Javascript

Segunda parte

Contador de clicks

Resolver el problema

¿Qué vamos a aprender?

- Cómo recordar cosas
  - Vamos a usar una **variable** para contar

Recordar datos

Como hacen las páginas para recordar datos?

- “Memoria a corto plazo”, o sea, sin usar Bases de Datos
- Se pueden utilizar:
  - Variables
  - Estructura de datos
- **OJO:** Luego de un refresh perdemos los datos!

Usar DOM para guardar datos. Ej: usar un hidden como “variable” global

Solución

Vamos a declarar una variable donde llevemos la cuenta de los clicks

```
contador = 0
```

cada vez que el usuario hace click vamos a incrementar el valor del contador en 1

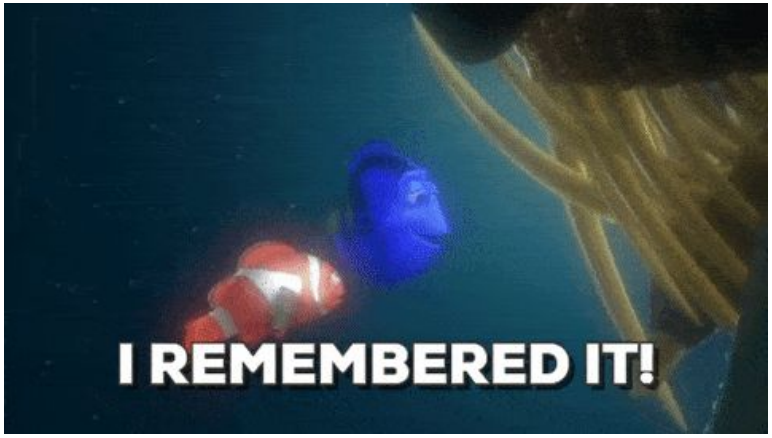
```
contador = contador + 1
```

que también se puede escribir como (abreviación para + 1)

```
contador++
```

Clicker!

```
<div class="container">
    <h1>Bienvenido a nuestro contador de click!</h1>
    <p>
      Hiciste <span id="spanContador"> 0 </span> clicks!
    </p>
    <button id="btn-click">Contar click!</button>
</div>
```



Clicker!


```
"use strict";
let btn = document.getElementById("btn-click");
btn.addEventListener("click", clickear);

let contador = 0;

function clickear() {
//incrementa el valor de contador
contador++;
//es lo mismo que contador = contador + 1
let valor = document.getElementById("spanContador");
valor.innerHTML = contador;
```

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

Variables Globales

Como la variable **contador** es global, se puede acceder y ver o modificar desde la consola del navegador!

```
console.log(contador);
contador = contador + 1000;
console.log(contador);
```

Esto no es bueno, incluso dos programas JS podrían usar el mismo nombre de variable y entonces la compartirán sin saberlo!

Evitar variables y funciones globales

- En lugar de declarar las variables y funciones como globales podemos incluirlas en un objeto para aislarlas.
- Crear un ámbito de todo el documento de JS para que las variables no estén disponibles desde fuera de él.
- Esa función la ejecutamos cuando el DOM se haya cargado

Otra forma de hacerlo

Podemos configurar todos los eventos una vez que ya se cargo el DOM. De este modo nos **aseguramos** que estén cargados todos los elementos del DOM antes de manipularlos

```
document.addEventListener("DOMContentLoaded",
  iniciarPagina);

function iniciarPagina(){
   // código de inicialización de eventos
}
```

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

Evitar variables y funciones globales

```
document.addEventListener('DOMContentLoaded', iniciarContador);

function iniciarContador() {
    "use strict";
    let btn = document.getElementById("btn-click");
    btn.addEventListener("click", clickear);

    let contador = 0;
    function clickear() {
        //incrementa el valor de contador
        contador++;
        //es lo mismo que contador = contador + 1
        let valor = document.getElementById("spanContador");
        valor.innerHTML = contador;
    }
}
```

leo variables de afuera, no de más adentro

"contador" no puede usarse desde más afuera


EVERY TIME YOU CREATE JAVASCRIPT GLOBAL VARIABLES
GOD KILLS A KITTEN
memegenerator.net

