


HENRY



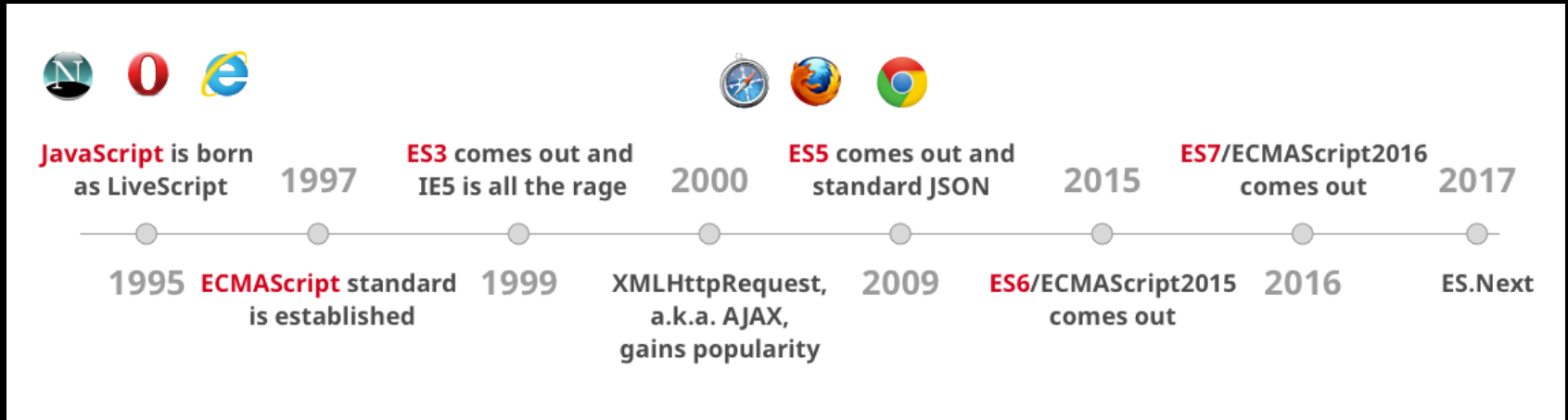
Text



ES6



ECMA



Ecma International es una organización internacional basada en membresías de estándares para la comunicación y la información.

La organización fue fundada en 1961 para estandarizar los sistemas computarizados en Europa. La membresía está abierta a las empresas que producen, comercializan o desarrollan sistemas computacionales o de comunicación en Europa.



Nuevas Features

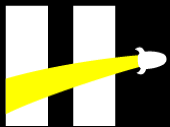
- Keywords *Let* y *Const*
- Funciones Flecha (Arrow Functions)
- Desestructuración (Destructuring)
- El operador Spread
- Literales de Template (Template Literals)

[Lista completa de features](#)



Let y Const

```
1  function f() {  
2    {  
3      let x;  
4      {  
5        // okay, block scoped name  
6        const x = "sneaky";  
7        // error, const  
8        x = "foo";  
9      }  
10     // error, already declared in block  
11     let x = "inner";  
12   }  
13 }
```



Let y var en Ciclo For

¿Qué pasaría si usamos let en vez de var?

```
1 var creaFuncion = function(){
2   var arreglo = [];
3   for ( var i=0; i < 3; i++){
4     arreglo.push(
5       function(){
6         console.log(i);
7       }
8     )
9   }
10  return arreglo;
11 }
12
13 var arr = creaFuncion();
14 arr[0]() // 3
15 arr[1]() // 3
16 arr[2]() // 3
```

<Demo />



Arrow Functions

```
1 // Cuerpos con Expresiones
2
3 var pares = impares.map(v => v + 1);
4 var nums = pares.map((v, i) => v + i);
5
6 // Cuerpos con Statements
7
8 nums.forEach(v => {
9   if (v % 5 === 0)
10     cincos.push(v);
11 });
12
13 // this
14 var bob = {
15   _name: "Bob",
16   _friends: [],
17   printFriends() {
18     this._friends.forEach(f =>
19       console.log(this._name + " knows " + f));
20   }
21 }
```



Class

```
1  class Persona {
2      constructor (nombre = 'Franco', apellido = "Etcheverri"){
3          this.nombre = nombre,
4          this.apellido = apellido
5      }
6
7      getNombre() {
8          return this.nombre + " " + this.apellido;
9      }
10 }
11
12 class Empleado extends Persona {
13     constructor (nombre, apellido, empleo, sueldo){
14         super(nombre, apellido);
15         this.empleo = empleo;
16         this.sueldo = sueldo;
17     }
18
19     getEmpleado() {
20         return this.empleo + "($ " + this.sueldo + ")";
21     }
22 }
23
```




