CURSO DESAROLLO WEB CON MEAN


# WEB FULL STACK DEVELOPER

*Germán Caballero Rodríguez*
*germanux@gmail.com*

# EcmaScript 6 y 7

# INDICE

# ECMAScript v6

## La evolución de JavaScript

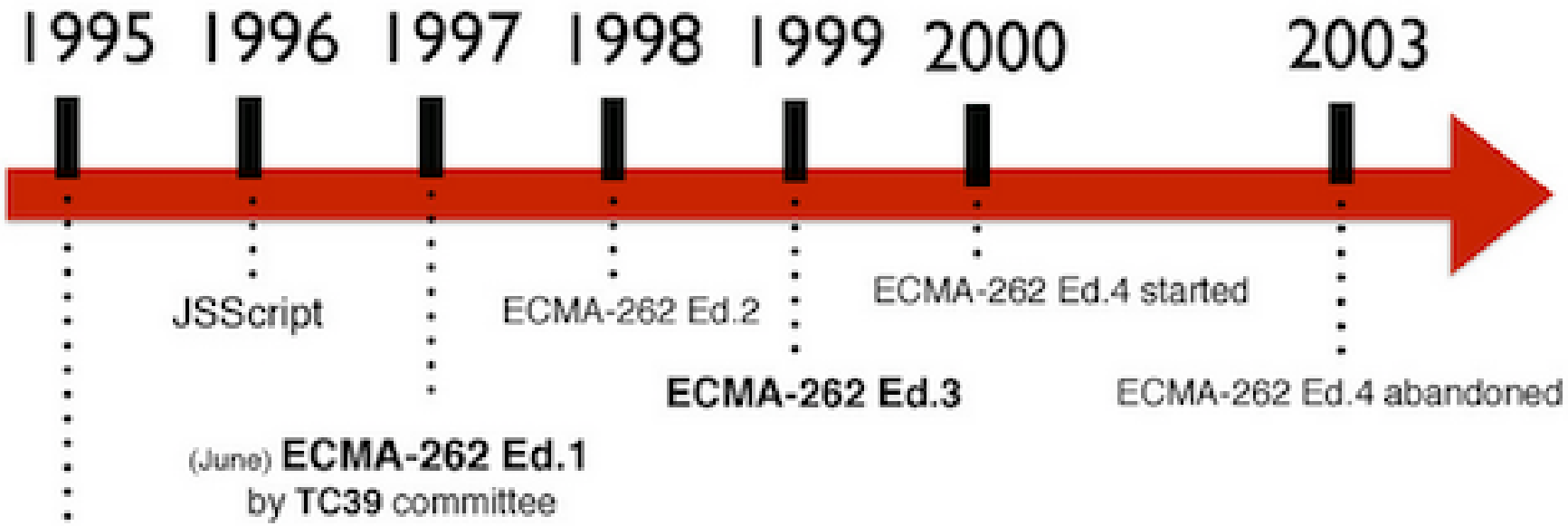
- Primero un poco de historia. En 1995 (hace 20 años!) Brendan Eich crea un lenguaje llamado Mocha cuando trabajaba en Netscape.
- En Septiembre de ese año lo renombra a LiveScript hasta que le cambiaron el nombre a JavaScript debido a una estrategia de marketing, ya que Netscape fue adquirida por Sun Microsystems, propietaria del lenguaje Java, muy popular por aquel entonces.

# ECMAScript v6

## La evolución de JavaScript

- En 1997 se crea un comité (TC39) para estadarizar JavaScript por la European Computer Manufacturers' Association, ECMA. Se diseña el estándar del DOM (Document Object Model) para evitar incompatibilidades entre navegadores.
- A partir de entonces los estándares de JavaScript se rigen por ECMAScript.

# ECMAScript v6




(May) **B. Eich invented Mocha**
(Sep) Mocha renamed to LiveScript
(Dec) LiveScript renamed to **JavaScript**

