Placing scripts at the bottom of the <body> element improves the display speed, because script interpretation slows down the display.

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension **.js**.

<script src="myScript.js"></script>

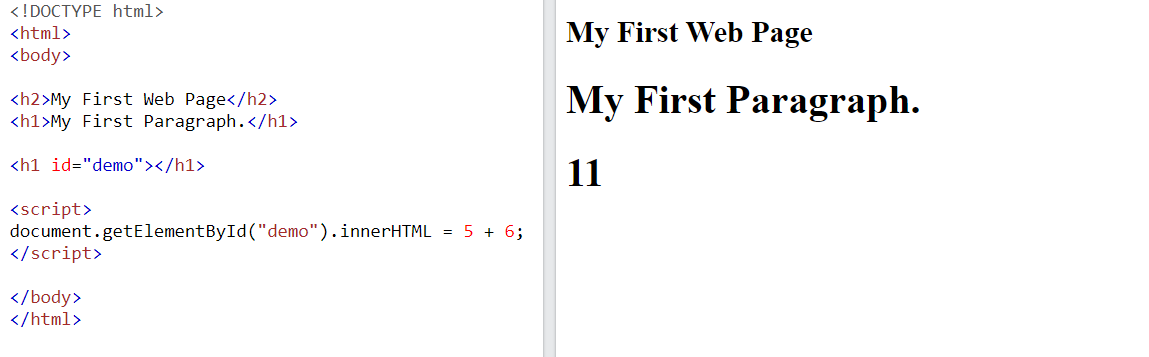
JavaScript Display Possibilities

JavaScript can "display" data in different ways:

* Writing into an HTML element, using innerHTML.
* Writing into the HTML output using document.write().
* Writing into an alert box, using window.alert().
* Writing into the browser console, using console.log().

Using innerHTML

* To access an HTML element, JavaScript can use the document.getElementById(id) method.
* The id attribute defines the HTML element. The innerHTML property defines the HTML content:



Using document.write()

Using document.write() after an HTML document is loaded, will **delete all existing HTML**

In JavaScript, the window object is the global scope object. This means that variables, properties, and methods by default belong to the window object. This also means that specifying the window keyword is optional

Using console.log()

For debugging purposes, you can call the console.log() method in the browser to display data.

JavaScript Programs

In HTML, JavaScript programs are executed by the web browser.

# **JavaScript Variables**

### **4 Ways to Declare a JavaScript Variable:**

* Using var
* Using let
* Using const
* Using nothing

## When to Use JavaScript var?

Always declare JavaScript variables with var,let, orconst.

The var keyword is used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

If you want your code to run in older browsers, you must use var.

It's a good programming practice to declare all variables at the beginning of a script.

## Value = undefined

A variable declared without a value will have the value undefined.

## Re-Declaring JavaScript Variables

If you re-declare a JavaScript variable declared with var, it will not lose its value.

The variable carName will still have the value "Volvo" after the execution of these statements:

var carName = "Volvo";  
var carName;

## Note

You cannot re-declare a variable declared with let or const.

This will not work:

let carName = "Volvo";  
let carName;

## JavaScript Dollar Sign $

Since JavaScript treats a dollar sign as a letter, identifiers containing $ are valid variable names

Using the dollar sign is not very common in JavaScript, but professional programmers often use it as an alias for the main function in a JavaScript library.

In the JavaScript library jQuery, for instance, the main function $ is used to select HTML elements. In jQuery $("p"); means "select all p elements".

## JavaScript Underscore (\_)

Since JavaScript treats underscore as a letter, identifiers containing \_ are valid variable names

Using the underscore is not very common in JavaScript, but a convention among professional programmers is to use it as an alias for "private (hidden)" variables

# **JavaScript Let**

The let keyword was introduced in [ES6 (2015)](https://www.w3schools.com/js/js_es6.asp).

Variables defined with let cannot be Redeclared.

Variables defined with let must be Declared before use.

## Let Hoisting

Variables defined with var are **hoisted** to the top and can be initialized at any time.

Meaning: You can use the variable before it is declared:

### **Example**

This is OK:

carName = "Volvo";  
var carName;

Variables defined with let have Block Scope.

Variables declared with the var keyword do NOT have block scope.

{  
  var x = 2;  
}  
// x CAN be used here

# **JavaScript Const**

[❮ Previous](https://www.w3schools.com/js/js_let.asp)[Next ❯](https://www.w3schools.com/js/js_operators.asp)

