G. B. N. Government Polytechnic Nilokheri

E-content

Computer Engineering Department

2nd Semester

Advanced IT (AIT)

DETAILED CONTENTS

UNIT I

HTML Fundamentals

Introduction to HTML, Characteristics of HTML language, Structure of a HTML page, Describing Tags, How to create a HTML document? Viewing HTML document, commonly used web browsers. HTML4 – List of Tags in HTML4, HTML tags: Container elements, empty elements. Using tags, Heading, Paragraph, Changing appearance of text (bold, italics, underline, subscript, superscript), center tag, title tag. Changing font size, text color and background, Changing the background color and background of HTML page, Top marging, left margin, & nbsp,<hr> and its attributes.

UNIT II

Working with HTML

Using list and images: Unordered lists: type attribute, Ordered lists: start attribute, type attribute, value attribute, Nested lists, Inserting images, aligning an image, centering image, adding border to a image, alternate text, setting height and width, adding space around the image, Working with links: Anchor elements, creating hyperlink to a document, Internal linking and external linking.

UNIT III

Designing with HTML Creating tables: Creating a table attributes of table tag (BORDER, BORDERCOLOR, BGCOLOR, ALIGN, CELLSPACING, CELLPADING, WIDTH) Attributes of table row <tr> and table data <td> tag (BORDERCOLOR, BGCOLOR, ALIGN, VALIGN, HEIGHT), Row span and Col span, Working with Frames, Use and creating frames, Introduction to Forms, Steps for developing a Website.

UNIT IV

JAVA Script Overview and Core Language Features, Introduction to Scripting Languages, JavaScript Implementation, ECMA Script, DOM, BOM, Values-Variables-Literals-Constants-Operators, Expressions, Regular Expressions, Conditional Branching Statements- Conditional Looping Statements-Functions-Creating Simple, Java Script page-Adding JavaScript page into HTML

UNIT V

Document Access, The Document Object Model: Mapping your HTML -Text Nodes-Attribute Nodes Accessing the Nodes you Want: Finding an Element by ID-Finding Elements by Tag Name-Finding Elements by ClassName; Navigating the DOM Tree-Interacting with Attributes -

Changing Styles: Changing Styles with Class and Id-Font-Table Layout-Text Properties- Padding, Borders and Margins

Unit 1: HTML Fundamentals

HTML provides basic and advanced concepts of HTML. If you are new in learning HTML, then you can learn HTML from basic to a professional level and after learning HTML with CSS and JavaScript you will be able to create your own interactive and dynamic website. But Now We will focus on HTML only from unit 1 to unit 4 and in unit 5 will cover JavaScript.

The major points or characteristics of HTML are given below:

- HTML stands for Hypertext Markup Language.
- HTML is used to create web pages and web applications.
- HTML is widely used language on the web.
- We can create a static website by HTML only.
- Technically, HTML is a Markup language rather than a programming language.

What is HTML

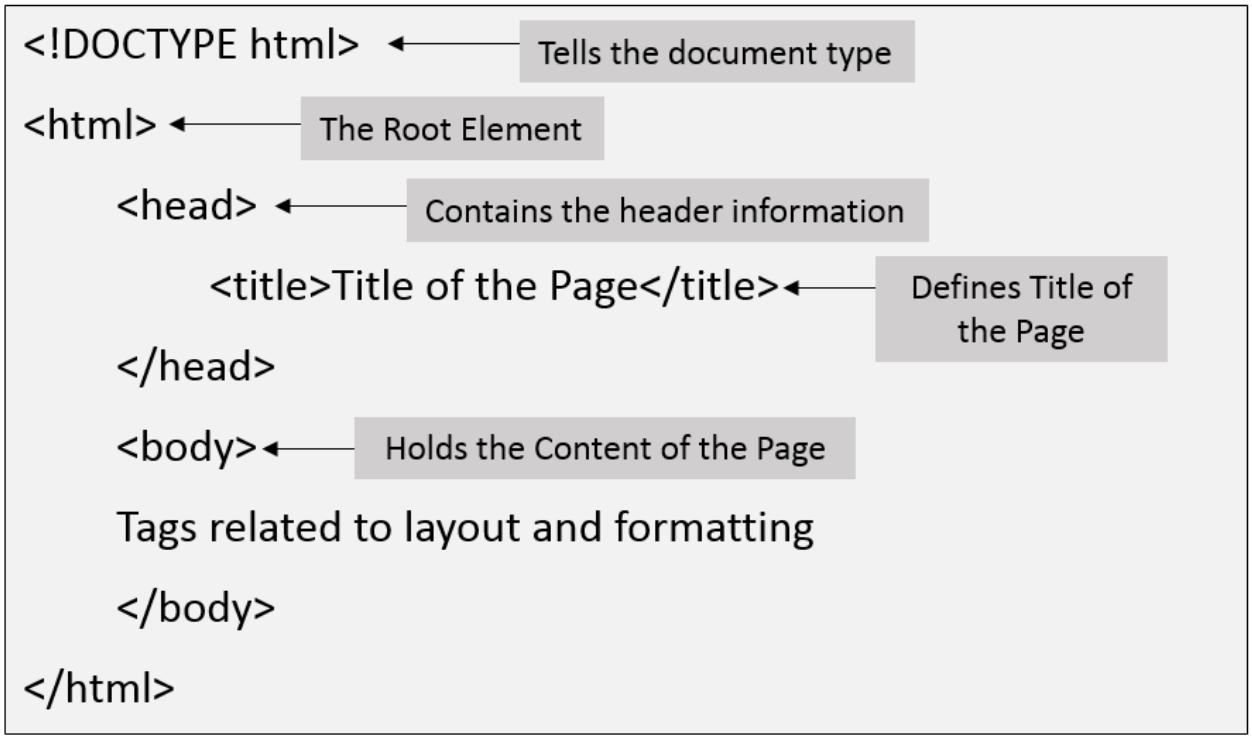
HTML is an acronym which stands for **Hyper Text Markup Language** which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. **With the help of HTML only, we can create static web pages**.

Hence, HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content.



Note: Inside <body> tag write <h1> and <p > tags

Description of HTML Example

<!DOCTYPE>: It defines the document type or it instruct the browser about the version of HTML.

<html > :This tag informs the browser that it is an HTML document. Text between html tag describes the web document. It is a container for all other elements of HTML except <!DOCTYPE>

<head>: It should be the first element inside the <html> element, which contains the metadata(information about the document). It must be closed before the body tag opens.

<title>: As its name suggested, it is used to add title of that HTML page which appears at the top of the browser window. It must be placed inside the head tag and should close immediately. (Optional)

<body> : Text between body tag describes the body content of the page that is visible to the end user. This tag contains the main content of the HTML document.

<h1> : Text between <h1> tag describes the first level heading of the webpage.

<p> : Text between <p> tag describes the paragraph of the webpage.

Brief History of HTML

In the late 1980's , a physicist, Tim Berners-Lee who was a contractor at CERN, proposed a system for CERN researchers. In 1989, he wrote a memo proposing an internet based hypertext system.

Tim Berners-Lee is known as the father of HTML. The first available description of HTML was a document called "HTML Tags" proposed by Tim in late 1991. The latest version of HTML is HTML5, which we will learn later in this tutorial.

HTML Versions

Since the time HTML was invented there are lots of HTML versions in market, the brief introduction about the HTML version is given below:

HTML 1.0: The first version of HTML was 1.0, which was the barebones version of HTML language, and it was released in1991.

HTML 2.0: This was the next version which was released in 1995, and it was standard language version for website design. HTML 2.0 was able to support extra features such as form-based file upload, form elements such as text box, option button, etc.

HTML 3.2: HTML 3.2 version was published by W3C in early 1997. This version was capable of creating tables and providing support for extra options for form elements. It can also support a web page with complex mathematical equations. It became an official standard for any browser till January 1997. Today it is practically supported by most of the browsers.

HTML 4.01: HTML 4.01 version was released on December 1999, and it is a very stable version of HTML language. This version is the current official standard, and it provides added support for stylesheets (CSS) and scripting ability for various multimedia elements.

