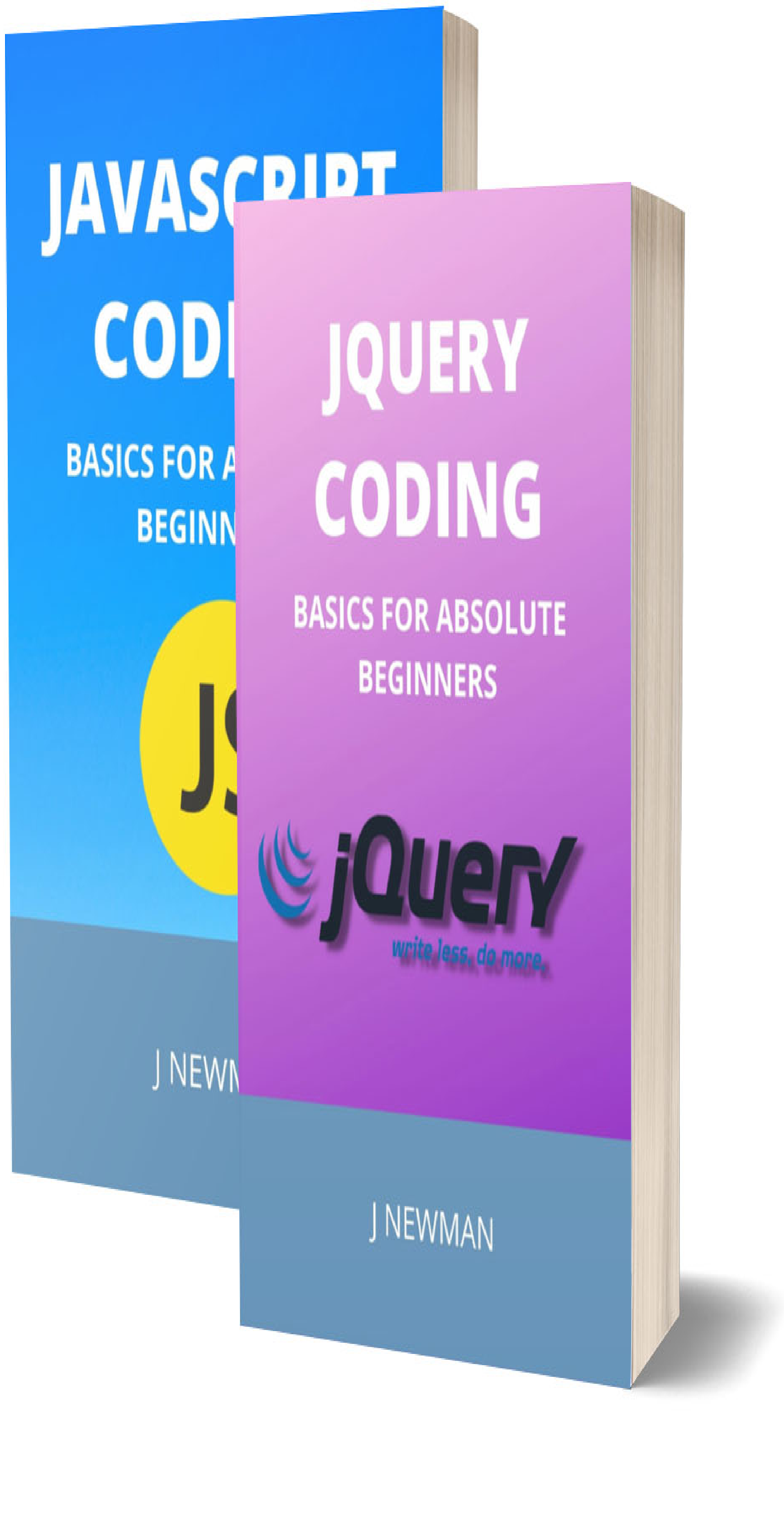

JAVASCRIPT
CODI
BASICS FOR A
BEGINN
JS
J NEWM
JQUERY
CODING
BASICS FOR ABSOLUTE
BEGINNERS
jQuery
write less, do more.
J NEWMAN

JQUERY AND JAVASCRIPT CODING:

BASICS FOR ABSOLUTE BEGINNERS

STEP BY STEP GUIDE TO LEARN CODING QUICKLY

J NEWMAN

JQUERY CODING:
BASICS FOR ABSOLUTE BEGINNERS

STEP BY STEP GUIDE TO LEARN CODING QUICKLY

J NEWMAN

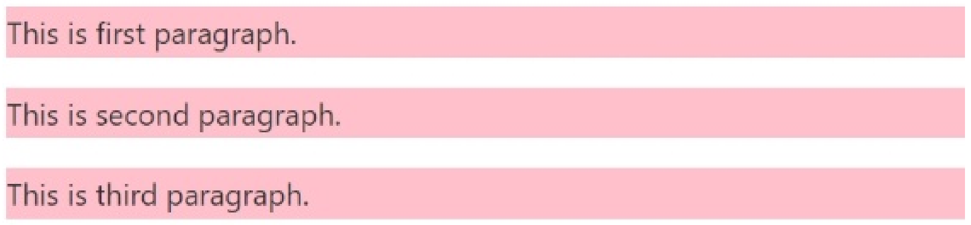
jQuery

- jQuery is a small and lightweight JavaScript library.
- jQuery is cross-platform.
- jQuery means "write less do more".
- jQuery simplifies AJAX call and DOM manipulation.

jQuery Example

```
<!DOCTYPE html>
<html>
<head>
 <title>First jQuery Example</title>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
 </script>
 <script type="text/javascript" language="javascript">
 $(document).ready(function() {
 $("p").css("background-color", "pink");
 });
 </script>
 </head>
<body>
<p>This is first paragraph.</p>
<p>This is second paragraph.</p>
<p>This is third paragraph.</p>
</body>
</html>
```

Output:



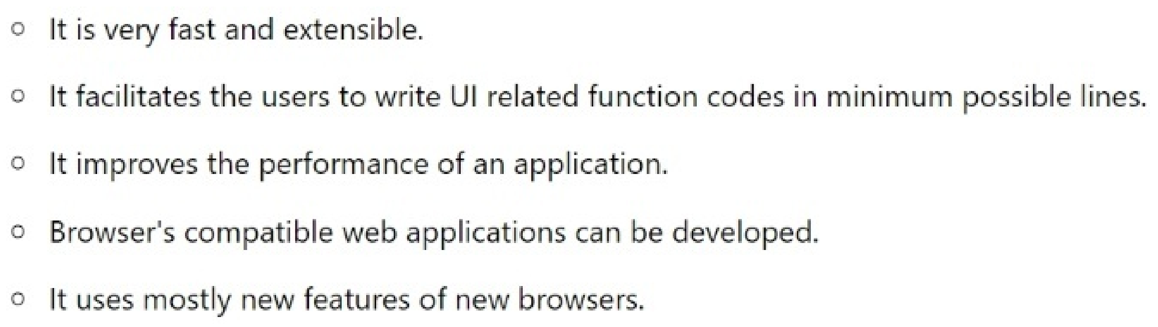
```
This is first paragraph.

This is second paragraph.

This is third paragraph.
```

jQuery Features

- HTML manipulation
- DOM manipulation
- DOM element selection
- CSS manipulation
- Effects and Animations
- Utilities
- AJAX
- HTML event methods
- JSON Parsing
- Extensibility through plug-ins

Why jQuery is required

A question may emerge as to why jQuery is required or what difference it makes to use jQuery instead of AJAX/ JavaScript. If jQuery takes the place of AJAX and JavaScript, what does it mean? You can give the following responses to all of these questions.



- It is very fast and extensible.
- It facilitates the users to write UI related function codes in minimum possible lines.
- It improves the performance of an application.
- Browser's compatible web applications can be developed.
- It uses mostly new features of new browsers.

Some of the companies on the web use jQuery are :

- Microsoft

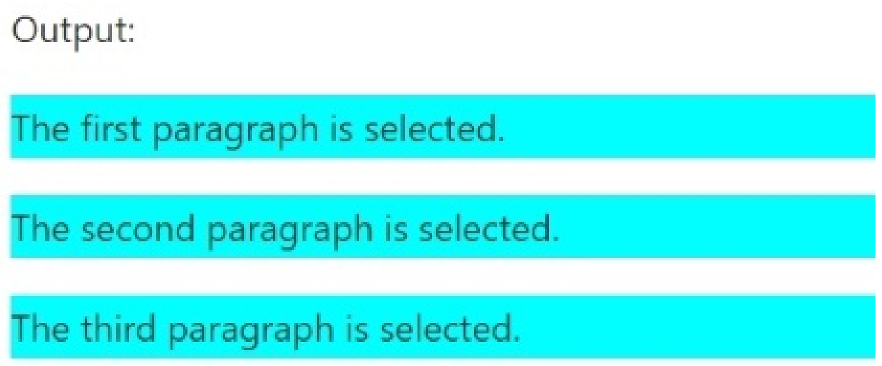
- Google
- IBM
- Netflix

jQuery Example

Google is the creator of jQuery. To make the first jQuery example, you'll need to use the jQuery JavaScript file. You can either use the absolute URL of the jQuery file or download it from jquery.com.

The absolute URL of the jQuery file is used in this jQuery example. The jQuery code is contained within the script tag.

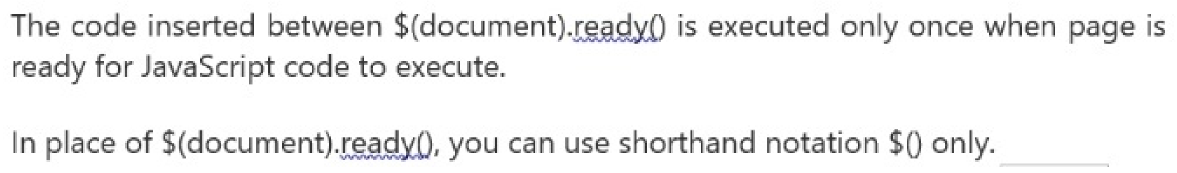
```
<!DOCTYPE html>
<html>
<head>
 <title>First jQuery Example</title>
 <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/
jquery/2.1.3/jquery.min.js">
 </script>
 <script type="text/javascript" language="javascript">
 $(document).ready(function() {
 $("p").css("background-color", "cyan");
          });
          </script>
          </head>
         <body>
         <p>The first paragraph is selected.</p>
         <p>The second paragraph is selected.</p>
         <p>The third paragraph is selected.</p>
         </body>
         </html>
```



Output:

```
The first paragraph is selected.

The second paragraph is selected.

The third paragraph is selected.
```

$(document).ready() and $()



The code inserted between $(document).ready() is executed only once when page is ready for JavaScript code to execute.

In place of $(document).ready(), you can use shorthand notation $() only.

```
$(document).ready(function() {
$("p").css("color", "red");
});
```

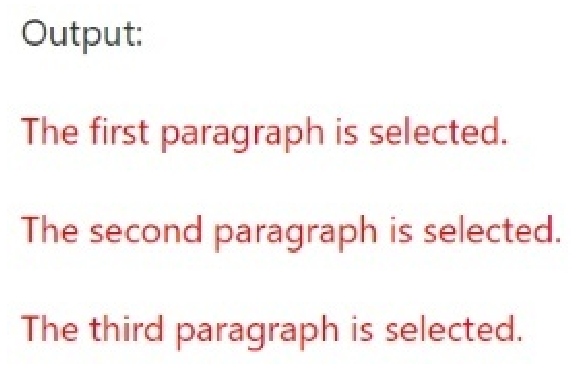
This code is equivalent to the code above.

```
$(function() {
$("p").css("color", "red");
});
```

Full example of jQuery - using shorthand notation $().

```
<!DOCTYPE html>
<html>
<head>
 <title>Second jQuery Example</title>
 <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/
jquery/2.1.3/jquery.min.js">
 </script>
 <script type="text/javascript" language="javascript">
```

```
$(function() {
$("p").css("color", "red");
          });
          </script>
          </head>
          <body>
          <p>The first paragraph is selected.</p>
          <p>The second paragraph is selected.</p>
          <p>The third paragraph is selected.</p>
          </body>
          </html>
```



```
Output:

The first paragraph is selected.

The second paragraph is selected.

The third paragraph is selected.
```

jQuery Selectors

Selecting and manipulating HTML components is done with the help of jQuery Selectors. They are an essential component of the jQuery library.

You can use jQuery selectors to find or pick HTML components from a DOM based on their id, classes, attributes, types, and more.

In simple terms, selectors are used in jQuery to choose one or more HTML elements, and once the element has been selected, you can execute various operations on it.

All jQuery selectors start with a dollor sign and parenthesis e.g. $(). It is known as the factory function.

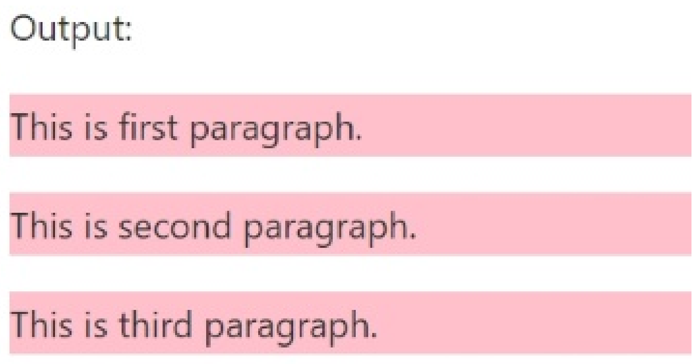
The $() factory function

Every jQuery selector start with this sign $(). This sign is known as the factory function. It uses the three basic building blocks while selecting an element in a given document.

