sustantiva

CON•SENTIDO

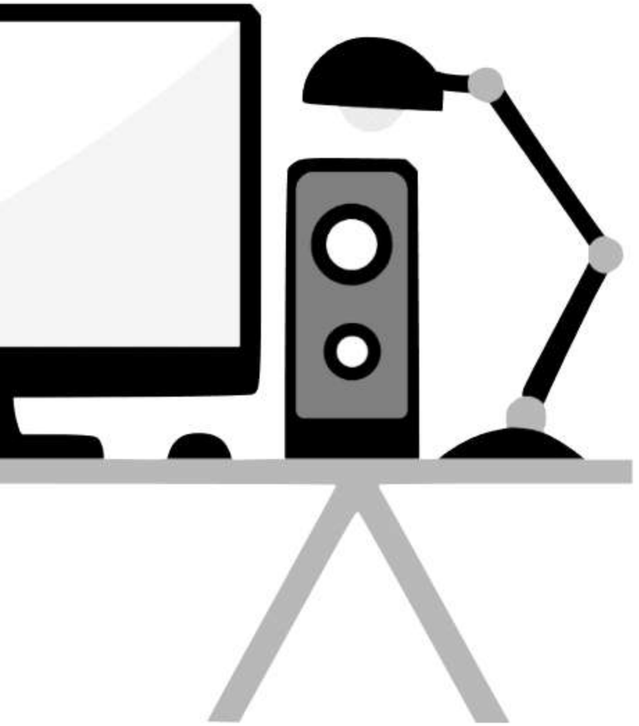
CARACTERÍSTICAS DE JS ES6

Lección 02

Competencia General

Utilizar las nuevas funcionalidades de la especificación ES6+ para la implementación de un algoritmo JavaScript que resuelve un problema Planteado.

Contenidos



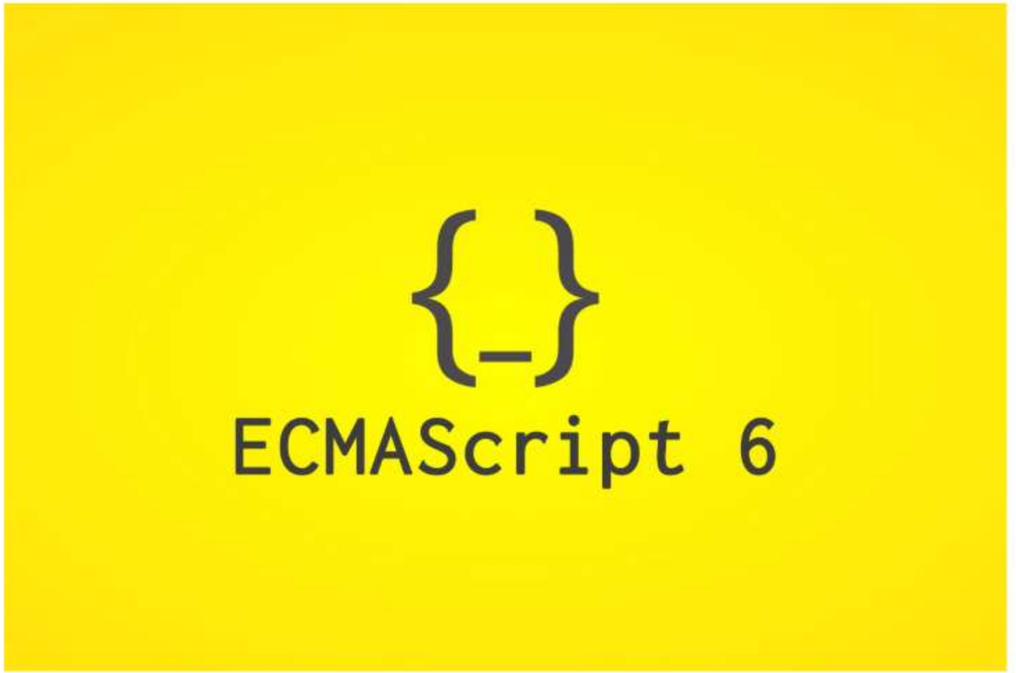
Especificación ES6+

1. Qué es ES6+
2. Compatibilidad ES6
3. Webpack
4. Babel
5. Polyfills
6. Características de ES6+

¿Qué es ES6+?

