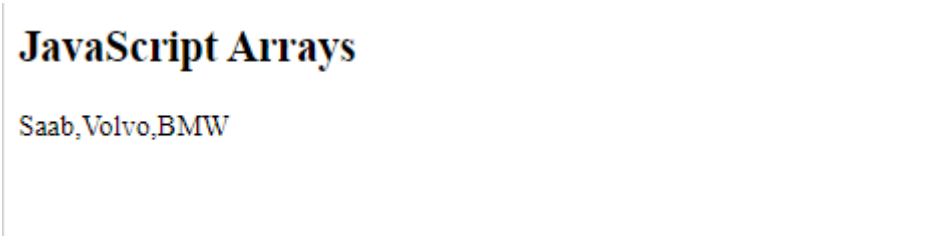
ARRAY:

1. **Display the array content.**

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

<h2>JavaScript Arrays</h2>

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

<script>
const cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars;
</script>

</body>
</html>
```




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

<h2>JavaScript Arrays</h2>

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

<script>
const cars = [
  "Saab",
  "Volvo",
  "BMW"
];
document.getElementById("demo").innerHTML = cars;
</script>

</body>
</html>
```

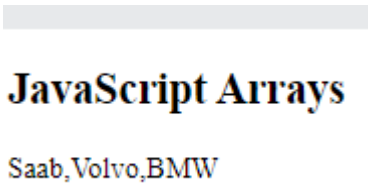
2. You can also create an array, and then provide the elements:

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

<h2>JavaScript Arrays</h2>

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

<script>
const cars = [];
cars[0]= "Saab";
cars[1]= "Volvo";
cars[2]= "BMW";
document.getElementById("demo").innerHTML = cars;
</script>

</body>
</html>
```

3.Using the JavaScript Keyword new

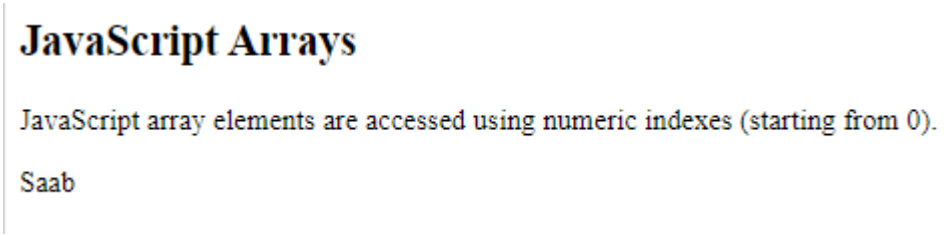
The following example also creates an Array, and assigns values to it.

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

<h2>JavaScript Arrays</h2>

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

<script>
const cars = new Array("Saab", "Volvo", "BMW");
document.getElementById("demo").innerHTML = cars;
</script>

</body>
</html>
```



4. Accessing Array Elements

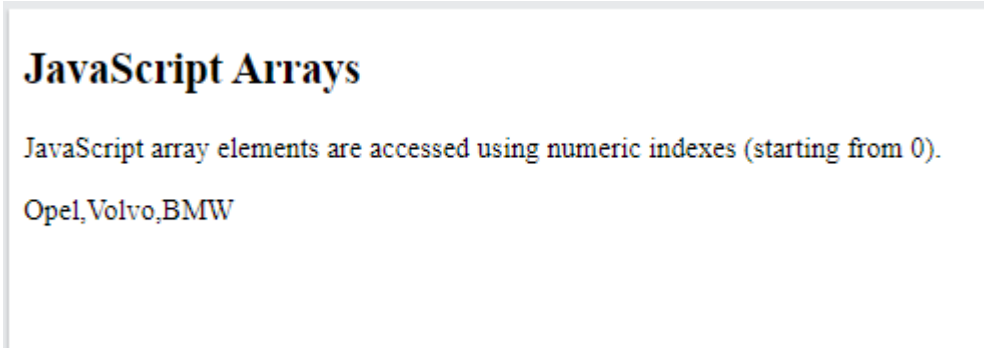
You access an array element by referring to the **index number**




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

<h2>JavaScript Arrays</h2>

<p>JavaScript array elements are accessed using numeric indexes (starting from 0).</p

