ES6 ~~is~~ **vs** Today

A walk through integrating
incoming JS features
where and when makes sense

by Andrea Giammarchi
@WebReflection

What's on the ES6 menu

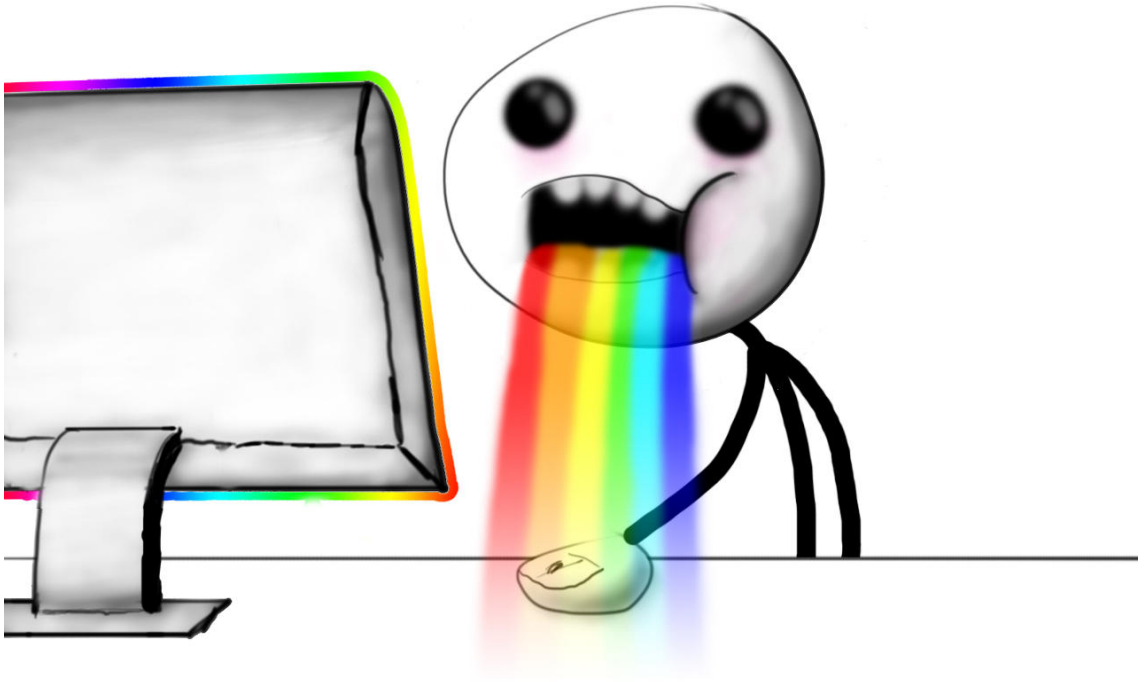
fat arrow, classes, enhanced object literals, template strings, destructuring, default/rest/spread arguments, let, const, iterators via for..of, generators, comprehension, unicode, modules, loaders, map, set, weakmap, weakset, proxies, symbols, subclassable built-ins, promises, math/number/string/object APIs, binary and octal literals, reflect API, tail calls

What's on the ES6 menu

fat arrow, classes, enhanced object literals, template strings, destructuring, default/rest/spread arguments, let, const, iterators via for..of, generators, comprehension, unicode, modules, loaders, map, set, weakmap, weakset, proxies, symbols, subclassable built-ins, promises, math/number/string/object APIs, binary and octal literals, reflect API, tail calls

