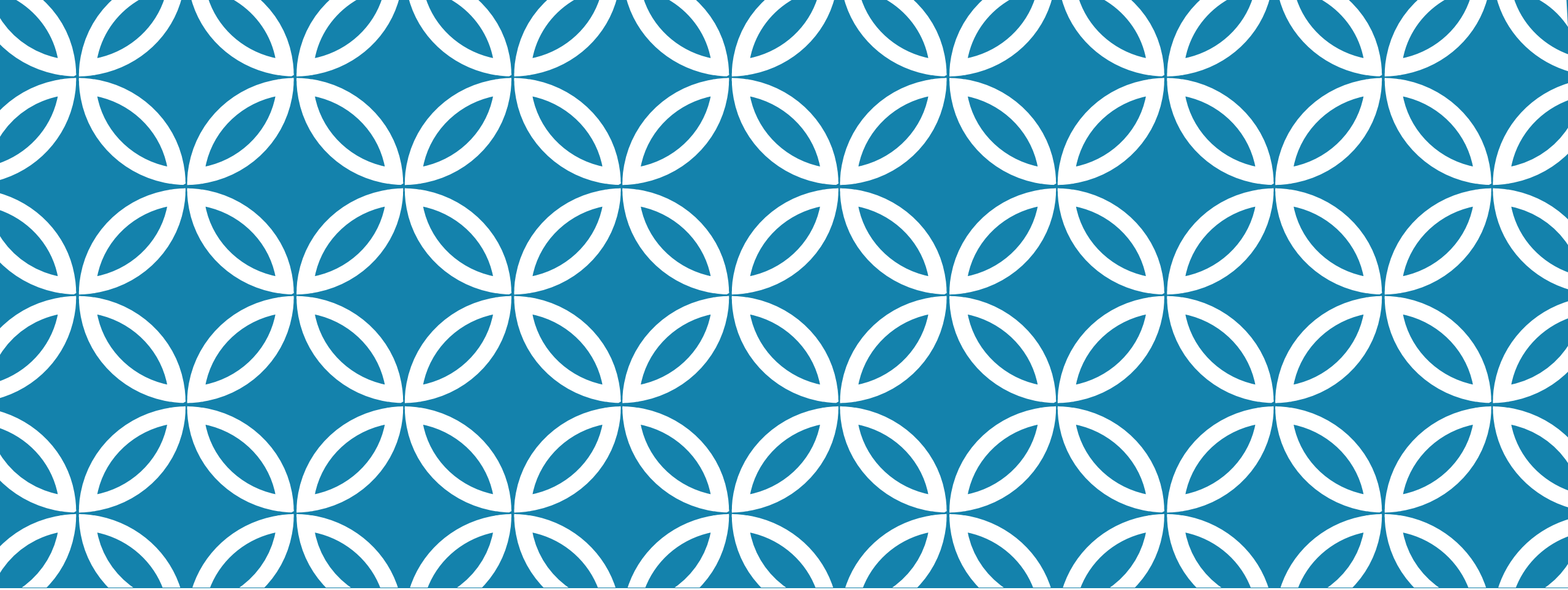


JAVASCRIPT

Lenguaje de programación.

AGENDA

- JavaScript
- Fundamentos JavaScript
- Ciclos
- Funciones en bloque y en expresión
- Object
- Array
- callback
- Promesas
- async / await
- Hoisting
- DOM manipulation

Gustavo Adolfo Garcia Blanco

JAVASCRIPT

- **JavaScript** (**JS**) es un lenguaje de programación ligero, interpretado, o compilado justo a tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js, Apache CouchDB y Adobe Acrobat. JavaScript es un lenguaje de programación basado en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional).

Gustavo Adolfo Garcia Blanco

FUNDAMENTOS JAVASCRIPT

- Variables

  `var, let, const`

- Tipos de datos primitivos

  `Undefined, Boolean, Number, String, BigInit, Symbol`

Gustavo Adolfo Garcia Blanco

- Tipos de datos no primitivos

  `Null, Object, Function`

- Principales métodos de conversión de tipos

  `Number(), String(), Boolean()`

FUNDAMENTOS JAVASCRIPT

- Operadores

  AND (&), OR (|), XOR (^), NOT (~), LEFT SHIFT (<<), RIGHT SHIFT (>>), ZERO-FILL RIGHT SHIFT (>>>)

- Comparaciones

  a > b, a < b, a >= b, a <= b, a == b, a === b, a != b

Gustavo Adolfo Garcia Blanco

- Condicionales

  if, else if, else

FUNDAMENTOS JAVASCRIPT

- Switch

```
switch(yourValueHere){
case 'optionOne' :
      //your code here
break;
case 'optionTwo':
      //your code here
break;
case 'optionThree':
      //your code here
break;
default:
      //your fallback code here
}
```

Gustavo Adolfo Garcia Blanco

CICLOS

- for

```
for (statement1; statement2; statement3) {
  // bloque de codigo
 }

for (let variable = 0; variable < 5; variable++) {
  console.log(variable);
 }
```

Gustavo Adolfo Garcia Blanco

- for In

```
for (key in object) {
  // bloque de codigo
 }

const persona = {nombre:"Gustavo”, apellido:”Garcia"}
for (let propiedad in persona) {
  console.log(propiedad);
 }
```

CICLOS

- for of

```
for (variable of iterable) {
  // bloque de codigo
 }
```

```
const arregloMarcas = [”BMW”,”Volvo”,”Mazda”]
for (let marca of arregloMarcas) {
  console.log(marca);
 }
```

Gustavo Adolfo Garcia Blanco

- while

```
while (condition) {
  // bloque de codigo
 }
```

```
 while (variable < 10) {
  console.log(variable);
  variable++;
 }
```

FUNCIONES EN BLOQUE Y EN E XPRESIÓN

- función en bloque

```
function functionname(parameters) {
  // bloque de codigo
 }

function suma(sumando1, sumando2) {
  return sumando1 + sumando2;
}
console.log(suma(1, 5));
```

Gustavo Adolfo Garcia Blanco

- expresión de función

```
var functionname = function(parameters) {
  // bloque de codigo
 }

 var suma = function(sumando1, sumando2) {
  return sumando1 + sumando2;
 }
 console.log(suma(1, 5));
```

OBJECT

- En JavaScript, los objetos son los reyes. Si entiendes los objetos, entiendes JavaScript.
- En JavaScript, casi "todo" es un objeto.
- Un objeto JavaScript es una colección de valores con nombre
- `{"propiedad": valor, …, "propiedadN": valorN }`

Gustavo Adolfo Garcia Blanco

OBJECT

- Los booleanos pueden ser objetos (si se definen con la `new` palabra clave)
- Los números pueden ser objetos (si se definen con la `new` palabra clave)
- Las cadenas pueden ser objetos (si se definen con la `new` palabra clave)
- Las fechas son siempre objetos.
- Las matemáticas son siempre objetos.
- Las expresiones regulares son siempre objetos.
- Los arreglos son siempre objetos.
- Las funciones son siempre objetos.
- Los objetos son siempre objetos.
- Todos los valores de JavaScript, excepto los primitivos, son objetos.

Gustavo Adolfo Garcia Blanco

ARRAY

- Una `Array` es una variable especial, que puede contener más de un valor.

Gustavo Adolfo Garcia Blanco

- El uso de un literal de matriz es la forma más fácil de crear una `Array` de JavaScript:

```
var arrayname = [item1, item2, ...];
```

CALLBACKS

- Un `callback` es una función que se pasa como argumento a otra función.
- Esta técnica permite que una función llame a otra función.
- Un `callback` puede ejecutarse después de que otra función haya finalizado.

Gustavo Adolfo Garcia Blanco

CALLBACKS

```
function saludar(nombre) {
  alert('Hola ' + nombre);
}

function procesarEntradaUsuario(callback) {
  var nombre = prompt('Por favor ingresa tu nombre.’);
  callback(nombre);
}

procesarEntradaUsuario(saludar);
```

Gustavo Adolfo Garcia Blanco

PROMESAS

- Una `Promise` es un objeto que representa la terminación o el fracaso de una operación asíncrona.

Gustavo Adolfo Garcia Blanco

- Esencialmente, una promesa es un objeto devuelto al cuál se adjuntan funciones `callback`, en lugar de pasar `callbacks` a una función.

PROMESAS

A diferencia de las funciones `callback` pasadas al "viejo estilo", una promesa viene con algunas garantías:

- Las funciones `callback` nunca serán llamadas antes de la terminación de la ejecución actual del bucle de eventos de JavaScript.

Gustavo Adolfo Garcia Blanco

- Las funciones `callback` añadidas con `then()` incluso después del éxito o fracaso de la operación asíncrona serán llamadas.

- Múltiples funciones `callback` pueden ser añadidas llamando a `then()` varias veces. Cada una de ellas es ejecutada una seguida de la otra, en el orden en el que fueron insertadas.

ASYNC / AWAIT

- "`async` y `await` hacen que las promesas sean más fáciles de escribir".

Gustavo Adolfo Garcia Blanco

- `async` hace que una función devuelva una promesa.

