Introduction To Javascript

Why Study JavaScript?

JavaScript is one of the **3 languages** all web developers **must** learn:

1. **HTML** to define the content of web pages
2. **CSS** to specify the layout of web pages
3. **JavaScript** to program the behavior of web pages


JS
Java™

JavaScript Can Change HTML Attributes

```html
<!DOCTYPE html>
<html>
<body>

<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">
    Turn on the light
</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">
    Turn off the light
</button>

</body>
</html>
```

JavaScript Can Change HTML Attributes(2)

```html
<!DOCTYPE html>
<html>
<head>
<script>

  function onLight() {
      document.getElementById('myImage').src='pic_bulbon.gif';
  }

  function offLight() {
      document.getElementById('myImage').src='pic_bulboff.gif';
  }

</script>
</head>
<body>

<button onclick="onLight()">
    Turn on the light
</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="offLight()">
    Turn off the light
</button>

</body>
</html>
```

JavaScript Can Change HTML Styles (CSS)

```html
<!DOCTYPE html>
<html>
<body>

<p id="demo">JavaScript can change the style of an HTML element.</p>

<button type="button"
onclick="document.getElementById('demo').style.fontSize='35px'">Click Me!
</button>


</body>
</html>
```

JavaScript can change the style of an HTML element.

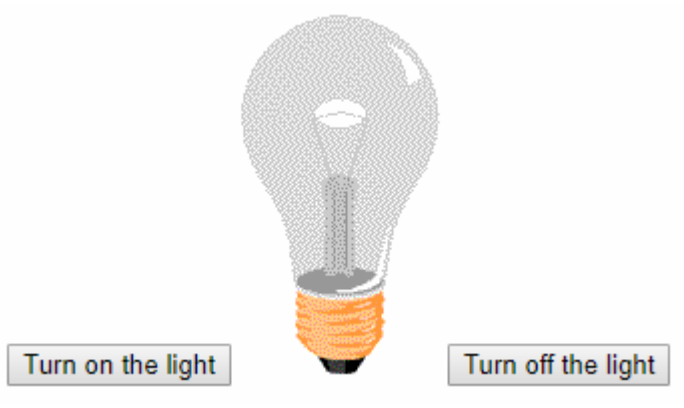
Click Me!

JavaScript can change the style of an HTML element.

Click Me!

Inline JavaScript

```html
<!DOCTYPE html>
<html>
<body>

<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">
    Turn on the light
</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">
    Turn off the light
</button>

</body>
</html>
```



Internal JavaScript

```html
<!DOCTYPE html>
<html>
<head>
<script>

  function onLight() {
      document.getElementById('myImage').src='pic_bulbon.gif';
  }

  function offLight() {
      document.getElementById('myImage').src='pic_bulboff.gif';
  }

</script>
</head>
<body>

<button onclick="onLight()">
    Turn on the light
</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="offLight()">
    Turn off the light
</button>

</body>
</html>
```

External JavaScript

```html
<!DOCTYPE html>
<html>
<head>
<script src="myScript.js"></script>
</head>
<body>

<button onclick="onLight()">
    Turn on the light
</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="offLight()">
    Turn off the light
</button>

</body>
</html>
```

