Using

ES6

The New Javascript

TODAY!

What does the

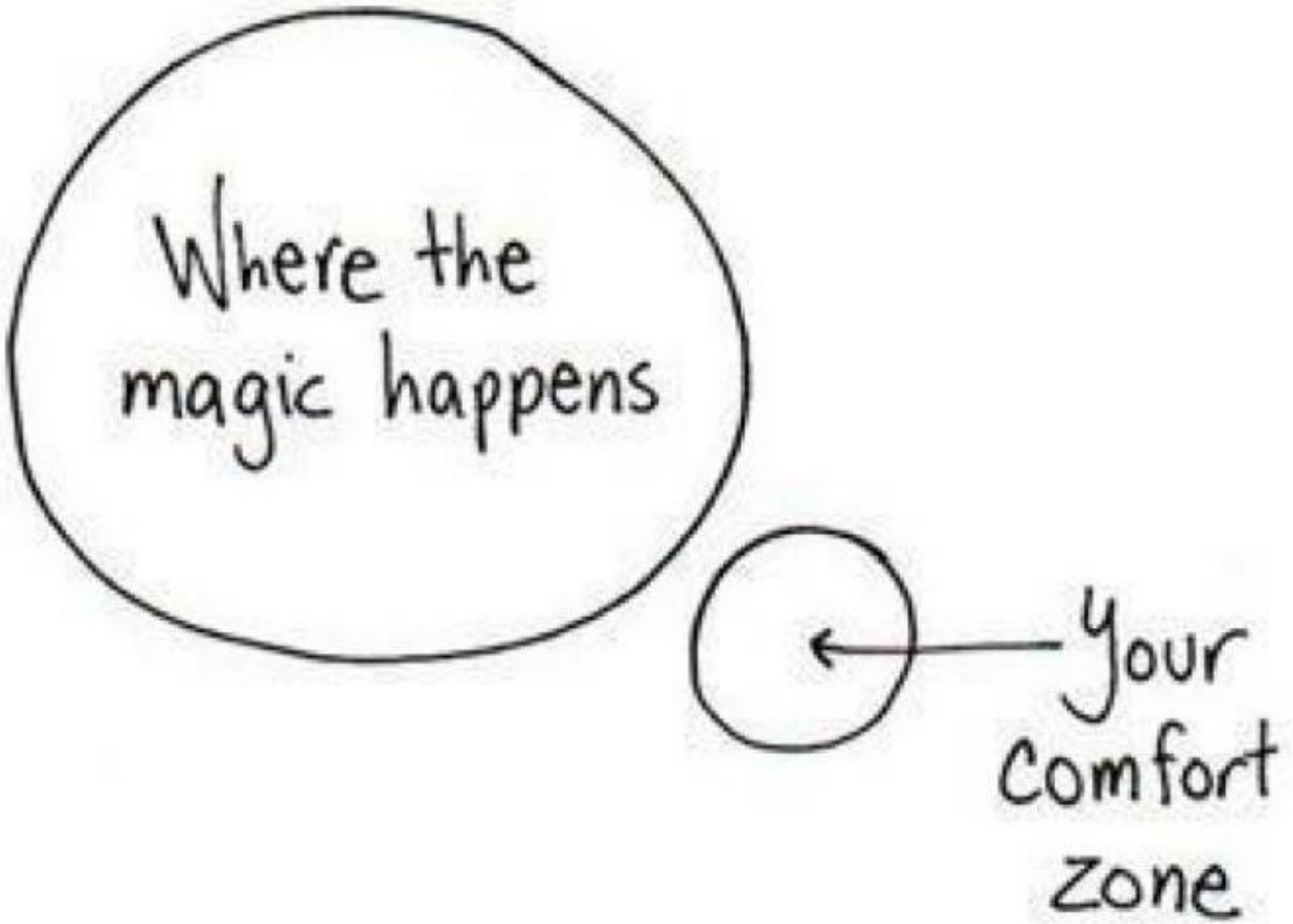
FUTURE

look like?


09
PICTURES

That's *so far*, no?

I DON'T GIVE A FUCK


Where the magic happens
Your comfort zone

and...

