Role of Babel in JS

Henry Zhu

- - Babel Overview
- - Why is Babel relevant

- - Open Questions

6to5 -> Babel

Sept 2014 Feb 2015

Goals

1. Transform new syntax into backwards compatible code

2. Avenue for developer feedback on TC39 proposals via config option (no specific agenda)

3. Toolchain for other static analysis tools (linter, fixer, etc)

Compiler Overview

Parser: babylon
Traversal: babel-traverse/babel-core
Printer: babel-generator

Plugins: babel-plugin-transform-x
Example: babel-plugin-transform-es2015-arrow-functions

Presets (a set of plugins): babel-preset-x
Example: babel-preset-es2015

babel / babel

1,126 | Fork | 1,842

Releases 542

Code | Issues

Babel is a compiler

babel | javascript | as

Edit

7,951 commits

MIT

Branch: 7.0

buunguyen committe

Clone or download

mit cfe0a84 14 hours ago

.github		2 months ago
doc		4 months ago
lib		a month ago
packages		2 days ago

babel-plugin-transform-fl...	Flow comments import export (#5675)	4 days ago
babel-plugin-transform-fl...	Add support for object type spread (#5525)	a month ago
babel-plugin-transform-fu..	v7.0.0-alpha.9	a month ago
babel-plugin-transform-js...	v7.0.0-alpha.9	a month ago
babel-plugin-transform-o...	v7.0.0-alpha.9	a month ago
babel-plugin-transform-o...	Fix incorrect property ordering with obj rest spread on nested (#5...	22 days ago
babel-plugin-transform-o...	v7.0.0-alpha.9	a month ago
babel-plugin-transform-p...	v7.0.0-alpha.9	a month ago
babel-plugin-transform-re..	Fix multiple var declarations in transform-react-constant-elements..	2 days ago
babel-plugin-transform-re..	Updated transform-react-display-name for createReactClass addo..	23 days ago
babel-plugin-transform-re..	v7.0.0-alpha.9	a month ago
babel-plugin-transform-re..	v7.0.0-alpha.9	a month ago
babel-plugin-transform-re..	v7.0.0-alpha.9	a month ago
babel-plugin-transform-re..	v7.0.0-alpha.9	a month ago
babel-plugin-transform-re..	Typecheck much more of the config loading process (#5642)	a month ago
babel-plugin-transform-re..	v7.0.0-alpha.9	a month ago
babel-plugin-transform-ru..	v7.0.0-alpha.9	a month ago
babel-plugin-transform-st...	v7.0.0-alpha.9	a month ago
babel-polyfill	v7.0.0-alpha.9	a month ago
babel-preset-es2015	Add test cases for bad options in babel-preset-es2015 (#5720)	10 days ago
babel-preset-es2016	v7.0.0-alpha.9	a month ago
babel-preset-es2017	v7.0.0-alpha.9	a month ago
babel-preset-flow	v7.0.0-alpha.9	a month ago
babel-preset-react	v7.0.0-alpha.9	a month ago
babel-preset-stage-0	v7.0.0-alpha.9	a month ago
babel-preset-stage-1	v7.0.0-alpha.9	a month ago
babel-preset-stage-2	v7.0.0-alpha.9	a month ago
babel-preset-stage-3	v7.0.0-alpha.9	a month ago
babel-register	add .mjs to list of well known extensions	23 days ago
babel-runtime	v7.0.0-alpha.9	a month ago
babel-template	v7.0.0-alpha.9	a month ago
babel-traverse	Fix multiple var declarations in transform-react-constant-elements..	2 days ago
babel-types	Fix ObjectProperty patterns (#5762)	2 days ago
README.md	6.0.0	2 years ago

Prototyping

In the browser

Babel built-ins

CLI	Require hook

Build systems

Broccoli	Browserify	Brunch	Duo	Gobble	Grunt	Gulp	jspm	Make
MSBuild	RequireJS	Rollup	Sprockets	Webpack	Webpack 1	Fly	Start	

Frameworks

Ember	Meteor	Rails	Sails

Test frameworks

AVA	Jasmine	Jest	Karma	Lab	Mocha

Utilities

Connect	Nodemon

Language APIs

C# / .NET	Node	Ruby

Polyfill

https://github.com/zloirock/core-js

Since Babel only transforms syntax (like arrow functions), you can use babel-polyfill in order to support new globals such as Promise or new native methods like String.padStart (left-pad). It uses core-js and regenerator. Check out our babel-polyfill docs for more info.

