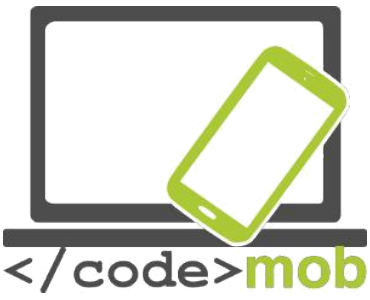
</code>mob

Programación
para usuarios finales

CodeMob: Programación para usuarios
Octubre 2017. http://codemob.eu/. Autores:

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

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



Índice de contenidos

¡Bienvenidos a este curso de programación!

Al final de este curso, deberías ser capaz de:

- construir un juego (por ejemplo, el memory) en JavaScript
- añadir un GUI (interfície de usuario básica) con HTML5 & CSS3
- entender y explicar los conceptos básicos de la programación.

En primer lugar, ten en cuenta que existen miles de recursos en línea. No sería posible proponer un curso completo aquí. Sin embargo, en la programación, encontrarás todas las respuestas gracias a una simple búsqueda en la web. Considera este curso como una introducción, paso a paso, en JavaScript.
¡A disfrutar!

¿Por qué aprendemos JavaScript?

No debe confundirse con Java: JavaScript permite crear sitios web interactivos. JavaScript se ha convertido en una tecnología web esencial junto con HTML y CSS, ya que la mayoría de los navegadores implementan JavaScript. Por lo tanto, es importante aprender JavaScript si se desea entrar en el ámbito del desarrollo web, y debe aprenderse bien si estamos pensando en convertirnos en un desarrollador de front-end o en el uso de JavaScript para el desarrollo de back-end.

Además, el uso de JavaScript se ha extendido ahora al desarrollo de aplicaciones para dispositivos móviles, al desarrollo de aplicaciones de escritorio y al desarrollo de juegos. En general, se ha hecho popular y ahora trabajar con JavaScript es una habilidad que resulta muy interesante adquirir.

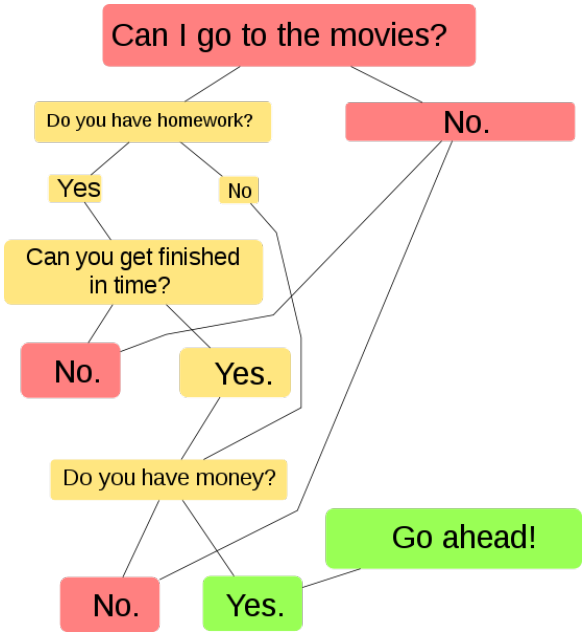

3
Beginner Friendliness
3
Scalability
5
Community
5
Career Opportunities
5
Future

Fuente:

Algoritmos

En matemáticas y ciencias de la computación, un **algoritmo** (ælgərɪðəm / AL-gə-ri-dhəm) es un conjunto de operaciones autónomas paso a paso. Los algoritmos realizan tareas de cálculo, procesamiento de datos y / o razonamiento automatizado.

Aquí está un ejemplo simple de lo que un algoritmo puede ser:



¿Cuál es la relación entre un **algoritmo** y un **programa**? Mira un **algoritmo** como una fórmula para trabajar algo. Un **programa** es una serie de instrucciones para el equipo que utiliza algoritmos para ejecutar la tarea deseada.

Vamos a entender esto a través de un ejemplo:

Algoritmo simple:

Área del rectángulo A = longitud por profundidad

El programa para trabajar el área sería:

Pedir al usuario que introduzca la longitud
Pedir al usuario que introduzca la profundidad
Usar el algoritmo de área para calcular el área
Mostrar respuesta

Introdución a HTML y CSS

Para crear un sitio web, debe proporcionar información al equipo. No basta con escribir el texto que estará en su sitio, también es necesario saber cómo colocar este texto, insertar imágenes, hacer enlaces, etc ... Para explicarle a la computadora lo que quiere, será necesario utilizar un lenguaje que entienda.

Hay lenguajes utilizados para crear programas, como C ++ o Java. Sin embargo, estos lenguajes son complejos y están destinados a personas que ya tienen algún conocimiento informático.

HTML y **CSS** se utilizan precisamente para crear sitios web, y se han creado para ser fáciles de usar. Cada uno de estos 2 lenguajes se utiliza para hacer algo específico, y los dos se complementan naturalmente para finalmente dar un sitio web:

• **HTML**: Es la abreviatura de HyperText Markup Language que apareció en 1991 cuando se lanzó la Web. Su función es gestionar y organizar el contenido. Por lo tanto, en HTML se escribe lo que se debe mostrar en la página: texto, enlaces, imágenes. . . Por ejemplo: este es mi título, este es mi menú, aquí está el texto principal de la página. ...
En este curso vamos a trabajar en la última versión de HTML (HTML5) que es hoy el lenguaje del futuro

• **CSS**: Esta es la abreviatura de **CASCADING STYLE SHEETS**. Este lenguaje solo sirve para diseñar la página web. Es en CSS que se dice: "Mis títulos están en rojo y están subrayados, mi texto está en fuente ARIAL, mi nombre está centrado, mi menú tiene un fondo blanco ..." etc. ... Con este lenguaje, podremos crear rápida y sencillamente el diseño de su sitio.

El editor

Una pregunta que debes hacerte es: "¿Qué programa necesito para crear mi sitio web?"

¡Notepad es suficiente para crear un sitio web! Pero para facilitar nuestro trabajo podemos usar un editor de código como Notepad ++ (la ventaja es

que automáticamente colorea el código HTML / CSS para hacerlo más legible).

Navegadores

¿Qué es un navegador?

El navegador es el programa quetle permite ver los sitios web. El trabajo del navegador es leer el código HTML / CSS que escribió y mostrar lo que representa. Si tu código CSS dice "los títulos son rojos", entonces el navegador mostrará los títulos en rojo.

Entre los navegadores que existen, aquí están los principales:
Internet Explorer - Mozilla Firefox - Opera - Netscape - Konqueror (para Linux) - Lynx (para Linux) - Apple Safari (para Mac) - etc.

Crea tu primer documento HTML

¿Qué es HTML?

Como hemos mencionado anteriormente, HTML es un lenguaje de marcado para describir documentos web (páginas web).

- HTML significa Hyper Text Markup Language
- Un lenguaje de marcado es un conjunto de etiquetas de marcado
- Los documentos HTML se describen mediante etiquetas HTML
- Cada etiqueta HTML describe diferentes contenidos de documentos

Un pequeño documento HTML
Ejemplo

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

<title>Page Title</title>
</head>
<body>

<h1>Mi primer encabezado</h1>
* MI primer párrafo. *

</body>
</html>

Ejemplo explicado

- La declaración <! DOCTYPE html> define este documento como HTML5
  El texto entre <html> y </html> describe un documento HTML
  El texto entre <head> y </head> proporciona información sobre el documento
  El texto entre <title> y </title> proporciona un título para el documento
  El texto entre <body> y </body> describe el contenido visible de la página
  El texto entre <h1> y </h1> describe un encabezado
  El texto entre y describe un párrafo
  Con esta descripción, un navegador web mostrará un documento con un encabezado y un párrafo.

