What should the average JS ninja know about upcoming JavaScript and Browser features?

Daniel Steigerwald
twitter.com/steida

mloc.js 2014

About me

- Google Developer Expert
- JS developer and speaker
- Este.js creator

JavaScript history

- JavaScript or EcmaScript?
  - JavaScript is umbrella term
- ES3
- ES5
- ES6 (aka ES.next, harmony, Ecma-262 Edition 6)

Plain old JavaScript is dead. Long live to JavaScript.

ES6

- A lot of new syntax sugar
- New language features

Why should I care?

- ES6 fixes what is broken.
- = Interoperability!

Block scoping - let, const

Block scoping instead of function scoping.

Why it's useful:

- No more "declaring your variables at the top of the function".
- Variables for iterations/enumerations only.

Function scope (var) Example

```
function foo(bar) {
  console.log(tmp);
  // undefined
  if (bar) {
    var tmp = x;
  }
}
```

Block scope (let, const) example

```
function foo(bar) {
  console.log(tmp);
  // ReferenceError: tmp is not defined
  if (bar) {
    let tmp = x;
  }
}
```

Block-local variable in for in loop

```
for (let i = 0; i < 3; i++) {
  console.log('i:', i);
}

// ReferenceError: i is not defined
console.log(i);
```

Block scoping - advice

- Use `let` instead of `var`.

Destructing assignment - array

Makes extracting values from complex arrays and objects more convenient.

```
// ES3
let first = someArray[0];
let second = someArray[1];
let third = someArray[2];

// ES6
let [first, second, third] = someArray;
```

Destructing assignment - array

Avoiding temporary variables. We can use destructuring assignment to swap two values.

```
// Btw, who might need that?
var a = 1;
var b = 3;
[a, b] = [b, a];
```

Destructing assignment - object

```
var robot = {
  name: "Bender",
  surname: "Rodríguez"
};

// ES3
var name = robot.name;
var surname = robot.surname;

// ES6
var {name, surname} = robot;
```

Destructing assignment - usage

- Extracting from deeply nested **third party** JSON's
- Multiple return values from functions

- Configuration object parameters

```
// ES3
function remove(options) { var url = options.url; ...}
// ES5
function remove({url, line, column, force}) {...}
```

- Importing a subset of a module

let {div, h3, textarea} = React.DOM;

Destructing assignment - advices

- Use it sparingly, think twice.
- Heavy querying tends to be an anti-pattern.
- Don't return more than one value from function unless you have pretty good explanation why you have to.

```
// Beware! Single Responsibility Principle violation.
function returnMultipleValues() {
  // Beware! Unnamed/anonymous object.
  return [1, 2];
}
var [foo, bar] = returnMultipleValues();
```

Arrow functions

- lambda syntax, less to type
- lexical this, no more that=this

```
// ES5
var squares = [1, 2, 3].map(function (x) {
  return x * x;
});
// ES6
var squares = [1, 2, 3].map(x => x * x);
```

Arrow functions - lexical this

```
// ES3
var that = this;
element.addEventListener('click', function (e) {
  this.foo();
});

// ES5
element.addEventListener('click', function (e) {
  this.foo();
}.bind(this));

// ES6
element.addEventListener('click', => this.foo());
```

Arrow functions - advices

- Use it for one-line functions.
- Don't use it for anonymous functions with more than one lines

```
// Anonymous function with more than one lines is anti-pattern.
$('#foo').onclick((e) =>
  // some code here
  // more code...
);

// Better. Handler deserves a name.
$('#foo').onclick(onFooClick);
```

Parameter handling

- Parameter default values
  - `function foo(x, y=0) {...}`
- Rest parameters
  - `function foo(...args) {...}`
- Named parameters
  - `function foo({force}) {...}`
- Named parameters with default value
  - `function foo({force=false}) {...}`

ES6, Classes, Modules, and OOP

- new class syntax
- new module syntax
- OOP sugar

Finally real interoperability across libraries.

A huge boost for ecosystem.

ES6 Classes

- Just another syntax sugar, but...
- ... no more custom prototype helpers!

  ~~`new Ext.Class({..}), Class.create()...`~~

```
class Animal {
  constructor(name) {
    this.name = name;
  }
  sayName() {
    console.log(this.name);
  }
}
```

ES6 Modules

- Finally... no more custom module libraries!
- Configurable sugar for CommonJS

```
// Profile.js
export var firstName = 'David';
export var lastName = 'Belle';

// ProfileView.js
import {firstName, lastName} from './Profile';
function setHeader(element) { element.textContent = firstName + ' ' + lastName; }
```

ES6 Classes and Modules consequences

- better tools
  - refactoring
  - code coverage
- better architecture!
  - github.com/angular/di.js (**Advice**: Check it!).

