

JavaScript

The JavaScript logo, consisting of the letters 'JS' in a bold, black, sans-serif font, centered within a solid yellow square.

JS

Objetivos

- Diferenciar las características del lenguaje JavaScript
- Comprender el uso de funciones y clases en JavaScript

JavaScript

- Es uno de los lenguajes de programación más usados y modernos en la actualidad
- Se puede utilizar en frontend, backend y aplicaciones de escritorio, y otros.
- Soporta la programación orientada a objetos.
- ECMAScript es el estándar para JavaScript. Los cambios más importantes fueron ES5 (2009) y ES6 (2015), siendo esta la versión estable más reciente.
- No tienen ninguna relación con Java.

Similitudes con Java o C++

- Estructuras de control
 - if
 - if-else
 - switch
- Operadores
 - Aritméticos
 - Relacionales
 - Lógicos
- Estructuras de repetición
 - while
 - do-while
 - for

Diferencias con otros lenguajes

- Es interpretado, o compilado justo a tiempo.
- Se ejecuta en los navegadores.
- Es opcional el uso de punto y coma (;)
- No es necesario definir tipos de variable.

Tipos de datos y variables

Todas las variables se deben declarar con **var** o **let**.

```
var length = 16;           // Number
var lastName = "Johnson"; // String
var x = {firstName:"John", lastName:"Doe"}; // Object
```

```
let str = "Hello";
let str2 = 'Single quotes are ok too';
let phrase = `can embed another ${str}`;
```

```
// no error
let message = "hello";
message = 123456;
```

- Number
- Object
- BigInt
- String
- Boolean
- null
- undefined

Funciones

Función expresión

```
function sum(a, b) {  
  return a + b;  
}
```

```
let result = sum(1, 2);  
alert( result ); // 3
```

```
let sayHi = function() {  
  alert( "Hello" );  
};
```

Uso de var

// aquí no se puede usar carName ni carYear

```
function miFuncion() {  
  var carName = "Volvo";  
  if (true) {  
    var carYear = 2019;  
  }  
}
```

// aquí se puede usar carName y carYear

// aquí NO se puede usar carName ni carYear

Uso de let

```
// aquí no se puede usar carName ni  
carYear  
function miFuncion() {  
  let carName = "Volvo";  
  if (true) {  
    let carName = "Toyota";  
    let carYear = 2019;  
  }  
  // aquí se puede usar carName, pero  
  no carYear  
}  
// aquí NO se puede usar carName ni  
carYear
```

const tiene un uso similar a let, con la diferencia de que solamente se permite asignarle valores una vez

Objetos

```
var person = {  
  firstName: "John",  
  lastName : "Doe",  
  id       : 5566,  
  fullName : function() {  
    return this.firstName + " " + this.lastName;  
  }  
};
```

Se puede acceder de las siguientes formas:

```
var x = person.firstName;  
x = person["firstName"];  
var z = person.fullName();
```

Arreglos

```
let fruits = ["Apple", "Orange", "Plum"];  
  
alert( fruits.length ); // 3
```

```
let cars = ["BMW", "Volvo", "Mini"];  
let text = "";  
  
for (let x of cars) {  
    text += x + "<br>";  
}
```

```
// mix of values  
let arr = [ 'Apple', { name: 'John' }, true, function() { alert('hello'); } ];  
  
// get the object at index 1 and then show its name  
alert( arr[1].name ); // John  
  
// get the function at index 3 and run it  
arr[3](); // hello
```

Tiene métodos como:

- push() //arr.push("final");
- unshift() //arr.unshift("inicio");
- slice()
- forEach()
- concat()
- map()
- find()
- filter()
- sort()

Función Constructor

Con funciones también se pueden crear instancias de objetos.

```
1 function User(name) {  
2   this.name = name;  
3   this.isAdmin = false;  
4 }  
5  
6 let user = new User("Jack");  
7  
8 alert(user.name); // Jack  
9 alert(user.isAdmin); // false
```