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

<script>
const cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars[0];
</script>

</body>
</html>
```

5. Changing an Array Element

This statement changes the value of the first element in `cars`:



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

<h2>JavaScript Arrays</h2>

<p>JavaScript array elements are accessed using numeric indexes (starting from 0).</p>

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

<script>
const cars = ["Saab", "Volvo", "BMW"];
cars[0] = "Opel";
document.getElementById("demo").innerHTML = cars;
</script>

</body>
</html>
```

7. Access the Full Array

With JavaScript, the full array can be accessed by referring to the array name:

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

<h2>JavaScript Arrays</h2>

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

<script>
const cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars;
</script>

</body>
</html>
```

8. Arrays are Objects

Arrays are a special type of objects. The `typeof` operator in JavaScript returns "object" for arrays.

But, JavaScript arrays are best described as arrays.

Arrays use **numbers** to access its "elements". In this example, `person[0]` returns John:

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

<h2>JavaScript Arrays</h2>

<p>Arrays use numbers to access its elements.</p>

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

<script>
const person = ["John", "Doe", 46];
document.getElementById("demo").innerHTML = person[0];
</script>

</body>
</html>
```

JavaScript Arrays

Arrays use numbers to access its elements

John

10. Objects use **names** to access its "members". In this example, `person.firstName` returns John:

Object:

```
const person = {firstName:"John", lastName:"Doe", age:46};
```

JavaScript Objects

JavaScript uses names to access object properties.

John

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

<h2>JavaScript Objects</h2>
<p>JavaScript uses names to access object properties.</p>
<p id="demo"></p>

<script>
const person = {firstName:"John", lastName:"Doe", age:46};
document.getElementById("demo").innerHTML = person.firstName;
</script>

</body>
</html>
```

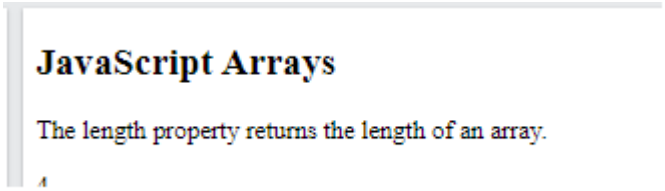
12. Array Properties and Methods

The real strength of JavaScript arrays are the built-in array properties and methods:

```
cars.length   // Returns the number of elements
cars.sort()   // Sorts the array
```

The length Property

The `length` property of an array returns the length of an array (the number of array elements).




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

<h2>JavaScript Arrays</h2>
<p>The length property returns the length of an array.</p>

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

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.length;
</script>

</body>
</html>
```

14. Accessing the First Array Element

Example

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let fruit = fruits[0];
```

15. Accessing the Last Array Element

Example

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let fruit = fruits[fruits.length - 1];
```

16. Looping Array Elements

One way to loop through an array, is using a `for` loop:

JavaScript Arrays

The best way to loop through an array is using a standard for loop:

- Banana
- Orange
- Apple
- Mango

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

<h2>JavaScript Arrays</h2>

<p>The best way to loop through an array is using a standard for loop:</p>

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

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let fLen = fruits.length;

let text = "<ul>";
for (let i = 0; i < fLen; i++) {
  text += "<li>" + fruits[i] + "</li>";
}
text += "</ul>";

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

</body>
</html>
```

16. You can also use the `Array.forEach()` function:

JavaScript Arrays

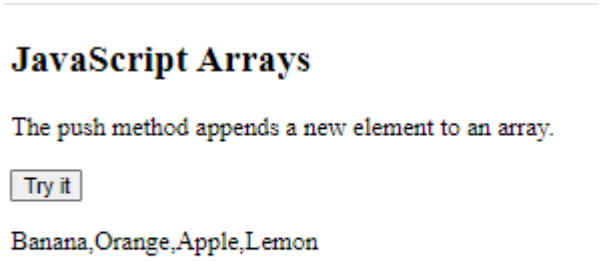
Array.forEach() calls a function for each array element.

- Banana
- Orange
- Apple
- Mango

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

<h2>JavaScript Arrays</h2>

<p>Array.forEach() calls a function for each array element.</p>

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

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];

let text = "<ul>";
fruits.forEach(myFunction);
text += "</ul>";
document.getElementById("demo").innerHTML = text;

function myFunction(value) {
  text += "<li>" + value + "</li>";
}
</script>

</body>
</html>
```

