

ECMAScript

DEV.F
DESARROLLAMOS(PERSONAS);

dev

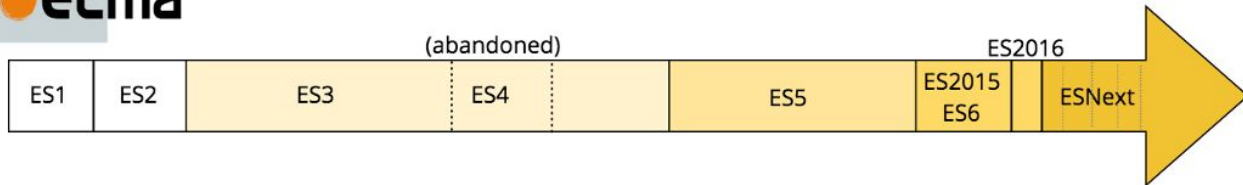
¿Qué es ECMA?

Ecma International es una organización internacional basada en membresías de estándares para la comunicación y la información. Adquirió el nombre Ecma International en 1994, cuando la European Computer Manufacturers Association (ECMA) cambió su nombre para expresar su alcance internacional.

Fue fundada en 1961 para estandarizar los sistemas informatizados en Europa.



ECMAScript



JS



ECMAScript

Es el estándar que la empresa del mismo nombre definió para JavaScript en el año 2015 (ES6) y encarga de regir como debe ser interpretado y cómo debe funcionar el lenguaje JavaScript.

Versión	Nombre	Publicación	Guía
ES11	ES2020	Junio del 2020	-
ES10	ES2019	Junio del 2019	-
ES9	ES2018	Junio del 2018	-
ES8	ES2017	Junio del 2017	-
ES7	ES2016	Junio del 2016	-
ES6	ES2015	Junio del 2015	-
ES5.1	ES5.1	Junio del 2011	-

The logo consists of the text 'DEV.F.' in a bold, white, sans-serif font. The 'F' is stylized with a grid of small squares at its right end. It is centered within a dark blue diamond shape.

DEV.F.

Características de esN

Let, const, templates y arrow



- **Let y const:** Para limitar el alcance a bloque (var solo lo hacía a nivel de función) y definir constantes.
- **Template literals (Template strings):** Para concatenar mensajes que utilizan variables sin abusar del operador +.
- **Arrow functions:** Forma más legible y que las expresiones de función. Funciona diferente en cuanto a this, super y su hoisting.

For of/in y módulos



- **For/of y for/in:** Ciclos con iteradores comodín para cada elemento de un array** y un objeto.
- **Módulos:** Mecanismo para dividir nuestros programas y solo invocar el código que sea realmente necesario.

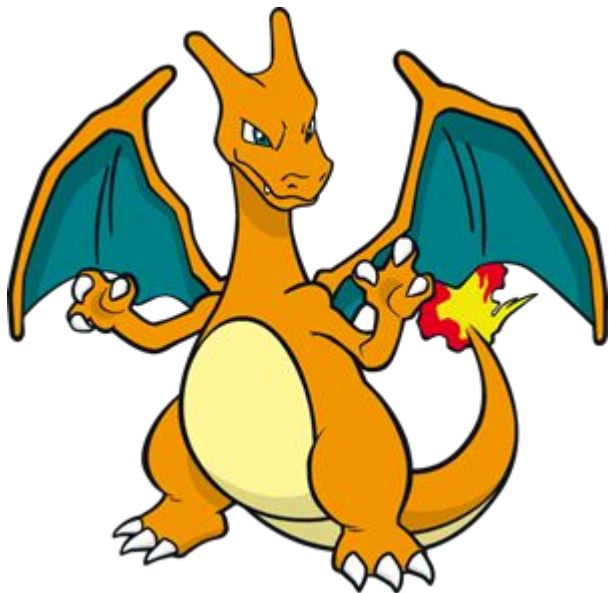
****** Para cualquier elemento que sea iterable en realidad.

Default parameters, spread y rest operator



- **Default parameters:** Para inicializar los valores que recibe una función.
- **Spread operator:** Para iterar las propiedades de un tipo de dato complejo. Uno de sus principales usos es clonar objetos sin referencia (considere deep copy / shallow copy).
- **Rest operator:** Es la utilización del spread operator pero para la recepción de parámetros en una función.

Destructuring, Map y Set



- **Destructuring:** Provee la capacidad de leer propiedades y posiciones de un objeto y un array respectivamente, para asignarlas a una nueva variable.
- **Map y Set:** Son estructuras de datos especialmente creadas para fines específicos como tener nombres de clave más complejos (Map) y no tener valores repetidos (Set) en un conjunto de datos.

Math y Numbers



Math

- random, pow, sqrt,
- cos, sin, tan, etc.
- round, floor, fround, trunc, exp y
abs.
- PI y E.