Resumen

Aprendimos a

- Usar variables globales para recordar cosas
- Limitar desde donde son accesibles esas variables
  - DOMContentLoaded

Contador Calorias

Contador de calorías

Quiero llevar contado la cantidad de calorías que consumo en el día

- En la página web se muestra la cantidad de calorías
- Hay un botón para incrementar la cantidad de calorías
- Hay un botón para decrementar la cantidad de calorías
- Hay una caja de texto para sumar muchas calorías en una sola acción

¿Qué vamos a aprender?

- Tipos de variables
  - Al escribir el texto, voy a tener un texto y lo quiero sumar como número
- Parámetros en las funciones
- Funciones anónimas

Comentarios

Se pueden hacer comentarios

```
/*  ACA VA EL COMENTARIO  */
```

¿Qué tanto comentar el código?

- Cada método o función (describiendo lo que hacen y los parámetros). Siempre y cuando el nombre y los parámetros no se auto-documenten.
- Código difícil de entender (switch-case, números mágicos, etc)
- Cosas que podrían parecer un error (usar = en lugar de == adrede, conviene aclararlo)

Tipado de Variables - Tipos

- El **tipado estático** nos obliga a definir desde el principio el tipo de una variable. Lenguajes con tipado estático son C++, Java, C# (casi) entre otros.

- El **tipado dinámico** nos da la facilidad de no definir los tipos al declarar una variable, algunos ejemplos son PHP, JavaScript, Grooby, Phyton, entre otros.

- ¿Se les ocurren pros y contras?

http://qbit.com.mx/blog/2012/01/16/tipado-estatico-vs-tipado-dinamico/

Tipos

- Javascript tiene tipos dinámicos.
- Una misma variable puede cambiar de tipo.
- Puede causar confusiones (y errores que no encuentro durante horas).

```
let nombre = "Pepe"; //nombre es un string

...

nombre = 2;  //nombre es un int (cambia tipo)
```

Mala práctica

Tipos de Datos

- String
- Number
- Boolean
- Null
- Undefined
- Object
  - Function
  - Array
  - Date
  - Expresiones Regulares (RegExp)

Undefined

- ***undefined*** es un tipo fundamental en Javascript
- Las variables sin inicializar valen ***undefined***
- Variables y miembros sin declarar valen ***undefined*** (salvo que uses “use strict” que causa una falla)
- Las funciones siempre devuelven un valor, si no tienen valor de retorno devuelven ***undefined***

0

null

undefined

Conversión de tipos

- **Cuidado** con los **tipos**, son dinámicos y no saber de qué tipo es una variable puede cambiar el resultado.

```
5 == "5"//true

"1" + 2 + 3;//"123"

//Conversion manual de tipos
parseInt("1", 10) + 2 + 3; //6
```

http://www.w3schools.com/jsref/jsref_parseint.asp

- **ES6** introduce una nueva forma de trabajar con Strings

```
'<p>Vos sos '+nombre+' '+apellido+'.</p>'

`<p>Vos sos ${nombre} ${apellido}.</p>`
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

Parámetros

Las funciones pueden tener parámetros y devolver valores.

