**UNIT-1**

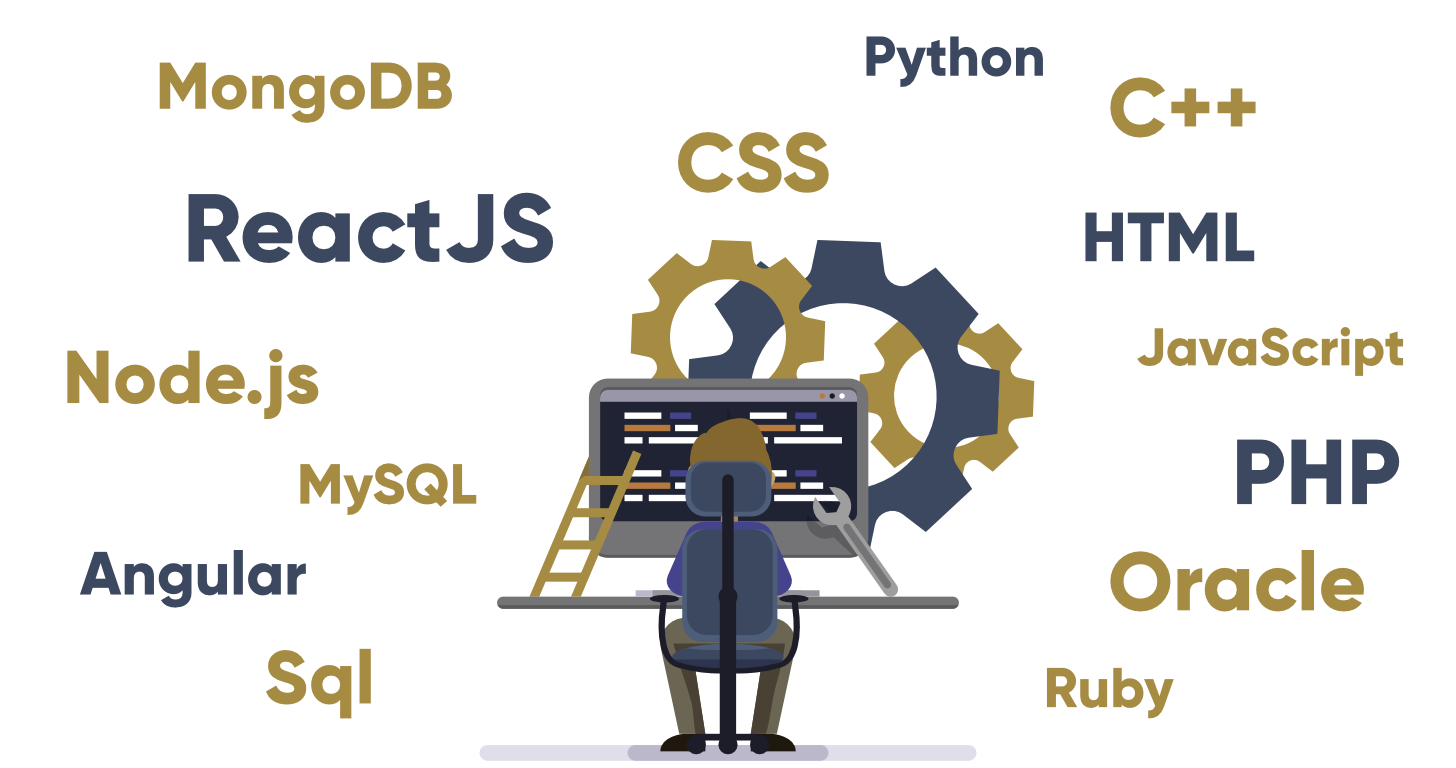
**Introduction:**

**Full stack:** In technology development, **full stack** refers to an **entire** computer system or application from the front end to the back end and the code that connects the two. The back end of a computer system encompasses “behind-the-scenes” technologies such as the database and operating system.

## Full Stack Web Developer: A full stack web developer is a person who can develop both client and server software. They are proficient in both front-end and back-end languages and frameworks, as well as in server, network and hosting environments.

## (or)

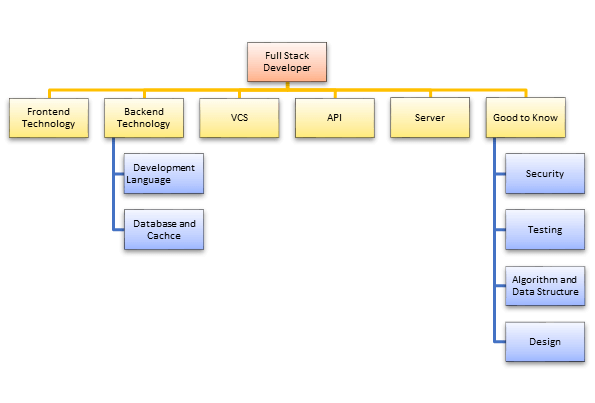
**Full Stack Developer** is an engineer who works on both client-side and server-side of the software application. This type of developer works on the Full Stack of a software application meaning Front end development, Back end development, Database, Server, API, and version controlling systems. Hence, the name "Full Stack" Developer.Full stack developer translates user requirements into the overall architecture and implement the new systems.



**Why Do You Need a Full-Stack Developer?**

* Full stack developer helps you to keep every part of the system running smoothly
* Full stack developer can provide help to everyone in the team and greatly reduce the time and technical costs of team communication
* If one person plays different roles, it saves your company's personnel, infrastructure and operational cost.

**Full Stack Developer Skills You Need to Know:**



**1) Front-end technology**

Full stack developer should be master of essential front-end technologies like HTML5, CSS3, JavaScript. Knowledge of third-party libraries like jQuery, LESS, [Angular](https://www.guru99.com/angularjs-tutorial.html) and [ReactJS](https://www.guru99.com/reactjs-tutorial.html) is desirable

**2) Development Languages**

Full stack engineer should know at least one server-side programming languages like Java, Python, Ruby, .Net etc.

**3) Database and cache**

Knowledge of various DBMS technology is another important need of full stack developer. MySQL, MongoDB, Oracle, SQLServer are widely used for this purpose. Knowledge of caching mechanisms like varnish, Memcached, Redis is a plus.

**4) Basic design ability**

In order to become a successful Full Stack web developer, the knowledge of designing is also recommended. Moreover, the person should know the principle of basic prototype design and UI /UX design.

**5) Server**

Exposure to handling Apache or nginx servers is desirable. A good background in Linux helps tremendously in administering servers.

**6) Version control system (VCS)**

A version control system allows full stack developers to keep track of all the changes made in the codebase. The knowledge of **Git** helps full stack developers to understand how to get the latest code, update parts of the code, make changes in other developer's code without breaking things.

**7) Working with API (REST & SOAP):**

Knowledge of web services or API is also important for full stack developers. Knowledge of creations and consumption of REST and SOAP services is desirable.

**SOAP** (Simple Object Access Protocol) is a standards-based web services access protocol that has been around for a long time. ... **REST** (Representational State Transfer) is another standard, made in response to **SOAP's** shortcomings. It seeks to fix the problems with **SOAP** and provide a simpler method of accessing web services.

**Other important points:**

1. Ability to write quality **unit tests**
2. He or she should have a complete understanding of automated processes for building testing, document, and deploying it at scale
3. An awareness of **security** concerns is important, as each layer has its own vulnerabilities
4. Knowledge of **Algorithms** and data structures is also an essential need for professional full stack developers.

**Java Full Stack Developer** :

A **Java Full Stack Developer** is a developer who has expertise and deep knowledge of frameworks and tools used in Java full stack development like Core Java, servlets, APIs, database, web architecture, etc. A Full Stack Java developer can build whole Java applications including front end, back-end, database, APIs, server and version control.

**Full Stack JavaScript:**

JavaScript has been around for over 20 years. It is the dominant programming language in web development.

In the beginning JavaScript was a language for the web client (browser). Then came the ability to use JavaScript on the web server (with Node.js).

Today the hottest buzzword is "Full Stack JavaScript".

The idea of "Full Stack JavaScript" is that all software in a web application, both client side and server side, should be written using JavaScript only.

**What is a Software Stack?**

Software stack is a collection of the programs which are used together to produce a specific result. It includes an operating system and its application. For example, a smartphone software stack includes OS along with the phone app, web browsers, and default applications.

**Popular Stacks:**

* LAMP stack: JavaScript - Linux - Apache - MySQL - PHP
* MERN: JavaScript - MongoDB - Express - React Js - Node.js
* MEAN stack: JavaScript - MongoDB - Express - AngularJS - Node.js
* Django stack: JavaScript - Python - Django - MySQL
* Ruby on Rails: JavaScript - Ruby - SQLite - Rails

**LAMP Stack:**

LAMP is a widely used model for web service stacks. Its name "LAMP" is an acronym of four open-source components.

* L= Linux: An open source operating system
* A= Apache: Widely used web server software
* M= MySQL: Popular open source database
* P=PHP: Server-side open source scripting language
* Many popular websites and web applications run on LAMP stack, Example: Facebook.

**MERN Stack:**

MERN is a collection of JavaScript-based technologies:

* M=MongoDB: Popular [nosql](https://www.guru99.com/nosql-tutorial.html) database
* E=Express: Light and portable web program framework
* R=React: A javascript library for building user interfaces
* N=Node.js: A server-side JavaScript run time

This stack currently in the huge demand as it is widely used to develop web applications.

**MEAN Stack:**

[MEAN](https://www.guru99.com/mean-stack-developer.html) Stack Application Development is witnessing a growing trend in usage. MEAN is an abbreviation of:

* M = MongoDB: nosql Database
* E = Express: Easy to use light and portable web program framework
* A = Angular.js: Robust framework for developing HTML5 and JavaScript- web programs
* N = Node.is: a server-side JavaScript run time

**What Does a Full Stack Developer Do?**

As a full stack developer, you may be involved in following activities:

* Translate user requirements into the overall architecture and implementation of new systems
* Manage Project and coordinate with the Client
* Write backend code in [Ruby](https://www.guru99.com/ruby-on-rails-tutorial.html), [Python](https://www.guru99.com/python-tutorials.html), [Java](https://www.guru99.com/java-tutorial.html), [PHP](https://www.guru99.com/php-tutorials.html) languages
* Writing optimized front end code HTML and [JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)
* Understand, create and debug database related queries
* Create test code to validate the applicaition against client requirement.
* Monitor the performance of web applications & infrastructure
* Troubleshooting web application with a fast and accurate a resolution

**Advantages:**

The advantage of being a full stack web developer is:

* You can master all the techniques involved in a development project
* You can make a prototype very rapidly
* You can provide help to all the team members
* You can reduce the cost of the project
* You can reduce the time used for team communication
* You can switch between front and back end development based on requirements
* You can better understand all aspects of new and upcoming technologies

**Disadvantages:**

* The solution chosen can be wrong for the project
* The solution chosen can be dependent on developer skills
* The solution can generate a key person risk
* Being a full stack developer is increasingly complex

**Technology related to full stack development:**

**Front end:** It is the visible part of website or web application which is responsible for user experience. The user directly interacts with the front end portion of the web application or website.

HTML,CSS,BOOTSTRAP,W3.CSS,JAVASCRIPT,ES5,HTML DOM,JSON,XML,JQUERY,ANGULAR,REACT,BACKBONE.JS,EMBER.JS,REDUX,STORYBOOK,GRAPHQL,METEOR.JS,GRUNT,GULF.

[**HTML:**](https://www.geeksforgeeks.org/html-tutorials/) HTML stands for Hyper Text Markup Language. It is used to design the front end portion of web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text documentation within tag which defines the structure of web pages.

[**CSS:**](https://www.geeksforgeeks.org/css-tutorials/) Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

[**JavaScript:**](https://www.geeksforgeeks.org/javascript-tutorial/) JavaScript is a famous scripting language used to create the magic on the sites to make the site interactive for the user. It is used to enhancing the functionality of a website to running cool games and web-based software.

**Front End Frameworks and Libraries:**

[AngularJS](https://www.geeksforgeeks.org/category/web-technologies/angular-js/)**:** AngularJs is a JavaScript open source front-end framework that is mainly used to develop single page web applications(SPAs). It is a continuously growing and expanding framework which provides better ways for developing web applications. It changes the static HTML to dynamic HTML. It is an open source project which can be freely used and changed by anyone. It extends HTML attributes with Directives, and data is bound with HTML.

[React.js](https://www.geeksforgeeks.org/react-js-introduction-working/)**:** React is a declarative, efficient, and flexible JavaScript library for building user interfaces. ReactJS is an open-source, component-based front end library responsible only for the view layer of the application. It is maintained by Facebook.

[Bootstrap](https://www.geeksforgeeks.org/bootstrap-tutorials/)**:** Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites.

[jQuery](https://www.geeksforgeeks.org/jquery-tutorials/)**:** jQuery is an open source JavaScript library that simplifies the interactions between an HTML/CSS document, or more precisely the Document Object Model (DOM), and JavaScript. Elaborating the terms, jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.