Object Literals

```
1 var obj = {
2   // __proto__
3   __proto__: theProtoObj, //extiende el prototipo
4   propiedad, // atajo para propiedad:propiedad
5   // Methods
6   toString() {
7     // Super calls
8     return "d " + super.toString();
9   },
10  // Computed (dynamic) property names
11  [ 'prop_' + (() => 42)() ]: 42
12 };
```



Template Strings

```
1 // Basic literal string creation
2 `In JavaScript '\n' is a line-feed.`
3
4 // Multiline strings
5 `In JavaScript this is
6   not legal.`
7
8 // String interpolation
9 var name = "Bob", time = "today";
10 `Hello ${name}, how are you ${time}?`
11
12 // Construct an HTTP request prefix is used to interpret
13 // the replacements and construction
14 POST`http://foo.org/bar?a=${a}&b=${b}
15     Content-Type: application/json
16     X-Credentials: ${credentials}
17     { "foo": ${foo},
18       "bar": ${bar}}`(myOnReadyStateChangeHandler);
```



Destructuring

```
1 // list matching
2 var [a, , b] = [1,2,3];
3
4 // object matching
5 var { op: a, lhs: { op: b }, rhs: c }
6     = getASTNode()
7
8 // object matching shorthand
9 // binds `op`, `lhs` and `rhs` in scope
10 var {op, lhs, rhs} = getASTNode()
11
12 // Can be used in parameter position
13 function g({name: x}) {
14     console.log(x);
15 }
16 g({name: 5})
17
18 // Fail-soft destructuring
19 var [a] = [];
20 a === undefined;
21
22 // Fail-soft destructuring with defaults
23 var [a = 1] = [];
24 a === 1;
```



Default + Rest + Spread

```
1 function f(x, y=12) {  
2   // y is 12 if not passed (or passed as undefined)  
3   return x + y;  
4 }  
5 f(3) == 15  
6  
7 function f(x, ...y) {  
8  
9   // y is an Array  
10  return x * y.length;  
11 }  
12 f(3, "hello", true) == 6  
13  
14 function f(x, y, z) {  
15   return x + y + z;  
16 }  
17 // Pass each elem of array as argument  
18 f(...[1,2,3]) == 6
```



Promises

```
1 function timeout(duration = 0) {
2   return new Promise((resolve, reject) => {
3     setTimeout(resolve, duration);
4   })
5 }
6
7 var p = timeout(1000).then(() => {
8   return timeout(2000);
9 }).then(() => {
10   throw new Error("hmm");
11 }).catch(err => {
12   return Promise.all([timeout(100), timeout(200)]);
13 })
```