  Para más información, mira el documento completo en

Elementos HTML que necesita

En esta lección, su tarea es crear un documento HTML simple que contenga el siguiente contenido:

No te olvides de usar los comentarios. Un **comentario** es una etiqueta **HTML** con una forma muy especial:

<!-- Esto es un comentario -->

Puedes colocarlo donde desees en tu código fuente: no tiene ningún impacto en tu página, pero puedes utilizar comentarios para explicar tu código, lo que

puede ayudarte cuando edites el código fuente en una fecha posterior. ¡Esto es especialmente útil si tienes un montón de código!

Atención: todos pueden ver el código HTML de tu página una vez que se sube a la web. Simplemente haces clic con el botón secundario en la página y seleccionas "Mostrar el código fuente de la página". Por lo tanto, hay que evitar poner información sensible como contraseñas en los comentarios ... y hay que cuidar la calidad de tu código fuente.

Crea un archivo CSS y utilízalo para diseñar tu documento HTML

¿Qué es CSS?

- CSS significa hojas de estilo en cascada.
- CSS describe cómo se muestran los elementos HTML en la pantalla, en el papel o en otros medios.
- CSS ahorra mucho trabajo. Puedes controlar el diseño de varias páginas web de una sola vez.
- Las hojas de estilo externas se almacenan en archivos CSS

¿Por qué usar CSS?

CSS se utiliza para definir estilos para sus páginas web, incluyendo el diseño, la disposición y las variaciones en la pantalla para diferentes dispositivos y tamaños de pantalla.

CSS resolvió un gran problema

HTML nunca fue diseñado para contener etiquetas para el formato de una página web. En cambio, HTML fue creado para describir el contenido de una página web, como:

<h1> Este es un encabezado </h1>
 Este es un párrafo.

Cuando se agregaron etiquetas como <font>, y atributos de color a la especificación HTML 3.2, comenzó una pesadilla para los desarrolladores web. El desarrollo de sitios web grandes, donde se agregaron fuentes e información de color a cada página, se convirtió en un proceso largo y costoso.

Para solucionar este problema, el World Wide Web Consortium (W3C) creó CSS.

¡CSS elimina el estilo de formato de la página HTML!

¡CSS ahorra mucho trabajo!

Las definiciones de estilo se guardan normalmente en archivos .css externos. Con un archivo de hoja de estilo externo, puede cambiar el aspecto de un sitio web entero mediante la modificación de un solo archivo.

Para obtener más información, consulta .

Hoja de estilo externa

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

Tres maneras de insertar CSS: hay tres maneras de insertar una hoja de estilo:

1. Hoja de estilo externa
2. Hoja de estilo interna
3. Estilo en línea

En esta lección vamos a insertar una **hoja de estilo externa**.

Con una hoja de estilo externa, puedes cambiar el aspecto de un sitio web completo ¡cambiando sólo un archivo!

Cada página debe incluir una referencia al archivo de hoja de estilo externo dentro del elemento <link>. El elemento <link> va dentro de la sección <head>:

Ejemplo:

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

Una hoja de estilo externa se puede escribir en cualquier editor de texto. El

archivo no debe contener ninguna etiqueta html. El archivo de hoja de estilo se debe guardar con una extensión .css.

Así es como se ve el "myStyle.css":

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

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

Nota: No agregues un espacio entre el valor de la propiedad y la unidad (como margin-left: 20 px;). La forma correcta es: margin-left: 20px;

Programación JS

Historia del JavaScript

El lenguaje JavaScript fue creado en **1995 por Brendan Eich** (Fundación Mozilla) para Netscape. El lenguaje, actualmente en la versión 1.8.5, es una implementación de la tercera versión de la norma ECMA262.

- En diciembre de 1995, Sun y Netscape anunciaron el lanzamiento de JavaScript. Microsoft reacciona entonces desarrollando JScript (el mismo idioma para un nombre diferente para evitar una disputa de marcas con Sun).
- Netscape envía JavaScript a Ecma International para estandarización en noviembre de 1996.
- DHTML (1996), un comienzo difícil e innecesario
- Ajax (2000), Un interés renovado gracias a las nuevas posibilidades
- JSon: una alternativa al XML basado en JavaScript
- Marcos de JavaScript: facilidad, productividad y estandarización, pero difícil elección
- node.js; Eventdriven & Serversided
- NodeWebkit & Cordova

¿Qué es JavaScript?

● Un lenguaje de secuencias de comandos (lenguaje de alto nivel)
  ○ Interpretado (en contraposición a los idiomas compilados)

● Un lenguaje del lado del cliente (se ejecuta en la máquina del usuario y no en el servidor).

  ○ Sin embargo node.js apareció en 2009 y cambió algo la situación.

● Capaz de cambiar el DOM:

  ○ Crear, editar y eliminar elementos HTML, sus atributos o CSS.
  ○ Crear, escuchar y borrar eventos.

● No tiene nada que ver con el lenguaje Java de Sun ...;)