- `await` hace que una función espere una Promesa

HOISTING

- En JavaScript, una variable se puede declarar después de que se haya utilizado. En otras palabras; una variable se puede utilizar antes de que se haya declarado. Para entender esto, hay que entender el término `hoisting`.

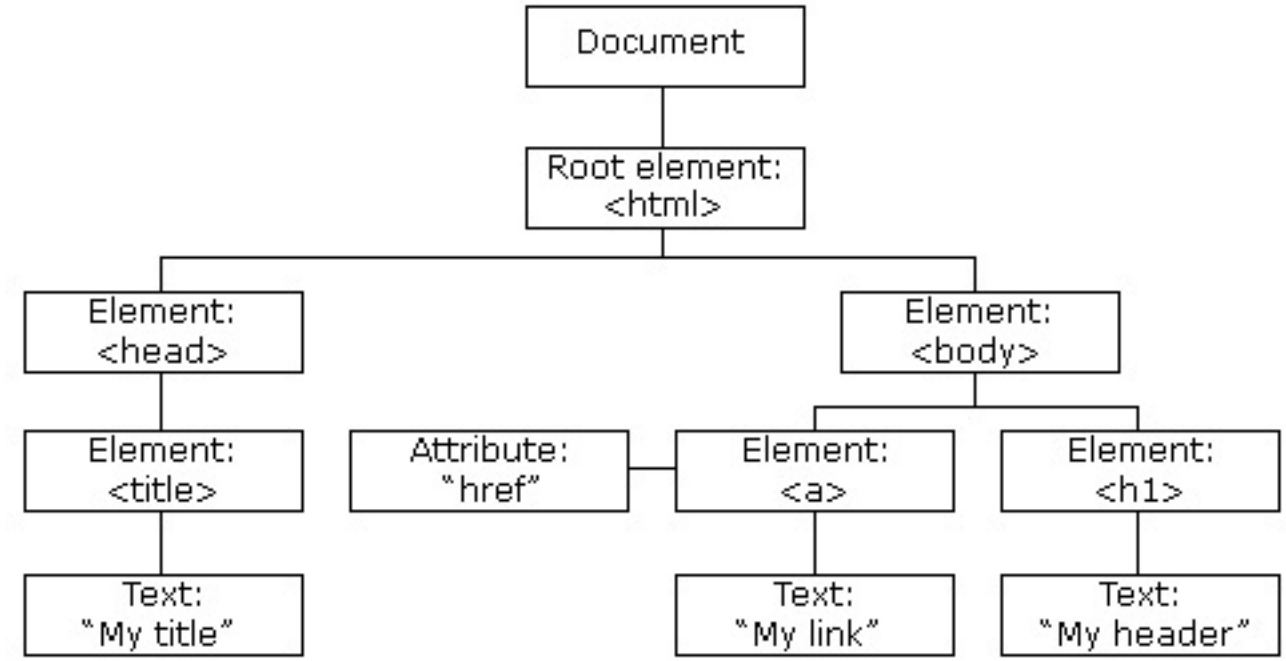
Gustavo Adolfo Garcia Blanco

- `hoisting` es el comportamiento predeterminado de JavaScript de mover todas las declaraciones a la parte superior del alcance actual (a la parte superior del script actual o de la función actual).

DOM MANIPULATION

- Con HTML **DOM**, JavaScript puede acceder y cambiar todos los elementos de un documento HTML.
- Cuando se carga una página web, el navegador crea un **Document Object Model** de la página. El modelo HTML **DOM** se construye como un árbol de Objetos.

Gustavo Adolfo Garcia Blanco



DOM MANIPULATION

Con el modelo de objetos, JavaScript obtiene todo el poder que necesita para crear HTML dinámico:

- JavaScript puede cambiar todos los elementos HTML en la página.
- JavaScript puede cambiar todos los atributos HTML en la página.
- JavaScript puede cambiar todos los estilos CSS en la página.
- JavaScript puede eliminar elementos y atributos HTML existentes.
- JavaScript puede agregar nuevos elementos y atributos HTML.
- JavaScript puede reaccionar a todos los eventos HTML existentes en la página.
- JavaScript puede crear nuevos eventos HTML en la página.

Gustavo Adolfo Garcia Blanco

DOM MANIPULATION

El HTML `DOM` es un modelo de objeto estándar y una interfaz de programación para HTML y define:

- Los elementos HTML como objetos.
- Las propiedades de todos los elementos HTML.
- Los métodos para acceder a todos los elementos HTML.
- Los eventos para todos los elementos HTML.

Gustavo Adolfo Garcia Blanco

En otras palabras: el HTML `DOM` es un estándar sobre cómo obtener, cambiar, agregar o eliminar elementos HTML.