End-users curriculum

Codinc

CodeMob: End-users curriculum - Codinc
October 2017. http://codemob.eu/. Authors:

Co-funded by the Erasmus+ Programme of the European Union

This publication has been co-funded by the European Commission's Ersamus+ Programme.

Contents

Welcome to this coding course!

At the end of it, you should be able to:

- make a game (e.g. memory) in JavaScript
- add a GUI (graphical user interface) with HTML5 & CSS3
- understand and explain the basic programming concepts

First of all, keep in mind that it exits thousands of resources online.
It would not be possible to propose a complete course here. However, in programming, you will find all your answers thanks to a simple search on the web.
Consider this course as an introduction, step by step, to JavaScript. Enjoy!

Why learn JavaScript?

Not to be confused with Java, JavaScript allows you to build interactive websites. JavaScript has become an essential web technology along with HTML and CSS, as most browsers implement JavaScript. Thus, You must learn JavaScript if you want to get into web development, and you must learn it well if you're planning on being a front-end developer or on using JavaScript for backend development.

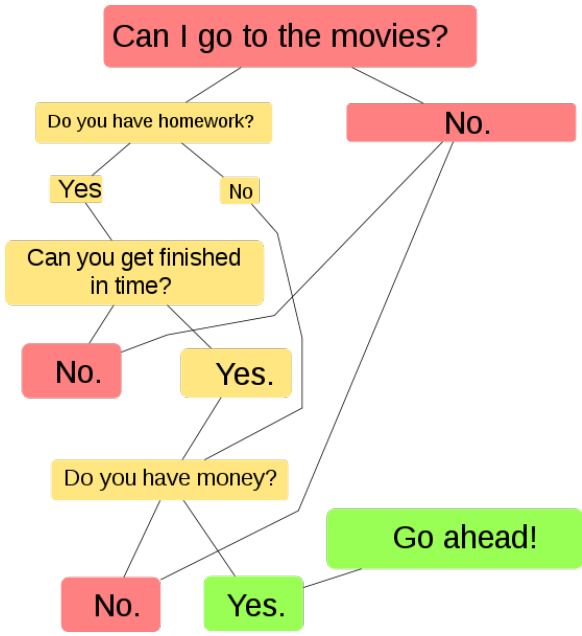
Furthermore, JavaScript usage has now extended to mobile app development, desktop app development, and game development. All in all, it has exploded in popularity and is now a very useful skill to learn.

Source:

Algorithmics

In mathematics and computer science, an **algorithm** (ælgərɪðəm/ ***AL**-gə-ri-dhəm*) is a self-contained step-by-step set of operations to be performed. Algorithms perform calculation, data processing, and/or automated reasoning tasks.

Here is a simple example of what an algorithm can be:



What is the relation between an **algorithm** and a **program**? Look at an **algorithm** as a formula for working something out. A **program** is a series of instructions for the computer that uses algorithms to execute the desired task.

Let's understand this through an example:

Simple algorithm Area of rectangle A = length times Depth
Program to work out area would be
Ask user to input length
Ask user to input depth
Use area algorithm to calculate area
Display answer

Introduction to HTML and CSS

To create a web site, you must provide information to the computer. It is not enough just to type the text that will be in his site, it is also necessary to know how to place this text, insert pictures, make links etc. ... To explain to the computer what you want, it will be necessary to use a language that it understands.

There are languages used to create programs, such as C ++ or Java. These languages are nevertheless complex and intended for people who already have some computer knowledge.

HTML and **CSS** are precisely used to create websites, and they have been created to be simple to use. Each of these 2 languages is used to do something specific, and the two complement each other naturally to finally give a website:

• **HTML**: it is the abbreviation of **HyperText Markup Language** it appeared in 1991 when the Web was launched. Its role is to manage and organize content. It is therefore in HTML that you write what must be displayed on the page: text, links, images. . . You will say for example: This is my title, this is my menu, here is the main text of the page. ...
In this course we will work on the latest version of HTML (HTML5) which is today the language of the future that everyone is encouraged to use.

• **CSS**: this is the abbreviation for **CASCADING STYLE SHEETS**. This language only serves to present the web page. It is in CSS that it will be said: "My titles are in red and are underlined, my text is in ARIAL font, my name is centered, my menu has a white background. .. "etc. ... With this language, we will be able to create quickly and simply the layout of your site.

The editor

One question you should definitely ask yourself is: "What software will I need to create my website?"

Notepad is enough to create a website! But to facilitate our work we can use a code editor such as Notepad ++ (the advantage is that it automatically colors the HTML / CSS code to make it more readable).

Browsers

What is a browser?

The browser is the program that allows you to see websites. The work of the browser is to read the HTML / CSS code you wrote, and to display what it represents. If your CSS code says "Titles are red," then the browser will display the titles in red.

Among the **browsers** that exist, here are **the main ones**:
Internet Explorer - Mozilla Firefox - Opera - Netscape - Konqueror (for Linux) - Lynx (for Linux) - Apple Safari (for Mac) - etc. ...

Create your first HTML document

What is HTML?
As said above, HTML is a markup language for describing web documents (web pages).

- HTML stands for Hyper Text Markup Language
- A markup language is a set of markup tags
- HTML documents are described by HTML tags
- Each HTML tag describes different document content

A Small HTML Document
Example

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Example Explained

- The <!DOCTYPE html> declaration defines this document to be HTML5

- The text between <html> and </html> describes an HTML document
- The text between <head> and </head> provides information about the document
- The text between <title> and </title> provides a title for the document
- The text between <body> and </body> describes the visible page content
- The text between <h1> and </h1> describes a heading
- The text between and describes a paragraph

Using this description, a web browser will display a document with a heading and a paragraph.

For more information see the full document at .

HTML elements you need

In this lesson, your task is to create a simple HTML document which contains the following content:

Don’t forget to use comments

A **comment** is an HTML **tag** with a very special shape:

<!-- This is a comment -->

You can put it where you want in your source code: it has no impact on your page, but you can use comments to explain your code, which can help you when you edit the source code at a later date. This is especially useful if you have a lot of code.Attention everyone can see the HTML code of your page once it's uploaded to the web. Just right click on the page and select "Show the source code of the page". Therefore, do not put sensitive information like passwords into comments ... and take care of your source code.

Create a CSS file and use it to style your HTML document

What is CSS?

- CSS stands for Cascading Style Sheets

- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

Why Use CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS Solved a Big Problem
HTML was NEVER intended to contain tags for formatting a web page!
HTML was created to describe the content of a web page, like:

<h1>This is a heading</h1>
This is a paragraph.

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.
CSS removed the style formatting from the HTML page!

CSS Saves a Lot of Work!
The style definitions are normally saved in external .css files.
With an external stylesheet file, you can change the look of an entire website by changing just one file!

For more information go to .

