~~Future~~ Web Applications

Javascript vNext and more

Who am I?

Ricardo Brandão

Full Stack Web Developer @ weListen
business • solutions

C#, Javascript, ServiceStack, RavenDB, Knockout, SQL

Agenda

ECMAScript

Babel

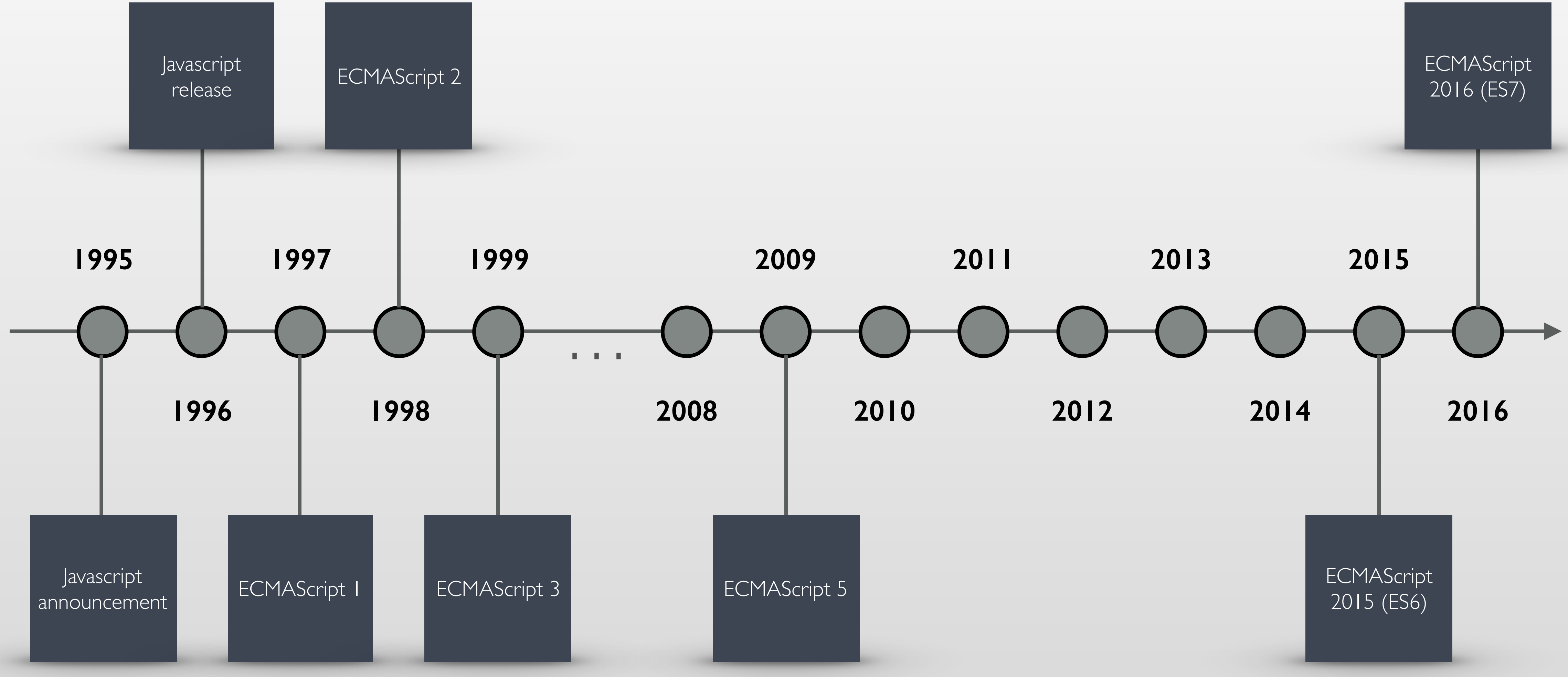
JSPM

Gulp

Demo

ECMAScript

ECMAScript Evolution



ECMAScript 2015

Formerly known as ES6

Specification is out!

Features

- Syntax
- Libraries
- Runtime

- arrow functions
- classes
- enhanced object literals
- template strings
- destructuring
- default + rest + spread
- let + const
- iterators + for...of
- generators
- unicode

- modules
- module loaders
- map + set + weakmap + weakset
- proxies
- symbols
- math + number + string + object APIs
- binary and octal literals
- reflect api
- tail calls

