

# Unit-1

Introduction to HTML, Web and  
Internet

# Introduction to Internet - WWW

- Internet is a global communication system that links together thousands of individual networks.
- It allows exchange of information between two or more computers on a network. Thus internet helps in transfer of messages through mail, chat, video & audio conference, etc.
- It has become mandatory for day-to-day activities: bills payment, online shopping and surfing, tutoring, working, communicating with peers, etc.

# Introduction to Internet - WWW

- Basics of Computer Networks
  - Computer network is an interconnection between two or more hosts/computers. Different types of networks include Local Area Network (LAN), Wide Area Network (WAN), etc.
- Internet Architecture
  - Internet is called the network of networks. It is a global communication system that links together thousands of individual networks.
  - Internet architecture is a meta-network, which refers to a congregation of thousands of distinct networks interacting with a common protocol

# Introduction to Internet - WWW

- Services on Internet

- Internet acts as a carrier for numerous diverse services, each with its own distinctive features and purposes.

- Communication on Internet

- communication can happen through the Internet by using Email, Internet Relay Chat, Video Conference etc.

- Web Browsing Software

- "World Wide Web" or simple "Web" is the name given to all the resources of internet. The special software or application program with which you can access web is called "Web Browser".

# Introduction to Internet - WWW

- Configuring Web Browser
  - Search Engine is an application that allows you to search for content on the web. It displays multiple web pages based on the content or a word you have typed.
- Search Engines
  - Search Engine is an application that allows you to search for content on the web. It displays multiple web pages based on the content or a word you have typed.

# Introduction to Internet - WWW

- [Accessing Web Browser](#)
  - There are several ways to access a web page like using URLs, hyperlinks, using navigating tools, search engine, etc.

# Understanding URL

- Every document on the Web has a unique address.
- This address is known as Uniform Resource Locator (URL).
- Several HTML/XHTML tags include a URL attribute value, including hyperlinks, inline images, and forms.
- All of them use the same syntax to specify the location of a web resource, regardless of the type or content of that resource. That's why it is known a Uniform Resource Locator.

# Understanding URL

- URL Elements

- A URL is made of up several parts, each of which offers information to the web browser to help find the page. It is easier to learn the parts of a URL, if you look at the example URL given below.
- There are three key parts: the scheme, the host address, and the file path.

**`http://www.tutorialspoint.com/index.htm`**



# Understanding URL

- The Scheme
- The scheme identifies the type of protocol and URL you are linking to and therefore, how the resource should be retrieved.
- For example, most web browsers use Hypertext Transfer Protocol (HTTP) to pass information to communicate with the web servers and this is the reason a URL **starts with http://**.
- There are other schemes available and you can use either of them based on your requirement:

# Understanding URL

- **http://**
  - Hypertext Transfer Protocol (HTTP) is used to request pages from Web servers and send them back from Web servers to browsers.
- **https://**
  - Secure Hypertext Transfer Protocol (HTTPS) encrypts the data sent between the browser and the Web server using a digital certificate.
- **ftp://**
  - File Transfer Protocol is another method for transferring files on the Web. While HTTP is a lot more popular for viewing Web sites because of its integration with browsers, FTP is still commonly used protocol to transfer large files across the Web and to upload source files to your Web server.
- **file://**
  - Used to indicate that a file is on the local hard disk or a shared directory on a LAN.

# Understanding URL

- The Host Address
  - The host address is where a website can be found, either the IP address (four sets of numbers between 0 and 255, for example 68.178.157.132 ) or more commonly the domain name for a site such as [www.tutorialspoint.com](http://www.tutorialspoint.com).
  - Note that "www" is not actually part of the domain name although it is often used in the host address.

# Understanding URL

- The File Path
  - The filepath always begins with a forward slash character, and may consist of one or more directory or folder names.
  - Each directory name is separated by forward slash characters and the filepath may end with a filename at the end.
  - Here index.htm is the filename which is available in html directory:

**<https://www.tutorialspoint.com/html/index.htm>**

# Understanding URL

- Other Parts of the URL
  - Using credentials is a way of specifying a username and password for a password-protected part of a site.
  - The credentials come before the host address, and they are separated from the host address by an @ sign.
  - Note how the username is separated from the password by a colon.
  - The following URL shows the username *admin* and the password *admin123*:

**`https://admin:admin123@tutorialspoint.com/admin/index.htm`**

- Using the above URL, you can authenticate administrator and if provided ID and Password are correct then administrator will have access on index.htm file available in admin directory.

# Understanding URL

- You can use a telnet URL to connect to a server as follows :

**telnet://user:password@tutorialspoint.com:port/**

- Another important information is Web Server *Port Number*.
- By default HTTP Server runs on port number 80.
- But if you are running a server on any other port number then it can be provided as follows, assuming server is running on port 8080:

**https://www.tutorialspoint.com:8080/index.htm**

# Understanding URL

- *Fragment identifiers* can be used after a filename to indicate a specific part of the page that a browser should go immediately.
- Following is an example to reach to the top of page **html\_text\_links.htm**.

**[https://www.tutorialspoint.com/html/html\\_text\\_links.htm#top](https://www.tutorialspoint.com/html/html_text_links.htm#top)**

- You can pass some information to the server using URL. When you use a form on a webpage, such as a search form or an online order form, the browser can append the information you supply to the URL to pass information from your browser to the server as follows

**<https://www.tutorialspoint.com/bin/search.cgi?searchTerm=HTML>**

- Here, **searchTerm=HTML** is passed to the server where search.cgi script is used to parse this passed information and take further action.

# Understanding URL

- Absolute and Relative URLs
- You may address a URL in one of the following two ways:
- **Absolute** – An absolute URL is the complete address of a resource.
- For example

**`http://www.tutorialspoint.com/html/html_text_links.htm`**

- **Relative** – A relative URL indicates where the resource is in relation to the current page. Given URL is added with the `<base>` element to form a complete URL.
- For example **`/html/html_text_links.htm`**



# Introduction to HTML

- **HTML** stands for **Hyper Text Markup Language**, which is the most widely used language on Web to develop web pages.
- **HTML** was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995.
- HTML 4.01 was a major version of HTML and it was published in late 1999.
- Though HTML 4.01 version is widely used but currently we are having HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012.

# Introduction to HTML

Year	Version
1989	Tim Berners-Lee invented www
1991	Tim Berners-Lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML Working Group defined HTML 2.0
1997	W3C Recommendation: HTML 3.2
1999	W3C Recommendation: HTML 4.01
2000	W3C Recommendation: XHTML 1.0
2008	WHATWG HTML5 First Public Draft
2012	WHATWG HTML5 Living Standard
2014	W3C Recommendation: HTML5
2016	W3C Candidate Recommendation: HTML 5.1
2017	W3C Recommendation: HTML5.1 2nd Edition
2017	W3C Recommendation: HTML5.2

# Introduction to HTML

- HTML stands for **H**ypertext **M**arkup **L**anguage, and it is the most widely used language to write Web Pages.
- **Hypertext** refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a **Markup Language** which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.
- Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.
- Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

# Introduction to HTML

- Why to Learn HTML?
  - Originally, **HTML** was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.
  - Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.
  - **HTML** is a **MUST** for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain.

# Introduction to HTML

- **Some of the key advantages of learning HTML:**
  - **Create Web site** - You can create a website or customize an existing web template if you know HTML well.
  - **Become a web designer** - If you want to start a carrer as a professional web designer, HTML and CSS designing is a must skill.
  - **Understand web** - If you want to optimize your website, to boost its speed and performance, it is good to know HTML to yield best results.
  - **Learn other languages** - Once you understands the basic of HTML then other related technologies like javascript, php, or angular are become easier to understand.

# Introduction to HTML

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>This is document title</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>This is a heading</h1>
```

```
    <p>Hello World!</p>
```

```
  </body>
```

```
</html>
```

# Introduction to HTML

- **Applications of HTML**
- **Web pages development** - HTML is used to create pages which are rendered over the web. Almost every page of web is having html tags in it to render its details in browser.
- **Internet Navigation** - HTML provides tags which are used to navigate from one page to another and is heavily used in internet navigation.
- **Responsive UI** - HTML pages now-a-days works well on all platform, mobile, tabs, desktop or laptops owing to responsive design strategy.
- **Offline support** - HTML pages once loaded can be made available offline on the machine without any need of internet.
- **Game development**- HTML5 has native support for rich experience and is now useful in gaming development arena as well.

# Structure of an HTML Document

- An HTML document has two main parts: the **head** and the **body**. But firstly every HTML document should start by declaring that it is an HTML document.
- These tags are of the form:
  - `<html>` Should appear at the beginning of your document.
  - `</html>` Should appear at the end of your document.

## HTML Tags

- This leads us nicely on to say more about [X]HTML tags. All formatting tags (or elements) are of the general form:
- `<tag_on>` Switches the tag sequence on. For example, to **bold** some text add a `<strong>` at the beginning of the text.
- `</tag_off>` Switches the tag sequence off. The `tag_on` and `tag_off` tags are the same except the off tag has an / character in front of it.
- For example, to switch off the bolding add a `</strong>` character sequence at the end of the text that is to be given the attribute of bolding.



# HTML Tags

- HTML Tags
- As told earlier, HTML is a markup language and makes use of various tags to format the content. These tags are enclosed within angle braces **<TagName>**. Except few tags, most of the tags have their corresponding closing tags. For example, **<html>** has its closing tag **</html>** and **<body>** tag has its closing tag **</body>** tag etc.

# HTML Tags

- **<!DOCTYPE...>**
  - This tag defines the document type and HTML version.
- **<html>**
  - This tag encloses the complete HTML document and mainly comprises of document header which is represented by `<head>...</head>` and document body which is represented by `<body>...</body>` tags.
- **<head>**
  - This tag represents the document's header which can keep other HTML tags like `<title>`, `<link>` etc.

# HTML Tags

- **<title>**
  - The <title> tag is used inside the <head> tag to mention the document title.
- **<body>**
  - This tag represents the document's body which keeps other HTML tags like <h1>, <div>, <p> etc.
- **<h1>**
  - This tag represents the heading.
- **<p>**
  - This tag represents a paragraph.

# HTML Tags

- **DOCTYPE TAG**
- The HTML !DOCTYPE tag is the very first thing that every compliant web document should have.
- It's purpose is to inform the browser the type of document it's about to process.
- In HTML5, the !DOCTYPE declaration remains only for legacy reasons pertaining processing modes in browsers.
- The Document Type Declaration (DTD) is unique for each version of HTML and must be written exactly as it is, to be of some use.
- For this reason, it's better if you just copy and paste the code for the declararion you're about to use.

# HTML Tags

- Below is a list of DTDs for all the versions of HTML (XHTML included), and others like SVG and MathML. But remember that, as of today, there's no need to use any other declaration than the one for HTML5, as this is, by far, the preferred version (unless you're working on a very particular project).
- **HTML5**
- `<!DOCTYPE html>`
- **XHTML 1.1**
- `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">`

# Introduction to Web Technologies

- **HTML (Hyper Text Mark-up Language)** is the glue which holds together every internet site.
- You construct a foundation, like building a home. HTML is the base. HTML is an open source speech (i.e. not possessed by anybody), which is not difficult to learn, and needs no more elaborate (or expensive! ) ) Packs to begin with it.
- All you will need is something to sort with, for example, Windows Notepad, and also a great deal of patience and time.
- HTML functions on a 'label' system, in which the material is effected by every label.
- It is although confined alone. Like your house's base, HTML is strong enough to help many types of languages.

# Introduction to Web Technologies

- **JavaScript** : JavaScript is a dynamic computer programming language. 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.
- It is an interpreted programming language with object-oriented capabilities.
- JavaScript was first known as **LiveScript**, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java.
- Its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**.
- The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.
- The [ECMA-262 Specification](#) defined a standard version of the core JavaScript language.
  - JavaScript is a lightweight, interpreted programming language.
  - Designed for creating network-centric applications.
  - Complementary to and integrated with Java.
  - Complementary to and integrated with HTML.
  - Open and cross-platform

# Introduction to Web Technologies

- **CSS : Cascading Style Sheets**, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.
- CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.
- CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document.
- Most commonly, CSS is combined with the markup languages HTML or XHTML.



# Introduction to Web Technologies

- **XML** stands for **Extensible Markup Language**. It is a text-based markup language derived from Standard Generalized Markup Language (SGML).
- XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data.
- XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML.
- There are three important characteristics of XML
  - **XML is extensible** – XML allows you to create your own self-descriptive tags, or language, that suits your application.
  - **XML carries the data, does not present it** – XML allows you to store the data irrespective of how it will be presented.
  - **XML is a public standard** – XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

# Introduction to Web Technologies

- **XHTML**
- XHTML stands for **EX**tensible **HyperText Markup Language**. It is the next step in the evolution of the internet. The XHTML 1.0 is the first document type in the XHTML family.
- XHTML is almost identical to HTML 4.01 with only few differences. This is a cleaner and stricter version of HTML 4.01. If you already know HTML, then you need to give little attention to learn this latest version of HTML.
- XHTML was developed by World Wide Web Consortium (W3C) to help web developers make the transition from HTML to XML. By migrating to XHTML today, web developers can enter the XML world with all of its benefits, while still remaining confident in the backward and future compatibility of the content.

# Introduction to Web Technologies

- **AJAX**
- AJAX is a web development technique for creating interactive web applications. If you know JavaScript, HTML, CSS, and XML, then you need to spend just one hour to start with AJAX.
- AJAX is Based on Open Standards
- AJAX is based on the following open standards –
- Browser-based presentation using HTML and Cascading Style Sheets (CSS).
- Data is stored in XML format and fetched from the server.
- Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.
- JavaScript to make everything happen.

# Introduction to Web Technologies

- **ASP.NET** is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC, as well as mobile devices.
- ASP.NET is a part of Microsoft .Net platform.
- ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework.
- These codes can use the entire hierarchy of classes in .Net framework.
- The ASP.NET application codes can be written in any of the following languages:
  - C#, Visual Basic.Net, Jscript, J#
- ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages.

# Introduction to Web Technologies

- PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.
- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

# Introduction to Web Technologies

- **Web services** are open standard (XML, SOAP, HTTP, etc.) based web applications that interact with other web applications for the purpose of exchanging data. Web services can convert your existing applications into web applications.

# Unit-2

## Basic HTML Tags

# <Head> Tag

- The <head> tag is a container of various important tags like <title>, <meta>, <link>, <base>, <style>, <script>, and <noscript> tags.
- **The HTML <title> Tag**
  - The HTML <title> tag is used for specifying the title of the HTML document. Following is an example to give a title to an HTML document .
- **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.



# <Head> Tag

- **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 –

# <Head> Tag

- **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 –

- **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 –

# <Head> Tag

- **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 simple JavaScript function –

# <Body> Tag

- The HTML <body> tag is used for indicating the main content section of the HTML document.
- The HTML <section> tag specifies a section in a document.

# <Body> Tag

Attribute	Value	Description
alink	rgb(x,x,x) #xxxxxx colorname	<i>Deprecated</i> – Specifies the color of the active links in the document.
background	URL	<i>Deprecated</i> – Specifies the background image file path.
bgcolor	rgb(x,x,x) #xxxxxx colorname	<i>Deprecated</i> – Specifies the background color.
link	rgb(x,x,x) #xxxxxx colorname	<i>Deprecated</i> – Specifies the color of all the links in the document.
text	rgb(x,x,x) #xxxxxx colorname	<i>Deprecated</i> – Specifies the color of the text in the document.
vlink	rgb(x,x,x) #xxxxxx colorname	<i>Deprecated</i> – Specifies the color of the visited links in the document.

# HTML Elements

- All the HTML elements can be categorized into two categories (a) Block Level Elements (b) Inline Elements.
- **Block Elements**
  - Block elements appear on the screen as if they have a line break before and after them. For example, the <p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <ul>, <ol>, <dl>, <pre>, <hr />, <blockquote>, and <address> elements are all block level elements. They all start on their own new line, and anything that follows them appears on its own new line.
- **Inline Elements (Text Level)**
  - Inline elements, on the other hand, can appear within sentences and do not have to appear on a new line of their own. The <b>, <i>, <u>, <em>, <strong>, <sup>, <sub>, <big>, <small>, <li>, <ins>, <del>, <code>, <cite>, <dfn>, <a>, <mark> <kbd>, <img> and <var> elements are all inline elements.

# HTML Elements

- **Grouping HTML Elements**

- There are two important tags which we use very frequently to group various other HTML tags (i) `<div>` tag and (ii) `<span>` tag

- **The `<div>` tag**

- This is the very important block level tag which plays a big role in grouping various other HTML tags and applying CSS on group of elements.
- Even now `<div>` tag can be used to create webpage layout where we define different parts (Left, Right, Top etc.) of the page using `<div>` tag.
- This tag does not provide any visual change on the block but this has more meaning when it is used with CSS.

# HTML Elements

- **The <span> tag**

- The HTML <span> is an inline element and it can be used to group inline-elements in an HTML document. This tag also does not provide any visual change on the block but has more meaning when it is used with CSS.
- **The difference between the <span> tag and the <div> tag is that the <span> tag is used with inline elements whereas the <div> tag is used with block-level elements.**



# HTML Elements

- **Line Break Tag**

- Whenever you use the `<br />` element, anything following it starts from the next line.
- This tag is an example of an **empty** element, where you do not need opening and closing tags, as there is nothing to go in between them.

- **Centering Content**

- You can use `<center>` tag to put any content in the center of the page or any table cell.

# HTML Elements

- **Horizontal Lines**
- Horizontal lines are used to visually break-up sections of a document. The `<hr>` tag creates a line from the current position in the document to the right margin and breaks the line accordingly.
- Again `<hr />` tag is an example of the **empty** element, where you do not need opening and closing tags, as there is nothing to go in between them.
- The `<hr />` element has a space between the characters `hr` and the forward slash. If you omit this space, older browsers will have trouble rendering the horizontal line, while if you miss the forward slash character and just use `<hr>` it is not valid in XHTML

# HTML Elements

- **Preserve Formatting**

- Sometimes, you want your text to follow the exact format of how it is written in the HTML document. In these cases, you can use the preformatted tag **<pre>**.
- Any text between the opening **<pre>** tag and the closing **</pre>** tag will preserve the formatting of the source document.

- **Nonbreaking Spaces**

- Suppose you want to use the phrase "12 Angry Men." Here, you would not want a browser to split the "12, Angry" and "Men" across two lines –
- An example of this technique appears in the movie "12 Angry Men." In cases, where you do not want the client browser to break text, you should use a nonbreaking space entity **&nbsp;**; instead of a normal space. For example, when coding the "12 Angry Men" in a paragraph,

# HTML Elements

- An **HTML element** is defined by a starting tag. If the element contains other content, it ends with a closing tag,
- So here `<p>....</p>` is an HTML element, `<h1>...</h1>` is another HTML element. There are some HTML elements which don't need to be closed, such as `<img.../>`, `<hr />` and `<br />` elements. These are known as **void elements**.
- HTML documents consists of a tree of these elements and they specify how HTML documents should be built, and what kind of content should be placed in what part of an HTML document.

# HTML Elements

- HTML Tag vs. Element

- An HTML element is defined by a *starting tag*. If the element contains other content, it ends with a *closing tag*.
- For example, **<p>** is starting tag of a paragraph and **</p>** is closing tag of the same paragraph but **<p>This is paragraph</p>** is a paragraph element.

- Nested HTML Elements

- It is very much allowed to keep one HTML element inside another HTML element

**Ex: <h1>This is <i>italic</i> heading</h1>**

# HTML - Attributes

- We have seen few HTML tags and their usage like heading tags **<h1>**, **<h2>**, paragraph tag **<p>** and other tags. We used them so far in their simplest form, but most of the HTML tags can also have attributes, which are extra bits of information.
- An attribute is used to define the characteristics of an HTML element and is placed inside the element's opening tag. All attributes are made up of two parts – a **name** and a **value**
  - The **name** is the property you want to set. For example, the paragraph **<p>** element in the example carries an attribute whose name is **align**, which you can use to indicate the alignment of paragraph on the page.
  - The **value** is what you want the value of the property to be set and always put within quotations. The below example shows three possible values of align attribute: **left**, **center** and **right**.

# HTML - Attributes

- Core Attributes
- The four core attributes that can be used on the majority of HTML elements (although not all) are
  -
- Id
- Title
- Class
- Style

# HTML - Attributes

- **The Id Attribute**
- The **id** attribute of an HTML tag can be used to uniquely identify any element within an HTML page. There are two primary reasons that you might want to use an id attribute on an element –
- If an element carries an id attribute as a unique identifier, it is possible to identify just that element and its content.
- If you have two elements of the same name within a Web page (or style sheet), you can use the id attribute to distinguish between elements that have the same name.
- We will discuss style sheet in separate tutorial. For now, let's use the id attribute to distinguish between two paragraph elements as shown below.



# HTML - Attributes

- **The title Attribute**
- The **title** attribute gives a suggested title for the element. The syntax for the **title** attribute is similar as explained for **id** attribute –
- The behavior of this attribute will depend upon the element that carries it, although it is often displayed as a tooltip when cursor comes over the element or while the element is loading.
- **The class Attribute**
- The **class** attribute is used to associate an element with a style sheet, and specifies the class of element. You will learn more about the use of the class attribute when you will learn Cascading Style Sheet (CSS). So for now you can avoid it.
- The value of the attribute may also be a space-separated list of class names. For example –

# HTML - Attributes

- **The style Attribute**

- The style attribute allows you to specify Cascading Style Sheet (CSS) rules within the element.

# HTML - Attributes

- Internationalization Attributes

- There are three internationalization attributes, which are available for most (although not all) XHTML elements.
  - dir
  - lang
  - xml:lang
- **The dir Attribute**
  - The **dir** attribute allows you to indicate to the browser about the direction in which the text should flow. The dir attribute can take one of two values, as you can see in the table that follows

Value	Meaning
ltr	Left to right (the default value)
rtl	Right to left (for languages such as Hebrew or Arabic that are read right to left)

# HTML - Attributes

- When *dir* attribute is used within the <html> tag, it determines how text will be presented within the entire document. When used within another tag, it controls the text's direction for just the content of that tag.
- **The lang Attribute**
  - The **lang** attribute allows you to indicate the main language used in a document, but this attribute was kept in HTML only for backwards compatibility with earlier versions of HTML.
  - This attribute has been replaced by the **xml:lang** attribute in new XHTML documents.
  - The values of the *lang* attribute are ISO-639 standard two-character language codes. Check [HTML Language Codes: ISO 639](#) for a complete list of language codes.

# HTML - Attributes

- **The `xml:lang` Attribute**

- The *xml:lang* attribute is the XHTML replacement for the *lang* attribute.
- The value of the *xml:lang* attribute should be an ISO-639 country code as mentioned in previous section.

- **Generic Attributes**

- Here's a table of some other attributes that are readily usable with many of the HTML tags.

# HTML - Attributes

Attribute	Options	Function
align	right, left, center	Horizontally aligns tags
valign	top, middle, bottom	Vertically aligns tags within an HTML element.
bgcolor	numeric, hexadecimal, RGB values	Places a background color behind an element
background	URL	Places a background image behind an element
id	User Defined	Names an element for use with Cascading Style Sheets.
class	User Defined	Classifies an element for use with Cascading Style Sheets.
width	Numeric Value	Specifies the width of tables, images, or table cells.
height	Numeric Value	Specifies the height of tables, images, or table cells.
title	User Defined	"Pop-up" title of the elements.

# HTML – Formatting Tags

- Bold Text
- Anything that appears within `<b>...</b>` element, is displayed in bold
- Italic Text
- Anything that appears within `<i>...</i>` element is displayed in italicized
- Underlined Text
- Anything that appears within `<u>...</u>` element, is displayed with underline
- Strike Text
- Anything that appears within `<strike>...</strike>` element is displayed with strikethrough, which is a thin line through the text

# HTML – Formatting Tags

- Monospaced Font
- The content of a **<tt>...</tt>** element is written in monospaced font. Most of the fonts are known as variable-width fonts because different letters are of different widths (for example, the letter 'm' is wider than the letter 'i'). In a monospaced font, however, each letter has the same width.
- Superscript Text
- The content of a **<sup>...</sup>** element is written in superscript; the font size used is the same size as the characters surrounding it but is displayed half a character's height above the other characters.
- Subscript Text
- The content of a **<sub>...</sub>** element is written in subscript; the font size used is the same as the characters surrounding it, but is displayed half a character's height beneath the other characters



# HTML – Formatting Tags

- Inserted Text
- Anything that appears within **<ins>...</ins>** element is displayed as inserted text.
- Deleted Text
- Anything that appears within **<del>...</del>** element, is displayed as deleted text.
- Larger Text
- The content of the **<big>...</big>** element is displayed one font size larger than the rest of the text surrounding
- Smaller Text
- The content of the **<small>...</small>** element is displayed one font size smaller than the rest of the text surrounding

# HTML – Phrase Tags

- Emphasized Text
- Anything that appears within `<em>...</em>` element is displayed as emphasized text.
- Marked Text
- Anything that appears with-in `<mark>...</mark>` element, is displayed as marked with yellow ink.
- Strong Text
- Anything that appears within `<strong>...</strong>` element is displayed as important text.
- Text Abbreviation
- You can abbreviate a text by putting it inside opening `<abbr>` and closing `</abbr>` tags. If present, the title attribute must contain this full description and nothing else.

# HTML – Formatting Tags

- Acronym Element
- The **<acronym>** element allows you to indicate that the text between **<acronym>** and **</acronym>** tags is an acronym.
- At present, the major browsers do not change the appearance of the content of the **<acronym>** element.
- Text Direction
- The **<bdo>...</bdo>** element stands for Bi-Directional Override and it is used to override the current text direction.
- Special Terms
- The **<dfn>...</dfn>** element (or HTML Definition Element) allows you to specify that you are introducing a special term. It's usage is similar to italic words in the midst of a paragraph.
- Typically, you would use the **<dfn>** element the first time you introduce a key term. Most recent browsers render the content of a **<dfn>** element in an italic font.

# HTML – Formatting Tags

- Quoting Text
- When you want to quote a passage from another source, you should put it in between **<blockquote>...</blockquote>** tags.
- Text inside a **<blockquote>** element is usually indented from the left and right edges of the surrounding text, and sometimes uses an italicized font.
- Short Quotations
- The **<q>...</q>** element is used when you want to add a double quote within a sentence.
- Text Citations
- If you are quoting a text, you can indicate the source placing it between an opening **<cite>** tag and closing **</cite>** tag
- As you would expect in a print publication, the content of the **<cite>** element is rendered in italicized text by default.

# HTML – Formatting Tags

- Computer Code
- Any programming code to appear on a Web page should be placed inside **<code>...</code>** tags. Usually the content of the **<code>** element is presented in a monospaced font, just like the code in most programming books.
- Keyboard Text
- When you are talking about computers, if you want to tell a reader to enter some text, you can use the **<kbd>...</kbd>** element to indicate what should be typed in, as in this example.
- Programming Variables
- This element is usually used in conjunction with the **<pre>** and **<code>** elements to indicate that the content of that element is a variable.

# HTML – Formatting Tags

- Program Output
- The **<samp>...</samp>** element indicates sample output from a program, and script etc. Again, it is mainly used when documenting programming or coding concepts.
- Address Text
- The **<address>...</address>** element is used to contain any address.

# HTML – Comments

- **Comment** is a piece of code which is ignored by any web browser. It is a good practice to add comments into your HTML code, especially in complex documents, to indicate sections of a document, and any other notes to anyone looking at the code. Comments help you and others understand your code and increases code readability.
  - HTML comments are placed in between `<!-- ... -->` tags. So, any content placed with-in `<!-- ... -->` tags will be treated as comment and will be completely ignored by the browser.
- **Valid vs Invalid Comments**
  - Comments do not nest which means a comment cannot be put inside another comment. Second the double-dash sequence `--` may not appear inside a comment except as part of the closing `-->` tag. You must also make sure that there are no spaces in the start-of comment string.

# HTML – Comments

- **Multiline Comments**

- So far we have seen single line comments, but HTML supports multi-line comments as well.
- You can comment multiple lines by the special beginning tag `<!--` and ending tag `-->` placed before the first line and end of the last line

- **Conditional Comments**

- Conditional comments only work in Internet Explorer (IE) on Windows but they are ignored by other browsers. They are supported from Explorer 5 onwards, and you can use them to give conditional instructions to different versions of IE.

- **Using Comment Tag**

- There are few browsers that support `<comment>` tag to comment a part of HTML code.



# HTML - Images

- Images are very important to beautify as well as to depict many complex concepts in simple way on your web page. This tutorial will take you through simple steps to use images in your web pages.
- **Insert Image**
  - You can insert any image in your web page by using **<img>** tag. Following is the simple syntax to use this tag.
  - `<img src = "Image URL" ... attributes-list/>` The **<img>** tag is an empty tag, which means that, it can contain only list of attributes and it has no closing tag.

# HTML - Images

- You can use PNG, JPEG or GIF image file based on your comfort but make sure you specify correct image file name in **src** attribute. Image name is always case sensitive.
- The **alt** attribute is a mandatory attribute which specifies an alternate text for an image, if the image cannot be displayed.

# HTML - Images

- **Set Image Location**
- Usually we keep all the images in a separate directory. So let's keep HTML file test.htm in our home directory and create a subdirectory **images** inside the home directory where we will keep our image test.png.

```

```

# HTML - Images

- **Set Image Width/Height**
- You can set image width and height based on your requirement using **width** and **height** attributes. You can specify width and height of the image in terms of either pixels or percentage of its actual size.

```
<img src = "/html/images/test.png" alt = "Test Image" width = "150" height = "100"/>
```

# HTML - Images

- **Set Image Border**
- By default, image will have a border around it, you can specify border thickness in terms of pixels using border attribute. A thickness of 0 means, no border around the picture.

```
<img src = "/html/images/test.png" alt = "Test Image" border = "3"/>
```

# HTML - Images

- Set Image Alignment
- By default, image will align at the left side of the page, but you can use **align** attribute to set it in the center or right.