External style sheet

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

Three Ways to Insert CSS
There are three ways of inserting a style sheet:

1. External style sheet
2. Internal style sheet
3. Inline style

In this lesson we will be inserting an external style sheet.
With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section:

Example:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

Here is how the "myStyle.css" looks:

```
body {
    background-color: lightblue;
}

h1 {
    color: navy;
    margin-left: 20px;
}
```

Note: Do not add a space between the property value and the unit (such as margin-left: 20 px;). The correct way is: margin-left: 20px;

Programming JS

History of JavaScript

The JavaScript language was created in **1995** by **Brendan Eich** (Mozilla Foundation) on behalf of Netscape. The language, currently in version 1.8.5, is an implementation of the 3rd version of the ECMA262 standard.

● In December 1995, Sun and Netscape announced the release of JavaScript. Microsoft then reacts by developing JScript (same language for a different name to avoid a Trademark dispute with Sun).
● Netscape submits JavaScript to Ecma International for standardization in November 1996.
● **DHTML** (1996), a difficult and unnecessary start
● **Ajax** (2000), A renewed interest thanks to new possibilities
● **JSon**: an alternative to JavaScript-based XML
● **JavaScript frameworks**, ease, productivity and standardization but difficult choice
● **node.js**; Eventdriven & Serversided
    ○ NodeWebkit & Cordova

What is JavaScript?

● Scripting language (high-level language)
    ○ Interpreted (as opposed to compiled languages)
● Client-side language (It runs on the user's machine and not on the server)
    ○ Nevertheless node.js appeared in 2009 changes somewhat the situation.
● Able to change the DOM:
    ○ Create, edit, and delete HTML elements, their attributes, or CSS.
    ○ Create, listen, and delete events.
● Nothing to do with Sun's Java language ...;)

Example- & the Polyfills

Installation

Scripts can be placed in an HTML page either in the <head> or in the <body> (just before the </ body>) and They can be **internal or external**. The **type attribute is optional** because its default value is correct (JavaScript is the only language accepted in HTML).

```
<script type="text/javascript">
//<![CDATA[
...(Your JS code)
//]]>
</script>

<! If there is an external source the tag must be empty  >
<script src="./js/script.js"></script>
```

</code>mob

JavaScript example

Let's try to make our first JavaScript file. In this example we will be creating a simple alert box. We will be doing this by linking an external .js file to our .html document.

The first thing you need to do is go to the folder in which you have created your first .html an .css files. Create a new .js file and save it in the same folder as (for example) *myscript.js*.

Now add the following script to your .js file:

```
alert("I am an alert box!");
```

Finally link the myscript.js to your .html document:

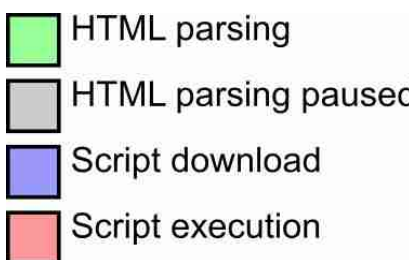
```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

You can place an external script reference in <head> or <body> as you like. The script will behave as if it was located exactly where the <script> tag is located.

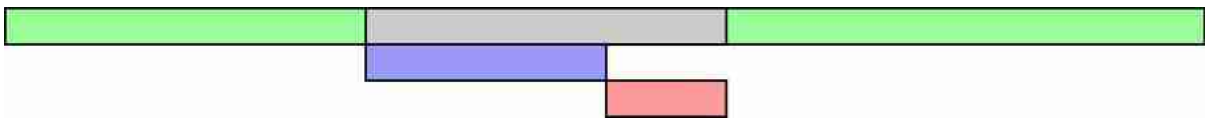
Script execution order - Defer & async

The timing of a script depends on its position in the HTML page and the defer and async attributes.

The defer and async attributes are only taken into account for **external** scripts.

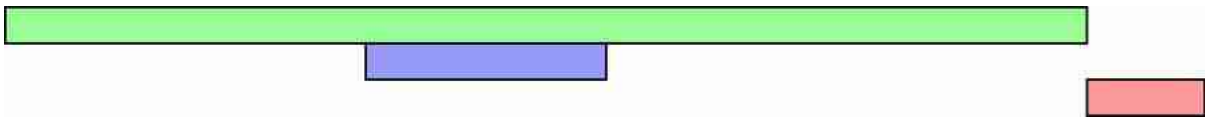


Normal execution



Common use: if two scripts follow each other, the second will never run before the end of the first because any resource is 'blocking'.

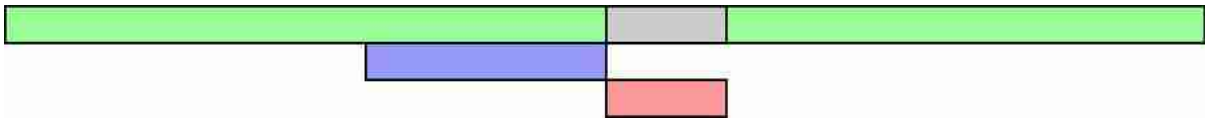
Defer



Never use document.write() or equivalent (with defer).

Always pay attention to the execution order of scripts having the same priority, it is supposed to be respected but bugs in IE4-9.

Async (HTML 5)



Ideal for a script whose execution time is not important (eg: Google Annalytics) but the order of execution of the scripts is not guaranteed!

Never use document.write() or equivalent (with async).

Layout Trashing

When JavaScript reads and rewrites or changes the DOM, the layout is invalidated and must be recalculated at some point in the future. Browsers try to wait until the end of a frame, but if they have to do so before, it can have a serious impact on performance.

In short, use as little as possible the manipulations and changes of the DOM, or at worst, group them as much as possible.

If you are interested in the topic, you can inquire about requestAnimationFrame and libraries like FastDOM ().

The JavaScript Console

It is easy to get lost writing JavaScript code without a debugger. Debugging is the process of testing, finding, and reducing bugs (errors) in computer programs.

If a script does not work (bug), we rarely see the error ... So we need to display the Javascript console to see the **error messages** (file & line of code, color code, script on the fly) .

- **Firefox**: F12> Console (JavaScript)
- **Chrome**: Ctrl + Shift + I> Console

You can also debug with **console.log(…)**; which accepts in parameter what must be displayed. It is also possible to add **break points** or use the **debugger** command.

The console.log() Method

If your browser supports debugging, you can use console.log() to display JavaScript values in the debugger window:

Example


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

<h1>My First Web Page</h1>

<script>
a = 5;
b = 6;
c = a + b;
console.log(c);
</script>

</body>
</html>
```

Setting Breakpoints

In the debugger window, you can set breakpoints in the JavaScript code.

At each breakpoint, JavaScript will stop executing, and let you examine JavaScript values.

After examining values, you can resume the execution of code (typically with a play button).

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.

```
var x = 15 * 5;
debugger;
document.getElementbyId("demo").innerHTML = x;
```

Keep in mind that the console object is not part of the standards even if it is, at least, well implemented by Firefox and Chrome.

