ktunotes.in

GET IT ON Google Play

KTU NOTES
The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

Website: www.ktunotes.in

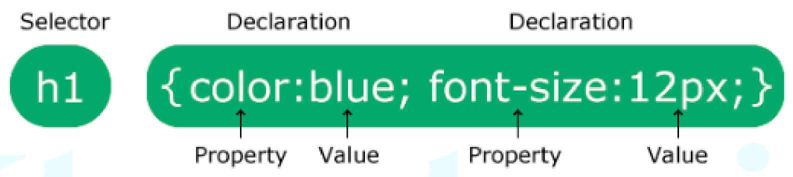
MODULE 2

STYLE WITH CSS

INTRODUCTION TO CSS

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

CSS Syntax



Ktunotes.in

The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)

DOWNLOADED FROM KTUNOTES.IN

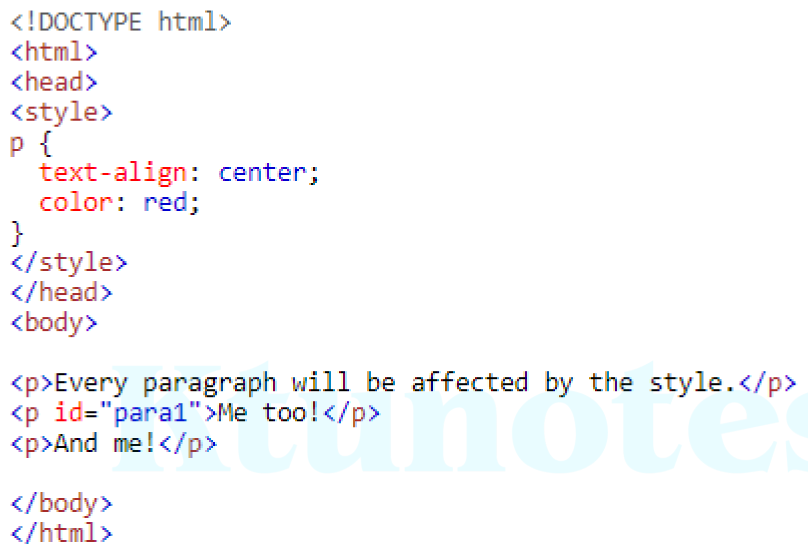
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

The CSS element Selector

The element selector selects HTML elements based on the element name.

Example:

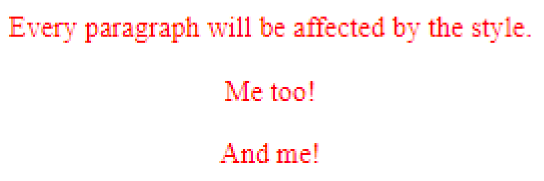
Here, all <p> elements on the page will be center-aligned, with a red text color:



```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p>Every paragraph will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

Ktunotes.in

OUTPUT:



```
Every paragraph will be affected by the style.

                 Me too!

                 And me!
```

The CSS id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

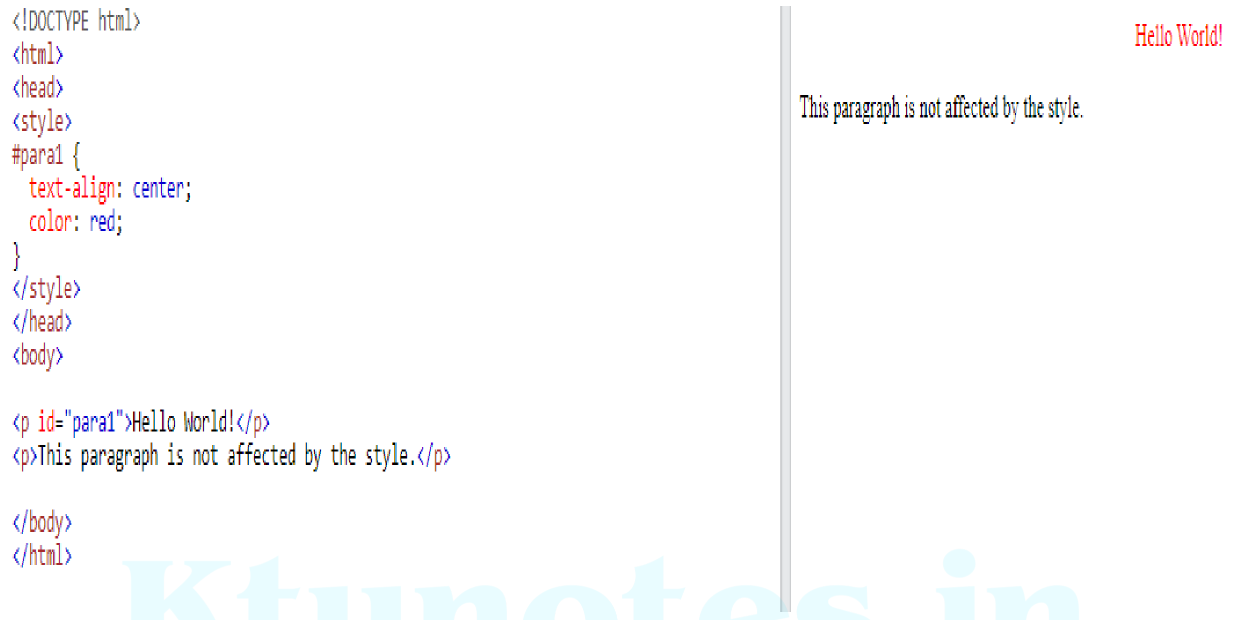
The id of an element is unique within a page, so the id selector is used to select one unique element!

DOWNLOADED FROM KTUNOTES.IN

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example:

The CSS rule below will be applied to the HTML element with id="para1":



```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>

</body>
</html>
```

Hello World!

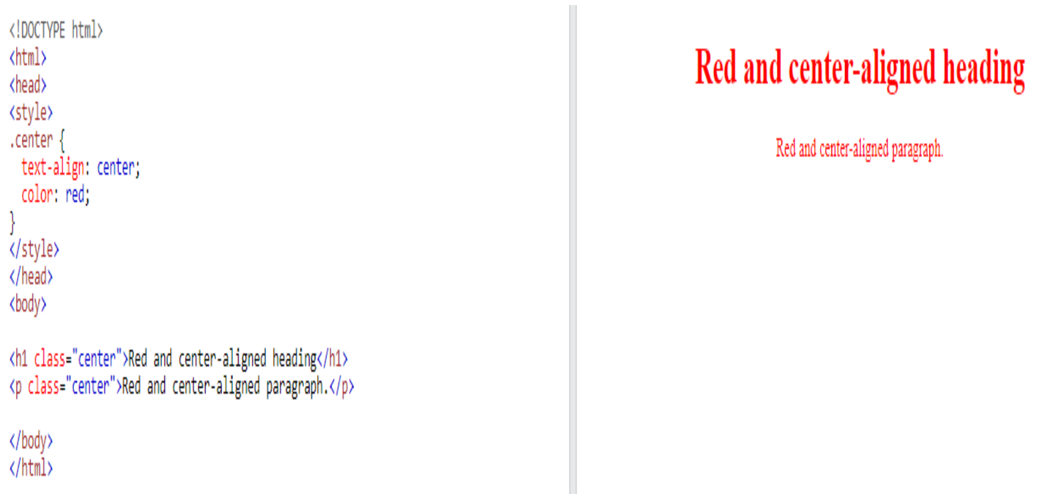
This paragraph is not affected by the style.

Ktunotes.in

The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.



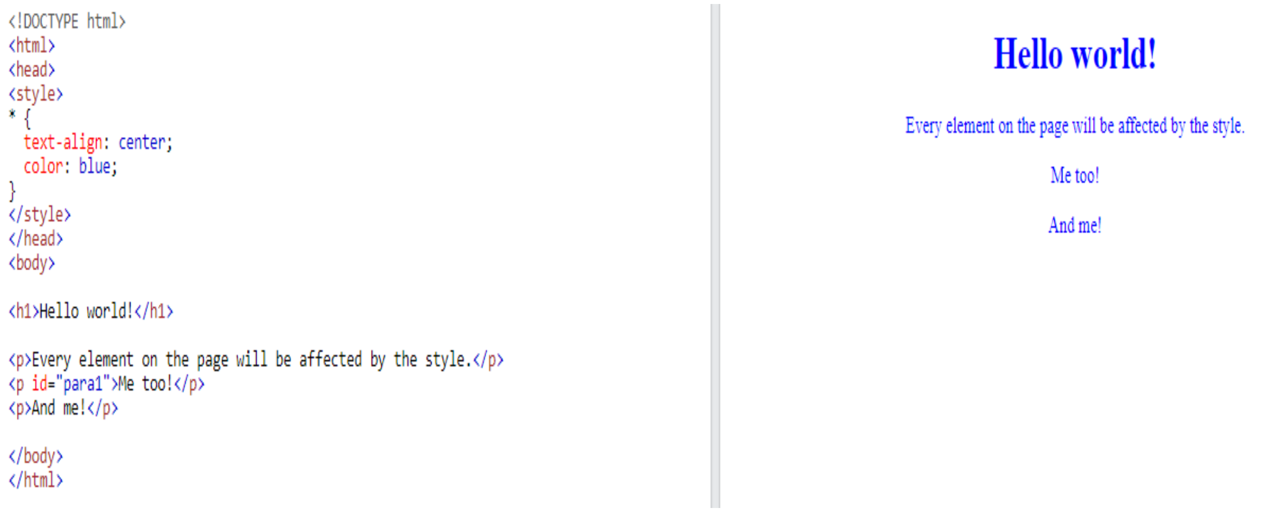
```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

