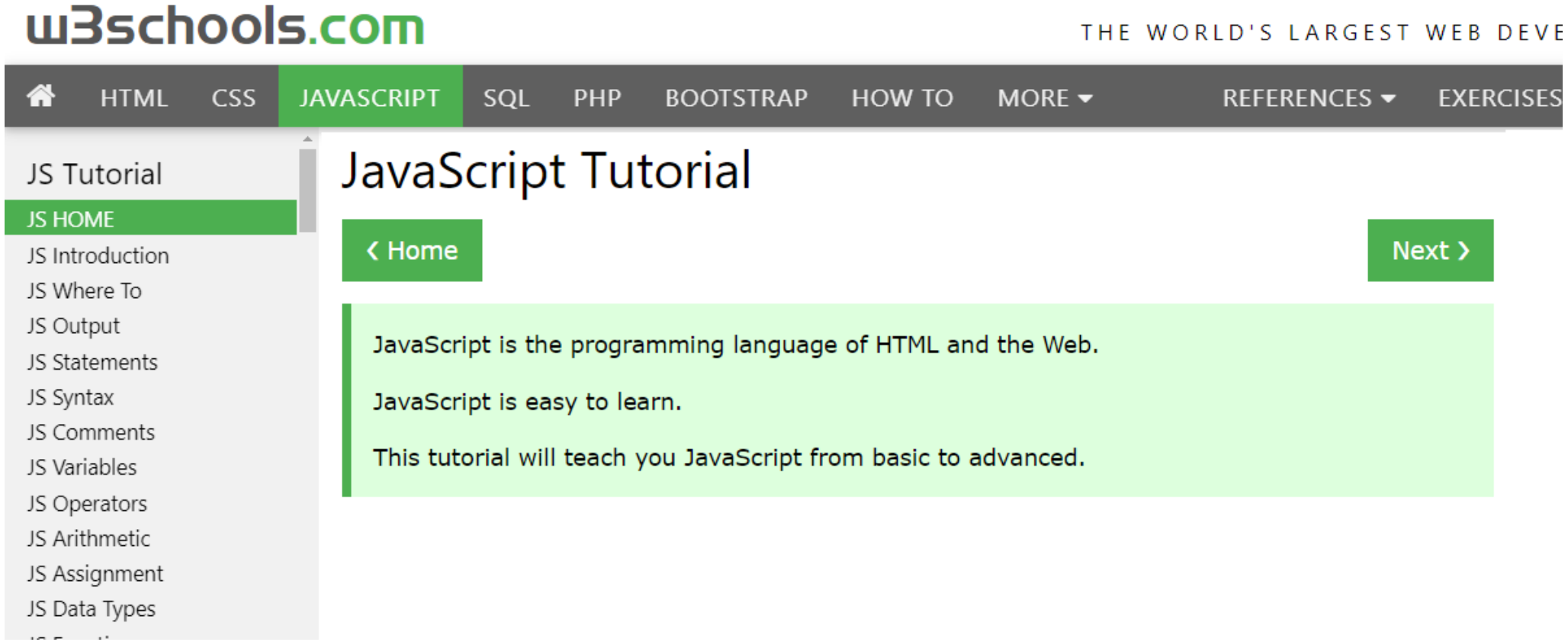
JAVASCRIPT, PART 2

Web Application Development

SOURCE: W3SCHOOLS



AGENDA

- Objects
- Events
- Strings
- String Methods
- Numbers
- Number Methods
- Exercise

OBJECTS

JavaScript

JAVASCRIPT OBJECTS

Real Life Objects, Properties, and Methods

In real life, a car is an **object**.

A car has **properties** like weight and color, and **methods** like start and stop:

Object	Properties	Methods
	car.name = Fiat car.model = 500 car.weight = 850kg car.color = white	car.start() car.drive() car.brake() car.stop()

All cars have the same **properties**, but the property **values** differ from car to car.

All cars have the same **methods**, but the methods are performed **at different times**.

JAVASCRIPT OBJECTS

JavaScript Objects

You have already learned that JavaScript variables are containers for data values.