Numbers

- NaN.
- parseFloat, parseInt, y toString.
- toFixed, toLocaleString y
valueOf

Classes, this y regex



Classes: Simplificación de la sintaxis para uso de prototipos y POO (Sugar syntax).

This, Call, bind y apply: Tipos de referenciadores de objeto, ayudan a definir/cambiar el ejecutor de una función.

RegExp: Las expresiones regulares son patrones para comparar la coincidencia de caracteres.

Promises



Promises: Es un objeto especial de JS del que no se conoce su respuesta al momento de crearla y ejecutarla.

Async / await: Para manejo más limpio de las promesas como mejora los callback y a then/catch.

Finally: Para ejecutar algo al final de la respuesta de una promise (tanto en status resolve como reject).

Symbol: Objetos de tipo primitivo que son “únicos”. Suele usarse en keys para objetos y nombres de eventos.

String, Object y Array

DEV.F
DESARROLLAMOS(PERSONAS);

dev

String

- `charAt`.
- `concat`.
- `indexOf`.
- `replace` y `replaceAll`.
- `trim`, `substr` y `substring`.
- `includes`, `startsWith` y `endsWith`.

Object

- create y assign.
- entries, keys y values.
- seal y frozen.

Array

- Map y forEach(diferencias).
- Reduce.
- Push.
- Splice.
- Includes.
- Filter.
- Find.
- FindIndex.

```
[1, 2, 3].push(4) // [1,2,3,4]
[1, 2, 3].pop() // [1,2]
[1, 2, 3].shift() // [2,3]
[1, 2, 3].unshift(0) // [0,1,2,3]
['a', 'b'].concat('c') // ['a','b','c']
['a', 'b', 'c'].join('-') // a-b-c
['a', 'b', 'c'].slice(1) // ['a','b']
['a', 'b', 'c'].indexOf('b') // 1
['a', 'b', 'c'].includes('c') // true
[3, 5, 6, 8].find((n) => n % 2 === 0) // 6
[2, 4, 3, 5].findIndex((n) => n % 2 !== 0) // 2
[3, 4, 8, 6].map((n) => n * 2) // [6,8,16,12]
[1, 4, 7, 8].filter((n) => n % 2 === 0) // [4,8]
[2, 4, 3, 7].reduce((acc, cur) => acc + cur) // 16
[2, 3, 4, 5].every((x) => x < 6) // true
[3, 5, 6, 8].some((n) => n > 6) // true
[1, 2, 3, 4].reverse() // [4,3,2,1]
[3, 5, 7, 8].at(-2) // 7
```


Un poco más de EsN

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Más de esN

- Valor por defecto de una variable (||).
- Encadenamiento opcional (cortocircuito ?).
- Nullish operator (??).
- Delete, type of, instanceof.
- Tipos de funciones en JS.
 - void
 - return



Conocimiento de programación

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Conocimiento de programación

- Tipos de datos.
- Operadores y tablas de verdad.
- Sentencias condicionales.
- Ciclos y concepto de iterador.
- Objetos.
- Arrays.
- Funciones o métodos.
- Validación de datos.
- Manejo de excepciones.



Cosas importantes de programar en JS

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Cosas importantes de programar en JS

- Hoisting.
- Uso correcto de var, let y const.
- Destructuring.
- Eventos y el DOM (WebApi).
- Localstorage.
- Comparaciones de valores null, undefined, vacío (0 y emptyString), NaN, etc.
- Event Loop.



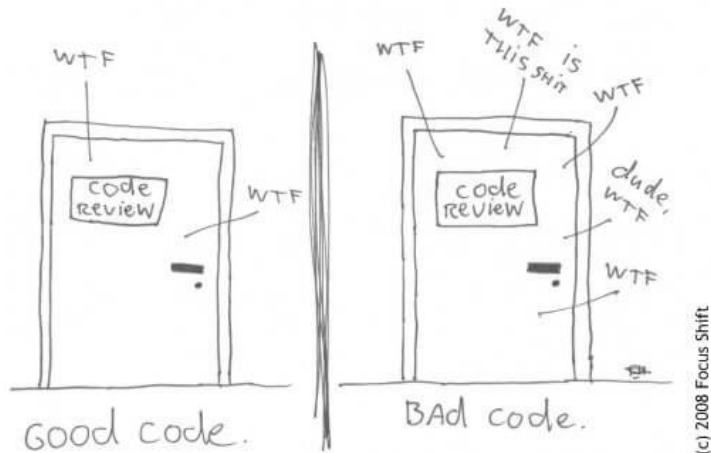
Clean code

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Cómo construir una casa...

The ONLY valid measurement
OF code QUALITY: WTFs/minute



Clean code

- Longitud de línea e indentación.
- Tener un código (de honor) para codear.
- Nombrado de variables y funciones.
- Reutilización y definición de funciones.
- Inmutabilidad.
- Documentación.
- Uso de dummies y constantes.
- Normalización de datos (patrón selector).