New Gen JavaScript

Marcin Kopacz (marcin.p.kopacz@jpmorgan.com)

Object Oriented Developer in Kapital Team

Available at: http://slides.com/chyzwar/new-js

git: https://github.com/Chyzwar/TechSymposium.git

Plan:

1. Brief History Lesson.
2. Whats new.
3. Can I use it?
4. Why You should care.
5. Challenges & Problems.
6. Q&A Session.
7. Lets make Twitter.

Brief History Lesson

1. Beginning

1970s - Scheme dynamic functional language. Lisp dialect created in MIT Guy L. Steele and Gerald Jay Sussman

1980 - Smalltalk is released as first modern Object Oriented language, Alan Kay in   Xerox PARC

1980s - Self high performance dialect of Smalltalk (no classes, proptype inheritance), Sun Microsystems

1995 - Java general-purpose language, class based object oriented with C like syntax. Sun Microsystem

[Brief History of JavaScript] http://foorious.com/articles/brief-history-of-javascript/

2. What could possibly go wrong?

1995 Brendan Eich was hire by Netscape to build scripting language for browser in 10 days.

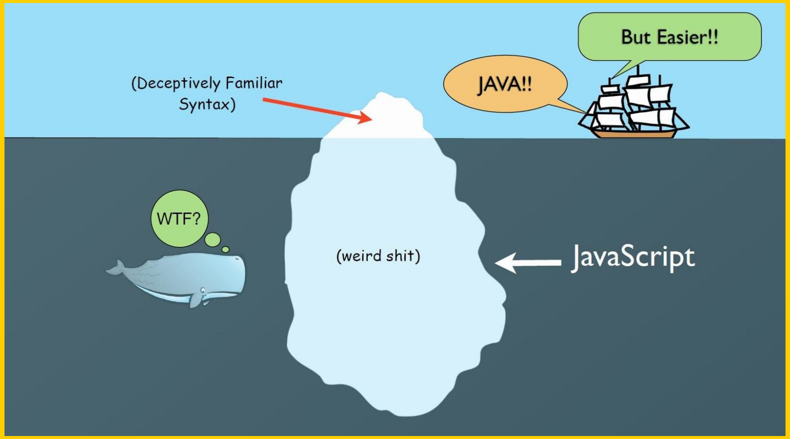
JavaScript = Java + Scheme(Lisp) + *Self*(Smalltalk)

High-level, dynamic, untyped, and interpreted programming language. Prototype-based with first-class functions.

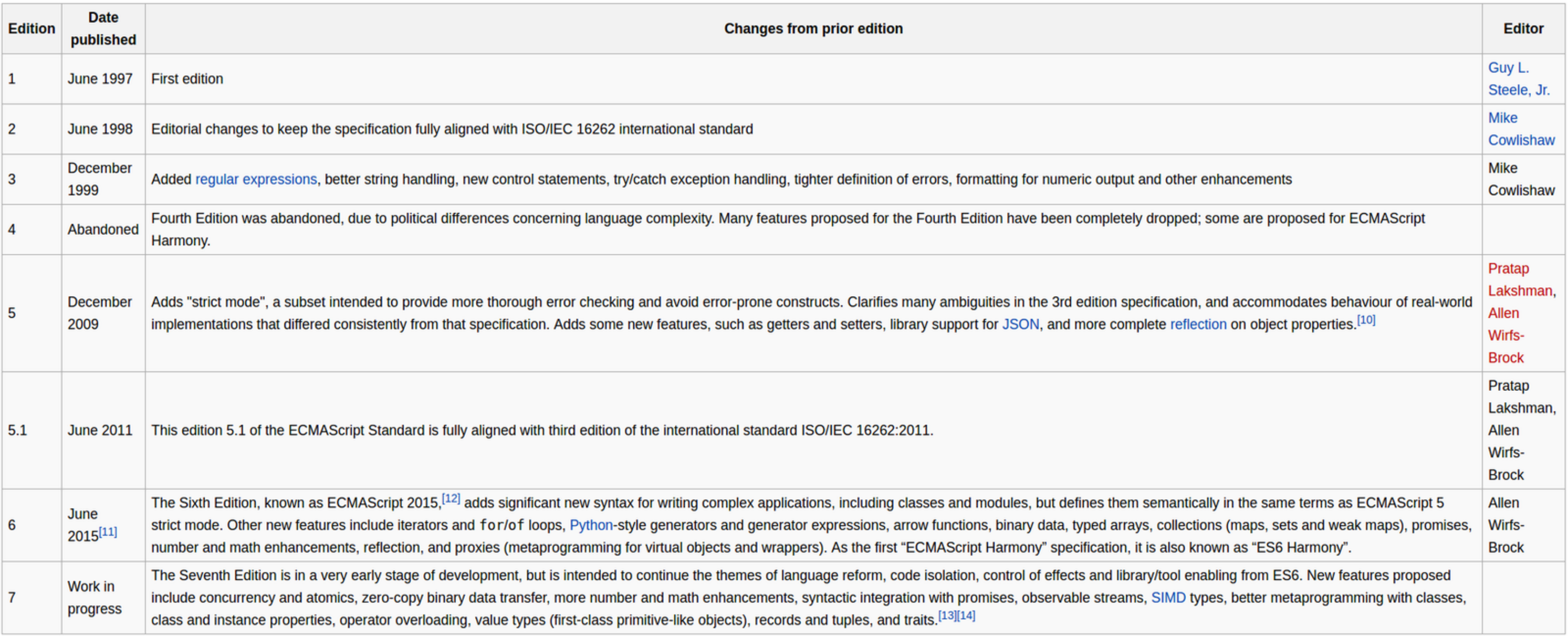
[Brendan Eich Blog] https://brendaneich.com/2011/06/new-javascript-engine-module-owner/

3. Why so much hate.

- Overuse of global scope
- Functional scope
- Browser incompatibilities
- DOM API (jQuery)
- Type coercion
- Prototype chain
- Lack of modules, spaghetti code



4. What happened next.



Edition	Date published	Changes from prior edition	Editor
1	June 1997	First edition	Guy L. Steele, Jr.
2	June 1998	Editorial changes to keep the specification fully aligned with ISO/IEC 16262 international standard	Mike Cowlishaw
3	December 1999	Added regular expressions, better string handling, new control statements, try/catch exception handling, tighter definition of errors, formatting for numeric output and other enhancements	Mike Cowlishaw
4	Abandoned	Fourth Edition was abandoned, due to political differences concerning language complexity. Many features proposed for the Fourth Edition have been completely dropped; some are proposed for ECMAScript Harmony.	
5	December 2009	Adds "strict mode", a subset intended to provide more thorough error checking and avoid error-prone constructs. Clarifies many ambiguities in the 3rd edition specification, and accommodates behaviour of real-world implementations that differed consistently from that specification. Adds some new features, such as getters and setters, library support for JSON, and more complete reflection on object properties.[10]	Pratap Lakshman, Allen Wirfs-Brock
5.1	June 2011	This edition 5.1 of the ECMAScript Standard is fully aligned with third edition of the international standard ISO/IEC 16262:2011.	Pratap Lakshman, Allen Wirfs-Brock
6	June 2015[11]	The Sixth Edition, known as ECMAScript 2015,[12] adds significant new syntax for writing complex applications, including classes and modules, but defines them semantically in the same terms as ECMAScript 5 strict mode. Other new features include iterators and `for/of` loops, Python-style generators and generator expressions, arrow functions, binary data, typed arrays, collections (maps, sets and weak maps), promises, number and math enhancements, reflection, and proxies (metaprogramming for virtual objects and wrappers). As the first “ECMAScript Harmony” specification, it is also known as “ES6 Harmony”.	Allen Wirfs-Brock
7	Work in progress	The Seventh Edition is in a very early stage of development, but is intended to continue the themes of language reform, code isolation, control of effects and library/tool enabling from ES6. New features proposed include concurrency and atomics, zero-copy binary data transfer, more number and math enhancements, syntactic integration with promises, observable streams, SIMD types, better metaprogramming with classes, class and instance properties, operator overloading, value types (first-class primitive-like objects), records and tuples, and traits.[13][14]	

https://en.wikipedia.org/wiki/ECMAScript

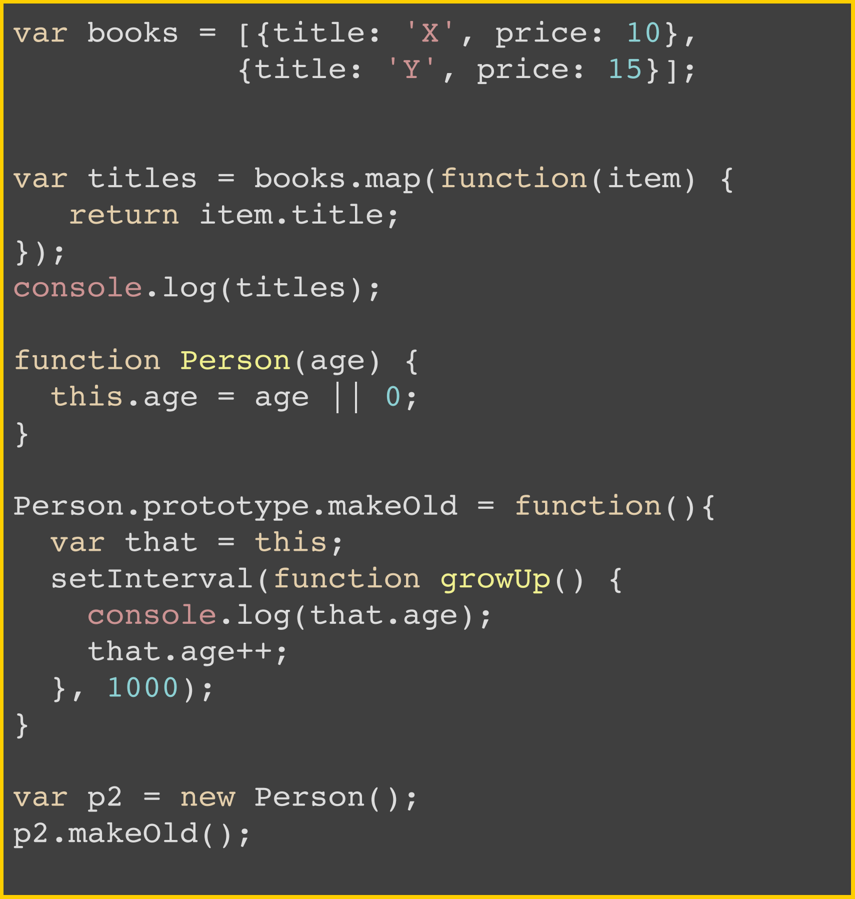
Get a repl

Babel Repl: https://babeljs.io/repl/

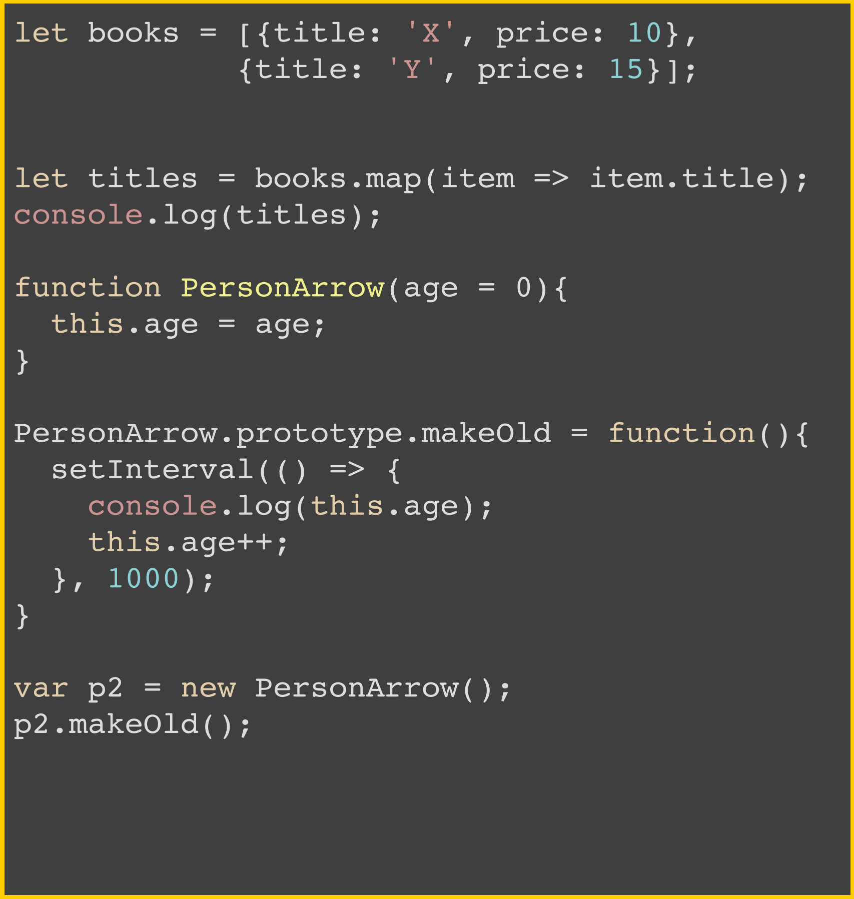
Scratch.js: https://goo.gl/9E6uUY

What New?