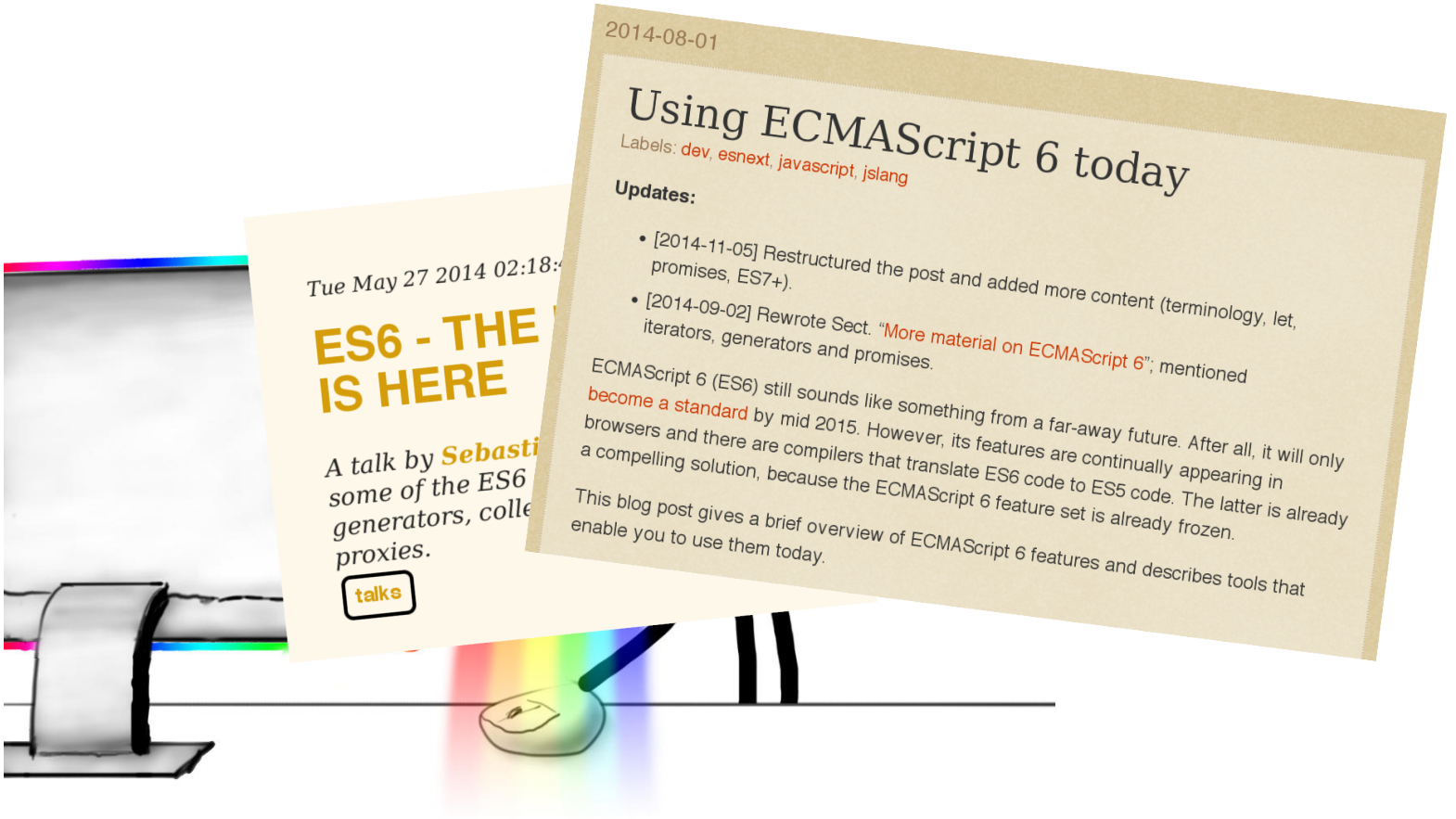
How Developers React

How Developers React



How Developers React

How Developers React



How Developers React

ECMAScript 6 Tools

Transpilers

- 6to5 - Turn ES6+ code into vanilla ES5 with no runtime
- Traceur compiler - ES6 features > ES5. Includes classes, generators, promises, destructuring patterns, default parameters & more.
- es6ify - Traceur compiler wrapped as a Browserify v2 transform
- 6to5ify - 6to5 transpiler wrapped as a Browserify transform
- es6-transpiler - ES6 > ES5. Includes classes, destructuring, default parameters, spread
- Square's es6-module-transpiler - ES6 modules to AMD or CJS
- Square's esnext - next generation JavaScript to today's JavaScript transformer
- es6now - ES6 to ES5 compiler and ES6 runtime environment
- Facebook's regenerator - transform ES6 yield/generator functions to ES5
- Facebook's jstransform - A simple utility for pluggable JS syntax transforms. Comes with a small set of ES6 -> ES5 transforms
- defs - ES6 block-scoped const and let variables to ES3 vars
- es6_module_transpiler-rails - ES6 Modules in the Rails Asset Pipeline
- Some Sweet.js macros that compile from ES6 to ES5
- Bitovi's transpile - Converts ES6 to AMD, CJS, and StealJS.
- regexpu — Transform Unicode-aware ES6 regular expressions to ES5