**SASS:** It is the most reliable, mature and robust CSS extension language. It is used to extend the functionality of an existing CSS of a site including everything from variables, inheritance, and nesting with ease.Some other libraries and frameworks are: Semantic-UI, Foundation, Materialize, Backbone.js, Express.js, Ember.js etc.

**Other Important Points:**

* + Work with text editors to use shortcuts and its facilities i.e. Visual studio, Atom, Sublime etc.
  + Make UI responsible using grid system.
  + Git and git commands like init, add, commit etc for version control and to work with team.
  + Other tools like npm & yarn package managers, sass css pre-processor, browser DevTools i.e. chrome devtools.
  + Understand using HTTP, JSON, GraphQL APIs to fetch data using axios or other tools.
  + It also requires some design skill to make layout and look better.

**Back end:** It refers to the server-side development of web application or website with a primary focus on how the website works. It is responsible for managing the database through queries and APIs by client-side commands. This type of website mainly consists of three parts front end, back end, and database.

PHP,ASP,C++,C#,JAVA,PYTHON,NODE.JS,EXPRESS.JS,RUBY,REST,GO,SQL,MONGODB,FIREBASE.COM,SASS,LESS,PARSE.COM,

PASS(AZURE,HEROKU)

The back end portion is built by using some libraries, frameworks, and languages which are discussed below:

* [**PHP:**](https://www.geeksforgeeks.org/php/) PHP is a server-side scripting language designed specifically for web development. Since, PHP code executed on server side so it is called server side scripting language.
* [**C++**](https://www.geeksforgeeks.org/c-plus-plus/) It is a general purpose programming language and widely used now a days for competitive programming. It is also used as backend language.
* [**Java:**](https://www.geeksforgeeks.org/java/) Java is one of the most popular and widely used programming language and platform. It is highly scalable. Java components are easily available.
* [**Python:**](https://www.geeksforgeeks.org/python-programming-language/) Python is a programming language that lets you work quickly and integrate systems more efficiently.
* [**JavaScript:**](https://www.geeksforgeeks.org/javascript-tutorial/) Javascript can be used as both (front end and back end) programming languages.
  + [Node.js](https://www.geeksforgeeks.org/introduction-to-nodejs/)**:** Node.js is an open source and cross-platform runtime environment for executing JavaScript code outside of a browser. You need to remember that NodeJS is not a framework and it’s not a programming language. Most of the people are confused and understand it’s a framework or a programming language. We often use Node.js for building back-end services like APIs like Web App or Mobile App. It’s used in production by large companies such as Paypal, Uber, Netflix, Wallmart and so on.
  + **Back End Frameworks:** The list of back end frameworks are: Express, Django, Rails, Laravel, Spring etc.
  + The other back end program/scripting languages are: C#, Ruby, REST, GO etc.

**Other Important Points:**

* + Structuring the data in efficient way.
  + Handle request-response of APIs for storing and retrieve data.
  + Security of data is important.

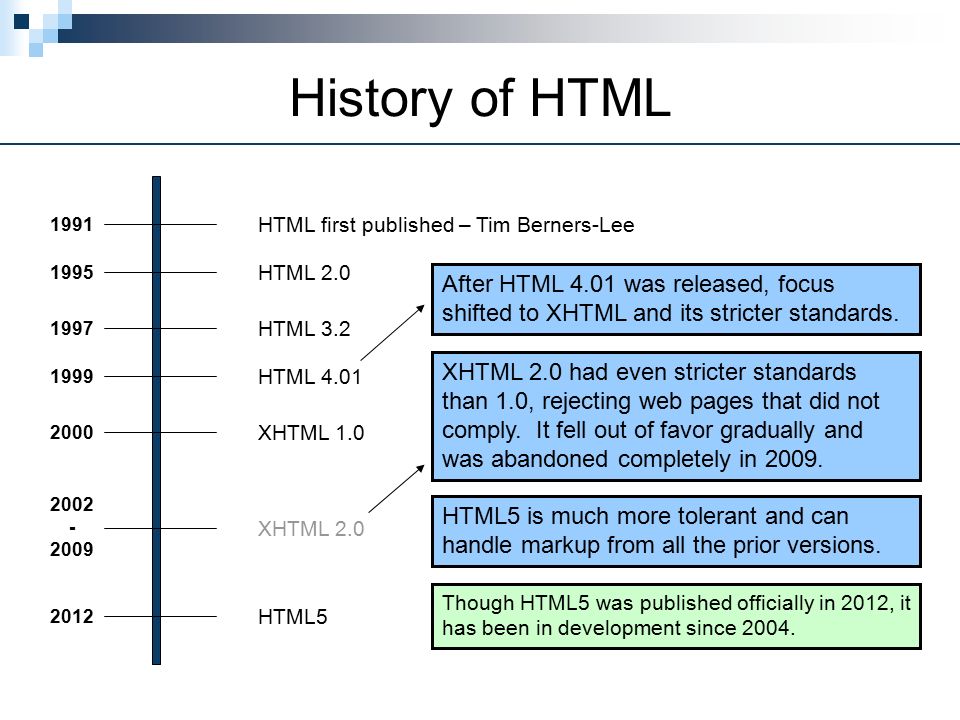
[Database](https://www.geeksforgeeks.org/dbms/)**:** Database is the collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc.

* + **Oracle:** Oracle database is the collection of data which is treated as a unit. The purpose of this database is to store and retrieve information related to the query. It is a database server and used to manages information.
  + [MongoDB](https://www.geeksforgeeks.org/mongodb-an-introduction/)**:** MongoDB, the most popular NoSQL database, is an open source document-oriented database. The term ‘NoSQL’ means ‘non-relational’. It means that MongoDB isn’t based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data.
  + [Sql](https://www.geeksforgeeks.org/sql-tutorial/)**:** Structured Query Language is a standard Database language which is used to create, maintain and retrieve the relational database.

**Full Stack Developer is "jack of all trade, master on none".**

**Getting Started with HTML**

* HTML stands for Hyper Text Markup Language
* HTML is the standard markup language for creating Web pages
* HTML describes the structure of a Web page
* HTML consists of a series of elements
* HTML elements tell the browser how to display the content
* HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.
* HTML **elements** are the building blocks of HTML pages.
* HTML elements are represented by **<> tags.**
* **An HTML file is simply a text file saved with an .html or .htm extension.**



**i) HTML Documents:**

All HTML documents must start with a document type declaration: <!DOCTYPE html>.

The HTML document itself begins with <html> and ends with </html>.

The visible part of the HTML document is between <body> and </body>.

### Example

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Heading</h1>  
<p>My first paragraph.</p>  
  
</body>  
</html>

**The <!DOCTYPE> Declaration:**

The <!DOCTYPE> declaration represents the document type, and helps browsers to display web pages correctly.

It must only appear once, at the top of the page (before any HTML tags).

The <!DOCTYPE> declaration is not case sensitive.

The <!DOCTYPE> declaration for HTML5 is:

<!DOCTYPE html>

**ii) HTML Headings**

HTML headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading:

### Example

<h1>This is heading 1</h1>  
<h2>This is heading 2</h2>  
<h3>This is heading 3</h3>

**iii) HTML Paragraphs**

Paragraph element is used to publish text on the web pages.

Paragraphs are defined with the <p> tag. Paragraph tag is a very basic and typically the first tag you will need to publish your text on the web pages.

HTML paragraphs are defined with the <p> tag:

### Example

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

**iv) HTML Links**

HTML links are defined with the <a> tag:

A link or hyperlink could be a word, group of words, or image.

<a href="*url*">Link text</a>

Anything between the opening <a> tag and the closing </a> tag becomes the part of the link that the user sees and clicks in a browser.

### Example

<a href="https://www.w3schools.com">This is a link</a>

The link's destination is specified in the href attribute.

Attributes are used to provide additional information about HTML elements.

**v) HTML Images**

HTML images are defined with the <img> tag.

The source file (src), alternative text (alt), width, and height are provided as attributes:

### Example

<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">

# vi) HTML Elements:

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

The HTML **element** is everything from the start tag to the end tag:

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

Examples of some HTML elements:

<h1>My First Heading</h1>

<p>My first paragraph.</p>

**Nested HTML Elements:**

HTML elements can be nested (this means that elements can contain other elements).

All HTML documents consist of nested HTML elements.

The following example contains four HTML elements (<html>, <body>, <h1> and <p>):

### Example

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Heading</h1>  
<p>My first paragraph.</p>  
  
</body>  
</html>

The <html> element is the root element and it defines the whole HTML document.

It has a start tag <html> and an end tag </html>.

Then, inside the <html> element there is a <body> element:

<body>  
  
<h1>My First Heading</h1>  
<p>My first paragraph.</p>  
  
</body>

The <body> element defines the document's body.

It has a start tag <body> and an end tag </body>.

Then, inside the <body> element there are two other elements: <h1> and <p>:

<h1>My First Heading</h1>  
<p>My first paragraph.</p>

The <h1> element defines a heading.

It has a start tag <h1> and an end tag </h1>:

<h1>My First Heading</h1>

The <p> element defines a paragraph.

It has a start tag <p> and an end tag </p>:

<p>My first paragraph.</p>

**vii) Empty HTML Elements:**

HTML elements with no content are called empty elements.

The <br> tag defines a line break, and is an empty element without a closing tag:

### Example:

<p>This is a <br> paragraph with a line break.</p>

**HTML is Not Case Sensitive:**

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

The HTML standard does not require lowercase tags, but W3C **recommends** lowercase in HTML, and **demands** lowercase for stricter document types like XHTML.

**HTML Attributes:** Attributes define additional characteristics or properties of the element such as width and height of an image. Attributes are always specified in the start tag (or opening tag) and usually consists of name/value pairs like name="value". Attribute values should always be enclosed in quotation marks.

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

**a) The href Attribute:**

The <a> tag defines a hyperlink. The href attribute specifies the URL of the page the link goes to:

### Example

<a href="https://www.w3schools.com">Visit W3Schools</a>

**b) The src Attribute:**

The <img> tag is used to embed an image in an HTML page. The src attribute specifies the path to the image to be displayed:

### Example

<img src="img\_girl.jpg">

**c) The width and height Attributes:**

The <img> tag should also contain the width and height attributes, which specifies the width and height of the image (in pixels):

### Example

<img src="img\_girl.jpg" width="500" height="600">

**d) The alt Attribute:**

The required alt attribute for the <img> tag specifies an alternate text for an image, if the image for some reason cannot be displayed. This can be due to slow connection, or an error in the src attribute, or if the user uses a screen reader.

### Example

<img src="img\_girl.jpg" alt="Girl with a jacket">

**1. General Purpose Attributes:**

There are some attributes, such as id, title, class, style, etc. that you can use on the majority of HTML elements. The following section describes their usages.

### a) The id Attribute:

The id attribute is used to give a unique name or identifier to an element within a document. This makes it easier to select the element using CSS or JavaScript.

#### Example

<input type="text" id="firstName">

<div id="container">Some content</div>

<p id="infoText">This is a paragraph.</p>

**Note:** The id of an element must be unique within a single document. No two elements in the same document can be named with the same id, and each element can have only one id.

### b) The class Attribute:

Like id attribute, the class attribute is also used to identify elements. But unlike id, the class attribute does not have to be unique in the document. This means you can apply the same class to multiple elements in a document.

#### Example

<input type="text" class="highlight">

<div class="box highlight">Some content</div>

<p class="highlight">This is a paragraph.</p>

**c) The style Attribute:**

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

### Example

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

**d) The lang Attribute:**

You should always include the lang attribute inside the <html> tag, to declare the language of the Web page. This is meant to assist search engines and browsers.

The following example specifies English as the language:

<!DOCTYPE html>  
<html lang="en">  
<body>  
...  
</body>  
</html>

**e) The title Attribute:**

The title attribute defines some extra information about an element.

The value of the title attribute will be displayed as a tooltip when you mouse over the element:

### Example

<p title="I'm a tooltip">This is a paragraph.</p>

**HTML Headings:**

HTML headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

### Example

<h1>Heading 1</h1>  
<h2>Heading 2</h2>  
<h3>Heading 3</h3>  
<h4>Heading 4</h4>

<h5>Heading 5</h5>  
<h6>Heading 6</h6>

**HTML Paragraphs:**

The HTML <p> element defines a paragraph.

A paragraph always starts on a new line, and browsers automatically add some white space (a margin) before and after a paragraph.

### Example

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

**HTML Horizontal Rules:**

The <hr> tag defines a thematic break in an HTML page, and is most often displayed as a horizontal rule.

The <hr> element is used to separate content (or define a change) in an HTML page:

### Example

<h1>This is heading 1</h1>  
<p>This is some text.</p>  
<hr>  
<h2>This is heading 2</h2>  
<p>This is some other text.</p>  
<hr>

The <hr> tag is an empty tag, which means that it has no end tag.

**HTML Line Breaks**

The HTML <br> element defines a line break.