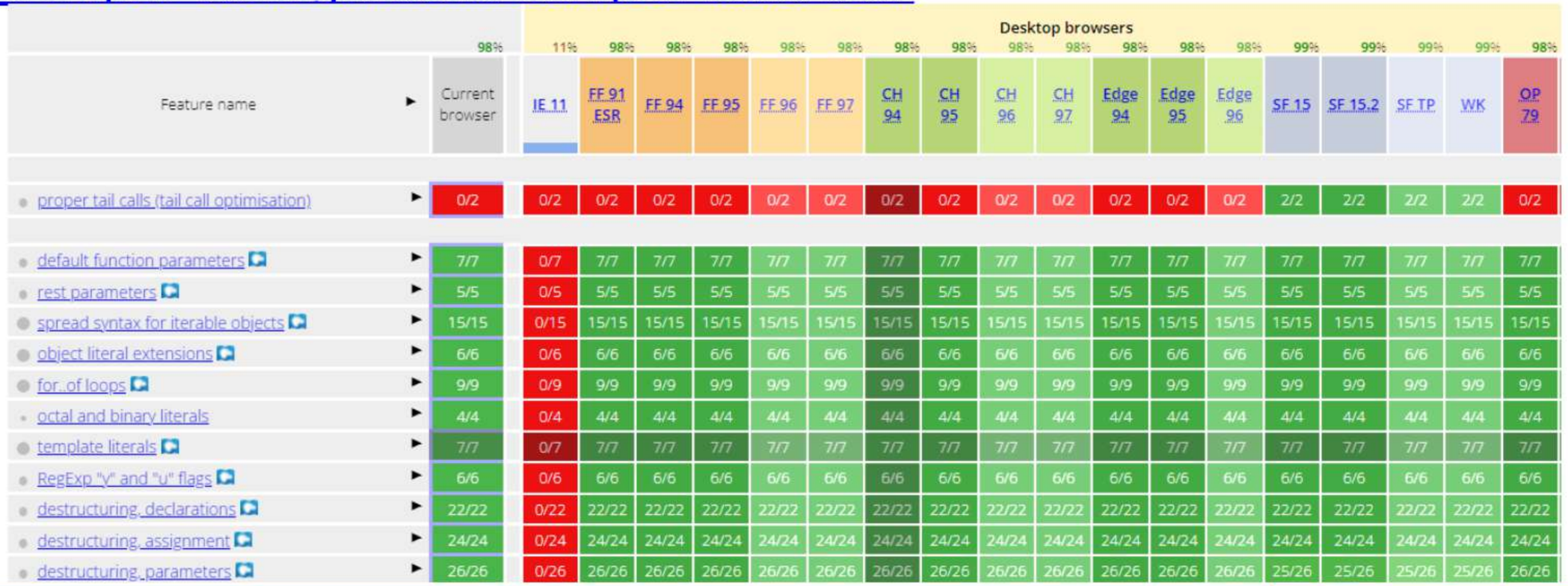
ECMAScript 2015 o ES6, es la nueva versión de JavaScript, aprobada el 2015.

Esta versión esta orientada a clases y herencias, además, incluye otras novedades interesantes, como el uso de módulos, funciones flecha, entre otros.



Compatibilidad de ES6

Podemos obtener una lista detallada de cada nueva característica en ES6 y su compatibilidad con un navegador ingresando al siguiente link:
https://compat-table.github.io/compat-table/es6/



		Desktop browsers																	
	98%	11%	98%	98%	98%	98%	98%	98%	98%	98%	98%	98%	98%	98%	99%	99%	99%	99%	98%
Feature name	Current browser	IE 11	FF 91 ESR	FF 94	FF 95	FF 96	FF 97	CH 94	CH 95	CH 96	CH 97	Edge 94	Edge 95	Edge 96	SF 15	SF 15.2	SF TP	WK	OP 79
proper tail calls (tail call optimisation)	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	2/2	2/2	2/2	2/2	0/2
default function parameters	7/7	0/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7
rest parameters	5/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
spread syntax for iterable objects	15/15	0/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15
object literal extensions	6/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6
for..of loops	9/9	0/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9
octal and binary literals	4/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4
template literals	7/7	0/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7
RegExp "y" and "u" flags	6/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6
destructuring, declarations	22/22	0/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22
destructuring, assignment	24/24	0/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24
destructuring, parameters	26/26	0/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	25/26	25/26	25/26	25/26	26/26

Webpack

Webpack nos brinda una serie de características a JavaScript:

- Gestión de dependencias de un proyecto
- Conversión de formatos
- Carga y uso de módulos

Se instala usando el administrador de dependencias npm:

```
npm install -g webpack
```

```
var webpack = require("webpack");

module.exports = {
    entry: './src/app.js',
    output: {
        filename: './bin/app.bundle.js'
    },
    plugins: [
        new webpack.optimize.UglifyJsPlugin()
    ]
};
```

Conviene usarlo en aplicaciones grandes las cuales requieren muchos archivos JavaScript.