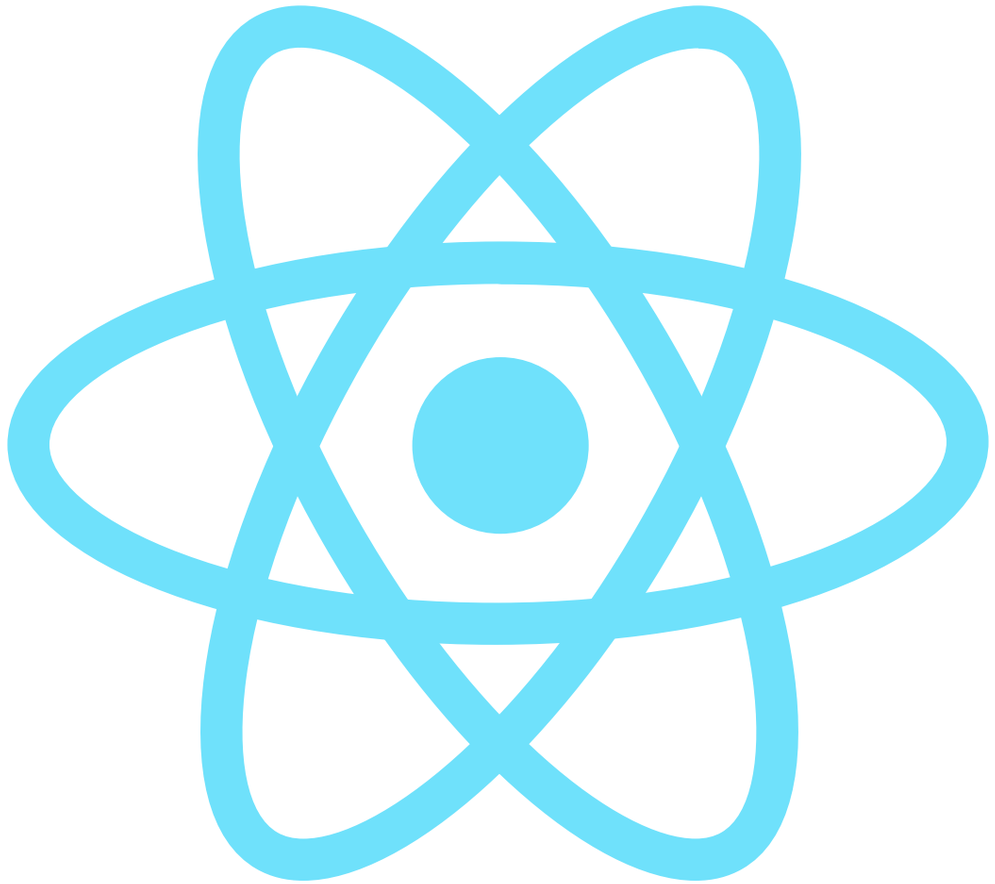
What will be the

FUTURE OF

the Javascript?

I sure that

will be..



"Geeks love fight"

Rasmus Lerdof


BULLSHIT

The future of

ANY

programming

LANGUAGE...

is itself...

and your
environment...

When one
language become
dependent of
just one tool...

Pedro Nauck

FRONTEND DEVELOPER

@pedronauck

pedronauck.com

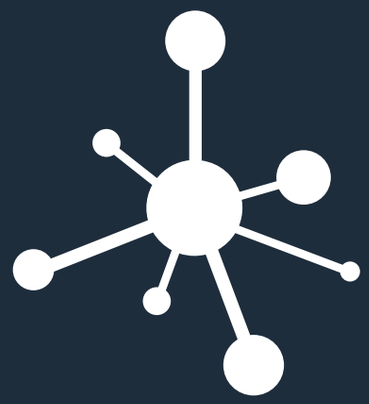
Agenda

...

A lot of code

A lit bit *about*

HISTORY



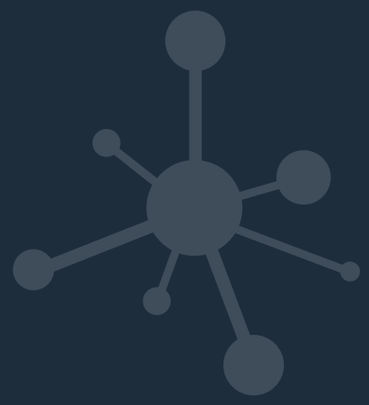
First proposal of World Wide Web

by Tim Bernes Lee

1989

1994

Born Netscape as the first real webbrowser



First proposal of World Wide Web

by Tim Bernes Lee

1989

1994

Born Netscape as the first real webbrowser

JS

1995

Brendan Eich

Works in the creation of the first version of what will be Javascript in the future

1997

Javascript is submitted to ECMA International

ecma INTERNACIONAL

1995

MochaScript

LiveScript

Javascript

1995

Influences

SCHEME

Dialecto of Lisp
Funcional Programming

SELF

Dialect of Smalltalk
Prototype Based
Dinamically Type
Just-in-time Compilation

JAVA

Syntax and name


after
10 days

1995

JS *Characteristics*

- ✓ Imperative and Structured (Have a lot of things about C. Except scope)
- ✓ Weakly typed and Dynamic typed
- ✓ Object and Event based (Asyncronous)
- ✓ Functional Language (Callback, closures, function scoping)
- ✓ Prototype Based with constructors. Non-classical OOP.

JS

1995

Brendan Eich

Works in the creation of the first version of what will be Javascript in the future

1997

Javascript is submitted to ECMA International

ecma INTERNACIONAL

1997

ecmaScript

- ✓ Attempts to standardize (IE Sucks)
- ✓ Netscape submitted JS to ECMA Internacional
- ✓ ECMAScript was created (ECMA-262 Edition 1)

ES2

Just some editorial changes to meet ISO requirements.

1998

1999

ES3

Start to work with RegExp

Better strings handling

Exception suport with Try/Catch block

ES2

Just some editorial changes to meet ISO requirements.

1998

1999

ES3

Start to work with RegExp

Better strings handling

Exception suport with Try/Catch block

ES4 >

EX4, Classes, Iterators, Generators, Block Scope

Suport with Flash/ActionsScript

Was abandoned in 2004

2000

2002

Simple object-based structure

To explore Ajax capabilities

< JSON

ES4

EX4, Classes, Iterators, Generators, Block Scope

Suport with Flash/ActionsScript

Was abandoned in 2004

2000

2002

Simple object-based structure

To explore Ajax capabilities

JSON

jQuery
write less, do more.

A better way to manipulate DOM and Ajax methods

2006

2009

Ryan Dahl introduces NodeJS at JSConf EU conference

Javascript in the server side using V8

A better way to manipulate DOM and Ajax methods

2006

2009

Ryan Dahl introduces NodeJS at JSConf EU conference

Javascript in the server side using V8

ES5 >

Strict mode

Native JSON support

Function.prototype.bind

A better Array and Object suport

2011

2013

Fix all these things! < ES6

ES5

2011

- Strict mode
- Native JSON support
- Function.prototype.bind
- A better Array and Object suport

2013

ES6

Fix all these things!

