even with text as well. The **onmouseover** event triggers when you bring your mouse over any element and the **onmouseout** triggers when you move your mouse out from that element. Try the following example.

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function over() {
               document.write ("Mouse Over");
            }
            function out() {
               document.write ("Mouse Out");
            }
         //-->
      </script>
   </head>

   <body>
      <p>Bring your mouse inside the division to see the result:</p>
      <div onmouseover = "over()" onmouseout = "out()">
         <h2> This is inside the division </h2>
      </div>
   </body>
</html>
```

# 5. HTML 5 Standard Events

The standard HTML 5 events are listed here for your reference. Here script indicates a Javascript function to be executed against that event.

| Attribute | Value | Description |
|---|---|---|
| Offline | script | Triggers when the document goes offline |
| Onabort | script | Triggers on an abort event |
| onafterprint | script | Triggers after the document is printed |
| onbeforeonload | script | Triggers before the document loads |
| onbeforeprint | script | Triggers before the document is printed |
| Onblur | script | Triggers when the window loses focus |
| oncanplay | script | Triggers when media can start play, but might has to stop for buffering |
| oncanplaythrough | script | Triggers when media can be played to the end, without stopping for buffering |
| onchange | script | Triggers when an element changes |
| Onclick | script | Triggers on a mouse click |
| oncontextmenu | script | Triggers when a context menu is triggered |
| ondblclick | script | Triggers on a mouse double-click |
| Ondrag | script | Triggers when an element is dragged |
| ondragend | script | Triggers at the end of a drag operation |
| ondragenter | script | Triggers when an element has been dragged to a valid drop target |
| ondragleave | script | Triggers when an element is being dragged over a valid drop target |
| ondragover | script | Triggers at the start of a drag operation |
| ondragstart | script | Triggers at the start of a drag operation |
| Ondrop | script | Triggers when dragged element is being dropped |
| ondurationchange | script | Triggers when the length of the media is changed |
| onemptied | script | Triggers when a media resource element suddenly becomes empty. |
| onended | script | Triggers when media has reach the end |
| onerror | script | Triggers when an error occur |
| onfocus | script | Triggers when the window gets focus |
| onformchange | script | Triggers when a form changes |
| onforminput | script | Triggers when a form gets user input |
| onhaschange | script | Triggers when the document has change |
| oninput | script | Triggers when an element gets user input |
| oninvalid | script | Triggers when an element is invalid |
| onkeydown | script | Triggers when a key is pressed |
| onkeypress | script | Triggers when a key is pressed and released |
| onkeyup | script | Triggers when a key is released |
| Onload | script | Triggers when the document loads |
| onloadeddata | script | Triggers when media data is loaded |

| Attribute | Value | Description |
|---|---|---|
| onloadedmetadata | script | Triggers when the duration and other media data of a media element is loaded |
| onloadstart | script | Triggers when the browser starts to load the media data |
| onmessage | script | Triggers when the message is triggered |
| onmousedown | script | Triggers when a mouse button is pressed |
| onmousemove | script | Triggers when the mouse pointer moves |
| onmouseout | script | Triggers when the mouse pointer moves out of an element |
| onmouseover | script | Triggers when the mouse pointer moves over an element |
| onmouseup | script | Triggers when a mouse button is released |
| onmousewheel | script | Triggers when the mouse wheel is being rotated |
| onoffline | script | Triggers when the document goes offline |
| Onoine | script | Triggers when the document comes online |
| ononline | script | Triggers when the document comes online |
| onpagehide | script | Triggers when the window is hidden |
| onpageshow | script | Triggers when the window becomes visible |
| onpause | script | Triggers when media data is paused |
| Onplay | script | Triggers when media data is going to start playing |
| onplaying | script | Triggers when media data has start playing |
| onpopstate | script | Triggers when the window's history changes |
| onprogress | script | Triggers when the browser is fetching the media data |
| onratechange | script | Triggers when the media data's playing rate has changed |
| onreadystatechange | script | Triggers when the ready-state changes |
| Onredo | script | Triggers when the document performs a redo |
| onresize | script | Triggers when the window is resized |
| onscroll | script | Triggers when an element's scrollbar is being scrolled |
| onseeked | script | Triggers when a media element's seeking attribute is no longer true, and the seeking has ended |
| onseeking | script | Triggers when a media element's seeking attribute is true, and the seeking has begun |
| onselect | script | Triggers when an element is selected |
| onstalled | script | Triggers when there is an error in fetching media data |
| onstorage | script | Triggers when a document loads |
| onsubmit | script | Triggers when a form is submitted |
| onsuspend | script | Triggers when the browser has been fetching media data, but stopped before the entire media file was fetched |
| ontimeupdate | script | Triggers when media changes its playing position |
| Onundo | script | Triggers when a document performs an undo |
| onunload | script | Triggers when the user leaves the document |
| onvolumechange | script | Triggers when media changes the volume, also when volume is set to "mute" |
| onwaiting | script | Triggers when media has stopped playing, but is expected to resume |

# JavaScript - Dialog Boxes

## 1. Introduction

JavaScript supports three important types of dialog boxes. These dialog boxes can be used to raise and alert, or to get confirmation on any input or to have a kind of input from the users.

## 2. Alert Dialog Box

An alert dialog box is mostly used to give a warning message to the users. For example, if one input field requires to enter some text but the user does not provide any input, then as a part of validation, you can use an alert box to give a warning message.

Nonetheless, an alert box can still be used for friendlier messages. Alert box gives only one button "OK" to select and proceed.

**Example**

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function Warn() {
               alert ("This is a warning message!");
               document.write ("This is a warning message!");
            }
         //-->
      </script>
   </head>

   <body>
      <p>Click the following button to see the result: </p>
      <form>
         <input type = "button" value = "Click Me" onclick = "Warn();" />
      </form>
   </body>
</html>
```

## 3. Confirmation Dialog Box

A confirmation dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: **OK** and **Cancel**.