S.No.	Selector	Description
1)	Tag Name:	It represents a tag name available in the DOM. For example: $('p') selects all paragraphs'p'in the document.
2)	Tag ID:	It represents a tag available with a specific ID in the DOM. For example: $('#real-id') selects a specific element in the document that has an ID of real-id.
3)	Tag Class:	It represents a tag available with a specific class in the DOM. For example: $('real-class') selects all elements in the document that have a class of real-class.

Simple example use of Tag selector

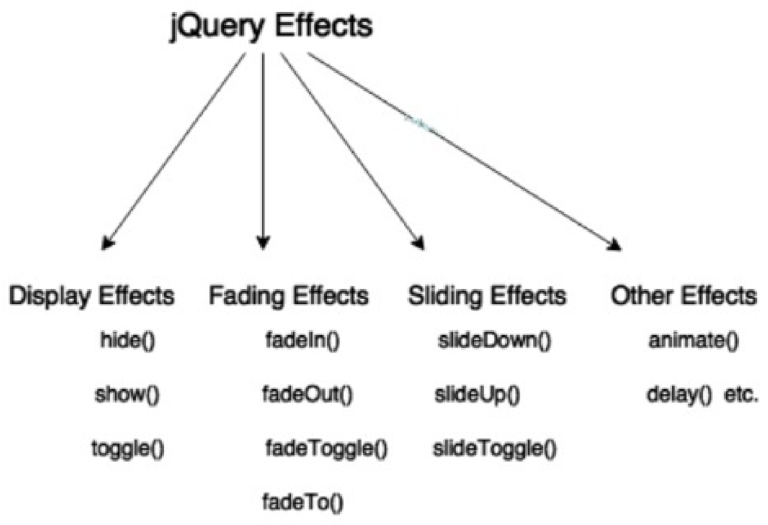
```
<!DOCTYPE html>
<html>
<head>
 <title>First jQuery Example</title>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
 </script>
 <script type="text/javascript" language="javascript">
 $(document).ready(function() {
 $("p").css("background-color", "pink");
 });
 </script>
 </head>
<body>
<p>This is first paragraph.</p>
<p>This is second paragraph.</p>
<p>This is third paragraph.</p>
</body>
</html>
```



```
Output:

This is first paragraph.

This is second paragraph.

This is third paragraph.
```

jQuery Effects

We can use jQuery to add effects to a web page. Fading, sliding, hiding/showing, and animation effects are all types of jQuery effects.



jQuery has a lot of methods for creating effects on a web page. The following is a complete list of jQuery effect methods:

No.	Method	Description
1)	animate()	performs animation.
2	clearQueue()	It is used to remove all remaining queued functions from the selected elements.
3)	delay()	sets delay execution for all the queued functions on the selected elements.
4	dequeue()	It is used to remove the next function from the queue, and then execute the function.
5)	fadein()	shows the matched elements by fading it to opaque. In other words, it fades in the selected elements.
6)	fadeout()	shows the matched elements by fading it to transparent. In other words, it fades out the selected elements.
7)	fadeto()	adjusts opacity for the matched element. In other words, it fades in/out the selected elements.
8)	fadetoggle()	shows or hides the matched element. In other words, toggles between the fadeIn() and fadeOut() methods.
9)	finish()	It stops, removes and complete all queued animation for the selected elements.
10)	hide()	hides the matched or selected elements.
11)	queue()	shows or manipulates the queue of methods i.e. to be executed on the selected elements.
12)	show()	displays or shows the selected elements.
13)	slidedown()	shows the matched elements with slide.

14)	slidetoggle()	shows or hides the matched elements with slide. In other words, it is used to toggle between the slideUp() and slideDown() methods.
15)	slideup()	hides the matched elements with slide.
16)	stop()	stops the animation which is running on the matched elements.
17)	toggle()	shows or hides the matched elements. In other words, it toggles between the hide() and show() methods.

jQuery hide()

The selected components are hidden using the jQuery hide() technique.

Syntax:

```
$(selector).hide();
$(selector).hide(speed, callback);
$(selector).hide(speed, easing, callback);
```

speed: It's a non-mandatory parameter. It specifies the delay's pace. Slow, fast, and milliseconds are all conceivable values.

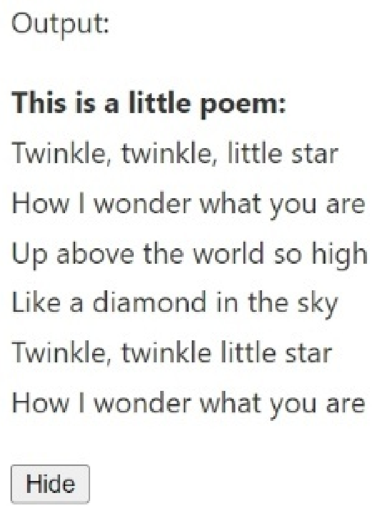
easing: It specifies the transition easing function to be utilised.

callback: It's also a non-mandatory parameter. It defines the function that will be invoked after the hide() action has been completed.

EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#hide").click(function(){
        $("p").hide();
    });
        });
        </script>
        </head>
        <body>
        <p>
        <b>This is a little poem: </b><br/>
```

```
Twinkle, twinkle, little star<br/>
How I wonder what you are<br/>
Up above the world so high<br/>
Like a diamond in the sky<br/>
Twinkle, twinkle little star<br/>
How I wonder what you are
</p>
<button id="hide">Hide</button>
</body>
</html>
```



Output:

This is a little poem:

Twinkle, twinkle, little star

How I wonder what you are

Up above the world so high

Like a diamond in the sky

Twinkle, twinkle little star

How I wonder what you are

jQuery show()

The selected components are displayed using the jQuery show() method.

Syntax:

1. $(selector).show();
2. $(selector).show(speed, callback);
3. $(selector).show(speed, easing, callback);

speed: It's a non-mandatory parameter. It specifies the delay's pace. Slow, fast, and milliseconds are all conceivable values.

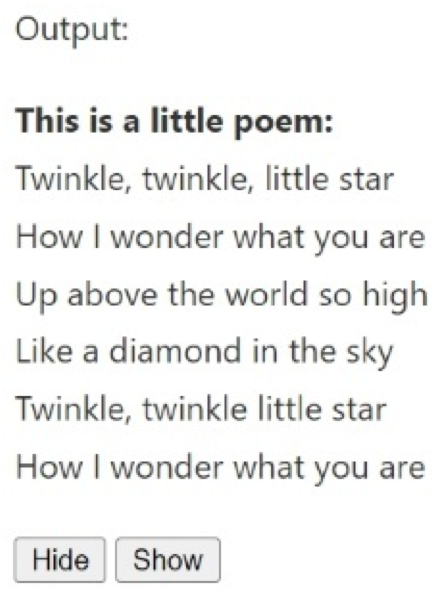
easing: It specifies the transition easing function to be utilised.

callback: It's also a non-mandatory parameter. It defines the function that will be invoked after the show() effect has finished.

EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
      $("#hide").click(function(){
      $("p").hide();
   });
            $("#show").click(function(){
               $("p").show();
            });
         });
         </script>
         </head>
```

```
<body>
<p>
<b>This is a little poem: </b><br/>
Twinkle, twinkle, little star<br/>
How I wonder what you are<br/>
Up above the world so high<br/>
Like a diamond in the sky<br/>
Twinkle, twinkle little star<br/>
How I wonder what you are
</p>
<button id="hide">Hide</button>
<button id="show">Show</button>
</body>
</html>
```



Output:

This is a little poem:

Twinkle, twinkle, little star

How I wonder what you are

Up above the world so high

Like a diamond in the sky

Twinkle, twinkle little star

How I wonder what you are

jQuery show() effect with speed parameter

EXAMPLE

```
$(document).ready(function(){
    $("#hide").click(function(){
    $("p").hide(1000);
  });
```

```
    $("#show").click(function(){
        $("p").show(1500);
    });
});
```

jQuery toggle()

The toggle() method in jQuery is a particular sort of method that toggles between the hide() and show() methods. It both reveals and conceals the hidden elements.

Syntax:

```
$(selector).toggle();
$(selector).toggle(speed, callback);
$(selector).toggle(speed, easing, callback);
$(selector).toggle(display);
```

speed: It's a non-mandatory parameter. It specifies the delay's pace. Slow, fast, and milliseconds are all conceivable values.

easing: It specifies the transition easing function to be utilised.

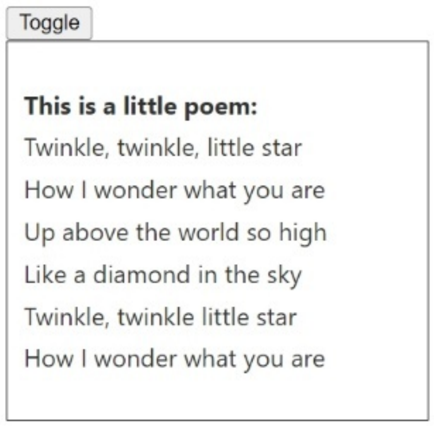
callback: It's also a non-mandatory parameter. It provides the function that will be invoked after the toggle() effect has finished.

display: If true, the element is displayed. If false, the element is hidden.

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("div.d1").toggle();
    });
});
</script>
</head>
<body>
<button>Toggle</button>
<div class="d1" style="border:1px solid black;padding:10px;width:250px">
<p><b>This is a little poem: </b><br/>
Twinkle, twinkle, little star<br/>
How I wonder what you are<br/>
Up above the world so high<br/>
```

```
Like a diamond in the sky<br/>
Twinkle, twinkle little star<br/>
How I wonder what you are</p>
</div>
</body>
</html>
```

Output:



This is a little poem:
Twinkle, twinkle, little star
How I wonder what you are
Up above the world so high
Like a diamond in the sky
Twinkle, twinkle little star
How I wonder what you are

jQuery fadeIn()

The element is faded in using the jQuery fadeIn() technique.

Syntax:

```
$(selector).fadein();
$(selector).fadeIn(speed,callback);
$(selector).fadeIn(speed, easing, callback);
```

speed: It's a non-mandatory parameter. It specifies the delay's pace. Slow, fast, and milliseconds are all conceivable values.

easing: It specifies the transition easing function to be utilised.

callback: It's also a non-mandatory parameter. It provides the function that will be invoked when the fadein() effect has finished.

EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").fadeIn();
        $("#div2").fadeIn("slow");
                $("#div3").fadeIn(3000);
            });
        });
        </script>
        </head>
        <body>
    <p>See the fadeIn() method example with different parameters.</p>
  <button>Click to fade in boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>
<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>
  </body>
  </html>
```

jQuery fadeOut()

The element is faded away using the jQuery fadeOut() technique.

Syntax:

```
$(selector).fadeOut();
$(selector).fadeOut(speed,callback);
$(selector).fadeOut(speed, easing, callback);
```

speed: It's a non-mandatory parameter. It specifies the delay's pace. Slow, fast, and milliseconds are all conceivable values.

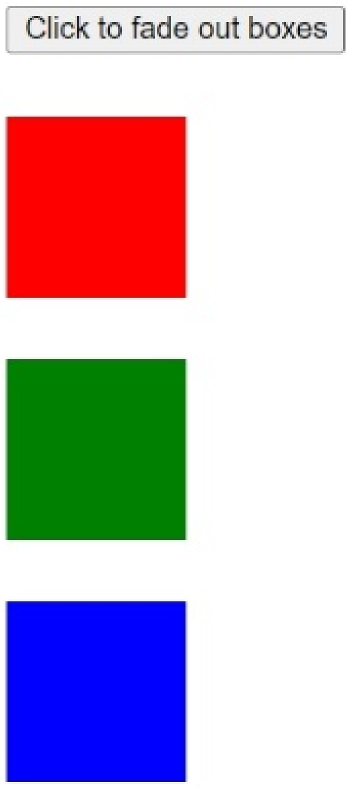
easing: It specifies the transition easing function to be utilised.

callback: It's also a non-mandatory parameter. It provides the method that will be invoked once the fadeOut() effect has finished.

EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").fadeOut();
        $("#div2").fadeOut("slow");
                    $("#div3").fadeOut(3000);
                });
            });
            </script>
            </head>
            <body>
        <p>See the fadeOut() method example with different parameters.</p>
        <button>Click to fade out boxes</button><br><br>
            <div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>
            <div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>
            <div id="div3" style="width:80px;height:80px;background-color:blue;"></div>
        </body>
        </html>
```

OUTPUT



jQuery fadeToggle()

Toggle between the fadeIn() and fadeOut() methods with the jQuery fadeToggle() method. If the elements are faded in, they will be faded out, and if the elements are faded out, they will be faded in.

Syntax:

```
$(selector).fadeToggle();
$(selector).fadeToggle(speed,callback);
$(selector).fadeToggle(speed, easing, callback);
```

speed: It's a non-mandatory parameter. It specifies the delay's pace. Slow, fast, and milliseconds are all conceivable values.

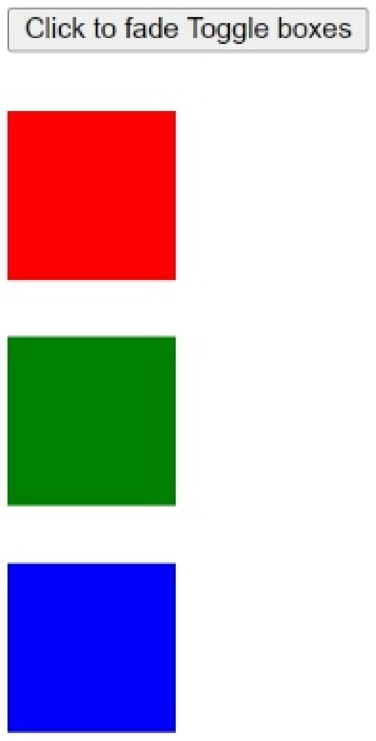
easing: It specifies the transition easing function to be utilised.

callback: It's also a non-mandatory parameter. It provides the method that will be invoked when the fadeToggle() effect has finished.

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").fadeToggle();
        $("#div2").fadeToggle("slow");
        $("#div3").fadeToggle(3000);
    });
});
</script>
</head>
<body>
<p>See the fadeToggle() method example with different parameters.</p>
<button>Click to fade Toggle boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>
```

```
<div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>
</body>
</html>
```

Output:



jQuery fadeTo()

The fadeTo() method in jQuery is used to fade to a specified opacity.

Syntax:

```
$(selector).fadeTo(speed, opacity);
$(selector).fadeTo(speed, opacity, callback);
$(selector).fadeTo(speed, opacity, easing, callback);
```

speed: It specifies the delay's pace. Slow, fast, and milliseconds are all conceivable values.

opacity: It determines the degree of opacity. The opacity value might be anywhere from 0 to 1.

easing: It specifies the transition easing function to be utilised.

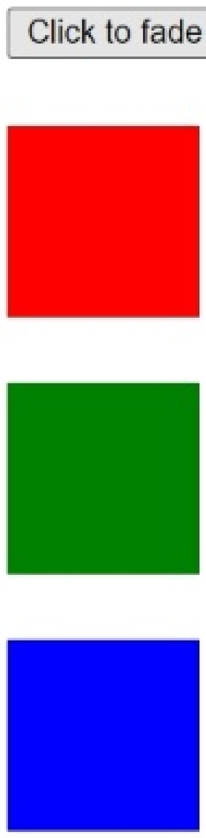
callback: It's also a non-mandatory parameter. It provides the method that will be invoked when the fadeToggle() effect has finished.

EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").fadeTo("slow", 0.3);
        $("#div2").fadeTo("slow", 0.4);
        $("#div3").fadeTo("slow", 0.5);
    });
});
</script>
</head>
<body>
```

```
<p>See the fadeTo() method example with different parameters.</p>
<button>Click to fade boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>
<div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>
</body>
</html>
```

Output:



jQuery slideDown()

To slide down an element, use the jQuery slideDown() method.

Syntax:

```
$(selector).slideDown(speed);
$(selector).slideDown(speed, callback);
$(selector).slideDown(speed, easing, callback);
```

speed: It specifies the delay's pace. Slow, fast, and milliseconds are all conceivable values.

easing: It specifies the transition easing function to be utilised.

callback: It's also a non-mandatory parameter. It defines the method that will be invoked after the slideDown() effect has finished.

EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#flip").click(function(){
        $("#panel").slideDown("slow");
    });
        });
        </script>
         <style>
        #panel, #flip {
            padding: 5px;
            text-align: center;
            background-color: #00FFFF;
            border: solid 1px #c3c3c3;
        }
```

```
#panel {
   padding: 50px;
   display: none;
}
</style>
</head>
<body>
<div id="flip">Click to slide down panel</div>
<div id="panel">Hello jquerypoint.com!
 It is the best tutorial website to learn jQuery and other languages.
</div>
</body>
</html>
```

jQuery slideUp()

To slide up an element, use the jQuery slideDown() method.

Syntax:

```
$(selector).slideUp(speed);
$(selector).slideUp(speed, callback);
$(selector).slideUp(speed, easing, callback);
```

speed: It specifies the delay's pace. Slow, fast, and milliseconds are all conceivable values.

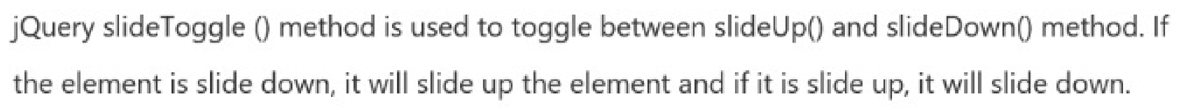
easing: It specifies the transition easing function to be utilised.

callback: It's also a non-mandatory parameter. It defines the function that will be invoked after the slideUp() effect has finished.

EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#flip").click(function(){
        $("#panel").slideUp("slow");
    });
        });
        </script>
         <style>
        #panel, #flip {
            padding: 5px;
        text-align: center;
        background-color: #00FFFF;
        border: solid 1px #c3c3c3;
    }
    #panel {
        padding: 50px;
    }
    </style>
    </head>
    <body>
    <div id="flip">Click to slide up panel</div>
    <div id="panel">Hello jquerypoint.com!
    It is the best tutorial website to learn jQuery and other languages.
</div>
    </body>
    </html>
```

jQuery slideToggle()



jQuery slideToggle () method is used to toggle between slideUp() and slideDown() method. If the element is slide down, it will slide up the element and if it is slide up, it will slide down.

Syntax:

1. $(selector).slideToggle(speed);
2. $(selector).slideToggle(speed, callback);
3. $(selector).slideToggle(speed, easing, callback);

speed: It specifies the delay's pace. Slow, fast, and milliseconds are all conceivable values.

easing: It specifies the transition easing function to be utilised.

callback: It's also a non-mandatory parameter. It provides the method that will be invoked after the slideToggle() action has finished.

EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#flip").click(function(){
        $("#panel").slideToggle("slow");
    });
});
</script>
 <style>
#panel, #flip {
    padding: 5px;
    text-align: center;
    background-color: #00FFFF;
    border: solid 1px #c3c3c3;
}
#panel {
    padding: 50px;
    display:none;
}
</style>
</head>
<body>
<div id="flip">Click to slide toggle panel</div>
<div id="panel">Hello jquerypoint.com!
It is the best tutorial website to learn jQuery and other languages.</div>
</body>
</html>
```

jQuery animate()

The animate() method in jQuery allows you to make custom animations.

Syntax:

```
$(selector).animate({params}, speed, callback);
```

The **params** parameter specifies the CSS properties that will be animated in this case.

The duration of the effect is determined by the **speed** parameter, which is optional. It has three options: "slow," "rapid," and milliseconds.

The **callback** parameter is also optional, and it specifies a function that will be run when the animation has finished.

EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("div").animate({left: '450px'});
    });
});
</script>
</head>
<body>
<button>Start Animation</button>
<p>A simple animation example:</p>
<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>
</body>
</html>
```

jQuery animate() method - using multiple properties

You can animate with several properties at the same time.

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("div").animate({
            left: '250px',
                        opacity: '0.5',
                        height: '150px',
                        width: '150px'
                    });
                });
            });
            </script>
            </head>
            <body>
            <button>Start Animation</button>
<div style="background:#125f21;height:100px;width:100px;position:absolute;"></div>
            </body>
            </html>
```

jQuery animate() method - using relative values

By placing += or -= in front of the value, you can define relative values (values that are relative to the element's current value).

```
<!DOCTYPE html>
<html>
<head>
<script  src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("div").animate({
            left: '250px',
                        height: '+=150px',
                        width: '+=150px'
                    });
                });
            });
            </script>
            </head>
            <body>
            <button>Start Animation</button>
<div  style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>
            </body>
            </html>
```

jQuery animate() method - using predefined value

You can also specify "display," "hide," or "toggle" as the animation value for a property.

We're using the "toggle" option for height in this example, which means it will show or conceal the selected element.

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("div").animate({
            height: 'toggle'
                });
            });
        });
        </script>
        </head>
        <body>
        <button>Start Animation</button>
<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>
        </body>
        </html>
```

jQuery Color animation

The characteristics of elements can also be animated between hues.

```
<!doctype html>
<html lang="en">
<head>
```

```
<meta charset="utf-8">
<title>jQuery UI Effects - Animate demo</title>
 <link rel="stylesheet" href="http://code.jquery.com/ui/1.11.4/themes/s
moothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
<style>
            .toggler { width: 500px; height: 200px; position: relative; }
            #button { padding: .5em 1em; text-decoration: none; }

  #effect { width: 240px; height: 135px; padding: 0.4em; position: relati
ve; background: #fff; }
            #effect h3 { margin: 0; padding: 0.4em; text-align: center; }
           </style>
           <script>
           $(function() {
             var state = true;
             $( "#button" ).click(function() {
               if ( state ) {
                 $( "#effect" ).animate({
                   backgroundColor: "#aa0000",
                   color: "#fff",
                   width: 500
                 }, 1000 );
               } else {
                 $( "#effect" ).animate({
                   backgroundColor: "#fff",
                   color: "#000",
                   width: 240
                 }, 1000 );
               }
               state = !state;
             });
           });
           </script>
          </head>
          <body>
```

```
<div class="toggler">
  <div id="effect" class="ui-widget-content ui-corner-all">
    <h3 class="ui-widget-header ui-corner-all">Animate</h3>
    <p>Javatpoint.com is the best tutorial website to learn Java and other programming languages.</p>
  </div>
</div>
<button id="button" class="ui-state-default ui-corner-all">Toggle Effect</button>
</body>
</html>
```

jQuery html()

To alter the complete content of the selected components, use the jQuery html() method. It replaces the content of the selected element with fresh material.

Note that, due to its API documentation, it is a very helpful function that only operates in a limited area.
Three method signatures make up the API documentation for the jQuery html function.

Because the first method signature does not take an argument, it simply returns the HTML contained within that element.
The remaining two signatures take only one argument: a string or a string-returning function.

Syntax:

$(selector).html()

$(selector).html(content)

$(selector).html(function (index, currentcontent))

By calling a function, it is utilised to set content.

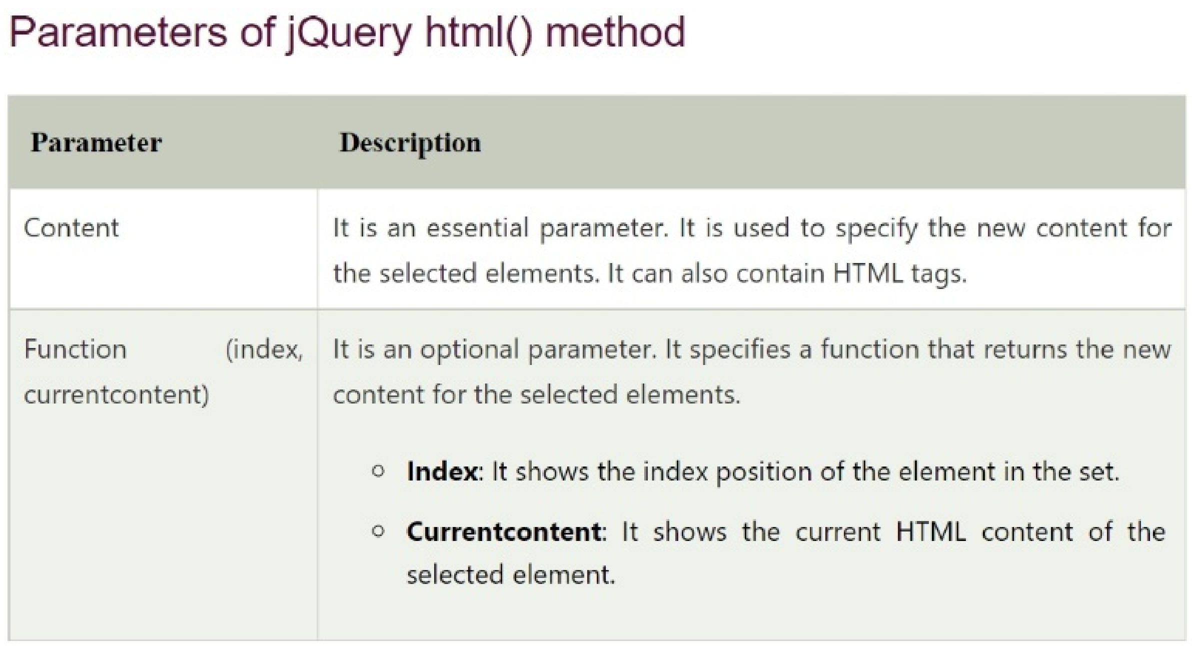
The html() method of jQuery is used to either set or return the content of selected elements.

To set content: This technique overwrites the content of all matched elements when used to set content.

To return content: This function returns the content of the first matched element when used to return content.

Only the text content of the selected components is set or returned using the text() function.

Because the first method signature does not take an argument, it simply returns the HTML contained within that element. The remaining two signatures take only one argument: a string or a string-returning function.



Parameters of jQuery html() method

Parameter	Description
Content	It is an essential parameter. It is used to specify the new content for the selected elements. It can also contain HTML tags.
Function (index, currentcontent)	It is an optional parameter. It specifies a function that returns the new content for the selected elements. ◦ **Index**: It shows the index position of the element in the set. ◦ **Currentcontent**: It shows the current HTML content of the selected element.

Example - jQuery html() method

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script>
```

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").html("Hello <b>Javatpoint.com</b>");
    });
});
</script>
</head>
<body>
<button>Click here to change the content of all p elements</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

jQuery html() - example 2

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        alert($("p").html());
    });
});
</script>
</head>
<body>

<button>Return the content of p element</button>

<p>This is first <b>paragraph</b>.</p>
<p>This is another <b>paragraph</b>.</p>
</body>
</html>
```

jQuery html() - example 3

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>html demo</title>
  <style>
  p {
    margin: 8px;
    font-size: 20px;
    color: blue;
    cursor: pointer;
  }
  b {
    text-decoration: underline;
  }
  </style>
  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>
<p>
  <b>Click</b> here to change the <span id="tag">html</span> to text
</p>
<script>
$( "p" ).click(function() {
  var htmlString = $( this ).html();
  $( this ).text( htmlString );
});
</script>
</body>
</html>
```

jQuery text()

The text() function of the jQuery library is used to set or return the text content of selected elements.