2014-08-01

Using ECMAScript 6 today

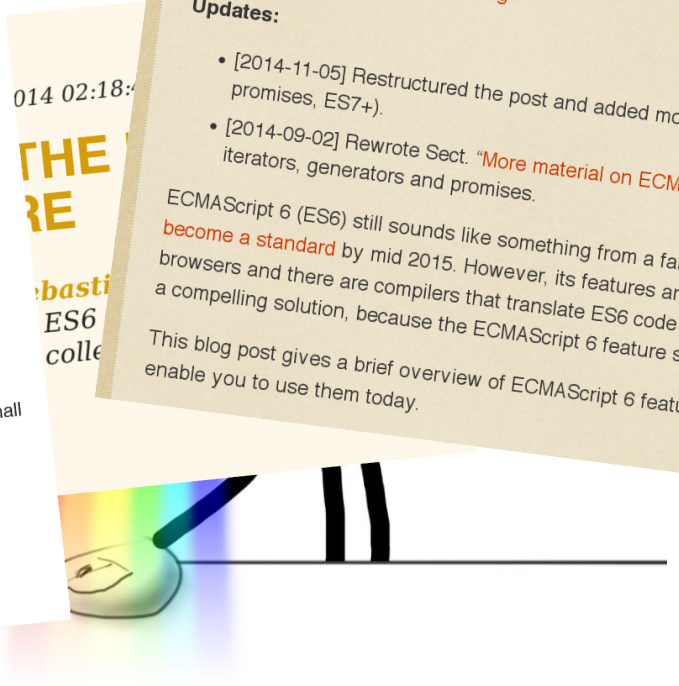
Labels: dev, esnext, javascript, jslang

Updates:

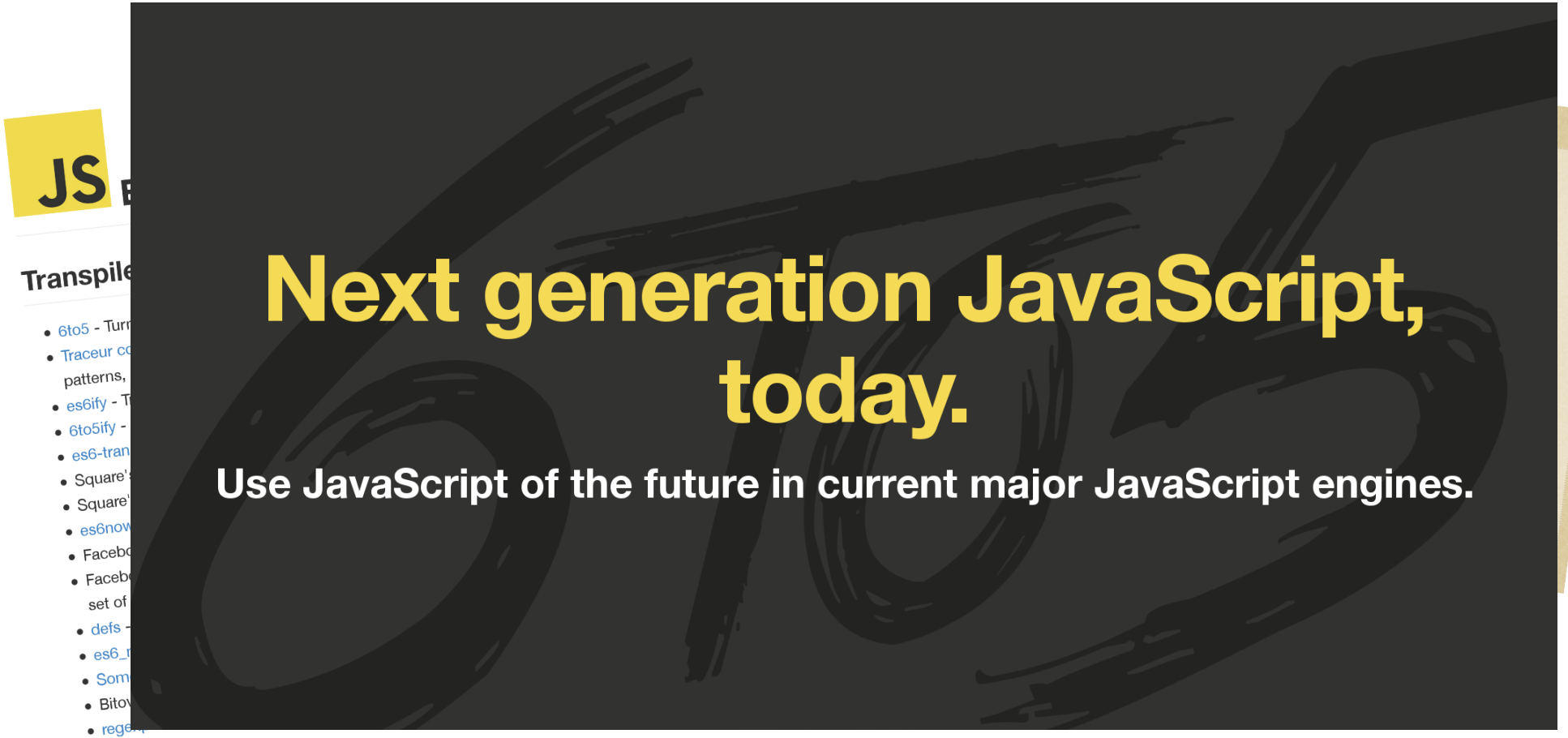
- [2014-11-05] Restructured the post and added more content (terminology, let, promises, ES7+).
- [2014-09-02] Rewrote Sect. "More material on ECMAScript 6"; mentioned iterators, generators and promises.

