Javascript

CFL Programador full-stack

Parte II

Mostrar/Ocultar detalles

CFL
Programador
full-stack

Ejemplo

Crear un botón “Ver Más”, que muestre/oculte el contenido de un div.

El botón debe poder reutilizarse y funcionar de manera independiente del resto de los botones de la página.

Qué vamos a aprender

Qué vamos a aprender?

- Desacoplar HTML y JS
  - funciones anónimas
- Obtener múltiples elementos del DOM
- Recorrer el DOM
- Asignar clase a un elemento del DOM

Eventos en ES6

Asignar handlers en HTML es mala práctica porque mezcla HTML y JavaScript.

```
<button type="button" class="btn" onclick="verificarFormulario();">Enviar</button>
```

No recomendado

Debería recordar las funciones escritas en JS.

Eventos en ES6

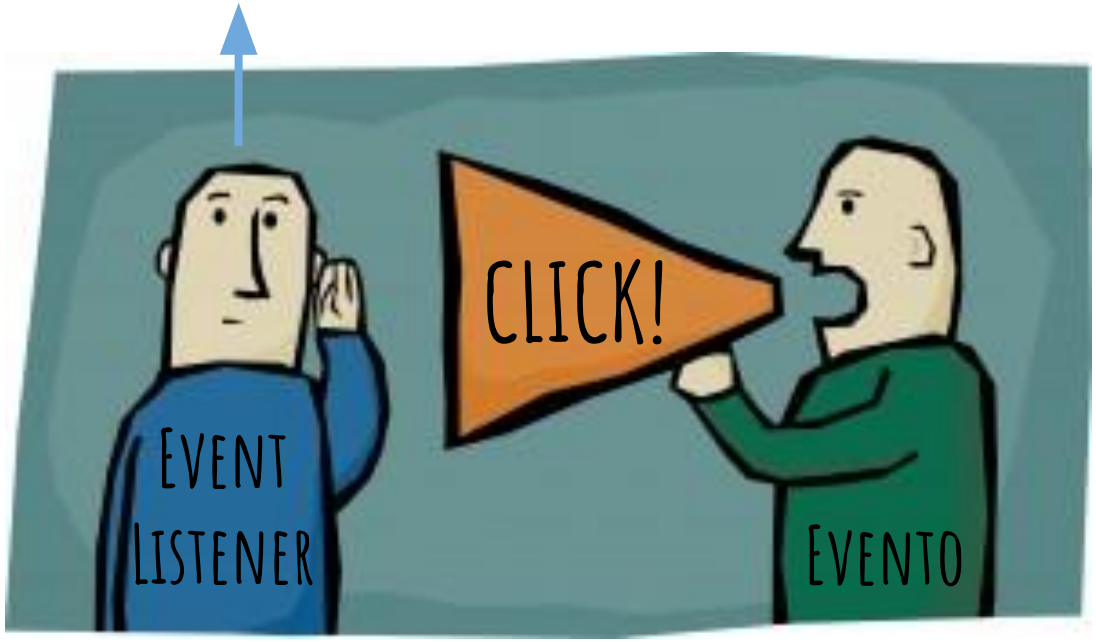
En **ES6**, primero buscamos el elemento y luego le asociamos el handler

HTML

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

JS

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



Está pendiente de escuchar un evento para ejecutarlo solo en el momento en que suceda

Eventos en ES6 con parámetros

¿Qué pasa si la función que queremos ejecutar respondiendo al evento requiere de parámetros?

Debemos pasar los parametros pero luego que el evento suceda.

