

# JAVASCRIPT



**JS**

# Contenido

## ★ Web Storage

- localStorage
- sessionStorage

## ★ ES6 - ECMAScript 2015

- let & const
- Template literals
- Arrow functions
- Spread operator
- Destructuring
- Default parameters

Web Storage es una herramienta que JS nos da para poder almacenar información en el navegador del usuario. Es algo como `$_COOKIE` y `$_SESSION` de PHP.

La diferencia, esta información se almacena en el cliente, NO en el servidor.



Web Storage nos provee de dos  
métodos para almacenar información:

- localStorage
- sessionStorage

# Web Storage

## .localStorage

Guarda información sin tiempo de expiración. Sirve para leer y escribir datos. Es un objeto literal en el cual podemos setear propiedades y valores.

```
localStorage.setItem("userName", "Juana"); // Setea el atributo userName  
localStorage.getItem("userName"); // Juana  
localStorage.removeItem("userName"); // Elimina el atributo userName
```

```
// Al igual que las COOKIES, localStorage es información única que se  
guarda por dominio y navegador.
```

# Web Storage

## .sessionStorage

Guarda información mientras se mantenga abierto el navegador. Sirve para leer y escribir datos. Es un objeto literal en el cual podemos setear propiedades y valores.

```
sessionStorage.setItem("userID", 32); // Setea el atributo userID  
sessionStorage.getItem("userID"); // 32  
sessionStorage.removeItem("userID"); // Elimina el atributo userID
```

```
// Al igual que las SESSION, sessionStorage se borra al cerrar la ventana  
del navegador.
```

ES6 o ECMAScript 2015 es la nueva especificación del lenguaje.

Trae consigo un montón de nuevas y poderosas funcionalidades que hacen de JS un lenguaje mucho más potente y poderoso.

# ECMAScript 2015

## let

Sirve para definir variables. A diferencia de **var**, **let** es una variable de bloque. Esto nos permite crear variable con el mismo nombre sin sobrescribir sus valores.

```
let name = "Ada Lovelace";  
if(true) {  
    let name = "Tim Berners Lee";  
    console.log(name); // "Tim Berners Lee"  
}  
console.log(name); // "Ada Lovelace"
```



# ECMAScript 2015

## const

**const** nos sirve para declarar constantes. Es decir, es una *"variable"* que **NUNCA** podrá cambiar su valor. A menos que sea un objeto literal.

```
const DNI = 23456987; // Crea una constante llamada DNI
DNI = "PAS-324567"; // ERROR: Assignment to constant variable.
const STUDENT_DATA = {
  code: "FS345618TN"
}
STUDENT_DATA.email = "adalovelace_ok@gmail.com"; // Permitido
```

# ECMAScript 2015

## Template literals

Nos permite una nueva y mejor manera de "concatenar" diferentes valores en un string. Usa comillas francesas `` e interpolación de variables.

```
let price = 950.45;  
let product = "Remera para nena";  
let message = `La ${product} cuesta ${price}`;  
console.log(message); // "La Remera para nena cuesta 950.45"
```

// Dentro de la interpolación `${}` podemos operar con cualquier funcionalidad de JS

# ECMAScript 2015

## Arrow Functions

Una nueva forma de escribir una función en la cual no es necesaria la palabra reservada ***function*** ni incluso, en ocasiones, el ***return***.

```
let suma = (n1, n2) => n1 + n2;  
let sayHello = () => "Hello world!";  
let contactForm = document.querySelector(".contact-form");  
contactForm.addEventListener("submit", e => e.preventDefault())
```

// Para usar como callback es una gran herramienta

# ECMAScript 2015

## Spread operator (...)

Nos permite "esparcir" datos dentro de un Array u Objeto de manera sencilla. Podemos copiar información de un lugar y trasladarla a otro fácilmente.

```
let arrayOne = ["Ada", "Grace"];  
let arrayTwo = ["Tim", "Vin"];  
arrayOne = [...arrayOne, ...arrayTwo]; // Ada, Grace, Tim, Vin  
let arrayThree = [...arrayOne, "Brendan"]; // Ada, Grace, Tim, Vin, Brendan  
  
// Funciona de manera similar para los objetos literales
```

# ECMAScript 2015

## Destructuring

Es una forma sencilla de extraer datos de un Array u Objeto y guardarlos en una variable.

```
let greatPeople = ["Ada", "Grace", "Tim", "Vin", "Brendan"];  
let [woman1, woman2] = greatPeople;  
console.log(woman1); // Ada  
console.log(woman2); // Grace
```

// Funciona de manera similar para los objetos literales, teniendo en cuenta las variables a extraer deben coincidir con las propiedades del objeto.

# ECMAScript 2015

## Default parameters

Una manera sencilla de definir valores para los parámetros que se pasan a una función.

```
let sayHello = (name = "Stranger") => `Hello ${name}`;  
sayHello("Ada"); // Hello Ada  
sayHello(); // Hello Stranger
```

// Una herramienta muy funcional, pues dentro de la función, podemos implementar toda la lógica deseada.

**¡A practicar!**

Practica Integradora



```
// To Do  
console.log("Practice Time");
```