If the user clicks on the OK button, the window method **confirm()** will return true. If the user clicks on the Cancel button, then **confirm()** returns false. You can use a confirmation dialog box as follows.

**Example**

```
<html>
   <head>
```

```
      <script type = "text/javascript">
         <!--
            function getConfirmation() {
               var retVal = confirm("Do you want to continue ?");
               if( retVal == true ) {
                  document.write ("User wants to continue!");
                  return true;
               } else {
                  document.write ("User does not want to continue!");
                  return false;
               }
            }
         //-->
      </script>
   </head>

   <body>
      <p>Click the following button to see the result: </p>
      <form>
         <input type = "button" value = "Click Me" onclick =
"getConfirmation();" />
      </form>
   </body>
</html>
```

## 4. Prompt Dialog Box

The prompt dialog box is very useful when you want to pop-up a text box to get user input. Thus, it enables you to interact with the user. The user needs to fill in the field and then click OK.

This dialog box is displayed using a method called **prompt()** which takes two parameters: (i) a label which you want to display in the text box and (ii) a default string to display in the text box.

This dialog box has two buttons: **OK** and **Cancel**. If the user clicks the OK button, the window method **prompt()** will return the entered value from the text box. If the user clicks the Cancel button, the window method **prompt()** returns **null**.

**Example**

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function getValue() {
               var retVal = prompt("Enter your name : ", "your name here");
               document.write("You have entered : " + retVal);
            }
         //-->
      </script>
   </head>
```

```
   <body>
      <p>Click the following button to see the result: </p>
      <form>
         <input type = "button" value = "Click Me" onclick =
"getValue();" />
      </form>
   </body>
</html>
```

## 5. JavaScript - Void Keyword

**void** is an important keyword in JavaScript which can be used as a unary operator that appears before its single operand, which may be of any type. This operator specifies an expression to be evaluated without returning a value.

The syntax of **void** can be either of the following two −

```
<head>
   <script type = "text/javascript">
      <!--
         void func()
         javascript:void func()
         or:
         void(func())
         javascript:void(func())
      //-->
   </script>
</head>
```

**Example 1**

The most common use of this operator is in a client-side *javascript:* URL, where it allows you to evaluate an expression for its side-effects without the browser displaying the value of the evaluated expression.

Here the expression **alert ('Warning!!!')** is evaluated but it is not loaded back into the current document −

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
         //-->
      </script>
   </head>

   <body>
      <p>Click the following, This won't react at all...</p>
      <a href = "javascript:void(alert('Warning!!!'))">Click me!</a>
   </body>
</html>
```

**Example 2**

Take a look at the following example. The following link does nothing because the expression "0" has no effect in JavaScript. Here the expression "0" is evaluated, but it is not loaded back into the current document.

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
         //-->
      </script>
   </head>

   <body>
      <p>Click the following, This won't react at all...</p>
      <a href = "javascript:void(0)">Click me!</a>
   </body>
</html>
```

**Example 3**

Another use of **void** is to purposely generate the **undefined** value as follows.

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function getValue() {
               var a,b,c;

               a = void ( b = 5, c = 7 );
               document.write('a = ' + a + ' b = ' + b +' c = ' + c );
            }
         //-->
      </script>
   </head>

   <body>
      <p>Click the following to see the result:</p>
      <form>
         <input type = "button" value = "Click Me" onclick =
"getValue();" />
      </form>
   </body>
</html>
```

## HTTP STATUS MESSAGES

### HTML Error Messages

When a browser requests a service from a web server, an error might occur, and the server might return an error code like "404 Not Found".

It is common to name these errors HTML error messages.

But these messages are something called HTTP status messages. In fact, the server always returns a message for every request. The most common message is 200 OK.

Below is a list of HTTP status messages that might be returned:

- 1xx: Information. E.G. 100 Continue, 101 Switching Protocols,
- 2xx: Successful. E.g. 200 OK, 201 Created, 202 Accepted, 204 No Content
- 3xx: Redirection: E.g. 300 Multiple Choices, 301 Moved Permanently
- 4xx: Client Error: E.g. 400 Bad Request, 403 Forbidden
- 5xx: Server Error: E.g. 500 Internal Server Error, 503 Service Unavailable

### Features of Web Authoring Tools

https://www.washington.edu/accesscomputing/webd2/student/unit7/module1/lesson1.html

### Lesson 1: Basic Features of Web Authoring Software

#### Overview

This lesson summarizes the basic features common to most popular web authoring software programs. It describes the overall working environment, and provides a summary of the HTML techniques you learned earlier in this course, and how to implement them using most web authoring tools. Use this handout as a starting point for exploring the web authoring software used in your class.

#### Learner Outcomes

At the completion of this exercise, you will:

- have a basic understanding of the features that are common to most web authoring software programs.
- be able to add HTML elements to a web page using web authoring software rather than writing code.
- be able to modify the attributes or properties of HTML elements using web authoring software.

#### Feature #1: Views

Most web authoring software provides multiple views of the web page you're working on.

