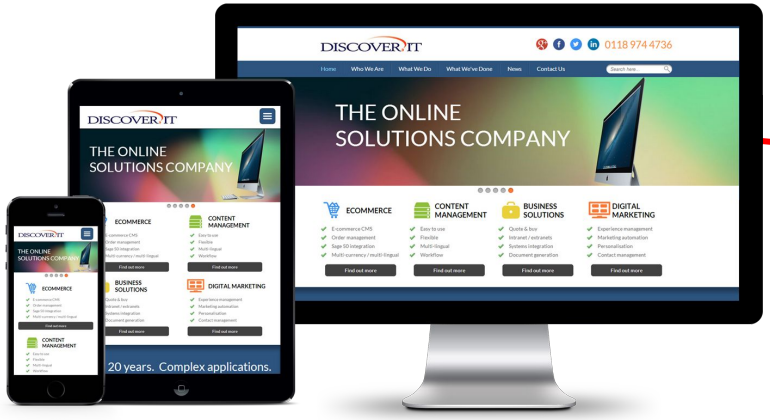
Javascript #1

WEB 1 TUDAI 2024



HTML

Markup Language
Content

CSS

Style sheet Language
Presentation

JS

Programming Language
Behavior

Javascript

Javascript (JS) es uno de los lenguajes de programación más populares del mundo.

“JavaScript es el lenguaje de programación de la Web”

- Es un lenguaje de programación interpretado.
- Es uno de los lenguajes más utilizados hoy en dia.

Nace con la necesidad de generar *dinamismo e interacción* en las páginas web.

Un poco de historia...

1995 — Brendan Eich desarrolla la primera versión llamada “LiveScript”

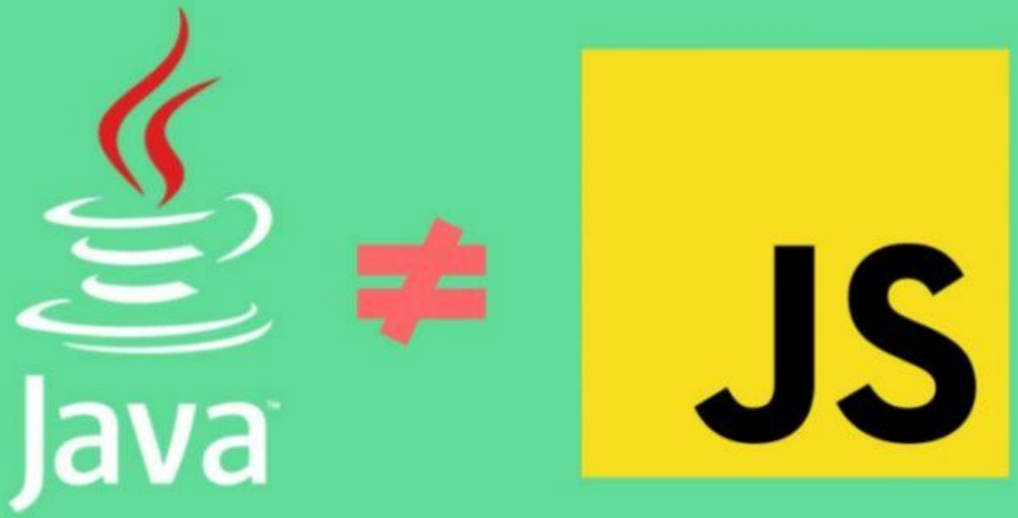
1997 — Es desarrollado para Netscape 2 y se convierte en estándar: ECMAScript (ECMA-262 / ES1)

1999 — Se lanza al tercer versión (ES3) y se populariza con la salida de Internet Explorer 5

2000/2005 — Gana popularidad con la moda de las aplicaciones web SPA

2009 — Se lanza ES5 siendo la primera revisión mayor del lenguaje

2015 — Luego de varios años aparece ES6 incorporando grandes cambios de sintaxis y estructura.



"Lo único que tienen en común
son las primeras 4 letras"

Para qué se usa JAVASCRIPT?

Si bien JS nació para ser ejecutado del **lado del cliente** (frontend) hoy en día es un potente lenguaje que es usado en muchos entornos fuera del navegador.

- **BACKEND**
  node.js
- **MOBILE APPS**
  react native
- **IoT**
  cyclon.js
- **DESKTOP APPS**
  electron

Javascript ¿para qué?

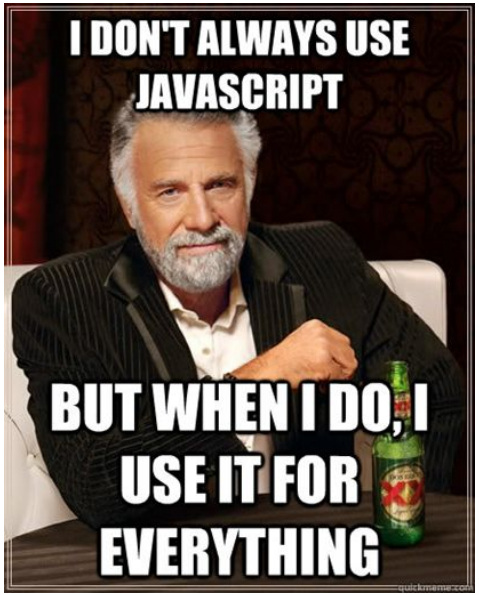
Algunos ejemplos de lo que vamos a hacer con JS en WEB1:

- Validar formularios (lo que no permita HTML5)
- Reaccionar a lo que haga el usuario (click, teclear, etc)
- Cambiar algo al pasar el mouse
- Partes de páginas que se muestran/ocultan
- Hacer cálculos complejos
- Carga dinámica de contenido (AJAX)
- **Hacer Single Page Applications (SPAs)**

Atwood's Law

Corolario del *Principle of Least Power*

Any application that can be written in JavaScript, will eventually be written in JavaScript.



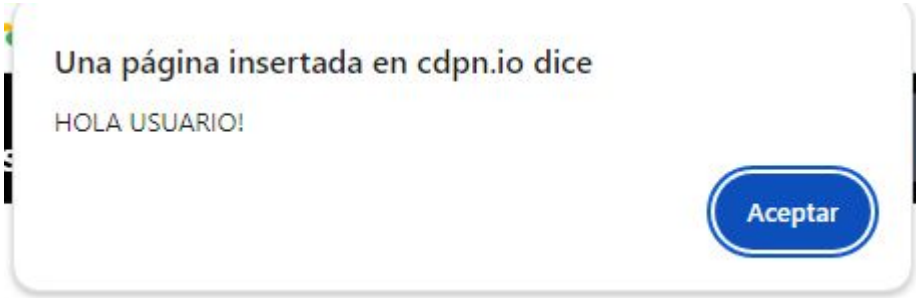
Ejemplos

Vamos aprender JS con ejemplos

JS - Ejemplo 1

Ejemplo 1: Mensaje al usuario

Mostrar un mensaje saludando al usuario cuando éste entra a la página.



