**JavaScript String Methods**

## String Length

The **length** property returns the length of a string:

### Example

var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
var sln = txt.length;

## Finding a String in a String

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

### Example

var str = "Please locate where 'locate' occurs!";  
var pos = str.indexOf("locate");

The **lastIndexOf()** method returns the index of the **last** occurrence of a specified text in a string:

### Example

var str = "Please locate where 'locate' occurs!";  
var pos = str.lastIndexOf("locate");

JavaScript counts positions from zero.  
0 is the first position in a string, 1 is the second, 2 is the third ...

## Searching for a String in a String

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

### Example

var str = "Please locate where 'locate' occurs!";  
var pos = str.search("locate");

## Extracting String Parts

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

* slice(start, end)
* substring(start, end)
* substr(start, length)

## The slice() Method

**slice()** extracts a part of a string and returns the extracted part in a new string.

The method takes 2 parameters: the starting index (position), and the ending index (position).

This example slices out a portion of a string from position 7 to position 13:

### Example

var str = "Apple, Banana, Kiwi";  
var res = str.slice(7, 13);

## The substring() Method

**substring()** is similar to slice().

The difference is that substring() cannot accept negative indexes.

### Example

var str = "Apple, Banana, Kiwi";  
var res = str.substring(7, 13);

## Replacing String Content

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

### Example

str = "Please visit Microsoft!";  
var n = str.replace("Microsoft", "W3Schools");

The replace() method does not change the string it is called on. It returns a new string.

By default, the replace() function replaces **only the first** match:

## Converting to Upper and Lower Case

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

### Example

var text1 = "Hello World!";       // String  
var text2 = text1.toUpperCase();  // text2 is text1 converted to upper

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

### Example

var text1 = "Hello World!";       // String  
var text2 = text1.toLowerCase();  // text2 is text1 converted to lower

## The concat() Method

**concat()** joins two or more strings:

### Example

var text1 = "Hello";  
var text2 = "World";  
var text3 = text1.concat(" ", text2);

## Extracting String Characters

There are 2 **safe** methods for extracting string characters:

* charAt(position)
* charCodeAt(position)

## The charAt() Method

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

### Example

var str = "HELLO WORLD";  
str.charAt(0);            // returns H

## The charCodeAt() Method

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

### Example

var str = "HELLO WORLD";  
  
str.charCodeAt(0);         // returns 72

## Accessing a String as an Array is Unsafe

You might have seen code like this, accessing a string as an array:

var str = "HELLO WORLD";  
  
str[0];                   // returns H

This is **unsafe** and **unpredictable:**

* It does not work in all browsers (not in IE5, IE6, IE7)
* It makes strings look like arrays (but they are not)
* str[0] = "H" does not give an error (but does not work)

If you want to read a string as an array, convert it to an array first.

# JavaScript Number Methods

## The toString() Method

**toString()** returns a number as a string.

All number methods can be used on any type of numbers (literals, variables, or expressions):

## The toFixed() Method

**toFixed()** returns a string, with the number written with a specified number of decimals:

### Example

var x = 9.656;  
x.toFixed(0);           // returns 10  
x.toFixed(2);           // returns 9.66  
x.toFixed(4);           // returns 9.6560  
x.toFixed(6);           // returns 9.656000

toFixed(2) is perfect for working with money.

## The toPrecision() Method

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

### Example

var x = 9.656;  
x.toPrecision();        // returns 9.656  
x.toPrecision(2);       // returns 9.7  
x.toPrecision(4);       // returns 9.656  
x.toPrecision(6);       // returns 9.65600

## The valueOf() Method

**valueOf()** returns a number as a number.

### Example

var x = 123;  
x.valueOf();            // returns 123 from variable x  
(123).valueOf();        // returns 123 from literal 123  
(100 + 23).valueOf();   // returns 123 from expression 100 + 23

## Converting Variables to Numbers

There are 3 JavaScript methods that can be used to convert variables to numbers:

* The Number() method
* The parseInt() method
* The parseFloat() method

## Global Methods

JavaScript global methods can be used on all JavaScript data types.