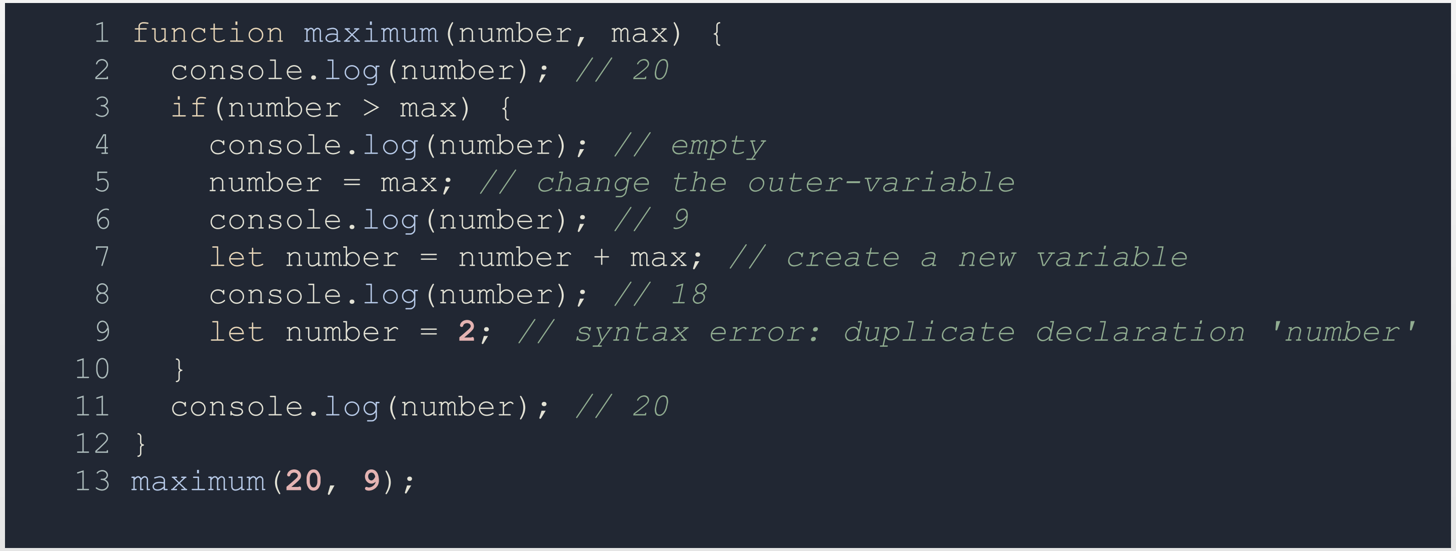
Variables

let

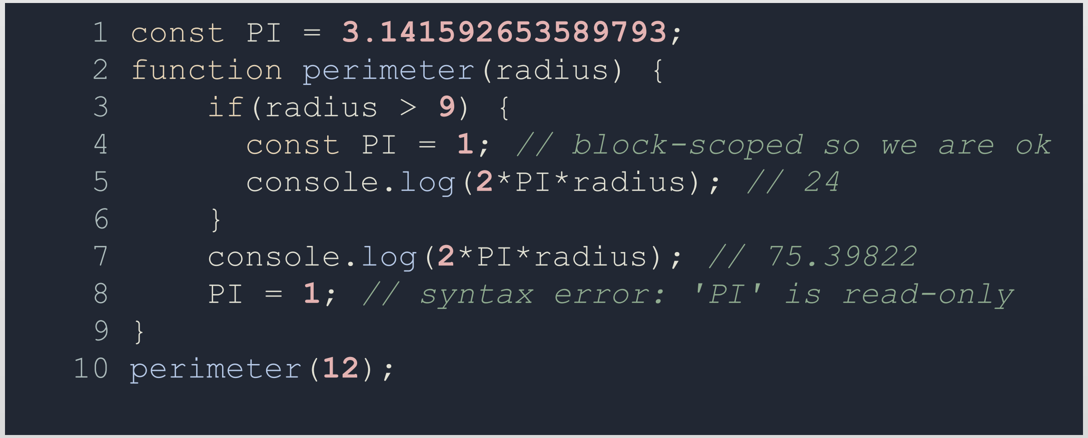
- block-scoped

const

- block-scoped
- defined once



```
function maximum(number, max) {
  console.log(number); // 20
  if(number > max) {
    console.log(number); // empty
    number = max; // change the outer-variable
    console.log(number); // 9
    let number = number + max; // create a new variable
    console.log(number); // 18
    let number = 2; // syntax error: duplicate declaration 'number'
  }
  console.log(number); // 20
}
maximum(20, 9);
```



