

DIPLOMADO

Front End



UVM

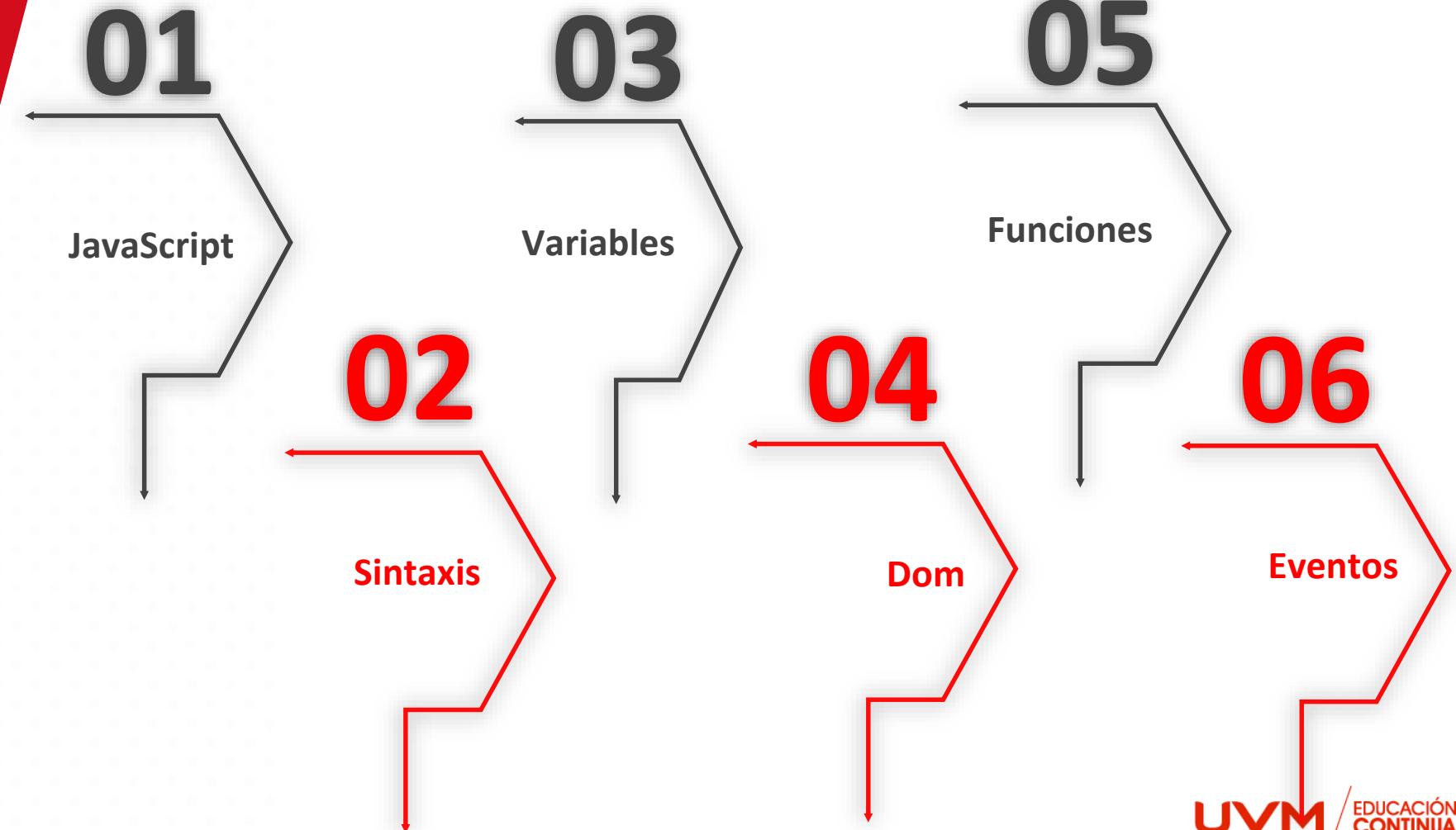
Introducción

Objetivo

Comprender el uso de JavaScript para controlar una página web. Aprenderás qué es el Modelo de objetos de documento (**DOM**) y utilizarás JavaScript y DOM para dictar el contenido y las interacciones de la página. Obtendrás experiencia trabajando con **Eventos** del navegador y administrando el rendimiento del sitio web controlando la creación de contenido de manera eficiente.



Temario



Temario

07

Listener

09

Objetos

11

Angular

08

Array

10

Librerías

12

React

DIPLOMADO

Front End

```
<script>
2   (function () {
3     window._harvestPlatformConfig = {
4       "application": "eCompany"
5       "permalink": "http://eCompany.simple.com"
6     };
7     var s = document.createElement("script");
8     s.src = "https://eCompany.simple.com/_harvest/config.js?nocache=true";
9     var ph = document.getElementsByTagName("head")[0];
10    ph.appendChild(s);
11  })();
12 </script>
```

UVM

TEMA 1

Javascript

Javascript



- JavaScript es un **lenguaje de programación o de secuencias de comandos** que te permite implementar funciones complejas en páginas web y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc.
- JavaScript (abreviado comúnmente **JS**) es un lenguaje de programación interpretado, dialecto del estándar **ECMAScript**. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Características

- **Imperativo y estructurado:**

JavaScript es compatible con gran parte de la estructura de programación de C (por ejemplo, sentencias if, bucles for, sentencias switch, etc.).

- **Lenguaje del lado del cliente:**

Cuando se dice que un lenguaje es del **lado del cliente**, nos referimos a que **se ejecuta en la máquina del propio cliente a través de un navegador**. Algunos de estos lenguajes son el propio javascript, HTML, CSS o Java.

Esta categoría de lenguajes se diferencia de la otra gran categoría: los lenguajes del **lado del servidor**. **Estos lenguajes se ejecutan e interpretan por el propio servidor** y necesitan un tratamiento antes de mostrarlos al usuario final. Algunos de los lenguajes de programación del lado del servidor más conocidos son **PHP**, **ASP** o **PERL**.

Características

- **Dinámicos.**
 - Como en la mayoría de lenguajes de scripting, el tipo está asociado al valor, no a la variable. Por ejemplo, una variable x en un momento dado puede estar ligada a un número y más adelante, religada a una cadena.
- **Objetual.**
 - JavaScript está formado casi en su totalidad por objetos. Los objetos en JavaScript son arrays asociativos, mejorados con la inclusión de prototipos. Evaluación en tiempo de ejecución.
- **Funcional.**
 - Funciones de primera clase
 - A las funciones se les suele llamar ciudadanos de primera clase; son objetos en sí mismos.

Características

- **Prototípico.**
 - JavaScript usa **prototipos** en vez de clases para el uso de herencia. Es posible llegar a emular muchas de las características que proporcionan las clases en lenguajes orientados a objetos tradicionales por medio de prototipos en JavaScript.
- **Funciones como métodos.**
 - A diferencia de muchos lenguajes orientados a objetos, **no hay distinción entre la definición de función y la definición de método**. Más bien, la distinción se produce durante la llamada a la función; una función puede ser llamada como un método.
- **Arrays y la definición literal de objetos.**
 - **Expresiones regulares.**

Uso

- Se utiliza principalmente **del lado del cliente**, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas y JavaScript del lado del servidor (Server-side JavaScript o SSJS)
- Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del **Document Object Model (DOM)**.
- Dado que el código JavaScript puede ejecutarse localmente en el navegador, este puede responder a las acciones del usuario con rapidez, haciendo una aplicación más sensible.
- Por otra parte, el código JavaScript puede detectar acciones de los usuarios que HTML por sí sola no puede, como pulsaciones de teclado
- La tendencia cada vez mayor por el uso de la programación **Ajax** explota de manera similar esta técnica.

Uso

El uso más común de JavaScript es escribir funciones embebidas o incluidas en páginas HTML y que interactúan con el **Document Object Model (DOM o Modelo de Objetos del Documento)** de la página. Algunos ejemplos sencillos de este uso son:

- Cargar nuevo contenido para la página o enviar datos al servidor a través de **AJAX** sin necesidad de recargar la página (por ejemplo, una red social puede permitir al usuario enviar actualizaciones de estado sin salir de la página).
- **Animación de los elementos de página**, hacerlos desaparecer, cambiar su tamaño, moverlos, etc.
- **Contenido interactivo**, por ejemplo, juegos y reproducción de audio y vídeo.
- **Validación de los valores de entrada de un formulario** web para asegurarse de que son aceptables antes de ser enviado al servidor.
- **Transmisión de información sobre los hábitos de lectura de los usuarios y las actividades de navegación a varios sitios web**. Las páginas Web con frecuencia lo hacen para hacer análisis web, seguimiento de anuncios, la personalización o para otros fines

TEMA 2

DIPLOMADO

Front End

Sintaxis Javascript

```
javac -classpath .:/run_dir/junit-4.12.jar -d . Main.  
java -classpath .:/run_dir/junit-4.12.jar Main  
Edades[0] = 23  
Edades[1] = 50  
Edades[2] = 80  
Edades[3] = 18  
Edades[4] = 20  
Numeros[0] = 1  
Numeros[1] = 3  
Numeros[2] = 4  
Numeros[3] = 5  
Numeros[4] = 5  
Mascotas[0] = Mascota{nombre:  
Mascotas[1] = Mascota{nombr
```

Sintaxis

Las variables en JavaScript se definen usando la palabra clave **var**.



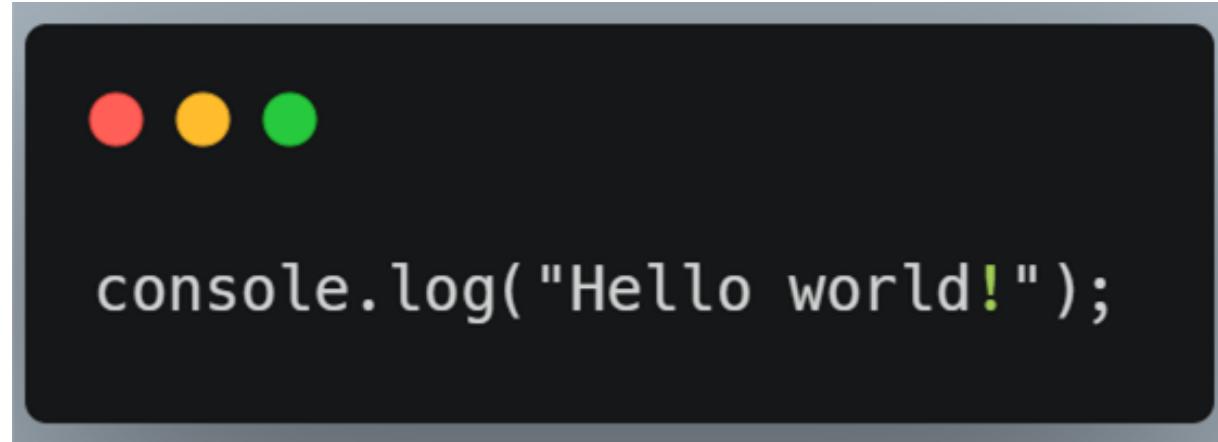
```
/*
Ejemplos de comentarios
*/
var variable; // define la variable sin ningún valor asignado
var otravariable = 100; // define la variable con valor asignado
```

objetos principales:

- Declaración de variables.
- Comentarios.

Imprimir Resultados

- Sin embargo, la mayoría de los entornos de ejecución tiene un objeto llamado console que puede ser usado para imprimir por el flujo de salida de la consola de depuración. He aquí un simple programa que imprime “Hello world!>:



Operadores aritméticos y conversiones

Operador	Nombre	Propósito
+	Adición	Suma dos números juntos.
-	Resta	Resta el numero de la derecha del de la izquierda.
*	Multiplicación	Multiplica dos números juntos.
/	División	Divide el número de la izquierda por el de la derecha.

Método	Descripción
parseFloat()	Analiza una cadena y devuelve un número de coma flotante
parseInt()	Analiza una cadena y devuelve un entero

DIPLOMADO

Front End

UVM

TEMA 3

Variables

Variables - var, let, const

En JavaScript la palabra reservada utilizada tradicionalmente para la definición de variables es **var**. Utilizada desde los inicios del lenguaje.

En JavaScript **ECMAScript 6** aparecen dos nuevas palabras reservadas para la definición de variables: **let** y **const**.

Las variables definidas con **var** tienen un ámbito de **función**. Por lo tanto, pueden ser utilizadas en cualquier punto de la función, incluso antes de ser definidas.

var

```
function explainVar(){
    var a = 10;
    console.log(a); // salida 10
    if(true){
        var a = 20;
        console.log(a); // salida 20
    }
    console.log(a); // salida 20
}
```

Podemos ver que cuando la variable es actualizada dentro del condicional if, el valor de esta se actualiza a **20 a nivel global**.

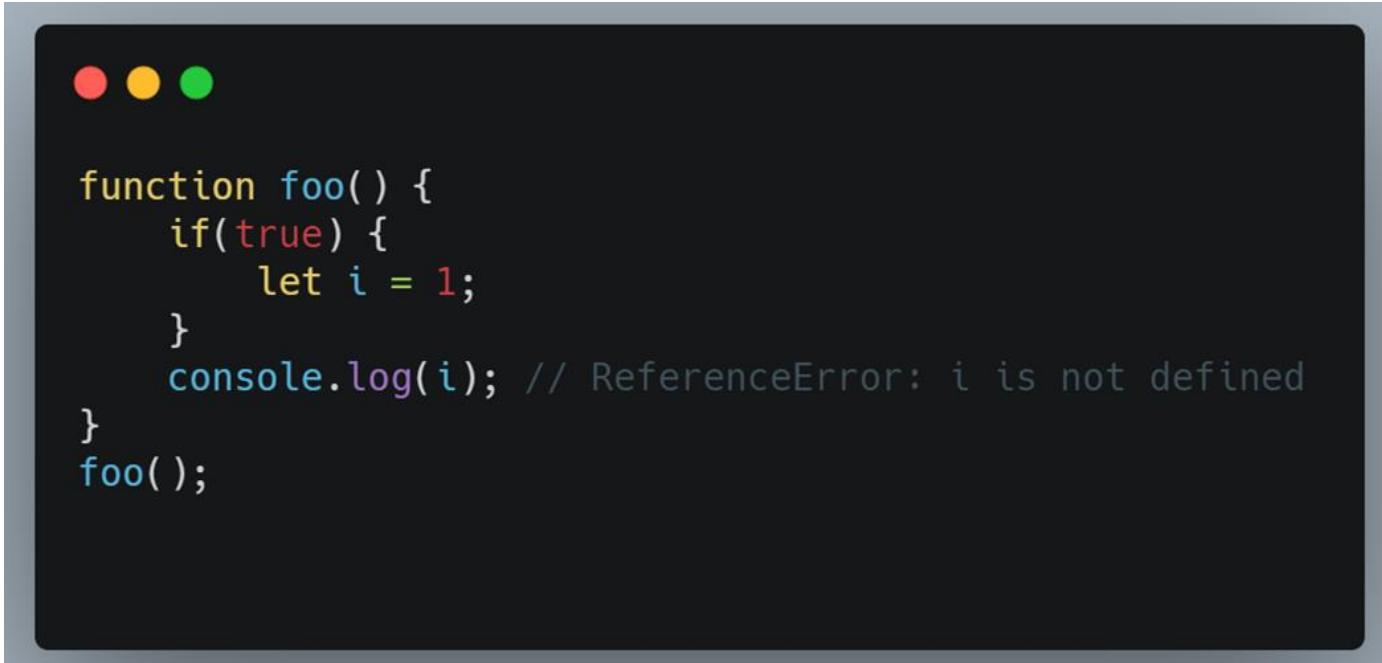
let

- Si declaramos una variable con **let** dentro un bloque que a su vez está dentro de una función, la variable pertenece solo a ese bloque:

```
function foo() {  
    let i = 0;  
    if(true) {  
        let i = 1; // Sería otra variable i solo para el bloque if  
        console.log(i); // 1  
    }  
    console.log(i); // 0  
}  
foo();
```

Let fuera del scope

Fuera del bloque donde se declara con **let**, la variable no está definida:

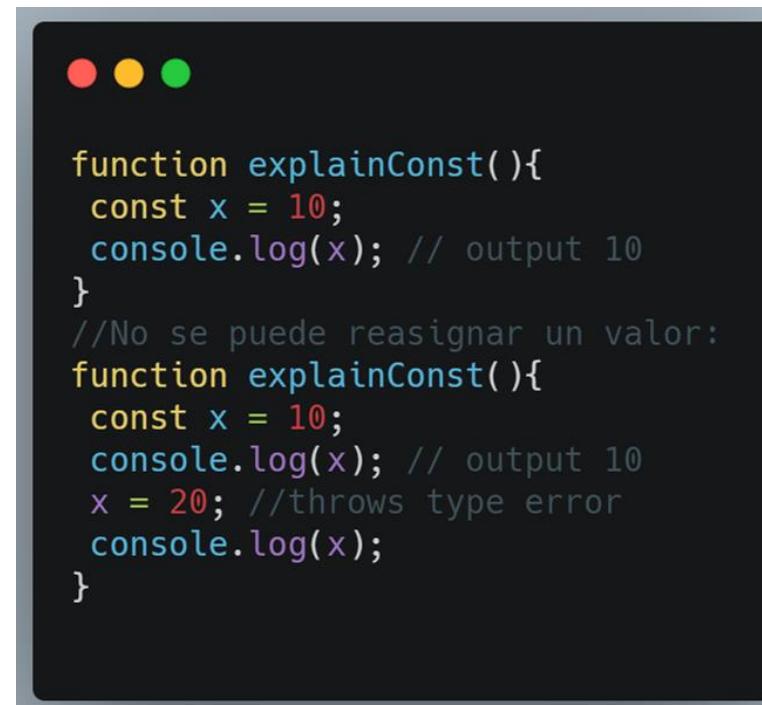


```
function foo() {
  if(true) {
    let i = 1;
  }
  console.log(i); // ReferenceError: i is not defined
}
foo();
```

//Mostrará error porque solo se declaró dentro del bloque if

Const

- **Const** es similar que **let**, con una pequeña gran diferencia: no puedes reasignar su valor.



```
function explainConst(){
  const x = 10;
  console.log(x); // output 10
}
//No se puede reasignar un valor:
function explainConst(){
  const x = 10;
  console.log(x); // output 10
  x = 20; //throws type error
  console.log(x);
}
```

Mostrará un error cuando tratemos de reasignar el valor de una variable const:
Error Message : Uncaught TypeError: Assignment to constant variable.

Ámbito de las palabras claves let y const

- La diferencia entre **let** y **const** es que las variables definidas mediante **la primera se pueden modificar** durante la ejecución, mientras que las creadas con **la segunda son constantes**. Indicando así tanto al intérprete de JavaScript, permitiendo reservar la memoria de forma más eficiente.
- Así cuando se sabe que el valor asignado a una variable no va a cambiar durante la ejecución es mejor utilizar **const** para su definición.

DIPLOMADO

Front End

EJERCICIO 1

Definiciones de Variables
y Comentarios

UVM

DIPLOMADO

Front End



UVM

TEMA 4

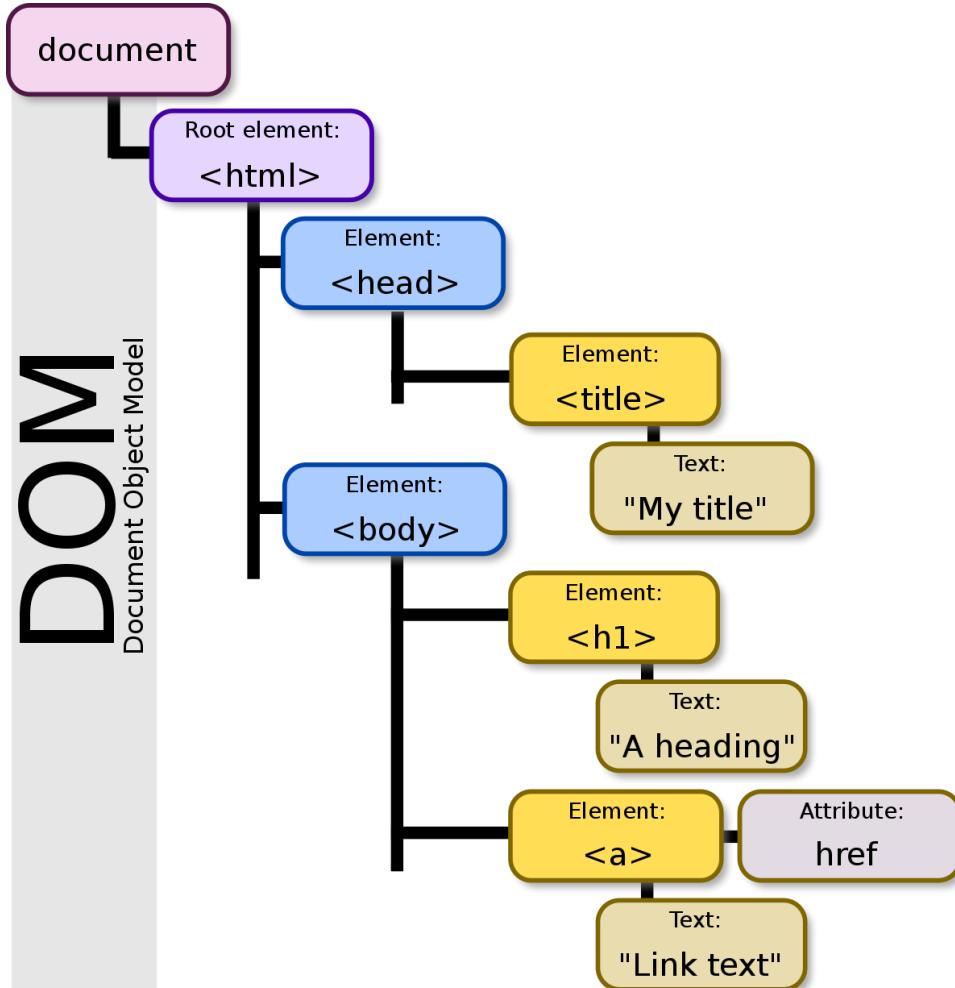
DOM

HTML (HyperText Markup Language)



- Es el componente más básico de la Web. **Define el significado y la estructura del contenido web.** Además de **HTML**, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página web (**CSS**) o la funcionalidad/comportamiento (**JavaScript**).
- El modelo de objeto de documento (**DOM**) proporciona otras formas de presentar, guardar y manipular este mismo documento. El DOM es una representación completamente orientada al objeto de la página web y puede ser modificado con un lenguaje de script como JavaScript.

DOM



- En aplicaciones web tenemos un concepto llamado **DOM** (Document Object Model) es la forma en que internamente el navegador organiza todo el HTML dentro de una estructura de árbol.
- Un documento HTML se carga en un navegador web, se convierte en un objeto de documento.
- El DOM es el nodo raíz del documento HTML.

Ejemplos de Propiedades - DOM

El DOM es nuestro lugar de trabajo en la web

navigator: El objeto que contiene las funciones del navegador.

window: El objeto que maneja cada una de las pestañas.

document: El objeto que contiene todo lo que vemos dentro de nuestra página

Propiedades / Metodos	
<u>attr.isId</u>	
<u>attr.name</u>	
<u>attr.value</u>	
<u>attr.specified</u>	
<u>nodeMap.getNamedItem()</u>	
<u>nodeMap.item()</u>	
<u>nodeMap.length</u>	
<u>nodeMap.removeNamedItem()</u>	
<u>nodeMap.setNamedItem()</u>	

Métodos de acceso del DOM

- **document.getElementById(id)**
- **document.getElementsByClassName(nombreDeClase)**
- **document.getElementsByTagName(nombreDeEtiqueta)**
- **document.querySelector(selectorCss)**
- **document.querySelectorAll(selectorCss)**

¿Qué es lo que regresan?

Hay dos métodos que regresan un único objeto Element, **getElementById** y **querySelector**:

```
var elemento = document.getElementById("elemento");
elemento.innerHTML = "Acceder al dom";
```

Los métodos **getElementsByName** y **getElementsByClassName** regresan un objeto **HTMLCollection** que actúa como un arreglo. Esa colección está "viva", lo que significa que la colección se actualiza si al documento se le agregan elementos adicionales con nombre de etiqueta o nombre de clase.

```
var teamMembers = document.getElementsByClassName("team-member");
for (var i = 0; i < teamMembers.length; i++) {
    console.log(teamMembers[i].innerHTML);
}
```

¿Qué es lo que regresan?

- El método **querySelectorAll()** regresa un **NodeList**, que también actúa como un arreglo. Esa lista es "**estática**", lo que significa que la lista no se actualizará aunque se le agreguen nuevos elementos coincidentes a la página.

```
var teamMembers = document.querySelectorAll(".team-member");
for (var i = 0; i < teamMembers.length; i++) {
    console.log(teamMembers[i].innerHTML);
}
```

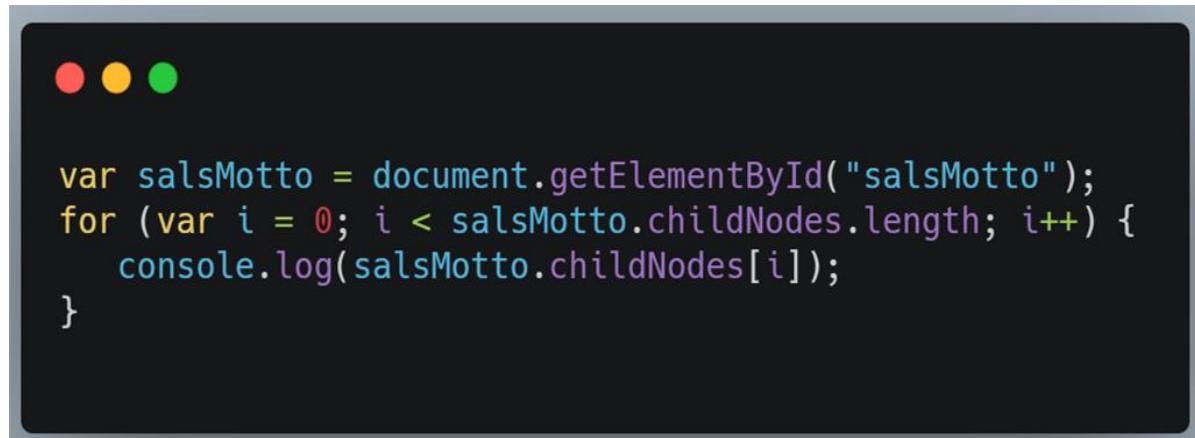
Acceder con Subconsultas

- Una vez que hayas encontrado un elemento, puedes hacer subconsultas sobre él al usar los métodos que acabamos de mostrar. Por ejemplo:

```
// encuentra el elemento con esa ID
var salsMotto = document.getElementById("salsMotto");
// encuentra los spans dentro de ese elemento:
var mottoWords = salsMotto.getElementsByTagName("span");
// escribe en consola cuántos hay
console.log(mottoWords.length);
```

Leer DOM

- Otra manera de acceder a los elementos es "atravesar" el árbol del DOM. Cada elemento tiene propiedades que apuntan a los elementos relacionados con él:
 - **firstElementChild**
 - **lastElementChild**
 - **nextElementChild/nextElementSibling**
 - **previousElementChild/previousElementSibling**
 - **childNodes**
 - **childElementCount**



```
var salsMotto = document.getElementById("salsMotto");
for (var i = 0; i < salsMotto.childNodes.length; i++) {
  console.log(salsMotto.childNodes[i]);
}
```

Estas propiedades no están disponibles en los nodos Text, solo en los nodos Element. Para asegurarte de que puedas atravesar un elemento, puedes revisar sus propiedades `nodeType/nodeValue`.

Técnicas de Modificación del DOM

Modificar atributos



```
imgEl.src = "http://www.dogs.com/dog.gif";
```

Puedes establecer un atributo en un elemento, al establecer la propiedad con el mismo nombre. Por ejemplo, para cambiar el src de una :



```
imgEl.setAttribute("src", "http://www.dogs.com/dog.gif");
```

Si quieres **quitar un atributo**, debes hacerlo con **removeAttribute**, como para quitar el atributo disabled de un botón, habilitándolo:

Modificar Estilos

- Puedes cambiar estilos igual que como cambias los atributos, al acceder a la propiedad style del elemento, y establecer la propiedad correspondiente. Por ejemplo, para cambiar el color:
- Recuerda usar notación "**camelCase**" en los nombres de las propiedades de CSS de varias palabras, puesto que los guiones no son nombres de propiedad válidos en JS:

```
head.style.backgroundColor = "valor";
```

Crear Elementos desde Cero

- Hay todo un conjunto de funciones que puedes utilizar para crear elementos completamente nuevos y agregarlos a la página.
- Para crear un nuevo elemento, utiliza el acertadamente llamado:
createElement:
var imgEl = document.createElement("img");
- Para anexarlo a la página, llama **appendChild** en el elemento padre de destino:
document.body.appendChild(imgEl);
- Del mismo modo, también puedes utilizar **insertBefore**, **replaceChild**, **removeChild** e **insertAdjacentHTML**.

DIPLOMADO

Front End



UVM

TEMA 5

Funciones

¿Por qué se requieren Funciones?

Cuando se desarrolla una aplicación compleja, **es muy habitual utilizar una y otra vez las mismas instrucciones.**

Ejemplo

Un script para una **tienda de comercio electrónico**, por ejemplo, tiene que calcular el precio total de los productos varias veces, para añadir los impuestos y los gastos de envío.

Cuando una serie de instrucciones se repiten una y otra vez, se complica demasiado el código fuente de la aplicación, ya que:

El código de la aplicación es mucho más largo porque muchas instrucciones están repetidas.

Si se quiere modificar alguna de las instrucciones repetidas, se deben hacer tantas modificaciones como veces se haya escrito esa instrucción, lo que se convierte en un trabajo muy pesado y muy propenso a cometer errores.

Las funciones son la solución a todos estos problemas, tanto en JavaScript como en el resto de lenguajes de programación..

Funciones

- El nombre de la función se utiliza para llamar a esa función cuando sea necesario. Se les asigna un nombre único para poder utilizarlas dentro del código. Después del nombre de la función, se incluyen dos paréntesis. Por último, los símbolos { y } se utilizan para encerrar todas las instrucciones que pertenecen a la función.



```
function suma_y_muestra() {  
    resultado = numero1 + numero2;  
    alert("El resultado es " + resultado);  
}  
var resultado;  
var numero1 = 3;  
var numero2 = 5;  
suma_y_muestra();  
numero1 = 10;  
numero2 = 7;  
suma_y_muestra();  
numero1 = 5;  
numero2 = 8;  
suma_y_muestra();  
...
```

Funciones

Se requiere una función para código que se repite:

```
● ● ●

var resultado;
var numero1 = 3;
var numero2 = 5;
// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);
numero1 = 10;
numero2 = 7;
// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);
numero1 = 5;
numero2 = 8;
// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);
...
...
```

Sintaxis - Funciones

Se requiere utilizar la palabra reservada **function**.

```
//Recursividad
function factorial(n) {
    if (n === 0) {
        return 1;
    }
    return n * factorial(n - 1);
}
//Función anónima
var displayClosure = function() {
    var count = 0;
    return function () {
        return ++count;
    };
}
var inc = displayClosure();
inc(); // devuelve 1
inc(); // devuelve 2
inc(); // devuelve 3
```

Funciones y Parámetros

Las funciones más **sencillas no necesitan ninguna información para producir sus resultados**. Sin embargo, la mayoría de funciones de las aplicaciones reales deben acceder al valor de algunas variables para producir sus resultados.

Las variables que necesitan las funciones se llaman argumentos. Antes de que pueda utilizarlos, la función debe indicar cuántos argumentos necesita y cuál es el nombre de cada argumento. Además, al **invocar la función, se deben incluir los valores que se le van a pasar a la función**. Los argumentos se indican dentro de los paréntesis que van detrás del nombre de la función y se separan con una coma (,).



```
//Definición de la función
function suma_y_muestra(primerNumero,segundoNumero){
    var resultado = primerNumero + segundoNumero;
    alert("El resultado es "+resultado);
}
//Declaración de las variables
var numero1=3;
var numero2=5;
//Llamada a la función
suma_y_muestra(numero1,numero2);
```

Funciones



```
//Definición de la función
function calculaPrecioTotal(precio){
var impuestos = 1.16;
var gastosEnvio = 10;
var precioTotal = (precio*impuestos) + gastosEnvio;
console.log(precioTotal);
}

calculaPrecioTotal(23.34);
```

DIPLOMADO

Front End

UVM

EJERCICIO 2

Crear una función de operaciones
Crear un función de bienvenida – con parámetro
Calculadora

DIPLOMADO

Front End

UVM

TEMA 6

Eventos

Eventos

- En Javascript existe un concepto llamado **evento**, que no es más que una notificación de que alguna característica interesante acaba de ocurrir. Dichas características pueden ser de múltiples tipos y muy variadas: desde un **click** de ratón hasta la **pulsación de una tecla de teclado o la reproducción de un audio o video, entre muchas otras.**

Eventos - Tipos

- **Tipos de eventos.**

En este modelo, **cada elemento o etiqueta HTML define su propia lista de posibles eventos que se le pueden asignar**. Un mismo tipo de evento (por ejemplo, dar clic en el botón izquierdo del ratón) puede estar definido para varios elementos HTML diferentes y un mismo elemento HTML puede tener asociados varios eventos diferentes.

El nombre de cada evento se construye mediante **el prefijo on**, seguido del nombre en inglés de la acción asociada al evento. Así, el evento de dar clic a un elemento con el ratón se denomina **onclick** y el evento asociado a la acción de mover el ratón se denomina **onmousemove**.

Eventos - Tipos

Evento	Descripción
onblur	Un elemento pierde el foco
onchange	Un elemento ha sido modificado
onclick	Pulsar y soltar el ratón
ondblclick	Pulsar dos veces seguidas con el ratón
onfocus	Un elemento obtiene el foco
onkeydown	Pulsar una tecla y no soltarla
onkeypress	Pulsar una tecla
onkeyup	Soltar una tecla pulsada
onload	Página cargada completamente
onmousedown	Pulsar un botón del ratón y no soltarlo
onmousemove	Mover el ratón

Evento	Descripción
onmouseout	El ratón "sale" del elemento
onmouseover	El ratón "entra" en el elemento
onmouseup	Soltar el botón del ratón
onreset	Inicializar el formulario
onresize	Modificar el tamaño de la ventana
onselect	Seleccionar un texto
onsubmit	Enviar el formulario
onunload	Se abandona la página, por ejemplo al cerrar el navegador

Eventos

- En el caso de la Web, **los eventos se desencadenan dentro de la ventana del navegador** y tienden a estar unidos a un elemento específico que reside en ella — **podría ser un solo elemento, un conjunto de elementos**—, el documento HTML cargado en la pestaña actual o toda la ventana del navegador. Hay muchos tipos diferentes de eventos que pueden ocurrir, por ejemplo:
 - **El usuario hace clic** con el mouse sobre un elemento determinado o coloca el cursor sobre un elemento determinado.
 - El usuario **presiona una tecla en el teclado**.
 - El usuario **cambia el tamaño o cierra la ventana** del navegador.
 - **Una página web termina de cargar**.
 - Un formulario se **envía**.
 - Un video se reproduce, pausa o finaliza la reproducción.
 - Un error ocurre.

Eventos – Más Utilizados

- Los eventos **son acciones u ocurrencias** que suceden en el sistema que está programando y que el sistema le informa para que pueda responder de alguna manera si lo desea.
 - **Por ejemplo, si el usuario hace clic en un botón en una página web, es posible que desee responder a esa acción mostrando un cuadro de información.**
- Los eventos más utilizados en las aplicaciones web tradicionales son **onload** para esperar a que se cargue la página por completo, los eventos **onclick**, **onmouseover**, **onmouseout** para controlar el ratón y **onsubmit** para controlar el envío de los formularios.

Ejemplo

- En el siguiente ejemplo, tenemos un solo <button>, que cuando se presiona, hará que el fondo cambie a un color aleatorio:

```
<button>Cambiar color</button>
```



```
function bgChange() {  
    var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) + ')';  
    document.body.style.backgroundColor = rndCol;  
}  
  
<button onclick="bgChange()">Press me</button>
```

Eventos - Ejemplo

- En este código, **almacenamos una referencia al botón** dentro de una variable llamada btn, usando document.querySelector ().
- También **definimos una función** que devuelve un número aleatorio.
- La tercera parte del código es el **controlador de eventos**.

```
const btn = document.querySelector('button');

function random(number) {
    return Math.floor(Math.random() * (number+1));
}

btn.onclick = function() {
    const rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) + ')';
    document.body.style.backgroundColor = rndCol;
}
```



DIPLOMADO

Front End

```
9 JSlider slider1
10
11 public void display()
12 {
13     f1=new JFrame("Kamleshut");
14     f2=new JFrame("Change");
15     p1=new JPanel();
16     l1=new JLabel("Ch");
17     slider1=new JSli
18     f1.getContentPane();
19     p1.add(l1);
20     p1.add(sli
21
22     f1.set
23     f1
24
25 }
```

UVM

TEMA 7

Listener

Eventos Listener

- Los **Listeners** se encuentran entre las estructuras de JavaScript más utilizadas en el diseño web.
- Nos permiten agregar **funcionalidad interactiva** a los elementos HTML al “escuchar” diferentes eventos que tienen lugar en la página, como cuando el usuario hace clic en un botón, presiona una tecla o cuando se carga un elemento. Cuando ocurre un evento, podemos ejecutar algo.

Sintaxis - addEventListener



```
element.addEventListener(event, function, useCapture);
```

Definición en línea//

```
element.addEventListener("click", function(){ alert("Hello World!"); });
```

Función separada//

```
element.addEventListener("click", myFunction);
```

```
function myFunction() { alert ("Hello World!");}
```

Ejemplo

Declaraciones de Event Listener

```
<button id="btn">Presiona</button>

const button = document.getElementById('btn')

button.addEventListener('click',function(event){
    alert('El evento se activó através del clic')
});

button.addEventListener('click',{
    handleEvent: function(event){
        alert('El evento se activó mediante la propiedad handleEvent');
    }
});
```

Sintaxis - removeEventListener

El método **removeEventListener()** elimina un controlador de eventos que se adjuntó con el método `addEventListener ()`.

```
// Agregar evento al elemento <div>
document.getElementById("myDIV").addEventListener("mousemove", myFunction);

// Eliminar el evento al elemento <div>
document.getElementById("myDIV").removeEventListener("mousemove", myFunction);
```

DIPLOMADO

Front End

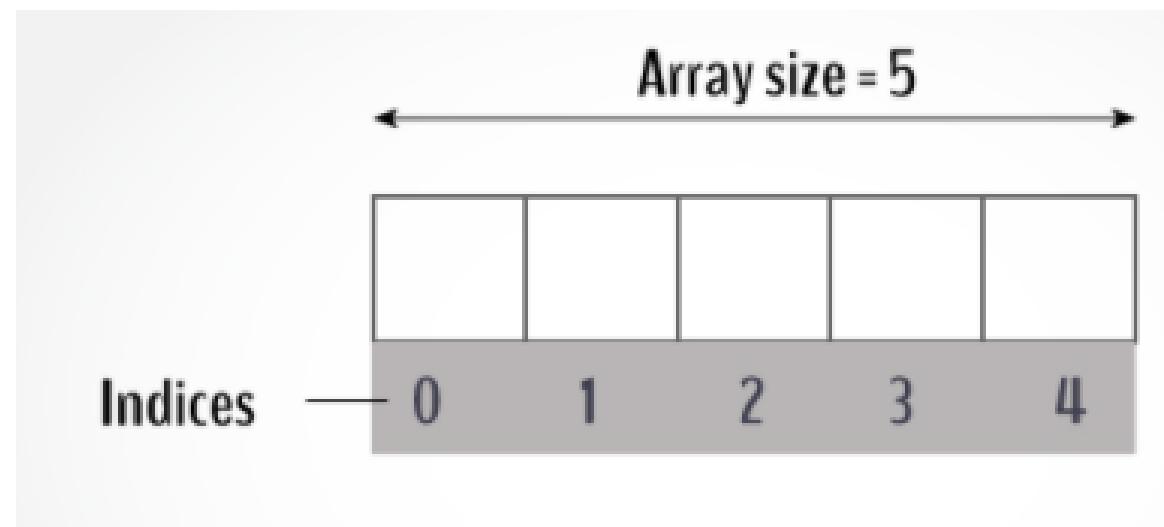
UVM

TEMA 8

Array y Objetos

Array

- Los **arrays son objetos similares a una lista** cuyo prototipo proporciona métodos para efectuar operaciones de recorrido y de mutación. Tanto la longitud como el tipo de los elementos de un array son variables.



Operaciones Habituales

Trabajar con los arrays

```
//Crear un Array
let frutas = ["Manzana", "Banana"]
console.log(frutas.length)
// 2
//Acceder a un elemento de Array mediante su índice
let primero = frutas[0]
// Manzana
let ultimo = frutas[frutas.length - 1]
// Banana
```

Operaciones Habituales

Trabajar con los arrays

```
//Recorrer un Array
frutas.forEach(function(elemento, indice, array) {
    console.log(elemento, indice);
})
// Manzana 0
// Banana 1
//Añadir un elemento al final de un Array
let nuevaLongitud = frutas.push('Naranja') // Añade "Naranja" al final
// ["Manzana", "Banana", "Naranja"]
//Eliminar el último elemento de un Array
let ultimo = frutas.pop() // Elimina "Naranja" del final
// ["Manzana", "Banana"]
//Tamaño
frutas[5] = 'fresa'
console.log(frutas[5])          // 'fresa'
console.log(Object.keys(frutas)) // ['0', '1', '2', '5']
console.log(frutas.length)      // 6
```

Operaciones Habituales

Trabajar con los arrays



```
//Añadir un elemento al principio de un Array
let nuevaLongitud = frutas.unshift('Fresa') // Añade "Fresa" al inicio
// ["Fresa", "Manzana", "Banana"]
//Eliminar el primer elemento de un Array
let primero = frutas.shift() // Elimina "Fresa" del inicio
// ["Manzana", "Banana"]
//Encontrar el índice de un elemento del Array
frutas.push('Fresa')
// ["Manzana", "Banana", "Fresa"]
let pos = frutas.indexOf('Banana') // (pos) es la posición para abbreviar
// 1
```

Accesos a Elementos de un Array

- **Los índices de los arrays de JavaScript comienzan en cero**, es decir, el índice del primer elemento de un array es 0, y el del último elemento es igual al valor de la propiedad **length del array restándole 1**.
- Si se utiliza un número de índice no válido, se obtendrá undefined.
 - `let arr = ['este es el primer elemento', 'este es el segundo elemento', 'este es el último elemento']`
 - `console.log(arr[0])` // escribe en consola 'este es el primer elemento'
 - `console.log(arr[1])` // escribe en consola 'este es el segundo elemento'
 - `console.log(arr[arr.length - 1])` // escribe en consola 'este es el último elemento'

DIPLOMADO

Front End

TEMA 9

Objetos

UVM

Objetos

- En JavaScript, **un objeto es una entidad independiente con propiedades y tipos. Por ejemplo** un coche, un coche es un objeto con propiedades. Tiene un **color**, un **diseño**, un **peso**, un **material** del que está hecho, etc.
- **Un objeto de JavaScript tiene propiedades asociadas a él.** Las propiedades de un objeto definen las características del objeto.
- Accedes a las propiedades de un objeto con una simple notación de puntos:
 - `objectName.propertyName`

Objetos

- Crear un objeto es tan simple como esto: En javascript podemos declarar un objeto así:

```
var myCar = new Object();  
  
myCar.make = 'Ford';  
  
myCar.model = 'Mustang';  
  
myCar.year = 2021;
```

El ejemplo **anterior también se podría escribir usando un objeto**, que es una lista delimitada por comas, pares de nombres de propiedad y valores asociados de un objeto, encerrados entre llaves ({}):

```
var myCar = {  
  
  make: 'Ford',  
  
  model: 'Mustang',  
  
  year: 2021  
};
```

Objetos

Trabajar con los objetos



```
//Objeto  
a= {clave1:"valor", clave2:22};  
//En a tenemos a.clave1 y a.clave2 con valores "valor" y 22 respectivamente
```

Funciones en los Objetos

```
let yo = {  
    nombre: 'Rodolfo'  
    edad: 29,  
    hablar: function(){  
        console.log(this.nombre);  
    }  
};  
  
//Invocar a la función  
yo.hablar();
```

```
let decirNombre = function(obj) {  
    obj.hablar = function() {  
        console.log(this.nombre);  
    };  
};  
const Mateo = {  
    nombre: 'Mateo',  
    edad: 22  
};  
const juan = {  
    nombre: 'Juan',  
    edad: 25  
};  
decirNombre(juan);  
decirNombre(Mateo);  
juan.hablar(); // Juan  
Mateo.hablar(); // Mateo
```

Objetos `this`

- JavaScript tiene una palabra clave especial, **this**, que puedes usar dentro de un método para referirte al objeto actual.
- ***this es una referencia que se crea cuando una función es invocada***, no declarada. El valor de esa referencia depende al 100% del lugar en la que esa invocación se realice, llamado ***call-site***.

Ejemplo

```
●●●  
  
const Maestro = {  
    nombre: "Rodolfo",  
    edad: 27,  
    ocupacion: "Maestro"  
}  
  
const Alumno = {  
    nombre: "Isai",  
    edad: 21,  
    ocupacion: "Alumno"  
}  
  
function saludar(){  
    console.log('Hola, mi nombre es:' + this.nombre);  
}  
  
//Agregar la función saludar a los objetos  
Maestro.saludar = saludar;  
Alumno.saludar = saludar;  
Maestro.saludar(); //Hola, mi nombre es Rodolfo  
Alumno.saludar(); //Hola, mi nombre es Isai
```

Por ejemplo, supongamos que tienes 2 objetos, **Maestro y Alumno**.

Cada objeto tiene su propio **nombre,edad y ocupación**.

En la función **saludar()**, observa que hay **this.name**.

Cuando se agregan a los 2 objetos, se pueden llamar y devuelve el 'Hola, mi nombre es' y luego agrega el valor name de ese objeto específico.

Objeto this

- **this** se refiere al objeto en el que se encuentra. Puedes crear una nueva función llamada Edad() que registra una oración que dice cuántos años tiene la persona.



```
function edad() {  
    console.log('Tengo ' + this.edad + ' años.')  
}  
Maestro.edad = edad;  
Maestro.edad() // Tengo 27 años.
```

DIPLOMADO

Front End

UVM

EJERCICIO 3

Crear un formulario con eventos de envío
Evento de carga de la información
Cambiar elementos en el DOM
Listener

DIPLOMADO

Front End

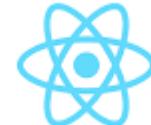
TEMA 10

Librerías



Librerías de JavaScript

- Javascript posee un amplio repertorio de librerías para diversas funcionalidades gracias a su amplia comunidad de desarrollo. Algunas de las librerías o frameworks más utilizadas en el mercado y sus características principales:
- **jQuery**: La librería jQuery es una de las librerías más conocidas para programar en javascript, y cuenta con una gran comunidad de usuarios y desarrolladores.
- **React**: es una librería **creada por Facebook** en 2011 y planeada explícitamente para construir interfaces de usuario dinámicas, rápidas e interactivas.
- **AngularJS**: es un **framework** desarrollado **por Google** en 2009 y de código abierto.
- **Vue.js**: Con total seguridad, **Vue** es el framework de Javascript que más ha crecido en popularidad.
- **Node js**: es la librería *opensource* más utilizada para el desarrollo backend con millones de desarrolladores en todo el mundo.





DIPLOMADO

Front End

```
2 import { BrowserModule } from '@angular/  
2 import { NgModule } from '@angular/  
  RouterModule } from '@angular/  
HttpModule } from '@angular/  
AppComponent  
ProjectsC  
routes  
{ pat  
{
```

UVM

TEMA 11

Angular

¿Qué es Angular JS?



- Es un framework **SPA** para javascript que nos permite realizar aplicaciones web de una sola página implementando el patrón MVC.
- Al usar un patrón MVVM (model view view-model) **separamos la lógica de la de diseño** pero mantenemos ambas partes conectadas (data binding). De manera que la capa visual no sabe lo que está pasando en la capa lógica pero manteniendo control sobre el DOM (el cuerpo de la web) y actualizar su contenido como queramos.

¿Qué ventajas ofrece Angular JS?

- **Ahorras tiempo.** Cuando empiezas a pensar en cómo crear una aplicación web, tienes que tomar una serie de decisiones; por ejemplo, **la arquitectura de la aplicación, su organización**, etc. Angular ya lo hace por ti, por lo que puedes centrarte en cosas más importantes, como las funcionalidades de tu aplicación web.
- Usa lenguaje **TypeScript**. Las aplicaciones son fáciles de mantener. Al usar TypeScript, cualquier cambio que deba hacerse en la aplicación podrá llevarse a cabo rápidamente y sin errores.
- Hace posible intercambiar o **añadir programadores en los proyectos**. Esto es una gran ventaja a la hora de trabajar en equipo o de retomar proyectos inacabados.
- Utiliza **componentes web**. Un componente web es una porción de código que puede reutilizarse en otros proyectos hechos con Angular.
- **El futuro es estable**, por lo general, hay muchos menos cambios que con JavaScript.

AngularJS Overview

- **Modelo**

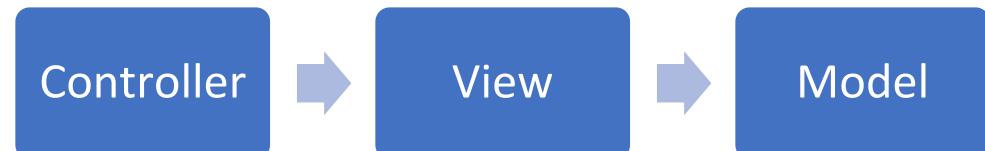
- El modelo representa la **estructura de datos** de una aplicación de software. Es un error común pensar que el modelo es la base de datos que está detrás de la aplicación, sin embargo, es mucho mejor ver el modelo como el cuerpo de código que representan los datos.