ES6 The New Javascript

Coolest Features

ES6

The New Javascript

Let/Const and Block Scoping

Prototype

Arrow Functions

Template Literals

Spread Operator

Rest Parameters

For-of

Classes

Generators

Destructuring Assigment

Default Parameters

Object Literals Improvement

Modules

Prototype

A lot new API's

Array *Prototype*

```
// ES5

var arr = new Array(1, 2, 3, 4);
// [1,2,3,4]

var arr = new Array(4);
// [undefined, undefined, undefined, undefined]
```

Array *Prototype*

```
// ES5
var arr = new Array(1, 2, 3, 4);
// [1,2,3,4]

var arr = new Array(4);
// [undefined, undefined, undefined, undefined]
```

Array *Prototype*

```
// ES6

var arr = Array.of(4); // [4]
```

Array *Prototype*

```
// ES5

var users = [
  { name: 'Peter' },
  { name: 'Joe' },
  { name: 'Michael' }
];

var joe;

users.forEach(function(user) {
  if (user.name === 'Joe') joe = user;
});
```

Array *Prototype*

```
// ES5

var users = [
  { name: 'Peter' },
  { name: 'Joe' },
  { name: 'Michael' }
];

var joe;

users.forEach(function(user) {
  if (user.name === 'Joe') joe = user;
});
```

Array *Prototype*

```
// ES5

var users = [
  { name: 'Peter' },
  { name: 'Joe' },
  { name: 'Michael' }
];

var joe;

users.forEach(function(user) {
  if (user.name === 'Joe') joe = user;
});
```

Array *Prototype*

```
// ES6

var joe = users.find(function(user, index, arr) {
  if (user.name === 'Joe') return user;
});
```

Array *Prototype*

```
// ES6
var findByName = function(name) {
  return function(item, index, arr) {
    if (item.name === name) return item;
  };
};

var joe = users.find(findByName('Joe'));
```

Array *Prototype*

```
// ES6
var findByName = function(name) {
  return function(item, index, arr) {
    if (item.name === name) return item;
  };
};

var joe = users.find(findByName('Joe'));
```

Array *Prototype*

```
// ES6
var findByName = function(name) {
  return function(item, index, arr) {
    if (item.name === name) return item;
  }
};

var joe = users.find(findByName('Joe'));
```

Array *Prototype*

`Array.from()`

`Array.of()`

`Array.prototype.find()`

`Array.prototype.findIndex()`

`Array.prototype.fill()`

`Array.prototype.entries()`

`Array.prototype.keys()`

`Array.prototype.copyWithin()`

http://bit.ly/es6-array-prototype

String *Prototype*

`String.fromCodePoint()`
`String.prototype.codePointAt()`
`String.prototype.startsWith()`
`String.prototype.endsWith()`
`String.prototype.contains()`
`String.prototype.repeat()`
`String.prototype.normalize()`
`String.raw()`

http://bit.ly/es6-string-prototype

Number *Prototype*

`Number.isNaN()`

`Number.isFinite()`

`Number.isInteger()`

`Number.parseInt()`

`Number.parseFloat()`

`Number.isSafeInteger()`

`Number.EPSILON`

`Number.MAX_SAFE_INTEGER`

`Number.MIN_SAFE_INTEGER`

http://bit.ly/es6-number-prototype

For Of

Replaces for-in

New iterator (other, yeah!)

Works fine with Array and Object

For Of

```
var student = {
  name: 'John',
  age: 23,
  city: 'Miami'
};

for (var prop in student) {
  console.log(prop) // name, age, city
}

for (var prop of student) {
  console.log(prop) // John, 23, Miami
}
```

For Of

```
var student = {
  name: 'John',
  age: 23,
  city: 'Miami'
};

for (var prop in student) {
  console.log(prop) // name, age, city
}

for (var prop of student) {
  console.log(prop) // John, 23, Miami
}
```

For Of

```
var student = {
  name: 'John',
  age: 23,
  city: 'Miami'
};

for (var prop in student) {
  console.log(prop) // name, age, city
}

for (var prop of student) {
  console.log(prop) // John, 23, Miami
}
```

For Of

```
var student = {
  name: 'John',
  age: 23,
  city: 'Miami'
};

for (var prop in student) {
  console.log(prop) // name, age, city
}

for (var prop of student) {
  console.log(prop) // John, 23, Miami
}
```

For Of

```
// ES6
var names = ['John', 'Peter', 'Michael'];

for (var name of names) {
  console.log(name) // John, Peter, Michael
}
```

For Of Array **Comprehensions**

```
// ES5
var students = [
  { name: 'John', age: 16 },
  { name: 'Peter' age: 23 },
  { name: 'Michael', age: 25 }
];

var adults = students.filter(function(student) {
  return student.age > 18;
});
```

For Of Array **Comprehensions**

```
// ES6
var students = [
   { name: 'John', age: 16 },
   { name: 'Peter' age: 23 },
   { name: 'Michael', age: 25 }
];

var adults = [for (s of students) if (s.age > 18)];
```

For Of Array **Comprehensions**

```
// ES6
var students = [
  { name: 'John', age: 16 },
  { name: 'Peter' age: 23 },
  { name: 'Michael', age: 25 }
];

var adults = [for (s of students) if (s.age > 18)];
```

Let (Block Scoping)

The new var

Fix hoisting problems

Let (Block Scoping)

```
// ES5
(function() {
  console.log(message); // hello world
  var message = 'hello world';
})();
```

Let (Block Scoping)

```
// ES5
(function() {
  var message = 'hello world';
  console.log(message); // hello world
})();
```

Let (Block Scoping)

```
// ES6
(function() {
  console.log(message); // ReferenceError
  let message = 'hello world';
})();
```

Let (Block Scoping)

```
// ES5
var handlers = [];

for (var count = 0; count < 3; count++) {
  handlers[count] = function () {
    console.log(count);
  }
}

handlers[0](); // "2"
handlers[1](); // "2"
handlers[2](); // "2"
```

Let (Block Scoping)

```
// ES5
var handlers = [];

for (var count = 0; count < 3; count++) {
  (function (i) {
    handlers[i] = function () {
      console.log(i)
    };
  })(count);
}

handlers[0](); // "0"
handlers[1](); // "1"
handlers[2](); // "2"
```

