ECMAScript

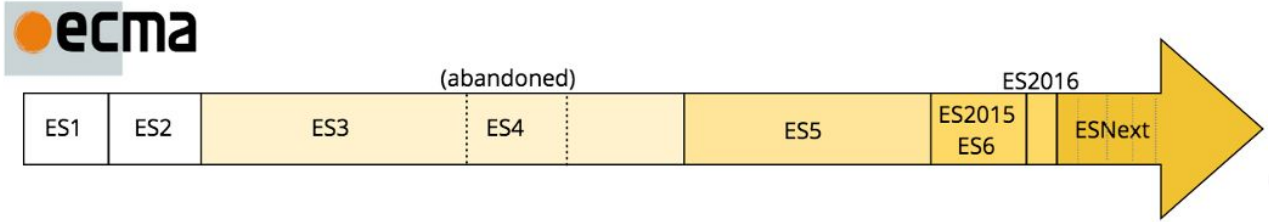
DEV.F:.
DESARROLLAMOS(PERSONAS);

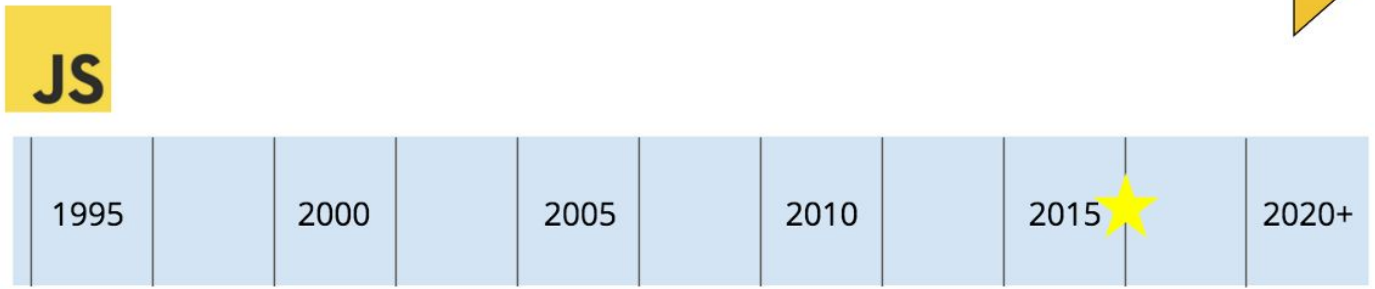
¿Qué es ECMA?

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


ecma
(abandoned)
ES2016
ES1
ES2
ES3
ES4
ES5
ES2015
ES6
ESNext


JS
1995
2000
2005
2010
2015
2020+
iOS

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	-

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.

DEV.F::

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.**

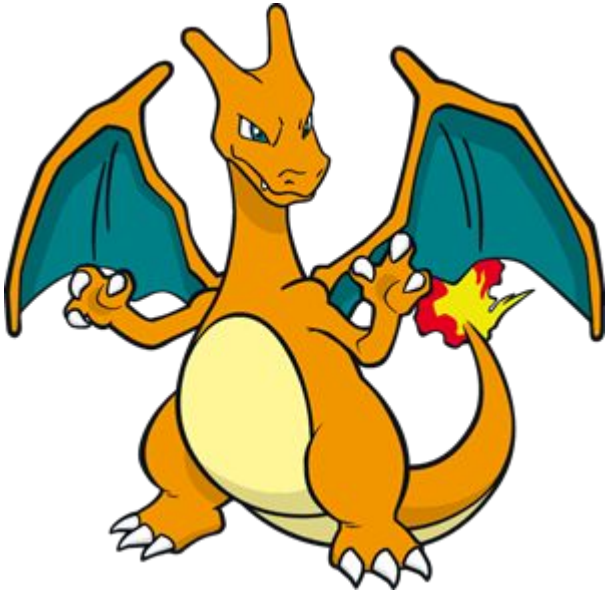
DEV.F::

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.

DEV.F::

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.

DEV.F::

Math y Numbers

Math

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

Numbers

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

DEV.F::

Classes, this y regex

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

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.

DEV.F::

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.