You can install the polyfill with

Shell Copy

```
npm install --save-dev babel-polyfill
```

Use it by requiring it at the top of the entry point to your application or in your bundler config.

Jakob Gruber

May 22 · 3 min read

es6.regexp.constructor polyfill moves RegExps onto slow path in V8 #5771

Open **schuay** opened this issue 2 hours ago · 2 comments

schuay Jakob Gruber - commented 2 hours ago

0 First Issue in this repo

See https://medium.com/@schuay/js-regexp-fast-and-slow-d29d6b77b06 and mishoo/UglifyJS2#1964 for more context.

es6.regexp.constructor (and possibly other regexp polyfills) cause all regexps to take the slow path in V8 by modifying RegExp.prototype.

Repro instructions here: mishoo/UglifyJS2#1964 (comment)

Input Code

```
require("babel-polyfill");
var str = "fooBar";
console.log( str.split(/(?=[A-Z])/) );
```

test.js hosted with ❤ by GitHub view raw

Ass

Lab

pe

Proj

Non

Mile

No

Not

Other tools powered via Babel

- Linting: ESLint + babel-eslint parser
- Formatting: prettier
- Minification: babel-minify (Babili)
- Codemods: babel-codemod, lebab 😄
- Code coverage: babel-plugin-instanbul

Babylon is a JavaScript parser used in Babel.

- The latest ECMAScript version enabled by default (ES2017).
- Comment attachment.
- Support for JSX and Flow.
- Support for experimental language proposals (accepting PRs for anything at least stage-0).

typescript is in progress!

Example

```
require("babylon").parse("code", {
  // parse in strict mode and allow module declarations
  sourceType: "module",

  plugins: [
    // enable special syntax
    "jsx",
    "dynamicImport"
  ]
});
```

Will Babylon support a plugin system?

Previous issues: babel/babel#1351, #500.

We currently aren't willing to commit to supporting the API for plugins or the resulting ecosystem (there is already enough work maintaining Babel's own plugin system). It's not clear how to make that API effective, and it would limit out ability to refactor and optimize the codebase.

Our current recommendation for those that want to create their own custom syntax is for users to fork Babylon.