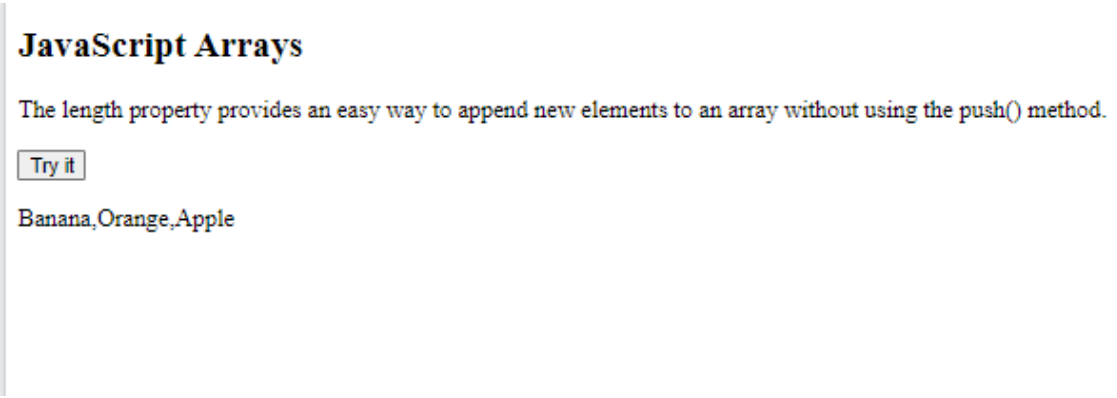
19. Adding Array Elements

The easiest way to add a new element to an array is using the `push()` method:



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

<h2>JavaScript Arrays</h2>

<p>The push method appends a new element to an array.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
const fruits = ["Banana", "Orange", "Apple"];
document.getElementById("demo").innerHTML = fruits;

function myFunction() {
  fruits.push("Lemon");
  document.getElementById("demo").innerHTML = fruits;
}
</script>

</body>
</html>
```

20. New element can also be added to an array using the `length` property:



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

<h2>JavaScript Arrays</h2>

<p>The length property provides an easy way to append new elements to an array without using t
push() method.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
const fruits = ["Banana", "Orange", "Apple"];
document.getElementById("demo").innerHTML = fruits;

function myFunction() {
  fruits[fruits.length] = "Lemon";
  document.getElementById("demo").innerHTML = fruits;
}
</script>

</body>
</html>
```

21. Adding elements with high indexes can create undefined "holes" in an array:

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

<h2>JavaScript Arrays</h2>

<p>Adding elements with high indexes can create undefined "holes" in an array.</p>

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

<script>
const fruits = ["Banana", "Orange", "Apple"];
fruits[6] = "Lemon";

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

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

</body>
</html>
```

JavaScript Arrays

Adding elements with high indexes can create undefined "holes" in an array.

Banana
Orange
Apple
undefined
undefined
undefined
Lemon

22. How to Recognize an Array

A common question is: How do I know if a variable is an array?

The problem is that the JavaScript operator `typeof` returns "`object`":

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

<h2>JavaScript Arrays</h2>

<p>The typeof operator, when used on an array, returns object:</p>

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

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = typeof fruits;
</script>

</body>
</html>
```

JavaScript Arrays

The typeof operator, when used on an array, returns object:

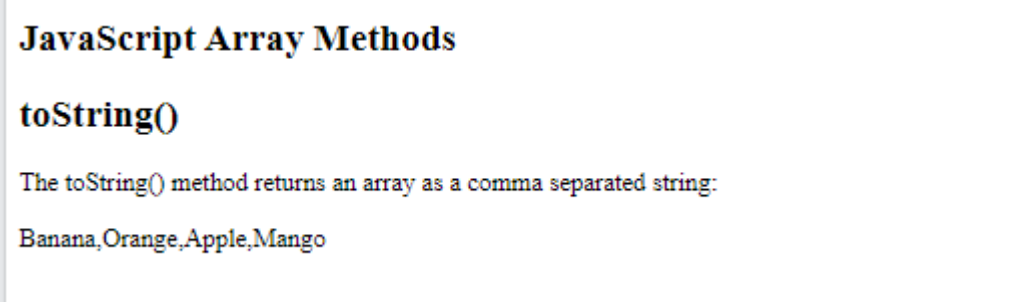
object

23. Converting Arrays to Strings

The JavaScript method `toString()` converts an array to a string of (comma separated) array values.