ECMAScript 6 (ES6) still sounds like something from a far-away future. After all, it will only become a standard by mid 2015. However, its features are continually appearing in browsers and there are compilers that translate ES6 code to ES5 code. The latter is already a compelling solution, because the ECMAScript 6 feature set is already frozen.

This blog post gives a brief overview of ECMAScript 6 features and describes tools that enable you to use them today.



How Developers React



JS ... a hype driven community

JS ... a hype driven community

- just because is new, it MUST be used ASAP

JS ... a hype driven community

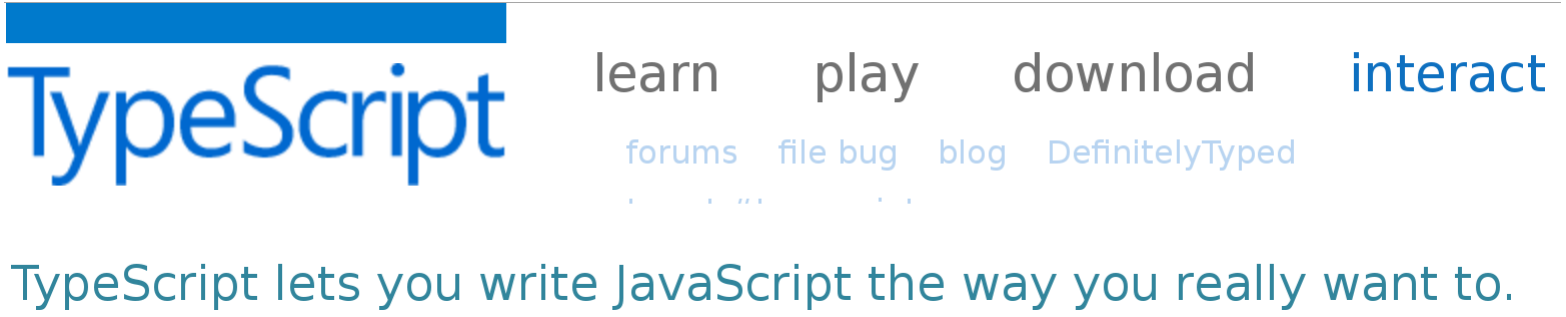
- just because is new, it MUST be used ASAP
- sometimes not even sure what it is supposed to solve

JS ... a hype driven community

- just because is new, it MUST be used ASAP
- sometimes not even sure what it is supposed to solve
- often not a clue if it will solve “the problem” properly

JS ... a hype driven community

- just because is new, it MUST be used ASAP
- sometimes not even sure what it is supposed to solve
- often not a clue if it will solve “the problem” properly