To return content: This function returns the combined text content of all matched elements without the HTML markup when used to return content.

To set content: This technique overwrites the content of all matched elements when used to set content.

Difference - jQuery text() method - jQuery html() method

This is caused by the fact that both methods are used to set or return html content. The jQuery text() method, on the other hand, is not the same as the html() method.

The primary distinctions are as follows:

The jQuery text() method sets or returns html content without HTML markup, whereas the html() method sets or returns innerHtml (text + HTML markup).

The jQuery text() function is compatible with both XML and HTML documents, however the jQuery html() method is not.

Syntax:

To return text content:

```
$(selector).text()
```

To set text content:

```
$(selector).text(content)
```

To set text content using a function:

```
$(selector).text(function(index,currentcontent))
```

Parameters of jQuery text() method

Parameter	Description
Content	It is a mandatory parameter. It specifies the new text content for the selected elements. The special characters will be encoded in this parameter.
Function (index,currentcontent)	It is an optional parameter. It specifies the function that returns the new text content for the selected elements. ◦ **Index:** It provides the index position of the element in the set. ◦ **Currentcontent:** It provides the current content of the selected elements.

Example - jQuery text() method

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>text demo</title>
  <style>
  p {
    color: blue;
    margin: 8px;
  }
  b {
    color: red;
  }
  </style>
  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>
<p><b>Hello! </b>jquerypoint.com</p>
<p></p>
<script>
var str = $( "p:first" ).text();
$( "p:last" ).html( str );
</script>
</body>
</html>
```

EXAMPLE 2

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").text("Welcome to jquerypoint.com!");
    });
});
</script>
</head>
<body>
```

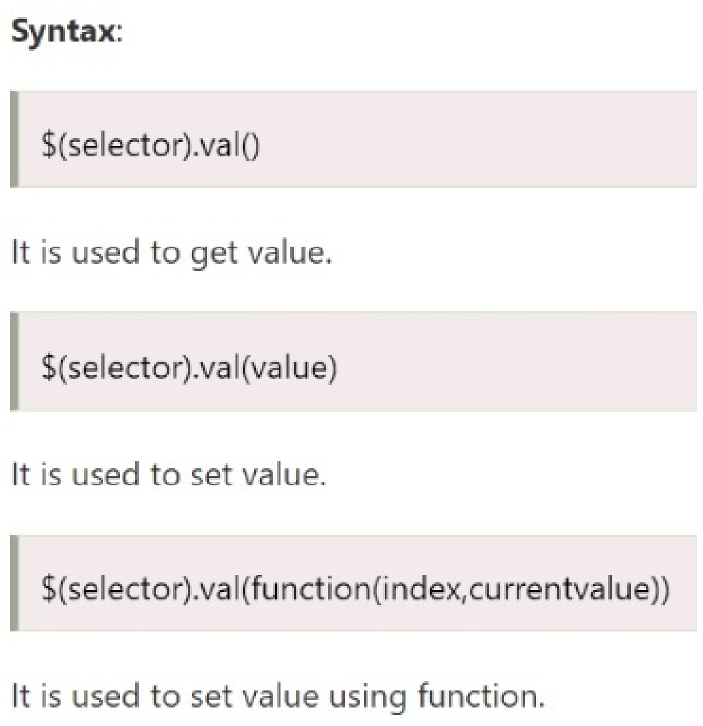
```
<button>Click here to set text content for all p elements</button>
<p>Hello Guys!</p>
<p>Looking for online training....</p>
</body>
</html>
```

jQuery val()

The val() method in jQuery can be used in two ways.

It's used to get the current value of the first matched element in the set.

It's used to provide each matching element a value.



Syntax:

```
$(selector).val()
```

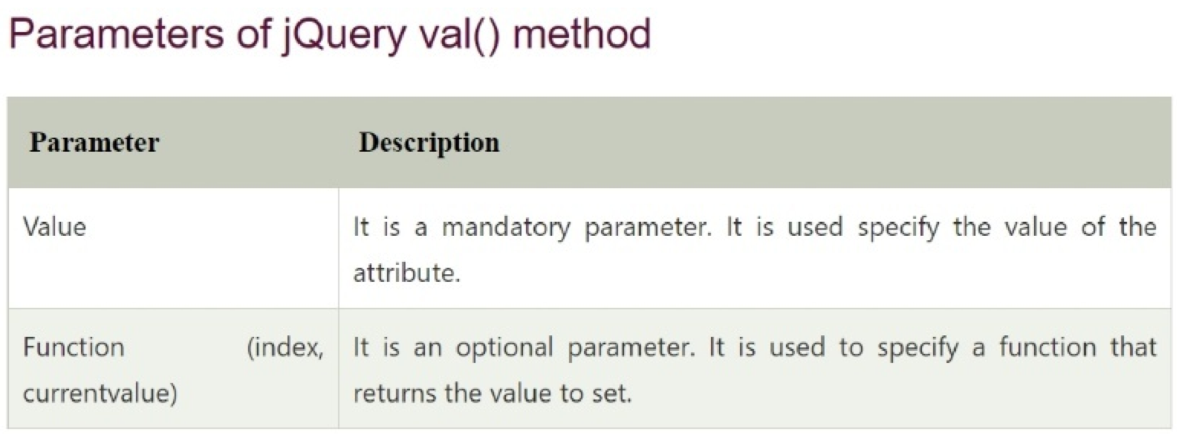
It is used to get value.

```
$(selector).val(value)
```

It is used to set value.

```
$(selector).val(function(index,currentvalue))
```

It is used to set value using function.



Parameters of jQuery val() method

Parameter	Description
Value	It is a mandatory parameter. It is used specify the value of the attribute.
Function (index, currentvalue)	It is an optional parameter. It is used to specify a function that returns the value to set.

jQuery val() example

The val() function is generally used to retrieve form element data. There are no arguments accepted by this procedure. When no choices are selected, this function returns NULL, and when one or more options are selected, it returns an array holding the values of each selected option.

EXAMPLE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>val demo</title>
  <style>
  p {
    color: red;
    margin: 4px;
          }
          b {
            color: blue;
          }
          </style>
```

```
                        <script src="https://code.jquery.com/jquery-1.10.2.js">
</script>
          </head>
          <body>
          <p></p>
          <select id="single">
            <option>Single</option>
            <option>Double</option>
           <option>Triple</option>
          </select>
          <script>
          function displayVals() {
            var singleValues = $( "#single" ).val();
            $( "p" ).html( "<b>Value:</b> " + singleValues);
          }
          $( "select" ).change( displayVals );
          displayVals();
          </script>
          </body>
          </html>
```

Let's look at how to use the val() method in jQuery with single and multiple select boxes.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>val demo</title>
  <style>
  p {
    color: red;
    margin: 4px;
          }
          b {
            color: blue;
          }
          </style>
                <script src="https://code.jquery.com/jquery-1.10.2.js">
</script>
</head>
<body>
<p></p>
<select id="single">
          <option>Single</option>
          <option>Single2</option>
          <option>Single3</option>
</select>
<select id="multiple" multiple="multiple">
          <option selected="selected">Multiple</option>
          <option>Multiple2</option>
          <option>Multiple3</option>
</select>
<script>
function displayVals() {
          var singleValues = $( "#single" ).val();
          var multipleValues = $( "#multiple" ).val() || [];
          $( "p" ).html( "<b>Single:</b> " + singleValues +
            " <b>Multiple:</b> " + multipleValues.join( ", " ) );
}
$( "select" ).change( displayVals );
displayVals();
</script>
</body>
</html>
```

JAVASCRIPT CODING:
BASICS FOR ABSOLUTE BEGINNERS

STEP BY STEP GUIDE TO LEARN CODING QUICKLY

J NEWMAN

Learn JavaScript

JavaScript is a lightweight, cross-platform object-based scripting language.

JavaScript is a translated language, not a compiled language. The JavaScript Translator (which is built within the browser) is in charge of translating JavaScript code for web browsers.

JavaScript (js) is a lightweight object-oriented programming language that is used to script webpages by a number of websites.

When applied to an HTML document, it is an interpreted, full-featured programming language that enables dynamic interactivity on websites. It was first released in 1995 to allow users to add programs to webpages in the Netscape Navigator browser.

Since then, all other graphical web browsers have embraced it. Users can use JavaScript to create modern web applications that interact immediately without having to reload the page every time. Js is used on a standard website to provide many sorts of interactivity and simplicity.

JavaScript, on the other hand, has no connection to the Java programming language.

Features of JavaScript

Because they have built-in execution environments, all popular web browsers support JavaScript.

The grammar and structure of JavaScript are based on the C programming language. As a result, it is classified as a structured programming language.

JavaScript is a weakly typed language with implicit casting of certain types (depending on the operation).

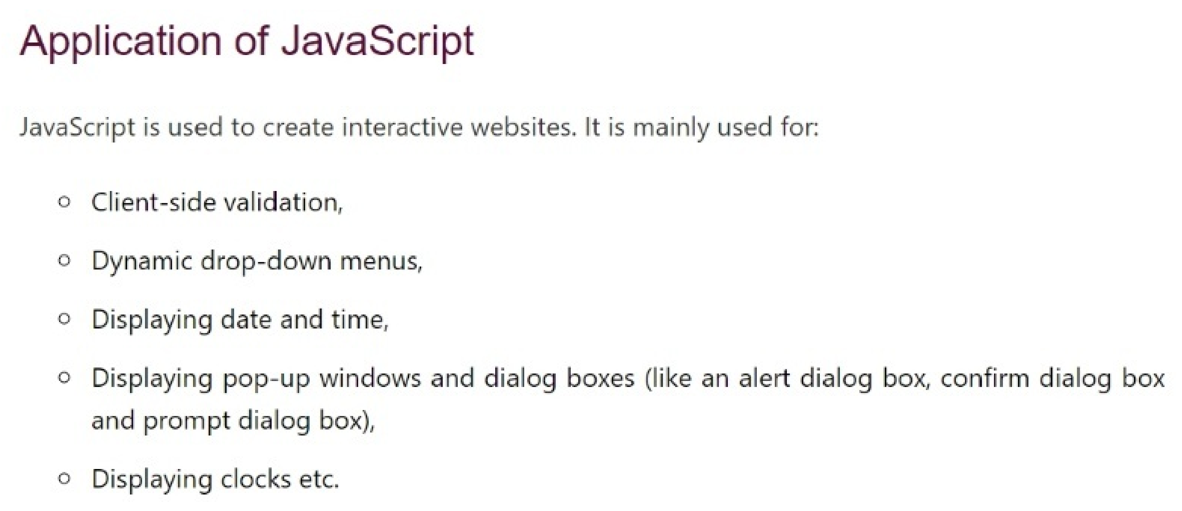
JavaScript is an object-oriented programming language that inherits through prototypes rather than classes.

It's a simple and interpretable language.

The language is case-sensitive.

JavaScript is compatible with a variety of operating systems, including Windows, macOS, and Linux.

It gives consumers complete control over their web browsers.



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

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

JavaScript Comment

JavaScript comments are an effective means of conveying information. It's used to include code-related information, such as warnings or suggestions, so that the end user can understand it.

The JavaScript engine, which is incorporated in the browser, ignores the JavaScript comment.

Advantages - JavaScript comments

To make coding more understandable It can be used to explain the code so that the end user can understand it.

To avoid having to write unneeded code It can also be used to prevent the code from running. We sometimes add code to do a certain action. However, the code may need to be disabled after some time. It is preferable to utilise comments in this situation.

JavaScript Comments - Types

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

Single line Comment

Double forward slashes (//) are used to indicate it. It's appropriate to use it both before and after the sentence.

Let's look at an example of a single-line comment, which is one that is placed before the statement.

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

```
</script>
```

EXAMPLE

```
<script>
var a=10;
var b=20;
var c=a+b;//It adds values of a and b variable
document.write(c);//prints sum of 10 and 20
</script>
```

Multi line Comment

It can be used to create both single and multi-line comments. As a result, it is more practical.

It's symbolised by a forward slash followed by an asterisk, then another forward slash. Consider the following scenario:

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

It might appear before, after, or in the middle of a 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 nothing more than a storage location's name. In JavaScript, there are two sorts of variables: local variables and global variables.

When declaring a JavaScript variable, there are some guidelines to follow (also known as identifiers).

A letter (a to z or A to Z), underscore(_), or dollar($) sign must begin the name.

We can put numerals (0 to 9) after the initial letter, for example value1.

Variables in JavaScript are case sensitive, so x and X are two separate variables.

