Ägile Innova
‹Academy›

Semana 2
Sesión 4

- **Integración HTML-CSS-JS**
- **Bucles**
- **Funciones**
- **Eventos**

¿ Qué es el DOM ?

Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina árbol DOM (*o simplemente DOM*). Así que cuando nos referimos al DOM, estamos hablando de dicha estructura que podemos modificar de forma dinámica añadiendo nuevas etiquetas, modificando o eliminando otras, cambiando sus atributos HTML, añadiendo clases, cambiando el contenido de texto, etc.

Seleccionar elementos del DOM

1 getElementById("id del elemento"): Cuando necesitamos utilizar un elemento en específico, utilizamos este método, que realiza una búsqueda por todo el DOM hasta conseguir el ID especificado (DEBIDO A QUE EL ID ES UN IDENTIFICADOR ÚNICO, SOLO RETORNARÁ UN ELEMENTO)

NOTA: El ID del elemento a buscar SIEMPRE se coloca dentro de los paréntesis, y este se recibirá como un STRING.

2. querySelector(“id, clase o etiqueta”): Este método nos permite capturar un elemento específico, sin embargo, podemos buscar mediante el ID, clase o etiqueta del elemento, solo retorna el primer elemento que coincida con el parámetro de búsqueda especificado dentro de los paréntesis(dicho parámetro debe ser un string)

NOTA: Si vamos a buscar un elemento por su ID, debemos utilizar su selector (“#idDelElemento”), de igual forma si vamos a buscar por su clase (“.claseDelElemento”); y si solo vamos a buscar una etiqueta, se coloca el nombre de la misma (“div”)

Otros métodos para seleccionar elementos del DOM

- .getElementsByClassName(class)
- .getElementsByName(name)
- .getElementsByTagName(tag)
- .querySelectorAll(sel)

Métodos para manipular el DOM

- **setAtributte("atributo", "valor"): Nos permite actualizar los atributos de una etiqueta, y si ese atributo no existe, lo crea con el valor que le coloquemos**

HTML:

```
<!-- Imagen sin el atributo src -->
<img alt="perrito">
```

Manipulando el CSS desde JS

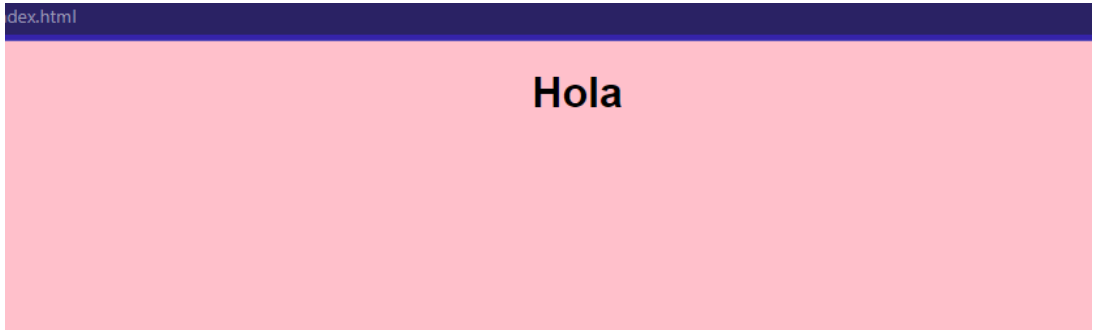
HTML:

```
<body>

    <h1>Hola</h1>

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

JS:

```
// Capturamos el elemento
let body = document.querySelector("body");


//Actualizando diferentes estilos
body.style.background = "pink";
body.style.fontFamily = "Helvetica";
body.style.textAlign = "center"
```

Resultado:



Métodos importantes para la manipulación del CSS desde JS

- **classList:** nos permite listar todas las clases que contiene un elemento

- **classList.add("nombre de la clase a agregar"):** Nos permite agregarle una clase a un elemento.

- **classList.remove("nombre de la clase a eliminar"):** Nos permite eliminar una clase de un elemento.

- **classList.contains("nombre de la clase"): retorna true si la clase existe en el elemento, de lo contrario retorna false**

- **classList.replace ("primer clase", "segunda clase"): Nos permite reemplazar clases en un elemento, la primera posición nos indica el nombre de la clase que vamos a reemplazar, y la segunda nos indica el nombre de la clase por el cual será reemplazada**

Crear elementos en el DOM

- **createElement("etiqueta a crear"): Nos permite crear una etiqueta, sin embargo, esta se crea TOTALMENTE VACÍA.**

