ES2015

OBS:

- Muitos slides, talvez pulemos alguns

Novidades e compatibilidade

- ES5: dezembro de 2009
- ES5.1: junho de 2011
- ES6 = ES2015: junho de 2015
- ES7 = ES2016: junho de 2016
    - http://kangax.github.io/compat-table/es6/
- Comparação de novidades com o http://kangax.github.io/compat-table/es2016plus/

The TC39 process for ECMAScript features: http://www.2ality.com/2015/11/tc39-process.html

Strict mode - lembrete

```
console.log(a); // Daria ReferenceError

a = 1; // declaração SEM var, let
console.log(a);
// 1
```

```
"use strict”;
a = 1; // declaração SEM var, let
console.log(a);

// ReferenceError: a is not defined (escrita ou leitura)
```

```
(function() {
  variavelNaoDeclarada = 'foo';
}());
console.log(variavelNaoDeclarada);

// Error
```

```
(function() {
 "use strict";
  variavelNaoDeclarada = 'foo';
}());
console.log(variavelNaoDeclarada);

//  ReferenceError: variavelNaoDeclarada is not defined
```

Mais diferenças em https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode

var x let

```
"use strict";
{  var a = 1; }
console.log(a);

// Error
```

```
"use strict";
{  let a = 1; }
console.log(a);

// ReferenceError: a is not defined
```

```
"use strict";
let a = 1;
{let a = 2};
teste();
console.log(a); // 1

function teste() {
  let a = 3;
}
```

var x let

```
"use strict";
console.log(a);
var a = 1;

// undefined
```

```
"use strict";
console.log(a);
let a = 1;

// ReferenceError: a is not defined
```

```
"use strict";
let funcoesRetornamNumero = [];
for (var i=0 ; i<2 ; i++) {
  funcoesRetornamNumero.push(
     function() { return i; }
  );
}
console.log( funcoesRetornamNumero[0]() );

// 2
```

```
"use strict";
let funcoesRetornamNumero = [];
for (let i=0 ; i<2 ; i++) {
  funcoesRetornamNumero.push(
     function() { return i; }
  );
}
console.log( funcoesRetornamNumero[0]() );

// 0 (no debugger default do Firefox)
```

const

```
"use strict";
const A = 1;
console.log(A);

// 1
```

```
"use strict";
const A;
console.log(A);

// SyntaxError: Missing initializer in const declaration
```

```
"use strict";
console.log(A);
const A = 1;


// ReferenceError: A is not defined
```

```
"use strict";
const A = 1;

{const A = 2};
console.log(A);

// 1
```

arrow functions

```
"use strict";
var soma = (a, b) => a + b;
console.log(soma(1,2));

// 3
```

```
function Pessoa() {
    this.idade = 10;
    var self = this;

    // setTimeout === window.setTimeout
    setTimeout(function envelhecer() {
        self.idade++;
        console.log(p.idade);
    }, 1000);
}
var p = new Pessoa();

// 11
```

```
function Pessoa() {
    this.idade = 10;

    // setTimeout === window.setTimeout
    setTimeout( () => {
        this.idade++
        console.log(p.idade);
    } , 1000);
}
var p = new Pessoa();


// 11
```

arrow functions

```
document.addEventListener('click', function() {
  console.log(this);
} );

// document
```

```
document.addEventListener('click', () => { console.log(this); });. //?


// Window
```

arrow functions

```
<html>
  <head></head>
  <body>
    <button id='botao'>botao</button>

    <script>
      var handlerArrow = () => {  // window.handlerArrow === handlerArrow
        console.log(this);
      }
      document.getElementById('botao').addEventListener('click', handlerArrow); // window
      document.addEventListener('click', handlerArrow); // window

      var handlerConvencional = function() {  // window.handlerConvencional === handlerConvencional
        console.log(this);
      }
      document.getElementById('botao').addEventListener('click', handlerConvencional); // button
      document.addEventListener('click', handlerConvencional); // document
    </script>
  </body>
</html>
```

default function parameters

```
function a (b=1) {
  console.log(b);
}
a();

// 1
```

```
function a(b = 1, c = 2) {
  console.log(arguments.length);
}
a();

// 0
```

operadores rest e spread (...)

```
// rest

let a = (b, ...c) => console.log(c);
a(); // []

a(1, 2, 3); // [ 2, 3 ]
```

```
// spread

console.log(Math.max(1,9,5,3));
// 9

console.log(Math.max(...[1,9,5,3]));
// 9

console.log(Math.max(..."5814"));
// 8
```

for of

```
var jogadores = ["Jose", "Joao", "Danilo"];

for (let jogador of jogadores) {
  console.log(jogador);
}
// Jose
// Joao
// Danilo

// Uma das formas ES5
for (var i in jogadores) {
  console.log(jogadores[i]);
}
// Jose
// Joao
// Danilo
```