External file: myScript.js

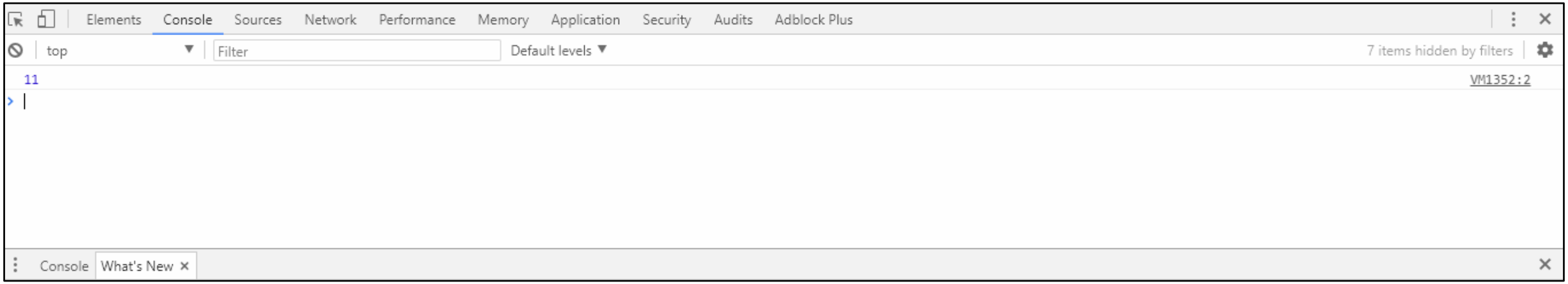
```html
<script>

  function onLight() {
      document.getElementById('myImage').src='pic_bulbon.gif';
  }

   function offLight() {
      document.getElementById('myImage').src='pic_bulboff.gif';
  }

</script>
```

console.log()

```html
<!DOCTYPE html>
<html>
<body>

<h2>Activate debugging with F12</h2>

<p>Select "Console" in the debugger menu. Then click Run again.</p>

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

Activate debugging with F12

Select "Console" in the debugger menu. Then click Run again.



JavaScript Syntax

```
var x, y;           // How to declare variables
x = 5; y = 6;       // How to assign values
z = x + y;          // How to compute values
```

JavaScript Comments

```
var x = 5;   // I will be executed

// var x = 6;   I will NOT be executed
```

```
/*
The code below will change
the heading with id = "myH"
and the paragraph with id = "myP"
in my web page:
*/
```

JavaScript is Case Sensitive

All JavaScript identifiers are **case sensitive**.

The variables **lastName** and **lastname**, are two different variables.

```
var lastname, lastName;
lastName = "Doe";
lastname = "Peterson";
```

JavaScript and Camel Case

Historically, programmers have used different ways of joining multiple words into one variable name:

Hyphens:

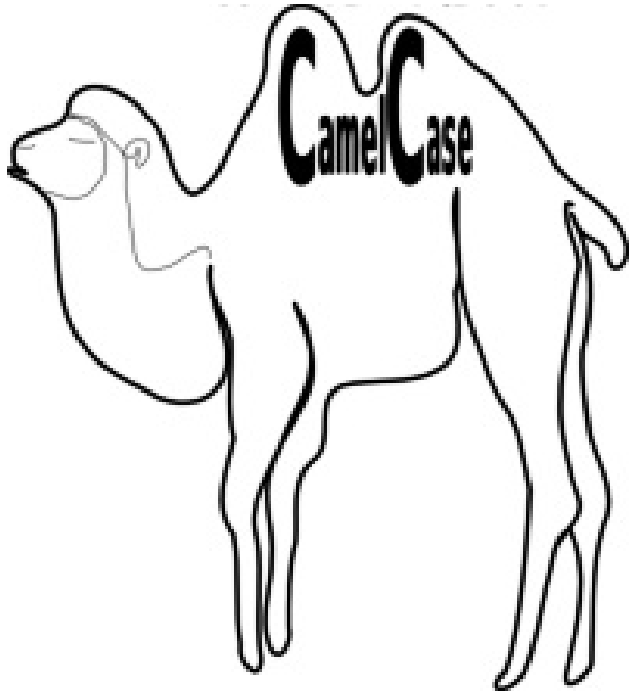
first-name, last-name, master-card, inter-city.

Underscore:

first_name, last_name, master_card, inter_city.

Upper Camel Case (Pascal Case):

FirstName, LastName, MasterCard, InterCity.



JavaScript Variables

```
var price1 = 5;
var price2 = 6;
var total = price1 + price2;
```

```
var pi = 3.14;
var person = "John Doe";
var answer = 'Yes I am!';
```

```
var person = "John Doe", carName = "Volvo", price = 200;
```

```
var person = "John Doe",
carName = "Volvo",
price = 200;
```

JavaScript Data Types

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

JavaScript Types are Dynamic

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

The typeof Operator