Ejemplo: & the Polyfills

Instalación

Los scripts pueden colocarse en una página HTML, ya sea en el <head> o en el <Body> (justo antes del </body>) y pueden ser **internos o externos.** El **atributo type es opcional** porque su valor predeterminado es correcto (JavaScript es el único idioma aceptado 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>
```

Ejemplo JavaScript

Vamos a intentar hacer nuestro primer archivo JavaScript. En este ejemplo vamos a crear un cuadro de alerta simple. Estaremos haciendo esto enlazando un archivo .js externo a nuestro documento .html.

Lo primero que tienes que hacer es ir a la carpeta en la que has creado tu primer archivo Html y archivo .css. Crea un nuevo archivo .js y guárdalo en la misma carpeta que (por ejemplo) myscript.js.

Ahora agrega la siguiente secuencia de comandos al archivo .js:

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

Finalmente, vincula el myscript.js a tu documento .html:

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

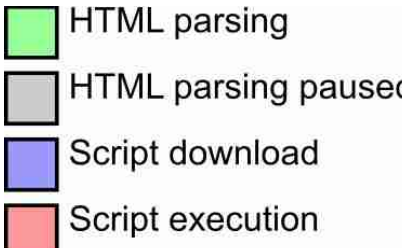
```
<body>
<script src="myScript.js"></script>
</body>
</html>
```

Puedes colocar una referencia de guión externa en <head> o <body> según prefieras. El script se comportará como si estuviera ubicado exactamente donde se encuentra la etiqueta <script>.

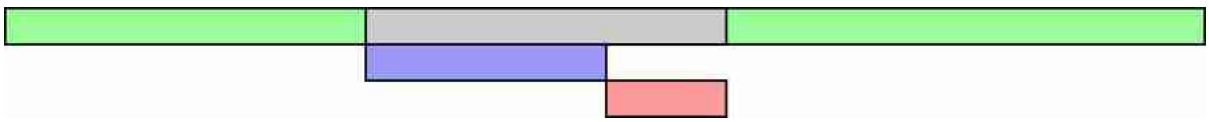
Orden de ejecución de secuencias - Defer & async

El tiempo de un script depende de su posición en la página HTML y los atributos de diferimiento y asincronismo.

Los atributos de diferimiento y asíncrono sólo se tienen en cuenta para los scripts externos.

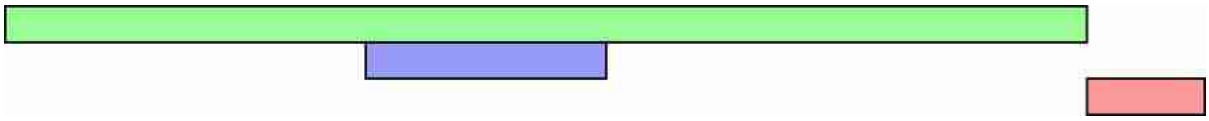


Normal execution



Uso común: si dos secuencias de comandos se siguen, la segunda nunca se ejecutará antes del final de la primera porque cualquier recurso está 'bloqueando'.

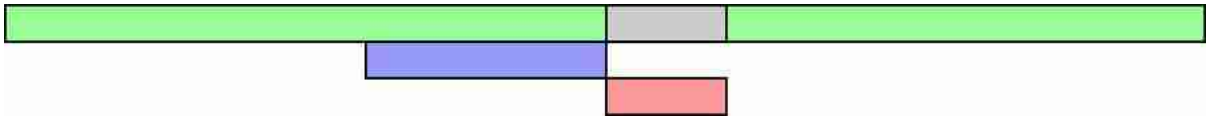
Defer



Nunca utilice document.write() or equivalent (con defer).

Presta siempre atención a la orden de ejecución de los scripts que tienen la misma prioridad; se supone que debe ser respetado, pero tenemos los errores en IE4-9.

Async (HTML 5)



Ideal para un script cuyo tiempo de ejecución no es importante (por ejemplo: Google Analytics), pero el orden de ejecución de los scripts no está garantizado!

Nunca uses document.write() o equivalente (con async).

Trucos para el diseño

Cuando JavaScript lee y reescribe o cambia el DOM, el diseño se invalida y debe volver a calcularse en algún momento en el futuro. Los navegadores tratan de esperar hasta el final de un marco, pero si tienen que hacerlo antes, puede tener un serio impacto en el rendimiento.

En pocas palabras, utilizar lo menos posible las manipulaciones y cambios de los DOM, o en el peor de los casos, agruparlos tanto como sea posible.

Si estás interesado en el tema, puedes informarte sobre requestAnimationFrame y bibliotecas como FastDOM ().

La consola JavaScript

Es fácil perderse escribiendo código JavaScript sin depurador.
La depuración es el proceso de probar, encontrar y reducir errores en los programas informáticos.

Si un script no funciona (error), rara vez se ve el error a simple vista ... Por lo tanto, tenemos que mostrar la consola de JavaScript para ver los **mensajes de error** (archivo y línea de código, código de color, script sobre la marcha).

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

También puede depurar con **console.log** (...); el cual acepta en parámetro lo que debe mostrarse. También es posible añadir **puntos de interrupción (break points)** o utilizar el comando **debugger**.

El método console.log()

Si tu navegador admite la depuración, puedes utilizar **console.log ()** para mostrar valores JavaScript en la ventana del depurador:

Ejemplo:

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

<h1>My First Web Page</h1>

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

</body>
</html>
```

Establecimiento de puntos de interrupción (Setting Breakpoints)

En la ventana del depurador, puedes establecer puntos de interrupción en el código JavaScript.

En cada punto de interrupción, JavaScript dejará de ejecutarse y te permitirá examinar los valores de JavaScript.

Después de examinar los valores, puedes reanudar la ejecución del código (normalmente con un botón de reproducción).

El depurador Keyword

El **debugger** keyword (depurador Keyword) detiene la ejecución de JavaScript y llama (si está disponible) a la función de depuración. Esto tiene la misma función que establecer un punto de interrupción en el depurador.

Si no hay depuración disponible, la instrucción depurador no tiene ningún efecto.

Con el depurador activado, este código dejará de ejecutarse antes de ejecutar la tercera línea.

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

Ten en cuenta que el objeto de consola no forma parte de los estándares, aunque al menos esté bien implementado por Firefox y Chrome.

Ejemplo: API Console Firefox

Comentarios

Los comentarios JavaScript se pueden utilizar para **explicar el código JavaScript** y para hacerlo más legible.

Los comentarios de JavaScript también se pueden utilizar para **evitar la ejecución,** al probar el código alternativo.

Comentarios de línea única

Los comentarios de una sola línea comienzan con //.

Cualquier texto entre // y el final de la línea será ignorado por JavaScript (no se ejecutará).

Este ejemplo utiliza un comentario de una sola línea antes de cada línea de código:

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

Este ejemplo utiliza un comentario de línea simple al final de cada línea para explicar el código:

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

Comentarios multi-linea

Los comentarios de varias líneas comienzan con /* y terminan con */.

Cualquier texto entre /* y */ será ignorado por JavaScript.

Este ejemplo utiliza un comentario de varias líneas (un bloque de comentario) para explicar el código:

```
/*
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.";
```

Uso de comentarios para evitar la ejecución

El uso de comentarios para evitar la ejecución de código es adecuado para las pruebas de código.

Agregar // delante de una línea de código cambia las líneas de código de una línea ejecutable a un comentario.

Este ejemplo utiliza // para evitar la ejecución de una de las líneas de código:

Example

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

Este ejemplo utiliza un bloque de comentario para evitar la ejecución de varias líneas:

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

Los comentarios sobre una sola línea comienzan con // y los de múltiples líneas están rodeados por /*...*/. A continuación, un caso un poco más complicado para insertar JavaScript en un documento XHTML.

Utiliza siempre los comentarios para comentar sobre tu **lógica**, los parámetros esperados por una función, que devuelve o desactiva un pedazo de código sin borrarlo.

<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>
```

Cosas que no hay que hacer: Eval & javascript:

No utilice (casi nunca) **eval()**, ya que permite ejecutar código arbitrario, representa un riesgo de seguridad.

La función **Setinterval()** que llama a su primer parámetro después de cierto tiempo utiliza eval() si recibe una cadena como parámetro. Por lo tanto, prefiere una función como un argumento.

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

Mantenga la separación entre HTML y JavaScript (como con CSS por razones de mantenimiento y lógica) evitando el document.write (...) y separando el JavaScript del diseño.

```
<!-- 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>
```

Operadores matemáticos

- *, /, +,- (orden de prioridad y paréntesis)
- % (Módulo)
- +=, =, *= & /= (x = x + 5;)
    - ++ Incremento. Operador unitario. Suma uno a su variable. Si se utiliza como prefijo ++x devuelve el valor de su variable después de añadirle uno; si se utiliza como sufijo x++ devuelve el valor de su variable antes de anadirle uno.
    - -- Decremento. Operador unitario. Resta uno a su variable. El valor devuelto es igual que para el operador incremental.

Constantes

Similar a las variables excepto la ya declarada, su valor no puede ser cambiado. A menudo se utilizan como parámetros predefinidos. Por convención y en casi todos los idiomas, los nombres de constantes se escriben en letras mayúsculas separadas por subrayado. Atención, no hay soporte para constantes en IE10.

```
// Constant
const MY_FAVORITE_NUMBER =9;

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

Variable & tipo de datos

Las variables JavaScript son un lenguaje de escritura **dinámico** como PHP, utilizan una **sintaxis pointed** y es **sensible a mayúsculas y minúsculas**, así que has de ser metódico y riguroso porque el tipo de variable puede cambiar en tiempo de ejecución.