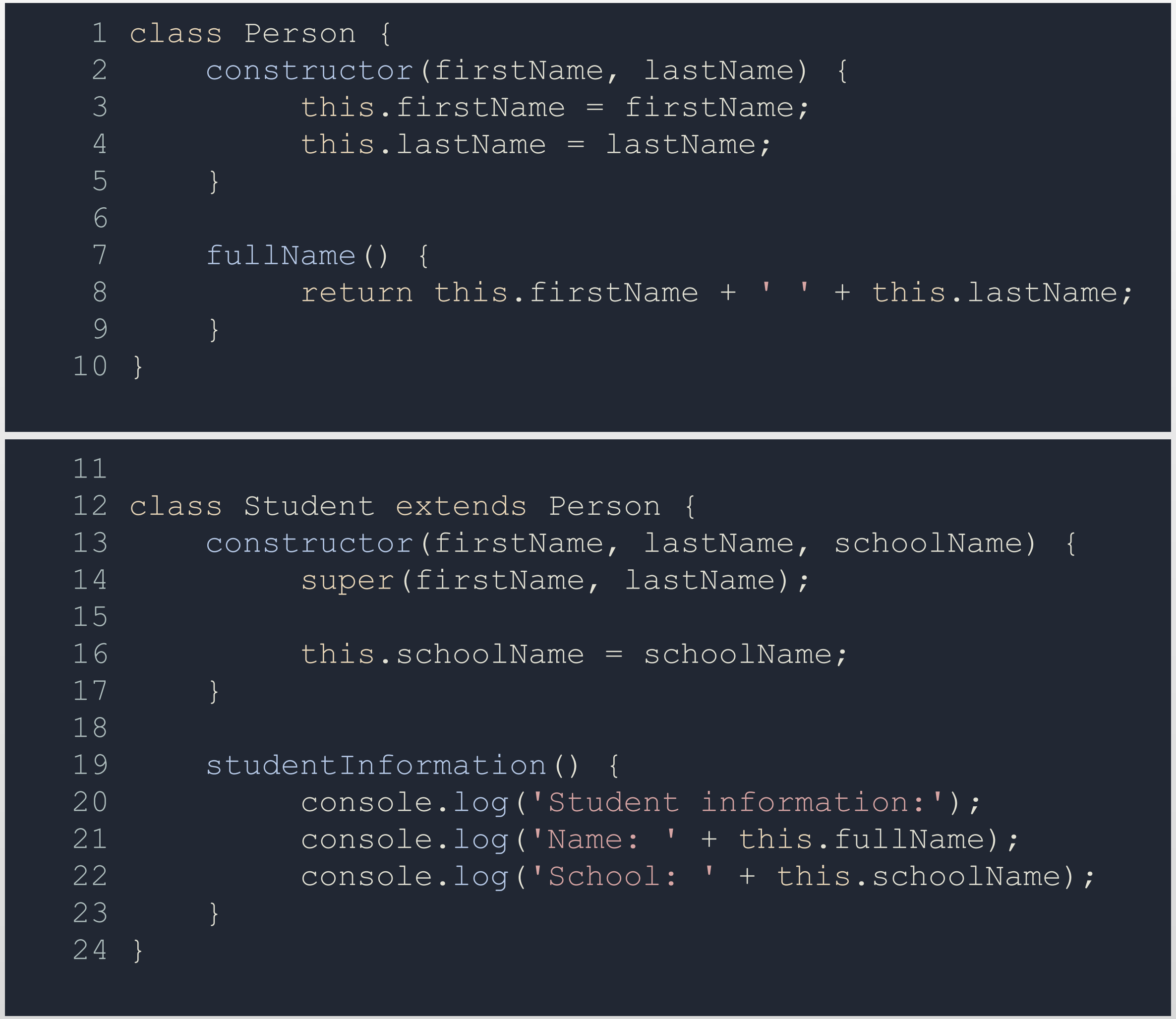
```
const PI = 3.141592653589793;
function perimeter(radius) {
    if(radius > 9) {
      const PI = 1; // block-scoped so we are ok
      console.log(2*PI*radius); // 24
    }
    console.log(2*PI*radius); // 75.39822
    PI = 1; // syntax error: 'PI' is read-only
}
perimeter(12);
```