# ECMAScript v6

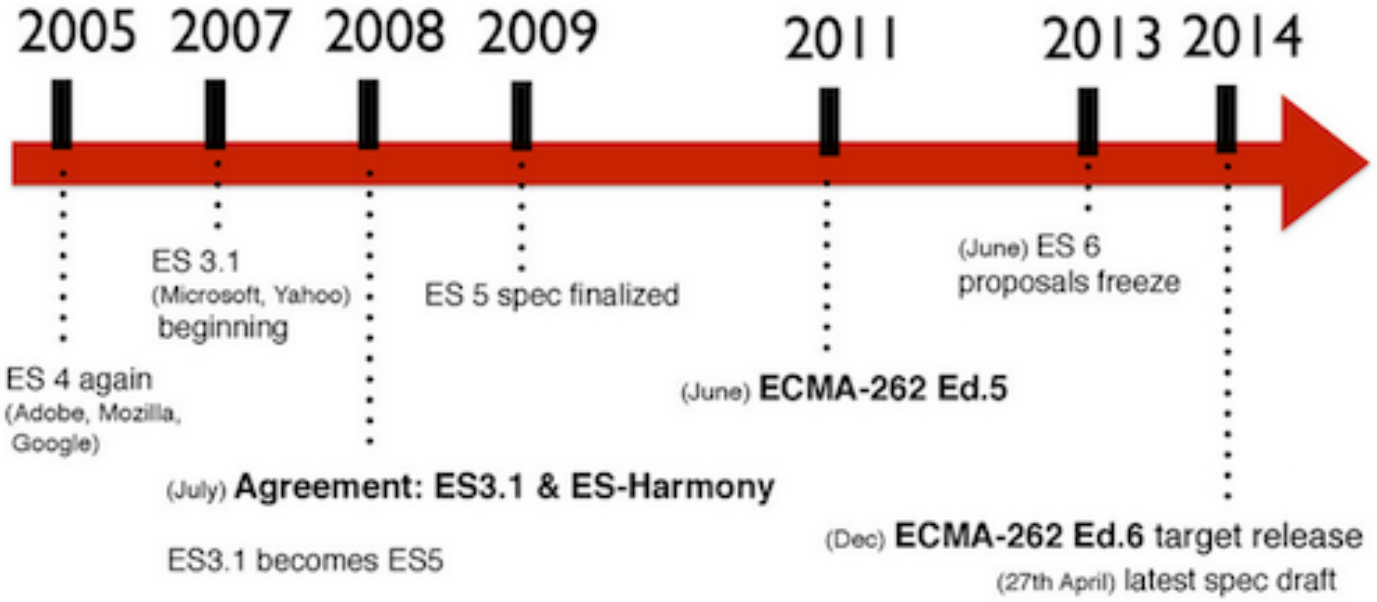
## La evolución de JavaScript

- En 1999 aparece la 3a versión del estándar ECMAScript, que se mantendría vigente hasta hace pocos años.
- Hubo pequeños intentos de escribir la versión 4, pero hasta 2011 no se aprobó y se estandarizó la versíon 5 (ES5) que es la que usamos hoy en día.

# ECMAScript v6

## La evolución de JavaScript

- En junio de 2013 quedó parado el borrador de la versión 6, pero en Diciembre de 2014 se aprobó al fin y se espera su estandarización a partir de Junio de 2015.