Extensões de objetos literais

```
"use strict";
let nome = "Marcos"; let dinamico = "sobrenome";

let usuario = {
    nome: "Wesley",
    [dinamico]: "Silva",
    getNome() {
        return this.nome;
    }
}
usuario.sobrenome = "Silva";

console.log(usuario.getNome());
// Wesley
```

Algumas outras extensões

```
let raios = [1, 2, 3];
let areas = Array.from(raios, (raio) => Math.PI * raio * raio);
console.log(areas); // testar
// [ 3.141592653589793, 12.566370614359172, 28.274333882308138 ]

console.log(Math.sign(100)); // 1
console.log(Math.sign(-200)); // -1

let v = 'NaN';
console.log(isNaN(v));
// true
console.log(Number.isNaN(v));
// false
```

Números em binário e octal

```
"use strict";
console.log(0o10);
// 8

console.log(010);
// 8 sem strict mode
// Syntax Error com strict mode

console.log(0b10);
// 2

console.log(0x10);
// 16 - Já existia no ES5
```

Template literals

```
"use strict";
var idade = 10;
console.log(`Idade: ${idade}`);

// Idade: 10
```

```
"use strict";
let idade = 10;

function mostrarIdade(template) {
  idade = 20;
  console.log(template);
}

mostrarIdade(`Idade: ${idade}`);

//Error
```

Template literals

```
"use strict";
var idade = 10;
var nome = "Joana";

function mostrarIdade(estaticos, ...dinamicos) {
  console.log(estaticos);
  console.log(dinamicos);
}

// CHAMADA DA FUNÇÃO SEM PARÊNTESES
mostrarIdade `A ${nome} tem ${idade} anos`;
// [ 'A ', ' tem ', ' anos' ] ??
// [ 'Joana', 10 ]
```

Desestruturação

```
"use strict";
let [r, s, t] = [1, 2, 3];
console.log(r, s, t);
// 1  2  3

let {x: a, y: b, z: c} = {x: 1, y: 2, z: 3};
console.log(a, b, c);
// 1  2  3

let [inicio, , ...fim] = "ABCDE";
console.log(inicio);
console.log(fim);
// A
// [ 'C', 'D', 'E' ]
```

Módulos

```
// 01.js
console.log("inicio");
import {nome, sobrenome as nomeVisualizacao} from "./02.js";
console.log(nomeVisualizacao);

// 02.js
export let nome = "Jose";
export let sobrenome = "Bonitinho";
console.log("Na externa.js");

// Saídas:
// Na externa.js
// inicio
// Bonitinho
```

Módulos

```
// 01.js
import teste from "./02.js";
console.log(teste);

// 02.js
let padrao = 100;
export default padrao;
export let teste=30;

// Saída:
// 100
```

Módulos

```
// 01.js
import * as lib from "./02.js";
lib.digaONumero(10);


// 02.js
export function digaONumero(numero) {
  console.log(numero);
}

// Saída:
// 10
```

Classes

```
class Pessoa {
  constructor() { console.log("Estou no construtor !") };

  getNome() { return this.nome; }

  setNome(nome) {
    this.nome = nome;
    return this;
  }

  static digaOla() { console.log("Olá"); }
}

console.log(typeof Pessoa); // function

let jose = new Pessoa; // Estou no construtor !

console.log(typeof jose); // object

console.log(jose.setNome("José").getNome()); // José

Pessoa.digaOla(); // Olá
```

Classes - herança

```
class Veiculo {
  constructor() {
    console.log("no veículo");
  }
}

class Carro extends Veiculo {
  constructor() {
    console.log("no carro");
    // Obrigatório, tendo ou não o constructor no pai
    // Caso contrário, this is not defined
    // Funciona também em objetos literais
    super();
  }
}

let carro = new Carro;
// no carro
// no veículo
```