1. Arrow function (fat arrow).

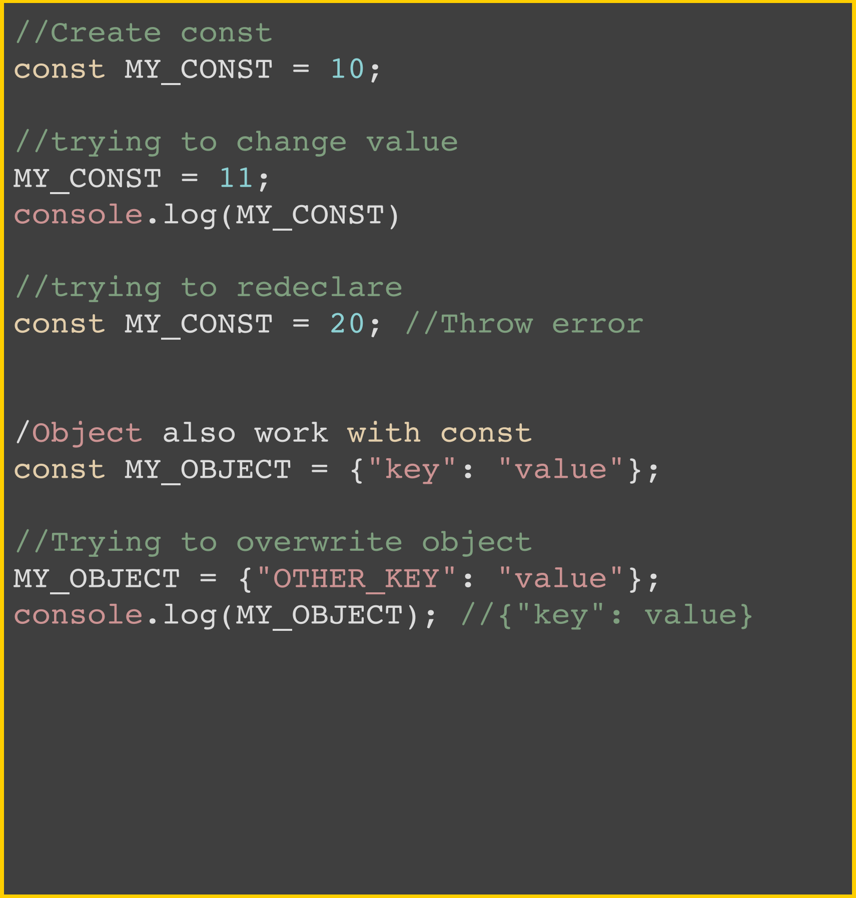


```
var books = [{title: 'X', price: 10},
            {title: 'Y', price: 15}];


var titles = books.map(function(item) {
   return item.title;
});
console.log(titles);

function Person(age) {
  this.age = age || 0;
}

Person.prototype.makeOld = function(){
  var that = this;
  setInterval(function growUp() {
    console.log(that.age);
    that.age++;
  }, 1000);
}

var p2 = new Person();
p2.makeOld();
```

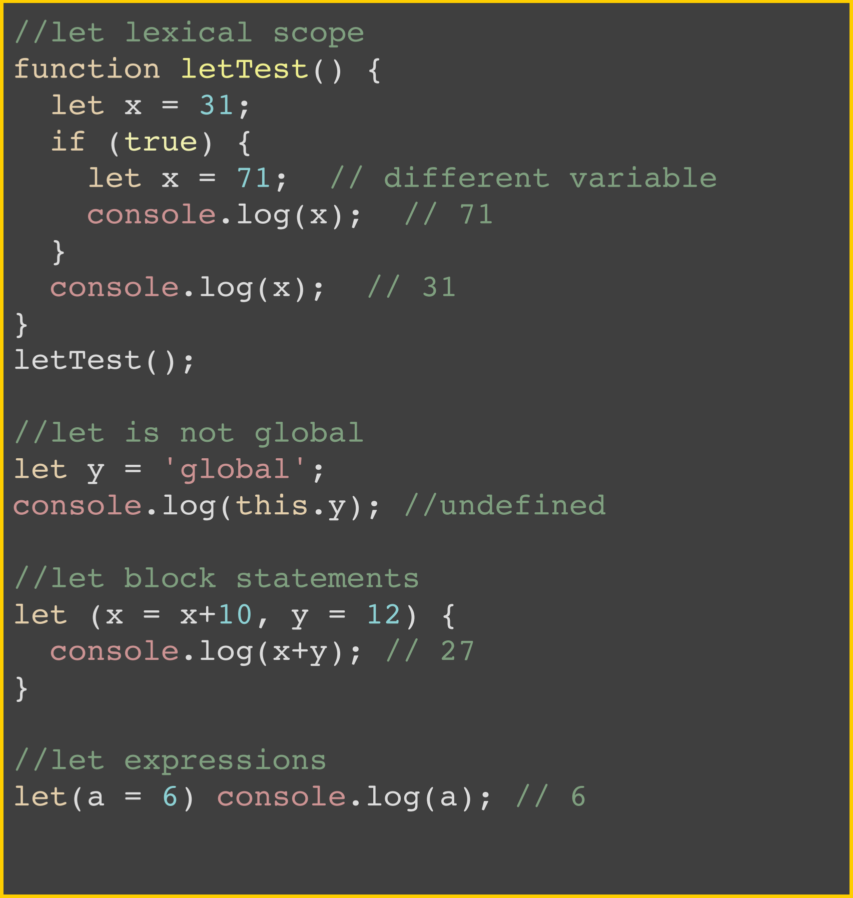


```
let books = [{title: 'X', price: 10},
            {title: 'Y', price: 15}];


let titles = books.map(item => item.title);
console.log(titles);

function PersonArrow(age = 0){
  this.age = age;
}

PersonArrow.prototype.makeOld = function(){
  setInterval(() => {
    console.log(this.age);
    this.age++;
  }, 1000);
}

var p2 = new PersonArrow();
p2.makeOld();
```

2. Let and const.

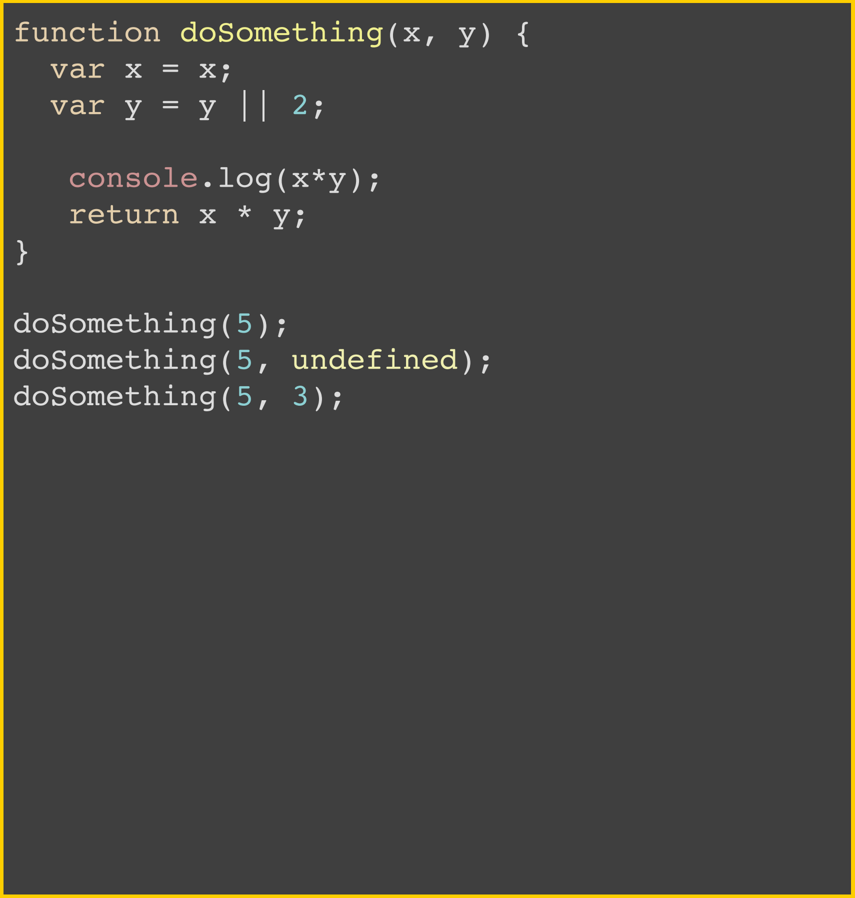


```
//Create const
const MY_CONST = 10;

//trying to change value
MY_CONST = 11;
console.log(MY_CONST)

//trying to redeclare
const MY_CONST = 20; //Throw error


/Object also work with const
const MY_OBJECT = {"key": "value"};

//Trying to overwrite object
MY_OBJECT = {"OTHER_KEY": "value"};
console.log(MY_OBJECT); //{"key": value}
```

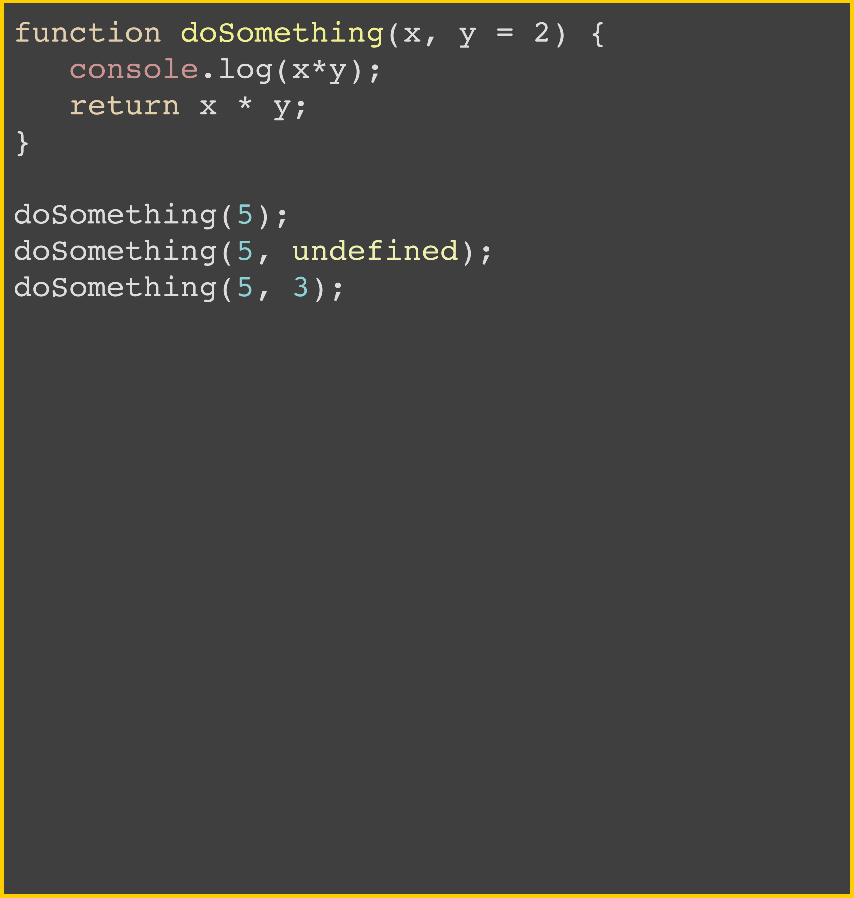


```
//let lexical scope
function letTest() {
  let x = 31;
  if (true) {
    let x = 71;  // different variable
    console.log(x);  // 71
  }
  console.log(x);  // 31
}
letTest();

//let is not global
let y = 'global';
console.log(this.y); //undefined

//let block statements
let (x = x+10, y = 12) {
  console.log(x+y); // 27
}

//let expressions
let(a = 6) console.log(a); // 6
```