Example

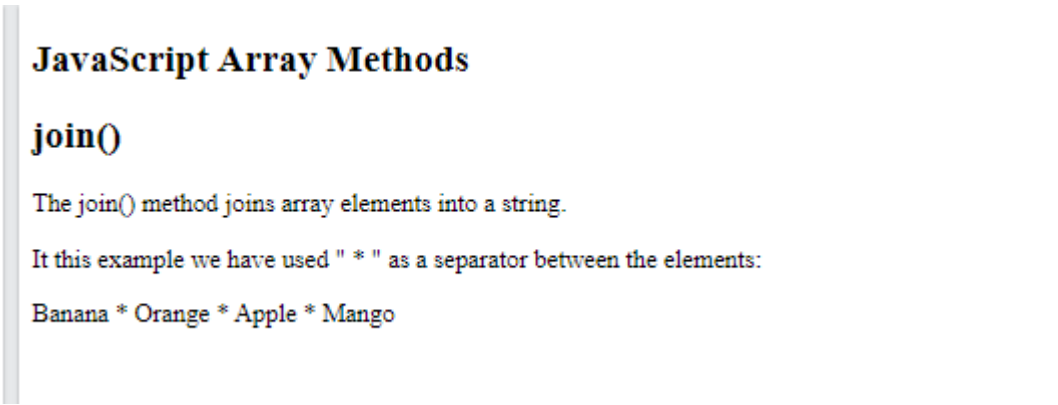
```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
```




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

<h2>JavaScript Array Methods</h2>
<h2>toString()</h2>
<p>The toString() method returns an array as a comma separated string:</p>

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

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
</script>

</body>
</html>
```

23.The `join()` method also joins all array elements into a string.

It behaves just like `toString()`, but in addition you can specify the separator:



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

<h2>JavaScript Array Methods</h2>
<h2>join()</h2>
<p>The join() method joins array elements into a string.</p>
<p>It this example we have used " * " as a separator between the elements:</p>

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

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.join(" * ");
</script>

</body>
</html>
```

24. Popping and Pushing

When you work with arrays, it is easy to remove elements and add new elements.

This is what popping and pushing is:

Popping items **out** of an array, or pushing items **into** an array.

JavaScript Array pop()

The `pop()` method removes the last element from an array:

Example

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();
```

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

<h2>JavaScript Array Methods</h2>
<h2>pop()</h2>
<p>The pop() method removes the last element from an array.</p>

<p id="demo1"></p>
<p id="demo2"></p>

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;
fruits.pop();
document.getElementById("demo2").innerHTML = fruits;
</script>

</body>
</html>
```

JavaScript Array Methods

pop()

The pop() method removes the last element from an array.

Banana,Orange,Apple,Mango

Banana,Orange,Apple

JavaScript Array Methods

concat()

The concat() method merges (concatenates) arrays:

Cecilie,Lone,Emil,Tobias,Linus

25.

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

<h2>JavaScript Array Methods</h2>
<h2>concat()</h2>
<p>The concat() method merges (concatenates) arrays:</p>

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

<script>
const myGirls = ["Cecilie", "Lone"];
const myBoys = ["Emil", "Tobias", "Linus"];
const myChildren = myGirls.concat(myBoys);

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

</body>
</html>
```

JavaScript Array Sort

The sort() method sorts an array alphabetically:

Banana,Orange,Apple,Mango

Apple,Banana,Mango,Orange

26.

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

<h2>JavaScript Array Sort</h2>
<p>The sort() method sorts an array alphabetically:</p>

<p id="demo1"></p>
<p id="demo2"></p>

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;

fruits.sort();
document.getElementById("demo2").innerHTML = fruits;
</script>

</body>
</html>
```

27. Reversing an Array

The `reverse()` method reverses the elements in an array.