HTML5 : HTML5 is the newest version of HyperText Markup language. The first draft of this version was announced in January 2008. There are two major organizations one is W3C (World Wide Web Consortium), and another one is WHATWG(Web Hypertext Application Technology Working Group) which are involved in the development of HTML 5 version, and still, it is under development.

Features or characteristics of HTML

1) It is a very **easy and simple language**. It can be easily understood and modified.

2) It is very easy to make an **effective presentation** with HTML because it has a lot of formatting tags.

3) It is a **markup language**, so it provides a flexible way to design web pages along with the text.

4) It facilitates programmers to add a **link** on the web pages (by html anchor tag), so it enhances the interest of browsing of the user.

5) It is **platform-independent** because it can be displayed on any platform like Windows, Linux, and Macintosh, etc.

6) It facilitates the programmer to add **Graphics, Videos, and Sound** to the web pages which makes it more attractive and interactive.

7) HTML is a case-insensitive language, which means we can use tags either in lower-case or upper-case.

NOTE: It is recommended to write all tags in lower-case for consistency, readability, etc.

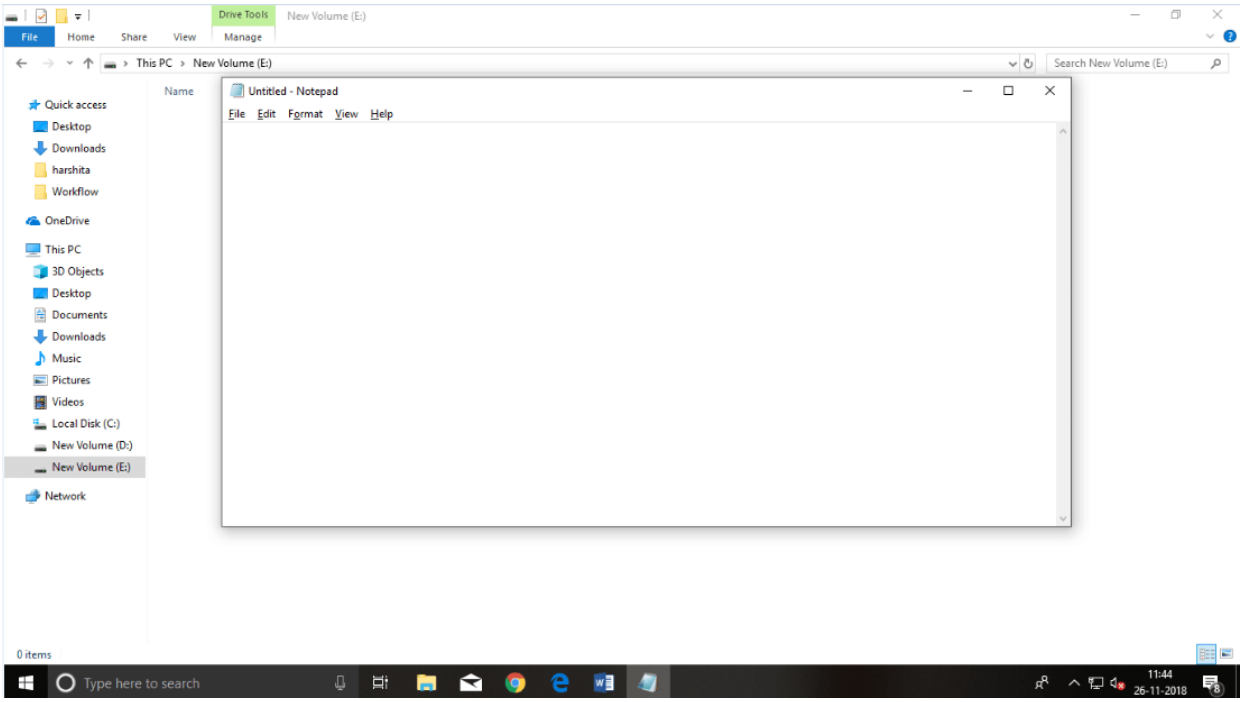
HTML text Editors:

- An HTML file is a text file, so to create an HTML file we can use any text editors.
- Text editors are the programs which allow editing in a written text, hence to create a web page we need to write our code in some text editor.
- There are various types of text editors available which you can directly download, but for a beginner, the best text editor is Notepad (Windows) or TextEdit (Mac).
- After learning the basics, you can easily use other professional text editors which are, **Notepad++, Sublime Text, Vim, etc**.
- In our tutorial, we will use Notepad and sublime text editor. Following are some easy ways to create your first web page with Notepad, and sublime text.

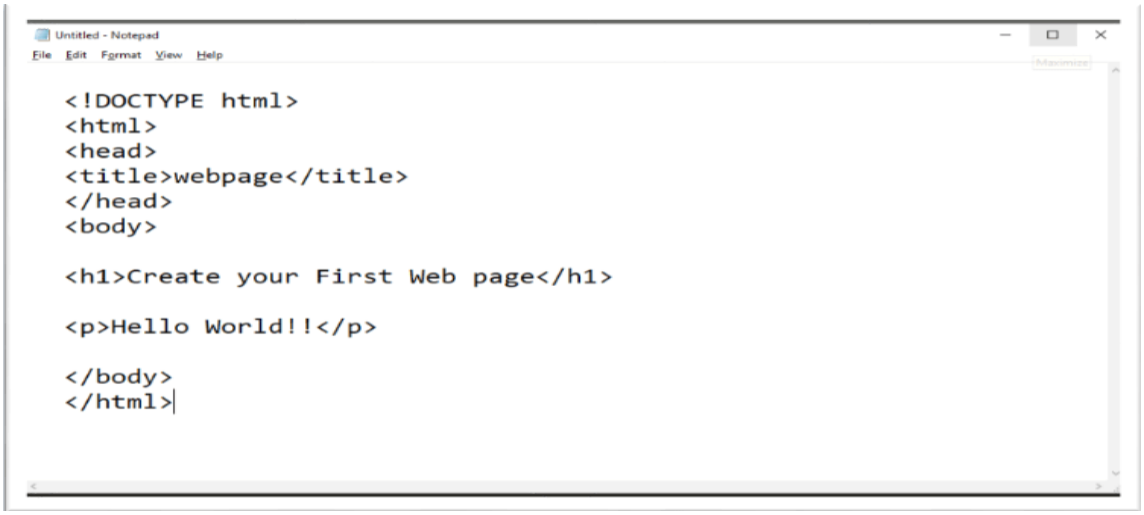
A. HTML code with Notepad. (Recommended for Beginners)

Notepad is a simple text editor and suitable for beginners to learn HTML. It is available in all versions of Windows, from where you easily access it.

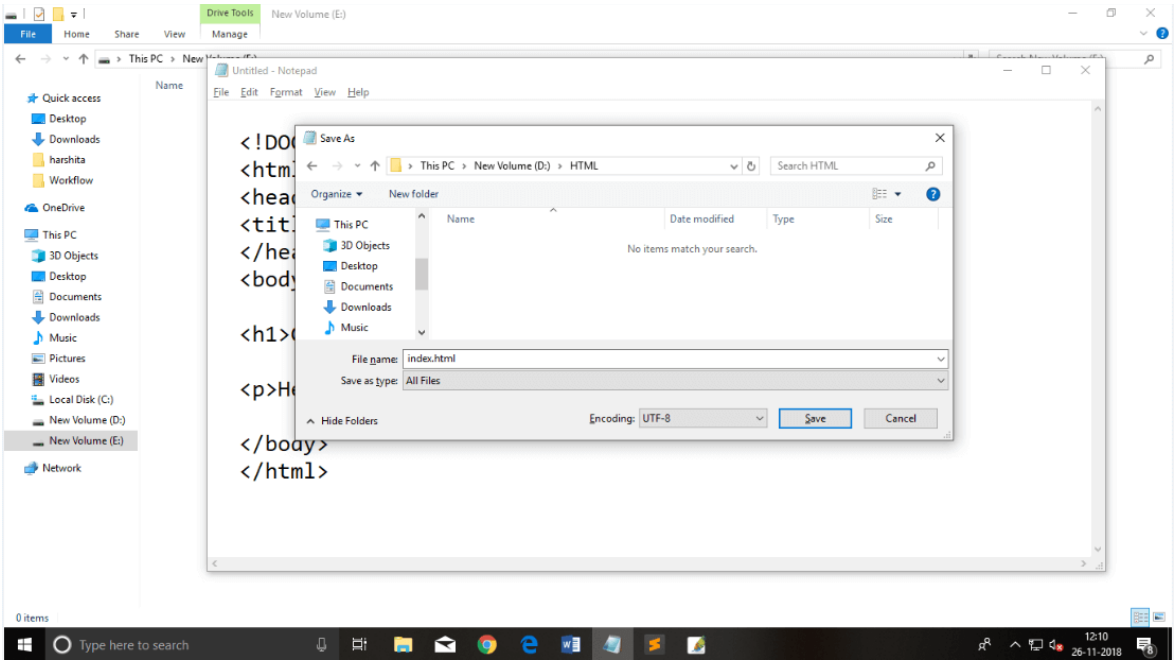
Step 1: Open Notepad (Windows)