```
typeof ""                    // Returns "string"
typeof "John"                // Returns "string"
typeof "John Doe"            // Returns "string"
```

```
typeof 0                      // Returns "number"
typeof 314                    // Returns "number"
typeof 3.14                   // Returns "number"
typeof (3)                    // Returns "number"
typeof (3 + 4)                // Returns "number"
```

Primitive Data

```
typeof "John"                // Returns "string"
typeof 3.14                  // Returns "number"
typeof true                  // Returns "boolean"
typeof false                 // Returns "boolean"
typeof x                     // Returns "undefined" (if x has no value)
```

Complex Data

```
typeof {name:'John', age:34} // Returns "object"
typeof [1,2,3,4]             // Returns "object" (not "array", see note below)
typeof null                  // Returns "object"
typeof function myFunc(){}   // Returns "function"
```

The typeof operator returns "object" for arrays because in JavaScript arrays are objects.

Undefined

```
var car;                    // Value is undefined, type is undefined
```

```
var car = "";                // The value is "", the typeof is "string"
```

Null

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
person = null;          // Now value is null, but type is still an object
```

Difference Between Undefined and Null

```
typeof undefined            // undefined
typeof null                 // object

null === undefined          // false
null == undefined           // true
```

JavaScript Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

JavaScript Assignment Operators

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

JavaScript Functions

```
function name(parameter1, parameter2, parameter3) {
    code to be executed
}
```

```
function myFunction(p1, p2) {
    return p1 * p2;              // The function returns the product of p1 and p2
}
```

```
var x = myFunction(4, 3);    // Function is called, return value will end up in x

function myFunction(a, b) {
    return a * b;            // Function returns the product of a and b
}
```

JavaScript Objects

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

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

Object Properties

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

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

Object Methods

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

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

Object Definition

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

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

Accessing Object Properties

```
objectName.propertyName
```

or

```
objectName["propertyName"]
```

Example

```
person.lastName;
```

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

Accessing Object Methods

```
objectName.methodName()
```

Example

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

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

JavaScript Function Scope

In JavaScript there are two types of scope:

- Local scope
- Global scope

JavaScript has function scope: Each function creates a new scope.

Scope determines the accessibility (visibility) of these variables.

Variables defined inside a function are not accessible (visible) from outside the function.

Local JavaScript Variables

```
// code here can not use carName

function myFunction() {
    var carName = "Volvo";

    // code here can use carName

}
```

Global JavaScript Variables

```
var carName = " Volvo";

// code here can use carName

function myFunction() {

    // code here can use carName

}
```

JavaScript Arrays

```
var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
```

Creating an Array

```
var array_name = [item1, item2, ...];
```

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

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

Access the Elements of an Array

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

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

```
var name = cars[0];
```

This statement modifies the first element in cars:

```
cars[0] = "Opel";
```

JavaScript If...Else Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- Use **if** to specify a block of code to be executed, if a specified condition is true
- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed

The if Statement

Use the **if** statement to specify a block of JavaScript code to be executed if a condition is true.

Syntax

```
if (condition) {
    block of code to be executed if the condition is true
}
```

> Note that **if** is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error.

Example

Make a "Good day" greeting if the hour is less than 18:00:

```
if (hour < 18) {
    greeting = "Good day";
}
```

The else Statement

Use the **else** statement to specify a block of code to be executed if the condition is false.

```
if (condition) {
    block of code to be executed if the condition is true
} else {
    block of code to be executed if the condition is false
}
```

Example If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
```

The else if Statement

Use the **else if** statement to specify a new condition if the first condition is false.

Syntax

```
if (condition1) {
    block of code to be executed if condition1 is true
} else if (condition2) {
    block of code to be executed if the condition1 is false and condition2 is true
} else {
    block of code to be executed if the condition1 is false and condition2 is false
}
```

Example

If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 20:00, create a "Good day" greeting, otherwise a "Good evening":

```
if (time < 10) {
    greeting = "Good morning";
} else if (time < 20) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
```

JavaScript For Loop

The for loop is often the tool you will use when you want to create a loop.

The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3) {
    code block to be executed
}
```

Statement 1 is executed before the loop (the code block) starts.