Example API Console Firefox

Comments

JavaScript comments can be used **to explain JavaScript code**, and to make it more readable.

JavaScript comments can also be used **to prevent execution**, when testing alternative code.

Single Line Comments

Single line comments start with //.

Any text between // and the end of the line will be ignored by JavaScript (will not be executed).

This example uses a single-line comment before each code line:

```
// Change heading:
document.getElementById("myH").innerHTML = "My First Page";
// Change paragraph:
document.getElementById("myP").innerHTML = "My first paragraph.";
```

This example uses a single line comment at the end of each line to explain the code:


```
var x = 5;       // Declare x, give it the value of 5
var y = x + 2;   // Declare y, give it the value of x + 2
```

Multi-line Comments

Multi-line comments start with /* and end with */.

Any text between /* and */ will be ignored by JavaScript.

This example uses a multi-line comment (a comment block) to explain the code:

```
/*
The code below will change
the heading with id = "myH"
and the paragraph with id = "myP"
in my web page:
*/
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
```

Using Comments to Prevent Execution

Using comments to prevent execution of code is suitable for code testing.

Adding // in front of a code line changes the code lines from an executable line to a comment.

This example uses // to prevent execution of one of the code lines:

Example

```
//document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
```

This example uses a comment block to prevent execution of multiple lines:


```
/*
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
*/
```

Comments on a single line start with // and those on multiple lines are surrounded by /*...*/. Below, a slightly more complicated case for validly inserting JavaScript into an XHTML document.

Always use comments to comment on **your logic**, the parameters expected by a function, which it returns or deactivate a piece of code without erasing it.

```
<scripttype="text/javascript"> //<!
[CDATA[
varmyFirstMessage ='Hello World';

console.log(myFirstMessage);

functionmyFirstFunction(){
        vardemo =document.getElementById("demo"); demo.
        style.color ="#990000";
}

myFirstFunction(); //]]>
</script>
```

Things not to do: Eval & javascript:

Use (almost) never **eval()**, as it allows to execute arbitrary code, it represents a security risk.

The **Setinterval() function** that calls its first parameter after a certain amount of time uses eval () if it receives a string as a parameter. Therefore, prefer a function as an argument.

```
setInterval( function(){myFunction(param1,param2);},5000);
```

Keep the separation between HTML and JavaScript (as with CSS for reasons of maintenance and logic) avoiding the document.write(…) and separating the JavaScript from the layout.


```
<! Never do this ... >
<a href="javascript:myFunction();">....</a>

<! ...and avoid as much as possible this >
<a title="Click to do some JS stuff" href="enablejs.html" onclick=
"MyFunction();return false;">link</a>
```

Mathematical operators

- *, /, +, (Order of precedence & parentheses)
- % (Modulo)
- +=, =, *= & /= (x = x + 5;)
    - x++ & x (postincrement or decrement : x = x + 1)
    - ++x & x (preincrement or decrement)

Constants

Similar to variables except that once declared, their value can not be changed anymore. They are often used as predefined parameters. By convention and in almost all languages, constants names are written in uppercase letters separated by underscore. Attention, there is no support for constants on IE10.

```
// Constant
const MY_FAVORITE_NUMBER =9;

// 'False' constant only by convention
var MY_LUCKY_NUMBER =7;
```

Variable & their data type

The variables JavaScript is a **dynamic typing** language like PHP, uses a **pointed syntax** and is **case sensitive**, so be methodical and rigorous because the type of a variable may change at runtime.

The use of the Lower Camel Case for variable names (first letter of each word except the first in upper case, no space) is considered best practice. JavaScript has a Garbage Collector that destroys variables that are no longer used (or set to null).

Do not use syntax var a = new Array(); or dash in variable names (for example: var causeAnError = 5;).

```
varuserIdNumber; // Variable declaration with var tag
userIdNumber =5; // Initialization of the same variable

varuserIdNumber =5; // Simultaneous declaration and initialization

varvariable1 ="Hello World!";
varvariable2 =42;

varvariable1 ="Hello World!",
    variable2 =42;          // Multiple declaration
```

Data type

- Primitive
  - **Strings** (character strings) : "Hello World"
  - **Numbers** (Integer and floating-point numbers) : 3.14 or 42
  - **Booleans** (Boolean values) : true or false
  - Undefined (A variable that has not yet been declared or does not match any type)
  - null (Empty variable, that is to say to which one has assigned a zero value)µ

- **Objects**
  - **Functions**
  - **Arrays**
  - Date, Math
  - RegExp (Regular expressions)

Casting

It is possible to change the type of a variable (Casting) to another via **Number**(…), **String**(…) and **Boolean**(…) or even some methods like .split() or .join().

```
varnbr2Boo =Boolean(0);                 // false
varstr2Nbr =Number('12');               // 12
varboo2Str =String(true);               // "true"
```

Strings

● String concatenation is executed via the "**+**"
● Strings are delimited either by double quotation marks (**double quote**) or single quotation marks (single quote).

  ○ Unlike PHP, there is no difference between the two.

- Special characters can be inserted via \(Escape character)
  - \ ", \ n (return to the line), \\, ...
  - \ at each return to the line

● The length is accessible via the **.length** property

● Strings have many methods;

- .charAt(); indexOf(), substr, substring(), toLowerCase(), toUpperCase(), ...

● Do not access a string as if it were an Array.

Exemple

Numbers

● Only one type (Integer & Float) stored under a 64bit Floating Point.

● The Infinity & -Infinity constants obviously represent the positive infinity and the negative.

● **NaN** (Not a Number) will be returned for mathematical impossibility as a division by a string for example.

● The **.toString()** method converts them to strings

- .toString(2): conversion to binary
- .toString(16): conversion to hexadecimal

● The **Math** object offers many features such as:

- Math.random(): Returns a random number between 0 inclusive and 1 exclusive
- Math.min(…,…)& Math.max(…,…): Minimum & Maximum
- Math.round(), Math.ceil() & Math.floor(): rounded, rounded up and down

- Main.sin(), Math.cos(), Math.PI, …

Example http://www.w3schools.com/js/js_number_methods.asp

Exercise: Randomly display the number 0 or 1 in the console each time you reload the page (F5).

Array

- Arrays are instantiated using empty brackets **[]** and use them to access a specific index.
  - `var myArray =[];// New empty array`
- The index starts as in almost all computer languages at 0.
  - `myArray[0]=42;`
- It is possible to find the length of an array via the **.length** property
  - Never modify an Array in which you are prompted.
- Caution, these are not Associative Arrays
- Array has methods for just about everything; .push(), .pop(), .splice(…), …
  - It is possible to sort the character strings of an array using the method **.sort()**
  - and digits via .sort() using an argument comparison function

.sort(function(a, b) {return a - b})

ExamplesW3Schools& www.w3schools.com/js/js_array_methods.asp

Exercise: Display the last element of an array whose length you do not know.