JavaScript is finally catching up mature ecosystems.

ES6 Symbols

- a new kind of primitive value typeof 'symbol'
- each symbol is unique
- supposed to replace "objects as enums"
- can be used as object property

```
// ES3
var Color = {red: 1};

// ES6
let red = Symbol();
```

ES6 Symbols - advice

Don't use them for private properties simulation. Please. Don't.

curiosity-driven.org/private-properties-in-javascript (Note it's not even a link!)

Why?

- a false sense of safety
- wrong direction and waste of time
- runtime hack (instead of design time certainty)
- another level of abstraction
- there are much better options (Google Closure, TypeScript)

ES6 - what else?

- iterables and iterators
- generators
- for-of: A better loop
- Comprehensions
- proxies
- Maps and Sets
- template strings
- ... and more

How to use ES6 today?

- Node.js
    - node --harmony --use-strict
- transpilers
    - Google Traceur: github.com/google/traceur-compiler
    - Facebook Regenerator: facebook.github.io/regenerator
    - Microsoft TypeScript: typescriptlang.org

Should I use ES6 today?

Yes

- If you are willing to learn new things
- If you want to be "future compatible"
- If you already know clean code principles
- If your dev stack is well tuned

No, If you answered **no** at least once.

Should I use ES6 today?

An inconvenient truth:

- software developers are uneducated
- software developers does not use proper dev tools

= SDD (stress driven development)

= Let's focus on truly important stuff first.

Developers are uneducated (me too)

- We all write a shitty code, but only a few of us are aware of that.
- Such code is not only pain to maintain, but also pain to write.
- scripting versus programming (That's why drag and drop/copy and past development will never scale).
- Clean code is panacea.

cleancoders.com

Developers does not use proper dev tools

- build tools and task runners
  - linting
  - building
  - transpiling
  - optimizing
  - = automatize everything as much as possible
- client dependency managers
- MVC or whatever third party libraries

Build tools and task runners

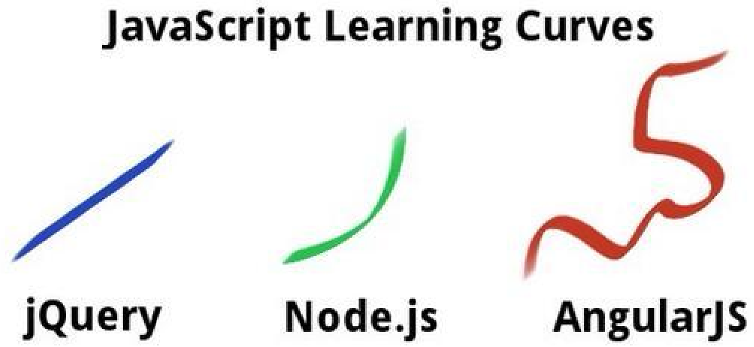
- Choose Grunt.js or gulp.js
- Use it
- Profit!

Client dependency/packages managment

- Choose Bower.js or Bower.js
- Use it
- Profit!

MVC or whatever third party libraries

Should I use Angular or Polymer or flexible, dependency-free, lightweight, device-agnostic, modular, baked-in, component framework MVC library shoelacestrap to help you kickstart your responsive CSS-based app architecture backbone kitchensink tweetybirds? (via html9responsiveboilerstrapjs.com)



To rewrite, or not to rewrite: that is the question.

- Not invented here syndrome versus reinvent wheel.
- Este.js as example of meta MVC framework.

Keywords for 2014

- Promises (jQuery is wrong - http://domenic.me/2012/10/14/youre-missing-the-point-of-promises)
- WebRTC Peer-to-peer connections
- Facebook React
- Functional programming (put state where it belongs)
- Flexible box layout
- Polymer stuff (Shadow DOM, Pointer Events, Custom Elements)
  - Do we really need that?
- Isomorphic applications

Internet Explorer ugh?

Need a bigger market share? Localize your app instead of hacking obsolete **never-green** browsers.

Evergreen Web Browser: a web browser that automatically updates itself.

[theie8countdown.com]

[theie9countdown.com]

Still have a time? Probably no, so ask me after talk.

- Let's talk about MVC frameworks
  - todomvc.com
- code versus configuration
- Future of MVC frameworks

Links

- ECMAScript Support Matrix: pointedears.de/scripts/test/es-matrix
- ECMAScript 6 specification: wiki.ecmascript.org/doku.php?id=harmony:specification_drafts
- Compatibility tables for support of HTML5, CSS3, SVG and more: caniuse.com
- ES6 list of tools by Addy Osmani: http://github.com/addyosmani/es6-tools
- gruntjs.com
- gulpjs.com
- todomvc.com
- polymer-project.org

Thank you!

Daniel Steigerwald

twitter.com/steida