Babel

Babel es una herramienta que permite transformar código JS de ES6+ a cualquier navegador o versión de node.js.
Babel funciona por medio de plugins para que le indiquemos que cosas queremos que transforme


```
node_modules/.bin/babel script.js -o script-compiled.js
```

Enviamos por parámetro el script a transformar y nos retorna un JS ya transformado.

Polyfill

Polyfill es un módulo o fragmento de código que permite hacer que las nuevas funciones estén disponibles en navegadores antiguos que no tienen compatibilidad con dicha función.

POLYFILL

Ejemplo de función startsWith( ) disponible para navegadores que no incluyen disponibilidad con esta función

```
if (!String.prototype.startsWith) {
  String.prototype.startsWith = function (searchString, position) {
    position = position || 0;
    return this.indexOf(searchString, position) === position;
  };
}
```

Características de ES6+

Variables

En ES6 si se introduce una nueva forma de declarar variables, que es mediante la palabra reservada let. El ámbito de las variables así definidas está limitado al bloque donde se han declarado

```
function foo(){
  var a = 3;
  if (a < 5){
    let b = 10;
    console.log(a + b); //devuelve 13
  }

  console.log(b); //b no está definida fuera del if
}
```

Características de ES6+

Diferencias entre let var y const

Característica	const	let	var
Alcance global	NO	NO	SI
Alcance en una función	SI	SI	SI
Alcance en un bloque	SI	SI	NO
¿Se puede modificar?	NO	SI	SI

Características de ES6+

Funciones

ECMAScript6 (ES6) trae muchas novedades muy interesantes:

- Argumentos por defecto.
- Operador ... para indicar resto de argumentos.
- Funciones flecha.
- Generadores.

Parámetros por defecto

Parámetros predeterminados de función permiten que los parámetros con nombre se inicien con valores predeterminados si no se pasa ningún valor o undefined.

```
function multiply(a, b = 1) {
  return a * b
}

multiply(5, 2)          // 10
multiply(5)             // 5
multiply(5, undefined)  // 5
```

Características de ES6+

Funciones flecha

Una expresión de función flecha es una alternativa compacta a la función tradicional de JavaScript.

No se puede utilizar como constructor de una clase.

```
// Función tradicional
function suma (a){
  return a + 100;
}

// Función flecha
let suma = (a) => (a+100);
```

Interpolación de cadenas

Podemos utilizar “Placeholders” para asignar variables en una cadena de String. Una forma mas visual de concatenar String

```
var nombre = "Juan";
console.log(`¡Hola ${nombre}!`);
// ¡Hola Juan!
```

Características de ES6+

Desestructuración de objetos y arreglos

La desestructuración permite extraer valores de arreglos u objetos utilizando una sintaxis de asignación en el lado izquierdo. Esto facilita declarar qué valores se quieren desempaquetar y en qué variables almacenarlos.

```
// Destructuring arrays
const x = [1, 2, 3, 4, 5];
const [y, z] = x;
console.log(y); // 1
console.log(z); // 2


const user = {
    id: 42,
    is_verified: true
};
// Destructuring object
const {id, is_verified} = user;

console.log(id); // 42
console.log(is_verified); // true
```

Características de ES6+

Operador spread

Permite a un elemento iterable ser expandido en lugares donde cero o más pares de valores clave son esperados

```
const numbers = [1, 2, 3];

console.log(...numbers);
// output: 1 2 3
```

Características de ES6+

Operador Rest

Permite representar un número indefinido de argumentos como un arreglo

```
function sum(...theArgs) {
  return theArgs.reduce((previous, current) => {
    return previous + current;
  });
}

console.log(sum(1, 2, 3));
// output: 6

console.log(sum(1, 2, 3, 4));
// output: 10
```

Asignación concisa de atributos

La asignación concisa de atributos permite crear objetos de forma más limpia y legible cuando el nombre de la propiedad y el nombre de la variable son iguales.

```
const nombre = "Ana";
const edad = 28;

const persona = {
  nombre: nombre,
  edad: edad
};
```

```
const nombre = "Ana";
const edad = 28;

const persona = { nombre, edad
};
```