¿Qué vamos a aprender?

1. Incluir un archivo Javascript y ejecutarlo
2. Mostrar una alerta (diálogo) por pantalla

Como incluir un Javascript

Existen varias maneras de incluir Javascript dentro un sitio web. Nosotros vamos a recomendar siempre crear un archivo nuevo para cada script:

```
<script type="text/javascript" src="js/main.js"></script>
```

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/style.css">
    <title>Ejemplo JS 1 (prende)</title>

    <script  type="text/javascript" src="js/main.js"></script>
</head>
```

DENTRO DEL HEAD DEL HTML

```
<body>
    <h1>Ejemplo JS 1</h1>
    <p>Lorem ipsum dolor sit</p>

    <script  type="text/javascript" src="js/main.js"></script>
</body>
```

AL FINAL DEL BODY

El código se ejecuta en donde se lo incluye

Función Alert

`alert(msg)`

La función alert nos muestra una alerta en el navegador

```
alert("mensaje")
```

No se suele usar en páginas reales, ya que no se integra visualmente con el resto del sitio

http://codepen.io/webUnicen/pen/eZMvzo

Función Alert

index.html main.js

classlist > index.html > html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/style.css">
    <title>Ejemplo JS 1 (prende)</title>
</head>
<body>
    <h1>Ejemplo JS 1</h1>

    <script  type="text/javascript" src="js/main.js"></script>
</body>
</html>
```

index.html main.js

classlist > js > main.js

```javascript
/*
Mi codigo inicial de Javascript
muestra un alert para comprobar que
el codigo se esta ejecutando.
*/

alert("HOLA USUARIO!");
```

El código Javascript incluido se ejecuta automáticamente al cargar la página

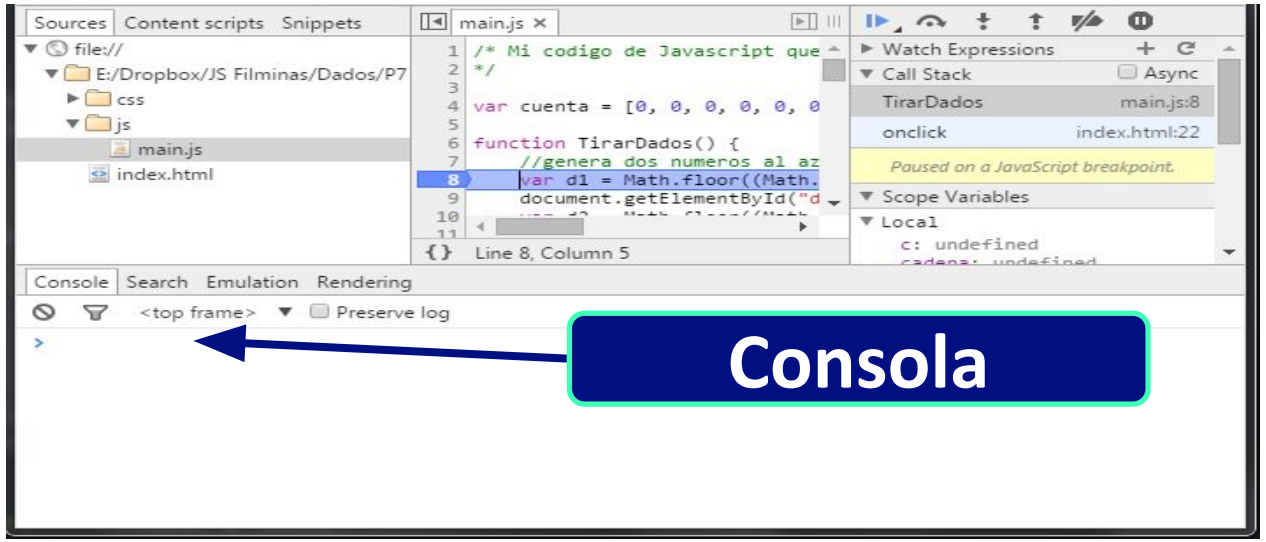
¿Qué pasa si hay dos alert() seguidos?

JS - Consola de desarrollo

Siempre abrimos la consola al desarrollar JS

`Menú > Más herramientas > Herramientas Desarrollador`

- Algunos errores del código se detectan al cargar el JS
- Otros errores luego de ejecutar esa línea



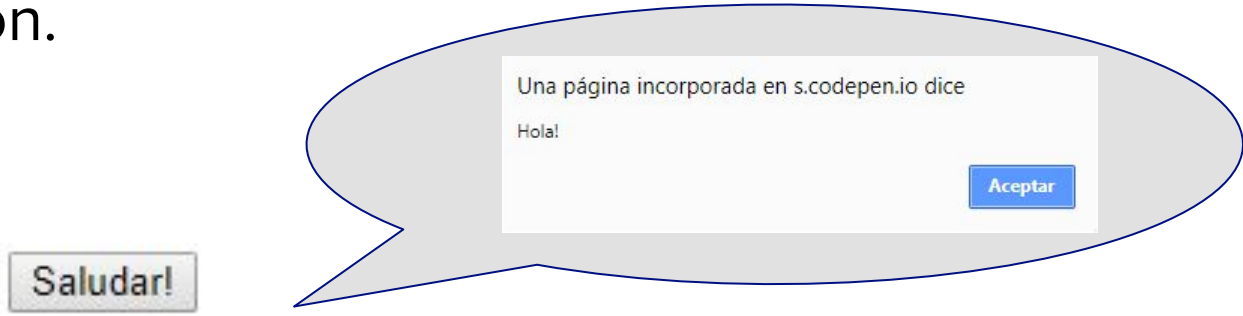
Atajo de Teclado:

- Ctrl + Shift + I
- F12

JS - Ejemplo 2

Ejemplo 2: Botón para saludar

Mostrar un mensaje saludando cuando el usuario haga click en un botón.



¿Qué vamos a aprender?

Ejecutar un código (**función**) al hacer click en un botón (**escuchar un evento**)

Funciones

En Javascript no hay diferencia entre funciones y procedimientos. Una función puede o no devolver datos

```
function saludar() {
alert("Hola");
}
```

Creación

Para llamar a esta función tengo que invocar su “firma”.

```
saludar();
```

Ejecución
(llamar a la función)

http://codepen.io/webUnicen/pen/YZmNwZ

Funciones: Para qué?

Una función le da un nombre a una porción de código

Sirve principalmente para:

- dividir un problema grande en varios problemas chicos
- reutilizar el código
- facilitar lectura del código

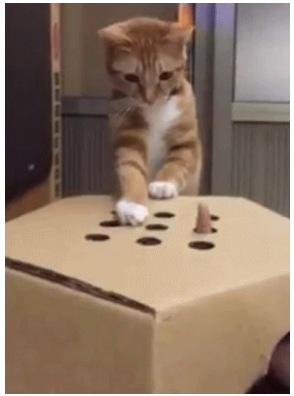
¿Cómo hago para ejecutar la función cuando el usuario aprieta un botón?

