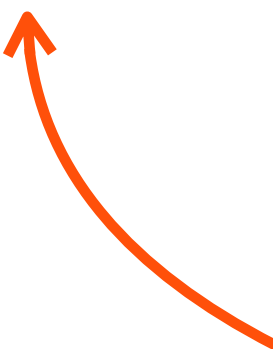


ES6



new features for JavaScript

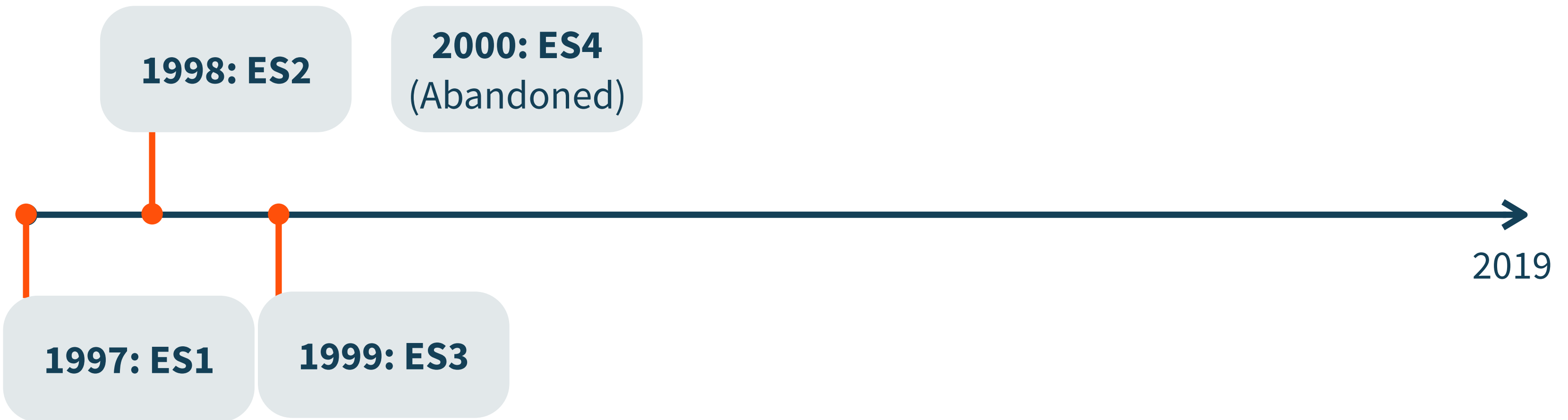
WHAT IS ES6?

ES (ECMAScript) is a **scripting language standard**.

JavaScript **implements** ECMAScript.

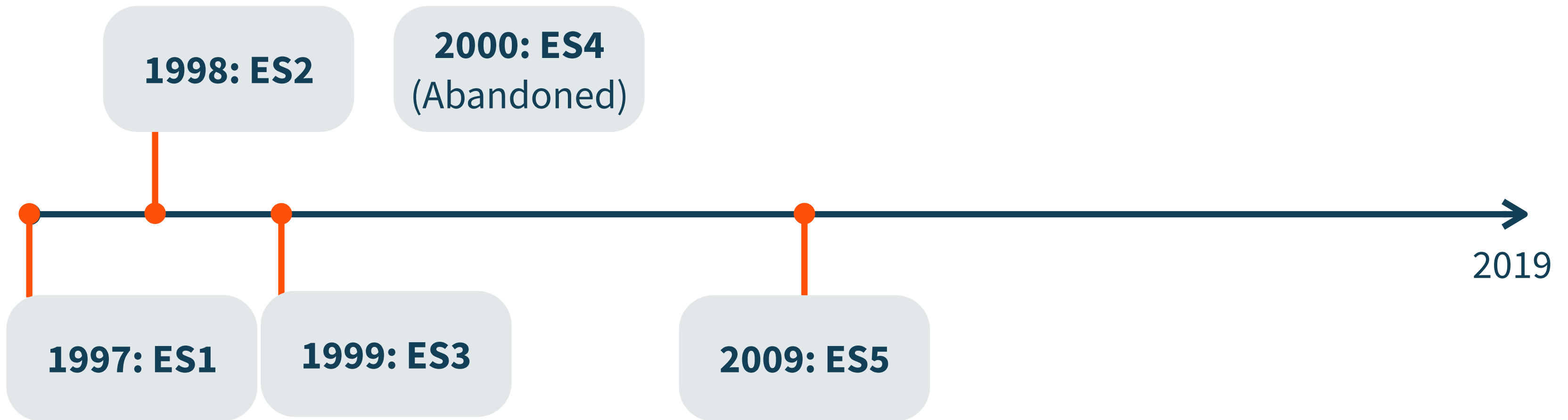
ES6 means the **6th Edition of ECMAScript**.