You can use it to sort an array in descending order:

> **JavaScript Array Sort Reverse**
>
> The reverse() method reverses the elements in an array.
>
> By combining sort() and reverse() you can sort an array in descending order:
>
> Banana,Orange,Apple,Mango
>
> Orange,Mango,Banana,Apple

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

<h2>JavaScript Array Sort Reverse</h2>

<p>The reverse() method reverses the elements in an array.</p>
<p>By combining sort() and reverse() you can sort an array in descending order:</p>

<p id="demo1"></p>
<p id="demo2"></p>

<script>
// Create and display an array:
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;

// First sort the array
fruits.sort();

// Then reverse it:
fruits.reverse();

document.getElementById("demo2").innerHTML = fruits;
</script>

</body>
</html>
```

28. String Length

To find the length of a string, use the built-in `length` property:

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

<h2>JavaScript String Properties</h2>

<p>The length property returns the length of a string:</p>

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

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

</body>
</html>
```

JavaScript String Properties

The length property returns the length of a string:

26

29. Extracting String Parts

There are 3 methods for extracting a part of a string:

- `slice(start, end)`
- `substring(start, end)`
- `substr(start, length)`

slice

JavaScript String Methods

The slice() method extract a part of a string and returns the extracted parts in a new string:

ple, Banana

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

<h2>JavaScript String Methods</h2>

<p>The slice() method extract a part of a string
and returns the extracted parts in a new string:</p>

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

<script>
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML = str.slice(2,13);
</script>

</body>
</html>
```

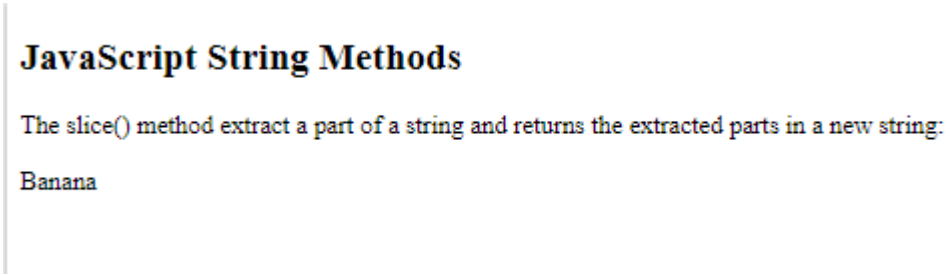
27. If a parameter is negative, the position is counted from the end of the string.

This example slices out a portion of a string from position -12 to position -6:

Example

```
let str = "Apple, Banana, Kiwi";
let part = str.slice(-12, -6);
```

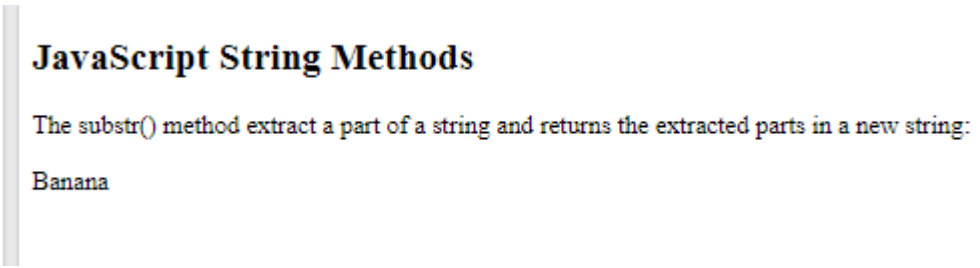
```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The slice() method extract a part of a string
and returns the extracted parts in a new string:</p>

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

<script>
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML = str.slice(-12,-6);
</script>

