**JavaScript Arrays**

An array is a special variable, which can hold more than one value at a time.

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

Syntax:

var *array\_name* = [*item1*, *item2*, ...];

## Using the JavaScript Keyword new

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

### Example

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

## Access the Elements of an Array

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

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

var name = cars[0];

### Example

var cars = ["Saab", "Volvo", "BMW"];  
document.getElementById("demo").innerHTML = cars[0];

## Access the Full Array

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

### Example

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

## Avoid new Array()

There is no need to use the JavaScript's built-in array constructor new Array().

**Use [] instead.**

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

var points = new Array();     // Bad  
var points = [];              // Good

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

var points = new Array(40, 100, 1, 5, 25, 10); // Bad  
var points = [40, 100, 1, 5, 25, 10];          // Good

## 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** |
| https://www.w3schools.com/js/objectExplained.gif | 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

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";

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"};

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.

## Object Definition

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

### Example

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

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

### Example

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

## Object Properties

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

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

## Accessing Object Properties

You can access object properties in two ways:

*objectName.propertyName*

or

*objectName["propertyName"]*

### Example1

person.lastName;

### Example2

person["lastName"];

## Object Methods

Objects can also have **methods**.

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

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

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

A method is a function stored as a property.

### Example

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

## 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](https://www.w3schools.com/js/js_this.asp).

## Accessing Object Methods

You access an object method with the following syntax:

*objectName.methodName()*

### Example

name = person.fullName();

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

### Example

name = person.fullName;

## 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