Step 2: Write code in HTML

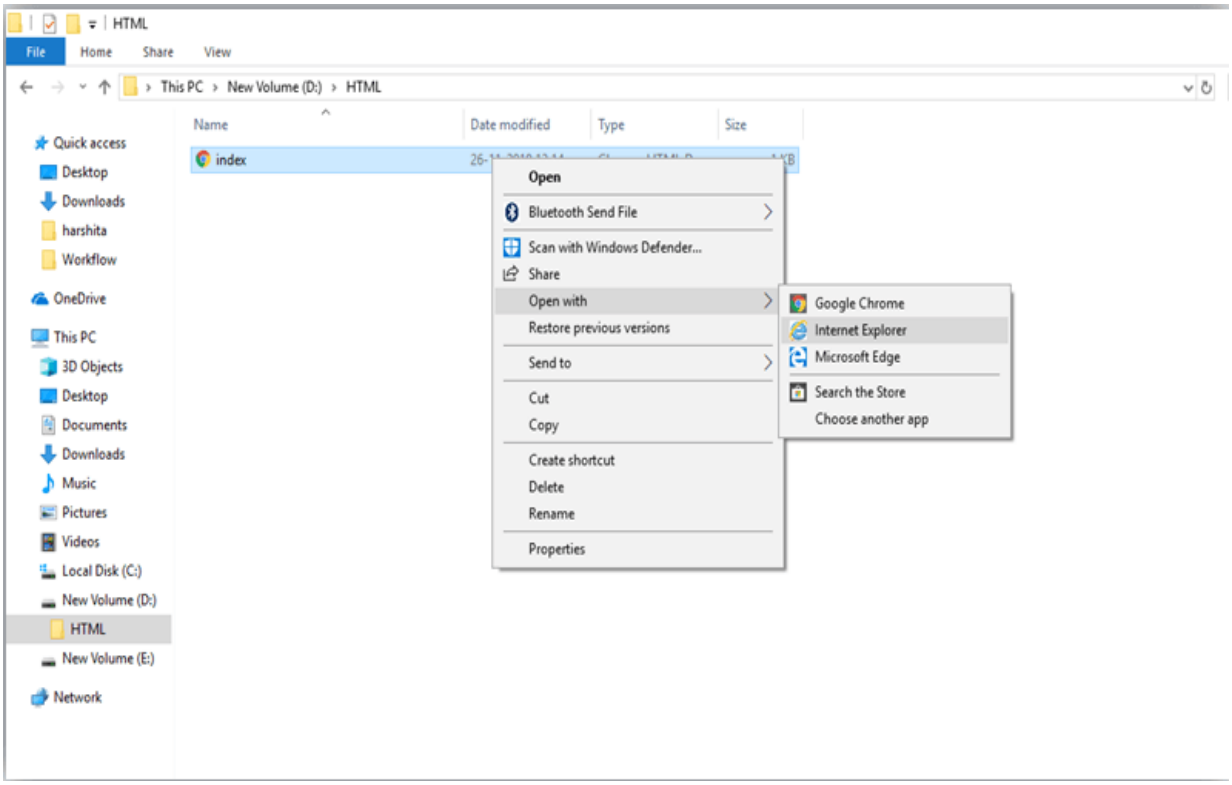


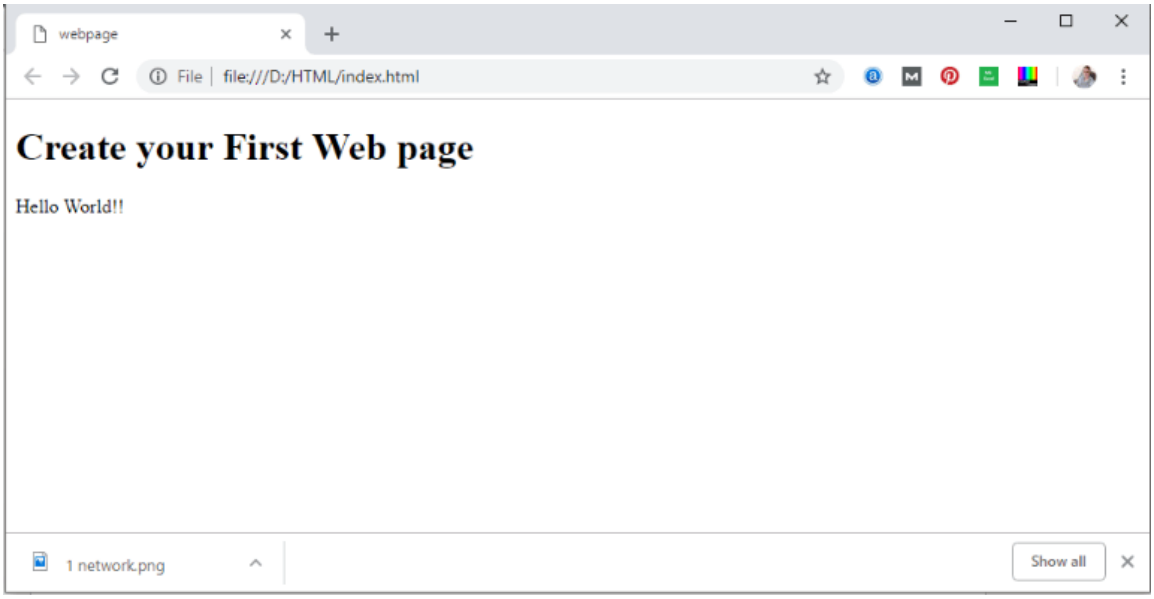
Step 3: Save the HTML file with .htm or .html extension.



Step 4: Open the HTML page in your web browser.

To run the HTML page, you need to open the file location, where you have saved the file and then either double-click on file or click on open with option





Note: You can execute HTML file in any browser, but there are some tags which are not supported by Some Web browser.

Commonly used web browsers:

Web Browsers are software installed on your PC. To access the Web, you need a web browser, such as Netscape Navigator, Microsoft Internet Explorer or Mozilla Firefox.

Currently you must be using any sort of Web browser while you are navigating through our site tutorialspoint.com. On the Web, when you navigate through pages of information, this is commonly known as web browsing or web surfing.

There are four leading web browsers – Explorer, Firefox, Netscape, and Safari, but there are many others browsers available. You might be interested in knowing Complete Browser Statistics. Now we will see these browsers in bit more detail.

While developing a site, we should try to make it compatible to as many browsers as possible. Especially sites should be compatible to major browsers like Explorer, Firefox, Chrome, Netscape, Opera, and Safari.

Internet Explorer

Internet Explorer (IE) is a product from software giant Microsoft. This is the most commonly used browser in the universe. This was introduced in 1995 along with Windows 95 launch and it has passed Netscape popularity in 1998.