- **Standard, normal, or design view** - This is typically the default view, which is a blank screen on which you type, paste, or insert content. This is very similar to a word processor screen.
- **Code view -** Allows you to view and work directly with the HTML code.
- **Split -** Both of the above views are displayed simultaneously in separate windows.

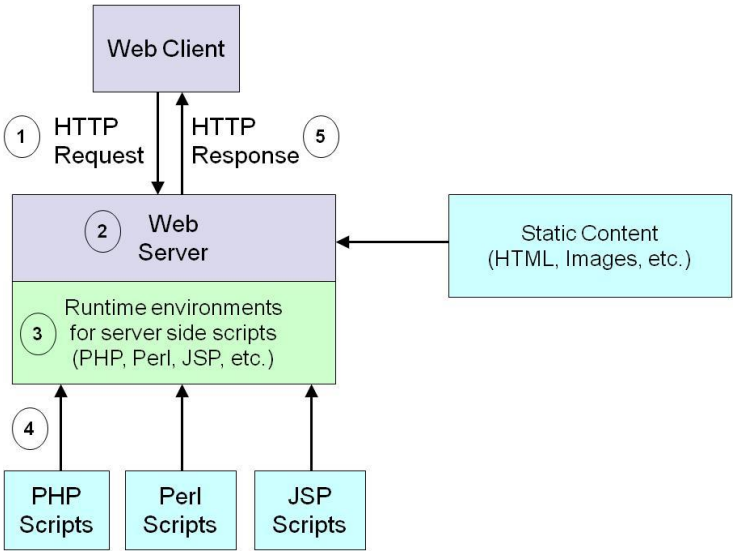
Typically you can switch between views by selecting a relevant item from the program's main menu (usually within the *View* menu) or by selecting a relevant tab or button.

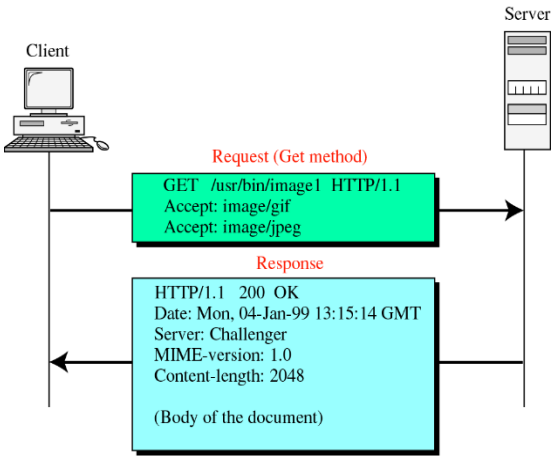
## Activity

- Find how to switch between views in your web authoring software. Does the software provide more than one way to do this? Try typing something on the screen in Normal or Design View, then switch to Code View to see the HTML that was generated by the web authoring tool.
- Ask your instructor for instructions on how to open and save files with your web authoring software in your school's computing environment.
- Now open your portfolio file tools.html in your web authoring software. At this point the page should have a "skip to main content" link, banner image, main H1 heading, and a navigation menu. Practice switching between views and exploring your page using your web authoring tool.

# SERVER SIDE TECHNOLOGY

- A web server responds to client requests (typically from a web browser) by providing resources such as XHTML documents.
- The web server maps the URL to a resource on the server (or to a file on the server's network) and returns the requested resource to the client.
- A web server and a client communicate using HTTP, a protocol for transferring requests and files over the Internet or an intranet.




Web client sends a HTTP request to a server that consists of:

- A request method: GET, POST, HEAD, PUT, etc. (GET and POST are the two most common used methods)
- A URI that identifies the requested resource
- Header fields
- A body (which can be empty)

Web server determines how to retrieve the requested resource.

- In the web server configuration file, one can specify how a particular kind of resources is to be handled:
  - Files with .php extension → To be handled by the PHP module
  - Files in folder /xxx/yyy/ →To be treated as CGI scripts

Runtime environment responsibilities are:

- Interpreting/executing the server-side scripts
- Maintaining sessions
- Parsing incoming HTTP request and generating outgoing HTTP response
- Caching generated output, frequently-used scripts, etc.

Different scripting languages may require different runtime environments

The requested script is processed by the corresponding runtime environment and the generated output is placed in the body of a HTTP response.

The HTTP response is sent to the web client.

# ACCESSING WEB SERVERS

To request documents from web servers, users must know the hostnames on which the web server software resides.

Users can request documents from local web servers or remote web servers.

Local web servers can be accessed through localhost - a hostname that references the local machine and normally translates to the IP address 127.0.0.1 (loopback address).

# Client-side versus Server-side Scripting

## Client-side scripting

- Client-side scripting
  - Validates user input
  - Accesses the browser
  - Enhances Web pages
  - Manipulates browser documents
- Client-side validation
  - Reduces number of requests that need to be passed to server
- Client-side scripting limitations
  - Browser dependency
  - Viewable to users through **View Source** command
- JavaScript is the most popular client-side script

## Server-Side scripts

Server-side scripts

- Provides programmers greater flexibility
- Generates custom responses for clients
- Contains greater programmatic capabilities than client-side equivalents
- Has access to server-side software that extend server functionality

Common Features of Server Side Technologies

- All server-side frameworks share a common set of features
  - Read data submitted by the user
  - Generate HTML dynamically based on user input
  - Determine information about the client browser
  - Access database systems
  - Exploit the HTTP protocol
- When evaluating which server-side technology to use, you need to consider a number of critical factors
  - Ease of development
    - How easily can you build new applications
  - Performance
    - How fast can the technology respond to queries
  - Scalability
    - Can the technology scale to thousands, even millions of users?
  - Security
    - Are there any inherent security vulnerabilities?