El uso del Lower Camel Case para nombres de variables (primera letra de cada palabra, excepto el primero en mayúsculas, sin espacio) se considera la mejor práctica. JavaScript tiene un recolector de basura que destruye las variables que ya no se utilizan (o las establece como nulas).

No utilice la sintaxis var a = new Array(); o guión en los nombres de las variables (por ejemplo: 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
```

Tipos de datos

- Primitivos
  - **Strings** (carácteres tipo texto) : "Hello World"
  - **Numbers** (enteros y números con coma flotante) : 3.14 or 42
  - **Booleans** (valores Booleanos): true or false (verdadero o falso)

- Undefined (Una variable que no ha sido declarada o que no coincide con ningún tipo)
  Null (variable vacía, es decir, a la que se ha asignado un valor cero) µ

- **Objetos**
  - **Functions**
  - **Arrays**
  - Date, Math
  - RegExp (expresiones regulares)

Función (cambiar el tipo de variable)

Es posible cambiar el tipo de una variable (Casting) a otra mediante Number (...), String (...) y Boolean (...) o incluso algunos métodos como .split () o .join ().

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

Cadenas

● La concatenación de cadenas se ejecuta a través del signo "+"
● Las cadenas están delimitadas por comillas dobles (double quote) o comillas simples (single quote).

○ A diferencia de PHP, no hay diferencia entre los dos.

○ Se pueden insertar caracteres especiales a través de \ (carácter de escape)

■ \", \n (volver a la línea), \\, ...

■ \ en cada vuelta a la línea

● La longitud es accesible a través de la propiedad .length

● Las cadenas tienen muchos métodos;

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

● No acceda a una cadena como si fuera una matriz (Array)

Ejemplo:

Números

● Sólo un tipo (**Integer & Float**) almacenado bajo un punto flotante de 64 bits.

● Las constantes **Infinity & -Infinity** representan obviamente el infinito positivo y el negativo.

● **NaN** (Not a Number) será devuelto por una imposibilidad matemática como una división por una cadena, por ejemplo.

● El método **.toString()** los convierte en cadenas

- .toString(2): conversión a binario
- .toString(16): conversión a hexadecimal

● El objeto **Math** ofrece muchas características como:

○ Math.random(): Devuelve un número aleatorio entre 0 inclusive y 1 exclusivo

○ Math.min(..., ...) & Math.max(..., ...): Mínimo y Máximo

○ Math.round(), Math.ceil() & Math.floor(): redondeado, redondeado hacia arriba y hacia abajo

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

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

Ejercicio: Mueva al azar el número 0 o 1 en la consola cada vez que vuelva a cargar la página (F5).

Array (Matriz)

● Las matrices son instanciadas usando corchetes vacíos [] y usándolas para acceder a un índice específico.

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

● El índice empieza como en casi todos los idiomas de la computadora a 0.

○ `myArray[0]=42;`

● Es posible encontrar la longitud de una matriz a través de la propiedad .length

○ Nunca modifique una matriz en la que se le solicite.

● Atención, los Array no son matrices asociativas.

● Los Array tienen métodos para casi todo; .push(), .pop(), .splice(…), …

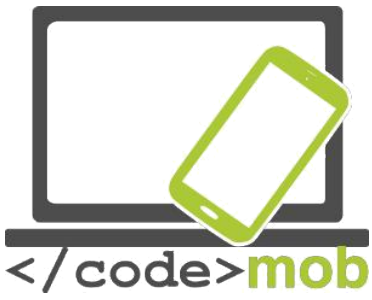
○ Es posible ordenar las cadenas de caracteres de una matriz usando el método .sort ()

○ y los dígitos a través de .sort () utilizando una función de comparación de argumentos

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

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

Ejercicio: Muestra el último elemento de una matriz cuya longitud no conoce.



Probando una Variable Type

Hay muchos métodos para probar el tipo de una variable.

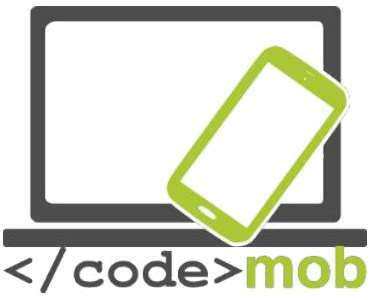
- **typeof** myVariable
    - **typeof** myVar ==='undefined'
    - El tipo de dato NaN es un número
    - El tipo de dato de un Array es un objeto.
    - El tipo de dato de Date es un objeto.
    - El tipo de dato Null es un objeto
    - El tipo de dato de una variable indefinida es un “undefined”

Ejemplos:

```
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 con **var** es local y global si se declara sin ello. En resumen: es recomendable **siempre declarar las variables** (evitan satural el alcance global, ECMA6, ...).

Su vida útil es diferente: una variable local se destruye al final de la función que lo contiene (contexto de ejecución actual), una variable global cuando se cierra la página HTML.

Hay, por supuesto, más sutilezas que eso, pero ten en cuenta que debemos usar **var** si declaramos una variable sin asignarla y que solo se puede borrar una variable no declarada (lo cual es desaconsejable de todos modos).

Alcance

En JavaScript, los objetos y las funciones también son variables.

El ámbito es el conjunto de variables, funciones y objetos a los que tiene acceso en un momento dado. Por lo tanto, cambia dentro de una función (ámbito de la función).

El ámbito siempre se remonta desde lo local a lo general, es decir, si JavaScript no encuentra una variable en su ámbito, entonces lo buscará en Global y finalmente generará un error si no encuentra nada.

Hoisting (utilización de variables antes de ser declaradas)

Javascript siempre declara las variables y funciones antes de ejecutar cualquier código: en primer lugar, las mueve al principio del bloque al que pertenecen.

Siempre es **recomendable declarar variables y funciones al comienzo** de su alcance.

Condiciones

Las sentencias condicionales se utilizan para realizar **diferentes acciones basadas en diferentes condiciones.**

Declaraciones condicionales

Muy a menudo cuando se escribe código, se desea realizar diferentes acciones para diferentes decisiones.

Puedes utilizar declaraciones condicionales en tu código para hacerlo.

En JavaScript tenemos las siguientes declaraciones condicionales:

- Utiliza **if** para especificar un bloque de código a ejecutar, si una condición especificada es verdadera
- Utiliza **else if** para especificar un bloque de código a ejecutar, si la misma condición es falsa
- Utiliza **else** si especifica una nueva condición para probar, cuando la primera condición es falsa
- Utiliza **switch** para especificar muchos bloques alternativos de código a ejecutar

If ... then ... else ...
Observa el espacio entre **else** y **if** (y no **elseif** como en PHP por ejemplo).

```
if(money <0){

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

elseif(money <10000){
```

```
        status ="Could be worse...";

}else{

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

Operador condicional

Una escritura compacta de un **If ... then**, útil para declaraciones condicionales de variables por ejemplo.

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

Switch

Es una forma sencilla de navegar por muchas opciones.

Tenga en cuenta que un **switch** hace posible tener varios casos apuntando a la misma variable de código y que utiliza una **strict comparison** **(**comparación 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 : )";

}
```

Operadores lógicos

- == (diferente de =), !=
- ===, !== : Comparación de valor y tipo
- >, >=, <, <=
- && (y)
- || (o)
- ! (no)
    - toggle = !toggle // verdadero se convierte en falso y falso se convierte en verdadero.

JavaScript, como todos los idiomas, se detiene en el primer false en una condición con && (y), lo que hace posible probar primero la existencia de una variable y luego trabajarla sin error si no existe.