```
<img src = "/html/images/test.png" alt = "Test  
Image" border = "3" align = "right"/>
```

# HTML - Tables

- The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells.
- The HTML tables are created using the **<table>** tag in which the **<tr>** tag is used to create table rows and **<td>** tag is used to create data cells.
- The elements under **<td>** are regular and left aligned by default.
- The **border** is an attribute of **<table>** tag and it is used to put a border across all the cells. If you do not need a border, then you can use **border = "0"**.

# HTML - Tables

- **Table Heading**
- Table heading can be defined using **<th>** tag.
- This tag will be put to replace **<td>** tag, which is used to represent actual data cell.
- Normally you will put your top row as table heading as shown below, otherwise you can use **<th>** element in any row.
- Headings, which are defined in **<th>** tag are centered and bold by default.



# HTML - Tables

- **Cellpadding and Cellspacing Attributes**
- There are two attributes called *cellpadding* and *cellspacing* which you will use to adjust the white space in your table cells.
- The cellspacing attribute defines space between table cells, while cellpadding represents the distance between cell borders and the content within a cell.

# HTML - Tables

- **Colspan and Rowspan Attributes**

- You will use **colspan** attribute if you want to merge two or more columns into a single column.
- Similar way you will use **rowspan** if you want to merge two or more rows.

- **Tables Backgrounds**

- You can set table background using one of the following two ways –
  - **bgcolor** attribute – You can set background color for whole table or just for one cell.
  - **background** attribute – You can set background image for whole table or just for one cell.
- You can also set border color also using **bordercolor** attribute.

# HTML - Tables

- **Table Height and Width**

- You can set a table width and height using **width** and **height** attributes.
- You can specify table width or height in terms of pixels or in terms of percentage of available screen area.

- **Table Caption**

- The **caption** tag will serve as a title or explanation for the table and it shows up at the top of the table.
- This tag is deprecated in newer version of HTML/XHTML.

# HTML - Tables

- **Table Header, Body, and Footer**
  - Tables can be divided into three portions – a header, a body, and a foot.
  - The head and foot are rather similar to headers and footers in a word-processed document that remain the same for every page, while the body is the main content holder of the table.
- The three elements for separating the head, body, and foot of a table are –
  - **<thead>** – to create a separate table header.
  - **<tbody>** – to indicate the main body of the table.
  - **<tfoot>** – to create a separate table footer.

# HTML - Tables

- A table may contain several `<tbody>` elements to indicate *different pages* or groups of data.
- But it is notable that `<thead>` and `<tfoot>` tags should appear before `<tbody>`
- **Nested Tables**
  - You can use one table inside another table. Not only tables you can use almost all the tags inside table data tag `<td>`.

# HTML - Tables

# HTML - Tables

# HTML - Tables



# HTML - Tables

# HTML - Tables

# Unit-3

## Introduction to HTML5

# Introduction to HTML5

- HTML5 is the next major revision of the HTML standard superseding HTML 4.01, XHTML 1.0, and XHTML 1.1. HTML5 is a standard for structuring and presenting content on the World Wide Web.
- HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).
- The new standard incorporates features like video playback and drag-and-drop that have been previously dependent on third-party browser plug-ins such as Adobe Flash, Microsoft Silverlight, and Google Gears.

# Introduction to HTML5

- **Browser Support**

- The latest versions of Apple Safari, Google Chrome, Mozilla Firefox, and Opera all support many HTML5 features and Internet Explorer 9.0 will also have support for some HTML5 functionality.
- The mobile web browsers that come pre-installed on iPhones, iPads, and Android phones all have excellent support for HTML5.

# Features of HTML5

- HTML5 introduces a number of new elements and attributes that can help you in building modern websites. Here is a set of some of the most prominent features introduced in HTML5.
- **New Semantic Elements** – These are like `<header>`, `<footer>`, and `<section>`.
- **Forms 2.0** – Improvements to HTML web forms where new attributes have been introduced for `<input>` tag.
- **Persistent Local Storage** – To achieve without resorting to third-party plugins.
- **WebSocket** – A next-generation bidirectional communication technology for web applications.

# Features of HTML5

- **Server-Sent Events** – HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).
- **Canvas** – This supports a two-dimensional drawing surface that you can program with JavaScript.
- **Audio & Video** – You can embed audio or video on your webpages without resorting to third-party plugins.
- **Geolocation** – Now visitors can choose to share their physical location with your web application.
- **Microdata** – This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
- **Drag and drop** – Drag and drop the items from one location to another location on the same webpage.

# Features of HTML5

- Backward Compatibility
  - HTML5 is designed, as much as possible, to be backward compatible with existing web browsers. Its new features have been built on existing features and allow you to provide fallback content for older browsers.
  - It is suggested to detect support for individual HTML5 features using a few lines of JavaScript.
  - If you are not familiar with any previous version of HTML, I would recommend that you go through our **HTML Tutorial** before exploring the features of HTML5.



# Basic Structure of HTML5

- The HTML 5 language has a "custom" HTML syntax that is compatible with HTML 4 and XHTML1 documents published on the Web, but is not compatible with the more esoteric SGML features of HTML 4.
- HTML 5 does not have the same syntax rules as XHTML where we needed lower case tag names, quoting our attributes, an attribute had to have a value and to close all empty elements.

# Basic Structure of HTML5

- HTML5 structure comes with a lot of flexibility and it supports the following features –
  - Uppercase tag names.
  - Quotes are optional for attributes.
  - Attribute values are optional.
  - Closing empty elements are optional.

# Basic Structure of HTML5

- **The DOCTYPE**

- DOCTYPEs in older versions of HTML were longer because the HTML language was SGML based and therefore required a reference to a DTD.
- HTML 5 authors would use simple syntax to specify DOCTYPE as follows –
- `<!DOCTYPE html>` The above syntax is case-insensitive.

# Basic Structure of HTML5

- **HTML5 Elements**

- HTML5 elements are marked up using start tags and end tags. Tags are delimited using angle brackets with the tag name in between.
- The difference between start tags and end tags is that the latter includes a slash before the tag name.
- Following is the example of an HTML5 element –
  - `<p>...</p>` HTML5 tag names are case insensitive and may be written in all uppercase or mixed case, although the most common convention is to stick with lowercase.
  - Most of the elements contain some content like `<p>...</p>` contains a paragraph.
  - Some elements, however, are forbidden from containing any content at all and these are known as void elements. For example, **br**, **hr**, **link**, **meta**, etc.

# Basic Structure of HTML5

- **HTML5 Attributes**

- Elements may contain attributes that are used to set various properties of an element.
- Some attributes are defined globally and can be used on any element, while others are defined for specific elements only. All attributes have a name and a value and look like as shown below in the example.
- Following is the example of an HTML5 attribute which illustrates how to mark up a div element with an attribute named class using a value of "example" –  
**<div class = "example">...</div>**

# Basic Structure of HTML5

- Attributes may only be specified within start tags and must never be used in end tags.
- HTML5 attributes are case insensitive and may be written in all uppercase or mixed case, although the most common convention is to stick with lowercase.


# Validating HTML5 Document

- There are only few validators available on the net.
- But following two validators are very authentic and can be used –

## (1) The W3C Markup Validator

- The [W3C Markup Validator](https://validator.w3.org/#validate_by_uri+with_options) [[https://validator.w3.org/#validate\\_by\\_uri+with\\_options](https://validator.w3.org/#validate_by_uri+with_options)] checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. This validator is part of Unicorn, W3C's unified validator service.
- To use this validator for HTML5, you need to use **More Options** and select **Document Type** as **HTML5 (experimental)** as shown below.

# Validating HTML5 Document

**Markup Validation Service**  
Check the markup (HTML, XHTML, ...) of Web documents

Validate by URI

Validate by File Upload

Validate by Direct Input

### Validate by URI

Validate a document online:

Address:

▼ More Options

**Character Encoding**

☐ Only if missing

**Document Type**

☐ Only if missing

☒ List Messages Sequentially

☐ Group Error Messages by Type

☐ Show Source

☐ Clean up Markup with HTML Tidy

☐ Show Outline

☐ Validate error pages

☐ Verbose Output



# Validating HTML5 Document

## (2) The Validator.nu (X)HTML5 Validator

- Here is the another currently known HTML5 validators: Henri's [Validator.nu \(X\)HTML5 Validator](https://html5.validator.nu/) [https://html5.validator.nu/] (Highly Experimental) –
- This validator has compatibility problem, so you can try it in lower versions of IE or Mozilla.

# Section Elements

- The following tags have been introduced for better structure
- **Body** – It is use to define content of HTML page
- **section** – This tag represents a generic document or application section. It can be used together with h1-h6 to indicate the document structure.
- **article** – This tag represents an independent piece of content of a document, such as a blog entry or newspaper article.
- **aside** – This tag represents a piece of content that is only slightly related to the rest of the page.
- **header** – This tag represents the header of a section.

# Section Elements

- **footer** – This tag represents a footer for a section and can contain information about the author, copyright information, et cetera.
- **nav** – This tag represents a section of the document intended for navigation.
- **dialog** – This tag can be used to mark up a conversation.
- **figure** – This tag can be used to associate a caption together with some embedded content, such as a graphic or video.
- **Address**- It is normally define at the haeder or footer of an HTML page and use to display information such as contact.

# HTML5 Elements

- **Root Element**

- `<HTML>` is the root element which comes after `<!DOCTYPE>` element and within which other HTML elements are specified.

- **Metadata Elements**

- HTML metadata is data about the HTML document. Metadata is not displayed.
- Metadata typically define the document title, character set, styles, scripts, and other meta information.
- The `<head>` element is a container for metadata (data about data)
- The `<head>` element is placed between the `<html>` tag and the `<body>` tag

# HTML5 Elements

**Following are the Meta elements:**

- 1) Title
- 2) Base
- 3) Link
- 4) Command
- 5) Meta
- 6) Script
- 7) NoScript
- 8) Style

# HTML5 Elements

- The **<title>** tag defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.
- The **<title>** tag is required in HTML documents!
  - The contents of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.
- The **<title>** element:
  - defines a title in the browser toolbar
  - provides a title for the page when it is added to favorites
  - displays a title for the page in search-engine results

# HTML5 Elements

- Here are some tips for creating good **titles**:
  - Go for a longer, descriptive title (avoid one- or two-word titles)
  - Search engines will display about 50-60 characters of the title, so try not to have titles longer than that
  - Do not use just a list of words as the title (this may reduce the page's position in search results)
  - So, try to make the title as accurate and meaningful as possible!
- Attributes of the title element:
  - Class
  - Id
  - Lang
  - Style

# HTML5 Elements

- Definition and Usage
  - The <base> tag specifies the base URL and/or target for all relative URLs in a document.
  - The <base> tag must have either an href or a target attribute present, or both.
  - There can only be one single <base> element in a document, and it must be inside the <head> element.

Attribute	Value	Description
<a href="#"><u>href</u></a>	<i>URL</i>	Specifies the base URL for all relative URLs in the page
<a href="#"><u>target</u></a>	<i>_blank</i> <i>_parent</i> <i>_self</i> <i>_top</i>	Specifies the default target for all hyperlinks and forms in the page



# HTML5 Elements

- Definition and Usage

- The `<link>` tag defines the relationship between the current document and an external resource.
- The `<link>` tag is most often used to link to external style sheets.
- The `<link>` element is an empty element, it contains attributes only.

**`<head>`**

**`<link rel="stylesheet" href="styles.css">`**

**`</head>`**

# HTML5 Elements

Link Attribute	Value	Description
crossorigin	anonymous use-credentials	Specifies how the element handles cross-origin requests
<a href="#"><u>href</u></a>	<i>URL</i>	Specifies the location of the linked document
<a href="#"><u>hreflang</u></a>	<i>language_code</i>	Specifies the language of the text in the linked document
<a href="#"><u>media</u></a>	<i>media_query</i>	Specifies on what device the linked document will be displayed
referrerpolicy	no-referrer no-referrer-when-downgrade origin origin-when-cross-origin unsafe-url	Specifies which referrer to use when fetching the resource

# HTML5 Elements

Link Attribute	Value	Description
<a href="#"><u>rel</u></a>	Alternate, author, dns-prefetch, help, icon, license, next, pingback, preconnect, prefetch, preload, prerender, prev, search, stylesheet,	Required. Specifies the relationship between the current document and the linked document
<a href="#"><u>sizes</u></a>	<i>HeightxWidth</i> any	Specifies the size of the linked resource. Only for rel="icon"
title		Defines a preferred or an alternate stylesheet
<a href="#"><u>type</u></a>	<i>media_type</i>	Specifies the media type of the linked document

# HTML5 Elements

- **Definition and Usage <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.
- There is a method to let web designers take control over the viewport (the user's visible area of a web page), through the <meta> tag (See "Setting The Viewport" example below).

# HTML5 Elements

Meta Attribute	Value	Description
<a href="#"><u>charset</u></a>	<i>character_set</i>	Specifies the character encoding for the HTML document
<a href="#"><u>content</u></a>	<i>text</i>	Specifies the value associated with the http-equiv or name attribute
<a href="#"><u>http-equiv</u></a>	content-security-policy content-type default-style refresh	Provides an HTTP header for the information/value of the content attribute
<a href="#"><u>name</u></a>	application-name author description generator keywords viewport	Specifies a name for the metadata

# HTML5 Elements

- **Define keywords for search engines:**

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

- **Define a description of your web page:**

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

- **Define the author of a page:**

```
<meta name="author" content="John Doe">
```

- **Refresh document every 30 seconds:**

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

- **Setting the viewport to make your website look good on all devices:**

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

# HTML5 Elements

- **Definition and Usage HTML <command>**
  - The type attribute specifies the type of command.
- Syntax

**<command type="command|checkbox|radio">**

**Ex:**

**<menu>**

**<command type="command" label="Save" onclick="save()">**

**Save </command> </menu>**

Value	Description
command	Default. Specifies a normal command with an action
checkbox	Specifies a command that can be toggled using a checkbox
radio	Specifies a command that can be toggled using a radio button

# HTML5 Elements

- **Definition and Usage <noscript>**

- 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.
- The <noscript> element can be used in both <head> and <body>. When used inside <head>, the <noscript> element could only contain <link>, <style>, and <meta> elements.
- Ex:

**<script>**

**document.write("Hello World!")**

**</script>**

**<noscript>Your browser does not support JavaScript!</noscript>**



# HTML5 Elements

- **Definition and Usage <script>**
  - The <script> tag is used to embed a client-side script (JavaScript).
  - The <script> element either contains scripting statements, or it points to an external script file through the src attribute.
  - Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.

Example:

```
<script>  
  document.getElementById("demo").innerHTML = "Hello  
  JavaScript!";  
</script>
```

# HTML5 Elements

Script Attribute	Value	Description
<a href="#"><u>async</u></a>	async	Specifies that the script is executed asynchronously (only for external scripts)
<a href="#"><u>charset</u></a>	<i>charset</i>	Specifies the character encoding used in an external script file
<a href="#"><u>defer</u></a>	defer	Specifies that the script is executed when the page has finished parsing (only for external scripts)
<a href="#"><u>src</u></a>	<i>URL</i>	Specifies the URL of an external script file
<a href="#"><u>type</u></a>	<i>media_type</i>	Specifies the media type of the script

# HTML5 Elements

- Definition and Usage <style>
  - The <style> tag is used to define style information (CSS) for a document.
  - Inside the <style> element you specify how HTML elements should render in a browser.

Attribute	Value	Description
<a href="#">media</a>	<i>media_query</i>	Specifies what media/device the media resource is optimized for
<a href="#">type</a>	text/css	Specifies the media type of the <style> tag
Scoped	Scoped	Specifies the style elements, that is the parent element and its child element.

# Heading Elements

- The HTML <h1> to <h6> tag is used to define headings in an HTML document. <h1> defines largest heading and <h6> defines smallest heading.
- Specific Attributes
- The HTML <h1> to <h6> tag also supports the following additional attributes –

Attribute	Value	Description
align	left right center justify	<i>Deprecated</i> – Specifies the alignment of the content enclosed.

# HTML5 Flow Elements

**Flow elements are used in the body of HTML documents.**

Tag	Description
<a>	Specifies an anchor
<abbr>	Specifies an abbreviation
<address>	Specifies an address element
<area>	Specifies an area inside an image map
<article>	New Tag: Specifies an independent piece of content of a document, such as a blog entry or newspaper article
<aside>	New Tag: Specifies a piece of content that is only slightly related to the rest of the page.
<audio>	New Tag: Specifies an audio file.

# HTML5 Flow Elements

Tag	Description
<code>&lt;b&gt;</code>	Represents the bold text.
<code>&lt;bdo&gt;</code>	Specifies the direction of text display
<code>&lt;blockquote&gt;</code>	Specifies a long quotation
<code>&lt;br&gt;</code>	Inserts a single line break
<code>&lt;button&gt;</code>	Specifies a push button
<code>&lt;canvas&gt;</code>	New Tag:This is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
<code>&lt;cite&gt;</code>	Specifies a citation
<code>&lt;code&gt;</code>	Specifies computer code text

# HTML5 Flow Elements

Tag	Description
<command>	New Tag:Specifies a command the user can invoke.
<datalist>	New Tag:Together with the a new list attribute for input can be used to make comboboxes
<dfn>	Specifies a definition term
<del>	Specifies deleted text
<details>	New Tag:Specifies additional information or controls which the user can obtain on demand.
<dl>	Specifies a definition list
<div>	Specifies a section in a document

# HTML5 Flow Elements

Tag	Description
<embed>	New Tag:Defines external interactive content or plugin.
<footer>	New Tag:Specifies a footer for a section and can contain information about the author, copyright information, et cetera.
<em>	Specifies emphasized text
<fieldset>	Specifies a fieldset
<figure>	New Tag:Specifies a piece of self-contained flow content, typically referenced as a single unit from the main flow of the document.
<form>	Specifies a form
<h1> to <h6>	Specifies header 1 to header 6
<header>	New Tag:Specifies a group of introductory or navigational aids.
<hgroup>	New Tag:Specifies the header of a section.
<hr>	Specifies a horizontal rule



# HTML5 Flow Elements

Tag	Description
<i>	Specifies italic text
<iframe>	Specifies an inline sub window (frame)
<img>	Specifies an image
<input>	Specifies an input field
<ins>	Specifies inserted text
<keygen>	New Tag:Specifies control for key pair generation.
<kbd>	Specifies keyboard text
<label>	Specifies a label for a form control
<map>	Specifies an image map
<mark>	New Tag:Specifies a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.

# HTML5 Flow Elements

Tag	Description
<menu>	Deprecated: Specifies a menu list
<math>	Defines math expressions
<meter>	New Tag:Specifies a measurement, such as disk usage.
<nav>	New Tag:Specifies a section of the document intended for navigation.
<noscript>	Specifies a noscript section
<object>	Specifies an embedded object
<ol>	Specifies an ordered list
<output>	New Tag:Specifies some type of output, such as from a calculation done through scripting.
<p>	Specifies a paragraph
<pre>	Specifies preformatted text

# HTML5 Flow Elements

Tag	Description
<progress>	New Tag:Specifies a completion of a task, such as downloading or when performing a series of expensive operations.
<q>	Specifies a short quotation
<ruby>	New Tag:Together with <rt> and <rp> allow for marking up ruby annotations.
<script>	Specifies a script
<samp>	Specifies sample computer code
<small>	Specifies small text
<section>	New Tag:Represents a generic document or application section.
<select>	Specifies a selectable list
<strong>	Specifies strong text
<span>	Specifies a section in a document

# HTML5 Flow Elements

Tag	Description
<style>	Specifies a style definition
<sub>	Specifies subscripted text
<sup>	Specifies superscripted text
<svg>	Defines graphics in XHTML documents
<table>	Specifies a table
<textarea>	Specifies a text area
<time>	New Tag:Specifies a date and/or time.
<var>	Specifies a variable
<ul>	Specifies an unordered list
<video>	New Tag:Specifies a video file.
<wbr>	New Tag:Specifies a line break opportunity.
<text>	Represents text in an HTML document

# HTML5 Phrasing Elements

- Phrasing elements are used to represent the text of the HTML document.
- These elements are also used to mark up the HTML document text within the paragraph of the document.

# HTML5 Phrasing Elements

Tag	Description
<a>	Specifies an anchor
<abbr>	Specifies an abbreviation
<audio>	New Tag:Specifies an audio file.
<area>	Specifies an area inside an image map
<b>	Represents the bold text.
<bdo>	Specifies the direction of text display
 	Inserts a single line break
<button>	Specifies a push button
<canvas>	New Tag:This is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
<cite>	Specifies a citation
<code>	Specifies computer code text

# HTML5 Phrasing Elements

Tag	Description
<command>	New Tag:Specifies a command the user can invoke.
<datalist>	New Tag:Together with the a new list attribute for input can be used to make comboboxes
<dfn>	Specifies a definition term
<del>	Specifies deleted text
<embed>	New Tag:Defines external interactive content or plugin.
<em>	Specifies emphasized text
<i>	Specifies italic text
<iframe>	Specifies an inline sub window (frame)
<img>	Specifies an image
<input>	Specifies an input field
<ins>	Specifies inserted text

# HTML5 Phrasing Elements

Tag	Description
<keygen>	New Tag:Specifies control for key pair generation.
<kbd>	Specifies keyboard text
<label>	Specifies a label for a form control
<map>	Specifies an image map
<mark>	New Tag:Specifies a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.
<math>	Defines math expressions
<meter>	New Tag:Specifies a measurement, such as disk usage.
<noscript>	Specifies a noscript section
<object>	Specifies an embedded object
<output>	New Tag:Specifies some type of output, such as from a calculation done through scripting.



# HTML5 Phrasing Elements

Tag	Description
<progress>	New Tag:Specifies a completion of a task, such as downloading or when performing a series of expensive operations.
<q>	Specifies a short quotation
<ruby>	New Tag:Together with <rt> and <rp> allow for marking up ruby annotations.
<script>	Specifies a script
<samp>	Specifies sample computer code
<small>	Specifies small text
<section>	New Tag:Represents a generic document or application section.
<select>	Specifies a selectable list
<strong>	Specifies strong text
<span>	Specifies a section in a document

# HTML5 Phrasing Elements

Tag	Description
<sub>	Specifies subscripted text
<sup>	Specifies superscripted text
<svg>	Defines graphics in XHTML documents
<textarea>	Specifies a text area
<time>	New Tag:Specifies a date and/or time.
<var>	Specifies a variable
<video>	New Tag:Specifies a video file.
<wbr>	New Tag:Specifies a line break opportunity.
<text>	Represents text in an HTML document

# HTML5 Embedded Elements

- Embedded elements are used to import content from other resources into the HTML document.
- For Ex: The EMBED element is used as an integration point to plug in the content from other sources into the HTML document.

# HTML5 Embedded Elements

Tag	Description
<audio>	New Tag:Specifies an audio file.
<canvas>	New Tag:This is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
<embed>	New Tag:Defines external interactive content or plugin.
<iframe>	Specifies an inline sub window (frame)
<img>	Specifies an image
<math>	Defines math expressions
<object>	Specifies an embedded object
<svg>	Defines graphics in XHTML documents
<video>	New Tag:Specifies a video file.

# HTML5 Interactive Elements

- Interactive elements are specifically intended for user interaction.

# HTML5 Interactive Elements

Tag	Description
<a>	Specifies an anchor
<audio>	New Tag:Specifies an audio file.
<button>	Specifies a push button
<details>	New Tag:Specifies additional information or controls which the user can obtain on demand.
<embed>	New Tag:Defines external interactive content or plugin.
<iframe>	Specifies an inline sub window (frame)
<img>	Specifies an image
<input>	Specifies an input field
<keygen>	Generate key information in a form
<label>	Specifies a label for a form control
<menu>	Deprecated: Specifies a menu list

# HTML5 Interactive Elements

Tag	Description
<object>	Specifies an embedded object
<select>	Specifies a selectable list
<textarea>	Specifies a text area
<video>	New Tag:Specifies a video file.

# HTML5 Interactive Elements

Tag	Description



# HTML5 Interactive Elements

Tag	Description

# HTML5 Elements

Tag	Description
<frame>	Deprecated: Specifies a sub window (a frame)
<frameset>	Deprecated: Specifies a set of frames
<head>	Specifies information about the document
<html>	Specifies an html document
<isindex>	Deprecated: Specifies a single-line input field

# HTML5 Elements

Tag	Description
<keygen>	Generate key information in a form
<layer>	Specifies a layer
<legend>	Specifies a title in a fieldset
<li>	Specifies a list item

# HTML5 Elements

Tag	Description
<link>	Specifies a resource reference
<marquee>	Create a scrolling-text marquee
<multicol>	Specifies a multicolumn text flow
<nobr>	No breaks allowed in the enclosed text
<noembed>	Specifies content to be presented by browsers that do not support the <embed>tag

# HTML5 Elements

Tag	Description
<noframes>	Deprecated:Specifies a noframe section
<optgroup>	Specifies an option group
<option>	Specifies an option in a drop-down list
<param>	Specifies a parameter for an object
<dfn>	Specifies a definition term

# HTML5 Elements

`<samp>`

Specifies sample computer code

`<plaintext>`

Deprecated: Render the remainder of the document as preformatted plain text

# HTML5 Elements

Tag	Description
<code>&lt;s&gt;</code>	Deprecated: Specifies strikethrough text
<code>&lt;strike&gt;</code>	Deprecated: Specifies strikethrough text
<code>&lt;tbody&gt;</code>	Specifies a table body

# HTML5 Elements

Tag	Description
<td>	Specifies a table cell
<title>	Specifies the document title
<tr>	Specifies a table row
<u>	Deprecated: Specifies underlined text



# HTML5 Elements

Tag	Description
<video>	New Tag:Specifies a video file.
<wbr>	New Tag:Specifies a line break opportunity.
<wbr>	Indicate a potential word break point within a <nobr> section
<xmp>	Deprecated: Specifies preformatted text

# HTML5 Elements

# HTML5 Elements

- A complete list of standard tags available in HTML5 is given below. All the tags are ordered alphabetically along with an indication if they have been introduced newly or they have been deprecated in HTML5.

# HTML5 Elements

Tag	Description
<!--...-->	Specifies a comment
<!DOCTYPE>	Specifies the document type
<a>	Specifies an anchor
<abbr>	Specifies an abbreviation
<acronym>	Deprecated:Specifies an acronym
<address>	Specifies an address element
<applet>	Deprecated: Specifies an applet
<area>	Specifies an area inside an image map
<article>	New Tag: Specifies an independent piece of content of a document, such as a blog entry or newspaper article
<aside>	New Tag:Specifies a piece of content that is only slightly related to the rest of the page.
<audio>	New Tag:Specifies an audio file.

# HTML5 Elements

Tag	Description
<base>	Specifies a base URL for all the links in a page
<basefont>	Deprecated: Specifies a base font
<bdo>	Specifies the direction of text display
<bgsound>	Specifies the background music
<blink>	Specifies a text which blinks
<blockquote>	Specifies a long quotation
<body>	Specifies the body element
 	Inserts a single line break
<button>	Specifies a push button
<canvas>	New Tag: This is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
<caption>	Specifies a table caption

# HTML5 Elements

Tag	Description
<center>	Deprecated: Specifies centered text
<col>	Specifies attributes for table columns
<colgroup>	Specifies groups of table columns
<command>	New Tag:Specifies a command the user can invoke.
<comment>	Puts a comment in the document
<datalist>	New Tag:Together with the a new list attribute for input can be used to make comboboxes
<dd>	Specifies a definition description
<del>	Specifies deleted text
<details>	New Tag:Specifies additional information or controls which the user can obtain on demand.
<dir>	Deprecated: Specifies a directory list
<div>	Specifies a section in a document

# HTML5 Elements

Tag	Description
<dl>	Specifies a definition list
<dt>	Specifies a definition term
<embed>	New Tag:Defines external interactive content or plugin.
<fieldset>	Specifies a fieldset
<figure>	New Tag:Specifies a piece of self-contained flow content, typically referenced as a single unit from the main flow of the document.
<b>	Specifies bold text
<big>	Deprecated:Specifies big text
<i>	Specifies italic text
<small>	Specifies small text
<tt>	Deprecated:Specifies teletype text
<font>	Deprecated: Specifies text font, size, and color

# HTML5 Elements

Tag	Description
<footer>	New Tag:Specifies a footer for a section and can contain information about the author, copyright information, et cetera.
<form>	Specifies a form
<frame>	Deprecated:Specifies a sub window (a frame)
<frameset>	Deprecated:Specifies a set of frames
<head>	Specifies information about the document
<header>	New Tag:Specifies a group of introductory or navigational aids.
<hgroup>	New Tag:Specifies the header of a section.
<h1> to <h6>	Specifies header 1 to header 6
<hr>	Specifies a horizontal rule
<html>	Specifies an html document
<isindex>	Deprecated: Specifies a single-line input field



# HTML5 Elements

Tag	Description
<iframe>	Specifies an inline sub window (frame)
<ilayer>	Specifies an inline layer
<img>	Specifies an image
<input>	Specifies an input field
<ins>	Specifies inserted text
<keygen>	New Tag:Specifies control for key pair generation.
<keygen>	Generate key information in a form
<label>	Specifies a label for a form control
<layer>	Specifies a layer
<legend>	Specifies a title in a fieldset
<li>	Specifies a list item

# HTML5 Elements

Tag	Description
<link>	Specifies a resource reference
<map>	Specifies an image map
<mark>	New Tag:Specifies a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.
<marquee>	Create a scrolling-text marquee
<menu>	Deprecated: Specifies a menu list
<meta>	Specifies meta information
<meter>	New Tag:Specifies a measurement, such as disk usage.
<multicol>	Specifies a multicolumn text flow
<nav>	New Tag:Specifies a section of the document intended for navigation.
<nobr>	No breaks allowed in the enclosed text
<noembed>	Specifies content to be presented by browsers that do not support the <embed>tag

# HTML5 Elements

Tag	Description
<noframes>	Deprecated:Specifies a noframe section
<noscript>	Specifies a noscript section
<object>	Specifies an embedded object
<ol>	Specifies an ordered list
<optgroup>	Specifies an option group
<option>	Specifies an option in a drop-down list
<output>	New Tag:Specifies some type of output, such as from a calculation done through scripting.
<p>	Specifies a paragraph
<param>	Specifies a parameter for an object
<cite>	Specifies a citation
<code>	Specifies computer code text
<dfn>	Specifies a definition term

# HTML5 Elements

Tag	Description
<code>&lt;em&gt;</code>	Specifies emphasized text
<code>&lt;kbd&gt;</code>	Specifies keyboard text
<code>&lt;samp&gt;</code>	Specifies sample computer code
<code>&lt;strong&gt;</code>	Specifies strong text
<code>&lt;var&gt;</code>	Specifies a variable
<code>&lt;plaintext&gt;</code>	Deprecated: Render the remainder of the document as preformatted plain text
<code>&lt;pre&gt;</code>	Specifies preformatted text
<code>&lt;progress&gt;</code>	New Tag:Specifies a completion of a task, such as downloading or when performing a series of expensive operations.
<code>&lt;q&gt;</code>	Specifies a short quotation
<code>&lt;ruby&gt;</code>	New Tag:Together with <code>&lt;rt&gt;</code> and <code>&lt;rp&gt;</code> allow for marking up ruby annotations.
<code>&lt;script&gt;</code>	Specifies a script

# HTML5 Elements

Tag	Description
<section>	New Tag:Represents a generic document or application section.
<select>	Specifies a selectable list
<spacer>	Specifies a white space
<span>	Specifies a section in a document
<s>	Deprecated: Specifies strikethrough text
<strike>	Deprecated: Specifies strikethrough text
<style>	Specifies a style definition
<sub>	Specifies subscripted text
<sup>	Specifies superscripted text
<table>	Specifies a table
<tbody>	Specifies a table body

# HTML5 Elements

Tag	Description
<td>	Specifies a table cell
<textarea>	Specifies a text area
<tfoot>	Specifies a table footer
<th>	Specifies a table header
<thead>	Specifies a table header
<time>	New Tag:Specifies a date and/or time.
<title>	Specifies the document title
<tr>	Specifies a table row
<u>	Deprecated: Specifies underlined text
<ul>	Specifies an unordered list

# HTML5 Elements

Tag	Description
<video>	New Tag:Specifies a video file.
<wbr>	New Tag:Specifies a line break opportunity.
<wbr>	Indicate a potential word break point within a <nobr> section
<xmp>	Deprecated: Specifies preformatted text

# HTML5 Attributes

- Standard Attributes
- The attributes listed below are supported by almost all the HTML 5 tags.

Attribute	Options	Function
accesskey	User Defined	Specifies a keyboard shortcut to access an element.
align	right, left, center	Horizontally aligns tags
background	URL	Places an background image behind an element
bgcolor	numeric, hexadecimal, RGB values	Places a background color behind an element



# HTML5 Attributes

Attribute	Options	Function
class	User Defined	Classifies an element for use with Cascading Style Sheets.
contenteditable	true, false	Specifies if the user can edit the element's content or not.
contextmenu	Menu id	Specifies the context menu for an element.
data-XXXX	User Defined	Custom attributes. Authors of a HTML document can define their own attributes. Must start with "data-".
draggable	true,false, auto	Specifies whether or not a user is allowed to drag an element.
height	Numeric Value	Specifies the height of tables, images, or table cells.
hidden	hidden	Specifies whether element should be visible or not.
id	User Defined	Names an element for use with Cascading Style Sheets.

# HTML5 Attributes

Attribute	Options	Function
tem	List of elements	Used to group elements.
itemprop	List of items	Used to group items.
spellcheck	true, false	Specifies if the element must have it's spelling or grammar checked.
style	CSS Style sheet	Specifies an inline style for an element.
subject	User define id	Specifies the element's corresponding item.
tabindex	Tab number	Specifies the tab order of an element.
title	User Defined	"Pop-up" title for your elements.
valign	top, middle, bottom	Vertically aligns tags within an HTML element.
width	Numeric Value	Specifies the width of tables, images, or table cells.

# Custom Attributes

- A new feature being introduced in HTML 5 is the addition of custom data attributes.
- A custom data attribute starts with **data-** and would be named based on your requirement. Here is a simple example —