Use <br> if you want a line break (a new line) without starting a new paragraph:

### Example

<p>This is<br>a paragraph<br>with line breaks.</p>

The <br> tag is an empty tag, which means that it has no end tag.

**The HTML <pre> Element:**

Sometimes, using &nbsp;, <br>, etc. for managing spaces isn't very convenient. Alternatively, you can use the <pre> tag to display spaces, tabs, line breaks, etc. exactly as written in the HTML file. It is very helpful in presenting text where spaces and line breaks are important like poem or code.

The HTML <pre> element defines preformatted text.

The text inside a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks:

### Example

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

# HTML Styles:

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

**The HTML Style Attribute:**

Setting the style of an HTML element, can be done with the style attribute.

The HTML style attribute has the following syntax:

<tagname style="property:value;">

The ***property*** is a CSS property. The ***value*** is a CSS value.

**Background Color:**

The CSS background-color property defines the background color for an HTML element.

### Example

Set the background color for a page to powderblue:

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

**Text Color:**

The CSS color property defines the text color for an HTML element:

### Example

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

**Fonts:**

The CSS font-family property defines the font to be used for an HTML element:

### Example

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

**Text Size:**

The CSS font-size property defines the text size for an HTML element:

### Example

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

**Text Alignment:**

The CSS text-align property defines the horizontal text alignment for an HTML element:

### Example

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

## HTML Formatting Elements:

Formatting elements were designed to display special types of text:

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

## HTML <b> and <strong> Elements:

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

### Example

<b>This text is bold</b>

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

### Example

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

**HTML <i> and <em> Elements:**

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

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

### Example

<i>This text is italic</i>

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

### Example

<em>This text is emphasized</em>

## HTML <small> Element:

The HTML <small> element defines smaller text:

### Example

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

## HTML <mark> Element:

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

### Example

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

## HTML <del> Element:

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

### Example

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

## HTML <ins> Element:

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

### Example

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

## HTML <sub> Element:

The HTML <sub> element defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H2O:

### Example

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

## HTML <sup> Element:

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

### Example

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

**HTML Lists:** HTML lists are used to present list of information in well formed and semantic way. There are three different types of list in HTML and each one has a specific purpose and meaning.

* **Unordered list** — Used to create a list of related items, in no particular order.
* **Ordered list** — Used to create a list of related items, in a specific order.
* **Description list** — Used to create a list of terms and their descriptions.

**HTML Unordered Lists:**

An unordered list created using the <ul> element, and each list item starts with the <li> element.

The list items in unordered lists are marked with bullets.

#### Example

<ul>

<li>Chocolate Cake</li>

<li>Black Forest Cake</li>

<li>Pineapple Cake</li>

</ul>

**output :**

* Chocolate Cake
* Black Forest Cake
* Pineapple Cake

**HTML Ordered Lists:** An ordered list created using the <ol> element, and each list item starts with the <li> element. Ordered lists are used when the order of the list's items is important.

The list items in an ordered list are marked with numbers.

#### Example

<ol>

<li>Fasten your seatbelt</li>

<li>Starts the car's engine</li>

<li>Look around and go</li>

</ol>

**output: 1.** Fasten your seatbelt

2.Starts the car's engine

3.Look around and go

**HTML Description Lists:**

A description list is a list of items with a description or definition of each item.

The description list is created using <dl> element. The <dl> element is used in conjunction with the <dt> element which specify a term, and the <dd> element which specify the term's definition.

Browsers usually render the definition lists by placing the terms and definitions in separate lines, where the term's definitions are slightly indented.

#### Example

<dl>

<dt>Bread</dt>

<dd>A baked food made of flour.</dd>

<dt>Coffee</dt>

<dd>A drink made from roasted coffee beans.</dd>

</dl>

**output:**

Bread

A baked food made of flour.

Coffee

A drink made from roasted coffee beans.

**HTML Form:**

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called controls like inputbox, checkboxes, radio-buttons, submit buttons, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for further processing.

The [<form>](https://www.tutorialrepublic.com/html-reference/html-form-tag.php) tag is used to create an HTML form.

#### Example

<form>

<label>Username: <input type="text"></label>

<label>Password: <input type="password"></label>

<input type="submit" value="Submit">

</form>

**Input Element:**

This is the most commonly used element within HTML forms.

It allows you to specify various types of user input fields, depending on the type attribute. An input element can be of type text field, password field, checkbox, radio button, submit button, reset button, file select box.

**Text Fields:**

Text fields are one line areas that allow the user to input text.

Single-line text input controls are created using an <input> element, whose type attribute has a value of text.

#### Example

<form>

<label for="username">Username:</label>

<input type="text" name="username" id="username">

</form>

**Password Field:**

Password fields are similar to text fields. The only difference is; characters in a password field are masked, i.e. they are shown as asterisks or dots. This is to prevent someone else from reading the password on the screen. This is also a single-line text input controls created using an <input> element whose type attribute has a value of password.

#### Example

<form>

<label for="user-pwd">Password:</label>

<input type="password" name="user-password" id="user-pwd">

</form>

## Radio Buttons:

Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an <input> element whose type attribute has a value of radio.

<form>

<input type="radio" name="gender" id="male">

<label for="male">Male</label>

<input type="radio" name="gender" id="female">

<label for="female">Female</label>

</form>

## Checkboxes:

Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an <input> element whose type attribute has a value of checkbox.

#### Example

<form>

<input type="checkbox" name="sports" id="soccer">

<label for="soccer">Soccer</label>

<input type="checkbox" name="sports" id="cricket">

<label for="cricket">Cricket</label>

<input type="checkbox" name="sports" id="baseball">

<label for="baseball">Baseball</label>

</form>

## Textarea:

Textarea is a multiple-line text input control that allows a user to enter more than one line of text. Multi-line text input controls are created using an <textarea> element.

#### Example

<form>

<label for="address">Address:</label>

<textarea rows="3" cols="30" name="address" id="address"></textarea>

</form>

## Submit and Reset Buttons:

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's action attribute to process the submitted data.

A reset button resets all the forms control to default values.

<form action="action.php" method="post">

<label for="first-name">First Name:</label>

<input type="text" name="first-name" id="first-name">

<input type="submit" value="Submit">

<input type="reset" value="Reset">

</form>

**HTML Links - Hyperlinks:**

HTML links are hyperlinks.

You can click on a link and jump to another document.

When you move the mouse over a link, the mouse arrow will turn into a little hand.

**HTML Links - Syntax:**

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

<a href="*url*">*link text*</a>

The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

The link text is the part that will be visible to the reader.

Clicking on the link text, will send the reader to the specified URL address.

**Example:**

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

## HTML Images Syntax:

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

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

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

The <img> tag has two required attributes:

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

### Syntax

<img src="*url*" alt="alternatetext">

**The src Attribute:**

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

### Example:

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

**The alt Attribute:**

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

The value of the alt attribute should describe the image:

### Example

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

## Image Size - Width and Height:

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

### Example

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

**Define an HTML Table:**

The <table> tag defines an HTML table.

Each table row is defined with a <tr> tag. Each table header is defined with a <th> tag. Each table data/cell is defined with a <td> tag.

By default, the text in <th> elements are bold and centered.

By default, the text in <td> elements are regular and left-aligned.

### Example

A simple HTML table:

<table style="width:100%">  
  <tr>  
    <th>Firstname</th>  
    <th>Lastname</th>  
    <th>Age</th>  
  </tr>  
  <tr>  
    <td>Jill</td>  
    <td>Smith</td>  
    <td>50</td>  
  </tr>  
  <tr>  
    <td>Eve</td>  
    <td>Jackson</td>  
    <td>94</td>  
  </tr>  
</table>

## HTML Table - Add a Border:

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

### Example

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

## HTML Table - Collapsed Borders:

To let the borders collapse into one border, add the CSS border-collapse property:

### Example

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

## HTML Table - Add Cell Padding:

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

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

To set the padding, use the CSS padding property:

### Example

th, td {  
  padding: 15px;  
}

## HTML Table - Left-align Headings:

By default, table headings are bold and centered.

To left-align the table headings, use the CSS text-align property:

### Example

th {  
  text-align: left;  
}

## HTML Table - Add Border Spacing:

Border spacing specifies the space between the cells.

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

### Example

table {  
  border-spacing: 5px;  
}

# HTML Lists:

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

### Example

An unordered HTML list:

* Item
* Item
* Item
* Item

An ordered HTML list:

1. First item
2. Second item
3. Third item
4. Fourth item

## Unordered HTML List:

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

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

### Example

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

## Ordered HTML List:

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

The list items will be marked with numbers by default:

### Example

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

## HTML Description Lists:

HTML also supports description lists.

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

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

### Example

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

# HTML Block and Inline Elements:

Every HTML element has a default display value, depending on what type of element it is.

There are two display values: block and inline.

**Block-level Elements:**

A block-level element always starts on a new line.

A block-level element always takes up the full width available (stretches out to the left and right as far as it can).

A block level element has a top and a bottom margin, whereas an inline element does not.

The <div> element is a block-level element.

### Example

<div>Hello World</div>

## Inline Elements:

An inline element does not start on a new line.

An inline element only takes up as much width as necessary.

This is a <span> element inside a paragraph.

### Example

<span>Hello World</span>

## The <div> Element:

The <div> element is often used as a container for other HTML elements.

The <div> element has no required attributes, but style, class and id are common.

### Example

<div style="background-color:black;color:white;padding:20px;">  
  <h2>London</h2>  
  <p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>  
</div>

## The <span> Element:

The <span> element is an inline container used to mark up a part of a text, or a part of a document.

The <span> element has no required attributes, but style, class and id are common.

When used together with CSS, the <span> element can be used to style parts of the text:

### Example

<p>My mother has <span style="color:blue;font-weight:bold">blue</span> eyes and my father has <span style="color:darkolivegreen;font-weight:bold">dark green</span> eyes.</p>

**HTML5**

HTML5 is the latest and most enhanced version of HTML. Technically, HTML is not a programming language, but rather a markup language.

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.

## 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.

**New Features**:HTML5 introduces a number of new elements and attributes that can help you in building modern websites.

* **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.
* **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.

**Why use HTML5:**It is enriched with advance features which makes it easy and interactive for designer/developer and users.

It allows you to play a video and audio file.

It allows you to draw on a canvas.

It facilitate you to design better forms and build web applications that work offline.

It provides you advance features for that you would normally have to write JavaScript to do.

The most important reason to use HTML 5 is, we believe it is not going anywhere. It will be here to serve for a long time according to W3C recommendation.

**HTML5 Syntax:**

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.

HTML5 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.

**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.

 DOCTYPE declaration appears at the top of a web page before all other elements. According to the HTML specification or standards, every HTML document requires a valid document type declaration to insure that your web pages are displayed the way they are intended to be displayed.

**syntax:**

<!DOCTYPE html>

* The above syntax is case-insensitive.

# HTML Meta:

* The <meta> tags are typically used to provide structured metadata such as a document's keywords, description, author name, character encoding, and other metadata. Any number of meta tags can be placed inside the [head section](https://www.tutorialrepublic.com/html-tutorial/html-head.php) of an HTML or XHTML document.
* Metadata will not be displayed on the web page, but will be machine parsable, and can be used by the browsers, search engines like Google or other web services.

**syntax:**

<meta charset = "UTF-8">

The above syntax is case-insensitive.

**Declaring Character Encoding in HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Declaring Character Encoding</title>

<meta charset="utf-8">

</head>

<body>

<h1>Hello World!</h1>

</body>

</html>

## The <script> tag

It's common practice to add a type attribute with a value of "text/javascript" to script elements as follows −

<script type = "text/javascript" src = "scriptfile.js"></script>

HTML 5 removes extra information required and you can use simply following syntax −

<script src = "scriptfile.js"></script>

## The <link> tag

So far you were writing <link> as follows −

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

HTML 5 removes extra information required and you can simply use the following syntax −

<link rel = "stylesheet" href = "stylefile.css">

# HTML 5 Tags:

There is a list of newly included tags in HTML 5. These HTML 5 tags (elements) provide a better document structure. This list shows all HTML 5 tags in alphabetical order with description.