Características de ES6+

Clases

Permite propiedades y métodos privados en una clase evitando que estos se modifiquen.

```
class ClassWithPrivateField {
  #privateField
}

class ClassWithPrivateMethod {
  #privateMethod() {
    return 'hello world'
 }
}
```

Características de ES6+

Herencia

En ES6 podemos “extender” clases implementando funciones de la clase padre para que las clases utilicen herencia

```
// Clase padre
class Forma {
  constructor() {
    console.log("Soy una forma geométrica.");
  }
  gritar() {
    console.log("YEP!!");
  }
}
// Clases hijas
class Cuadrado extends Forma {
  constructor() {
    super();
    console.log("Soy un cuadrado.");
  }
}
class Circulo extends Forma {
  constructor() {
    super();
    console.log("Soy un círculo.");
  }
}
class Triangulo extends Forma {
  constructor() {
    super();
    console.log("Soy un triángulo.");
  }
}
```

Atributos en clases

Los atributos en una clase representan las propiedades que tendrán los objetos creados a partir de ella.
Se definen en el constructor utilizando this.

```
class Persona {
  constructor(nombre, edad) {
    this.nombre = nombre; // atributo
    this.edad = edad;     // atributo
  }

  saludar() {
    console.log(`Hola, soy ${this.nombre} y tengo
${this.edad} años.`);
  }
}

const persona1 = new Persona("Sofía", 30);
persona1.saludar(); // Hola, soy Sofía y tengo 30 años.
```

Características de ES6+

Sets y Maps

El objeto Map es una colección de datos identificados por clave, se diferencia de un objeto dado que permite cualquier tipo de clave

El objeto Set es una colección de valores sin clave donde cada valor puede aparecer solo una vez

```
let map = new Map();

map.set('1', 'str1');   // un string como clave
map.set(1, 'num1');     // un número como clave
map.set(true, 'bool1'); // un booleano como clave
```

```
let set = new Set();

let john = { name: "John" };
let pete = { name: "Pete" };
let mary = { name: "Mary" };

// visitas, algunos usuarios lo hacen varias veces
set.add(john);
set.add(pete);
set.add(mary);
set.add(john);
set.add(mary);

// set solo guarda valores únicos
alert( set.size ); // 3
```

Características de ES6+

Iteradores y generadores

JavaScript proporciona diversas formas de iterar sobre una colección, desde simples bucles for hasta métodos como map() y filter()

Los iteradores permiten recorrer una colección y devolver un valor al terminar.

Los generadores permiten definir un algoritmo iterativo.

```
function crearIterador(arreglo){
    var siguienteIndice = 0;

    return {
       next: function(){
           return siguienteIndice < arreglo.length ?
               {value: arreglo[siguienteIndice++], done: false} :
               {done: true};
       }
    }
}

var it = crearIterador(['yo', 'ya']);
console.log(it.next().value); // 'yo'
console.log(it.next().value); // 'ya'
console.log(it.next().done);  // true
```

Características de ES6+

Módulos

Un módulo es un archivo que contiene un script de JavaScript

Los módulos pueden cargarse entre si con las directivas de **export e import** de esta forma intercambian funcionalidades

```
// 📁 hola.js
export function decirHola(user)
{
  alert(`hola, ${user}!`);
}

// 📁 main.js
import {decirHola} from './hola.
js';

alert(decirHola); // function...
decirHola('John'); // hola, John
!
```

Características de ES6+

Promesas

Una promesa es un objeto que representa la terminación o fracaso de una operación asíncrona.

```
new Promise((resolver, rechazar) => {
    console.log('Inicial');

    resolver();
})
.then(() => {
    throw new Error('Algo falló');

    console.log('Haz esto');
})
.catch(() => {
    console.log('Haz aquello');
})
.then(() => {
    console.log('Haz esto sin que importe
lo que sucedió antes');
});
```

Async / await

Una función con una declaración ***async*** incluye una promesa, cuando la función devuelve un valor, entonces esa promesa quedo resuelta.

Una función async incluye una expresión ***await***, esta expresión pausará la ejecución de la función hasta que la promesa sea resuelta.

```
async function getProcessedData(url) {
  let v;
  try {
    v = await downloadData(url);
  } catch(e) {
    v = await downloadFallbackData(url);
  }
  return processDataInWorker(v);
}
```

sustantiva

CON•SENTIDO