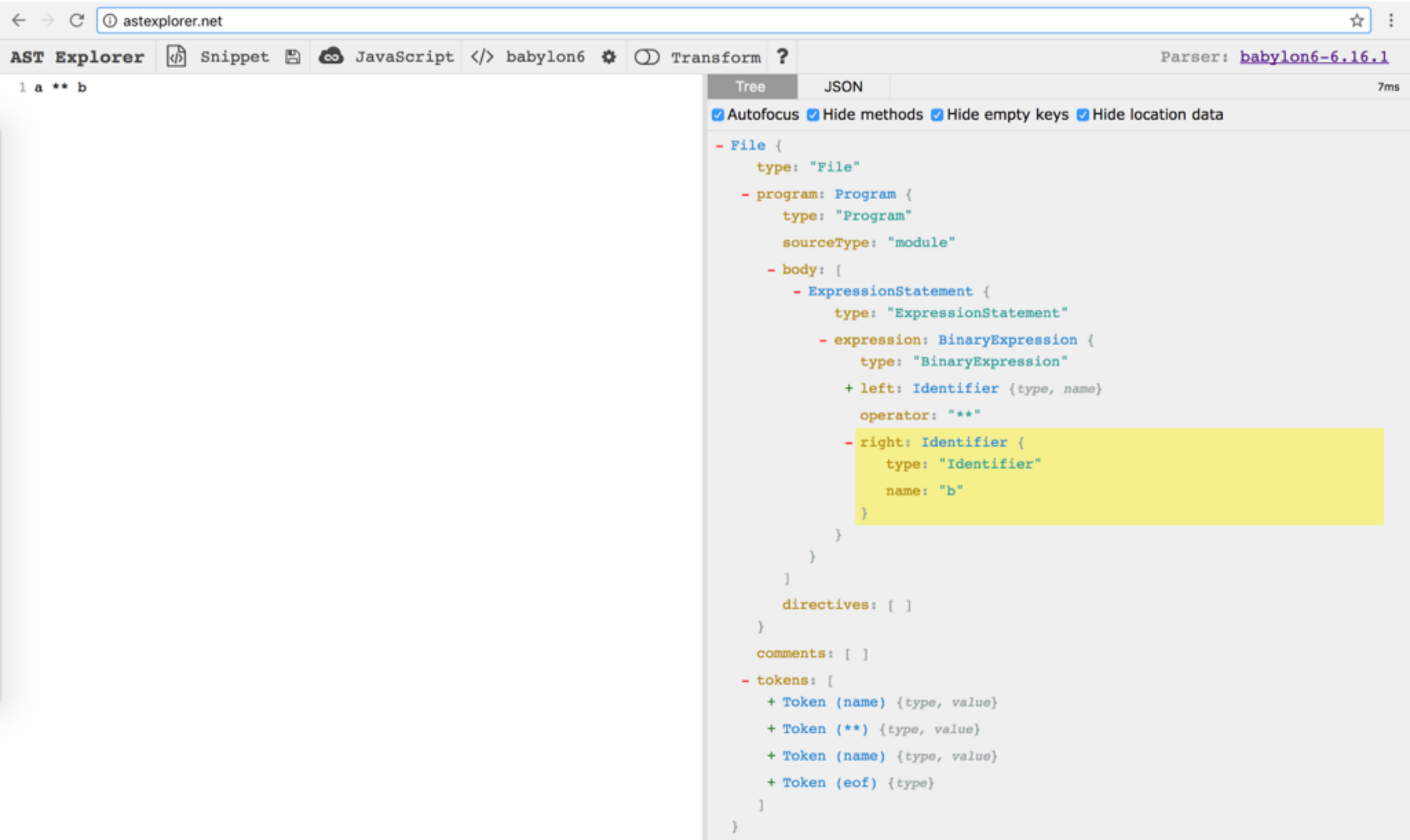
To consume your custom parser, you can add to your `.babelrc` via its npm package name or require it if using JavaScript,

```
{
  "parserOpts": {
    "parser": "custom-fork-of-babylon-on-npm-here"
  }
}
```

```
function square(n) {
  return n * n;
}
```


```
{
  type: "FunctionDeclaration",
  id: {
    type: "Identifier",
    name: "square"
  },
  params: [{
    type: "Identifier",
    name: "n"
  }],
  body: {
    type: "BlockStatement",
    body: [{
      type: "ReturnStatement",
      argument: {
        type: "BinaryExpression",
        operator: "*",
        left: {
          type: "Identifier",
          name: "n"
        },
        right: {
          type: "Identifier",
          name: "n"
        }
      }
    }]
  }
}
```

http://astexplorer.net/



AST Explorer | Snippet | JavaScript | babylon6 | Transform | ?

```
1 // literals
2 1 ** 2;
3
```

Tree | JSON

Autofocus | Hide methods | Hide empty key

```
BinaryExpression(path) {
  if (path.node.operator !== "**") return;
  const { left, right } = path.node;

  path.replaceWith(t.callExpression(
    t.memberExpression(
      t.identifier("Math"),
      t.identifier("pow")
    ),
    [left, right]
  ));
},
```

Plugin Output

```
24       )
25     }
26   }
27 };
28 }
```

Insertion, Removal, Replacement

```
path.insertBefore(node);
path.insertAfter(node);
path.unshiftContainer(node);
path.pushContainer(node);

path.remove();

path.replaceWith(node);
path.replaceWithMultiple(nodes);
```

```
$(el).after(node);
$(el).before(node);

$(el).remove();

$(el).replaceWith(node);
$(el).replaceAll(nodes);
```