# ECMAScript v6

https://kangax.github.io/compat-table/es6/

| Feature name | Current browser | Compilers/polyfills | | | | | Desktop browsers | | | | | | | | | | | | | | | | Servers/runtimes | | | | | | | | Mobile | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 92% | 56% | 71% | 48% | 59% | 18% | 5% | 11% | 83% | 93% | 95% | 86% | 92% | 94% | 94% | 97% | 97% | 97% | 97% | 54% | 100% | 100% | 100% | 4% | 66% | 96% | 59% | 21% | 52% | 97% | 97% | 5% | 10% | 25% | 54% | 100% |
| | | Traceur | Babel + core-js[2] | Closure | Type-Script + core-js | es6-shim | KQ 4.14[3] | IE 11 | Edge 13[4] | Edge 14[4] | Edge 15[4] | FF 45 ESR | FF 50 | FF 51 Beta | FF 52 Aurora | FF 53 Nightly | CH 55, OP 42[1] | CH 56, OP 43[1] | CH 57, OP 44[1] | SF 9 | SF 10 | SF TP | WK | PJS | Echo JS | XS6 | JXA | Node 0.12[5] | Node 4[5] | Node 6.5[5] | Node 7[5] | AN 4.4 | AN 5.0 | AN 5.1 | iOS 9 | iOS 10 |
| **Optimisation** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 2/2 | 2/2 | 2/2 | 0/2 | 0/2 | 2/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 2/2 |
| **Syntax** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| default function parameters | 4/7 | 4/7 | 4/7 | 4/7 | 5/7 | 0/7 | 0/7 | 0/7 | 0/7 | 7/7 | 7/7 | 4/7 | 4/7 | 6/7 | 6/7 | 7/7 | 7/7 | 7/7 | 7/7 | 0/7 | 7/7 | 7/7 | 7/7 | 0/7 | 4/7 | 7/7 | 0/7 | 0/7 | 0/7 | 7/7 | 7/7 | 0/7 | 0/7 | 0/7 | 0/7 | 7/7 |
| rest parameters | 5/5 | 4/5 | 3/5 | 2/5 | 4/5 | 0/5 | 0/5 | 0/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 0/5 | 5/5 | 5/5 | 5/5 | 0/5 | 3/5 | 5/5 | 0/5 | 0/5 | 0/5 | 5/5 | 5/5 | 0/5 | 0/5 | 0/5 | 0/5 | 5/5 |
| spread (...) operator | 15/15 | 15/15 | 13/15 | 12/15 | 4/15 | 0/15 | 0/15 | 0/15 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 | 9/15 | 15/15 | 15/15 | 15/15 | 0/15 | 10/15 | 15/15 | 11/15 | 0/15 | 0/15 | 15/15 | 15/15 | 0/15 | 0/15 | 0/15 | 9/15 | 15/15 |
| object literal extensions | 6/6 | 6/6 | 6/6 | 4/6 | 6/6 | 0/6 | 0/6 | 0/6 | 6/6 | 6/6 | 6/6 | 6/6 | 6/6 | 6/6 | 6/6 | 6/6 | 6/6 | 6/6 | 6/6 | 5/6 | 6/6 | 6/6 | 6/6 | 0/6 | 5/6 | 6/6 | 5/6 | 0/6 | 6/6 | 6/6 | 6/6 | 0/6 | 0/6 | 0/6 | 5/6 | 6/6 |
| for..of loops | 7/9 | 9/9 | 9/9 | 6/9 | 3/9 | 0/9 | 0/9 | 0/9 | 7/9 | 7/9 | 9/9 | 7/9 | 7/9 | 7/9 | 7/9 | 9/9 | 9/9 | 9/9 | 9/9 | 8/9 | 9/9 | 9/9 | 9/9 | 0/9 | 7/9 | 9/9 | 8/9 | 7/9 | 7/9 | 9/9 | 9/9 | 0/9 | 0/9 | 7/9 | 8/9 | 9/9 |
| octal and binary literals | 4/4 | 2/4 | 4/4 | 4/4 | 4/4 | 2/4 | 0/4 | 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 | 0/4 | 2/4 | 4/4 | 4/4 | 0/4 | 4/4 | 4/4 | 4/4 | 0/4 | 0/4 | 0/4 | 4/4 | 4/4 |
| template literals | 5/5 | 4/5 | 4/5 | 3/5 | 3/5 | 0/5 | 0/5 | 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 | 0/5 | 4/5 | 5/5 | 5/5 | 0/5 | 5/5 | 5/5 | 5/5 | 0/5 | 0/5 | 0/5 | 5/5 | 5/5 |
| RegExp "y" and "u" flags | 5/5 | 3/5 | 3/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 5/5 | 5/5 | 5/5 | 2/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 0/5 | 5/5 | 5/5 | 5/5 | 0/5 | 2/5 | 2/5 | 0/5 | 0/5 | 0/5 | 5/5 | 5/5 | 0/5 | 0/5 | 0/5 | 0/5 | 5/5 |
| destructuring, declarations | 21/22 | 20/22 | 21/22 | 19/22 | 15/22 | 0/22 | 0/22 | 0/22 | 0/22 | 21/22 | 22/22 | 19/22 | 21/22 | 21/22 | 21/22 | 22/22 | 22/22 | 22/22 | 22/22 | 19/22 | 22/22 | 22/22 | 22/22 | 0/22 | 12/22 | 21/22 | 19/22 | 0/22 | 0/22 | 22/22 | 22/22 | 0/22 | 0/22 | 0/22 | 19/22 | 22/22 |
| destructuring, assignment | 23/24 | 23/24 | 24/24 | 17/24 | 19/24 | 0/24 | 0/24 | 0/24 | 0/24 | 23/24 | 24/24 | 21/24 | 23/24 | 23/24 | 23/24 | 24/24 | 24/24 | 24/24 | 24/24 | 21/24 | 24/24 | 24/24 | 24/24 | 0/24 | 14/24 | 24/24 | 21/24 | 0/24 | 0/24 | 24/24 | 24/24 | 0/24 | 0/24 | 0/24 | 21/24 | 24/24 |
| destructuring, parameters | 19/23 | 19/23 | 20/23 | 18/23 | 15/23 | 0/23 | 0/23 | 0/23 | 0/23 | 22/23 | 23/23 | 18/23 | 19/23 | 20/23 | 20/23 | 23/23 | 23/23 | 23/23 | 23/23 | 18/23 | 23/23 | 23/23 | 23/23 | 0/23 | 12/23 | 23/23 | 18/23 | 0/23 | 0/23 | 23/23 | 23/23 | 0/23 | 0/23 | 0/23 | 18/23 | 23/23 |
| Unicode code point escapes | 1/2 | 1/2 | 1/2 | 1/2 | 1/2 | 0/2 | 0/2 | 0/2 | 2/2 | 2/2 | 2/2 | 1/2 | 1/2 | 1/2 | 1/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 0/2 | 2/2 | 2/2 | 2/2 | 0/2 | 2/2 | 2/2 | 2/2 | 0/2 | 0/2 | 0/2 | 2/2 | 2/2 |
| new.target | 2/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 1/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 0/2 | 2/2 | 2/2 | 2/2 | 0/2 | 2/2 | 2/2 | 0/2 | 0/2 | 0/2 | 2/2 | 2/2 | 0/2 | 0/2 | 0/2 | 0/2 | 2/2 |