EVENTOS

Un evento es “algo” que ocurre en el sistema originado por el usuario (u otra parte del sistema) el cual produce una notificación al sistema.

EJEMPLOS

Click en un botón
Interactuar con un form
Se terminó de cargar la página
El usuario pasó el mouse sobre una imagen



Programación dirigida por eventos

En este modelo los programas esperan sin realizar ninguna tarea hasta que se produzca un evento. Una vez producido, ejecutan alguna tarea asociada a la aparición de ese evento y cuando concluye, el programa vuelve al estado de espera.

Eventos

Formas de manejar eventos

Existen varias formas diferentes de manejar eventos en Javascript:

Mediante **atributos HTML**

```
<button onclick="..."></button>
```

Mediante **propiedades Javascript**

```
onclick = function() { ... }
```

Mediante **listeners del DOM**

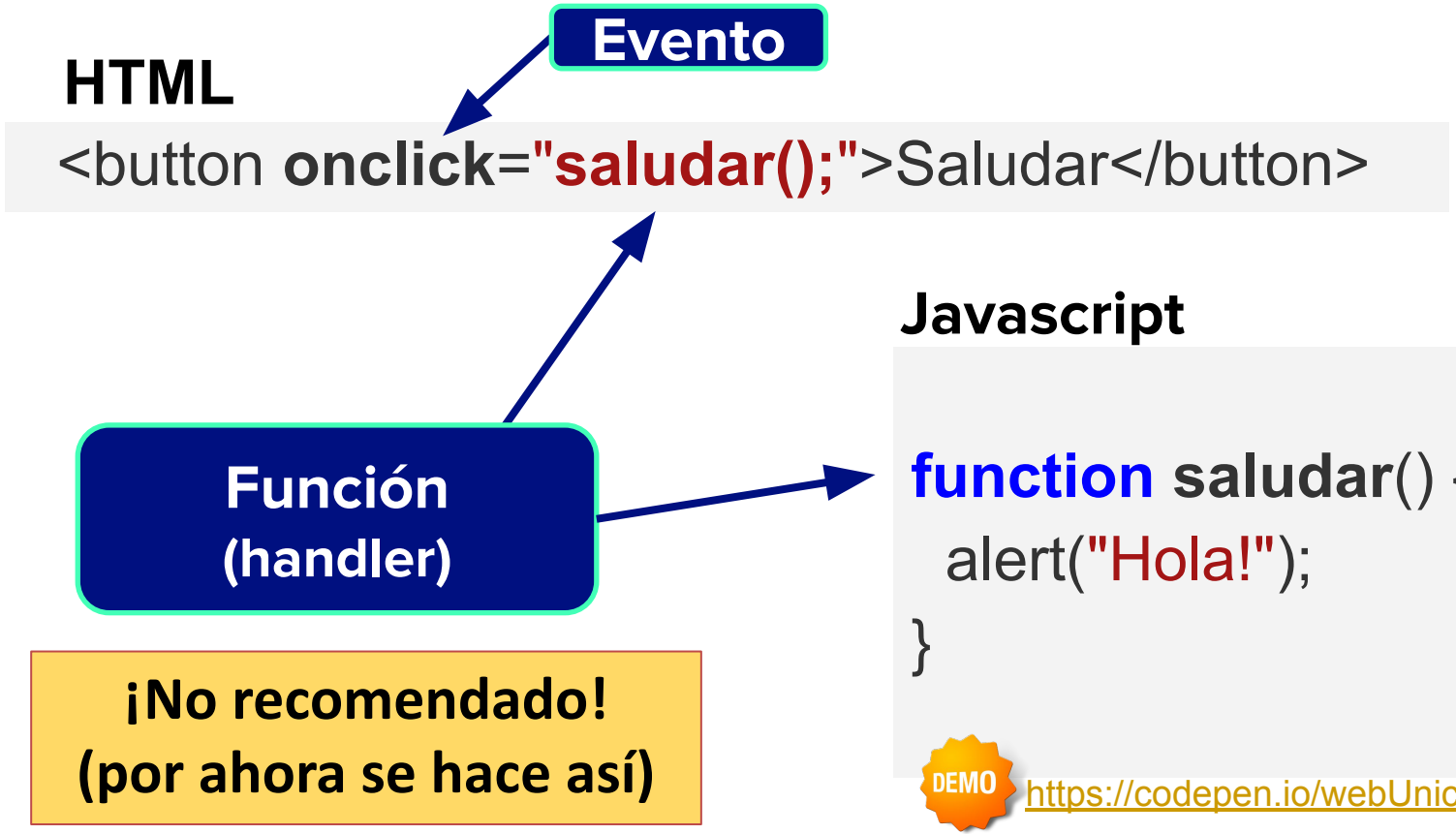
```
.addEventListener("click", ...)
```

Empezamos por esta solor por FACILIDAD

RECOMENDADA!

Eventos

Los eventos son capturados por manejadores (handlers).



Eventos

Ejemplos de eventos:

- onclick
- onkeydown
- onload
- onfocus
- onchange (recomendado para selects)
- ondrag
- oncopy
- onpause (para media)

Hay 50~100 eventos: http://www.w3schools.com/jsref/dom_obj_event.asp

Programación dirigida por eventos

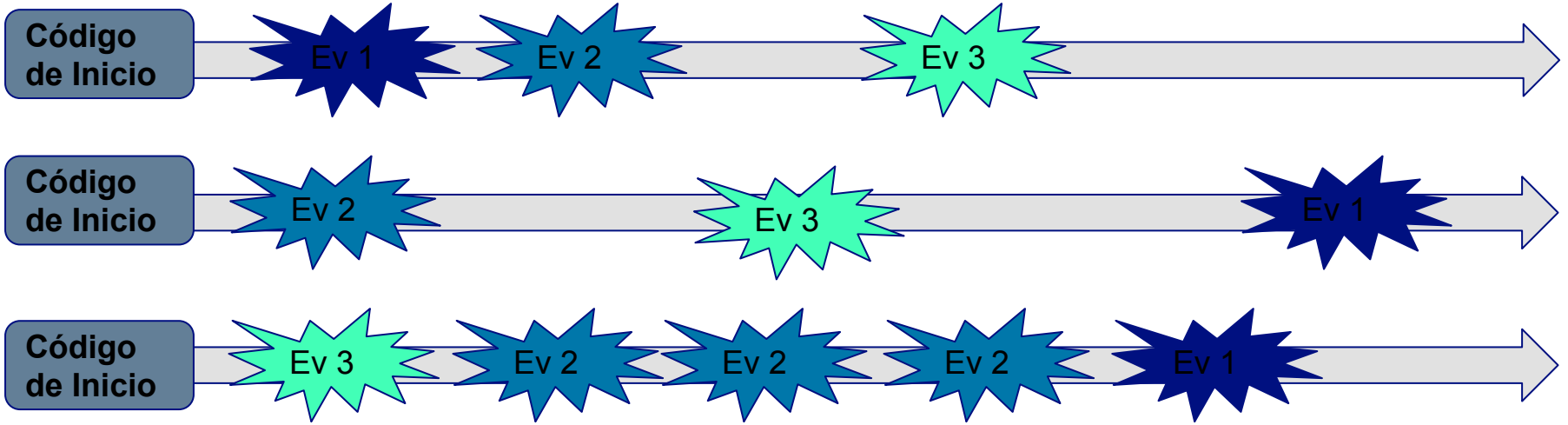
Programación secuencial

- Sabemos el flujo de la ejecución

Programamos dirigida por eventos

- No sabemos la secuencia exacta de ejecución
- Se disparan diferentes códigos con diferentes acciones

