

***RECUERDA PONER A GRABAR LA
CLASE***





¿DUDAS DEL ON-BOARDING?

MIRALO AQUI



Clase 10. JAVASCRIPT

WORKSHOP I



OBJETIVOS DE LA CLASE

- Hacer un breve repaso por todos los temas.
- Recomendaciones principales para el proyecto final.
- Avanzar individualmente con el proyecto final.

GLOSARIO:

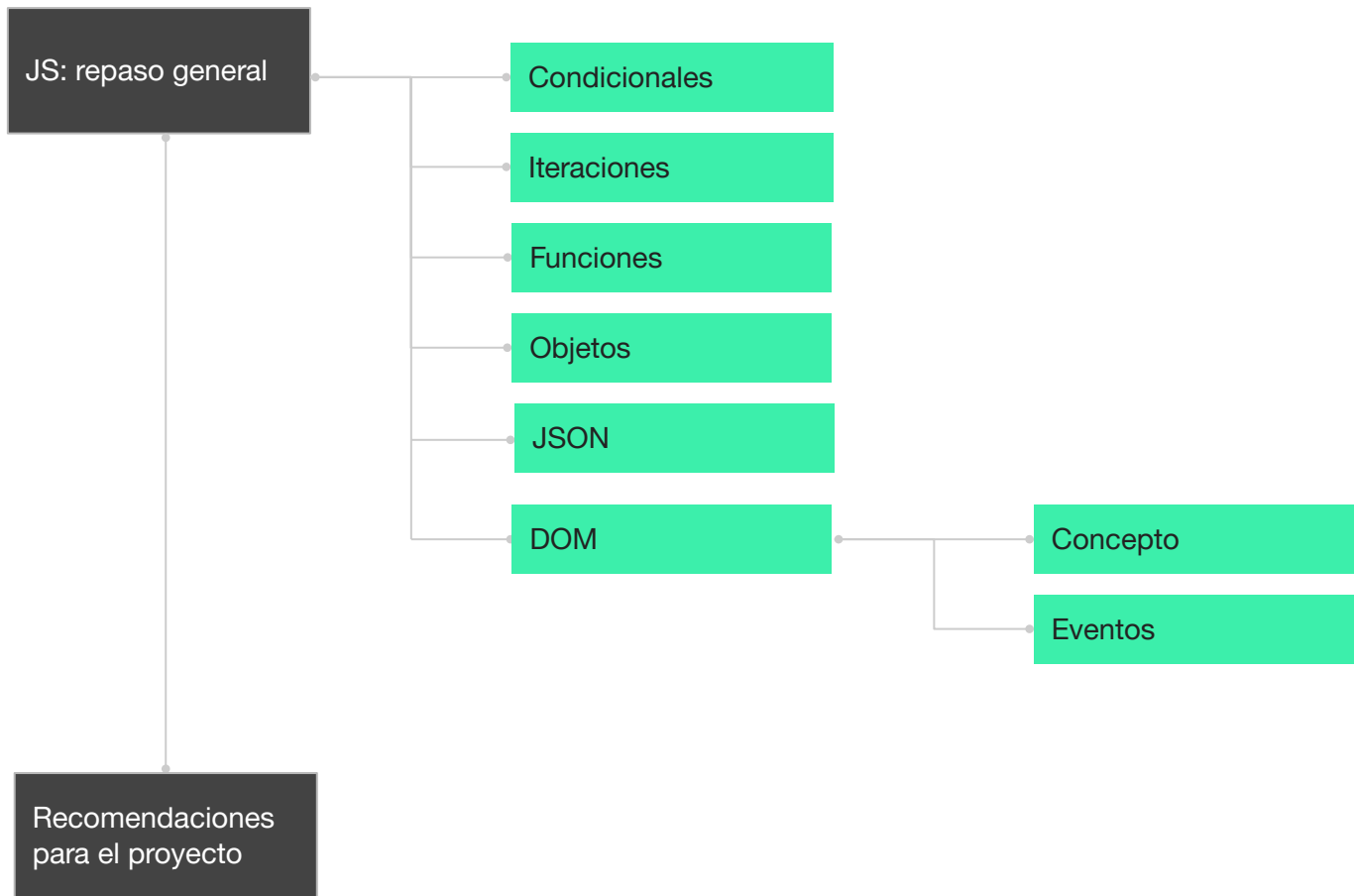
Clase 9

Evento: es la manera que tenemos en Javascript de controlar las acciones de los usuarios, y definir un comportamiento de la página o aplicación cuando se produzcan.

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 10

¡Para
recordar!



CRONOGRAMA DEL CURSO

Clase 9



Eventos



EJEMPLOS EN VIVO



INCORPORAR EVENTOS

Clase 10



Workshop I



EJEMPLOS EN VIVO



SEGUNDA ENTREGA DEL
PROYECTO FINAL

Clase 11



jQuery



EJEMPLO EN VIVO



JQUERY

JS: REPASO GENERAL

REPASEMOS...