3. Default Values.

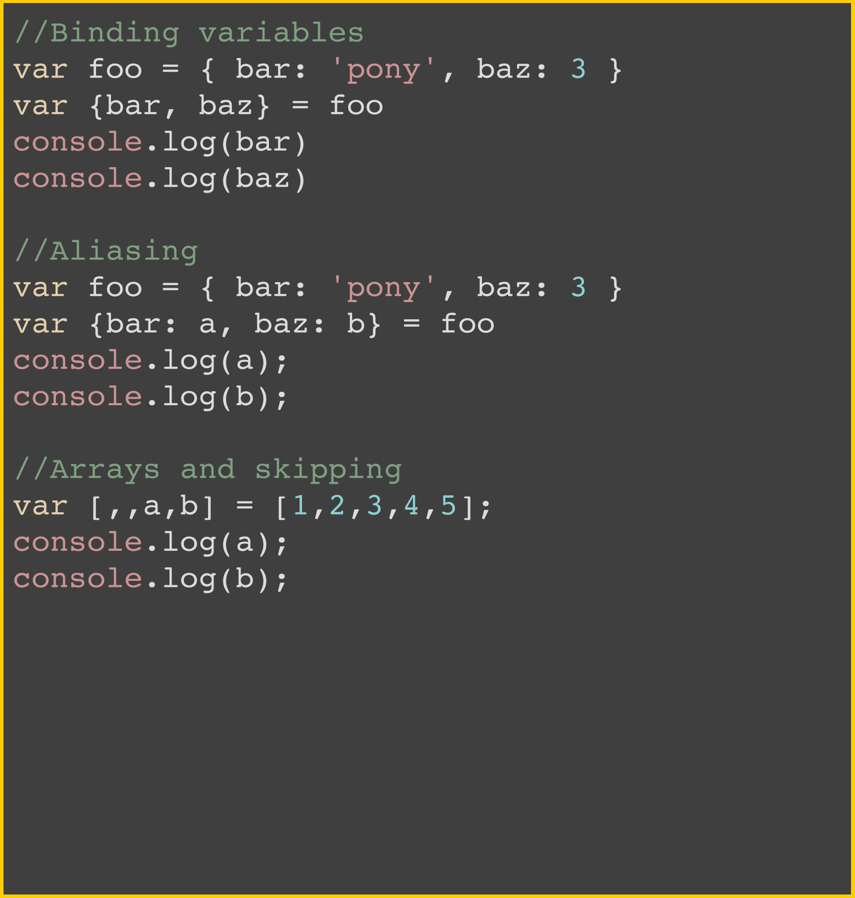


```
function doSomething(x, y) {
  var x = x;
  var y = y || 2;

   console.log(x*y);
   return x * y;
}

doSomething(5);
doSomething(5, undefined);
doSomething(5, 3);
```

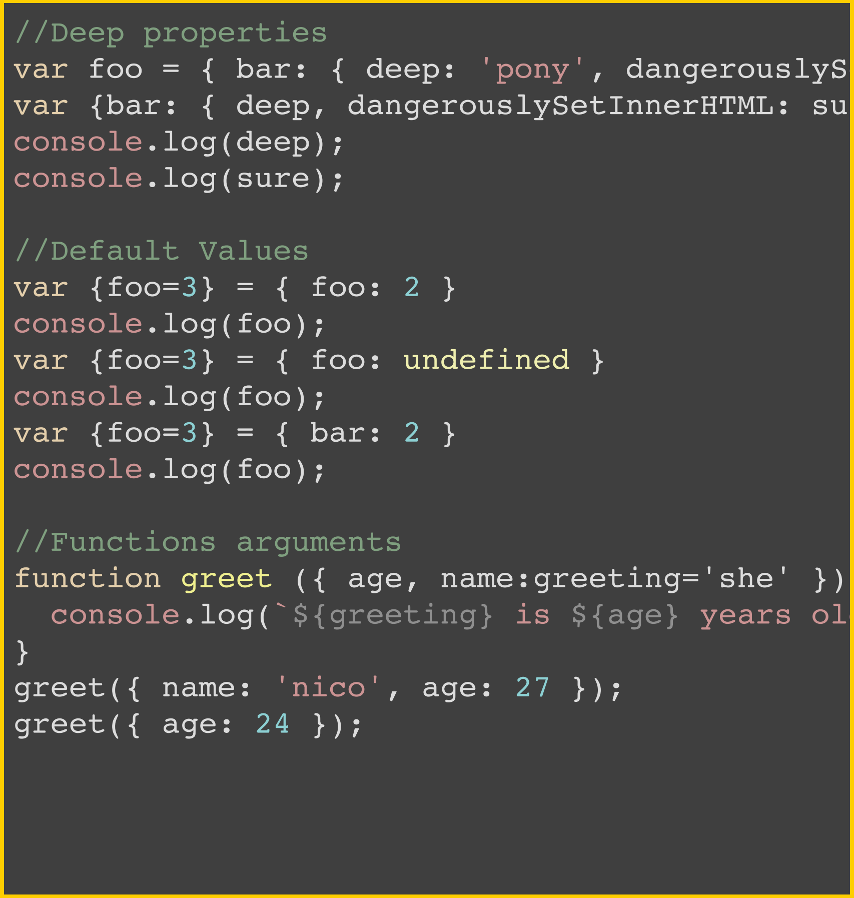


```
function doSomething(x, y = 2) {
   console.log(x*y);
   return x * y;
}

doSomething(5);
doSomething(5, undefined);
doSomething(5, 3);
```

4. Destructuring.



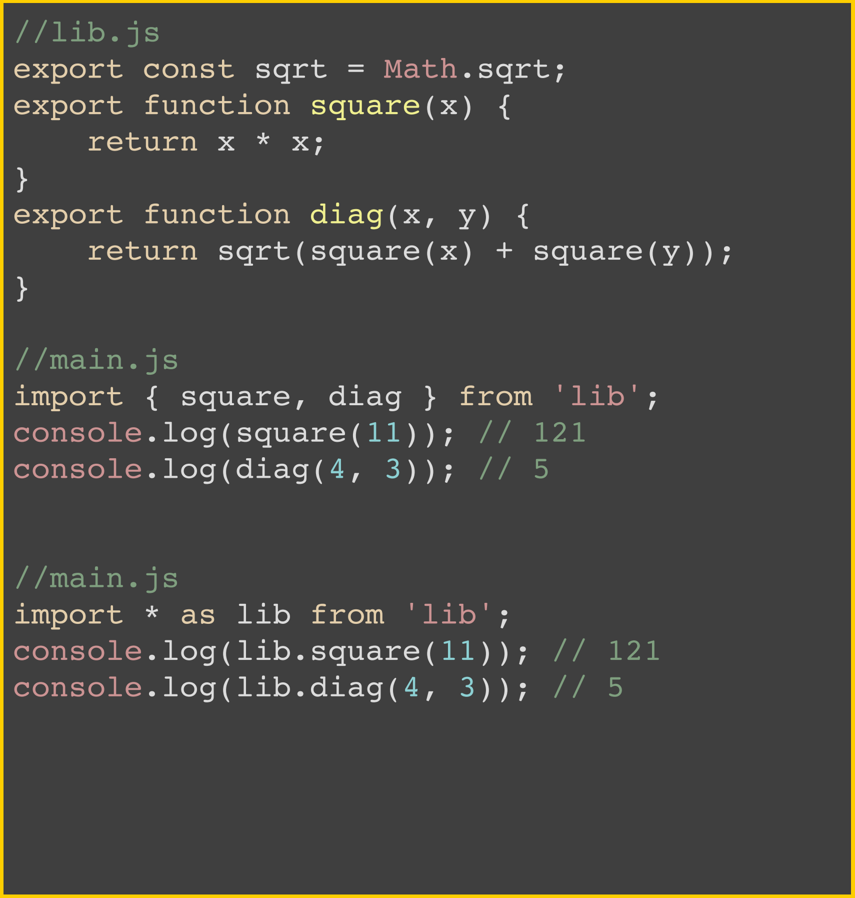
```
//Binding variables
var foo = { bar: 'pony', baz: 3 }
var {bar, baz} = foo
console.log(bar)
console.log(baz)

//Aliasing
var foo = { bar: 'pony', baz: 3 }
var {bar: a, baz: b} = foo
console.log(a);
console.log(b);

//Arrays and skipping
var [,,a,b] = [1,2,3,4,5];
console.log(a);
console.log(b);
```



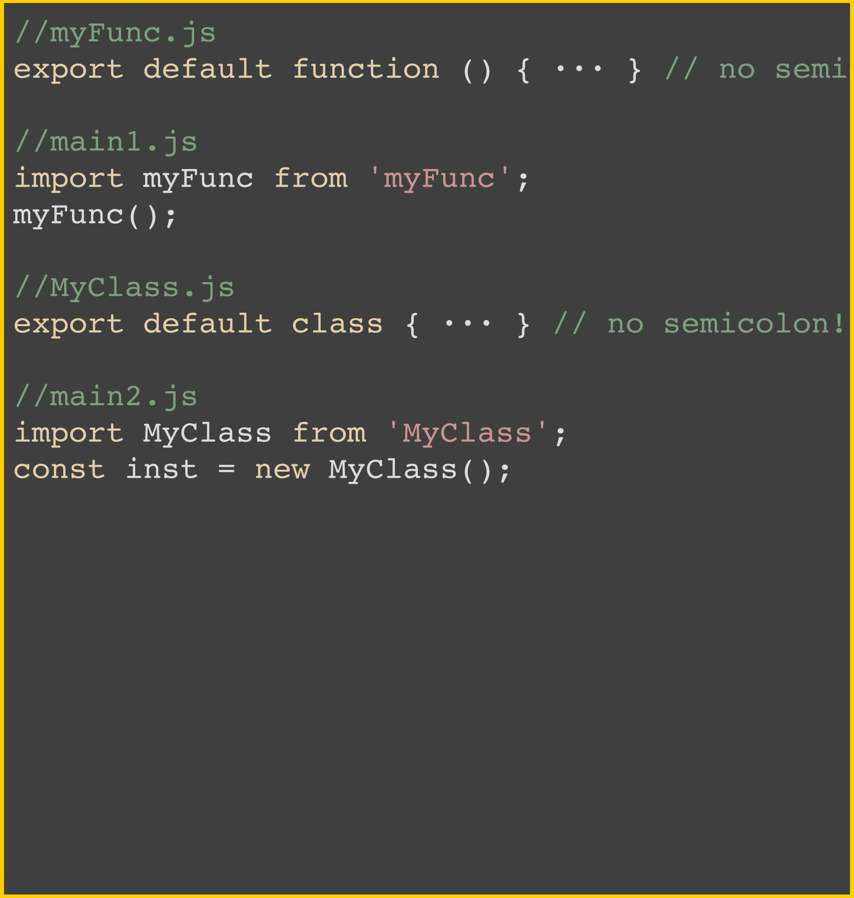
```
//Deep properties
var foo = { bar: { deep: 'pony', dangerouslyS
var {bar: { deep, dangerouslySetInnerHTML: su
console.log(deep);
console.log(sure);

//Default Values
var {foo=3} = { foo: 2 }
console.log(foo);
var {foo=3} = { foo: undefined }
console.log(foo);
var {foo=3} = { bar: 2 }
console.log(foo);

//Functions arguments
function greet ({ age, name:greeting='she' })
  console.log(`${greeting} is ${age} years ol
}
greet({ name: 'nico', age: 27 });
greet({ age: 24 });
```