```
let image = document.createElement('img');
```

Insertar elementos en el DOM

- **appendChild(elemento):** Permite añadir un elemento al final de un contenedor (solo recibe valores de tipo node)

HTML:

```
<body>

    <div id="container"></div>


    <script src="./script/script.js"></script>
</body>

</html>
```

JS:

```
1   // creamos una imagen
2   let image = document.createElement('img');
3   // capturamos ek contenedor
4   let contenedor = document.getElementById('container');
5
6   //actualizamos el atributo src
7   image.setAttribute('src', 'https://www.tooltyp.com/wp-cont
8
9   // al contenedor seleccionado insertamos la
10  // etiqueta creada en la linea 2
11  contenedor.appendChild(image);
12
```

- **innerHTML:** Permite agregar o reemplazar el contenido de un elemento padre, este método recibe código

HTML EN FORMA DE STRING

HTML:

```
<div id="container">
    <h1>Bienvenido</h1>
</div>
```

Reemplazo

Con el operador “=” reemplazamos todo el contenido del elemento padre.

```
1   // capturamos el contenedor
2   let contenedor = document.getElementById('container');
3
4   //Acá reemplazamos todo el contenido original
5   contenedor.innerHTML = `
6     <img src='https://cdn2.excelsior.com.mx/media/styles/ir
7   `;
8
9
```

Bucles e iteraciones

Una de las principales ventajas de la programación es la posibilidad de crear bucles y repeticiones para tareas específicas, y que no tengamos que realizarlas varias veces de forma manual. Existen muchas formas de realizar bucles, vamos a ver los más básicos, similares en otros lenguajes de programación:

- For
- While

Partes de un bucle

- **Condición:** Al igual que en los if, en los bucles se va a evaluar una condición para saber si se debe repetir el bucle o finalizarlo. Generalmente, si la condición es verdadera, se repite. Si es falsa, se finaliza.

- **Iteración:** Cada repetición de un bucle se denomina iteración. Por ejemplo, si un bucle repite una acción 10 veces, se dice que tiene 10 iteraciones.

- **Contador**: Muchas veces, los bucles tienen una variable que se denomina contador, porque cuenta el número de repeticiones que ha hecho, para finalizar desde que llegue a un número concreto. Dicha variable hay que inicializarla (crearla y darle un valor) antes de comenzar el bucle.

- **Incremento**: Cada vez que terminemos un bucle se suele realizar el incremento (o decremento) de una variable, generalmente la denominada variable contador.

Bucle while

El bucle while es uno de los bucles más simples que podemos crear. Vamos a repasar el siguiente ejemplo y todas sus partes, para luego repasar que ocurre en cada iteración del bucle:

```
// variable contadora;
let contador = 1;

//      condición
while(contador <= 10 ){
    //Cuerpo del condicional
     console.log("Esta es la iteración nº " + contador);


     //Incremento
     contador = contador + 1;
}
```

Bucle For

El bucle for es quizás uno de los más utilizados en el mundo de la programación. En Javascript se utiliza exactamente igual que en otros lenguajes como Java o C/C++. Veamos el ejemplo anterior utilizando un bucle for:

```
//    variable          condición    incremento
for(let contador = 1; contador<=10; contador++){
    //Cuerpo del condicional
    console.log("Estamos en la vuenta n° " + contador);
}
```

Segmento de ejercicios + socialización 1

- **Escribir un programa que pregunte al usuario su edad y muestre por pantalla todos los años que ha cumplido (desde 1 hasta su edad).**

- **Escribir un programa que pida al usuario un número entero positivo y muestre por pantalla la cuenta atrás desde ese número hasta cero.**

- **Escribir un programa que muestre por pantalla la tabla de multiplicar del 1 al 10.**

- **Escribir un programa que muestre lo que el usuario ingresó mediante el promt hasta que el usuario escriba "salir".**

Funciones en JavaScript

Las funciones son una forma de agrupar código que vamos a usar varias veces permitiéndonos además, pasar diferentes valores para obtener diferentes resultados.

Segmento de ejercicios + socialización 2

Control de horas

Una persona necesita calcular su salario semanal, el cual se obtiene de la siguiente manera:

1. Si trabaja 40 horas o menos se le paga a $16.000 por hora

2. Si trabaja más de 40 horas se le paga $16.000 por cada una de las primeras 40 horas y $20.000 por cada hora extra

Reto

Una universidad tiene el siguiente plan de tarifas de acuerdo a la cantidad de créditos que el estudiante desee matricular: Si un estudiante matrícula menos o igual de 6 créditos tiene el 50% de descuento sobre el valor de la matrícula, si matrícula más de 6 pero menos o igual de 10, tiene un 25% de descuento sobre el valor de la matrícula, y si el estudiante matricula más de 10 créditos deberá pagar el total del valor de la matrícula. Hacer un programa que calcule el valor a pagar de un estudiante teniendo en cuenta que el valor de la matrícula es de $900.000

Eventos

Los eventos son acciones que suceden en el sistema que está programando y que el sistema le informa para que pueda responder de alguna manera si lo desea.

Podemos identificar como eventos cuando el usuario mueve el mouse, clic sobre algún elemento, se escribe algo en un input, cuando el usuario borra algo, cuando el usuario redimensiona una ventana.

element.addEventListener

Registra un evento a un objeto en específico. Es un método que recibe dos parámetros, en el primero se indica el evento a escuchar y en el segundo se declara la función a ejecutar

```
// capturamos el elemento
let botonSubmit = document.getElementById('button');

// Le agregamos un evento al elemento capturado
botonSubmit.addEventListener('evento', function(){})
```

Eventos del mouse

- **Evento click**

```
<body>

    <button id="boton">Presióname</button>

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

```
// Capturamos el elemento;
let boton = document.getElementById('boton');

//Agregamos el evento Click
boton.addEventListener('click', function(){
  alert("Este es el evento click");
})
```

- **Evento mouseOver**

```
// Capturamos el elemento;
let boton = document.getElementById('boton');

//Agregamos el evento MouseOver
boton.addEventListener('mouseover', function(){
  boton.style.background = "green";
  boton.style.padding = "10px";
  boton.style.color = "white";
})
```

- **Evento mouseout**

```
// Capturamos el elemento;
let boton = document.getElementById('boton');

//Agregamos el evento MouseOut
boton.addEventListener('mouseout', function(){
    boton.style.background = "orange";
    boton.style.padding = "100px";
    boton.style.color = "black";
})
```

Eventos de focos

```
<body>

    <form>
        <input type="text" id="inputText">
    </form>


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

- Focus

```
// Capturamos el Input
let input = document.getElementById('inputText');

input.addEventListener('focus', function(){
  console.log("Se ejecutó el evento focus")
})
```

- Blur

```
// Capturamos el Input
let input = document.getElementById('inputText');

input.addEventListener('blur', function(){
  console.log("Se ejecutó el evento blur")
})
```

Eventos del DOM

- **DOMContentLoaded**

```
// Evento que se ejecutá al cargar la estructura básica del HTML
document.addEventListener('DOMContentLoaded', function(){
  alert("Se ejecutó el evento");
})
```

Otros eventos

- **submit:** se activa cuando el formulario es enviado, normalmente se utiliza para validar el formulario antes de ser enviado al servidor o bien para abortar el envío y procesarlo con JavaScript.

- **change:** se dispara para elementos <input>, <select>, y <textarea> cuando una alteración al valor de un elemento es confirmada por el usuario.