Correct JavaScript variables

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

Incorrect JavaScript variables

```
var  123=30;
var *aa=320;
```

Example - JavaScript variable

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

```
</script>
```

Output

30

JavaScript local variable

A local variable in JavaScript is declared within a block or function. It can only be accessed from within the function or block. Consider the following scenario:

```
<script>
function abc(){
var x=10;//local variable
}
</script>
```

Or,

```
<script>
If(10<13){
var y=20;//JavaScript local variable
}
</script>
```

JavaScript global variable

A global variable in JavaScript can be accessed from any function. A global variable is one that is declared outside of the function or with the window object. Consider the following scenario:

```
<script>
var data=200;//gloabal variable
```

```
function a(){
document.writeln(data);
}
function b(){
document.writeln(data);
}
a();//calling JavaScript function
b();
</script>
```

JavaScript Global Variable

A global variable in JavaScript is declared outside of the function or with the window object. It's possible to go to it from any function.

Let's look at a simple JavaScript example of a global variable.

```
<script>
var value=50;//global variable
function a(){
alert(value);
}
function b(){
alert(value);
}
</script>
```

Declaring a global variable in JavaScript within a function

You must utilise the window object to declare JavaScript global variables inside a function. Consider the following scenario:

```
window.value=90;
```

It can now be declared within any function and accessed from within any function. Consider the following scenario:

```
function m(){
```

```
window.value=100;//declaring global variable by window object
}
function n(){
alert(window.value);//accessing global variable from other function
}
```

In JavaScript, the internals of a global variable

When you declare a variable outside of a function, it is automatically added to the window object. You can also use the window object to get to it. Consider the following scenario:

```
var value=50;
function a(){
alert(window.value);//accessing global variable
}
```

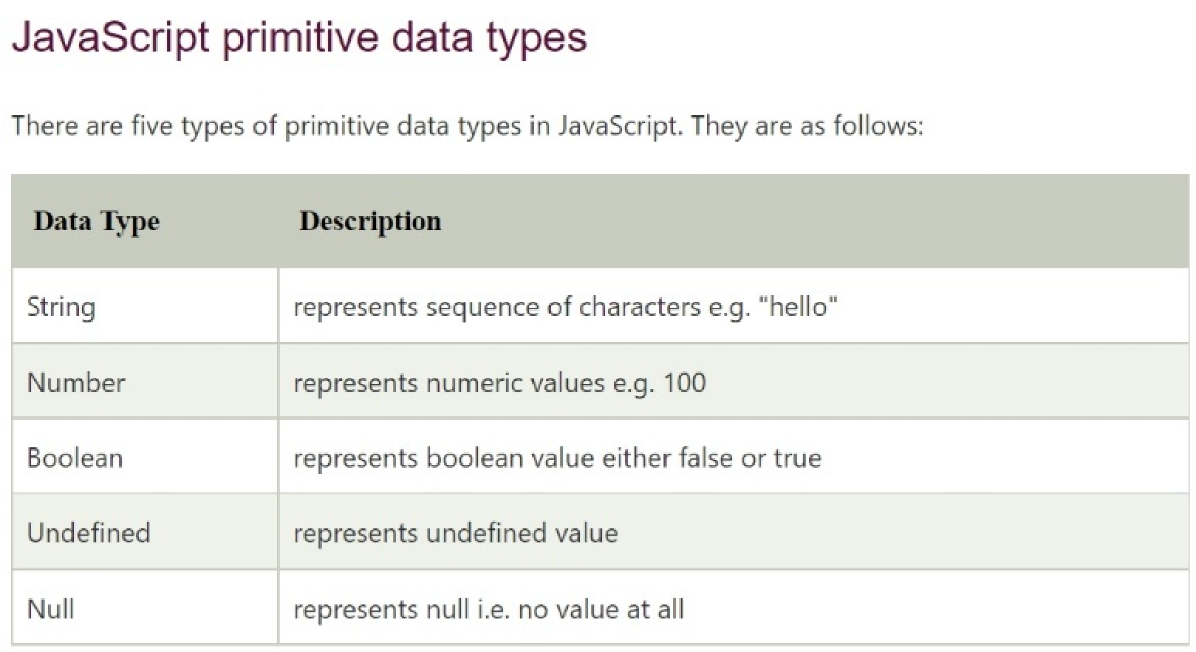
Javascript Data Types

Different data types are available in JavaScript to hold various types of values. In JavaScript, there are two sorts of data types.

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

JavaScript is a dynamic type language, which means you don't have to specify the type of a variable because the JavaScript engine uses it dynamically. To specify the data type, you must use var. It may store any type of value, including numbers, strings, and so on. Consider the following scenario:

```
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

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

Operators in JavaScript are symbols that perform operations on operands. Consider the following scenario:

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

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

To conduct arithmetic operations on the operands, arithmetic operators are employed. JavaScript arithmetic operators are the operators listed below.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

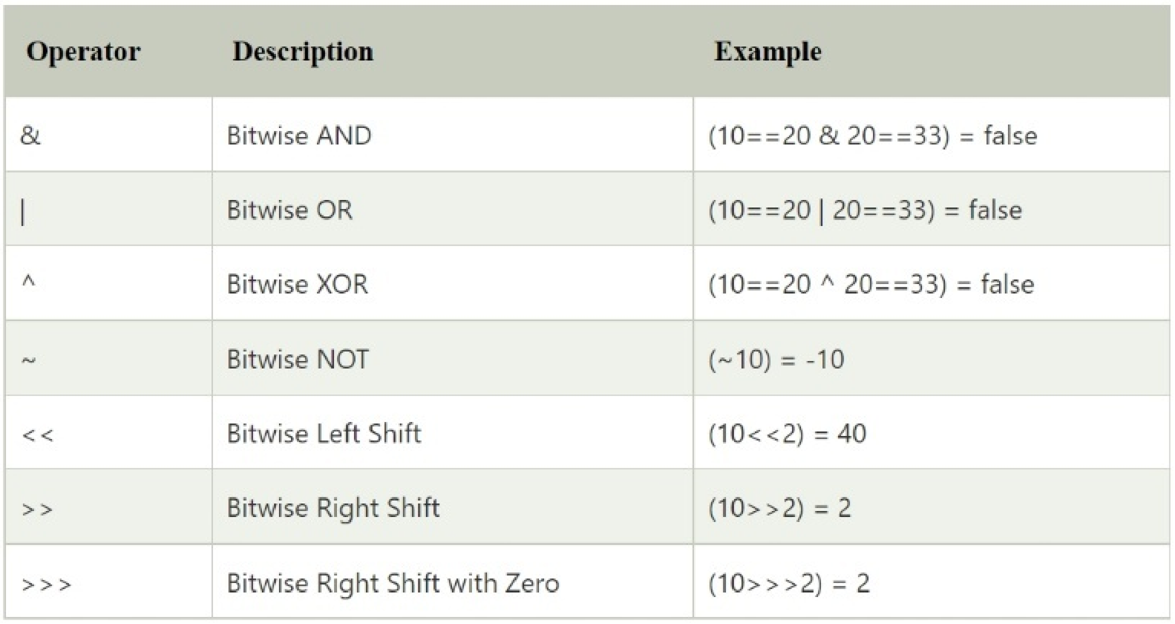
JavaScript Comparison Operators

The comparison operator in JavaScript compares the two operands. The following are the comparison operators:

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

JavaScript Bitwise Operators

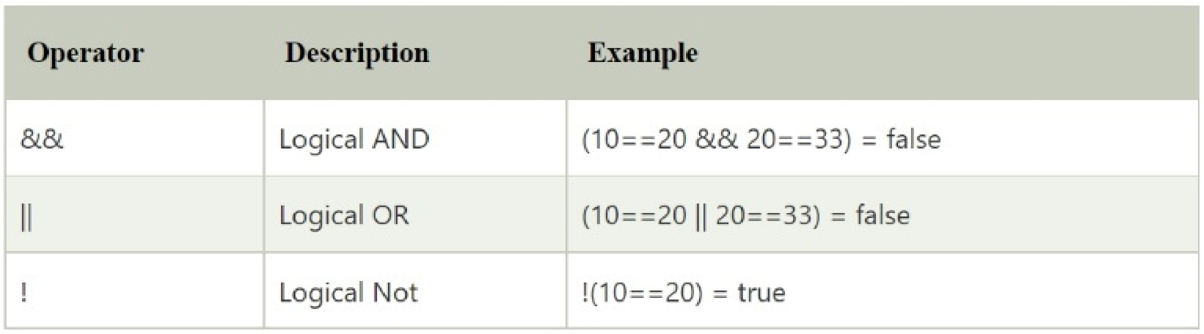
On operands, the bitwise operators perform bitwise operations. The following are the bitwise operators:



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

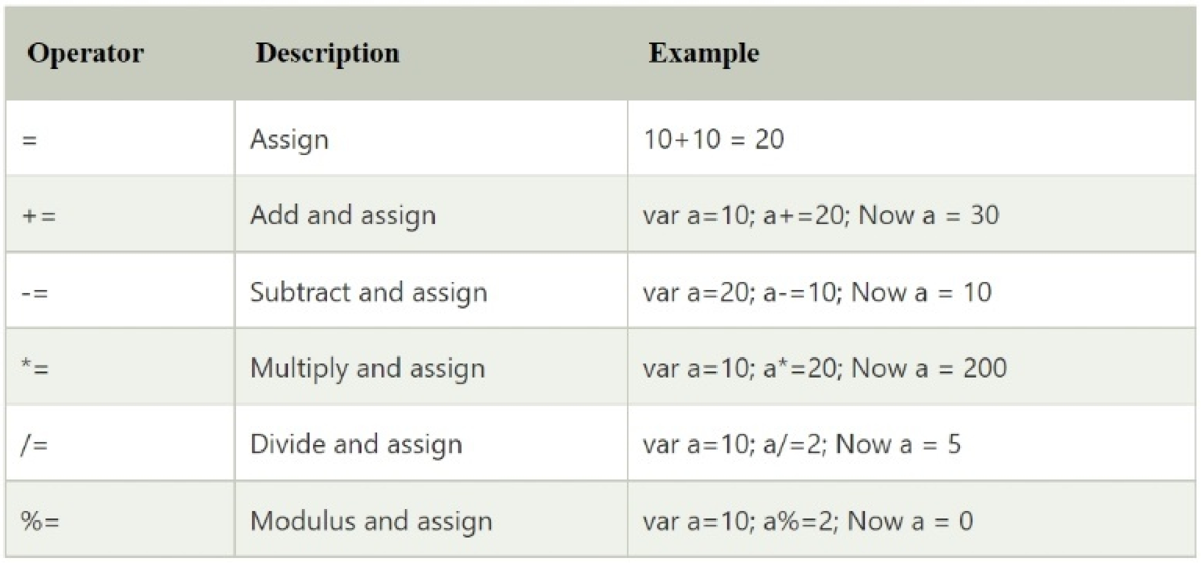
JavaScript Logical Operators

JavaScript logical operators are the operators listed below.



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



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

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
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 If-else

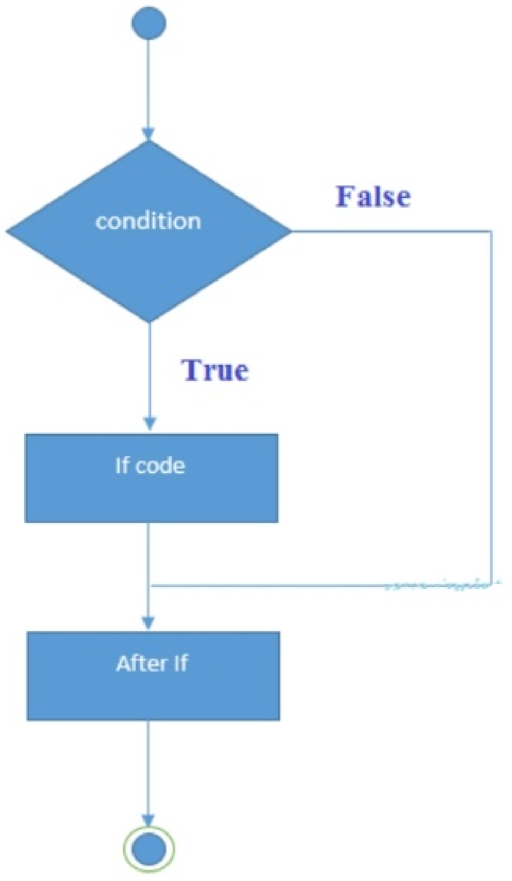
The **JavaScript if-else statement** is used *to execute the code whether condition is true or false.* There are three forms of if statement in JavaScript.

1. If Statement
2. If else statement
3. if else if statement

JavaScript If statement

Only if expression is true does it assess the content. The if statement in JavaScript has the following signature.