## List of HTML 5 Tags

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <article> | This element is used to define an independent piece of content in a document, that may be a blog, a magazine or a newspaper article. |
| <aside> | It specifies that article is slightly related to the rest of the whole page. |
| <audio> | It is used to play audio file in HTML. |
| <bdi> | The bdi stands for bi-directional isolation. It isolates a part of text that is formatted in other direction from the outside text document. |
| <canvas> | It is used to draw canvas. |
| <data> | It provides machine readable version of its data. |
| <datalist> | It provides auto complete feature for textfield. |
| <details> | It specifies the additional information or controls required by user. |
| <dialog> | It defines a window or a dialog box. |
| <figcaption> | It is used to define a caption for a <figure> element. |
| <figure> | It defines a self-contained content like photos, diagrams etc. |
| <footer> | It defines a footer for a section. |
| <header> | It defines a header for a section. |
| <main> | It defines the main content of a document. |
| <mark> | It specifies the marked or highlighted content. |
| <menuitem> | It defines a command that the user can invoke from a popup menu. |
| <meter> | It is used to measure the scalar value within a given range. |
| <nav> | It is used to define the navigation link in the document. |
| <progress> | It specifies the progress of the task. |
| <rp> | It defines what to show in browser that don't support ruby annotation. |
| <rt> | It defines an explanation/pronunciation of characters. |
| <ruby> | It defines ruby annotation along with <rp> and <rt>. |
| <section> | It defines a section in the document. |
| <summary> | It specifies a visible heading for <detailed> element. |
| <svg> | It is used to display shapes. |
| <time> | It is used to define a date/time. |
| <video> | It is used to play video file in HTML. |
| <wbr> | It defines a possible line break. |

|  |  |
| --- | --- |
| **HTML5 Form Tags** | |
|  |  |
| <datalist> | It represent predefined list for input <option> element. |
| <output> | It is used a container element to represent the output of a calculation or outcome of user action. |
| Graphics Tags | |
|  |  |
| <canvas> | It allows drawing graphics and animations via scripting. |
| <svg> | It is used to draw scalable vector graphics. |
| **HTML5 Media Tags** | |
|  |  |
| <audio> | It defines sound content. |
| <embed> | It defines a container for external files/application/media. |
| <source> | It defines multiple media resources for the media elements. |
| <track> | It defines text tracks for <audio> and <video> files |
| <video> | It defines video content within HTML document. |

## HTML5 New <input> types:

|  |  |
| --- | --- |
| **Type** | **Description** |
| color | It represents an input field which defines a color selector. |
| date | It represents an input field to define a date selector. |
| datetime | It defines full date and time display with time zone information. |
| datetime-local | It defines date and time without time zone information. |
| email | It defines an input field with email pattern Validation. |
| month | It defines the input field to enter month for the particular year |
| number | It defines field which selects a numeric value only. |
| range | It defines a numeric value selector with a given range of 1 to 100. |
| search | It is used to define a search field. |
| tel | It represents a control to enter a telephone number. |
| time | It represents a control to enter time value with no time zone. |
| url | It represents an input field to enter a URL |
| week | It defines a selector for week value for the particular year. |

## 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.

## 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.

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>

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.

## Global Attributes in HTML5:

In addition to the element specific attributes, the HTML5 defines few attributes that are common to all elements. These attributes may be specified on all elements, with some exceptions where it is not relevant, such as elements found inside the [<head>](https://www.tutorialrepublic.com/html-reference/html-head-tag.php) section of the document, e.g. [<base>](https://www.tutorialrepublic.com/html-reference/html-base-tag.php), [<script>](https://www.tutorialrepublic.com/html-reference/html-script-tag.php), [<title>](https://www.tutorialrepublic.com/html-reference/html-title-tag.php) etc.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value** | **Description** |
| accesskey | *shortcut key* | Specifies a keyboard shortcut to activate or focus the element. |
| class | *classname* | Assigns a class name or space-separated list of class names to an element. |
| contenteditable | true false | Indicates whether the content of an element is editable by the user or not. |
| contextmenu | *menu-id* | Specifies a context menu for an element. A context menu is a menu that appears when the user clicks the right mouse button on the element. |
| data-\* | *data* | Specified on any HTML element, to store custom data specific to the page. |
| dir | ltr rtl | Specifies the base direction of directionality of the element's text. |
| draggable | true false | Specifies whether an element is draggable or not. |
| dropzone | copy move link | Specifies whether the dragged data is copied, moved, or linked, when dropped. |
| hidden | hidden | Indicates that the element is not yet, or is no longer, relevant. |
| id | *name* | Specifies a unique identifier (ID) for an element which must be unique in the whole document. |
| lang | *language-code* | Specifies the primary language for the element's text content. |
| spellcheck | true false | Specifies whether the element may be checked for spelling errors or not. |
| style | *style* | Specifies inline style information for an element. |
| tabindex | *number* | Specifies the tabbing order of an element. |
| title | *text* | Provides advisory information related to the element. It would be appropriate for a tooltip. |
| translate | yes no | Specifies whether the text content of an element should be translated or not. |
| xml:lang | *language-code* | Specifies the primary language for the element's text content, in XHTML documents. |

## Event Attributes in HTML5

The following event attributes can be applied to the most of the elements for the execution of JavaScript when certain events occur, with some exceptions where it is not relevant such as elements found inside the [<head>](https://www.tutorialrepublic.com/html-reference/html-head-tag.php) section, e.g. [<title>](https://www.tutorialrepublic.com/html-reference/html-title-tag.php), [<base>](https://www.tutorialrepublic.com/html-reference/html-base-tag.php), [<link>](https://www.tutorialrepublic.com/html-reference/html-link-tag.php) etc.

## Window Events

Events related to the window object (applies to the [<body>](https://www.tutorialrepublic.com/html-reference/html-body-tag.php) tag):

| **Attribute** | **Value** | **Description** |
| --- | --- | --- |
| Onafterprint | *script* | Fires after the associated document is printed. |
| Onbeforeprint | *script* | Fires before the associated document is printed. |
| Onbeforeunload | *script* | Fires before a document being unloaded. |
| Onerror | *script* | Fires when document errors occur. |
| Onhashchange | *script* | Fires when the fragment identifier part of the document's URL i.e. the portion of a URL that follows the sign (#) changes. |
| Onload | *script* | Fires when the document has finished loading. |
| Onmessage | *script* | Fires when the message event occurs i.e. when user sends a cross-document message or a message is sent from a worker with postMessage() method. See [HTML5 Web Workers](https://www.tutorialrepublic.com/html-tutorial/html5-web-workers.php). |
| Onoffline | *script* | Fires when the network connection fails and the browser starts working offline. |
| Ononline | *script* | Fires when the network connections returns and the browser starts working online. |
| Onpagehide | *script* | Fires when the page is hidden, such as when a user is moving to another webpage. |
| Onpageshow | *script* | Fires when the page is shown, such as when a user navigates to a webpage. |
| Onpopstate | *script* | Fires when changes are made to the active history. |
| Onresize | *script* | Fires when the browser window is resized. |
| Onstorage | *script* | Fires when a [Web Storage](https://www.tutorialrepublic.com/html-tutorial/html5-web-storage.php) area is updated. |
| Onunload | *script* | Fires immediately before the document is unloaded or the browser window is closed. |

## Form Events

Events that occur due to the user interacting with the HTML form controls.

| **Attribute** | **Value** | **Description** |
| --- | --- | --- |
| Onblur | *script* | Fires when an element loses focus. |
| Onchange | *script* | Fires when the value or state of the element is changed. |
| Onfocus | *script* | Fires when the element receives focus. |
| Oninput | *script* | Fires when the value of an element is changed by the user. |
| Oninvalid | *script* | Fires when a submittable element do not satisfy their constraints during form validation. |
| Onreset | *script* | Fires when the user resets a form. |
| Onselect | *script* | Fires when some text is being selected or the current selection is changed by the user. |
| Onsearch | *script* | Fires when the user writes something in a [search input](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#search-input) field. |
| Onsubmit | *script* | Fires when a form is submitted. |

## Mouse Events

Events that occur due to the user interacting with a pointing device such as a mouse:

| **Attribute** | **Value** | **Description** |
| --- | --- | --- |
| Onclick | *script* | Fires when the user clicks the left mouse button on the element. |
| Ondblclick | *script* | Fires when the user double-clicks on the element. |
| Oncontextmenu | *script* | Fires when a context menu is triggered by the user through right-click on the element. |
| Ondrag | *script* | Fires when the user drags an element. The ondrag event fires throughout the drag operation. |
| Ondragend | *script* | Fires when the user releases the mouse button at the end of a drag operation. |
| Ondragenter | *script* | Fires when the user drags an element to a valid drop target. |
| Ondragleave | *script* | Fires when an element leaves a valid drop target during a drag operation. |
| Ondragover | *script* | Fires when an element is being dragged over a valid drop target. |
| Ondragstart | *script* | Fires when the user starts to drag a text selection or selected element. |
| Ondrop | *script* | Fires when the mouse button is released during a drag-and-drop operation i.e. when dragged element is being dropped. |
| Onmousedown | *script* | Fires when the mouse button is pressed over an element. |
| Onmousemove | *script* | Fires when the user moves the mouse pointer over an element. |
| Onmouseout | *script* | Fires when the user moves the mouse pointer outside the boundaries of an element. |
| Onmouseover | *script* | Fires when the user moves the mouse pointer onto an element. |
| Onmouseup | *script* | Fires when the user releases the mouse button while the mouse is over an element. |
| Onmousewheel | *script* | Use the onwheel attribute instead. |
| Onscroll | *script* | Fires when the user scrolls the contents of an element by scrolling the element's scrollbar. |
| Onshow | *script* | Fires when a [contextmenu](https://www.tutorialrepublic.com/html-reference/html5-event-attributes.php" \l "contextmenu) event was fired onto an element that has a contextmenu attribute. |
| Ontoggle | *script* | Fires when the user opens or closes the [<details>](https://www.tutorialrepublic.com/html-reference/html5-details-tag.php) element. |
| Onwheel | *script* | Fires when the user scrolls the contents of an element by rolling the mouse wheel up or down over an element. |

## Keyboard Events

Events that occur by the user interaction with the keyboard:

| **Attribute** | **Value** | **Description** |
| --- | --- | --- |
| Onkeydown | *script* | Fires when the user presses a key. |
| Onkeypress | *script* | Fires when the user presses an alphanumeric key. |
| Onkeyup | *script* | Fires when the user releases a key. |

## Clipboard Events

Events related to modification of the clipboard, that is copy, cut and paste:

| **Attribute** | **Value** | **Description** |
| --- | --- | --- |
| Oncopy | *script* | Fires when the user copies the element or selection, adding it to the system clipboard. |
| Oncut | *script* | Fires when the element or selection is removed from the document and added to the system clipboard. |
| Onpaste | *script* | Fires when the user pastes data, transferring the data from the system clipboard to the document. |

## Media Events

Events that occur when handling media elements that are embedded inside the HTML documents, such as [<audio>](https://www.tutorialrepublic.com/html-reference/html5-audio-tag.php) and [<video>](https://www.tutorialrepublic.com/html-reference/html5-video-tag.php) elements:

| **Attribute** | **Value** | **Description** |
| --- | --- | --- |
| Onabort | *script* | Fires when playback is aborted, but not due to an error. |
| Oncanplay | *script* | Fires when enough data is available to play the media, at least for a couple of frames, but would require further buffering. |
| Oncanplaythrough | *script* | Fires when entire media can be played to the end without requiring to stop for further buffering. |
| Oncuechange | *script* | Fires when the text track cue in a [<track>](https://www.tutorialrepublic.com/html-reference/html5-track-tag.php) element changes. |
| Ondurationchange | *script* | Fires when the duration of the media changes. |
| Onemptied | *script* | Fires when the media element is reset to its initial state, either because of a fatal error during load, or because the load() method is called to reload it. |
| Onended | *script* | Fires when the end of playback is reached. |
| Onerror | *script* | Fires when an error occurs while fetching the media data. |
| Onloadeddata | *script* | Fires when media data is loaded at the current playback position. |
| onloadedmetadata | *script* | Fires when metadata of the media (like duration and dimensions) has finished loading. |
| Onloadstart | *script* | Fires when loading of the media begins. |
| Onpause | *script* | Fires when playback is paused, either by the user or programmatically. |
| Onplay | *script* | Fires when playback of the media starts after having been paused i.e. when the play() method is requested. |
| Onplaying | *script* | Fires when the audio or video has started playing. |
| Onprogress | *script* | Fires periodically to indicate the progress while downloading the media data. |
| Onratechange | *script* | Fires when the playback rate or speed is increased or decreased, like slow motion or fast forward mode. |
| Onseeked | *script* | Fires when the seek operation ends. |
| Onseeking | *script* | Fires when the current playback position is moved. |
| Onstalled | *script* | Fires when the download has stopped unexpectedly. |
| Onsuspend | *script* | Fires when the loading of the media is intentionally stopped. |
| Ontimeupdate | *script* | Fires when the playback position changed, like when the user fast forwards to a different playback position. |
| Onvolumechange | *script* | Fires when the volume is changed, or playback is muted or unmuted. |
| Onwaiting | *script* | Fires when playback stops because the next frame of a video resource is not available. |

**Video And Audio**

# HTML Video:

The HTML <video> element is used to show a video on a web page.

HTML 5 supports <video> tag also. The HTML video tag is used for streaming video files such as a movie clip, song clip on the web page.

Currently, there are three video formats supported for HTML video tag:

1. mp4
2. webM
3. ogg

|  |  |  |  |
| --- | --- | --- | --- |
| **Browser** | **mp4** | **webM** | **ogg** |
| Internet Explorer | yes | no | no |
| Google Chrome | yes | yes | yes |
| Mozilla Firefox | yes | yes | yes |
| Opera | no | yes | yes |
| Apple Safari | yes | no | no |

**Example:**

<!DOCTYPE html>

<html>

<body>

<video width="400" controls>

<source src="mov\_bbb.mp4" type="video/mp4">

</video>

<p>

<a href="https://www.bigbuckbunny.org/" target="\_blank">Big Buck Bunny</a>.