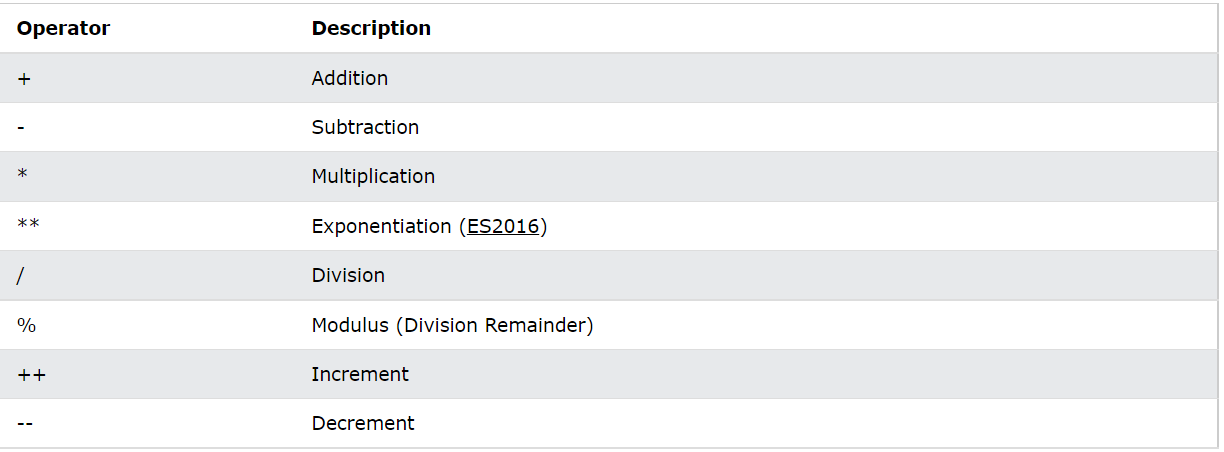
The const keyword was introduced in [ES6 (2015)](https://www.w3schools.com/js/js_es6.asp).

Variables defined with const cannot be Redeclared.

Variables defined with const cannot be Reassigned.

Variables defined with const have Block Scope.

## JavaScript Arithmetic Operators



## JavaScript Assignment Operators

## JavaScript Comparison Operators

## JavaScript Logical Operators

## JavaScript Type Operators

# **JavaScript Data Types**

JavaScript variables can hold different data types: numbers, strings, objects and more:

let length = 16;                               // Number  
let lastName = "Johnson";                      // String  
let x = {firstName:"John", lastName:"Doe"};    // Object

## JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types

## JavaScript Strings

You can use single or double quotes

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

## JavaScript Numbers

JavaScript has only one type of numbers.

Numbers can be written with, or without decimals

## JavaScript Arrays

JavaScript arrays are written with square brackets.

Array items are separated by commas.

const cars = ["Saab", "Volvo", "BMW"];

## JavaScript Objects

JavaScript objects are written with curly braces {}.

Object properties are written as name:value pairs, separated by commas.

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

## The typeof Operator

You can use the JavaScript typeof operator to find the type of a JavaScript variable.

# **JavaScript Objects**

It is a common practice to declare objects with the const keyword.

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

## Object Definition

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

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

## Accessing Object Properties

You can access object properties in two ways:

*objectName.propertyName*

or

*objectName["propertyName"]*

## Object Methods

Objects can also have **methods**.

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

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

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

You access an object method with the following syntax:

*objectName.methodName()*

## 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:

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

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

# **JavaScript Events**

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

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

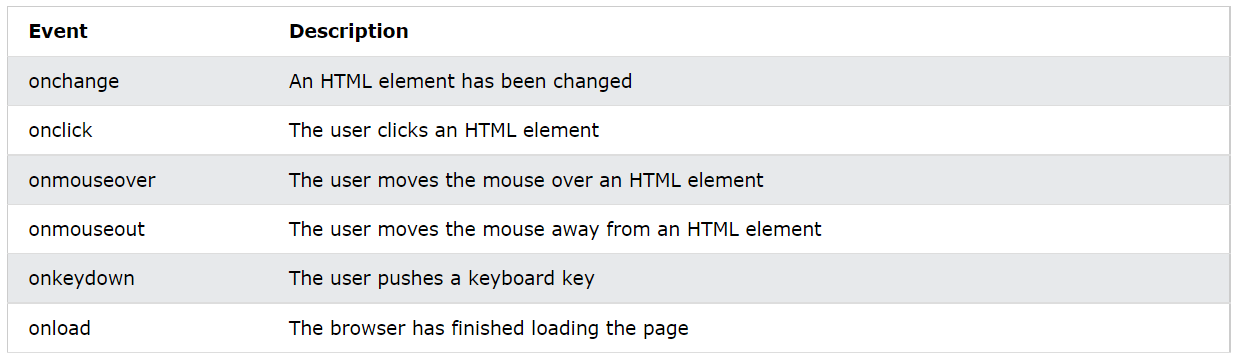
An HTML event can be something the browser does, or something a user does.