```
function sumar(parametro1, parametro2) {
   return parametro1 + parametro2;
}
```

Al llamarla con:

```
let valor = 2;
let resultado = sumar(3,valor); //devuelve 5
```

El primer parámetro va a valer 3, el segundo va a valer 2, por lo que la suma dará 5.

Esto hace el código más **genérico y reutilizable**.

Los parámetros son la **entrada** del código, y el valor que devuelve es la **salida**.

Eventos en ES6 - Funciones anónimas

- Se usan para no crear tantas funciones que se usan en un solo lugar
- Es una función sin nombre que se escribe directamente donde la quería pasar de parámetro
- En este caso encapsula a la función que si pasa parámetros

Una nueva función anónima que llama a verificarFormulario con sus parámetros, es igual al **verificarSinParametros**

```
btn.addEventListener("click", function(e){
 verificarFormulario(inputEmail,inputConsulta)
} )
function verificarFormulario(email, consulta)
{...
```

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

Resolver el problema

Asignar eventos a los 3 botones

```
<button id="btn-restar">--</button>
<button id="btn-sumar">++</button>
<button id="btn-sumar-input">Sumar</button>
```

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

JS + Función anónima:

```
let btnSumar = document.getElementById("btn-sumar");
btnSumar.addEventListener("click", function () {SumarCantidad(+1) });

let btnRestar = document.getElementById("btn-restar");
btnRestar.addEventListener("click", function () { SumarCantidad(-1)
});

let btn = document.getElementById("btn-sumar-input");
btn.addEventListener("click", SumarInput);
```

Resumen

Aprendimos

- Comentarios
- Tipos
- Conversiones de tipos
- Funciones anónimas

Carrito de compras

Ejemplo

Un cliente quiere realizar compras y saber total que va a gastar entre todos los elementos.

Crear un formulario para cargar los datos de los Items y utilizando los datos cargados por el cliente sumar el total gastado.

Qué vamos a aprender?

- Qué son los arreglos
- Recorrer un arreglo
- Agregar elementos a un arreglo
- Vaciar un arreglo

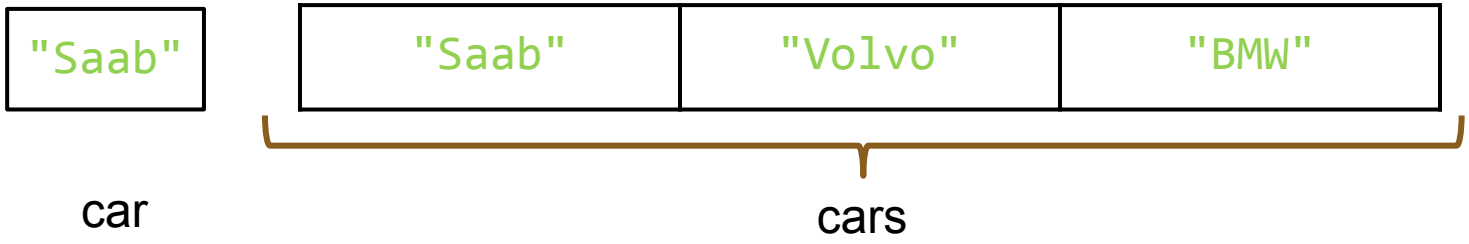
Arreglos

Los arreglos almacenan elementos en una colección de datos ordenados.
Los elementos se acceden por índice (comienzan en 0).

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

Acceder a un arreglo:

```
let car = cars[0];
console.log(car); //”Saab”
```



Arreglos

```
let arr = [1,2,3]
// definición
arr[0]
// 1
arr[5]
// undefined
arr[5] = “texto”
// [1,2,3,undefined, undefined, “texto”]
delete arr[2]
// [1,2,undefined,undefined, undefined, “texto”]
```

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

Recorriendo Arreglos

```
let arr = [1,2,3]
// Antes
for (let i=0; i<arr.length; i++) { // 1, 2, 3
    let elem = arr[i];
    console.log(elem);
}

// foreach de ES6
for (let elem of arr) { // 1, 2, 3
    console.log(elem);
}
```

Agregando elementos y borrando

```
let frutas = ["Manzana", "Pera", "Naranja"];

//Agregar elemento
frutas.push("Mandarina");

//Borra Ultimo
frutas.pop()
//Borra el último con un método más genérico y los
parámetros adecuados
frutas.splice(frutas.length-1);