Examples of Server-side Technologies include:

- Common Gateway Interface (CGI)
- CGI/Perl
- Python (Frameworks like Django, Flask,)
- Active Server Pages (ASP)
- PHP
- Java Script (Node.js)
- Java Servlets
- Java Server Page (JSP)

## Common Gateway Interface (CGI)

- CGI represents one of the earliest, practical methods for generating web content
- Primarily written in the Perl programming language
- Unfortunately, some CGI programs suffer from scalability and performance problems

### ASP (Active Server Pages)

- Developed by Microsoft
- Runs on Microsoft's web server: IIS (Internet Information Server)
- Developers add ASP code directly to their HTML pages
- When a client requests a page, the web server takes the HTML page, runs the ASP code within the page, and returns the result in another HTML page

### Internet Information Services (IIS)

- Microsoft IIS is a Windows web server that allows that computer to serve documents.
- You place documents that will be requested from IIS either in the default directory or in a virtual directory.

### PHP Scripting

- PHP is an open source serverside scripting language
- Provides simple and powerful database access via Apache server
- Good for Rapid Application Development

### Apache HTTP Server

- The Apache HTTP Server, maintained by Apache Software Foundation. It is open source software that runs on UNIX, Linux, Mac OS X, Windows and numerous other platforms.
- Documents requested from an Apache Server must be either in the default htdocs directory or in a directory for which an Apache alias is configured. An alias is equivalent to IIS's virtual directory.
- The httpd.conf file contains all the information that the Apache HTTP Server needs to run correctly and serve web documents.

## IMPLEMENTING ASP

ASP supports many different development models:

- Classic ASP
- ASP.NET Web Forms
- ASP.NET MVC
- ASP.NET Web Pages
- ASP.NET API
- ASP.NET Core

You need to have a server to run ASP

- If you use Windows 7: Windows 7 includes Internet Information Server 7.5 (IIS) and can therefore run ASP.
- Windows XP Professional includes Internet Information Server 5.1 (IIS) and can therefore run ASP.
- Internet Information Services (IIS) on your Windows 10 computer is a good method.
- IIS is a full-featured web and FTP server with some powerful admin tools, strong security features, and can be used to host ASP.NET and PHP applications on the same server. You can even host WordPress sites on IIS.

Internet Information Services (IIS) is also available on Microsoft Visual Studio Integrated Development Environments like. *Visual Studio 2017, Visual Studio 2018, Visual Studio 2019, Visual Studio 2022 IDE's.* Download Community Edition( free for students)

**Enabling IIS and required IIS components on Windows 10**

1. Open Control Panel and click Programs and Features > Turn Windows features on or off.
2. Enable Internet Information Services.
3. Expand the Internet Information Services feature and verify that the web server components listed in the next section are enabled.
4. Click OK.

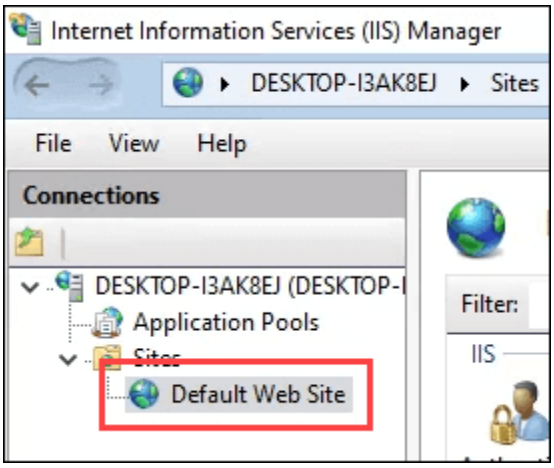
**Required IIS components**

The IIS components listed below satisfy the minimum requirements to run the Web Adaptor. If other IIS components are enabled, they do not need to be removed.

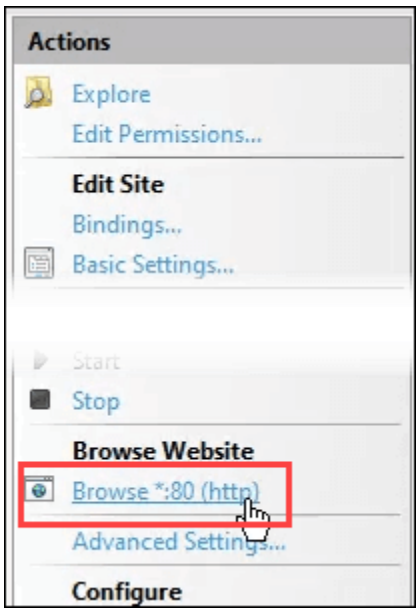
- Web Management Tools
    - IIS 6 Management Compatibility
        - IIS Metabase and IIS 6 configuration compatibility
    - IIS Management Console
    - IIS Management Scripts and Tools
    - IIS Management Service
- World Wide Web Services

- Application Development Features
  - .NET Extensibility 4.5
  - ASP.NET 4.5
  - ISAPI Extensions
  - ISAPI Filters
  - WebSocket Protocol
- Common HTTP Features
  - Default Document
  - Static Content
- Security
  - Basic Authentication
  - Request Filtering
  - Windows Authentication