# ECMAScript v6

https://kangax.github.io/compat-table/es6/

| | | | Compilers/polyfills | | | | | Desktop browsers | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 92% | 56% | 71% | 48% | 59% | 18% | 5% | 11% | 83% | 93% | 95% | 86% | 92% | 94% | 94% |
| Feature name | ▶ | Current browser | Traceur | Babel + core-js[2] | Closure | Type-Script + core-js | es6-shim | KQ 4.14[3] | IE 11 | Edge 13[4] | Edge 14[4] | Edge 15[4] | FF 45 ESR | FF 50 | FF 51 Beta | FF 52 Aurora |
| **Optimisation** | | | | | | | | | | | | | | | | |
| 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 | 0/2 |
| **Syntax** | | | | | | | | | | | | | | | | |
| default function parameters | ▶ | 4/7 | 4/7 | 4/7 | 4/7 | 5/7 | 0/7 | 0/7 | 0/7 | 0/7 | 7/7 | 7/7 | 4/7 | 4/7 | 6/7 | 6/7 |
| rest parameters | ▶ | 5/5 | 4/5 | 3/5 | 2/5 | 4/5 | 0/5 | 0/5 | 0/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| spread (...) operator | ▶ | 15/15 | 15/15 | 13/15 | 12/15 | 4/15 | 0/15 | 0/15 | 0/15 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 |
| object literal extensions | ▶ | 6/6 | 6/6 | 6/6 | 4/6 | 6/6 | 0/6 | 0/6 | 0/6 | 6/6 | 6/6 | 6/6 | 6/6 | 6/6 | 6/6 | 6/6 |
| for..of loops | ▶ | 7/9 | 9/9 | 9/9 | 6/9 | 3/9 | 0/9 | 0/9 | 0/9 | 7/9 | 7/9 | 9/9 | 7/9 | 7/9 | 7/9 | 7/9 |
| octal and binary literals | ▶ | 4/4 | 2/4 | 4/4 | 4/4 | 4/4 | 2/4 | 0/4 | 0/4 | 4/4 | 4/4 | 4/4 | 4/4 | 4/4 | 4/4 | 4/4 |
| template literals | ▶ | 5/5 | 4/5 | 4/5 | 3/5 | 3/5 | 0/5 | 0/5 | 0/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| RegExp "y" and "u" flags | ▶ | 5/5 | 3/5 | 3/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 5/5 | 5/5 | 5/5 | 2/5 | 5/5 | 5/5 | 5/5 |
| destructuring, declarations | ▶ | 21/22 | 20/22 | 21/22 | 19/22 | 15/22 | 0/22 | 0/22 | 0/22 | 0/22 | 21/22 | 22/22 | 19/22 | 21/22 | 21/22 | 21/22 |
| destructuring, assignment | ▶ | 23/24 | 23/24 | 24/24 | 17/24 | 19/24 | 0/24 | 0/24 | 0/24 | 0/24 | 23/24 | 24/24 | 21/24 | 23/24 | 23/24 | 23/24 |
| destructuring, parameters | ▶ | 19/23 | 19/23 | 20/23 | 18/23 | 15/23 | 0/23 | 0/23 | 0/23 | 0/23 | 22/23 | 23/23 | 18/23 | 19/23 | 20/23 | 20/23 |
| Unicode code point escapes | ▶ | 1/2 | 1/2 | 1/2 | 1/2 | 1/2 | 0/2 | 0/2 | 0/2 | 2/2 | 2/2 | 2/2 | 1/2 | 1/2 | 1/2 | 1/2 |
| new.target | ▶ | 2/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 1/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 |

# ECMAScript v6

https://kangax.github.io/compat-table/es6/

| | | Compilers/polyfills | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 92% | 56% | 71% | 48% | 59% | 18% | 5% | 11% | 83% | 93% | 95% | 86% | 92% |
| Feature name | Current browser | Traceur | Babel + core-js[2] | Closure | TypeScript + core-js | es6-shim | KQ 4.14[3] | IE 11 | Edge 13[4] | Edge 14[4] | Edge 15[4] | FF 45 ESR | FF 50 |
| **Functions** | | | | | | | | | | | | | |
| arrow functions | 13/13 | 11/13 | 9/13 | 10/13 | 9/13 | 0/13 | 0/13 | 0/13 | 13/13 | 13/13 | 13/13 | 13/13 | 13/13 |
| class | 24/24 | 17/24 | 19/24 | 13/24 | 19/24 | 0/24 | 0/24 | 0/24 | 24/24 | 24/24 | 24/24 | 24/24 | 24/24 |
| super | 8/8 | 7/8 | 4/8 | 5/8 | 7/8 | 0/8 | 0/8 | 0/8 | 8/8 | 8/8 | 8/8 | 8/8 | 8/8 |
| generators | 25/27 | 24/27 | 24/27 | 16/27 | 0/27 | 0/27 | 0/27 | 0/27 | 27/27 | 27/27 | 27/27 | 25/27 | 25/27 |
| **Built-ins** | | | | | | | | | | | | | |
| typed arrays | 45/46 | 0/46 | 45/46 | 0/46 | 45/46 | 0/46 | 8/46 | 16/46 | 44/46 | 46/46 | 46/46 | 42/46 | 45/46 |
| Map | 18/19 | 14/19 | 19/19 | 14/19 | 19/19 | 15/19 | 0/19 | 8/19 | 18/19 | 18/19 | 19/19 | 18/19 | 18/19 |
| Set | 18/19 | 14/19 | 19/19 | 14/19 | 19/19 | 15/19 | 0/19 | 8/19 | 18/19 | 18/19 | 19/19 | 18/19 | 18/19 |
| WeakMap | 11/12 | 6/12 | 12/12 | 9/12 | 12/12 | 0/12 | 0/12 | 6/12 | 11/12 | 11/12 | 12/12 | 10/12 | 11/12 |
| WeakSet | 10/11 | 5/11 | 11/11 | 8/11 | 11/11 | 0/11 | 0/11 | 0/11 | 10/11 | 10/11 | 11/11 | 9/11 | 10/11 |
| Proxy[17] | 34/34 | 0/34 | 0/34 | 0/34 | 0/34 | 0/34 | 0/34 | 0/34 | 34/34 | 34/34 | 34/34 | 30/34 | 34/34 |
| Reflect[18] | 20/20 | 0/20 | 14/20 | 14/20 | 18/20 | 14/20 | 0/20 | 0/20 | 20/20 | 20/20 | 20/20 | 20/20 | 20/20 |
| Promise | 8/8 | 4/8 | 8/8 | 7/8 | 8/8 | 7/8 | 0/8 | 0/8 | 8/8 | 8/8 | 8/8 | 8/8 | 8/8 |
| Symbol | 12/12 | 4/12 | 9/12 | 2/12 | 8/12 | 2/12 | 0/12 | 0/12 | 12/12 | 12/12 | 12/12 | 12/12 | 12/12 |
| well-known symbols[19] | 22/26 | 1/26 | 14/26 | 1/26 | 15/26 | 0/26 | 0/26 | 0/26 | 9/26 | 10/26 | 14/26 | 8/26 | 22/26 |
| **Built-in extensions** | | | | | | | | | | | | | |
| Object static methods | 4/4 | 3/4 | 3/4 | 2/4 | 3/4 | 2/4 | 1/4 | 1/4 | 4/4 | 4/4 | 4/4 | 4/4 | 4/4 |
| function "name" property | 12/17 | 0/17 | 8/17 | 0/17 | 3/17 | 0/17 | 3/17 | 0/17 | 14/17 | 16/17 | 16/17 | 10/17 | 12/17 |

