</code>mob

Programació

CodeMob: Programació.
Octobre 2017. http://codemob.eu/**. Generat per:**

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

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

Índex de continguts

Benvinguts/des a aquest curs de programació!

Al final d'aquest curs, hauries de ser capaç de:

- construir un joc (per exemple, el memory) en JavaScript
- afegir un GUI (interfície d'usuari bàsica) amb HTML5 & CSS3
- entendre i explicar els conceptes bàsics de la programació.

En primer lloc, tingues en compte que hi ha milers de recursos en línia. No seria possible proposar un curs complet aquí. No obstant això, en la programació, trobaràs totes les respostes gràcies a una simple cerca a la web. Considera aquest curs com una introducció, pas a pas, en JavaScript. A gaudir!

Per què aprenem JavaScript?

No s'ha de confondre amb Java: JavaScript permet crear llocs web interactius. JavaScript s'ha convertit en una tecnologia web essencial juntament amb HTML i CSS, ja que la majoria dels navegadors implementen JavaScript. Per tant, és important aprendre JavaScript si es vol entrar en l'àmbit del desenvolupament web, i s'ha d'aprendre bé si estem pensant en convertir-nos en un desenvolupador de front-end o en l'ús de JavaScript per al desenvolupament de back-end.

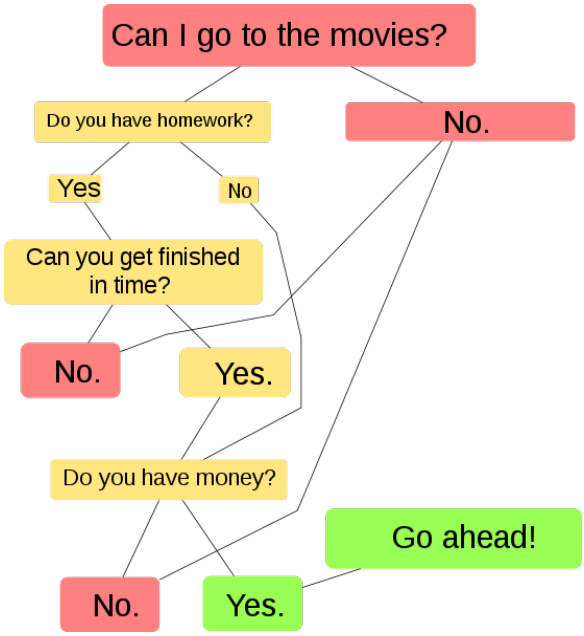
A més, l'ús de JavaScript s'ha estès ara al desenvolupament d'aplicacions per a dispositius mòbils, al desenvolupament d'aplicacions d'escriptori i al desenvolupament de jocs. En general, s'ha fet popular i ara treballar amb JavaScript és una habilitat que resulta molt interessant adquirir.

Font: http://www.bestprogramminglanguagefor.me/why-learn-javascript

Algoritmes

En matemàtiques i ciències de la computació, un algoritme (ælgərɪðəm / A-gə-ri-dhəm) és un conjunt d'operacions autònomes pas a pas. Els algoritmes realitzen tasques de càlcul, processament de dades i/o raonament automatitzat.

Aquí tens un exemple simple del que un algoritme pot ser:



Quina és la relació entre un algoritme i un programa? Mira un algoritme com una fórmula per treballar alguna cosa. Un programa és una sèrie d'instruccions per a l'equip que utilitza algoritmes per executar la tasca desitjada.

Intentem entendre això a través d'un exemple:

Algoritme simple:

Àrea del rectangle A = longitud per profunditat

El programa per treballar l'àrea seria:

Demanar a l'usuari que introdueixi la longitud
Demanar a l'usuari que introdueixi la profunditat
Utilitza l'algoritme d'àrea per calcular l'àrea
Mostra resposta

Introducció a HTML i CSS

Per crear un lloc web, has de proporcionar informació a l'equip. No n'hi ha prou amb escriure el text que estarà al seu lloc, també cal saber com col·locar aquest text, inserir imatges, fer enllaços, etc. Per explicar-li a l'ordinador el que vol, serà necessari utilitzar un llenguatge que entengui.

Hi ha llenguatges utilitzats per crear programes, com C ++ o Java. No obstant això, aquests llenguatges són complexos i estan destinats a persones que ja tenen algun coneixement informàtic.

HTML i **CSS** s'utilitzen precisament per crear llocs web, i s'han creat per ser fàcils d'utilitzar. Cada un d'aquests 2 llenguatges s'utilitza per fer alguna cosa específica, i els dos es complementen naturalment per finalment donar un lloc web:

• **HTML**: És l'abreviatura de HyperText Markup Language que va aparèixer el 1991 quan es va llançar la web. La seva funció és gestionar i organitzar el contingut. Per tant, en HTML s'escriu el que s'ha de mostrar a la pàgina: text, enllaços, imatges... Per exemple: aquest és el meu títol, aquest és el meu menú, aquí hi ha el text principal de la pàgina...
En aquest curs treballarem en l'última versió d'HTML (HTML5), que és avui el llenguatge del futur.

• **CSS**: Aquesta és l'abreviatura de **CASCADING STYLE SHEETS**. Aquest llenguatge només serveix per dissenyar la pàgina web. És en CSS que diu: "Els meus títols estan en vermell i estan subratllats, el meu text està en font ARIAL, el meu nom està centrat, el meu menú té un fons blanc…". Amb aquest llenguatge, podrem crear ràpidament i senzillament el disseny d'un lloc web.

L'editor

Una pregunta que t'has de fer és: "Quin programa necessito per crear el meu lloc web?"

Notepad és suficient per crear un lloc web! Però per facilitar el nostre treball

podem utilitzar un editor de codi com Notepad ++ (l'avantatge és que automàticament acoloreix el codi HTML / CSS per fer-lo més llegible).

Navegadors

Què és un navegador?

El navegador és el programa que et permet veure els llocs web. El treball del navegador és llegir el codi HTML / CSS que s'ha escrit i mostrar el que representa. Si el teu codi CSS diu "els títols són vermells", llavors el navegador mostrarà els títols en vermell.

Entre els navegadors que existeixen, aquí hi ha els principals:
Internet Explorer - Mozilla Firefox - Opera - Netscape - Konqueror (per a Linux) - Lynx (per a Linux) - Apple Safari (per a Mac) - etc.

Crea el teu primer document HTML

Què es HTML?

Com hem esmentat anteriorment, HTML és un llenguatge de marcat per descriure documents web (pàgines web).

- HTML vol dir Hyper Text Markup Language
- Un llenguatge de marcat és un conjunt d'etiquetes de marcat
- Els documents HTML es descriuen mitjançant etiquetes HTML
- Cada etiqueta HTML descriu diferents continguts de documents