```
<div class = "example" data-subject = "physics" data-level = "complex"> ... </div>
```

- The above code will be perfectly valid HTML5 with two custom attributes called *datasubject* and *data-level*.
- You would be able to get the values of these attributes using JavaScript APIs or CSS in similar way as you get for standard attributes.

# HTML5 Elements

# HTML5 Elements

# HTML5 Elements

# HTML5 Elements

# HTML5 Elements



# Unit-4

## Working with Text, Links and Tables in HTML5

# Text Formatting with Physical Style Elements

- HTML provides a number of [tags](#) such as HEAD and BODY to define the [structure](#) and appearance of a document. By default, the HTML document is stored in the plain text format, it means that it does not contain any type of formatting. To format the text in an HTML document, you need to use the various kinds of [HTML tags](#).
- [Formatting](#) the text not only enhanced the visual appearance of the content but also improves the readability and comprehensibility.
- The content of a formatted document is presented in an organized manner, which allows readers to easily differentiate between various types of information. For instance, any important piece of information in an HTML document can be enclosed between B or I tags to make it appear in bold or italics respectively.

# Text Formatting with Physical Style Elements

- Sometimes, you may need to highlight, mark or emphasize certain important or technical terms to differentiate them from the normal text content. So HTML provides you with the MARK and the EM tag to highlight and emphasize the text content.
- **HTML Text Formatting Tags**
- HTML provide a set of tags to change the appearance of the text by applying various formatting features, such as bold, italics, subscript, and superscript. These HTML tags are used with starting and ending tags.

# Text Formatting with Physical Style Elements

Tag	Description
B	Displays the text in bold
I	Displays the text in italics
SMALL	Displays the text in small font size
SUB	Displays the text as subscript
SUP	Displays the text as superscript

# Text Formatting with Physical Style Elements

- **HTML Physical Style Tags Attributes**
- The table given below list and describes all attributes of physical style tags available in HTML used to format the text in an HTML document.

Attribute	Description
class	Indicates a class name for a tag
dir	Indicates the directionality of the text, such as left to right or right to left
id	Indicates an unique id for a tag
lang	Indicates the language code for the content in a tag
style	Indicates an inline style for a tag
title	Specifies a title for a tag

# HTML Text Formatting with Logical Style Tags

- In HTML, the logical style tags specify that the enclosed text has a specific meaning, context, or usage. For example, the ABBR tag conveys to the Web browser that the text enclosed inside this tag is an abbreviation. The browser change the appearance of the text depending upon the meaning of the tags.
- The advantage of using the logical style tags rather than the physical style tags is that the meaning related to the tag is more precisely conveyed to the users

# HTML Text Formatting with Logical Style Tags

Tag	Description
ABBR	Displays the abbreviation on the Web page
CODE	Refers to the program code
SAMP	Displays the sample program output on the Web page
KBD	Refers to the keyboard keys
EM	Emphasized text
STRONG	Emphasized text by increasing boldness
DFN	Displays new terms on the Web page
Q	Displays the short quotations on the Web page

# HTML Text Formatting with Logical Style Tags

Tag	Description
BLOCKQUOTE	Displays the long quotations on the Web page
INS	Displays the inserted text
DEL	Displays the deleted text
VAR	Represent a variable
BDO	Changes the direction of text



# HTML Text Formatting with Logical Style Tags

- **HTML Logical Style Tags Attributes**
- The following table list and describes all attributes used with the logical style tags to format the text in an HTML document:

Attribute	Description
class	Indicates a class name for a tag
dir	Indicates the directionality of text, such as left-to-right or right-to-left
id	Indicates a unique id for a tag
lang	Indicates a language code for the content in a tag
style	Indicates an inline style for the tag
title	Indicates the title for a tag

# HTML Text Formatting with Logical Style Tags

- In addition to the above attributes, there are some other attributes, such as cite and datetime, which can only be used with the INS and DEL tags.
- The INS and DEL tags are used to display the insertion of content by underlining and deletion of content by striking through it in an HTML document.
- The cite attribute is used to indicate the reason for insertion and deletion.
- The value of the cite attribute is a Universal Resource Locator (URL), which points to some other document that describes the inserted or deleted text

# HTML Text Formatting with Logical Style Tags

- The datetime attributed is used to indicate the time of insertion or deletion. This attribute takes a single value, which is an encoded date and time stamp.
- The format of the datetime attribute value must be YYYY-MM-DDThh:mm:ssTZD
- **Components of the datetime Value Attribute**
- The following table shows the components of the datetime value attribute :

# HTML Text Formatting with Logical Style Tags

Components	Description
YYYY	Indicates the year, such as 1996 or 2010
MM	Indicates the month, such as 05 for May
DD	Indicates the date, such as 01 to 31
T	Specifies that the next section displays the time
hh	Indicates the hour in a 24-hour format
mm	Indicates the minutes in an hour
ss	Indicates the seconds, 00 to 59
TZD	Indicates the Time-Zone Designator (TZD)

# HTML Text Formatting with Logical Style Tags

- **HTML MARK tag**
- The HTML MARK tag is used to mark or highlight the text in an HTML document.
  - **Note** - By default, the MARK tag displays the text with yellow background color
- **HTML MARK Tag Attributes**
- The table given below list and describes attributes that can be used with the MARK tag to format the text in an HTML document:

Attribute	Description
class	Indicates a class name for the MARK tag
id	Indicates a unique id for the MARK tag
style	Indicates an inline style for the MARK tag
title	Indicates a title for the MARK tag

# HTML Text Formatting with Logical Style Tags

- **HTML STRONG tag**
- The HTML STRONG tag is used to emphasize the important text with bold. this tag increases the font weight of the text and makes the text appears as bold. So that to help us to recognize the important text in the HTML document
- **HTML CODE tag**
- The HTML CODE tag is used to represent the computer code in an HTML document. So if you want to display some article on the Web page, which contains some text and code of a program, then you can write the program code within the CODE tag so that the reader of the article can differentiate between the normal text and the code.

# HTML Text Formatting with Logical Style Tags

- **HTML SMALL tag**
- The HTML SMALL tag is used to display the text as a side comment or in small print. The small prints include the disclaimers, legal restrictions, and copyrights.

# HTML Character Entities

- Sometimes, we need to include some special character, such > and < in the content of an HTML element. These characters are called as entities.
- So, HTML enables us to include such symbols in the content of an element by using the various character entities.
- World Wide Web Consortium (W3C) calls the character entities as character encodings.
- There are some character, such as Greek and Chinese, which do not appear on the keyboard of the user. Therefore, you cannot write them directly from the keyboard.



# HTML Character Entities

- However, you can overcome form this problem by using the character entities in an HTML document. The character entity is composed of the following parts in the sequential manner :
  - Ampersand symbol (&)
  - Name of the entity
  - A terminating semicolon

# HTML Organizing Text

- In a Web page, the content is organized into the different formats, such as layers, [paragraphs](#), lines, [tables](#), and divisions that we have already learned.
- Organizing text refers to the proper placement of all the [HTML tags](#) and their content in a Web page.
- By default, a Web browser wraps [text](#) in a Web page and displays the enclosed text in a single block by avoiding the line and paragraph breaks.
- Now, if the content of a page is not separated by any line or paragraph breaks, it becomes very tedious for the readers to understand the given information.

# HTML Organizing Text

- By default, a Web browser wraps [text](#) in a Web page and displays the enclosed text in a single block by avoiding the line and paragraph breaks. Now, if the content of a page is not separated by any line or paragraph breaks, it becomes very tedious for the readers to understand the given information.
- If text in the Web page is not arranged then the readers will find difficulty in reading the desired information.

# HTML Organizing Text

- To overcome this problem, you can [arrange the text](#) in different ways, such as paragraphs, lines, and tables.
- HTML provides a number of tags to arrange text into paragraphs and tables.
- For instance, you can display the text on the Web page as paragraphs by using the P tag, or display a [horizontal line](#) in a Web page representing a break in the text.
- HTML also allows you to change to format of a specific text using the SPAN tag.

# HTML Organizing Text

- HTML allows you to arrange the text into different formats and styles, such as [paragraphs](#), layers, and [tables](#).
- By default, the Web browser displays all the text of an HTML page as a single paragraph.
- You can format this text in different ways by using the various [tags](#). For example, the P tag is used to arrange the text in multiple paragraphs and the BR tag allows you to continue text from the next line.

# HTML Organizing Text

- There are the following tasks are involved in arranging the text of a Web page :
  - Allowing word breaks
  - Defining the preformatted text
  - Defining the DIV tag
  - Defining the Span tag
  - Formatting the text in tables
  - Defining the Ruby (captioned) text

# HTML Organizing Text

- **Allowing Word Breaks**
- HTML provides the WBR tag to insert word breaks between the words or paragraphs in an HTML document.
- However, if a break is required between the words, then you can use the WBR tag.
- The WBR tag does not force the Web browser to break the line; whereas, the BR tag break the line irrespective of the space available in the Web page.
- **Defining the Preformatted Text**
- In HTML, the preformatted tag, PRE, instructs the browser that the enclosed text is a formatted text and should not be formatted again.

# HTML Organizing Text

- **HTML PRE Tag Attributes**
- The following table given here list and describes attributes available in the PRE tag:

Attribute	Description
class	Indicates the class name for the PRE tag
dir	Indicates the direction of the enclosed text
id	Indicates an unique id for tag
lang	Indicates the base language used for the tag
style	Indicates an inline style for the tag
title	Indicates an extra information about the tag



# HTML Organizing Text

- **HTML DIV tag**
- In HTML, the division (DIV) tag is used to divide the Web page into different divisions or sections. The DIV tag basically works as a container for the other HTML tags.
- Using the DIV tag, you can group the HTML tags in sections and apply Cascading Style Sheet ([CSS](#)) on them.
- The DIV tag is a block level tag and cannot be used inside the P tag. The P tag is only used for creating the paragraphs while the DIV tag is used to divide the Web page into different sections.

# HTML Organizing Text

Attribute	Description
class	Defines the class name for a tag
dir	Defines the text direction for the content in a tag
id	Defines an unique id for a tag
lang	Defines the language code for the content in a tag
style	Defines an inline style for a tag
title	Defines an extra information about a tag

# HTML Organizing Text

- **HTML SPAN tag**
- The SPAN tag is used to change to style of the text enclosed within the tag using the style attribute.
- The SPAN tag provides the additional formatting capabilities to the HTML tags and is used with the style sheets to set presentational attributes that define the style of the content.

# HTML Organizing Text

Attribute	Description
class	Defines the class name for a tag
dir	Defines the direction for the content in a tag
id	Defines an unique id for a tag
lang	Defines the language code for the content in a tag
style	Defines an inline style for a tag
title	Defines the title for a tag

# HTML Organizing Text

- **Formatting Text in Tables**
- HTML provides the TABLE tag for arranging the text in a tabular format, i.e. in rows and columns.
- The TABLE tag is used along with the TH, TR, and TD tags. The TH tag is used to define the header of a table column.
- The TR tag is used to specify the content of the table rows.
- The TD tag is used to specify the content of a table cell.
- You can set the width and the height of the table columns and rows using the style attribute in the TABLE tag.
- Using the style attribute, you can also define the borders for the table, font style for the table content, and the font size for the table.
- You can also specify the table caption, which indicates the content of the table, by using the CAPTION tag

# HTML Organizing Text

- **HTML Ruby (Captioned) Text**
- You can define the caption for your text using the RUBY tag in you Web page.
- This tag is used to specify ruby annotations that provide reading or pronunciation guide of a particular language, such as Chinese and Japanese.
- A ruby is a part of an annotation pair in which a piece of text (known as ruby text) is associated with another set of text (known as ruby base).

# HTML Organizing Text

- Ruby annotations are frequently used in many Asian countries, such as Japan and China to provide the pronunciation guide in various kinds of publications, including school books and magazines.
- The RUBY tag contains the following tags :
  - **RP** - Contains parenthesis
  - **RT** - Acts as the container for the ruby text
- These tags are used to provide a structural association between the ruby text and ruby base.
- Structural association refers to various ways in which the ruby text and ruby base can be connected with each other.
- In the RUBY tag, the text appearing before the RT tag is called the base text. Whereas the text appearing after the RT tag is called the notation.

# HTML Organizing Text

Ruby Tag Attribute	Description
class	Indicates the class name for the RUBY tag
cols	Indicates the number of columns used in the table
dir	Indicates the direction of the enclosed text
id	Indicates an unique id for a tag
lang	Indicates the base language used for a tag
language	Indicates the scripting language used for a tag
style	Indicates an inline style for a tag
title	Indicates the title of a tag
name	Indicates the name of a tag



# List Elements

- HTML allows you to display the information in the form of [lists](#).
- These information in these lists may be presented in an ordered and sequential manner or in an unordered and random manner.
- Here are the list of [tags](#) used in displaying the list in an HTML document:

# List Elements

Tag	Description
OL	used to display the list of information in an ordered of sequence
UL	used to display the information in an unordered list
LI	defines a list item
DL	allows you to include the description of the items in the UL or OL lists
DT	defines the term in a description list
DD	defines the description in a description list

# List Elements

- HTML list helps us to display the information in the form of lists. There are the following three forms a list can be:
  - Ordered Lists
  - Unordered Lists
  - Description Lists
- **HTML Ordered List**

```
<ol>
```

```
<li>codes</li>
```

```
<li>cracker</li>
```

```
<li>html</li>
```

```
<li>list</li>
```

```
<li>tutorial</li>
```

```
<li>examples</li>
```

```
</ol>
```

# List Elements

- **HTML Ordered List Type Attributes**
- A type attribute can be added to an ordered list, to define the type of the marker. The table given here list and describes the list of marker types used in ordered list.

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

# List Elements

```
<ol type="1">  
  <li>html</li>  
  <li>list</li>  
</ol>
```

```
<ol type="A">  
  <li>html</li>  
  <li>list</li>  
</ol>
```

```
<ol type="a">  
  <li>html</li>  
  <li>list</li>  
</ol>
```

# List Elements

- **HTML Unordered List**
- HTML unordered list starts with the <UL> tag and each items starts with the <LI> tag. Here is an example of unordered list.

```
<ul>
```

```
  <li>codes</li>
```

```
  <li>cracker</li>
```

```
  <li>html</li>
```

```
  <li>list</li>
```

```
</ul>
```

# List Elements

- **HTML Unordered Lists Style Type Attributes**
- A style attribute can be added to an unordered list, to define the style of the marker. The table given below list and describes attributes of the HTML unordered list style type.
- **EX:** `<ul style="list-style-type:disc">`

Style	Description
list-style-type:circle	The list items will be marked with circles
list-style-type:disc	The list items will be marked with bullets (default)
list-style-type:square	The list items will be marked with squares
list-style-type:none	The list items will not be marked

# List Elements

- **HTML Description Lists**
- A description list, is a list of terms, with a description of each term.
- The `<DL>` tag defines a description list, the `<DT>` tag defines the term (name), and the `<DD>` tag defines the data (description).

`<dl>`

`<dt>html</dt>`

`<dd>hyper text markup language</dd>`

`<dt>list</dt>`

`<dd>organized, unorganized, description list</dd>`

`</dl>`



# List Elements

- **HTML Nested Lists**
  - You are free to nest one list to another, also called as nested list.
- **HTML Create Horizontal Lists**
  - You can also create horizontal list in HTML using [CSS](#).

# HTML Working with Links and URLs

- In HTML, hypertext refers to a point of references for detailed text on the same or different Web pages.
- Generally, a hyperlink is an underlined word or phrase or can also be an image or icon that contains a specific address of a Web page.
- The address is provided in the [hyperlink](#) in the form of the Uniform Resource Locator ([URL](#)), which is a unique address allotted to every Web page. With the help of URL, you can navigate to the specific Web page.
- Hyperlinks also allows you to access the main system of you computer.

# HTML Working with Links and URLs

- For instance, most of the websites provide an e-mail address to users to contact website administrator or to send any query, suggestions, and comments regarding the respective website.
- This e-mail address is provided on a Web page in the form of hyperlink.
- When a user selects that hyperlink, a new mail message window automatically appears on the screen, in which the user can write the query or suggestion in that window and sent it to that e-mail address.
- You can also access various news groups or blogs over the Internet with the help of hyperlinks by providing the URL of a news group and blog.

# HTML Links

- A hyperlink interconnects the current Web page with the other Web pages available on the Internet. In HTML, you can create a hyperlink by using the anchor element (A).
- The hyperlink redirects the user to the another HTML page, image, or file.
- The A element uses the href attribute to specify the target resource or document that you want to open when the user clicks the hyperlink.
- The term **href** stands for Hypertext Reference. The href attribute sets the URL of the target resource.
- In HTML, links are defined with the <a> tag. For example,  
<A href=<http://www.google.com>>Google</A>

# HTML Links

- **HTML Link - The target Attribute**
- The A element uses the target attribute to specify a window where you want to open a document when a hyperlink is clicked.
- For instance, you can open a document in the same window or in another window just by using the target attribute.
- **HTML Link target Attribute Values**

Value	Description
_blank	Opens the linked document in a new unnamed window
_self	Opens the linked document in the current window (this is default value)
_parent	Opens the linked document in the parent window
_top	Opens the linked document in the topmost window
framename	Opens the linked document in a named frame

# HTML Links

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
  <a href="http://abc.com" target="_blank">Click Here</a>
```

```
</body>
```

```
</html>
```

# HTML Links

- **HTML Link - The id Attribute**
- The A element uses the id attribute to create a fragment identifier within a document. A fragment identifier specifies a particular location within a document. You can navigate different locations within a document just by using the id attribute.
- The id attribute takes a character string as a value. This value must be unique in the same document. However, it can be reused in different documents.
- The id attribute can be used to create bookmarks inside HTML documents. Bookmarks are not displayed in any special way. They are invisible to the reader.

# HTML Links

- **Example**
- Here is an example demonstrates id attribute in HTML link. Add an id attribute to any `<a>` element.

**`<a id="topS">Go To Top</a>`**

- Then create a link to the `<a>` element (Go To Top):

**`<a href="#topS">Go To Top</a>`**

- Click on the link below to watch the effect. After clicking on the below link, you will go to the top of this web page.



# HTML Links

- **HTML Local Links**
- As you can see, the above example used an absolute URL (A full web address)
- A local link (link to the same web site) is specified with a relative URL (without http://www....)
- Here is an example demonstrates local links in HTML:

**`<a href="html-links.htm">This is a Link</a>`**

# HTML Links

- **HTML Links Colors**
- When you place your mouse cursor over a link, then you will watch the following two things will happen normally:
  - the arrow of the mouse will turn into a little hand
  - the color of the link element will change
- By default, the link will appear as in all browsers:

Link	Appearance
unvisited link	underline and blue
visited link	underline and purple
active link	underline and red

# Working with Links and URLs

- **Change Link Style in HTML**
- Here is an example shows how to change the defaults of the link, using [CSS](#)
- **See Example: HTMLLink.html**
- **HTML Image Link**
- You can also use any image as a link in HTML. Here is an example demonstrates how to provide image link in HTML:

```
<a href="http://codescracker.com" target="_blank">  
      
</a>
```

# Working with Links and URLs

- **HTML LINK Tag**
- HTML provide the LINK tag to link a Web page with an external resource or document, such as a Cascading Style Sheet ([CSS](#)) file or an Extensible Markup Language (XML) file.
- The **rel** attribute of the LINK tag is used to create relation with the other resources.
- This attributes takes different values that specify which resource is to be linked with the Web page. For example, the stylesheet value specifies that the page is linked with a stylesheet file.
- Here is an example of the LINK tag:  
**<link rel="stylesheet" href="mystylesheet.css">**
- In addition to the stylesheet, there are some other values of the rel attribute which are described in the table given here:

# Links Relations

Name	Description
alternate	Specifies the link on the basis of the other attributes of the LINK tag
archives	Creates a relation to the collection of records, documents, or other materials of historical events
author	Provides a hyperlink to the page that contains the information about the author of the current document
bookmark	Indicates that the current document is a permalink or bookmark
external	Refers to the content or a document that is not the part of the current website
feed	Provides the hyperlink to a syndication feed, which is data format that provides the user updated content of a document
first, next, prev, last	Defines the relation between the pages that are the part of a series
help	Represents that the content of the referenced document provides help for the current tag

# HTML Working with Links and URLs

Name	Description
icon	Specifies an image that indicates the icon of the current document
index	Specifies the document that contains the table of content or index of the current document
license	Indicates that the current document provides copyright license terms
nofollow	Does not allow author or publisher to endorse the referenced document
noreferrer	Secures the referrer details and other information
pingback	Notifies a blog automatically when the other blogs are linked to it
prefetch	Specifies that the highly required resource is to be stored in the cache memory for later use
search	Provides a hyperlink to the resource to specifically search the current document and its related resource
sidebar	Signifies that the referenced document is displayed in the sidebar of the browser. The left-side portion of a Web page is known as sidebar

# HTML Working with Links and URLs

Name	Description
start	Represents the first document from a list of documents
stylesheet	Refers to the stylesheet for the current document
tag	Provides a tag to the current document that represents the referenced document
up	Provides a hyperlink to a document that is used to go back to the first page or the home page of the document

# Unit-5

Working with Forms, Images and Media in  
HTML5



# HTML5 Forms

- Here you will learn all about HTML form handling with examples. HTML form provides user interaction with websites.
- Sometimes, you need to develop a website that require user interaction.
- For example, you need to develop a railway website that allows users to check the arrival or departure status of trains on the Internet.
- This website uses a registration or login form to collect the user information and submits the information to the server for further processing.
- These types of forms provides the medium of interaction between a website and its users on the Internet.

# HTML5 Forms

- A form is an area of a Web page that allows the users to provide their information in a variety of ways, such as by entering the text field or by selecting one or more available options from the provided list.
- HTML enables you to add a form in a Web page by using the FORM tag.
- After adding the form on the Web page, you can add various controls, such as buttons and text fields, on the form by using a variety of tags. Some examples of these elements are [INPUT](#), [BUTTON](#), [TEXTAREA](#), and [DATALIST](#).

# HTML5 Forms

- The INPUT tag allows you to enter different types of values, such as date, time, and e-mail address.
- The BUTTON tag enables you to add buttons, such as submit and cancel buttons on the form to submit or cancel form's detail.
- The TEXTAREA tag enables you to enter text in the provided area.
- The DATALIST tag allows you to enter the text in a text field by providing a set of predefined values.

# HTML5 Forms

- Form Tags
- Input Tag Types
- Button Tags
- Multiple-Choice Tags
- Select Tag
- Option Tag
- Optgroup Tag
- Textarea and Label Tags
- Fieldset and Legend Tags
- Datalist Tag
- Keygen Tag
- Output Tag
- Progress Tag
- Meter Tag
- Form Submitting
- enctype Attribute
- action Attribute
- Method Attribute  
(Get Method and Post Method)

# HTML5 Forms

- **Form Tag**
- A [form](#) is an area on the Web page that consists of plain text, [HTML tags](#), and controls.
- Plain text and HTML tags are used to structure the form, whereas, controls, which are also known as form fields, are used to make the form interactive by allowing a user to enter the information.
- You can create a form in a web page using the FORM tag.

# HTML5 Forms

Form Tag Attribute	Description
action	Refers to the Uniform Resource Locator (URL) of the program in the server that processes the form
autocomplete	Enables the autocomplete feature in a form
accept-character	Refers to the character set in the form that can be accepted by the server
enctype	Specifies how the information in the form should be encoded before sending it to the server
method	Specifies how the information is sent from the browser to the server
name	Refers to the name of the form
novalidate	Specifies that the form should not validate while submitting
target	Opens the action URL in the specified target, such as in the same window, in a new window, or in a new tab

# HTML5 Forms

- **INPUT Types Tag**
- A form is composed of controls, such as text box, drop-down list, check-box, and radio button that enable the user to enter the information.
- You can create these controls by using the INPUT tag. To do this, you need to set the type attribute of the INPUT tag to the name of the control.
- For example, `<INPUT type="text">` creates a control of text box type.

# HTML5 Forms

## Types of INPUT Tag

- text and search
- tel
- url
- email
- password
- datetime-local
- datetime, date, month, week, and time
- number and range
- file
- hidden
- checkbox
- radio
- submit
- reset



# HTML5 Forms

- **HTML text and search Type**
- A text box (or text field) is a rectangular-shaped box that facilitates a user to enter the information.
- It is used to collect single line information, such as name, date of birth, telephone number, or e-mail, from the user.
- You can create a text box inside a form by setting the type attribute of the INPUT tag to text; for instance, `<INPUT type="text">`.
- In addition, the attributes of the input tag that specifies the features of the text box are as follows :

# HTML5 Forms

- **Name** - Provides a name to the text box, so that the program that handles the information provided in the form can identify the text box. For example, `<INPUT type="text" name="username">`
- **Size** - Defines the size of the text box. It defines the number of visible characters that can be seen in the text box. For example, `<INPUT type="text" name="username" size="30">`
- **Maxlength** - Specifies the maximum number of characters the user can enter in the text box.
- **Value** - Specifies the default text that you want to display in the text box when the form is first loaded

# HTML5 Forms

- **HTML tel Type**
- The tel type of the input control is used to enter the telephone numbers.
- In this type, you can only enter numbers, as it does not accept alphabets.
- There is no specific syntax to specify the telephone number in the input control, as there is a wide variety of valid telephone numbers.
- You can also specify particular pattern to enter the telephone number by using the pattern attribute of the INPUT tag.

# HTML5 Forms

- **HTML url Type**
- The url type of the input control is used to enter a valid path of a Web page.
- A valid url of a website consists of a protocol, a domain name, and a pathname.
- The most widely used protocols are Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP).
- The value of the url field is automatically validated when you submit the form.

# HTML5 Forms

- **HTML email Type**
- the email type of the input control is used to enter a valid e-mail address, which should contain a @ symbol and a dot (.).
- The value of the e-mail field is automatically validated when you submit the form.
- You can also enter more than one e-mail address in the e-mail field by using the multiple attribute, which specifies that the multiple e-mail addresses can be entered in the input control.

# HTML5 Forms

- **HTML password Type**
- A password is used to protect the secret information that a user does not want to share with others.
- You can create a password field by using the INPUT tag and setting its type attribute to password.
- The password field is similar to the text box but the text entered in the password field is not in a readable format, as the asterisk symbol or dot is displayed in the place of the characters.

# HTML5 Forms

- Therefore, it is also referred as a masked text box. The attributes of the INPUT tag that specifies the features of the password field are as follows :

**Name** - Provides an internal name to the password field, so that the program that handles the form identifies the field. The name attribute is set through the INPUT tag.

**Size** - Defines the length of the password field. If the size is not specified, the browser takes the default size, which is 20.

**Maxlength** - Specifies the maximum number of characters that the user can enter in the password field.

**Value** - Specifies the value that is to be displayed in the password field in the masked form, which means that the value is reflected as dots or asterisks.

# HTML5 Forms

- **HTML datetime-local Type**
- The datetime-local type of the input control is used to enter the date and time in the input control.
- When the date and time is defined as the type of the input control, a date time picker is provided on a form to select the date, month, year, and time.
- You can select the date and time according to your appropriate time zone.
- In addition, you can set the minimum and maximum date and time values by using the min and max attributes.
- If you want to convert the seconds into the milliseconds, you can use the step attribute.



# HTML5 Forms

- **HTML datetime, date, month, week, and time Types**
- You can also set the type of the input control to datetime, date, month, week, and time separately.
- The datetime type allows you to enter year, month, day, hour, minute, second, and fraction of a second in the Coordinated Universal Time (UTC).
- The date type allows you to enter the day, month, and year by using the date time picker.
- The month type allows you to enter the month in the input control. You can also set the minimum and maximum month by specifying the min and max attributes of the month type, respectively.
- The week type is used to enter a week of a month.
- The time type is used to enter only time in the input control.

# HTML5 Forms

- **HTML number and range Types**
- The number types is used to enter only numbers in the input control.
- You cannot enter any alphabet as the number type does not accept it and prompts you to enter only numbers at the time of submitting the form.
- The min and max attributes can be used to specify the minimum and maximum number to be entered in the input control. The range type of the input control allows you to enter a value within a specific range.

# HTML5 Forms

- **HTML file Type**
- The file type allows a user to select a file stored in the local computer and send it to the server on submission on the form.
- You need to set the type attribute to the file to create a file selection field.
- A user can either type the pathname of the file directly into the file selection field or use the browse option to select the pathname of the file from a system-specific dialog box.

# HTML5 Forms

- In case of using the browse option, even if the pathname exceeds the maxlength specified, the browser accepts the complete pathname. The attributes of the file selection field are as follows :
- **Size** - Defines the width of the visible text on the file selection field
- **Maxlength** - Specifies the maximum number of character that can be entered in the file selection field
- **Accept** - Specifies the type of files that can be submitted through a file upload

# HTML5 Forms

- **HTML hidden Type**
- A hidden field is used to pass along variables and values from one form to another, without forcing the user to re-enter the information.
- In addition, it is not displayed by the browser. You can create a hidden field by using the INPUT tag and setting its type attribute to hidden. The attribute of the INPUT tag that specifies the features of the hidden field are as follows :
- **Name** - Specifies a name of the hidden field
- **Value** - Specifies the value that is to be displayed by default in the hidden field

# HTML5 Forms

- **HTML checkbox Type**
- A checkbox is used to select or deselect one or more items from a given set of items that are displayed on the form. It helps a user to select the items quickly and easily by providing the options.
- You can create a checkbox by setting the type attribute of the INPUT tag to checkbox. The attribute of the INPUT tag that specifies the features of the checkbox field are as follows :
- **Name** - Provides a name to the checkbox field. The following example shows the use of the name attribute :  
`<input type="checkbox" name="cars">`

# HTML5 Forms

- **Value** - Determines the value that should be transmitted to the server after selecting the corresponding option on the checkbox.
- The following example shows the user of the value attribute :  
`<input type="checkbox" name="cars" value="audi">`
- **Checked** - Provides the default item, which has to be sent to the server along with the items that are selected by the user. In addition, a user can clear the checked item as per the requirement.
- The following example shows the user of the checked attribute :  
`<input type="checkbox" name="cars" value="audi" checked="yes">`

# HTML5 Forms

- **HTML radio Type**
- The usage of a radio button is almost similar to a checkbox except that, in a check box more than one item can be selected, however, in a radio button, only one item can be selected.
- Therefore, a radio button can be described as a field having a list of items from which a user has to select a single item.
- The radio button field can be created by setting the type attribute of the INPUT tag to radio.
- The attributes of the INPUT tag that specifies the features of the radio button field are as follows :



# HTML5 Forms

- **Name** - Provides a name to the radio button field. Unlike the check box where different items can be named differently, each radio button in a group has the same name.
- The following example shows the use of the name attribute :  
`<input type="radio" name="cars">`
- **Value** - Refers to the value that should be transmitted to the server when the corresponding option for the radio button is selected.
- The following example shows the use of the value attribute :  
`<input type="radio" name="cars" value="audi">`
- **Checked** - Provides the default item, which is selected and sent to the server if the user does not select any item.
  - If the user selects any other item from the list, the default item is automatically deselected.
- The following example shows the use of the checked attribute :  
`<input type="radio" name="cars" value="audi" checked="yes">`

# HTML5 Forms

- **HTML submit Type**
- When a user clicks the submit button, the form is sent to the address specified by the URL.
- You can create a submit button by setting the value of the type attribute of the INPUT tag to submit.
- The attributes of the INPUT tag that specifies the features of the submit button are as follows :
  - **Name** - Specifies the name of the submit button
  - **Value** - Specifies the label that is displayed on the submit button

# HTML5 Forms

- **HTML reset Type**
- When a user clicks the reset button, information of all the fields in the form are erased and set to the default values.
- The reset button is created by setting the type attribute of the INPUT tag to reset.
- The attributes of the INPUT tag that specifies the features of the reset button are as follows :
  - **Name** - Specifies the name of the reset button
  - **Value** - Specifies the label that is displayed on the reset button

# HTML5 Forms

## **BUTTON tag**

- In HTML, the BUTTON tag is used to add a button control on the [form](#).
- A button control can be used to perform various tasks, such as submitting or resetting the details of the form.
- The BUTTON tag can be used with the conjunction of FORM tag to display the controls on the form.
- A button control can also be placed on the form by using the [INPUT tag](#), but the difference is that you do not have the scope to change the appearance of the button control, except changing the text on the button control.
- However, you can change the appearance of the button control in case you have created the button control by using the BUTTON tag.

# HTML5 Forms

- The button control is created by using the opening and the closing tags of the `BUTTON` tag.
- The text, image, or any multimedia embedded between the opening and the closing tags of the `BUTTON` tag become the content of the button control.
- The `BUTTON` tag provides a `type` attribute that allows you to create three kinds of button controls i.e. submit button, reset button, and normal button.
- The submit button is used to submit the form, whereas the reset button is used to erase all the text entered in the text box of the form and set the default values.
- The submit button is created by setting the `type` attribute to `submit`, the reset button is created by setting the `type` attribute to `reset`, and the normal button is created by setting the value of the `type` attribute to `button`.

# HTML5 Forms

Button Tag Attributes	Description
autofocus	Allows the button control to get the focus as soon as page loads
disabled	Disables the button control
form	Refers to the id of the FORM tag
formaction	Refers to the value of the action attribute of the current form
formenctype	Specifies a value that is used to encode the content while submitting to the server. The possible values are application/x-www-form-urlencoded (default), multipart/form-data, text/plain
formmethod	Specifies the methods of the HTTP at the time of the submitting the button control. The possible values are get, post, put, and delete
formnovalidate	Specifies that the form is not to validate at the time of submitting the button control

# HTML5 Forms

Button Tag Attributes	Description
formtarget	Specifies the destination, such as a new tab or a new window, to load the browsing context
name	Provides a name of the button control
type	Specifies the type of the button control. The possible values are submit, reset, and button
value	Provides a value to the button control. You can use this attribute if the form attribute is present

# HTML5 Forms

- **Multiple-Choice Tags**
- Multiple-Choice tags offer multiple choices to the user in a Web page, such as check boxes and radio buttons.
- For instance, you can use these multiple-choice tags in a Web page to display the multiple choice questions and answers.
- However, the Web page appears cumbersome when all the choices are displayed on it.
- Therefore, the multiple-choice tags, such as the SELECT, OPTION, and CAPTION tags, are used to manage choices.
- This is divided into the following three parts:
  - Select Tag
  - Option Tag
  - Optgroup Tag



# HTML5 Forms

- The **SELECT** tag allows the user to select a single item from number of options.
- Unlike the radio button, the SELECT tag does not provide any default option that is to be transmitted to the server when no option is selected by the user.
- The options are written within the opening and the closing tags of the SELECT tag by using the **OPTION** tag.

# HTML5 Forms

Select Tag Attribute	Description
disabled	Implies that the drop-down list is disabled
name	Refers to the name of the drop-down list
size	Refers to the number of visible options shown in the drop-down list
autofocus	Allows the button control to get the focus as soon as the page loads
form	Refers to the id of the FORM
multiple	Specifies that the multiple items can be selected from the drop-down list

# HTML5 Forms

- The **OPTION** tag is used to define the options written within the SELECT tag.
- The options are created by embedding the OPTION tag within the opening and the closing tags of the OPTION tag.
- Each option is separately written within a separate set of OPTION tag can not contain any other tag within it.

# HTML5 Forms

Option Tag Attribute	Description
label	Refers to the heading of the several groups
disabled	Disables the OPTION tag
selected	Refers to the option that is to be displayed as default
value	Refers to the value that is sent to the server

# HTML5 Forms

- The **OPTGROUP** tag is used to create nested and cascading drop-down lists.
- In both type of lists, the related items are grouped under specific headings.
- The following table describes the various attributes of the OPTGROUP tag.
  - **Label** - Refers to the heading of the several groups in the cascading menu
  - **Disabled** - Disables the OPTGROUP tag

# HTML5 Forms

- **HTML TEXTAREA and LABEL Tags**
- **Textarea** is similar to the text box except in the text box, you can enter only a single line of information, whereas, in the textarea, you can enter multiple lines of information.
- The content provided within the starting and the ending tags of the TEXTAREA tag should only be plain text.
- You can adjust the size of the textarea by using the two attributes i.e. cols and rows.
- The wrap attribute of the TEXTAREA tag defines how the text appears in the textarea field when it reaches the end of every row. Wrapping can have soft, hard, and off settings.

# HTML5 Forms

- The soft setting forces the words to wrap inside the textarea, but when the form is submitted, it includes the line breaks.
- The hard setting wraps the words inside the textarea and places line breaks at the end of each line so that when the form is submitted, it appears exactly as it appears in the textarea.
- The offsetting ignores all the wrapping in the textarea and places the text into one ongoing line.

# HTML5 Forms

Textarea Attribute	Description
cols	Refers to the visible width of the textarea control
rows	Refers to the permitted number of rows in the textarea control
disabled	Disables the textarea
name	Refers to the name of the textarea
readonly	Specifies that the textarea is read-only and you cannot write in it
accesskey	Refers to the shortcut key on the keyboard
autofocus	Allows the control to get the focus as soon as the page loads



# HTML5 Forms

Textarea Attribute	Description
dirname	Specifies the name of the input control that indicates the text direction of the textarea
maxlength	Specifies the maximum number of character that can be entered in the textarea
placeholder	Helps the user to fill the respective textarea by providing the hint for the input
required	Specifies that the value of the input field is required to submit the form
wrap	Allows the textarea to wrap the text

# HTML5 Forms

- Some controls, such as button control, do not require any description as they already have labels associated with them, whereas, some controls, such as text boxes, check boxes, and radio buttons, need description.
- You can provide the description of the control by adding a label on the form using the **LABEL tag**.
- Each LABEL is associated with exactly one control.
- The following table describes the attributes of the LABEL tag.
  - For -- Associates the label with a specific control. The value of this attribute must match the id attribute of its associated control.
  - Form -- Refers to the id of a form

# HTML5 Forms

- **HTML FIELDSET Tag**
- The FIELDSET tag is used to group related controls in a single box.
- Grouping the related controls displays the form fields in a more organized manner.
- Suppose you have a form that is used for user registration as well as for the login process.
- Using the FIELDSET tag, you can group the controls for the login process and the user registration process in two separate boxes.
- This makes the user easily understand the purpose of each control on the form.

# HTML5 Forms

- **Attributes of the FIELDSET tag**
- The following table describes the attributes of the FIELDSET tag.
  - Disabled -- Disables the textarea
  - Name -- Refers to the name of the textarea
  - Form -- Refers to the id of a form
- **HTML LEGEND Tag**
- The LEGEND tag is used to provide caption for the FIELDSET tag.
- Using the LEGEND tag, you can provide the caption for the login process and the user registration process as Sign-In and Create an Account, respectively.

# HTML5 Forms

- **HTML DATALIST Tag**
- The DATALIST tag is used to display the list of the predefined options that the user may want to select as input.
- This tag enables the auto complete feature on the forms. This means that when a user start typing in a text box, a list of predefined words is dropped down to choose.
- The DATALIST tag is used with the [INPUT tag](#), in which the list attribute is specified.
- The value of the list attribute is similar to the id attribute of the DATALIST tag to link the INPUT tag with the DATALIST tag.
- The [OPTION tag](#) used as the child tag of the DATALIST tag is used to specify the list of the options that are to be displayed.

# HTML5 Forms

```
<INPUT type="text" list="name">
```

```
<datalist id="name">
```

```
  <option value="Richard">
```

```
  <option value="John">
```

```
  <option value="Tony">
```

```
</datalist>
```

- Here, we have used the INPUT tag and the DATALIST tag. We have specified the same value for the list attribute of the INPUT tag and the id attribute of the DATALIST tag.
- We have also defined the OPTION tags and specified their value attributes as the options to be displayed.

# HTML5 Forms

- **HTML KEYGEN Tag**
- The KEYGEN tag is used to generate the key pair. When a form is submitted, a key pair, which contains the private and public keys, is generated using the KEYGEN tag to secure the content of the form.
- The private key from the generated key pair is encrypted and stored in the key database on local computer.
- The public key is encrypted and submitted to the server along with the form.

# HTML5 Forms

- Following is the example of the KEYGEN tag, as shows in the following code snippet.

```
<KEYGEN name="key_name"  
challenge="987621">
```



# HTML5 Forms

Keygen Attribute	Description
autofocus	Allows the control to get the focus as soon as the page loads
challenge	Specifies a string that is used for the verification at the time of submission of a form
disabled	Disables the input control
form	Refers to the id of the FORM tag
keytype	Specifies the type of the key to generate
name	Provides a name to the input control

# HTML5 Forms

- **HTML OUTPUT Tag**
- The OUTPUT tag is used to display the result of the calculation, which can be written using the [JavaScript](#).
- The OUTPUT tag has three attributes i.e. form, name, and for.
- The form attribute is used to specify the name of the form in which the output is displayed.
- The name attribute is used to specify the name of the current tag and the for attribute is used to specify the name of the control in which the result is displayed.