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

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

## Common HTML Events

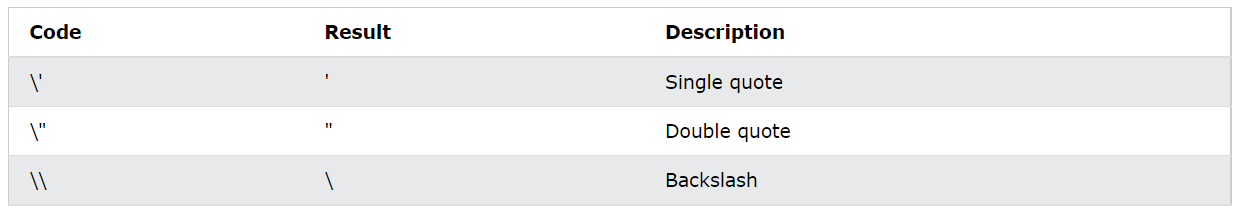
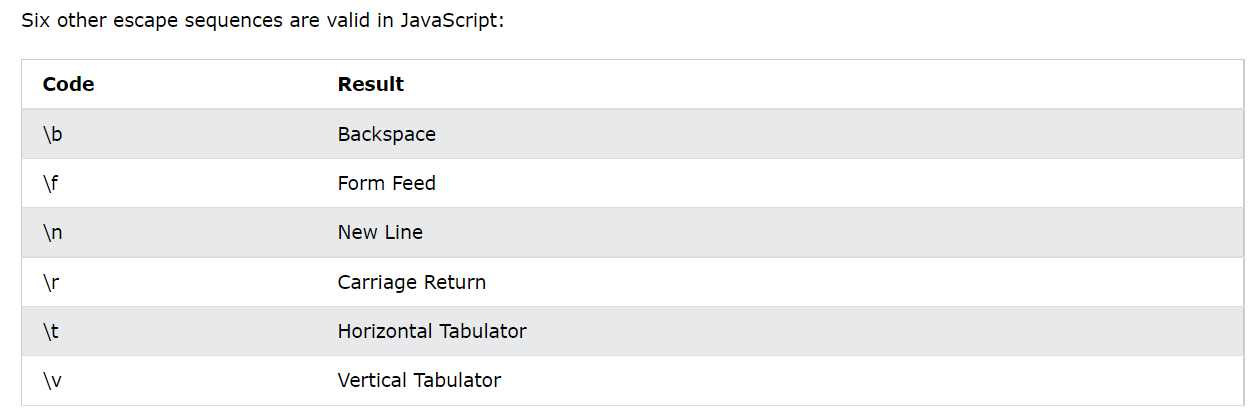


# **JavaScript Strings**

## String Length

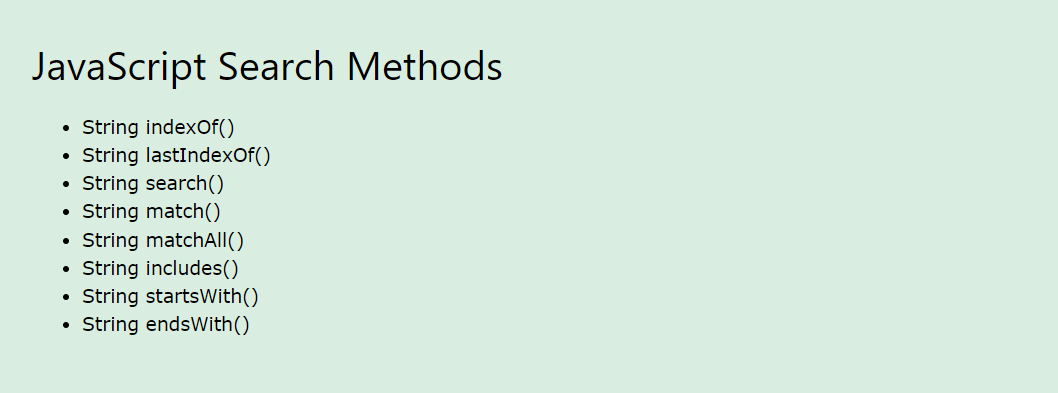
let length = text.length;

## Escape Character

# **JavaScript String Methods**

# **JavaScript String Search**



# **JavaScript Template Literals**

## Interpolation

**Template literals** provide an easy way to interpolate variables and expressions into strings.

The method is called string interpolation.

The syntax is:

${...}

let text = `Welcome ${firstName}, ${lastName}!`;

## Infinity

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

Division by 0 (zero) also generates Infinity:

# **JavaScript BigInt**

JavaScript BigInt variables are used to store big integer values that are too big to be represented by a a normal JavaScript Number.

## How to Create a BigInt

To create a BigInt, append n to the end of an integer or call BigInt():

let x = 1234567890123456789012345n;  
let y = BigInt(1234567890123456789012345)