You can download a latest version of this browser by clicking here → **Download Internet Explorer**

 Google Chrome

This web browser is developed by Google and its beta version was first released on September 2, 2008 for Microsoft Windows. Today, chrome is known to be one of the most popular web browser with its global share of more than 50%.

You can download a latest version of this browser by clicking here → **Download Google Chrome**

 Mozilla Firefox

Firefox is a new browser derived from Mozilla. It was released in 2004 and has grown to be the second most popular browser on the Internet.

You can download a latest version of this browser by clicking here → **Download Firefox**

 Safari

Safari is a web browser developed by Apple Inc. and included in Mac OS X. It was first released as a public beta in January 2003. Safari has very good support for latest technologies like XHTML, CSS2 etc.

You can download a latest version of this browser by clicking here → **Download Safari**

 Opera

Opera is smaller and faster than most other browsers, yet it is full- featured. Fast, user-friendly, with keyboard interface, multiple windows, zoom functions, and more. Java and non Java-enabled versions available. Ideal for newcomers to the Internet, school children, handicap and as a front-end for CD-Rom and kiosks.

You can download a latest version of this browser by clicking here → **Download Opera**

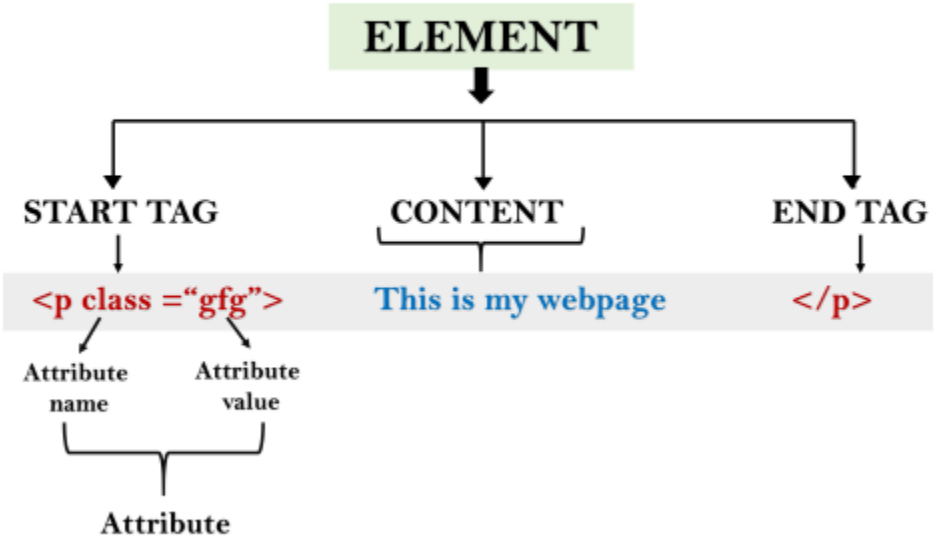
Building blocks of HTML

An HTML document consist of its basic building blocks which are:

- **Tags:** An HTML tag surrounds the content and apply meaning to it. It is written between < and > brackets.
- **Attribute:** An attribute in HTML provides extra information about the element, and it is applied within the start tag. An HTML attribute contains two fields: name & value.

Syntax

1. <tag name attribute_name= " attr_value"> content </ tag name>
    - **Elements:** An HTML element is an individual component of an HTML file. In an HTML file, everything written within tags are termed as HTML elements.



Example:

```
<!DOCTYPE html>
<html>
 <head>
  <title>The basic building blocks of HTML</title>
 </head>
 <body>
    <h2>The building blocks</h2>
    <p>This is a paragraph tag</p>
    <p style="color: red">The style is attribute of paragraph tag</p>
    <span>The element contains tag, attribute and content</span>
 </body>
</html>
```

Output:

The building blocks

This is a paragraph tag

The style is attribute of paragraph tag

The element contains tag, attribute and content

HTML Tags

HTML tags are like keywords which define that how web browser will format and display the content. With the help of tags, a web browser can distinguish between an HTML content and a simple content. HTML tags contain three main parts: opening tag, content and closing tag. But some HTML tags are unclosed tags.

When a web browser reads an HTML document, browser reads it from top to bottom and left to right. HTML tags are used to create HTML documents and render their properties. Each HTML tags have different properties.

An HTML file must have some essential tags so that web browser can differentiate between a simple text and HTML text. You can use as many tags you want as per your code requirement.

- All HTML tags must enclosed within < > these brackets.
- Every tag in HTML performs different tasks.
- If you have used an open tag <tag>, then you must use a close tag </tag> (except some tags)

Syntax

<tag> content </tag>

HTML Tag Examples

Note: HTML Tags are always written in lowercase letters. The basic HTML tags are given below:

 Paragraph Tag

<h2> Heading Tag </h2>

 Bold Tag

 Italic Tag

<u> Underline Tag</u>

Test it Now

Unclosed HTML Tags

Some HTML tags are not closed, for example br and hr.

**
 Tag**: br stands for break line, it breaks the line of the code.

<hr> Tag: hr stands for Horizontal Rule. This tag is used to put a line across the webpage.

HTML Meta Tags

DOCTYPE, title, link, meta and style

HTML Text Tags

<p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, , , <abbr>, <acronym>, <address>, <blockquote>, <cite>, <q>, <code>, <ins>, <del>, <pre>, and

HTML Link Tags

<a>

HTML Image and Object Tags

<img>, <area>, <map>, <param> and <object>

HTML List Tags

, <ol>, <li>, <dl>, <dt> and <dd>

HTML Table Tags

table, tr, td, th, tbody, thead, tfoot, col, colgroup and caption

---

HTML Form Tags

form, input, textarea, select, option, button, label, fieldset and legend

---

HTML Scripting Tags

script and noscript

> *Note: We will see examples using these tags in later charters.*

HTML Attribute

- HTML attributes are special words which provide additional information about the elements or attributes are the modifier of the HTML element.
- Each element or tag can have attributes, which defines the behaviour of that element.
- Attributes should always be applied with start tag.
- The Attribute should always be applied with its name and value pair.
- The Attributes name and values are case sensitive, and it is recommended by W3C that it should be written in Lowercase only.
- You can add multiple attributes in one HTML element, but need to give space between two attributes.

Syntax

```
<element attribute_name="value">content</element>
```

HTML Heading

A HTML heading or HTML h tag can be defined as a title or a subtitle which you want to display on the webpage. When you place the text within the heading tags <h1>.....</h1>, it is displayed on the browser in the bold format and size of the text depends on the number of heading.

There are six different HTML headings which are defined with the <h1> to <h6> tags, from highest level h1 (main heading) to the least level h6 (least important heading).

h1 is the largest heading tag and h6 is the smallest one. So h1 is used for most important heading and h6 is used for least important.

Headings in HTML helps the search engine to understand and index the structure of web page.

> *Note: The main keyword of the whole content of a webpage should be display by h1 heading tag.*

See this example:

```
<h1>Heading no. 1</h1>
<h2>Heading no. 2</h2>
<h3>Heading no. 3</h3>
<h4>Heading no. 4</h4>
<h5>Heading no. 5</h5>
<h6>Heading no. 6</h6>
```

Output:

Heading no. 1

Heading no. 2

Heading no. 3

Heading no. 4

Heading no. 5

Heading no. 6

HTML Paragraph

HTML paragraph or HTML p tag is used to define a paragraph in a webpage. Let's take a simple example to see how it work. It is a notable point that a browser itself add an empty line before and after a paragraph. An HTML <p> tag indicates starting of new paragraph.

> *Note: If we are using various <p> tags in one HTML file then browser automatically adds a single blank line between the two paragraphs.*

See this example:

```
1. <p>This is first paragraph.</p>
2. <p>This is second paragraph.</p>
3. <p>This is third paragraph.</p>
```

Test it Now

Output:

This is first paragraph.

This is second paragraph.

This is third paragraph.