Un petit document HTML
Exemple

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

<h1>El primer encapçalament</h1>
<p>El primer paràgraf.</p>

</body>
</html>
```

Exemple explicat

- La declaració <! DOCTYPE html> defineix aquest document com HTML5
  El text entre <html> i </html> descriu un document HTML
  El text entre <head> i </head> proporciona informació sobre el document
  El text entre <title> i </title> proporciona un títol per al document
  El text entre <body> i </body> descriu el contingut visible de la pàgina
  El text entre <h1> i </h1> descriu un encapçalament
  El text entre i descriu un paràgraf
  Amb aquesta descripció, un navegador web mostrarà un document amb una capçalera i un paràgraf.

  Per a més informació, mira el document complet en w3schools

Elements HTML que necessites

En aquesta lliçó, la teva tasca és crear un document HTML simple que contingui el següent contingut:

- LinkTitle
- Text paragraph
- Image
- Link

No t'oblidis de fer servir els comentaris. Un comentari és una etiqueta HTML amb una forma molt especial:

<! - Això és un comentari ->

El pots posar on vulguis en el teu codi font: no té cap impacte a la teva pàgina, però pots utilitzar comentaris per explicar el teu codi, el que pot ajudar-te quan editis el codi font en una data posterior. Això és especialment útil si tens un munt de codi!

Atenció: tots poden veure el codi HTML de la teva pàgina una vegada que puja al web. Simplement fas clic amb el botó secundari a la pàgina i selecciones "Mostrar el codi font de la pàgina". Per tant, cal evitar posar informació sensible com contrasenyes en els comentaris... i cal cuidar la qualitat del teu codi font.

Crea un arxiu CSS i utilitza'l per dissenyar el teu document HTML

Què és CSS?

- CSS significa fulls d'estil en cascada.
- CSS descriu com es mostren els elements HTML a la pantalla, en el paper o en altres mitjans.
- CSS estalvia molta feina. Pots controlar el disseny de diverses pàgines web d'una sola vegada.
- Els fulls d'estil externs s'emmagatzemen en arxius CSS.

Per què utilitzar CSS?

CSS s'utilitza per definir estils per a les seves pàgines web, incloent-hi el disseny, la disposició i les variacions en la pantalla per diferents dispositius i mides de pantalla.

CSS va resoldre un gran problema

HTML mai va ser dissenyat per contenir etiquetes per al format d'una pàgina web. En canvi, HTML va ser creat per descriure el contingut d'una pàgina web, com:

<H1> Aquest és un encapçalament </ h1>

<P> Aquest és un paràgraf. </ P>

Quan es van agregar etiquetes com <font>, i atributs de color a l'especificació HTML 3.2, va començar un malson per als desenvolupadors web. El desenvolupament de llocs web grans, on es van agregar fonts i informació de color a cada pàgina, es va convertir en un procés llarg i costós.

Per solucionar aquest problema, el World Wide Web Consortium (W3C) va crear CSS.

CSS elimina l'estil de format de la pàgina HTML!

CSS estalvia molta feina!

Les definicions d'estil es guarden normalment en arxius .css externs.
Amb un arxiu de full d'estil extern, pot canviar l'aspecte d'un lloc web sencer mitjançant la modificació d'un sol arxiu.

Per a més informació, consulta .

Full d'estil extern

Cuando un navegador lee una hoja de estilo, formatea el documento HTML de acuerdo con la información de la hoja de estilo.

Quan un navegador llegeix un full d'estil, formata el document HTML d'acord amb la informació del full d'estil:

1. Full d'estil extern
2. Full d'estil intern
3. Estil en línia

En aquesta lliçó anem a inserir un **full d'estil extern**.

Amb un full d'estil extern, pots canviar l'aspecte d'un lloc web complet canviant només un arxiu!

Cada pàgina ha d'incloure una referència a l'arxiu de full d'estil extern dins de l'element <link>. L'element <link> va dins de la secció <head>:

Exemple:

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

Un full d'estil extern es pot escriure en qualsevol editor de text. L'arxiu no ha de contenir cap etiqueta html. L'arxiu de full d'estil s'ha de guardar amb una extensió .css.

Així és com es veu el "myStyle.css":

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

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

Nota: No hi afegiu un espai entre el valor de la propietat i la unitat (com margin-left: 20 px;). La forma correcta és: margin-left: 20px;

Programació JS

Història del JavaScript

El llenguatge JavaScript va ser creat el **1995** per **Brendan Eich** (Fundació Mozilla) per Netscape. El llenguatge, actualment en la versió 1.8.5, és una implementació de la tercera versió de la norma ECMA262.

- Al desembre de 1995, Sun i Netscape van anunciar el llançament de JavaScript. Microsoft reacciona llavors desenvolupant JScript (la mateixa llengua per a un nom diferent per evitar una disputa de marques amb Sun).
- Netscape envia JavaScript a ECMA International per estandardització al novembre de 1996.
- **DHTML** (1996), un començament difícil i innecessari
- **Ajax** (2000), un interès renovat gràcies a les noves possibilitats
- **Json**: una alternativa al XML basat en JavaScript
- **Marcs de JavaScript**: facilitat, productivitat i estandardització, però difícil elecció
- N**ODE.JS**; Eventdriven & Serversided
  - NodeWebkit & Cordova

Què és JavaScript?

● Un llenguatge de seqüència (llenguatge d'alt nivell).

  ○ Interpretat (en contraposició als idiomes compilats).

● Un llenguatge del costat del client (s'executa en la màquina de l'usuari i no al servidor).

  ○ No obstant això NODE.JS va aparèixer el 2009 i va canviar una mica la situació.

● Capaç de canviar el DOM:

- Crear, editar i eliminar elements HTML, els seus atributs o CSS.
- Crear, escoltar i esborrar esdeveniments.

● No té res a veure amb el llenguatge Java de Sun... ;)

Exemple: & the Polyfills

Instal·lació

Els scripts poden col·locar-se en una pàgina HTML, ja sigui en el <head> o en <Body> (just abans del </ body>) i poden ser interns o externs. L'atribut type és opcional perquè el seu valor per omissió és correcte (i Javascript és l'únic idioma acceptat en 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>
```

Exemple JavaScript

Intentarem fer el nostre primer arxiu JavaScript. En aquest exemple crearem un quadre d'alerta simple. Estarem fent això enllaçant un arxiu .js extern al nostre document .html.

El primer que has de fer és anar a la carpeta en què has creat el teu primer arxiu Html i arxiu .css. Crea un nou arxiu .js i guarda a la mateixa carpeta que (per exemple) myscript.js.

Ara afegeix la següent seqüència d'ordres a l'arxiu .js:

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

Finalment, vincula el myscript.js al teu document .html:

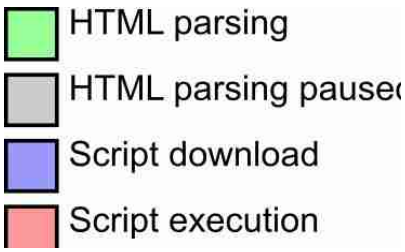
```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

Pots posar una referència de guió externa en <head> o <body> segons prefereixis. L'script es comportarà com si estigués situat exactament on es troba l'etiqueta <script>.

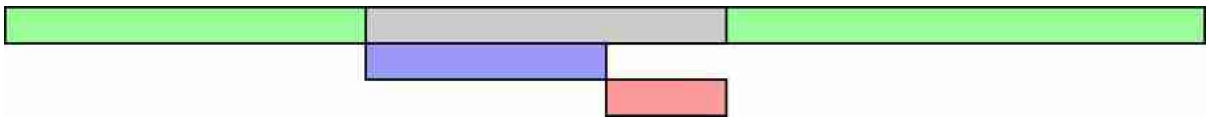
Ordre d'execució de seqüències - Defer & async

El temps d'un script depèn de la seva posició a la pàgina HTML i els atributs d'ajornament i asincronisme.

Els atributs d'ajornament i asincronisme només es tenen en compte per als scripts externs.

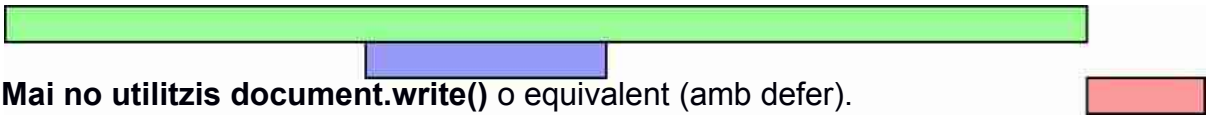


Normal execution



Ús comú: si dues seqüències d'ordres se segueixen, la segona mai s'executarà abans del final de la primera perquè qualsevol recurs està 'bloquejant'.

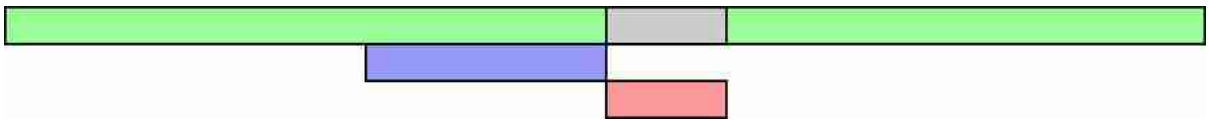
Defer



Mai no utilitzis document.write() o equivalent (amb defer).

Presta sempre atenció a l'ordre d'execució dels scripts que tenen la mateixa prioritat; se suposa que ha de ser respectat, però tenim els errors en IE4-9.

Async (HTML 5)



Ideal per un script, el temps d'execució del qual no és important (per exemple: Google Analytics), però l'ordre d'execució dels scripts no està garantit!

Mai no utilitzis document.write() o equivalent (amb async).

Trucs per al disseny

Quan estigui habilitat llegeix i reescriu o canvia el DOM, el disseny s'invalida i s'ha de tornar a calcular-se en algun moment en el futur. Els navegadors tracten d'esperar fins al final d'un marc, però si han de fer-ho abans, pot tenir un seriós impacte en el rendiment.

En poques paraules, utilitzar el menys possible les manipulacions i canvis dels DOM, o en el pitjor dels casos, agrupar-los tant com sigui possible.

Si estàs interessat en el tema, pots informar-te sobre requestAnimationFrame i biblioteques com FastDOM ().

La consola JavaScript

És fàcil perdre's escrivint codi JavaScript sense depurador.
La depuració és el procés de provar, trobar i reduir errors en els programes informàtics.

Si un script no funciona (error), poques vegades es veu l'error a simple vista... Per tant, hem de mostrar la consola de JavaScript per veure **els missatges d'error** (arxiu i línia de codi, codi de color, script sobre la marxa).

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

També pots depurar amb **console.log(...)**; el qual accepta en paràmetre el que ha de mostrar-se. També és possible afegir punts d'interrupció (**break points**) o utilitzar el comandament **debugger**.

El mètode console.log()

Si el teu navegador admet la depuració, pots utilitzar console.log() per mostrar valors JavaScript a la finestra del depurador:

Exemple:

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

<h1>My First Web Page</h1>

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

</body>
</html>
```