Red and center-aligned heading

Red and center-aligned paragraph.

The CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.



```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>

<h1>Hello world!</h1>

<p>Every element on the page will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

```
Hello world!

Every element on the page will be affected by the style.

Me too!

And me!
```

The CSS Grouping Selector

Ktunotes.in

The grouping selector selects all the HTML elements with the same style definitions.

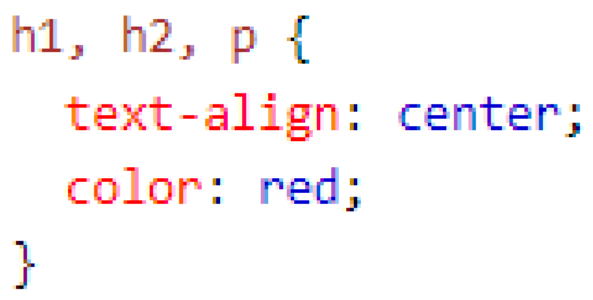
Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

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

It will be better to group the selectors, to minimize the code.

DOWNLOADED FROM KTUNOTES.IN

To group selectors, separate each selector with a comma.



```
h1, h2, p {
  text-align: center;
  color: red;
}
```

How To Add CSS In HTML

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

Ktunotes.in

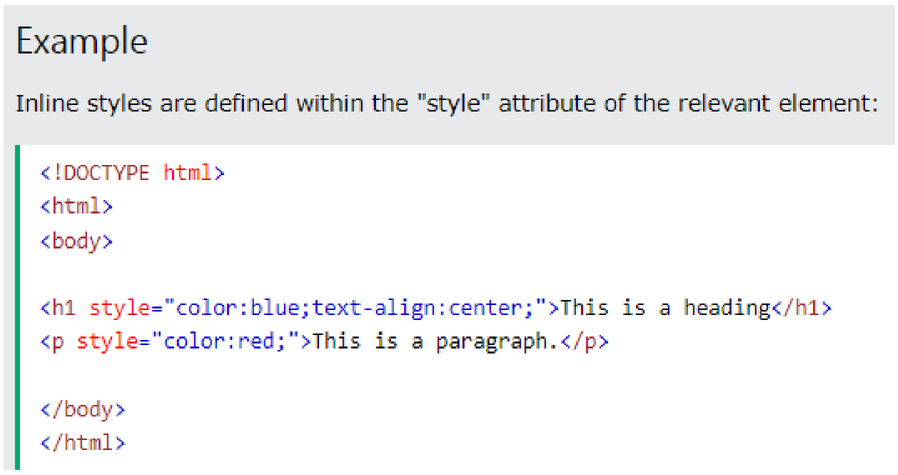
- External CSS
- Internal CSS
- Inline CSS

Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

DOWNLOADED FROM KTUNOTES.IN



Example

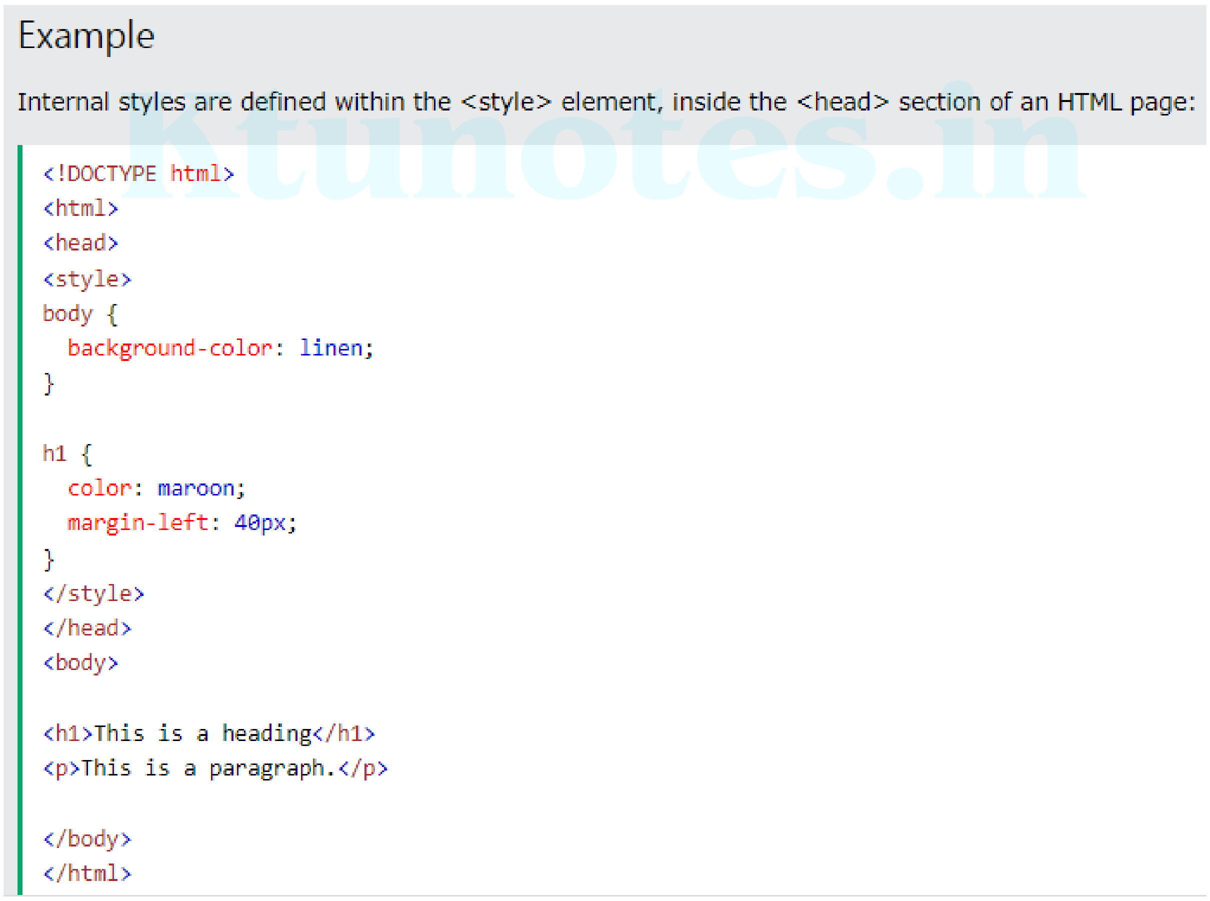
Inline styles are defined within the "style" attribute of the relevant element:

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

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

</body>
</html>
```

Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.



Example

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

ktunotes.in

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

DOWNLOADED FROM KTUNOTES.IN

External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

Example

External styles are defined within the <link> element, inside the <head> section of an HTML page:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Ktunotes.in

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

The external .css file should not contain any HTML tags.

Here is how the "mystyle.css" file looks:

"mystyle.css"

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

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

DOWNLOADED FROM KTUNOTES.IN

COLORS

CSS uses color values to specify a color. Typically, these are used to set a color either for the foreground of an element or else for the background of the element. They can also be used to affect the color of borders and other decorative effects. You can specify your color values in various formats.