Space inside HTML Paragraph

If you put a lot of spaces inside the HTML p tag, browser removes extra spaces and extra line while displaying the page. The browser counts number of spaces and lines as a single one.

```
<p>
I am
going to provide
you a tutorial on HTML
```

```
and hope that it will
be very beneficial for you.
</p>
<p>
Look, I put here a lot
of spaces                but          I know, Browser will ignore it.
</p>
<p>
You cannot determine the display of HTML</p>
<p>because resized windows may create different result.
</p>
```

Output:

```
I am going to provide you a tutorial on HTML and hope that it will be very beneficial for you.

Look, I put here a lot of spaces but I know, Browser will ignore it.

You cannot determine the display of HTML

because resized windows may create different result.
```

As you can see, all the extra lines and unnecessary spaces are removed by the browser.

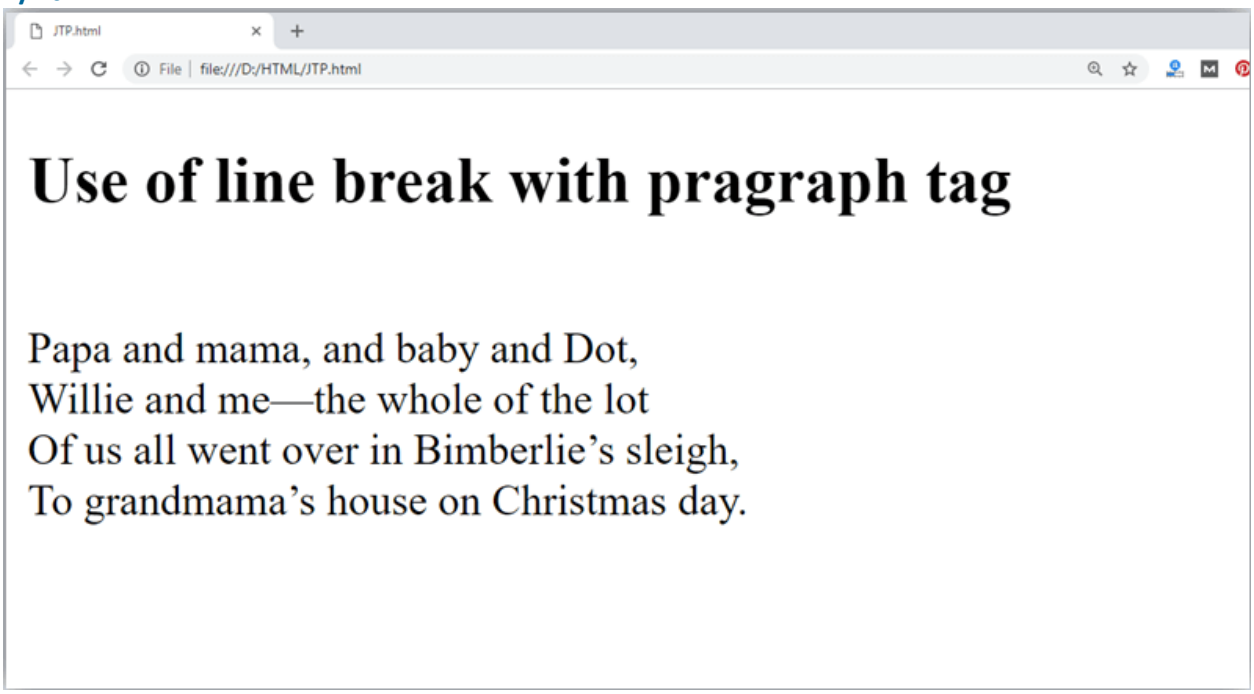
How to Use
 and <hr> tag with paragraph?

An HTML
 tag is used for line break and it can be used with paragraph elements. Following is the example to show how to use
 with <p> element.

Example:

```
<!DOCTYPE html>
<html>
   <head>
  </head>
 <body>
    <h2> Use of line break with pragraph tag</h2>
       <p><br>Papa and mama, and baby and Dot,
   <br>Willie and me?the whole of the lot
          <br>Of us all went over in Bimberlie's sleigh,
```

```
        <br>To grandmama's house on Christmas day.
      </p>
  </body>
</html>
```

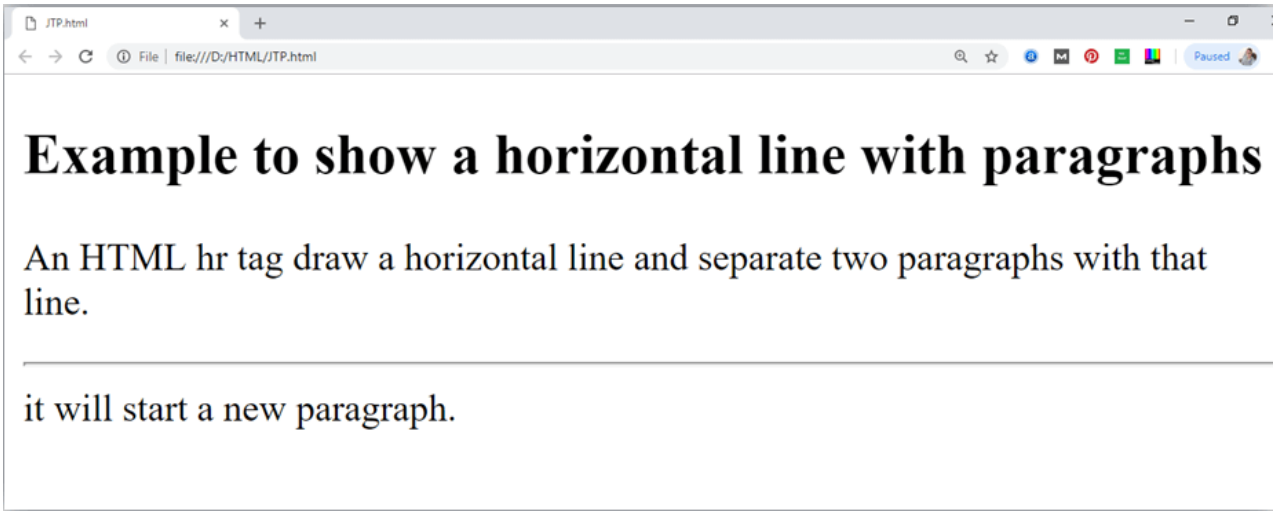


An HTML <hr> tag is used to apply a horizontal line between two statements or two paragraphs. Following is the example which is showing use of <hr> tag with paragraph.

Example:

```
<!DOCTYPE html>
<html>
 <head>
   </head>
 <body>
  <h2> Example to show a horizontal line with paragraphs</h2>
    <p> An HTML hr tag draw a horizontal line and separate two paragraphs with that line.<hr> it
 will start a new paragraph.
   </p>
 </body>
</html>
```

Output:



Changing appearance of Text tags:

Formatting elements were designed to display special types of text:

- <b> - Bold text
- - Important text
- <i> - Italic text
- - Emphasized text
- <mark> - Marked text
- <small> - Smaller text
- <del> - Deleted text
- <ins> - Inserted text
- <sub> - Subscript text
- <sup> - Superscript text

HTML <b> and Elements

The HTML <b> element defines bold text, without any extra importance.

Example

```
<b>This text is bold</b>
```

The HTML element defines text with strong importance. The content inside is typically displayed in bold.

Example

```
<strong>This text is important!</strong>
```

HTML <i> and Elements

The HTML <i> element defines a part of text in an alternate voice or mood. The content inside is typically displayed in italic.

Tip: The <i> tag is often used to indicate a technical term, a phrase from another language, a thought, a ship name, etc.

Example

```
<i>This text is italic</i>
```

The HTML element defines emphasized text. The content inside is typically displayed in italic.

Tip: A screen reader will pronounce the words in with an emphasis, using verbal stress.