# ECMAScript v6

https://kangax.github.io/compat-table/es6/

| | | | | | | Servers/runtimes | | | | | | | | Mobile | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 97% | 97% | 54% | 100% | 100% | 100% | 4% | 66% | 96% | 59% | 21% | 52% | 97% | 97% | 5% | 10% | 25% | 54% |
| CH 56, OP 43[1] | CH 57, OP 44[1] | SF 9 | SF 10 | SF TP | WK | PJS | Echo JS | XS6 | JXA | Node 0.12[5] | Node 4[5] | Node 6.5[5] | Node 7[5] | AN 4.4 | AN 5.0 | AN 5.1 | iOS 9 |
| 13/13 | 13/13 | 0/13 | 13/13 | 13/13 | 13/13 | 0/13 | 7/13 | 12/13 | 0/13 | 0/13 | 9/13 | 13/13 | 13/13 | 0/13 | 0/13 | 0/13 | 0/13 |
| 24/24 | 24/24 | 16/24 | 24/24 | 24/24 | 24/24 | 0/24 | 22/24 | 24/24 | 18/24 | 0/24 | 0/24 | 24/24 | 24/24 | 0/24 | 0/24 | 0/24 | 16/24 |
| 8/8 | 8/8 | 6/8 | 8/8 | 8/8 | 8/8 | 0/8 | 6/8 | 8/8 | 7/8 | 0/8 | 0/8 | 8/8 | 8/8 | 0/8 | 0/8 | 0/8 | 6/8 |
| 27/27 | 27/27 | 0/27 | 27/27 | 27/27 | 27/27 | 0/27 | 16/27 | 27/27 | 0/27 | 0/27 | 20/27 | 27/27 | 27/27 | 0/27 | 0/27 | 16/27 | 0/27 |
| 46/46 | 46/46 | 18/46 | 46/46 | 46/46 | 46/46 | 18/46 | 37/46 | 46/46 | 46/46 | 23/46 | 43/46 | 46/46 | 46/46 | 19/46 | 19/46 | 23/46 | 18/46 |
| 19/19 | 19/19 | 18/19 | 19/19 | 19/19 | 19/19 | 0/19 | 17/19 | 19/19 | 18/19 | 13/19 | 17/19 | 19/19 | 19/19 | 0/19 | 0/19 | 16/19 | 18/19 |
| 19/19 | 19/19 | 18/19 | 19/19 | 19/19 | 19/19 | 0/19 | 18/19 | 19/19 | 18/19 | 13/19 | 17/19 | 19/19 | 19/19 | 0/19 | 0/19 | 16/19 | 18/19 |
| 12/12 | 12/12 | 12/12 | 12/12 | 12/12 | 12/12 | 0/12 | 9/12 | 11/12 | 11/12 | 7/12 | 11/12 | 12/12 | 12/12 | 0/12 | 6/12 | 9/12 | 12/12 |
| 11/11 | 11/11 | 11/11 | 11/11 | 11/11 | 11/11 | 0/11 | 9/11 | 10/11 | 10/11 | 6/11 | 10/11 | 11/11 | 11/11 | 0/11 | 5/11 | 8/11 | 11/11 |
| 34/34 | 34/34 | 0/34 | 34/34 | 34/34 | 34/34 | 0/34 | 27/34 | 34/34 | 0/34 | 0/34 | 0/34 | 34/34 | 34/34 | 0/34 | 0/34 | 0/34 | 0/34 |
| 20/20 | 20/20 | 0/20 | 20/20 | 20/20 | 20/20 | 0/20 | 14/20 | 16/20 | 0/20 | 0/20 | 0/20 | 20/20 | 20/20 | 0/20 | 0/20 | 0/20 | 0/20 |
| 8/8 | 8/8 | 6/8 | 8/8 | 8/8 | 8/8 | 0/8 | 4/8 | 8/8 | 0/8 | 4/8 | 7/8 | 8/8 | 8/8 | 0/8 | 5/8 | 5/8 | 6/8 |
| 12/12 | 12/12 | 11/12 | 12/12 | 12/12 | 12/12 | 0/12 | 4/12 | 12/12 | 12/12 | 9/12 | 10/12 | 12/12 | 12/12 | 0/12 | 0/12 | 10/12 | 11/12 |
| 25/26 | 26/26 | 3/26 | 26/26 | 26/26 | 26/26 | 0/26 | 23/26 | 25/26 | 22/26 | 2/26 | 3/26 | 26/26 | 26/26 | 0/26 | 0/26 | 3/26 | 3/26 |