Testing a Variable Type

There are many methods to test the type of a variable.

- **typeof** myVariable
    - **typeof**myVar ===**'undefined'**
    - The NaN data type is 'number'
    - The data type of an array is 'object'
    - The data type of the Date object is 'object'
    - The data type of null is 'object'
    - The data type of an indefinite variable is 'undefined'

Examples

- ```
  typeof "John"                 // Returns "string"
  typeof 3.14                   // Returns "number"
  typeof false                  // Returns "boolean"
  typeof [1,2,3,4]              // Returns "object" (not "array", see note
  below)
  typeof {name:'John', age:34} // Returns "object"
  ```

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

isNaN(…)

- isFinite(…)
- Array.isArray(…);
- …

Var or not var?

In a function, a **variable** declared **with var is local** and global if declared without. For short, it is advisable to **always declare its variables** (Avoid overloading the global scope, ECMA6, ...).
  ```

Their lifetime is different: a local variable is destroyed at the end of the function that contains it (Current execution context), a global variable when the HTML page is closed.

There are of course more subtleties than that but keep in mind that we must use var if we declare a variable without assigning it and that only an undeclared variable can be deleted (which is inadvisable anyway).

Scope

In JavaScript, objects and functions are also variables.

The scope is the set of variables, functions, and objects that you have access to at a given time. It therefore changes within a function (Function scope).

The scope always goes back from the local to the general, that is if JavaScript does not find a variable in its scope, then it will look for it in the Global and finally generate an error if it finds nothing.

Hoisting

Javascript always declares the variables and functions before any code is executed: first of all, it moves them to the beginning of the block to which they belong.

It is **always advisable to declare variables and functions at the beginning** of their scope.

Condition

Conditional statements are used to ***perform different actions based on different conditions***.

Conditional 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

If ... then ... else ...
Note the space between else and if (and not elseif as in PHP for example).

```
if(money <0){

      status ="I'm broke..."; }

elseif(money <10000){

      status ="Could be worse...";

}else{

      status ="I'm rich!";
}
```

Ternary operator

A compact write of an If ... then, useful for conditional declarations of variables for example.

```
varadmissibleAtI3 =(sex =="F")?true : false;
```

Switch

A simple way to browse many choices.

Note that a switch makes it possible to have several cases pointing to the same piece of code and that it uses a **strict comparison** (===).

```
switch (newDate().getDay()){

        case1:


        case2:
        case3:
        default:

                text ="Looking forward to the Weekend
                ";
                break;

        case4:
        case5:

                text ="I'm tired and bored";
                break;

        case0:
        case6:
                text ="Time to play : )";

}
```

Logical operators

- == (different from =), !=
- ===, !== : comparison of value and type
- >, >=, <, <=
- && (And)
- || (Or)

- ! (Not)
    - toggle = !toggle // true becomes false and false becomes true.

JavaScript, like all languages, stops at the first false in a condition with &&(and), which makes it possible to first test the existence of a variable and then to work on it without error if it does not exist .

JavaScript does not have an XOR (exclusive OR) but it is easily replaceable by a function.

	&& (AND)	|| (OR)	XOR
True & True	True	True	False
True & False	False	True	True
False & True	False	True	True
False & False	False	False	False

Exercise: Create a condition with two parameters that replace XOR.

```
vara =true;              // Test 4 times avec true, true, false, false
varb =true;              //                      true, false, false, true

if(...){
    console.log('XOR is true');
}else{
    console.log('XOR is false');
};
```

Loops

```
// Loop FOR
for(vari =0;i <10;i++){...};
// Optimized loop FOR
for(vari =0,len =myArray.length;i <len;i++){...};
// Loop FOR IN To iterate the properties of an object
for(prop inobj){
```


```
        /    prop
        /    obj[prop]
};
// Loop
vari =0;

while(i <10){
          // ...
           i++; // Without this increment, the loop is infinite
};


// Loop executed at least once

do{
        // ...
        i++;
} while(i <10);
```

Browsers all have a maximum execution time for JavaScript.

Break, continue & label

Break and continue give control over the iterations of a loop.

Label, it specifies a place in the script to which it is possible to point. It must be declared before break or continuous.

- **break**: leaves the loop
  - break label: can output from any block of code (multiple loops)
- **continuous**: passes the current iteration of the loop

Exercise: Create a bulleted list that scans three times the contents of an array (with a short explanation of getElementById & innerHTML).

Try catch

A try catch block can 'test' a bit of code and recover the error it could generate.

```
try{
// A block of code that could generate an error.
}
catch(e){
// Code block to handle possible error e.
}
finally{
// Block of code that will be executed independently of the result of the
// try catch (return or other throw exception)
}
```

The reserved word **throw** allows to throw an error (including your own type).

```
throw404;
thrownewError("Invalid age");
```

Functions

Functions have many advantages:

- Grouping of code
- Improves legibility
- Reuse & refactoring
    - Generalization (same code, different values)

```
functionsum(arg1,arg2){returnarg1 +arg2 };
```

Make a habit of systematically performing functions that always return something and whose data type is consistent.

Excess arguments are ignored and undeclared arguments have 'undefined' as a value; do your own validation tests in the function via the **arguments** object which is globally similar to an array (.length and index via [...]).

JavaScript Function Syntax

A JavaScript **function** is defined with the function keyword, followed by a **name**, followed by parentheses **()**.
Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
The parentheses may include parameter names separated by commas:
(**parameter1, parameter2**, ...)

The code to be executed, by the function, is placed inside curly brackets: **{}**

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

Function **parameters** are the **names** listed in the function definition.
Function **arguments** are the real **values** received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variables.

Function Invocation and Return

Function Invocation

The code inside the function will execute when "something" **invokes** (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Return

When JavaScript reaches a **return statement**, the function will stop executing. If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement. Functions often compute a **return value**. The return value is "returned" back to the "caller":

Example: Calculate the product of two numbers, and return the result:


```
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
}
```

The result in x will be: 12

Call and function reference

A variable can, of course, reference a function via its name. The call is done via parentheses () and possible arguments.

High order function is any function that accepts one or more other functions as a parameter or returns another function.

Anonymous Functions

```
varmyFunction =function(message){ alert(message); }; myFunction('This is a test'); // Displays : This is a test
```

Exercise: Create a factorial function(n){…} which returns the factorial of n.

Closure

The subject is specific to JS and too broad for an introduction but remember at least that when you see the word function in another function, the internal function has access to the variables of the external function (much like a 'regional' scope between The local and the global).

Example http://www.w3schools.com/js/js_function_closures.asp

Objects

Unlike String, Number, or Boolean variables, objects can contain multiple values as **pairs name: value**.