JavaScript no tiene un XOR (OR exclusivo) pero es fácilmente reemplazable por una función.

	&& (AND)	|| (OR)	XOR
Cierto & Cierto	Cierto	Cierto	False
Cierto & Falso	Falso	Cierto	Cierto
Falso & Cierto	Falso	Cierto	Cierto
Falso & Falso	Falso	Falso	Falso

Ejercicio: Cree una condición con dos parámetros que reemplacen XOR.

```
var a =true;            // Prueba 4 veces sobre cierto, cierto, falso, falso.
var b =true;            //                          cierto, falso, falso, cierto

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

```
};
```

Loops (bucles)

```
// Bucle FOR
for(vari =0;i <10;i++){...};
// Bucle optimizado FOR
for(vari =0,len =myArray.length;i <len;i++){...};
// Bucle FOR IN para iterar las propiedades de un objeto.
for(prop in obj){
        /    prop
        /    obj[prop]
};
// Bucle
vari =0;

while(i <10){
          // ...
           i++; // Sin el incremento, el bucle es infinito.
};


// Bucle ejecutado al menos una vez.

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

Los navegadores tienen un tiempo de ejecución máximo para JavaScript.

Break, continue & label

Break y **continue** ofrecen control sobre las iteraciones de un bucle.

Label, especifica un lugar en el script al que es posible apuntar. Debe declararse antes de la pausa o continua.

- **break**: deja el bucle
  - break label: Puede salir de cualquier bloque de código (múltiples bucles)

- **continuous**: Pasa la iteración actual del bucle

Ejercicio: Cree una lista con viñetas que escanee tres veces el contenido de una Array (con una breve explicación de getElementById & innerHTML).

Try catch

Un bloque try catch puede 'probar' un poco de código y recuperar el error que podría generar.

```
try{
// Un bloque de código puede generar un error.
}
catch(e){
// Bloque de código para gestionar un error.
}
finally{
// Bloque de código que será ejecutado independientemente del resultado
// try catch (vuelve u otra excepción)
}
```

La palabra **throw** permite lanzar un error (incluyendo tu propio tipo).

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

Funciones

Las funciones tienen muchas ventajas:

- Agrupación de código
- Mejora la legibilidad
- Reutilización y refactorización
- Generalización (mismo código, valores diferentes)

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

Construye un hábito para realizar sistemáticamente funciones que siempre devuelven algo y cuyo tipo de datos es coherente.

Los argumentos en exceso se ignoran y los argumentos no declarados tienen 'undefined' como un valor; haz tus propias pruebas de validación en la función a través del objeto **arguments** que es globalmente similar a una array (.length and index via [...]).

JavaScript Function Syntax (Sintaxis de la función JavaScript)

Una **función** JavaScript se define con la palabra clave **function**, seguida de un nombre seguido de paréntesis ().

Los nombres de funciones pueden contener letras, dígitos, subrayados y signos de dólar (las mismas reglas que las variables).

Los paréntesis pueden incluir nombres de parámetros separados por comas: (**parameter1, parameter2**, ...)

El código a ejecutar por la función se coloca dentro de los paréntesis: **{}**

```
function name(parameter1, parameter2, parameter3) {
    código a ejecutar
}
```

Los **parámetros** de función son los **names (**nombres) enumerados en la definición de la función.
Los argumentos de **función** son los **values (**valores) reales recibidos por la función cuando se invoca.

Dentro de la función, los argumentos (los parámetros) se comportan como variables locales.

Función, Invocacion y Retorno

Invocación de las funciones

El código dentro de la función se ejecutará cuando invocas o llamas a "algo" **invokes** (invoca o llama) la función:

- Cuando se produce un evento (cuando un usuario hace clic en un botón)
- Cuando se invoca (se llama) desde código JavaScript
- Automáticamente (auto invocado)

Retorno de funciones

Cuando JavaScript llega a una declaración de devolución (**return statement)**, la función dejará de ejecutarse. Si la función se ha invocado desde una sentencia, JavaScript "devolverá" para ejecutar el código después de la instrucción invocación. Las funciones a menudo calculan un **return value** (valor de retorno). El valor de retorno se devuelve al "llamar la función":

Ejemplo: Calcula el producto de dos números y devuelve el resultado:

```
var x = myFunction(4, 3);        // la función es llamada y devuelve el valor a la variable x
function myFunction(a, b) {
    return a * b;                // la función devuelve el producto de a por b
}
```

El resultado de x será 12

Call and function reference (Referencia de llamada y función)

Una variable puede, por supuesto, hacer referencia a una función a través de su nombre. La llamada se realiza mediante paréntesis () y sus posibles argumentos.

High order function (Función de orden superior) es cualquier función que acepta una o más funciones como un parámetro o devuelve otra función.

Anonymous Functions (funciones anónimas)

```
Var myFunction =function(message){ alert(message); }; myFunction('Esto es un test'); // Muestra: esto es un test
```

Ejercicio: Crear una función factorial (n) {...} que devuelva el factorial de n.

Closure (cierre)

El sujeto es específico de JS y demasiado amplio para una introducción, pero recuerda al menos que cuando veas la palabra función en otra función, la función interna tiene acceso a las variables de la función externa (como un ámbito "regional" entre el local y el global).

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

Objects (objetos)

A diferencia de las variables String, Number o Boolean, los objetos pueden contener varios valores como **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 (objetos anónimos)

Un objeto, al igual que otros tipos de datos, no necesariamente tiene que ser nombrado. Esto suele ser el caso de un objeto de configuración utilizado como parámetro de una clase.

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

Common objects (objetos comunes)

Muchos objetos útiles están directamente disponibles: documento, ventana, Math, etc.

- Object: por ejemplo document
  - Propiedad: por ejemplo .innerHTML o .textContent
  - Método: por ejemplo .getElementById()
  - Palabra clave: this

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

La palabra clave devuelve el objeto que 'posee' la pieza de código, por lo que en un objeto será el propio objeto. Esto se puede discutir en detalle en jQuery.

Asignación por valor y por referencia

En JavaScript, los tipos de datos complejos (matriz y objeto) se asignan por referencia y no por valor. Dependiendo de sus necesidades, tendrá que codificar un script para copiar su tabla u objeto (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 es el DOM?)

El DOM es un standar de W3C (World Wide Web Consortium).

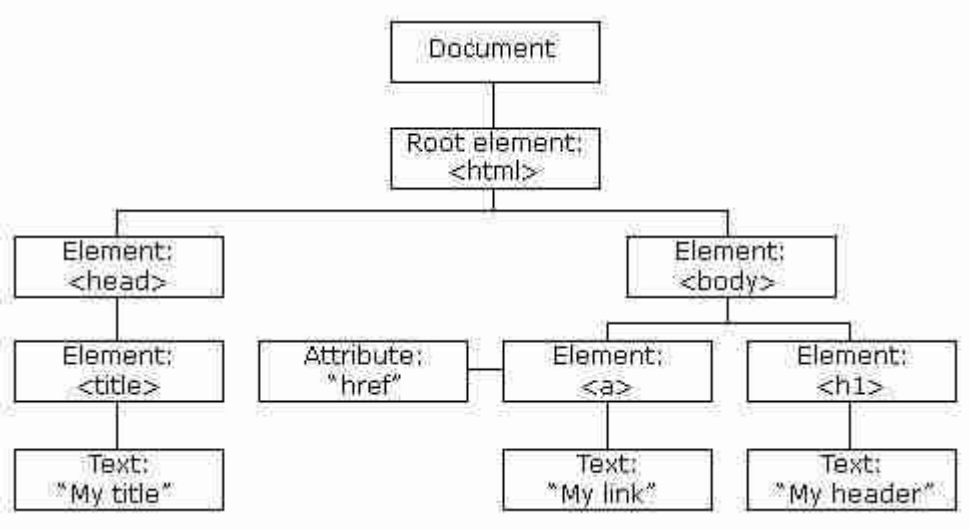
El DOM define un estándar para acceder a los documentos:

"El Modelo de Objetos de Documento (DOM) del W3C es una plataforma y una interfaz de lenguaje neutral que permite a los programas y scripts acceder y actualizar dinámicamente el contenido, la estructura y el estilo de un documento".