- **Vista**

- La vista es el **cuerpo de código que representa la interfaz del usuario**, es decir, todo aquello que el usuario puede ver en la pantalla e interactuar. Una aplicación normalmente tiene múltiples vistas, cada una representa alguna parte del modelo.

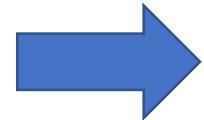
- **Controlador**

- El controlador se puede ver como **el intermediario entre la vista y el modelo**. El modelo y la vista nunca se relacionan directamente.



Ejemplo

```
<!DOCTYPE html>
<html ng-app>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/
      angularjs/1.8.2/angular.min.js"></script>
    <link
      rel="stylesheet"
      href="https://netdna.bootstrapcdn.com/twitter-bootstrap/
        2.0.4/css/bootstrap-combined.min.css"
    />
  </head>
  <body>
    <div>
      <label>Name:</label>
      <input type="text" ng-model="Nombre"
        placeholder="Ingresa tu nombre" />
      <hr />
      <h1>Hola: {{Nombre}}!</h1>
    </div>
  </body>
</html>
```



Name:

Rodolfo

Hola: ¡Rodolfo!

DIPLOMADO

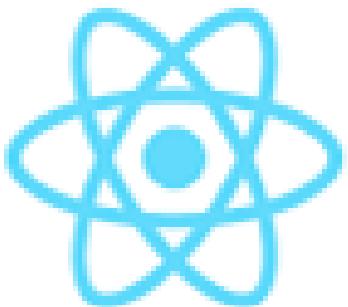
Front End

React

TEMA 12

UVM

¿Qué es React?



- React es una librería de Javascript para la generación de interfaces visuales.
- React es un proyecto desarrollado por la empresa Facebook y tiene sus inicios en el año 2013
- React nos facilita la implementación de páginas **SPA (Single-Page Application)** o aplicaciones de página única en un sitio web, este tipo de aplicaciones tiene por objetivo dar al visitante una experiencia más parecida a las aplicaciones de escritorio.
- Utiliza la metodología de programación orientada a componentes en forma similar a sus competidores Vue y Angular.

¿Por qué React?

- React aporta una serie de claras ventajas frente a la forma clásica de realizar una web, **sus facilidades para el desarrollo unido al rendimiento**, la flexibilidad y organización del código la convierten en una de las mejores opciones.
- Una de las principales razones para que esto sea posible es el uso del **DOM virtual**, React es capaz de generar el **DOM de forma dinámica**, hace los cambios en una copia en **memoria** y después la compara con la versión actual del DOM, **de esta forma evita renderizar toda la página cada vez que haya cambios**, simplemente se aplica dicho cambio al componente que haya sido actualizado, simple y rápido. **Esto propicia una mejor experiencia de usuario, además de un rendimiento y una fluidez impresionante.**

Ventajas

- React intenta ayudar a los desarrolladores a construir aplicaciones que usan **datos que cambian todo el tiempo**.
- Su objetivo es ser **sencillo, declarativo y fácil de combinar**.
- React sólo maneja la interfaz de usuario en una aplicación; React es la Vista en un contexto en el que se use el patrón **MVC** (Modelo-Vista-Controlador) o MVVM (Modelo-vista-modelo de vista).

Características

- **Declarativo.**
 - React te ayuda a crear interfaces de usuario interactivas de forma sencilla. Diseña vistas simples para cada estado en tu aplicación, y React se encargará de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien.
- **Basado en componentes.**
 - Crea componentes encapsulados que manejen su propio estado, y conviértelos en interfaces de usuario complejas.
 - Ya que la lógica de los componentes está escrita en JavaScript y no en plantillas, puedes pasar datos de forma sencilla a través de tu aplicación y mantener el estado fuera del DOM.
- **Aprende una vez, escríbelo donde sea.**
 - En React no dejamos fuera al resto de tus herramientas tecnológicas, así que podrás desarrollar nuevas características sin necesidad de volver a escribir el código existente.
 - React puede también **renderizar desde el servidor usando Node**, así como potencializar aplicaciones móviles usando **React Native**.

Características

- **Virtual DOM.**
 - React mantiene un **virtual DOM** propio, en lugar de confiar solamente en el DOM del navegador. Esto deja a la biblioteca determinar qué partes del DOM han cambiado comparando contenidos entre la versión nueva y la almacenada en el virtual DOM, y utilizando el resultado para determinar **cómo actualizar eficientemente el DOM..**
- **Las propiedades.**
 - Las propiedades (también conocidas como 'props') pueden definirse como los atributos de configuración para dicho componente. Éstas son recibidas desde un nivel superior, normalmente al realizar la instancia del componente y por definición son inmutables.
- **El Estado.**
 - El estado de un componente se define como una representación del mismo en un momento concreto, es decir, una instantánea del propio componente. Existen dos tipos de componentes con y sin estado, denominados statefull y stateless.
- **Ciclos de vida.**
 - El ciclo de vida es una serie de estados por los cuales pasan los componentes statefull a lo largo de su existencia. Se pueden clasificar en tres etapas de montaje o **inicialización, actualización y destrucción.**

DOM Virtual en React

- En vez de centrarse en el “data-binding”, como lo hace la mayoría de las librerías alternativas, React enfatiza la necesidad de actualizar las vistas cuando el desarrollador lo requiera, y lo consigue mediante el **DOM Virtual**.
- El DOM Virtual es una de las características principales de React. Este concepto hace referencia a **una representación del DOM pero en memoria**, usado para **aumentar el rendimiento de los componentes y aplicaciones con las que interactúa el usuario** directamente. Así ofrece una **mejor experiencia de usuario** y una mejor fluidez.
- Lo que hace el DOM Virtual es elaborar selectivamente sub-árboles de los nodos sobre la base de variaciones de estados, logrando la **menor manipulación del DOM posible**, para mantener los **componente actualizados y los datos estructurados**.

React y su Arquitectura

- El elemento más importante de React es **el componente**, que es, en esencia, una pieza de la interfaz de usuario. Como norma general, al diseñar una aplicación con React, lo que estamos haciendo es crear **componentes independientes y reusables para**, poco a poco, **crear interfaces de usuario más complejas**.

```
● ● ●

import React from "react";

class ComponentExample extends React.Component {
  state = {
    data: ""
  }

  render () {
    return (
      Hello World!
    )
  }
}

export default ComponentExample;
```

Ejemplo

React utiliza una sintaxis parecida a HTML, llamada **JSX**. No es necesaria para utilizar React, sin embargo, hace el código más legible, y escribirlo es una experiencia similar a HTML.

LIVE JSX EDITOR JSX? RESULT

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hola {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="¡Rodolfo!" />,
  document.getElementById('hello-example')
);
```

Hola ¡Rodolfo!

LIVE JSX EDITOR JSX? RESULT

```
class HelloMessage extends React.Component {
  render() {
    return React.createElement(
      "div",
      null,
      "Hola ",
      this.props.name
    );
  }
}

ReactDOM.render(React.createElement(HelloMessage, { name:
"\xA1Rodolfo!" }), document.getElementById('hello-example'));
```

Hola ¡Rodolfo!