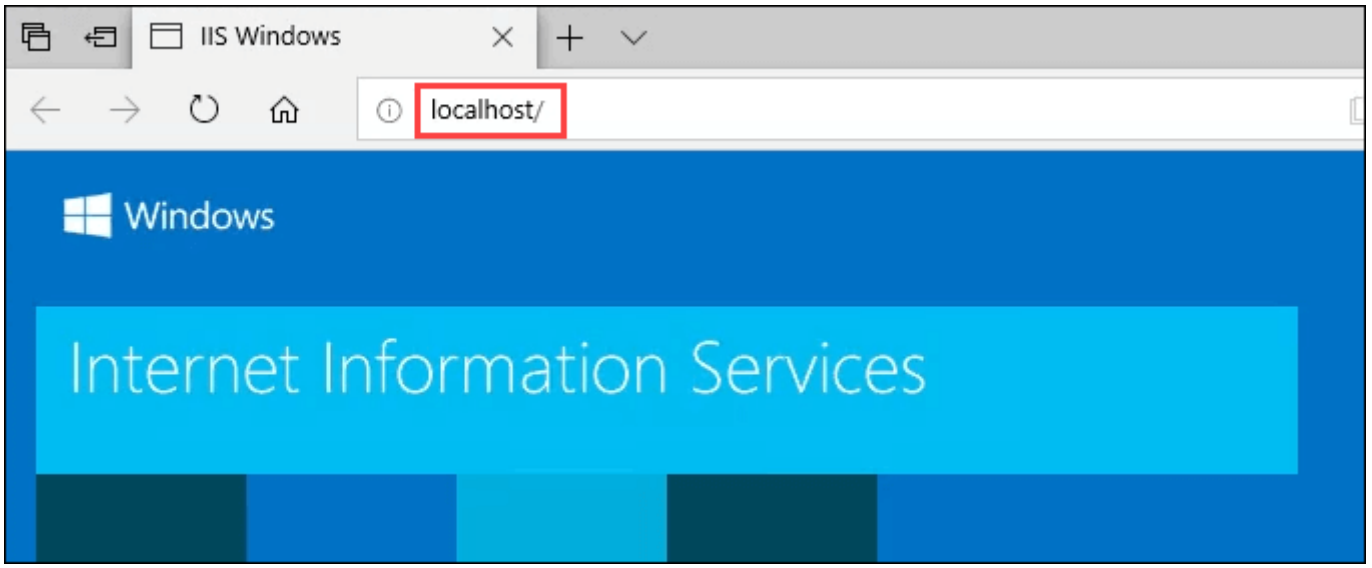
When IIS Manager opens, look in the left pane of the window under **Connections**. Expand the tree menu until you see **Default Web Site.** That's a placeholder site that is installed with IIS. Click on it to select it.




On the right side of the IIS Manager look under the **Browse Website** section. Click on **Browse *:80 (http).** That will open the default web site in your default web browser.




You'll see a web page like the following. Notice in the address bar that it says **localhost**. That's the address to type in to go to your new website.