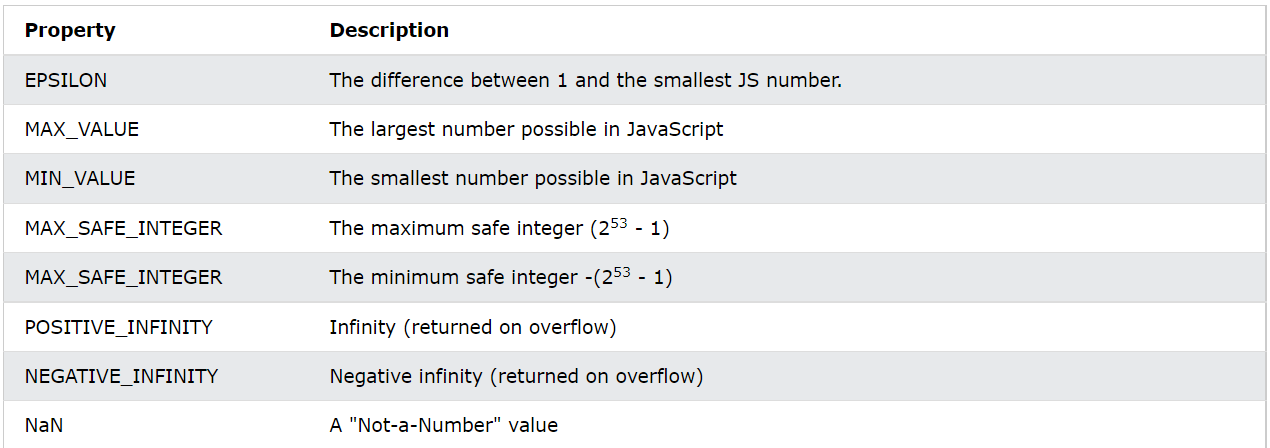
Arithmetic between a BigInt and a Number is not allowed (type conversion lose information).

## JavaScript Number Methods

# 

# **JavaScript Arrays**

# **JavaScript Number Properties**



# **JavaScript Arrays**

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

const cars = [];  
cars[0]= "Saab";  
cars[1]= "Volvo";  
cars[2]= "BMW";

## Arrays are Objects

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

## Array Elements Can Be Objects

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

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
  
let text = "<ul>";  
fruits.forEach(myFunction);  
text += "</ul>";  
  
function myFunction(value) {  
  text += "<li>" + value + "</li>";  
}

## Adding Array Elements

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

const fruits = ["Banana", "Orange", "Apple"];  
fruits.push("Lemon");  // Adds a new element (Lemon) to fruits

## How to Recognize an Array

[JS Dates](https://www.w3schools.com/js/js_dates.asp)[JS Date Formats](https://www.w3schools.com/js/js_date_formats.asp)[JS Date Get Methods](https://www.w3schools.com/js/js_date_methods.asp)[JS Date Set Methods](https://www.w3schools.com/js/js_date_methods_set.asp)[JS Math](https://www.w3schools.com/js/js_math.asp)[JS Random](https://www.w3schools.com/js/js_random.asp)[JS Booleans](https://www.w3schools.com/js/js_booleans.asp)[JS Comparisons](https://www.w3schools.com/js/js_comparisons.asp)[JS If Else](https://www.w3schools.com/js/js_if_else.asp)[JS Switch](https://www.w3schools.com/js/js_switch.asp)[JS Loop For](https://www.w3schools.com/js/js_loop_for.asp)[JS Loop For In](https://www.w3schools.com/js/js_loop_forin.asp)[JS Loop For Of](https://www.w3schools.com/js/js_loop_forof.asp)[JS Loop While](https://www.w3schools.com/js/js_loop_while.asp)[JS Break](https://www.w3schools.com/js/js_break.asp)[JS Iterables](https://www.w3schools.com/js/js_iterables.asp)[JS Sets](https://www.w3schools.com/js/js_sets.asp)[JS Maps](https://www.w3schools.com/js/js_maps.asp)[JS Typeof](https://www.w3schools.com/js/js_typeof.asp)[JS Type Conversion](https://www.w3schools.com/js/js_type_conversion.asp)[JS Bitwise](https://www.w3schools.com/js/js_bitwise.asp)[JS RegExp](https://www.w3schools.com/js/js_regexp.asp)[JS Precedence](https://www.w3schools.com/js/js_precedence.asp)[JS Errors](https://www.w3schools.com/js/js_errors.asp)[JS Scope](https://www.w3schools.com/js/js_scope.asp)[JS Hoisting](https://www.w3schools.com/js/js_hoisting.asp)[JS Strict Mode](https://www.w3schools.com/js/js_strict.asp)[JS this Keyword](https://www.w3schools.com/js/js_this.asp)[JS Arrow Function](https://www.w3schools.com/js/js_arrow_function.asp)[JS Classes](https://www.w3schools.com/js/js_classes.asp)[JS Modules](https://www.w3schools.com/js/js_modules.asp)[JS JSON](https://www.w3schools.com/js/js_json.asp)[JS Debugging](https://www.w3schools.com/js/js_debugging.asp)[JS Style Guide](https://www.w3schools.com/js/js_conventions.asp)[JS Best Practices](https://www.w3schools.com/js/js_best_practices.asp)[JS Mistakes](https://www.w3schools.com/js/js_mistakes.asp)[JS Performance](https://www.w3schools.com/js/js_performance.asp)[JS Reserved Words](https://www.w3schools.com/js/js_reserved.asp)