Entonces necesitamos encapsular la función con otra y pasar los parámetros

Eventos en ES6

Utilizamos una **función anónima** sin parámetros que encapsula a la función que si pasa parámetros

JS

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

function verificarSinParametros(e){
 verificarFormulario(inputEmail,inputConsulta)
}
function verificarFormulario(email, consulta)
{
    ...
```

El parámetro **"e"** lo pasa JS, tiene info del evento (el click en este caso)

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

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)
{...
```

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

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

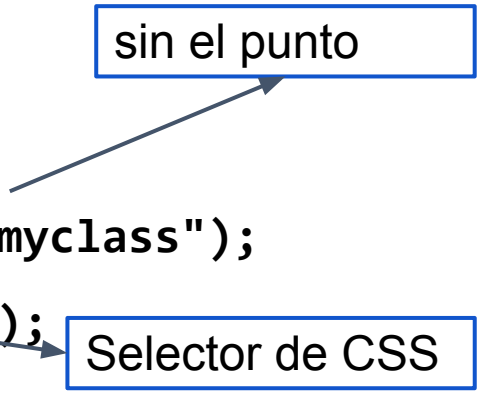
```
let elem = document.getElementById("identificador");

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

Obtener múltiples nodos del DOM

Obteniendo elementos del DOM con la misma clase

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

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

`manyElements` es un **arreglo** con los elementos que poseen la clase

`manyElements.length` largo del arreglo y cantidad de nodos con esa clase

`manyElements[0]` es el primer elemento con clase .myclass

Recorrer el árbol DOM

Los elementos del DOM se pueden recorrer como un árbol y ser localizados:

- element.children, encuentra los elementos hijos
- element.parentElement , encuentra el elemento padre
- element.nextElementSibling , encuentra el siguiente hermano
- element.previousElementSibling , encuentra el hermano anterior
- element.firstElementChild , encuentra el primer hijo
- element.lastElementChild , el último hijo

Editar estilo desde Javascript

```
// div es una referencia a un elemento <div>
div.classList.add("clase");
div.classList.remove("clase");
div.classList.toggle("clase");

alert(div.classList.contains("clase"));

// agregar o quitar múltiples clases
div.classList.add("clase-1", "clase-2", "clase-3");
```

Buena Práctica!
Cambiar estilos con clases

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

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

this

En el contexto de Eventos *this* representa el elemento involucrado en el evento

```
let el =
document.getElementById('miDiv');
el.addEventListener('click', function(e){
  this.classList.toggle("clase");
  //toggle de clase del div miDiv click
  console.log(e); //ver E
});
```

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

Resolver el problema

Debemos localizar todos los elementos que correspondan a una clase, y luego asignarle a cada uno el evento.

// Búsqueda de todos los botones con una clase

```
let btns = document.querySelectorAll('.btn');


// asignación de evento a todos los elementos
for(let i = 0; i < btns.length; i++) {
    btns[i].addEventListener('click', miFuncion);
}
```

Resolver el problema

Luego, mediante una función anónima individualizamos el botón que dispara el evento y buscamos su hermano en el DOM.

```
for(let i = 0; i < btns.length; i++) {
    btns[i].addEventListener('click', function(e){
    //busca el hermano inmediato