Example

```
<em>This text is emphasized</em>
```

HTML <small> Element

The HTML <small> element defines smaller text:

Example

```
<small>This is some smaller text.</small>
```

HTML <mark> Element

The HTML <mark> element defines text that should be marked or highlighted:

Example

```
<p>Do not forget to buy <mark>milk</mark> today.</p>
```

HTML <del> Element

The HTML <del> element defines text that has been deleted from a document. Browsers will usually strike a line through deleted text:

Example

```
<p>My favorite color is <del>blue</del> red.</p>
```

HTML <ins> Element

The HTML <ins> element defines a text that has been inserted into a document. Browsers will usually underline inserted text:

Example

```
<p>My favorite color is <del>blue</del> <ins>red</ins>.</p>
```

HTML <sub> Element

The HTML <sub> element defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H_2O:

Example

```
<p>This is <sub>subscripted</sub> text.</p>
```

HTML <sup> Element

The HTML <sup> element defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font. Superscript text can be used for footnotes, like WWW[1]:

Example

```
<p>This is <sup>superscripted</sup> text.</p>
```

HTML Comments

HTML comments are not displayed in the browser, but they can help document your HTML source code.

HTML Comment Tag

You can add comments to your HTML source by using the following syntax:

```
<!-- Write your comments here -->
```

Notice that there is an exclamation point (!) in the start tag, but not in the end tag.

Note: Comments are not displayed by the browser, but they can help document your HTML source code.

Add Comments

With comments you can place notifications and reminders in your HTML code:

Example

```
<!-- This is a comment -->

<p>This is a paragraph.</p>

<!-- Remember to add more information here -->
```

Hide Content

Comments can be used to hide content.

This can be helpful if you hide content temporarily:

Example

```
<p>This is a paragraph.</p>

<!-- <p>This is another paragraph </p> -->

<p>This is a paragraph too.</p>
```

You can also hide more than one line. Everything between the <!-- and the --> will be hidden from the display.

Background Color

You can set the background color for HTML elements:

Hello World

Background color feature

Example

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

Text Color

You can set the color of text:

Hello World

Welcome to HTML

How are you...

Example

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Welcome to HTML</p>
<p style="color:MediumSeaGreen;">How are you...</p>
```

Non-breaking Space

A commonly used entity in HTML is the non-breaking space: ** **

A non-breaking space is a space that will not break into a new line.

Two words separated by a non-breaking space will stick together (not break into a new line). This is handy when breaking the words might be disruptive.

Examples:

- § 10
- 10 km/h
- 10 PM

Another common use of the non-breaking space is to prevent browsers from truncating spaces in HTML pages.

If you write 10 spaces in your text, the browser will remove 9 of them. To add real spaces to your text, you can use the ** ** character entity.

Tip: The non-breaking hyphen (‑) is used to define a hyphen character (-) that does not break into a new line.

HTML <hr> Tag

Example

Use the <hr> tag to define thematic changes in the content:

```
<h1>The Main Languages of the Web</h1>

<p>HTML is the standard markup language for creating Web pages. HTML describes the structure of a Web page, and consists of a series of elements. HTML elements tell the browser how to display the content.</p>

<hr>

<p>CSS is a language that describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work, because it can control the layout of multiple web pages all at once.</p>

<hr>

<p>JavaScript is the programming language of HTML and the Web. JavaScript can change HTML content and attribute values. JavaScript can change CSS. JavaScript can hide and show HTML elements, and more.</p>
```

Definition and Usage

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.

Example

Align a <hr> element (with CSS):

```
<hr style="width:50%;text-align:left;margin-left:0">
```

Example

A noshaded <hr> (with CSS):

```
<hr style="height:2px;border-width:0;color:gray;background-color:gray">
```

Example

Set the height of a <hr> element (with CSS):

```
<hr style="height:30px">
```

Example

Set the width of a <hr> element (with CSS):

```
<hr style="width:50%">
```

UNIT 4

What is JavaScript?

JavaScript (JS) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.

It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market.

Another popular use of JS: In addition to web browsers, databases such as Couch DB and Mongo DB use JavaScript as their scripting and query language.

Features of JavaScript

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

JavaScript History

JavaScript was invented by **Brendan Eich** in 1995.

It was developed for **Netscape 2**, and became the **ECMA-262** standard in 1997.

After Netscape handed JavaScript over to ECMA, the Mozilla foundation continued to develop JavaScript for the Firefox browser. Mozilla's latest version was 1.8.5. (Identical to ES5).

Internet Explorer (IE4) was the first browser to support ECMA-262 Edition 1 (ES1).

Year	ECMA	Browser
1995		JavaScript was invented by Brendan Eich
1997		JavaScript became an ECMA standard (ECMA-262)
1997	ES1	ECMAScript 1 was released

1998	ES2	ECMAScript 2 was released
2009	ES5	ECMAScript 5 was released
2015	ES6	ECMAScript 6 was released
2017	ES6	Full support for ES6 in Firefox 54
2017	ES6	Full support for ES6 in Edge 15
2018	ES6	Full support for ES6 in all browsers **

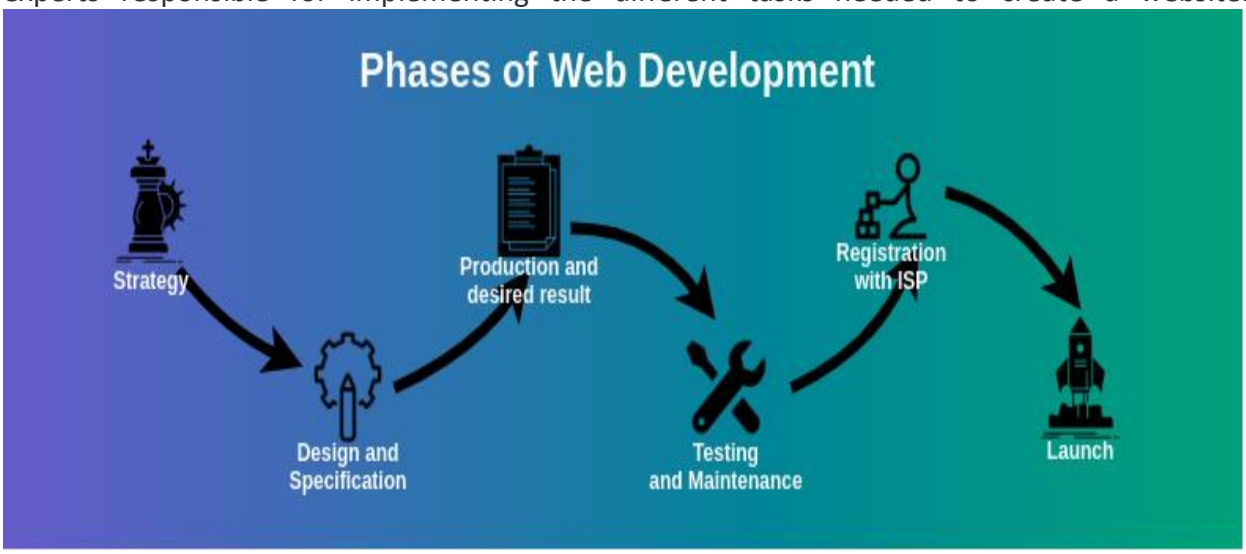
The main difference between client-side and server-side scripting is given below.

Basis	Client-side Scripting	Server-side Scripting
Primary Function	The primary function of client-side scripting is to provide the requested output to the end-user	The primary function of server-side scripting is to manipulate and give access to the required database as per request.
Uses	The client side is used as the front end, where the user gets to see what we have browsed.	The server side is used as a back end where data is processed and is not visible to the client user.
Code allowance	On the client side, the user is allowed to access the code written after verifying the user's need.	Server-side scripting allows the back-end developer to hide the source code from the user.
Processing	The client-side does not need any interaction with the server.	Server-side scripting on the other hand is all about communicating with the servers.
Function	Used for the visibility and getting out the required data from servers' database	Used for the customization or modification of the data to change the website dynamically.
Dependability	Client-side scripting depends upon the user's browser version.	Serve-side does not depend on the client.

Basis	Client-side Scripting	Server-side Scripting
Security	This way of scripting is less secure than Server-side scripting because of the accessibility of code provided to the client.	Server-side scripting is considered a more secure way of working on a web application
Connectivity	The client-side does not connect to the database at the webserver.	The server side helps connect with the database, which is already stored in the server database.
File Access	It does have any access to the files present on the web servers. But we have the option to upload files from the front end	It has total access to the files which are stored in the web database server.
Occurrence	It occurs when the browser processes all the codes, and then it reacts according to the client's query.	It only acts after the client initiates the browsing request.
Running	It runs on the end-user's system.	It runs on the web server.
Languages	HTML, JavaScript, and CSS are used to display the request	PHP, Python, Ruby, Node js are some of the programming languages used on server-side

Phases of web Development

Web Development refers to a term that includes all the processes involved in developing a web project or website. It contains the various phases such as planning, designing, Implementation & testing, and launching of the web project. The web development process requires a team of experts responsible for implementing the different tasks needed to create a website.



The various stages that are needed in order to develop a web project in web development are as following:

Strategy: The first step in the web development process for a developer is to make a strategy for developing a web page or web site. In the strategy phase, web developer has to done the following:

- Deciding goals and objectives
- Developing team

- Make the appropriate analysis associated with problem and review the analysis
- Formulate a list of tasks
- Proposal of project to web team for developing

Specification and Design: After the strategy-making, the next step in the web development process is to develop a planned work. Web developer has to determine the schedule and the specifications. The tasks in this phase are as follows.

- Developing approach
- Selection of front end(client side) or back end(server side) programming languages
- Planning of contents needed for use
- Making of rough design for project
- Making of final design from rough design, if there is no considerable modification in rough design.
- Frame a prototype(dummy website) of project for developing
- Test the prototype

If prototype is accomplish, then go to next phase phase-3 otherwise repeat the phase 2 until prototype is accomplish.

Production of desired result: In this phase of the web development process the actual functional site is built. After the proper testing of the prototype, the developer has to work on developing the actual live web project. The actual live web project is built according to the requirements of the client. Web developer has to consider all the situations from the design phase to create all the features in the web project.
This phase involves both front end development and back end development of the website.
Front end development comprises of the writing codes with the basic technologies like HTML, CSS, JavaScript etc. according to the web standards. It generally starts by developing the home page first and then other pages.
Back end development is also completed in this phase by installing and configuring the content management systems, databases, and frameworks using PHP, Python, Ruby, SQL etc.

After completing all the steps that were finalized in the strategy and design phase by which the original website becomes functional, it is tested in the next phase.

Testing and Maintenance: Testing is an important phase in the web development process. Testing is performed by the developers and testers to ensure the client's requirements after completion of the web project.

Registration with ISP: After completion of the Testing and Maintenance and removing all the bugs from the project, the next step or phase is to register the web project with the regional ISP to make the web project legal.
The client has to select and decide the ISP which provides domain name registration and web hosting services.

Launch: This is the last phase of the web development process. Project is launched after getting registered with ISP. After launching, web project is publicly accessed by the users of the particular domain. The tasks performed in the launch phase are as follows.

- Migration of data
- Launching of server
- Merging of code
- Redirecting domain name

Application of JavaScript

JavaScript is used to create interactive websites. It is mainly used for:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

NOTE: Javascript example is easy to code. JavaScript provides 3 places to put the JavaScript code:

a. within body tag
b. within head tag
c. external JavaScript file(.js file)

Example1(within body Tag)