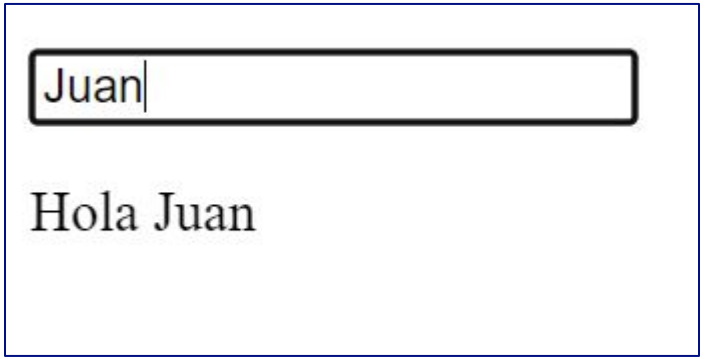
Ejemplo de tres ejecuciones diferentes:



JS - Ejemplo 3

Ejemplo 3: Saludar con nombre

A medida que escribo mi nombre en un input, la página irá escribiendo “**Bienvenido {NOMBRE}**”.



¿Qué vamos a aprender?

- Usar otro evento (que no es “onclick”)
- Editar el html desde Javascript
- Calcular el largo de una cadena

JS - Ejemplo 3

Pensemos en pasos:

1. **Leer el valor** que introduce el usuario mientras escribe

2. Guardar ese valor en una **variable**

3. **Modificar el HTML** y escribir el valor de esa variable

Variables y Constantes

Variables:

- Una variable es un nombre que le damos a un valor que puede cambiar (o no) con el tiempo (durante la ejecución del programa)

```
// declarar una variable
let nombre = "Pepe";
```


```
// antes de ES6
var nombre = "Pepe";
```

Constantes:

- Una constante es un tipo INMUTABLE, no puede cambiar una vez definida.
- Se usan para aumentar la legibilidad del programa

```
// declarar una constante
const MAX_LIMIT = 10;
```

Use Strict

Las variables pueden declararse de forma implícita.

La primera vez que uso una variable se declara “automáticamente”.

```
numero = 2; //declaró variable número mágicamente
```

ESTO PUEDE CAUSAR PROBLEMAS

Es una buena práctica escribir al comenzar un archivo Javascript

```
"use strict"
```

“El use strict hace que el navegador se ponga la gorra!”

- **Convierte en obligatoria la declaración de variables**
- Restringe otros posibles errores de sintaxis

Más info : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode

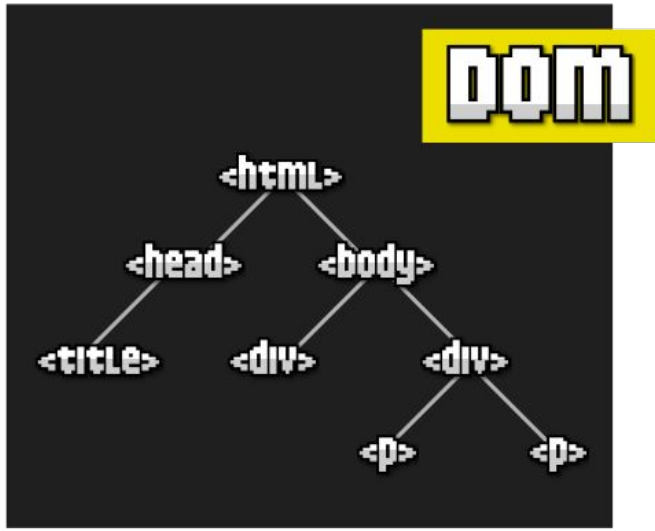
MANIPULACIÓN DEL DOM

Nos permite pasar de una página totalmente estática a una página más “dinámica” utilizando Javascript

El DOM (**Document Object Model**) es una manera de comprender las dependencias y relaciones entre los elementos HTML de una página web mediante un **diagrama de árbol**.

```
<html>
<head>
  <title>Title</title>
</head>
<body>
  <div></div>
  <div>
    <p>Párrafo 1</p>
    <p>Párrafo 2</p>
  </div>
</body>
</html>
```

HtML

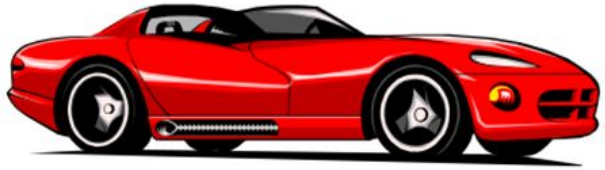


DOM - Objetos

El DOM es un árbol de objetos...

- En la vida real todos los objetos tienen una serie de características y un comportamiento.
- En programación, un objeto es una combinación de
  - **Campos o atributos**: almacenan datos. Estos datos pueden ser de tipo primitivo y/o otro tipo de objeto
  - **Rutinas o métodos**: lleva a cabo una determinada acción o tarea con los atributos.

```
Acceso a atributos y llamado de métodos con .
let auto = …
auto.arranca();
auto.color = rojo;
alert(auto.ruedas);
```



- Atributos:
  - color
  - velocidad
  - ruedas
  - motor
- Métodos:
  - arranca()
  - frena()
  - dobla()

Como manipular el HTML a través del DOM en dos simples pasos:

En Javascript, la forma de acceder al DOM es a través de un objeto llamado `document`