Let (Block Scoping)

```
// ES5
var handlers = [];

for (var count = 0; count < 3; count++) {
  (function (i) {
    handlers[i] = function () {
      console.log(i)
    };
  })(count);
}

handlers[0](); // alerts "0"
handlers[1](); // alerts "1"
handlers[2](); // alerts "2"
```

Let (Block Scoping)

```
// ES6
let handlers = [];

for (let count = 0; count < 3; count++) {
  handlers[count] = function () {
    console.log(count);
  }
}

handlers[0](); // "0"
handlers[1](); // "1"
handlers[2](); // "2"
```

Let (Block Scoping)

```
// ES6
let handlers = [];
for (let count = 0; count < 3; count++) {
  handlers[count] = function () {
    console.log(count);
  }
}

handlers[0](); // "0"
handlers[1](); // "1"
handlers[2](); // "2"
```

Const

Read-only variables

Block scoping

Const

```
// ES6
const TIMEOUT = 600;

TIMEOUT = 300;
// Throw: TIMEOUT is read-only;
```

Arrow Functions

New token =>

Bye bye bind()

It isn't just a Syntatic Sugar

Arrow Functions

```
// ES5
var blueProducts;

categories.forEach(function(category) {
  blueProducts = category.products.filter(function(product) {
    return product.color === 'blue';
  });)
});
```

Arrow Functions

22 **unnecessary** chars

```
// ES5
var blueProducts;

categories.forEach(function(category) {
  blueProducts = category.products.filter(function(product) {
    return product.color === 'blue';
  });
});
```

Arrow Functions

```
// ES6
let blueProducts;

categories.forEach(c => {
  blueProducts = c.products.filter(p => p.color === 'blue');
});
```

Arrow Functions

```
// ES6
let blueProducts;

categories.forEach(c => {
  blueProducts = c.products.filter(p => p.color === 'blue');
});
```

Arrow Functions

```
// ES5
...
getAverages: function(students) {
  return students.map(function(student) {
    return this.calcAverage(student.grades);
  }
}
...
```

Arrow Functions

```
// ES5
...
getAverages: function(students) {
  return students.map(function(student) {
    return this.calcAverage(student.grades);
  }
}
...
```

// ERROR

Arrow Functions

```
// ES5
...
getAverages: function(students) {
  return students.map(function(student) {
    return this.calcAverage(student.grades);
  }.bind(this));
}
...
```

Arrow Functions

```
// ES6
...
getAverages(students) {
  return students.map(s => this.calcAverage(s.grades);
}
...
```

Template Literal

Allows string literals with embedded expressions

Multiline suport


esc
F1
F2
F3
F4
~
`
!
1
@
2
#
3
$
4
%
5
tab
Q
W
E
R
A
S
D
F

Template Literal

```
// ES5
var name = 'John';
var age = 23;

console.log('My name is ' + name + '. I\'m ' + age + '.');
```

Template Literal

```
// ES6
var name = 'John';
var age = 23;

console.log(`My name is ${name}. I\'m ${age}`);
```

Template Literal

```
// ES6
var name = 'John';
var age = 23;

console.log(`My name is ${name}. I\'m ${age}`);
```

Template Literal

```
// ES5
var className = 'myClass';
var content = 'Hello world';

var div = '<div class="' + className + '">' +
          '<p>' + content + '</p>' +
          '</div>';
```

Template Literal

```
// ES6
let className = 'myClass';
let content = 'Hello world';

let div = `
  <div class="${className}">
    <p>${content}</p>
  </div>`;
```

Default Parameters

Default Parameters

```
// ES5
var isApproved = function(grades, base) {
  base = base || 7;

  var average = grades.reduce(function(sum, num) {
    return sum + num;
  }) / grades.length;

  return average >= base;
};
```

Default Parameters

```
// ES5
var isApproved = function(grades, base) {
  base = base || 7;

  var average = grades.reduce(function(sum, num) {
    return sum + num;
  }) / grades.length;

  return average >= base;
};
```

Default Parameters

```
// ES6
let isApproved = (grades, base = 7) => {
  let len = grades.length;
  let avg = grades.reduce((sum, num) => {
    return (sum + num) / len;
  };

  return avg >= base;
};
```

Default Parameters

```
// ES6
let isApproved = (grades, base = 7) => {
  let len = grades.length;
  let avg = grades.reduce((sum, num) => {
    return (sum + num) / len;
  };

  return avg >= base;
};
```

Rest Parameters

Allow functions with variable number of params

Rest Parameters

rest parameters

```
store.add('fruit',     'apple');
store.add('dairy',     'milk', 'cheese', 'yoghurt');
store.add('pastries', 'donuts', 'croissants');
```

Rest Parameters

```
// ES3, ES5
store.add = function(category) {
  var items = [].slice.call(arguments, 1);

  items.forEach(function (item) {
    store.aisle[category].push(item);
  });
};
```

Rest Parameters

```
// ES3, ES5
store.add = function(category) {
  var items = [].slice.call(arguments, 1);

  items.forEach(function (item) {
    store.aisle[category].push(item);
  });
};
```

Rest Parameters

```
// ES6
store.add = (category, ...items) => {
  items.forEach(item => {
    store.aisle[category].push(item)
  });
};
```

Rest Parameters

```
// ES6
store.add = (category, ...items) => {
  items.forEach(item => {
    store.aisle[category].push(item)
  });
};
```

Spread Operator

Allow to spread variables into parameters

Spread Operator

```
// ES5
var calcAverage = function(x, y, z) {
  return (x + y + z) / 3;
};

var grades = [70, 80, 85];

calcAverage.apply(null, grades);
```

Spread Operator

```
// ES5
var calcAverage = function(x, y, z) {
  return (x + y + z) / 3;
};

var grades = [70, 80, 85];

calcAverage.apply(null, grades);
```