[Destructuring] https://ponyfoo.com/articles/es6-destructuring-in-depth

5. Modules

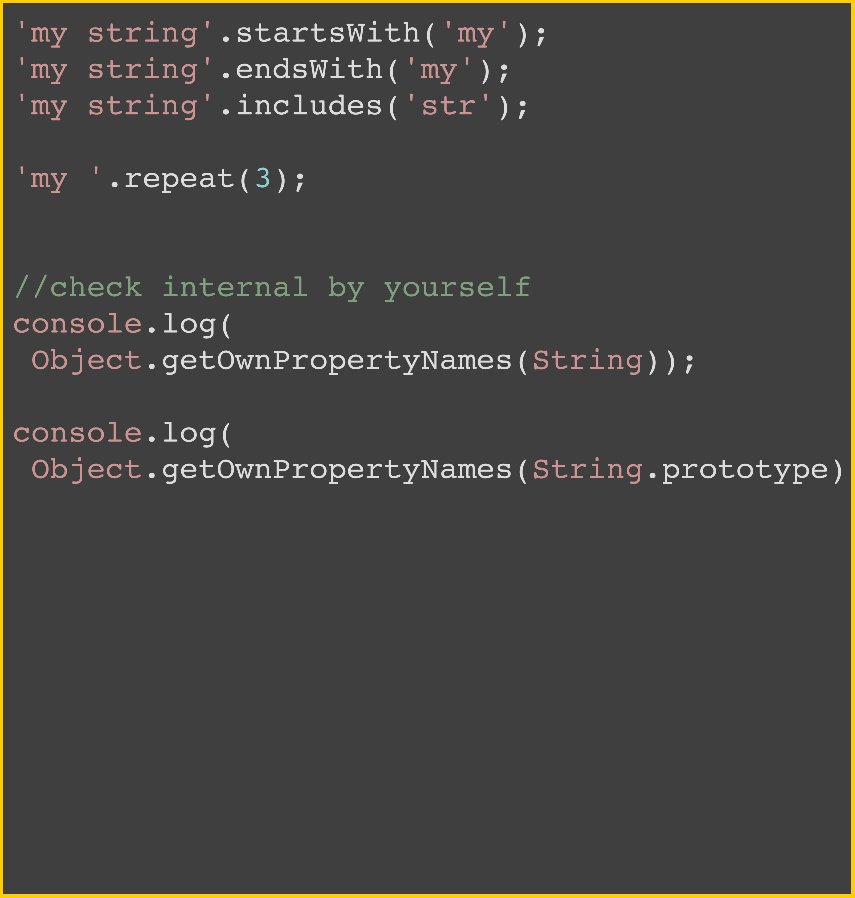


```
//lib.js
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}

//main.js
import { square, diag } from 'lib';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5


//main.js
import * as lib from 'lib';
console.log(lib.square(11)); // 121
console.log(lib.diag(4, 3)); // 5
```

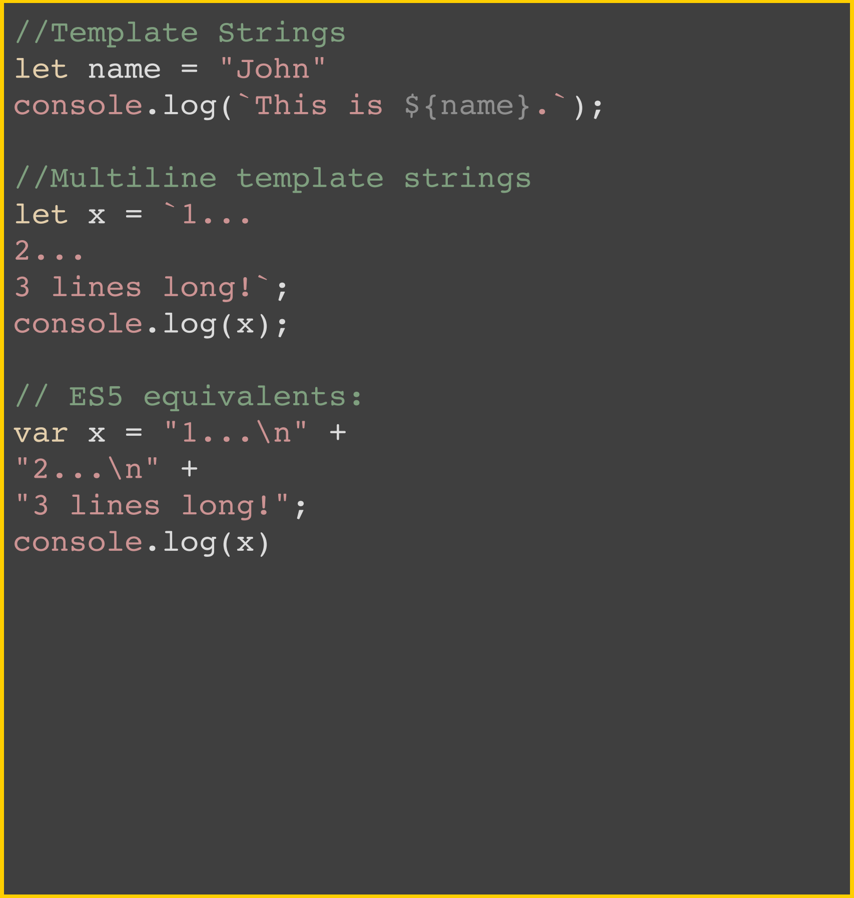


```
//myFunc.js
export default function () { ··· } // no semi

//main1.js
import myFunc from 'myFunc';
myFunc();

//MyClass.js
export default class { ··· } // no semicolon!

//main2.js
import MyClass from 'MyClass';
const inst = new MyClass();
```

6. Better Strings and Templates.

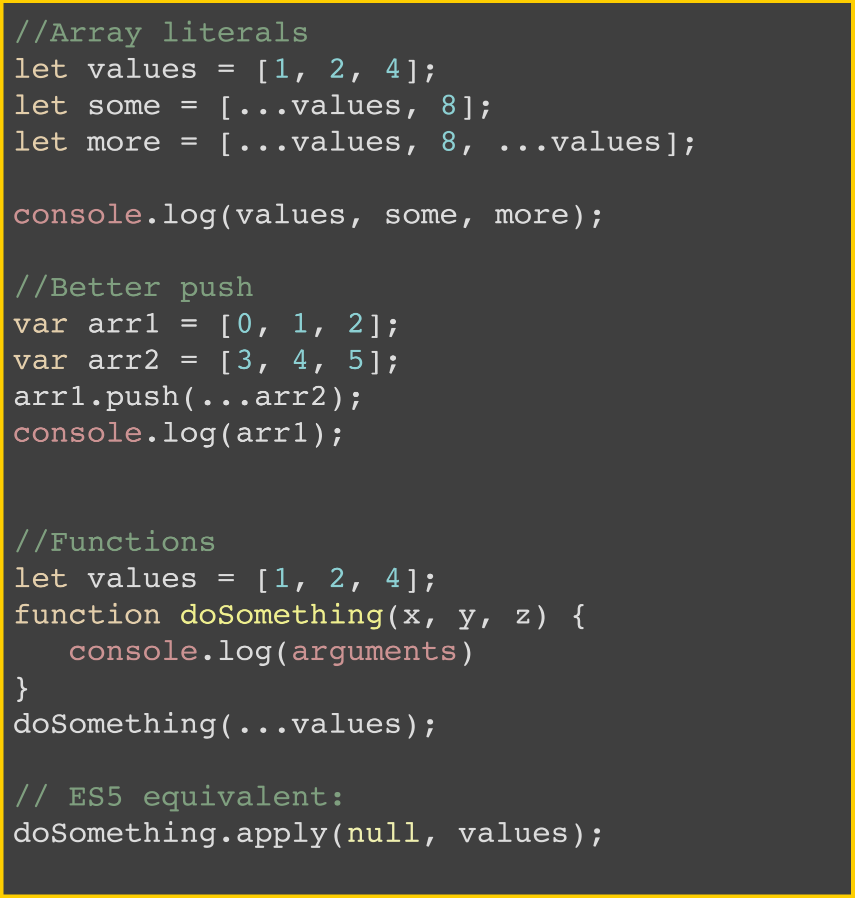


```
'my string'.startsWith('my');
'my string'.endsWith('my');
'my string'.includes('str');

'my '.repeat(3);


//check internal by yourself
console.log(
 Object.getOwnPropertyNames(String));

console.log(
 Object.getOwnPropertyNames(String.prototype)
```

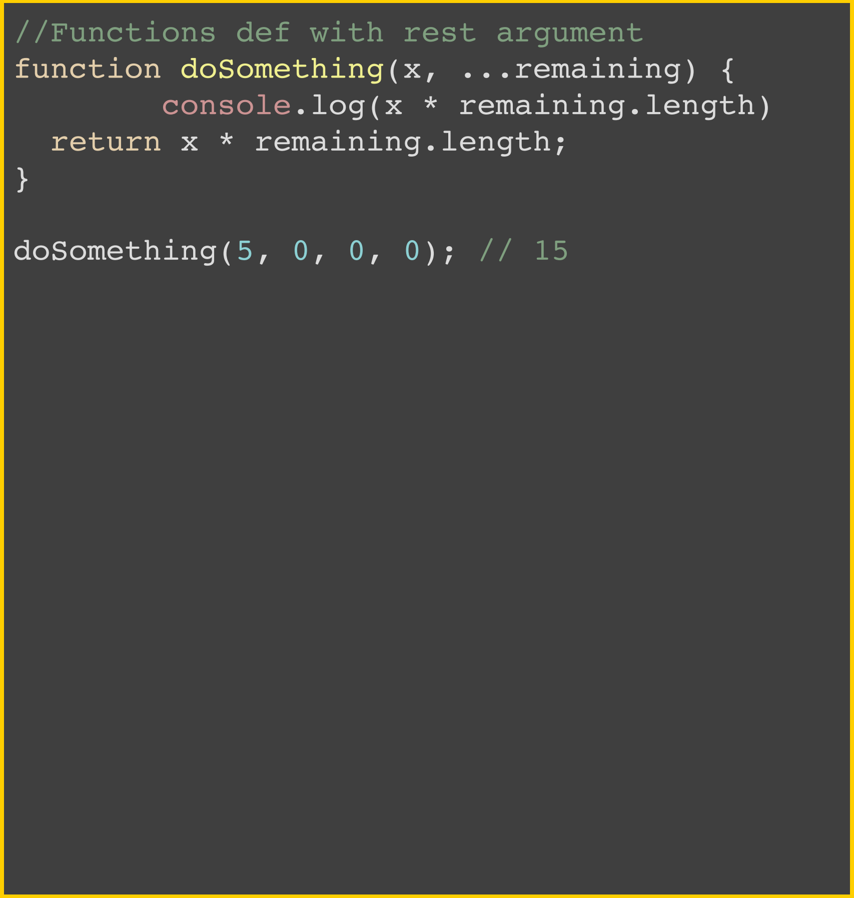


```
//Template Strings
let name = "John"
console.log(`This is ${name}.`);

//Multiline template strings
let x = `1...
2...
3 lines long!`;
console.log(x);

// ES5 equivalents:
var x = "1...\n" +
"2...\n" +
"3 lines long!";
console.log(x)
```

7. Spread and Rest operators.

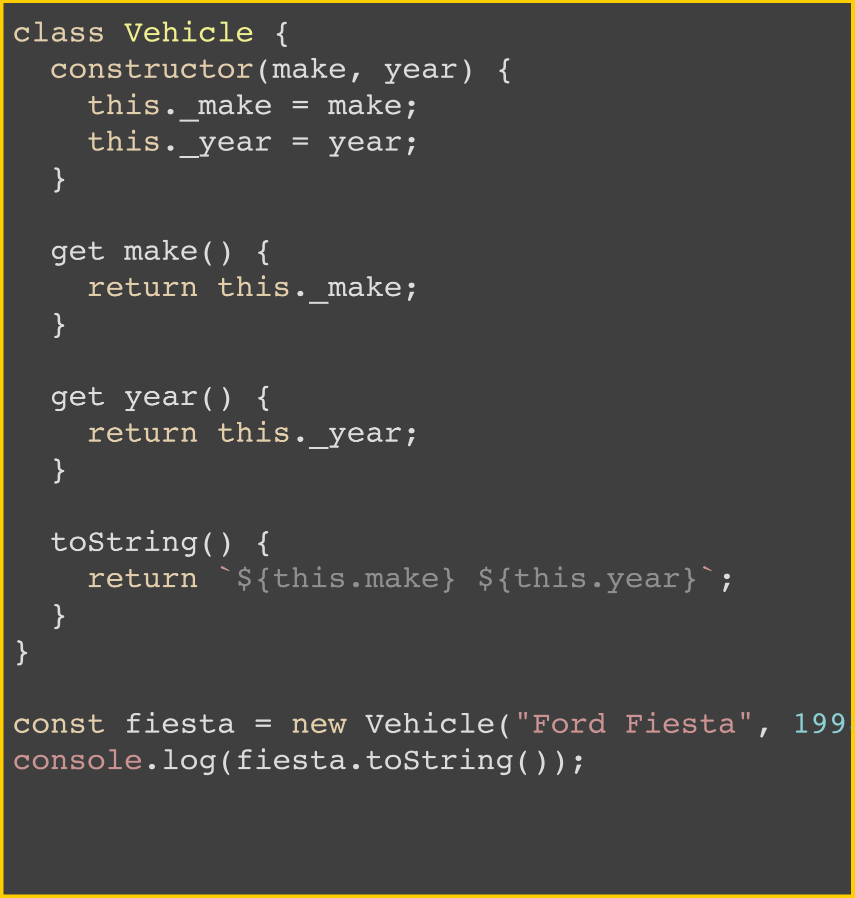


```
//Array literals
let values = [1, 2, 4];
let some = [...values, 8];
let more = [...values, 8, ...values];

console.log(values, some, more);

//Better push
var arr1 = [0, 1, 2];
var arr2 = [3, 4, 5];
arr1.push(...arr2);
console.log(arr1);


//Functions
let values = [1, 2, 4];
function doSomething(x, y, z) {
   console.log(arguments)
}
doSomething(...values);

// ES5 equivalent:
doSomething.apply(null, values);
```

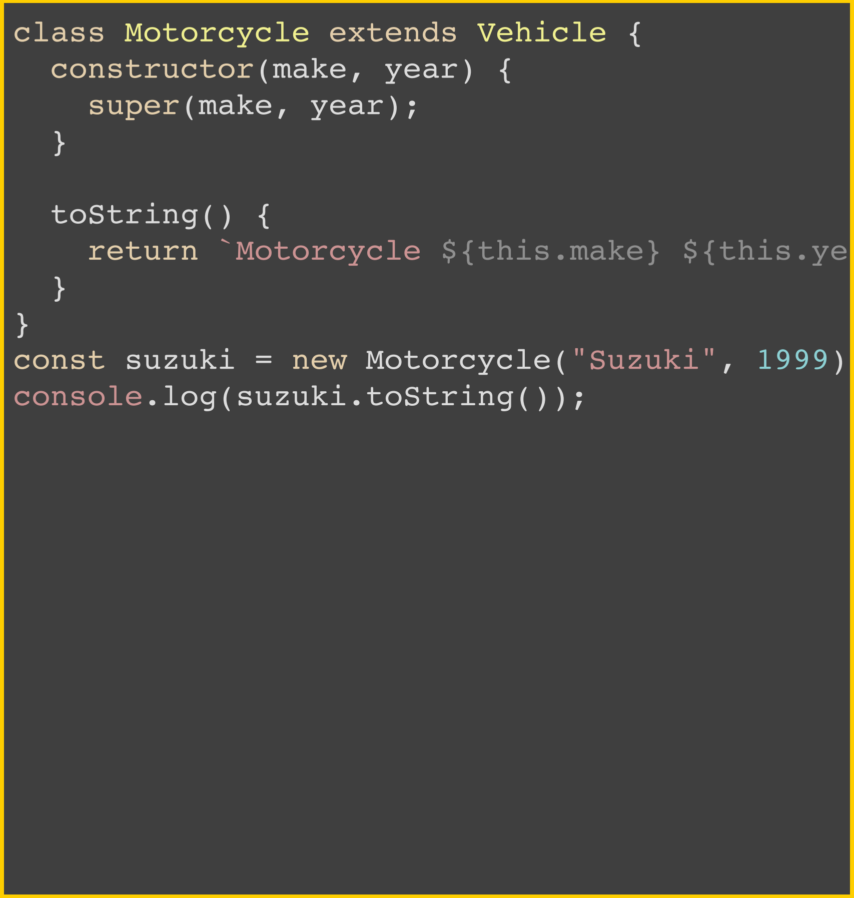


```
//Functions def with rest argument
function doSomething(x, ...remaining) {
        console.log(x * remaining.length)
  return x * remaining.length;
}

doSomething(5, 0, 0, 0); // 15
```

8. Classes.

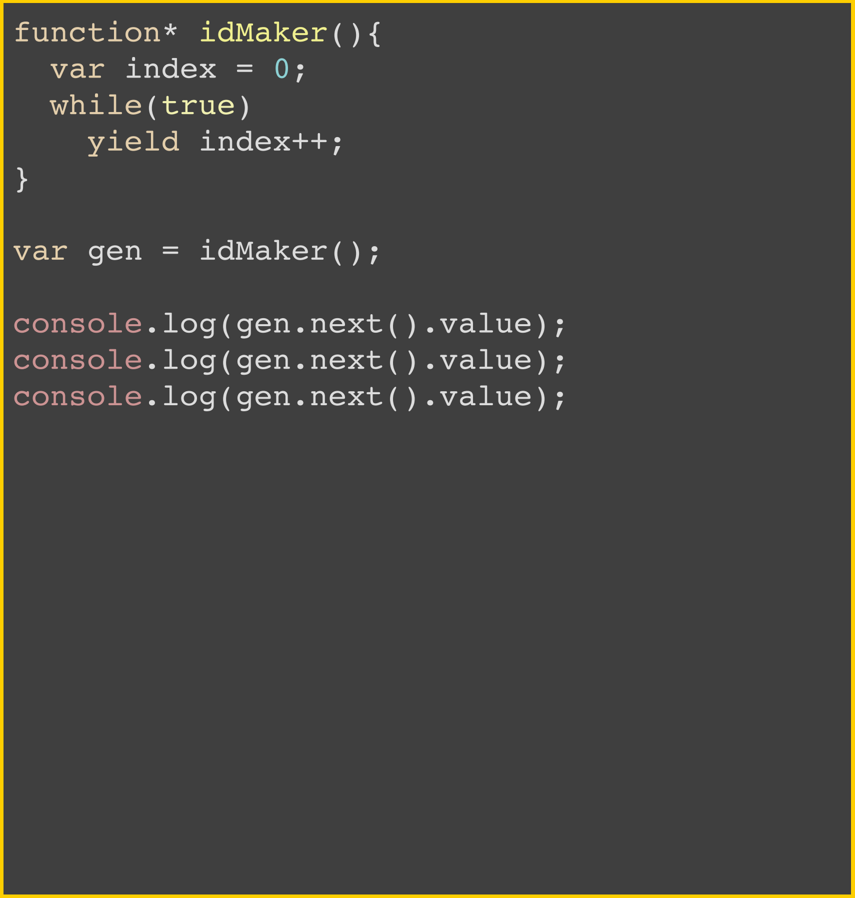


```
class Vehicle {
  constructor(make, year) {
    this._make = make;
    this._year = year;
  }

  get make() {
    return this._make;
  }

  get year() {
    return this._year;
  }

  toString() {
    return `${this.make} ${this.year}`;
  }
}

const fiesta = new Vehicle("Ford Fiesta", 199
console.log(fiesta.toString());
```

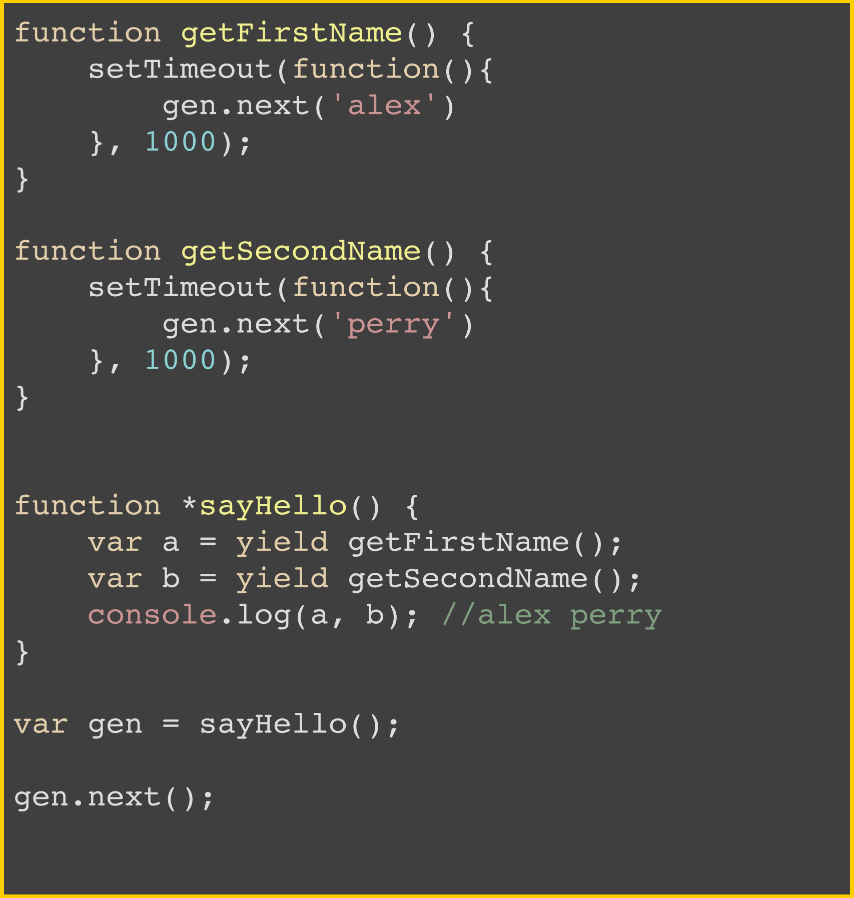


```
class Motorcycle extends Vehicle {
  constructor(make, year) {
    super(make, year);
  }

  toString() {
    return `Motorcycle ${this.make} ${this.ye
  }
}
const suzuki = new Motorcycle("Suzuki", 1999)
console.log(suzuki.toString());
```

9. Generators.



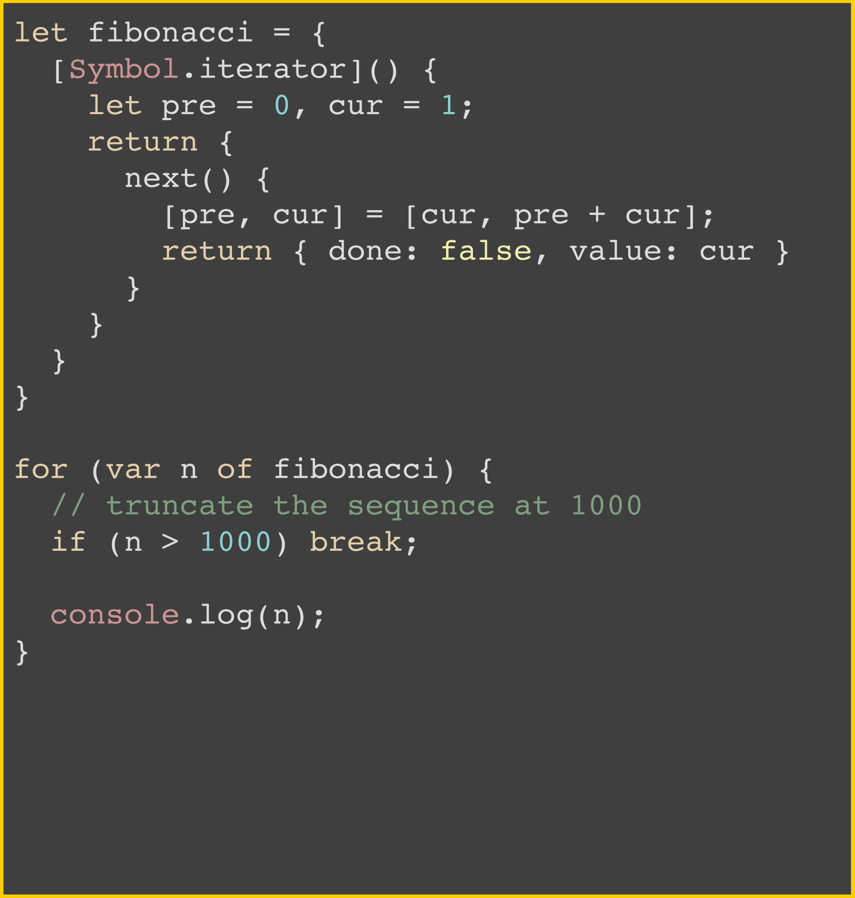
```
function* idMaker(){
  var index = 0;
  while(true)
    yield index++;
}

var gen = idMaker();

console.log(gen.next().value);
console.log(gen.next().value);
console.log(gen.next().value);
```



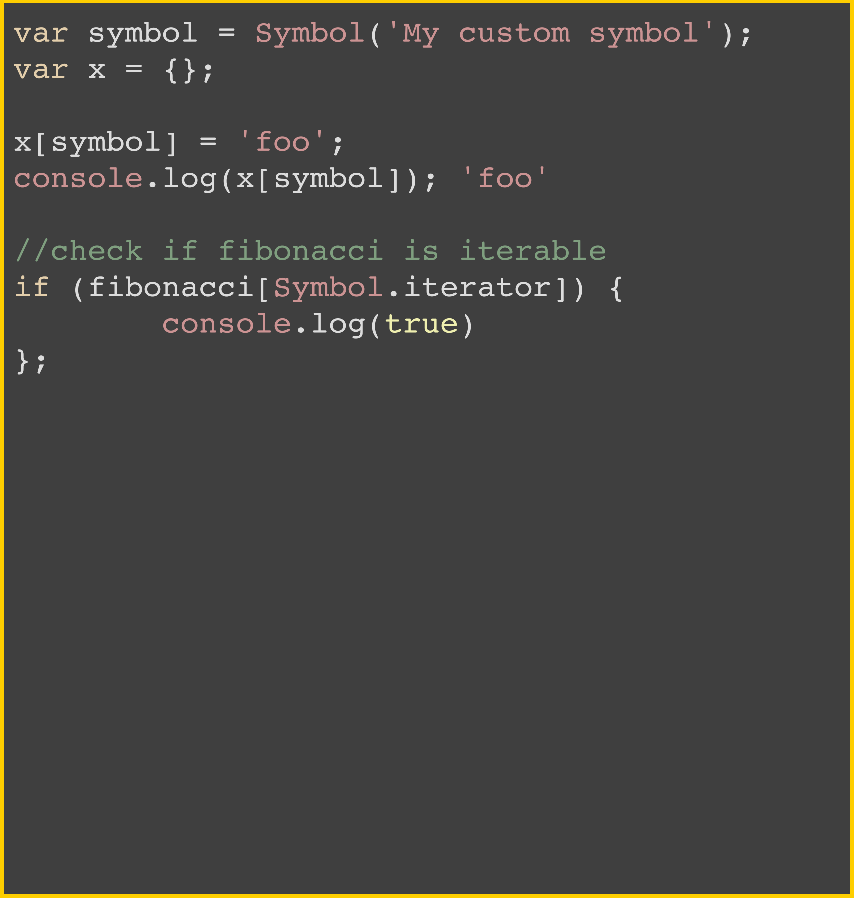
```
function getFirstName() {
    setTimeout(function(){
        gen.next('alex')
    }, 1000);
}

function getSecondName() {
    setTimeout(function(){
        gen.next('perry')
    }, 1000);
}


function *sayHello() {
    var a = yield getFirstName();
    var b = yield getSecondName();
    console.log(a, b); //alex perry
}

var gen = sayHello();

gen.next();
```

[Generators] http://alexperry.io/javascript/2015/09/17/es6-generators-and-asynchronous-javascript.html

10. Iterators and Symbols.

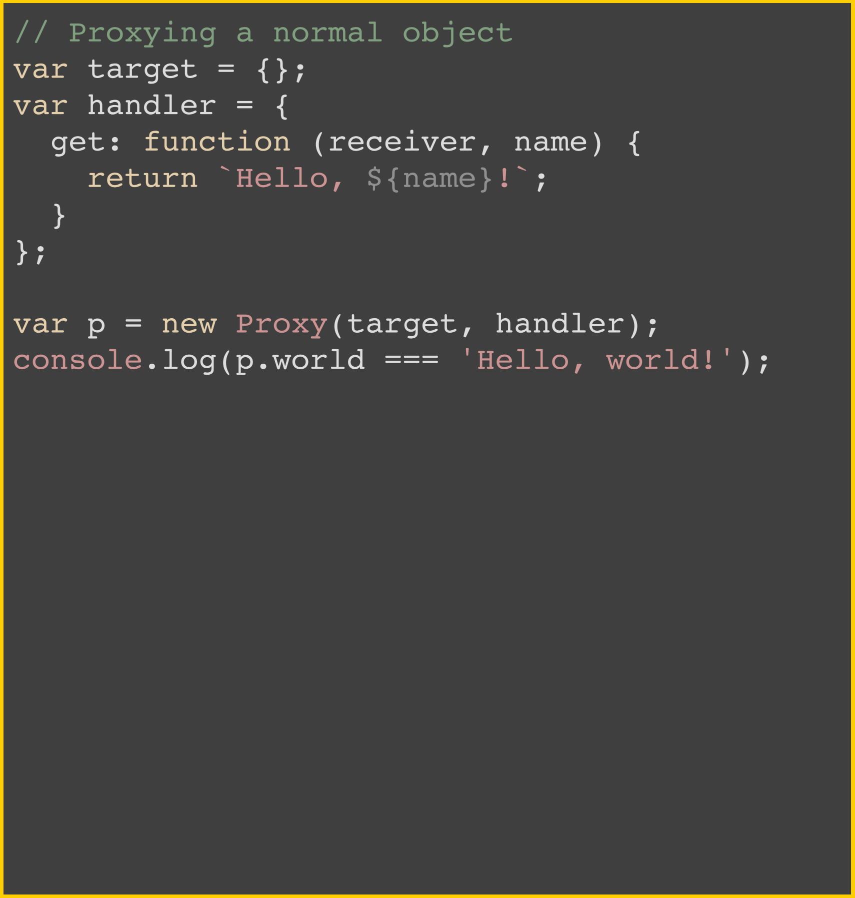


```
let fibonacci = {
  [Symbol.iterator]() {
    let pre = 0, cur = 1;
    return {
      next() {
        [pre, cur] = [cur, pre + cur];
        return { done: false, value: cur }
      }
    }
  }
}

for (var n of fibonacci) {
  // truncate the sequence at 1000
  if (n > 1000) break;

  console.log(n);
}
```

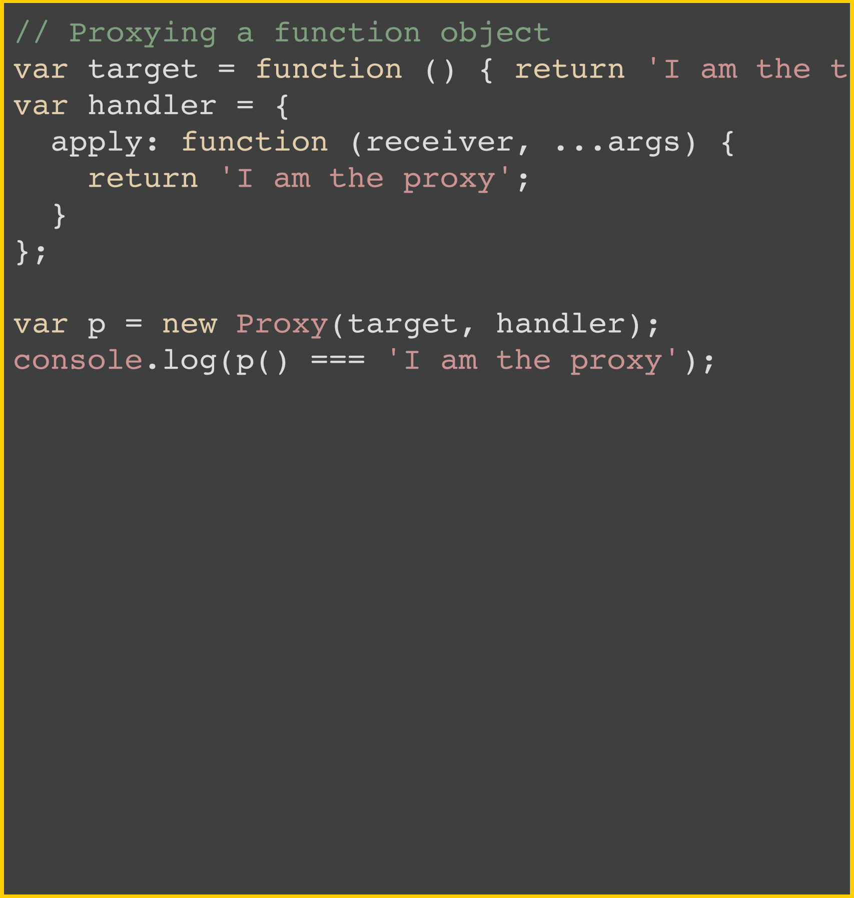


```
var symbol = Symbol('My custom symbol');
var x = {};

x[symbol] = 'foo';
console.log(x[symbol]); 'foo'

//check if fibonacci is iterable
if (fibonacci[Symbol.iterator]) {
        console.log(true)
};
```

11. Proxy.

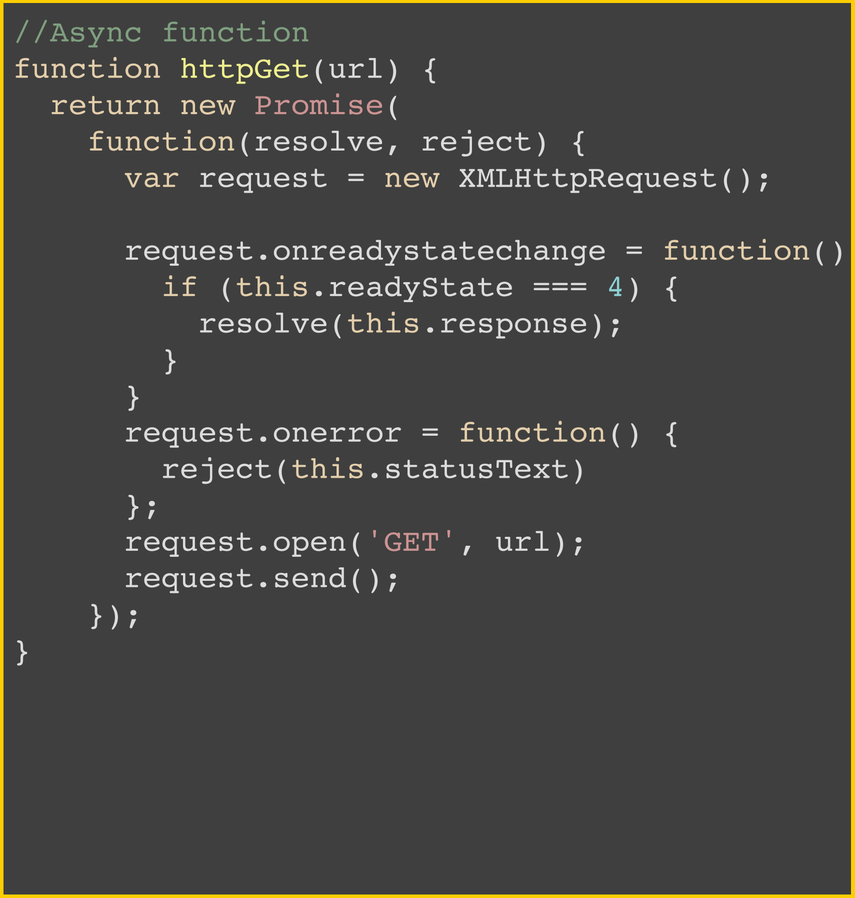


```
// Proxying a normal object
var target = {};
var handler = {
  get: function (receiver, name) {
    return `Hello, ${name}!`;
  }
};

var p = new Proxy(target, handler);
console.log(p.world === 'Hello, world!');
```

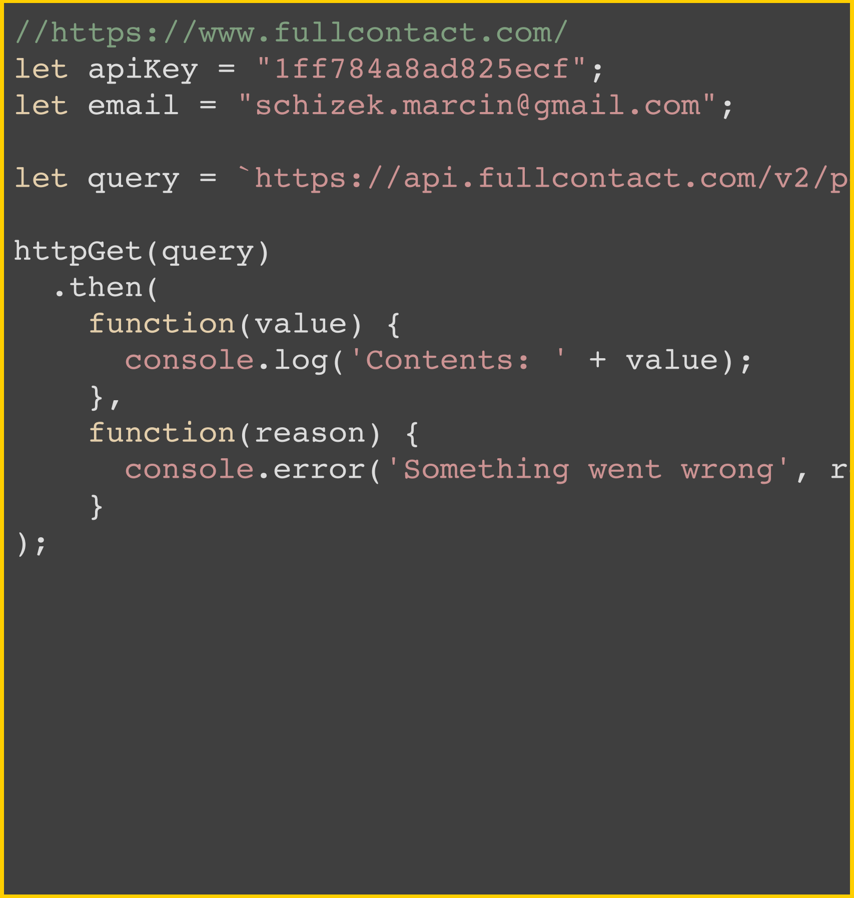


```
// Proxying a function object
var target = function () { return 'I am the t
var handler = {
  apply: function (receiver, ...args) {
    return 'I am the proxy';
  }
};

var p = new Proxy(target, handler);
console.log(p() === 'I am the proxy');
```

12. Promises.



```
//Async function
function httpGet(url) {
  return new Promise(
    function(resolve, reject) {
      var request = new XMLHttpRequest();

      request.onreadystatechange = function()
        if (this.readyState === 4) {
          resolve(this.response);
        }
      }
      request.onerror = function() {
        reject(this.statusText)
      };
      request.open('GET', url);
      request.send();
    });
}
```



```
//https://www.fullcontact.com/
let apiKey = "1ff784a8ad825ecf";
let email = "schizek.marcin@gmail.com";

let query = `https://api.fullcontact.com/v2/p

httpGet(query)
  .then(
    function(value) {
      console.log('Contents: ' + value);
    },
    function(reason) {
      console.error('Something went wrong', r
    }
);
```

13. Is that all??

14. and much more....

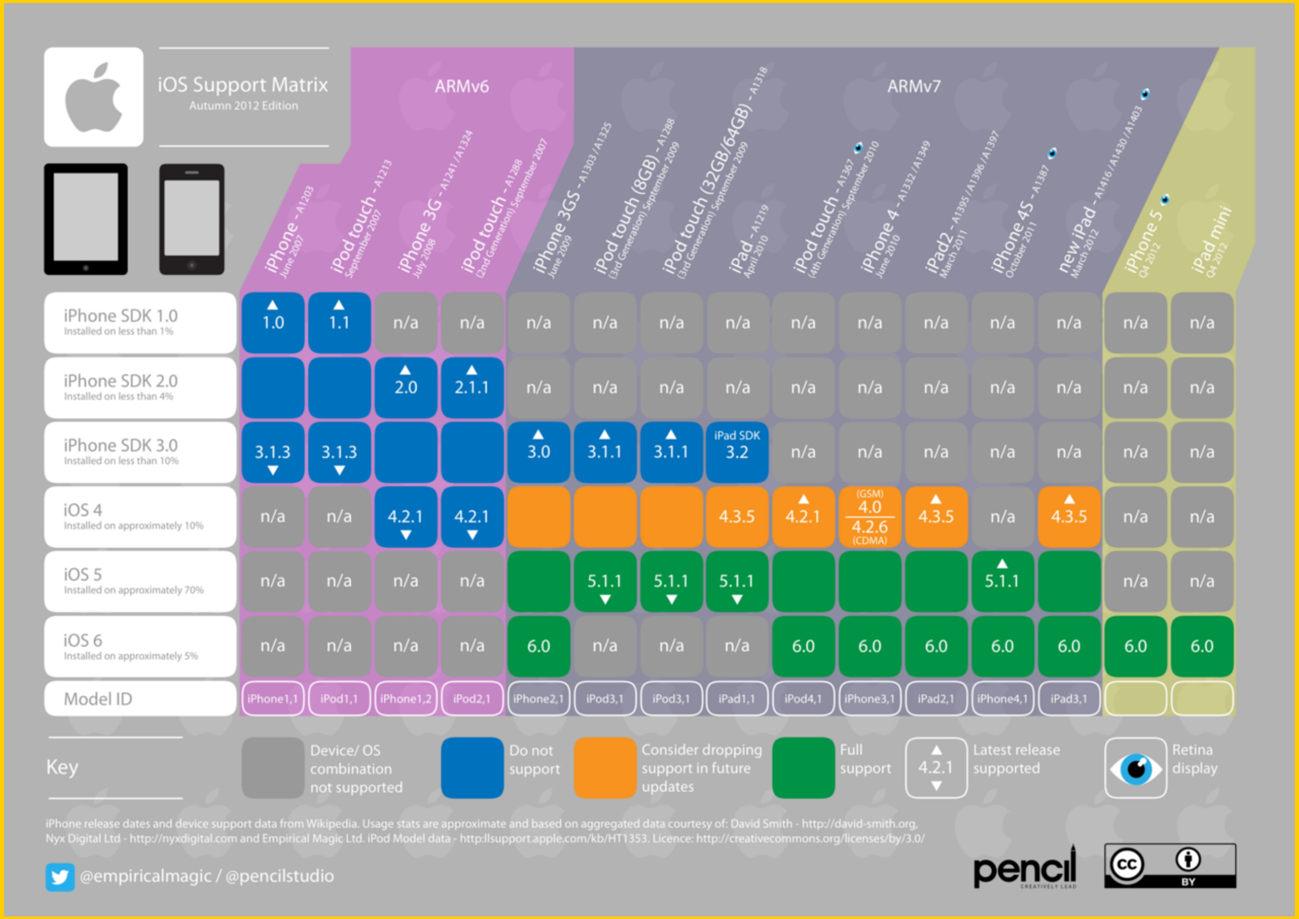
- enhanced object literals
- unicode
- module loaders
- map + set + weakmap + weakset
- subclassable built-ins
- math + number + array + object
- binary and octal literals
- reflect api
- tail calls

Can I use it?

What now??

Why You should care?

1. No Platform fragmentation. (matrix from hell)



iOS Support Matrix
Autumn 2012 Edition

	ARMv6				ARMv7										
	iPhone - A1203 June 2007	iPod touch - A1213 September 2007	iPhone 3G - A1241 / A1324 July 2008	iPod touch - A1288 (2nd Generation) September 2007	iPhone 3GS - A1303 / A1325 June 2009	iPod touch (8GB) - A1288 (3rd Generation) September 2009	iPod touch (32GB/64GB) - A1318 (3rd Generation) September 2009	iPad - A1219 April 2010	iPod touch - A1367 👁 (4th Generation) September 2010	iPhone 4 - A1332 / A1349 June 2010	iPad2 - A1395 / A1396 / A1397 March 2011	iPhone 4S - A1387 👁 October 2011	new iPad - A1416 / A1430 / A1403 👁 March 2012	iPhone 5 👁 Q4 2012	iPad mini Q4 2012
iPhone SDK 1.0 Installed on less than 1%	▲ 1.0	▲ 1.1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
iPhone SDK 2.0 Installed on less than 4%	🟦	🟦	▲ 2.0	▲ 2.1.1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
iPhone SDK 3.0 Installed on less than 10%	3.1.3 ▼	3.1.3 ▼	🟦	🟦	▲ 3.0	▲ 3.1.1	▲ 3.1.1	iPad SDK 3.2	n/a	n/a	n/a	n/a	n/a	n/a	n/a
iOS 4 Installed on approximately 10%	n/a	n/a	4.2.1 ▼	4.2.1 ▼	🟧	🟧	🟧	4.3.5	▲ 4.2.1	(GSM) 4.0 4.2.6 (CDMA)	▲ 4.3.5	n/a	▲ 4.3.5	n/a	n/a
iOS 5 Installed on approximately 70%	n/a	n/a	n/a	n/a	🟩	5.1.1 ▼	5.1.1 ▼	5.1.1 ▼	🟩	🟩	🟩	▲ 5.1.1	🟩	n/a	n/a
iOS 6 Installed on approximately 5%	n/a	n/a	n/a	n/a	6.0	n/a	n/a	n/a	6.0	6.0	6.0	6.0	6.0	6.0	6.0
Model ID	iPhone1,1	iPod1,1	iPhone1,2	iPod2,1	iPhone2,1	iPod3,1	iPod3,1	iPad1,1	iPod4,1	iPhone3,1	iPad2,1	iPhone4,1	iPad3,1		

Key

- ▨ Device/ OS combination not supported
- 🟦 Do not support
- 🟧 Consider dropping support in future updates
- 🟩 Full support
- ▲ 4.2.1 ▼ Latest release supported
- 👁 Retina display

iPhone release dates and device support data from Wikipedia. Usage stats are approximate and based on aggregated data courtesy of: David Smith - http://david-smith.org, Nyx Digital Ltd - http://nyxdigital.com and Empirical Magic Ltd. iPod Model data - http:llsupport.apple.com/kb/HT1353. Licence: http://creativecommons.org/licenses/by/3.0/

@empiricalmagic / @pencilstudio

pencil
CREATIVELY LEAD

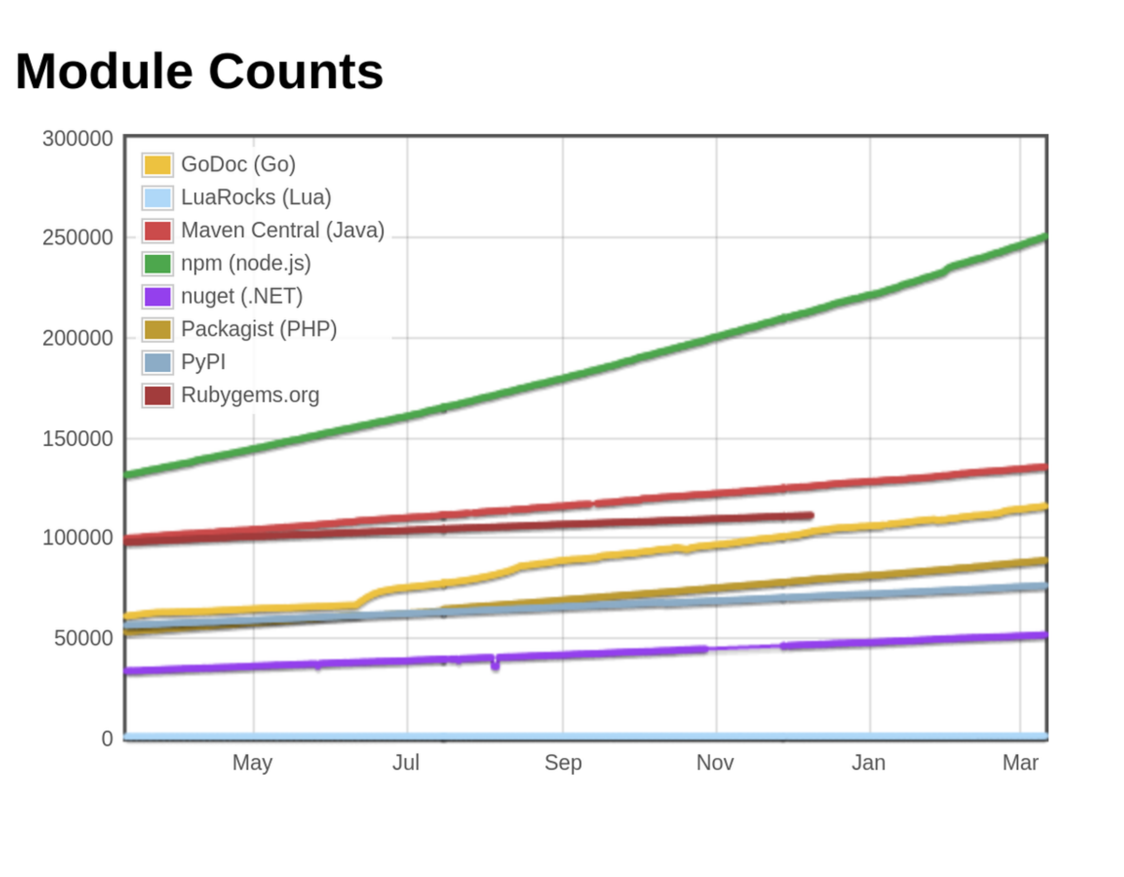
2. JavaScript is ambiguous.

Electron VS. NW.js
React
Vue.js
ANGULARJS by Google
React Native
ionic
APACHE CORDOVA™
node JS™
express
CYLON.JS

3. Better development process.

1. No compilation & fast feedback.
2. Full-stack development.
3. Less opinionated, good selection.
4. Asyc is good enough concurrency
5. Open-sourced, well supported

4. Community and modules



5. Perfomance

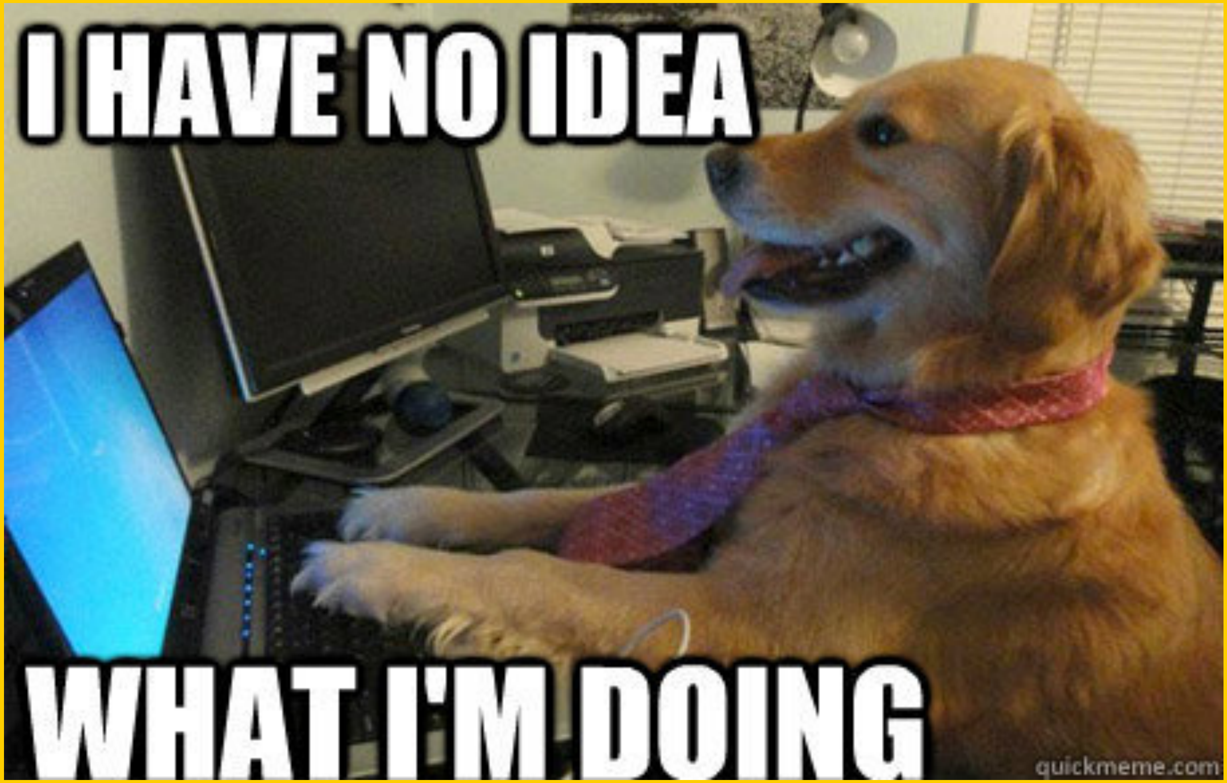
V8 vs Python 3 http://goo.gl/hKX9rV

V8 vs Java http://goo.gl/H6ywV8

Challenges & issues

- Transpilation is not free.
- Most modules are still ES5.
- No idiomatic way of writing.
- CSS and HTML are difficult.
- High velocity of modules/libraries

Q&A Session



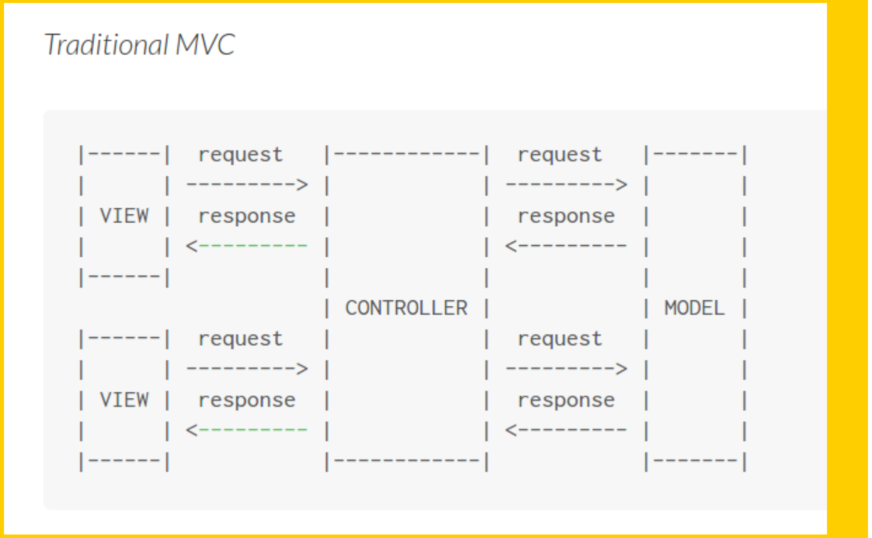
Lets Build something.

1. Setup envirotment

Git Repo: https://github.com/Chyzwar/TechSymposium

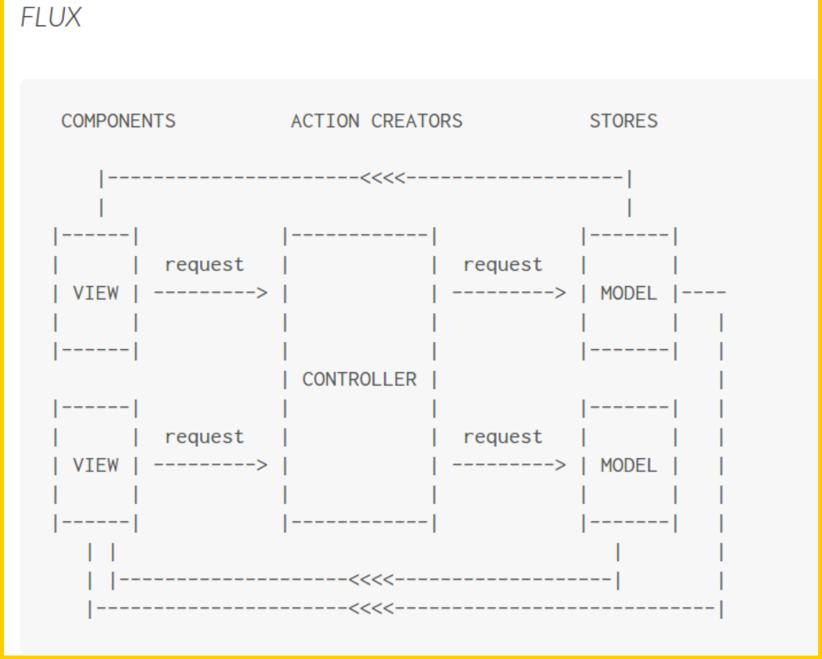
Cloud IDE:

2. Flux Architecture



Traditional MVC

```
|------|  request   |------------|  request   |-------|
|      | ---------> |            | ---------> |       |
| VIEW |  response  |            |  response  |       |
|      | <--------- |            | <--------- |       |
|------|            |            |            |       |
                    | CONTROLLER |            | MODEL |
|------|  request   |            |  request   |       |
|      | ---------> |            | ---------> |       |
| VIEW |  response  |            |  response  |       |
|      | <--------- |            | <--------- |       |
|------|            |------------|            |-------|
```

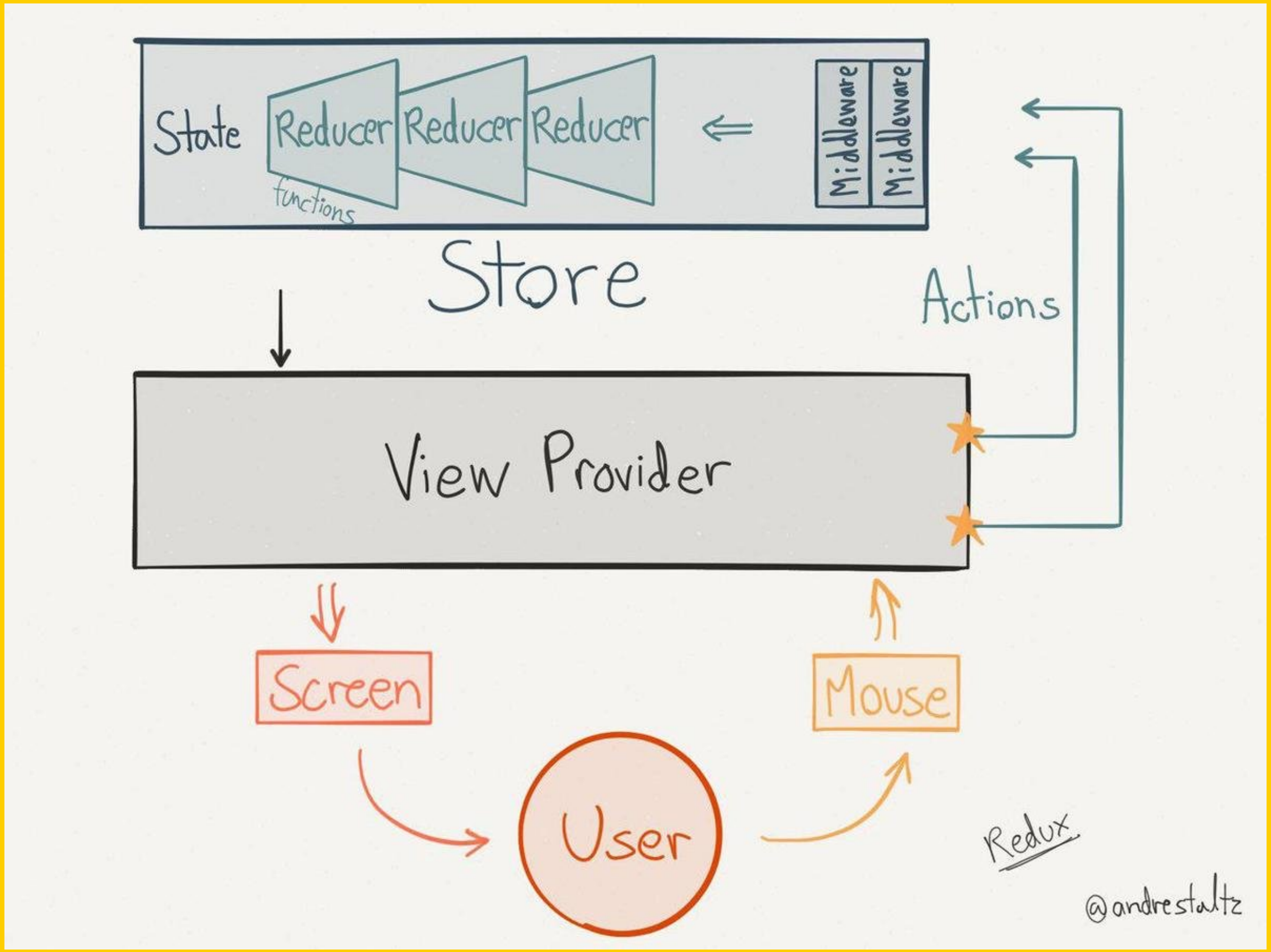


FLUX

```
 COMPONENTS           ACTION CREATORS           STORES

     |----------------------<<<<--------------------|
     |                                              |
|------|            |------------|            |-------|
|      |  request   |            |  request   |       |
| VIEW | ---------> |            | ---------> | MODEL |----
|      |            |            |            |       |   |
|------|            |            |            |-------|   |
                    | CONTROLLER |                        |
|------|            |            |            |-------|   |
|      |  request   |            |  request   |       |   |
| VIEW | ---------> |            | ---------> | MODEL |   |
|      |            |            |            |       |   |
|------|            |------------|            |-------|   |
   | |                                          |         |
   | |--------------------<<<<------------------|         |
   |--------------------<<<<--------------------------------|
```

[MVC and FLux] http://www.christianalfoni.com/articles/2015_08_02_Why-we-are-doing-MVC-and-FLUX-wrong

3. Redux


State
Reducer
Reducer
Reducer
functions
Middleware
Middleware
Store
Actions
View Provider
Screen
Mouse
User
Redux
@andrestaltz