```
varx1 ={};// New object

varperson ={firstName: "David", lastName: "
                    Collignon", age: 39
                };

console.log(person.firstName); console.
log(person['firstName']);
```

Anonymous Objects

An object, like other types of data, does not necessarily need to be named. This is often the case for a configuration object used as a parameter of a class.

```
$('.bxslider').bxSlider({mode:'fade', captions:true});
```

Common objects

Many useful objects are directly available: document, window, Math, etc.

- Object: eg document
  - Property: for example .innerHTML or .textContent
  - Method: for example .getElementById()
  - Keyword: this

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

The keyword this returns the object that 'owns' the piece of code, so in an object it will be the object itself. This can be discussed in detail in jQuery.

Assignment by value and by reference

In JavaScript, complex data types (array & object) are assigned by reference and not by value. Depending on your needs, you will need to code a script to copy your table or object (Deep copy).

// Short examples of Deep copy JSON.parse(JSON.stringify(obj))// only if there is no fn varnewObject =jQuery.extend(true,{},oldObject); varnewArray =jQuery.extend(true,[],oldArray);

The DOM

What is the DOM?

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

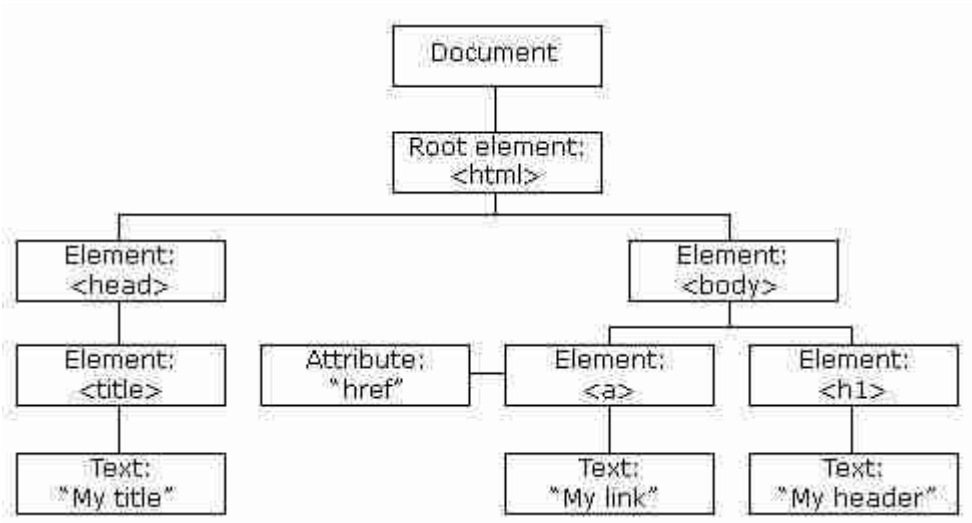
The DOM defines a standard for accessing documents:

"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."

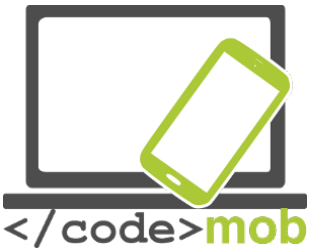
The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

JavaScript allows you to browse, read and modify the Document Object Model (DOM)



Exercise: Convert the diagram above into HTML page.



The document object and select a portion of the HTML

The document object represents the entire HTML document (root) when it is loaded by the browser window. It proposes many methods to target a part like for example:

```
varnode1 =document.getElementById("demo"); // #demo varnode2
=document.querySelector("p"); // The first p found
```

GetElementByID and querySelector respectively return the corresponding id and only the first occurrence found or **null** in the opposite cases.

```
varnodeList1 =document.getElementsByClassName("demo"); // .demo varnodeList2 =
document.getElementsByTagName("p");// All the p varnodeList3 =document.querySelectorAll(
"p"); // All the p
```

These methods return a list of nodes (Node list) that has the **.length** property whose indexes are accessed via the brackets **[]**, so it can be traversed through a for loop. Nevertheless, this is not an array.

Keep in mind that for complex CSS selectors the .querySelector method is still not better than specific libraries like jQuery neither in terms of performance nor in terms of ease of use. Moreover, since its conception, some people (like John Resig) have criticized its implementation.

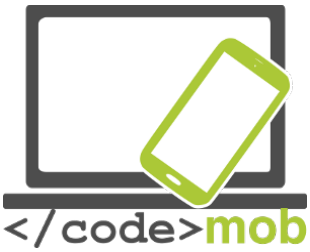
Read & change HTML

What is the HTML DOM?

Here is a great explanation of it:

You can quickly change the content and attributes of an HTML tag with .innerHTML (in Read & Write) and a Getter/Setter for attributes.

```
// Get & Set
varhref =document.getElementById("myLink").getAttribute("href"); document.getElementById("
myLink").setAttribute("href",newValue); // Tests the existence
```




```
varhasH =document.getElementById("myLink").hasAttribute("href"); // Deletion document.
getElementById("myLink").removeAttribute("href");

// Inject HTML document.getElementById("demo").innerHTML ="New text"; // Add text
content document.getElementById("demo").textContent ="New text";
```

Note the difference between .innerHTML and .textContent, the first is parsed and thus accepts HTML and the second is not. **TextContent** will be *faster* and more **secure**.

Change CSS

All CSS properties are accessible via the syntax pointed under the style property with the following features:

- Properties are written in Lower Camel Case (fontsize → fontSize)
- Values are always Strings (for example, not 250 but "250px")
- The new value will be added in **inline style**!
- Position is by .offsetTop and .offsetLeft (neither right nor bottom)
- Total widths and heights (padding, border) via .offsetWidth and .offsetHeigth

```
varelement =document.getElementById("demo"); element.style.backgroundColor ="
green";// camel Cased element.style["backgroundcolor"]="green"; // standard CSS

varelement =document.getElementById("demo"); element.className =element.className
+"otherclass";// space
```

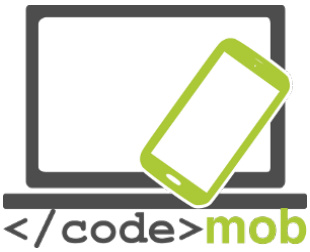
JavaScript usually retrieves the inline style. If you want the resulting CSS styles from an external style sheet or the <style> tag, use .getComputedStyle():