Statement 2 defines the condition for running the loop (the code block).

Statement 3 is executed each time after the loop (the code block) has been executed.

Example

```
for (i = 0; i < 5; i++) {
    text += "The number is " + i + "<br>";
}
```

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4

The For/In Loop

```
var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
    text += person[x];
}
```

John Doe 25

JavaScript While Loop

The while loop loops through a block of code as long as a specified condition is true.

Syntax

```
while (condition) {
    code block to be executed
}
```

Example

```
while (i < 10) {
    text += "The number is " + i;
    i++;
}
```

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9

Comparing For and While

```
var cars = ["BMW", "Volvo", "Saab", "Ford"];
var i = 0;
var text = "";

for (;cars[i];) {
    text += cars[i] + "<br>";
    i++;
}
```

```
var cars = ["BMW", "Volvo", "Saab", "Ford"];
var i = 0;
var text = "";

while (cars[i]) {
    text += cars[i] + "<br>";
    i++;
}
```

JavaScript Forms Validation

```html
<!DOCTYPE html>
<html>
<head>
<script>

function validateForm() {

    var x = document.forms["myForm"]["fname"].value;

    if (x == "") {
        alert("Name must be filled out");
        return false;
    }
}

</script>
</head>
<body>

<form name="myForm" action="/action_page.php"onsubmit="return validateForm()" method="post">

    Name:   <input type="text" name="fname">

    <input type="submit" value="Submit">

</form>

</body>
```

Name: Submit

Name must be filled out

ตกลง

JavaScript Can Validate Numeric Input

```html
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
    var x, text;

    // Get the value of the input field with id="numb"
    x = document.getElementById("numb").value;

    // If x is Not a Number or less than one or greater than 10
    if (isNaN(x) || x < 1 || x > 10) {
        text = "Input not valid";
    } else {
        text = "Input OK";
    }
    document.getElementById("demo").innerHTML = text;
}
</script>
</head>
<body>

<p>Please input a number between 1 and 10:</p>

<input id="numb">

<button type="button" onclick="myFunction()">Submit</button>

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

</body>
</html>
```

Please input a number between 1 and 10:

2 Submit

Input OK

Please input a number between 1 and 10:

dddd Submit

Input not valid

{JSON}

JavaScript Object Notation

JavaScript JSON

JSON is a format for storing and transporting data.

JSON is often used when data is sent from a server to a web page.

What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is 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.

JSON Example

This JSON syntax defines an employees object: an array of 3 employee records (objects):

JSON Example

```
{
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
}
```

JSON Syntax Rules

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

JSON Data - A Name and a Value

JSON data is written as name/value pairs, just like JavaScript object properties.

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

```
"firstName":"John"
```

> JSON names require double quotes. JavaScript names do not.

JSON Objects

JSON objects are written inside curly braces.

Just like in JavaScript, objects can contain multiple name/value pairs:

```
{"firstName":"John", "lastName":"Doe"}
```

JSON Arrays

JSON arrays are written inside square brackets.

Just like in JavaScript, an array can contain objects:

```
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
```

In the example above, the object "employees" is an array. It contains three objects.

Each object is a record of a person (with a first name and a last name).

Converting a JSON Text to a JavaScript Object

A common use of JSON is to read data from a web server, and display the data in a web page.

For simplicity, this can be demonstrated using a string as input.

First, create a JavaScript string containing JSON syntax:

```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

Then, use the JavaScript built-in function JSON.parse() to convert the string into a JavaScript object:

```
var obj = JSON.parse(text);
```

```
obj.employees[1].firstName + " " + obj.employees[1].lastName;
```


Quiz

Reference

- https://www.w3schools.com/