# HTML5 Forms

- Following is an example of using the OUTPUT tag.

```
<FORM>
```

```
  <OUTPUT name="result">
```

```
  </OUTPUT>
```

```
</FORM>
```

- **OUTPUT Tag Attributes**

- **For**-- Associates the output with a specific control. The value of this attribute must match the id attribute of its associated control.
- **Form**--Refers to the id of a form
- **Name**-- Specifies the name of the OUTPUT tag

# HTML5 Forms

- **HTML PROGRESS Tag**
- The PROGRESS tag is used to display the progress of a particular task that is being performed.
- The PROGRES tag is used in conjunction with [JavaScript](#) to display the progress or the process of a task.
- The PROGRES tag has only two attributes i.e. value and max.
- The value attribute is used to specify how much the task has been completed and the max attributes is used to specify the total progress to be made.
- **PROGRESS Tag Example**  
**<PROGRESS value="100" max="500"> </PROGRESS>**
- **Attributes used within the PROGRESS tag**
  - **Value** -- Specifies the value of the PROGRESS tag
  - **Max** --Specifies the maximum value of the PROGRESS tag

# HTML5 Forms

- **HTML METER Tag**
- The METER tag is used to define the scalar measurement. This tag is mostly useful when you need to display the disk usage and the relevance of a search result, or to show some other measurement.
- You cannot use the METER tag to display a single number as it is used to display a range.
- **METER Tag Example**  
`<METER min="50" max="165"> 72 </METER>`  
`<METER>200 from 500</METER>`

# HTML5 Forms

- Here, in the first example, we have used the min and max attributes of the METER tag to specify the minimum and maximum values for the tag.
- This implies that you cannot define the value of the METER tag less than 0 and more than 10.
- And in the second example, we have displayed the tag without any attribute.

# HTML5 Forms

Meter tag Attribute	Description
Value	Specifies the value of the METER tag
min	Specifies the minimum value of the METER tag
max	Specifies the maximum value of the METER tag
low	Specifies a range which is considered as low value
high	Specifies a range which is considered as high value
optimum	Specifies a range which is considered as the optimum or the best value
form	Refers to the id of the FORM tag

# HTML5 Forms

- **HTML Submit Form**
- After the information in a [form](#) is entered by a user, the user clicks the submit button to send the information contained in the form to the server for processing.
- The whole process of transmission of the information from browser to the server is performed by the [enctype](#), [action](#), and [method](#) attributes of the [FORM tag](#).
- The enctype attribute encodes the information from the form before transmitting it from the browser to the server.
- The action attribute specifies the URL of the program that handles the information contained in the form and the method attribute specifies how the information is transmitted from the browser to the server.



# HTML5 Forms

- **HTML enctype Attribute**
- The enctype attribute is used to encode the information in the form before sending it to the server so that the information is not corrupted during the transmission. The encoding is performed by the browser.
- Three types of encoding process are application/x-www-form-urlencoded, multipart/form-data, and text/plain.
- The application/x-www-form-urlencoded encoding is the standard form of encoding. If no value is set for the enctype attribute, browser accepts the standard type.
- The multipart/form-data encoding is required for those forms that contain file selection field.
- You can use the text/plain attribute along with the mailto URL to send the form information to an e-mail address rather than a server.

# HTML5 Forms

- **The application/x-www-form-urlencoded Encoding**
- The application/x-www-form-urlencoded encoding acts as a default type of encoding for the browser.
- This implies that if the enctype attribute is not set with any value, the browser by default uses the application/x-www-form-urlencoded encoding.
- In this encoding, the browser converts a space into a plus sign (+), a non-alphanumeric character into a percent sign (%) followed by two hexadecimal digits that are ASCII code of character, and line breaks into CR/LF pairs (%0D%0A).
- The standard encoding also contains the name of each form controls that is present in the form.
- Each control name is separated from other control name by an ampersand (&).

# HTML5 Forms

- **Example of application/x-www-form-urlencoded Encoding**
- Let's consider a form, which contains four controls named username, password, name, and address. The values of these controls are richjones, rich23, richard, and Sydney, Australia respectively.
- In such a case, the encoding of the information in the form is done in the following way :

**Username=richjones&password=rich23&name=richard&address=Sydney,+Australia**

# HTML5 Forms

- **The multipart/form-data Encoding**
- The multipart/form-data encoding is used only when the method attribute of the form is set to post.
- In this type of encoding, the controls in the form are segregated in several parts.
- This encoding is more cumbersome and longer than the application/x-www-form-urlencoded encoding.
- **Example of multipart/form-data Encoding**
- Let's consider the example used in the application/x-www-form-urlencoded encoding.
- The form contains four controls named username, password, name, and address, and their values are richjones, rich23, richard, and Sydney, Australia, respectively. The encoding of the given example in the multipart/form-data encoding is as follows:

# HTML5 Forms

-----1234567897987

Content-Disposition: form-data; name="username"

richjones

-----1234567897987

Content-Disposition: form-data; name="password"

rich23

-----1234567897987

Content-Disposition: form-data; name="name"

richard

-----1234567897987

Content-Disposition: form-data; name="address"

Sydney, Australia

-----1234567897987

# HTML5 Forms

- As already discussed this type is preferred only when the form contains one or more file selection control.
- Let's consider an example where the form contains a file selection control named thefile and the filename is file.
- In this case, the multipart/form-data encoding is given as follows:

```
-----1234567897987
Content-Disposition: form-data; name="thefile"; filename="file"
```

Content of the file...

```
-----1234567897987
```

# HTML5 Forms

- **The text/plain Encoding**
- The text/plain encoding is used when the browser does not have access to the server that contains the form-handling program.
- In this case, the information from the form is sent through an e-mail and the action attribute is set to a mailto URL.
- In this type of encoding, each control is written in a single line, with name and value separated by an equal to sign (=).
- Space is denoted by space and line break is denoted by CR/LF pairs (%0D%0A).

# HTML5 Forms

- **Example of text/plain Encoding**
- Let's again consider the example used in the application/x-www-form-urlencoded encoding in which, there are four controls named username, password, name, and address.
- The encoding of the information in the form is given as follows:

**username=richjones**

**password=rich23**

**name=richard**

**address=Sydney, Australia**



# HTML5 Forms

- **HTML form action Attribute**
- The action attribute of the FORM tag provides the URL of the program (which is in the server) that receives the information from the form and processes it.
- This URL is also referred as form's action URL. Most of the servers keep these form processing programs in a bin known as Common Gateway Interface-binaries (cgi-bin)

- **HTML form action Attribute Example**

**<FORM action="http://codescracker.com/cgi-bin/update">**

- Here, the name of the program that handles the form is updated, which is placed in the CGI-BIN directory. This directory is placed in the codescracker.com domain

# HTML5 Forms

- **HTML method Attribute**
- The method attribute specifies the method by which the information in the form is sent to the server for processing.
- There are the two acceptable values for the method attribute are [get](#) and [post](#).
- The get method sends the information by adding it in the URL of the Web page. This method is used when the data is small and there is no issue for securing it.
- And the post method first connects to the server and then sends the data. This method is more secure than the get method.

# HTML5 Forms

- **HTML Get Method**

- The get method is the default method for browsers to submit the information. In the case of get method, the browser connects to the server, which processes the form and sends the information in a single transmission step. The browser appends the data to the form's action URL, separated by a question mark.
- As the get method places the information directly in the form's action URL, the information can easily be captured.
- The information transmitted through the get method is the default value for the method attribute.

- **Example of Get Method**

**<FORM method=get action="http://codescracker.com/update">**

- The get method is commonly used for the transmission of short information

# HTML5 Forms

- **HTML Post Method**
- The post method is more appropriate than the get method for the transmission of large amount of information.
- In the case of post method, the browser sends the information in two steps.
- In the first step, browser contacts to the form-processing program as specified in the form's action URL.
- In the second step, the browser sends the information to the server.
- **Example of Post Method**  
**<FORM method=post**  
**action='http://codescracker.com/update'>**

# HTML5 Images, Colors and Canvas

- While creating a website, developers try to make it as attractive as possible by providing the information in an effective and reader friendly manner.
- A website provides information in the two forms, namely [text](#) and [images](#). Information presented in the form of images is easier to retain for users in comparison to the information written in plain text.
- HTML provides a tag named IMG which is used to add or insert images in a Web page.
- You can use images in various forms, such as a logo, a diagram or an icon. You can also apply an images as a background of a Web page or website.

# HTML5 Images, Colors and Canvas

- In addition to images, you can use different [colors](#) in the content of your website to change its appearance.
- This can be done by using one of the following three ways:
  - color names
  - Hexadecimal (Hex) value
  - Red Green Blue (RGB) value
- To further enhance the appearance of a website, HTML also provides [CANVAS](#) tag.
- This tag allows you to make your images more appealing by changing their brightness and contrast. You can also draw 2D shapes and graphs on a Web page by using the CANVAS tag.

# HTML5 Images, Colors and Canvas

- **Insert Image in Web page**
- HTML allows you to insert an image in a Web page with the help of the IMG tag. This tag uses several attributes, such as src, id, lang, dir, and alt.
- **Note** - Out of all the attributes of the IMG tag, only the src attribute is necessary.
- The src attribute provides the information about the path of the image file to the Web browser. If your image and HTML files are stored in the same folder, then there is no need to specify the full path of the image file in the src attribute.
- If your image and HTML files are in different folders, you have to specify the full path of the image file in the src attribute of the IMG tag

# HTML5 Images, Colors and Canvas

Img tag Attribute	Description
id	Assigns a unique identifier to a tag. This identifier must be used only once in a document. This attribute is optional
class	Assigns a single name or a set of class names to a tag. However, one or more tags can be assigned with a same class name. This attribute is optional
lang	Specifies the base language used for the IMG tag. This attribute is optional
dir	Assigns a direction to entire or a section of HTML file. This attribute is optional
title	Describes the objective of the use of the IMG tag. This attribute is optional
style	Applies inline CSS style on individual tags in an HTML file. This attribute is optional
src	Specifies the URL or the location of the image. This attribute is mandatory



# HTML5 Images, Colors and Canvas

Img tag Attribute	Description
alt	Specifies the alternate text to be used, if the Web browser cannot render the image. This attribute is optional
height	Specifies the height of the image. This attribute is optional
width	Specifies the width of the image. This attribute is optional
ismap	Indicates that the image is used as an image map. This attribute is optional
usemap	Associates a tag with an image map. This attribute is optional

# HTML5 Images, Colors and Canvas

- **HTML Screen Readers**
- Screen readers are software programs that can read what is displayed on a screen
- Used on the web, screen readers can "reproduce" HTML as text-to-speech, sound icons, or braille output
- Screen readers are used by people who are blind, visually impaired, or learning disabled
- **Note** : Screen readers can read the alt attribute

# HTML5 Images, Colors and Canvas

- **HTML Image Formats**
- You can use different types of image formats in Web pages. Some of the commonly used image formats are:
  - **GIF** - used to create illustrations, such as logos or cartoons
  - **JPEG** - used to generate complex images, such as photographs
  - **PNG** - used with the images that have more number of colors
- GIF and JPEG are supported by all the Web browser whereas, PNG is supported by some of the Web browsers.

# HTML5 Images, Colors and Canvas

- **The GIF Image Format**
- The GIF image format was developed for CompuServe (an online information service) before the Internet was invented. Converting an image to an GIF format is safe, as no data is lost at all time of conversion. There are the following two types of GIF images:
  - **GIF87** - Does not support the image transparency and animation
  - **GIF89a** - Supports image transparency and animation
- Both types of GIF images support 256 colors and have the .gif file extension

# HTML5 Images, Colors and Canvas

- **The JPEG Image Format**
- In comparison to GIF, JPEG supports unlimited number of colors and has the .jpeg or .jpg extension. JPEG uses a complex compression algorithm, known as the JPEG algorithm.
- You can convert any image format into the JPEG image format by using the JPEG algorithm. This algorithm focuses on various important aspects, such as brightness and contrast of the image.
- The JPEG image format has certain standards that specify codecs, which define the process to compress an image into bytes and again decompress the bytes into the image.
- Exchangeable Image File Format (EXIF) and Joint Photographic Experts Group (JFIF) are the most commonly used standards of JPEG.

# HTML5 Images, Colors and Canvas

- **The PNG Image Format**
- The PNG image format has all the features of GIF. PNG uses a lossless compression algorithm and supports an unlimited number of colors. The lossless compression algorithm prevents any loss that occurs while compressing an image.
- PNG provides alpha transparency, which allows you to have variable levels of opacity as compared to GIF. In the true alpha transparency, each pixel of the image can be stored with an alpha value that indicates the amount of transparency of the pixel.
- The alpha value can be adjusted to make a PNG image completely transparent or completely opaque.

# HTML5 Images, Colors and Canvas

- **HTML Color Codes**
- In HTML, you can specify the colors for the [text](#) and [background](#) of a Web page to make it more attractive and appealing.
- HTML defines 16 widely known colors that you can apply to a Web page by specifying the name of the color as the value of the text attribute of the BODY element.
- Alternately, you can use the hexadecimal (Hex) values of colors, which imply that the value of the text attribute must be the hexadecimal value of the color. You can also apply colors to a Web page by using the RGB color mode. In this case, the text attribute contains the RGB value of the color.
- You can apply colors in a Web page with the help of
  - Color names
  - Hex values
  - RGB configuration
  - Web-safe colors

# HTML5 Images, Colors and Canvas

- **HTML Color Names**
- You can use a color in a Web page by simply specifying its name. However, there are only 16 colors that can be specified by their names. For example, you need to design a Web page, in which you want to display the text in red color. You can do so by using the text attribute of the BODY element. You can specify a color by typing the name of the color as the value of the text attribute in the BODY element.
- **HTML Color Hex Values**
- The Hex values, also referred as hexadecimal numbers, are 6 digits or three bytes number that start with the # sign. The three bytes present in every hexadecimal number represent the following :
  - **Byte 1** - Represents the red color
  - **Byte 2** - Represents the green color
  - **Byte 3** - Represents the blue color



# HTML5 Images, Colors and Canvas

- Each byte contains some numbers from 00 to 99 and alphabets from AA to FF in hexadecimal notation.
- The Hex values are defined through the combination of values assigned to red, green, and blue colors. When you use a Hex value to apply a color in an image, you can use numerals as well as alphabets.
- In Hex value, 0 represents the darkest color that is black and F represents the lightest color that is white.
- For example, the hexadecimal number for black is #000000. The first two digits (00) represent the amount of red color.
- The second two digits (00) represent the amount of green color, and the last two digits (00) represent the amount of blue color.

# HTML5 Images, Colors and Canvas

- **HTML Color RGB Values**
- The process of displaying colors by using the different combinations of red, green, and blue is known as the RGB configuration.
- It is a set of three dials, where the first dial represents red, second dial represents green, and the third dial represents the blue color.
- In the RGB configuration, the value of each color (red, green, and blue) ranges from 0 to 255 in decimal.
- The color specified by 255, 0, 0 represents the red color because its first dial contains the value 255 that represents the red color, while the second dial representing the green color and the third dial representing the blue colors contains 0 (means no color).

# HTML5 Images, Colors and Canvas

- In the same way, the color specified by 0, 255, 0 represents the green color, and the color specified by 0, 0, 255 represents the blue color.
- If all the three dials are turned off, such as 0, 0, 0 then this is the condition of the absence of color or simply black color.
- On the other hand, if all the dials are turned on, such as 255, 255, 255 then it represents the white color.

# HTML5 Images, Colors and Canvas

Color Name	RGB Value	Color Name	RGB Value
Black	rgb(0, 0, 0)	Lime	rgb(0, 255, 0)
Silver	rgb(192, 192, 192)	Olive	rgb(128, 128, 0)
Gray	rgb(128, 128, 128)	Yellow	rgb(255, 255, 0)
White	rgb(255, 255, 255)	Navy	rgb(0, 0, 128)
Maroon	rgb(128, 0, 0)	Blue	rgb(0, 0, 255)
Red	rgb(255, 0, 0)	Teal	rgb(0, 128, 128)
Purple	rgb(128, 0, 128)	Aqua	rgb(0, 255, 255)
Fuchsia	rgb(255, 0, 255)	Green	rgb(0, 255, 0)

# HTML5 Images, Colors and Canvas

- **HTML Web-safe Colors**
- Earlier, a computer system is used to support only 256 colors, out of which 216 colors are Web standard colors displayed correctly by all computer systems.
- These colors are known as Web-safe colors that are only defined by using the values, 00, 33, 66, 99, AA, CC, or FF.
- A Web-safe color can use any combination of these values. For example, #99FF00 is a Web-safe color but #221144 is not a Web-safe color.
- The 00 represents the darkest value for each color and the FF represents the brightest value for each color.

# HTML5 Images, Colors and Canvas

- **HTML Canvas**
- The CANVAS tag of HTML is introduced in HTML5, used to display 2D shapes and graphics on the Web page. This tag allows you to use graphs, games, and other visuals in the Web page.
- You can also use the CANVAS tag to apply various transformations, such as rotate and blur on an [image](#). The content that is defined between the starting and the ending tags of the CANVAS tag is only displayed when the Web browser does not support the tag.
- The [attributes](#) of the CANVAS tag are height and width, which specify the height and width of the canvas.
- **HTML CANVAS Example**  
`<CANVAS id="canvas" width="50" height="100"> </CANVAS>`
- As shown in the above code snippet, we have also defined another attribute of the CANVAS tag that is the id attribute, which is used to specify a unique name to the tag. The id attribute of the CANVAS tag is used with the [JavaScript](#) code to display graphics on the Web page.

# HTML Interactive Web Tags

- Interactivity is the process of presenting the information on the Web page and allowing the users to communicate with that Web page.
- An interactive Web page allows a user to easily perform the tasks related to the content of the Web page, such as retrieving the details of an article or executing commands using the radio buttons or check boxes.
- You can display the content on a Web page by using the different elements that help the users to interact with the Web page.
- For instance, if your Web page contains an article, then you can display the name of the author, publishing the date, and summary of the article using the [DETAILS and SUMMARY tags](#).

# HTML Interactive Web Tags

- You can also display a menu on your Web page by using the [MENU tag](#), which provides a list of options to the users, such as Home, About Us, and Contact.
- Sometime, you need to display keyboard shortcut keys in the content of your Web page for the users.
- By default, HTML displays the textual content as plain text. As a result, users might not be able to identify or differentiate between the shortcut keys specified in the content and the normal text.
- In this situation, the KBD tag proves to be very beneficial as it changes the font style of the shortcut keys.



# HTML Interactive Web Tags

- **HTML DETAILS and SUMMARY Tags**
- The DETAILS tag is used to display the additional details about a document, such as publishing date of the document or the name of the author, whereas, the SUMMARY tag is used to provide the summary or the caption of the current document.
- It can also be used in the conjunction with the SUMMARY tag to create a header that can expand or collapse according to the requirement.
- The open attribute of the DETAILS tag is used to specify the details of the document that are displayed to the users.
- The absence of the open attribute hides the details from the users.

# HTML Interactive Web Tags

- Following is an example of using the DETAILS and SUMMARY tags :

**<DETAILS>**

**<SUMMARY>**This section contains the summary of the document

**</SUMMARY>**

**</DETAILS>**

# HTML Interactive Web Tags

- **HTML MENU Tag**
- The MENU tag is used to display a list of menu items on a Web page. It contains one or more LI tags that define the items of the list.
- You can display the menu items as a toolbar, a context menu, or a list of items by using the type attribute of the MENU tag.
- **Attributes of MENU tag**
  - Label -- Specifies a label of the menu, which is visible on the Web page
  - Type -- Defines the type of the menu, such as context menu, toolbar, or list (default)

# HTML Interactive Web Tags

- Following is an example of using the MENU tag :

```
<MENU>
```

```
  <LI><INPUT type="radio" />Audi</LI>
```

```
  <LI><INPUT type="radio" />Ferrari</LI>
```

```
</MENU>
```

# HTML Interactive Web Tags

- **HTML COMMAND Tag**
- The COMMAND tag is introduced in HTML5 and is used to define the command button, such as radio button, check box, or button.
- You can use the COMMAND tag in conjunction with the MENU tag.
- The COMMAND tag can also be used without the MENU tag, but only for specifying the keyboard shortcuts.
-

# HTML Interactive Web Tags

Attribute	Description
checked	Defines the selection of the command in case of radio button and checkbox
disabled	Disables the command
icon	Specifies the location of the graphical image that represents the command
label	Shows the name of the command to the user
radiogroup	Specifies the name of the group if the command type radio is defined
type	Defines the type of the command, such as command (default), checkbox, and radio

# HTML Interactive Web Tags

- Following is an example of using the COMMAND tag :

```
<MENU>
```

```
<COMMAND type="command">
```

```
Execute
```

```
</COMMAND>
```

```
</MENU>
```

# HTML Interactive Web Tags

- **HTML KBD Tag**
- The KBD tag is used to display the keyboard input text in a Web page.
- The KBD tag can also be used to identify the other inputs, such as voice commands.
- When you use the KBD tag, you need to define the keyboard input text between its starting and ending tags. The Web browser changes the font of the text to Courier New.
- You can also change the font of the keyboard input text according to your requirement with the help of the Cascading Style Sheet ([CSS](#)).



# HTML Interactive Web Tags

- Following is an example of using the KBD tag :

<P>To open the Task Manager, press

<KBD>CTRL</KBD> +

<KBD>ALT</KBD> +

<KBD>DELETE</KBD>.

</P>

- In the above example, CTRL+ALT+DELETE are the keys of the keyboard, which displayed in the Courier New font on the Web browser.

# HTML Interactive Web Tags

- **HTML TIME Tag**
- The TIME tag is newly introduced in HTML5 and is used to define the date and/or time on a Web page.
- The TIME tag displays the time in 24-hour clock format and the date in the proleptic Gregorian calendar format.
- The TIME tag provides the datetime attribute, which is used to specify the valid date and time.
- **Attributes of TIME Tag**
  - Datetime -- Specifies the date and/or time
  - Pubdate -- Specifies the publishing date and time of a document

# HTML Interactive Web Tags

- Following is an example of using the TIME tag :

<P>This example is written on

    <TIME datetime="2009-11-10">Friday</TIME>.

</P>

- In addition to the datetime attribute, the TIME tag also contains the pubdate attribute. This attribute is more useful when the TIME tag is defined inside the ARTICLE tag.
- The pubdate attribute indicates that the date and time specified in the TIME tag is the publication date of the article defined in the nearest ancestor ARTICLE tag.

# HTML Multimedia

- The term media refers to various means of communicating and disseminating information, such as text, images, graphics, audio, video, and animation. All these mediums of communication are collectively termed as multimedia.
- In other words, the term multimedia encompasses all means of communication, from a single magazine containing text and images to an advance application containing audio, video, and graphics.
- A combination of video and audio files can also be used in websites to gain popularity in terms of viewership or provide information and entertainment to the users.
- HTML helps you to add multimedia files on your website by providing various multimedia tags. These tags include [AUDIO](#), [VIDEO](#), [EMBED](#), and [OBJECT](#).

# HTML Multimedia

- The AUDIO tag is used to display the audio file on the Web page, whereas the VIDEO tag is used to display the video files on the Web page.
- The EMBED and OBJECT tags display the multimedia files on a Web page as well as embed the files from other websites.
- HTML5 has introduced two new multimedia tags, [AUDIO and VIDEO](#), for displaying the audio and video streams on a Web page.
- You can play the multimedia files, which are stored in your local computer, on the Web page by specifying their location. The src attribute is used to specify the multimedia file to play it on the Web page.
- If the Web browser does not support AUDIO and VIDEO tags, then the text defined between the starting and the closing tags of these tags are displayed on the Web page.

# HTML Multimedia

- The AUDIO tag of HTML5 supports only three audio file formats i.e. .ogg, .mp3, .wav

Audio Attribute	Description
autoplay	Plays the audio file as soon as the Web page loads
controls	Displays the controls on the Web page, such as play and pause buttons
loop	Replays the audio file
preload	Specifies whether the audio file is preloaded on the Web page or not
src	Provides the location of the audio file to play

# HTML Multimedia

- You can use the VIDEO tag to display a video file on the Web page. The VIDEO tag supports the .gov and .mp4 file formats.

Video Attribute	Description
audio	Controls the default state of the video's audio channel
autoplay	Plays the audio file as soon as the Web page loads
controls	Displays the controls on a Web page, such as play and pause buttons
height	Specifies the height of the VIDEO tag
loop	Replays the video file
preload	Specifies whether the video file is preloaded on the Web page or not
poster	Provides an image to be displayed when the video file is not available
src	Provides the location of the video file to play
width	Specifies the width of the VIDEO tag

# HTML Multimedia

- You can also use the SOURCE tag within the opening and the closing tags of the VIDEO tag to provide the source of the video file.
- The SOURCE tag is used in a situation when the location of the video file is not confirmed. In this case, the VIDEO tag plays the first video file located in the specified path. The following code snippet shows the use of the VIDEO tag :

```
<VIDEO src="video.ogv" autoplay="true" loop="3" controls>  
</VIDEO>
```

- In the above code snippet, we have defined a video.ogv file in the src attribute.
- We have also set the autoplay attribute to true, which implies that the video file start playing as soon as the Web page loads.
- The loop attribute is set to 3, which implies that the video file will be played three times. In addition, the controls attribute displays the controls on the video player.



# HTML Multimedia

- **HTML Audio**
- An audio file is used to store audio data on various data, such as a computer system, mp3 players, and mobile phones.
- To store an audio data, you need to convert it into a digital format.
- The process of converting audio data into a digital file is called encoding of the raw audio data. It involves taking samples of audio data and storing them in a compressed format to reduce the file size.
- An audio player decodes these compressed sample files to make the audio waves audible. The process of converting a digital file into the audio data is known as decoding. A codec is performs the encoding and decoding of the raw audio data.

# HTML Multimedia

Format	File Extension	Description
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface). Main format for all electronic music devices like synthesizers and PC sound cards. MIDI files do not contain sound, but digital notes that can be played by electronics.
WMA	.wma	WMA (Windows Media Audio). Developed by Microsoft. Commonly used in music players. Plays well on Windows computers, but not in web browsers
RealAudio	.rm .ram	RealAudio. Developed by Real Media to allow streaming of audio with low bandwidths. Does not play in web browsers
WAV	.wav	WAV. Developed by IBM and Microsoft. Plays well on Windows, Macintosh, and Linux operating systems. Supported by HTML5
AAC	.aac	AAC (Advanced Audio Coding). Developed by Apple as the default format for iTunes. Plays well on Apple computers, but not in web browsers
MP3	.mp3	MP3 is the most popular format for music players.
Ogg	.ogg	Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5
MP4	.mp4	MP4 is a video format, but can also be used for audio. MP4 video is the upcoming video format on the internet.

# HTML Multimedia

- **HTML <audio> tag**
- To play an [audio file in HTML](#), use the <audio> tag.

**<audio controls>**

**<source src="songs.ogg" type="audio/ogg">      <source  
src="songs.mp3" type="audio/mpeg"> Your browser  
does not support the audio tag.**

**</audio>**

- **HTML Audio Working**
- The controls attribute adds audio controls, like play, pause, and volume.
- Text between the <audio> and </audio> tags will display in browsers that do not support the <audio> tag.
- Multiple <source> tags can link to different audio files. The browser will use the first recognized format.

# HTML Multimedia

- **HTML Video**
- A video file is a collection of images that is displayed in a sequence representing scenes in motion.
- Similar to the audio file system, video files are also encoded or decoded using the various video codecs, such as DivX and QuickTime.

# HTML Multimedia

Format	File	Description
MPEG	.mpg .mpeg	MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Used to be supported by all browsers, but it is not supported in HTML5 (See MP4)
MPEG-4 or MP4	.mp4	MP4. Developed by the Moving Pictures Expert Group. Based on QuickTime. Commonly used in newer video cameras and TV hardware. Supported by all HTML5 browsers. Recommended by YouTube
AVI	.avi	AVI (Audio Video Interleave). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers
QuickTime	.mov	QuickTime. Developed by Apple. Commonly used in video cameras and TV hardware. Plays well on Apple computers, but not in web browsers. (See MP4)

# HTML Multimedia

Format	File	Description
Flash	.swf .flv	Flash. Developed by Macromedia. Often requires an extra component (plug-in) to play in web browsers
RealVideo	.rm .ram	RealVideo. Developed by Real Media to allow video streaming with low bandwidths. It is still used for online video and Internet TV, but does not play in web browsers
WMV	.wmv	WMV (Windows Media Video). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers
WebM	.webm	WebM. Developed by the web giants, Mozilla, Opera, Adobe, and Google. Supported by HTML5
Ogg	.ogg	Theora Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5

# HTML Multimedia

- **HTML <video> tag**
- To show a [video in HTML](#), use the <video> tag. Let's look at the following example, to know how to embed video into your webpage.

```
<video width="320" height="240" controls>
```

```
  <source src="song.mp4" type="video/mp4"> <source src="song.ogg"  
  type="video/ogg">
```

Your browser does not support the video tag.

```
</video>
```

- **HTML Video Working**
- The controls attribute adds video controls, like play, pause, and volume.
- It is a good idea to always include width and height attributes. If height and width are not set, the browser does not know the size of the video. The effect will be that the page will change (or flicker) while the video loads.
- Text between the <video> and </video> tags will only display in browsers that do not support the <video> tag.
- Multiple <source> tags can link to different video files. The browser will use the first recognized format.

# HTML Multimedia

- **HTML <video> Autoplay**
- To start a video automatically use the autoplay attribute.
- Let's look at the following example, to know how to embed video and set to autoplay when web page loads.

```
<video width="320" height="240" autoplay>  
  <source src="songs.mp4" type="video/mp4">  
  <source src="songs.ogg" type="video/ogg">
```

Your browser does not support the video tag

```
</video>
```



# HTML Embed Multimedia

- Sometimes you need to add Multimedia on the web like sound, music, videos, movies, and animations.
- Multimedia comes in many different formats. It can be almost anything you can hear or see. For Examples: Pictures, music, sound, videos, records, films, animations, and many more.
- Web pages often contains multimedia tags of different types and formats.
- **History of Browser Supports**
- The first web browsers had support for text only and limited to a single font in a single color. Later came browsers with support for colors and fonts, and even support for pictures.
- The support for sounds, animations, and videos is handled differently by various browsers.
- Different types and formats are supported, and some formats requires extra helper programs (plug-ins) to work.
- Hopefully this will become history. HTML5 multimedia promises an easier future for multimedia.

# HTML Embed Multimedia

- **Multimedia Formats**
- Multimedia tags (like [audios or videos](#)) are stored in media files.
- The most common way to discover the type of a file, is to look at the file extension. When a browser sees the file extension .htm or .html, it will treat the file as an HTML file. The .xml extension indicates an XML file, and the .css extension indicates a style sheet file. Pictures are recognized by extensions like .gif, .png and .jpg
- Multimedia files also have their own formats and different extensions like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi
- HTML allows you to embed plug-ins in a Web page using the EMBED tag.
- This tag lets you embed multimedia in a Web page and play it while opening the page.
- The EMBED tag is supported by Internet Explorer as well as Netscape Navigator. It is also supported across the Windows and Mac platforms.
- The EMBED tag uses the three mandatory attributes, namely src, height, and width.

# HTML Embed Multimedia

- Following is an example of using the EMBED tag.

```
<EMBED src="Music.mp3" width=600 height=100>  
</EMBED>
```

- In the above example, we have defined the src attribute to specify the source multimedia file to be played while the Web page loads in the browser.
- We have also defined the height and width attributes to specify the height and width of the embedded multimedia component in the page, respectively.
- Following table lists various attributes of the EMBED tag in HTML.
  - Height -- Specifies the height of the embedded component
  - Hspace -- Sets the horizontal padding around the tag
  - Type-- Specifies the Multipurpose Internet Mail Extension (MIME) type for the components
  - Width Sets the width of the embedded component in the page

# HTML Embed Multimedia

- **HTML OBJECT Tag**
- HTML uses the OBJECT tag to include objects, such as images, audios, videos, Java applets, ActiveX controls, Portable Document Format (PDF), and Flash objects, in a Web page.
- The OBJECT tag allows you to specify the code that can be used to display or manipulate that data.
- An OBJECT tag can also be used inside the BODY tag. The text between the starting and the ending tags of the OBJECT tag is the alternate text for browsers that do not support this tag.
- The OBJECT tag initializes the object by passing the parameters to the object, which can be done using the PARAM tag.

# HTML Embed Multimedia

Attribute	Description
data	Specifies the URL of the object's data
form	Specifies one or more forms to which the object belongs
height	Specifies the height of the object in pixels
name	Specifies the object's name
type	Specifies the MIME type for the component
usemap	Specifies the URL
width	Sets the width of the embedded components in the page

# HTML Embed Multimedia

- In HTML, you can use the PARAM tag to define parameters or variables for an OBJECT tag. An OBJECT tag can contains multiple PARAM tags, as shown in the following code snippet.

```
<OBJECT data="movie.avi" type="video/quicktime" id="video"  
width="200" height="100">  
  <PARAM name="BorderStyle" value="1" />  
  <PARAM name="autoplay" value=true />  
</OBJECT>
```

- In the above code snippet, we have defined an OBJECT tag, which includes a video file in a Web page.
- We have also used the PARAM tags to pass the parameters for the OBJECT tag.
- Attributes of the PARAM tag.
  - Name -- Specifies the name of the parameter
  - Value -- Specifies the value of the parameter

# HTML Embed Multimedia

- **HTML FIGURE and FIGCAPTION Tags**
- The FIGURE tag is newly introduced in HTML5 and is used to group or annotate various diagrams, images, illustrations, and code snippets.
- You can define various tags, such as IMG, CODE, and PRE, inside the FIGURE tag. The FIGCAPTION tag is used inside the FIGURE tag to provide the caption of the content.
- **Example of FIGURE and FIGCAPTION Tags**

<FIGURE>

    <FIGCAPTION>Listing 1: Showing the alert box</FIGCAPTION>

    <PRE><CODE>alert('Hello World!'); </CODE></PRE>

</FIGURE>

- In the above example, we have defined the FIGURE tag, which contains the FIGCAPTION tag to display the figure caption on a Web page. In addition, we have also defined the PRE and the CODE tags to display the code in the predefined formatting.

# Unit-6

## Cascaded Style Sheet (CSS)



# Introduction to CSS

- **What is CSS ?**
  - CSS stands for Cascading Style Sheet, is a text file with **.css** extension and is commonly used to define styles and layouts of Web pages written in HTML and Extensible Hypertext Markup Language (XHTML).
- **A Brief History of CSS**
  - CSS was invented by *Hakon Wium Lie* on 10th Oct, 1994 and maintained by a group of people within World Wide Web Consortium (W3C).
- **Why CSS ?**
  - CSS simplifies the task of maintaining a Web document by separating its style information, such as font size, font color, line width, and background color etc. This separation allows you to apply the same style rules to multiple Web pages. CSS also allows you to apply a style multiple times in a single Web page.