```
varstyle =window.getComputedStyle(document.querySelector('nav'),null); var
bgc =style.getPropertyValue('backgroundcolor');//rgb(255, 191, 0)
```

Change the DOM

Some methods to create and insert HTML tags in the DOM.

- **document.createElement("htmlTag")** Creates an HTML tag



- **document.createTextNode("Hello world")** Creates text content
- **.removeChild(childToBeRemoved)**Removes a child element
- **.appendChild(newContent)**Adds a new element to the end of the parent
- **.insertBefore(newElement, currentElement)**Adds a new element
- **.replaceChild(newElement, currentElement)**Replaces an element

Some useful methods to browse the DOM.

- **.parentNode**Targets the parent node
- **.children[index]** Targets children through an index starting at 0
- **.nextElementSibling** Target the next node 'sister' (who has the same parent)
  - Do not confuse with .nextSibling which also returns spaces between nodes and comments!
- **.previousElementSibling** Targets the previous node 'sister'
- **…**

Try it yourself!

JavaScript HTML DOM Elements:

Changing HTML Content: http://JavaScript HTML DOM - Changing HTML

Changing CSS:

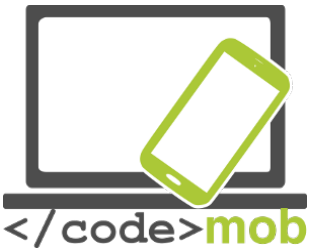
Full list of properties and methods

Examples W3Schools [1][3]

The Screen object

The screen object provides information about the user's screen. This information is **not accessible** by a **server-side** scripting language.

- **screen.width**& **screen.height**
  - **screen.availHeight**& **screen.availWidth**(Windows taskbar less)
- **screen.colorDepth**

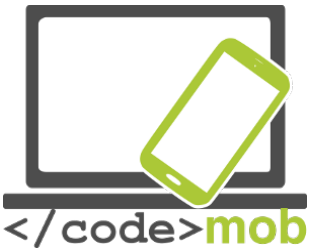


The Navigator object

The navigator object provides information about the user's browser.

- **navigator.userAgent** The full name of the browser
- **navigator.platform**The user’s OS
- **navigator.onLine**If the user is online or offline
- **navigator.language**The interface language of the browser
- **navigator.geolocation** Geolocation after agreement of the user
- **navigator.cookieEnabled**If the user accepts cookies
- ...

Example http://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_nav_geolocation



Events

It is possible to link any event (click, double click, hover, error, resize, ...) to an element of the DOM via the following syntax:

element.addEventListener(event, function, useCapture);

```
document.getElementById("myBtn").addEventListener("click", doStuff);

function doStuff(ev){...}
```

The name of the event parameter is a string that ignores the 'on' of HTML attributes (example: <a href="" onClick="…" /> → click).

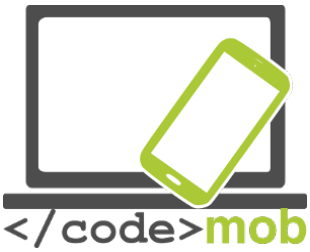
Complete list on W3Schools:

Basically, HTML events are "things" that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can "react" on these events.
Events are actions or occurrences that happen in the system you are programming, which the system tells you about so you can respond to them in some way if desired. For example if the user clicks a button on a webpage, you might want to respond to that action by displaying an information box.

In the case of the Web, events are fired inside the browser window, and tend to be attached to a specific item that resides in it — this might be a single element, set of elements, the HTML document loaded in the current tab, or the entire browser window. There are **a lot of different types of event** that can occur, for example:

- The user clicking the mouse over a certain element, or hovering the cursor over a certain element.
- The user pressing a key on the keyboard.
- The user resizing or closing the browser window.



- A web page finishing loading.
- A form being submitted.
- A video being played, or paused, or finishing play.
- An error occuring.

Each available event has an **event handler**, which is a block of code usually defined by the developer that will be run when the event fires. When such a block of code is defined to be run in reponse to an event firing, we say we are **registering an event handler**. Note that event handlers are sometimes called **event listeners** — they are pretty much interchangeable for our purposes, although strictly speaking they work together. The listener listens out for the event happening, and the handler is the code that is run in response to it happening.
A simple example

In the following example, we have a single , which when pressed, will make the background change to a random color:

```
<button>Change color</button>
```

The JavaScript looks like so:

```
var btn = document.querySelector('button');

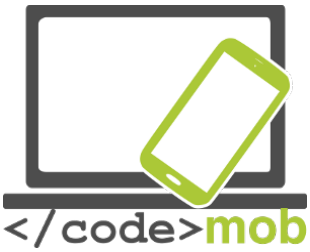
function random(number) {
  return Math.floor(Math.random()*number);
}

btn.onclick = function() {
  var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255)
  document.body.style.backgroundColor = rndCol;
}
```

In this code, we store a reference to the button inside a variable called btn, using the function. We also define a function that returns a random number. The third part of the code is the event handler. The btn variable points to a <button> element, and this type of object has a number of events that can fire on it, and therefore, event handlers available. We are listening for the click event firing, by setting the onclick event handler property to equal an anonymous function containing code that generated a random RGB color and sets the <body> background-color equal to it.

This code will now be run whenever the click event fires on the <button> element, i.e., whenever a user clicks on it.

Code and result:



Event handler properties

Returning to the previous example:

```
var btn = document.querySelector('button');

btn.onclick = function() {
  var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(25
  document.body.style.backgroundColor = rndCol;
}
```

The property is the event handler property being used in this situation. It is essentially a property just like any other one available on the button (e.g. , or), but it is a special type — when you set it to be equal to some code, that code will be run when the event fires on the button.

Event Management

The Event object represents any DOM event. As it is passed in argument, we can use it in our function that handles the event.

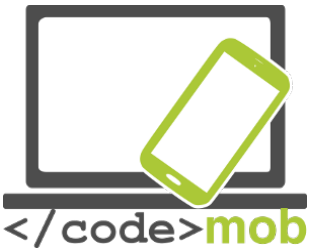
It has many properties and methods ...

- **event.target**The object that issued the event
- **event.currentTarget**Returns the element to which the event was attached
- **event.preventDefault()**Cancels normal behavior
- **event.stopPropagation()**Cancels the propagation of the event

Capturing & Bubbling

The **capture** is the **descending** phase (from the outer element to the inner element) and the **bubbling** the **ascending** phase (Like a fisherman who dives and ascends).

The default value of useCapture is false and IE9 only supports the bubbling event.



Remove event handlers

Just use .removeEventListener () but it is not possible to remove anonymous functions. In addition, each event handler must be removed separately, one event in bubbling and one event in capturing on the same event are two different eventListener.