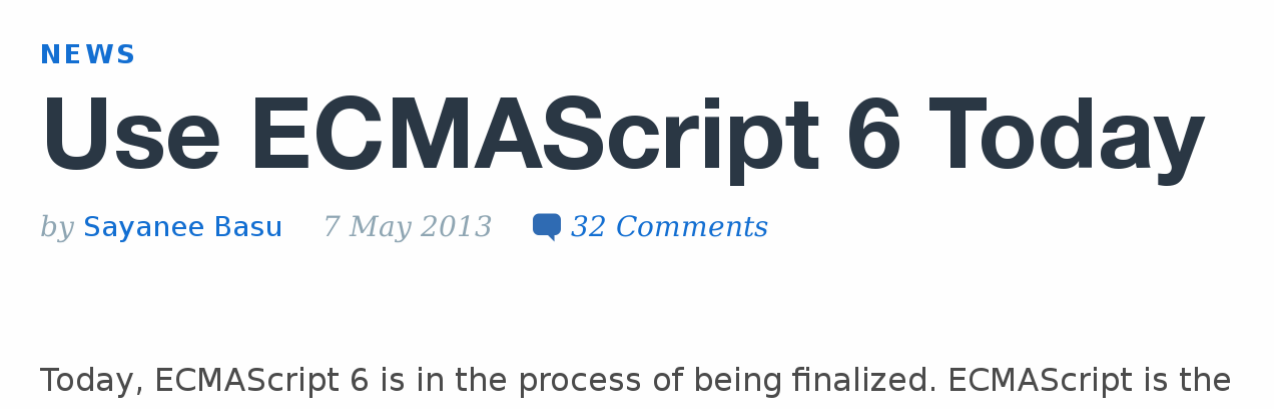
JS ... a hype driven community

- just because is new, it MUST be used ASAP
- sometimes not even sure what it is supposed to solve
- often not a clue if it will solve “the problem” properly

```
do {
    magic();
} while (
    i.sleep()
);
```

JS ... a hype driven community

- just because is new, it MUST be used ASAP
- sometimes not even sure what it is supposed to solve
- often not a clue if it will solve “the problem” properly



Tomorrow VS Today

Tomorrow VS Today

```
function shoutOut(name, {
  greeting = 'hello',
  target = 'world'
}={}) {
  return `${name} says ${greeting} ${target}!`;
}

shoutOut('Bob');
// returns: "Bob says hello world!"

shoutOut('Jake', {
  greeting: 'go away'
});
// returns: "Jake says go away world!"
```

Jake Archibald @jaffathecake · Jan 8

The future of JS is great: Destructuring & defaults make optional args really simple pic.twitter.com/1LtSZaMivW

349 310

Tomorrow VS Today

```
function shoutOut(name, opts) {
  if (arguments.length < 2) {
    opts = {};
  }

  if (!'greeting' in opts) {
    opts.greeting = 'hello';
  }

  if ('target' in opts) {
    opts.target = 'world';
  }

  var greeting = opts.greeting;
  var target = opts.target;

  return name + ' says ' + greeting + ' ' + target;
}
```

Jake Archibald @jaffathecake · Jan 8
@Paul_Kinlan here's the ES5 version pic.twitter.com/n98PrbvJu4

5

Tomorrow VS Today

```
function shoutOut(name, opts) {
  if (arguments.length < 2) {
    opts = {};
  }

  if (!'greeting' in opts) {
    opts.greeting = 'hello';
  }

  if ('target' in opts) {
    opts.target = 'world';
  }

  var greeting = opts.greeting;
  var target = opts.target;

  return name + ' says ' + greeting + ' ' + target;
}
```

gistfile1.js

```
1  function shoutOut(name, opts) {
2    opts || (opts = {});
3    return [
4      name, 'says',
5      opts.greeting || 'greeting',
6      opts.target || 'target'
7    ].join(' ');
8  }
```

Jake Archibald @jaffathecake · Jan 8
@Paul_Kinlan here's the ES5 version pic.twitter.com/n98PrbvJu4

5

Tomorrow VS Today

```
1 function greetings(name) {
2   return `Hello ${name}!`;
3 }
4
5 greetings('chaps');
```