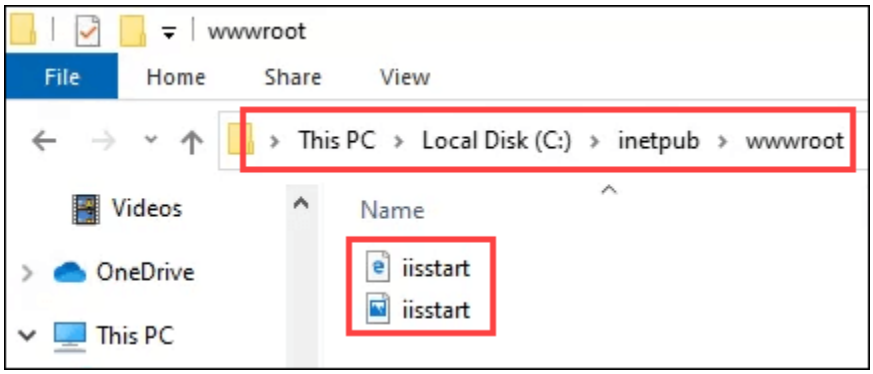



### Create Your First Web Page for IIS

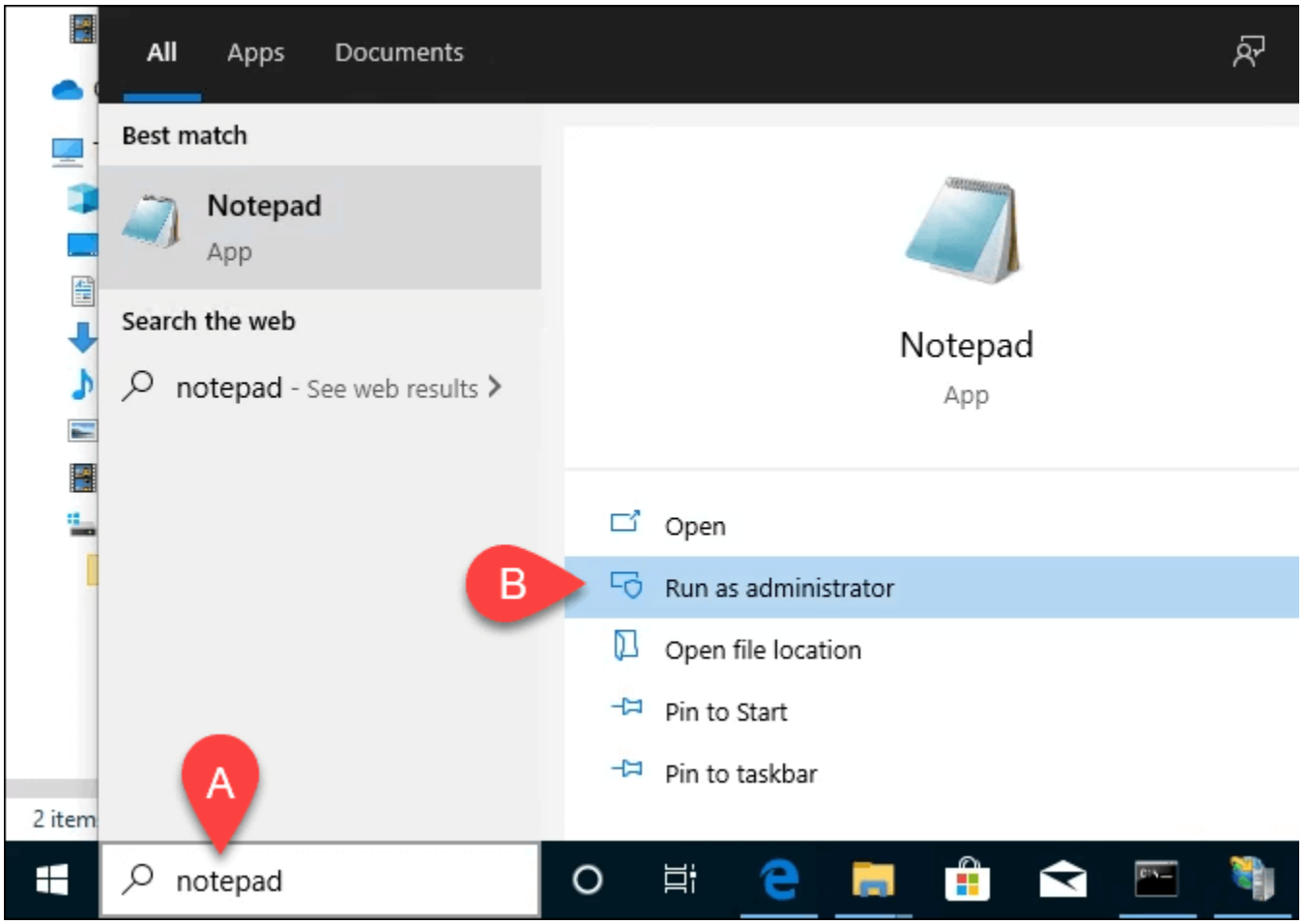
Before we go into the other two methods of installing IIS, let's see where the files that make your website are stored. We'll also make a very basic web page. Once you know how to do this, you'll know the basics to jump into learning web design and development.

1. After IIS is installed, open **File Explorer**. Navigate to **C:\intepub\wwwroot**. That's where the files that make up the website need to be stored. You'll see the default IIS web

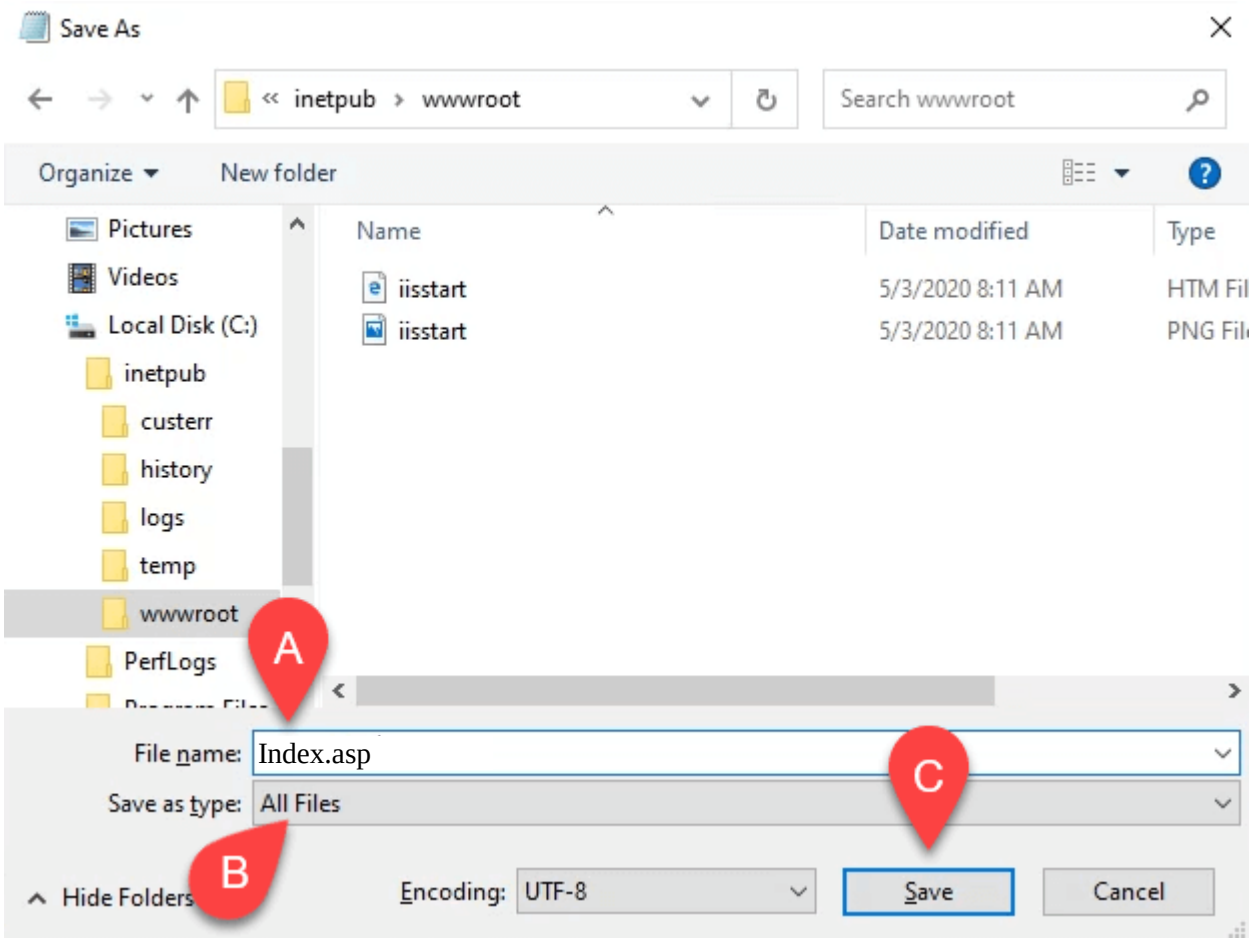
page file, **iisstart.html**, and the image shown on the page, **iisstart.png**. This is where you'll save your first web page.