```
vare =document.getElementById("myDIV") e.addEventListener("
mousemove",myFunction); e.removeEventListener("mousemove",
myFunction);
```

Quizz

Here is a little quiz you should be able to pass now 🕹
You will find the answers at the end of it. Try to not cheat!

Questions

Introduction

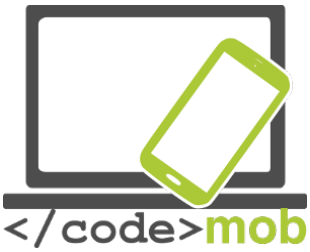
1. Whenever your program isn't working, the first thing to do is always to look at the error messages. How do you open the web console in both Firefox and Chrome ?

- F12 (Chrome and Firefox)
- Open Menu > Developer > Web Console (Firefox)
  Open Menu > More Tools > Developer Tools > Console Tab (Chrome)
- Ctrl + Shift + K (Firefox)
  Ctrl + Shift + I (Chrome)
- Ctrl + C (Chrome and Firefox)

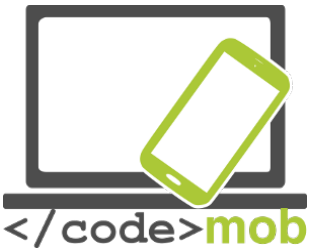
2. You can declare a variable both with and without the var keyword. Which way should you use ?

- Both, there isn't any difference.
- A variable declared with the 'var' keyword belongs to the local scope, the other belongs to the global scope. Always use var to avoid polluting the global scope.
- A variable declared with the 'var' keyword belongs to the global scope, the other belongs to the local scope. Never use var to avoid polluting the global scope.

3. Why should you always write comments ?



- It's critical to be able to explain some finer points of a software program. Both for yourself if you're working again on an old project or when you're part of a team.
- Comments are a waste of time, nobody write them anyway.
- It's a convenient way of temporary disabling some part of a code without losing it.



Conditions

1. Is there some logical difference between one 'if else' structure and two consecutive 'if' having opposite condition ?

- No but the 'if else' way is far better because it's both more readable and less error prone.
- Yes, they aren't the same.

2. How can you loop through each and every item of an array of unknown length ?

- Through the use of the array property .length.
- Through the use of the array method .length().
- Through the use of the function count().

3. *Which improvement the following code*
for (var i = 0, len = a.length; i < len, i++) { /* ... */ }
brings compared to
for (var i = 0; i < a.length, i++) { /* ... */ }
?

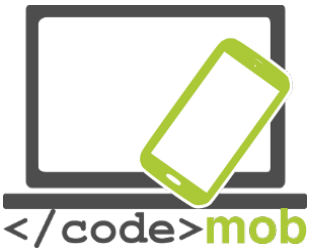
- None
- The variable 'len' is only evaluated once instead of once per iteration.
- The number of iteration is different.

Functions

1. Is the order of the arguments of a function called 'divide' (which return the result of a division of the first argument by the second one) important ?

- Yes, it change the result.
- No

2. What is the difference between a 'function call' and a 'function reference' ?



- A 'function call' (its name followed by parenthesis) ask the function code execution right at the moment of the call. A 'function reference' (in a call back for example) simply store the function name for future reference.
- There isn't any difference if the function hasn't any argument.

3. Which reserved keyword is used to specify which value is returned by a function call ?

- Return
- Pass
- Exit

DOM

1. Which methods do not refer to the first or only occurrence found?

- document.querySelector()
- document.querySelectorAll()
- document.getElementById()
- document.getElementsByTagName()
- document.getElementsByClassName
- document.getElementsByName()

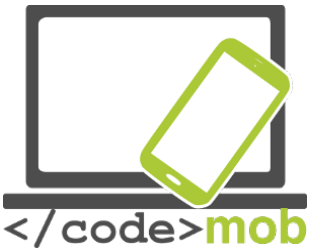
2. In the following code, which argument is received by the callback ?
document.getElementById('myID').addEventListener('click', myCallBack);

- The event which activated the callback.
- No argument was passed through the callback.

3. How can you get the value of an HTML control (UI) such as the input tag ?

- Via the property .value which return the content of the 'value' attribute.
- Via the method .getValue() which return the content of the 'value' attribute.
- Via the method ..val() which return the content of the 'value' attribute.

CSS



1. Which CSS selector allow to only target each paragraph which is a direct child of a div ?

- Div > p
- Div p
- Div, p

2. Which CSS selector allow to target only the first paragraph regardless of the number of HTML siblings he has.

- p:nth-of-type(1)
- p:first-child
- p:nth-child(1)

3. If two CSS selectors (.red et #blue) both target the same element, which will be his color ?

- Blue
- Red
- Purple
- It depends ; whichever is declared last.

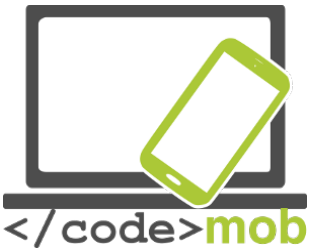
Answers

Introduction

1. Whenever your program isn't working, the first thing to do is always to look at the error messages. How do you open the web console in both Firefox and Chrome ?

- F12 (Chrome and Firefox)
- Open Menu > Developer > Web Console (Firefox)
  Open Menu > More Tools > Developer Tools > Console Tab (Chrome)
- Ctrl + Shift + K (Firefox)
  Ctrl + Shift + I (Chrome)

2. You can declare a variable both with and without the var keyword. Which way should you use ?



- A variable declared with the 'var' keyword belongs to the local scope, the other belongs to the global scope. Always use var to avoid polluting the global scope.

3. Why should you always write comments ?

- It's critical to be able to explain some finer points of a software program. Both for yourself if you're working again on an old project or when you're part of a team.
- It's a convenient way of temporary disabling some part of a code without losing it.

Conditions

1. Is there some logical difference between one 'if else' structure and two consecutive 'if' having opposite condition ?

- No but the 'if else' way is far better because it's both more readable and less error prone.

2. How can you loop through each and every item of an array of unknown length ?

- Through the use of the array property .length.

3. Which improvement the following code
for (var i = 0, len = a.length; i < len, i++) { /* ... */ }
brings compared to
for (var i = 0; i < a.length, i++) { /* ... */ }
?

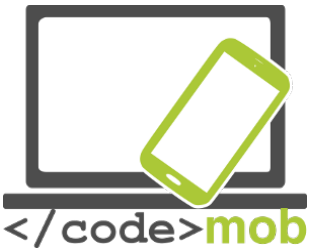
- The variable 'len' is only evaluated once instead of once per iteration.

Functions

1. Is the order of the arguments of a function called 'divide' (which return the result of a division of the first argument by the second one) important ?

- Yes, it change the result.

2. What is the difference between a 'function call' and a 'function reference' ?



- A 'function call' (its name followed by parenthesis) ask the function code execution right at the moment of the call. A 'function reference' (in a call back for example) simply store the function name for future reference.

3. Which reserved keyword is used to specify which value is returned by a function call ?

- Return

DOM

1. Which methods do not refer to the first or only occurrence found?

- document.querySelector()
- document.getElementById()

2. In the following code, which argument is received by the callback ?
document.getElementById('myID').addEventListener('click', myCallBack);

- The event which activated the callback.

3. How can you get the value of an HTML control (UI) such as the input tag ?

- Via the property .value which return the content of the 'value' attribute.

CSS

1. Which CSS selector allow to only target each paragraph which is a direct child of a div ?

- Div > p

2. Which CSS selector allow to target only the first paragraph regardless of the number of HTML siblings he has.

- p:nth-of-type(1)

3. If two CSS selectors (.red et #blue) both target the same element, which will be his color ?

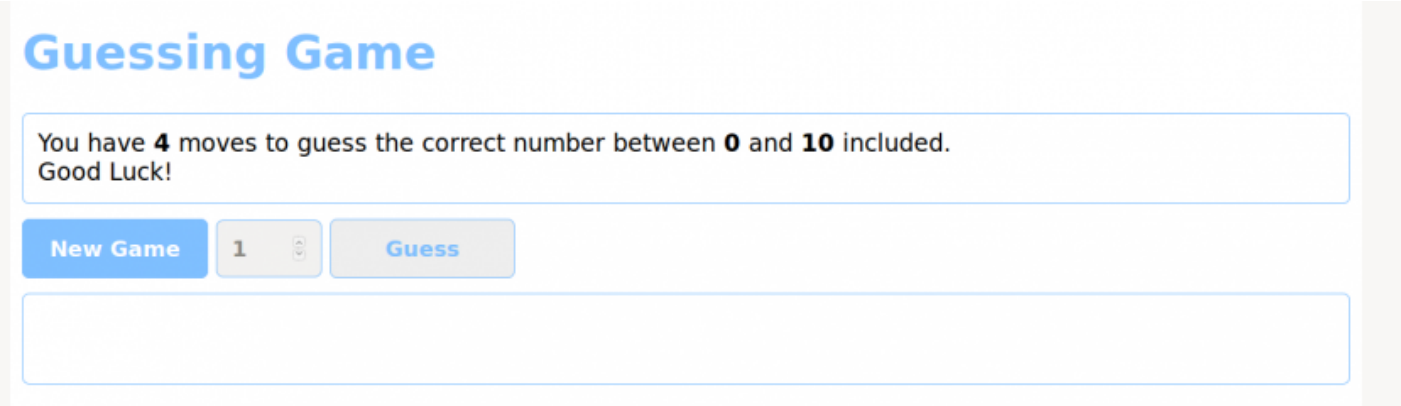
- Blue

Coding lab: Guessing game

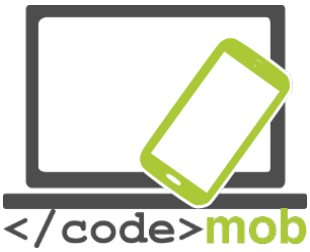
Now that you understand the basics of HTML / CSS / JavaScript, it is time to have fun and make your first game...
In this lesson, you will recreate the Guessing Game. That means that you will be creating:

1. HTML file in which you will put all the elements required by the game
2. CSS file through which you will style your game (any way you like ;-)
3. JavaScript file which will contain all the functions required by the game

This s basically how it should be (not especially look):



Next page, you will find a step-by-step guide to do it, if needed…



Getting started

1. We'll need at least one numeric stepper and a button to test the result, but also some sort of 'container' in which we'll write/log the results even if it start empty or invisible.

Try to think about whether or not the 'New game' button is needed both as a user and as a coder standpoint and whether it is useful or not. If you want, you may at the end of the exercise change the code to allow a user to abandon the current game for example.

2. Try to think about all the steps needed to code this mini game. This will be the start of your different blocks of code. If a problem seems hard to solve, you'll most likely need to once again cut it into smaller chunks.

If you need to reuse multiple time the same code/logic, it's best to turn it into a function especially if the starting parameters might change.

Always, try to use variables instead of hard-coded values so that you can easily change the game parameters like the number of available guesses for example. It improve readability, maintainability and make your code more configurable.

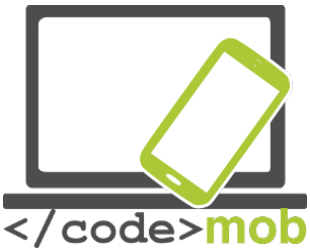
Always comment your code (by explaining the logic behind the code and not the code) for you, your future you ;) and your teammates.

3. getRandomIntegerBetween() is a simple example of a function that improve both readability and turn a piece of code into something easily reusable multiple time.

Take into account that true randomness is not something trivial in computer sciences. Computers are deterministic, which means that if you ask the same question you'll get the same answer every time. In fact, such machines are specifically and carefully programmed to eliminate randomness in results. See pseudo random in Google.

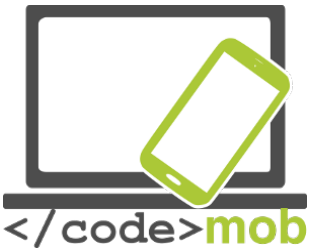
In checkResponse(), we see that return can also be used to exit a function without returning something, bypassing the useless 'game over' test when the game is already won.

Finally we see that web technologies like JavaScript are all about accessing and updating the HTML DOM (Document Object Model) with things like



document.getElementById('userGuess') and resultP.innerHTML or the CSS. Each of those relatively easy technologies work in tandem.

Next page, some clues for the structure and functions…



STRUCTURE AND FUNCTIONS