El estándar W3C DOM está dividido en 3 partes diferentes:

- Core DOM - modelo estándar para todos los tipos de documentos
- XML DOM: modelo estándar para documentos XML
- HTML DOM - modelo estándar para documentos HTML

JavaScript le permite navegar, leer y modificar el DOM (Document Object Model)



Ejercicio: Convierte el diagrama anterior en una página HTML.

El documento object y la selección de una porción de HTML

El objeto de documento representa todo el documento HTML (raíz) cuando se carga por la ventana del navegador. Propone muchos métodos para apuntar una parte como por ejemplo:

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

GetElementByID y querySelector respectivamente devuelven el id correspondiente y solo la primera ocurrencia **encontrada** o **nula** en los casos opuestos.

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

Estos métodos devuelven una lista de nodos (lista de nodos) que tiene la propiedad **.length a** cuyos índices se accede a través de los corchetes [], por lo que se puede recorrer a través de un bucle (loop). Sin embargo, esto no es una matriz.

Tenga en cuenta que para los selectores CSS complejos, el método .querySelector todavía no es mejor que las bibliotecas específicas como jQuery ni en términos de rendimiento ni en términos de facilidad de uso. Además, desde su concepción, algunas personas (como John Resig) han criticado su implementación.

Leer & cambiar HTML

¿Qué es el DOM HTML?

Aquí hay una gran explicación de ello:

Puede cambiar rápidamente el contenido y los atributos de una etiqueta HTML con .innerHTML (en Lectura y Escritura) y un Getter / Setter para los atributos.

```
// 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";
```

Observe la diferencia entre .innerHTML y .textContent, el primero es analizado y por lo tanto acepta HTML y el segundo no lo es. **TextContent** será más rápido y más **seguro**.

Cambiar el CSS

Todas las propiedades CSS son accesibles a través de la sintaxis señalada bajo la propiedad style con las siguientes características:

Las propiedades se escriben en la casilla Lower Camel (fontsize → fontSize)

Los valores son siempre cadenas (por ejemplo, no 250 pero sí "250px")

El nuevo valor se agregará en el estilo en línea

La posición es por .offsetTop y .offsetLeft (ni derecha ni abajo)

Anchos y alturas totales (relleno, borde) a través de .offsetWidth y .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 normalmente recupera el estilo en línea. Si desea que los estilos CSS resultantes de una hoja de estilo externa o la etiqueta <style>, utilice .getComputedStyle ():

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

Cambiar el DOM

Algunos métodos para crear e insertar etiquetas HTML en el DOM.

- **Document.createElement ("htmlTag")** Crea una etiqueta HTML
- **Document.createTextNode ("Hello world")** Crea contenido de texto
- **.removeChild (childToBeRemoved)** Elimina un elemento secundario
- **.appendChild (newContent)** Agrega un nuevo elemento al final del padre
- **.insertBefore (newElement, currentElement)** Agrega un elemento nuevo
- **.replaceChild (newElement, currentElement)** Sustituye un elemento

Algunos métodos útiles para explorar el DOM.

- **ParentNode** Muestra el nodo principal
- .**children [index]** Se dirige a los hijos a través de un índice a partir de 0
- **.nextElementSibling** señala el siguiente nodo semejante (que tiene el mismo padre)
  - No confunda con .nextSibling que también devuelve espacios entre nodos y comentarios.
- **.previousElementSibling** Se dirige al nodo anterior semejante

...

¡Prueba tu mismo!

JavaScript HTML DOM Elements:

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

Changing CSS:

Lista completa de propiedades y métodos

Ejemplos W3Schools [1][3]

El objeto Screen

El objeto **screen** proporciona información sobre la pantalla del usuario. Esta información **no es accesible** por un lenguaje de scripting del lado del servidor.

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

El objeto Navegador

El objeto Navegador ofrece información sobre el navegador del usuario.

- **navigator.userAgent:** El nombre completo del navegador
- **navigator.platform:** El sistema operativo del usuario
- **navigator.onLine:** Si el usuario está en línea o fuera de línea
- **navigator.language:** El idioma de la interfaz del navegador
- **navigator.geolocation:** Geolocalización tras el acuerdo del usuario
- **navigator.cookieEnabled:** Si el usuario acepta cookies
- …

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

Events (eventos)

Es posible vincular cualquier evento (clic, doble clic, sobre, error, redimensionar, ...) a un elemento del DOM a través de la siguiente sintaxis:

element.addEventListener(event, function, useCapture);

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

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

El nombre del parámetro event es una cadena que ignora el 'on' de atributos HTML (ejemplo: <a href="" onClick="..." /> → clic).

Lista completa en W3Schools:

Básicamente, los eventos HTML son "acciones" que suceden a los elementos HTML.

Cuando JavaScript se utiliza en páginas HTML, JavaScript puede "reaccionar" en estos eventos.

Los eventos son acciones o ocurrencias que ocurren en el sistema que está programando, de las que el sistema le informa para poder responder a ellas de alguna manera, si así lo desea. Por ejemplo, si el usuario hace clic en un botón de una página web, puede responder a esa acción mostrando un cuadro de información.

En el caso de la Web, los eventos se desencadenan dentro de la ventana del navegador y tienden a conectarse a un elemento específico que reside en él: puede ser un solo elemento, un conjunto de elementos, el documento HTML cargado en la pestaña actual o toda la ventana del navegador. Hay muchos tipos diferentes de eventos que pueden ocurrir, por ejemplo:

- El usuario hace clic con el ratón sobre un determinado elemento o coloca el cursor sobre un determinado elemento.
- El usuario presiona una tecla en el teclado.
- El usuario cambia el tamaño o el cierre de la ventana del navegador.
- La carga de una página web ha acabado.
- Se envía un formulario.
- Se reproduce un video, se pausa o se termina la reproducción.
- Se ha producido un error.

Cada evento disponible tiene un controlador de eventos (**event handler)**, que es un bloque de código normalmente definido por el desarrollador que se ejecutará cuando se desencadene el evento. Cuando dicho bloque de código se define para ser ejecutado en respuesta a un disparador de evento, decimos que estamos registrando un manejador de eventos (**registering an event handler)**. Tenga en cuenta que los *event handlers* a veces se llaman **event listeners (oyentes)** - son bastante intercambiables para nuestros propósitos, aunque en sentido estricto trabajan juntos. El oyente escucha el acontecimiento que sucede, y el encargado (handler) es el código que se corre en respuesta a él que sucede.