Classes

Easier to use prototype-based OO pattern

Support

- class definition
- class constructor
- class methods
- inheritance



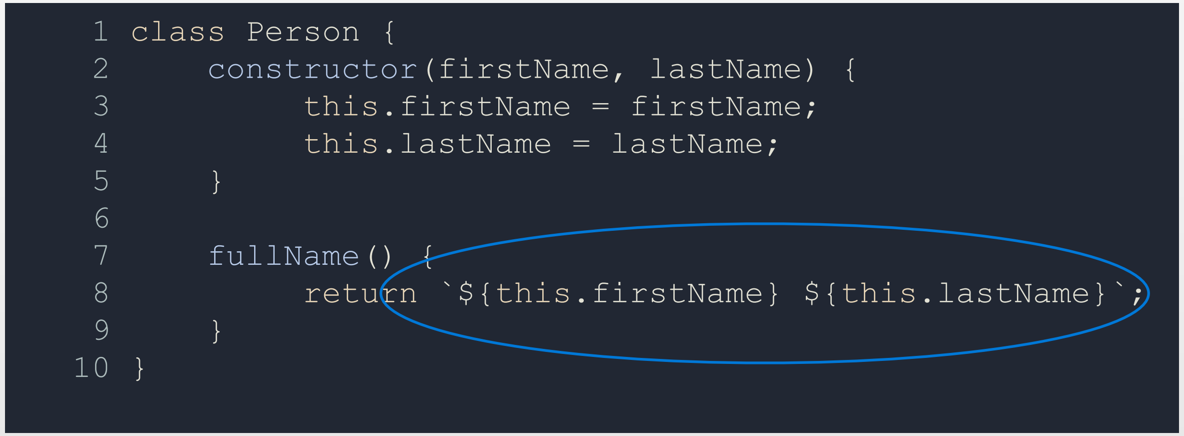
```
class Person {
    constructor(firstName, lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    fullName() {
        return this.firstName + ' ' + this.lastName;
    }
}

class Student extends Person {
    constructor(firstName, lastName, schoolName) {
        super(firstName, lastName);

        this.schoolName = schoolName;
    }

    studentInformation() {
        console.log('Student information:');
        console.log('Name: ' + this.fullName);
        console.log('School: ' + this.schoolName);
    }
}
```

Template Strings

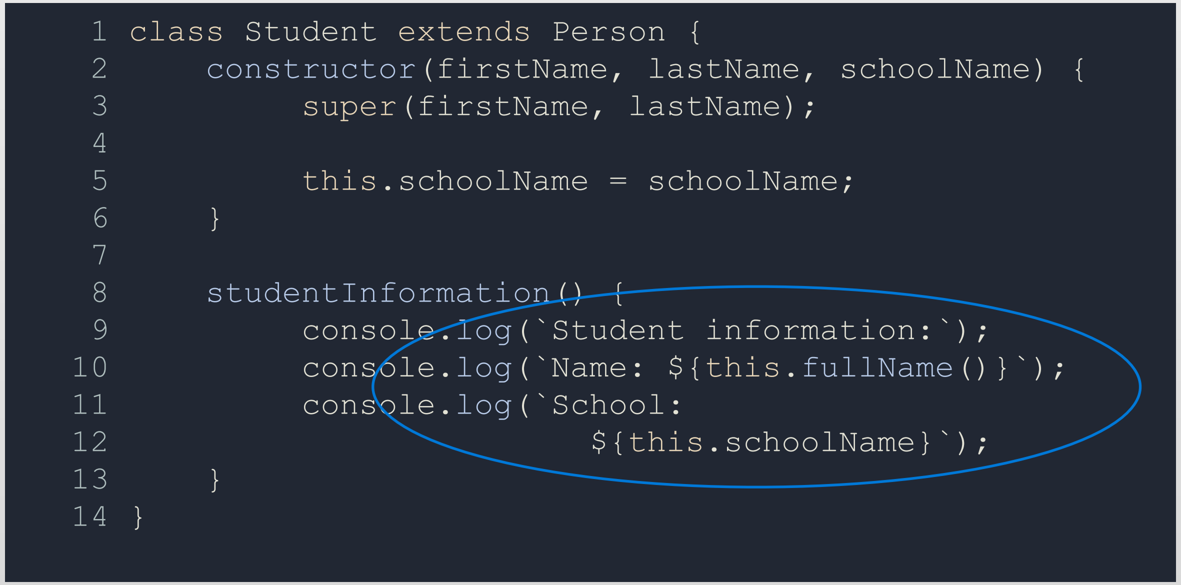
String syntactic sugars

Support

- basic literal strings
- interpolate variable bindings
- multiline strings



```
class Person {
    constructor(firstName, lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    fullName() {
        return `${this.firstName} ${this.lastName}`;
    }
}
```



```
class Student extends Person {
    constructor(firstName, lastName, schoolName) {
        super(firstName, lastName);

        this.schoolName = schoolName;
    }

    studentInformation() {
        console.log(`Student information:`);
        console.log(`Name: ${this.fullName()}`);
        console.log(`School:
                        ${this.schoolName}`);
    }
}
```