CSS Colors - Hex Codes

A hexadecimal is a 6 digit representation of a color. The first two digitsRR represent a red value, the next two are a green valueGG, and the last are the blue valueBB. A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Jasc Paintshop Pro, or even using Advanced Paint Brush. Each hexadecimal code will be preceded by a pound or hash sign '#'. Following are the examples to use Hexadecimal notation.

Ex: #FFFF00

CSS Colors - RGB Values

Ktunotes.in

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

NOTE: All the browsers does not support rgb property of color so it is recommended not to use it.

Ex: rgb255, 0, 255

CSS Colors - RGBA Values

This color value is specified using the rgba(red,green,blue,alpha) property.The alpha value which represents opacity—can be any value in the range 0.0(fully transparent) through 1.0(fully opaque).

Ex: rgba(255,0,0,0.5)

CSS Colors – HSL/HSLA Values

DOWNLOADED FROM KTUNOTES.IN

- **HSL** (hue,saturation,lightness)
- **HSLA**(hue,saturation,lightness,alpha)

The hue is a color or shade expressed as a value from 0 to 359 representing the degree on a color wheel.The colors on the wheel progress in the ode of the colors of the rainbow.

The saturation- the intensity of the hue is expressed as a percentage,where 100% is fully saturated and 0% is gray.

Lightness- the intensity of light or luminance of the hue

CSS Background Color

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

CSS Text Color

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

Ktunotes.in

<u>FONT PROPPERTIES</u>

The font property is a shorthand property for:

- font-style
- font-variant
- font-weight
- font-size/line-height
- font-family
  - The font-family property specifies the font for an element.
  - The font-size property sets the size of a font.

DOWNLOADED FROM KTUNOTES.IN

Eg:

```
div.a {
 font-size: 15px;
}

div.b {
 font-size: large;
}

div.c {
 font-size: 150%;
}
```

- The font-style property specifies the font style for a text.

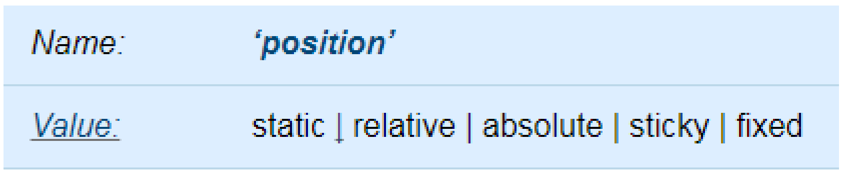
Eg:

```
p.a {
  font-style: normal;
}

p.b {
 font-style: italic;
}

p.c {
font-style: oblique;
}
```

Ktunotes.in

Layout and Positioning

position property



Name:	***'position'***
Value:	static | relative | absolute | sticky | fixed

DOWNLOADED FROM KTUNOTES.IN

The position property determines which of the ***positioning schemes*** is used to calculate the position of a box. Values other than static make the box a ***positioned box***, and cause it to establish an absolute positioning containing block for its descendants. Values have the following meanings:

static

The box is not a positioned box, and is laid out according to the rules of its parent formatting context. The inset properties do not apply.

relative

The box is laid out as for static, then offset from the resulting position. This offsetting is a purely visual effect, and does not affect the size or position of any other box, except insofar as it increases the scrollable overflow area of its ancestors. This positioning scheme is called ***relative positioning***.

sticky

Identical to relative, except that its offsets are automatically adjusted in reference to the nearest ancestor scroll container's scrollport (as modified by the inset properties) in whichever axes the inset properties are not both auto, to try to keep the box in view within its containing block as the user scrolls. This positioning scheme is called ***sticky positioning***.

absolute

The box is taken out of flow such that it has no impact on the size or position of its siblings and ancestors, and does not participate in its parent's formatting context.

Instead, the box is positioned and sized solely in reference to its absolute positioning containing block, as modified by the box's inset properties, see § 4 Absolute Positioning Layout Model. It can overlap in-flow content or other absolutely positioned elements, and is included in the scrollable overflow area of the box that generates is containing block. This positioning scheme is called ***absolute positioning***.

fixed

Same as absolute, except the box is positioned and sized relative to a fixed positioning containing block (usually the viewport in continuous media, or the page area in paged

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

media). The box's position is fixed with respect to this reference rectangle: when attached to the viewport it does not move when the document is scrolled, and when attached to the page area is replicated on every page when the document is paginated. This positioning scheme is called ***fixed positioning*** and is considered a subset of absolute positioning.

The precise location of a positioned box is controlled by the ***inset properties***: the physical inset properties top, right, bottom, left; the flow-relative inset properties inset-block-start, inset-inline-start, inset-block-end, and inset-inline-end; and their shorthands, inset-block, inset-inline, and inset.

The interpretation of these inset properties varies by positioning scheme:

- for absolute positioning, they represent insets from the containing block.
- for relative positioning, they represent insets from the box's original margin edge.
- for sticky positioning, they represent insets from the scrollport edge.

Ktunotes.in

Relative Positioning

For a relatively positioned box, the inset properties move the box inward from the respective edge, without changing its size. left moves the box to the right, right moves it to the left, etc. Since boxes are not split or stretched as a result of relative positioning opposing used values in a given axis must be negations of each other:

- If opposing inset properties in an axis both compute to auto (their initial values), their used values are zero (i.e., the boxes stay in their original position in that axis).
- If only one is auto, its used value becomes the negation of the other, and the box is shifted by the specified amount.
- If neither is auto, the position is over-constrained; (with respect to the writing mode of its containing block) the computed end side value is ignored, and its used value becomes the negation of the start side.

DOWNLOADED FROM KTUNOTES.IN

Sticky positioning

Sticky positioning is similar to relative positioning except the offsets are automatically calculated in reference to the nearest scrollport.

For a sticky positioned box, the inset properties represent insets from the respective edges of the nearest scrollport, defining the ***sticky view rectangle*** used to constrain the box's position. (For this purpose an auto value represents a zero inset.) If this results in a sticky view rectangle size in any axis less than the size of the border box of the sticky box in that axis, then the effective end-edge inset in the affected axis is reduced (possibly becoming negative) to bring the sticky view rectangle's size up to the size of the border box in that axis (where end is interpreted relative to the writing mode of the containing block).

Absolute (and Fixed) Positioning

For an absolutely positioned box, the inset properties effectively reduce the containing block into which it is sized and positioned by the specified amounts.

If only one inset property in a given axis is auto, it is set to zero. If both inset properties in a given axis are auto, then, depending on the box's self-alignment property in the relevant axis (treating normal as start and any distributed, baseline, or stretch alignment value as its fallback alignment):

- for self-start alignment or its equivalent

  Set its start-edge inset property to the static position, and its end-edge inset property to zero.

- for self-end alignment or its equivalent
  Set its end-edge inset property to the static position, and its start-edge inset property to zero.

- for center alignment
  Let *start distance* be the distance from the center of its static position rectangle to the start edge of its containing block, and *end distance* be the distance from the center of its static position rectangle to the end edge of its containing block. If *start distance* is less than or equal to *end distance*, then set the start-edge inset property to zero, and set

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

the end-edge inset property to (*containing block size* - 2 × |*start distance*|); otherwise, set the end-edge inset property to zero and the start-edge inset property to (*containing block size* - 2 × |*end distance*|).

If these adjustments result in an effective containing block size in any axis less than zero, then the weaker inset in the affected axis is reduced (possibly becoming negative) to bring that size up to zero. In the case that only one inset is auto, that is the ***weaker inset***; otherwise the weaker inset is the inset of the end edge (where end is interpreted relative to the writing mode of the containing block).

If its self-alignment property in an axis is normal, then the resolved value of its weaker inset in that axis is the value necessary to match that edge of its inset-modified containing block to the corresponding edge of its margin box after layout.

Block Layout

The static position rectangle is a zero-thickness rectangle spanning between the inline-axis sides of the box's static-position containing block; and positioned at its block-start static position.

Ktunotes.in

Inline Layout

The static position rectangle is a zero-thickness rectangle spanning between the line-over/line-under sides of the line box that would have contained its "hypothetical box"; and positioned at its inline-start static position.

Flex Layout

The static position rectangle of the child of a flex container corresponds to the content edges of the flex container.