```
if(expression){
//content to be evaluated
}
```



EXAMPLE

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

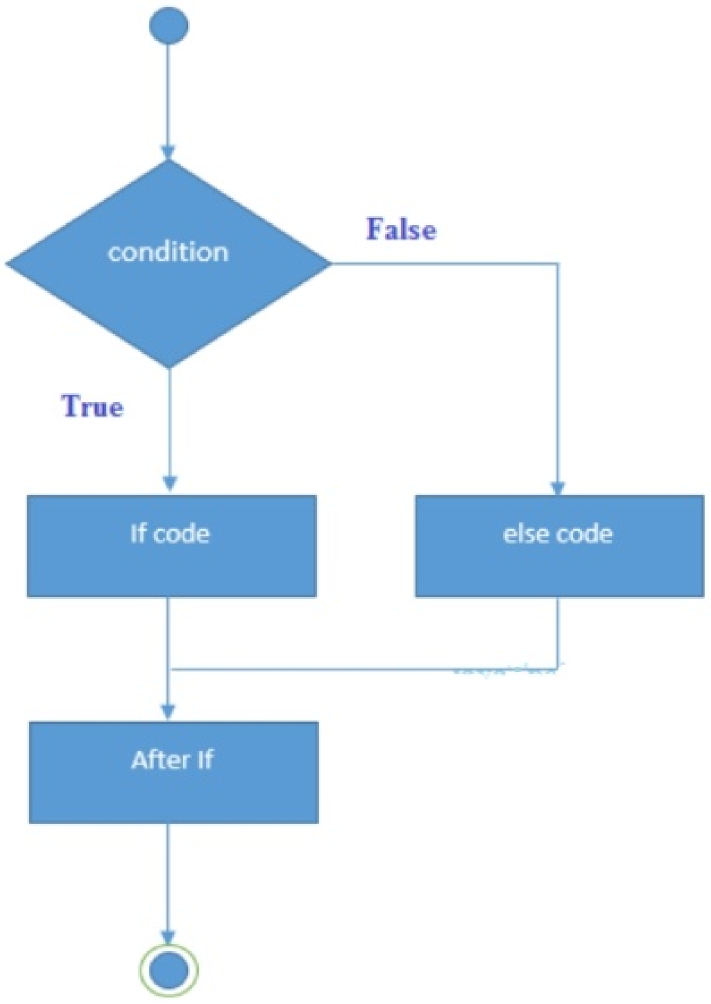
OUTPUT

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

JavaScript If...else Statement

It determines if the condition is true or false by evaluating the content. The syntax for the if-else statement in JavaScript is shown below.

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



EXAMPLE

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

OUTPUT

```
a is even number
```

If...else if statement – in Javascript

It only analyses the content if one or more expressions are true. The if else if statement in JavaScript has the following signature.

```
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
        }
```

EXAMPLE

```
<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>
```

OUTPUT

```
a is equal to 20
```

JavaScript Switch

The JavaScript switch statement is used to combine numerous expressions into a single code. It's the same as the else if statement we learnt on the previous page. It is, however, more useful than if..else..if because it may be used with integers, characters, and other variables.

The JavaScript switch statement's signature is shown 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;
}
```

Simple example - switch statement - 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

```
B Grade
```

Behaviour of switch statement

```
<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>
```

OUTPUT

```
undefined B Grade C Grade No Grade
```

JavaScript Loops

The for, while, do while, and for-in loops are used to iterate the piece of code in JavaScript. It reduces the size of the code. It's most commonly found in arrays.

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

1) JavaScript For loop

The loop in JavaScript iterates the components for a set number of times. If the number of iterations is known, it should be used. The for loop's syntax is seen below.

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

EXAMPLE

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

Output:

```
1
2
3
4
5
```

2) JavaScript while loop

The while loop in JavaScript iterates the components an infinite amount of times. If the number of iterations is unknown, it should be utilised. The while loop's syntax is seen below.

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

EXAMPLE

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

Output:

```
11
12
13
14
15
```

3) Do while loop

The JavaScript do while loop, like the while loop, iterates the elements indefinitely. Regardless of whether the condition is true or false, the code is performed at least once. The following is the syntax for the do while loop.

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

EXAMPLE

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

Output:

```
21
22
23
24
25
```

4) JavaScript for in loop

The JavaScript for in loop is used to cycle through an object's properties.

JavaScript Functions

To conduct operations, JavaScript functions are needed. To reuse the code, we can call the JavaScript function many times.

Benefits of Using JavaScript

JavaScript functions have primarily two advantages.

Reusability of code: Because we can call a function multiple times, we can save time coding.

Less coding means our program is more compact. To accomplish a common task, we don't need to write many lines of code each time.

JavaScript Function Syntax

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

EXAMPLE

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

JavaScript Function Arguments

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

Function with Return Value

We can use a function that returns a value in our program by calling it. Let's look at a function that returns a value as an example.

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

OUTPUT

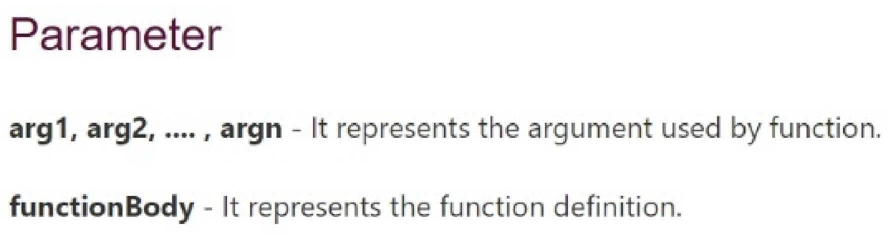
```
hello javatpoint! How r u?
```

JavaScript Function Object

The Function function Object() { [native code] } in JavaScript is used to generate a new Function object. It runs the code on a global scale. When we call the function Object() { [native code] } directly, however, a function is constructed dynamically but insecurely.

Syntax

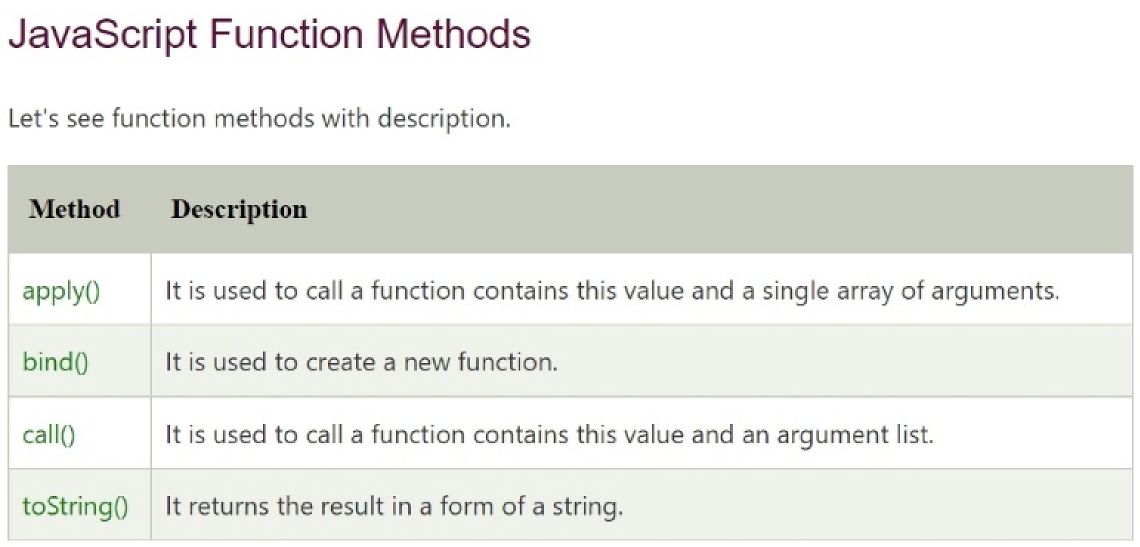
```
new Function ([arg1[, arg2[, ....argn]],] functionBody)
```



Parameter

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

functionBody - It represents the function definition.



JavaScript Function Methods

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 Function Object Example

```
<script>
var add=new Function("num1","num2","return num1+num2");
```

```
document.writeln(add(2,5));
</script>
```

Output:

```
7
```

Example 2

```
<script>
var pow=new Function("num1","num2","return Math.pow(num1,num2)");
document.writeln(pow(2,3));
</script>
```

Output:

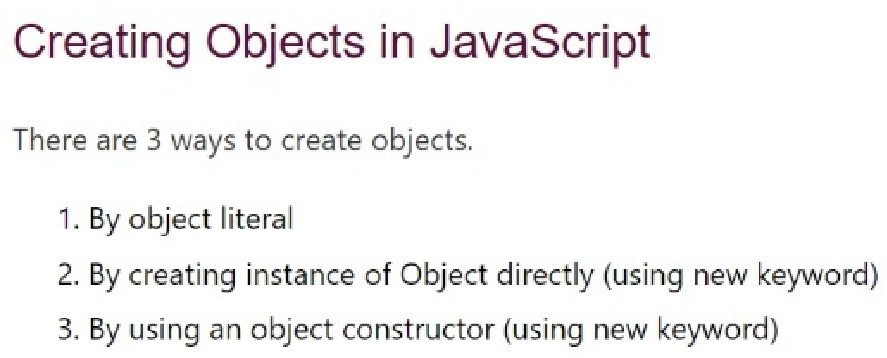
```
8
```

JavaScript Objects

A javaScript object is a state and behavior-based entity (properties and method). For example, a car, a pen, a bicycle, a chair, a glass, a keyboard, and a monitor.

JavaScript is an object-oriented programming language. In JavaScript, everything is an object.

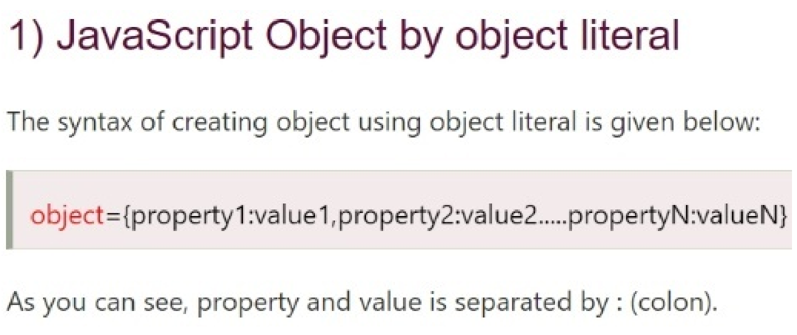
JavaScript is a template-based language, not a class-based one. To get the object, we don't need to build a class. However, we direct the creation of objects.



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}
```

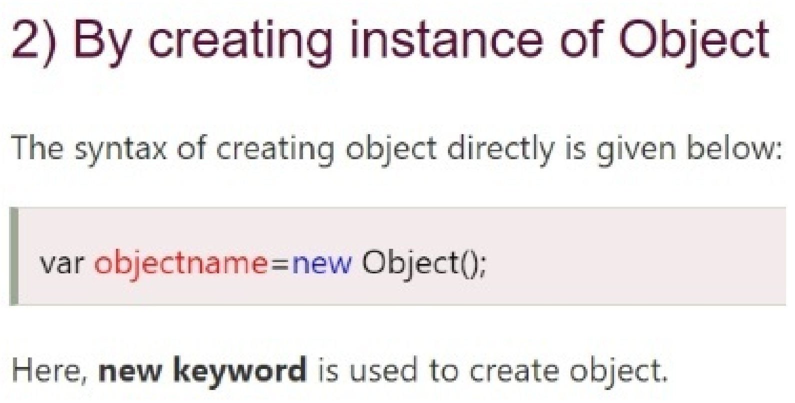
As you can see, property and value is separated by : (colon).

EXAMPLE

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

OUTPUT

```
102 Shyam Kumar 40000
```



2) By creating instance of Object

The syntax of creating object directly is given below:

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

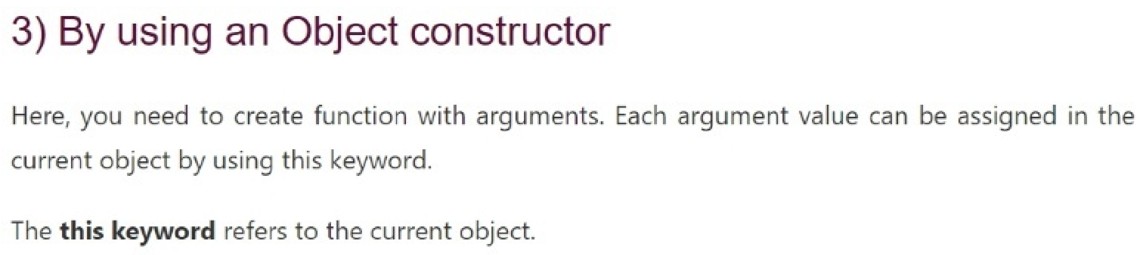
Here, **new keyword** is used to create object.

EXAMPLE

```
<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

```
101 Ravi 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.

EXAMPLE