//Borra TODOS del primero al último
frutas.splice(0,frutas.length);
//o equivalente, le vuelvo a asignar el arreglo vacío
frutas = [];
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice

Resolviendo el problema

Creo dos arreglos uno de items y otro de precios, el contenido de cada posición tiene correspondencia item - precio

items	precios
Computadora	8900
Banana	25
Frutilla	60
Silla	600
Pera	25
Casa	1000000

Resolviendo el problema

Creo 4 botones (Agregar, Sumar, Reset, Borrar Último), 2 inputs para agregar item de compra y precio

Creo funciones agregar(), sumar(), reset(), borrar() y mostrar()
Llamo a mostrar cada vez que se modifica el arreglo.

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

Se usan arreglos para:

[TBC]

Resumen

Aprendimos

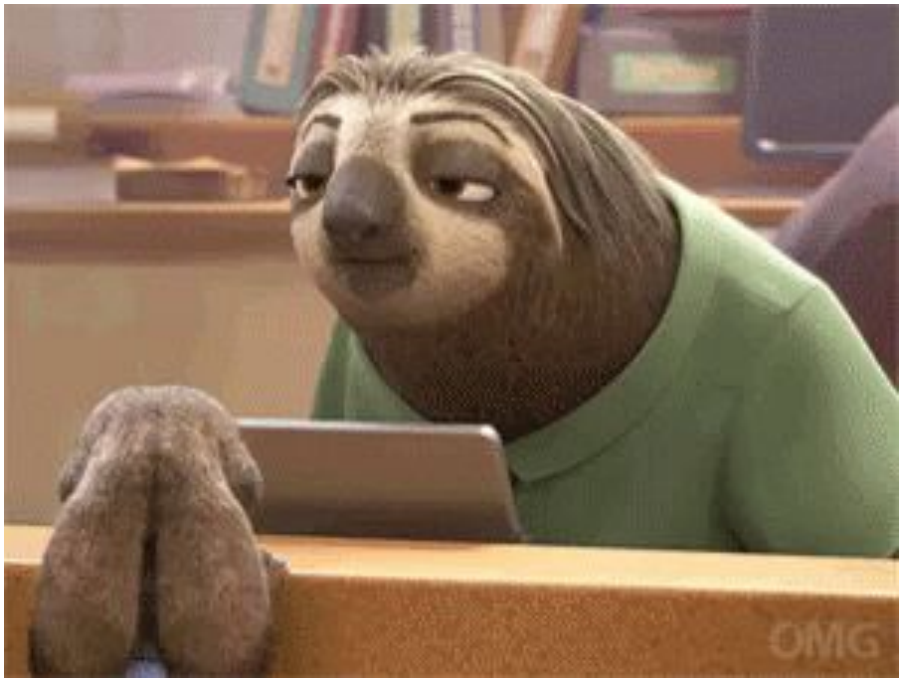
- Arreglos
- Manipulaciones sencillas sobre arreglos

Javascript - Buenas prácticas

Usar Cache de Selectores

Las consultas al DOM son MUY lentas.

Editarlo también.



Usar Cache de Selectores

- Salvar los resultados de selectores optimizan mucho el código.

```
function sinCache () {
    let i, val;
    for(i=0; i<50; i++){
      val =
document.getElementById('title
').innerHTML
    }
 }
```

```
function conCache (){
    let i, val;
    let h1 =
document.getElementById('title'
);
    for(i=0; i<50; i++){
      val = h1.innerHTML;
    }
 }
```

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

Separar los "event handlers"

```
// Mal hecho
function handleClick(event){
    let popup =
document.getElementById("popup");
    popup.style.left = event.clientX +
"px";
    popup.style.top = event.clientY + "px";
    popup.className = "reveal";
}
// Mejor, pero sigue estando mal
function handleClick(event){
    showPopup(event);
}
function showPopup(event){
    let popup =
document.getElementById("popup");
    popup.style.left = event.clientX +
"px";
    popup.style.top = event.clientY + "px";
    popup.className = "reveal";
}
```

```
// La solucion correcta
function handleClick(event){
    showPopup(event.clientX,
event.clientY);
}
function showPopup(x, y){
    let popup =
document.getElementById("popup");
    popup.style.left = x + "px";
    popup.style.top = y + "px";
    popup.className = "reveal";
}
```