</p>

</body>

</html>

## HTML Video Tag Example:

Let's see the code to play mp4 file using HTML video tag.

<!DOCTYPE>

<html>

<body>

<video controls>

<source src="movie.mp4" type="video/mp4">

</video>

</body>

</html>

## Attributes of HTML Video Tag:

Let's see the list of HTML 5 video tag attributes.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| controls | It defines the video controls which is displayed with play/pause buttons. |
| height | It is used to set the height of the video player. |
| width | It is used to set the width of the video player. |
| poster | It specifies the image which is displayed on the screen when the video is not played. |
| autoplay | It specifies that the video will start playing as soon as it is ready. |
| loop | It specifies that the video file will start over again, every time when it is completed. |
| muted | It is used to mute the video output. |
| preload | It specifies the author view to upload video file when the page loads. |
| src | It specifies the source URL of the video file. |

## HTML Video Tag Attribute Example

Let's see the example of video tag in HTML where are using height, width, autoplay, controls and loop attributes.

<!DOCTYPE>

<html>

<body>

<video width="320" height="240" controls autoplay loop>

<source src="movie.mp4" type="video/mp4">

</video>

</body>

</html>

# HTML5 Audio Tag:

**HTML audio tag** is used to define sounds such as music and other audio clips. Currently there are three supported file format for HTML 5 audio tag.

1. mp3
2. wav
3. ogg

HTML5 supports <video> and <audio> controls. The Flash, Silverlight and similar technologies are used to play the multimedia items.

This table defines that which web browser supports which audio file format.

|  |  |  |  |
| --- | --- | --- | --- |
| **Browser** | **mp3** | **wav** | **ogg** |
| Internet Explorer | yes | no | no |
| Google Chrome | yes | yes | yes |
| Mozilla Firefox | yes\* | yes | yes |
| Opera | no | yes | yes |
| Apple Safari | yes | yes | no |

## HTML Audio Tag Example:

## <!DOCTYPE>

## <html>

## <body>

## <audio controls>

## <source src="koyal.mp3" type="audio/mpeg">

## </audio>

## </body>

## </html>

## Attributes of HTML Audio Tag:

There is given a list of HTML audio tag.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| controls | It defines the audio controls which is displayed with play/pause buttons. |
| autoplay | It specifies that the audio will start playing as soon as it is ready. |
| loop | It specifies that the audio file will start over again, every time when it is completed. |
| muted | It is used to mute the audio output. |
| preload | It specifies the author view to upload audio file when the page loads. |
| src | It specifies the source URL of the audio file. |

## HTML Audio Tag Attribute Example:

Here we are going to use controls, autoplay, loop and src attributes of HTML audio tag.

<!DOCTYPE>

<html>

<body>

<audio controls autoplay loop>

<source src="koyal.mp3" type="audio/mpeg"></audio>

</body>

</html>

# HTML Canvas Tag:

The **HTML canvas element** provides HTML a bitmapped surface to work with. It is used to draw graphics on the web page.

The **HTML 5 <canvas> tag** is used to draw graphics using scripting language like JavaScript.

HTML5 element <canvas> gives you an easy and powerful way to draw graphics using JavaScript. It can be used to draw graphs, make photo compositions or do simple (and not so simple) animations.

The <canvas> element is only a container for graphics, you must need a scripting language to draw the graphics. The <canvas> element allows for dynamic and scriptable rendering of 2D shapes and bitmap images.

It is a low level, procedural model that updates a bitmap and does not have a built-in scene. There are several methods in canvas to draw paths, boxes, circles, text and add images.

## How to create a HTML canvas?

A canvas is a rectangle like area on an HTML page. It is specified with canvas element. By default, the <canvas> element has no border and no content, it is like a container.

**<canvas** id = "mycanvas" width ="200" height ="100"**>** **</canvas>**

## HTML 5 Canvas Tag Example:

## <!DOCTYPE>

## <html>

## <body>

## <canvas id="myCanvas" width="300" height="200" style="border:2px solid;">

## </canvas>

## </body>

## </html>

## Supporting Browsers:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Element | Chrome | IE | Firefox | Opera | Safari |
| <canvas> | Yes | Yes | Yes | Yes | Yes |

## HTML Canvas Tag with JavaScript

A canvas is a two dimensional grid.

Coordinates (0,0) defines the upper left corner of the canvas. The parameters (0,0,200,100) is used for fillRect() method. This parameter will fill the rectangle start with the upper-left corner (0,0) and draw a 200 \* 100 rectangle.

1. **<canvas** id="myCanvas" width="250" height="150" style="border:1px solid #c3c3c3;"**>**
2. Your browser does not support the HTML5 canvas tag.
3. **</canvas>**
4. **<script>**
5. var c = document.getElementById("myCanvas");
6. var cctx = c.getContext("2d");
7. ctx.fillStyle = "#FF0000";
8. ctx.fillRect(0,0,200,100);
9. **</script>**

## Drawing Line on Canvas:

If you want to draw a straight line on the canvas, you can use the following two methods.

**moveTo(x,y):** It is used to define the starting point of the line.

**lineTo(x,y):** It is used to define the ending point of the line.

If you draw a line which starting point is (0,0) and the end point is (200,100), use the stroke method to draw the line.

1. **<canvas** id="myCanvasLine" width="200" height="100" style="border:1px solid #d3d3d3;"**>**
2. Your browser does not support the HTML5 canvas tag.**</canvas>**
3. **<script>**
4. var c = document.getElementById("myCanvasLine");
5. var cctx = c.getContext("2d");
6. ctx.moveTo(0,0);
7. ctx.lineTo(200,100);
8. ctx.stroke();
9. **</script>**

## Drawing Circle on Canvas

If you want to draw a circle on the canvas, you can use the arc() method:

1. arc(x, y, r, start, stop)

To sketch circle on HTML canvas, use one of the ink() methods, like stroke() or fill().

1. **<canvas** id="myCanvasCircle" width="200" height="100" style="border:1px solid #d3d3d3;"**>**
2. Your browser does not support the HTML5 canvas tag.**</canvas>**
3. **<script>**
4. var c = document.getElementById("myCanvasCircle");
5. var cctx = c.getContext("2d");
6. ctx.beginPath();
7. ctx.arc(95,50,40,0,2\*Math.PI);
8. ctx.stroke();
9. **</script>**

## Drawing text on canvas

There are property and methods used for drawing text on the canvas.

**font property:** It is used to define the font property for the text.

**fillText(text,x,y) method:** It is used to draw filled text on the canvas. It looks like bold font.

**strokeText(text,x,y) method:** It is also used to draw text on the canvas, but the text is unfilled.

Let's see **fillText()** method example.

1. **<canvas** id="myCanvasText1" width="300" height="100" style="border:1px solid #d3d3d3;"**>**
2. Sorry! Your browser does not support the HTML5 canvas tag.**</canvas>**
3. **<script>**
4. var c = document.getElementById("myCanvasText1");
5. var cctx = c.getContext("2d");
6. ctx.font = "30px Arial";
7. ctx.fillText("Hello JavaTpoint",10,50);
8. **</script>**

Let's see **strokeText()** method example.

1. **<canvas** id="myCanvasText2" width="300" height="100" style="border:1px solid #d3d3d3;"**>**
2. Sorry!Upgrade your browser. It does not support the HTML5 canvas tag.**</canvas>**
3. **<script>**
4. var c = document.getElementById("myCanvasText2");
5. var cctx = c.getContext("2d");
6. ctx.font = "30px Arial";
7. ctx.strokeText("Hello JavaTpoint",10,50);
8. **</script>**

**Basic Styling using CSS3**

Cascading Style Sheets (CSS) is a language that is used to illustrate the look, style, and format of a document written in any markup language. In simple words, it is used to style and organize the layout of Web pages. CSS3 is the latest version of an earlier CSS version, CSS2.

CSS3 is the latest standard for CSS.

CSS3 is completely backwards-compatible with earlier versions of CSS.

CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces).

Some of the most important CSS3 modules are:

* Selectors
* Box Model
* Backgrounds and Borders
* Image Values and Replaced Content
* Text Effects
* 2D/3D Transformations
* Animations
* Multiple Column Layout
* User Interface

### Features of CSS3:

The features of the CSS3 are as follows:

#### 1. Selectors

Selectors allow the designer to select on more precise levels of the web page. They are structural pseudo-classes that perform partial matches to help match attribute and attribute values. New selectors target a pseudo-class to style the elements targeted in the URL. Selectors also include a checked pseudo-class to style checked elements such as checkboxes and radio buttons.

#### 2. Text Effects and Layout

With CSS3, we can change the justification of text, whitespace adjustment of the document, and style the hyphenation of words.

#### 3. First-Letter and First-Line Pseudo-Classes

CSS 3 includes properties that help with kerning (adjusting the spacing between characters to achieve a visually pleasing effect) and positioning drop-caps (large decorative capital letter at the starting of a paragraph).

#### 4. Paged Media and Generated Content

CSS 3 has additional choices in Paged Media, such as page numbers and running headers and footers. There are additional properties for printing Generated Content as well, like properties for cross-references and footnotes.

#### 5. Multi-Column Layout

This feature includes properties to allow designers to present their content in multiple columns with options like the column-count, column-gap, and column-width.

### Advantages of CSS3

* CSS3 provides a consistent and precise positioning of navigable elements.
* It is easy to customize a web page as it can be done by merely altering a modular file.
* Graphics are easier in CSS3, thus making it easy to make the site appealing.
* It permits online videos to be seen without using third-party plug-ins.
* CSS3 is economical, time-saving, and most browsers support it.

## Including CSS in HTML Documents:

CSS can either be attached as a separate document or embedded in the HTML document itself. There are three methods of including CSS in an HTML document:

* **Inline styles** — Using the style attribute in the HTML start tag.
* **Embedded styles** — Using the [<style>](https://www.tutorialrepublic.com/html-reference/html-style-tag.php) element in the head section of a document.
* **External style sheets** — Using the [<link>](https://www.tutorialrepublic.com/html-reference/html-link-tag.php) element, pointing to an external CSS file.

**i) Inline Styles:**

Inline styles are used to apply the unique style rules to an element by putting the CSS rules directly into the start tag. It can be attached to an element using the style attribute.

The style attribute includes a series of CSS property and value pairs. Each "property: value" pair is separated by a semicolon (;), just as you would write into an embedded or external style sheets. But it needs to be all in one line i.e. no line break after the semicolon, as shown here:

**Example:** <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of CSS Inline Styles</title>

</head>

<body>

<h1 style="color:red; font-size:30px;">This is a heading</h1>

<p style="color:green; font-size:22px;">This is a paragraph.</p>

<div style="color:blue; font-size:14px;">This is some text content.</div>

</body>

</html>

**ii) Embedded Style Sheets:**

Embedded or internal style sheets only affect the document they are embedded in.

Embedded style sheets are defined in the [<head>](https://www.tutorialrepublic.com/html-tutorial/html-head.php) section of an HTML document using the [<style>](https://www.tutorialrepublic.com/html-reference/html-style-tag.php) element. You can define any number of <style> elements in an HTML document but they must appear between the <head> and </head> tags.

**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of CSS Embedded Style Sheet</title>

<style>

body { background-color: YellowGreen; }

p { color: #fff; }

</style>

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

**iii) External Style Sheets:**

An external style sheet is ideal when the style is applied to many pages of the website.

An external style sheet holds all the style rules in a separate document that you can link from any HTML file on your site. External style sheets are the most flexible because with an external style sheet, you can change the look of an entire website by changing just one file.

An external style sheet can be linked to an HTML document using the [<link>](https://www.tutorialrepublic.com/html-reference/html-link-tag.php) tag. The <link> tag goes inside the [<head>](https://www.tutorialrepublic.com/html-reference/html-head-tag.php) section.

**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of CSS External Style Sheet</title>

<link rel="stylesheet" href="/examples/css/style.css">

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

## Positioning And Background Images

## CSS Positioning Methods:

Positioning elements appropriately on the web pages is a necessity for a good layout design. There are several methods in CSS that you can use for positioning elements.

**i) Static Positioning:**

A static positioned element is always positioned according to the normal flow of the page. HTML elements are positioned static by default. Static positioned elements are not affected by the [top](https://www.tutorialrepublic.com/css-reference/css-top-property.php), [bottom](https://www.tutorialrepublic.com/css-reference/css-bottom-property.php), [left](https://www.tutorialrepublic.com/css-reference/css-left-property.php), [right](https://www.tutorialrepublic.com/css-reference/css-right-property.php), and [z-index](https://www.tutorialrepublic.com/css-reference/css-z-index-property.php) properties.