Tomorrow VS Today

```
1 function greetings(name) {
2   return `Hello ${name}!`;
3 }
4
5 greetings('chaps');
```

vs

```
1 'Hello ${name}!'.template({
2   name: 'chaps'
3 });
```

Tomorrow VS Today

```
function greetings(name) {
  return `Hello ${name}!`;
}

greetings('chaps');
```

vs

```
'Hello ${name}!'.template({
  name: 'chaps'
});
```

https://gist.github.com/WebReflection/8f227532143e63649804

Tomorrow VS Today

```
String.prototype.template = function (object) {
  // Andrea Giammarchi - WTFPL License
  var
    stringify = JSON.stringify,
    re = /\$\{([\S\s]*?)\}/g,
    evaluate = [],
    i = 0,
    m
  ;
  while (m = re.exec(this)) {
    evaluate.push(
      stringify(this.slice(i, re.lastIndex - m[0].length)),
      '(' + m[1] + ')'
    );
    i = re.lastIndex;
  }
  evaluate.push(stringify(this.slice(i)));
  // Function is needed to opt out from possible "use strict" directive
  return Function('with(this)return' + evaluate.join('+')).call(object);
};
```

https://gist.github.com/WebReflection/8f227532143e63649804

Tomorrow VS Today

```
function toBeDestructured() {
  return {
    a: 'a',
    b: 'b',
    c: 'c'
  };
}

var {a, b, c} = toBeDestructured();
console.log('as simple as ' + a + b + c);
```

Tomorrow VS Today

```
function toBeDestructured() {
  return {
    a: 'a',
    b: 'b',
    c: 'c'
  };
}

with(toBeDestructured()) {
  console.log('as simple as ' + a + b + c);
}
```

different types of sugar

different types of sugar

It's OK to think about ES6 as a superset of ES5 where almost everything can desugar into compatible ES5 syntax

different types of sugar

It's OK to think about ES6 as a superset of ES5 where almost everything can desugar into compatible ES5 syntax

almost

different types of sugar

- fat arrow
- templates
- property methods assignment
- enhanced object literals

different types of sugar

- fat arrow
- templates
- property methods assignment
- enhanced object literals

different types of sugar

- classes
- Weak{Map,Set}
- builtin subclassing
- generators
- Symbols

different types of sugar

- classes
- Weak{Map,Set}
- builtin subclassing
- generators
- Symbols

different types of sugar

- classes
- Weak{Map,Set}
- builtin subclassing
- generators
- Symbols

different types of sugar

- classes
- Weak{Map,Set}
- builtin subclassing
- generators
- Symbols

different types of sugar

- classes
- Weak{Map,Set}
- builtin subclassing
- generators
- Symbols

try not to overload

- polyfills and transpilers bring new features to old engines

try not to overload

- polyfills and transpilers bring new features to old engines
- old engines are usually slower

try not to overload

- polyfills and transpilers bring new features to old engines
- old engines are usually slower
- polyfills and transpilers adds a lot of boilerplate even to simple operations

try not to overload

- polyfills and transpilers bring new features to old engines
- old engines are usually slower
- polyfills and transpilers adds a lot of boilerplate even to simple operations
- the rule here: try not to end up in a situation like the following one shown in the picture

try not to overload

A closer look to transpiled classes

A closer look to transpiled classes

A closer look to transpiled classes

Live Demo

Tomorrow VS Today

```
class TestB extends TestA {
  method1() {
    return super.method1();
  }
  get value1() {
    return super.value1;
  }
  method2() {
    return new WeakMap;
  }
}
```

Tomorrow VS Today

```
var TestB = Class({ extends: TestA,
  method1() {
    return this.super();
  },
  get value1() {
    return this.super();
  },
  method2() {
    return new WeakMap;
  }
});
```

https://github.com/WebReflection/es-class

A closer look to transpiled classes

based on test.js parsing

- * **6to5**: 2.2K output and constructors degraded up to the following:

- **traceur**: 863B output + 126.5K runtime + degraded perf
- * **TypeScript**: 1.2K output and best performances beside empty extended constructors (however, not an ES6 substitute)
- **es-class**: 477B output + 1.2K Class utility

* exposed some potential global scope conflict, e.g. this._get = ... or __extends = ...

es-class also includes

- lightweight traits / mixins included
- grouped static properties and methods
- lightweight interfaces with one-shot runtime check
- compatible down to IE6 if no {get,set}ters are used
- usable as hybrid solution for 6to5 transformers

  (e.g. 6to5 –whitelist=arrowFunctions,propertyMethodAssignment main.js)

https://github.com/WebReflection/es-class

ES6 ~~is~~ **vs** Today

it does not have to be all at once
it doesn't have to break the Web
in some case it might be even superfluous

ES6 ~~is~~ **vs** Today

Thank You!