Classes - herança

```
class Somador extends Array {
    sum() {
        let total = 0;
        this.map( v => total += v );
        return total;
    }
}

let s = new Somador;
s[0] = 4;
s.push(5);

console.log(s.sum());
// 9
```

Classes - namespace global

```
var nome = "Tobias";
console.log(window.nome);
// "Tobias"

function mostrarNome() { }
console.log(window.mostrarNome);
// function mostrarNome() { }

class Teste {
}
console.log(window.Teste);
// undefined
```

Promise

Um objeto **Promise** é usado para processamento assincrono e demorado.

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Promise

Promises are an alternative to callbacks for delivering the results of an asynchronous computation.

http://exploringjs.com/es6/ch_promises.html

Promise

```
consultar().then(function(resultado) {
    console.log(`Consulta resolvida. Resultado: ${resultado}`);
});

console.log("depois da invocação de consultar()");

function consultar() {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            console.log("resolvendo a promise de consulta...");
            resolve('a consulta retornou o nome João');
        }, 1000);
        console.log("final da declaração da promise de consulta");
    });
}

// final da declaração da promise de consulta
// depois da invocação de consultar()
// resolvendo a promise de consulta...
// Consulta resolvida. Resultado: a consulta retornou o nome João
```

Promise

```
$.ajax({
  url: "http://fiddle.jshell.net/favicon.png",
  beforeSend: function( xhr ) {
    xhr.overrideMimeType( "text/plain; charset=x-user-defined" );
  }
})
  .done(function( data ) {
    if ( console && console.log ) {
      console.log( "Sample of data:", data.slice( 0, 100 ) );
    }
  });
```

The jqXHR objects returned by `$.ajax()` as of jQuery 1.5 implement the Promise interface, giving them all the properties, methods, and behavior of a Promise (see Deferred object for more information). These methods take

Promise

```
consultar().then(
    () => console.log("Consulta resolvida"),
    () => console.log("Consulta rejeitada")
);

console.log("depois da invocação de consultar()");

function consultar() {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            console.log("resolvendo a promise de consulta...");
            reject('a consulta retornou o nome João');
        }, 1000);
        console.log("final da declaração da promise de consulta");
    });
}

// final da declaração da promise de consulta
// depois da invocação de consultar()
// resolvendo a promise de consulta...
// Consulta rejeitada
```

Promise

```
consultar()
    .then(function(resultado) {
        console.log(`Consulta resolvida. Resultado: ${resultado}`);
        return atualizar();
      })
     .then((resultado) => console.log(“reject”));

function consultar() {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            resolve('a consulta retornou o nome João');
        }, 1000);
    });
}

function atualizar() {
    return new Promise((resolve, reject) => resolve('Atualização efetuada via promise'));
}

// Consulta resolvida. Resultado: a consulta retornou o nome João
// Atualizacao resolvida. Resultado: Atualização efetuada via promise
```

Symbol

// “O **symbol** é um tipo de dado único e imutável e pode ser usado como um identificador para propriedades de objeto.”

// https://developer.mozilla.org/

```
var sym1 = Symbol("foo"); // sem new. Com new gera TypeError
var sym2 = Symbol("foo");

console.log(sym1 == sym2);
// false

console.log(typeof sym1);
// symbol
```

Alguns símbolos existentes

```
class Pessoa {
    constructor(nome) {
        this.nome = nome;
    }

    get [Symbol.toStringTag]() {
        return "Pessoa";
    }

    [Symbol.toPrimitive](hint) {
        console.log("Acionada a transformação para primitivo");
        if (hint === "string") return this.nome;
    }
};

let pessoa = new Pessoa("Genésio")
console.log(pessoa.toString()); // [object Pessoa] e não  [object Object]

console.log(`Estou falando com o ${pessoa}`);
// Acionada a transformação para primitivo
// Estou falando com o Genésio
```

Map e WeakMap

```
class Comparador {
    equals(a, b) {
        return a === b;
    }
}

let register = new Map();
register.set('comparador', new Comparador);

console.log(register.get('comparador').equals(1, 2));
// false

console.log(register.has('comparador'));
// true

console.log(register.size);
// 1

// WeakMap:
// - Garbage collector remove chaves do WeakMap que não são mais referenciadas
// - Não tem .size
```