Establiment de punts d'interrupció (Setting Breakpoints)

A la finestra del depurador, pots establir punts d'interrupció en el codi JavaScript.

A cada punt d'interrupció, JavaScript deixarà d'executar-se et permetrà examinar els valors de JavaScript.

Després d'examinar els valors, pots reprendre l'execució del codi (normalment amb un botó de reproducció).

El depurador Keyword

El **debugger** keyword (depurador Keyword) deté l'execució de JavaScript i crida (si està disponible) a la funció de depuració.

Això té la mateixa funció que establir un punt d'interrupció en el depurador.

Si no hi ha depuració disponible, la instrucció depurador no té cap efecte.

Amb el depurador activat, aquest codi deixarà d'executar-se, abans d'executar la tercera línia.

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

Tingues en compte que l'objecte de consola no forma part dels estàndards, encara que almenys estigui ben implementat per Firefox i Chrome.

Exemple: API Console Firefox

Comentaris

Els comentaris JavaScript es poden utilitzar per **explicar el codi JavaScript** i per fer-lo més llegible.

Els comentaris de JavaScript també es poden utilitzar per **evitar l'execució**, al provar el codi alternatiu.

Comentaris de línia única

Els comentaris d'una sola línia comencen amb //.

Qualsevol text entre // i el final de la línia serà ignorat per JavaScript (no s'executarà).

Aquest exemple utilitza un comentari d'una sola línia abans de cada línia de codi:

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

Aquest exemple utilitza un comentari de línia simple al final de cada línia per explicar el codi:

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

Comentaris multi-línia

Els comentaris de diverses línies comencen amb / * i acaben amb * /.

Qualsevol text entre / * i * / serà ignorat per JavaScript.

Aquest exemple utilitza un comentari de diverses línies (un bloc de comentari) per explicar el codi:

```
/*
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.";
```

Ús de comentaris per evitar l'execució

L'ús de comentaris per evitar l'execució de codi és adequat per a les proves de codi.

Afegir // davant d'una línia de codi canvia les línies de codi d'una línia executable a un comentari.

Aquest exemple utilitza // per evitar l'execució d'una de les línies de codi:

Example

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

Aquest exemple utilitza un bloc de comentari per evitar l'execució de diverses línies:

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

Els comentaris sobre una sola línia comencen amb // i els de múltiples línies estan envoltats per /*...*/. A continuació, un cas una mica més complicat per inserir JavaScript en un document XHTML.

Utilitza sempre els comentaris per comentar sobre la teva lògica, els paràmetres esperats per una funció, que retorna o desactiva un tros de codi sense esborrar-lo.

<script type="text/javascript"> //<![CDATA[var myFirstMessage =' Hello World';console.log(myFirstMessage);function myFirstFunction(){var demo =document.getElementById("demo"); demo.style.color ="#990000";}myFirstFunction(); //]]></script>

```
<script type="text/javascript">
//<![CDATA[
var myFirstMessage ='Hello World';

console.log(myFirstMessage);

function myFirstFunction(){
	var demo =document.getElementById("demo");
	demo.style.color ="#990000";
}

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

Coses que no s'han de fer: Eval & javascript:

No utilitzis (gairebé mai) **eval ()**, ja que permet executar codi arbitrari, representa un risc de seguretat.

La **funció Setinterval ()** que crida al seu primer paràmetre després de cert temps utilitza eval () si rep una cadena com a paràmetre. Per tant, prefereix una funció com un argument.

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

Manté la separació entre HTML i JavaScript (com amb CSS per raons de manteniment i lògica) evitant el document.write (...) i separant el JavaScript del disseny.

```
<!-- 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>
```

Operadors matemàtics

- *, /, +,- (ordre de prioritat i parèntesi)
- % (Mòdul)
- +=, =, *= & /= (x = x + 5;)
    - ++ Increment. Operador unitari. Suma un al seu variable. Si s'utilitza com a prefix ++ x torna el valor de la seva variable després d'afegir-un; si s'utilitza com a sufix x ++ torna el valor de la seva variable abans d'afegir-ne un.
    - -- Decrement. Operador unitari. Resta un a la seva variable. El valor retornat és igual que per a l'operador incremental.

Constants

Similar a les variables excepte la ja declarada, el seu valor no pot ser canviat. Sovint s'utilitzen com a paràmetres predefinits. Per convenció i en gairebé tots els idiomes, els noms de constants s'escriuen en lletres majúscules separades per subratllat. Atenció, no hi ha suport per constants en IE10.

```
// Constant
const MY_FAVORITE_NUMBER =9;

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

Variable & tipus de dades

Les variables JavaScript són un llenguatge d'**escriptura dinàmica** com PHP, utilitzen una **sintaxis pointed** i és **sensible a majúscules i minúscules**, així que has de ser metòdic i rigorós perquè el tipus de variable pot canviar en temps d'execució.

L'ús del Lower Camel Case per a noms de variables (primera lletra de cada paraula, excepte el primer en majúscules, sense espai) es considera la millor pràctica. JavaScript té un recol·lector d'escombraries que destrueix les variables que ja no s'utilitzen (o les estableix com nul·les).