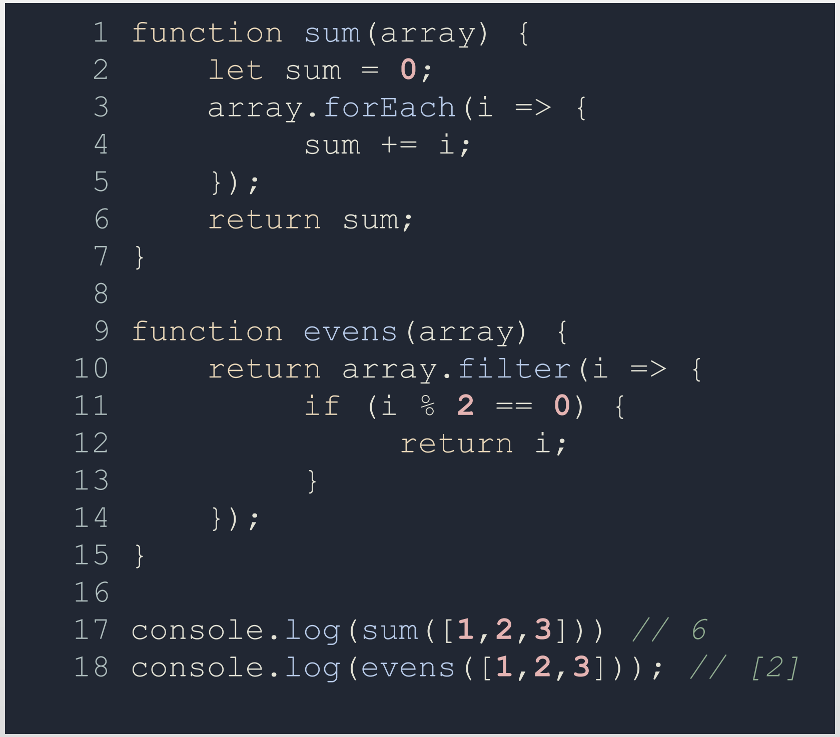
Arrow Functions

Shorthand for functions

Support

- statement bodies
- expression bodies

Lexical *this* binding



```
function sum(array) {
    let sum = 0;
    array.forEach(i => {
         sum += i;
    });
    return sum;
}

function evens(array) {
    return array.filter(i => {
         if (i % 2 == 0) {
              return i;
         }
    });
}

console.log(sum([1,2,3])) // 6
console.log(evens([1,2,3])); // [2]
```