1. Al “documento” le pedimos el nodo del elemento que queremos editar. Puede ser por su “identificador”.

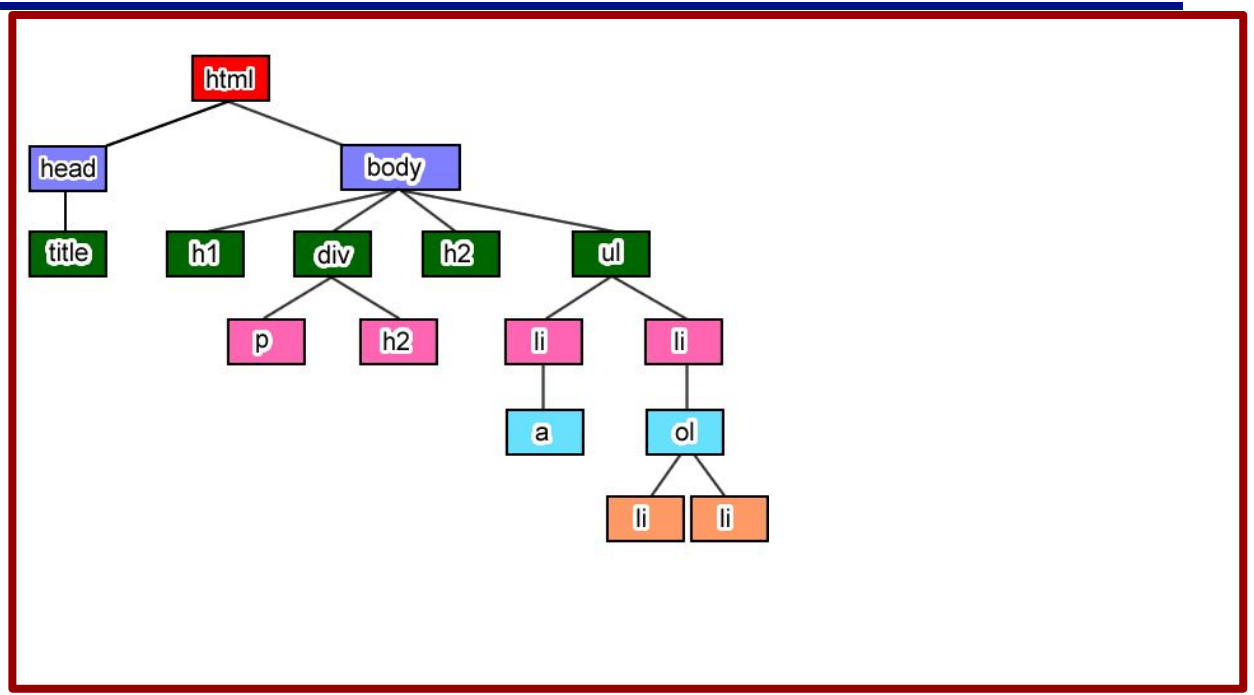
```
let elem = document.getElementById("identificador");
let elem = document.querySelector("#identificador");
```

2. A ese objeto (el nodo del arbol en cuestion) le modificamos los atributos que necesitemos con un nuevo valor.

```
elem.innerHTML = "nuevo valor";
```

DOM - Edición

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo Arbol</title>
    <link rel="stylesheet" href="estilo.css">
  </head>
  <body>
    <h1>Titulo 1</h1>
    <div>Div1
      <p>Parrafo dentro de div</p>
      <h2>Titulo 2 en div</h2>
    </div>
    <h2>Titulo 2</h2>
    <ul>
      <li>Elemento 1 <a href="...">Link1</a></li>
      <li>Elemento 2
        <ol>
          <li id="prim">Primero</li>
          <li id="seg">Segundo</li>
        </ol>
      </li>
    </ul>
  </body>
</html>
```

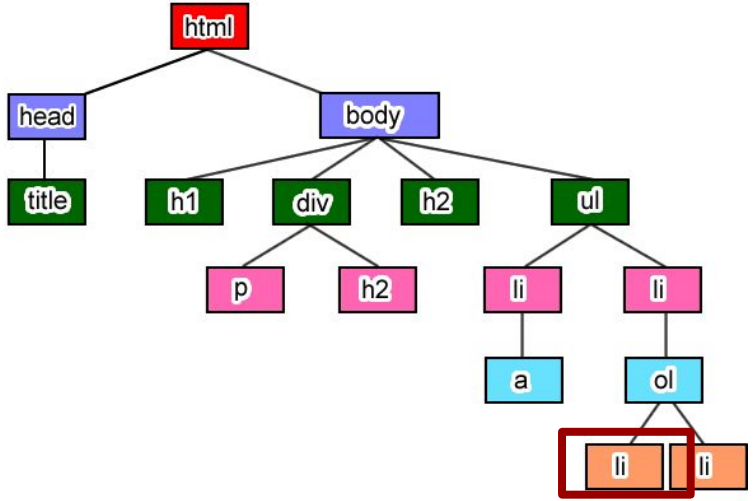



```
document.querySelector('#prim').innerHTML = "1- Primero";
```

Al **documento**

DOM - Edición

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo Arbol</title>
    <link rel="stylesheet" href="estilo.css">
  </head>
  <body>
    <h1>Titulo 1</h1>
    <div>Div1
      <p>Parrafo dentro de div</p>
      <h2>Titulo 2 en div</h2>
    </div>
    <h2>Titulo 2</h2>
    <ul>
      <li>Elemento 1 <a href="...">Link1</a></li>
      <li>Elemento 2
        <ol>
          <li id="prim">Primero</li>
          <li id="seg">Segundo</li>
        </ol>
      </li>
    </ul>
  </body>
</html>
```

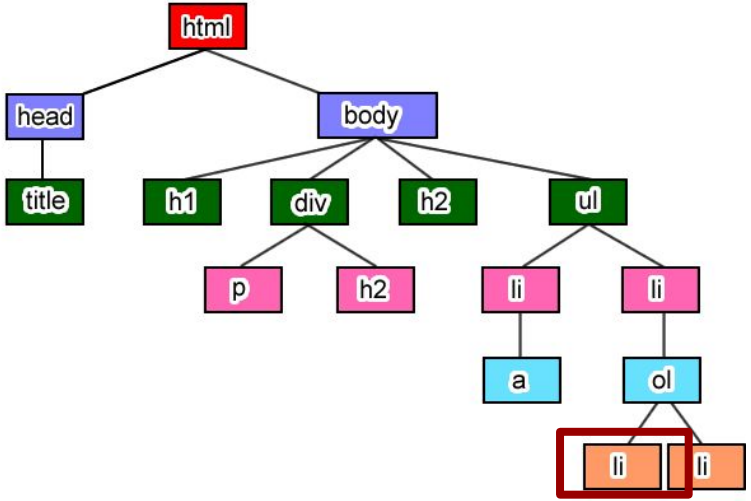



```
document.querySelector('#prim').innerHTML = "1- Primero";
```

