INDICE

Contenido

[Etiquetas 2](#_Toc188397429)

[Script 2](#_Toc188397430)

[Variables 2](#_Toc188397431)

[Let: 2](#_Toc188397432)

## Etiquetas

Para que nosotros podamos escribir código de javascript tenemos que hacerlo dentro de la etiqueta:

### Script

El *elemento HTML Script* (**<script>**) se utiliza para insertar o hacer referencia a un script ejecutable dentro de un documento HTML o XHTML.

Tiene su etiqueta de apertura y cierre.

Esto va dentro de la etiqueta (<body>) en HTML y al final de ella.

#### Atributos

* [src](https://developer.mozilla.org/es/docs/Web/SVG/Element/script#src): Este atributo especifica la URI del script externo; este puede ser usado como alternativa a scripts embebidos directamente en el documento. Si el script tiene el atributo src, no debería tener código dentro de la etiqueta.

//Se coloca así (<script src=””></script>).

## Variables

Let: es una variable reservada y es usada para definir variables.

//Declara una variable local con alcance de bloque, inicializándola opcionalmente en un valor.

let nombre = “Hola Mundo”;

Nota:

* No se puede crear una variable con un número por delante.
* No se puede usar el nombre de una variable con una palabra reservada.

//Todas las variables declaradas pueden cambiar su valor en el futuro. Basta con volver a escribir el nombre de la variable y cambiar el contenido.

Ojo que JS es “case sensitive”, esto quiere decir que dependiendo del nombre y la convención que usemos en el nombre de la variable va a ser distinto.

Ej:

* let NombreCompleto; //chanchito
* let nombreCompleto; //Felipe 🡪 el más usado en JS

Osea, al momento de usarlo, cada uno es distinto del otro.

Const: Declara una constante de solo lectura con alcance de bloque.

Nota:

* Es usada para que no se pueda cambiar el contenido de una variable.
* Nos va a lanzar un error si cambiamos el contenido de una variable.

Cuándo usar “let” y “const”

* Como regla general nosotros nunca deberíamos cambiar el valor de una variable.
* Usamos let cuando queramos cambiar el valor de la variable en el código.
* Usamos const cuando no vamos a cambiar el valor de una variable en el código.

## Tipos de Lenguajes de Programación

* **Tipado estático**
  + String nombre = “hola”;
  + nombre = 42; //no va a dejar hacer esto.
* **Tipado dinámico**
  + nombre = “Hola”;
  + nombre = 24; //no lanza error

Lenguajes con tipado estático tenemos a java, y con tipado dinámico tenemos a JavaScript.

## Objetos

Son Tipo de referencia, quiere decir que guardamos la dirección en la memoria ram.

Son agrupación de datos que hacen sentido tenerlos juntos.

Ej, inicializamos así:

*//Personajes de TV*

let nombre = "Homero Simpsson";

let anime = "Animado";

let edad = 45;

let personaje = {

    nombre: "Homero Simpsson",    *// par llave-valor, atributo, propiedad.*

    anime: "Animado",

    edad: 45,       *//recomendable la coma al final de cada propiedad*

};

La fila completa -> nombre: “Tanjiro”, se lo conoce como par llave-valor, atributo, propiedad.

**Para acceder a una sola propiedad se usa el punto:**

console.log(personaje.nombre);  *//para acceder a una propiedad de un objeto se usa el punto.*

console.log(personaje['edad']); *//otra forma de acceder a una propiedad de un objeto.*

**Para modificar la propiedad de un objeto:**

*//Modificar una propiedad de un objeto*

personaje.edad = 56;

personaje['edad'] = 58;

let llave = 'edad';

personaje[llave] = 60;

**Para eliminar una propiedad de un objeto:**

*//Para eliminar una propiedad de un objeto*

delete personaje.anime;

## Array

Array = arreglo : significa colección

Los elementos de un array comienzan desde le cero.

*// Definir un array*

let animales = ['perro', 'gato', 'pez', 'loro', 'tortuga'];

*// Acceder a un elemento*

console.log(animales[0]); *// perro*

*// Modificar un elemento*

animales[0] = 'caballo';

*// Añadir un elemento*

animales[5] = 'conejo';

console.log(animales);

*// ojo cuando añadimos de más*

animales[10] = 'elefante';

Funciones

**Términos Interesantes**

**Refactorizar**

Cambiar la estructura interna, sin alterar el comportamiento externo, a esto le llamamos refactorizar.

//Hacer el código más bonito, más legible o que este tenga mejor rendimiento.

Ej:

* Cambiamos el código, pero seguimos obteniendo el mismo resultado.

**Tipos de Convenciones**

* UperCamelCase
* camelCase
* snake\_case