Default + Rest + Spread

Default

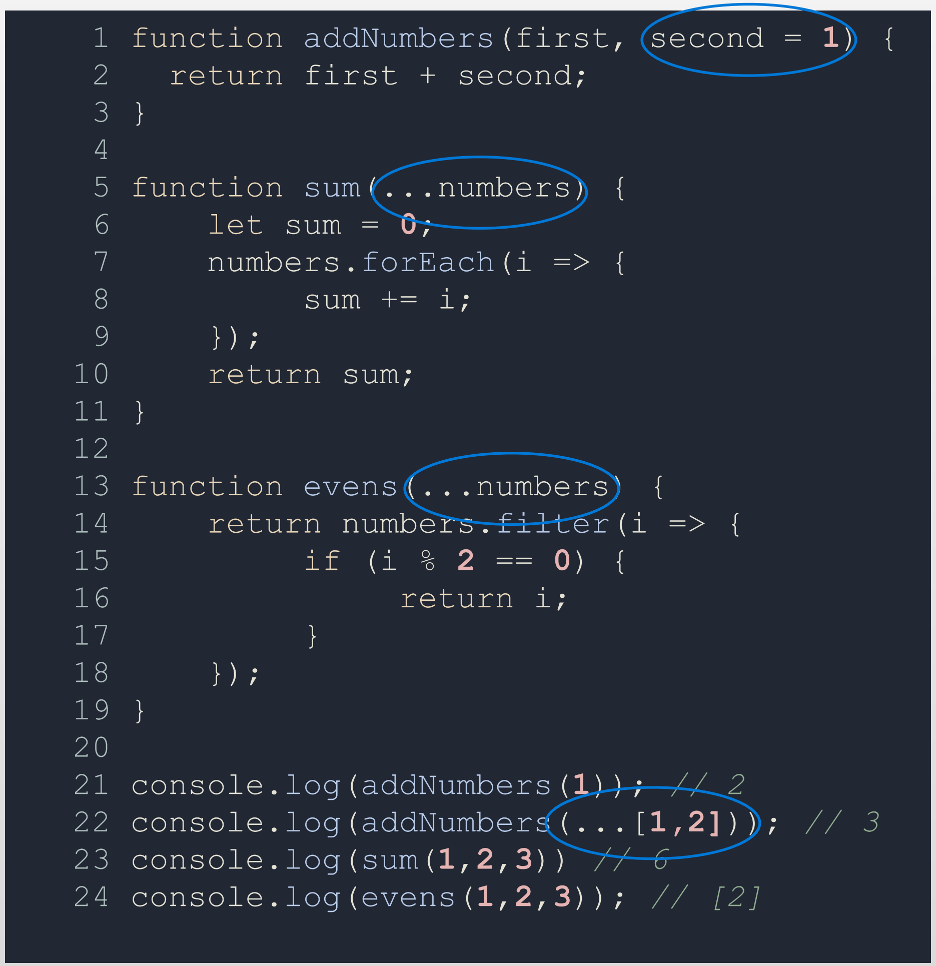
- provide default parameter values

Rest

- bind trailing parameters to an array

Spread

- turn an array into function arguments



```
function addNumbers(first, second = 1) {
 return first + second;
}

function sum(...numbers) {
  let sum = 0;
  numbers.forEach(i => {
    sum += i;
  });
  return sum;
}

function evens(...numbers) {
  return numbers.filter(i => {
    if (i % 2 == 0) {
       return i;
    }
  });
}

console.log(addNumbers(1)); // 2
console.log(addNumbers(...[1,2])); // 3
console.log(sum(1,2,3)) // 6
console.log(evens(1,2,3)); // [2]
```