Spread Operator

```
// ES5
var calcAverage = function(x, y, z) {
  return (x + y + z) / 3;
};

var grades = [70, 80, 85];

calcAverage.apply(null, grades);
```

Spread Operator

```
// ES6
let calcAverage = (x, y, z) => (x + y + z) / 3;
let grades = [70, 80, 85];

calcAverage(...grades);
```

Spread Operator

```
// ES6
let calcAverage = (x, y, z) => (x + y + z) / 3;
let grades = [70, 80, 85];

calcAverage(...grades);
```

Spread Operator

```
// ES6
let names = ['Peter', 'John', ‘Michael'];

let firstArr = [];
let secondArr = [];

firstArr.push(...names);
secondArr.push(names);

console.log(firstArr.length); // 3
console.log(secondArr.length); // 1
```

Spread Operator

```
// ES6
let names = ['Peter', 'John', ‘Michael'];

let firstArr = [];
let secondArr = [];

firstArr.push(...names);
secondArr.push(names);

console.log(firstArr.length); // 3
console.log(secondArr.length); // 1
```

Spread Operator

```
// ES6
let names = ['Peter', 'John', ‘Michael'];

let firstArr = [];
let secondArr = [];

firstArr.push(...names);
secondArr.push(names);

console.log(firstArr.length); // 3
console.log(secondArr.length); // 1
```

Spread Operator

```
// ES6
let names = ['Peter', 'John', ‘Michael'];

let firstArr = [];
let secondArr = [];

firstArr.push(...names);
secondArr.push(names);

console.log(firstArr.length); // 3
console.log(secondArr.length); // 1
```

Spread Operator

```
// ES6
let names = ['Peter', 'John', ‘Michael'];

let firstArr = [];
let secondArr = [];

firstArr.push(...names);
secondArr.push(names);

console.log(firstArr.length); // 3
console.log(secondArr.length); // 1
```

Destructuring Assigment

Use data-structure to declare elements

Functions with multiple returns

Destructuring Array

```
// ES5
var name = 'John';
var city = 'California';
var age = 23;

// ES6
let [name, city, age] = ['John', 'California', 23];
```

Destructuring Array

```
// ES5
var name = 'John';
var city = 'California';
var age = 23;

// ES6
let [name, city, age] = ['John', 'California', 23];
```

Destructuring Array

```
// ES5
var foo = function(a, b) {
  return [ a + 1, b + 2 ];
};

var a = foo()[0];
var b = foo()[1];

// ES6
let foo = (a, b) => [a + 1, b + 2];
let [a, b] = foo();
```

Destructuring Array

```
// ES3, ES5
var foo = function(a, b) {
  return [ a + 1, b + 2 ];
};

var a = foo()[0];
var b = foo()[1];

// ES6
let foo = (a, b) => [a + 1, b + 2];
let [a, b] = foo();
```

Destructuring Object

```
// ES5
var location = document.location;
var protocol = location.protocol;
var hostname = location.hostname;
var port = location.port;

// ES6
let { protocal, hostname, port } = document.location;
```

Destructuring Object

```
// ES5
var location = document.location;
var protocol = location.protocol;
var hostname = location.hostname;
var port = location.port;

// ES6
let { protocal, hostname, port } = document.location;
```

Destructuring Object

```
// ES5
var foo = function() { return 'foo'; }
var bar = function() { return 'bar'; }

module.exports = {
  foo: foo,
  bar: barr
};

// ES6
let foo = () => 'foo';
let foo = () => 'bar';

module.exports = { foo, bar };
```

Destructuring Object

```
// ES5
var foo = function() { return 'foo'; }
var bar = function() { return 'bar'; }

module.exports = {
  foo: foo,
  bar: barr
};
```

```
// ES6
let foo = () => 'foo';
let foo = () => 'bar';

module.exports = { foo, bar };
```

Destructuring Object

```
// ES5
var foo = function(opts) {
  var bar = opts.bar;
  var baz = opts.baz;

  return bar + baz;
};

// ES6
let foo = ({bar, baz}) => bar + baz;

// ES6
let foo = ({b1: bar, b2: baz}) => b1 + b2;
```

Destructuring Object

```
// ES5
var foo = function(opts) {
  var bar = opts.bar;
  var baz = opts.baz;

  return bar + baz;
};

// ES6
let foo = ({bar, baz}) => bar + baz;

// ES6
let foo = ({b1: bar, b2: baz}) => b1 + b2;
```

Destructuring Object

```
// ES5
var foo = function(opts) {
  var bar = opts.bar;
  var baz = opts.baz;

  return bar + baz;
};

// ES6
let foo = ({bar, baz}) => bar + baz;

// ES6
let foo = ({b1: bar, b2: baz}) => b1 + b2;
```

Generators

Better way to manage flows

Yield token allow control returns

delivery by part

Generators

```
let idMaker = function* () {
  let index = 0;
  while(true) {
    yield index++
  }
};

let gen = idMaker();

console.log(gen.next()) // { value: 1, done: false }
console.log(gen.next()) // { value: 2, done: false }
console.log(gen.next()) // { value: 3, done: false }
```

Generators

```
let idMaker = function* () {
  let index = 0;

  while(true) {
    yield index++
  }
};

let gen = idMaker();

console.log(gen.next()) // { value: 1, done: false }
console.log(gen.next()) // { value: 2, done: false }
console.log(gen.next()) // { value: 3, done: false }
```

Generators

```
let idMaker = function* () {
  let index = 0;

  while(true) {
    yield index++
  }
};

let gen = idMaker();

console.log(gen.next()) // { value: 1, done: false }
console.log(gen.next()) // { value: 2, done: false }
console.log(gen.next()) // { value: 3, done: false }
```

Generators

```
user.tasks = (function* () {
  let a = yield $.ajax('http://get/user');
  let b = yield $.ajax('http://process/user', a);
  yield $.ajax('http://save/user', b);
})();
```