# Introduction to CSS

- **Why to Use CSS ?**

- Suppose, you have a Web page that contains multiple tables and you want to apply some style on the table caption, table header, and table cells.
- To do this, you just need to write the code once in a CSS style sheet and apply this style sheet to all the tables of your Web page.
- This reduces the complexity and redundancy of code in the Web page and saves your time, as you do not need to write the same code again and again.

- **CSS Recommendation**

- The CSS file contains the style code for the structure, such as headings, paragraphs, and links. The styles patterns and layouts defined in a CSS file can be modified by making the required changes in the code of the CSS file.
- CSS also provides a pattern that helps in applying the style rules on specific elements. This pattern is known as a selector. Some of the most-commonly used CSS selectors are universal, type, and class.

# Introduction to CSS

- **Advantages of CSS**
  - CSS saves more time
  - CSS provides faster page loads
  - CSS is easy to maintain
  - CSS has superior styles to HTML
  - CSS provides multiple device compatibility
  - CSS provides global web standards
  - CSS provides offline browsing
  - CSS is platform independence

# Introduction to CSS

- we are going to create a CSS file, to style the above HTML document (Web page). The file contains:

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

```
h1{color:green;}
```

```
h2{color:blue;}
```

- Save the above CSS code in a file ending with **.css** extension like **filename.css**. Now to use the above CSS code to style the HTML document, we have to provide link (address) of this CSS file in the HTML document as shown here:

# Introduction to CSS

```
<!DOCTYPE HTML>
```

```
<html>
```

```
<head>
```

```
<title>CSS Tutorial</title>
```

```
<link rel="stylesheet" href="filename.css">
```

```
</head>
```

```
<body>
```

```
<h1>CSS</h1>
```

```
<p>You are learning CSS at codescracker.com</p>
```

```
<h2>CSS Tutorial</h2>
```

```
<p>This is CSS Tutorial at codescracker.com</p>  
</body> </html>
```

# Introduction to CSS

- Later, if you want to change the heading, paragraph, or background color, you have to only make changes in the CSS file. After that, your whole web page changed.
- You can also apply CSS in a HTML document at the time of creating HTML elements i.e., inline CSS.

# CSS Syntax

- Syntax can be defined as a rule that defines the structure or the order of the statements used in a programming language. It also specifies how words and symbols are put together to form statements and expressions.
- CSS also uses syntax to apply CSS rules in an [HTML](#) document.
- The CSS syntax is divided into two the following different parts.
  - selector
  - declaration
- [Selectors](#) defines an HTML element to which the CSS style is applied and the declaration contains the CSS properties as well as the value of these properties.

# CSS Syntax

- The following code fragment shows the syntax of a CSS document.

**Selector**

**{**

**1st property : value;**

**2nd property : value;**

**3rd property : value;**

**...**

**Nth property : value;**

**}**

- The preceding syntax consists of a selector and CSS properties with their values.
- The selector is the name of the element to which you want to apply the CSS properties.
- You can put CSS Style Rule Syntax as follows:

**selector { property: value }**



# CSS Syntax

```
span
{
    background:#fff;
    padding:0 3px 0 0;
    float:left;
    position:absolute;
    text-decoration:none;
}

ol
{
    list-style:lower-roman;
    margin:1.5em 0 1em 5%;
    padding:0;
    background:#fff;
    float:left;
    display:block;
    width:95%;
}
```

# CSS Syntax

```
a
{
    padding:0 0 0 3px;
    right:0;
}
```

```
Li
{
    clear:left;
    border-bottom:dashed 1px #aaa;
    height:1.05em;
    margin-top:10px;
    position:relative;
}
```

# CSS Syntax

- In the above code fragment, span, ol, a, and li are the four different selectors of a CSS file. The span selector has properties, such as [background](#), [padding](#), [float](#), [position](#), and [text-decoration](#).
- Each property is assigned with its corresponding values, for example, the background property has #fff as value while the value of the padding property is 0 3px 0 0.
- Similarly, all the selectors have their corresponding properties with each property having its corresponding value.

# CSS Syntax

- **CSS Syntax Example**
- A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly braces:

**p {color:red;text-align:center;}**

- Here **p** is the selector, **color** is the property and **red** is the value.
- To make the CSS code more readable, you can put one declaration on each line.
- Look at the following example where all the <p> elements will be center-aligned, with a red text color.

# CSS Syntax

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS Syntax Example</title>
  <style>
    p
    {
      color: red;
      text-align: center;
    }
  </style>
</head>
<body>
  <p>Hello Browser!</p>
  <p>I am center-aligned with red color.</p>
</body>
</html>
```

# CSS Selectors

- A selector is a pattern that is used to select an element to apply the CSS style rules.
- Selectors can be used as a condition or a CSS rule to determine the elements that match with the selector. As the CSS rule is divided into the following two parts.
  - selectors
  - declaration
- The declaration is a part that appears within the braces of the CSS rule followed by the selector.
- The rules defined in the declaration part are applied to the elements specified by the selector.

# CSS Selectors

- The different types of selectors are as follows.
  - The universal selector
  - The type selector
  - The class selector
  - The id selector
  - The child selector
  - The descendant selector
  - The adjacent sibling selector
  - The attribute selector
  - The query selector

# CSS Selectors

- **CSS Universal Selector**
- The universal selector selects all the elements that are present in an [HTML](#) document.
- You can use this selector to apply the same rule to all the elements of an HTML or XHTML document. The universal selector is represented by an asterisk symbol, as shown in the following code snippet.

`*{ }`

- The following code fragment shows the use of universal selector.

`* { margin:0; padding:0; color: green; }`

- In the above code fragment, the margin and the padding properties are set to 0 for all the elements in the HTML or XHTML document on which the CSS rule is applied.
- All the font have green color for all elements.



# CSS Selectors

- **CSS Type Selector**
- The type selector matches all the elements specified in a list with the given value to determine the elements to which the CSS rules are to be applied.
- The rules applied to several elements of an HTML or XHTML document are similar to the ones applied to a CSS file.
- The following code fragment shows how to use the type selector in CSS.  
**h1, h2, h3, p { font-family: sans-serif }**
- In the above code fragment, we have specified the font-family property for the different heading elements (h1, h2, h3) and for the paragraph element (p).

# CSS Selectors

- **CSS Class Selector**
- The class selector allows you to apply CSS rules to the elements that carry a class attribute whose value matches with the class attribute specified in the class selector.
- Let's consider that you have an element, H1, with a class attribute whose value is intro, as shown in the following code fragment.

```
<H1 class="intro">Header 1</H1>
```

- You can use a class selector in either of the two ways.
  - (i) By applying the CSS rule to all the elements that have the class attribute of the same value. The following code fragment shows how to apply the CSS rule.

```
.intro { font-family: sans-serif}
```

- In the above code fragment, a period is followed by a value. The value is followed by braces which embeds the CSS rule within it.
- The CSS rule is applied to all the elements having the class attribute with *intro* as its value.

# CSS Selectors

- (ii) By applying the CSS rule to the H1 elements, whose class attribute contains *intro* as its value. The following code fragment shows how to apply the CSS style on H1 elements.

**h1.intro { font-family: sans-serif}**

- In the above code fragment, an element name is followed by a value.
- The value is followed by braces, which embeds the CSS rule within it.
- The CSS rule is applied to all the H1 elements having the class attribute with *intro* as its value.

# CSS Selectors

```
<style>
```

```
  h2.green
```

```
  {
```

```
    color: green;
```

```
  }
```

```
  p.red
```

```
  {
```

```
    color: red;
```

```
  }
```

```
</style>
```

```
<h2 class="green">CSS Class Selector Tutorial</h2>
```

```
<p class="green">This is tutorial on CSS class selector.</p>
```

```
<h2 class="red">CSS Class Selector Example</h2>
```

```
<p class="red">This is example on CSS class selector.</p>
```

# CSS Selectors

- **CSS ID Selector**

- The value of the id attribute is unique within a document; therefore, the selector is applied only to the content of one element.
- The following code fragment shows the h1 element having myHeader as the value of the id attribute.

```
<H1 id="myHeader">Hello World!</H1>
```

- The following code fragment shows the id selector, which is represented by a hash symbol (#) and followed by the value of the id attribute.

```
#myHeader{ font-family: sans-serif }
```

- In the above code fragment, myHeader is the value of the id attribute.
- The CSS rule is applied to the value of the id attribute.

# CSS Selectors

```
<style>
```

```
  #green
```

```
{
```

```
  color: green;
```

```
}
```

```
</style>
```

```
<h2 id="green">CSS ID Selector Tutorial</h2>
```

# CSS Selectors

- **CSS Child Selector**
- The child selector matches the element that is an immediate child of another element.
- In the child selector, greater than symbol (>) is used as the combinator, as shown in the following code fragment.

**TD>b{ font-family: sans-serif }**

- A combinator is a symbol, such as >, <, and +, which shows the relationship between the two elements.
- In the preceding code fragment, the B element is the immediate child of the TD element.
- The CSS rule is applied to all the B elements that are immediate children of TD elements.

# CSS Selectors

```
<style>  
  body > p  
  {  
    color: green;  
  }  
</style>
```

```
</head>
```

```
<body>
```

```
<h2>CSS Child Selector</h2>
```

```
<hr/>
```

```
<p>Hello Browser!</p>
```

```
<p>This is CSS child selector</p>
```

```
<hr/>
```

```
<div>
```

```
<p>Paragraph in <div></p>
```

```
</div>
```

```
<p>Paragraph out of </div></p>
```



# CSS Selectors

- **CSS Descendant Selector**
- The descendant selector matches an element that is a descendant of another element.
- A descendant element is an element that is nested inside another element.
- In case of the descendant selector, white space is used as the combinator, as shown in the following code fragment.  
**table b { font-family: sans-serif }**
- In the above code fragment, CSS is applied to all the b elements that are nested within the table element.

# CSS Selectors

```
<style>
```

```
    ul em
```

```
    {
```

```
        color: green;
```

```
    }
```

```
</style>
```

```
<h2>CSS Descendant Selector</h2>
```

```
<hr/>
```

```
<p>Hello Browser!</p>
```

```
<p>This is CSS descendant selector.</p>
```

```
<hr/>
```

```
Here is the unorganized lists:<br/>
```

```
<ul>
```

```
    <li>list <em>item</em> 1</li>
```

```
    <li>list <em>item</em> 2</li>
```

```
    <li>list <em>item</em> 3</li>
```

```
</ul>
```

# CSS Selectors

- **CSS Adjacent Sibling Selector**
- The adjacent sibling selector selects all the elements that are adjacent siblings of a specified element. Sibling elements must have the same parent element.
- The word adjacent means side-by-side, so no other element could exist between the adjacent sibling elements. To use an adjacent sibling selector, the plus symbol (+) is used as its combinator, as shown in the following code fragment of a CSS file.

```
H2+P { font-family: sans-serif }
```

- Let's apply the preceding code fragment of a CSS file to the following [HTML](#) code fragment.

```
<H2>Heading</H2>
```

```
<P>The selector above matches this paragraph.</P>
```

```
<P>The selector above does not match this paragraph.</P>
```

- In the above code fragment, the first paragraph matches the adjacent sibling selector, H2+P, because the P element is an adjacent sibling to the H2 element.
- The second paragraph does not match with the selector. Although it is a sibling of the H2 element, it is not adjacent to the element.

# CSS Selectors

- **CSS Attribute Selector**
- The CSS attribute selector selects elements on the basis of some specific attributes or attribute values.
- The following table describes most common types of attribute selectors.
- Example:

**<style>**

```
input[type="text"]  
{  
    color: green;  
}
```

**</style>**

# CSS Selectors

Name	Syntax	Match	Example
Hyphen selector	[attribute =value]	Matches if the element has an attribute with a value followed by a hyphen	[lang =fr] { background-color:red; }
Existence selector	[attribute]	Matches if the element has a specific attribute	a[title] { color:green; }
Equality selector	[attribute=value]	Matches if the element has an attribute with a specific value	a[href=http://codescracker.com/] { font-weight:bold; }
Space selector	[attribute~=value]	Matches if the element has an attribute with space separated items that match with the value	a[title~=Web] { background-color:red; }

# CSS Selectors

- **CSS Query Selector**
- The *querySelector()* and *querySelectorAll()* methods accept CSS selectors as parameters and return the matching element node in the document tree.
- The *querySelector()* method helps in querying the entire document or a specific element of the document.
- You can use all the CSS selectors with this method as parameters.
- If multiple elements are available, CSS selectors return the first matching element; or returns null, if no element is available.
- The *querySelectorAll()* method returns all the available elements as a single static collection of elements known as *staticNodeList*.
- This collection of elements is not affected by any change made in the document tree, for instance removing or inserting a node does not affect *staticNodeList*.

# CSS Selectors

- **CSS element Selectors**
- The element selector selects elements based on the element name.
- You can select all <p> elements on a page as shown in the example given below.

```
<head>
  <title>Element Selector Example</title>
  <style>
    p
    {
      text-align: center;
      color: red;
    }
  </style>
</head>
<body>

<p>Hello Browser!</p>
<p>This is CSS element selector.</p>
```

# CSS Selectors

- **CSS Multiple Style Rules**
- You may need to define multiple style rules for a single element.
- You can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example.

```
h1  
{  
    color: #36C;  
    font-weight: normal;  
    letter-spacing: .4em;  
    margin-bottom: 1em;  
    text-transform: lowercase;  
}
```

- Here all the property and value pairs are separated by a semi colon (;). You can keep them in a single line or multiple lines.
- For better readability we keep them into separate lines.



# CSS Selectors

- **CSS Grouping Selectors**
- You can apply a style to many selectors if you like. Just separate the selectors with a comma as given in the following example.

**h1, h2, h3**

```
{  
    color: #36C;  
    font-weight: normal;  
    letter-spacing: .4em;  
    margin-bottom: 1em;  
    text-transform: lowercase;  
}
```

- This define style rule will be applicable to h1, h2 and h3 element as well.
- The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.
- You can combine various class selectors together as shown below.

**#content, #footer, #supplement**

```
{  
    position: absolute;  
    left: 510px;  
    width: 200px;  
}
```

# Inserting CSS in HTML Document

- You can use a CSS style sheet with an [HTML](#) document by learning how to link the CSS code with the HTML document.
- A CSS style sheet can be linked to an HTML document in a variety of ways, where each way has its own advantages and disadvantages.
- There are the following three ways to apply CSS style to your HTML document.
  - The internal style sheet - Embedded CSS
  - The external style sheet - External CSS
  - The in-line style - Inline CSS

# Inserting CSS in HTML Document

- **The Internal Style Sheet - Embedded CSS**
- The internal style sheet is written the HEAD element of the HTML document.
- This style is applied only to the documents in which it is defined and not referenced by any other Web document.
- The syntax of internal style sheet is written as follows.  
**<STYLE type="text/css">**  
**selector { property: value; }**  
**</STYLE>**
- In the above syntax contains the starting and the ending tags of the STYLE element.
- The STYLE element contains a type attribute with value text/css. The opening and the closing tags of the STYLE element embeds the style declaration.
- The declaration consists of a selector followed by a curly braces. The curly braces hold a property followed by a colon, which is further followed by a value, and finally that value is followed by a semicolon.

# Inserting CSS in HTML Document

- The following code fragment shows a sample CSS document written using the internal style sheet.

```
<head>  
<style type="text/css">  
p { font-family: sans-serif; color:#c00; }  
h1 { font-family: sans-serif; color:#f00; }  
</style>  
</head>
```
- In the above code fragment, the style element is placed inside the HEAD element.
- The CSS statements are written within the style element. The two selectors are p and h1.
- The properties of the p element are font-family and color. The value of font-family is sans-serif and the value of color is #c00. Similarly, the properties of h1 element are font-family and color. The value of font-family is sans-serif and the value of color is #f00.

# Inserting CSS in HTML Document

- The following code fragment shows a sample CSS document written using the internal style sheet.

```
<!DOCTYPE html>
<html>
<head>
  <title>Embedded CSS Example</title>
  <style>
    p
    {
      text-align: center;
      color: red;
    }
  </style>
</head>
<body>
  <p>Hello Browser!</p>
  <p>This is embedded CSS example.</p>
</body>
</html>
```

# Inserting CSS in HTML Document

- **Advantages and Disadvantages of internal style sheets (Embedded CSS)**
- Advantages of using the internal style sheets.
  - Affects only the page in which they are placed. If you are working on a large site and need to test styles before loading them on the site as a whole, internal style sheets can be a great tool as they allow you to test the styles on a single page.
  - Allows you to change the style of the same HTML file in which you are working.
- Internal style sheets have the following disadvantages.
  - Affects only the page to which they are applied. If you want to use same styles in several documents, you need to repeat them for every page.
  - Increases the page load time because the entire CSS file needs to be implemented first to apply CSS.

# Inserting CSS in HTML Document

- **The External Style Sheet - External CSS**
  - The syntax to create an external style sheet is same as that of creating an internal style sheet. In case of internal style sheet, the CSS file is placed inside the HTML document; whereas, in case of external style sheet, the CSS file is written outside the HTML document and the reference of the CSS file is placed in the HTML document.
  - In an external style sheet, the style sheet rules are saved into a text file with the .css extension.
  - Once you have a style sheet document, you can link it with your Web pages in the following two ways.
- (i) Linking** - Refers to the HTML LINK element, which is used to link a style sheet. This element has the following three attributes.
- rel
  - type
  - href

# Inserting CSS in HTML Document

- The rel attribute specifies what you are linking (style sheet in this case).
- The type specifies the MIME type for the browser, and the href attribute specifies the path of the .css file, as shown in the following code fragment.

```
<link rel="stylesheet" type="text/css" href="test.css" />
```

- In the above code fragment, the value of the rel attribute is set to stylesheet, value of the type attribute is set to text/css, and the value of the href attribute is set to test.css.

**(ii) Importing** - Helps you in accessing the style rules from other CSS style sheets. The @import keyword is used, followed by the Uniform Resource Identifier (URI) of the style sheet to which you want to import the style rules.



# Inserting CSS in HTML Document

- The following code fragment shows an example of using the @import rule.

```
<style type="text/css">  
    @import url("mystylesheet.css")  
    h1 { color: blue }  
</style>
```

- In the above code fragment, we have used the @import keyword followed by the URL of the style sheet, named mystylesheet.css.
- In addition to the @import rule, the @media rule of CSS helps you in applying the styles to the media device depending on the type of the device a page is displaying.
- Some of the media devices supported by the CSS are computer screens, printers, televisions, handhelds, speech synthesizers, and projectors.
- Please note that all the media types are not supported by all the Web browsers, for example, the Internet Explorer 4.5 and Netscape Navigator 4.7 browsers or their later versions, support the computer screen, printer, and projector media types.

# Inserting CSS in HTML Document

- The following code fragment shows an example of using @media rule.

```
<style type="text/css">  
    @media screen  
    {  
        body { font-size: 13px; }  
    }  
</style>
```

- In the preceding code fragment, we have defined the @*media* rule for the screen media type.

# Inserting CSS in HTML Document

- **Example**
- An external style sheet is a separate text file with .css extension. You define all the Style rules within this text file and then you can include this file in any HTML document using <link> element.
- Open your text editor and type the following code into it.

```
p
{
    text-align:center;
    color:red;
}
```

- Save the file as filename.css and close it.

# Inserting CSS in HTML Document

Now this time type the below html code into your text editor like notepad for windows user.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>External CSS Example</title>
```

```
    <link rel="stylesheet" href="full path of css file">
```

```
</head>
```

```
<body>
```

```
<p>Hello Browser</p>
```

```
<p>This is external CSS example.</p>
```

```
</body>
```

```
</html>
```

- Open your file named **filename.css** into your web browser like google chrome, and copy the url (full path) of this file.
- In the place of **full path of css file**, place the full path (copied from browser) of your css file (filename.css).

# Inserting CSS in HTML Document

- **Advantages and Disadvantages of External CSS**
- Advantages of using the external style sheets.
  - Allows you to control the look and feel of several documents in one go and do not need to define a specific style for every element.
  - Allows you to easily group your styles in a more efficient way
- External style sheets have the following disadvantages.
  - Increases the download time as the entire CSS file has to be downloaded to apply the style to the HTML document. When the styles are less in number, applying external style sheet can make the document complicated.
  - Displays the Web page only after the entire style sheets is loaded.

# Inserting CSS in HTML Document

- **The Inline Style - Inline CSS**
- The inline style properties are written in a single line separated by semicolons.
- These properties are placed inside the style attribute of the HTML element, on which you want to apply the style.

```
<p style="background:#ccc; color:#fff;  
border:1px solid black;">
```

- In the above code fragment, the p element is styled.

# Inserting CSS in HTML Document

- **Advantages and Disadvantages of Inline CSS**
- Advantages of applying inline style in an HTML document.
  - Provides highest precedence over internal and external style sheets. Therefore, if you want some styles to be compulsorily applied, use the inline style.
  - Provides an easy and quick approach to add a style sheet in a Web page. You do not need to create a whole new document (as with external style sheets) or edit a new element in the head of your document (as with internal style sheets) to add an inline style.

# Inserting CSS in HTML Document

- Inline styles have the following disadvantages.
  - Makes a document difficult to maintain and increases the download time. Inline styles must be applied to every element on which you want to apply a style.
  - Therefore, if you want all your headings to have the font-family Arial, you have to add an inline style to each heading element in your document.
  - Does not allow to style pseudo-elements, which are used to add special effects to the selectors.
  - For example, in external and internal style sheets, you can provide different styles or colors to differentiate between active, visited, or non-visited links.



# Inserting CSS in HTML Document

- **CSS Inheritance**

- In CSS, a property that is applied to an element is also inherited by the child elements of that element.
- For example, if the font-family property is declared for the BODY element, it is automatically applied to all the elements present inside the BODY element.
- This inheritance saves your time in writing the repeated code for every single element that constitutes the Web page. The following code fragment shows inheritance in CSS.

```
<div style="font-family:serif; border:1px solid red; padding:10px;>
```

**This text will be in a serif font.**

```
<p>
```

**This text is also in a serif font, because font is inherited by default.**

**But the border and padding properties are not inherited from the parent div.**

```
</p>
```

```
</div>
```

- In the above code fragment, font-family is automatically inherited by the p element from the parent div element. In this case, border and padding properties are not inherited from the parent element.

# Inserting CSS in HTML Document

- Following table describes CSS properties that are inherited automatically.

Properties	Description
border-collapse	Represents the border display
border-spacing	Represents the thickness of the border
caption-side	Represents the place of the table caption
color	Represents the color of a text
cursor	Represents the cursor to be displayed
direction	Represents the direction of a text
empty-cells	Specifies whether to display the border and background on empty cells
font	Specifies all the font properties
font-family	Represents the font family of a text
font-stretch	Represents text as stretched or condensed

# Inserting CSS in HTML Document

- Following table describes CSS properties that are inherited automatically.

Properties	Description
font-size	Represents the size of a text
font-size-adjust	Represents the size of the text on the basis of an aspect value
font-style	Represents the style of a text
font-variant	Represents the fonts variant, such as small caps
font-weight	Represents the font as bold
letter-spacing	Represents the space between the characters in a text
line-height	Represents the height of a line
list-style	Represents the style of the list
list-style-image	Represents an image as a list marker

# Inserting CSS in HTML Document

- Following table describes CSS properties that are inherited automatically.

Properties	Description
list-style-type	Represents the type of a list marker
quotes	Represents the quotation marks in a text
text-align	Represents the horizontal alignment of a text
text-indent	Represents the indentation of a text
text-transform	Represents the transformation of text, such as uppercase, capitalize, or lowercase
white-space	Handles the spaces in an element
word-spacing	Represents the spacing between the words in a text

# CSS Backgrounds

- A Web developer always tries to create an impressive website. This is because, unless the website looks appealing, a user is not attracted towards the content of the website.
- To create attractive websites, Web developers use background [images](#) and [color](#) gradients.
- Prior to Hypertext Markup Language (HTML) 4, the background of a Web page was set by using the bgcolor attribute of the BODY element.
- However, from HTML4 onwards, the background is controlled by using Cascading Style Sheet ([CSS](#)).
- CSS not only helps in setting the background of an entire Web page, but also helps in adding different colors to the various elements of the Web page.

# CSS Backgrounds

- For example, you can set the green color for the entire Web page and display the content of a paragraph in the orange base color.
- Therefore, by using different elements with different background colors, you can easily highlight various areas of a Web page, such as [paragraphs](#) and [headings](#).
- The new version of [CSS](#), [CSS3](#), allows you to set the color of a background on the basis of various color specifications, such as Red Green Blue (RGB) and Hue Saturation Lightness (HSL).
- It also enables you to set the transparency level of a color that has to be displayed in the background of an element.
- Using CSS3, you can also use an image as the background of a website and specify how many times the image should be repeated.

# CSS Backgrounds

- **Background of a Web Page**
- Background of a Web page is the area on which the content of the Web page, such as [text](#) , [tables](#), [border](#), and [images](#), is displayed.
- A Web page should have a background that expresses the motto of the Web page.
- It is important that background should have lighter color than the text written on it, to provide the better readability.
- Similarly, while creating a Web page for entertainment, the background should preferably have vibrant color.

# CSS Backgrounds

- CSS provides you various properties to set the background of a Web page. These properties are as follows.

- [background-color](#)
- [background-image](#)
- [background-repeat](#)
- [background-attachment](#)
- [background-position](#)
- [background-clip](#)
- [background-origin](#)
- [background-size](#)
- [background-quantity](#)
- [background-spacing](#)
- [background](#)



# CSS Backgrounds

- **The background-color Property**
- The background-color property is used to set the color of the background area on which an element is displayed. It can be applied to almost any element.
- The background-color property takes any of the following three values:
  - [Color Name](#)
  - [Hexadecimal Equivalence of the Color](#)
  - [RGB Color Value](#)
- An example of using the background-color property is given as follows.  
**h1 { background-color: #FFFFFF; }**
- Apart from the preceding three values, the background-color property can also take two more values. They are transparent and inherit.

# CSS Backgrounds

- When the value is set to transparent, the background color of the element becomes transparent to its parent element so that the background color of the parent element is visible through it.
- It is important to note that the element does not take or inherit the color of the parent element; it just becomes transparent so that the background color of parent element is visible through it.
- Transparent is a default value, that is, if no value is set for the background-color property; it takes transparent as its default value.
- When you set the value of the background-color property to inherit, it takes or inherits the color of its immediate parent elements.

# CSS Backgrounds

- **The background-image Property**
- The background-image property is used to set an image in the background of an element.
- It is similar to the background attribute of the body element of HTML 4.0.
- In case of the background attribute, the image could only be set for the body element; whereas, using the background-image property, you can virtually set background image for all elements.
- The background-image property is specified using two values either url, to specify the image, or more, when no image is used.

# CSS Backgrounds

- Some examples of using the background-image property are given in the following code fragment.

```
body { background-image: url('picture.jpg'); }  
p { background-image: none; }
```

- In the above code fragment, an image, picture.jpg is specified for the BODY element and no image is used for the p element of HTML.
- The picture.jpg image is located in the local server. If the image is located in a different server, you have to provide the complete path of the image in the Uniform Resource Locator (URL), as shown in the following code fragment.

```
body { background-image:  
    url('http://codescracker.com/pictures/pic1.jpg'); }
```

# CSS Backgrounds

- To provide local url (image located in your pc):

```
body { background-image:  
    url('file:///D:/Images/wallpaper/Waterfall.jpg'  
); }
```

- When you set a background image for an element, you should also specify a background color that will be used when the image is unavailable.
- If the image is available, it is placed on top of the background color, and in fact the color is visible in the transparent parts of the image.

# CSS Backgrounds

- Here are some other properties used to decorate background image:
  - [background image repeat](#)
  - [background image attachment](#)
  - [background image position](#)
  - [background image origin](#)
  - [background image size](#)
  - [background image quantity](#)
  - [background image spacing](#)
  - [background image clip](#)

# Background image property

- **The background-repeat Property**
- The background-repeat property allows you to tile the [background images](#) along x-axis and y-axis of an element.
- **Note** - The background-repeat property is used along with the [background-image property](#) only.
- The background-repeat property can take either of the following values:
  - **repeat-x** - Tiles an image horizontally
  - **repeat-y** - Tiles an image vertically
  - **repeat** - Tiles an image both horizontally and vertically
  - **no-repeat** - Does not tile an image

# Background image property

- **Repeat Background Image Horizontally or Vertically**
- By default, the background-image property repeats an image both horizontally and vertically.
- To make background image to repeat horizontally, then below is the syntax:

**body**

```
{  
    background-image: url('bg_image.jpg');  
    background-repeat: repeat-x;  
}
```

**To make background image to repeat vertically, then below is the syntax:**

**body**

```
{  
    background-image: url('bg_image.jpg');  
    background-repeat: repeat-y;  
}
```



# Background image property

- **Background Image no-repeat**
- Showing the image only once is specified by the background-repeat property:

**body**

```
{  
  background-image: url('bg_cloud_once.png');  
  background-repeat: no-repeat;  
}
```

# Background image property

- **The background-attachment Property**
- The background-attachment property is used to fix or scroll the [background image](#) along with the text and other content displayed on it.
- The background-attachment property takes either of the two values describes in the table given below:
  - Fixed -- If the value is set to fixed, the background image does not move with the text when the Web page is scrolled.
  - Scroll -- If the value is set to scroll, the background image scrolls along with the text written over it. The default value of the background-attachment property is scroll.
- The following code fragment shows how to use the background-attachment property:

```
body {  
    background-image: url('myimage.gif');  
    background-attachment: scroll;  
}
```

# Background image property

- **The background-position Property**
- The background-position property sets the position of a [background image](#) on a Web page. This property is used along with the [background-image property](#).
- You can set the position of an image by performing any of the following tasks:
  - (i) Representing position in pixels, as shown in the following code fragment:

```
body
{
    background-image: url('pic.jpg');
    background-position: 200px 200px;
}
```

- (ii) Representing position in percentage, as shown in the following code fragment:

```
body
{
    background-image: url('pic.jpg');
    background-position: 50% 50%;
}
```

# Background image property

(iii) Representing position by words, such as left, right, and center, for x-axis, and top, down, and center for y-axis, as shown in the following code fragment:

```
body  
{  
    background-image: url('pic.jpg');  
    background-position: left bottom;  
}
```

In the preceding code fragment, the position of the background image is set using the left bottom value.

# Background image property

- **The background-origin Property**
- The background-origin property is used to determine the starting position of the [background image](#) in a box like shape.
- The background-origin property allows you to specify the starting point of the position of your background image.
- There are three values that can be specified with the background-origin property. These values are described in the table given below:
  - padding-box -- Specifies the position of the background image in relation to the outer edge of the padding or inner edge of the border. This is the default value of the background-origin property.
  - border-box -- Specifies the position of the background image in relation to the outer edge of the border.
  - content-box -- Specifies the position of the background image in relation to the outer edge of the content or inner edge of the padding.

# Background image property

- **CSS Background Size**
- **The background-size Property**
- The background-size property is used to specify the size of the image that is used as a background for an element.
- You can specify the size by using any of the following values:
  - the auto keyword
  - the length parameter
  - the percentage parameter

## **CSS auto Keyword**

- Sets the image to its original size. Some examples of using the auto keywords to specify the size are given as follows:

**background-size: auto;**

**background-size: auto auto;**

# Background image property

## CSS Length Parameter

- Sets an image of specified size in terms of height and width. Some examples of using the background-size property to specify length parameter of an image are given as follows :

**background-size: 15px;**

**background-size: 50px 100px;**

## CSS Percentage Parameter

- Sets the size of an image with respect to the specified height and width of the area in which the image has to be displayed.
- Some examples of using the percentage parameter to specify the size are given as follows :

**background-size: 50%;**

**background-size: 50% auto;**

# Background image property

- **The background-quantity Property**
- The background-quantity property is used to specify the number of times to repeat an image. It takes following values:
  - the infinite keyword
  - the integer parameter

## CSS infinite Keyword

- Repeats an image infinitely. The following code fragment shows the use of the infinite keyword :

**background-quantity: infinite;**

## CSS Integer Parameter

- Repeats an image the specified number of times. The following code fragment shows the use of the integer parameter :

**background-quantity: 5;**



# Background image property

- **The background-spacing Property**
- The background-spacing property is used to specify the distance between the images that are repeated in the background of an element.
- It takes the value to specify the horizontal and vertical space, related to the coordinates specified by the background-position property.
- The following code fragment shows the use of the background-spacing property:

```
<body style="background-image: url('picture.gif');  
background-repeat: repeat-x;  
background-spacing: 20px 50px;" >
```

- It should be noted that if you specify a single value, this value gives the horizontal and vertical spacing both. However, if you give two values, the first value gives the horizontal spacing and the second gives the vertical spacing.

# Background image property

- **The background Property**
  - The background property is the shortcut of specifying several background properties at the same place in a style sheet.
  - It can be used to specify the values for the [background-color](#), [background-image](#), [background-repeat](#), [background-attachment](#), [background-position](#), and [background-size](#) properties.
  - Some examples of using the background property are given as follows :

```
body { background: green }                /* Example 1 */  
p { background: url("people.png") blue 50% repeat fixed } /* Example 2 */  
E { background: orange url("metal.jpg")/100% auto          /* Example 3 */
```

- Here is the explanation of the above CSS code fragment:
  - **Example 1** - Specifies that a color is defined for background
  - **Example 2** - Specifies that all properties are specified individually, which includes an image, color, image size, image repetition, and its attachment
  - **Example 3** - Specifies that both a background color and a background image are set. In addition, the image is stretched to the full width of the element

# CSS Fonts

- A font can be defined as a set of characters, including letters, numbers, punctuation marks, and symbols, of a certain size and style.
- The height of characters in a font is measured in points, where each point is approximately  $1/72$  inch.
- The width is measured in pitch, which refers to how many characters can fit in an inch.
- The pitch of a character also defines its boldness and thickness.
- Apart from fonts, you can also apply various text styles, such as italic, bold, and underline, on a document using a Cascading Style Sheet (CSS).
- These styles can be applied by setting various text properties of a document, such as text-decoration, text-shadow, text-indent, text-stroke, and text-wrap.
- Application of these properties helps the reader to easily understand and identify important terms and information in a document.

# CSS Fonts

- Font represents the style and size of the text that is displayed in a Web browser.
- Apart from imparting a visual appealing to the content, fonts are also used to help users discriminate between different types of information.
- For example, you can easily distinguish a main level heading from its subheadings by applying different font sizes and styles to different levels of headings.
- The fonts are categorized under different font families describes in the given table.

