

1. ECMAScript

Es la especificación del lenguaje propuesto por ECMA Internacional, que es una institución encargada de los estándares y JavaScript es el lenguaje de programación que utiliza esta especificación para trabajar sobre estas características que van siendo añadidas año con año a partir del dos mil quince que fue lanzada la versión seis.

ECMAScript 6 se implemento en el año 2015

2. Default Params y Concatenación

Funcionalidad en las funciones a partir de ECMAScript

Antes de ECMAScript6

```
function newFunction(name, age, country) {  
  var name = name || 'oscar';  
  var age = age || 32;  
  var country = country || 'MX';  
  console.log(name, age, country);  
}
```

Con ECMAScript6

```
function newFunction2(name = 'oscar', age = 32, country = "MX") {  
  console.log(name, age, country);  
}  
  
newFunction2('Ricardo', '23', 'CO');
```

Template Literals: permiten separar o unir varios elementos

Antes de ECMAScript6

```
let hello = "Hello";  
let world = "World";  
let epicPhrase = hello + ' ' + world;  
console.log(epicPhrase);
```

Con ECMAScript6

```
let hello = "Hello";  
let world = "World";  
  
let epicPhrase2 = `${hello} ${world}`;  
console.log(epicPhrase2);
```

3. LET y CONST, Multilínea, Spread Operator y Desestructuración

Antes de ECMAScript6

```
let lorem = "Qui consequat. Comodi. Ipsum vel dui yet minima \n"  
+ "otra frase epica que necesitamos."  
console.log(lorem);
```

Con ECMAScript6

```
let lorem2 = `otra frase epica que necesitamos  
ahora es otra frase epica`;  
console.log(lorem2);
```

Antes de ECMAScript6

```
let person = {  
  'nombre': 'oscar',  
  'age': 32,  
  'country': 'MX'  
}  
  
console.log(person.nombre, person.age);
```

Con ECMAScript6

```
let person = {  
  'nombre': 'oscar',  
  'age': 32,  
  'country': 'MX'  
}  
  
let { nombre, age, country } = person;  
  
console.log(nombre, age, country);
```

Antes de ECMAScript6

```
let team1 = ['Oscar', 'Julian', 'Ricardo'];  
let team2 = ['Valeria', 'Yesica', 'Camila'];  
  
let educacion = ['David', 'Oscar', 'Julian', 'Ricardo', 'Valeria', 'Yesica', 'Camila'];
```

Con ECMAScript6

```
let team1 = ['Oscar', 'Julian', 'Ricardo'];
let team2 = ['Valeria', 'Yesica', 'Camila'];

let educacion = ['David', ...team1, ...team2];

console.log(educacion);
```

4. Arrow Functions, Promesas y Parámetros en objetos

Antes de ECMAScript6

```
let nombre = 'oscar';
let age = 32;

obj = { nombre: nombre, age: age };

console.log(obj);
```

Con ECMAScript6

```
let nombre = 'oscar';
let age = 32;

obj2 = { nombre, age };

console.log(obj2);
```

Antes de ECMAScript6

Funciones anónimas

```
const nombres = [
  {nombre: 'Oscar', age: 32},
  {nombre: 'Yesica', age: 27}
]

let listOfNames = nombres.map(function (item) {
  console.log(item.nombre);
})
```

Con ECMAScript6

Arrow Functions

```
const nombres = [  
  {nombre: 'Oscar', age: 32},  
  {nombre: 'Yesica', age: 27}  
]  
  
let listOfNames2 = nombres.map(item => console.log(item.nombre));
```

```
const listOfNames3 = (nombre, age, country) => {  
  
}  
  
const listOfNames4 = nombre => {  
  
}  
  
const square = num => num * num;  
console.log(square(2));
```

Promesas (para trabajar el asincronismo)

```
const helloPromise = () => {  
  return new Promise((resolve, reject) => {  
    if(true) {  
      resolve('Hey!');  
    } else {  
      reject('Ups!!');  
    }  
  });  
}  
  
helloPromise()  
  .then(response => console.log(response))  
  .then(() => console.log('hola'))  
  .catch(error => console.log(error));
```

5. Clases, Módulos y Generadores

```
class calculator {
  constructor() {
    this.valueA = 0;
    this.valueB = 0;
  }
  sum(valueA, valueB) {
    this.valueA = valueA;
    this.valueB = valueB;
    return this.valueA + this.valueB;
  }
}

const calc = new calculator();
console.log(calc.sum(2,2));
```

Módulos

modulo.js

```
const hello = () => {
  return 'hello';
}

export default hello;
```

Importo mi modulo a mi archivo js

```
import { hello } from 'module';

hello();
```

Generadores

```
function* helloWorld() {
  if (true) {
    yield 'Hello, ';
  }
  if (true) {
    yield 'World';
  }
};

const generatorHello = helloWorld();
console.log(generatorHello.next().value);
```

```
console.log(generatorHello.next().value);  
console.log(generatorHello.next().value);
```

* Un caso de generators sería Fibonacci

6. ¿Qué se implementó en ES7?