Generators

```
user.tasks.next() // gets the user
  .then(user => {
    // we do something with the user and then...
    return user.tasks.next(user);
  })
  .then(user => {
    // we do something else after the user
    // has been processed and then...
    return user.tasks.next(user);
  })
  .then(user => {
    // final operations here
  });
```

Generators

```
user.tasks.next() // gets the user
  .then(user => {
    // we do something with the user and then...
    return user.tasks.next(user);
  })
  .then(user => {
    // we do something else after the user
    // has been processed and then...
    return user.tasks.next(user);
  })
  .then(user => {
    // final operations here
  });
```

Generators

```
user.tasks.next() // gets the user
  .then(user => {
    // we do something with the user and then...
    return user.tasks.next(user);
  })
  .then(user => {
    // we do something else after the user
    // has been processed and then...
    return user.tasks.next(user);
  })
  .then(user => {
    // final operations here
  });
```

Generators

```
user.tasks.next() // gets the user
  .then(user => {
    // we do something with the user and then...
    return user.tasks.next(user);
  })
.then(user => {
  // we do something else after the user
  // has been processed and then...
  return user.tasks.next(user);
})
.then(user => {
  // final operations here
});
```

Generators

```
user.tasks.next() // gets the user
  .then(user => {
    // we do something with the user and then...
    return user.tasks.next(user);
  })
  .then(user => {
    // we do something else after the user
    // has been processed and then...
    return user.tasks.next(user);
  })
  .then(user => {
    // final operations here
  });
```

Classes

Classical OOP

Allow single inheritance

Classes

```
class Vehicle {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
  get name() {
    return `This is a ${this.name}`;
  }
  get year() {
    return `Year of manufacture: ${this.year}`;
  }
  set name(name) {
    this.name = name.toTitleize();
  }
}
```

Classes

```
class Vehicle {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
  get name() {
    return `This is a ${this.name}`;
  }
  get year() {
    return `Year of manufacture: ${this.year}`;
  }
  set name(name) {
    this.name = name.toTitleize();
  }
}
```

Classes

```
class Vehicle {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
  get name() {
    return `This is a ${this.name}`;
  }
  get year() {
    return `Year of manufacture: ${this.year}`;
  }
  set name(name) {
    this.name = name.toTitleize();
  }
}
```

Classes

```
var fusca = new Vehicle('Fusca', 1976);
fusca.name = 'fuxxca';
// Fuxxca

console.log(fusca.name);
// This is a Fuxxca

console.log(fusca.year);
// Year of manufacture: 1976
```

Classes

```
class Car extends Vehicle {
  constructor(name, brand, year) {
    super(name, year)
    this.brand = brand;
  }
}

let fusca = new Car('Fusca', 'volkswagen', 1976);
```

Classes

```
class Car extends Vehicle {
  constructor(name, brand, year) {
    super(name, year)
    this.brand = brand;
  }
}

let fusca = new Car('Fusca', 'volkswagen', 1976);
```

Classes

```
Car = function(Vehicle) {
  function Car(name, brand, year) {
    Vehicle.call(this, name, year)
    this.brand = brand;
  }

  Car.prototype = Object.create(Vehicle.prototype, {
    constructor: {
      value: Car,
      enumerable: false,
      writable: true,
      configurable: true
    }
  });

  Car.__proto__ = Vehicle;

  return Car;
}(Vehicle);
```

Classes

```
Car = function(Vehicle) {
  function Car(name, brand, year) {
    Vehicle.call(this, name, year)
    this.brand = brand;
  }

  Car.prototype = Object.create(Vehicle.prototype, {
    constructor: {
      value: Car,
      enumerable: false,
      writable: true,
      configurable: true
    }
  });

  Car.__proto__ = Vehicle;

  return Car;
}(Vehicle);
```

Classes

```
Car = function(Vehicle) {
  function Car(name, brand, year) {
    Vehicle.call(this, name, year)
    this.brand = brand;
  }

  Car.prototype = Object.create(Vehicle.prototype, {
    constructor: {
      value: Car,
      enumerable: false,
      writable: true,
      configurable: true
    }
  });

  Car.__proto__ = Vehicle;

  return Car;
}(Vehicle);
```

Classes

```
Car = function(Vehicle) {
  function Car(name, brand, year) {
    Vehicle.call(this, name, year)
    this.brand = brand;
  }

  Car.prototype = Object.create(Vehicle.prototype, {
    constructor: {
      value: Car,
      enumerable: false,
      writable: true,
      configurable: true
    }
  });

  Car.__proto__ = Vehicle;

  return Car;
}(Vehicle);
```

Modules

Native way to build app with a modular approach

Modules

Named Exports

```
// foobar.js
var foo = 'foo';
var bar = 'bar';

export { foo, bar };

// app.js
import { foo, bar } from 'foobar';

console.log(foo); // foo
```

Modules

Default **Exports**

```
// car.js
export default class Car() {
  constructor(name) {
    this.name = name;
  }
}

// main.js
import Car from 'car';
let fusca = new Car('Fusca');
```

Modules

All Exports

```
// foobar.js
let foo = 'foo';
let bar = 'bar';

export { foo, bar };

// main.js
module foobar from ‘foobar';

console.log(foobar.foo) // 'foo'
console.log(foobar.bar) // 'bar'
```

How can I use ES6 *today?*

Modern browser supports partially

NodeJS supports many features (—harmony flag)

Transpilers and polyfills

June 2015

6to5 transpiler

github.com/sebmck/6to5

Have a lot of features implemented

CLI-Tool working as a Node REPL

Have a good integration with Gulp/Grunt


```
var gulp = require('gulp');
var to5 = require('gulp-6to5');

gulp.task('default', function () {
  return gulp.src('src/app.js')
    .pipe(to5())
    .pipe(gulp.dest('dist'));
});
```

github.com/sindresorhus/gulp-6to5

```
require('load-grunt-tasks')(grunt);

grunt.initConfig({
  '6to5': {
    options: { sourceMap: true },
    dist: {
      files: {
        'dist/app.js': 'src/app.js'
      }
    }
  }
});

grunt.registerTask('default', ['6to5']);
```

github.com/sindresorhus/grunt-6to5

pedronauck / hacker-news-es6

Hacker News feed built with ECMAScript 6 and jQuery http://pedronauck.github.io/hacker-news-es6 — Edit