Grid Layout

By default, the static position rectangle of the child of a grid container corresponds to the content edges of the grid container. However, if that grid container also establishes the box's actual containing block, then the grid area specified by the grid-placement properties establishes its static position rectangle instead.

DOWNLOADED FROM KTUNOTES.IN

For the purposes of calculating the static position rectangle, the containing block of fixed positioned elements is the initial containing block instead of the viewport, and all scroll containers should be assumed to be scrolled to their initial scroll position. Additionally, all auto margins on the box itself are treated as zero.

How to change image size in CSS?

Sometimes, it is required to fit an image into a certain given dimension. We can resize the image by specifying the width and height of an image. A common solution is to use the **max-width: 100%;** and **height: auto;** so that large images do not exceed the width of their container. The **max-width** and **max-height** properties of CSS works better, but they are not supported in many browsers.

Another way of resizing the image is by using the **object-fit** property, which fits the image. This CSS property specifies how a video or an image is resized to fit its content box. It defines how an element fits into the container with an established width and height.

Ktunotes.in

The **object-fit** property is generally applied to image or video. This property defines how an element responds to the width and height of its container. Mainly there are five values of **object-fit** property such as **fill, contain, cover, none, scale-down, initial**, and **inherit**. The default value of this property is **"fill"**.

Eg:

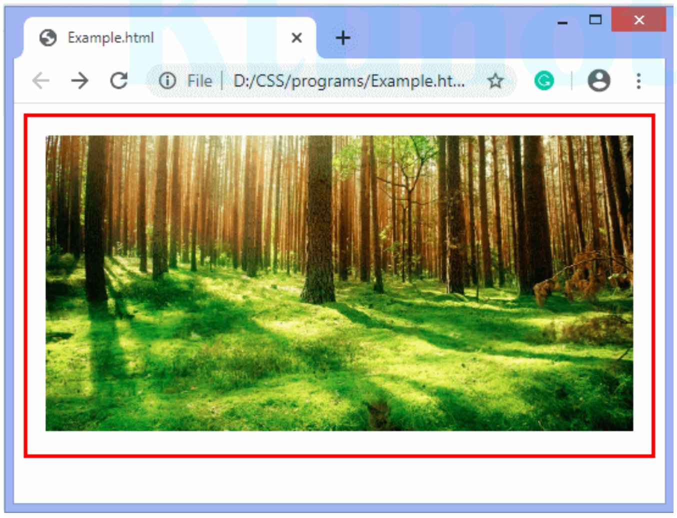
```
<!DOCTYPE html>
<html>
<head>
<title>cell padding</title>
<style>
div {
width: auto;
text-align: center;
padding: 15px;
border: 3px solid red;
```

DOWNLOADED FROM KTUNOTES.IN

```
}
img {
max-width: 100%;
height: auto;
}
</style>
</head>
<body>
<div>
<img src= "forest.jpg">
</div>
</body>
</html>
```

Ktunotes.in

Output



How to align images in CSS?

Images are an important part of any web application. Including a lot of images in a web application is generally not recommended, but it is important to use the images wherever they required. CSS helps us to control the display of images in web applications.

DOWNLOADED FROM KTUNOTES.IN

Aligning an image means to position the image at center, left and right. We can use the **float** property and **text-align** property for the alignment of images.

If the image is in the div element, then we can use the **text-align** property for aligning the image in the div.

Example

In this example, we are aligning the images by using the **text-align** property. The images are in the div element.

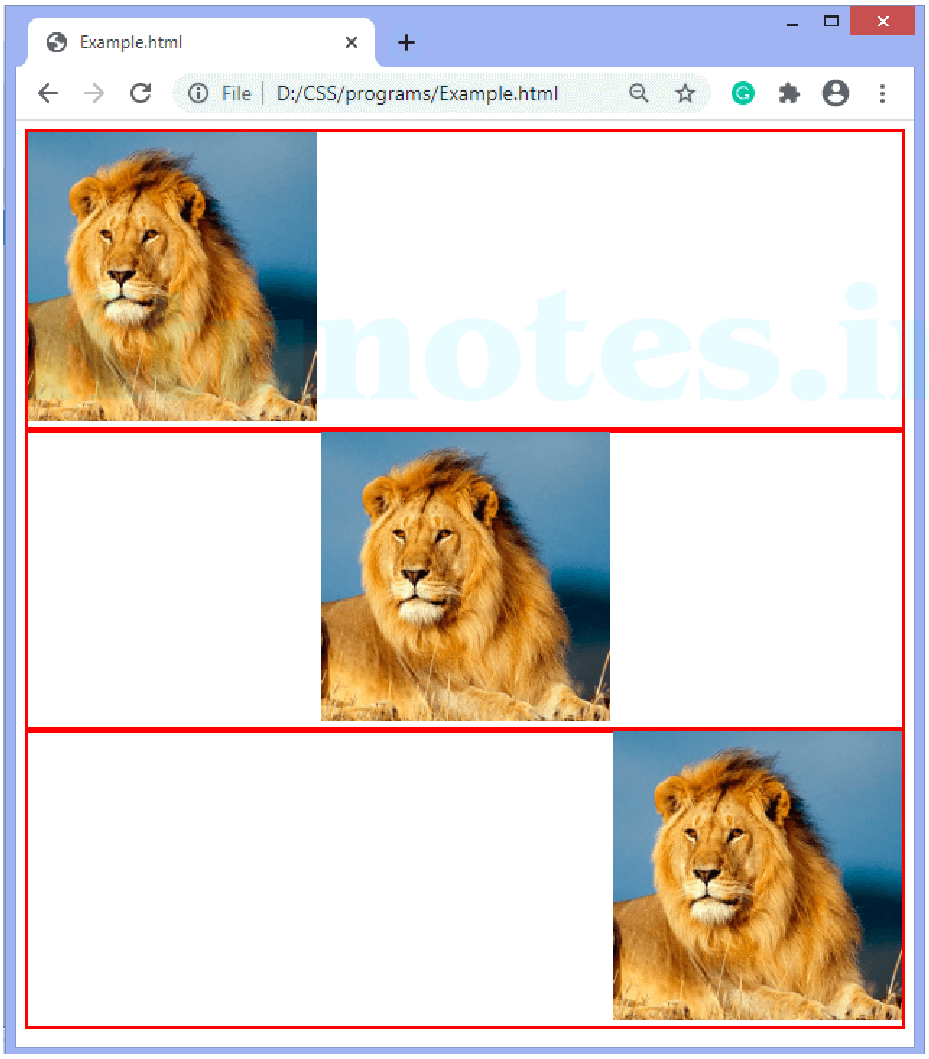
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 border: 2px solid red;
 }
 img{
 height: 250px;
 width: 250px;
 }
#left {
 text-align: left;
 }
 #center {
 text-align: center;
 }
 #right{
 text-align: right;
 }</style>
</head>
<body>
<div id ="left">
<img src="lion.png">
```

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

```
</div>
<div id ="center">
<img src="lion.png">
</div>
<div id ="right">
<img src="lion.png">
</div>
</body>
</html>
```

Output



DOWNLOADED FROM KTUNOTES.IN

Using float property

The CSS float property is a positioning property. It is used to push an element to the left or right, allowing other elements to wrap around it. It is generally used with images and layouts.

Elements are floated only horizontally. So it is possible only to float elements left or right, not up or down. A floated element may be moved as far to the left or the right as possible. Simply, it means that a floated element can display at the extreme left or extreme right.

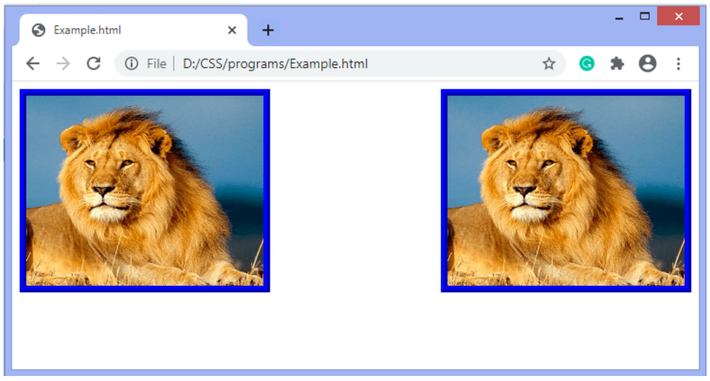
Example