**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of CSS Static Positioning</title>

<style>

.box{

color: #fff;

background: #7dc765;

padding: 20px;

}

.container{

padding: 50px;

margin: 50px;

position: relative;

border: 5px solid black;

font-family: Arial, sans-serif;

}

.container p{

line-height: 50px;

}

</style>

</head>

<body>

<div class="container">

<div class="box">

<h2>Static Positioned Box</h2>

<div><strong>Note:</strong> This box is positioned static, which is default. It is always positioned according to the normal flow of the page.</div>

</div>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam eu sem tempor, varius quam at, luctus dui. Mauris magna metus, dapibus nec turpis vel, semper malesuada ante. Vestibulum id metus ac nisl bibendum scelerisque non non purus. Suspendisse varius nibh non aliquet sagittis. In tincidunt orci sit amet elementum vestibulum. Vivamus fermentum in arcu in aliquam. Quisque aliquam porta odio in fringilla. Vivamus nisl leo, blandit at bibendum eu, tristique eget risus. Integer aliquet quam ut elit suscipit, id interdum neque porttitor. Integer faucibus ligula.</p>

<p>Quis quam ut magna consequat faucibus. Pellentesque eget nisi a mi suscipit tincidunt. Ut tempus dictum risus. Pellentesque viverra sagittis quam at mattis. Suspendisse potenti. Aliquam sit amet gravida nibh, facilisis gravida odio. Phasellus auctor velit at lacus blandit, commodo iaculis justo viverra. Etiam vitae est arcu. Mauris vel congue dolor. Aliquam eget mi mi. Fusce quam tortor, commodo ac dui quis, bibendum viverra erat. Maecenas mattis lectus enim, quis tincidunt dui molestie euismod. Curabitur et diam tristique, accumsan nunc eu.</p>

</div>

</body></html>

**ii) Relative Positioning:**

A relative positioned element is positioned relative to its normal position.

In the relative positioning scheme the element's box position is calculated according to the normal flow. Then the box is shifted from this normal position according to the properties — top or bottom and/or left or right.

## <!DOCTYPE html>

## <html lang="en">

## <head>

## <meta charset="utf-8">

## <title>Example of CSS Relative Positioning</title>

## <style>

## .box{

## position: relative;

## left: 100px;

## color: #fff;

## background: #00c4cc;

## padding: 20px;

## }

## .container{

## padding: 50px;

## margin: 50px;

## border: 5px solid black;

## font-family: Arial, sans-serif;

## }

## .container p{

## line-height: 50px;

## }

## </style>

## </head>

## <body>

## <div class="container">

## <div class="box">

## <h2>Relative Positioned Box</h2>

## <div><strong>Note:</strong> The left margin edge of this DIV box is shifted to right by 100px from its original position. The whitespace generated is preserved.</div>

## </div>

## <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam eu sem tempor, varius quam at, luctus dui. Mauris magna metus, dapibus nec turpis vel, semper malesuada ante. Vestibulum id metus ac nisl bibendum scelerisque non non purus. Suspendisse varius nibh non aliquet sagittis. In tincidunt orci sit amet elementum vestibulum. Vivamus fermentum in arcu in aliquam. Quisque aliquam porta odio in fringilla. Vivamus nisl leo, blandit at bibendum eu, tristique eget risus. Integer aliquet quam ut elit suscipit, id interdum neque porttitor. Integer faucibus ligula.</p>

## <p>Quis quam ut magna consequat faucibus. Pellentesque eget nisi a mi suscipit tincidunt. Ut tempus dictum risus. Pellentesque viverra sagittis quam at mattis. Suspendisse potenti. Aliquam sit amet gravida nibh, facilisis gravida odio. Phasellus auctor velit at lacus blandit, commodo iaculis justo viverra. Etiam vitae est arcu. Mauris vel congue dolor. Aliquam eget mi mi. Fusce quam tortor, commodo ac dui quis, bibendum viverra erat. Maecenas mattis lectus enim, quis tincidunt dui molestie euismod. Curabitur et diam tristique, accumsan nunc eu.</p>

## </div>

## </body>

## </html>

## iii) Absolute Positioning:

An absolutely positioned element is positioned relative to the first parent element that has a position other than static. If no such element is found, it will be positioned on a page relative to the 'top-left' corner of the browser window. The box's offsets further can be specified using one or more of the properties top, right, bottom, and left.

Absolutely positioned elements are taken out of the normal flow entirely and thus take up no space when placing sibling elements. However, it can overlap other elements depending on the [z-index](https://www.tutorialrepublic.com/css-reference/css-z-index-property.php) property value. Also, an absolutely positioned element can have [margins](https://www.tutorialrepublic.com/css-reference/css-margin-property.php), and they do not collapse with any other margins.

**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of CSS Absolute Positioning</title>

<style>

.box{

position: absolute;

top: 200px;

left: 100px;

color: #fff;

width: 60%;

background: #4cafdf;

padding: 20px;

}

.container{

padding: 50px;

margin: 50px;

position: relative;

border: 5px solid black;

font-family: Arial, sans-serif;

}

.container p{

line-height: 50px;

}

</style>

</head>

<body>

<div class="container">

<div class="box">

<h2>Absolute Positioned Box</h2>

<div><strong>Note:</strong> This box is absolutely positioned relative to the container DIV element. It is scroll with the page.</div>

</div>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam eu sem tempor, varius quam at, luctus dui. Mauris magna metus, dapibus nec turpis vel, semper malesuada ante. Vestibulum id metus ac nisl bibendum scelerisque non non purus. Suspendisse varius nibh non aliquet sagittis. In tincidunt orci sit amet elementum vestibulum. Vivamus fermentum in arcu in aliquam. Quisque aliquam porta odio in fringilla. Vivamus nisl leo, blandit at bibendum eu, tristique eget risus. Integer aliquet quam ut elit suscipit, id interdum neque porttitor. Integer faucibus ligula.</p>

<p>Quis quam ut magna consequat faucibus. Pellentesque eget nisi a mi suscipit tincidunt. Ut tempus dictum risus. Pellentesque viverra sagittis quam at mattis. Suspendisse potenti. Aliquam sit amet gravida nibh, facilisis gravida odio. Phasellus auctor velit at lacus blandit, commodo iaculis justo viverra. Etiam vitae est arcu. Mauris vel congue dolor. Aliquam eget mi mi. Fusce quam tortor, commodo ac dui quis, bibendum viverra erat. Maecenas mattis lectus enim, quis tincidunt dui molestie euismod. Curabitur et diam tristique, accumsan nunc eu.</p>

</div>

</body>

</html>

**iv) Fixed Positioning:**

Fixed positioning is a subcategory of absolute positioning.

The only difference is, a fixed positioned element is fixed with respect to the browser's [viewport](https://www.tutorialrepublic.com/definitions.php#viewport) and does not move when scrolled.

**Example:**

## <!DOCTYPE html>

## <html lang="en">

## <head>

## <meta charset="utf-8">

## <title>Example of CSS Fixed Positioning</title>

## <style>

## .box{

## position: fixed;

## top: 200px;

## left: 100px;

## color: #fff;

## width: 60%;

## background: #f44712;

## padding: 20px;

## }

## .container{

## padding: 50px;

## margin: 50px;

## position: relative;

## border: 5px solid black;

## font-family: Arial, sans-serif;

## }

## .container p{

## line-height: 50px;

## }

## </style>

## </head>

## <body>

## <div class="container">

## <div class="box">

## <h2>Fixed Positioned Box</h2>

## <div><strong>Note:</strong> The position of this box is fixed relative to the document's viewport. It doesn't scroll with the page.</div>

## </div>

## <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam eu sem tempor, varius quam at, luctus dui. Mauris magna metus, dapibus nec turpis vel, semper malesuada ante. Vestibulum id metus ac nisl bibendum scelerisque non non purus. Suspendisse varius nibh non aliquet sagittis. In tincidunt orci sit amet elementum vestibulum. Vivamus fermentum in arcu in aliquam. Quisque aliquam porta odio in fringilla. Vivamus nisl leo, blandit at bibendum eu, tristique eget risus. Integer aliquet quam ut elit suscipit, id interdum neque porttitor. Integer faucibus ligula.</p>

## <p>Quis quam ut magna consequat faucibus. Pellentesque eget nisi a mi suscipit tincidunt. Ut tempus dictum risus. Pellentesque viverra sagittis quam at mattis. Suspendisse potenti. Aliquam sit amet gravida nibh, facilisis gravida odio. Phasellus auctor velit at lacus blandit, commodo iaculis justo viverra. Etiam vitae est arcu. Mauris vel congue dolor. Aliquam eget mi mi. Fusce quam tortor, commodo ac dui quis, bibendum viverra erat. Maecenas mattis lectus enim, quis tincidunt dui molestie euismod. Curabitur et diam tristique, accumsan nunc eu.</p>

## </div>

## </body>

## </html>

## Background Images: The CSS3 provides several new properties to manipulate the background of an element like background clipping, multiple backgrounds, and the option to adjust the background size.

## i) CSS3 background-size Property:

The background-size property can be used to specify the size of the background images.  Prior to CSS3, the size of the background images was determined by the actual size of the images. The background image size can be specified using the pixels or percentage values as well as the keywords auto, contain, and cover. Negative values are not allowed.

## Example:

## <!DOCTYPE html>

## <html lang="en">

## <head>

## <meta charset="utf-8">

## <title>Example of Setting background-size of an Element</title>

## <style>

## .box {

## width: 250px;

## height: 150px;

## background: url("/examples/images/sky.jpg") no-repeat;

## background-size: contain;

## border: 6px solid #333;

## }

## </style>

## </head>

## <body>

## <div class="box"></div>

## <p><strong>Note:</strong> The original size of the background image is 500x300 pixels, but using the background-size CSS property we are still able to show the complete image inside the smaller box.</p>

## </body>

## </html>

## ii) CSS3 background-clip Property:

The background-clip property can be used to specify whether an element's background extends into the border or not. The background-clip property can take the three values: border-box, padding-box, content-box.

## Example:

## <!DOCTYPE html>

## <html lang="en">

## <head>

## <meta charset="utf-8">

## <title>Example of CSS3 Background Clipping</title>

## <style>

## .box {

## width: 250px;

## height: 150px;

## padding: 10px;

## border: 6px dashed #333;

## background: orange;

## }

## .clip1 {

## background-clip: border-box;

## }

## .clip2 {

## background-clip: padding-box;

## }

## .clip3 {

## background-clip: content-box;

## }

## </style>

## </head>

## <body>

## <h2>Default Background Behavior</h2>

## <div class="box"></div>

## <h2>Background Clipping Using border-box</h2>

## <div class="box clip1"></div>

## <h2>Background Clipping Using padding-box</h2>

## <div class="box clip2"></div>

## <h2>Background Clipping Using content-box</h2>

## <div class="box clip3"></div>

## </body>

## </html>

## iii) CSS3 background-origin Property:

The background-origin property can be used to specify the positioning area of the background images. It can take the same values as background-clip property: border-box, padding-box, content-box.

**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of Setting background-origin of an Element</title>

<style>

.box {

width: 250px;

height: 150px;

padding: 10px;

border: 6px dashed #333;

background: url("/examples/images/sky.jpg") no-repeat;

background-size: contain;

background-origin: content-box;

}

</style>

</head>

<body>

<div class="box"></div>

</body>

</html>

**iv) CSS3 Multiple Backgrounds:**

CSS3 gives you ability to add multiple backgrounds to a single element. The backgrounds are layered on the top of one another. The number of layers is determined by the number of comma-separated values in the [background-image](https://www.tutorialrepublic.com/css-reference/css-background-image-property.php) or [background](https://www.tutorialrepublic.com/css-reference/css-background-property.php) shorthand property.

**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of CSS3 Multiple Backgrounds</title>

<style>

.box {

width: 100%;

height: 500px;

background: url("/examples/images/birds.png") no-repeat center, url("/examples/images/clouds.png") no-repeat center, url("/examples/images/sun.png") no-repeat 10% 30%, lightblue;

}

</style>

</head>

<body>

<div class="box"></div>

</body></html>

**High Level Programming**

**What is JavaScript?**

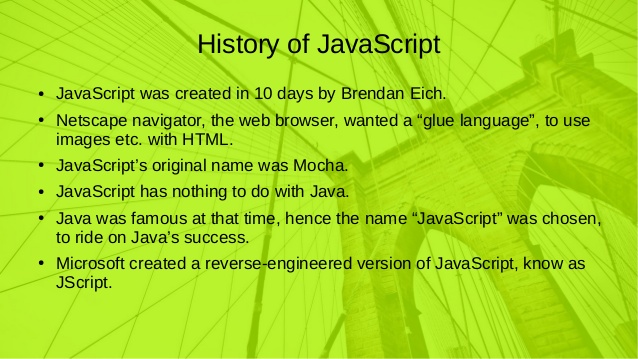
JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages . It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

## Features of JavaScript:

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

**History of JavaScript:**



**Application of JavaScript**

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

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

### JavaScript Example:

<html>

<body>

<h2>Welcome to JavaScript</h2>

<script>

document.write("Hello JavaScript by JavaScript");

</script>

</body>

</html>

## Output: Welcome to JavaScript

Hello JavaScript by JavaScript

# JavaScript Variable

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable. There are some rules while declaring a JavaScript variable (also known as identifiers).