2. Open **Notepad** as administrator. To save to the **wwwroot** location, you must be an administrator.




2. Save the file to the **wwwroot** location. Name it **index.asp** and change the **Save as type:** to **All Files**. Then click the **Save** button.




4. Now that it's saved as the right filetype, let's put some content in it. Enter the following HTML code for a very basic web page and save it:

```
<html>
<head>
<title>My first ASP Page</title>
</head>
   <body>
   <%
        Response.Write "<h1>Hello World!</h1>"
        Response.Write Now
   %>
   </body>
</html>
```

**ASP Scripts**

First, we need to tell the server when the ASP will start and end. In ASP you use the tags **<%** and **%>** to mark the start and end for the ASP codes that the server must execute

ASP scripts can be written in different languages

- Microsoft Visual Basic Scripting Edition (VBScript), but could also be written in another language - eg. JScript.

ASP.NET application codes can be written in any of the following languages:

- C#
- Visual Basic.Net
- Jscript
- J#

Because ASP. NET code is executed on the server, you cannot view the code in your browser. You will only see the output as plain HTML.

ASP.NET Web Pages use Razor markup with C# or VB code

Razor is a simple **markup syntax** for embedding server code (C# or VB) into ASP.NET web pages.

## Razor Syntax for C#

- C# code blocks are enclosed in @{ ... }
- Inline expressions (variables or functions) start with @
- Code statements end with semicolon
- Variables are declared with the var keyword, or the datatype (int, string, etc.)
- Strings are enclosed with quotation marks
- C# code is case sensitive
- C# files have the extension .cshtml

**C# Example**

```
<!-- Single statement block -->
@{ var myMessage = "Hello World"; }

<!-- Inline expression or variable -->
<p>The value of myMessage is: @myMessage</p>

<!-- Multi-statement block -->
@{
    var greeting = "Welcome to our site!";
    var weekDay = DateTime.Now.DayOfWeek;
    var greetingMessage = greeting + " Today is: " + weekDay;
}
<p>The greeting is: @greetingMessage</p>
```

**Razor Syntax for VB**

- VB code blocks are enclosed in @Code ... End Code
- Inline expressions (variables or functions) start with @
- Variables are declared with the Dim keyword
- Strings are enclosed with quotation marks
- VB code is not case sensitive
- VB files have the extension .vbhtml

To insert VBScript code onto your website, you must once again make use of the <script> tag.

Below is the correct code to insert VBScript code onto your site.

HTML Code:

```
<script type="text/vbscript">
     <!--script
     ***The VBScript code should go in this spot***
     -->
</script>
```

For VBScript, you set the type attribute equal to "text/vbscript", which is similar to specifying CSS.

We also include a comment around the VBScript code. This will prevent browsers that do not support

VBScript or have had VBScript disabled from displaying the VBScript code in the web browser.

**VB Example**

```
<!-- Single statement block  -->
@Code dim myMessage = "Hello World" End Code

<!-- Inline expression or variable -->
<p>The value of myMessage is: @myMessage</p>

<!-- Multi-statement block -->
@Code
dim greeting = "Welcome to our site!"
dim weekDay = DateTime.Now.DayOfWeek
dim greetingMessage = greeting & " Today is: " & weekDay
End Code

<p>The greeting is: @greetingMessage</p>
```

Razor is not a programming language. It's a server side markup language.

Razor is a markup syntax that lets you embed server-based code (Visual Basic and C#) into web pages.

Server-based code can create dynamic web content on the fly, while a web page is written to the browser. When a web page is called, the server executes the server-based code inside the page before it returns the page to the browser. By running on the server, the code can perform complex tasks, like accessing databases.

Razor is based on ASP.NET, and designed for creating web applications. It has the power of traditional ASP.NET markup, but it is easier to use, and easier to learn.

## ASP Uses VBScript

## The default scripting language in ASP is VBScript.

A scripting language is a lightweight programming language.

VBScript is a light version of Microsoft's Visual Basic.

ASP files can be ordinary HTML files. In addition, ASP files can also contain server scripts.

Scripts surrounded by **<% and %>** are executed on the server.

The **Response.Write()** method is used by ASP to write output to HTML.