These are the most relevant methods, when working with numbers:

|  |  |
| --- | --- |
| **Method** | **Description** |
| Number() | Returns a number, converted from its argument. |
| parseFloat() | Parses its argument and returns a floating point number |
| parseInt() | Parses its argument and returns an integer |

## The Number() Method

**Number()** can be used to convert JavaScript variables to numbers:

### Example

Number(true);          // returns 1  
Number(false);         // returns 0  
Number("10");          // returns 10  
Number("  10");        // returns 10  
Number("10  ");        // returns 10  
Number("10 20");       // returns NaN   
Number("John");        // returns NaN

## The parseInt() Method

**parseInt()** parses a string and returns a whole number. Spaces are allowed. Only the first number is returned:

### Example

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

## The parseFloat() Method

**parseFloat()** parses a string and returns a number. Spaces are allowed. Only the first number is returned:

### Example

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

## Number Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| MAX\_VALUE | Returns the largest number possible in JavaScript |
| MIN\_VALUE | Returns the smallest number possible in JavaScript |
| NEGATIVE\_INFINITY | Represents negative infinity (returned on overflow) |
| NaN | Represents a "Not-a-Number" value |
| POSITIVE\_INFINITY | Represents infinity (returned on overflow) |

# JavaScript Date Formats

## JavaScript Date Input

There are generally 4 types of JavaScript date input formats:

|  |  |
| --- | --- |
| **Type** | **Example** |
| ISO Date | "2015-03-25" (The International Standard) |
| Short Date | "03/25/2015" |
| Long Date | "Mar 25 2015" or "25 Mar 2015" |
| Full Date | "Wednesday March 25 2015" |

The ISO format follows a strict standard in JavaScript.

The other formats are not so well defined and might be browser specific.

## JavaScript ISO Dates

ISO 8601 is the international standard for the representation of dates and times.

The ISO 8601 syntax (YYYY-MM-DD) is also the preferred JavaScript date format:

### Example (Complete date)

var d = new Date("2015-03-25");

The computed date will be relative to your time zone.  
Depending on your time zone, the result above will vary between March 24 and March 25.

## ISO Dates (Year and Month)

ISO dates can be written without specifying the day (YYYY-MM):

### Example

var d = new Date("2015-03");

## ISO Dates (Only Year)

ISO dates can be written without month and day (YYYY):

### Example

var d = new Date("2015");

## JavaScript Short Dates.

Short dates are written with an "MM/DD/YYYY" syntax like this:

### Example

var d = new Date("03/25/2015");

## JavaScript Long Dates.

Long dates are most often written with a "MMM DD YYYY" syntax like this:

### Example

var d = new Date("Mar 25 2015");

## Date Get Methods

Get methods are used for getting a part of a date. Here are the most common (alphabetically):

|  |  |
| --- | --- |
| **Method** | **Description** |
| getDate() | Get the day as a number (1-31) |
| getDay() | Get the weekday as a number (0-6) |
| getFullYear() | Get the four digit year (yyyy) |
| getHours() | Get the hour (0-23) |
| getMilliseconds() | Get the milliseconds (0-999) |
| getMinutes() | Get the minutes (0-59) |
| getMonth() | Get the month (0-11) |
| getSeconds() | Get the seconds (0-59) |
| getTime() | Get the time (milliseconds since January 1, 1970) |

## The getTime() Method

**getTime()** returns the number of milliseconds since January 1, 1970:

### Example

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

## The getFullYear() Method

**getFullYear()** returns the year of a date as a four digit number:

### Example

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

## Displaying Dates

In this tutorial we use a script to display dates inside a <p> element with id="demo":

### Example

<p id="demo"></p>  
  
<script>  
document.getElementById("demo").innerHTML = Date();  
</script>

The script above says: assign the value of Date() to the content (innerHTML) of the element with id="demo".

You will learn how to display a date, in a more readable format, at the bottom of this page.

## Creating Date Objects

The Date object lets us work with dates.

A date consists of a year, a month, a day, an hour, a minute, a second, and milliseconds.

Date objects are created with the **new Date()** constructor.

There are **4 ways** of initiating a date:

new Date()  
new Date(milliseconds)  
new Date(dateString)  
new Date(year, month, day, hours, minutes, seconds, milliseconds)