Uso de this

- El uso de **this** es diferente a los demás lenguajes y puede variar dependiendo del caso.
- **En tiempo de ejecución**, el **this** se convierte en el objeto que invoca a la función.

```
var o = {prop: 37};

function independent() {
  return this.prop;
}

o.f = independent;

console.log(o.f()); // logs 37
```

Callbacks

- Una función de callback es una función que se pasa a otra función como un argumento, que luego se invoca dentro de la función externa para completar algún tipo de rutina o acción.

```
let result = arr.map(function(item, index, array) {  
    // returns the new value instead of item  
});
```

JSON

- **JavaScript Object Notation** es un formato de intercambio o estructura de datos.
- Su sintaxis está en el lenguaje JavaScript.
- Es ampliamente usado para envío de información del Servidor al Cliente y viceversa.

Objeto en JSON

```
{  
  "id_libro": 308,  
  "titulo": "Web APIs",  
  "autor": "Microsoft",  
  "anio": 2011,  
  "multapordia": 50.00  
}
```


Lista de Objetos en JSON

```
[  
  {  
    "id_libro": 101,  
    "titulo": "Programacion en .NET",  
    "autor": "Deitel y Deitel",  
    "anio": 2014,  
    "multapordia": 81.00  
  },  
  {  
    "id_libro": 102,  
    "titulo": "Programacion en C++",  
    "autor": "Deitel y Deitel",  
    "anio": 2010,  
    "multapordia": 12.55  
  }  
]
```

Características de un JSON

- Requiere usar comillas dobles para las cadenas y los nombres de propiedades. Las comillas simples no son válidas.
- Una coma o dos puntos mal ubicados pueden producir que un archivo JSON no funcione.

Características de un JSON

```
{
  "orders": [
    {
      "orderno": "748745375",
      "date": "June 30, 2088 1:54:23 AM",
      "trackingno": "TN0039291",
      "custid": "11045",
      "customer": [
        {
          "custid": "11045",
          "fname": "Sue",
          "lname": "Hatfield",
          "address": "1409 Silver Street",
          "city": "Ashland",
          "state": "NE",
          "zip": "68003"
        }
      ]
    }
  ]
}
```

Cada campo puede tener cualquier tipo de dato, incluyendo objetos y arreglos, siempre y cuando respeten el formato JSON.

Clases

Dentro de la clase

```
class Greeter {  
  constructor(message) {  
    this.greeting = message;  
  }  
  greet() {  
    return "Hello, " + this.greeting;  
  }  
}
```

Fuera de la clase

```
let greeter = new Greeter("world");  
console.log(greeter.greet());
```

Funciones Arrow: Caso 1

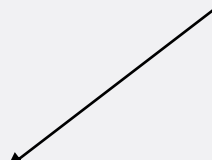
//Función expresión

```
nums = evens.map(  
  function (v, i) {  
    return v + i;  
  }  
);
```

//Función Arrow

```
nums = evens.map( (v, i) => v + i );
```

Simplifica el callback
en una sola línea



Funciones Arrow: Caso 2

```
nums = evens.map(  
    function (v, i) {  
        var x = v + i;  
        return x;  
    }  
);
```

//Arrow

```
nums = evens.map(  
    (v, i) => {  
        var x = v + i;  
        return x;  
    }  
);
```

Funciones Arrow: Caso 3

```
pairs = evens.map(  
    function (v) {  
        return {  
            even: v,  
            odd: v + 1  
        };  
    }  
);  
  
//Arrow  
pairs = evens.map(v => ({ even: v, odd: v + 1 }));
```

Funciones Arrow: Caso 4

```
pairs = evens.map(  
  function (v) {  
    var obj = { even: v, odd: v + 1 };  
    return obj;  
  }  
);
```

//Arrow

```
pairs = evens.map(  
  v => {  
    var obj = { even: v, odd: v + 1 };  
    return obj;  
  }  
);
```


JS