```
<html>
<head><title>My first JS program</title></head>
   <body>
   <h2>Welcome to JavaScript</h2>
     <script>
     document.write("JavaScript is a simple language for learners");
     </script>
    </body>
</html>
```

Output:
Welcome to JavaScript

JavaScript is a simple language for learners

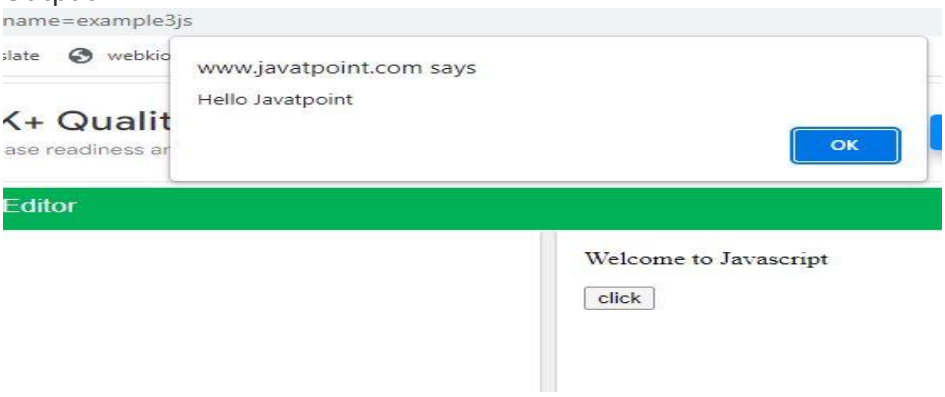
Highlights of Example1:

The **document.write()** function is used to display dynamic content through JavaScript. We will learn about document object in detail later.

Example 2: within head section

```
<html>
<head>
    <script>
    function msg()
      {
     alert("Hello Javatpoint");
      }
    </script>
</head>
<body>
<p>Welcome to Javascript</p>
<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</body>
</html>
```

Output:



How to add comments in JS: The **JavaScript comments** are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code.

The JavaScript comment is ignored by the JavaScript engine i.e. embedded in the browser.

Advantages of JavaScript comments:

There are mainly two advantages of JavaScript comments.
To make code easy to understand: It can be used to elaborate the code so that end user can easily understand the code.
To avoid the unnecessary code: It can also be used to avoid the code being executed. Sometimes, we add the code to perform some action. But after sometime, there may be need to disable the code. In such case, it is better to use comments.

Types of JavaScript Comments

There are two types of comments in JavaScript.
Single-line Comment
Multi-line Comment

JavaScript Single line Comment

It is represented by double forward slashes (//). It can be used before and after the statement.

Let's see the example of single-line comment i.e. added before the statement.

```
<script>
// It is single line comment
document.write("hello javascript");
</script>
```

JavaScript Multi line Comment

It can be used to add single as well as multi line comments. So, it is more convenient.
It is represented by forward slash with asterisk then asterisk with forward slash at the end.
For example:

```
/* your code here */
```

It can be used before, after and middle of the statement.

```
<script>
/* It is multi line comment.
It will not be displayed */
document.write("example of javascript multiline comment");
</script>
```

JavaScript Variable:

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).
1. Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.

2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

Example:

```
<html>
<head><title>Variable in JS program</title></head>
<body>
<script>
var x = 10;
var y = 20;
var z=x+y;
document.write(z);
</script>
</body>
</html>
```

Output: 30

Operators in Java Script:

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2

%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

JavaScript Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
~	Bitwise NOT	(~10) = -10

JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

JavaScript Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
new	creates an instance (object)
typeof	checks the type of object.

void	it discards the expression's return value.

Ex1:

Program of addition, subtraction, multiply, and division in Java Script

```
<!doctype html>
<html>
<body>
<script>
  var numOne=12, numTwo=10, res;
  res = numOne + numTwo;
  document.write("Add = " + res + "<br/>");
  res = numOne - numTwo;
  document.write("Subtract = " + res + "<br/>");
  res = numOne * numTwo;
  document.write("Multiply = " + res + "<br/>");
  res = numOne/numTwo;
  document.write("Divide = " + res + "<br/>");
</script>
</body>
</html>
```

Output:

Add = 22
Subtract = 2
Multiply = 120
Divide = 1.2

Different ways to declare Variables in JavaScript:

- Using var
- Using let

- Using const
- Using nothing

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Variables</h2>
<p>In this example, x, y, and z are variables.</p>
<p id="demo"></p>
<script>
 let x = 5;
 let y = 6;
 let z = x + y;
document.getElementById("demo").innerHTML ="The value of z is: " + z;
document.write("The value of z is:" + z);
</script>
</body>
</html>
```

JavaScript Variables

In this example, x, y, and z are variables.

The value of z is: 11

The value of z1 is:11

When to Use JavaScript var?

Always declare JavaScript variables with var, let, or const.