Al documento **le pido el elemento con ID “prim”**

DOM - Edición

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo Arbol</title>
    <link rel="stylesheet" href="estilo.css">
  </head>
  <body>
    <h1>Titulo 1</h1>
    <div>Div1
      <p>Parrafo dentro de div</p>
      <h2>Titulo 2 en div</h2>
    </div>
    <h2>Titulo 2</h2>
    <ul>
      <li>Elemento 1 <a href="...">Link1</a></li>
      <li>Elemento 2
        <ol>
          <li id="prim">Primero</li>
          <li id="seg">Segundo</li>
        </ol>
      </li>
    </ul>
  </body>
</html>
```

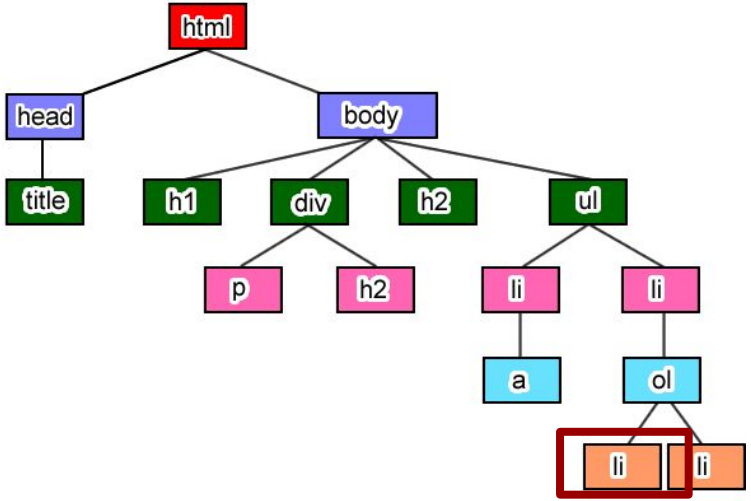



```
document.querySelector('#prim').innerHTML = "1- Primero";
```

Al documento le pido el elemento con ID “prim”, **y al contenido**

DOM - Edición

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo Arbol</title>
    <link rel="stylesheet" href="estilo.css">
  </head>
  <body>
    <h1>Titulo 1</h1>
    <div>Div1
      <p>Parrafo dentro de div</p>
      <h2>Titulo 2 en div</h2>
    </div>
    <h2>Titulo 2</h2>
    <ul>
      <li>Elemento 1 <a href="...">Link1</a></li>
      <li>Elemento 2
        <ol>
          <li id="prim">1- Primero</li>
          <li id="seg">Segundo</li>
        </ol>
      </li>
    </ul>
  </body>
</html>
```




```
document.querySelector('#prim').innerHTML = "1- Primero";
```

Al documento le pido el elemento con ID “prim”, y al contenido **le asigno un nuevo valor**

Leer/Editar el DOM

Se pueden manipular cualquiera de los atributos de un elemento:

Editar el **contenido**:

```
let unDiv = document.querySelector("unDiv");
unDiv.innerHTML = "Contenido Nuevo";
```

Editar un **atributo**:

```
let lampImg = document.querySelector("lamp");
lampImg.src = "foto.png";
```

Leer el **valor** de un input:

```
let inputNombre = document.querySelector("#inputNombre");
console.log(inputNombre.value);
```

Resultado

```
<input type="text" id="txtNombre" oninput="actualizarSaludo()" />
<p id="txtSaludo">ACA VA EL SALUDO</p>
```

EVENTO INPUT

```
function actualizarSaludo() {
    //lee el nombre
    let nodoInput = document.querySelector("#txtNombre");
    let nombre = nodoInput.value;

    //lo muestra en consola (opcional, para debug)
    console.log(nombre);

    //lo muestra en el DOM
    let nodoSaludo = document.querySelector("#txtSaludo");
    nodoSaludo.innerHTML = "Hola " + nombre;
}
```

PIDE NODO

LEE VALOR

PIDE NODO

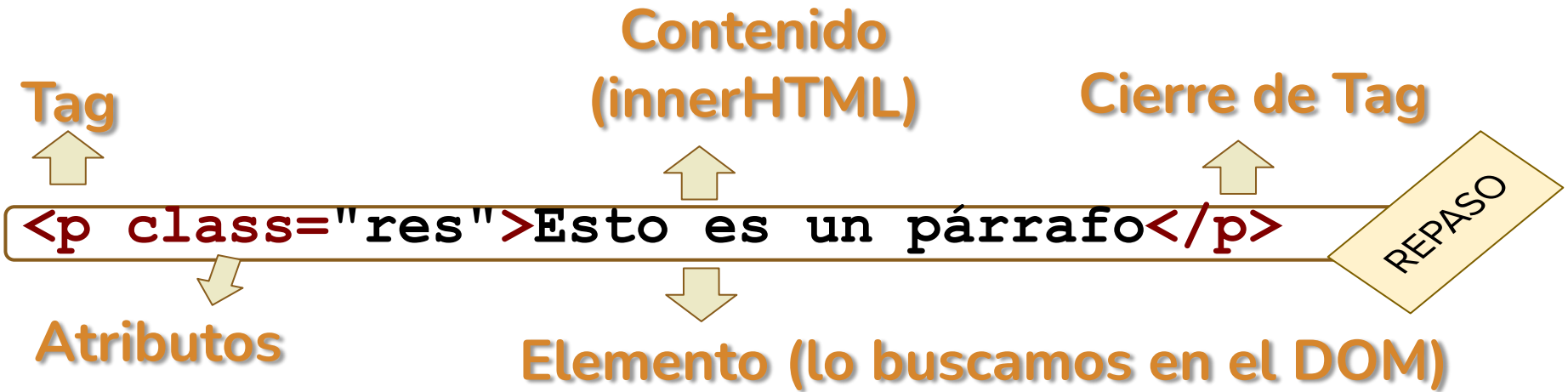
ESCRIBE VALOR

https://codepen.io/webUnicen/pen/geRRLe

Resumen DOM

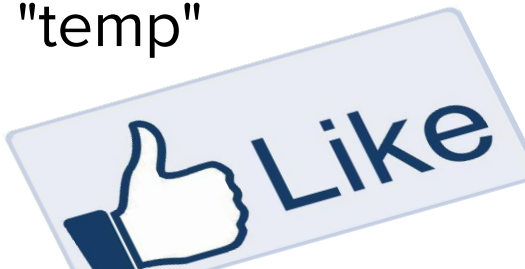
Las propiedades del DOM (HTML) se pueden leer/editar desde Javascript.

```
let unDiv = document.getElementById("unDiv");
unDiv.innerHTML = "Contenido Nuevo";
```



Naming - Buenas Prácticas

- Variables:
  - se nombran con sustantivos
- Funciones:
  - Comienzan con verbos
  - En el caso de funciones booleanas, deben comenzar con "is", ej: isValid()
- **Usar nombres descriptivos**
  - nunca son demasiado largos!
- Evitar nombres sin significado como "aux" y "temp"



Debug

Usando la consola en JS

Debug

Debug es el término conocida para encontrar los errores de un programa y solucionarlos.

DEBUG

DE(lete) **BUG**

“Eliminar Bichos”

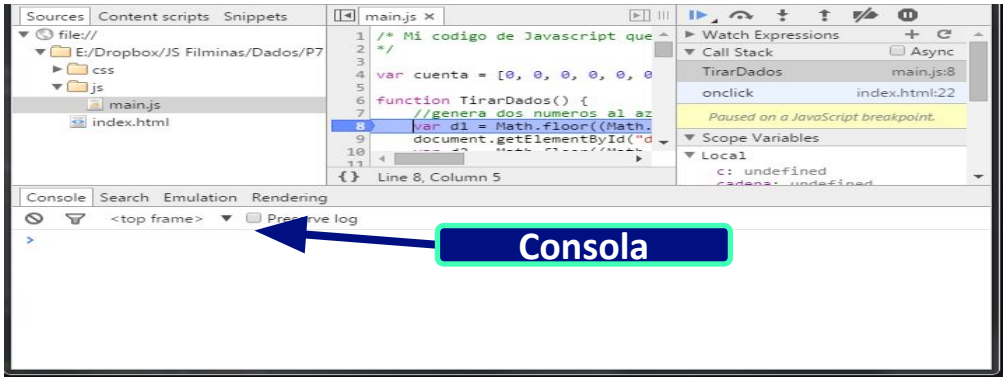
También se dice “debug” al proceso de hacer un seguimiento del código en la máquina y comprender mejor cómo funciona.