Timeline



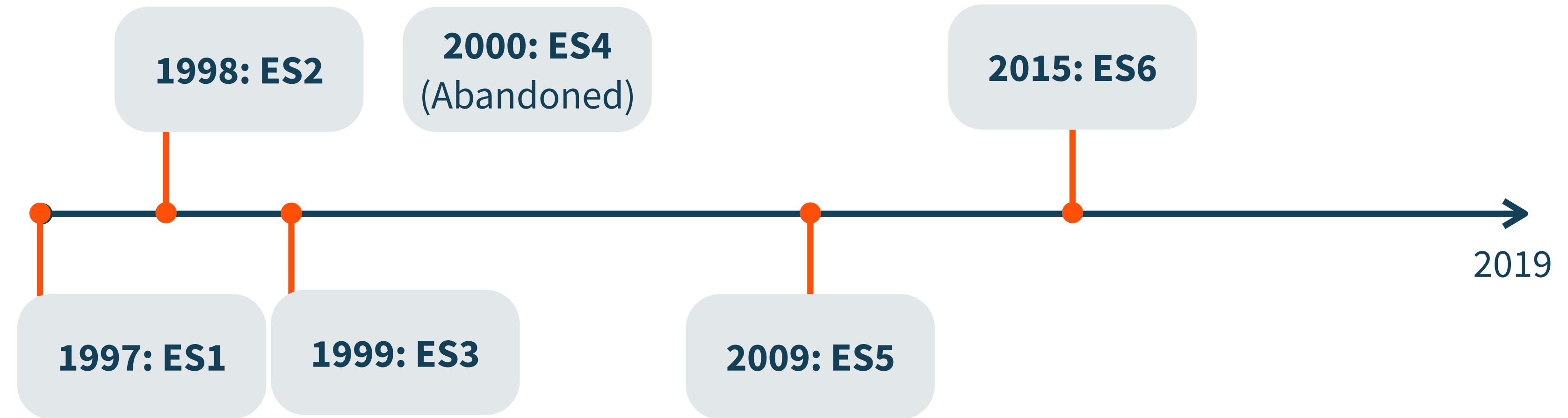
Started in 1997, yearly updates up to 2000

