Javascript

Ken Ratri

2018

Objective

- String Method
- Number Methods
- Array
- Array Method
- Date

String Methods

- String methods help you to work with strings

Method	Description
charAt()	Returns the character at the specified index (position)
concat()	Joins two or more strings, and returns a new joined strings
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
match()	Searches a string for a match against a regular expression, and returns the matches
replace()	Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced
search()	Searches a string for a specified value, or regular expression, and returns the position of the match
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices

String Method

- String methods help you to work with strings.
- The **length** property returns the length of a string

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

<script>
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
document.getElementById("demo").innerHTML = txt.length;
</script>
```

Number Methods

- Number methods help you work with numbers.
- **toFixed()** returns a string, with the number written with a specified number of decimals:

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

<script>
var x = 9.656;
document.getElementById("demo").innerHTML =
    x.toFixed(0) + "<br>" +
    x.toFixed(2) + "<br>" +
    x.toFixed(4) + "<br>" +
    x.toFixed(6);
</script>
```

Number Methods

- **toPrecision()** returns a string, with a number written with a specified length:

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

<script>
var x = 9.656;
document.getElementById("demo").innerHTML =
    x.toPrecision() + "<br>" +
    x.toPrecision(2) + "<br>" +
    x.toPrecision(4) + "<br>" +
    x.toPrecision(6);
</script>
```

Number Methods

- **parseInt()** parses a string and returns a whole number. Spaces are allowed.
- **parseFloat()** parses a string and returns a number. Spaces are allowed.

```
parseInt("10");          // returns 10
parseInt("10.33");       // returns 10
parseInt("10 20 30");    // returns 10
parseInt("10 years");    // returns 10
parseInt("years 10");    // returns NaN
```

```
parseFloat("10");          // returns 10
parseFloat("10.33");       // returns 10.33
parseFloat("10 20 30");    // returns 10
parseFloat("10 years");    // returns 10
parseFloat("years 10");    // returns NaN
```

Array

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

```
<h2>JavaScript Arrays</h2>

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

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

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

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

Array Methods

- The concat() method creates a new array by merging (concatenating) existing arrays

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

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

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

Array Methods

- The **slice()** method slices out a piece of an array into a new array.
- This example slices out a part of an array starting from array element 1 ("Orange"):

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

<script>
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(1);
document.getElementById("demo").innerHTML = fruits + "<br><br>" + citrus;
</script>
```

Array Methods

- The sort() method sorts the items of an array.
- The sort order can be either alphabetic or numeric, and either ascending (up) or descending (down).

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

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

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

function myFunction() {
    fruits.sort();
    fruits.reverse();
    document.getElementById("demo").innerHTML = fruits;
}
</script>
```

Date

- The Date object is used to work with dates and times.
- Date objects are created with **new Date()**.

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

<script>
function myFunction() {
    var d = new Date();
    var n = d.getDay()
    document.getElementById("demo").innerHTML = n;
}
</script>
```

Date Object Methods

Method	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1 1970, and a specified date

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

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

<script>
function myFunction() {
    var d = new Date();
    var weekday = new Array(7);
    weekday[0] = "Sunday";
    weekday[1] = "Monday";
    weekday[2] = "Tuesday";
    weekday[3] = "Wednesday";
    weekday[4] = "Thursday";
    weekday[5] = "Friday";
    weekday[6] = "Saturday";

    var n = weekday[d.getDay()];
    document.getElementById("demo").innerHTML = n;
}
</script>
```

```
<script>
function startTime()
{
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;
t=setTimeout(function(){startTime()},500);
}

function checkTime(i)
{
if (i<10)
  {
  i="0" + i;
  }
return i;
}
</script>
</head>

<body onload="startTime()">
<div id="txt"></div>
```