Consola

La **consola** Javascript es una zona del navegador ubicada en las DevTools donde podemos escribir pequeños fragmentos de código y observar los resultados, así como revisar mensajes de información, error u otros detalles similares.

```
console.log("HOLA MUNDO");
```

imprime algo en la consola del debugger.



Atajo de teclado:

- CTRL+SHIFT+I
- F12

Resultado

```
console.log("Paso 1: declarando funciones");
function actualizarSaludo() {
    //lee el nombre
    let nodoInput = document.getElementById("txtNombre");
    let nombre = nodoInput.value;

    //lo muestra en consola (opcional, para debug)
    console.log(nombre);

    //lo muestra en el DOM
    let nodoSaludo = document.getElementById("txtSaludo");
    console.log("Paso 3: Saludo:" + nodoSaludo.innerHTML);
    nodoSaludo.innerHTML = "Hola " + nombre;
    console.log("Paso 4: Saludo:" + nodoSaludo.innerHTML);
}
console.log("Paso 2: continua ejecución");
```

https://codepen.io/webUnicen/pen/ZZzgxv

Consola: Otros usos

Aunque el `console.log()` es uno de los métodos más utilizados (a veces el único), existen muchos otros:

`console.debug()`	Muestra información con todo nivel de detalle.
`console.info()`	Muestra mensajes de información.
`console.warn()`	Muestra información de advertencia. Aparece destacado en amarillo.
`console.error()`	Muestra información de error. Aparece destacado en rojo.
`console.trace()`	Muestra la traza del mensaje
`console.table()`	Muestra una tabla con los datos. Ideal para arreglos de objetos

SIEMPRE PROBÁ EL CÓDIGO JAVASCRIPT CON LA CONSOLA ABIERTA

Prende y Apaga

Otro ejemplo más :)

JS - Ejemplo 4

Ejemplo 4: Prende y Apaga

Crear dos botones para “prender” y “apagar” la página.

PRENDER: fondo blanco + imagen lámpara prendida

APAGAR: fondo oscuro + imagen lámpara apagada

¿Qué vamos a aprender?

- Cambiar el estilo de un elemento desde Javascript
- Agregar event listeners (manera recomendada)

Eventos

Repaso

Formas de manejar eventos

Existen varias formas diferentes de manejar eventos en Javascript:

Mediante **atributos HTML**

```
<button onclick="..."></button>
```

Mediante **propiedades Javascript**

```
onclick = function() { ... }
```

Mediante **listeners del DOM**

```
.addEventListener("click", ...)
```

Asignar handlers en HTML es mala práctica porque mezcla HTML y JavaScript y es menos flexible

RECOMENDADA!

Eventos

MÉTODO `addEventListener(event, handler);`

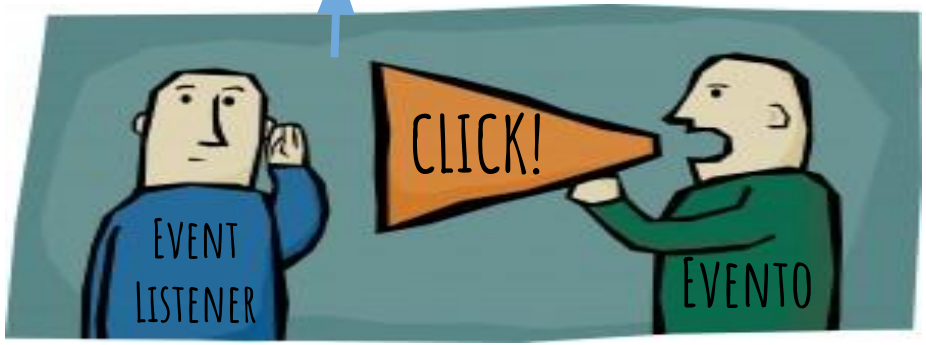
Con el método `.addEventListener()` permite añadir una escucha del evento indicado (primer parámetro), y en el caso de que ocurra, se ejecutará la función asociada indicada (segundo parámetro).

HTML

```html
<button type="button" id="btn-saludar">Enviar</button>
```

JS

```js
let btn = document.getElementById("btn-saludar");
btn.addEventListener("click", saludar);
```



La función handler va sin "()"

No la está llamando en esta sentencia, solo le dice que función llamar después

Comparativa

```
<button type="button" onclick="saludar();">
    Enviar
</button>
```

Es lo mismo, pero en lugar de escribir Javascript en el HTML, **desde JS** pedimos el nodo del DOM y le agregamos el handler del evento.

¡Cada cosa en su lugar!

```
<button type="button" id="btn-enviar">Enviar</button>
```

```
let btn = document.getElementById("btn-saludar");
btn.addEventListener("click", saludar);
```

Más Info

https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener#Why_use_addEventListener

Why use `addEventListener()`?

`addEventListener()` is the way to register an event listener as specified in W3C DOM. The benefits are as follows:

- It allows adding more than a single handler for an event. This is particularly useful for AJAX libraries, JavaScript modules, or any other kind of code that needs to work well with other libraries/extensions.
- It gives you finer-grained control of the phase when the listener is activated (capturing vs. bubbling).
- It works on any DOM element, not just HTML elements.

The alternative, older way to register event listeners, is described below.

Editar estilo desde Javascript

.classList

Se trata de un objeto especial que nos facilita trabajar con las clases de los elementos de una forma más intuitiva y lógica.

```
let div = document.querySelector("#divDerecho");

// agrega una clase
div.classList.add("destacado");
// elimina una clase
div.classList.remove("destacado");
// alterna una clase
div.classList.toggle("destacado");

// pregunta si existe una clase
if (div.classList.contains("clase"))

// agrega múltiples clases
div.classList.add("clase-1", "clase-2", "clase-3");

// estilos vía JS (Mala práctica)
div.style.font-size = "20px";
```

Buena Práctica!