## JS Versions

[JS Versions](https://www.w3schools.com/js/js_versions.asp)[JS 2009 (ES5)](https://www.w3schools.com/js/js_es5.asp)[JS 2015 (ES6)](https://www.w3schools.com/js/js_es6.asp)[JS 2016](https://www.w3schools.com/js/js_2016.asp)[JS 2017](https://www.w3schools.com/js/js_2017.asp)[JS 2018](https://www.w3schools.com/js/js_2018.asp)[JS 2019](https://www.w3schools.com/js/js_2019.asp)[JS 2020](https://www.w3schools.com/js/js_2020.asp)[JS 2021](https://www.w3schools.com/js/js_2021.asp)[JS IE / Edge](https://www.w3schools.com/js/js_ie_edge.asp)[JS History](https://www.w3schools.com/js/js_history.asp)

## JS Objects

[Object Definitions](https://www.w3schools.com/js/js_object_definition.asp)[Object Properties](https://www.w3schools.com/js/js_object_properties.asp)[Object Methods](https://www.w3schools.com/js/js_object_methods.asp)[Object Display](https://www.w3schools.com/js/js_object_display.asp)[Object Accessors](https://www.w3schools.com/js/js_object_accessors.asp)[Object Constructors](https://www.w3schools.com/js/js_object_constructors.asp)[Object Prototypes](https://www.w3schools.com/js/js_object_prototypes.asp)[Object Iterables](https://www.w3schools.com/js/js_object_iterables.asp)[Object Sets](https://www.w3schools.com/js/js_object_sets.asp)[Object Maps](https://www.w3schools.com/js/js_object_maps.asp)[Object Reference](https://www.w3schools.com/js/js_object_es5.asp)

## JS Functions

[Function Definitions](https://www.w3schools.com/js/js_function_definition.asp)[Function Parameters](https://www.w3schools.com/js/js_function_parameters.asp)[Function Invocation](https://www.w3schools.com/js/js_function_invocation.asp)[Function Call](https://www.w3schools.com/js/js_function_call.asp)[Function Apply](https://www.w3schools.com/js/js_function_apply.asp)[Function Bind](https://www.w3schools.com/js/js_function_bind.asp)[Function Closures](https://www.w3schools.com/js/js_function_closures.asp)

## JS Classes

[Class Intro](https://www.w3schools.com/js/js_class_intro.asp)[Class Inheritance](https://www.w3schools.com/js/js_class_inheritance.asp)[Class Static](https://www.w3schools.com/js/js_class_static.asp)

## JS Async

[JS Callbacks](https://www.w3schools.com/js/js_callback.asp)[JS Asynchronous](https://www.w3schools.com/js/js_asynchronous.asp)[JS Promises](https://www.w3schools.com/js/js_promise.asp)[JS Async/Await](https://www.w3schools.com/js/js_async.asp)

## JS HTML DOM

[DOM Intro](https://www.w3schools.com/js/js_htmldom.asp)[DOM Methods](https://www.w3schools.com/js/js_htmldom_methods.asp)[DOM Document](https://www.w3schools.com/js/js_htmldom_document.asp)[DOM Elements](https://www.w3schools.com/js/js_htmldom_elements.asp)[DOM HTML](https://www.w3schools.com/js/js_htmldom_html.asp)[DOM Forms](https://www.w3schools.com/js/js_validation.asp)[DOM CSS](https://www.w3schools.com/js/js_htmldom_css.asp)[DOM Animations](https://www.w3schools.com/js/js_htmldom_animate.asp)[DOM Events](https://www.w3schools.com/js/js_htmldom_events.asp)[DOM Event Listener](https://www.w3schools.com/js/js_htmldom_eventlistener.asp)[DOM Navigation](https://www.w3schools.com/js/js_htmldom_navigation.asp)[DOM Nodes](https://www.w3schools.com/js/js_htmldom_nodes.asp)[DOM Collections](https://www.w3schools.com/js/js_htmldom_collections.asp)[DOM Node Lists](https://www.w3schools.com/js/js_htmldom_nodelist.asp)

## JS Browser BOM

[JS Window](https://www.w3schools.com/js/js_window.asp)[JS Screen](https://www.w3schools.com/js/js_window_screen.asp)[JS Location](https://www.w3schools.com/js/js_window_location.asp)[JS History](https://www.w3schools.com/js/js_window_history.asp)[JS Navigator](https://www.w3schools.com/js/js_window_navigator.asp)[JS Popup Alert](https://www.w3schools.com/js/js_popup.asp)[JS Timing](https://www.w3schools.com/js/js_timing.asp)[JS Cookies](https://www.w3schools.com/js/js_cookies.asp)

## JS Web APIs

[Web API Intro](https://www.w3schools.com/js/js_api_intro.asp)[Web Forms API](https://www.w3schools.com/js/js_validation_api.asp)[Web History API](https://www.w3schools.com/js/js_api_history.asp)[Web Storage API](https://www.w3schools.com/js/js_api_web_storage.asp)[Web Worker API](https://www.w3schools.com/js/js_api_web_workers.asp)[Web Fetch API](https://www.w3schools.com/js/js_api_fetch.asp)[Web Geolocation API](https://www.w3schools.com/js/js_api_geolocation.asp)