# Principales novedades de ES6

## Destructuring

- Hasta ahora hemos usado éstas:

```
1  // Las conocidas:
2
3  var miArray = new Array();
4  miArray[0] = "Node Hoteles";
5  miArray[1] = 12;
6  miArray[2] = [2, "Array en línea"];
7
8  var miObjeto = {
9      propiedad: "Array asociativo",
10     otraProp:  "Sin habitaciones"
11 };
```

# Principales novedades de ES6

## Destructuring

- Ahora tenemos nuevas formas de asignar valores a Arrays y a Objetos.

```
 var [a, b] = ["hola", "mundo"];
console.log(a); // "hola"
console.log(b); // "mundo"

var obj = { nombre: "Carlos", apellido: "Azaustre" };
var { nombre, apellido } = obj;
console.log(nombre); // "Carlos"
```

https://carlosazaustre.es/blog/ecmascript-6-el-nuevo-estandar-de-javascript/

# Principales novedades de ES6

## Destructuring

- ¿No te ha estallado el cerebro todavía? Pues mira esto:

```
 var foo = function() {
     return ["175", "75"];
};
var [estatura, peso] = foo();
console.log(estatura); //175
console.log(peso); //75
```

# Principales novedades de ES6

## Función Arrow

- ¿Cuántas veces has programado un código con una estructura similar a la siguiente?

```
 // ES5
// Imaginemos una variable data que incluye un array de objectos
var data = [{...}, {...}, {...}, ...];
data.forEach(function(elem){
    // Tratamos el elemento
    console.log(elem)
});
```

# Principales novedades de ES6

## Función Arrow

- Con la función arrow => de ES6, el código anterior se sustituiría por:

```
 //ES6
var data = [{...}, {...}, {...}, ...];
data.forEach(elem => {
    console.log(elem);
});
```

- Mucho más limpio y claro. CoffeeScript (un metalenguaje que compila a JavaScript) usa algo parecido.

# Principales novedades de ES6

## Función Arrow

- Incluso la podemos utilizar así:

```
 // ES5
var miFuncion = function(num) {
    return num + num;
}
```

```
 // ES6
var miFuncion = (num) => num + num;
```

# Principales novedades de ES6

## Clases

- Ahora JavaScript tiene clases, muy parecidas las funciones constructoras de objectos que realizabamos en el estándar anterior, pero ahora bajo el paradigma de clases, con todo lo que eso conlleva, como por ejemplo, **herencia**.

# Principales novedades de ES6

## Clases

- Herencia:

```
class LibroTecnico extends Libro {
    constructor(tematica, paginas) {
        super(tematica, paginas);
        this.capitulos = [];
        this.precio = "";
        // ...
    }
    metodo() {
        // ...
    }
}
```

# Principales novedades de ES6

## Palabra this

- La variable this muchas veces se vuelve un dolor de cabeza.
- Antíguamente teníamos que cachearlo en otra variable ya que solo hace referencia al contexto en el que nos encontremos.