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

The general rules for constructing names for variables (unique identifiers) are:

* Names can contain letters, digits, underscores, and dollar signs.
* Names must begin with a letter
* Names can also begin with $ and \_ (but we will not use it in this tutorial)
* Names are case sensitive (y and Y are different variables)
* Reserved words (like JavaScript keywords) cannot be used as names
* JavaScript identifiers are case-sensitive.

## Declaring (Creating) JavaScript Variables:

Creating a variable in JavaScript is called "declaring" a variable.

You declare a JavaScript variable with the var keyword:

**var  carName;**

**Correct JavaScript variables:**

1. var  x = 10;
2. var \_value="sonoo";

## Incorrect JavaScript variables:

1. var  123=30;
2. var \*aa=320;

## Example of JavaScript variable:

## <html>

## <body>

## <script>

## var x = 10;

## var y = 20;

## var z=x+y;

## document.write(z);

## </script>

## </body>

## </html>

## Output:30

## i) JavaScript local variable:

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.

1. **<script>**
2. function abc(){
3. var  x=10;//local variable
4. }
5. **</script>**

Or,

1. **<script>**
2. If(10**<13**){
3. var  y=20;//JavaScript local variable
4. }
5. **</script>**

## ii) JavaScript global variable:

A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable.

**Example:**

<html>

<body>

<script>

var data=200;//global variable

function a(){

document.writeln(data);

}

function b(){

document.writeln(data);

}

a();//calling JavaScript function

b();

</script>

</body>

</html>

**Output:** 200 200

**JavaScript Arrays**

JavaScript arrays are used to store multiple values in a single variable.

## What is an Array?

An array is a special variable, which can hold more than one value at a time.

var car1 = "Saab";  
var car2 = "Volvo";  
var car3 = "BMW";

**i) Creating an Array:**

Using an array literal is the easiest way to create a JavaScript Array.

**Syntax:var  *array name* = [*item1*, *item2*, ...];**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Arrays</h2>

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

<script>

var cars = ["Saab", "Volvo", "BMW"];

document.getElementById("demo").innerHTML = cars;

</script>

</body></html>

**Output:**

**JavaScript Arrays**

Saab,Volvo,BMW

Spaces and line breaks are not important. A declaration can span multiple lines:

**Example:**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Arrays</h2>

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

<script>

var cars = [

"Saab",

"Volvo",

"BMW"

];

document.getElementById("demo").innerHTML = cars;

</script>

</body>

</html>

## Output: JavaScript Arrays

## Saab,Volvo,BMW

## ii) Using the JavaScript Keyword new:

The syntax of creating array directly is given below:

var arrayname=new Array();

Here, **new keyword** is used to create instance of array.

**Example:**

## <html>

## <body>

## <script>

## var i;

## var emp = new Array();

## emp[0] = "Arun";

## emp[1] = "Varun";

## emp[2] = "John";

## for (i=0;i<emp.length;i++){

## document.write(emp[i] + "<br>");

## }

## </script>

## </body>

## </html>

## Output:

## Arun Varun John

**iii) Access the Elements of an Array:**

You access an array element by referring to the **index number**.

This statement accesses the value of the first element in cars:

var name = cars[0];

**iv) JavaScript array constructor (new keyword):**

 To create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

The example of creating object by array constructor is given below.

<html>

<body>

<script>

var emp=new Array("Jai","Vijay","Smith");

for (i=0;i<emp.length;i++){

document.write(emp[i] + "<br>");

}

</script>

</body>

</html>

**Output:** Jai  
 Vijay  
 Smith

**JavaScript Array Methods:**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| [concat()](https://www.javatpoint.com/javascript-array-concat-method) | It returns a new array object that contains two or more merged arrays. |
| [copywithin()](https://www.javatpoint.com/javascript-array-copywithin-method) | It copies the part of the given array with its own elements and returns the modified array. |
| [entries()](https://www.javatpoint.com/javascript-array-entries-method) | It creates an iterator object and a loop that iterates over each key/value pair. |
| [every()](https://www.javatpoint.com/javascript-array-every-method) | It determines whether all the elements of an array are satisfying the provided function conditions. |
| [flat()](https://www.javatpoint.com/javascript-array-flat-method) | It creates a new array carrying sub-array elements concatenated recursively till the specified depth. |
| [flatMap()](https://www.javatpoint.com/javascript-array-flatmap-method) | It maps all array elements via mapping function, then flattens the result into a new array. |
| [fill()](https://www.javatpoint.com/javascript-array-fill-method) | It fills elements into an array with static values. |
| [from()](https://www.javatpoint.com/javascript-array-from-method) | It creates a new array carrying the exact copy of another array element. |
| [filter()](https://www.javatpoint.com/javascript-array-filter-method) | It returns the new array containing the elements that pass the provided function conditions. |
| [find()](https://www.javatpoint.com/javascript-array-find-method) | It returns the value of the first element in the given array that satisfies the specified condition. |
| [findIndex()](https://www.javatpoint.com/javascript-array-findindex-method) | It returns the index value of the first element in the given array that satisfies the specified condition. |
| [forEach()](https://www.javatpoint.com/javascript-array-foreach-method) | It invokes the provided function once for each element of an array. |
| [includes()](https://www.javatpoint.com/javascript-array-includes-method) | It checks whether the given array contains the specified element. |
| [indexOf()](https://www.javatpoint.com/javascript-array-indexof-method) | It searches the specified element in the given array and returns the index of the first match. |
| [isArray()](https://www.javatpoint.com/javascript-array-isarray-method) | It tests if the passed value ia an array. |
| [join()](https://www.javatpoint.com/javascript-array-join-method) | It joins the elements of an array as a string. |
| [keys()](https://www.javatpoint.com/javascript-array-keys-method) | It creates an iterator object that contains only the keys of the array, then loops through these keys. |
| [lastIndexOf()](https://www.javatpoint.com/javascript-array-lastindexof-method) | It searches the specified element in the given array and returns the index of the last match. |
| [map()](https://www.javatpoint.com/javascript-array-map-method) | It calls the specified function for every array element and returns the new array |
| [of()](https://www.javatpoint.com/javascript-array-of-method) | It creates a new array from a variable number of arguments, holding any type of argument. |
| [pop()](https://www.javatpoint.com/javascript-array-pop-method) | It removes and returns the last element of an array. |
| [push()](https://www.javatpoint.com/javascript-array-push-method) | It adds one or more elements to the end of an array. |
| [reverse()](https://www.javatpoint.com/javascript-array-reverse-method) | It reverses the elements of given array. |
| [reduce(function, initial)](https://www.javatpoint.com/javascript-array-reduce-method) | It executes a provided function for each value from left to right and reduces the array to a single value. |
| [reduceRight()](https://www.javatpoint.com/javascript-array-reduceright-method) | It executes a provided function for each value from right to left and reduces the array to a single value. |
| [some()](https://www.javatpoint.com/javascript-array-some-method) | It determines if any element of the array passes the test of the implemented function. |
| [shift()](https://www.javatpoint.com/javascript-array-shift-method) | It removes and returns the first element of an array. |
| [slice()](https://www.javatpoint.com/javascript-array-slice-method) | It returns a new array containing the copy of the part of the given array. |
| [sort()](https://www.javatpoint.com/javascript-array-sort-method) | It returns the element of the given array in a sorted order. |
| [splice()](https://www.javatpoint.com/javascript-array-splice-method) | It add/remove elements to/from the given array. |
| [toLocaleString()](https://www.javatpoint.com/javascript-array-tolocalestring-method) | It returns a string containing all the elements of a specified array. |
| [toString()](https://www.javatpoint.com/javascript-array-tostring-method) | It converts the elements of a specified array into string form, without affecting the original array. |
| [unshift()](https://www.javatpoint.com/javascript-array-unshift-method) | It adds one or more elements in the beginning of the given array. |
| [values()](https://www.javatpoint.com/javascript-array-values-method) | It creates a new iterator object carrying values for each index in the array. |

# JavaScript Objects

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

**Creating Objects in JavaScript:**

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

**1) JavaScript Object by object literal:**  syntax of creating object using object literal is

object={property1:value1,property2:value2.....propertyN:valueN}

property and value is separated by : (colon).

**Example:**

<html>

<body>

<script>

emp={id:102,name:"Shyam Kumar",salary:40000}

document.write(emp.id+" "+emp.name+" "+emp.salary);

</script>

</body>

</html>

**Output:** 102 Shyam Kumar 40000

**Object Properties:**

The **name: values** pairs in JavaScript objects are called **properties**:

**Example: Property Property Value**

1. age 50

2.firstName John

**Accessing Object Properties:** You can access object properties in two ways:

*objectName.propertyName / objectName["propertyName"]*

**2) By creating instance of Object:**

The syntax of creating object directly is given below:

var  objectname=new Object();

Here, **new keyword** is used to create object.

example of

<html>

<body>

<script>

var emp=new Object();

emp.id=101;

emp.name="Ravi Malik";

emp.salary=50000;

document.write(emp.id+" "+emp.name+" "+emp.salary);

</script>

</body>

</html>

**Output:** 101 Ravi Malik 50000

**3) By using an Object constructor:**

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.The **this keyword** refers to the current object.

In a function definition, this refers to the "owner" of the function.

**Example:**

<html>

<body>

<script>

function emp(id,name,salary){

this.id=id;

this.name=name;

this.salary=salary;

}

e=new emp(103,"Vimal Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);

</script>

</body>

</html>

**Output:** 103 Vimal Jaiswal 30000

**Defining method in JavaScript Object:**

Objects can also have **methods**.Methods are **actions** that can be performed on objects.Methods are stored in properties as **function definitions**.But before defining method, we need to add property in the function with same name as method.

**Accessing Object Methods:**

**Syntax:** *objectName.methodName()*

***Example:***

<html>

<body>

<script>

function emp(id,name,salary){

this.id=id;

this.name=name;

this.salary=salary;

this.changeSalary=changeSalary;

function changeSalary(otherSalary){

this.salary=otherSalary;

}

}

e=new emp(103,"Sonoo Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);

e.changeSalary(45000);

document.write("<br>"+e.id+" "+e.name+" "+e.salary);

</script>

</body>

</html>

**Output:** 103 Sonoo Jaiswal 30000  
 103 Sonoo Jaiswal 45000

**JavaScript Object Methods:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Methods** | **Description** |
| 1 | [Object.assign()](https://www.javatpoint.com/javascript-object-assign-method) | This method is used to copy enumerable and own properties from a source object to a target object |
| 2 | [Object.create()](https://www.javatpoint.com/javascript-object-create-method) | This method is used to create a new object with the specified prototype object and properties. |
| 3 | [Object.defineProperty()](https://www.javatpoint.com/javascript-object-defineproperty-method) | This method is used to describe some behavioral attributes of the property. |
| 4 | [Object.defineProperties()](https://www.javatpoint.com/javascript-object-defineproperties-method) | This method is used to create or configure multiple object properties. |
| 5 | [Object.entries()](https://www.javatpoint.com/javascript-object-entries-method) | This method returns an array with arrays of the key, value pairs. |
| 6 | [Object.freeze()](https://www.javatpoint.com/javascript-object-freeze-method) | This method prevents existing properties from being removed. |
| 7 | [Object.getOwnPropertyDescriptor()](https://www.javatpoint.com/javascript-object-getownpropertydescriptor-method) | This method returns a property descriptor for the specified property of the specified object. |
| 8 | [Object.getOwnPropertyDescriptors()](https://www.javatpoint.com/javascript-object-getownpropertydescriptors-method) | This method returns all own property descriptors of a given object. |
| 9 | [Object.getOwnPropertyNames()](https://www.javatpoint.com/javascript-object-getownpropertynames-method) | This method returns an array of all properties (enumerable or not) found. |
| 10 | [Object.getOwnPropertySymbols()](https://www.javatpoint.com/javascript-object-getownpropertysymbols-method) | This method returns an array of all own symbol key properties. |
| 11 | [Object.getPrototypeOf()](https://www.javatpoint.com/javascript-object-getprototypeof-method) | This method returns the prototype of the specified object. |
| 12 | [Object.is()](https://www.javatpoint.com/javascript-object-is-method) | This method determines whether two values are the same value. |
| 13 | [Object.isExtensible()](https://www.javatpoint.com/javascript-objects) | This method determines if an object is extensible |
| 14 | [Object.isFrozen()](https://www.javatpoint.com/javascript-objects) | This method determines if an object was frozen. |
| 15 | [Object.isSealed()](https://www.javatpoint.com/javascript-objects) | This method determines if an object is sealed. |
| 16 | [Object.keys()](https://www.javatpoint.com/javascript-objects) | This method returns an array of a given object's own property names. |
| 17 | [Object.preventExtensions()](https://www.javatpoint.com/javascript-object-preventextensions-method) | This method is used to prevent any extensions of an object. |
| 18 | [Object.seal()](https://www.javatpoint.com/javascript-object-seal-method) | This method prevents new properties from being added and marks all existing properties as non-configurable. |
| 19 | [Object.setPrototypeOf()](https://www.javatpoint.com/javascript-object-setprototypeof-method) | This method sets the prototype of a specified object to another object. |
| 20 | [Object.values()](https://www.javatpoint.com/javascript-object-values-method) | This method returns an array of values. |

# JavaScript Loops

The **JavaScript loops** are used *to iterate the piece of code* using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

**1) JavaScript For loop:**

The **JavaScript for loop** *iterates the elements for the fixed number of times*. It should be used if number of iteration is known.