Acoplamiento HTML,CSS,JS

Buena práctica

- Mantener cada uno en su tarea.
- No escribir código HTML o CSS desde JS.
  - Usar los métodos específicos para editar el DOM.

Mantener a JS fuera del HTML

- Lenguajes acoplados:

```
let element = document.getElementById("container");
element.innerHTML = "<div class=\"popup\"></div>";
```

- Solución:
  - Create Element

Nota: crear html con texto se suele usar por ser más eficiente para cosas grandes. Hay que analizar el trade-off (que conviene en cada caso).

Nombres para las clases

- Es fácil al hacer un cambio en el HTML romper un estilo o un script si están muy acoplados.
- Se recomienda:
  - Usar selectores simples (tag>tag>tag es facil de romper en el HTML, mejor usar una clase) tanto en CSS como JS

Más info: http://philipwalton.com/articles/decoupling-html-css-and-javascript/

Nombres para las clases

Se recomienda:

- js-class para clases que se usen para asignar comportamiento JS.
- is-class para clases que cambian el estilo visual (ej, is-visible)

Más info: http://philipwalton.com/articles/decoupling-html-css-and-javascript/

Repasemos

- Las convenciones de código facilitan la comunicación.
- Bajar el acoplamiento entre capas (lenguajes) facilita los cambios y el debug.
- Las buenas prácticas de programación permiten...
  - [TBC]

Reloj

Reloj - Bomba

Simular la cuenta regresiva de una bomba.

Con un botón activarla y dejar 5 segundos para escapar y comenzar la cuenta regresiva.

El valor de la cuenta regresiva se ingresa por un input

Qué vamos a aprender

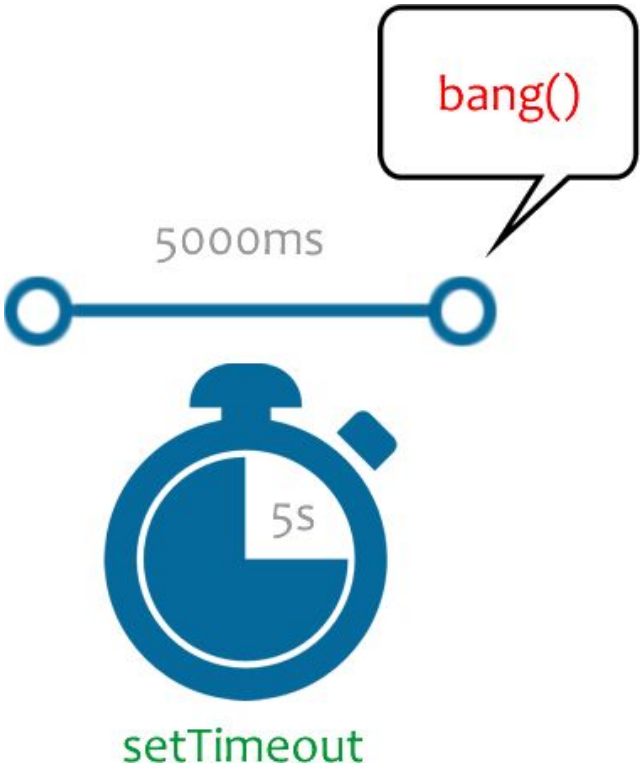
Qué vamos a aprender?

- Ejecutar eventos diferidos en tiempo, o retardados
- Ejecutar eventos que se repiten en intervalos de tiempo hasta que hagamos un reset.