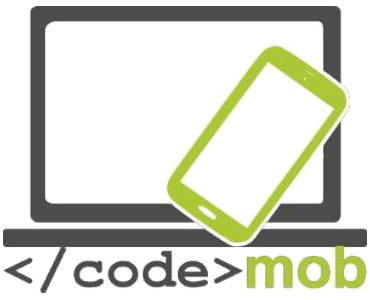
No utilitzis la sintaxi var a = new Array (); o guió en els noms de les variables (per exemple: var causeAnError = 5;).

```
var userIdNumber; // Declaración de la variable con var tag
userIdNumber =5; // Inicialización de la misma variable.

var userIdNumber =5; // Declaración e inicialización simultáneas

var variable1 ="Hello World!";
var variable2 =42;


var variable1 ="Hello World!",
    variable2 =42;          // Multiples declaraciones
```

Tipus de dades

- Primitives
  - **Strings** (caràcters tipus text) : "Hello World"
  - **Numbers** (sencers i nombres com coma flotant) : 3.14 or 42
  - **Booleans** (valors Booleans): true or false (veritable o fals)
  - Undefined (Una variable que no ha estat declarada o que no coincideix amb cap tipus)
  - Null (variable buida, és a dir, a la qual s'ha assignat un valor zero) µ

- **Objetes**
  - **Functions**
  - **Arrays**
  - Date, Math
  - RegExp (expressions regulars)



Funció (canviar el tipus de variable)

És possible canviar el tipus d'una variable (Casting) a una altra mitjançant **Number**(...), **String**(...) i **Boolean**(...) o fins i tot alguns mètodes com .split () o .join ().

```
var nbr2Boo =Boolean(0);                    // falso
var str2Nbr =Number('12');                  // 12
var boo2Str =String(true);                  // cierto
```

Cadenes

● La concatenació de cadenes s'executa a través del signe "+"
● Les cadenes estan delimitades per cometes dobles (double quote) o cometes simples (single quote).

○ A diferència de PHP, no hi ha diferència entre els dos.

○ Es poden inserir caràcters especials a través de \ (caràcter d'escapament)

■ \", \n (tornar a la línia), \\, ...

■ \ en cada volta a la línia

● La longitud és accessible a través de la propietat .length

● Les cadenes tenen molts mètodes;

○ .charAt(); IndexOf(), substr, substring(), toLowerCase(), toUpperCase(), ...

● No accedeixis a una cadena com si fos una matriu (Array)

Exemple:

Nombres

● Només un tipus (Integer & Float) emmagatzemat sota un punt flotant de 64 bits.

● Les constants Infinity & -Infinity representen òbviament l'infinit positiu i el negatiu.

● **NaN** (Not a Number) serà retornat per una impossibilitat matemàtica com una divisió per una cadena, per exemple.

● El mètode **.toString()** els converteix en cadenes

- .toString(2): conversió a binari
- .toString(16): conversió a hexadecimal

● L'objecte **Math** ofreix moltes característiques com:

○ Math.random(): Retorna un nombre aleatori entre 0 inclusivament i 1 exclusiu

○ Math.min(..., ...) & Math.max(..., ...): Mínim i Màxim

○ Math.round(), Math.ceil() & Math.floor(): arrodonit, arrodonit cap amunt i cap avall

○ Main.sin(), Math.cos(), Math.PI, ...

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

Exercici: Mou a l'atzar el nombre 0 o 1 en la consola cada vegada que torni a carregar la pàgina (F5).

Array (Matriu)

● Les matrius són instanciades usant claudàtors buits [] i usant-les per accedir a un índex específic.

```
○ var myArray =[];// Nueva matriz vacía
```

● L'índex comença com en gairebé tots els idiomes de l'ordinador a 0.

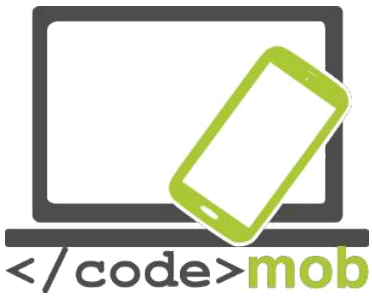
```
○ myArray[0]=42;
```

- És possible trobar la longitud d'una matriu a través de la propietat .length
    - Mai modifiquis una matriu que et li sol·liciti.
- Atenció, els Array no sn matrius associatives.
- Els Array tenen mètodes per a gairebé tot; .push(), .pop(), .splice(…), …
    - És possible ordenar les cadenes de caràcters d'una matriu usant el mètode .sort ()
    - i els dígits a través d'.sort () utilitzant una funció de comparació d'arguments

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

Exemples: W3Schools& www.w3schools.com/js/js_array_methods.asp

Exercici: Mostra l'últim element d'una matriu, la longitud del qual no coneixes.



Provant una Variable Type

Hi ha molts mètodes per provar el tipus d'una variable.

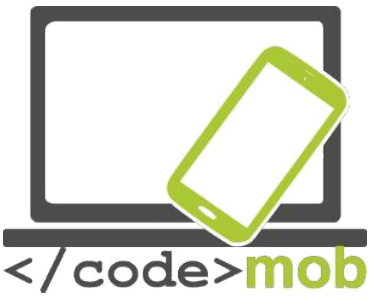
- **typeof** myVariable
    - **typeof** myVar ===**'undefined'**
    - El tipus de dada NaN és un nombre
    - El tipus de dada d'un Array és un objecte.
    - El tipus de dada de Dóna't és un objecte.
    - El tipus de dada Null és un objecte
    - El tipus de dada d'una variable indefinida és un "undefined"

Exemples:

```
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 o no var?

En una **Funció**, una **variable** declarada amb **var** és local i global si es declara sense això. En resum: és recomanable **sempre declarar les variables** (evita sobrecarregar l'àmbit global, ECMA6...).

La seva vida útil és diferent: una variable local es destrueix al final de la funció que el conté (context d'execució actual), una variable global quan es tanca la pàgina HTML.

Hi ha, per descomptat, més subtileses que això, però tingues en compte que hem de fer servir var si declarem una variable sense assignar-la i que només es pot esborrar una variable no declarada (la qual cosa és desaconsellable de totes maneres).

Abast

En JavaScript, els objectes i les funcions també són variables.

L'àmbit és el conjunt de variables, funcions i objectes als quals té accés en un moment donat. Per tant, canvia dins d'una funció (àmbit de la funció).

L'àmbit sempre es remunta des del local al general, és a dir, si JavaScript no troba una variable en el seu àmbit, llavors ho buscarà a Global i finalment generarà un error si no troba res.

Hoisting (utilització de variables abans de ser declarades)

Javascript sempre declara les variables i funcions abans d'executar qualsevol codi: en primer lloc, les mou al principi del bloc al qual pertanyen.

Sempre és **recomanable declarar variables i funcions al començament** del seu abast.

Condicions

Les sentències condicionals s'utilitzen per realitzar **diferents accions basades en diferents condicions.**

Declaracions condicionals

Molt sovint quan s'escriu codi es desitja realitzar diferents accions per a diferents decisions.

Pots utilitzar declaracions condicionals en el teu codi per fer-ho.

En JavaScript tenim les següents declaracions condicionals:

- UUtilitza **if** per especificar un bloc de codi a executar, si una condició especificada és veritable
- Utilitza **else if** per especificar un bloc de codi a executar, si la mateixa condició és falsa
- Utilitza **else** si s'especifica una nova condició per provar, quan la primera condició és falsa
- Utilitza **switch** per especificar molts blocs alternatius de codi a executar

If ... then ... else ...
Observa l'espai entre **else** i **if** (i no **elseif** com en PHP, per exemple).

```
if(money <0){
    status ="I'm broke..."; }
elseif(money <10000){
    status ="Could be worse...";
```

```
}else{

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

Operador condicional

Una escriptura compacta d'un **If ... then**, útil per a declaracions condicionals de variables, per exemple.

```
var admissibleAtl3 =(sex =="F")?true : false;
```

Switch

És una forma senzilla de navegar per moltes opcions.

Recorda que un **switch** fa possible tenir diversos casos apuntant a la mateixa variable de codi i que utilitza una **strict comparison** (comparació estricta) (===).

```
switch (new Date().
        getDay()){

        case 1:
        case 2:
        case 3:
        default

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

        case 4:
        case 5:

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

        case 0:
        case 6:

                text ="Time to play : )";

}
```

Operadors lògics

- == (diferent de =), !=
- ===, !== : Comparació de valor i tipus
- >, >=, <, <=
- && (y)
- || (o)
- ! (no)
    - toggle = !toggle // veritable es converteix en fals i fals es converteix en veritable.

JavaScript, com tots els idiomes, s'atura en el primer false en una condició amb && (i), el que fa possible provar primer l'existència d'una variable i després treballar sense error si no existeix.