Set e WeakSet

```
let fabricantes = new Set;

fabricantes.add('Motorola');
fabricantes.add('Samsung');
fabricantes.add('Motorola');

console.log(fabricantes.values());
// SetIterator { 'Motorola', 'Samsung' }

for (fabricante of fabricantes) {
    console.log(fabricante);
}
// Motorola
// Samsung

console.log(fabricantes.size);
// 2


// WeakSet
    -    Semelhante à comparação entre Map e WeakMap
```

Iterators

“Um iterador acessa os itens de uma coleção um por vez, enquanto mantém controle da sua posição atual na sequência. Ele tem um método next() que retorna o próximo item da sequência. Esse método retorna um objeto com duas propriedades: done e value”

https://github.com/alexmoreno/ES6-para-humanos#15-iteradores-iterators

Iterators

```
let valores = [100, 200];
let iteradorValores = valores[Symbol.iterator]();

console.log(iteradorValores.next());
// { value: 100, done: false }

console.log(iteradorValores.next());
// { value: 200, done: false }

console.log(iteradorValores.next());
// { value: undefined, done: true }

let iteradorValores2 = valores[Symbol.iterator]();
for (item of iteradorValores2) {
    console.log(item);
}
// 100
// 200
```

Iterators

```
let valores = [2, 4, 6, 8];
valores[Symbol.iterator] = function() {
    return {
        indiceAtual: -1,
        next(item) {
            this.indiceAtual++;
            if (this.indiceAtual >= valores.length) return {value: undefined, done: true}

            return {value: 10 * valores[this.indiceAtual], done: false};
        }
    }
}

for (let item of valores) {
    console.log(item);
}
// 20
// 40
// 60
// 80
// no Firefox
```

Generators

> “Funções geradoras são um novo recurso que permite que uma função gere quantos valores forem necessários retornando um objeto que pode ser iterado para puxar mais valores da função, um por vez.”

https://github.com/alexmoreno/ES6-para-humanos#15-iteradores-iterators

Generators

```
// function* generator() {
function *generator() {
    yield 100;
    yield 200;
}

let iterador = generator();

console.log(iterador.next()); // { value: 100, done: false }

console.log(iterador.next()); // { value: 200, done: false }

console.log(iterador.next()); // { value: undefined, done: true }
```

Generators

```
function *gerador() {
    yield* [100, 200];
}

let iterador = gerador();

for (let item of iterador) {
    console.log(item);
}
// 100
// 200
```

Generators

```
let contador = 0;
function *generator() {
    contador++;
    yield 100;
    console.log("O generator foi chamado entre 100 e 200");
    yield 200;
    console.log(`O generator foi chamado ${contador}`);
}

let iterador = generator();
console.log(iterador.next()); // { value: 100, done: false }

console.log(iterador.next());
// O generator foi chamado entre 100 e 200
// { value: 200, done: false }

console.log(iterador.next());
// O generator foi chamado 1
// { value: undefined, done: true }

console.log(iterador.next()); // { value: undefined, done: true }
```

Proxy e Reflect API

"The **Proxy** object is used to define custom behavior for fundamental operations (e.g. property lookup, assignment, enumeration, function invocation, etc)."

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Proxy

"**Reflect** is a built-in object that provides methods for interceptable JavaScript operations. "

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Reflect

Proxy API

```
class Piloto {
    constructor(nome, equipe, salario) {
        this._nome = nome;
        this._equipe = equipe;
        this._salario = salario;
    }
}

let massa = new Piloto('Felipe Massa', 'Williams', 5000000);

let massaProxy = new Proxy(massa, {
    get: function(target, prop, receiver) {
        if (prop === 'salario') {
            return 'R$ ' + Reflect.get(target, '_salario') + ",00";
        }
    }
});

console.log(massaProxy.salario);
// R$ 5000000,00
```

Reflect API

```
let carro = {
    cor: 'Azul',
    ano: 2009
}

console.log(Reflect.ownKeys(carro));
// [ 'cor', 'ano' ]

Reflect.deleteProperty(carro, 'ano');
carro.combustivel = 'Gasolina';
Reflect.preventExtensions(carro);
carro.placa = 'XTS-9911';
console.log(Reflect.ownKeys(carro));
// [ 'cor', 'combustivel' ]

console.log(Reflect.getOwnPropertyDescriptor(carro, 'cor'));
// { value: 'Azul',  writable: true,  enumerable: true,  configurable: true }

// OBS: Semelhante ao Object.getOwnPropertyDescriptor
// https://mzl.la/2bBNu9E
```

Mas não é só isto. Falta...