Un ejemplo sencillo:

En el ejemplo siguiente, tenemos un solo elemento <button>, que cuando se presiona, hará que el fondo cambie a un color aleatorio:

```
1 <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 este código, almacenamos una referencia al botón dentro de una variable llamada btn, utilizando la función Document.querySelector (). También definimos una función que devuelve un número aleatorio. La tercera parte del código es el manejador de eventos. La variable btn apunta a un elemento <**button**>, y este tipo de objeto tiene un número de eventos que pueden disparar sobre él, y por lo tanto, los manejadores de eventos disponibles. Estamos escuchando la activación del evento click, estableciendo la propiedad **onclick event handler** como igual a una función anónima que contiene código que generó un color RGB aleatorio y establece el color de fondo <**body**> igual a él.

Este código se ejecutará ahora cada vez que se active el evento de clic en el elemento <button>, es decir, cada vez que un usuario haga clic en él.

Código y resultado:

Código y resultado: https://codepen.io/pen/

Event handler properties (Propiedades del controlador de eventos)

Volviendo al ejemplo anterior:

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

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

La propiedad **onclick** es la propiedad del manejador de eventos que se utiliza en esta situación. Es esencialmente una propiedad como cualquier otra disponible en el botón (por ejemplo, btn.textContent, o btn.style), pero es un tipo especial - cuando se establece que sea igual a algún código, ese código se ejecutará cuando se presiona el botón.

Event Management (gestor de eventos)

El objeto **Event** representa cualquier evento DOM. A medida que se pasa en el argumento, podemos utilizarlo en nuestra función que maneja el evento.

Tiene muchas propiedades y métodos ...

- **Event.target** El objeto que emitió el evento
- **Event.currentTarget** Devuelve el elemento que se adjuntó el evento
- **Event.preventDefault ()** Cancela el comportamiento normal
- **Event.stopPropagation ()** Cancela la propagación del evento

Capturing & Bubbling

Capture es la fase descendente (desde el elemento exterior hasta el elemento interior) y el **bubbling** la fase ascendente (Como un buceador que se sumerge y asciende).

El valor predeterminado de useCapture es falso y IE9 sólo admite el evento de **bubbling**.

Eliminar los controladores de eventos

Usa **.removeEventListener ()** pero no es posible eliminar funciones anónimas. Además, cada manejador de eventos debe eliminarse por separado, un evento en **bubbling** y un evento en la **capturing** en el mismo evento son dos **eventListener** diferentes.

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

Quizz

Aquí hay una pequeña prueba que deberías ser capaz de pasar ahora 🛳
Encontrarás las respuestas al final de la misma. ¡Trata de no hacer trampa!

Preguntas

Introducción

1. Cuando tu programa no funciona, lo primero que debes hacer es mirar los mensajes de error. ¿Cómo se abre la consola web tanto en Firefox como en 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. Puedes declarar una variable con y sin la palabra clave **var**. ¿Qué camino debes utilizar?*

- Ambos, no hay ninguna diferencia.
- Una variable declarada con la palabra clave 'var' pertenece al ámbito local, la otra pertenece al ámbito global. Utilice siempre var para evitar contaminar el ámbito global.
- Una variable declarada con la palabra clave 'var' pertenece al ámbito global, la otra pertenece al ámbito local. Nunca utilice var para evitar contaminar el ámbito global.

3. ¿Por qué siempre debes escribir comentarios?

- Es bueno explicar la lógica y los detalles de un programa. Tanto para ti mismo si estás trabajando en un proyecto nuevo, en uno antiguo o cuando eres parte de un equipo.
- Los comentarios son una pérdida de tiempo, nadie los escribe de todos modos.
- Es la manera correcta de desactivar temporalmente alguna parte de un código sin perderlo.

Condiciones

1. ¿Hay alguna diferencia lógica entre una estructura 'if else' y dos consecutivas 'if' que tienen la condición opuesta?

- No, pero el 'if else' es mucho mejor porque es más legible y menos propenso a errores.

- Sí, no son lo mismo.

2. ¿Cómo puedes recorrer todos y cada uno de los elementos de una matriz de longitud desconocida?

- Mediante el uso de la propiedad array .length.

- Mediante el uso del método array .length ().

- Mediante el uso de la función count ().

3. *¿Qué mejora el siguiente código?:*
for (var i = 0, len = a.length; i < len, i++) { /* ... */ }
comparado con

for (var i = 0; i < a.length, i++) { /* ... */ }
?

- Nada
- La variable 'len' sólo se evalúa una vez en lugar de una vez por iteración.
- El número de iteración es diferente.

Funciones

1. ¿Es importante el orden de los argumentos de una función llamada 'divide' (que devuelve el resultado de una división del primer argumento por la segunda)?

- Sí, cambia el resultado.
- No

2. ¿Cuál es la diferencia entre una 'function call' y una 'function reference' ?

- Una 'function call' (su nombre seguido de paréntesis) solicita la ejecución del código de función en el momento de la llamada. Una 'function reference' (en una llamada atrás, por ejemplo) simplemente almacena el nombre de la función para referencias futuras.
- No hay ninguna diferencia si la función no tiene ningún argumento.

3. ¿Qué palabra clave reservada se utiliza para especificar el valor devuelto de la llamada de una función?

- Return
- Pass
- Exit

DOM

1. ¿Qué métodos no se refieren a la primera o única ocurrencia encontrada?

- document.querySelector()
- document.querySelectorAll()

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

2. En el código siguiente, ¿qué argumento es recibido por la devolución de llamada?
document.getElementById('myID').addEventListener('click', myCallBack);

- El evento que activó la devolución de llamada.
- No se pasó ningún argumento a través de la devolución de llamada.

3. ¿Cómo puedes obtener el valor de un elemento HTML (UI) como la de una etiqueta?

- A través del the property .value que devuelve el contenido del atributo 'valor'.
- Mediante el método .getValue() que devuelve el contenido del atributo 'value'.
- Mediante el método .val() que devuelve el contenido del atributo 'value'.

CSS

1. ¿Qué selector CSS permite seleccionar sólo el párrafo que es hijo directo de la caja div?

- div > p
- div p
- div, p

2. ¿Qué selector CSS permite seleccionar sólo el primer párrafo independientemente del número de hermanos HTML que tenga?

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

3. Si dos selectores CSS (.red y #blue) apuntan al mismo elemento, ¿cuál será su color?

- Blue
- Red
- Purple
- Depende: el que sea declarado último.

Respuestas

Introducción

1. Cuando tu programa no funcione, lo primero que debes hacer es mirar los mensajes de error. ¿Cómo se abre la consola web tanto en Firefox como en 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. Puedes declarar una variable con y sin la palabra clave **var**. ¿Qué camino debes utilizar?*