Timeline



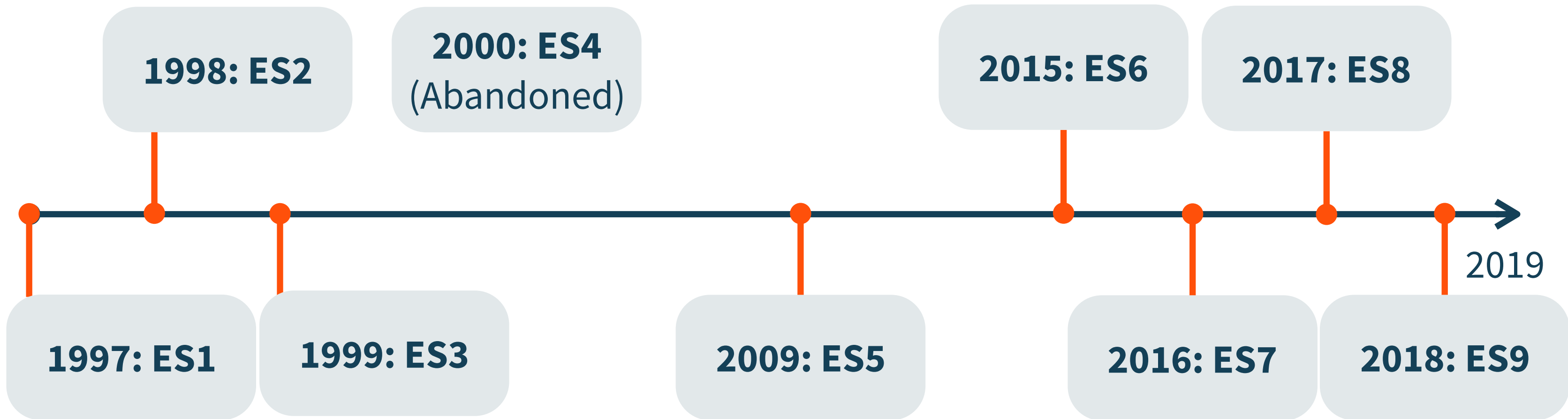
10 years later, ES5. Not much new stuff.

Timeline



6 years later, ES6! Plethora of new features!

Timeline



Yearly updates since ES6.

New, Exciting Features Every Year!

ES6 FEATURES

Let & Const

Rest & Spread

Arrow Functions

New Array Methods

Default Parameters

Classes

Template Literals

Modules

Destructuring

and more!

LET

before

```
var x = 1;

if (x === 1) {
  var x = 2;
  console.log(x); // 2
}

console.log(x); // 2
```

after

```
let x = 1;

if (x === 1) {
  let x = 2;
  console.log(x); // 2
}

console.log(x); // 1
```

CONST

before

```
var number = 42;  
  
try {  
  number = 99;  
} catch(err) {  
  console.log(err); // no error  
}  
  
console.log(number); // 99
```

after

```
const number = 42;  
  
try {  
  number = 99;  
} catch(err) {  
  console.log(err); // TypeError  
}  
  
console.log(number); // 42
```

ARROW FUNCTIONS

before

```
const arr = [1, 2, 3];  
const squares = arr.map(function (x) { return x * x });
```

after

```
const arr = [1, 2, 3];  
const squares = arr.map(x => x * x);
```

ARROW FUNCTIONS

before

```
function MyComponent() {  
  const button = document.getElementById('myButton');  
  const obj = this;  
  button.addEventListener('click', function () {  
    obj.handleClick();  
  });  
}
```

after

```
function MyComponent() {  
  const button = document.getElementById('myButton');  
  button.addEventListener('click', () => {  
    this.handleClick();  
  });  
}
```

DEFAULT PARAMETERS

before

```
function magic(x, y) {  
  x = x || 0;  
  y = y || 0;  
  ...  
}
```

after

```
function magic(x = 0, y = 0) {  
  ...  
}
```

TEMPLATE LITERALS

before

```
function greet(first, last) {  
  console.log('Hello' + first + ' ' + last + '!');  
}
```

after

```
function greet(first, last) {  
  console.log(`Hello ${first} ${last}!`);  
}
```


TEMPLATE LITERALS

before

```
var HTML5_SKELETON =  
  '<!doctype html>\n' +  
  '<html>\n' +  
  '  <head>\n' +  
  '    <meta charset="UTF-8">\n' +  
  '    <title></title>\n' +  
  '  </head>\n' +  
  '  <body>\n' +  
  '  </body>\n' +  
  '</html>\n';
```

after

```
const HTML5_SKELETON = `  
  <!doctype html>  
  <html>  
    <head>  
      <meta charset="UTF-8">  
      <title></title>  
    </head>  
    <body>  
    </body>  
  </html>`;
```