- Aprofudamento nos tópicos
  - Ex: Generators
    - return
    - throw
    - recebimento de valores no generator
  - Reflect API
  - Proxy API
  - etc
- Unicode code point escapes
- ArrayBuffer, Dataview
- etc

Babel

Babel


Babel is a JavaScript compiler.
Use next generation JavaScript, today.

Babel x Traceour

Babel:

- Mais stars
- Mais novo
- Código gerado mais legível
- Maior percentual no kangax

Setup básico

```
$ npm init
$ npm install --save-dev babel-cli
$ npm install babel-preset-es2015 --save-dev
$ echo '{"presets": ["es2015"]}' > .babelrc

# Presets?
# Plugins?
```

Teste simples

```
$ mkdir src dist
$ cd src
$ vi 01.js

$ cd ..
$ ./node_modules/.bin/babel src -d dist
```

```
// src/02.js

function dobrador(i) {
    return i * 2;
}
```

```
// src/01.js

import dobrador from "./02.js";
console.log(dobrador(10));
```

```
// dist/01.js

"use strict";
var dobrador = function dobrador(i) {
  return i * 2;
};
```

Gulp

```
$ # Complemento do setup básico

$ npm install --save-dev gulp gulp-babel
$ gulp
```

```
// gulpfile.js
// Node 6 (sintaxe ES2015) no gulpfile.js

const gulp = require('gulp');
const babel = require('gulp-babel');

gulp.task('default', () => {
    gulp.src('src/*.js')
        .pipe(babel())
        .pipe(gulp.dest('dist'));
});
```

Browserify


browserify
Browserify lets you require('modules') in the browser by bundling up all of your dependencies.

Import/export ES6 no Node 6

```
$ # Somente com o setup básico
$ # Ainda sem o Browserify

$  ./node_modules/.bin/babel-node src/index.js
# 20
```

```
// src/index.js

import {dobrador} from './dobrador';
console.log(dobrador(10));
```

```
// src/dobrador.js

export let dobrador = i => i * 2;
```

Browserify

```
$ npm install --save-dev browserify babelify
$ ./node_modules/.bin/browserify -t babelify -e src/index.js -o dist/bundle.js

$ # Execução via node (apenas para teste)
$ node dist/bundle.js
$ # 20
```

```
// index.js

import {dobrador} from './dobrador';
console.log(dobrador(10));
```

```
// dobrador.js

export let dobrador = i => i * 2;
```

Browserify

```
// index.html
<!DOCTYPE html>
<html>
    <head>
        <script src="bundle.js"></script>
    </head>
    <body></body>
</html>
```

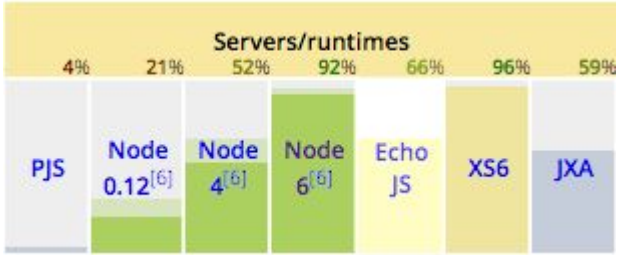
Polyfill

```
$ npm install --save babel-polyfill
```

```
// index.html
<!DOCTYPE html>
<html>
    <head>
        <script src="node_modules/babel-polyfill/dist/polyfill.js"></script>
        ...
    </head>
    <body></body>
</html>
```

Node.js

Usando com Node



- Diretamente (sem módulos ES2015)
- ./node_modules/.bin/babel-node
  - REPL
  - transpile seguido de execução (em 1 etapa)
- Transpile via babel e execução via node (em 2 etapas)

Referência

http://lmgtfy.com/?q=es2015

“Cabô”