The following example writes "Hello World" into HTML:

```
!DOCTYPE html>
<html>
<body>
      <%
      Response.Write("Hello World!")
      %>
</body>
</html>
```

HTML tags can be a part of the output:

Example

```
<!DOCTYPE html>
<html>
<body>
      <%
      Response.Write("<h2>You can use HTML tags to format
      the text!</h2>")
      %>
</body>
</html>
```

### Using JavaScript in ASP

To set JavaScript as the scripting language for a web page you must insert a language specification at the top of the page:

Example

```
<%@ language="javascript"%>
<!DOCTYPE html>
<html>
<body>
<%
Response.Write("Hello World!")
%>
</body>
</html>
```

# ASP Variables

Variables are "containers" for storing information.

### VBScript Variables

As with algebra, VBScript variables are used to hold values or expressions.

A variable can have a short name, like x, or a more descriptive name, like carname.

Rules for VBScript variable names:

- Must begin with a letter
- Cannot contain a period (.)
- Cannot exceed 255 characters

In VBScript, all variables are of type *variant* that can store different types of data. example demonstrates how to declare a variable, assign a value to it, and use the value in a text.

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

You can declare VBScript variables with the Dim, Public or the Private statement. Like this

```
Dim x
Dim carname
```

Option Explicit statement. This statement forces you to declare all your variables with the dim, public or private statement.

```
Option Explicit
Dim carname
carname=some value
```

**VBScript Array Variables**

An array variable is used to store multiple values in a single variable.

```
Dim names(2)
<html>
<body>
     <%
          Dim x(2,2)
          x(0,0)="Volvo"
          x(0,1)="BMW"
          x(0,2)="Ford"
          x(1,0)="Apple"
          x(1,1)="Orange"
          x(1,2)="Banana"
          x(2,0)="Coke"
          x(2,1)="Pepsi"
          x(2,2)="Sprite"
          for i=0 to 2
              response.write("<p>")
              for j=0 to 2
                  response.write(x(i,j) & "<br />")
              next
              response.write("</p>")
          next
     %>
</body>
</html>
```

**The Lifetime of Variables**

A variable declared outside a procedure can be accessed and changed by any script in the ASP file.

A variable declared inside a procedure is created and destroyed every time the procedure is executed. No scripts outside the procedure can access or change the variable.

To declare variables accessible to more than one ASP file, declare them as session variables or application variables.

**Session Variables**

Session variables are used to store information about ONE single user, and are available to all pages in one application. Typically information stored in session variables are name, id, and preferences.

**Application Variables**

Application variables are also available to all pages in one application. Application variables are used to store information about ALL users in one specific application.

**ASP Procedures**

The ASP source code can contain procedures and functions:

```
<!DOCTYPE html>
<html>
<head>
<%
     sub vbproc(num1,num2)
     response.write(num1*num2)
     end sub
%>
</head>
<body>

<p>Result: <%call vbproc(3,4)%></p>

</body>
</html>
```

Insert the <%@ language="*language*" %> line above the <html> tag to write the procedure/function in another scripting language:

```
<%@ language="javascript" %>
<!DOCTYPE html>
<html>
<head>
<%
     function jsproc(num1,num2)
     {
          Response.Write(num1*num2)
     }
%>
</head>
```

```
<body>

<p>Result: <%jsproc(3,4)%></p>

</body>
</html>
```

**Differences between VBScript and JavaScript**

When calling a VBScript or a JavaScript procedure from an ASP file written in VBScript, you can use the "call" keyword followed by the procedure name. If a procedure requires parameters, the parameter list must be enclosed in parentheses when using the "call" keyword. If you omit the "call" keyword, the parameter list must not be enclosed in parentheses. If the procedure has no parameters, the parentheses are optional.

When calling a JavaScript or a VBScript procedure from an ASP file written in JavaScript, always use parentheses after the procedure name.

**VBScript Procedures**

VBScript has two kinds procedures:

1. Sub procedure
2. Function procedure

**VBScript Sub Procedures**

A Sub procedure:

- is a series of statements, enclosed by the Sub and End Sub statements
- can perform actions, but **does not return** a value
- can take arguments

```
Sub mysub()
  some statements
End Sub
```

Or

```
Sub mysub(argument1,argument2)
  some statements
End Sub
```

**VBScript Conditional Statements**

**Conditional Statements**

Conditional statements are used to perform different actions for different decisions.

In VBScript we have four conditional statements:

- **If statement** - executes a set of code when a condition is true

- **If...Then...Else statement** - select one of two sets of lines to execute
- **If...Then...ElseIf statement** - select one of many sets of lines to execute
- **Select Case statement** - select one of many sets of lines to execute

### VBScript Looping

### Looping Statements

Looping statements are used to run the same block of code a specified number of times.

In VBScript we have four looping statements:

- **For...Next statement** - runs code a specified number of times
- **For Each...Next statement** - runs code for each item in a collection or each element of an array
- **Do...Loop statement** - loops while or until a condition is true
- **While...Wend statement** - Do not use it - use the Do...Loop statement instead

### ASP Forms and User Input

The Request.QueryString and Request.Form commands are used to retrieve user input from forms.

### User Input

The Request object can be used to retrieve user information from forms.

User input can be retrieved with the Request.QueryString or Request.Form command.

### Request.QueryString

The Request.QueryString command is used to collect values in a form with method="get".

Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.

```
<!DOCTYPE html>
<html>
<body>
<form action="demo_reqquery.asp" method="get">
Your name: <input type="text" name="fname" size="20" />
<input type="submit" value="Submit" />
</form>
<%
dim fname
fname=Request.QueryString("fname")
If fname<>"" Then
    Response.Write("Hello " & fname & "!<br>")
    Response.Write("How are you today?")
End If
%>
</body>
</html>
```