9 commits 2 branches 0 releases 1 contributor

branch: master hacker-news-es6 / +

change README again ;p

pedronauck authored a day ago latest commit 081e0119a6

src	add 6to5-browserify transformer	a day ago
.gitignore	initial commit	3 days ago
README.md	change README again ;p	a day ago
gulpfile.js	add gulp-gh-pages to deploy	a day ago
package.json	add gulp-gh-pages to deploy	a day ago

Code
Boards
Issues 0
Pull Requests 0
Wiki
Pulse
Graphs
Settings

SSH clone URL

git@github.com:pec

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP

README.md

Hacker News ES6

Just a simple Hacker News Feed built using these ES6 features:

- Modules
- Classes
- Template Literals

ES6 references

ECMAScript 5 **6** 7 non-standard compatibility table

Please note that *some of these tests* represent **existence**, not functionality or full conformance.

Sort by number of features? ☐ Show obsolete browsers? ☐

			Compilers				Desktop browsers											
		39%	59%	55%	66%	30%	5%	21%	74%	45%	49%	58%	60%	47%	54%	9%	9%	24%
Feature name		Current browser	Traceur	6to5 + polyfill	EJS	Closure Compiler	IE 10	IE 11	IE Technical Preview[1]	FF 31	FF 33	FF 34	FF 35	CH 38, OP 25[2]	CH 39, OP 26[2]	SF 6	SF 7.0	SF 7.1, SF 8
proper tail calls (tail call optimisation)	C	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
arrow functions	▶	0/9	7/9	7/9	7/9	6/9	0/9	0/9	8/9	7/9	7/9	7/9	7/9	3/9	4/9	0/9	0/9	0/9
const	▶	1/8	6/8	4/8	8/8	6/8	0/8	8/8	8/8	3/8	3/8	3/8	3/8	5/8	5/8	1/8	1/8	1/8
let	▶	0/10	8/10	6/10	10/10	8/10	0/10	8/10	8/10	0/10	0/10	0/10	0/10	5/10	5/10	0/10	0/10	0/10
default function parameters	▶	0/5	3/5	3/5	3/5	3/5	0/5	0/5	0/5	3/5	3/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5
rest parameters	C	No	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
spread (...) operator	▶	0/8	8/8	8/8	6/8	2/8	0/8	0/8	4/8	6/8	6/8	6/8	6/8	0/8	0/8	0/8	0/8	2/8
class	▶	0/8	7/8	7/8	6/8	6/8	0/8	0/8	8/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8
super	▶	0/3	3/3															0/3
object literal extensions	▶	0/3	3/3															1/3
for..of loops	▶	4/4	4/4	4/4	3/4	3/4	0/4	0/4	4/4	3/4	3/4	3/4	3/4	4/4	4/4	0/4	0/4	1/4
generators	▶	5/6	6/6	6/6	0/6	3/6	0/6	0/6	0/6	4/6	4/6	4/6	5/6	5/6	6/6	0/6	0/6	0/6
octal and binary literals	▶	4/4	2/4	2/4	4/4	4/4	0/4	0/4	2/4	2/4	2/4	2/4	2/4	4/4	4/4	0/4	0/4	0/4
template strings	▶	0/2	2/2	1/2	2/2	2/2	0/2	0/2	1/2	0/2	0/2	2/2	2/2	0/2	0/2	0/2	0/2	0/2
RegExp "y" and "u" flags	▶	0/2	1/2	1/2	0/2	0/2	0/2	0/2	1/2	1/2	1/2	1/2	1/2	0/2	1/2	0/2	0/2	0/2
typed arrays	▶	21/40	0/40	0/40	18/40	0/40	16/40	16/40	40/40	18/40	18/40	19/40	19/40	21/40	21/40	18/40	18/40	18/40
Map	▶	11/11	10/11	11/11	9/11	0/11	0/11	5/11	11/11	10/11	11/11	11/11	11/11	10/11	11/11	0/11	0/11	9/11
Set	▶	11/11	10/11	10/11	9/11	0/11	0/11	5/11	11/11	10/11	11/11	11/11	11/11	10/11	11/11	0/11	0/11	9/11
WeakMap	▶	4/4	0/4	0/4	1/4	0/4	0/4	2/4	4/4	2/4	3/4	3/4	3/4	4/4	4/4	0/4	0/4	3/4
WeakSet	▶	4/4	0/4	0/4	1/4	0/4	0/4	0/4	4/4	0/4	0/4	4/4	4/4	4/4	4/4	0/4	0/4	0/4
Proxy	▶	0/17	0/17	0/17	8/17	0/17	0/17	0/17	14/17	11/17	12/17	13/17	13/17	0/17	0/17	0/17	0/17	0/17
Reflect	▶	0/14	0/14	0/14	14/14	0/14	0/14	0/14	13/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14
block-level function declaration[11]	C	No	Yes	No	No	Yes	No	Yes	Yes	No	No	No	No	Yes	Yes	No	No	No
destructuring	▶	0/11	8/11	6/11	5/11	8/11	0/11	0/11	0/11	5/11	5/11	7/11	7/11	0/11	0/11	0/11	0/11	5/11
Promise	C	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes
Object static methods	▶	3/4	3/4	0/4	4/4	0/4	0/4	1/4	3/4	2/4	2/4	3/4	3/4	3/4	3/4	0/4	0/4	0/4
function "name" property	▶	2/16	0/16	0/16	0/16	0/16	0/16	0/16	0/16	3/16	3/16	4/16	4/16	2/16	3/16	3/16	3/16	3/16

ECMAScript Discussion Archives

http://esdiscuss.org

This site aims to provide an easily browsable, nicely formatted archive of all the email correspondence on the es-discuss@mozilla.org mailing list.