```
// Return an integer between min and max included
    function getRandomIntegerBetween(min, max)

// Activate User Interface for new game
    function activateUI()

// Game over: Deactivate User Interface
    function deactivateUI()

    function init(){
      // Reset to maximum
      // Clear logs
      // Get a new random number
      // Cheet ;)
      activateUI();
    }

// check the answer
function checkResponse()

// Number of available try for the player before game over

    // Random number to be guessed by the player

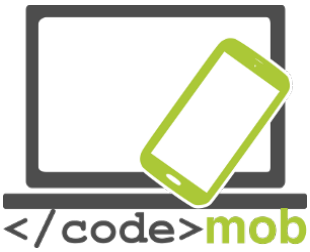
    // Minimum and maximum values (included) to choose from

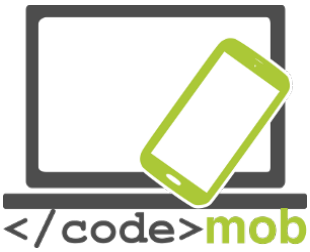
    // Display the rules

    // Player log

    // UI

    // Can you guess the optimal way to always win this game ?
```


</code>mob



References & Resources

JAVASCRIPT

[EN] JavaScript

– -

[EN] JavaScript Style Guide

JavaScript for the Total Non-Programmer

Defer & async [EN]

-

Modernizr

-

What is the function of the var keyword and when to use it ?

[EN] The infamous 'this' keyword

-

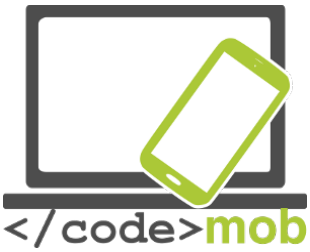
[EN] ECMA 6 The future of JavaScript

http://es6-features.org/#Constants

The JavaScript Source is an excellent JavaScript resource with tons of "cut and paste" **JavaScript examples for your Web pages**. All for free!

HTML & CSS

HTML5 Tutorial



CSS3 Tutorial

And so much more on Google, YouTube and the whole web!

Thanks to David Collignon, for the precious help and course notes.