DEV.F::

String, Object y Array

DEV.F:: DESARROLLAMOS(PERSONAS);

String

- **charAt:** Dado un string y una posición permite obtener el carácter del string en en esa posición.
- **concat:** Permite concatenar un string con los n parámetros que se le indiquen.
- **indexOf:** Obtiene la posición de la primer coincidencia de un caracter dentro de un string.
- **replace y replaceAll:** Permiten cambiar (uno y todos respectivamente) caracteres de un string por otro.
- **trim:** Quitar espacios de un string.
- **substring:** Obtiene una parte de un string delimitado por dos índices.
- **includes:** Determina si una cadena de texto puede ser encontrada dentro de otra cadena de texto, devolviendo true o false según corresponda.
- **startsWith y endsWith:** Indican si una cadena de texto comienza/termina con los caracteres de una cadena de texto concreta.

Object

- **create:** Crea un nuevo objeto a partir de un prototipo.
- **assign:** Copia todas las propiedades de un objeto en otro para devolver un nuevo objeto.
- **keys:** Retorna un arreglo de strings con los nombres de todas las keys del objeto.
- **values:** Retorna un arreglo de strings con los valores de c/key objeto.
- **entries:** Retorna un arreglo de arreglos donde c/subarreglo es un par [‘key1’, ‘value1’]
- **seal:** Sella un objeto para evitar volverlo configurable (con prototype o herencia por ejemplo).
- **freeze:** Hace lo mismo que seal y además no permite modificar ningún valor de las propiedades de los objetos (vuelve las props constantes)

Array

- **Map y forEach:** Ambos recorren un arreglo pero map retorna un array con c/item y forEach solo itera.
- **Push:** Agrega elementos a un array.
- **Pop:** Elimina elementos de un array.
- **Splice:** Elimina los N elementos a partir de cierta posición.
- **Slice:** Obtiene un nuevo arreglo delimitado por dos posiciones del arreglo original.
- **Includes:** Devuelve true o false si encuentra la coincidencia de un elemento el array.
- **Filter:** Devuelve un nuevo array a partir de que cumpla con una determinada condición.
- **Find / FindIndex:** Para devolver el elemento o posición del elemento que satisface una cierta condicion.
- Reduce: ??.

```
[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
```

DEV.F

Un poco más de EsN

DEV.F::
DESARROLLAMOS(PERSONAS);

Más de esN

- Valor por defecto de una variable (||).
- Encadenamiento opcional (cortocircuito ?).
- Nullish operator (??).
- Delete, type of, instanceof.
- Destructuring.

Conocimientos de programación

DEV.F::
DESARROLLAMOS(PERSONAS);

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);

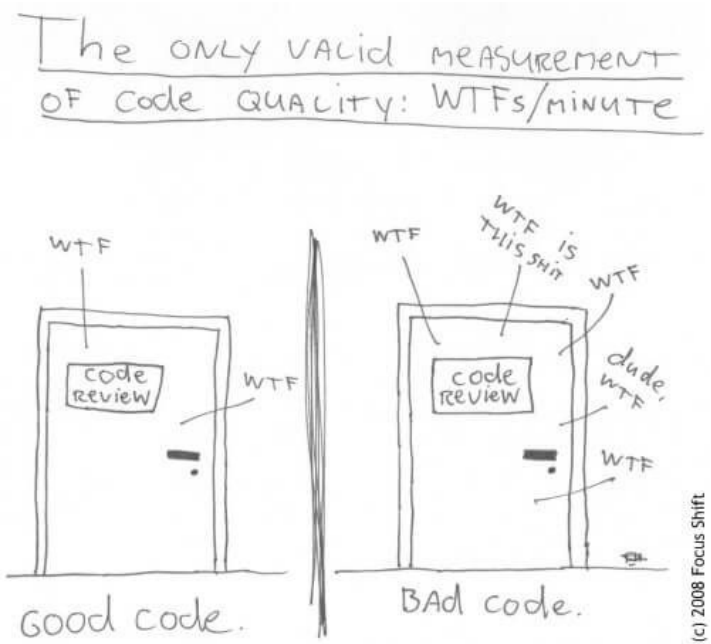
Cosas importantes de programar en JS

- Uso correcto de var, let y const.
- Destructuring.
- Paso por valor y por referencia.
- Comparaciones de valores null, undefined, vacío (0 y emptyString), NaN, false, etc.
- Tipos de funciones
  - Named.
  - Anonyms.
  - Arrows.
  - IIFE.
- Hoisting.
- Localstorage.
- *Eventos y el DOM (WebApi).*
- *Event Loop.*

Clean code

DEV.F::
DESARROLLAMOS(PERSONAS);

Cómo construir una casa...



Clean code

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

DEV.F::