```
<!DOCTYPE html>
<html>
<head>
<style>
 img{
 height: 200px;
 width: 250px;
 border: 7px ridge blue;
 }
#left{
 float: left;
 }
 #right{
 float: right;
 }</style>
</head>
<body>
<img src="lion.png" id ="left">
<img src="lion.png" id="right">
</body>
</html>
```

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

Output



How to center images in CSS?

CSS helps us to control the display of images in web applications. The centering of images or texts is a common task in CSS. To center an image, we have to set the value of **margin-left** and **margin-right** to **auto** and make it a block element by using the **display: block;** property.

Ktunotes.in

If the image is in the div element, then we can use the **text-align: center;** property for aligning the image to center in the div.

The <img> element is said to be an inline element that can easily be centered by applying the **text-align: center;** property of CSS to the parent element that contains it.

We can use the shorthand property **margin** and set it to **auto** for aligning the image at the center, rather than using the **margin-left** and **margin-right** property.

Let's see how to center an image by applying **text-align: center;** property to its parent element.

Example

In this example, we are aligning the images by using the **text-align: center;** property. The image is in the div element.

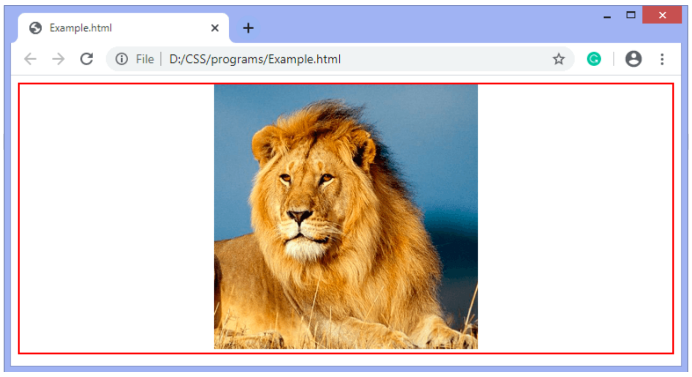
1. <!DOCTYPE html>

DOWNLOADED FROM KTUNOTES.IN

```
<html>
<head>
<style>
div {
border: 2px solid red;
}
img{
height: 300px;
width: 300px;
}
#center {
text-align: center;
}
</style>
</head>
<body>
<div id ="center">
<img src="lion.png">
</div>
</body>
</html>
```

Ktunotes.in

Output



DOWNLOADED FROM KTUNOTES.IN

How to add background image in CSS?

The **background-image** property in CSS is used to set an image as the background of an element. Using this CSS property, we can set one or more than one background image for an element.

By default, the image is positioned at the top-left corner of an element and repeated both horizontally as well as vertically. The background image should be chosen according to the text color. The bad combination of text and background image may be a cause of poorly designed and not readable webpage.

The **url()** value of this property allows us to include a file path to any image. It will show the element's background. We can use multiple images or a mixture of gradients and images for the background. If the background-image is failed to load or if we are using the gradients, but they are not supported on the corresponding browser then, we can use the fallback value (the value used as the substitution) as the background color of the element.

Syntax

background-image: url();

Ktunotes.in

Values

url(): It is the URL to the image. We can separate the URLs by a comma if we want to specify more than one image.

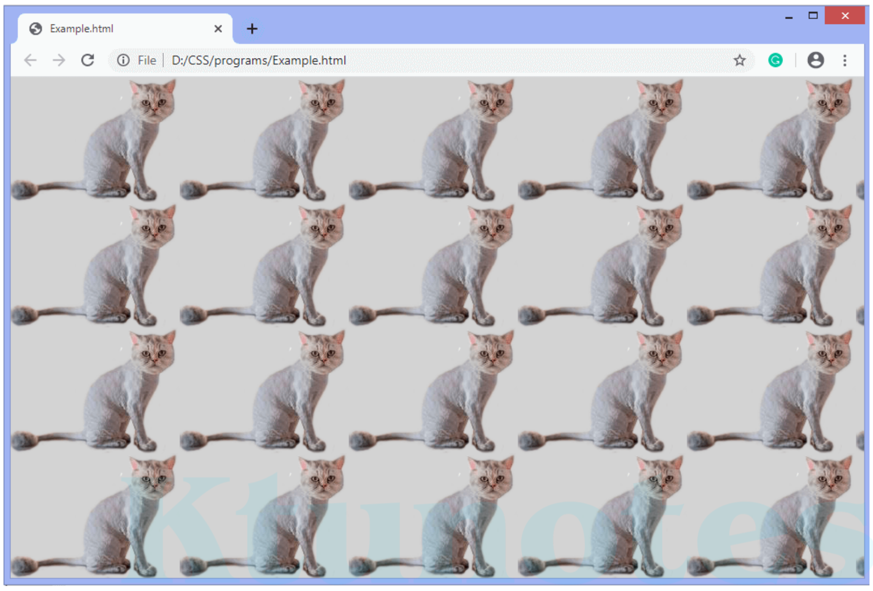
Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("cat.png");
  background-color: lightgray;
  }
</style>
```

DOWNLOADED FROM KTUNOTES.IN

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

Output



Ktunotes.in

<u>CSS Gradient</u>

CSS gradient is used to display smooth transition within two or more specified colors.

Why CSS Gradient

These are the following reasons to use CSS gradient.

- You don't have to use images to display transition effects.
- The download time and bandwidth usage can also be reduced.
- It provides better look to the element when zoomed, because the gradient is generated by the browser.

There are two types of gradient in CSS3.

DOWNLOADED FROM KTUNOTES.IN

1. Linear gradients
2. Radial gradients

1) CSS Linear Gradient

The CSS3 linear gradient goes up/down/left/right and diagonally. To create a CSS3 linear gradient, you must have to define two or more color stops. The color stops are the colors which are used to create a smooth transition. Starting point and direction can also be added along with the gradient effect.

2) CSS Radial Gradient

You must have to define at least two color stops to create a radial gradient. It is defined by its center.

Media Queries

Media queries are a way to target browser by certain characteristics, features, and user preferences, then apply styles or run other code based on those things. Perhaps the most common media queries in the world are those that target particular viewport ranges and apply custom styles, which birthed the whole idea of responsive design.

Ktunotes.in

There are lots of other things we can target beside viewport width. That might be screen resolution, device orientation, operating system preference, or even more among a whole bevy of things we can query and use to style content.

DOWNLOADED FROM KTUNOTES.IN

INTRODUCTION TO JAVASCRIPT

JavaScript is a programming language we can use to make a website interactive. When we search something on Google or click a link, our website changes — that's what JavaScript allows us to do.

JavaScript is used to create client-side dynamic pages.

JavaScript is *an object-based scripting language* which is lightweight and cross-platform.

JavaScript is not a compiled language, but it is a translated language. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.

What is JavaScript

Ktunotes.in

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

There are following features of JavaScript:

DOWNLOADED FROM KTUNOTES.IN

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.

Application of JavaScript

Ktunotes.in

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

```
<script>
document.write("Hello JavaScript by JavaScript");
</script>
```

DOWNLOADED FROM KTUNOTES.IN

JavaScript Comment

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

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

Advantages of JavaScript comments

There are mainly two advantages of JavaScript comments.

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

Types of JavaScript Comments

Ktunotes.in

There are two types of comments in JavaScript.

1. Single-line Comment
2. Multi-line Comment

JavaScript Single line Comment

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

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

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

DOWNLOADED FROM KTUNOTES.IN

JavaScript Multi line Comment

It can be used to add single as well as multi line comments. So, it is more convenient.