JavaScript no té un XOR (OR exclusiu) però és fàcilment reemplaçable per una funció.

	&& (AND)	|| (OR)	XOR
Cert & Cert	Cert	Cert	Fals
Cert & Fals	Fals	Cert	Cert
Fals & Cert	Fals	Cert	Cert
Fals & Fals	Fals	Fals	Fals

Exercici: Crea una condició amb dos paràmetres que reemplacin XOR.

```
var a =true;            // Prova 4 cops sobre cert, cert, fals, fals.
var b =true;            //                    cert, fals, fals, cert

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

```
};
```

Loops (bucles)

```
// Bucle FOR
for(vari =0;i <10;i++){...};
// Bucle optimitzat FOR
for(vari =0,len =myArray.length;i <len;i++){...};
// Bucle FOR IN per iterar les propietats d'un objecte.
for(prop in obj){
        /    prop
        /    obj[prop]
};
// Bucle
vari =0;

while(i <10){
          // ...
           i++; // Sense l'increment, el bucle és infinit.
};


// Bucle executat al menys una vegada.

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

Els navegadors tenen un temps d'execució màxim per a JavaScript.

Break, continue & label

Break i **continue** ofereixen control sobre les iteracions d'un bucle.

Label, especifica un lloc a l'script al qual és possible apuntar. Cal declarar-ho abans de la pausa o continua.

- **break**: deixa el bucle
  - break label: Pot sortir qualsevol bloc de codi (múltiples bucles)
- **continuous**: Passa la iteració actual del bucle

Exercici: Crea una llista amb vinyetes que escanegis tres cops el contingut d'una Array (amb una breu explicació de getElementById & innerHTML).

Try catch

Un bloc try catch pot 'provar' una mica de codi i recuperar l'error que podria generar.

```
try{
// Un bloc de codi pt generar un error.
}
catch(e){
// Bloc de codi per gestionar un error.
}
finally{
// Bloc de codi que serà executat independentment del resultat
// try catch (torna o una altra excepció)
}
```

La palaura **throw** permet llançar un error (incloent el teu propi tipus).

```
throw 404;
thrownew Error("Invalid age");
```

Funcions

Les funcions tenen molts avantatges:

- Agrupació de codi
- Millora la llegibilitat
- Reutilització i refactorització
- Generalització (mateix codi, valors diferents)

```
function sum(arg1,arg2){return arg1 +arg2 };
```

Construeix un hàbit per realitzar sistemàticament funcions que sempre retornen alguna cosa i el tipus de dades és coherent.

Els arguments en excés s'ignoren i els arguments no declarats tenen 'undefined' com un valor; fes les teves pròpies proves de validació a la funció a través de l'objecte **arguments** que és globalment similar a una matriu (.length and index via [...]).

JavaScript Function Syntax (Sintaxis de la funció JavaScript)

Una **funció** JavaScript es defineix amb la paraula clau function, seguida d'un nom seguit de parèntesi().

Els noms de funcions poden contenir lletres, dígits, subratllats i signes de dòlar (les mateixes regles que les variables).

Els parèntesis poden incloure noms de paràmetres separats per comes: (**parameter1, parameter2**, ...)

El codi a executar per la funció es col·loca dins dels parèntesis: **{}**

```
function name(parameter1, parameter2, parameter3) {
    codi a executar
}
```

Els **paràmetres** de funció són els **names (**noms) enumerats en la definició de la funció.
Els arguments de **funció** són els **values (**valors) reals rebuts per la funció quan s'invoca.

Dins de la funció, els arguments (els paràmetres) es comporten com a variables locals.

Funció, Invocació i Retorn

Invocació de les funcions

El codi dins de la funció s'executarà quan invoques o flames a "alguna cosa" **invokes** (invoca o crida) la funció:

- Quan es produeix un esdeveniment (quan un usuari fa clic a un botó)
- Quan s'invoca (es diu) des de codi JavaScript
- Automàticament (auto invocat)

Retorn de funcions

Quan estigui habilitat arriba a una declaració de devolució (**return statement**), la funció deixarà d'executar-se. Si la funció s'ha invocat des d'una sentència, JavaScript "tornarà" per executar el codi després de la instrucció invocació. Les funcions sovint calculen un **return value** (valor de retorn). El valor de retorn es retorna al "cridar l'opció":

Exemple: Calcula el producte de dos nombres i retorna el resultat:

```
var x = myFunction(4, 3);        // la funció és cridada i retorna el valor a la variable x
function myFunction(a, b) {
    return a * b;                // la funció retorna el producte d'a per b
}
```

El resultat d'x serà 12

Call and function reference (Referència de trucada i funció)

Una variable pot, per descomptat, fer referència a una funció a través del seu nom. La trucada es realitza mitjançant parèntesi () i les seves possibles arguments.

High order function (Funció d'ordre superior) és qualsevol funció que accepta una o més funcions com un paràmetre o retorna una altra funció.

Anonymous Functions (funcions anònimes)

```
Var myFunction =function(message){ alert(message); }; myFunction('Això és un test'); // Mostra: això és un test
```

Exercici: Crear una funció factorial (n) {...} que retorni el factorial de n.

Closure (tancament)

El subjecte és específic de JS i massa ampli per a una introducció, però recorda almenys que quan vegis la paraula funció en una altra funció, la funció interna té accés a les variables de la funció externa (com un àmbit "regional" entre el local i el global).

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

Objects (objectes)

A diferència de les variables String, Number o Boolean, els objectes poden contenir diversos valors com **pairs name: value**.

```
var x1 ={};// New object
```

```
var person ={firstName: "David",
                    lastName: "Collignon",
                    age: 39
                  };

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

Anonymous Objects (objectes anònims)

Un objecte, igual que altres tipus de dades, no necessàriament ha de ser nomenat. Això sol ser el cas d'un objecte de configuració utilitzat com a paràmetre d'una classe.

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

Common objects (objectes comuns)

Molts objectes útils estan directament disponibles: document, finestra, Math, etc.

- Object: per exemple document
  - Propietat: per exemple .innerHTML o .textContent
  - Mètode: per exemple .getElementById()
  - Paraula clau: this

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

La paraula clau retorna l'objecte que 'posseeix' la peça de codi, de manera que en un objecte serà el propi objecte. Això es pot discutir en detall en jQuery.

Assignació per valor i per referència

En JavaScript, els tipus de dades complexes (matriu i objecte) s'assignen per referència i no per valor. Depenent de les seves necessitats, hauràs de codificar un script per copiar la seva taula o objecte (Deep copy).

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

The DOM

What is the DOM? (Qué és el DOM?)

El DOM és un estàndard de W3C (World Wide Web Consortium).

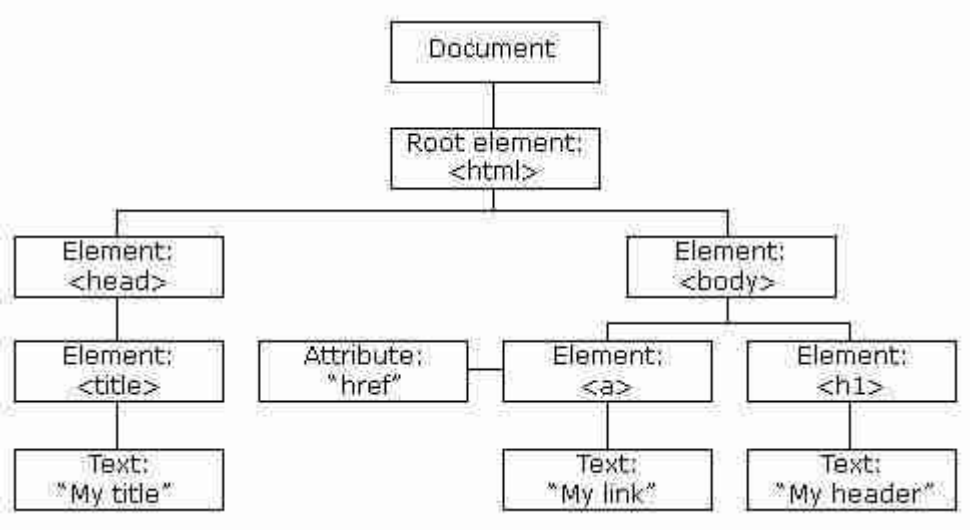
El DOM defineix un estàndard per accedir als documents:

"El Model d'Objectes de Document (DOM) del W3C és una plataforma i una interfície de llenguatge neutral que permet als programes i scripts accedir i actualitzar dinàmicament el contingut, l'estructura i l'estil d'un document".

L'estàndard W3C DOM està dividit en 3 parts diferents:

- Core DOM - model estàndard per a tots els tipus de documents
- XML DOM: model estàndard per a documents XML
- HTML DOM - model estàndard per a documents HTML

JavaScript permet navegar, llegir i modificar el DOM (Document Object Model)



Exercici: Converteix el diagrama anterior en una pàgina HTML.

El document object i la selecció d'una porció d'HTML

L'objecte de document representa tot el document HTML (arrel) quan es carrega per la finestra del navegador. Proposa molts mètodes per apuntar una part com per exemple:

```
var node1 =document.getElementById("demo"); // #demo var node2
=document.querySelector("p"); // The first p found
```

GetElementByID i querySelector respectivament retornen l'id corresponent només la primera ocurrència **trobada** o **nula** en els casos oposats.

```
var nodeList1 =document.getElementsByClassName("demo");
// .demo var nodeList2 =document.getElementsByTagName("p");
// All the p var nodeList3 =document.querySelectorAll("p");
// All the p
```

Aquests mètodes retornen una llista de nodes (llista de nodes) que té la propietat **.length a** aquests índexs s'accedeix a través dels claudàtors [], de manera que es pot recórrer a través d'un bucle (loop). No obstant això, això no és una matriu.

Recordeu que per als selectors CSS complexos, el mètode .querySelector encara no és millor que les biblioteques específiques com jQuery ni en termes de rendiment ni en termes de facilitat d'ús. A més, des de la seva concepció, algunes persones (com John Resig) han criticat la seva implementació.

Llegir & canviar HTML

Què és el DOM HTML?

Aquí trobaràs una gran explicació:

Pots canviar ràpidament el contingut i els atributs d'una etiqueta HTML amb .innerHTML (en Lectura i Escriptura) i un Getter / Setter per als atributs.

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

```
var hasH =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";
```

Observa la diferència entre .innerHTML i .textContent, el primer és analitzat i per tant accepta HTML i el segon no ho és. **TextContent** serà més ràpid i més **segur**.

Canviar el CSS

Totes les propietats CSS són accessibles a través de la sintaxi assenyalada sota la propietat style amb les següents característiques:

Les propietats s'escriuen en la casella Lower Camel (fontsize → fontsize)

Els valors són sempre cadenes (per exemple, no 250 però sí "250px")

El nou valor s'agregarà en l'estil en línia

La posició és per .offsetTop i .offsetLeft (ni dreta ni a baix)

Amples i altures totals (farcit, vora) a través d'.offsetWidth i .offsetHeigth

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

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

JavaScript normalment recupera l'estil en línia. Si vols que els estils CSS resultants d'un full d'estil extern o l'etiqueta <style>, utilitza .getComputedStyle ():

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

Canviar el DOM

Alguns mètodes per crear i inserir etiquetes HTML en el DOM.

- **Document.createElement ("htmlTag")** Crea una etiqueta HTML
- **Document.createTextNode ("Hello world")** Crea contingut de text
- **.removeChild (childToBeRemoved)** Elimina un element secundari
- **.appendChild (newContent)** Afegeix un nou element al final del pare
- **.insertBefore (newElement, currentElement)** Afegeix un element nou
- **.replaceChild (newElement, currentElement)** Substitueix un element

Alguns mètodes útils per explorar el DOM.

- **ParentNode** Mostra el node principal
- **.children [index]** S'adreça als fills a través d'un índex a partir de 0
- .nextElementSibling assenyala el següent node semblant (que té el mateix pare)
  - No confonguis amb .nextSibling que també retorna espais entre nodes i comentaris.
- **.previousElementSibling** S'adreça al node anterior semblant

...

Prova tu mateix/a!

JavaScript HTML DOM Elements:

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

Changing CSS:

Llista completa de propietats i mètodes

Exemples W3Schools [1][3]

L'objecte Screen

L'objecte **screen** proporciona informació sobre la pantalla de l'usuari. Aquesta informació **no és accessible** per un llenguatge de scripting del costat del servidor.

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

L'objecte Navegador

L'objecte Navegador ofereix informació sobre el navegador de l'usuari.

- **navigator.userAgent:** El nom complet del navegador
- **navigator.platform:** El sistema operatiu de l'usuari
- **navigator.onLine:** Si l'usuari està en línia o fora de línia
- **navigator.language:** L'idioma de la interfície del navegador
- **navigator.geolocation:** Geolocalització després de l'acord de l'usuari
- **navigator.cookieEnabled:** Si l'usuari accepta cookies
- …

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

Events (esdeveniments)

És possible vincular qualsevol esdeveniment (clic, doble clic, sobre, error, redimensionar, ...) a un element del DOM a través de la següent sintaxi:

element.addEventListener(event, function, useCapture);

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

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

El nom del paràmetre event és una cadena que ignora el 'on' d'atributs HTML (exemple: <a href="" onClick="..." /> → clic).

Llista completa en W3Schools:

Bàsicament, els esdeveniments HTML són "accions" que succeeixen als elements HTML.

Quan JavaScript s'utilitza en pàgines HTML, JavaScript pot "reaccionar" en aquests esdeveniments.

Els esdeveniments són accions o ocurrències que ocorren en el sistema que està programant, de les que el sistema l'informa per poder respondre-hi d'alguna manera, si així ho desitja. Per exemple, si l'usuari fa clic a un botó d'una pàgina web, pot respondre a aquesta acció mostrant un quadre d'informació.

En el cas de la web, els esdeveniments es desencadenen dins de la finestra del navegador i tendeixen a connectar-se a un element específic que resideix en ell: pot ser un sol element, un conjunt d'elements, el document HTML carregat a la pestanya actual o tota la finestra del navegador. Hi ha molts tipus diferents d'esdeveniments que poden ocórrer, per exemple:

- L'usuari fa clic amb el ratolí sobre un determinat element o col·loca el cursor sobre un determinat element.
- L'usuari pressiona una tecla en el teclat.
- L'usuari canvia la mida o el tancament de la finestra del navegador.
- La càrrega d'una pàgina web s'ha acabat.
- S'envia un formulari.
- Es reprodueix un vídeo, es pausa o s'acaba la reproducció.
- S'ha produït un error.

Cada esdeveniment disponible té un controlador d'esdeveniments (**event handler**), que és un bloc de codi normalment definit pel desenvolupador que s'executarà quan es desencadeni l'esdeveniment. Quan aquest bloc de codi es defineix per ser executat en resposta a un disparador d'esdeveniment, diem que estem registrant un gestor d'esdeveniments (**Registering an event handler**). Recorda que els event handlers de vegades es diuen **event listeners (oients)** - són bastant intercanviables per als nostres propòsits, encara que en sentit estricte treballen junts. L'oient escolta l'esdeveniment que succeeix, i l'encarregat (handler) és el codi que es corre en resposta a el que succeeix.

Un exemple senzill:

En l'exemple següent, tenim un sol element <button>, que quan es pressiona, farà que el fons canviï a un color aleatori:

```
<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;
}
```

En aquest codi, emmagatzemem una referència al botó dins d'una variable anomenada btn, utilitzant la funció Document.querySelector (). També definim una funció que retorna un nombre aleatori. La tercera part del codi és el gestor d'esdeveniments. La variable btn apunta a un element **<button>**, i aquest tipus d'objecte té un nombre d'esdeveniments que poden disparar sobre ell, i per tant, els controladors d'esdeveniments disponibles. Estem escoltant l'activació de l'esdeveniment clic, establint la propietat **onclick event handler** com igual a una funció anònima que conté codi que va generar un color RGB aleatori i estableix el color de fons **<body>** igual a ell.

Aquest codi s'executarà ara cada vegada que s'activi l'esdeveniment de clic en l'element <button>, és a dir, cada vegada que un usuari faci clic.

Codi i resultat: https://codepen.io/pen/

Event handler properties (Propietats del controlador d'esdeveniments)

Tornant a l'exemple anterior:

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

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

La propietat **onclick** és la propietat del gestor d'esdeveniments que s'utilitza en aquesta situació. És essencialment una propietat com qualsevol altra disponible al botó (per exemple, btn.textContent, o btn.style), però és un tipus especial - quan s'estableix que sigui igual a algun codi, aquest codi s'executarà quan es pressiona el botó.

Event Management (gestor d'esdeveniments)

L'objecte Event representa qualsevol esdeveniment DOM. A mesura que es passa en l'argument, podem utilitzar-lo en la nostra funció que maneja l'esdeveniment.

Té moltes propietats i mètodes...

- **Event.target** L'objecte que va emetre l'esdeveniment
- **Event.currentTarget** Retorna l'element en què es va adjuntar l'esdeveniment
- **Event.preventDefault ()** Cancel·la el comportament normal
- **Event.stopPropagation ()** Cancel·la la propagació de l'esdeveniment

Capturing & Bubbling

Capture és la fase descendent (des de l'element exterior fins l'element interior) i el **bubbling** la fase ascendent (Com un bussejador que se submergeix i ascendeix).

El valor per omissió de useCapture és fals i IE9 només admet l'esdeveniment de **bubbling**.

Eliminar els controladors d'esdeveniments

Utilitza **.removeEventListener ()** però no és possible eliminar funcions anònimes. A més, cada gestor d'esdeveniments s'ha d'eliminar per separat, un esdeveniment en **bubbling** i un esdeveniment a la **capturing** en el mateix esdeveniment són dos **eventListener** diferents.

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

Quizz

Aquí hi ha una petita prova que hauries de ser capaç de passar ara 🏝
Trobaràs les respostes al final de la mateixa. Intenta de no fer trampa!

Preguntes

Introducció

1. Quan el teu programa no funciona, el primer que has de fer és mirar els missatges d'error. Com s'obre la consola web tant en Firefox com a 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. Pots declarar una variable amb i sense la paraula clau var. Quin camí has d'utilitzar?

- Tots dos, no hi ha cap diferència.
- Una variable declarada amb la paraula clau 'var' pertany a l'àmbit local, l'altra pertany a l'àmbit global. Utilitzeu sempre var per evitar contaminar l'àmbit global.

- Una variable declarada amb la paraula clau 'var' pertany a l'àmbit global, l'altra pertany a l'àmbit local. No utilitzi mai var per evitar contaminar l'àmbit global.

3. Per què sempre has d'escriure comentaris?

- És bo explicar la lògica i els detalls d'un programa. Tant per a tu mateix si estàs treballant en un projecte nou, en un antic o quan ets part d'un equip.
- Els comentaris són una pèrdua de temps, ningú els escriu de totes maneres.
- És la manera correcta de desactivar temporalment alguna part d'un codi sense perdre-ho.

Condicions

1. Hi ha alguna diferència lògica entre una estructura 'if else' i dues consecutives 'if' que tenen la condició oposada?

- No, però el 'if else' és molt millor perquè és més llegible i menys propens a errors.
- Sí, no són el mateix.

2. Com pots recórrer tots i cadascun dels elements d'una matriu de longitud desconeguda?

- Mitjançant l'ús de la propietat array .length.
- Mitjançant l'ús del mètode array .length ().
- Mitjançant l'ús de la funció count ().

3. *Què millora el següent codi?:*
for (var i = 0, len = a.length; i < len, i++) { /* ... */ }
comparat amb
for (var i = 0; i < a.length, i++) { /* ... */ }
?

- Res

- La variable 'len' només s'avalua una vegada en lloc d'una vegada per iteració.
- El nombre d'iteració és diferent.

Funcions

1. És important l'ordre dels arguments d'una funció anomenada 'divideix' (que retorna el resultat d'una divisió del primer argument per la segona)?

- Sí, canvia el resultat.
- No

2. Quina és la diferència entre una 'function call' i una 'function reference'?

- Una 'function call' (el seu nom seguit de parèntesis) sol·licita l'execució del codi de funció en el moment de la trucada. Una 'function reference' (en una trucada enrere, per exemple) simplement emmagatzema el nom de la funció per a referències futures.
- No hi ha cap diferència si la funció no té cap argument.

3. Quina paraula clau reservada s'utilitza per especificar el valor retornat de la trucada d'una funció?

- Return
- Pass
- Exit

DOM

1. Quins mètodes no es refereixen a la primera o única ocurrència trobada?

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

2. En el codi següent, quin argument és rebut per la devolució de trucada?
document.getElementById('myID').addEventListener('click', myCallBack);

- L'esdeveniment que va activar la devolució de trucada.
- No va passar cap argument a través de la devolució de trucada.

3. Com pots obtenir el valor d'un element HTML (UI) com la d'una etiqueta?

- Mitjançant el the property .value que retorna el contingut de l'atribut 'valor'.
- Mitjançant el mètode .getValue () que retorna el contingut de l'atribut 'value'.
- Mitjançant el mètode .VAL () que retorna el contingut de l'atribut 'value'.

CSS

1. Què selector CSS permet seleccionar només el paràgraf que és fill directe de la caixa div?

- div > p
- div p
- div, p

2. Què selector CSS permet seleccionar només el primer paràgraf independentment del nombre de germans HTML que tingui?

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

3. Si dos selectors CSS (.red y #blue) apunten al mateix element, quin serà el seu color?

- Blue
- Red
- Purple
- Depèn: el que sigui declarat últim.

Respostes

Introducció

1. Quan el teu programa no funcioni, el primer que has de fer és mirar els missatges d'error. Com s'obre la consola web tant en Firefox com a 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. Pots declarar una variable amb i sense la paraula clau var. Quin camí has d'utilitzar?

- Una variable declarada amb la paraula clau 'var' pertany a l'àmbit local, l'altra pertany a l'àmbit global. Utilitzeu sempre var per evitar contaminar l'àmbit global.

3. Per què sempre has d'escriure comentaris?

- És bo explicar la lògica i els detalls d'un programa. Tant per a tu mateix si estàs treballant en un projecte nou, en un antic o quan ets part d'un equip.
- És la manera correcta de desactivar temporalment alguna part d'un codi sense perdre-ho.

Condicions

1. Hi ha alguna diferència lògica entre una estructura 'if else' i dues consecutives 'if' que tenen la condició oposada?

- No, però el 'if else' és molt millor perquè és més llegible i menys propens a errors.

2. Com pots recórrer tots i cadascun dels elements d'una matriu de longitud desconeguda?

- Mitjançant l'ús de la propietat array .length.

- Mitjançant l'ús del mètode array .length ().
- Mitjançant l'ús de la funció count ().

3. *¿Què millora el següent codi:*
for (var i = 0, len = a.length; i < len, i++) { /* ... */ }
comparat amb
for (var i = 0; i < a.length, i++) { /* ... */ }
?

- La variable 'len' només s'avalua una vegada en lloc d'una vegada per iteració.

Funcions

1. És important l'ordre dels arguments d'una funció anomenada 'divideix' (que retorna el resultat d'una divisió del primer argument per la segona)?

- Sí, canvia el resultat.

2. Quina és la diferència entre una 'function call' i una 'function reference'?

- Una 'function call' (el seu nom seguit de parèntesis) sol·licita l'execució del codi de funció en el moment de la trucada. Una 'function reference' (en una trucada enrere, per exemple) simplement emmagatzemi el nom de la funció per a referència futura.

3. Quina paraula clau reservada s'utilitza per especificar el valor retornat de la trucada d'una funció?

- Return

DOM

1. Quins mètodes no es refereixen a la primera o única ocurrència trobada?

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

2. En el codi següent, quin argument és rebut per la devolució de trucada?
document.getElementById('myID').addEventListener('click', myCallBack);

- L'esdeveniment que va activar la devolució de trucada.

3. Com pot obtenir el valor d'un element HTML (UI) com l'entrada d'una etiqueta?

- Mitjançant el the property .value que retorna el contingut de l'atribut 'valor'.

CSS

1. Què selector CSS permet seleccionar només el paràgraf que és fill directe de la caixa div?

- div > p

2. Què selector CSS permet seleccionar només el primer paràgraf independentment del nombre de germans HTML que tingui?

- p:nth-of-type(1)

3. Si dos selectors CSS (.Xarxa i #blue) apunten al mateix element, quin serà el seu color?

- Blue

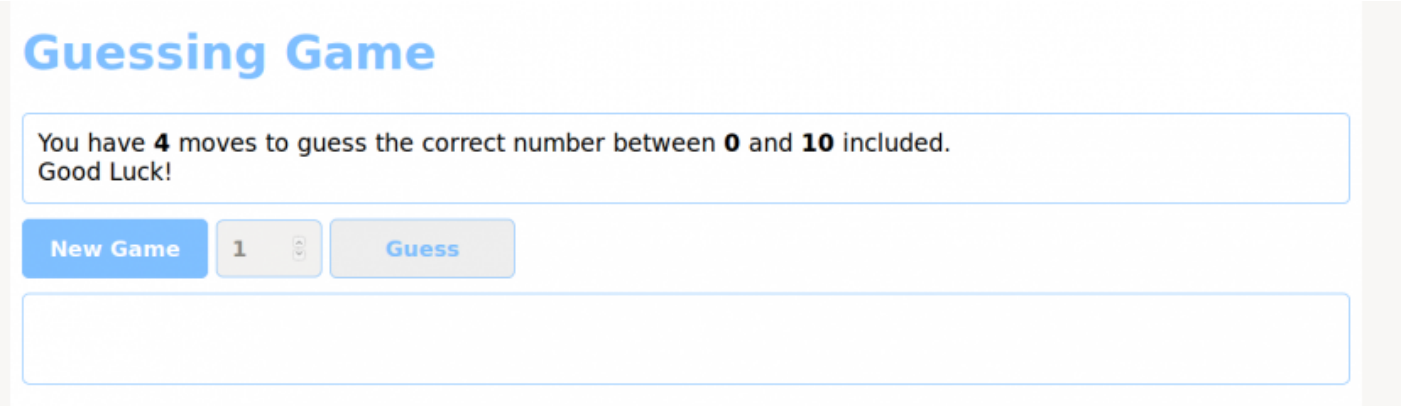
Coding lab: joc d'endevinalles

Ara que entens els conceptes bàsics d'HTML / CSS / JavaScript, és hora de divertir-te i fer el teu primer joc...

En aquesta lliçó, recrearàs el Joc d'endevinalles. Això vol dir que vas a crear:

1. Un fitxer HTML en el qual es posaran tots els elements requerits pel joc
2. Un arxiu CSS a través del qual es va a dissenyar el teu joc (de la manera que vulguis ;-)
3. Un arxiu JavaScript que contindrà totes les funcions requerides pel joc

Això és bàsicament com ha de ser (encara que l'aspecte pot variar):



A la pàgina següent, trobaràs una guia pas a pas per fer-ho, per si la necessites.

Començant

1. Ens caldrà almenys un camp numèric i un botó per obtenir el resultat, però també algun tipus de "contenidor" en què escriurem/anotarem els resultats fins i tot si comença buit o invisible.

Tracta de pensar si es necessita o no el botó 'Nou joc' tant com a usuari com com a punt de vista del programador; i si és útil o no. Si ho desitja, al final de l'exercici pots canviar el codi per permetre que un usuari abandoni el joc actual, per exemple.

2. Tracta de pensar en tots els passos necessaris per a programar aquest mini joc. Aquest serà el començament de les teves diferents blocs de codi. Si un problema sembla difícil de resoldre, el més probable és que necessitis tallar-ho en trossos més petits.

Si necessites reutilitzar diverses vegades el mateix codi/lògica, és millor convertir-lo en una funció, especialment si els paràmetres d'inici poden canviar.

Sempre has d'intentar utilitzar variables en lloc de valors programats, de manera que puguis canviar fàcilment els paràmetres del joc com el nombre d'hipòtesis disponibles, per exemple. Millora la llegibilitat, la facilitat de manteniment i fes que el teu codi sigui més configurable.

Comenta sempre el teu codi (explicant la lògica darrere del codi i no el codi) per a tu, per a futures ocasions, i també pensant en els teus companys d'equip. :)

3. getRandomIntegerBetween () és un exemple simple d'una funció que millora la llegibilitat i converteix un tros de codi en alguna cosa fàcilment reutilitzable.

Tingues en compte que la veritable aleatorietat no és una cosa trivial en les ciències de la programació. Els ordinadors són deterministes, el que significa que si fas la mateixa pregunta, obtindràs la mateixa resposta cada vegada. De fet, com màquines són específicament i acuradament programades per eliminar l'aleatorietat en els resultats. Veure pseudo aleatori a Google.

En checkResponse (), veiem que return també pot usar-se per sortir d'una funció sense tornar alguna cosa, passant per alt la prova inútil de 'game over' quan el joc ja està guanyat.

Finalment, veiem que les tecnologies web com JavaScript intenten accedir i actualitzar el DOM HTML (Document Object Model) amb coses com document.getElementById ('userGuess') i result.innerHTML o el CSS. Cadascuna d'aquestes tecnologies relativament fàcils treballen en tàndem.

A continuació, trobaràs algunes pistes per a l'estructura i funcions ...

ESTRUCTURA I FUNCIONS