This code assigns a **simple value** (Fiat) to a **variable** named car:

```
var car = "Fiat";
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_objects_variable

JAVASCRIPT OBJECTS

JavaScript Objects Continued

Objects are variables too. But objects can contain many values.

This code assigns **many values** (Fiat, 500, white) to a **variable** named car:

```
var car = {type:"Fiat", model:"500", color:"white"};
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_objects_object

The values are written as **name:value** pairs (name and value separated by a colon).

JavaScript objects are containers for **named values** called properties or methods.

JAVASCRIPT OBJECTS

Object Definition

You define (and create) a JavaScript object with an object literal:

```
var person = {firstName:"John", lastName:"Doe", age:50,
eyeColor:"blue"}
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_objects_create_1

Spaces and line breaks are not important. An object definition can span multiple lines:

```
var person = {
    firstName:"John",
    lastName:"Doe",
    age:50,
    eyeColor:"blue"
};
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_objects_create_2

JAVASCRIPT OBJECTS

Object Properties

The **name:values** pairs in JavaScript objects are called **properties**:

Property	**Property Value**
firstName	John
lastName	Doe
age	50
eyeColor	blue

JAVASCRIPT OBJECTS

Accessing Object Properties

You can access object properties in two ways:

objectName.propertyName

or

objectName["propertyName"]

Example 1:

```
person.lastName;
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_objects_properties_1

Example 2:

```
person["lastName"];
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_objects_properties_2

JAVASCRIPT OBJECTS

Object Methods

Objects can also have **methods**.

Methods are **actions** that can be performed on objects.

Methods are stored in properties as **function definitions**.

A method is a function stored as a property.

Property	**Property Value**
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

JAVASCRIPT OBJECTS

Object Methods

Example:

```
var person = {
    firstName: "John",
    lastName : "Doe",
    id       : 5566,
    fullName : function() {
        return this.firstName + " " + this.lastName;
    }
};
```

JAVASCRIPT OBJECTS

The **this** Keyword

In a function definition, **this** refers to the "owner" of the function.

In the example above, **this** is the **person object** that "owns" the **fullName** function.

In other words, **this.firstName** means the **firstName** property of **this object**.

Read more about the **this** keyword at JS this Keyword.

JAVASCRIPT OBJECTS

Accessing Object Methods

You access an object method with the following syntax:

objectName.methodName()

Example:

```
name = person.fullName();
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_objects_method

If you access a method **without** the () parentheses, it will return the **function definition**:

```
name = person.fullName;
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_objects_function

JAVASCRIPT OBJECTS

Do Not Declare Strings, Numbers, and Booleans as Objects!

When a JavaScript variable is declared with the keyword "new", the variable is created as an object:

```
var x = new String();        // Declares x as a String object
var y = new Number();        // Declares y as a Number object
var z = new Boolean();       // Declares z as a Boolean object
```

Avoid String, Number, and Boolean objects. They complicate your code and slow down execution speed.

You will learn more about objects later in this tutorial.

JAVASCRIPT OBJECTS

Test Yourself with Exercises!

- Exercise 1
- Exercise 2
- Exercise 3

EVENTS

JavaScript

JAVASCRIPT EVENTS

HTML events are "**things**" that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can "**react**" on these events.

JAVASCRIPT EVENTS

HTML Events

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

JAVASCRIPT EVENTS

HTML Events

With single quotes:

```
<element event='some JavaScript'>
```

With double quotes:

```
<element event="some JavaScript">
```

In the following example, an onclick attribute (with code), is added to a button element:

```
<button onclick="document.getElementById('demo').innerHTML =
Date()">The time is?</button>
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick1

In the example above, the JavaScript code changes the content of the element with id="demo".

JAVASCRIPT EVENTS

HTML Events Continued

In the next example, the code changes the content of its own element (using **this**.innerHTML):

```
<button onclick="this.innerHTML = Date()">The time is?</button>
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick

JavaScript code is often several lines long. It is more common to see event attributes calling functions:

```
<button onclick="displayDate()">The time is?</button>
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_events1

JAVASCRIPT EVENTS

Common HTML Events

Here is a list of some common HTML events:

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

The list is much longer: W3Schools JavaScript Reference HTML DOM Events.

JAVASCRIPT EVENTS

What can JavaScript Do?

Event handlers can be used to handle, and verify, user input, user actions, and browser actions:

- Things that should be done every time a page loads
- Things that should be done when the page is closed
- Action that should be performed when a user clicks a button
- Content that should be verified when a user inputs data
- And more ...

Many different methods can be used to let JavaScript work with events:

- HTML event attributes can execute JavaScript code directly
- HTML event attributes can call JavaScript functions
- You can assign your own event handler functions to HTML elements
- You can prevent events from being sent or being handled
- And more ...
- You will learn a lot more about events and event handlers in the HTML DOM chapters.

You will learn a lot more about events and event handlers in the HTML DOM chapters.

STRINGS

JavaScript

JAVASCRIPT STRINGS

JavaScript Strings

JavaScript strings are used for storing and manipulating text.

A JavaScript string is zero or more characters written inside quotes.

```
var x = "John Doe";
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string

You can use single or double quotes:

```
var carname = "Volvo XC60";   // Double quotes
var carname = 'Volvo XC60';   // Single quotes
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_quotes

JAVASCRIPT STRINGS

JavaScript Strings Continued

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

```
var answer = "It's alright";
var answer = "He is called 'Johnny'";
var answer = 'He is called "Johnny"';
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_quotes_mixed

JAVASCRIPT STRINGS

String Length

The length of a string is found in the built in property **length**:

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_length

JAVASCRIPT STRINGS

Special Characters

Because strings must be written within quotes, JavaScript will misunderstand this string:

```
var x = "We are the so-called "Vikings" from the north.";
```

The string will be chopped to "We are the so-called ".

The solution to avoid this problem, is to use the **backslash escape character**.

The backslash (\) escape character turns special characters into string characters:

Code	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

JAVASCRIPT STRINGS

Special Characters Continued

The sequence \" inserts a double quote in a string:

```
var x = "We are the so-called \"Vikings\" from the north.";
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_escape_quotes2

The sequence \' inserts a single quote in a string:

```
var x = 'It\'s alright.';
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_escape_quotes1

JAVASCRIPT STRINGS

Special Characters Continued

The sequence \\ inserts a backslash in a string:

```
var x = "The character \\ is called backslash.";
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_escape_backslash

Six other escape sequences are valid in JavaScript:

Code	Result
\b	Backspace
\f	Form Feed
\n	New Line
\r	Carriage Return
\t	Horizontal Tabulator
\v	Vertical Tabulator

The 6 escape characters were originally designed to control typewriters, teletypes, and fax machines. They do not make any sense in HTML.

JAVASCRIPT STRINGS

Breaking Long Code Lines

For best readability, programmers often like to avoid code lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it is after an operator:

```
document.getElementById("demo").innerHTML =
"Hello Dolly!";
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_statements_linebreak

JAVASCRIPT STRINGS

Breaking Long Code Lines Continued

You can also break up a code line **within a text string** with a single backslash:

```
document.getElementById("demo").innerHTML = "Hello \
Dolly!";
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_break

The \ method is not the preferred method. It might not have universal support.
Some browsers do not allow spaces behind the \ character.

JAVASCRIPT STRINGS

Breaking Long Code Lines Continued

A safer way to break up a string, is to use string addition:

```
document.getElementById("demo").innerHTML = "Hello " +
"Dolly!";
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_break_ok

You cannot break up a code line with a backslash:

```
document.getElementById("demo").innerHTML = \
"Hello Dolly!";
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_strings_codebreak

JAVASCRIPT STRINGS

Strings Can be Objects

Normally, JavaScript strings are primitive values, created from literals:

var firstName = "John";

But strings can also be defined as objects with the keyword new:

var firstName = new String("John");

```
var x = "John";
var y = new String("John");

// typeof x will return string
// typeof y will return object
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_object

Don't create strings as objects. It slows down execution speed.
The **new** keyword complicates the code. This can produce some unexpected results:

JAVASCRIPT STRINGS

Strings Can be Objects Continued

When using the == operator, equal strings are equal:

```
var x = "John";
var y = new String("John");

// (x == y) is true because x and y have equal values
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_object1

JAVASCRIPT STRINGS

Strings Can be Objects Continued

When using the === operator, equal strings are not equal, because the === operator expects equality in both type and value.