The var keyword is used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

If you want your code to run in older browsers, you must use var.

When to Use JavaScript const?

If you want a general rule: always declare variables with const.

If you think the value of the variable can change, use let.

In this example, price1, price2, and total, are variables:

Example

```
const price1 = 5;
const price2 = 6;
let total = price1 + price2;
```

The two variables price1 and price2 are declared with the const keyword.

These are constant values and cannot be changed.

The variable total is declared with the let keyword.

This is a value that can be changed.

Different Kinds of Loops

JavaScript supports different kinds of loops:

- for - loops through a block of code a number of times
- while - loops through a block of code while a specified condition is true
- do/while - also loops through a block of code while a specified condition is true

The For Loop

The for statement creates a loop with 3 optional expressions:

```
for (expression 1; expression 2; expression 3)
 {
  // code block to be executed
}
```

Expression 1(Initialization) is executed (one time) before the execution of the code block.

Expression 2(Condition) defines the condition for executing the code block.

Expression 3(Increment/Decrement) is executed (every time) after the code block has been executed.

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

<h2>JavaScript For Loop</h2>
```

```
<p id="demo"></p>

<script>
let text = "";
for (let i = 0; i < 5; i++)
{
  text += "The number is " + i + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

From the example above, you can read:

Expression 1 sets a variable before the loop starts (let i = 0).

Expression 2 defines the condition for the loop to run (i must be less than 5).

Expression 3 increases a value (i++) each time the code block in the loop has been executed.

Another Example:

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

<h2>JavaScript For Loop</h2>

<p id="demo"></p>

<script>
const p_lang = ["html", "css", "java script", "java", "c", "c++"];

let text = "";
for (let i = 0; i < p_lang.length; i++) {
  text += p_lang[i] + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

JavaScript While Loop

Loops can execute a block of code as long as a specified condition is true.

The While Loop: The while loop loops through a block of code as long as a specified condition is true.

Syntax

```
while (condition) {
  // code block to be executed
}
```

Example

In the following example, the code in the loop will run, over and over again, as long as a variable (i) is less than 10:

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

<h2>JavaScript While Loop</h2>

<p id="demo"></p>

<script>
let text = "";
let i = 0;
while (i < 10) {
  text += "<br>The number is " + i;
  i++;
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

JavaScript While Loop

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4

The number is 5
The number is 6
The number is 7
The number is 8
The number is 9

If you forget to increase the variable used in the condition, the loop will never end. This will crash your browser.

The Do While Loop

The do while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {
  // code block to be executed
}
while (condition);
```

Example

The example below uses a do while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

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

<h2>JavaScript Do While Loop</h2>

<p id="demo"></p>

<script>
let text = ""
let i = 0;

do {
  text += "<br>The number is " + i;
  i++;
}
```

```
while (i < 10);

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Output:

JavaScript Do While Loop

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
```

JavaScript Functions

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

Why Functions?

Code Reusability: With functions you can reuse code and we can call a function several times so it saves coding.

You can write code that can be used many times.

You can use the same code with different arguments(input values), to produce different results.

Less Coding: It makes our program compact. We don’t need to write many lines of code each time to perform a common task.

JavaScript Function Syntax

A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include arguments/parameter names separated by commas:
(*parameter1, parameter2, ...*)

The code to be executed, by the function, is placed inside curly brackets: **{}**

function *name*(*parameter1, parameter2, parameter3*)

```
{
  // code to be executed
}
```

Function **parameters** are listed inside the parentheses () in the function definition.

Function **arguments** are the **values** received by the function when it is invoked/call.

Inside the function, the arguments (the parameters) behave as local variables.

Example:

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Functions</h1>

<p>Call a function which performs a calculation and returns the result:</p>

<p id="demo"></p>

<script>
function myFunction(p1, p2) {
  return p1 * p2;
}
let n1=+prompt("Enter n1");
let n2=+prompt("Enter n2");
let result = myFunction(n1,n2);
document.getElementById("demo").innerHTML = result;
</script>
```

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

Output:

JavaScript Functions

Call a function which performs a calculation and returns the result:

24

Function Invocation

The code inside the function will execute when "something" **invokes** (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Return

When JavaScript reaches a return statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Functions often compute a **return value**. The return value is "returned" back to the "caller":

The () Operator

The () operator invokes (calls) the function:

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

<h1>JavaScript Functions</h1>

<p>Invoke (call) a function that converts from Fahrenheit to Celsius:</p>
<p id="demo"></p>

<script>
function FahToCelsius(f)      //FahToCelsius(77)
{
```

```
  return (5/9) * (f-32);
}
let n1=+prompt("Enter the value of n1 in fah");
let value = FahToCelsius(n1);    //fun calling
document.getElementById("demo").innerHTML = value;
</script>

</body>
</html>
```

Accessing a function with incorrect parameters can return an incorrect answer:

Suppose 77 is not passed here then it will return NaN

Accessing a function without () returns the function and not the function result:

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

<h1>JavaScript Functions</h1>

<p>Accessing a function without () returns the function and not the function result:</p>
<p id="demo"></p>

<script>
function toCelsius(f) {
  return (5/9) * (f-32);
}

let value = toCelsius;
document.getElementById("demo").innerHTML = value;
</script>

</body>
</html>
```

OutPut:

JavaScript Functions

Accessing a function without () returns the function and not the function result:

```
function toCelsius(f) { return (5/9) * (f-32); }
```

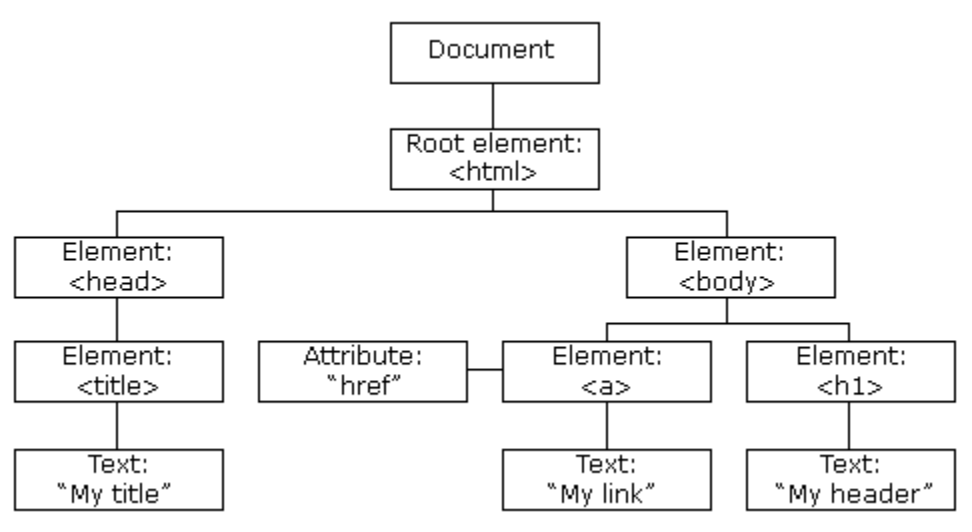
Unit-5

The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

Methods of document object

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

1. Example of getElementsById() method:

```
<!DOCTYPE html>

<html>
<body>

<h2>JavaScript HTML DOM</h2>

<p id="intro">Finding HTML Elements by Id</p>
<p>This example demonstrates the <b>getElementsById</b> method.</p>

<p id="demo"></p>

<script>
const element = document.getElementById("intro");

document.getElementById("demo").innerHTML =
"The text from the intro paragraph is: " + element.innerHTML;
</script>
</body>
</html>
```

2. Using getElementsByTagName() method:

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

<h2>JavaScript HTML DOM</h2>

<p>Finding HTML Elements by Tag Name.</p>
<p>This example demonstrates the <b>getElementsByTagName</b> method.</p>

<p id="demo"></p>

<script>
const element = document.getElementsByTagName("p");

document.getElementById("demo").innerHTML = 'The text in first paragraph (index 0) is: ' +
element[0].innerHTML;
</script>

</body>
</html>
```

Try more CSS properties and change image, padding, border, and margin