</body>
</html>
```



29. JavaScript String substr()

`substr()` is similar to `slice()`.

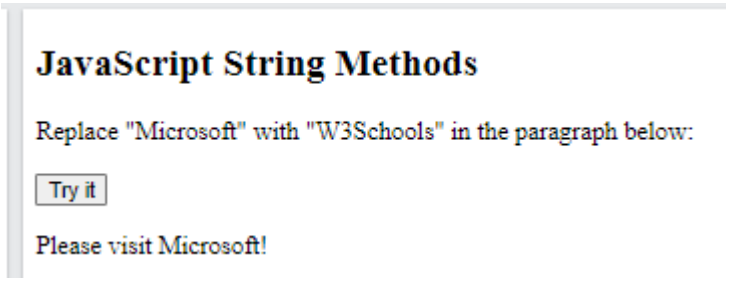
The difference is that the second parameter specifies the **length** of the extracted part.

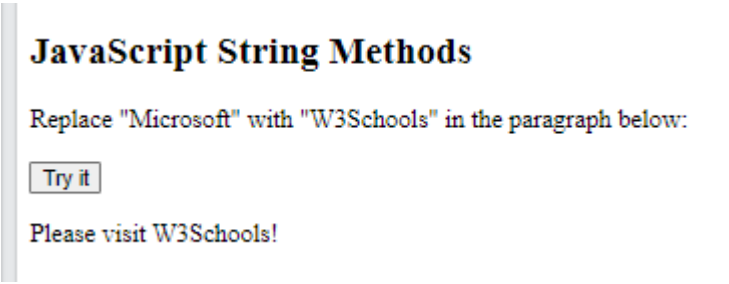



```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>The substr() method extract a part of a string
and returns the extracted parts in a new string:</p>

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

<script>
let str = "Apple, Banana, Kiwi";
document.getElementById("demo").innerHTML = str.substr(7,6);
</script>

</body>
</html>
```

Replacing String Content

The `replace()` method replaces a specified value with another value in a string:






```
<!DOCTYPE html>
<html>

<body>

<h2>JavaScript String Methods</h2>

<p>Replace "Microsoft" with "W3Schools" in the paragraph below:</p>

<button onclick="myFunction()">Try it</button>

<p id="demo">Please visit Microsoft!</p>

<script>
function myFunction() {
  let text = document.getElementById("demo").innerHTML;
  document.getElementById("demo").innerHTML =
  text.replace("Microsoft","W3Schools");
}
</script>

</body>
</html>
```

Converting to Upper and Lower Case

A string is converted to upper case with `toUpperCase()`:

A string is converted to lower case with `toLowerCase()`:

```
let text1 = "Hello World!";
let text2 = text1.toUpperCase();
```

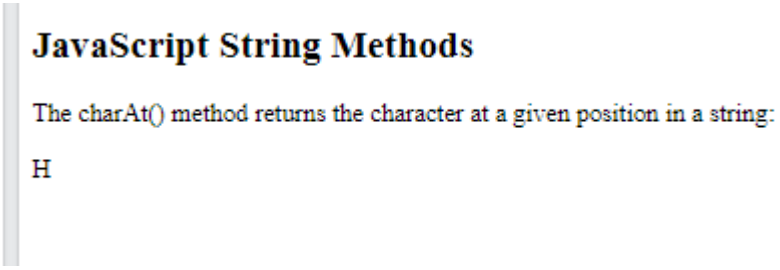
Extracting String Characters

There are 3 methods for extracting string characters:

- `charAt(position)`
- `charCodeAt(position)`
- Property access [ ]

JavaScript String charAt()

The `charAt()` method returns the character at a specified index (position) in a string:




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

<h2>JavaScript String Methods</h2>

<p>The charAt() method returns the character at a given position in a string:</p>

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

<script>
var text = "HELLO WORLD";
document.getElementById("demo").innerHTML = text.charAt(0);
</script>

</body>
</html>
```

JavaScript String charCodeAt()

The `charCodeAt()` method returns the unicode of the character at a specified index in a string:

The method returns a UTF-16 code (an integer between 0 and 65535).