```
var x = "John";
var y = new String("John");

// (x === y) is false because x and y have different types (string
and object)
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_object2

JAVASCRIPT STRINGS

Strings Can be Objects Continued

Or even worse. Objects cannot be compared:

```
var x = new String("John");
var y = new String("John");

// (x == y) is false because x and y are different objects
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_object3

```
var x = new String("John");
var y = new String("John");

// (x === y) is false because x and y are different objects
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_object4

Note the difference between (x==y) and (x===y).
Comparing two JavaScript objects will **always** return false.

JAVASCRIPT STRINGS

Test Yourself with Exercises!

- Exercise 1
- Exercise 2
- Exercise 3

STRING METHODS

JavaScript

JAVASCRIPT STRING METHODS

String Methods and Properties

String methods help you to work with strings.

Primitive values, like "John Doe", cannot have properties or methods (because they are not objects).

But with JavaScript, methods and properties are also available to primitive values, because JavaScript treats primitive values as objects when executing methods and properties.

JAVASCRIPT STRING METHODS

String Length

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

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_length

JAVASCRIPT STRING METHODS

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:

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_indexof

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

JAVASCRIPT STRING METHODS

Finding a String in a String Continued

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

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

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_lastindexof

Both indexOf(), and lastIndexOf() return -1 if the text is not found.

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_indexof_1

JAVASCRIPT STRING METHODS

Finding a String in a String Continued

Both methods accept a second parameter as the starting position for the search:

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate",15);
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_indexof_2

JAVASCRIPT STRING METHODS

Searching for a String in a String

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

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

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_search_locate

JAVASCRIPT STRING METHODS

Did You Notice?

The two methods, indexOf() and search(), are **equal?**

They accept the same arguments (parameters), and return the same value?

The two methods are **NOT** equal. These are the differences:

- The search() method cannot take a second start position argument.
- The indexOf() method cannot take powerful search values (regular expressions).

You will learn more about regular expressions in a later chapter.

JAVASCRIPT STRING METHODS

Extracting String Parts

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

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

JAVASCRIPT STRING METHODS

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:

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

The result of res will be:

```
Banana
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_slice

JAVASCRIPT STRING METHODS

The slice() Method Continued

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:

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

The result of res will be:

```
Banana
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_slice_negative

JAVASCRIPT STRING METHODS

The slice() Method Continued

If you omit the second parameter, the method will slice out the rest of the string:

```
var res = str.slice(7);
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_slice_rest

or, counting from the end:

```
var res = str.slice(-12);
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_slice_rest_negative

Negative positions do not work in Internet Explorer 8 and earlier.

JAVASCRIPT STRING METHODS

The substring() Method

substring() is similar to slice().

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

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

The result of *res* will be:

```
Banana
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_substring

If you omit the second parameter, substring() will slice out the rest of the string.

JAVASCRIPT STRING METHODS

The substr() Method

substr() is similar to slice().

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

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(7, 6);
```

The result of res will be:

```
Banana
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_substr

JAVASCRIPT STRING METHODS

The substr() Method Continued

If you omit the second parameter, substr() will slice out the rest of the string.

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(7);
```

The result of res will be:

```
Banana, Kiwi
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_substr1

JAVASCRIPT STRING METHODS

The substr() Method Continued

If the first parameter is negative, the position counts from the end of the string.

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(-4);
```

The result of res will be:

```
Kiwi
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_substr2

JAVASCRIPT STRING METHODS

Replacing String Content

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

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_replace

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

JAVASCRIPT STRING METHODS

Replacing String Content Continued

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

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

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_replace_first

By default, the replace() function is case sensitive. Writing MICROSOFT (with upper-case) will not work:

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

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_replace_case

JAVASCRIPT STRING METHODS

Replacing String Content Continued

To replace case insensitive, use a **regular expression** with an **/i** flag (insensitive):

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

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_replace_insensitive

Note that regular expressions are written without quotes.

JAVASCRIPT STRING METHODS

Replacing String Content Continued

To replace all matches, use a **regular expression** with a **/g** flag (global match):

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

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_replace_global

You will learn a lot more about regular expressions in the chapter JavaScript Regular Expressions.

JAVASCRIPT STRING METHODS

Converting to Upper and Lower Case

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

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_toupper

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

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_tolower

JAVASCRIPT STRING METHODS

The concat() Method

concat() joins two or more strings:

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_concat

JAVASCRIPT STRING METHODS

The concat() Method Continued

The **concat()** method can be used instead of the plus operator. These two lines do the same:

```
var text = "Hello" + " " + "World!";
var text = "Hello".concat(" ", "World!");
```

All string methods return a new string. They don't modify the original string.
Formally said: Strings are immutable: Strings cannot be changed, only replaced.

JAVASCRIPT STRING METHODS

String.trim()

String.trim() removes whitespace from both sides of a string.

```
var str = "       Hello World!        ";
alert(str.trim());
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_trim

String.trim() is not supported in Internet Explorer 8 or lower.

If you need to support IE 8, you can use String.replace with a regular expression instead:

```
var str = "       Hello World!        ";
alert(str.replace(/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g, ''));
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_trim_regexp

JAVASCRIPT STRING METHODS

String.trim() Continued

You can also use the replace solution above to add a trim function to the JavaScript String.prototype:

```
if (!String.prototype.trim) {
    String.prototype.trim = function () {
    return this.replace(/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g, '');
};
var str = "       Hello World!        ";
alert(str.trim());
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_trim_polyfill

JAVASCRIPT STRING METHODS

Extracting String Characters

There are 3 methods for extracting string characters:

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

JAVASCRIPT STRING METHODS

The charAt() Method

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

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_charat

JAVASCRIPT STRING METHODS

The charCodeAt() Method

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).

```
var str = "HELLO WORLD";

str.charCodeAt(0);          // returns 72
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_charcodeat

JAVASCRIPT STRING METHODS

Property Access

ECMAScript 5 (2009) allows property access [ ] on strings:

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_prop

Property access might be a little **unpredictable:**

- It does not work in Internet Explorer 7 or earlier
- It makes strings look like arrays (but they are not)
- If no character is found, [ ] returns undefined, while charAt() returns an empty string.
- It is read only. str[0] = "A" gives no error (but does not work!)

JAVASCRIPT STRING METHODS

Property Access Continued

```
var str = "HELLO WORLD";
str[0] = "A";                // Gives no error, but does not work
str[0];                      // returns H
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_prop2

If you want to work with a string as an array, you can convert it to an array.

JAVASCRIPT STRING METHODS

Converting a String to an Array

A string can be converted to an array with the **split()** method:

```
var txt = "a,b,c,d,e";    // String
txt.split(",");           // Split on commas
txt.split(" ");           // Split on spaces
txt.split("|");           // Split on pipe
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_split

If the separator is omitted, the returned array will contain the whole string in index [0].

JAVASCRIPT STRING METHODS

Converting a String to an Array Continued

If the separator is "", the returned array will be an array of single characters:

```
var txt = "Hello";          // String
txt.split("");              // Split in characters
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_split_char

JAVASCRIPT STRING METHODS

Complete String Reference

For a complete reference, go to our Complete JavaScript String Reference.

The reference contains descriptions and examples of all string properties and methods.

JAVASCRIPT STRING METHODS

Test Yourself with Exercises!

- Exercise 4
- Exercise 5
- Exercise 6
- Exercise 7
- Exercise 8
- Exercise 9

NUMBERS

JavaScript

JAVASCRIPT NUMBERS

JavaScript Numbers

JavaScript has only one type of number. Numbers can be written with or without decimals.

```
var x = 3.14;     // A number with decimals
var y = 3;        // A number without decimals
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers1

Extra large or extra small numbers can be written with scientific (exponent) notation:

```
var x = 123e5;     // 12300000
var y = 123e-5;    // 0.00123
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers2

JAVASCRIPT NUMBERS

JavaScript Numbers are Always 64-bit Floating Point

Unlike many other programming languages, JavaScript does not define different types of numbers, like integers, short, long, floating-point etc.

JavaScript numbers are always stored as double precision floating point numbers, following the international IEEE 754 standard.