```
// Tornar un enter entre un valor mínim i un valor màxim inclòs.
    function getRandomIntegerBetween(min, max)

// Activa la interfície d'usuari per a un joc nou.
    function activateUI()

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

    function init(){
        // Reajusta el màxim.
          // Neteja el registre de logs
          // Inicialitza un nou valor aleatori
        activateUI();
    }

// comprova la resposta
function checkResponse()

// jugades disponibles abans de perdre el joc

    // Nombre aleatori estimat pel jugador.

    // Valors mínims i màxims (ambdós inclosos) entre els que pots escollir.

    // Regles de visualització

    // Registre de logs

    // Pots endevinar la forma òptima de guanyar sempre aquest joc?
```

Referències & Recursos

JAVASCRIPT

[EN] JavaScript

http://www.w3schools.com/js/default.asp
http://www.w3schools.com/js/js_performance.asp
http://eloquentjavascript.net https://developer.mozilla.org/en-US/en%C2%ADUS/docs/Web/JavaScript
https://developer.mozilla.org/en/docs/Web/API/Event
https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript

[EN] JavaScript Style Guide

http://www.w3schools.com/js/js_conventions.asp

JavaScript for the Total Non-Programmer

http://www.webteacher.com/javascript/

Defer & async [EN]

http://www.growingwiththeweb.com/2014/02/asyncvsdeferattributes.html

Modernizr

http://modernizr.com/

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

http://stackoverflow.com/questions/1470488/what-is-the-purpose-of-the-var-keyword-and-when-to-use-it-or-omit-it

[EN] The infamous 'this' keyword

https://www.sitepoint.com/mastering-javascripts-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!
http://www.javascriptsource.com/
HTML & CSS

HTML5 Tutorial

http://www.w3schools.com/html/default.asp

CSS3 Tutorial
http://www.w3schools.com/css/default.asp

I molt més a Google, YouTube i tota la web!

Gràcies a David Collignon, per les notes d'ajuda i cursos fantástics.