# Principales novedades de ES6

## Palabra **this**

- Por ejemplo, en el siguiente código si no hacemos **var that = this** dentro de la función **document.addEventListener**, **this** haría referencia a la función que pasamos por Callback y no podríamos llamar a **foo()**

# Principales novedades de ES6

## Palabra this

```
//ES3
var obj = {
    foo : function() {...},
    bar : function() {
        var that = this;
        document.addEventListener("click", function(e) {
            that.foo();
        });
    }
}
```

# Principales novedades de ES6

## Palabra this

- Con ECMAScript5 la cosa cambió un poco, y gracias al método **bind** podíamos indicarle que **this** hace referencia a un contexto y no a otro.

```
 //ES5
var obj = {
    foo : function() {...},
    bar : function() {
        document.addEventListener("click", function(e) {
            this.foo();
        }.bind(this));
    }
}
```

# Principales novedades de ES6

## Palabra this

- Ahora con ES6 y la función Arrow => la cosa es todavía más visual y sencilla.

```
 //ES6
var obj = {
    foo : function() {...},
    bar : function() {
        document.addEventListener("click", (e) => this.foo());
    }
}
```

# Principales novedades de ES6

## let y const

- Ahora podemos declarar variables con let en lugar de var si no queremos que sean accesibles más allá de un ámbito.
- Por ejemplo:

# Principales novedades de ES6

## let y const

```
 //ES5
(function() {
    console.log(x); // x no está definida aún.
    if(true) {
        var x = "hola mundo";
    }
    console.log(x);
    // Imprime "hola mundo", porque "var" hace que sea global
    // a la función;
})();
```

# Principales novedades de ES6

## let y const

```
//ES6
(function() {
    if(true) {
        let x = "hola mundo";
    }
    console.log(x);
    //Da error, porque "x" ha sido definida dentro del "if"
})();
```

# Principales novedades de ES6

## Template Strings

- También podemos tener String multilínea sin necesidad de concatenarlos con +.

```
 //ES5
var saludo = "ola " +
"que " +
"ase ";

//ES6
var saludo = "ola
que
ase";

console.log("hola
que
ase");
```

# Principales novedades de ES6

## Template Strings

- Con ES6 podemos interpolar Strings de una forma más sencilla que como estábamos haciendo hasta ahora.

```
 //ES6
let nombre1 = "JavaScript";
let nombre2 = "awesome";
console.log(`Sólo quiero decir que ${nombre1} is ${nombre2`);
// Solo quiero decir que JavaScript is awesome
```

# Principales novedades de ES6

## let y const

- Ahora con const podemos crear constantes que sólo se puedan leer...

```
(function() {
    const PI;
    PI = 3.15;
    // ERROR, porque ha de asignarse un valor en la
    // declaración
})();
```

# Principales novedades de ES6

## let y const

- … y no modificar a lo largo del código.

```
(function() {
    const PI = 3.15;
    PI = 3.14159;
    // ERROR de nuevo, porque es sólo-lectura
})();
```

# Principales novedades de ES6

## Valores por defecto

- Otra novedad es asignar valores por defecto a las variables que se pasan por parámatros en las funciones.
- Antes teníamos que comprobar si la variable ya tenía un valor.
- Ahora con ES6 se la podemos asignar según creemos la función.

# Principales novedades de ES6

## Valores por defecto

- 

```
 //ES5
function(valor) {
    valor = valor || "foo";
}

//ES6
function(valor = "foo") {...};
```

# Principales novedades de ES6

## Módulos

- A esto lo llamo un browserify nativo.
- Ahora JavaScript se empieza a parecer a lenguajes como Python o Ruby.
- Llamamos a las funciones desde los propios Scripts, sin tener que importarlos en el HTML, si usamos JavaScript en el navegador.

# Principales novedades de ES6

## Módulos

- 

```
 //File: lib/person.js
module "person" {
    export function hello(nombre) {
        return nombre;
    }
}
```

# Principales novedades de ES6

## Módulos

- Y para importar en otro fichero:

```
 //File: app.js
import { hello } from "person";
var app = {
    foo: function() {
        hello("Carlos");
    }
}
export app;
```

# Principales novedades de ES6

## Iteradores

- Un objeto es un iterador cuando sabe como acceder a los elementos de una colleccion, mientras mantiene un registro de su posición actual dentro de esa secuencia.
- En JavaScript un iterador es un objeto que proporciona un método next() que devuelve el siguiente elemento en la secuencia.
- Este método devuelve un objeto con dos propiedades: done y value.

# Principales novedades de ES6

## Iteradores

- Una vez creado, un objeto iterador puede utilizarse explícitamente llamando repetidamente al método  next().

```
function makeIterator(array){
    var nextIndex = 0;

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

# Principales novedades de ES6

## Iteradores

- Una vez inicializado, puede llamar al método next() para acceder a su vez,  a parejas de valores clave-valor desde el objeto:

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

# Principales novedades de ES6

## Generators

- Procesar cada uno de los elementos en una colección es un tipo de operación muy común.
- JavaScript proporciona diversas formas de iterar a través de los elementos de una colección, desde simples bucles for hasta map(), y filter().

# Principales novedades de ES6

## Generators

- Los iteradores y los generadores acercan el concepto de iteración directamente al núcleo del lenguaje y proporcionan un mecanismo para personalizar el comportamiento de los bucles for...of.
- Para más información, también puedes ver:
  - for...of *(se verá después)*.
  - function* y Generator *(se verá después)*.
  - yield y yield* *(se verá después)*.
  - Protocolos de iteración
  - Generador de comprensiones

# Principales novedades de ES6

## Generators

- Los generadores son funciones de las que se puede salir y volver a entrar.
- Su contexto (asociación de variables) será conservado entre las reentradas.
- La llamada a una función generadora no ejecuta su cuerpo inmediatamente; se devuelve un objeto iterador para la función en su lugar

# Principales novedades de ES6

## Generators

- La declaración function* (la palabra clave function seguida de una asterisco) define una función generadora, que devuelve un objeto Generator.
- También puedes definir funciones generadoras usando el constructor GeneratorFunction y una function* expression.

# Principales novedades de ES6

## Generators

### Sintaxis