This format stores numbers in 64 bits, where the number (the fraction) is stored in bits 0 to 51, the exponent in bits 52 to 62, and the sign in bit 63:

Value (aka Fraction/Mantissa)	**Exponent**	**Sign**
52 bits (0 - 51)	11 bits (52 - 62)	1 bit (63)

JAVASCRIPT NUMBERS

Precision

Integers (numbers without a period or exponent notation) are accurate up to 15 digits.

```
var x = 999999999999999;   // x will be 999999999999999
var y = 9999999999999999;  // y will be 10000000000000000
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_inaccurate1

The maximum number of decimals is 17, but floating point arithmetic is not always 100% accurate:

```
var x = 0.2 + 0.1;         // x will be 0.30000000000000004
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_inaccurate2

JAVASCRIPT NUMBERS

Precision Continued

To solve the problem above, it helps to multiply and divide:

```
var x = (0.2 * 10 + 0.1 * 10) / 10;       // x will be 0.3
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_inaccurate3

JAVASCRIPT NUMBERS

Adding Numbers and Strings

WARNING !!

JavaScript uses the + operator for both addition and concatenation.

Numbers are added. Strings are concatenated.

If you add two numbers, the result will be a number:

```
var x = 10;
var y = 20;
var z = x + y;              // z will be 30 (a number)
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_add

JAVASCRIPT NUMBERS

Adding Numbers and Strings Continued

If you add two strings, the result will be a string concatenation:

```
var x = "10";
var y = "20";
var z = x + y;              // z will be 1020 (a string)
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_add_strings1

If you add a number and a string, the result will be a string concatenation:

```
var x = 10;
var y = "20";
var z = x + y;              // z will be 1020 (a string)
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_add_strings2

JAVASCRIPT NUMBERS

Adding Numbers and Strings Continued

If you add a string and a number, the result will be a string concatenation:

```
var x = "10";
var y = 20;
var z = x + y;              // z will be 1020 (a string)
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_add_strings5

A common mistake is to expect this result to be 30:

```
var x = 10;
var y = 20;
var z = "The result is: " + x + y;
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_add_strings3

JAVASCRIPT NUMBERS

Adding Numbers and Strings Continued

A common mistake is to expect this result to be 102030:

```
var x = 10;
var y = 20;
var z = "30";
var result = x + y + z;
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_add_strings4

The JavaScript compiler works from left to right.
First 10 + 20 is added because x and y are both numbers.
Then 30 + "30" is concatenated because z is a string.

JAVASCRIPT NUMBERS

Numeric Strings

JavaScript strings can have numeric content:

```
var x = 100;          // x is a number

var y = "100";        // y is a string
```

JavaScript will try to convert strings to numbers in all numeric operations:

This will work:

```
var x = "100";
var y = "10";
var z = x / y;        // z will be 10
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_string1

JAVASCRIPT NUMBERS

Numeric Strings Continued

This will also work:

```
var x = "100";
var y = "10";
var z = x * y;          // z will be 1000
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_string2

And this will work:

```
var x = "100";
var y = "10";
var z = x - y;          // z will be 90
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_string3

JAVASCRIPT NUMBERS

Numeric Strings Continued

But this will not work:

```
var x = "100";
var y = "10";
var z = x + y;      // z will not be 110 (It will be 10010)
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_string4

In the last example JavaScript uses the + operator to concatenate the strings.

JAVASCRIPT NUMBERS

NaN - Not a Number

NaN is a JavaScript reserved word indicating that a number is not a legal number.

Trying to do arithmetic with a non-numeric string will result in NaN (Not a Number):

```
var x = 100 / "Apple";  // x will be NaN (Not a Number)
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_divide_string

However, if the string contains a numeric value , the result will be a number:

```
var x = 100 / "10";     // x will be 10
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_divide_number

JAVASCRIPT NUMBERS

NaN - Not a Number Continued

You can use the global JavaScript function isNaN() to find out if a value is a number:

```
var x = 100 / "Apple";
isNaN(x);                    // returns true because x is Not a Number
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_isnan_true

Watch out for NaN. If you use NaN in a mathematical operation, the result will also be NaN:

```
var x = NaN;
var y = 5;
var z = x + y;              // z will be NaN
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_nan_math

JAVASCRIPT NUMBERS

NaN - Not a Number Continued

Or the result might be a concatenation:

```
var x = NaN;
var y = "5";
var z = x + y;              // z will be NaN5
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_nan_concat

NaN is a number: typeof NaN returns number:

```
typeof NaN;                 // returns "number”
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_nan_typeof

JAVASCRIPT NUMBERS

Infinity

Infinity (or -Infinity) is the value JavaScript will return if you calculate a number outside the largest possible number.