    let el = this.nextElementSibling;
    //toggle de clase del hermano

    el.classList.toggle("ver");
  });
}
```

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

Resumen: conexión entre HTML, CSS y JS

- La unica conexion debieran ser las clases
- Las clases son el contrato entre los tres lenguajes
- En lo único que se tienen que poner de acuerdo es en qué significa cada clase
  - HTML le va a poner las clases a lo que corresponda
  - CSS va a hacer que se vea como dice el acuerdo
  - JS va a hacer que se comporte como dice el acuerdo
- El nombre de la clase debe ser representativo de este contrato

Reloj

CFL Programador full-stack

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

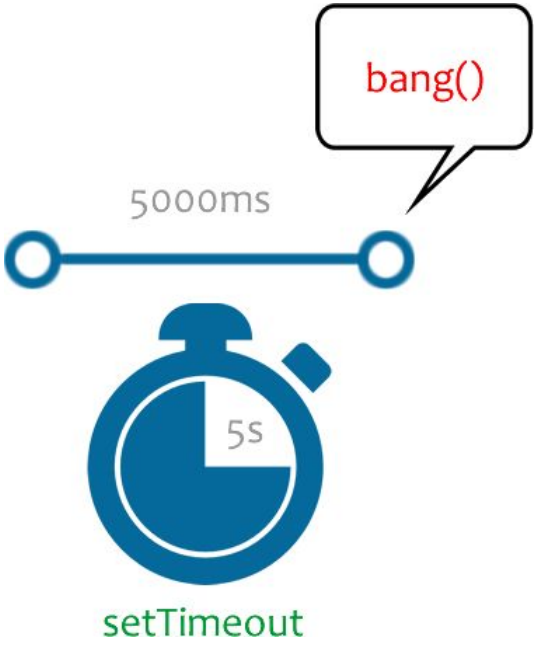
Qué vamos a aprender?

- 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.

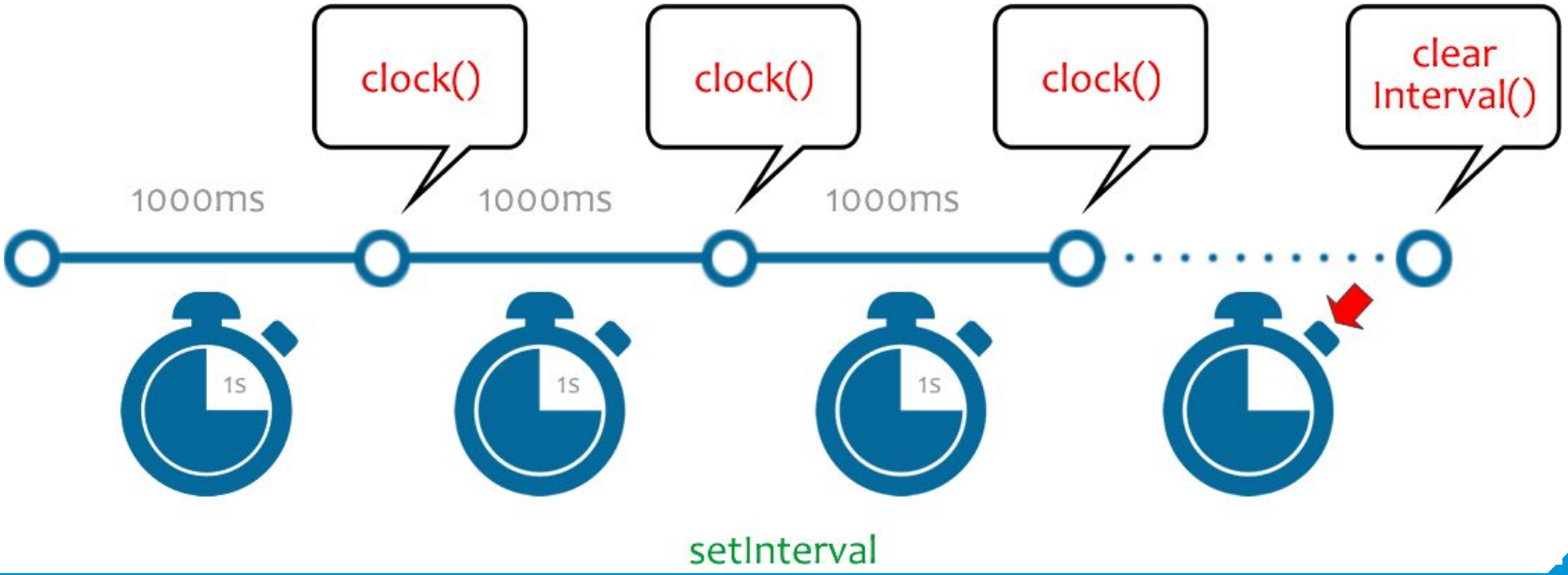
```
//dispara (ejecuta bang) en 5 segundos
let timer = setTimeout(bang, 5000);
```



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

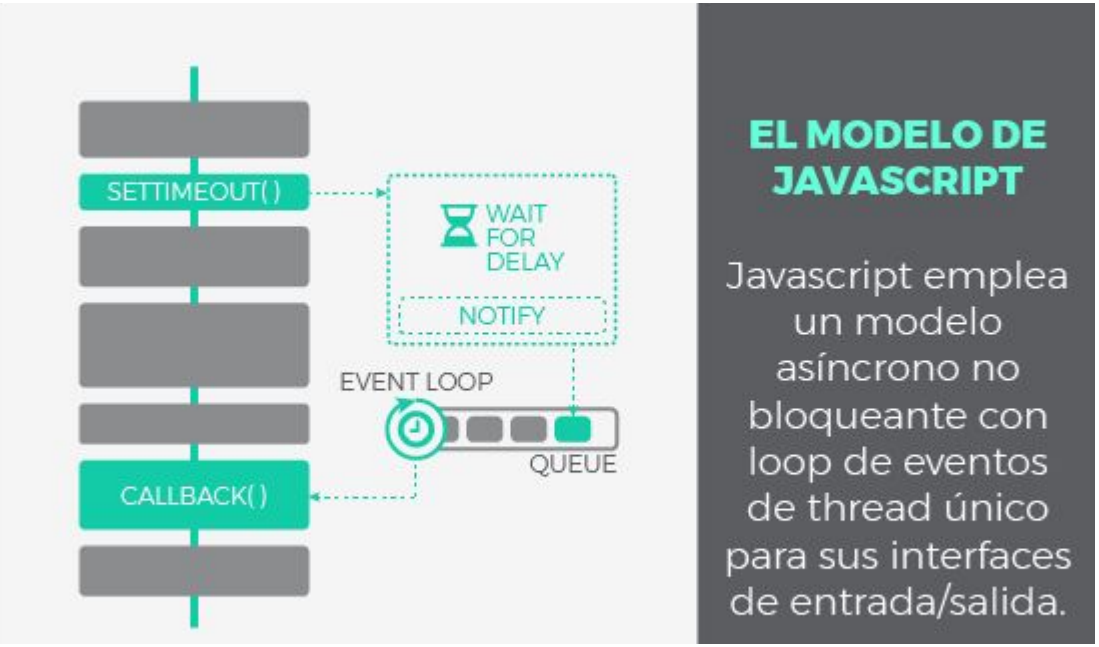
Cuenta Regresiva