# CSS Fonts

Font Family	Description
Serif	Refers to the font family in which the width of characters is proportional, and the characters are displayed with serifs (serifs are semi-structural details on the ends of some of the strokes). Proportional width means each letter in the text has different width according to its height. Some of the fonts that are included in this family are Times New Roman, Georgia, Palatino Linotype, Sylfaen, Garamond, Book Antiqua, Bookman Old Style, Perpetua, Rockwell, and Cambria.
Sans-serif	Refers to the font family in which the width of characters is proportional, but does not have serifs. Some of the fonts that are included in this family are Microsoft Sans Serif, Verdana, Tahoma, Arial, Trebuchet MS, Arial Black, Lucida Sans Unicode, Franklin Gothic Medium, Arial Narrow, and Century Gothic.
Cursive	Refers to the font family in which characters appear as human hand writing. Some of the fonts that are included in this family are Comic Sans MS, Monotype Corsiva, Bradley Hand ITC, French Script MT, Tempus Sans ITC, Mistral, Kristen ITC, Edwardian Script ITC, Maiandra GD, Blackadder ITC, and Vivaldi.

# CSS Fonts

Font Family	Description
Fantasy	Refers to the font family in which characters cannot be characterized under a single rule. Some of the fonts included in this family are Impact, Haettenschweiler, Papyrus, Copperplate Gothic Light, Copperplate Gothic Bold, Curlz MT, Felix Titling, Rockwell Extra Bold, Engravers MT, Juice ITC, Jokerman, Imprint MT Shadow, Goudy Stout, Castellar, Agency FB, Perpetua Titling MT, and Cooper Black.
Monospace	Refers to the font family in which characters resemble the text written with a type writer. The characters are not proportional, which means, the width of each character is same. Some of the fonts of this family are Courier New, Lucida Console, OCR A Extended, Consolas, Lucida Sans Typewriter, Bitstream Vera Sans Mono, Andale Mono IPA, Andale Mono, OCRB, Monaco, and Terminal.

# CSS Fonts

- **CSS Font Properties**
- In Hypertext Markup Language ([HTML](#)), you can change the size, style, and family of fonts using various CSS properties.
- CSS provides the following properties to perform different tasks that can be grouped according to their functionalities related to fonts and text.
  - font-family
  - font-size-adjust
  - font-stretch
  - font-style
  - font-variant
  - font-weight
  - font

# CSS Fonts

- **CSS font-family**

- The font-family property used to specify the name of a font family to apply the specified font style on the text.
- Please note that if you have specified a font family and that font family is not installed on your computer, then the Web browser displays the text in another font.
- You can specify more than one font family in the font-family property, so that, if one font is not installed on your computer then the Web browser can display the second specified font.
- The following code fragment shows an example of using the font-family property.

```
body { font-family: sans-serif; }
```

```
h1 { font-family: sans-serif, monospace; }
```



# CSS Fonts

- In the preceding code fragment, we have specified the font-family property of the BODY element as sans-serif.
- We have also specified the font-family property of the H1 element as sans-serif and monospace. In case, the sans-serif font family is not installed, the Web browser will display the heading in the monospace font.

# CSS Fonts

- **CSS font-size**
- The font-size property is used to change the size of the text.
- The value of the font-size property is often specified in pixels, as shown in the following code fragment.  
**p { font-size: 12px; }**
- The font size can be specified in the following three different ways.
  - Using absolute values
  - Using relative values
  - Using a percentage value
- Let's now discuss each of these ways in details.

# CSS Fonts

- **CSS Absolute Values**
- Absolute value refers to the absolute size of the font. Absolute sizes are predefined fixed sizes that cannot be changed by a user. The following font sizes are categorized under the absolute values.
  - xx-small
  - x-small
  - small
  - medium
  - large
  - x-large
  - xx-large
- The xx-small is the smallest text size, xx-large is the largest text size, and rest comes in between.
- The following code fragment shows an example of the font-size property using the absolute values.  
**p { font-size: x-small; }**

# CSS Fonts

- **CSS Relative Values**
- Relative values refer to the values that are not fixed value and are calculated on the basis of the current font values.
- Consider a case in which an element has a child element and the font size of the child element is set with a relative value, which you can increase or decrease by specifying as smaller or larger.
- The smaller value displays the child element with font size one unit smaller than the font size of the parent element and the larger value displays the child element with font size one unit larger than the font of the parent element.
- The following code fragment is an example of the font-size property using a relative value.  
**p { font-size: larger; }**
- In the preceding code fragment, the font of the child element, p, is displayed one size larger than its parent element.

# CSS Fonts

- **CSS Percentage Value**
- You can also increase or decrease the font size of the text by specifying a percentage value in the font-size property.
- The percentage value is relative to the size of the parent element, which is the base value.
- For instance, if you set the percentage value for the font size to 50%, then the font size increases 50% to its current size.
- The following code fragment is an example of the font-size property using the percentage values.  
**p { font-size: 20%; }**
- In the preceding code fragment, the font size of the text will be increased by 20%.
- In case, you want to decrease the font size, specify the percentage in negative, such as -20%.

# CSS Fonts

- **CSS font-size-adjust**
- The font-size-adjust property is used to change the aspect value of the text on a Web page.
- The aspect value is the ratio between the font height of a lowercase letter and the actual height of the font.
- This ratio is also known as the x-height.
- For example, the aspect value of the Verdana font is 0.58, which means that when the font size of 100px, the height of a character written in lowercase of Verdana font is 58 pixels.
- In case of Times New Roman font, when the font size is 100px, its x-height is 46 pixels. This means that the aspect value of the Times New Roman font is 0.46.
- You can increase or decrease the height of the font by modifying its aspect value.
- The following code fragment is an example of using the font-size-adjust property.  
**p { font-size-adjust: 0.5; }**
- In the preceding code fragment, the font size is adjusted to 0.5 x-height.

# CSS Fonts

- **CSS font-stretch**
- The font-stretch property is used to change the width of a font. Using this property, you can condense or expand the width of the font by specifying the following values.
  - ultra-condensed
  - extra-condensed
  - condensed
  - semi-condensed
  - normal
  - semi-expanded
  - expanded
  - extra-expanded
  - ultra-expanded
- The following code fragment is an example of using the font-stretch property.  
**p { font-stretch: condensed; }**
- In the preceding code fragment, the width of the font is decreased or condensed than its original size.

# CSS Fonts

- **CSS font-style**
- The font-style property is used to specify the style of the font.
- The possible values of the font-style property are normal italic, and oblique.
- The following code fragment is an example of using the font-style property.  

```
p { font-style: italic; }
```
- In the preceding code fragment, the style of the font is set to italic, which means that the text will appear in italics.



# CSS Fonts

- **CSS font-variant**
- The font-variant property is used to display a font as normal or in small-caps.
- When you set the font-variant property of a font to small-caps, the font written in lower case displays in the smaller version of the uppercase letter.
- The following code fragment is an example of the font-variant property.

```
p { font-variant: small-caps; }
```

- In the preceding code fragment, we have specified the font-variant property of the p element to small-caps.
- The text written in the p element displays in the smaller version of the uppercase letters.

# CSS Fonts

- **CSS font-weight**
- The font-weight property is used to specify the weight of the font, such as the font boldness or thickness.
- Font weight is a term used to signify the extent of boldness or thickness assigned to a character, when a particular font is applied to it.
- For example, the font weight of a character written in the Cooper Black font, A, is more than the same letter, A written in the Arial font.
- Here are the list of possible values for the font-weight property.
  - lighter
  - normal
  - bold
  - bolder
  - numbers from 100 to 900

# CSS Fonts

- The following table describes details of the numbers that are used with the font-weight property.

Values	Description
100	Represents the thin font
200	Represents the extra light (ultra light) font
300	Represents the light font
400	Represents the normal font
500	Represents the medium font
600	Represents the semi bold font
700	Represents the bold font
800	Represents the extra bold (ultra bold) font
900	Represents the black (heavy) font

# CSS Fonts

- **CSS font Property**
- Instead of defining all the properties, such as font-style, font-weight, and font-style, separately, you can specify the values of all these properties in the font property.
- The following is an example of using the font property.

```
p { font: bold italic 30px verdana; }
```

- In the preceding code fragment, we have specified the values of the font-weight, font-style, font-size, and font-family properties as bold, italics, 30px, and Verdana, respectively, as the value of the font property.

# CSS Fonts

- **CSS Web Font**
- Web font is a feature that allows you to write text in fonts other than those existing in your system.
- The Web font feature is introduced in the latest version of CSS, that is CSS3.
- This feature eliminates the restriction of using the limited number of fonts that are installed on your computer.
- You can also use the fonts that are available online by specifying their Uniform Resource Locator (URL) in the style sheet.
- In CSS3, you can define the name of the desired font by using the *@font-face* keyword in the style sheet.

# CSS Fonts

- The following is the syntax for defining the Web font in your style sheet.

```
@font-face { font-family: <name> src:  
<source> }
```

- In the preceding syntax, you need to define the name of the font that you want to use in your Web page in the font-family attribute, and the address of the URL of the font in the *src* attribute.

# CSS Control Display of Element

- CSS allows you to control the display of an HTML element by using the *display* and the *visibility* property. The property used to control the display of an element using CSS given below.
  - Display -- The display property specifies how to display an element.
  - Visibility -- The visibility property specifies whether the element should be visible or hidden.
- The following code snippet shows the use of display property:  
**h1 { display: none; }**
- The following code snippet shows the use of visibility property:  
**p { visibility: visible; }**
- CSS also allows you to display the content of an HTML element as inline or [blocks](#).
- When you display the content of an element as block, then it takes the full width of a Web page and is preceded and followed by a [line break](#).

# CSS Control Display of Element

Value	Description
none	Does not display an element.
block	Generates a block box, which means a line break before and after an element.
inline	Generates an inline box, which means no line break before and after an element. This is the default value.
inline-block	Generates a block box, laid out as an inline box.
inline-table	Generates an inline table element without any line break before and after the element.
list-item	Generates an element as an item of a list element.
table	Generates a table element with a line break before and after the element.



# CSS Control Display of Element

Value	Description
table-caption	Displays an element as a table caption.
table-cell	Displays an element as a table cell.
table-column	Displays an element as a table column.
table-column-group	Displays an element as a group of table columns.
table-footer-group	Displays an element as a group of table footer.
table-header-group	Displays an element as a group of table headers.
table-row	Displays an element as a table row.
table-row-group	Displays as element as a group of rows.
inherit	Inherits the value of the display property from the parent element.

# CSS Control Display of Element

- **CSS visibility Property**

- The visibility property specifies whether an element is visible on a Web page or not. It takes four values, visible, hidden, collapse, and inherit.

Value	Description
visible	Makes an element visible
hidden	Hides an element
collapse	Represent a value that is applied only to a table element and removes its rows and columns
inherit	Inherits the value of visibility property from the parent element

- **CSS visibility Property Values**

- You can hide an element either by setting its display property as none or by setting its visibility property as hidden. The difference is that the visibility property hides the element, but it preserves the space required by the element on the document. However, the display property does not preserve any space.

# CSS Control Display of Element

- **CSS Position**
- CSS provides a property, position, which controls the position of elements with respect to the normal flow of the content on a Web page.
- You can apply the position property on any HTML element, such as P, DIV, TABLE, FORM, and TEXTAREA.
- The syntax to use the position property is given as follows:

**position: [value];**

# CSS Control Display of Element

- The position property takes values described in the table given below:

Value	Description
relative	Specifies relative position of an element with respect to the normal flow the content
absolute	Specifies the position of a block element with respect to the normal flow of the content
fixed	Fixes the position of an element with respect to the normal flow of the content
static	Specifies the normal position of an element
inherit	Specifies that an element uses the same settings of position as of its parent element

# CSS Control Display of Element

- The following code fragment shows how to use the position property:

```
p { position: fixed; }
```

- CSS also provides some properties that specify the offset position of an element with respect to the normal flow of the content of a Web page.
- These properties are described in the following table:

Property	Description
top	Offsets an element in the top direction of a Web page
bottom	Offsets an element in the bottom direction of a Web page
left	Offsets an element in the left direction of a Web page
right	Offsets an element in the right direction of a Web page

# CSS Control Display of Element

- The following code fragment shows how to set the offset positions for an element:  
**p { position: fixed; top: 10px; right: 5px; }**
- In the preceding code fragment, the offset position of the P element is top-right and the position is fixed with respect to the normal flow of the Web page content.
- Let's explore more about the positioning of an element in the following sections:
  - Fixed positioning
  - Static positioning
  - Absolute positioning
  - Relative positioning

# CSS Control Display of Element

- **CSS Fixed Positioning**
- As discussed, the fixed value of the position property is used to set the fixed position for an element.
- It keeps an element fixed with respect to the remaining content of the Web page.
- An element assigned with fixed positioning does not change its position when the Web page is scrolled.
- This type of positioning is often used when a fixed header or footer needs to be specified in each page of a website irrespective of the scrolling of the page. This is also used to create a frame whose header and side bar are kept constant and the remaining content keeps changing on scrolling.
- The following code fragment shows how to fix the position of the P element:

```
<style type="text/css">
```

```
p { position: fixed; bottom: 10px; right: 5px; }
```

```
</style>
```

# CSS Control Display of Element

- **CSS Static Positioning**
- HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the web page.
- Static positioned elements are not affected by the top, bottom, left, and right properties.



# CSS Control Display of Element

- **CSS Absolute Positioning**
- The absolute value of the position property is used to set the absolute position of an element with respect to the content of its parent element.
- An absolutely positioned element and its parent element are placed independently.
- The layout of each absolutely positioned element is independent of other elements.
- By default, an absolutely positioned element is placed just above (in z-space coordinate) its parent element.

# CSS Control Display of Element

- The following code fragment shows how to set absolute position using CSS position property:

```
<style type="text/css">
```

```
.absolute
```

```
{ position: absolute; left: 100px; top: 150px; }
```

```
.absolute2
```

```
{ position: absolute; left: 10px; top: 50px; }
```

```
</style>
```

- **Note** - Using CSS, you can also overlap the content of two elements. To overlap two elements, you should specify the position of one element as absolute.
- CSS provides the z-index property to overlap the content of two elements. This property specifies a stack order, that positions one element over another.
- The following code fragments overlaps the content of the P element over the content of the parent element:

```
<style type="text/css">
```

```
p { position: absolute; left: 0px; top: 0px; z-index: -1; } </style>
```

# CSS Control Display of Element

- **CSS Relative Positioning**
- The relative value of the position property is used to set the relative position of an element with respect to the content of its parent element.
- This type of positioning allows an element to retain its natural formatting.
- Relative positioning provides the following advantages :
  - Adjusts the z-order coordinate relative to the other elements that may occupy the same area.
  - Positions the child elements that are underlying or overlaying the content of the parent element.
  - Moves or hides an element dynamically using a scripting language.
- Relative positioning defines a new coordinate system for child elements, with the original located at the position where the first child element is rendered.

# CSS Control Display of Element

- The following code fragment shows how to set relative position using the CSS position property:

```
<style type="text/css">
```

```
.pos_left { position: relative; left: -20px; }
```

```
.pos_right { position: relative; left:20px; }
```

```
</style>
```

# CSS Control Display of Element

- **Floating an element using CSS**
- Sometimes you need to use the text wrap feature in a document to properly arrange the position of images with respect to their description.
- For instance, you need to set an image as floated to right side and rest of the content in the left side.
- CSS allows you to implement the text wrap feature in a Web page by using the float property.
- Float property makes an HTML element as a floated element and defines the side where other elements are displayed.
- The syntax to use the float property is given as follows :

**float: [value];**

# CSS Control Display of Element

- The float property supports values described in the table given here:

Value	Description
left	Floats an element to the left with respect to the content.
right	Floats an element to the right with respect to the content.
none	Does not float an element.
inherit	Floats an element using the same float settings as specified for its parent element.

# CSS Control Display of Element

- The following code fragment floats an image in the right side :

**<p>**

** The text is in left direction.**

**</p>**

- You can disable the effect of the float property by using the clear property. This means that you can turn off the effect of the float property by using the clear property.
- The syntax to use the clear property is given as follows:

**clear: [value];**

# CSS Control Display of Element

- The clear property supports values described in the following table:

Value	Description
left	Disables the effect of a left floated element.
right	Disables the effect of a right floated element.
both	Disables the effect of both left and right floated.
none	Does not disable the effect of a floated element.
inherit	Uses the same clear settings as specified for its parent element.

- An example of using the clear property is given as follows:

```
img { clear: both; }
```

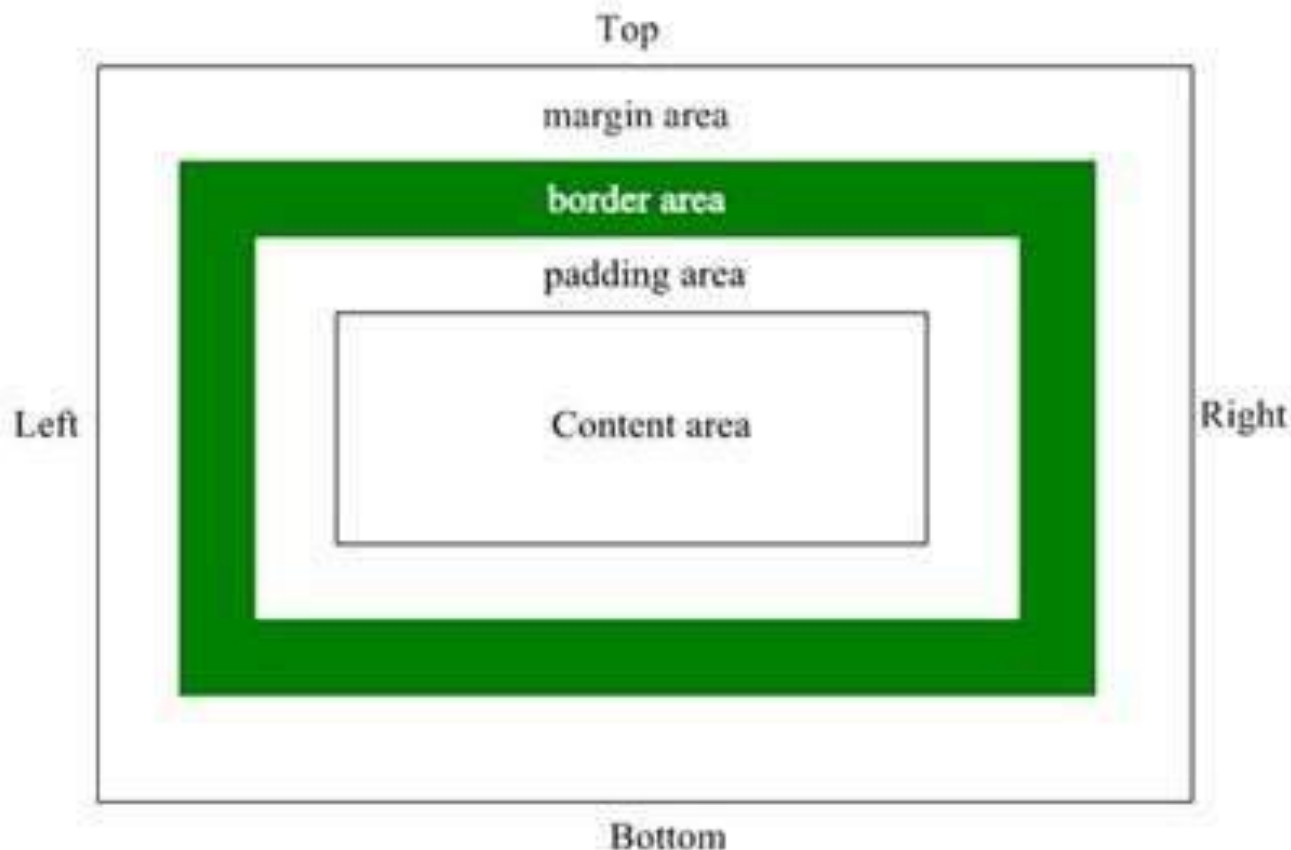


# CSS Box Model

- CSS treats an HTML document as a hierarchical tree of elements, where each element can have zero or more child elements arranged in an ordered way.
- The topmost element of this tree is called as the root element or the parent element.
- These elements display their content at a specific position, which is defined by using CSS properties.
- CSS converts the data of HTML elements in the form rectangular boxes, by using a layout model, called the box model, to set the design and [layout of HTML documents](#).
- This means that the box model determines how HTML`elements are displayed as boxes.
- The box model allows placing a [border](#) around the elements and also provides space between elements.

# CSS Box Model

- **Areas of Box Model**
- Following figure shows the various areas and edges of a box:



# CSS Box Model

- The table given below describes the area of the box model as shown in the above figure:

Area	Description
Content area	Displays the content of a document, such as <a href="#">text</a> and <a href="#">images</a> . This is bounded by a rectangle, which is called as the content edge. Note that the content area always appears inside the <a href="#">padding</a> area.
Padding area	Specifies the area around the content area. This is bounded by the padding edge. Outside the padding is the border area and the outside boundary of that area is the border edge. Finally, outside the border is the <a href="#">margin</a> area whose outer edge is called the margin edge.
Border area	Specifies the area around the padding area. This is bounded by the border edge.
Margin area	Specifies the area around the border area. This is bounded by the margin edge.

# CSS Box Model

- A box model includes the following types of boxes (describes in the table given here):
  - block-level -- box Represents a box to show a paragraph
  - line box -- Represents a box to show a line of text
  - inline-level – box Represents a box to show the words of a line
- **Note** - A block-level box can contain either other block-level boxes, such as a section containing [paragraphs](#) or a [table](#) containing rows, or line boxes, such as a paragraph containing lines of text.
- **Note** - A line box contains inline-level boxes, such as a line with words in different styles.
- **Note** - An inline-level box can contain either text with other inline-level boxes or a block-level box, such as a small table that is in inline format.

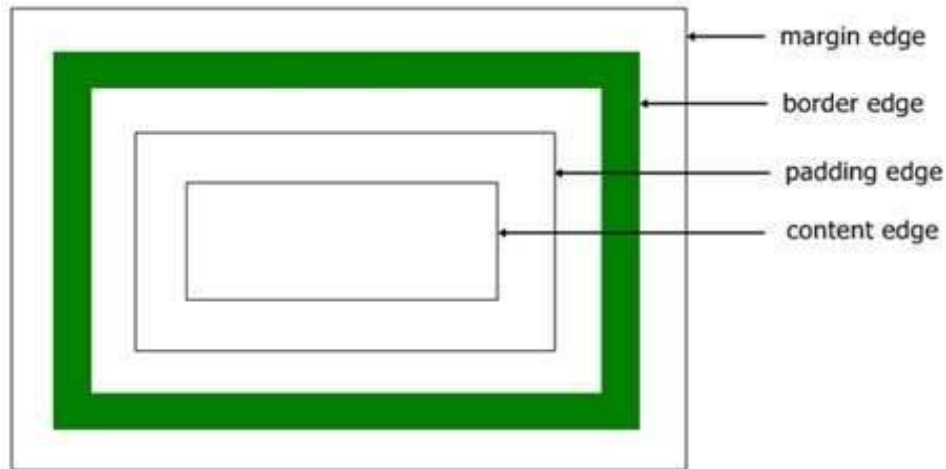
# CSS Box Model

- Let's explore various other aspects of the box model in the following sections:
  - [Box dimensions](#)
  - [The padding properties](#)
  - [The border properties](#)
  - [The margin properties](#)
  - [The width and height properties](#)
  - [Floating boxes](#)
  - [Overflowing of box content](#)
  - [The marquee properties](#)
  - [Rotating boxes](#)

## CSS Box Model Example

# CSS Box Model

- **Box Dimensions**
- All HTML elements in a [box model](#) are represented as rectangular boxes. The dimensions of the box model are calculated by using the height and width of the content area that gets applied to the element.
- Each box is associated with a content area and many optional areas, such as [padding](#), [border](#), and [margin](#).
- The size of each area is specified by using the box model dimensions shown in the following figure:



# CSS Box Model

- In the above figure, the perimeter of the content, padding, border, and margin is called an edge. Each box in the box model have four edges, describes in the table given below:

Edge	Description
content edge	Surrounds a rectangle specified by the width and height of the box. This is also called the inner edge. The four content edges represent the content area of the box.
padding edge	Surrounds the padding box. If the padding has 0 width, the padding edge becomes a content edge. The four padding edges represent the padding area of the box.
border edge	Surrounds the border of the box. If the border has 0 width, the border edge becomes a padding edge. The four edges represent the border area of the box.
margin edge	Surrounds the margin of the box. If the border has 0 width, the margin edge becomes a border edge. The margin edge is also called as outer edge. The four border edges represent the margin area of the box.

# CSS Box Model

- **CSS Dimension Properties**
- The table given here lists all the CSS Dimension Properties:

Property	Possible Values	Property	Possible Values
width	auto length % inherit	min-height	length % inherit
height	auto length % inherit	max-height	none length % inherit
min-width	length % inherit	max-width	none length % inherit



# CSS Box Model

- **CSS Padding Properties**
- Padding (space) in the box model specifies the distance between the border of an element and the content within it.
- The padding is affected by the background color of an element. The value of padding cannot be negative.
- The shorthand property in padding is used to change all the padding properties, such as padding-top, padding-bottom, padding-right, and padding-left at once.
- The syntax used to set padding properties is as follows:

**padding: value;**

- Padding can be set according to the values describes in the table given here:
  - Length -- Specifies the fixed padding in the pt and em units.
  - Percentage -- Specifies padding in percentage with respect to the width of a parent block.
  - Auto -- Specifies the default padding from the top, bottom, left, or right direction.

# CSS Box Model

- The following code fragment shows the specifications of padding properties in the box model:

**#container**

```
{ padding-top: 7px; padding-left: 25%;  
padding-right: auto; padding-bottom: 45px;  
border: 2px solid black;  
}
```

- In the preceding code fragment, you can see the following four properties:
  - padding-top
  - padding-bottom
  - padding-right
  - padding-left

# CSS Box Model

- **CSS Padding Shorthand**
- If the padding property has four values, then the values are applied to the top, right, bottom, and left paddings, respectively.
- The padding property can have from one to four values as shown here one by one.
- (i) If CSS padding contains four values like:  
**padding: 25px 50px 75px 100px;**
- Then, these four values represents:
  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px
  - left padding is 100px

# CSS Box Model

- (ii) If CSS padding contains three values like:  
**padding: 25px 50px 75px;**
  - Then, these three values represents:
    - top padding is 25px
    - right and left paddings are 50px
    - bottom padding is 75px
- (iii) If CSS padding contains two values like:  
**padding: 50px 100px;**
  - Then, these two values represents:
    - top and bottom paddings are 50px
    - right and left paddings are 100px
- (iv) And, if CSS padding contains only one values like:  
**padding: 100px;**
  - Then, this value represents:
    - all four paddings are 100px

# CSS Box Model

- **Border properties** (<https://www.w3.org/TR/CSS2/box.html#border-properties>)
- The border properties specify the width, color, and style of the [border area](#) of a box. These properties apply to all element
- **Border width:**
  - **border-top-width**
  - **border-right-width**
  - **border-bottom-width**
  - **border-left-width**
  - **border-width**
- The border width properties specify the width of the [border area](#). The properties defined in this section refer to the **<border-width>** value type, which may take one of the following values:
  - **Thin** -- A thin border.
  - **Medium** -- A medium border.
  - **Thick** -- A thick border.
  - [<length>](#) The border's thickness has an explicit value.
  - Explicit border widths cannot be negative. The interpretation of the first three values depends on the user agent. The following relationships must hold, however:  
**'thin' <= 'medium' <= 'thick'**

# CSS Box Model

- Furthermore, these widths must be constant throughout a document.
- **'border-top-width', 'border-right-width', 'border-bottom-width', 'border-left-width'**

*Value:* <border-width> | inherit

- These properties set the width of the top, right, bottom, and left border of a box.

*Value:* <border-width>{1,4} / inherit

# CSS Box Model

- This property is a shorthand property for setting 'border-top-width', 'border-right-width', 'border-bottom-width', and 'border-left-width' at the same place in the style sheet.
- If there is only one component value, it applies to all sides.
- If there are two values, the top and bottom borders are set to the first value and the right and left are set to the second.
- If there are three values, the top is set to the first value, the left and right are set to the second, and the bottom is set to the third.
- If there are four values, they apply to the top, right, bottom, and left, respectively.
- In the examples below, the comments indicate the resulting widths of the top, right, bottom, and left borders:

```
h1 { border-width: thin }    /* thin thin thin thin */
```

```
h1 { border-width: thin thick } /* thin thick thin thick */
```

```
h1 { border-width: thin thick medium } /* thin thick medium  
thick */
```

# CSS Box Model

- **Border color:**
  - border-top-color
  - border-right-color
  - border-bottom-color
  - border-left-color
  - border-color
- The border color properties specify the color of a box's border.
- 'border-top-color', 'border-right-color', 'border-bottom-color', 'border-left-color'

*Value:* <color> | transparent | inherit

- The 'border-color' property sets the color of the four borders. Values have the following meanings:



# CSS Box Model

- <color> -- Specifies a color value.
  - **Transparent** -- The border is transparent (though it may have width).
  - The 'border-color' property can have from one to four component values, and the values are set on the different sides as for 'border-width'.
  - If an element's border color is not specified with a border property, user agents must use the value of the element's 'color' property as the computed value for the border color.
  - In this example, the border will be a solid black line.
- ```
p { color: black; background: white; border: solid; }
```

# CSS Box Model

- **Border style:**
  - border-top-style
  - border-right-style
  - border-bottom-style
  - border-left-style
  - border-style
- The border style properties specify the line style of a box's border (solid, double, dashed, etc.). The properties defined in this section refer to the **<border-style>** value type, which may take one of the following values:
  - **None**-- No border; the computed border width is zero.
  - **Hidden** Same as 'none', except in terms of border conflict resolution for table elements.
  - **Dotted** The border is a series of dots.
  - **Dashed** The border is a series of short line segments.
  - **Solid** The border is a single line segment.
  - **Double** The border is two solid lines. The sum of the two lines and the space between them equals the value of border-width.
  - **Groove** The border looks as though it were carved into the canvas.
  - **Ridge** The opposite of 'groove': the border looks as though it were coming out of the canvas.
  - **Inset** The border makes the box look as though it were embedded in the canvas.
  - **Outset** The opposite of 'inset': the border makes the box look as though it were coming out of the canvas.

# CSS Box Model

- All borders are drawn on top of the box's background.
- The color of borders drawn for values of 'groove', 'ridge', 'inset', and 'outset' depends on the element's border color properties, but UAs may choose their own algorithm to calculate the actual colors used.
- For instance, if the 'border-color' has the value 'silver', then a UA could use a gradient of colors from white to dark gray to indicate a sloping border.
- The 'border-style' property sets the style of the four borders. It can have from one to four component values, and the values are set on the different sides as for 'border-width' above.

**#xy34 { border-style: solid dotted }**

- In the above example, the horizontal borders will be 'solid' and the vertical borders will be 'dotted'.
- Since the initial value of the border styles is 'none', no borders will be visible unless the border style is set.

# CSS Box Model

- **Margin Properties**
- The blank area around the border of an element is called margin.
- Margin is used to create an extra space around an element. It is completely transparent and does not contain any background color.
- Margin is also used to determine spacing around different elements. The margin property is used to set all the sides of an element, such as top, right, bottom, and left.
- **CSS Margin Syntax**
- The syntax to set the margin property is as follows:  
**p { margin: value; }**

# CSS Box Model

- **CSS Margin Possible Values**
- The table given below describes possible values of the margin property:

| Value   | Description                                                                           |
|---------|---------------------------------------------------------------------------------------|
| Auto    | Defines margin.                                                                       |
| Length  | Defines a fixed margin in px, pt, and cm. The default value of this property is 0px.  |
| %       | Defines margin in percentage. The value of margin is the height of the nearest block. |
| Inherit | Defines that margin should be inherited from the parent element.                      |

# CSS Box Model

- **Margin Individual Sides**
- A box has four sides; therefore, the margin in a box is divided into the following four properties:
  - margin-top
  - margin-bottom
  - margin-right
  - margin-left
- **CSS Margin Example**
- The following code fragment shows the specifications of margin property in the [box model](#):
- `#container{ margin-top: 7px; margin-left: 25%; margin-right: auto; margin-bottom: 45px; border: 2px solid black; }`
- The top, right, bottom, and left values of a margin can be changed independently by using different properties, such as margin-top, margin-right, margin-bottom, and margin-left.
- **Here is an example demonstrates margin property in CSS:**

# CSS Box Model

- **CSS margin Shorthand**
  - A margin shorthand property is used to change all the properties of margin at once. The values of margin property ranges between one and four. Let's take a look at each case one by one.
- (i) If CSS margin property contains four values like this:  
**margin: 25px 50px 75px 100px;**
- Then, these four values represents:
    - top margin is 25px
    - right margin is 50px
    - bottom margin is 75px
    - left margin is 100px
- (ii) If CSS margin property contains three values like this:  
**margin: 25px 50px 75px;**
- Then, these three values represents:
    - top margin is 25px
    - right and left margins are 50px
    - bottom margin is 75px

# CSS Box Model

(iii) If CSS margin property contains two values like this:

**margin: 50px 100px;**

- Then, these two values represents:
- top and bottom margins are 50px
- right and left margins are 100px

(iv) And, if CSS margin property contains only one value like this:

**margin: 100px;**

- Then, the above value represents:
- all four margins are 100px



# CSS Box Model

- **Width and Height Properties**
- The width and height properties specify the width and height of the content area, [padding](#) area, or [border](#) area of a [box](#).
- The syntax of these properties is given as follows:

**width/height: <length> | <percentage> | auto**

- In the preceding syntax, the <length> specifies the width of the content area; <percentage> specifies the width of the content area in the percentage; and auto specifies that the width depends on the values of other properties.
- **Example**
- Some examples of using these properties are given as follows:
- ```
p { width: 100px; height: 50px; }
```

# CSS Box Model

- **CSS min-width and min-height**
- The min-width and min-height properties are used to set the minimum width and height respectively, of a block-level element.
- The syntax of these properties is given as follows:

**min-width/min-height: <length> | <percentage> | inherit**

- In the preceding syntax, <length> specifies a fixed minimum width height; <percentage> specifies a minimum value for width or height as a percentage of the corresponding dimension of the containing block; and none specifies no limit on the width or height of the box.