```
var myNumber = 2;
while (myNumber != Infinity) {           // Execute until Infinity
    myNumber = myNumber * myNumber;
}
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_infinity

JAVASCRIPT NUMBERS

Infinity Continued

Division by 0 (zero) also generates Infinity:

```
var x =  2 / 0;          // x will be Infinity
var y = -2 / 0;          // y will be -Infinity
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_infinity_zero

Infinity is a number: typeof Infinity returns number.

```
typeof Infinity;        // returns "number"
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_infinity_typeof

JAVASCRIPT NUMBERS

Hexadecimal

JavaScript interprets numeric constants as hexadecimal if they are preceded by 0x.

```
var x = 0xFF;             // x will be 255
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_hex

Never write a number with a leading zero (like 07).
Some JavaScript versions interpret numbers as octal if they are written with a leading zero.

JAVASCRIPT NUMBERS

Hexadecimal Continued

By default, JavaScript displays numbers as **base 10** decimals.

But you can use the toString() method to output numbers from **base 2** to **base 36.**

Hexadecimal is **base 16.** Decimal is **base 10.** Octal is **base 8.** Binary is **base 2.**

```
var myNumber = 32;
myNumber.toString(10);  // returns 32
myNumber.toString(32);  // returns 10
myNumber.toString(16);  // returns 20
myNumber.toString(8);   // returns 40
myNumber.toString(2);   // returns 100000
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_tostring

JAVASCRIPT NUMBERS

Numbers Can be Objects

Normally JavaScript numbers are primitive values created from literals:

var x = 123;

But numbers can also be defined as objects with the keyword new:

var y = new Number(123);

```
var x = 123;
var y = new Number(123);

// typeof x returns number
// typeof y returns object
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_object

Do not create Number objects. It slows down execution speed.
The **new** keyword complicates the code. This can produce some unexpected results:

JAVASCRIPT NUMBERS

Numbers Can be Objects Continued

When using the == operator, equal numbers are equal:

```
var x = 500;
var y = new Number(500);

// (x == y) is true because x and y have equal values
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_object1

JAVASCRIPT NUMBERS

Numbers Can be Objects Continued

When using the === operator, equal numbers are not equal, because the === operator expects equality in both type and value.

```
var x = 500;
var y = new Number(500);

// (x === y) is false because x and y have different types
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_object2

JAVASCRIPT NUMBERS

Numbers Can be Objects Continued

Or even worse. Objects cannot be compared:

```
var x = new Number(500);
var y = new Number(500);

// (x == y) is false because objects cannot be compared
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_object3

Note the difference between (x==y) and (x===y).
Comparing two JavaScript objects will always return false.

NUMBER METHODS

JavaScript

JAVASCRIPT NUMBER METHODS

Number Methods and Properties

Number methods help you work with numbers.

Primitive values (like 3.14 or 2014), cannot have properties and methods (because they are not objects).

But with JavaScript, methods and properties are also available to primitive values, because JavaScript treats primitive values as objects when executing methods and properties.

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):

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

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_tostring

JAVASCRIPT NUMBER METHODS

The toExponential() Method

toExponential() returns a string, with a number rounded and written using exponential notation.

A parameter defines the number of characters behind the decimal point:

```
var x = 9.656;
x.toExponential(2);     // returns 9.66e+0
x.toExponential(4);     // returns 9.6560e+0
x.toExponential(6);     // returns 9.656000e+0
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_toexponential

The parameter is optional. If you don't specify it, JavaScript will not round the number.

JAVASCRIPT NUMBER METHODS

The toFixed() Method

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

```
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
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_tofixed

toFixed(2) is perfect for working with money.

JAVASCRIPT NUMBER METHODS

The toPrecision() Method

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

```
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
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_toprecision

JAVASCRIPT NUMBER METHODS

The valueOf() Method

valueOf() returns a number as a number.