**syntax:** for (initialization; condition; increment)

{

 code to be executed

}

**Example:**

<!DOCTYPE html>

<html>

<body>

<script>

for (i=1; i<=5; i++)

{

document.write(i + "<br/>")

}

</script>

</body>

</html>

**Output:**

1  
2  
3  
4  
5

**2) JavaScript while loop:** The **JavaScript while loop** iterates the elements for the infinite number of times. It should be used if number of iteration is not known.

**syntax :** while (condition)

{

    code to be executed

}

**Example:**

<!DOCTYPE html>

<html>

<body>

<script>

var i=11;

while (i<=15)

{

document.write(i + "<br/>");

i++;

}

</script>

</body>

</html>

**output:**

11  
12  
13  
14  
15

**3) JavaScript do while loop:**

The **JavaScript do while loop** *iterates the elements for the infinite number of times* like while loop. But, code is *executed at least* once whether condition is true or false.

**syntax:** do{

    code to be executed

}while (condition);

**Example:**

<!DOCTYPE html>

<html>

<body>

<script>

var i=21;

do{

document.write(i + "<br/>");

i++;

}while (i<=25);

</script>

</body>

</html>

**Output:**

21  
22  
23  
24  
25

**4) JavaScript for in loop:**

The **JavaScript for in loop** is used *to iterate the properties of an object*.

**Conditionals**

Conditional statements are used to perform different actions based on different conditions.

**Conditional Statements:** In JavaScript we have the following conditional statements:

* Use if to specify a block of code to be executed, if a specified condition is true
* Use else to specify a block of code to be executed, if the same condition is false
* Use else if to specify a new condition to test, if the first condition is false
* Use switch to specify many alternative blocks of code to be executed

### i) JavaScript If statement: It evaluates the content only if expression is true.

### Syntax

if (*condition*) {  
  //  block of code to be executed if the condition is true}

**Output:**

<html>

<body>

<script>

var a=20;

if(a>10){

document.write("value of a is greater than 10");

}

</script>

</body>

</html>

**output:** value of a is greater than 10

**ii)JavaScript If...else Statement**: It evaluates the content whether condition is true of false.

**Syntax:**

1. if(expression){
2. //content to be evaluated if condition is true
3. }
4. else{
5. //content to be evaluated if condition is false
6. }

**Example:**

<html>

<body>

<script>

var a=20;

if(a%2==0){

document.write("a is even number");

}

else{

document.write("a is odd number");

}

</script>

</body>

</html>

**output:** a is even number

### iii)JavaScript If...else if statement:

It evaluates the content only if expression is true from several expressions.

**Syntax:**

1. if(expression1){
2. //content to be evaluated if expression1 is true
3. }
4. else if(expression2){
5. //content to be evaluated if expression2 is true
6. }
7. else if(expression3){
8. //content to be evaluated if expression3 is true
9. }
10. else{
11. //content to be evaluated if no expression is true
12. }

**Example:**

<html>

<body>

<script>

var a=20;

if(a==10){

document.write("a is equal to 10");

}

else if(a==15){

document.write("a is equal to 15");

}

else if(a==20){

document.write("a is equal to 20");

}

else{

document.write("a is not equal to 10, 15 or 20");

}

</script>

</body>

</html>

**Output:** a is equal to 20

# JavaScript Switch Statement

The **JavaScript switch statement** is used to execute one code from multiple expressions.

Use the switch statement to select one of many code blocks to be executed.

### Syntax:

switch(expression) {  
  case x:  
    *// code block*    break;  
  case y:  
    *// code block*    break;  
  default:  
    // code block  
}

This is how it works:

* The switch expression is evaluated once.
* The value of the expression is compared with the values of each case.
* If there is a match, the associated block of code is executed.
* If there is no match, the default code block is executed.

**Example:**

<!DOCTYPE html>

<html>

<body>

<script>

var grade='B';

var result;

switch(grade){

case 'A':

result="A Grade";

break;

case 'B':

result="B Grade";

break;

case 'C':

result="C Grade";

break;

default:

result="No Grade";

}

document.write(result);

</script>

</body>

</html>

**output:** B Grade

**The break Keyword**

When JavaScript reaches a break keyword, it breaks out of the switch block.

This will stop the execution inside the switch block.

It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.

#### The switch statement is fall-through i.e. all the cases will be evaluated if you don't use break statement.

**Example:**

<!DOCTYPE html>

<html>

<body>

<script>

var grade='B';

var result;

switch(grade){

case 'A':

result+=" A Grade";

case 'B':

result+=" B Grade";

case 'C':

result+=" C Grade";

default:

result+=" No Grade";

}

document.write(result);

</script>

</body>

</html>

**output:** undefined B Grade C Grade No Grade

# JavaScript Functions

**JavaScript functions** are used to perform operations. We can call JavaScript function many times to reuse the code.

#### Advantage of JavaScript function:

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don’t need to write many lines of code each time to perform a common task.

**JavaScript Function Syntax:**

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

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

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

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

**function *name*(*parameter1, parameter2, parameter3*) {  
  // *code to be executed*  
}**

Function **parameters** are listed inside the parentheses () in the function definition.Function **arguments** are the **values** received by the function when it is invoked.Inside the function, the arguments (the parameters) behave as local variables.

**Example:** <html>

<body>

<script>

function msg(){

alert("hello! this is message");

}

</script>

<input type="button" onclick="msg()" value="call function"/>

</body>

</html>

call function

**output:**

**Function Invocation:**

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

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

**JavaScript Function Arguments:** We can call function by passing arguments.

**Example**: <html>

<body>

<script>

function getcube(number){

alert(number\*number\*number);

}

</script>

<form>

<input type="button" value="click" onclick="getcube(4)"/>

</form>

</body>

</html>

**Function with Return Value:**We can call function that returns a value and use it in our program.

<html>

<body>

<script>

function getInfo(){

return "hello javatpoint! How r u?";

}

</script>

<script>

document.write(getInfo());

</script>

</body>

</html>

**output:** hello javatpoint! How r u?

**JavaScript Function Object:**

In JavaScript, the purpose of **Function constructor** is to create a new Function object. It executes the code globally. However, if we call the constructor directly, a function is created dynamically but in an unsecured way.

**Syntax:**

new Function ([arg1[, arg2[, ....argn]],] functionBody)

**Parameter**

**arg1, arg2, .... , argn** - It represents the argument used by function.

**functionBody** - It represents the function definition.

**JavaScript Function Methods:**

|  |  |
| --- | --- |
| **Method** | **Description** |
| [apply()](https://www.javatpoint.com/javascript-function-apply-method) | It is used to call a function contains this value and a single array of arguments. |
| [bind()](https://www.javatpoint.com/javascript-function-bind-method) | It is used to create a new function. |
| [call()](https://www.javatpoint.com/javascript-function-call-method) | It is used to call a function contains this value and an argument list. |
| [toString()](https://www.javatpoint.com/javascript-function-tostring-method) | It returns the result in a form of a string. |

**Example:**

<!DOCTYPE html>

<html>

<body>

<script>

var add=new Function("num1","num2","return num1+num2");

document.writeln(add(2,5));

</script>

</body>

</html>

**output:7**

# JavaScript Events

The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When [javascript](https://www.javatpoint.com/javascript-tutorial) code is included in [HTML](https://www.javatpoint.com/html-tutorial), js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.

**For example**, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

**Mouse events:**

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| click | onclick | When mouse click on an element |
| mouseover | onmouseover | When the cursor of the mouse comes over the element |
| mouseout | onmouseout | When the cursor of the mouse leaves an element |
| mousedown | onmousedown | When the mouse button is pressed over the element |
| mouseup | onmouseup | When the mouse button is released over the element |
| mousemove | onmousemove | When the mouse movement takes place. |

**Keyboard events:**

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| Keydown & Keyup | onkeydown & onkeyup | When the user press and then release the key |

**Form events:**

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| focus | onfocus | When the user focuses on an element |
| submit | onsubmit | When the user submits the form |
| blur | onblur | When the focus is away from a form element |
| change | onchange | When the user modifies or changes the value of a form element |

**Window/Document events:**

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| load | onload | When the browser finishes the loading of the page |
| unload | onunload | When the visitor leaves the current webpage, the browser unloads it |
| resize | onresize | When the visitor resizes the window of the browser |

**Click Event:**

<html>

<head> Javascript Events </head>

<body>

<script language="Javascript" type="text/Javascript">

<!--

function clickevent()

{

document.write("This is JavaTpoint");

}

//-->

</script>

<form>

<input type="button" onclick="clickevent()" value="Who's this?"/>

</form>

</body>

</html>

**Keydown Event:**

<html>

<head> Javascript Events</head>

<body>

<h2> Enter something here</h2>

<input type="text" id="input1" onkeydown="keydownevent()"/>

<script>

<!--

function keydownevent()

{

document.getElementById("input1");

alert("Pressed a key");

}

//-->

</script>

</body>

</html>

**Load event:**

<html>

<head>Javascript Events</head>

</br>

<body onload="window.alert('Page successfully loaded');">

<script>

<!--

document.write("The page is loaded successfully");

//-->

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

**Form Validating**

It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.

JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation.

Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

**Example**: In this example, we are going to validate the name and password. The name can’t be empty and password can’t be less than 6 characters long.

Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

<html>

<body>

<script>

function validateform(){

var name=document.myform.name.value;

var password=document.myform.password.value;

if (name==null || name==""){

alert("Name can't be blank");

return false;

}else if(password.length<6){

alert("Password must be at least 6 characters long.");

return false;

}

}

</script>

<body>

<form name="myform" method="post" action="http://www.javatpoint.com/javascriptpages/valid.jsp" onsubmit="return validateform()" >

Name: <input type="text" name="name"><br/>

Password: <input type="password" name="password"><br/>

<input type="submit" value="register">

</form>

</body>

</html>

**JavaScript Retype Password Validation:**

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function matchpass(){

var firstpassword=document.f1.password.value;

var secondpassword=document.f1.password2.value;

if(firstpassword==secondpassword){

return true;

}

else{

alert("password must be same!");

return false;

}

}

</script>

</head>

<body>

<form name="f1" action="http://www.javatpoint.com/javascriptpages/valid.jsp" onsubmit="return matchpass()">

Password:<input type="password" name="password" /><br/>

Re-enter Password:<input type="password" name="password2"/><br/>

<input type="submit">

</form>

</body>

</html>

**JavaScript Number Validation:**

Let's validate the textfield for numeric value only. Here, we are using isNaN() function.

<!DOCTYPE html>

<html>

<head>

<script>

function validate(){

var num=document.myform.num.value;

if (isNaN(num)){

document.getElementById("numloc").innerHTML="Enter Numeric value only";

return false;

}else{

return true;

}

}

</script>

</head>

<body>

<form name="myform" action="http://www.javatpoint.com/javascriptpages/valid.jsp" onsubmit="return validate()" >

Number: <input type="text" name="num"><span id="numloc"></span><br/>

<input type="submit" value="submit">

</form>

</body>

</html>

**JavaScript validation with image:** Let’s see an interactive JavaScript form validation example that displays correct and incorrect image if input is correct or incorrect.

<html>

<body>

<script type="text/javascript">

function validate(){

var name=document.f1.name.value;

var passwordlength=document.f1.password.value.length;

var status=false;

if(name==""){

document.getElementById("namelocation").innerHTML=

" <img src='http://www.javatpoint.com/javascriptpages/images/unchecked.gif'/> Please enter your name";

status=false;

}else{

document.getElementById("namelocation").innerHTML=" <img src='http://www.javatpoint.com/javascriptpages/images/checked.gif'/>";

status=true;

}

if(passwordlength<6){

document.getElementById("passwordlocation").innerHTML=

" <img src='http://www.javatpoint.com/javascriptpages/images/unchecked.gif'/> Password must be greater than 6";

status=false;

}else{

document.getElementById("passwordlocation").innerHTML=" <img src='http://www.javatpoint.com/javascriptpages/images/checked.gif'/>";

}

return status;

}

</script>

<form name="f1" action="http://www.javatpoint.com/javascriptpages/valid.jsp" onsubmit="return validate()">

<table>

<tr><td>Name:</td><td><input type="text" name="name"/>

<span id="namelocation" style="color:red"></span></td></tr>

<tr><td>Password:</td><td><input type="password" name="password"/>

<span id="passwordlocation" style="color:red"></span></td></tr>

<tr><td colspan="2"><input type="submit" value="register"/> </td></tr>

</table>

</form>

</body>

</html>