Example

> **JavaScript String Methods**
>
> The charCodeAt() method returns the unicode of the character at a given position in a string:
>
> 72

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

<h2>JavaScript String Methods</h2>

<p>The charCodeAt() method returns the unicode of the character at a given position in a string:
</p>

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

<script>
let text = "HELLO WORLD";
document.getElementById("demo").innerHTML = text.charCodeAt(0);
</script>

</body>
</html>
```

JavaScript Search Methods

- String indexOf()
- String lastIndexOf()
- String startsWith()
- String endsWith()

JavaScript String indexOf()

The `indexOf()` method returns the index of (the position of) the `first` occurrence of a specified text in a string:

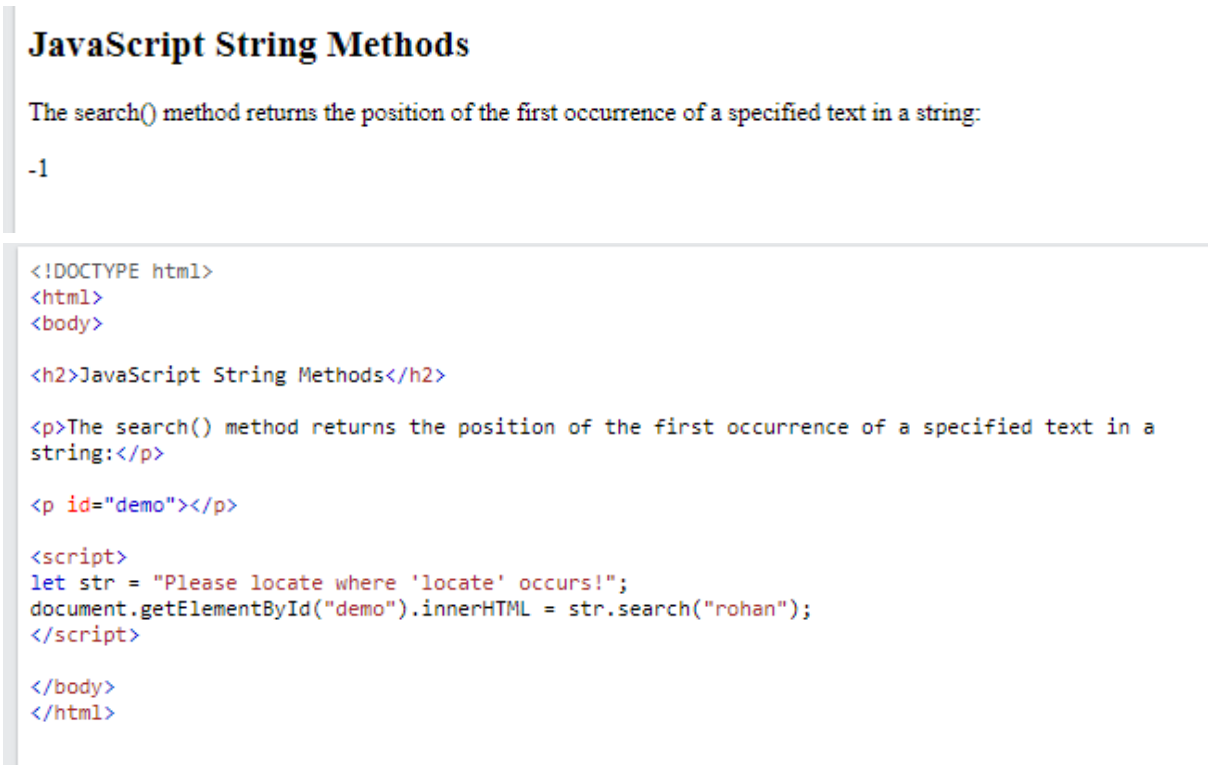
> **JavaScript String Methods**
>
> The indexOf() method returns the position of the first occurrence of a specified text:
>
> 7

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

<h2>JavaScript String Methods</h2>

<p>The indexOf() method returns the position of the first occurrence of a specified text:</p>

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

<script>
let str = "Please locate where 'locate' occurs!";
document.getElementById("demo").innerHTML = str.indexOf("locate");
</script>

</body>
</html>
```

JavaScript String search()

The `search()` method searches a string for a specified value and returns the position of the match:

Example




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

<h2>JavaScript String Methods</h2>

<p>The search() method returns the position of the first occurrence of a specified text in a
string:</p>

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

<script>
let str = "Please locate where 'locate' occurs!";
document.getElementById("demo").innerHTML = str.search("rohan");
</script>