Modules

export

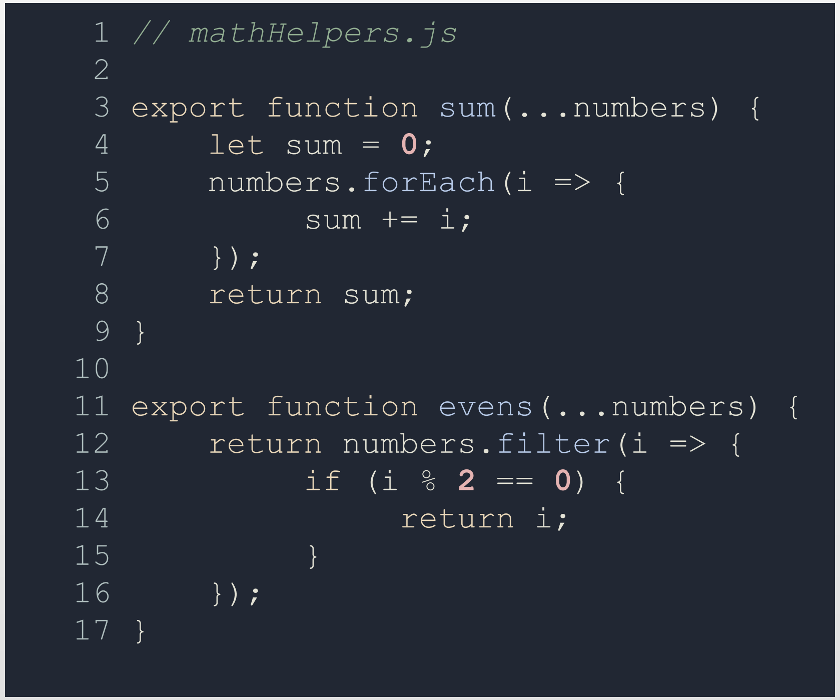
- define your public API

import

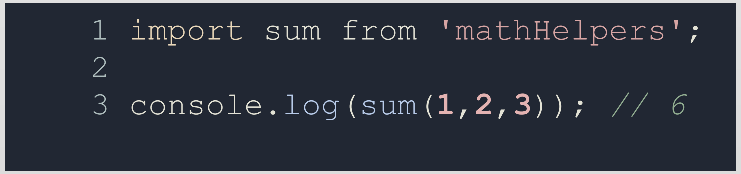
- use other modules

Support

- encapsulation
- modularisation



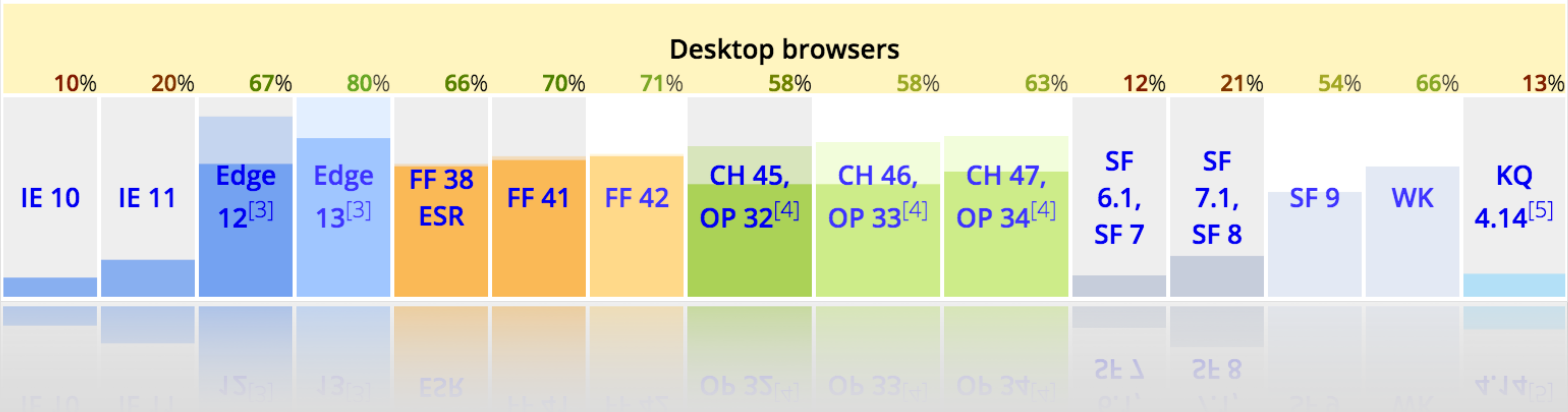
```javascript
// mathHelpers.js

export function sum(...numbers) {
    let sum = 0;
    numbers.forEach(i => {
         sum += i;
    });
    return sum;
}

export function evens(...numbers) {
    return numbers.filter(i => {
         if (i % 2 == 0) {
              return i;
         }
    });
}
```



```javascript
import sum from 'mathHelpers';

console.log(sum(1,2,3)); // 6
```

Browser Compatibility

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



Desktop browsers														
10%	20%	67%	80%	66%	70%	71%	58%	58%	63%	12%	21%	54%	66%	13%
IE 10	IE 11	Edge 12[3]	Edge 13[3]	FF 38 ESR	FF 41	FF 42	CH 45, OP 32[4]	CH 46, OP 33[4]	CH 47, OP 34[4]	SF 6.1, SF 7	SF 7.1, SF 8	SF 9	WK	KQ 4.14[5]

Babel

Overview

What is that?

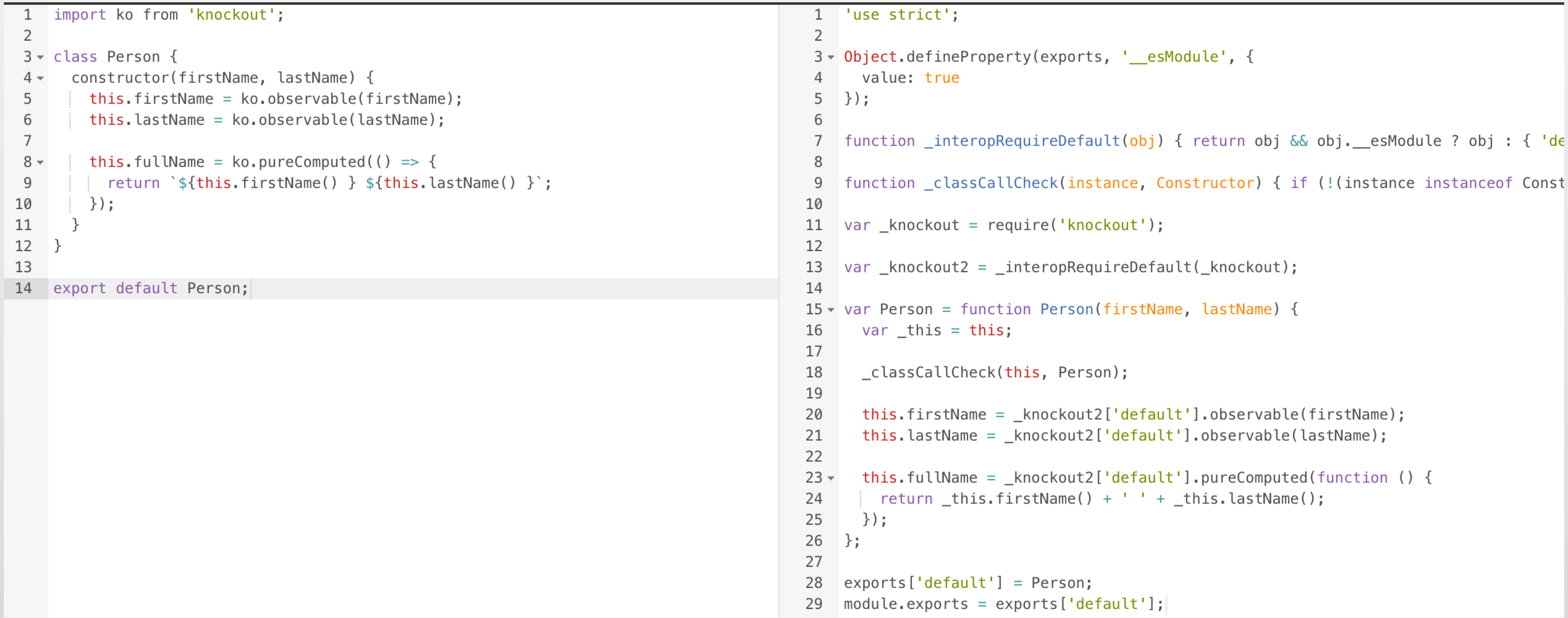
- Babel is a Javascript transpiler

Why transpile Javascript?