# CSS Box Model

- **CSS max-width and max-height**
- The max-width and max-height properties are used to set the maximum width and height, respectively, of the block-level element.
- The syntax of these properties is given as follows:

**max-width/max-height: <length> | <percentage> | none**

- In the preceding syntax, <length> specifies a fixed maximum width or height; <percentage> specifies a maximum value for width or height as a percentage of the corresponding dimension of the containing block; and none specifies no limit on the width or height of the box.
  - Some examples of using min-width, max-width, min-height, and max-height properties are given as follows :
- ```
p { min-width: 10px; min-height: 10px; max-width: 100px; max-height: 100px; }
```

# CSS Box Model

- **Floating Boxes**
- A floating box is a box that can display the content of an element in left or right direction with respect to the content of another element.
- For example, you can display a box containing the content of a paragraph in left direction and an image in the right direction. You can float an element by using the float property.
- The syntax to use the float property is given as follows :

**float: left | right | none | <page-floats>**

- An example of using the float property is given as follows :

```
img { float: left; }
```

# CSS Box Model

- **Overflowing Box Content**
- Sometimes, the amount of content placed in an HTML element is much larger than the capacity of the element. In such cases, the content overflows from the element, which may disrupt the display of the entire Web page.
- CSS provides various overflow properties to restrict the overflow of content in an element.
- **CSS Overflow Properties**
- The following table describes the overflow properties provided by CSS:
  - overflow-x -- Specifies clipping at the left and right edges
  - overflow-y -- Specifies clipping at the top and bottom edges
  - overflow-style -- Specifies a scrolling method for elements that overflow
- **CSS Overflow Syntax**
- The syntax to use the overflow-x and overflow-y properties is given as follows:  
**overflow-x/overflow-y: value;**

# CSS Box Model

- In the preceding syntax, the description of various values is given in the following table:

| Value      | Description                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------------------|
| visible    | Indicates that the content is not clipped                                                                                                |
| hidden     | Indicates that the content is clipped and that no scrolling mechanism should be provided to view the content outside the clipping region |
| scroll     | Indicates that the content is clipped and a scrolling mechanism should be displayed for a box                                            |
| auto       | Indicates that a scrolling mechanism has to be provided to a box, because the content of this box overflows from its content area        |
| no-display | Indicates that if the content does not fit in the content box, the box is removed completely                                             |
| no-content | Indicates that if content does not fit in the content box, the content is hidden completely                                              |

# CSS Box Model

- **CSS overflow-style Syntax**
- The syntax to use the overflow-style property is given as follows:

**overflow-style:auto |  
[scrollbar|panner|move|marquee][,[scrollbar|pa  
nner|move|marquee]]\*;**

- The overflow shorthand can be used as an alternate to set the values for the overflow-x and overflow-y properties.

# CSS Box Model

- **Rotating Boxes**
- CSS allows you to rotate the content of an HTML element by using the rotation and rotation-point properties.
- **CSS rotation**
- The rotation property rotates the content of an HTML element according to the specified values in degree. It rotates the element content around a given point defined by the rotation-point property.
- The syntax of using the rotation property is given as follows:

**rotation: <angle> ;**

- In the preceding syntax, <angle> specifies the number of degrees up to which the element needs to be rotated.



# CSS Box Model

- **CSS rotation-point**
- The rotation-point property defines a point as an offset from the top left border edge.
- The syntax of using the rotation-point property is given as follows :

**rotation-point: <bg-position> ;**

- In the preceding syntax, <bg-position> specifies the values that defines a point as an offset from the top left border edge.
- **CSS rotation and rotation-point Example**
- Following is an example of using the rotation and rotation-point properties is given as follows :

**h1 { rotation: 45deg; rotation-point: bottom left; }**

# CSS3 Line Box Model

- **CSS3 Line Box Model**
- CSS3 introduces a new model, called the line box model to enhance the [box model](#).
- The line box model helps to display elements and [text](#) in line with each other.
- It also enables you to display first line of an element in a specified format or to display the first character of an element in big [font](#) size.

# CSS3 Line Box Model

- **CSS3 Line Box Model Properties**

| Property                   | Description                                                                                                                      |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| alignment-adjust           | Allows you to specify alignment points for elements                                                                              |
| alignment-baseline         | Specifies the alignment of an inline-level element with respect to its parent                                                    |
| baseline-shift             | Allows you to reposition the dominant-baseline with respect to other dominant-baseline                                           |
| dominant-baseline          | Specifies a scaled-baseline-table                                                                                                |
| drop-initial-after-adjust  | Specifies an alignment point for the initial letter box with respect to the primary connection point                             |
| drop-initial-after-align   | Specifies an alignment line within the initial line box that is used at the primary connection point with the initial letter box |
| drop-initial-before-adjust | Specifies an alignment point of the drop initial for the secondary connection point                                              |

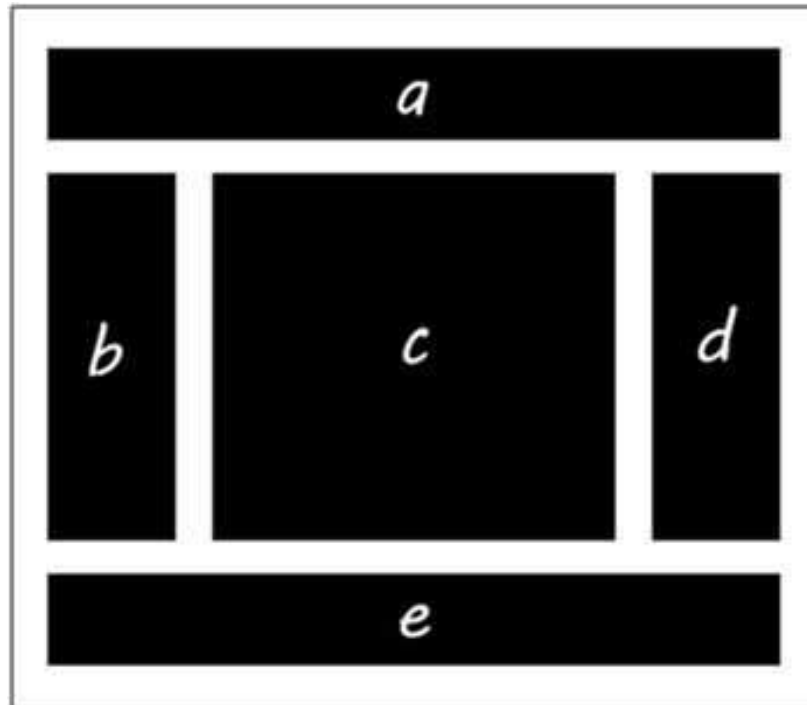
# CSS3 Line Box Model

- **CSS3 Line Box Model Properties**

| Property                  | Description                                                                                                                        |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| drop-initial-before-align | Specifies an alignment line within the initial line box that is used at the secondary connection point with the initial letter box |
| drop-initial-size         | Specifies a value to show the size of the initial letter                                                                           |
| drop-initial-value        | Specifies a value to activate a drop-initial effect                                                                                |
| inline-box-align          | Specifies the alignment of a line of a multi-line inline block with the previous and next inline elements within a line            |
| line-stacking             | Specifies a shorthand property for setting the line-stacking-strategy, line-stacking-ruby, and line-stacking-shift properties      |
| line-stacking-ruby        | Specifies the line stacking method for elements, which are ruby annotation elements                                                |
| line-stacking-shift       | Specifies the line stacking method for elements, which are baseline-shifted elements                                               |
| line-stacking-strategy    | Specifies the height of the text content area in an inline box                                                                     |

# CSS3 Template Layout Model

- CSS3 introduces a new layout model, called the template layout model, which allows you to present the content contained in elements into slots.
- The slots are the placeholders that can be created by declaring a grid structure using certain alphabetical characters as shown in the following figure (Using Alphabets to Create Slots in a Template):



# CSS3 Template Layout Model

- Using the template layout model, you can divide the layout of a Web page or a form in following two parts :
  - Grid of the page or window
  - Fonts, indents, and colors of the text and other objects
- The template layout model defines a layout policy, called template-based positioning, to give an invisible grid to an element for aligning with other elements.
- A template does not allow elements to overlap and provides layouts with different widths, heights, and alignments.
- An element mapped to its slot in a template by using the position property.
- The template itself is specified by using the display property for remapping of some elements.
- Let's learn about the display and position properties.

# CSS3 Template Layout Model

- **CSS3 display Property**
- The display property is used to set the layout for the content of elements in the rows and columns format.
- It provides a template value to an element, which is also called a template element.
- The syntax of using the display property is given as follows :

**display:inline? [ <string> [ / <row-height> ]? ]+  
<col-width>\*;**

# CSS3 Template Layout Model

- The description of the preceding syntax is given as follows :
- **The inline keyword** - Shows that an element is an inline-level element. Note that an element is considered to be a block-level element, in case the inline keyword is not used.
- **The <string> parameter** - Represents one or more letters, at signs (@), periods (.), or spaces. Each string stands for one row in the template and each character, other than a space, stands for one column in that row. In the <string> parameter, spaces have no meaning. They are just added to make better readable content.
- **The <row-height> parameter** - Sets the height of the preceding row. The default value for this parameter is auto. You can specify the length of the height of a row or use the asterisk (\*) symbol to represent that all the rows are of equal height.
- **The <col-width> parameter** - Sets the width of the preceding column. The default value for this parameter is auto. You can specify the length of the width of a column or use the asterisk (\*) symbol to represent that all the columns are of equal width.



# CSS3 Template Layout Model

- The symbols used in a template have the following meaning :
  - **Any letter** - Shows the slot of content
  - **at signs (@)** - Shows the default slot of content
  - **period (.)** - Shows white spaces
- **Note** - The letters used in a template are case-insensitive
- **CSS3 display Property Example**
- Following is an example of using the display property:

```
div { display:".aa..bb." ; }
```

# CSS3 Template Layout Model

- **CSS3 position Property**
- The position property is used to map the element slots with the elements. It specifies the row and column of a template in which an element will be placed.
- The syntax to use the position property is given as follows :  
**position: <letter> | same;**
- In the preceding syntax, the <letter> parameter can be a single letter or the @ symbol.
- The <letter> parameter places an element in the specified slot. The value, same, places an element to the same slot where the last element was placed.
- It means multiple elements can be put into the same slot.

# CSS3 Template Layout Model

- **CSS3 position Property Example**
- Following is an example of using the position property:

```
<style type="text/css">
```

```
body { height:100%;
```

```
display:
```

```
    "a . b . c" /2em
```

```
    ". . . . ." /1em
```

```
    "d . e . f"
```

```
    ". . . . ." /1em
```

```
    "g . h . i" /2em
```

```
    5em 1em * 1em 10em }
```

```
#logo {position: a}
```

```
#motto {position: b}
```

```
#date {position: c}
```

```
#main {position: e}
```

```
#adv {position: f}
```

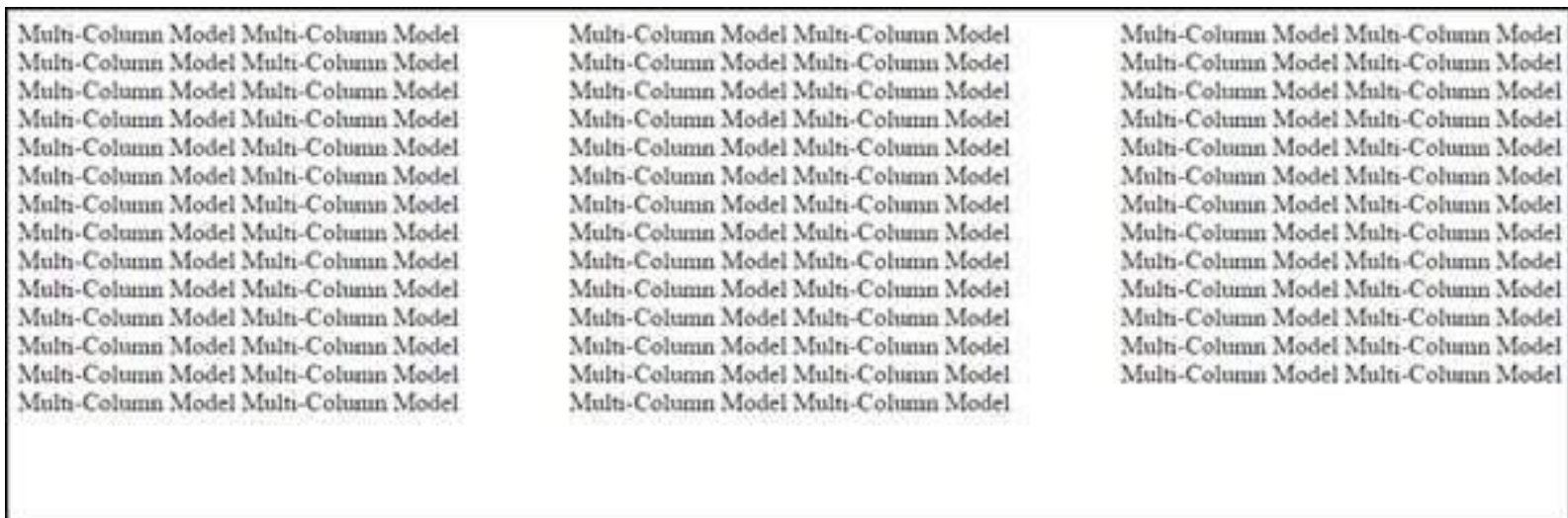
```
#copy {position: g}
```

```
#about {position: h}
```

```
</style>
```

# CSS Multi-Column Model

- At times, you need to display your Web content in multiple columns just like the columns of a newspaper.
- CSS enables you to show your element content in multiple columns. The W3C organization introduces the multi-column model to extend the current CSS box model.
- **Show Content of Web Page in Multiple Columns**
- Following figure shows the content of a Web page in multiple columns :



# CSS Multi-Column Model

- CSS provides the following properties to show the content in multiple columns :
  - column-count
  - column-width
  - column-gap
  - column-rule
  - column-fill
  - column-span
- Let's discuss these properties one by one in detail.
- **CSS column-count**
  - The column-count specifies the number of columns to display the content of an element. The column-count property describes the number of columns of the multicol element.
  - The syntax to use the column-count property is given as follows :  
**column-count: <integer> | auto;**
  - In the preceding syntax, the <integer> parameter specifies the number of columns in which the content of the element is displayed.
  - The auto keyword specifies that the number of columns is determined by other properties.

# CSS Multi-Column Model

- **CSS column-width**

- The column-width specifies the width of a column. The column-width property describes the width of columns in multicol elements.
- A multicol element is the element, whose column-width property is not set to auto.
- The syntax to use the column-width property is given as follows :

**column-width: <length> | auto;**

- In the preceding syntax, the <length> parameter stands for the column width of a column.
- Note that the specified width of a column can be wider or narrower with respect to the actual available space and the browser will show the content accordingly.
- The auto keyword specifies that the column width is determined by other properties.

# CSS Multi-Column Model

- **CSS column-gap**

- The column-gap specifies the padding between columns. The column-gap property sets the gap between columns.
- If there is a column rule between columns, it will appear in the middle of the gap.
- The syntax to use the column-gap property is given as follows :

**column-gap: <length> | normal;**

- In the preceding syntax, the <length> parameter stands for the length of the gap between columns.
- The normal value specifies the default value for the column gaps. Note that column gaps cannot be negative.

# CSS Multi-Column Model

- **CSS column-rule**

- The column-rule specifies a border between columns. The column-rule property inserts a line between columns.
- The syntax to use the column-rule property is given as follows :

**column-rule: <column-rule-width> || <border-style> || [ <color> | transparent ];**

- The column-rule property is a shorthand for setting all the other properties related to columns, such as column-rule-width, column-rule-style, and column-rule-color. These properties are described as follows :
  - **column-rule-width** - Sets the width of a column rule
  - **column-rule-style** - Sets the style of a column rule
  - **column-rule-color** - Sets the color of a column rule



# CSS Multi-Column Model

- **CSS column-fill**

- The column-fill balances the content of an element in all the specified columns equally.
- The syntax to use the column-fill property is given as follows :

**column-fill: auto | balance;**

- In the preceding syntax, the value, balance, specifies that the content of an element is equally divided between columns; while the value, auto, specifies that the columns are filled sequentially.

- **CSS column-span**

- The column-span specifies that elements are set to span across all columns. The column-span property is used to span the content of an element over multiple columns.
- It enables you to span an element across several columns.
- Note that an element that spans more than one columns is called a spanning element.
- The syntax to use the column-span property is given as follows :

**column-span: 1 | all;**

- In the preceding syntax, the value, 1, specifies that the element does not span multiple columns; while the value, all specifies that the element spans across all columns.

# CSS Multi-Column Model

- **CSS Multi-Column Example**

- Some examples of using the above properties are given as follows :

**Ibody{column-width:12em;}      /\* Example 1 \*/**

**IIbody{column-count:2;}      /\* Example 2 \*/**

**IIIbody{column-gap:1em;column-rule:thin solid black;}      /\* Example 3\* /**

**div{column-fill:balance;}      /\* Example 4 \*/**

**h2{column-span:all;}      /\* Example 5 \*/**

- The description of the preceding examples is given as follows:
- **Example 1** - Sets the BODY element to have columns at least of 12em wide. The number of columns depends on the available space.
- **Example 2** - Sets the number of columns, whose widths vary on the basis of the available space.
- **Example 3** - Sets the gap of 1em between two adjacent columns. In the middle of the gap, there is a rule, which is described by the column-rule property.
- **Example 4** - Sets the columns with appropriate and equal length.
- **Example 5** - Sets the H2 elements to span across all columns.

# CSS Multi-Column Model

- **CSS Shorthand Columns Property**

- You can also use the shorthand columns property to set any or all the properties in one declaration.
- The following example shows how to use the shorthand columns property :

**body { columns: 2 12em }**

- In the preceding example, the number and widths of columns is set.
- You should note that at the time of displaying the content of an element in multiple columns, you also need to determine where column breaks are placed.
- Some properties that are newly introduced in CSS3 to insert column breaks are given as follows :

**(i) break-before** - Specifies a break that should be inserted before the content of an element. The syntax to use the break-before property is given as follows :

**break-before auto | always | avoid | left | right | page | column | avoid-page | avoid-column;**

# CSS Multi-Column Model

**(ii) break-after** - Specifies a break that should be inserted after the content of an element. The syntax to use the break-after property is given as follows :

**break-after auto | always | avoid | left | right | page | column | avoid-page | avoid-column;**

**(iii) break-inside** - Specifies an internal break. The syntax to use the break-inside property is given as follows :

**break-inside auto | avoid | avoid-page | avoid-column;**

# CSS Multi-Column Model

- **Values Used in Column or Page Breaks**
- The preceding properties uses the values described in the following table :

| Value  | Description                                                                                                                        |
|--------|------------------------------------------------------------------------------------------------------------------------------------|
| auto   | Specifies that a page/column break is neither forced nor forbidden to generate before, after, or inside a box                      |
| always | Specifies that a page break is always forcedly generated before or after a box                                                     |
| avoid  | Specifies that a page/column break is avoided to be generated before, after, or inside a box                                       |
| left   | Specifies one or two page breaks to be forcedly generated before or after a box so that the next page is formatted as a left page  |
| right  | Specifies one or two page breaks to be forcedly generated before or after a box so that the next page is formatted as a right page |

# CSS Multi-Column Model

- **Values Used in Column or Page Breaks**
- The preceding properties uses the values described in the following table :

| Value        | Description                                                                             |
|--------------|-----------------------------------------------------------------------------------------|
| page         | Specifies that a page break is forcedly generated before or after a box                 |
| column       | Specifies that a column break is forcedly generated before or after a box               |
| avoid-page   | Specifies that a page break is avoided to be generated before, after, or inside a box   |
| avoid-column | Specifies that a column break is avoided to be generated before, after, or inside a box |

# CSS Multi-Column Model

- Some examples of inserting or avoiding column or page breaks are as follows :

**.multicol img { break-after: column }**

**/\*Example 1 \*/**

**p { break-inside: avoid-column } /\* Example 2 \*/**

- In the Example 1, a column break is generated after inserting an image; while in Example 2, a column break is avoided.

# Unit-7

## Introduction to Java Script



# Introduction to Java Script

- **What is JavaScript ?**
- JavaScript is a client and server-side object-based scripting language that is used to make interactive Web pages.
- A scripting language is a lightweight programming language with less complexity.
- JavaScript is the most usually used scripting language to add dynamism and interactivity to Web pages.
- This is because JavaScript, written on the client-side, executes on a client browser, thereby reducing the load on the server.

# Introduction to Java Script

- **JavaScript is an Interpreted Language**
- JavaScript is an interpreted language, which implies that scripts written to JavaScript are processed line by line.
- These scripts are interpreted by the JavaScript interpreter, which is a built-in component of the Web browser.
- JavaScript can be written on the client-side as well server-side.
- Client-side JavaScript allows you to validate only those programs that execute and produce the result on the client-machine. In counterpoint/contrast, server-side JavaScript validates only those programs that execute on the server.
- JavaScript includes various built-in [objects](#) and features that can be used to make your [HTML](#) pages dynamic.

# Introduction to Java Script

- **JavaScript is platform-independent**
  - JavaScript is platform-independent, which implies that you need to write the script once and can run it on any platform or browser without affecting the output of the script.
- **Why to Learn JavaScript ?**
- There are the three languages, all web developers must know, these are the following:
  - [HTML](#) - to define the content of web pages
  - [CSS](#) - to define the layout of web pages
  - [JavaScript](#) - to program the behavior of web pages
- So to program the behavior of Web pages, you must have to learn JavaScript.

# Introduction to Java Script

- **Advantages of JavaScript**
  - Less server interaction
  - More interactivity
  - Richer interfaces
  - Fast feedback to visitors

# Introduction to Java Script

- **Features of Java Script**
- **Imperative and Structured**
  - The Statement Control Syntax System of Java Script is very similar to Statement Control Syntax used in C Language.
  - Java Script allows more abilities to perform as HTML is only capable of designing websites and incapable of performing logic operations like condition checking, statements looping (while & for), statement decision making (else and if) at client edge and adding two numbers.
  - The only syntactical difference between C and Java Script is that, in Java Script semi-colon is not necessary to terminate a statement, whereas in C it is necessary to terminate a statement.

# Introduction to Java Script

- **Dynamic Typing**

- JavaScript supports dynamic typing which means types of the variable are defined based on the stored value.
- For example, if you declare a variable x then you can store either a string or a Number type value or an array or an object. This is known as dynamic typing.
- To understand this, in languages like Java, we explicitly mention that a particular variable will store a certain type of data, whereas in JavaScript we do not have to provide the data type while declaring a variable.
- In JavaScript, we just have to use var or let keyword before the variable name to declare a variable without worrying about its type.

# Introduction to Java Script

- **Functional Style**

- This implies that JavaScript uses a functional approach, even objects are created from the constructor functions and each constructor function represents a unique object-type.
- Also, functions in JavaScript can be used as objects and can be passed to other functions too.

- **Platform Independent**

- This implies that JavaScript is platform-independent or we can say it is portable; which simply means that you can simply write the script once and run it anywhere and anytime.
- In general, you can write your JavaScript applications and run them on any platform or any browser without affecting the output of the Script.

# Introduction to Java Script

- **Prototype-based Language**

- JavaScript is a prototype-based scripting Language. This means javascript uses prototypes instead of classes or inheritance.
- In languages like Java, we create a class and then we create objects for those classes. But in JavaScript, we define object prototype and then more objects can be created using this object prototype.



# JavaScript Syntax

- Syntax of a JavaScript program/code is nothing, it's just a set of rules that defines how JavaScript programs are constructed.
- A JavaScript program consists of JavaScript statements that is placed inside the HTML `<script> ... </script>` tag in a web page.
- You are free to place the `<script>` containing your JavaScript program anywhere within your web page, but to keep it within the `<head>` tag is the preferred way.
- Here is the general form shows how script tag plays a role in placing JavaScript code inside it:

**`<script ...>`**

**JavaScript code**

**`</script>`**

# JavaScript Syntax

- The script tag takes two important attributes:
  - **language** - The language attribute is used to specify what scripting language you are using
  - **type** - The type attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript"
- Syntax of your JavaScript code segment will be:

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

**JavaScript code**

```
</script>
```

# JavaScript Syntax

- You can place or include your JavaScript code anywhere in your HTML document, even you can also place your JavaScript code in an external file and include this file in any HTML document to use your JavaScript code present in that file.
- Here are the place, where you can include your JavaScript code:
  - In the `<head> .. </head>` section
  - in the `<body> .. </body>` section
  - In the `<body> .. </body>` and `<head> .. </head>` sections
  - In an external file and then include this file in the `<head> .. </head>` section

# JavaScript Syntax

- **Examples**
  - JavaScript Code in Head
  - JavaScript Code in Body
  - JavaScript Code in Body and Head
  - JavaScript Code in External File

# JavaScript Syntax

- **JavaScript Code in External File** is the most preferred and safe way to use your all JavaScript code from an external file. Because if you want to use same JavaScript code in multiple HTML file, then this is the most preferred way.
- You can write all your JavaScript code in a simple text file having name like **myjavascript.js**. Use **.js** extension after the filename.
- Now you can include this file in any HTML document to use all the code present in that file. And later if you want to change the JavaScript code, you only have to make changes in a single file (say **myjavascript.js**), and all the HTML document automatically changes.

# Fundamentals of JS

- Programming Fundamentals of JS
  - Lexical Structure
  - Variables
  - Operators
  - Control Flow Statements
  - Popup Boxes

# Fundamentals of JS

- **Lexical Structure of JS**
  - **The lexical Structure of JS defines the rules for the following aspects:**
    - **Character Set**
    - **Case Sensitivity**
    - **White spaces and Line breaking**
    - **Optional Semicolons**
    - **Comments**
    - **Literals**
    - **Identifiers**
    - **Reserved Keywords**

# Fundamentals of JS

- **Understanding Character Set**
  - Character Set is a set of character reserved to write JS programs. ASCII character set.
- **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.



# Fundamentals of JS

- Whitespace and Line Breaks
- JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.
- Semicolons are Optional
- Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

# Fundamentals of JS

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

```
<!--
```

```
    var1 = 10
```

```
    var2 = 20
```

```
//-->
```

```
</script>
```

- But when formatted in a single line as follows, you must use semicolons –

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

```
<!--
```

```
    var1 = 10; var2 = 20;
```

```
//-->
```

```
</script>
```

**Note** – It is a good programming practice to use semicolons.

# Fundamentals of JS

- Comments in JavaScript
  - JavaScript supports both C-style and C++-style comments, Thus –
  - Any text between a `//` and the end of a line is treated as a comment and is ignored by JavaScript.
  - Any text between the characters `/*` and `*/` is treated as a comment. This may span multiple lines.
  - 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 `//-->`.

# Fundamentals of JS

- Example
- The following example shows how to use comments in JavaScript.

```
<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>
```

-

# Fundamentals of JS

## JavaScript Literals

- JavaScript Literals are the fixed value that cannot be changed, you do not need to specify any type of keyword to write literals. Literals are often used to initialize variables in programming, names of variables are string literals.
- A JavaScript Literal can be a numeric, string, floating-point value, a boolean value or even an object. In simple words, any value is literal, if you write a string **"Studytonight"** is a literal, any number like **7007** is a literal, etc.

# Fundamentals of JS

- JavaScript supports various types of literals which are listed below:
  - Numeric Literal
  - Floating-Point Literal
  - Boolean Literal
  - String Literal
  - Array Literal
  - Regular Expression Literal
  - Object Literal

# Fundamentals of JS

- JavaScript Numeric Literal
  - It can be expressed in the decimal(base 10), hexadecimal(base 16) or octal(base 8) format.
  - Decimal numeric literals consist of a sequence of digits (0-9) without a leading 0(zero).
  - Hexadecimal numeric literals include digits(0-9), letters (a-f) or (A-F).
  - Octal numeric literals include digits (0-7). A leading 0(zero) in a numeric literal indicates octal format.
- JavaScript Numeric Literals Example
  - 120 // decimal literal**
  - 021434 // octal literal**
  - 0x4567 // hexadecimal literal**

# Fundamentals of JS

- JavaScript Floating-Point Literal
  - It contains a decimal point(.)
  - A fraction is a floating-point literal
  - It may contain an Exponent.
- JavaScript Floating-Point Literal Example
  - 6.99689 // floating-point literal**
  - 167.39894 // negative floating-point literal**
- JavaScript Boolean Literal
  - Boolean literal supports two values only either **true** or **false**.
- JavaScript Boolean Literal Example
  - true // Boolean literal**
  - false // Boolean literal**



# Fundamentals of JS

- JavaScript String Literal
  - A string literal is a combination of zero or more characters enclosed within a single(') or double quotation marks (").
- JavaScript String Literal Example
  - "Study" // String literal**
  - 'tonight' // String literal**

# Fundamentals of JS

- String literals can have some special characters too which are tabled below.

| Character | Description                                                                     |
|-----------|---------------------------------------------------------------------------------|
| \b        | It represents a backspace.                                                      |
| \f        | It represents a Form Feed.                                                      |
| \n        | It represents a new line.                                                       |
| \r        | It represents a carriage return.                                                |
| \t        | It represents a tab.                                                            |
| \v        | It represents a vertical tab.                                                   |
| \'        | It represents an apostrophe or single quote.                                    |
| \"        | It represents a double quote.                                                   |
| \\        | It represents a backslash character.                                            |
| \uXXXX    | It represents a Unicode character specified by a four-digit hexadecimal number. |

# Fundamentals of JS

- JavaScript Array Literal
  - An array literal is a list of zero or more expressions representing array elements that are enclosed in a square bracket([]).
  - Whenever you create an array using an array literal, it is initialized with the elements specified in the square bracket.
- JavaScript Array Literal Example

```
var emp = ['aman','anu','charita']; // Array  
literal
```

# Fundamentals of JS

- JavaScript Regular Expression Literal
  - Regular Expression is a pattern, used to match a character or string in some text. It is created by enclosing the regular expression string between forward slashes.
- JavaScript Regular Expression Example
  - var myregexp = /ab+c/;     // Regular Expression literal**
  - var myregexp = new RegExp('abc');     // Regular Expression literal**

# Fundamentals of JS

- JavaScript Object Literal
  - It is a collection of key-value pairs enclosed in curly braces({ }). The key-value pair is separated by a comma.
- JavaScript Object Literal Example

**var games = {cricket :11, chess :2, carom: 4} //**  
**Object literal**

# Fundamentals of JS

- JavaScript Identifiers
  - Identifiers are names.
  - In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).
  - The rules for legal names are much the same in most programming languages.
  - In JavaScript, the first character must be a letter, or an underscore (\_), or a dollar sign (\$).
  - Subsequent characters may be letters, digits, underscores, or dollar signs.
  - Numbers are not allowed as the first character.  
This way JavaScript can easily distinguish identifiers from numbers.

# Fundamentals of JS

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

# JavaScript - Variables

- **JavaScript Datatypes**
  - One of the most fundamental characteristics of a programming language is the set of data types it supports.
  - These are the type of values that can be represented and manipulated in a programming language.
- 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.



# JavaScript - Variables

- **JavaScript Variables**
- Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers.
- You can place data into these containers and then refer to the data simply by naming the container.
- Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

```
<script type = "text/javascript">
```

```
<!--
```

```
    var money;
```

```
    var name;
```

```
//-->
```

```
</script>
```

# JavaScript - Variables

- 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.
- For instance, you might create a variable named **money** and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

```
<script type = "text/javascript">
```

```
<!--
```

```
    var name = "Ali";
```

```
    var money;
```

```
    money = 2000.50;
```

```
//-->
```

```
</script>
```

# JavaScript - Variables

- **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.

# JavaScript - Variables

- **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.

# JavaScript - Variables

```
<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>
```

# JavaScript - Variables

- **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.

# JavaScript Operators

- Operators in JavaScript, are used to perform some mathematical or logical manipulations. There are following types of operators available in JavaScript:
  - Arithmetic Operators
  - Assignment Operators
  - Comparison Operators
  - Logical (Relational) Operators

# JavaScript Operators

- **JavaScript Arithmetic Operators**
- JavaScript arithmetic operators are used to perform arithmetical task. Here, the following table lists the arithmetic operators available in JavaScript:
- **Important** - JavaScript addition operator (+) works for numeric as well as strings. For example, "a" + 20 will give "a20"

Operator	Meaning
+	This operator adds two operands
-	This operator subtracts second operand from the first
/	This operator divide numerator by denominator
*	This operator multiply both operands
%	This is a modulus operator, used to calculate the remainder
--	This is a decrement operator, decreases integer value by one
++	This is a increment operator, increases integer value by one



# JavaScript Operators

- **JavaScript Assignment Operators**
- Assignment operators in JavaScript, are used to assign the values of right side operand to the left side operand. Here, this table lists the assignment operators available in JavaScript:

Operator	Meaning
=	This operator Assigns values from the right side operands to left side operand
-=	This operator subtracts right operand from the left operand and assign the result to the left operand
+=	This operator adds right operand to the left operand and assign the result to the left operand
/=	This operator divides left operand with the right operand and assign the result to the left operand
*=	This operator multiplies right operand with the left operand and assign the result to the left operand
%=	This operator takes modulus using two operands and assign the result to the left operand

# JavaScript Operators

- **JavaScript Comparison Operators**
- JavaScript comparison operators are basically used in logical statements which is to determine the equality or difference between variables or values. The following table lists the comparison operators available in JavaScript:

Operator	Name
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

# JavaScript Operators

- **JavaScript Logical Operators**
- JavaScript logical operators are used to determine the logic between the variables or values. The following table lists the logical operators available in JavaScript:

Operator	Name
	or
&&	and
!	not

# JavaScript Operators

- **JavaScript Conditional (Ternary) Operator**
- Conditional operators in JavaScript, are used to assign a value to a variable based on some condition. Here is the general form to use JavaScript conditional operators:

**variable\_name = (condition) ? value1:value2**

# JavaScript Operators

- **JavaScript Operator Precedence**
- The following table describes the operators in the decreasing order of their precedence:

Operator Name	Operator
member operator	. []
new operator	new
function call operator	()
increment operator	++
decrement operator	--
logical NOT operator	!
bitwise NOT operator	~
unary plus operator	+
unary negation operator	-

# JavaScript Operators

Operator Name	Operator
typedef operator	typedef
void operator	void
delete operator	delete
multiplication operator	*
division operator	/
modulus operator	%
addition operator	+
subtraction operator	-
bitwise shift operator	<< >> >>>
relational operator	< <= > >=

# JavaScript Operators

Operator Name	Operator
in operator	in
instanceof operator	instanceof
equality operator	== != === !==
bitwise AND operator	&
bitwise XOR operator	^
bitwise OR operator	
logical AND operator	&&
logical OR operator	
conditional operator	?:

# JavaScript Operators

Operator Name	Operator
assignment operator	= += -= *= /= %= <<= >>= >>>= &= ^=  =
comma operator	,



# JavaScript Conditional Statements

- JavaScript conditional statements or decision making statements are used to perform particular actions based on the given conditions.
- You will learn all about conditional statements in detail divided into the following three tutorials:
  - if Statement
  - if-else Statement
  - switch Statement

# JavaScript Conditional Statements

- **JavaScript if Statement**
- The if statement in JavaScript is used to execute a group of one or more than one script statements only when the given/particular condition is met.