Eventos de tiempo

Se puede programar un evento, para ejecutar una función dentro de M milisegundos.

```
//dispara (ejecuta bang) en 5 segundos
let timer = setTimeout(bang, 5000);
```

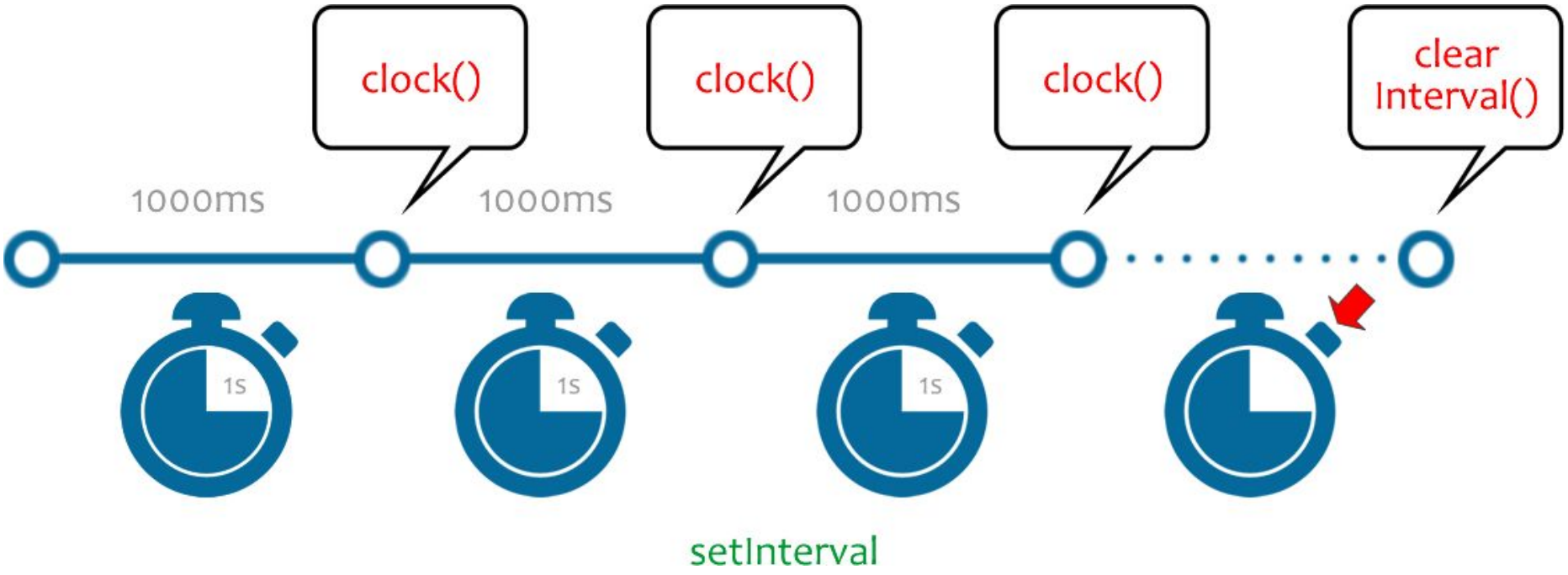


http://codepen.io/webUnicen/pen/jqzBpJ

Eventos de tiempo

```
let timer = setInterval(clock, 1000);
...
clearInterval(timer);
```

setInterval llama a la función cada 1000 milisegundos, hasta que se limpie el intervalo.



Resultado

Cuenta Regresiva

```
function cuentaRegre(){
    let intervalo = setInterval(function() {
        if (i === 0) {
            clearInterval(intervalo); // limpio intervalo
            alert('BOOOOOM!!');
        }
        else {
            i--;
        }
    }, 1000);
}
```

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

Implicancias de eventos de tiempo

[TBC]

Resumen

Aprendimos

- Usar temporizadores e intervals

Basta de ejemplos

Tengo el poder de Javascript!