DESTRUCTURING

```
// Assignment
const numbers = [1, 2, 3];
const [one, two, three] = numbers;

// Swapping
let a = 1;
let b = 2;
[a, b] = [b, a];

// Ignoring
const foo = () => [1, 2, 3];
let [p, , q] = foo();

// Works with Objects too!
const obj = { bar: 42, baz: true };
const { bar, baz } = obj;

// Can also rename vars
const { bar: qux, baz: quux } = obj;
```


REST

```
function secret(first, ...others) {  
  console.log(first);  
  console.log(others);  
}  
secret(1, 2, 3, 4, 5);
```

accessing the "rest" of the arguments

SPREAD

before

```
var arr1 = ['a', 'b'];  
var arr2 = ['c'];  
var arr3 = ['d', 'e'];  
  
console.log(arr1.concat(arr2, arr3));  
// [ 'a', 'b', 'c', 'd', 'e' ]
```

after

```
const arr1 = ['a', 'b'];  
const arr2 = ['c'];  
const arr3 = ['d', 'e'];  
  
console.log([...arr1, ...arr2, ...arr3]);  
// [ 'a', 'b', 'c', 'd', 'e' ]
```

SPREAD

```
const arr = ['a', 'b', 'c']  
const copyOfArr = [...arr]
```

copying an array

NEW ARRAY METHODS

```
const arr = [1, 2, 3];  
arr.forEach(n => console.log(n)); // prints 1, 2, 3  
arr.map(n => n * n); // returns [1, 4, 9]  
arr.find(n => n > 2); // returns 3  
arr.some(n => n < 2); // returns true  
arr.every(n => n > 0); // returns true  
arr.reduce((n, acc) => acc + n, 0) // returns 6
```

and more...

CLASSES

before

```
function Person(name) {  
  this.name = name;  
}  
  
Person.prototype.describe = function () {  
  return 'Person called ' + this.name;  
};
```

after

```
class Person {  
  constructor(name) {  
    this.name = name;  
  }  
  describe() {  
    return `Person called ${this.name}`;  
  }  
}
```

CLASSES

before

```
function Employee(name, title) {  
  Person.call(this, name);  
  this.title = title;  
}  
  
Employee.prototype = Object.create(Person.prototype);  
Employee.prototype.constructor = Employee;  
  
Employee.prototype.describe = function () {  
  return Person.prototype.describe.call(this)  
    + ' (' + this.title + ')';  
};
```

after

```
class Employee extends Person {  
  constructor(name, title) {  
    super(name);  
    this.title = title;  
  }  
  describe() {  
    return `${super.describe()} (${this.title})`;  
  }  
}
```


MODULES

before

```
//----- lib.js -----  
var sqrt = Math.sqrt;  
function square(x) {  
    return x * x;  
}  
function diag(x, y) {  
    return sqrt(square(x) + square(y));  
}  
  
module.exports = {  
    sqrt: sqrt,  
    square: square,  
    diag: diag  
};  
  
//----- main.js -----  
var square = require('lib').square;  
var diag = require('lib').diag;  
  
console.log(square(11)); // 121  
console.log(diag(4, 3)); // 5
```

after

```
//----- lib.js -----  
export const sqrt = Math.sqrt;  
export const square = x => x * x;  
export const diag = (x, y) => sqrt(square(x) + square(y))  
  
//----- main.js -----  
import { square, diag } from 'lib';  
console.log(square(11)); // 121  
console.log(diag(4, 3)); // 5
```

MODULES

before

```
//----- myFunc.js -----  
module.exports = function () { ... };  
  
//----- main.js -----  
var myFunc = require('myFunc');  
myFunc();
```

after

```
//----- myFunc.js -----  
export default function () { ... }  
  
//----- main.js -----  
import myFunc from 'myFunc';  
myFunc();
```


RESOURCES

<http://exploringjs.com/es6/>

<https://github.com/lukehoban/es6features>

<http://es6-features.org>