Includes

```
let numbers = [1, 2, 3, 7, 8];  
  
if(numbers.includes(7)) {  
  console.log('Si se encuentra el valor 7');  
} else {  
  console.log('No se encuentra');  
}
```

Potencias

```
let base = 4;  
let exponent = 3;  
let result = base ** exponent;  
  
console.log(result);
```

7. ¿Qué se implementó en ES8?

Esta versión fue lanzada en junio de 2017


Matriz

```
const data = {  
  frontend: 'Oscar',  
  backend: 'Isabel',  
  design: 'Ana',  
}  
  
const entries = Object.entries(data);  
console.log(entries);  
console.log(entries.length);
```

Object values

Muestra los valores del objeto

```
const data = {  
  frontend: 'Oscar',  
  backend: 'Isabel',  
  design: 'Ana',  
}  
  
const values = Object.values(data);  
console.log(values);  
console.log(values.length);
```



Trailing commas
(Ya no da error de sintaxis)

Pad

```
const string = 'hello';  
console.log(string.padStart(7, 'hi'));  
console.log(string.padEnd(12, ' -----'));  
console.log('food' .padEnd(12, ' -----'));
```

8. Async Await

Principal característica de ECMAScript 8, Async Await

```
const helloWorld = () => {
  return new Promise((resolve, reject) => {
    (true)
    ? setTimeout(() => resolve('Hello World'), 3000)
    : reject(new Error('Test Error'))
  })
};

const helloAsync = async () => {
  const hello = await helloWorld();
  console.log(hello);
};

helloAsync();

const anotherFunction = async () => {
  try {
    const hello = await helloWorld();
    console.log(hello);
  } catch (error) {
    console.log(error);
  }
};

anotherFunction();
```


9. ¿Qué se implementó en ES9?

Esta versión fue lanzada en junio de 2018

Operador de reposo

Puede extraer las propiedades de un objeto que aun no se ha construido

```
const obj = {
  nombre: 'Oscar',
  age: 32,
};

const obj1 = {
  ...obj,
  country: 'MX'
};

console.log(obj1);
```

Promise.finally

```
const helloWorld = () => {
  return new Promise((resolve, reject) => {
    (true)
    ? setTimeout(() => resolve('Hello World'), 3000)
    : reject(new Error('Test Error'))
  });
};

helloWorld()
  .then(response => console.log(response))
  .catch(error => console.log(error))
  .finally(() => console.log('Finalizo'))
```

Agrupar bloques de regex y poder acceder a cada uno de ellos

```
const regexData = /([0-9]{4})-([0-9]{2})-([0-9]{2})/
const match = regexData.exec('2018-04-20');
const year = match[1];
const month = match[2];
const day = match[3];

console.log(year, month, day);
```

10. ¿Qué se implementó en ES10?

Esta versión fue lanzada en junio de 2019

Flat

```
let array = [1,2,3, [1,2,3, [1,2,3]]];

console.log(array.flat(2));

let array = [1,2,3,4,5];

console.log(array.flatMap(value => [value, value * 2]));
```

Trim

```
let hello = '    hello world';

console.log(hello);
console.log(hello.trimStart());

let hello = 'hello world    ';
console.log(hello);
console.log(hello.trimEnd());
```

Try catch

Ahora viene el opcional catch binding, significa que podemos pasar de forma opcional el parámetro de error al valor de catch

```
try {

} catch {
  error
```

```
}
```

From Entries

Ahora podemos trabajar de objetos a arreglos y de arreglos a objetos

```
let entries = [{"nombre", "oscar"}, {"age", 32}];  
console.log(Object.fromEntries(entries));
```

Objeto de tipo simbolo

```
let mySymbol = `My Symbol`;  
let symbol = Symbol(mySymbol);  
console.log(symbol.description);
```

10. TC39

TC39 es un comité compuesto por un grupo de desarrolladores que están a cargo de revisar propuestas y revisar estándares para implementar nuevas versiones de ECMAScript.

A través del sitio web es posible contribuir y poder ver su estado:

<https://tc39.es/>