## JS AJAX

[AJAX Intro](https://www.w3schools.com/js/js_ajax_intro.asp)[AJAX XMLHttp](https://www.w3schools.com/js/js_ajax_http.asp)[AJAX Request](https://www.w3schools.com/js/js_ajax_http_send.asp)[AJAX Response](https://www.w3schools.com/js/js_ajax_http_response.asp)[AJAX XML File](https://www.w3schools.com/js/js_ajax_xmlfile.asp)[AJAX PHP](https://www.w3schools.com/js/js_ajax_php.asp)[AJAX ASP](https://www.w3schools.com/js/js_ajax_asp.asp)[AJAX Database](https://www.w3schools.com/js/js_ajax_database.asp)[AJAX Applications](https://www.w3schools.com/js/js_ajax_applications.asp)[AJAX Examples](https://www.w3schools.com/js/js_ajax_examples.asp)

## JS JSON

[JSON Intro](https://www.w3schools.com/js/js_json_intro.asp)[JSON Syntax](https://www.w3schools.com/js/js_json_syntax.asp)[JSON vs XML](https://www.w3schools.com/js/js_json_xml.asp)[JSON Data Types](https://www.w3schools.com/js/js_json_datatypes.asp)[JSON Parse](https://www.w3schools.com/js/js_json_parse.asp)[JSON Stringify](https://www.w3schools.com/js/js_json_stringify.asp)[JSON Objects](https://www.w3schools.com/js/js_json_objects.asp)[JSON Arrays](https://www.w3schools.com/js/js_json_arrays.asp)[JSON Server](https://www.w3schools.com/js/js_json_server.asp)[JSON PHP](https://www.w3schools.com/js/js_json_php.asp)[JSON HTML](https://www.w3schools.com/js/js_json_html.asp)[JSON JSONP](https://www.w3schools.com/js/js_json_jsonp.asp)

## JS vs jQuery

[jQuery Selectors](https://www.w3schools.com/js/js_jquery_selectors.asp)[jQuery HTML](https://www.w3schools.com/js/js_jquery_elements.asp)[jQuery CSS](https://www.w3schools.com/js/js_jquery_css.asp)[jQuery DOM](https://www.w3schools.com/js/js_jquery_dom.asp)

## JS Graphics

[JS Graphics](https://www.w3schools.com/js/js_graphics.asp)[JS Canvas](https://www.w3schools.com/js/js_graphics_canvas.asp)[JS Plotly](https://www.w3schools.com/js/js_graphics_plotly.asp)[JS Chart.js](https://www.w3schools.com/js/js_graphics_chartjs.asp)[JS Google Chart](https://www.w3schools.com/js/js_graphics_google_chart.asp)[JS D3.js](https://www.w3schools.com/js/js_graphics_d3js.asp)

## JS Examples

[JS Examples](https://www.w3schools.com/js/js_examples.asp)[JS HTML DOM](https://www.w3schools.com/js/js_dom_examples.asp)[JS HTML Input](https://www.w3schools.com/js/js_input_examples.asp)[JS HTML Objects](https://www.w3schools.com/js/js_ex_dom.asp)[JS HTML Events](https://www.w3schools.com/js/js_events_examples.asp)[JS Browser](https://www.w3schools.com/js/js_ex_browser.asp)[JS Editor](https://www.w3schools.com/js/js_editor.asp)[JS Exercises](https://www.w3schools.com/js/js_exercises.asp)[JS Quiz](https://www.w3schools.com/js/js_quiz.asp)[JS Certificate](https://www.w3schools.com/js/js_exam.asp)

## JS References

[JavaScript ObjectsHTML DOM Objects](https://www.w3schools.com/jsref/default.asp)

# **JavaScript Arrays**

[❮ Previous](https://www.w3schools.com/js/js_number_properties.asp)[Next ❯](https://www.w3schools.com/js/js_array_methods.asp)

An array is a special variable, which can hold more than one value:

const cars = ["Saab", "Volvo", "BMW"];

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array)

## Why Use Arrays?

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

let car1 = "Saab";  
let car2 = "Volvo";  
let car3 = "BMW";

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

## Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

const array\_name = [item1, item2, ...];

It is a common practice to declare arrays with the const keyword.

Learn more about const with arrays in the chapter: [JS Array Const](https://www.w3schools.com/js/js_array_const.asp).

### **Example**

const cars = ["Saab", "Volvo", "BMW"];

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array)

Spaces and line breaks are not important. A declaration can span multiple lines:

### **Example**

const cars = [  
  "Saab",  
  "Volvo",  
  "BMW"  
];

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_newlines)

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

### **Example**

const cars = [];  
cars[0]= "Saab";  
cars[1]= "Volvo";  
cars[2]= "BMW";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_add_elements)

## Using the JavaScript Keyword new

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

### **Example**

const cars = new Array("Saab", "Volvo", "BMW");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_new)

The two examples above do exactly the same.