- Una variable declarada con la palabra clave 'var' pertenece al ámbito local, la otra pertenece al ámbito global. Utilice siempre var para evitar contaminar el ámbito global.

3. ¿Por qué siempre debes escribir comentarios?

- Es bueno explicar la lógica y los detalles de un programa. Tanto para ti mismo si estás trabajando en un proyecto nuevo, en uno antiguo o cuando eres parte de un equipo.
- Es la manera correcta de desactivar temporalmente alguna parte de un código sin perderlo.

Condiciones

1. ¿Hay alguna diferencia lógica entre una estructura 'if else' y dos consecutivas 'if' que tienen la condición opuesta?

- No, pero el 'if else' es mucho mejor porque es más legible y menos propenso a errores.

2. ¿Cómo puedes recorrer todos y cada uno de los elementos de una matriz de longitud desconocida?

- Mediante el uso de la propiedad array .length.
- Mediante el uso del método array .length().
- Mediante el uso de la función count().

3. *¿Qué mejora el siguiente código?:*
for (var i = 0, len = a.length; i < len, i++) { /* ... */ }
comparado con
for (var i = 0; i < a.length, i++) { /* ... */ }
?

- La variable 'len' sólo se evalúa una vez en lugar de una vez por iteración.

Funciones

1. ¿Es importante el orden de los argumentos de una función llamada 'divide' (que devuelve el resultado de una división del primer argumento por la segunda)?

- Sí, cambia el resultado.

2. ¿Cuál es la diferencia entre una 'function call' y una 'function reference' ?

- Una 'function call' (su nombre seguido de paréntesis) solicita la ejecución del código de función en el momento de la llamada. Una 'function reference' (en una llamada atrás, por ejemplo) simplemente almacene el nombre de la función para referencia futura.

3. ¿Qué palabra clave reservada se utiliza para especificar el valor devuelto de la llamada de una función?

- Return

DOM

1. ¿Qué métodos no se refieren a la primera o única ocurrencia encontrada?

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

2. En el código siguiente, ¿qué argumento es recibido por la devolución de llamada?
document.getElementById('myID').addEventListener('click', myCallBack);

- El evento que activó la devolución de llamada.

3. ¿Cómo puede obtener el valor de un elemento HTML (UI) como la entrada de una etiqueta?

- A través del the property .value que devuelve el contenido del atributo 'valor'.

CSS

1. ¿Qué selector CSS permite seleccionar sólo el párrafo que es hijo directo de la caja div?

- div > p

2. ¿Qué selector CSS permite seleccionar sólo el primer párrafo independientemente del número de hermanos HTML que tenga?

- p:nth-of-type(1)

3. Si dos selectores CSS (.red y #blue) apuntan al mismo elemento, ¿cuál será su color?

- Blue

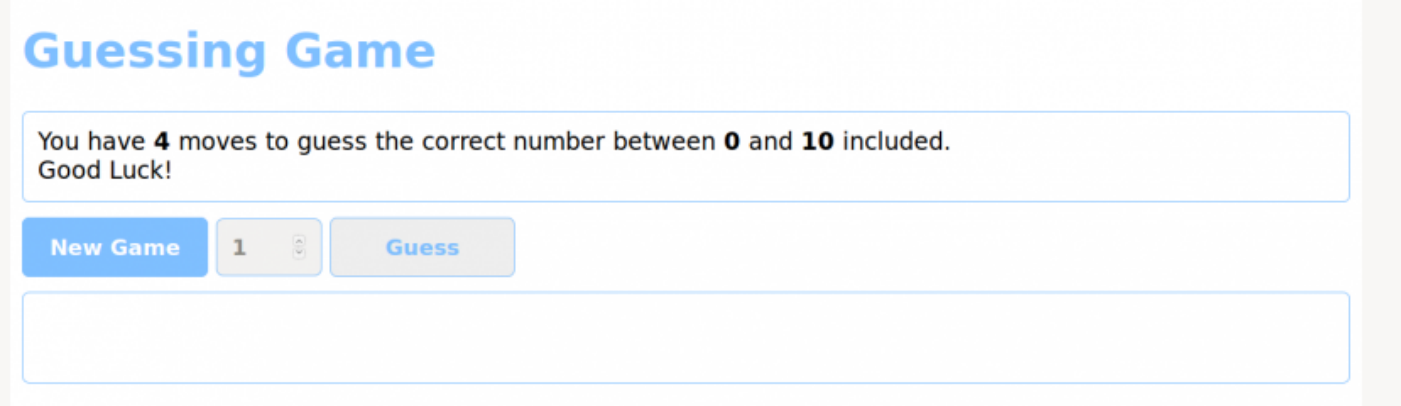
Coding lab: juego de adivinanzas

Ahora que entiendes los conceptos básicos de HTML / CSS / JavaScript, es hora de divertirte y hacer tu primer juego ...

En esta lección, recrearás el Juego de Adivinanzas. Eso significa que vas a crear:

1. Un archivo HTML en el que se pondrán todos los elementos requeridos por el juego
2. Un archivo CSS a través del cual se va a diseñar tu juego (de la forma que quieras ;-)
3. Un archivo JavaScript que contendrá todas las funciones requeridas por el juego

Esto es básicamente cómo debe ser (aunque el aspecto puede variar):



En la página siguiente, encontrarás una guía paso a paso para hacerlo, por si la necesitas.

Empezando

1. Necesitaremos al menos un campo numérico y un botón para obtener el resultado, pero también algún tipo de "contenedor" en el que escribiremos / anotaremos los resultados incluso si empieza vacío o invisible.

Trata de pensar si se necesita o no el botón 'Nuevo juego' tanto como usuario como como punto de vista del programador; y si es útil o no. Si lo deseas, al final del ejercicio puedes cambiar el código para permitir que un usuario abandone el juego actual, por ejemplo.

2. Trata de pensar en todos los pasos necesarios para programar este mini juego. Este será el comienzo de tus diferentes bloques de código. Si un problema parece difícil de resolver, lo más probable es que necesites cortarlo en trozos más pequeños.

Si necesitas reutilizar varias veces el mismo código / lógica, es mejor convertirlo en una función, especialmente si los parámetros de inicio pueden cambiar.

Siempre debes intentar utilizar variables en lugar de valores programados, de forma que puedas cambiar fácilmente los parámetros del juego como el número de hipótesis disponibles, por ejemplo. Mejora la legibilidad, la facilidad de mantenimiento y haz que tu código sea más configurable.

Comenta siempre tu código (explicando la lógica detrás del código y no el código) para ti, para futuras ocasiones, y también pensando en tus compañeros de equipo. :)

3. getRandomIntegerBetween() es un ejemplo simple de una función que mejora la legibilidad y convierte un pedazo de código en algo fácilmente reutilizable.

Ten en cuenta que la verdadera aleatoriedad no es algo trivial en las ciencias de la programación. Las computadoras son deterministas, lo que significa que si haces la misma pregunta, obtendrás la misma respuesta cada vez. De hecho, tales máquinas son específicamente y cuidadosamente programadas para eliminar la aleatoriedad en los resultados. Ver pseudo aleatorio en Google.

http://engineering.mit.edu/ask/can-computer-generate-truly-random-number

En checkResponse(), vemos que return también puede usarse para salir de una función sin devolver algo, pasando por alto la prueba inútil de 'game over' cuando el juego ya está ganado.

Finalmente, vemos que las tecnologías web como JavaScript tratan de acceder y actualizar el DOM HTML (Document Object Model) con cosas como document.getElementById ('userGuess') y result.innerHTML o el CSS. Cada una de esas tecnologías relativamente fáciles trabajan en tándem.

A continuación, encontrarás algunas pistas para la estructura y funciones ...

STRUCTURE AND FUNCTIONS

```
// Returna un entoro entre un valor mínimo y un valor máximo dado.
    function getRandomIntegerBetween(min, max)

// Activa la interfaz de usuario para un juego nuevo.
    function activateUI()

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

    function init(){
        // Reajusta el máximo.
        // Limpia el registro de logs
        // Inicializa un nuevo valor aleatorio
        activateUI();
    }

// chequea la respuesta
function checkResponse()

// jugadas disponibles antes de perder el juego

    // Número aleatorio estimado por el jugador.

    // Valores mínimos y máximos (ambos incluídos) entre los que se puede escoger.

    // Reglas de visualización

    // Registro de logs

    // ¿Puedes adivinar la forma óptima de ganar siempre este juego?
```

References & Resources

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

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

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