Compatibilidad

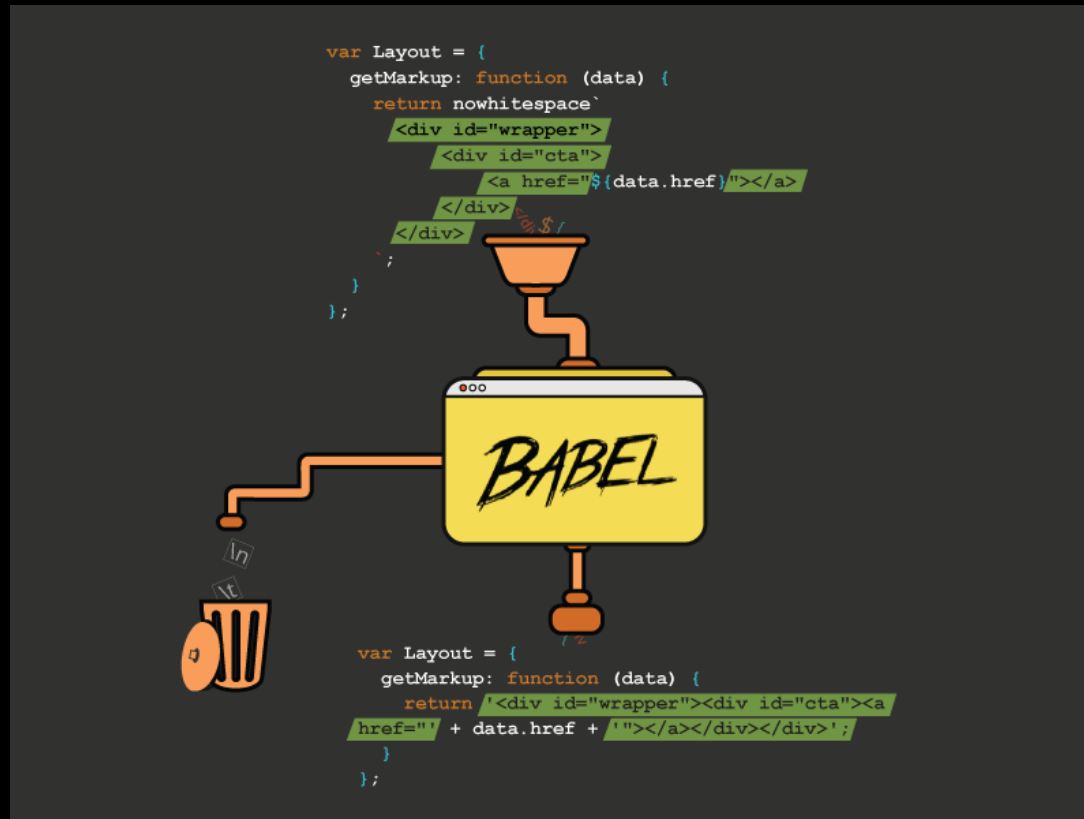
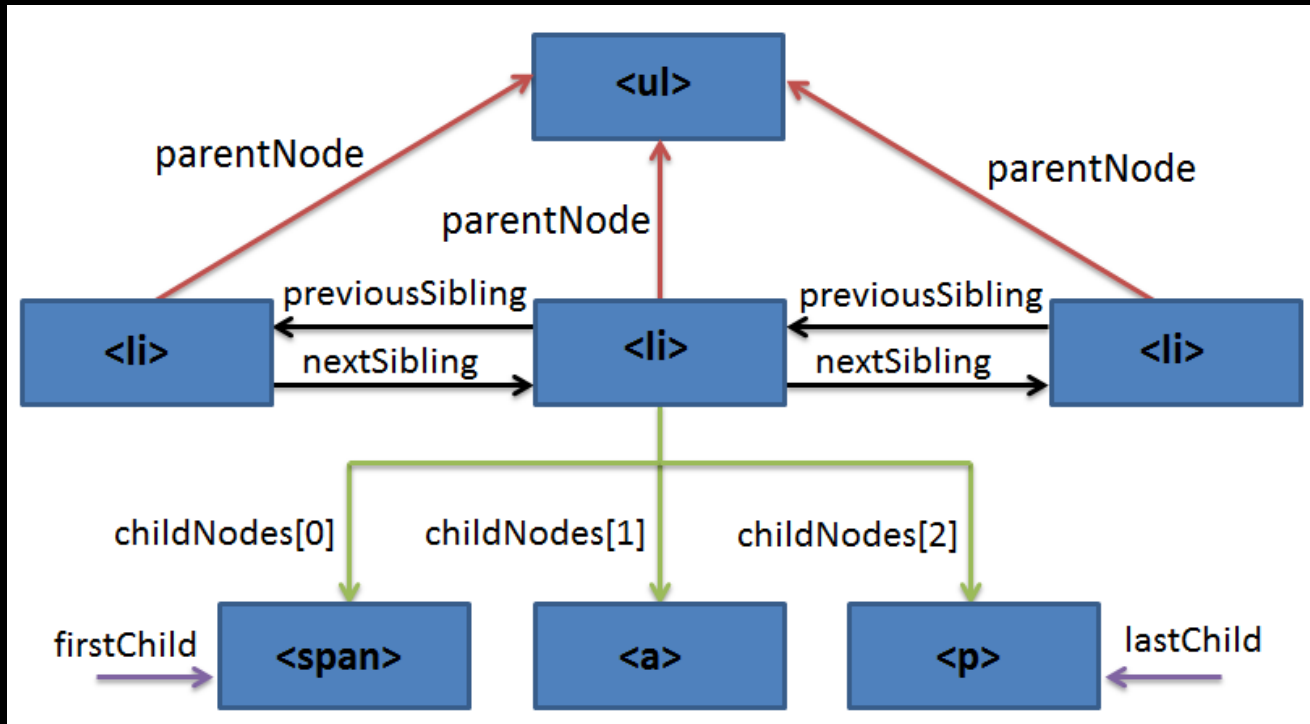


Tabla de compatibilidad



Homework



Github



Homework