Cambiar estilos con clases

DEMO

https://codepen.io/webUnicen/pen/qmVoMV

Resultado

```
"use strict";
function prender() {
 document.querySelector("body").classList.add("prendido");
 document.querySelector("body").classList.remove("apagado");
 document.querySelector("#lampara").src = "img/lampara_on.png";
}
function apagar() {
 document.querySelector("body").classList.remove("prendido");
 document.querySelector("body").classList.add("apagado");
 document.querySelector("#lampara").src = "img/lampara_off.png";
}
document.querySelector("#btnPrender")
     .addEventListener("click", prender);
document.querySelector("#btnApagar")
     .addEventListener("click", apagar);
```

CAMBIA CLASES Y SRC DE LA IMAGEN

https://codepen.io/webUnicen/pen/eoYOZP

Extra

Funciones con parámetros

En Javascript también podemos pasarle parámetros a las funciones:

```
function saludar(nombre) {
    alert("Hola" + nombre);
}
```

parámetro formal

Al llamarlo debo decirle el valor que tiene “nombre”

```
saludar("Javier1");
saludar("Javier2");
```

parámetro real
(“lo que le paso como parametro”)

Eventos y parámetros

En un event listener no podemos pasar parámetros a la función handler. El único parámetro es uno que el navegador le pasa a la función con datos del evento (e).

```
let inputEmail, inputConsulta...
...
btn.addEventListener("click", sumarCinco)

function sumarCinco(e){
   sumar(5);
}

function sumar(cantidad) {
  ...
}
```

Más adelante vamos a aprender a usar funciones anónimas en los listeners para pasaje de parámetros

https://codepen.io/webUnicen/pen/qmZOaV

Prendamos y apaguemos con el teclado

Modifiquen el ejercicio anterior para “prender” y “apagar” usando las teclas del teclado

- Detectar el evento “key-press”. ¿A quien se lo asigno?
  - Opción 1: Asignar una tecla para prender y otra para apagar
  - Opción 2: Asignar la misma tecla para prender para apagar

Ejemplo de la práctica

Juego de Dados

Ejemplo 5

JS - Ejemplo 5

Ejemplo 5: Tirar dados

Hacer una aplicación web que al apretar un botón simule el lanzamiento de dos dados, sumarlos y que muestre en en la página si salió un 8.

¿Qué vamos a aprender?

1. Generar un número aleatorio utilizando funciones matemáticas de JS
2. Uso de estructuras de control (if)
3. Uso de comparaciones

¿Qué tenemos que hacer?

Lo primero que tenemos que hacer es analizar el problema y separarlo en partes pequeñas que sean más fáciles de resolver

Analicemos y separemos el problema en pasos

1. Crear el html con la estructura básica para jugar
    a. dos imágenes para los dados
    b. un botón para lanzar
2. Una función para “lanzar” que genere dos número aleatorios
3. Verificar el resultado
4. Informar por pantalla si salió el número 8

Random

`Math.random();`

Es una función de Javascript para generar números al azar entre 0 y 1 (el 1 no se incluye: de 0 a 0.9999999999...)

Las caras de un dado son del 1 al 6 ¿cómo hacemos?

Podemos utilizar también la función matemática `Math.floor()`, que redondea al menor entero, y aplicar algunas operaciones matemáticas para convertir el número aleatorio.

```
Math.floor((Math.random() * 6) + 1);
// 0 <= Math.random() * 6 <= 5.999
```

Math es un objeto incorporado en JS que tiene propiedades y métodos para constantes y funciones matemáticas.

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Math

Random

Repasemos:

1. Arrancamos del número random (aleatorio)

```
0 <=  Math.random() <= 0.9999999
```

2. Multiplicamos por 6 (el número que queremos como límite)

```
0 <=  Math.random() * 6 <= 5.9999999
```

3. Sumamos 1

```
1 <=  (Math.random() * 6) + 1 <= 6.9999999
```

4. Nos quedamos con la parte entera

```
Math.floor((Math.random() * 6) + 1) ∈
{1,2,3,4,5,6}
```

http://codepen.io/webUnicen/pen/RaMpoV

Mostrar imágenes de los dados

Debo mostrar la imagen del dado que corresponde al número que salió.

Nombre de mis imágenes

dado1.png

...

dado6.png

¿ Dónde están esas imágenes ?

http://tudai1-1.alumnos.exa.unicen.edu.ar/web-1/material/archivos/

http://tudai1-1.alumnos.exa.unicen.edu.ar/web-1/material/archivos/dado1.png

Mostrar imágenes de los dados

Para mostrar las imágenes podemos cambiar la propiedad “src” de las dos <img> que tenemos en el html.

Debo generar un **string** con el nombre de la imagen que se corresponda con el número del dado que salió.

```
document.getElementById("dado1").src = "images/dado"+d1+".png";
```

¿ Ruta relativa o externa ?

Si estoy en Codepen o no están las imágenes en mi carpeta images el enlace es externo, debo concatenar usando: `http://tudai1-1.alumnos.exa.unicen.edu.ar/web-1/material/archivos/`

https://codepen.io/webUnicen/pen/bvXWxo

Sintaxis JavaScript - if , else

Verificamos si la suma de 8 => Uso de condicionales

```
if(condicion) {
    // codigo SI se cumple
    // la condicion
}
else {
    // codigo si la condicion
    // NO se cumple
}
```

Operadores de comparación

- = Asignación

```
let materia = “Web”
```

- == Igualdad (!=)
  - Convierte tipos

```
if( 5 == “5”) {
//true
}
```

- === Identidad (!==)

```
if( 5 === “5”) {
//false
}
```

Siempre usar ===

- Recordemos que (==) hace conversión automática de tipos
- A veces el resultado de esta conversión no es obvia.
- Evitar esto usando (===)
- Lo mismo para != (usar !==)

Resultado

```
let suma = d1 + d2;

if (suma === 8){ //verificacion si es 8 e incremento cuenta
    document.getElementById("resultado").innerHTML
        = "salió el 8";
}
```

https://codepen.io/webUnicen/pen/yrLBdr

Resumen

Aprendimos a

- Generar un número aleatorio en JS
- Uso de estructuras condicionales
- Cómo comparar valores de variables

Extra: 1000 veces!

Bucle

Problema:

Queremos simular que los dados se tiraron mil veces al apretar el botón (pero quedarnos con el último resultado)

Solución: Bucles (Estructuras de control iterativas)

```
for (let i = 0; i < 1000; i++) {
    // código ejecución durante el bucle
}
```

Inicialización: el bucle se ejecuta desde un valor inicial (i = 0)
Condición: Se ejecuta siempre que la condición sea verdadera (i<1000)
Actualización: En cada pasada del bucle i se modifica, i++ es sumarle 1 al anterior

DEMO https://codepen.io/webUnicen/pen/jROOOd

Bibliografia y Más Información

Enlaces

- https://lenguajejs.com/javascript/

Libros

- Javascript and Jquery : Interactive Front-End Web Development, Jon Duckett Willey 2014
- Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics, Jennifer Niederst Robbins O'Reilly Media 2012
- Standard: http://standardjs.com/rules.html
- Tutorial W3 Schools: http://www.w3schools.com/js/
- Javascript from birth to closure

Eventos

- http://www.elcodigo.net/tutoriales/javascript/javascript5.html
- http://dev.opera.com/articles/view/handling-events-with-javascript-es

AHORA LES TOCA PRACTICAR :D