It is represented by forward slash with asterisk then asterisk with forward slash. For example:

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

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

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

JavaScript Variable

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

Ktunotes.in

There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

Correct JavaScript variables

```
var x = 10;
var _value="sonoo";
```

Incorrect JavaScript variables

```
var  123=30;
```

DOWNLOADED FROM KTUNOTES.IN

```
2. var *aa=320;
```

Example of JavaScript variable

Let’s see a simple example of JavaScript variable.

```
<script>
var x = 10;
var y = 20;
var z=x+y;
document.write(z);
</script>
```

Output of the above example

```
30
```

Ktunotes.in

Javascript Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

```
var a=40;//holding number
var b="Rahul";//holding string
```

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

Ktunotes.in

JavaScript non-primitive data types

The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

JavaScript Operators

JavaScript operators are symbols that are used to perform operations on operands. For example:

DOWNLOADED FROM KTUNOTES.IN

```
1. var sum=10+20;
```

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

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

JavaScript Arithmetic Operators

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

Ktunotes.in

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11

DOWNLOADED FROM KTUNOTES.IN

--	Decrement	var a=10; a--; Now a = 9

JavaScript Comparison Operators

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

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

Ktunotes.in

JavaScript Bitwise Operators

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

DOWNLOADED FROM KTUNOTES.IN

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

Ktunotes.in

JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

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

JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

DOWNLOADED FROM KTUNOTES.IN

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

JavaScript Special Operators

Ktunotes.in

The following operators are known as JavaScript special operators.

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

DOWNLOADED FROM KTUNOTES.IN

New	creates an instance (object)
Typeof	checks the type of object.
Void	it discards the expression's return value.
Yield	checks what is returned in a generator by the generator's iterator.

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.

Ktunotes.in

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

The syntax of declaring function is given below.

```
function functionName([arg1, arg2, ...argN]){
 //code to be executed
}
```

JavaScript Functions can have 0 or more arguments.

JavaScript Function Example

DOWNLOADED FROM KTUNOTES.IN

Let’s see the simple example of function in JavaScript that does not has arguments.

```
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
```

JavaScript Function Arguments

We can call function by passing arguments. Let’s see the example of function that has one argument.

```
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcube(4)"/>
</form>
```

Ktunotes.in

Function with Return Value

We can call function that returns a value and use it in our program. Let’s see the example of function that returns value.

```
<script>
function getInfo(){
return "hello javatpoint! How r u?";
}
</script>
<script>
```

DOWNLOADED FROM KTUNOTES.IN

```
7. document.write(getInfo());
8. </script>
```

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

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

Parameter

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

functionBody - It represents the function definition.

Ktunotes.in

JavaScript Function Methods

Let's see function methods with description.

Method	**Description**
apply()	It is used to call a function contains this value and a single array of arguments.
bind()	It is used to create a new function.
call()	It is used to call a function contains this value and an argument list.
toString()	It returns the result in a form of a string.

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.

DOWNLOADED FROM KTUNOTES.IN

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

The syntax of creating object using object literal is given below:

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

Ktunotes.in

As you can see, property and value is separated by : (colon).

Let’s see the simple example of creating object in JavaScript.

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

Output of the above example

```
102 Shyam Kumar 40000
```

2) By creating instance of Object

DOWNLOADED FROM KTUNOTES.IN

The syntax of creating object directly is given below:

```
var objectname=new Object();
```

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

Let’s see the example of creating object directly.

```
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

Output of the above example

```
101 Ravi 50000
```

Ktunotes.in

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.

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

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
```

DOWNLOADED FROM KTUNOTES.IN

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

document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

Output of the above example

```
103 Vimal Jaiswal 30000
```

JavaScript Array

JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

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

Ktunotes.in

1) JavaScript array literal

The syntax of creating array using array literal is given below:

```
var arrayname=[value1,value2..... valueN];
```

As you can see, values are contained inside [ ] and separated by , (comma).

Let's see the simple example of creating and using array in JavaScript.

```
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

Test it Now

DOWNLOADED FROM KTUNOTES.IN

The .length property returns the length of an array.

Output of the above example

```
Sonoo
Vimal
Ratan
```

2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

```
var arrayname=new Array();
```

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

Let's see the example of creating array directly.

```
<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>
```

Output of the above example

```
Arun
Varun
John
```

3) JavaScript array constructor (new keyword)

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

Here, you need 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.

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Output of the above example

```
Jai
Vijay
Smith
```

Ktunotes.in

JavaScript Array Methods

Let's see the list of JavaScript array methods with their description.

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
copywithin()	It copies the part of the given array with its own elements and returns the modif array.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided funct conditions.

DOWNLOADED FROM KTUNOTES.IN

flat()	It creates a new array carrying sub-array elements concatenated recursively till specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a n array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
filter()	It returns the new array containing the elements that pass the provided funct conditions.
find()	It returns the value of the first element in the given array that satisfies the specif condition.
findIndex()	It returns the index value of the first element in the given array that satisfies specified condition.
forEach()	It invokes the provided function once for each element of an array.
includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the f match.
isArray()	It tests if the passed value ia an array.
join()	It joins the elements of an array as a string.
keys()	It creates an iterator object that contains only the keys of the array, then loops thro

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

	these keys.
lastIndexOf()	It searches the specified element in the given array and returns the index of the match.
map()	It calls the specified function for every array element and returns the new array
of()	It creates a new array from a variable number of arguments, holding any type argument.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
reduce(function, initial)	It executes a provided function for each value from left to right and reduces the arra a single value.
reduceRight()	It executes a provided function for each value from right to left and reduces the arra a single value.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

toLocaleString()	It returns a string containing all the elements of a specified array.
toString()	It converts the elements of a specified array into string form, without affecting original array.
unshift()	It adds one or more elements in the beginning of the given array.
values()	It creates a new iterator object carrying values for each index in the array.

JavaScript String

The **JavaScript string** is an object that represents a sequence of characters.

There are 2 ways to create string in JavaScript

1. By string literal
2. By string object (using new keyword)

Ktunotes.in

1) By string literal

The string literal is created using double quotes. The syntax of creating string using string literal is given below:

```
var stringname="string value";
```

Let's see the simple example of creating string literal.

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

Output:

```
This is string literal
```

2) By string object (using new keyword)

The syntax of creating string object using new keyword is given below:

```
1. var stringname=new String("string literal");
```

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

Let's see the example of creating string in JavaScript by new keyword.

```
1. <script>
2. var stringname=new String("hello javascript string");
3. document.write(stringname);
4. </script>
```

Output:

```
hello javascript string
```

Ktunotes.in

JavaScript String Methods

Let's see the list of JavaScript string methods with examples.

Methods	Description
charAt()	It provides the char value present at the specified index.
charCodeAt()	It provides the Unicode value of a character present at the specified index.
concat()	It provides a combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.

DOWNLOADED FROM KTUNOTES.IN

lastIndexOf()	It provides the position of a char value present in the given string by searchin character from the last position.
search()	It searches a specified regular expression in a given string and returns its position match occurs.
match()	It searches a specified regular expression in a given string and returns that reg expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified start position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as negative index.
toLowerCase()	It converts the given string into lowercase letter.
toLocaleLowerCase()	It converts the given string into lowercase letter on the basis of host?s current locale.
toUpperCase()	It converts the given string into uppercase letter.
toLocaleUpperCase()	It converts the given string into uppercase letter on the basis of host?s current locale.
toString()	It provides a string representing the particular object.
valueOf()	It provides the primitive value of string object.

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.

JavaScript Number Object

The **JavaScript number** object *enables you to represent a numeric value*. It may be integer or floating-point. JavaScript number object follows IEEE standard to represent the floating- point numbers.

By the help of Number() constructor, you can create number object in JavaScript. For example:

```
1. var n=new Number(value);
```

If value can't be converted to number, it returns NaN(Not a Number) that can be checked by isNaN() method.

You can direct assign a number to a variable also. For example:

```
1.  var x=102;//integer value
2.  var y=102.7;//floating point value
3.  var z=13e4;//exponent value, output: 130000
4.  var n=new Number(16);//integer value by number object
```

Output:

```
102 102.7 130000 16
```

JavaScript Number Constants

Let's see the list of JavaScript number constants with description.

Constant	Description

Ktunotes.in

MIN_VALUE	returns the largest minimum value.
MAX_VALUE	returns the largest maximum value.
POSITIVE_INFINITY	returns positive infinity, overflow value.
NEGATIVE_INFINITY	returns negative infinity, overflow value.
NaN	represents "Not a Number" value.

JavaScript Number Methods

Let's see the list of JavaScript number methods with their description.

Ktunotes.in

Methods	Description
isFinite()	It determines whether the given value is a finite number.
isInteger()	It determines whether the given value is an integer.
parseFloat()	It converts the given string into a floating point number.
parseInt()	It converts the given string into an integer number.
toExponential()	It returns the string that represents exponential notation of the given number.
toFixed()	It returns the string that represents a number with exact digits after a decimal point.
toPrecision()	It returns the string representing a number of specified precision.

DOWNLOADED FROM KTUNOTES.IN

toString()	It returns the given number in the form of string.

JavaScript Math

The **JavaScript math** object provides several constants and methods to perform mathematical operation. Unlike date object, it doesn't have constructors.

JavaScript Math Methods

Let's see the list of JavaScript Math methods with description.

Methods	Description
abs()	It returns the absolute value of the given number.
acos()	It returns the arccosine of the given number in radians.
asin()	It returns the arcsine of the given number in radians.
atan()	It returns the arc-tangent of the given number in radians.
cbrt()	It returns the cube root of the given number.
ceil()	It returns a smallest integer value, greater than or equal to the given number.
cos()	It returns the cosine of the given number.
cosh()	It returns the hyperbolic cosine of the given number.
exp()	It returns the exponential form of the given number.

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

floor()	It returns largest integer value, lower than or equal to the given number.
hypot()	It returns square root of sum of the squares of given numbers.
log()	It returns natural logarithm of a number.
max()	It returns maximum value of the given numbers.
min()	It returns minimum value of the given numbers.
pow()	It returns value of base to the power of exponent.
random()	It returns random number between 0 (inclusive) and 1 (exclusive).
round()	It returns closest integer value of the given number.
sign()	It returns the sign of the given number
sin()	It returns the sine of the given number.
sinh()	It returns the hyperbolic sine of the given number.
sqrt()	It returns the square root of the given number
tan()	It returns the tangent of the given number.
tanh()	It returns the hyperbolic tangent of the given number.
trunc()	It returns an integer part of the given number.

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

JavaScript Date Object

The **JavaScript date** object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

Constructor

You can use 4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

Ktunotes.in

JavaScript Date Methods

Let's see the list of JavaScript date methods with their description.

Methods	Description
getDate()	It returns the integer value between 1 and 31 that represents the day for the specif date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.

DOWNLOADED FROM KTUNOTES.IN

getHours()	It returns the integer value between 0 and 23 that represents the hours on the basi local time.
getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basi local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basi local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basi local time.
getUTCDate()	It returns the integer value between 1 and 31 that represents the day for the specif date on the basis of universal time.
getUTCDay()	It returns the integer value between 0 and 6 that represents the day of the week on basis of universal time.
getUTCFullYears()	It returns the integer value that represents the year on the basis of universal time.
getUTCHours()	It returns the integer value between 0 and 23 that represents the hours on the basi universal time.
getUTCMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basi universal time.
getUTCMonth()	It returns the integer value between 0 and 11 that represents the month on the basi universal time.

Ktunotes.in

getUTCSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basi universal time.
setDate()	It sets the day value for the specified date on the basis of local time.
setDay()	It sets the particular day of the week on the basis of local time.
setFullYears()	It sets the year value for the specified date on the basis of local time.
setHours()	It sets the hour value for the specified date on the basis of local time.
setMilliseconds()	It sets the millisecond value for the specified date on the basis of local time.
setMinutes()	It sets the minute value for the specified date on the basis of local time.
setMonth()	It sets the month value for the specified date on the basis of local time.
setSeconds()	It sets the second value for the specified date on the basis of local time.
setUTCDate()	It sets the day value for the specified date on the basis of universal time.
setUTCDay()	It sets the particular day of the week on the basis of universal time.
setUTCFullYears()	It sets the year value for the specified date on the basis of universal time.
setUTCHours()	It sets the hour value for the specified date on the basis of universal time.
setUTCMilliseconds()	It sets the millisecond value for the specified date on the basis of universal time.
setUTCMinutes()	It sets the minute value for the specified date on the basis of universal time.

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

setUTCMonth()	It sets the month value for the specified date on the basis of universal time.
setUTCSeconds()	It sets the second value for the specified date on the basis of universal time.
toDateString()	It returns the date portion of a Date object.
toISOString()	It returns the date in the form ISO format string.
toJSON()	It returns a string representing the Date object. It also serializes the Date object dur JSON serialization.
toString()	It returns the date in the form of string.
toTimeString()	It returns the time portion of a Date object.
toUTCString()	It converts the specified date in the form of string using UTC time zone.
valueOf()	It returns the primitive value of a Date object.

Ktunotes.in

Decision Making and Loops

Decision Making

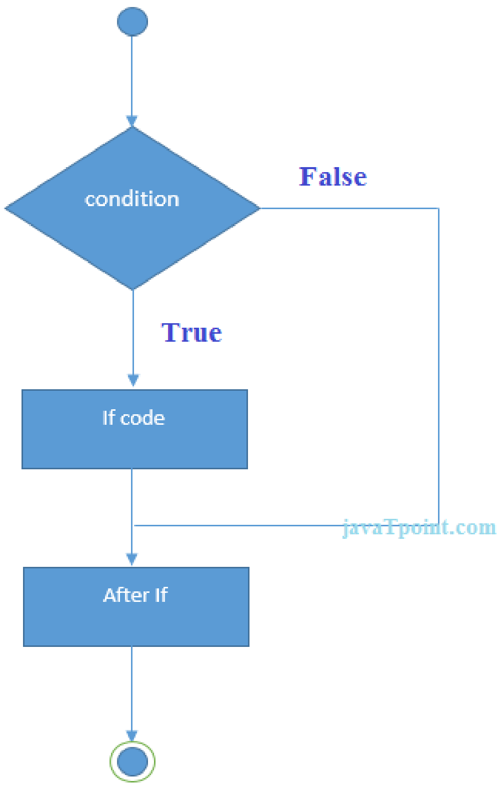
JavaScript If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

```
1. if(expression){
2. //content to be evaluated
3. }
```

DOWNLOADED FROM KTUNOTES.IN

Flowchart of JavaScript If statement



Ktunotes.in

Let’s see the simple example of if statement in javascript.

```
<script>
var a=20;
if(a>10){
document.write("value of a is greater than 10");
}
</script>
```

Output of the above example

```
value of a is greater than 10
```

JavaScript If...else Statement

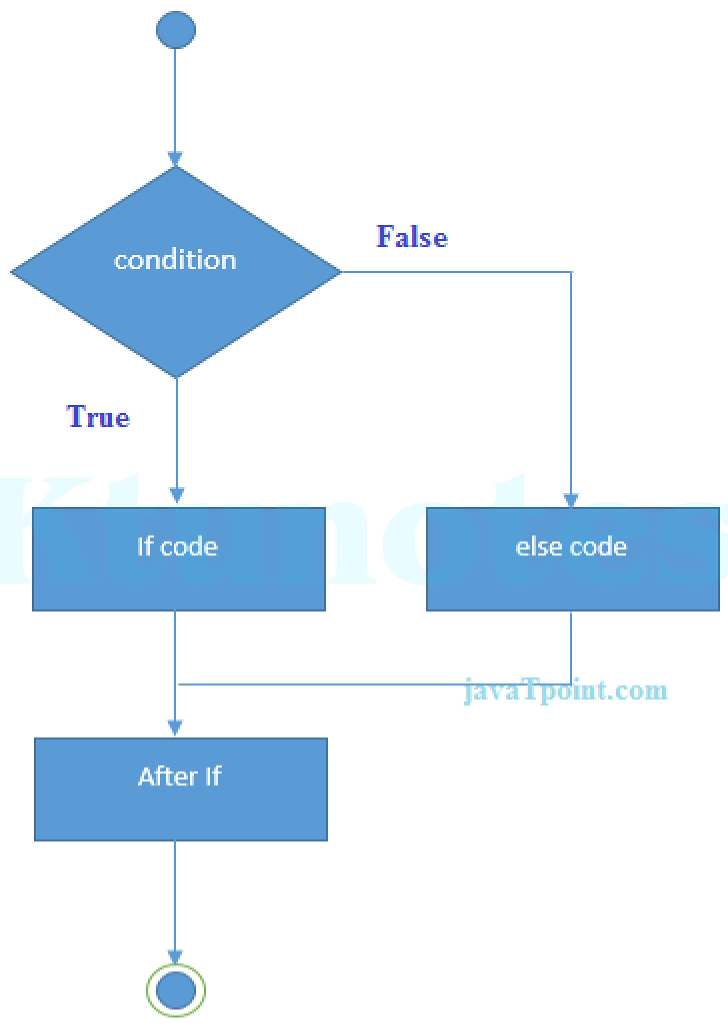
It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

```
if(expression){
```

DOWNLOADED FROM KTUNOTES.IN

```
//content to be evaluated if condition is true
}
else{
//content to be evaluated if condition is false
}
```

Flowchart of JavaScript If...else statement



Let’s see the example of if-else statement in JavaScript to find out the even or odd number.

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
```

DOWNLOADED FROM KTUNOTES.IN

```
else{
document.write("a is odd number");
}
</script>
```

Output of the above example

```
a is even number
```

JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

```
if(expression1){
//content to be evaluated if expression1 is true
}
else if(expression2){
//content to be evaluated if expression2 is true
}
else if(expression3){
//content to be evaluated if expression3 is true
}
else{
//content to be evaluated if no expression is true
}
```

Let’s see the simple example of if else if statement in javascript.

```
<script>
var a=20;
if(a==10){
document.write("a is equal to 10");
}
```

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

```
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>
```

Output of the above example

```
a is equal to 20
```

JavaScript Switch

ktunotes.in

The **JavaScript switch statement** is used *to execute one code from multiple expressions*. It is just like else if statement that we have learned in previous page. But it is convenient than *if..else..if* because it can be used with numbers, characters etc.