There is no need to use new Array().

For simplicity, readability and execution speed, use the array literal method.

## Accessing Array Elements

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

const cars = ["Saab", "Volvo", "BMW"];  
let car = cars[0];

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_element)

**Note:** Array indexes start with 0.

[0] is the first element. [1] is the second element.

## Changing an Array Element

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

cars[0] = "Opel";

### **Example**

const cars = ["Saab", "Volvo", "BMW"];  
cars[0] = "Opel";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_element_change)

## Access the Full Array

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

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_full)

## 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:

### **Array:**

const person = ["John", "Doe", 46];

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_array)

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

### **Object:**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_object)

## Array Elements Can Be Objects

JavaScript variables can be objects. Arrays are special kinds of objects.

Because of this, you can have variables of different types in the same Array.

You can have objects in an Array. You can have functions in an Array. You can have arrays in an Array:

myArray[0] = Date.now;  
myArray[1] = myFunction;  
myArray[2] = myCars;

## 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

Array methods are covered in the next chapters.

## The length Property

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

### **Example**

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
let length = fruits.length;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_length)

The length property is always one more than the highest array index.

## Accessing the First Array Element

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_first)

## Accessing the Last Array Element

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_last)

## Looping Array Elements

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

### **Example**

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>";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_loop)

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

### **Example**

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
  
let text = "<ul>";  
fruits.forEach(myFunction);  
text += "</ul>";  
  
function myFunction(value) {  
  text += "<li>" + value + "</li>";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_loop_foreach)

## Adding Array Elements

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

### **Example**

const fruits = ["Banana", "Orange", "Apple"];  
fruits.push("Lemon");  // Adds a new element (Lemon) to fruits

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_add_push)

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

### **Example**

const fruits = ["Banana", "Orange", "Apple"];  
fruits[fruits.length] = "Lemon";  // Adds "Lemon" to fruits

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_add)

**WARNING !**

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

### **Example**

const fruits = ["Banana", "Orange", "Apple"];  
fruits[6] = "Lemon";  // Creates undefined "holes" in fruits

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_holes)

## Associative Arrays

Many programming languages support arrays with named indexes.

Arrays with named indexes are called associative arrays (or hashes).

JavaScript does **not** support arrays with named indexes.

In JavaScript, **arrays** always use **numbered indexes**.

### **Example**

const person = [];  
person[0] = "John";  
person[1] = "Doe";  
person[2] = 46;  
person.length;    // Will return 3  
person[0];        // Will return "John"

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_associative_1)

**WARNING !!**  
If you use named indexes, JavaScript will redefine the array to an object.

After that, some array methods and properties will produce **incorrect results**.

### **Example:**

const person = [];  
person["firstName"] = "John";  
person["lastName"] = "Doe";  
person["age"] = 46;  
person.length;     // Will return 0  
person[0];         // Will return undefined

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_associative_2)

## The Difference Between Arrays and Objects

In JavaScript, **arrays** use **numbered indexes**.

In JavaScript, **objects** use **named indexes**.

Arrays are a special kind of objects, with numbered indexes.

## When to Use Arrays. When to use Objects.

* JavaScript does not support associative arrays.
* You should use **objects** when you want the element names to be **strings (text)**.
* You should use **arrays** when you want the element names to be **numbers**.

## JavaScript new Array()

JavaScript has a built-in array constructor new Array().

But you can safely use [] instead.

These two different statements both create a new empty array named points:

const points = new Array();  
const points = [];

These two different statements both create a new array containing 6 numbers:

const points = new Array(40, 100, 1, 5, 25, 10);  
const points = [40, 100, 1, 5, 25, 10];

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_literal)

The new keyword can produce some unexpected results:

// Create an array with three elements:  
const points = new Array(40, 100, 1);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_new_array_three)

// Create an array with two elements:  
const points = new Array(40, 100);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_new_array_two)

// Create an array with one element ???  
const points = new Array(40);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_new_error)

### **A Common Error**

const points = [40];

is not the same as:

const points = new Array(40);

// Create an array with one element:  
const points = [40];

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_one)

// Create an array with 40 undefined elements:  
const points = new Array(40);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_new_error2)

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

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

const fruits = ["Banana", "Orange", "Apple"];  
let type = typeof fruits;

### **Solution 1:**

## Array.isArray(fruits);

### **Solution 2:**

fruits instanceof Array;

# **JavaScript Sets**

A JavaScript Set is a collection of unique values.

Each value can only occur once in a Set.

# **JavaScript Maps**

A Map holds key-value pairs where the keys can be any datatype.

A Map remembers the original insertion order of the keys.

# **JavaScript Type Conversion**

* Converting Strings to Numbers
* Converting Numbers to Strings
* Converting Dates to Numbers
* Converting Numbers to Dates
* Converting Booleans to Numbers
* Converting Numbers to Booleans

## Converting Strings to Numbers

The global method Number() converts a variable (or a value) into a number.