</body>
</html>
```

JavaScript String Methods

The search() method returns the position of the first occurrence of a specified text in a string:

14

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

<h2>JavaScript String Methods</h2>

<p>The search() method returns the position of the first occurrence of a specified text in a
string:</p>

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

<script>
let str = "Please locate where 'locate' occurs!";
document.getElementById("demo").innerHTML = str.search("where");
</script>

</body>
</html>
```

JavaScript Date Objects

JavaScript new Date()

Thu Jun 09 2022 23:53:28 GMT+0530 (India Standard Time)

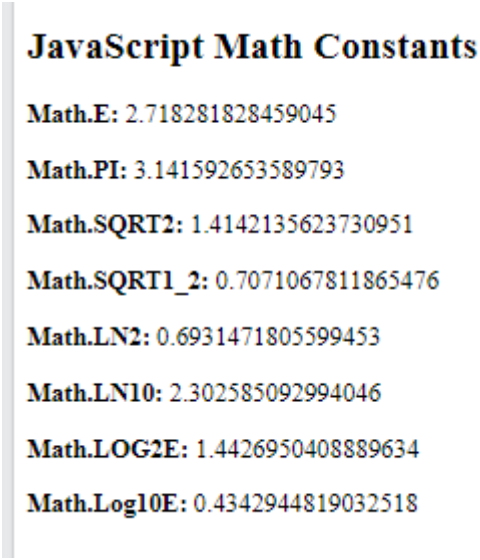
```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript new Date()</h2>

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

<script>
const d = new Date();
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript Math Object

Math Properties (Constants)

The syntax for any Math property is : `Math.property`.

JavaScript provides 8 mathematical constants that can be accessed as Math properties:



JavaScript Math Constants

Math.E: 2.718281828459045

Math.PI: 3.141592653589793

Math.SQRT2: 1.4142135623730951

Math.SQRT1_2: 0.7071067811865476

Math.LN2: 0.6931471805599453

Math.LN10: 2.302585092994046

Math.LOG2E: 1.4426950408889634

Math.Log10E: 0.4342944819032518

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

<h2>JavaScript Math Constants</h2>

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

<script>
document.getElementById("demo").innerHTML =
"<p><b>Math.E:</b> " + Math.E + "</p>" +
"<p><b>Math.PI:</b> " + Math.PI + "</p>" +
"<p><b>Math.SQRT2:</b> " + Math.SQRT2 + "</p>" +
"<p><b>Math.SQRT1_2:</b> " + Math.SQRT1_2 + "</p>" +
"<p><b>Math.LN2:</b> " + Math.LN2 + "</p>" +
"<p><b>Math.LN10:</b> " + Math.LN10 + "</p>" +
"<p><b>Math.LOG2E:</b> " + Math.LOG2E + "</p>" +
"<p><b>Math.Log10E:</b> " + Math.LOG10E + "</p>";
</script>

</body>
</html>
```

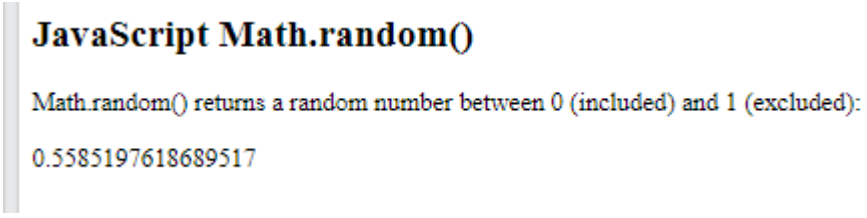
Number to Integer

There are 4 common methods to round a number to an integer:

Math.round(x)	Returns x rounded to its nearest integer
Math.ceil(x)	Returns x rounded up to its nearest integer
Math.floor(x)	Returns x rounded down to its nearest integer
Math.trunc(x)	Returns the integer part of x (new in ES6)

Math.random()

`Math.random()` returns a random number between 0 (inclusive), and 1 (exclusive):



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

<h2>JavaScript Math.random()</h2>

<p>Math.random() returns a random number between 0 (included) and 1 (excluded):</p>

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

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

</body>
</html>
```