- So that we can code in the next generation Javascript right now!

Example

http://babeljs.io/repl

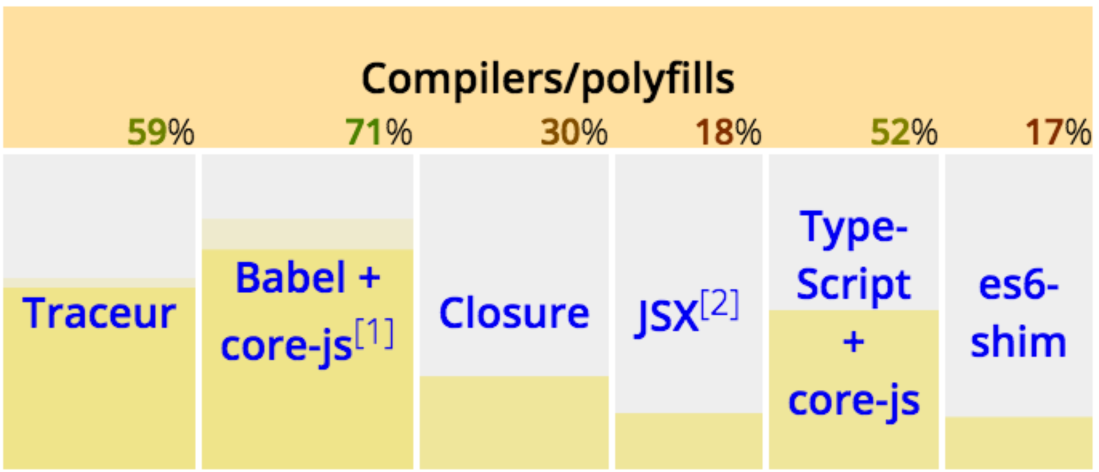


```
import ko from 'knockout';

class Person {
  constructor(firstName, lastName) {
    this.firstName = ko.observable(firstName);
    this.lastName = ko.observable(lastName);

    this.fullName = ko.pureComputed(() => {
      return `${this.firstName() } ${this.lastName() }`;
    });
  }
}

export default Person;
```

```
'use strict';

Object.defineProperty(exports, '__esModule', {
  value: true
});

function _interopRequireDefault(obj) { return obj && obj.__esModule ? obj : { 'de

function _classCallCheck(instance, Constructor) { if (!(instance instanceof Const

var _knockout = require('knockout');

var _knockout2 = _interopRequireDefault(_knockout);

var Person = function Person(firstName, lastName) {
  var _this = this;

  _classCallCheck(this, Person);

  this.firstName = _knockout2['default'].observable(firstName);
  this.lastName = _knockout2['default'].observable(lastName);

  this.fullName = _knockout2['default'].pureComputed(function () {
    return _this.firstName() + ' ' + _this.lastName();
  });
};

exports['default'] = Person;
module.exports = exports['default'];
```

Compilers Compatibility

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



Compilers/polyfills					
59%	71%	30%	18%	52%	17%
Traceur	Babel + core-js[1]	Closure	JSX[2]	Type-Script + core-js	es6-shim

JSPM

Overview

What is that?

- JSPM is a package manager for the SystemJS module loader
  - SystemJS was built on top of ES6 module loader

Why should I use it?

- Because it helps you managing your client-side dependencies
- To start using ES6 modules right away
- To not worry about module formats

Gulp

Overview

What is that?

- Gulp is a task/build runner
- Stream-based (like a pipeline)

Why do I need that?

- To automatize tasks (saving you some time!)

Demo

Patrocinadores “GOLD”

Gold Sponsor

isep Instituto Superior de Engenharia do Porto

Rumos

Patrocinadores “Silver”

Syncfusion®
Deliver innovation with ease®

LA.NET [PT]
Reflexões sobre C#, .NET e programação em geral

telerik
deliver more than expected

pluralsight
see what you can learn

Patrocinadores “Bronze”

http://bit.ly/netponto-aval-po-7

* Para quem não puder preencher durante a reunião,
iremos enviar um email com o link à tarde

survs
asking for you