```
<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>
```

OUTPUT

```
103 Vimal Jaiswal 30000
```

Defining method - JavaScript Object

In a JavaScript object, we can specify methods. However, before we define the method, we must add a property to the function with the same name as the method.

EXAMPLE

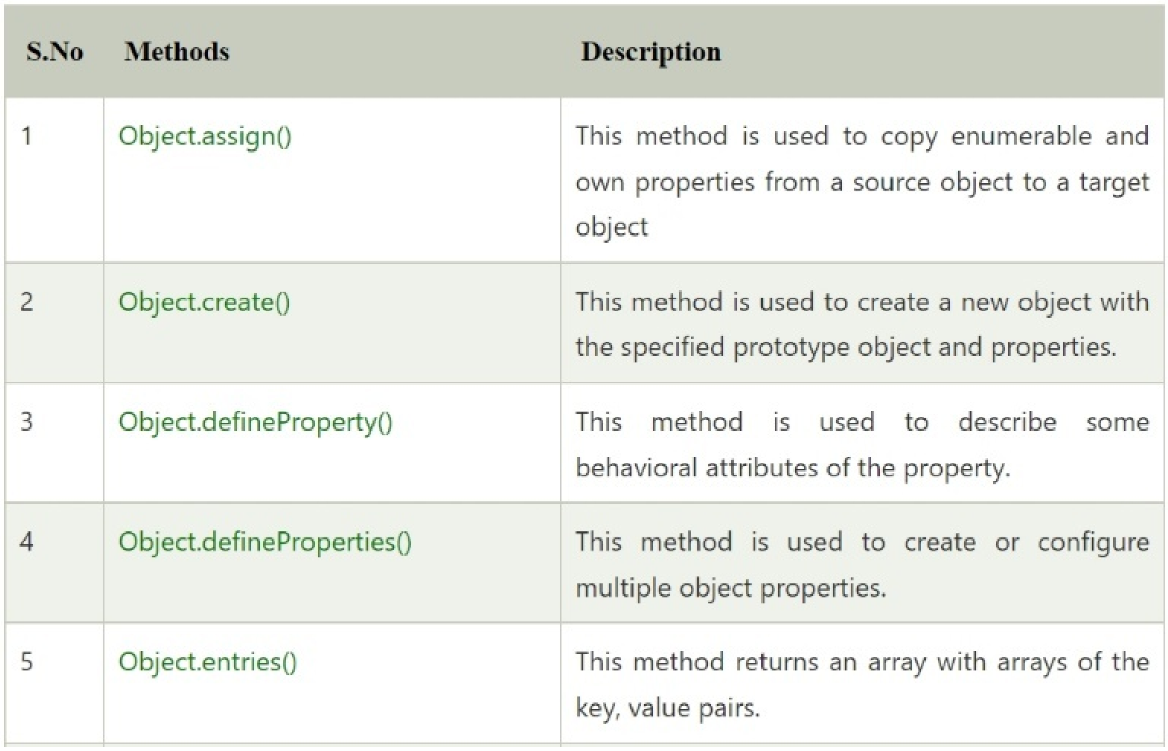
```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;

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>
```

OUTPUT

```
103 Sonoo Jaiswal 30000
103 Sonoo Jaiswal 45000
```

JavaScript Object Methods



S.No	Methods	Description
1	Object.assign()	This method is used to copy enumerable and own properties from a source object to a target object
2	Object.create()	This method is used to create a new object with the specified prototype object and properties.
3	Object.defineProperty()	This method is used to describe some behavioral attributes of the property.
4	Object.defineProperties()	This method is used to create or configure multiple object properties.
5	Object.entries()	This method returns an array with arrays of the key, value pairs.

6	Object.freeze()	This method prevents existing properties from being removed.
7	Object.getOwnPropertyDescriptor()	This method returns a property descriptor for the specified property of the specified object.
8	Object.getOwnPropertyDescriptors()	This method returns all own property descriptors of a given object.
9	Object.getOwnPropertyNames()	This method returns an array of all properties (enumerable or not) found.
10	Object.getOwnPropertySymbols()	This method returns an array of all own symbol key properties.
11	Object.getPrototypeOf()	This method returns the prototype of the specified object.
12	Object.is()	This method determines whether two values are the same value.
13	Object.isExtensible()	This method determines if an object is extensible
14	Object.isFrozen()	This method determines if an object was frozen.
15	Object.isSealed()	This method determines if an object is sealed.
16	Object.keys()	This method returns an array of a given object's own property names.
17	Object.preventExtensions()	This method is used to prevent any extensions of an object.

18	Object.seal()	This method prevents new properties from being added and marks all existing properties as non-configurable.
19	Object.setPrototypeOf()	This method sets the prototype of a specified object to another object.
20	Object.values()	This method returns an array of values.

JavaScript Array

A JavaScript array is an object that represents a group of objects of the same type.

In JavaScript, there are three ways to create an array.

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

1) JavaScript array literal

The following is the syntax for generating an array with an array literal:

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

Values are contained within [] and separated by, as you can see (comma).

Let's have a look at a simple example of how to create and use an array in JavaScript.

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

Output

```
Sonoo
Vimal
Ratan
```

2) JavaScript Array directly

The following is the syntax for directly creating an array:

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

In this case, the new keyword is used to create an array object.

Let's look at an example of generating an array from scratch.

```
<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

```
Arun
Varun
John
```

3) JavaScript array constructor

So that we don't have to provide value explicitly, you must create an instance of array by passing arguments to the function Object() { [native code] }.

The following is an example of using the array function Object() { [native code] } to create an object.

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

Output

```
Jai
Vijay
Smith
```

JavaScript Array Methods

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 modified 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 function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new 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 function conditions.
find()	It returns the value of the first element in the given array that satisfies the specified condition.
findIndex()	It returns the index value of the first element in the given array that satisfies the 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 first 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 through these keys.
lastIndexOf()	It searches the specified element in the given array and returns the index of the last 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 of 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 array to a single value.
reduceRight()	It executes a provided function for each value from right to left and reduces the array to 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.

sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.
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 the 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

A string in JavaScript is an object that represents a string of characters.

In JavaScript, there are two ways to make a string.

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

1) By string literal

Double quotations are used to construct the string literal. The following is the syntax for creating a string using a string literal:

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

EXAMPLE

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

Output:

This is string literal

2) By string object (using new keyword)

The following is the syntax for creating a string object with the new keyword:

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

The new keyword is used to create a string instance.

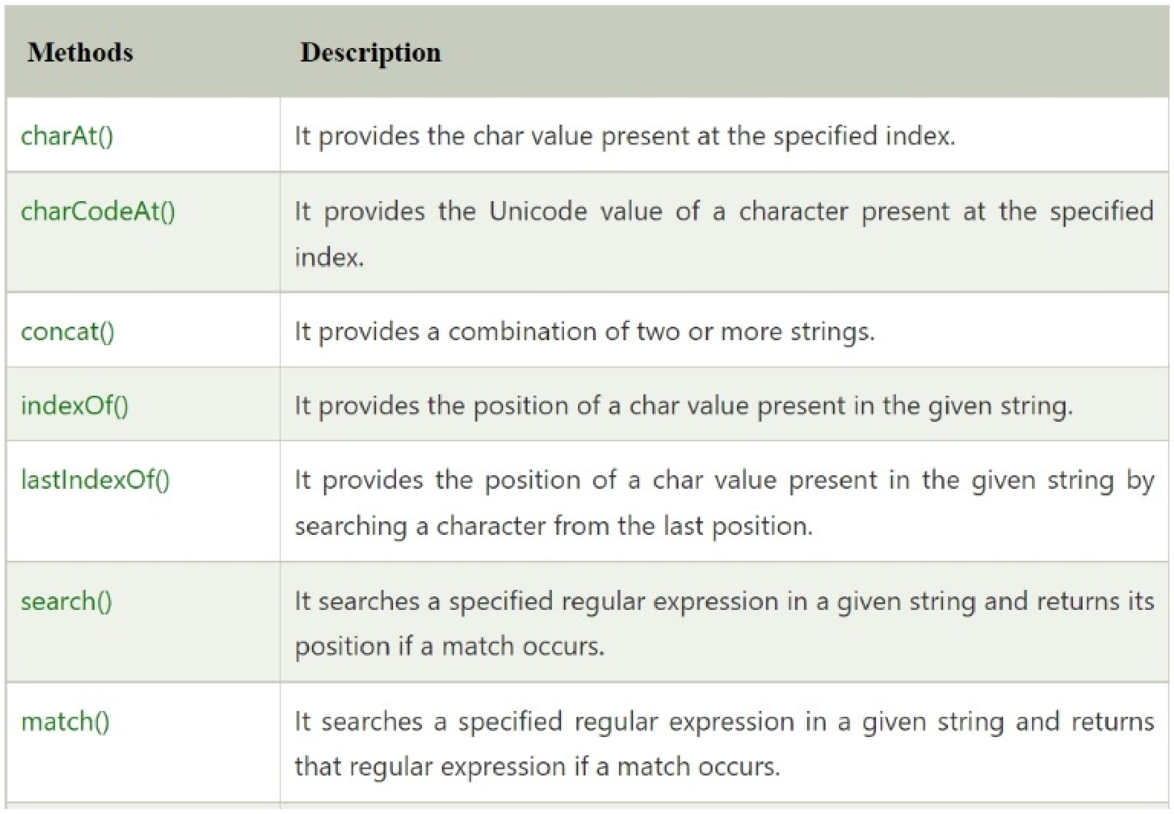
Let's look at an example of using the new keyword in JavaScript to create a string.

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

OUTPUT

```
hello javascript string
```

JavaScript String Methods



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.
lastIndexOf()	It provides the position of a char value present in the given string by searching a character from the last position.
search()	It searches a specified regular expression in a given string and returns its position if a match occurs.
match()	It searches a specified regular expression in a given string and returns that regular 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 starting 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 well 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.
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 Date Object

You may get the year, month, and day using the JavaScript date object. The JavaScript date object can be used to display a timer on a webpage.

To generate a date object, you can use a variety of Date constructors. It has methods for retrieving and setting the day, month, year, hour, minute, and second.

Constructor

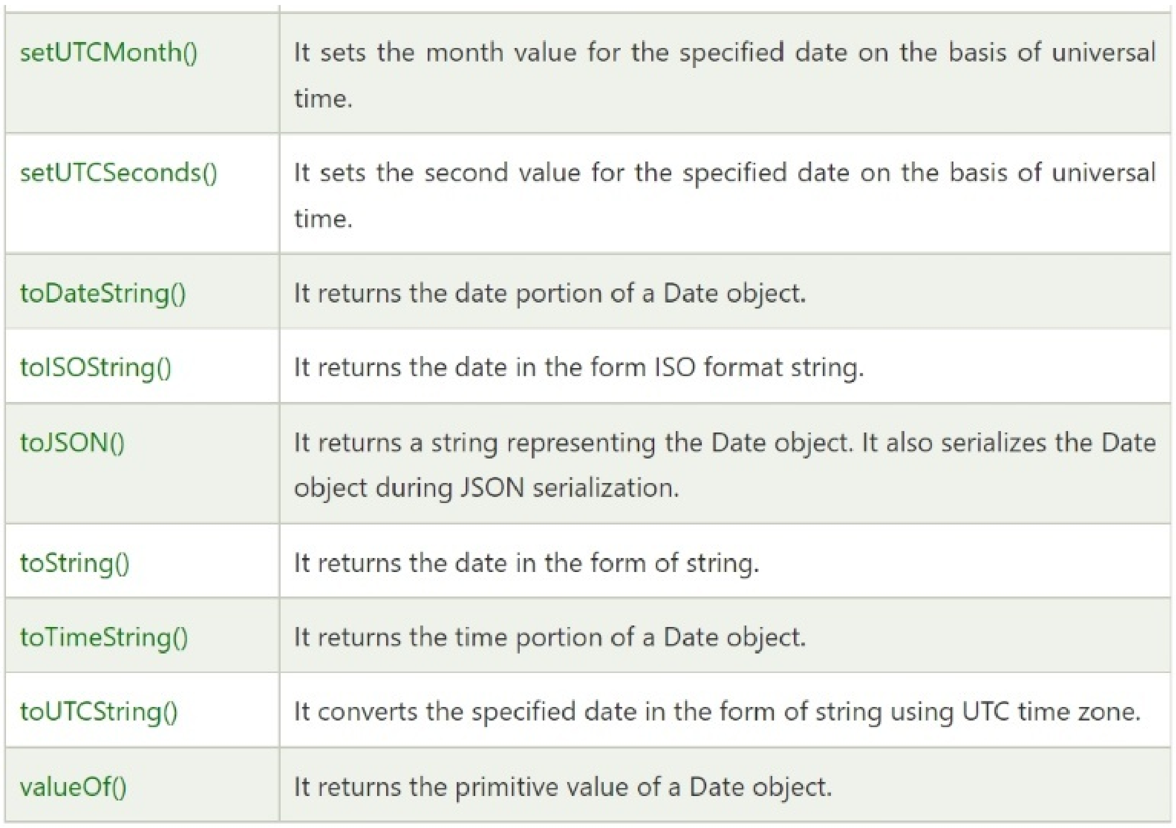
4 variant of Date constructor

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

JavaScript Date Methods

Methods	Description
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified 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 the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.
getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.
getUTCDate()	It returns the integer value between 1 and 31 that represents the day for the specified 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 the 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 basis of universal time.
getUTCMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of universal time.
getUTCMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of universal time.
getUTCSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of 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.



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 during 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.

JavaScript Date Example

```
Current Date and Time: <span id="txt"></span>
<script>
var today=new Date();
document.getElementById('txt').innerHTML=today;
</script>
```

OUTPUT

```
Current Date and Time: Sat Feb 12 2022 23:47:52 GMT+0530 (India Standard Time)
```

EXAMPLE 2

```
<script>
var date=new Date();
var day=date.getDate();
var month=date.getMonth()+1;
var year=date.getFullYear();
document.write("<br>Date is: "+day+"/"+month+"/"+year);
</script>
```

JavaScript Current Time Example

```
Current Time: <span id="txt"></span>
<script>
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
</script>
```

Digital Clock Example

Let's look at a simple example of using the JavaScript date object to display a digital clock.

In JavaScript, you can specify interval using the setTimeout() or setInterval() methods.

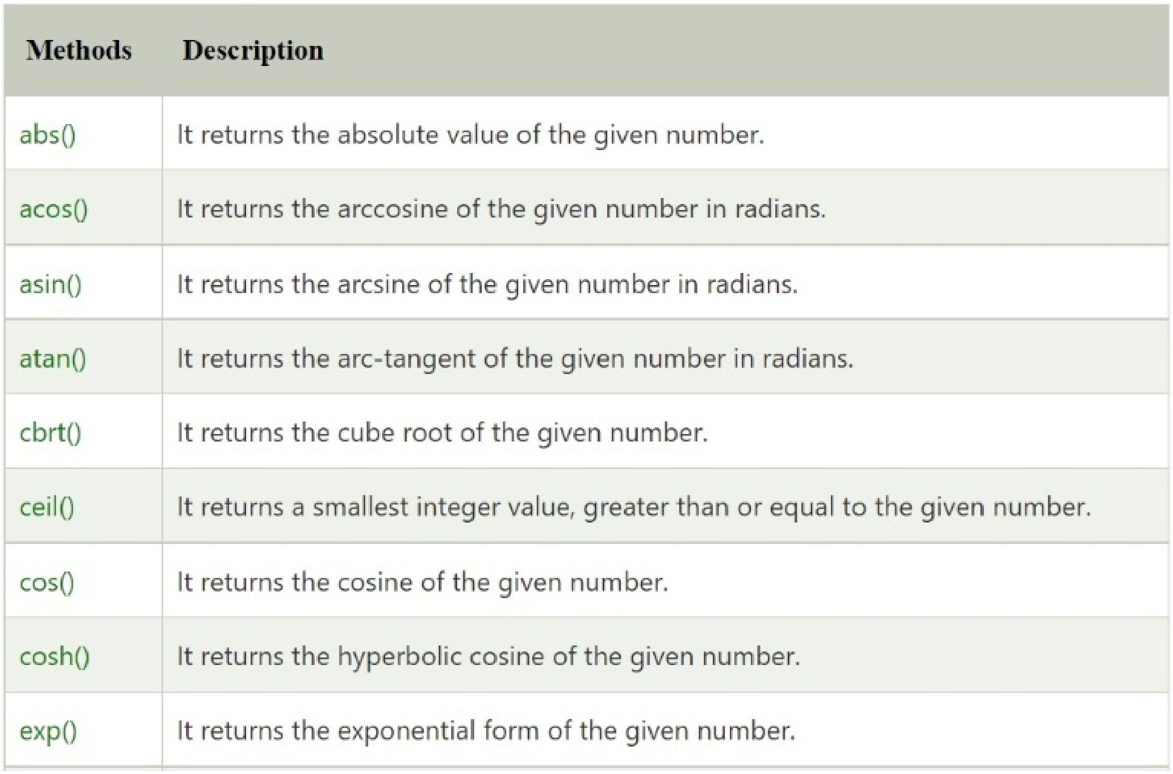
EXAMPLE

```
Current Time: <span id="txt"></span>
<script>
window.onload=function(){getTime();}
function getTime(){
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
          m=checkTime(m);
          s=checkTime(s);
          document.getElementById('txt').innerHTML=h+":"+m+":"+s;
          setTimeout(function(){getTime()},1000);
          }
          //setInterval("getTime()",1000);//another way
          function checkTime(i){
          if (i<10){
            i="0" + i;
           }
          return i;
          }
          </script>
```

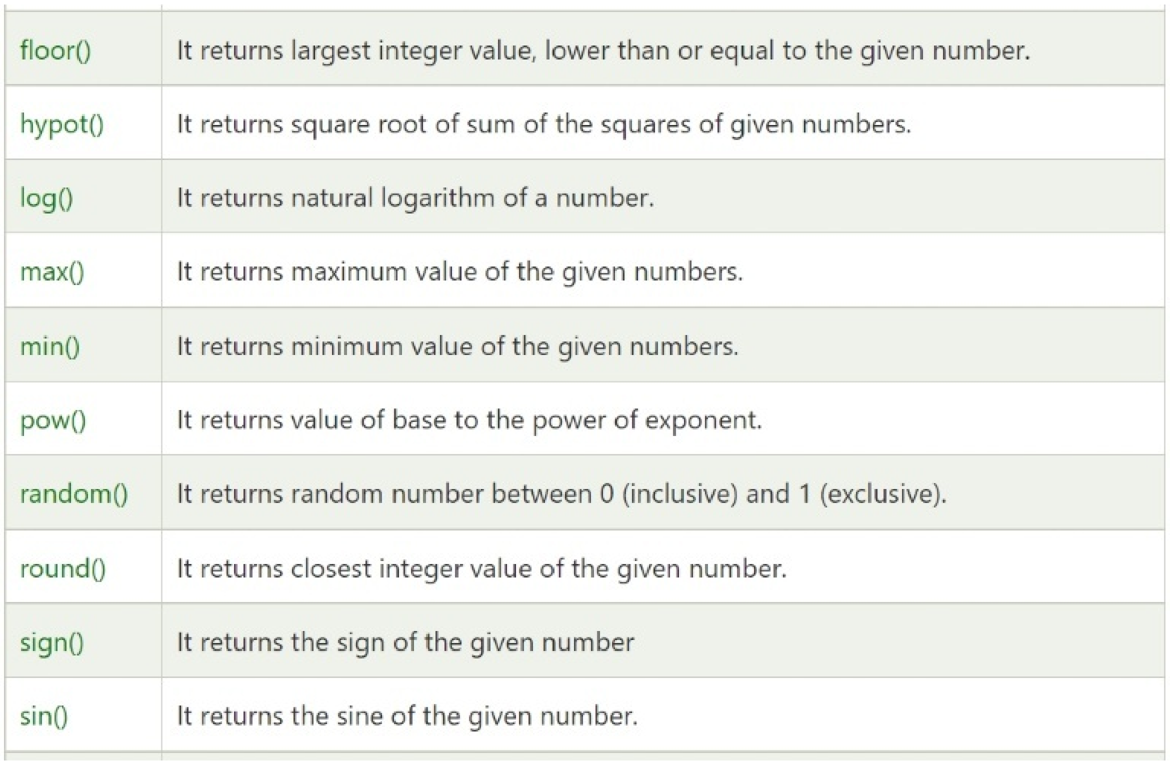
JavaScript Math

To conduct mathematical operations, the JavaScript math object exposes various constants and methods. It does not have constructors, unlike the date object.

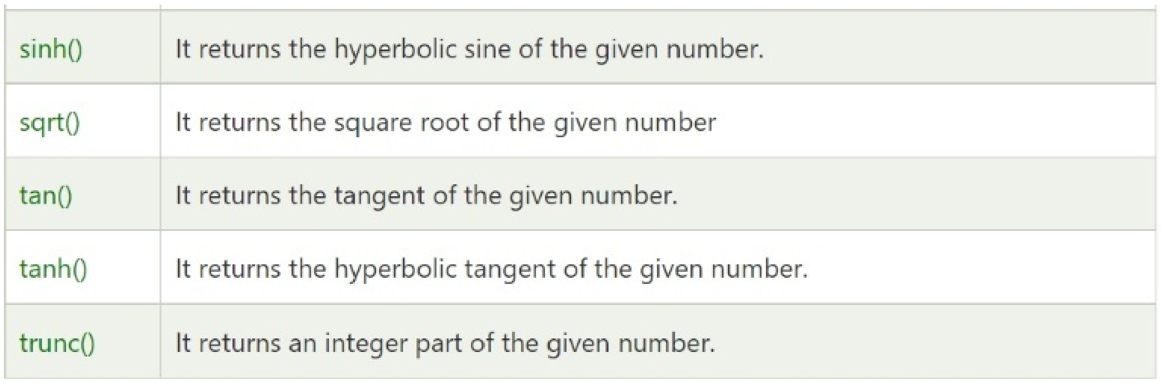
JavaScript Math Methods



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.



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.

Math.sqrt(n)

The square root of a number is returned by the JavaScript math.sqrt(n) method.

```
Square Root of 17 is: <span id="p1"></span>
<script>
document.getElementById('p1').innerHTML=Math.sqrt(17);
</script>
```

Output:

Square Root of 17 is: 4.123105625617661

Math.random()

The math.random() method in JavaScript returns a random number between 0 and 1.

```
Random Number is: <span id="p2"></span>
<script>
document.getElementById('p2').innerHTML=Math.random();
</script>
```

Output:

Random Number is: 0.02157876559753591

Math.pow(m,n)

```
3 to the power of 4 is: <span id="p3"></span>
<script>
document.getElementById('p3').innerHTML=Math.pow(3,4);
</script>
```

Output:

3 to the power of 4 is: 81

Math.floor(n)

The JavaScript math.floor(n) method returns the number's lowest integer. For instance, 3 for 3.7, 5 for 5.9, and so on.

```
Floor of 4.6 is: <span id="p4"></span>
<script>
document.getElementById('p4').innerHTML=Math.floor(4.6);
</script>
```

Output:

```
Floor of 4.6 is: 4
```

Math.ceil(n)

The math.ceil(n) method in JavaScript returns the largest integer for a given value. For instance, 4 for 3.7, 6 for 5.9, and so on.

```
Ceil of 4.6 is: <span id="p5"></span>
<script>
document.getElementById('p5').innerHTML=Math.ceil(4.6);
</script>
```

Output:

```
Ceil of 4.6 is: 5
```

Math.round(n)

The math in JavaScript. The round(n) method gives the nearest rounded integer for a supplied number. If the fractional component is equal to or higher than 0.5, it is assigned to the upper value 1; otherwise, it is assigned to the lower value 0. For instance, 4 for 3.7, 3 for 3.3, 6 for 5.9, and so on.

```
Round of 4.3 is: <span id="p6"></span><br>
Round of 4.7 is: <span id="p7"></span>
<script>
document.getElementById('p6').innerHTML=Math.round(4.3);
document.getElementById('p7').innerHTML=Math.round(4.7);
</script>
```

Output:

```
Round of 4.3 is: 4
Round of 4.7 is: 5
```

JavaScript Number Object

You can represent a numeric value with the JavaScript number object. It could be either an integer or a floating-point number. The IEEE standard for representing floating-point numbers is followed by the JavaScript number object.

In JavaScript, you can make a number object with the Number() function Object() { [native code] }. Consider the following scenario:

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

If the value cannot be converted to a number, it returns NaN (Not a Number), which the isNaN() method can validate.

You can also assign a number directly to a variable. Consider the following scenario:

```
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

Constant	Description
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

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

JavaScript Boolean

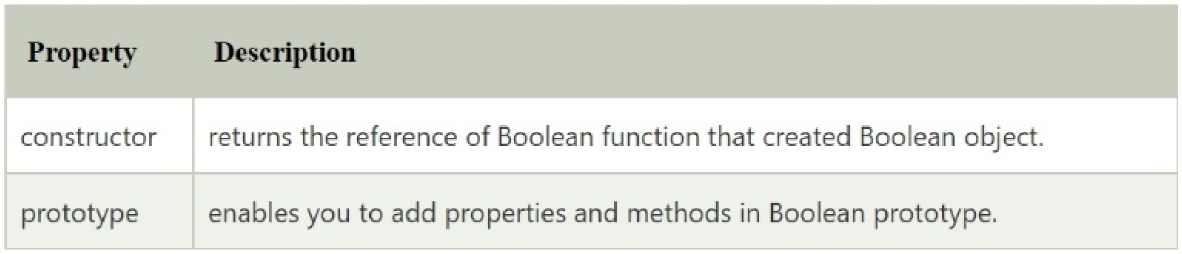
A Boolean object in JavaScript represents a value in one of two states: true or false. The Boolean() function Object() { [native code] } in JavaScript can be used to build a JavaScript Boolean object, as seen below.

```
Boolean b=new Boolean(value);
```

JavaScript Boolean Example

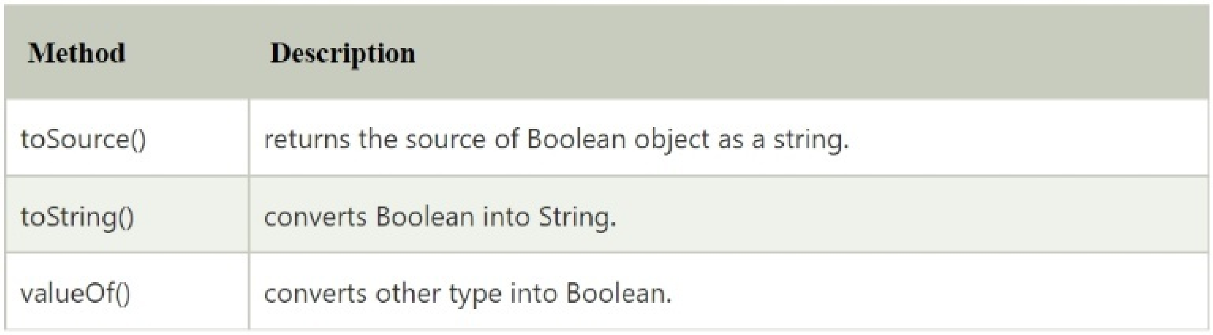
```
<script>
document.write(10<20);//true
document.write(10<5);//false
</script>
```

Boolean Properties



Property	Description
constructor	returns the reference of Boolean function that created Boolean object.
prototype	enables you to add properties and methods in Boolean prototype.

Boolean Methods



Method	Description
toSource()	returns the source of Boolean object as a string.
toString()	converts Boolean into String.
valueOf()	converts other type into Boolean.