1- *CONDICIONALES*

Los **condicionales** son sentencias que podemos utilizar para interpretar un conjunto de instrucciones en función del resultado de una comparación.

```
let nombreIngresado = prompt("Ingresar nombre");

if((nombreIngresado != "") && ((nombreIngresado == "EMA") || (nombreIngresado == "ema"))){
    alert("Hola Ema");
}else{
    alert("Error: Ingresar nombre valido");
}
```

2- BUCLES

Los **bucles** son sentencias que podemos utilizar para repetir un conjunto de instrucciones más de una vez de forma consecutiva.

```
let entrada = prompt("Ingresar un dato");  
//Repetimos con While hasta que el usuario ingresa "ESC"  
while(entrada != "ESC" ){  
    alert("El usuario ingresó "+ entrada);  
    //Volvemos a solicitar un dato. En la próxima iteración se evalúa si no es ESC.  
    entrada = prompt("Ingresar otro dato");  
}
```

3- FUNCIONES

Los **funciones** son un conjunto de instrucciones destinadas a resolver una situación en el programa. Podemos reutilizarlas y modificarlas fácilmente.

```
const suma = (a, b) => { return a + b };  
//Si es una función de una sola línea con retorno podemos evitar escribir el cuerpo.  
const resta = (a, b) => a - b ;  
console.log(suma(15,20));  
console.log(resta(20,5));
```

4- OBJETOS

Los **objetos** son estructuras que podemos definir para agrupar valores bajo un mismo criterio y asignarle comportamiento.

```
function Persona(nombre, edad, calle) {  
  this.nombre = nombre;  
  this.edad   = edad;  
  this.calle  = calle;  
  this.hablar = function() { console.log("HOLA SOY " + this.nombre)}  
}  
  
const persona1 = new Persona("Homero", 39, "Av. Siempreviva 742");  
const persona2 = new Persona("Marge", 36, "Av. Siempreviva 742");  
persona1.hablar();  
persona2.hablar();
```

5- ARRAY

Los **Arrays** son objetos que nos permite agrupar distintos elementos (incluso otros objetos). Son recursos muy útiles por sus métodos para aplicar filtros.

```
const numeros = [1, 2, 3, 4, 5];  
const porDos = numeros.map(x => x * 2); // porDos = [2, 4, 6, 8, 10]  
const masCien = numeros.map(x => x + 100); // porDos = [102, 104, 106, 108, 110]  
  
const nombres = ["Ana", "Ema", "Juan", "Elia"];  
const lengths = nombres.map(nombre => nombre.length); //lengths = [3, 3, 4, 4]
```

6- STORAGE Y JSON

```
const productos = [{ id: 1, producto: "Arroz", precio: 125 },
                    { id: 2, producto: "Fideo", precio: 70 },
                    { id: 3, producto: "Pan" , precio: 50},
                    { id: 4, producto: "Flan" , precio: 100}];
```

```
const guardarLocal = (clave, valor) => { localStorage.setItem(clave, valor) };
```

```
//Almacenar producto por producto
```

```
for (const producto of productos) {
    guardarLocal(producto.id, JSON.stringify(producto));
}
```

```
// o almacenar array completo
```

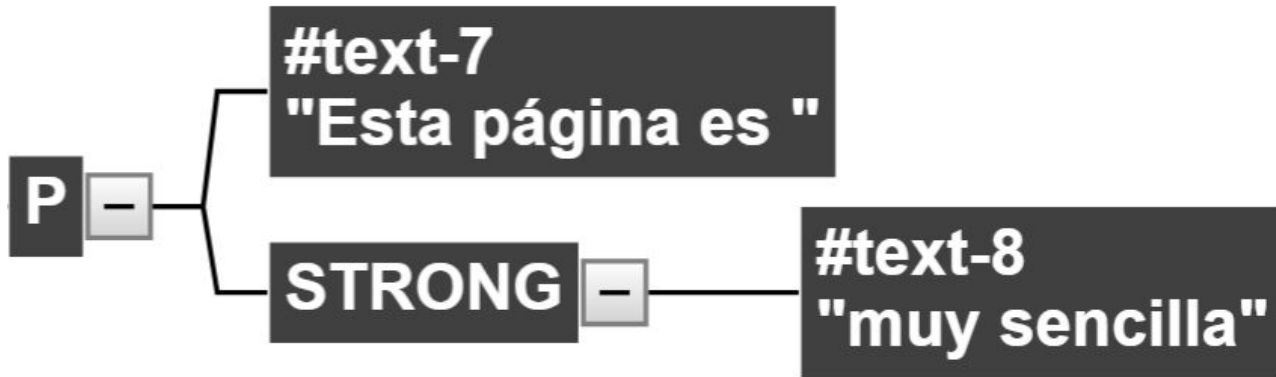
```
guardarLocal("listaProductos", JSON.stringify(productos));
```


7- CONCEPTO DE DOM

El **DOM** es la representación del documento HTML que podemos emplear para modificar la página actual dinámicamente.

```
<p>Esta página es <strong>muy sencilla</strong></p>
```

La etiqueta <p> se transforma en los siguientes nodos del DOM:



8- EVENTOS EN EL DOM

Los **eventos** son la manera que tenemos en Javascript de controlar las acciones de los usuarios y definir un comportamiento de aplicación cuando se produzcan.

```
//CODIGO HTML DE REFERENCIA
<form id="formulario">
  <input type="text">
  <input type="number">
  <input type="submit" value="Enviar">
</form>

//CODIGO JS
let miFormulario = document.getElementById("formulario");
miFormulario.addEventListener("submit", validarFormulario);

function validarFormulario(e){
  e.preventDefault();
  console.log("Formulario Enviado");
}
```



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

RECOMENDACIONES PARA EL PROYECTO

ESTRUCTURA DE ARCHIVOS

Asegúrate de mantener ordenados tus archivos del proyecto:

- Tendrás como mínimo un `archivo.html`, y en el `<head>` cargarás al menos un archivo `.js`
- Al menos un archivo `.css` para el estilo del proyecto.
- Podés tener más archivos `.js` si decidís agruparlos por separado. Por ejemplo: uno procesa todo lo que son operaciones de usuario, y otro todas las operaciones sobre el DOM.

CÓDIGO ORDENADO

Siempre es importante mantener el código ordenado.

No sólo por si otro programador necesita acceder, sino para ayudarte a vos mismo.

Tal vez dejas pasar unos días entre un cambio y otro, y por tener el código repetido, desordenado y sin comentarios, perdes mucho tiempo en entenderlo nuevamente.

PROYECTO ORDENADO

Siempre piensa y boceta (en tu cabeza o en un papel) cómo funcionará cada módulo del proyecto.

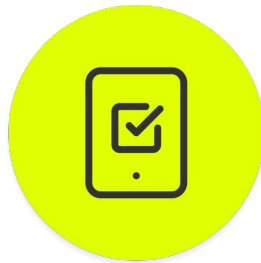
No improvises, busca referencias, ejemplos. Es más rápido desarrollar una vez una función, que tener que modificarla varias veces por notar fallas al momento de utilizarla.

USOS CORRECTOS EN JS

Recuerda cumplir las buenas prácticas en JS:

- No escribas mucho código js inline dentro del archivo html. (entre etiquetas <script>)
- Para el uso de funciones, métodos, etc recuerda siempre usar archivos externos de extensión .js.
- Si vas a usar objetos, también es ideal que cada uno tenga su propio archivo .js

***¡AHORA VAMOS A VER EL ESTADO DE
AVANCE DE SUS PROYECTOS!***



SEGUNDA ENTREGA DEL PROYECTO FINAL

Deberás agregar y entregar **uso de JSON y Storage, y DOM y eventos del usuario**, correspondientes a la segunda entrega de tu proyecto final.

SEGUNDA ENTREGA DEL PROYECTO FINAL

Formato: Página HTML y código fuente en JavaScript. Debe identificar el apellido del alumno/a en el nombre de archivo comprimido por “claseApellido”.

Sugerencia: En la segunda entrega buscamos programar el código esencial para garantizar dinamismo en el HTML con JavaScript.

En relación a la primer entrega, ya no usamos alert() como salida y prompt() como entrada, ahora modificamos el DOM para las salidas y capturamos los eventos del usuario sobre inputs y botones para las entradas. Verificar Rúbrica

Proyecto
Final



2

>>Objetivos Generales:

1. Codificar funciones de procesos esenciales y notificación de resultados por HTML, añadiendo interacción al simulador.
2. Ampliar y refinar el flujo de trabajo del script en términos de captura de eventos, procesamiento del simulador y notificación de resultados en forma de salidas por HTML, modificando el DOM.

>>Objetivos Específicos:

1. Definir eventos a manejar y su función de respuesta.
2. Declarar una estructura de datos de tipo JSON, para definir datos iniciales a consumir por el simulador.
3. Modificar el DOM, ya sea para definir elementos al cargar la página o para realizar salidas de un procesamiento.
4. Almacenar datos (clave-valor) en el Storage y recuperarlos.

SEGUNDA ENTREGA DEL PROYECTO FINAL

>>Se debe entregar:

- Implementación con uso de JSON y Storage.
- Modificación del DOM y detección de eventos de usuario.

¿PREGUNTAS?





***TE INVITAMOS A QUE COMPLEMENTES
LA CLASE CON LOS SIGUIENTES
CODERTIPS***



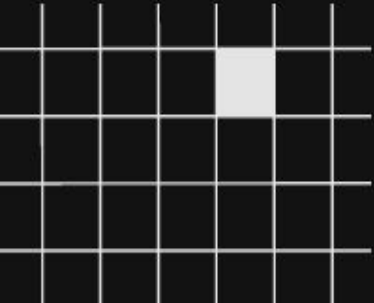
VIDEOS Y PODCASTS

- 5 Tips para tener un código limpio con Javascript | **CODERHOUSE** |
<https://www.youtube.com/watch?v=R2Nkv3P-9CU>



¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Repasamos conceptos generales de JS, y dudas.
 - Avanzamos en el proyecto integrador.
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE