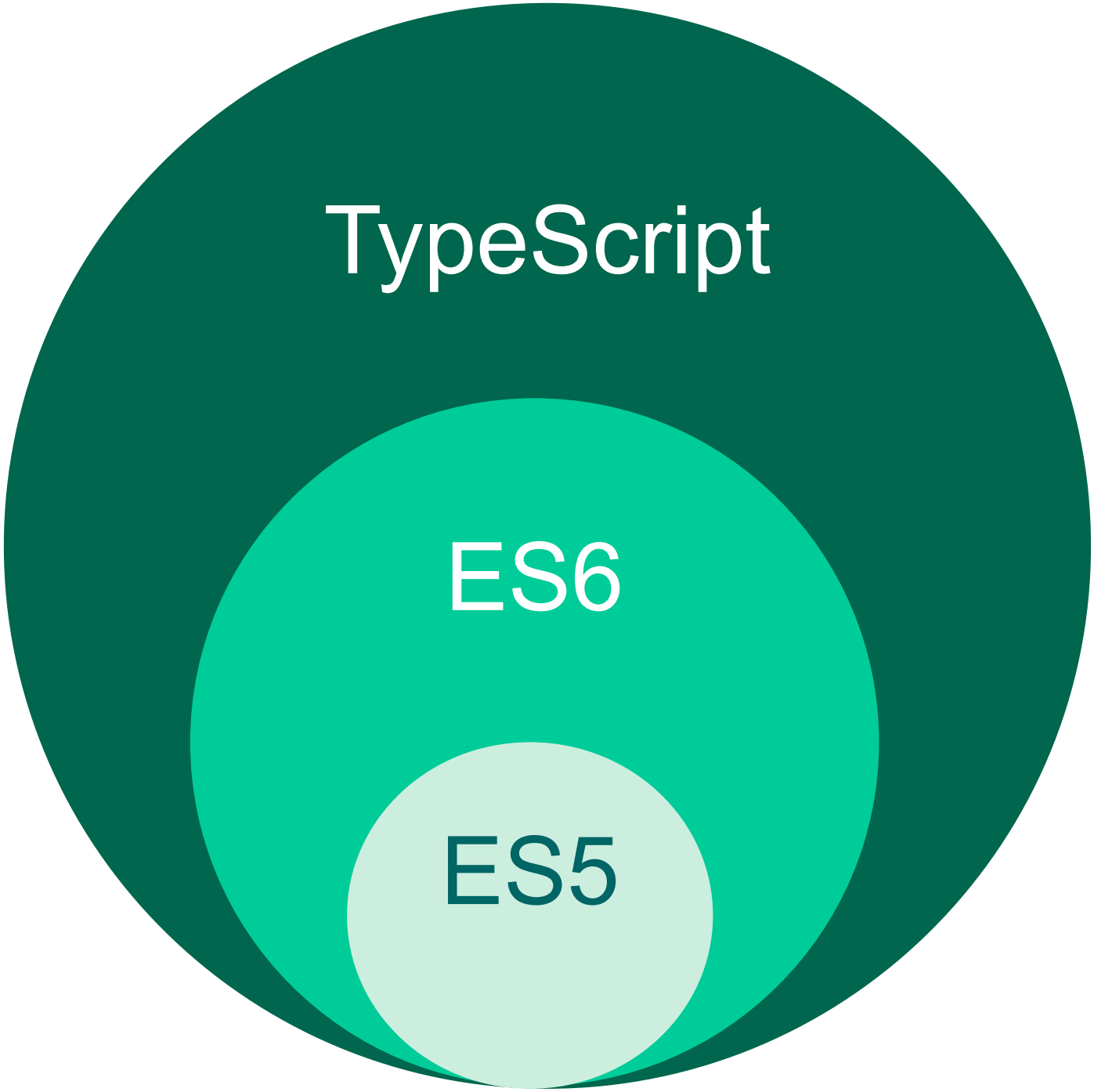
ES6/ECMAScript 2015

Peter Kassenaar – info@kassenaar.com

What wrong with JavaScript?

Maar we willen…

Programmeertalen


TypeScript
ES6
ES5

What is ECMAScript 2015?

Major update from JavaScript programming language

Current version: ES5 (2009)

Class-based object-orientation

New features and keywords

ES6: Superset of JavaScript

ECMAScript 5 6 7 intl 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 platforms? Show unstable platforms?

V8 SpiderMonkey JavaScriptCore Chakra Carakan KJS Other

Minor difference (1 point) Useful feature (2 points) Significant feature (4 points) Landmark feature (8 points)

		Compilers/polyfills						Desktop browsers											
	60%	61%	73%	32%	21%	53%	16%	6%	15%	59%	79%	67%	72%	73%	75%	60%	61%	11%	23%
Feature name	Current browser	Traceur	Babel + core-js[1]	Closure	JSX[2]	Type-Script + core-js	es6-shim	IE 10	IE 11	Edge 12[3]	Edge 13[3]	FF 38 ESR	FF 42	FF 43	FF 44	CH 47, OP 34[0]	CH 48, OP 35[0]	SF 6.1, SF 7	SF 7.1, SF 8
Optimisation																			
proper tail calls (tail call optimisation)	0/2	0/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
Syntax																			
default function parameters	0/7	4/7	6/7	4/7	0/7	4/7	0/7	0/7	0/7	0/7	0/7	3/7	3/7	4/7	4/7	0/7	0/7	0/7	0/7
rest parameters	5/5	4/5	4/5	2/5	3/5	3/5	0/5	0/5	0/5	5/5	5/5	4/5	4/5	5/5	5/5	5/5	5/5	0/5	0/5
spread (...) operator	15/15	15/15	13/15	3/15	2/15	4/15	0/15	0/15	0/15	12/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	0/15	5/15
object literal extensions	6/6	6/6	6/6	4/6	5/6	6/6	0/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	0/6	1/6
for..of loops	7/9	9/9	9/9	6/9	2/9	3/9	0/9	0/9	0/9	6/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	0/9	2/9
octal and binary literals	4/4	2/4	4/4	2/4	0/4	4/4	2/4	0/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	0/4	0/4
template strings	5/5	4/5	4/5	3/5	4/5	3/5	0/5	0/5	0/5	4/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	0/5
RegExp "y" and "u" flags	0/4	2/4	2/4	0/4	0/4	0/4	0/4	0/4	0/4	2/4	4/4	2/4	2/4	2/4	2/4	0/4	0/4	0/4	0/4
destructuring, declarations	0/22	19/22	21/22	14/22	12/22	14/22	0/22	0/22	0/22	0/22	0/22	19/22	19/22	19/22	19/22	0/22	0/22	0/22	9/22
destructuring, assignment	0/24	22/24	24/24	11/24	11/24	18/24	0/24	0/24	0/24	0/24	0/24	20/24	21/24	21/24	21/24	0/24	0/24	0/24	12/24
destructuring, parameters	0/23	18/23	21/23	14/23	12/23	14/23	0/23	0/23	0/23	0/23	0/23	18/23	18/23	18/23	18/23	0/23	0/23	0/23	10/23
Unicode code point escapes	2/2	1/2	1/2	1/2	0/2	1/2	0/2	0/2	0/2	2/2	2/2	0/2	1/2	1/2	1/2	2/2	2/2	0/2	0/2
new.target	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	1/2	0/2	2/2	2/2	2/2	2/2	2/2	0/2	0/2
Bindings																			
const	5/8	6/8	6/8	6/8	0/8	6/8	0/8	0/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	5/8	5/8	1/8	1/8
let	5/10	8/10	8/10	8/10	0/10	7/10	0/10	0/10	8/10	8/10	8/10	0/10	0/10	0/10	8/10	5/10	5/10	0/10	0/10
block-level function declaration[12]	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes	Yes	No	No	No	No	Yes	Yes	No	No
Functions																			
arrow functions	11/13	11/13	10/13	10/13	8/13	9/13	0/13	0/13	0/13	9/13	13/13	8/13	10/13	11/13	11/13	11/13	11/13	0/13	0/13
class	0/23	16/23	19/23	9/23	15/23	16/23	0/23	0/23	0/23	0/23	23/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23
super	0/8	7/8	6/8	4/8	7/8	7/8	0/8	0/8	0/8	0/8	8/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8

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

Exploring ES6
Homepage
Please support this book: buy it (PDF, EPUB, MOBI) or donate

Table of contents

http://exploringjs.com/es6/

“Superset van JavaScript”

ECMAScript 2015

JavaScript

```
var x;

function foo(){
  …
}
```

```
const x;

class Foo {
  …
}
```

Superset? In code…

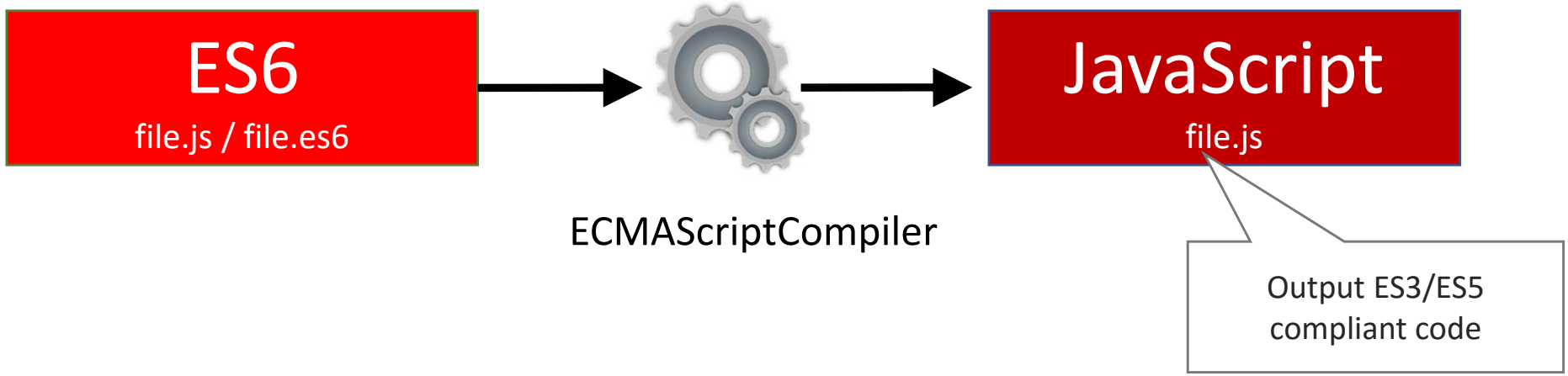
```
// Classic JavaScript - also valid ES6
function foo(){
   "use strict";
   var msg = 'Hello World from function';
   return msg;
}
console.log(foo());
```

```
// New in ES6 :classes w/ functions
class Bar{
   hello(){
      var msg = 'Hello World from class';
      return msg;
   }
}
var bar = new Bar();
console.log(bar.hello());
```

ES6/ECMAScript Compilation

ES6: compilatie nodig (*transpiling*).

Huidige generatie browsers begrijpt ES6 en TypeScript niet.



Transpiling options : Babel & Traceur

- **Liefst: server-sided compileren**

  Via npm, grunt of gulp

- **Voorkeur: Babel (clean code)**

  http://babeljs.io/

- **Ook mogelijk: Traceur (by Google)**

  https://github.com/google/traceur-compiler

Babel · The compiler

babeljs.io

BABEL Learn ES2015 Setup Plugins Usage Try it out Discuss Chat Issues Blog Twitter GitHub

Babel is a JavaScript compiler.

Use next generation JavaScript, today.

Babel Plugin Handbook Published!

Star 12,269

Babel transforms your JavaScript

You put JavaScript in	And get JavaScript out
`myJavaScript("foobar");`	`myNewTransformedJavaScript("yay!");`

✓ Arrow functions ✓ Generators

http://babeljs.io/

google/traceur-comp

GitHub, Inc. [US] https://github.com/google/traceur-compiler

This repository Search Pull requests Issues Gist

google / traceur-compiler

Watch 267 Star 6,050 Fork 382

Code Issues 276 Pull requests 7 Wiki Pulse Graphs

Traceur is a JavaScript.next-to-JavaScript-of-today compiler

1,651 commits 4 branches 58 releases 54 contributors

Branch: master New pull request New file Find file HTTPS https://github.com/google/t Download ZIP

arv Merge pull request #2040 from arv/fix-repl Latest commit 092c563 3 days ago

build	Update feature tests to es6.	6 months ago
demo	Fix repl.html	4 days ago
example	Remove unnecessary options from example	11 months ago
logo	Add a logo	2 years ago
src	Split src/traceur.js into compiler, loader, and util imports.	4 days ago
test	Merge pull request #2032 from arv/move-symbols-out-of-runtime-js	8 days ago
third_party	implement bulk re-exports for instantiate setter perf	6 months ago
tools	Add a simple tool for printing the dependencies	6 months ago
.editorconfig	Add EditorConfig	a year ago
.gitattributes	Add text=auto to .gitattributes.	a year ago
.gitignore	Add Make target bin/traceur-no-modules.js;	8 days ago
.travis.yml	Add node 0.10 to TravisCI and fix test	20 days ago
AUTHORS	Fix template literals precedence for NewExpression	9 months ago
CONTRIBUTING.md	add http://editorconfig.org	a year ago
LICENSE	Rename COPYING -> LICENSE	3 years ago
Makefile	Add Make target bin/traceur-no-modules.js;	8 days ago
README.md	Update README.md	9 months ago
gh-pages.gitignore	Fixup runner.html for the FreeVariableChecker.generated.js change	a year ago
index.html	Change index.html redirect URL to add dash	11 months ago
package.json	0.0.93	15 days ago
testRunner.html	Update feature tests to es6.	6 months ago
traceur	Command line traceur will now invoke the interpreter if no out param ...	3 years ago

https://github.com/google/traceur-compiler

Output

```
// Babel
'use strict';

//… internal definition omitted…

function foo() {
   "use strict";
   var msg = 'Hello World from function';
   return msg;
}
console.log(foo());

// New in ES6 :classes w/ function

var Bar = (function () {
   function Bar() {
      _classCallCheck(this, Bar);
   }

   // Don't forget: instantiate new Bar() (!)

   _createClass(Bar, [{
      key: 'hello',
      value: function hello() {
         var msg = 'Hello World from class';
         return msg;
      }
   }]);

   return Bar;
})();

var bar = new Bar();
console.log(bar.hello());
```

```
// Traceur
"use strict";
function foo() {
  "use strict";
  var msg = 'Hello World from function';
  return msg;
}
console.log(foo());
var Bar = function() {
  function Bar() {}
  return ($traceurRuntime.createClass)
           (Bar, {hello: function() {
      var msg = 'Hello World from class';
      return msg;
    }}, {});
}();
var bar = new Bar();
console.log(bar.hello());
```

Let op: Babel 5.x

- In deze demo's gebruiken we babel 5.8.
- Nieuwer: babel 6.
  - Maar: *complete overhaul.*
  - Meer opties, veel (!) complexer.

The Six Things You Need To Know About Babel 6

posted in Build Systems, ES6, Javascript on November 9, 2015 by James K Nelson

Over the last year, Babel has become the go-to tool for transforming ES2015 and JSX into boring old JavaScript. But seemingly overnight, *Babel 6 changed everything*. The `babel` package was deprecated, running `babel` doesn't actually transform ES2015 to ES5, and the old docs have basically *disappeared*.

But Don't Panic! To get you up to speed, I've put together a brief list of the *six most important* changes. And if you need a little more help, my Complete Guide to ES6 with Babel 6 will walk you through the practical details; including the CLI, Webpack, Mocha and Gulp.

1. **The `babel` npm package no longer exists.** Instead, Babel has been split into

James K Nelson

- **Wants to** help people create ama
- **Has used** JavaScript for more tha
- **Built** Memamug, an open-source
- **Currently** working on Unicorn St

Follow @james_k_nelson

Follow @jamesknelson

Guides with Cheatshee

- ES6 - The Bits You'll Actually Use
- Learn Raw React – no JSX, Flux, E
- Introduction to ES6 Promises

http://jamesknelson.com/the-six-things-you-need-to-know-about-babel-6/

Boek & blogs over Babel 6

②ality – JavaScript and more

About | Donate | Subscribe | ES5 book | ES6 book | ES2016 | Newsletter

Most popular (last 30 days)

ECMAScript 6 modules: the final syntax

The future of bundling JavaScript modules

Classes in ECMAScript 6 (final semantics)

Looking back on 2015: six exciting web technologies

Enumify: better enums for JavaScript

Managing the private data of ES6 classes

ES6 classes have inner names

Most popular (all time)

ECMAScript 6 modules: the final syntax

Classes in ECMAScript 6 (final semantics)

2015-11-29

Configuring Babel 6

Labels: babel, dev, esnext, javascript, jstools

Babel 6 is much more configurable than Babel 5, but also more difficult to configure. This blog post gives tips.

Follow-up blog posts:

- [2015-12-11] Babel 6: configuring ES6 standard library and helpers
- [2015-12-12] Babel 6: loose mode
- [2015-12-13] Babel and CommonJS modules

1. Installing Babel 6

The following are a few important npm packages. All Babel packages reside in a single repository on GitHub. Browsing their source code and their `package.json` files is instructive.

- `babel-core`: the core compilation machinery and plugin infrastructure for Babel. You will rarely need to install this package, because other packages such as `babel-cli` have it as a dependency, meaning that it will be automatically installed when they are installed.
- `babel-cli`: a command line interface to `Babel`. It includes the following commands:
  - `babel-doctor` detects common problems with your Babel installation.

(Ad, please don't block.)

Dr. Axel Rauschmayer

Free online books by Axel

Speaking JavaScript [up to ES5]

Exploring ES6

JavaScript training:

Ecmanauten

Tweets Follow

Scott Kelly @StationCDRKelly 21 Jan

Try this, Mary Poppins! Super-hydrophobic polycarbonate ping pong paddles and a water

http://www.2ality.com/2015/11/configuring-babel6.html

Transpiling Automation with Gulp

```
// gulpfile.js
var gulp = require('gulp'),
   babel = require('gulp-babel'),
   es6Path = 'es6/*.es6',
   compilePath = 'es6/compiled';

// 1. Define task for Babel
gulp.task('babel', function () {
   return gulp.src([es6Path])
      .pipe(babel())
      .pipe(gulp.dest(compilePath + '/babel'));
});

// 2. When file changes, update compiled version
gulp.task('watch', function () {
   gulp.watch([es6Path], ['babel']);
});

// 3. Bootstrap: start default tasks
gulp.task('default', ['babel', 'watch']);
```

Belangrijke ES6 Features

Maps/ Sets	Classes	Block Scope	Destructuring
Arrow Functions	Modules	Default/Rest Parameters	More...

Belangrijke ES6 Features

Maps/ Sets	Classes	Block Scope	Destructuring
Arrow Functions	Modules	Default/Rest Parameters	More...

Maps en Sets

New way of handling collections:

Maps store a collection of key/value pairs, with unique keys

Sets can store a collection of items (items must be unique)

Using Map()

```
//Using Map (*Simple example)
var map = new Map();
map.set('Finance','Process bills');
map.set('HR', 'Human Resources and Healthcare');
map.set('HR', 'Human Resources and Healthcare'); //Duplicate ignored
console.log('Getting HR: ' + map.get('HR'));
console.log(map.size);

if (map.has('Finance')) console.log('Found it!');
map.delete('Finance'); //Delete single item
map.clear(); //Clear all items
```

Map() interface

```
.set(key, val)
.get(key)
.has(key)
.delete(key)
.clear()
.size
```

- `.keys()`
- `.values()`
- `.entries()`

http://exploringjs.com/es6/ch_maps-sets.html#_maps

Using Set()

"A Set is a collection of unique elements"

(remember: a Map has to have unique *keys*)

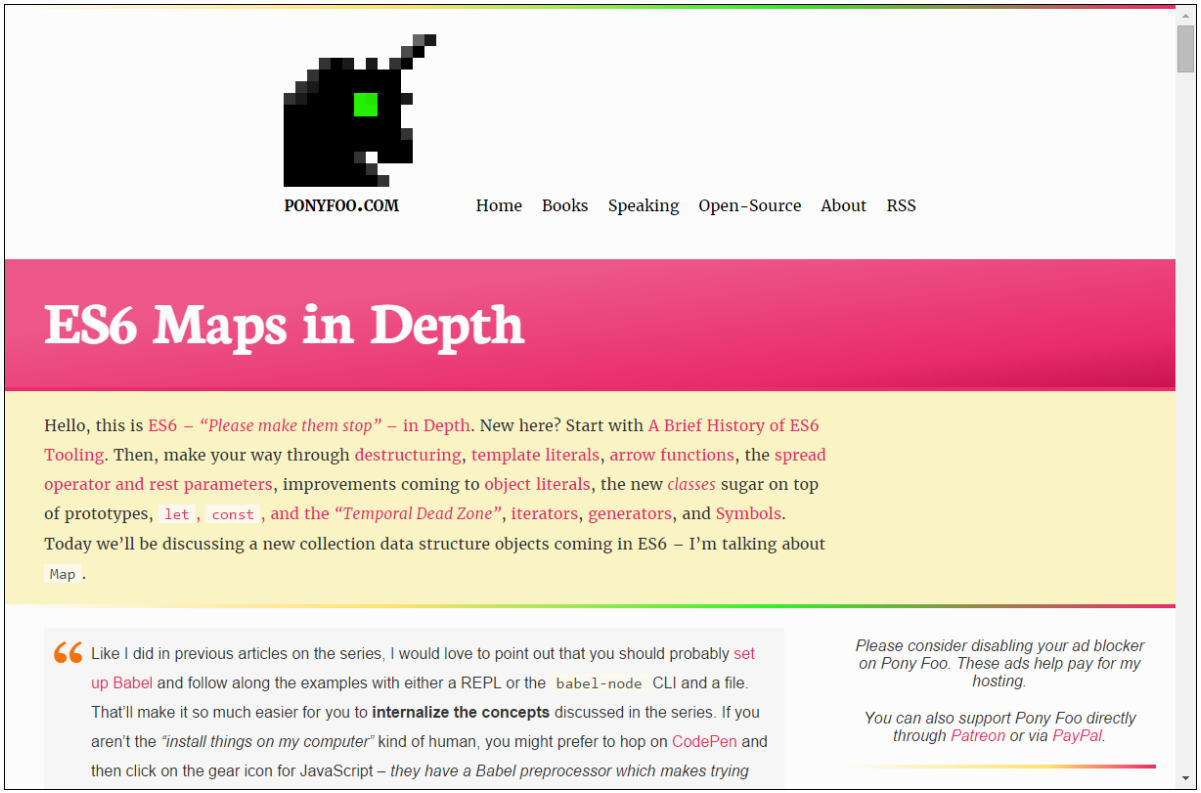
```
var set = new Set();
set.add('Finance');
console.log(set);
console.log('\n');

// num items in set
console.log('num items in set:', set.size);
console.log('\n');

// duplicates are ignored:
var colorSet = new Set(['red', 'green', 'green', 'blue', 'blue', 'yellow', 'red']);
console.log('items in colorSet: ', colorSet);
console.log('\n');
```

Set() interface

```
.add()
.has()
.delete()
.clear()
.size
```

Maps / Sets : wanneer gebruiken?

- Als vervanging/uitbreiding van Array's.
- Als vervanging/uitbreiding van custom Hash-maps



https://ponyfoo.com/articles/es6-maps-in-depth

Belangrijke ES6 Features

Maps/ Sets	Classes	Block Scope	Destructuring
Arrow Functions	Modules	Default/Rest Parameters	More...

Arrow Functions

- Kortere schrijfwijze voor functions, function body en callbacks
- Lexical this

```
// traditional ES5
var oldLogger = function (msg) {
   console.log(msg);
};
oldLogger('Hello oldskool ES5!');
```

```
// ES6, with arrow function
var newLogger = msg => console.log(msg); //Creates an anonymous function. OF
var newLogger2 = (msg) => console.log(msg)
newLogger('Hello ES6 arrow functions!');
```

Lexical this

- `this` verwijst in ECMAScript 2015 steeds naar dezelfde scope/function/object.
- Geen noodzaak meer voor caching (`var self = this`, etc).

```
//Working with "this" the "old" way
function Car() {
   var self = this; // caching 'this' in local variable
   self._seats = 4;

   self.timeout = function () {
      setTimeout(function () {
         console.log('we get more seats in this 2016 model!',
           ++self._seats);
      }, 1000);
   }
}
```

```
//Working with this using arrow functions
class CarWithArrow {

   constructor() {
      this._seats = 6;
   }

   timeout() {
      setTimeout(() => {
         console.log('we get even more seats in with ES6!',
           ++this._seats); // No need to cache 'this' !
      }, 1000);
   }
}
```

Invoking:

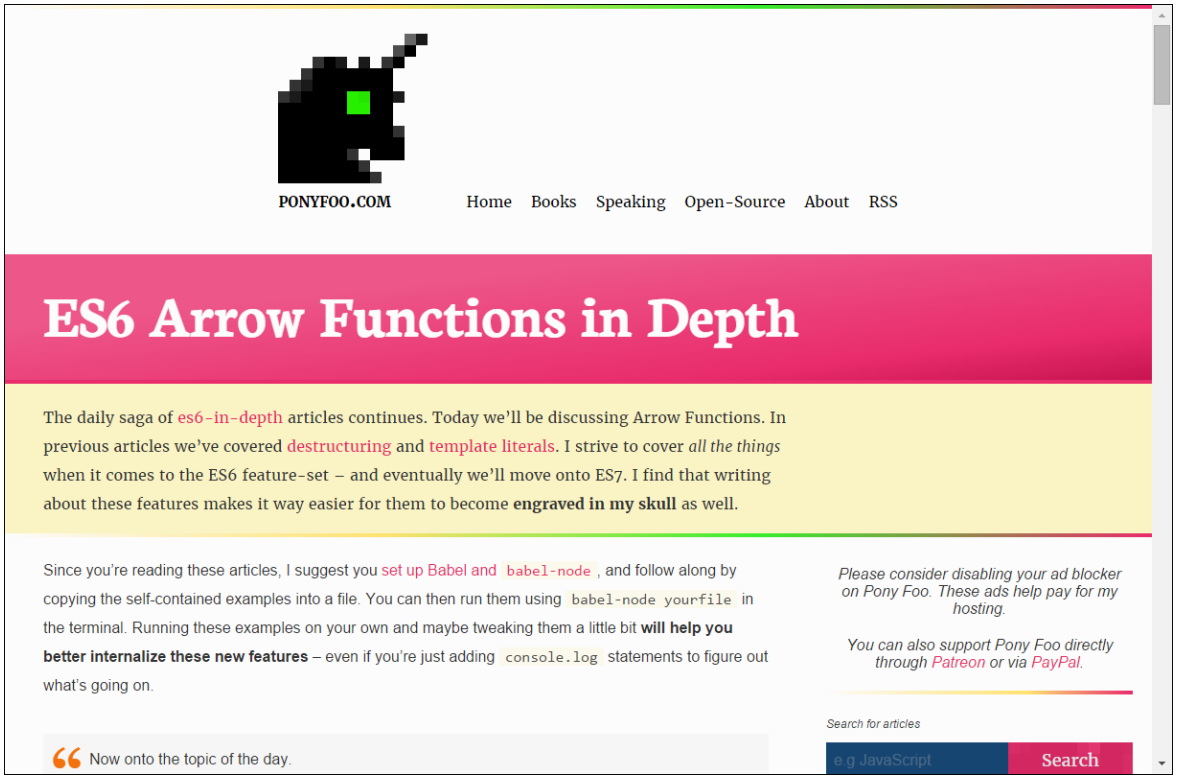
```
var c = new Car();
c.timeout();
var cArrow = new CarWithArrow();
cArrow.timeout();
```

Another arrow function example

```
class UiComponent {
   constructor() {
      let button = document.getElementById('myButton');
      button.addEventListener('click', () => {
         console.log('I\'m clicked!');
         this.handleClick(); // this verwijst naar de class!
      });
   }

   handleClick() {
      ///    ...
      alert('Hello world')
   }
}

var uiC = new UiComponent();
```

Arrow functions : wanneer gebruiken?

- Zoveel mogelijk!
  - Kortere notatie
  - Compatible met TypeScript
  - Lexical `this`



https://ponyfoo.com/articles/es6-arrow-functions-in-depth

Belangrijke ES6 Features

Maps/ Sets	Classes	Block Scope	Destructuring
Arrow Functions	Modules	Default/Rest Parameters	More...

Classes

Classes are mainly ‘syntactic sugar’ over old school functions and constructors

```
// class example - base class
class BaseLog {
   constructor() {
      this.logName = 'Log1';
   }

   log(msg) {
      console.log(this.logName + ': ' + msg);
   }
}
```

Classes – important features

- **No keyword `function` inside class**
- **The `this` refers always to the class instance**
- **The `constructor()` is always automatically called upon instantiation.**
- **Keywords `super`, `get` and `set`**
- **Class declarations are *not* hoisted, like functions.**

- **http://exploringjs.com/es6/ch_classes.html**

ES6 Class Example

```
class Auto {
    constructor(engine) {
        this._engine = engine;
    }

    get engine() {
        return this._engine;
    }

    set engine(val) {
            this._engine = val;
    }

    start() {
        console.log(this.engine);

    }
}
```

constructor

get/set property blocks

function

Subclasses

```
// subclass, inherits from base class
class Logger extends BaseLog {

   constructor(logName) {
      super(logName);
   }

   writeLine(msg) {
      super.log(msg + '\r\n');
   }
}
```

HTML:

```
<script>
    var logger = new Logger('MyLog');
    logger.writeLine('Logging via ES6 classes!');
</script>
```

Template Literal (‘template strings’)

Use backticks ` …` for multiple line-strings

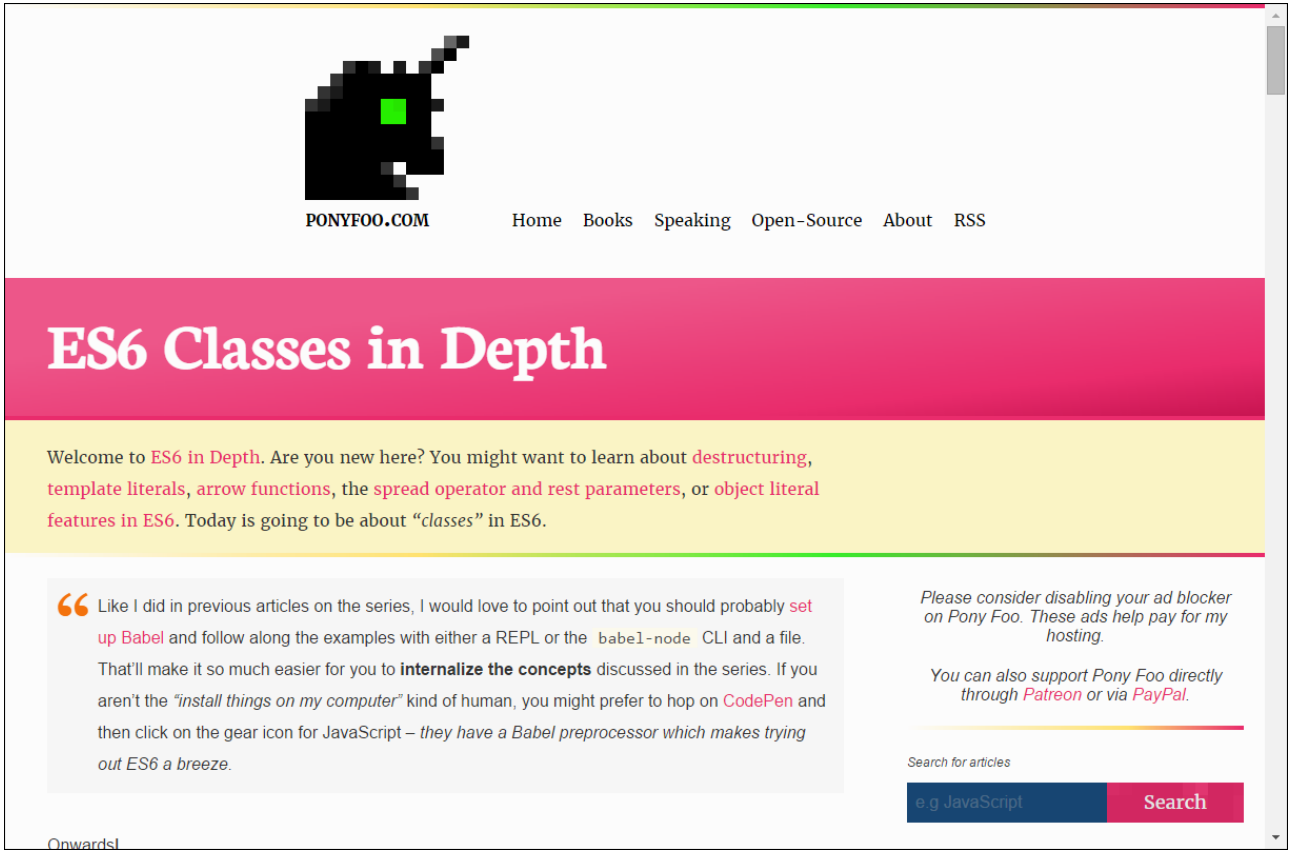
String interpolation: use ${…} for templates inside a string

No more ‘+’ –signs at the end of each line!

```
class Car {
    constructor(make, model, engine, price) {
        this._make = make;
        this._model = model;
        this._engine = engine;
        this._price = price;
        this._taxRate = .08;
    }
    …

    start() {
        //Use a template string
        return `${this.make} ${this.model} with a
                ${this.engine} engine is started!`;
    }

    getTotal() {
        return `Total cost is: ${(this._price * this._taxRate) +
                this._price}`;
    }
}
```

Classes : wanneer gebruiken?

- Hangt beetje af van persoonlijke voorkeur
  - Java/C#-developers: “Altijd!”
  - Functionele (F#, Haskell, JavaScript) – developers: “onzin”.



https://ponyfoo.com/articles/es6-classes-in-depth

Belangrijke ES6 Features

Maps/ Sets	Classes	Block Scope	Destructuring
Arrow Functions	Modules	Default/Rest Parameters	More...

Modules

“ES6 has built-in support for modules. Alas, no JavaScript engine supports them natively, yet.”

- ***Module loader*** **nodig**
- **Traceur of System.js**

ES6 keywords `export` en `import` voor werken met module

```javascript
// modules-foobar.js
export var foo = 'foo';
export var bar = 'bar';

export class Person{
   constructor(){
      this.name = 'Peter Kassenaar'
   }
   getName(){
      return this.name;
   }
}
```

```javascript
import { foo, bar } from '../es6/modules-foobar';
console.log(foo); // 'foo'

import * as foobar from '../es6/modules-foobar';
console.log(foobar.foo); // 'foo'
console.log(foobar.bar); // 'bar'
console.log('\n');

var person = new foobar.Person();
console.log('Naam van persoon', person.getName());
```

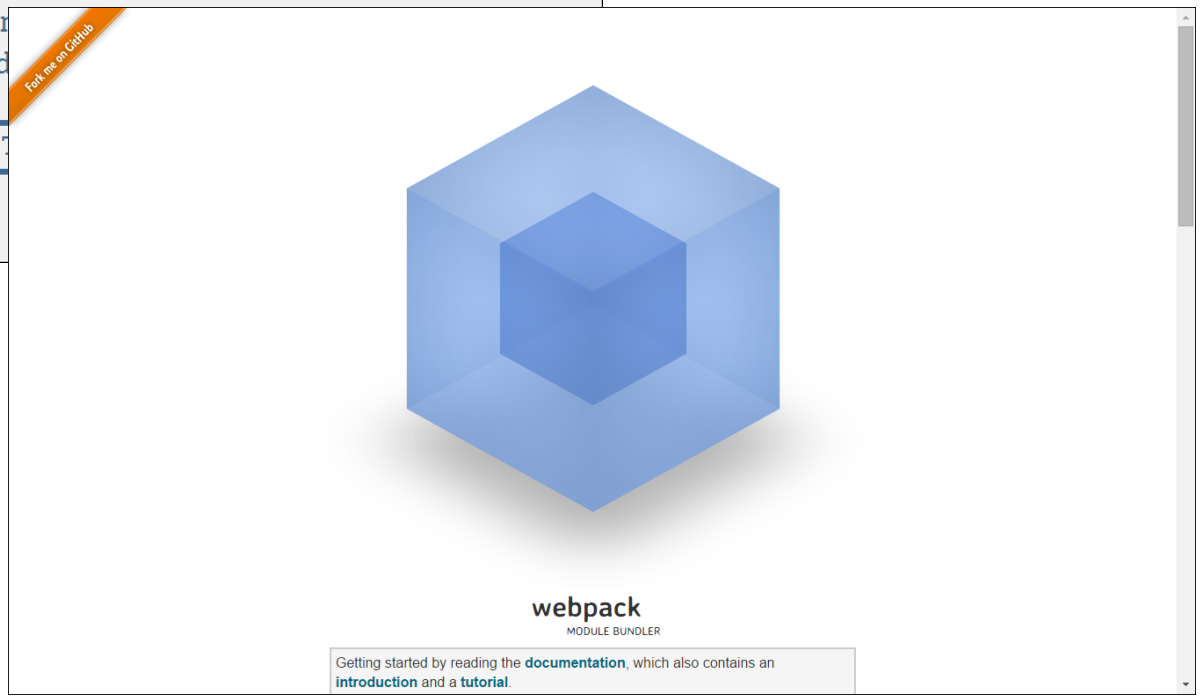
“In ECMAScript 6, modules are stored in files. There is exactly one module per file and one file per module.”

http://exploringjs.com/es6/ch_modules.html#ch_modules

Optional – Module loader Browserify

- Probleem:
  - Browser kan geen modules laden;
  - Browser kan geen scripts laden die modules laden met `import {}` …
- Oplossing:
  - Gebruik Browserify of WebPack om modules te bundelen en te `require('…')`-en

http://browserify.org/



https://webpack.github.io/

Globaal stappenplan

1. `npm install --save-dev browserify`

   Eventueel aanvullend : `npm install --save-dev watchify`

2. Npm scripts schrijven:

```
"scripts": {
  "start": "gulp",
  "build": "browserify js/app.js -o js/bundle.js",
  "watch": "watchify js/app.js -o js/bundle.js"
},
```

3. Index.html aanpassen, zodat wordt verwezen naar bundle.js

More info on browserify

- https://www.youtube.com/watch?v=CTAa8IcQh1U
- http://browserify.org/
- https://github.com/substack/browserify-handbook
- https://webpack.github.io/

Belangrijke ES6 Features

Maps/ Sets	Classes	Block Scope	Destructuring
Arrow Functions	Modules	Default/Rest Parameters	More...

Block scope – `let` en `const`

- **Standaard JavaScript: variabelen hebben *function scope*.**

"Both `let` and `const` create variables that are block-scoped – they only exist within the innermost block that surrounds them"

```
// New: Using let
for (let i = 0; i < 5; i++) {
    age += 5;
}

try {
    console.log(i);
} catch (e) {
    console.log('i is out of scope due to using let!');
}
```

Const

Variables created by `let` are mutable.

`Const` creates immutable variables:

```
let foo = 'foo';
foo = 'bar';
console.log(foo); // 'bar'

// const
const bar = 'baz';
bar = 'qux'; // TypeError: "bar is read-only"
```

Let en const : wanneer gebruiken?

- Gebruik in ECMAScript 2015 eigenlijk *altijd* `let` in plaats van `var`
- Goede upgrade-oefening: open een bestaand JavaScript-project en vervang alle ‘`var`’s door `let`’s.

- Gebruik liever nog `const` in plaats van `let`. Alleen als de variabele die je declareert ook daadwerkelijk gewijzig wordt, of –mag worden, gebruik dan `let`.

- Echter, praktijk: meeste developers gebruiken `let`. `Const` wordt alleen in specifieke gevallen (expliciet immutable) gebruikt.

Belangrijke ES6 Features

Maps/ Sets	Classes	Block Scope	Destructuring
Arrow Functions	Modules	Default/Rest Parameters	More...

Default parameters

Assign default value to a parameter

Similar to Java / C#

Provided parameters overrule default parameters

```
// ‘Default’ parameters in ES5
function foo(x, y) {
    x = x || 0;
    y = y || 0;
    // do something with x and y
}
```

ES6:

```
class Car {
    currentYear() {
        return new Date().getFullYear();
    }
    //The year parameter is a "default parameter"
    setDetails(make = 'No Make', model = 'No Model',
            year = this.currentYear()) {
        console.log(make + ' ' + model + ' ' + year);
    }
}
var car = new Car();
//Testing default parameters
car.setDetails('Toyota', 'Tundra');
car.setDetails();
```

Named Parameters

Provide object as parameter

- Not as nice as in Python, C#, IMO.

```
// Named parameters in ES6, with empty object as default parameter
function selectEntriesES6({ start=0, end=-1, step=1 } = {}) {
   console.log('ES6 start: ', start);
   console.log('ES6 end: ', end);
   console.log('ES6 step: ', step);
   // ...
}
// Calling ES6 function
selectEntriesES6({start: 10, end: 100, step: 10});
```

Rest Parameters

- Pass indefinite number of parameters to a funciton
- Three dots, followed by parameter name

```
...parameterName
```

- Simply "The rest of the parameters"
- Has nothing to do with RESTful Services

```
// Rest parameters in ES6
function foo(...args){
   for (let i of args){
      console.log('parameter: ', i);
   }
   //...
}
function bar(x, ...y) {
   //...
}
foo (1, 2, 3); // prints: 1, 2, 3
bar('a', 'b', 'c'); // x = 'a'; y = ['b', 'c']
```

```
setDetails(make = 'No Make', model = 'No Model',
         year = this.currentYear(), ...accessories) {
   console.log(make + ' ' + model + ' ' + year);
   if (accessories) {
      for (let i of accessories){
         console.log('\n' + i);
      }
   }
}
```

Belangrijke ES6 Features

Maps/ Sets	Classes	Block Scope	Destructuring
Arrow Functions	Modules	Default/Rest Parameters	More...

Destructuring

"ECMAScript 6 supports destructuring: a convenient way to extract values from data stored in (possibly nested) objects and Arrays."

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

Destructuring

```
// Destructure object
var {total2, tax2} = {total:9.99, tax:.50};


// Destructure array
var [red, yellow, green] = ['red', 'yellow', 'green'];
console.log(`Destructuring colors: ${red} ${yellow} ${green}`);
```

Ignoring Specific Members

```
var [red2, , green2] = ['red', 'yellow', 'green'];
console.log(`Destructuring with an ignore: ${red2}
             ${green2}`);
```

Destructuring

Praktijk:

Vaak gebruikt om bepaalde objecten of classes uit libraries te halen, als je niet de hele library nodig hebt:

```
import { Component, Input } from 'angular2/core';
```

Belangrijke ES6 Features

Maps/ Sets	Classes	Block Scope	Destructuring
Arrow Functions	Modules	Default/Rest Parameters	More...

More on ES6

- New array functions
- Typed arrays
- Generators/Yield
- New regex functions
- Promises
- http://exploringjs.com/es6/

Suggested upgrade path

- **Upgraden hoeft geen 'big bang' te zijn.**
- **Eerst kleine dingen veranderen, daarna meer.**
- **Suggesties:**
  1. Transpiler instellen (Babel + watch) !
  2. Eerst `var` vervangen door `let` en `const`
  3. Arrow functions gaan gebruiken
  4. Template literals inzetten in plaats van `'...' + '...' + ...`
  5. Classes gebruiken in plaats van functions
  6. Checken of je collections als `Map` en `Set` kunt gebruiken in plaats van arrays
  7. Bundling / minifying