- Syntax:

```
if(condition)  
{  
    Statement1  
}
```

# JavaScript Conditional Statements

- **JavaScript if-else Statement**
- The if-else statement in JavaScript is used to execute the code that is written in the if statement, if the specified condition is true. Otherwise, the code written in the else statement is executed.
- Syntax:

```
if(condition)
{
    Statement1
}
else
{
    Statement2
}
```
- **JavaScript Nested if-else Statement**
- You are free to define one if-else statement into another, such statements are called as nested if-else statements.
- Nested if-else statement are useful in situations when additional checking is required.

# JavaScript Conditional Statements

- **JavaScript switch Statement**
- The switch statement in JavaScript is used to select a particular group of statements to be executed among several other groups of statements.

- Syntax:

```
switch(expression)  
{  
    case value1: statement1  
                                break;  
    case value2: statement2  
                                break;  
    default : statement_default  
                                break;  
}
```

# JavaScript Loops

- Loops or iteration statements are control flow statements that allows you to execute group of statements several number of times.
- There are the following three loops provided by JavaScript:
  - for Loop
  - while Loop
  - do-while Loop

# JavaScript Loops

- **JavaScript for Loop**
- The for loop in JavaScript is used to execute a block of code or group of statements multiple times based on the given/particular condition.
- Here is the general form to use for loop in JavaScript:  
**for(initialization; condition-checker; incrementation)**  
**{**  
**// block of code, to be executed here**  
**}**

# JavaScript Loops

- **JavaScript while Loop**
- The while loop in JavaScript is used when you want to check the condition at the start of the loop and then the group of statements that is to be executed is specified after the condition.
- Here is the general form to use while loop in JavaScript:

```
while(condition-checker)  
{  
    // block of code to be executed here  
    incrementation;  
}
```

# JavaScript Loops

- **JavaScript do-while Loop**
- The do-while loop in JavaScript is used in the case where you want a group of statements to be executed at least once.
- Since the condition is placed at the end of the loop in case of do-while loop in JavaScript, therefore you can use this loop to executed group of statements at least once.
- Here is the general form of do-while loop in JavaScript:

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



# JavaScript Loops

- **JavaScript Break and Continue**
- JavaScript break and continue statement/keyword is used to control the [loop](#).
- **JavaScript break Statement**
- JavaScript break statement is used to exit from the loop when the required condition is met.
- **JavaScript continue Statement**
- JavaScript continue statement is used to skip the remaining part of code and goes to the condition-checker part, if the required condition is met.

# JavaScript Popup/Dialog Box

- JavaScript dialog box are of following three types:
  - JavaScript dialog box for getting confirmation on any user input (confirm Box)
  - JavaScript dialog box for raising an alert box (alert Box)
  - JavaScript dialog box for a kind of input from the user (prompt Box)

# JavaScript Popup/Dialog Box

- **JavaScript alert Box**
  - An alert box in JavaScript is mainly used to display an alert message while executing JavaScript code.
- **JavaScript confirm Box**
  - The confirm box in JavaScript is used to display a message and return a true or false value.
- **JavaScript prompt Box**
  - The prompt box in JavaScript contains a text box along with OK and Cancel buttons. It returns the text string when OK is clicked and null when Cancel is clicked.

# JavaScript Functions

- A function is a collection of statements that is executed when it is called at some point in a program.
- JavaScript functions are nothing, but a block of code designed to be perform a specific task.
- **Define Function in JavaScript**
  - To define a function in JavaScript, use **function** keyword, followed by the function name, which is followed by the parentheses (contains parameter list). Here is the general form shows how to define a function in JavaScript:

```
function funName(parameter1, parameter2, parameter3)  
{  
code to be executed here  
}
```

- **Note** - The arguments inside functions are used as local variables.
- Here is an example of defining a function in JavaScript:
- `<script type="text/javascript"> function funName(parameter-list) { statements list } </script>`

# JavaScript Functions

- Let's look at the following example of JavaScript functions, which takes no parameters called fun1 is defined below :

```
<script type='text/javascript'>
```

```
function fun1()
```

```
{
```

```
    alert('Hello JavaScript, I am Statements  
of Function Definition');
```

```
}
```

```
</script>
```

# JavaScript Functions

- **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.
- **Syntax**
- The basic syntax is shown here.

```
<script type = "text/javascript">  
    <!-- function functionname(parameter-list)  
        { statements } //-->  
</script>
```

# JavaScript Functions

- **Call a Function in JavaScript**
- To call a function in JavaScript, you have to simply write the name of the function which is going to be called. Here is an example:

```
<script type="text/javascript">  
    fun1();  
</script>
```

- After calling the above function **fun1()** which is define in the second above code fragment, the function will tells the browser to give an alert box containing the string **"Hello JavaScript, I am Statements of Function Definition"**

```
<body>  
    <form>  
<input type = "button" onclick = "fun1()" value = "Say  
    Hello"> </form>  
</body>
```

# JavaScript Functions

- **JavaScript Function with Parameters or Arguments**
- A function with parameters or arguments accepts some values when it is called.
- Basically arguments are the values that you pass to a function, which has corresponding parameters to store them.



# JavaScript Functions

- Built in Global Functions of JS

Function	Description
<b>eval()</b>	The <b>eval()</b> function evaluates JavaScript code represented as a string.
<b>isFinite()</b>	The global <b>isFinite()</b> function determines whether the passed value is a finite number. If needed, the parameter is first converted to a number.
<b>isNaN()</b>	The <b>isNaN()</b> function determines whether a value is <a href="#">NaN</a> or not. Note, coercion inside the isNaN function has <a href="#">interesting</a> rules; you may alternatively want to use <a href="#">Number.isNaN()</a> , as defined in ECMAScript 2015.
<b>parseFloat()</b>	The <b>parseFloat()</b> function parses an argument (converting it to a string first if needed) and returns a floating point number.
<b>parseInt()</b>	The <b>parseInt()</b> function parses a string argument and returns an integer of the specified <a href="#">radix</a> (the base in mathematical numeral systems).
<b>escape</b>	Returns the hexadecimal encoding of an argument in the ISO Latin-1 character set.
<b>unescape</b>	Returns the ASCII string for the specified value.
<b>Number()</b>	Converts a value of an object into a number.

# JavaScript Functions

- **JavaScript return Statement**
- The return statement in JavaScript returns a value from a [function](#).

```
<script type="text/javascript">
```

```
function calmult(a, b)
```

```
{
```

```
    var mult = a*b;
```

```
    return mult;
```

```
}
```

```
document.write("The multiplication result of any two  
number is calculated using calmult() function<br/>");
```

```
var m = calmult(10, 20);
```

```
document.write("Multiplication Result = " + m );
```

```
</script>
```

# JavaScript Functions

- **JavaScript Variable Scope**
- In JavaScript, the variable's scope is basically the region of your program where it is defined. There are following two types of variable scopes available in JavaScript:
  - Local Variables
  - Global Variables
- **JavaScript Local Variables**
  - A local variable in JavaScript, will be visible only that function where it is declared.
- **JavaScript Global Variables**
  - A global variable in JavaScript, will be visible anywhere in the JavaScript program.

# JavaScript Functions

```
<script type="text/javascript">
```

```
var variable_scope_var3 = "this is global"; //
```

```
variable_scope_var3 is declared as a global variable
```

```
function variable_scope_fun3()
```

```
{
```

```
    var variable_scope_var3 = "this is local"; //
```

```
variable_scope_var3 is declared as a local variable
```

```
    document.getElementById('variable_scope_para3')
```

```
.innerHTML = variable_scope_var3;
```

```
}
```

```
</script>
```

# JavaScript Functions

- **JavaScript setTimeout() Method**
- The setTimeout() method in JavaScript is used to specify the time interval after which the code executes.
- You can use the setTimeout() method to delay the execution of specific code.
- **JavaScript setInterval() Method**
- The setInterval() method in JavaScript is used to execute the code after every specified time intervals.

# JavaScript Functions

- **JavaScript clearTimeout(timer) Method**
  - Deactivates or cancels the timer that is set using the setTimeout().
- **JavaScript clearInterval() Method**
  - Deactivates or cancels the timer that is set using the setTimeout().

# JavaScript Events

- Events in JavaScript, refer to actions that are detected by a JavaScript program when you perform a particular task.
- For example, [onclick event](#) is detected by the program when you click the mouse button. Here, the following code fragment shows how to create an event handler in JavaScript:

**onEvent = 'code to handle the event'**

# JavaScript Events

- **Events triggered by actions inside a HTML form**

Attribute	Description
<a href="#">onblur</a>	Fires the moment that the element loses focus
<a href="#">onchange</a>	Fires the moment when the value of the element is changed
<a href="#">oncontextmenu</a>	Script to be run when a context menu is triggered
<a href="#">onfocus</a>	Fires the moment when the element gets focus
<a href="#">oninput</a>	Script to be run when an element gets user input
onforminput	Triggers when input is provided on a form
onformchange	Triggers when a form changes
<a href="#">oninvalid</a>	Script to be run when an element is invalid
<a href="#">onreset</a>	Fires when the Reset button in a form is clicked
<a href="#">onsearch</a>	Fires when the user writes something in a search field (for <input="search">)
<a href="#">onselect</a>	Fires after some text has been selected in an element
<a href="#">onsubmit</a>	Fires when a form is submitted



# JavaScript Events

- Keyboard Events

Attribute	Description
<a href="#"><u>onkeydown</u></a>	Fires when a user is pressing a key
<a href="#"><u>onkeypress</u></a>	Fires when a user presses a key
<a href="#"><u>onkeyup</u></a>	Fires when a user releases a key

# JavaScript Events

- Mouse Events

Attribute	Description
<a href="#"><u>onclick</u></a>	Fires on a mouse click on the element
<a href="#"><u>ondblclick</u></a>	Fires on a mouse double-click on the element
<a href="#"><u>onmousedown</u></a>	Fires when a mouse button is pressed down on an element
<a href="#"><u>onmousemove</u></a>	Fires when the mouse pointer is moving while it is over an element
<a href="#"><u>onmouseout</u></a>	Fires when the mouse pointer moves out of an element
<a href="#"><u>onmouseover</u></a>	Fires when the mouse pointer moves over an element
<a href="#"><u>onmouseup</u></a>	Fires when a mouse button is released over an element
onmousewheel	<b>Deprecated.</b> Use the <a href="#"><u>onwheel</u></a> attribute instead
<a href="#"><u>onwheel</u></a>	Fires when the mouse wheel rolls up or down over an element

# JavaScript Events

- Mouse Drag Events

Attribute	Description
<a href="#"><u>ondrag</u></a>	Script to be run when an element is dragged
<a href="#"><u>ondragend</u></a>	Script to be run at the end of a drag operation
<a href="#"><u>ondragenter</u></a>	Script to be run when an element has been dragged to a valid drop target
<a href="#"><u>ondragleave</u></a>	Script to be run when an element leaves a valid drop target
<a href="#"><u>ondragover</u></a>	Script to be run when an element is being dragged over a valid drop target
<a href="#"><u>ondragstart</u></a>	Script to be run at the start of a drag operation
<a href="#"><u>ondrop</u></a>	Script to be run when dragged element is being dropped
<a href="#"><u>onscroll</u></a>	Script to be run when an element's scrollbar is being scrolled

# JavaScript Events

- **Media Events**
- Events triggered by medias like videos, images and audio (applies to all HTML elements, but is most common in media elements, like `<audio>`, `<embed>`, `<img>`, `<object>`, and `<video>`).

Attribute	Description
onabort	Script to be run on abort
oncanplay	Script to be run when a file is ready to start playing (when it has buffered enough to begin)
oncanplaythrough	Script to be run when a file can be played all the way to the end without pausing for buffering
oncuechange	Script to be run when the cue changes in a <code>&lt;track&gt;</code> element
ondurationchange	Script to be run when the length of the media changes
onemptied	Script to be run when something bad happens and the file is suddenly unavailable (like unexpectedly disconnects)

# JavaScript Events

- **Media Events**

Attribute	Description
onended	Script to be run when the media has reach the end (a useful event for messages like "thanks for listening")
onerror	Script to be run when an error occurs when the file is being loaded
onloadeddata	Script to be run when media data is loaded
onloadedmetadata	Script to be run when meta data (like dimensions and duration) are loaded
onloadstart	Script to be run just as the file begins to load before anything is actually loaded
onpause	Script to be run when the media is paused either by the user or programmatically
onplay	Script to be run when the media is ready to start playing

# JavaScript Events

- **Media Events**

Attribute	Description
onplaying	Script to be run when the media actually has started playing
onprogress	Script to be run when the browser is in the progress of getting the media data
onratechange	Script to be run each time the playback rate changes (like when a user switches to a slow motion or fast forward mode)
onseeked	Script to be run when the seeking attribute is set to false indicating that seeking has ended
onseeking	Script to be run when the seeking attribute is set to true indicating that seeking is active
onstalled	Script to be run when the browser is unable to fetch the media data for whatever reason
onsuspend	Script to be run when fetching the media data is stopped before it is completely loaded for whatever reason

# JavaScript Events

- **Media Events**

Attribute	Description
ontimeupdate	Script to be run when the playing position has changed (like when the user fast forwards to a different point in the media)
onvolumechange	Script to be run each time the volume is changed which (includes setting the volume to "mute")
onwaiting	Script to be run when the media has paused but is expected to resume (like when the media pauses to buffer more data)

# JavaScript Events

- **Window Event Attributes**
- Events triggered for the browser/window object (applies to the `<body>` tag):

Attribute	Description
<a href="#"><u>onafterprint</u></a>	Script to be run after the document is printed
<a href="#"><u>onbeforeprint</u></a>	Script to be run before the document is printed
<a href="#"><u>onbeforeunload</u></a>	Script to be run when the document is about to be unloaded
<a href="#"><u>onerror</u></a>	Script to be run when an error occurs
<a href="#"><u>onhashchange</u></a>	Script to be run when there has been changes to the anchor part of the a URL
<a href="#"><u>onload</u></a>	Fires after the page is finished loading
<code>onmessage</code>	Script to be run when the message is triggered
<a href="#"><u>onoffline</u></a>	Script to be run when the browser starts to work offline



# JavaScript Events

- **Window Event Attributes**

Attribute	Description
<a href="#"><u>ononline</u></a>	Script to be run when the browser starts to work online
onpagehide	Script to be run when a user navigates away from a page
<a href="#"><u>onpageshow</u></a>	Script to be run when a user navigates to a page
onpopstate	Script to be run when the window's history changes
<a href="#"><u>onresize</u></a>	Fires when the browser window is resized
onstorage	Script to be run when a Web Storage area is updated
<a href="#"><u>onunload</u></a>	Fires once a page has unloaded (or the browser window has been closed)

# JavaScript Image Map

- An image map in JavaScript is an image on a web page that provides various links to navigate to other web pages or some sections of the same web page. You can say those various links as hotspots.
- In an image map in JavaScript to provide hotspot link, you can use any shape such as rectangle, circle, or polygon.
- If you want to create a rectangular image map, then you need two different co-ordinates, such as top right and bottom left.
- If you want to create a circular image map, then you need centre co-ordinate. If you want to create a polygon image map, then you need different number of co-ordinates.
- And last, if you want to create a pentagon shape image map, then you need five co-ordinates.

# JavaScript Image Map

- Use MAP element to define image maps. Every image map has a unique name.
- Therefore, the name attribute is required in the MAP element. You can add usemap attribute to the IMG element to associate an image map with an image.
- You can also add events to the image map using the AREA element.

# JavaScript Animation

- Animation is a type of optical illusion where the rapid display of a sequence of images or frames of 2D or 3D artwork creates an illusion of movement. An animation movie or cartoon film is an example of animation.
- **Two types of Animation**
  - **Sprite Animation**
    - Sprite animation defines a rectangular image in which the parts of the image are made transparent where you want to show the background.
    - It has the ability to move an image over another image.
  - **Frame Animation**
    - Frame animation allows you to create fast slide shows. In this animation, you can create a number of slides or frames by making small changes in each frame.
    - As the slide sets are displayed in quick succession, therefore, the changes appears as motion.

# JavaScript Animation

- A high-quality animation contains the combination of both the animations, that is, the sprite animation and the frame animation.
- JavaScript provides timing [functions](#) to create animation which are listed here in the following table.

Function	Description
<a href="#">setTimeout(function, duration)</a>	This function is used to execute the code sometime in the future
<a href="#">setInterval(function, duration)</a>	This function is used to execute the code after specified intervals of time
<code>clearTimeout(setTimeout_variable)</code>	This function is used to clear the timer set by the <code>setTimeout()</code> function

# JavaScript Objects

- As you know that the JavaScript language is totally based on objects.
- In JavaScript, you have the following two options to create an object:
  - by creating a direct instance of object
  - by creating an object using a function
- A direct instance of an object in JavaScript, is created by using the **new** keyword.
- Here is the general form shows how to create an object in JavaScript by creating a direct instance of object:

**Obj = new object();**

# JavaScript Objects

- You are free to add properties and methods to an object in JavaScript, just by using the dot (.) followed by a property or method name as shown in the below code fragment:

**Obj.name="Deepak";**

**Obj.branch="CSE";**

**Obj.rollno=15;**

**Obj.getValue();**

- From the above code fragment, **Obj** is the newly created object, **name**, **branch**, and **rollno** are its properties, and **getValue()** is its method.

# JavaScript Objects

- Another way to create an instance of an object is by creating a function template of the object.
- After defining the function template for an object, you need to create an instance of the object by using the **new** keyword.
- A function template for an object is created by using the **function** keyword.
- You can also add properties to the function template by using **this** keyword.
- Example of creating function template.

```
<script type = "text/javascript">  
function book(title, author)  
{  
    this.title = title;  
    this.author = author;  
}
```



# JavaScript Objects

- **The Object() Constructor**
- A constructor is a function that creates and initializes an object. JavaScript provides a special constructor function called **Object()** to build the object. The return value of the **Object()** constructor is assigned to a variable.
- The variable contains a reference to the new object. The properties assigned to the object are not variables and are not defined with the **var** keyword.

# JavaScript Objects

**<head>**

**<title>User-defined objects</title>**

**<script type = "text/javascript">**

**var book = new Object(); // Create the object**

**book.subject = "Perl"; // Assign properties to the object**

**book.author = "Mohtashim";**

**</script>**

**</head>**

**<body>**

**<script type = "text/javascript">**

**document.write("Book name is : " + book.subject + "<br>");**

**document.write("Book author is : " + book.author + "<br>");**

**</script>**

**</body>**

# JavaScript Objects

- **Defining Methods for an Object**
- The previous examples demonstrate how the constructor creates the object and assigns properties.
- But we need to complete the definition of an object by assigning methods to it.

# JavaScript Objects

```
<script type = "text/javascript">  
  // Define a function which will work as a method  
  function addPrice(amount) {  
    this.price = amount;  
  }  
  function book(title, author) {  
    this.title = title;  
    this.author = author;  
    this.addPrice = addPrice; // Assign that method as property.  
  }  
</script>  
</head>  
<body>  
  <script type = "text/javascript">  
    var myBook = new book('Perl', 'Mohtashim');  
    myBook.addPrice(100);  
    document.write('Book title is : ' + myBook.title + "<br>");  
    document.write('Book author is : ' + myBook.author + "<br>");  
    document.write('Book price is : ' + myBook.price + "<br>");  
  </script>
```

# JavaScript Objects

- Built In Java Script Objects
  - Number Object
  - Array Object
  - String Object
  - Boolean Object
  - Math Object
  - RegExp Object
  - Date Object

# JavaScript Objects

- **JavaScript String**

- A string is a sequence of characters.
- All the strings in JavaScript are represent as instances of the String object.
- You can use String object to perform operations on stored string like to find the length of the string.
- **JavaScript String Object Properties**

Property	Description
constructor	gives the function, creates the prototype of a String object
length	gives the length of a string
prototype	adds properties and methods to an object

# JavaScript Objects

- **String Object Methods**

Method	Description
charAt()	gives the character in the specified index
charCodeAt()	gives the Unicode equivalence of the character in the specified index
concat()	joins two strings
fromCharCode()	converts Unicode to character
indexOf()	gives the position of the first occurrence of the specified character in a string
lastIndexOf()	gives the position of the last occurrence of a specified value in a string
match()	looks for the match between a regular expression and a string and returns the matches
quote()	writes quotes in JavaScript
replace()	searches for the match between a regular expression and a string, and replaces the matched substring with a new substring

# JavaScript Objects

- **String Object Methods**

Method	Description
search()	searches for a match between a regular expression and a string, and returns the position of the match
slice()	gives a part of a string as new string
split()	splits a string into substrings
substr()	extracts characters from a string beginning at the specified start index for the specified number of characters
substring()	gives characters in a string between two specified indices
toLowerCase()	converts a String value into a lowercase letter
toSource()	returns an object literal representing the specified object
toString()	returns a string representing the specified object
toUpperCase()	converts a string into uppercase letter
valueOf()	gives the primitive value of String object



# JavaScript Objects

- Here is an example of string in JavaScript:

**var carname = "Audi A6";**

**var carname = 'Audi A9';**

- Here, both the strings are equal. To use single quote in a string, then quote the string in double quote like this:

**var bupp = "It's a double quote";**

**var bupp = "He is a 'Computer Hacker'";**

**var bupp = 'He is an "Internet Hacker"';**

# JavaScript Objects

- **Find Length of String in JavaScript**

- To find the length of the string in JavaScript, then use JavaScript built-in property length. Here is an example:

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
var sln = txt.length;
```

- Here the variable **sln** will hold the value, 26, which is the length of the string **ABCDEFGHIJKLMNOPQRSTUVWXYZ**

- **JavaScript Special Characters**

- As you know that, JavaScript string must be in quotes. Now let's look at the following code:

```
var y = "He is a "Computer Hacker" from New Delhi."
```

- Here, the above string will be chopped to "He is a ".
- To avoid this problem, is simply to use \ escape character. Here is an example:

# JavaScript Objects

- **Wrapper methods that return a String Object**
  - [anchor\(\)](#) - Creates an HTML anchor that is used as a hypertext target.
  - [big\(\)](#) - Creates a string to be displayed in a big font as if it were in a <big> tag.
  - [blink\(\)](#) - Creates a string to blink as if it were in a <blink> tag.
  - [bold\(\)](#) - Creates a string to be displayed as bold as if it were in a <b> tag.
  - [fixed\(\)](#) - Causes a string to be displayed in fixed-pitch font as if it were in a <tt> tag
  - [fontcolor\(\)](#) - Causes a string to be displayed in the specified color as if it were in a <font color="color"> tag.

# JavaScript Objects

- **Wrapper methods that return a String Object**
  - [fontsize\(\)](#) - Causes a string to be displayed in the specified font size as if it were in a <font size="size"> tag.
  - [italics\(\)](#) - Causes a string to be italic, as if it were in an <i> tag.
  - [link\(\)](#) - Creates an HTML hypertext link that requests another URL.
  - [small\(\)](#) - Causes a string to be displayed in a small font, as if it were in a <small> tag.
  - [strike\(\)](#) - Causes a string to be displayed as struck-out text, as if it were in a <strike> tag.
  - [sub\(\)](#) - Causes a string to be displayed as a subscript, as if it were in a <sub> tag
  - [sup\(\)](#) - Causes a string to be displayed as a superscript, as if it were in a <sup> tag

# JavaScript Objects

Character Code	Output
\n	newline
\t	tab
\r	carriage return
\'	single quote
\"	double quote
\\	backslash
\f	form feed
\b	backspace

# JavaScript Objects

- **JavaScript Boolean**
- The JavaScript Boolean object is a wrapper class and a member of global objects.
- It is used to convert the non-boolean values into boolean values.
- Boolean object has following two values:
  - true
  - false
- **JavaScript Boolean Object Properties**
  - Constructor-- returns the function which has created the prototype of the Boolean object
  - Prototype-- allows you to add properties and methods to an object

# JavaScript Objects

- **JavaScript Boolean Object Methods**
  - `toString()` -- converts the boolean value into string and returns the
  - `stringValueOf()` -- returns the primitive value of a Boolean object
- Here is the general form to create a boolean object in JavaScript:  
**`var val = new Boolean(value);`**
- **JavaScript Boolean() Function**
- The JavaScript `Boolean()` function is used to find out if an expression or variable is true or not.
- Here is an example:  
**`Boolean(20 > 9)      // returns true`**
- The above expression will return true.

# JavaScript Objects

- **Numbers Object**
- The **Number** object represents numerical data, either integers or floating-point numbers.
- In general, you do not need to worry about **Number** objects because the browser automatically converts number literals to instances of the number class.
- The syntax for creating a **number** object is as follows –  
**var val = new Number(number);**
- In the place of number, if you provide any non-number argument, then the argument cannot be converted into a number, it returns **NaN** (Not-a-Number).

**var num= new Number('12', '12');**



# JavaScript Objects

- **Number Object Properties**

Property	Description
constructor	holds the value of the constructor function that has created the object
MAX VALUE	gives the maximum value
MIN VALUE	gives the minimum value
NEGATIVE INFINITY	represents the value of negative infinity
POSITIVE INFINITY	represents the value of infinity
prototype	adds properties and methods to the Number object

# JavaScript Objects

- **Number Object Methods**

Method	Description
toExponential(x)	converts a number into an exponential notation
toFixed(x)	rounds up a number to x digits after the decimal
toPrecision(x)	rounds up a number to a length of x digits
toString()	gives a string value for the Number object
valueOf()	gives a primitive value for the Number object

# JavaScript Objects

- **The Array Object**
- JavaScript arrays are basically used in storing multiple values in a single [variable](#).
- Here is an example, shows how to create an array in JavaScript:
- First, using array constructor  
**var fruits = new Array( "Guava", "Apple", "Orange" );**
- Second, using the array literal notation  
**var fruits = [ "Guava", "Apple", "Orange" ];**
- The fruits[0] represents the first element which is "Guava" here, and the fruits[2] is the third element which is "Orange" here.

# JavaScript Objects

- **Array Object Properties**

Property	Description
constructor	holds the value of the constructor function that has created the object
length	holds the number of elements in an array
prototype	adds properties and methods to the Array object

# JavaScript Objects

- **Array Object Methods**

Method	Description
concat()	joins two or more arrays
join()	joins all the elements of an array into a string
pop()	removes the last element of an array and returns that element
push()	adds new element as the last element and the returns the length of the new array
reverse()	reverses the order of list of element in an array
shift()	removes the first element of an array and then returns that element
slice()	selects a part of an array and then returns that part as a new array
sort()	sorts the elements of an array
splice()	adds or removes the elements of an array
toString()	converts an array into a string and then returns the string
unshift()	adds new elements to an array and then returns the new length
valueOf()	returns the primitive value of an Array object

# JavaScript Objects

- **Math Object**
- The JavaScript Math object is used to perform simple and complex arithmetic operations.
- **Math Object Properties**

Property	Description
E	Euler's number (approx. value is 2.718)
LN2	natural logarithm of 2 (approx. value is 0.693)
LN10	natural logarithm of 10 (approx. value is 2.302)
LOG2E	base-2 logarithm of E (approx. value is 1.442)
LOG10E	base-10 logarithm of E (approx. value is 0.434)
PI	numerical value of PI (approx. value is 3.142)
SQRT1_2	square root of 1/2 (approx. value is 0.707)
SQRT2	square root of 2 (approx. value is 1.414)

# JavaScript Objects

- **Math Object Methods**

Method	Description
abs(x)	gives the absolute value of x
acos(x)	gives arccosine of x (in radian)
asin(x)	gives arcsine of x (in radian)
atan(x)	gives the arctangent of x
atan2(y,x)	gives the arctangent of the quotient on dividing y and x
ceil(x)	rounds up x to the nearest bigger integer
cos(x)	gives cosine value of x
exp(x)	gives the value of $e^x$
floor(x)	rounds up x to the nearest smaller integer
log(x)	gives the natural logarithmic value of x

# JavaScript Objects

- **Math Object Methods**

Method	Description
<code>max(x,y,z,...,n)</code>	gives the highest number from the given list
<code>min(x,y,z,...,n)</code>	gives the lowest number from the given list
<code>pow(x,y)</code>	returns x to the power of y
<code>random()</code>	returns a random number between 0 and 1
<code>round(x)</code>	rounds up x to the nearest integer
<code>sin(x)</code>	gives the sine value of x
<code>sqrt(x)</code>	gives the square root of x
<code>tan(x)</code>	gives the tangent value of x



# JavaScript Objects

- **Date Object**
- The Date object in JavaScript is used to display a date on a Web page.
- **Date Object Properties**

Property	Description
constructor	holds the value of the constructor function that has created the object
prototype	adds properties and methods to the Date object

# JavaScript Objects

- **Date Object Methods**

Method	Description
<code>getDate()</code>	returns the day of the month (ranges from 1 to 31)
<code>getDay()</code>	returns the numerical equivalence of the day of a week (ranges from 0 to 6). 0 for Monday
<code>getFullYear()</code>	returns the numerical equivalence of the year (in 4 digits)
<code>getHours()</code>	returns the hours (ranges from 0 to 23)
<code>getMilliseconds()</code>	returns the milliseconds (ranges from 0 to 999)
<code>getMinutes()</code>	returns minutes (ranges from 0 to 59)
<code>getMonth()</code>	returns the numerical equivalence of month (ranges from 0 to 11)
<code>getSeconds()</code>	returns the seconds (ranges from 0 to 59)
<code>getTime()</code>	returns the number of milliseconds since midnight Jan 1, 1970

# JavaScript Objects

- **Date Object Methods**

Method	Description
getTimezoneOffset()	returns the difference of time in minutes between GMT and the local time
getUTCDate()	returns the day of the month (ranges from 1 to 31) as per the universal time
getUTCDay()	returns the numerical equivalence of the day of the week (ranges from 0 to 6) as per the universal time
getUTCFullYear()	returns the year in four digits as per the universal time
getUTCHours()	returns the hour (ranges from 0 to 23) as per the universal time
getUTCMilliseconds()	returns the milliseconds (ranges from 0 to 9999) as per the universal time
getUTCMinutes()	returns the minutes (ranges from 0 to 59) as per the universal time
getUTCMonth()	returns the numerical equivalence of month (ranges from 1 to 31) as per the universal time

# JavaScript Objects

- **Date Object Methods**

Method	Description
getUTCSeconds()	returns the seconds (ranges from 0 to 59) as per the universal time
Parse()	parses a date string and returns the number of millisecond since the midnight of January 1, 1970
setDate()	sets the day of a month (ranges from 1 to 31)
setFullYear()	sets the year in four digits
setHours()	sets the hours (ranges from 0 to 23)
setMilliseconds()	sets the milliseconds (ranges from 0 to 999)
setMinutes()	sets the minutes (ranges from 0 to 59)
setMonth()	sets the numerical equivalence of month (ranges from 0 to 11)
setSeconds()	sets the seconds (ranges 0 from 59)

# JavaScript Objects

- **Date Object Methods**

Method	Description
setTime()	sets a date and time by adding or subtracting specified milliseconds to/from midnight January 1, 1970
setUTCDate()	sets the day of the month (ranges from 1 to 1) as per the universal time
setUTCFullYear()	sets the year in four digits as per the universal time
setUTCHours()	sets the hours (ranges from 0 to 23) as per the universal time
setUTCMilliseconds()	sets the milliseconds (ranges from 0 to 999)
setUTCMinutes()	sets the minutes (ranges from 0 to 59) as per the universal time
setUTCMonth()	sets the numerical equivalence of month (ranges from 0 to 11) as per universal time
setUTCSeconds()	sets the seconds (ranges from 0 to 59) as per the universal time

# JavaScript Objects

- **Date Object Methods**

Method	Description
<code>toString()</code>	converts the date into a string
<code>toLocaleDateString()</code>	converts the date into a string as per local conventions
<code>toLocaleTimeString()</code>	converts the time into a string as per local convention
<code>toLocaleString()</code>	converts the Date object into a string as per local convention
<code>toString()</code>	converts the Date object into a string
<code>getTimeString()</code>	converts time into a string
<code>toUTCString()</code>	converts the Date object into a string as per the universal time
<code>UTC()</code>	holds the number of millisecond since the midnight of January 1, 1970, as per the universal time
<code>valueOf()</code>	returns the primitive value of the Date object

# JavaScript Objects

- **JavaScript RegExp**
- A regular expression is an object that describes a pattern of characters.
- The JavaScript **RegExp** class represents regular expressions, and both **String** and **RegExp** define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.
- Syntax
- A regular expression could be defined with the **RegExp ()** constructor, as follows –

**var pattern = new RegExp(pattern, attributes);**

**or simply**

**var pattern = /pattern/attributes;**

Here is the description of the parameters –

- **pattern** – A string that specifies the pattern of the regular expression or another regular expression.
- **attributes** – An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multi-line matches, respectively.

# JavaScript Objects

- Following are three types of flag/attributes of Java Script:
- **i**
  - Perform case-insensitive matching.
- **m**
  - Specifies that if the string has newline or carriage return characters, the ^ and \$ operators will now match against a newline boundary, instead of a string boundary
- **g**
  - Performs a global match that is, find all matches rather than stopping after the first match.
- Brackets
- Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.



# JavaScript Objects

Expression & Description	
<b>[...]</b>	Any one character between the brackets.
<b>[^...]</b>	Any one character not between the brackets.
<b>[0-9]</b>	It matches any decimal digit from 0 through 9.
<b>[a-z]</b>	It matches any character from lowercase <b>a</b> through lowercase <b>z</b> .
<b>[A-Z]</b>	It matches any character from uppercase <b>A</b> through uppercase <b>Z</b> .
<b>[a-Z]</b>	It matches any character from lowercase <b>a</b> through uppercase <b>Z</b> .

- The ranges shown above are general; you could also use the range **[0-3]** to match any decimal digit ranging from 0 through 3, or the range **[b-v]** to match any lowercase character ranging from **b** through **v**.

# JavaScript Objects

- **Quantifiers**
- The frequency or position of bracketed character sequences and single characters can be denoted by a special character.
- Each special character has a specific connotation. The +, \*, ?, and \$ flags all follow a character sequence.

# JavaScript Objects

- JavaScript RegExp Object Quantifiers

Quantifier	Description
n+	specifies a string that contains at least one n
n*	specifies a string that contains zero or more occurrences of n
n?	specifies a string that contains zero or one occurrence of n
n{a}	specifies a string that contains n, a number of times
n{a, b}	specifies a string that contains n, a to b number of times
n{a, }	specifies a string that contains n, a or more number of times
n\$	specifies a string that ends with n
^n	specifies a string that begins with n
?=n	specifies a string that is followed by a specific string n
?!n	specifies a string that is not followed by a specific string n

# JavaScript Objects

- **JavaScript RegExp Object Properties**

Property	Description
global	refers that the g modifier is set
ignoreCase	refers that the i modifier is set
lastIndex	refers the index to start the next match
multiline	refers that the m modifier is set
source	refers the text of the RegExp pattern

# JavaScript Objects

- **JavaScript RegExp Object Methods**

Method	Description
compile()	compiles the RegExp object
exec()	tests for a match in a string and returns a result array
test()	tests for a match in a string and returns true or false