VIEW TOPICS

VIEW NOTES

ECMAScript 6 support in Mozilla

HIDE SIDEBAR

SEE ALSO

ECMAScript 5 support in Mozilla

ECMAScript 6 support in Mozilla

ECMAScript 7 support in Mozilla

Firefox JavaScript changelog

New in JavaScript 1.1

New in JavaScript 1.2

New in JavaScript 1.3

New in JavaScript 1.4

New in JavaScript 1.5

New in JavaScript 1.6

New in JavaScript 1.7

New in JavaScript 1.8

New in JavaScript 1.8.1

New in JavaScript 1.8.5

ECMAScript 6 is the next version of the standard, code-named "Harmony" or "ES.next". Specification drafts can be found on the official ECMA wiki. The first working draft based on ECMAScript 5.1, was published on July 12, 2011 as "ES.next". As of August 2014, ECMAScript 6 is already feature frozen, will be finished around the end of 2014 and will start to go into the official publication process starting in March 2015.

A channel for feedback on ES6 support is [illegible]

http://bit.ly/es6-mdn

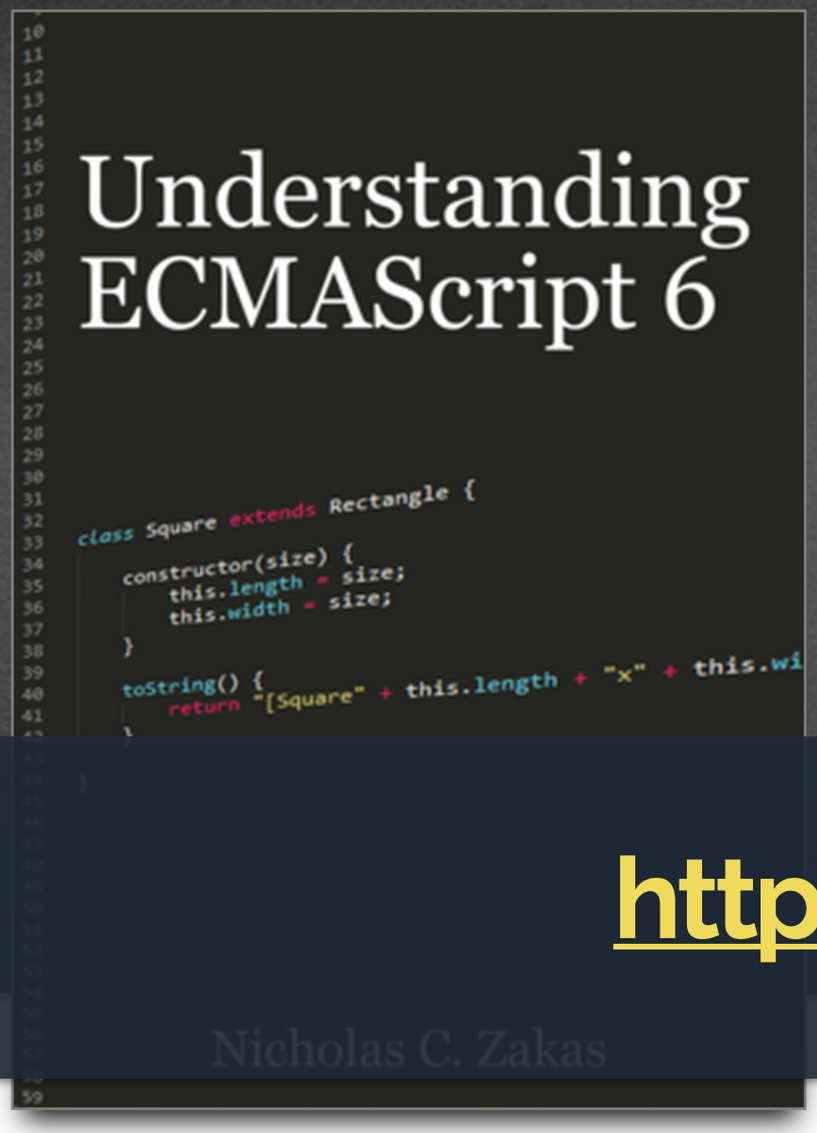
Already supported features

The following features are already implemented in Firefox:

Standard library

Additions to the `Array` object

- `Array` iteration with `for...of` (Firefox 13)
- `Array.from()` (Firefox 32)
- `Array.of()` (Firefox 25)
- `Array.prototype.fill()` (Firefox 31)
- `Array.prototype.find()`, `Array.prototype.findIndex()` (Firefox 25)
- `Array.prototype.entries()`, `Array.prototype.keys()` (Firefox 28)



Understanding ECMAScript 6

ECMAScript 6 is coming, are you ready? There's a lot of new concepts to learn and understand. Get a headstart with this book!

165 READERS

http://bit.ly/understanding-es6

by  Nicholas C. Zakas

Includes three convenient formats

- **PDF** *(for Mac or PC)*
- **EPUB** *(for iPad, iPhone, Android, and other ebook readers)*
- **MOBI** *(for Kindle)*

Buy Now

Minimum: $14.99
Suggested: $19.99+
100% HAPPINESS GUARANTEE

Free to read online | 165 readers | 81 pages | 19,807 words | 5% complete | English | CC BY NC ND | Updated 6 days ago

ES6ROCKS.com

to follow

@brendaneich

@rwaldron

@slicknet

@wycats

@littlecalculist

@lbljeffmo

@rauschma

@jaydson

@felipenmoura

I hope you *enjoyed*

Questions?