```
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
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_valueof

In JavaScript, a number can be a primitive value (typeof = number) or an object (typeof = object).

The valueOf() method is used internally in JavaScript to convert Number objects to primitive values.

There is no reason to use it in your code.

All JavaScript data types have a valueOf() and a toString() method.

JAVASCRIPT NUMBER METHODS

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

These methods are not **number** methods, but **global** JavaScript methods.

JAVASCRIPT NUMBER METHODS

Global JavaScript 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

JAVASCRIPT NUMBER METHODS

The Number() Method

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

```
Number(true);            // returns 1
Number(false);           // returns 0
Number("10");            // returns 10
Number("  10");          // returns 10
Number("10  ");          // returns 10
Number(" 10  ");         // returns 10
Number("10.33");         // returns 10.33
Number("10,33");         // returns NaN
Number("10 33");         // returns NaN
Number("John");          // returns NaN
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_global_number

If the number cannot be converted, NaN (Not a Number) is returned.

JAVASCRIPT NUMBER METHODS

The Number() Method Used on Dates

Number() can also convert a date to a number:

```
Number(new Date("2017-09-30"));     // returns 1506729600000
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_global_number_date

The Number() method above returns the number of milliseconds since 1.1.1970.

JAVASCRIPT NUMBER METHODS

The parseInt() Method

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

```
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
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_global_parseint

If the number cannot be converted, NaN (Not a Number) is returned.

JAVASCRIPT NUMBER METHODS

The parseFloat() Method

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

```
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
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_global_parsefloat

If the number cannot be converted, NaN (Not a Number) is returned.

JAVASCRIPT NUMBER METHODS

Number Properties

Property	Description
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
POSITIVE_INFINITY	Represents infinity (returned on overflow)
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
NaN	Represents a "Not-a-Number" value

JAVASCRIPT NUMBER METHODS

JavaScript MIN_VALUE and MAX_VALUE

```
var x = Number.MAX_VALUE;
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_max

```
var x = Number.MIN_VALUE;
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_min

JAVASCRIPT NUMBER METHODS

JavaScript POSITIVE_INFINITY

```
var x = Number.POSITIVE_INFINITY;
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_pos_infinity

POSITIVE_INFINITY is returned on overflow:

```
var x = 1 / 0;
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_pos_infinity2

JAVASCRIPT NUMBER METHODS

JavaScript NEGATIVE_INFINITY

```
var x = Number.NEGATIVE_INFINITY;
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_neg_infinity

NEGATIVE_INFINITY is returned on overflow:

```
var x = -1 / 0;
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_neg_infinity2

JAVASCRIPT NUMBER METHODS

JavaScript NaN - Not a Number

```
var x = Number.NaN;
```

Try it Yourself: https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_nan

NaN is a JavaScript reserved word indicating that a number is not a legal number.

Trying to do arithmetic with a non-numeric string will result in NaN (Not a Number):

```
var x = 100 / "Apple";  // x will be NaN (Not a Number)
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_numbers_divide_string

JAVASCRIPT NUMBER METHODS

Number Properties Cannot be Used on Variables

Number properties belongs to the JavaScript's number object wrapper called **Number**.

These properties can only be accessed as **Number**.MAX_VALUE.

Using *myNumber*.MAX_VALUE, where *myNumber* is a variable, expression, or value, will return undefined:

```
var x = 6;
var y = x.MAX_VALUE;     // y becomes undefined
```

Try it Yourself:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_max_undefined

JAVASCRIPT NUMBER METHODS

Complete JavaScript Number Reference

For a complete reference, go to our Complete JavaScript Number Reference.

The reference contains descriptions and examples of all Number properties and methods.

EXERCISE

JavaScript

EXERCISE

- Declare an array of objects.
- Create a form which allows a user to create elements of the array, **person** objects, including name, email address and phone number.
- Create 5 **person** objects using the form. Give each person object values for name, email address and phone number.
- As they add objects, the code should generate HTML to display the names in an unordered list. Each list item, <li>, should be clickable, i.e. should have an "onclick" attribute set calling the function, myFunction.
- Create a function, myFunction, which displays an alert box containing the name, email and phone number of a clicked <li> element.

THE END

JavaScript