The signature of JavaScript switch statement is given below.

```
switch(expression){
case value1:
 code to be executed;
 break;
case value2:
 code to be executed;
 break;
......

default:
 code to be executed if above values are not matched;
}
```

DOWNLOADED FROM KTUNOTES.IN

Let's see the simple example of switch statement in javascript.

```
<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>
```

Output of the above example

```
B Grade
```

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

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN

3. do-while 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. The syntax of for loop is given below.

```
for (initialization; condition; increment)
{
    code to be executed
}
```

Let’s see the simple example of for loop in javascript.

```
<script>
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
```

Ktunotes.in

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. The syntax of while loop is given below.

```
while (condition)
```

DOWNLOADED FROM KTUNOTES.IN

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

Let’s see the simple example of while loop in javascript.

```
<script>
var i=11;
while (i<=15)
{
document.write(i + "<br/>");
 i++;
}
</script>
```

Output:

Ktunotes.in

```
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. The syntax of do while loop is given below.

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

DOWNLOADED FROM KTUNOTES.IN

Let's see the simple example of do while loop in javascript.

```
<script>
var i=21;
do{
document.write(i + "<br/>");
i++;
}while (i<=25);
</script>
```

Output:

```
21
22
23
24
25
```

Ktunotes.in

Browser Object Model

The **Browser Object Model** (BOM) is used to interact with the browser.

The default object of browser is window means you can call all the functions of window by specifying window or directly. For example:

```
window.alert("hello javatpoint");
```

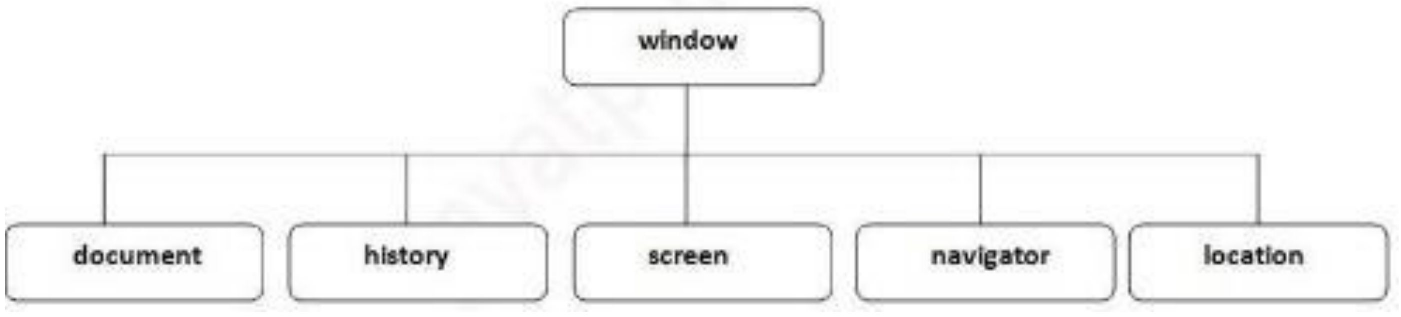
is same as:

```
alert("hello javatpoint");
```

You can use a lot of properties (other objects) defined underneath the window object like document, history, screen, navigator, location, innerHeight, innerWidth,

DOWNLOADED FROM KTUNOTES.IN

Note: The document object represents an html document. It forms DOM (Document Object Model).



Document Object Model

The **document object** represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

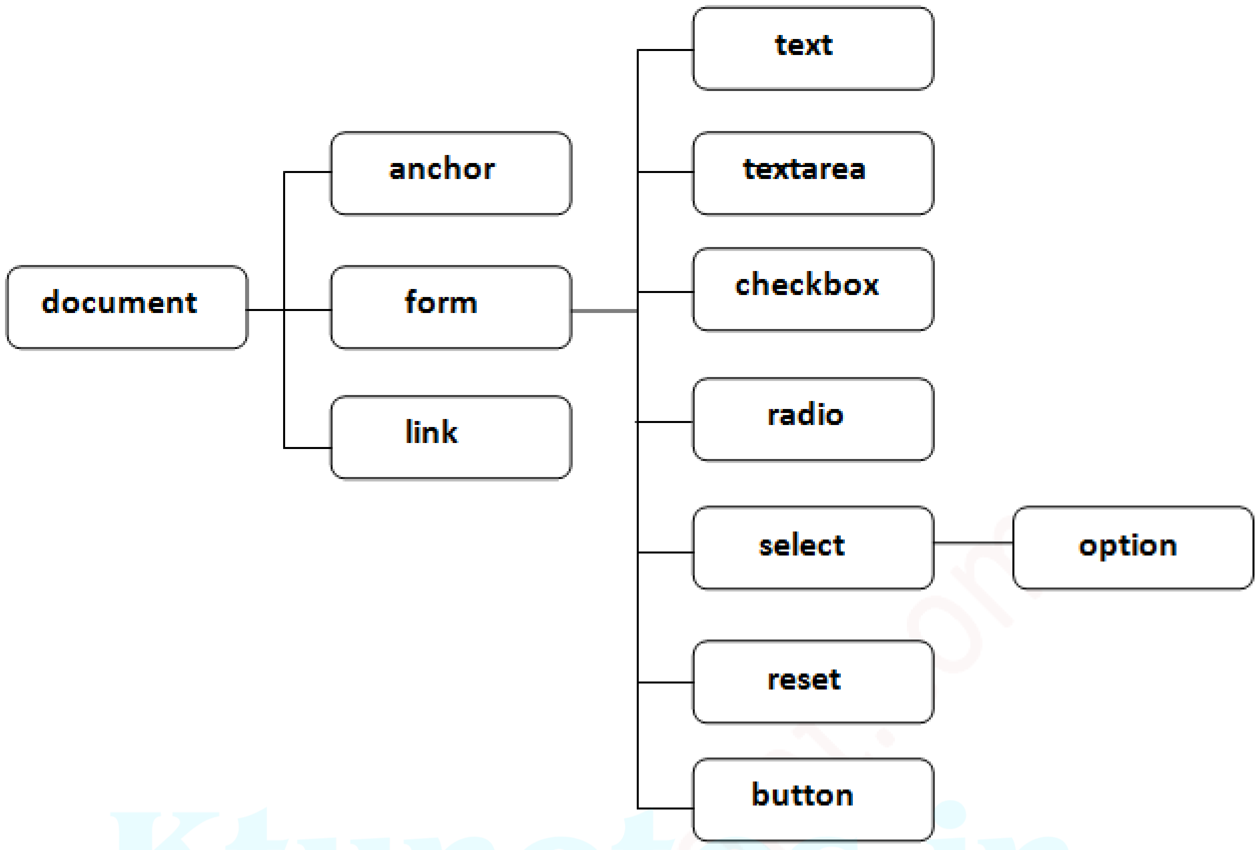
Ktunotes.in

Properties of document object

Let's see the properties of document object that can be accessed and modified by the document

DOWNLOADED FROM KTUNOTES.IN

object.



Ktunotes.in

Methods of document object

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

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.

DOWNLOADED FROM KTUNOTES.IN

getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

Ktunotes.in

DOWNLOADED FROM KTUNOTES.IN