```
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

Asíncrono: La finalización de la operación es notificada al programa principal. El procesado de la respuesta se hará en algún momento futuro.



Qué es un callback?

Un callback no es más que una función que se pasa como argumento de otra función, y que será invocada para completar algún tipo de acción.

- Son la pieza clave para que Javascript pueda funcionar de forma asíncrona
- Los patrones asíncronos de JS, de una forma u otra, están basados en callbacks

```
setTimeout(function(){
  console.log("Hola Mundo con retraso!");
  }, 1000)
```

Ejercicios

CFL Programador full-stack

Ejercicio 1

Utilizando lo visto en esta clase, crear una función Javascript que oculte y muestre un div que contiene información.

Ejercicio 2

Analizar cómo modificar el ejercicio anterior para que sea un código reutilizable (poder poner muchos botones que oculten o muestren un div respectivo)

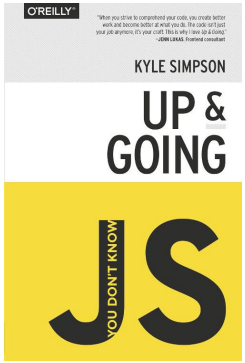
Ejercicio 3

Dada una lista de tareas, permitir que el usuario agregue dinámicamente nuevas tareas y mostrarlas sin necesidad de actualizar la página.

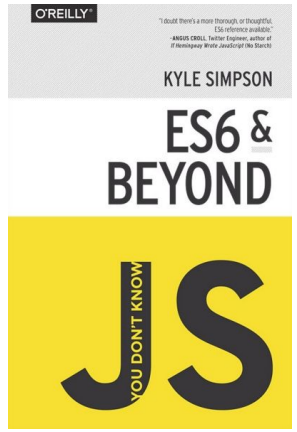
Más Información

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