```
function* nombre([param[, param[, ... param]]]) {
   instrucciones
}
```

- **nombre**
  - El nombre de la función.
- **param**
  - El nombre de los argumentos que se le van a pasar a la función. Una función puede tener hasta 255 argumentos.
- **instrucciones**
  - Las instrucciones que componen el cuerpo de la función.

# Principales novedades de ES6

## Generators

Ejemplo

```
function* idMaker(){
  var index = 0;
  while(index < 3)
    yield index++;
}

var gen = idMaker();

console.log(gen.next().value); // 0
console.log(gen.next().value); // 1
console.log(gen.next().value); // 2
console.log(gen.next().value); // undefined
// ...
```

# Principales novedades de ES6

## Generators

Ejemplo con yield*

```
function* anotherGenerator(i) {
  yield i + 1;
  yield i + 2;
  yield i + 3;
}

function* generator(i){
  yield i;
  yield* anotherGenerator(i);
  yield i + 10;
}

var gen = generator(10);

console.log(gen.next().value); // 10
console.log(gen.next().value); // 11
console.log(gen.next().value); // 12
console.log(gen.next().value); // 13
console.log(gen.next().value); // 20
```

# Principales novedades de ES6

## Generators

Ejemplo con yield*

```
function* anotherGenerator(i) {
  yield i + 1;
  yield i + 2;
  yield i + 3;
}

function* generator(i){
  yield i;
  yield* anotherGenerator(i);
  yield i + 10;
}

var gen = generator(10);

console.log(gen.next().value); // 10
console.log(gen.next().value); // 11
console.log(gen.next().value); // 12
console.log(gen.next().value); // 13
console.log(gen.next().value); // 20
```

# Principales novedades de ES6

## Iterables

- Un objeto es iterable si define el comportamiento de su iteración, como por ejemplo qué valores que recorren un bucle **for..of**.
- Algunos tipos de datos, como Array or Map, incorporan esta caracteristica por defecto, mientras que otras no (como Object).

# Principales novedades de ES6

## Iterables

- The for...of statement creates a loop iterating over iterable objects (including Array, Map, Set, String, TypedArray, arguments object and so on), invoking a custom iteration hook with statements to be executed for the value of each distinct property.

# Principales novedades de ES6

## Iterables

## Syntax

```
for (variable of iterable) {
  statement
}
```

**`variable`**

On each iteration a value of a different property is assigned to *variable*.

**`iterable`**

Object whose enumerable properties are iterated.

# Principales novedades de ES6

## Iterables

## Examples

### Iterating over an Array

```
let iterable = [10, 20, 30];

for (let value of iterable) {
  console.log(value);
}
// 10
// 20
// 30
```

# Principales novedades de ES6

## Iterables

- Se puede usar const en vez de let también, pàra no moficar la variable dentro del bloque.

```
let iterable = [10, 20, 30];

for (const value of iterable) {
  console.log(value);
}
// 10
// 20
// 30
```

# Principales novedades de ES6

## Iterables

- Iterating over a String

```
1  let iterable = "boo";
2
3  for (let value of iterable) {
4    console.log(value);
5  }
6  // "b"
7  // "o"
8  // "o"
```

# Principales novedades de ES6

## Iterables

- Iterating over a TypedArray

```
let iterable = new Uint8Array([0x00, 0xff]);

for (let value of iterable) {
  console.log(value);
}
// 0
// 255
```

# Principales novedades de ES6

## Iterables

- Iterating over a Map

```
let iterable = new Map([["a", 1], ["b", 2], ["c", 3]]);

for (let entry of iterable) {
  console.log(entry);
}
// [a, 1]
// [b, 2]
// [c, 3]

for (let [key, value] of iterable) {
  console.log(value);
}
// 1
// 2
// 3
```

# Principales novedades de ES6

## Iterables

- Iterating over a Set

```
let iterable = new Set([1, 1, 2, 2, 3, 3]);

for (let value of iterable) {
  console.log(value);
}
// 1
// 2
// 3
```

# Principales novedades de ES6

## Iterables

- Iterating over generators

```
function* fibonacci() { // a generator function
  let [prev, curr] = [1, 1];
  while (true) {
    [prev, curr] = [curr, prev + curr];
    yield curr;
  }
}

for (let n of fibonacci()) {
  console.log(n);
  // truncate the sequence at 1000
  if (n >= 1000) {
    break;
  }
}
```

# Principales novedades de ES6

## usar ES6

- Chrome es uno de los navegadores que está implementando las nuevas características de ES6 más rápidamente.
- De hecho se puede afirmar que todos los navegadores modernos soportan ECMAScript 6 casi por completo, y en breve también ES7.