A numeric string (like "3.14") converts to a number (like 3.14).

# **JavaScript Use Strict**

[❮ Previous](https://www.w3schools.com/js/js_hoisting.asp)[Next ❯](https://www.w3schools.com/js/js_this.asp)

"use strict"; Defines that JavaScript code should be executed in "strict mode".

## The "use strict" Directive

The "use strict" directive was new in ECMAScript version 5.

It is not a statement, but a literal expression, ignored by earlier versions of JavaScript.

The purpose of "use strict" is to indicate that the code should be executed in "strict mode".

With strict mode, you can not, for example, use undeclared variables.

All modern browsers support "use strict" except Internet Explorer 9 and lower

## Declaring Strict Mode

Strict mode is declared by adding "use strict"; to the beginning of a script or a function.

"use strict";  
x = 3.14;       // This will cause an error because x is not declared

# **JavaScript Arrow Function**

Arrow functions allow us to write shorter function syntax:

let myFunction = (a, b) => a \* b;

# **JavaScript Classes**

## JavaScript Class Syntax

Use the keyword class to create a class.

Always add a method named constructor():

### **Syntax**

class ClassName {  
  constructor() { ... }  
}

## Using a Class

When you have a class, you can use the class to create objects:

### **Example**

let myCar1 = new Car("Ford", 2014);  
let myCar2 = new Car("Audi", 2019);

# **JavaScript Modules**

JavaScript modules allow you to break up your code into separate files.

This makes it easier to maintain the code-base.

JavaScript modules rely on the import and export statements.

## Export

You can export a function or variable from any file.

Let us create a file named person.js, and fill it with the things we want to export.

There are two types of exports: Named and Default.

## Named Exports

You can create named exports two ways. In-line individually, or all at once at the bottom.

### **In-line individually:**

person.js

export const name = "Jesse";  
export const age = 40;

### **All at once at the bottom:**

person.js

const name = "Jesse";  
const age = 40;  
  
export {name, age};

## Default Exports

Let us create another file, named message.js, and use it for demonstrating default export.

You can only have one default export in a file.

### **Example**

message.js

const message = () => {  
const name = "Jesse";  
const age = 40;  
return name + ' is ' + age + 'years old.';  
};  
  
export default message;

## Import

You can import modules into a file in two ways, based on if they are named exports or default exports.

Named exports are constructed using curly braces. Default exports are not.

### **Import from named exports**

Import named exports from the file person.js:

import { name, age } from "./person.js";

### **Import from default exports**

Import a default export from the file message.js:

import message from "./message.js";

What is JSON?

* JSON stands for **J**ava**S**cript **O**bject **N**otation
* JSON is a lightweight data interchange format
* JSON is language independent **\***
* JSON is "self-describing" and easy to understand

 The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. Code for reading and generating JSON data can be written in any programming language.

## The debugger Keyword

The debugger keyword stops the execution of JavaScript, and calls (if available) the debugging function.

This has the same function as setting a breakpoint in the debugger.

If no debugging is available, the debugger statement has no effect.

With the debugger turned on, this code will stop executing before it executes the third line.

let x = 15 \* 5;  
debugger;  
document.getElementById("demo").innerHTML = x;

# Private class features

Private members are not native to the language before this syntax existed. In prototypical inheritance, its behavior may be emulated with [WeakMap](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/WeakMap#emulating_private_members) objects or [closures](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures#emulating_private_methods_with_closures), but they can't compare to the # syntax in terms of ergonomics.

class ClassWithPrivate {

#privateField;

#privateFieldWithInitializer = 42;

## Class Inheritance

To create a class inheritance, use the extends keyword.

# **JavaScript Static Methods**

Static class methods are defined on the class itself.

You cannot call a static method on an object, only on an object class.

# **JavaScript HTML DOM**

## The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:

### **The HTML DOM Tree of Objects**



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

* JavaScript can change all the HTML elements in the page
* JavaScript can change all the HTML attributes in the page
* JavaScript can change all the CSS styles in the page
* JavaScript can remove existing HTML elements and attributes
* JavaScript can add new HTML elements and attributes
* JavaScript can react to all existing HTML events in the page
* JavaScript can create new HTML events in the page\*

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

*"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

# **JavaScript HTML DOM Document**

The HTML DOM document object is the owner of all other objects in your web page.

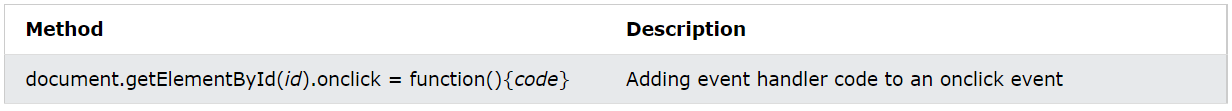
## Finding HTML Elements

## Changing HTML Elements

## Adding and Deleting Elements

## 

## Adding Events Handlers



## Finding HTML Objects