JS y sus detalles

Obtener nodos del DOM

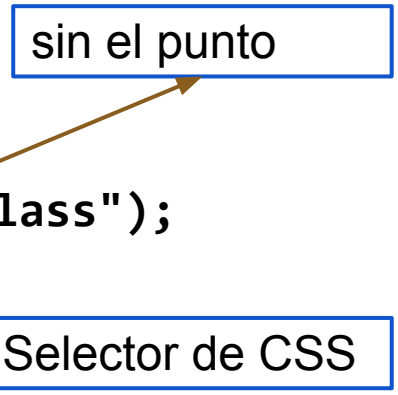
- Se pueden obtener elementos del DOM consultando por un ID, nombre, clase o un selector.
- Podemos obtener como resultado de uno o múltiples elementos del DOM

Retorna un nodo

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

Retorna uno o más

```
let manyElements = document.getElementsByClassName("myclass");
let manyElems = document.querySelectorAll(".myclass");
```



Más info https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector

Comparaciones cortas en JS

?

- If más corto:

```
let tmp = (bool) ? 1 : 2
```

es igual que

```
let tmp = 0;
if(bool)
   tmp=1;
else
   tmp=2;
```

- Solo sirve cuando es un valor que se asigna en la misma variable en las dos ramas.

Falsey evaluation

En Javascript, hay diferentes cosas que al convertirla a `bool`, se transforma a `false` automáticamente.

```
null == undefined == 0 == false == ""
```

No es tan así, pero es una buena simplificación.

```
let a = null; let b; //undefined

let c = 0; let d = false;

let e = "";

if (a), if (b), if (c), if (d), if (e) //false
```

Puedo pasarme horas revisando un bug en una comparación, que era por un `undefined` en la variable.

Falsey Evaluation

- Por costumbres de otros lenguajes, es normal escribir condicionales tipo C
- Los condicionales JS son más cortos y eficientes

```
1   /* C-style conditional */
2   if (val != null && val.length > 0){
3      ...
4   }
5
6   /* JavaScript style conditional */
7   if (val) {
8      ...
9   }
10
```

Largo de una cadena

Existen muchas funciones que ya trae Javascript

Para calcular el largo de una cadena puedo usar:

```
let  largo = str.length("cadena");
```

El valor calculado se **devuelve** y debe guardarse en una variable

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/length

Ejercicios

Ejercicios

Ejercicio #1

- Utilizando lo visto en esta clase, crear una función Javascript que oculte y muestre un div que contiene información.
- Analizar cómo modificar el ejercicio para que sea un código reutilizable (poder poner muchos botones que oculten o muestren un div respectivo)

Ejercicio #2

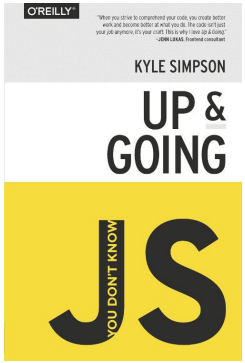
- Tengo una lista de tareas, y quiero dinámicamente (sin refrescar la página) agregar tareas.

AHORA LES TOCA PRACTICAR :D

Más Información

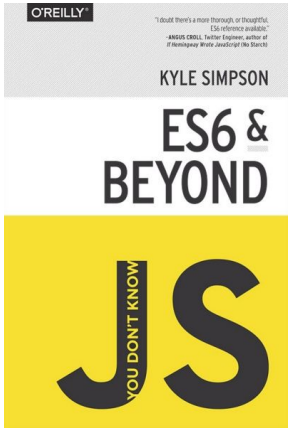
Libros

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



O'Reilly “You don’t know JS, up going”
https://github.com/getify/You-Dont-Know-JS/blob/master/up%20&%20going/README.md#you-dont-know-js-up--going

O'Reilly “You don’t know JS, ES6 and beyond”
https://github.com/getify/You-Dont-Know-JS/tree/master/es6%20%26%20beyond



Eventos

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