DOM API

...

```
el.insertAdjacentHTML
el.appendChild
```

...

http://henryzoo.com/babel-plugin-slides/

So what?

https://www.npmjs.com/package/babel-core

Stats

330,690 downloads in the last day

1,906,272 downloads in the last week

8,035,147 downloads in the last month

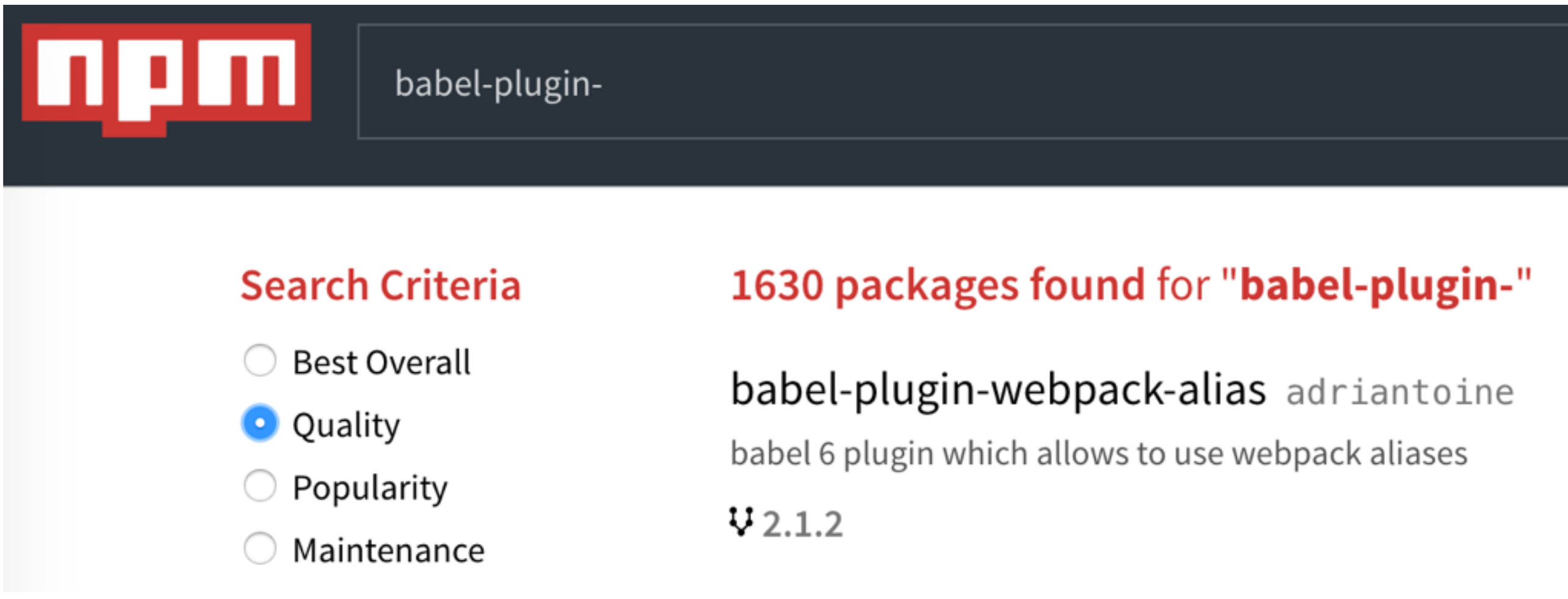
See who's using Babel

http://babeljs.io/users

Logos are added by company representatives. These companies may or may not be using Babel on their main web properties, but they're definitely using it somewhere in their organizations :)

facebook
CLOUDFLARE
NETFLIX
mozilla
YAHOO!
PayPal
airbnb
reddit
Spotify
Atlassian
NFL
bocoup
SOUNDCLOUD
zendesk
vimeo
npm
ember

Community Plugins

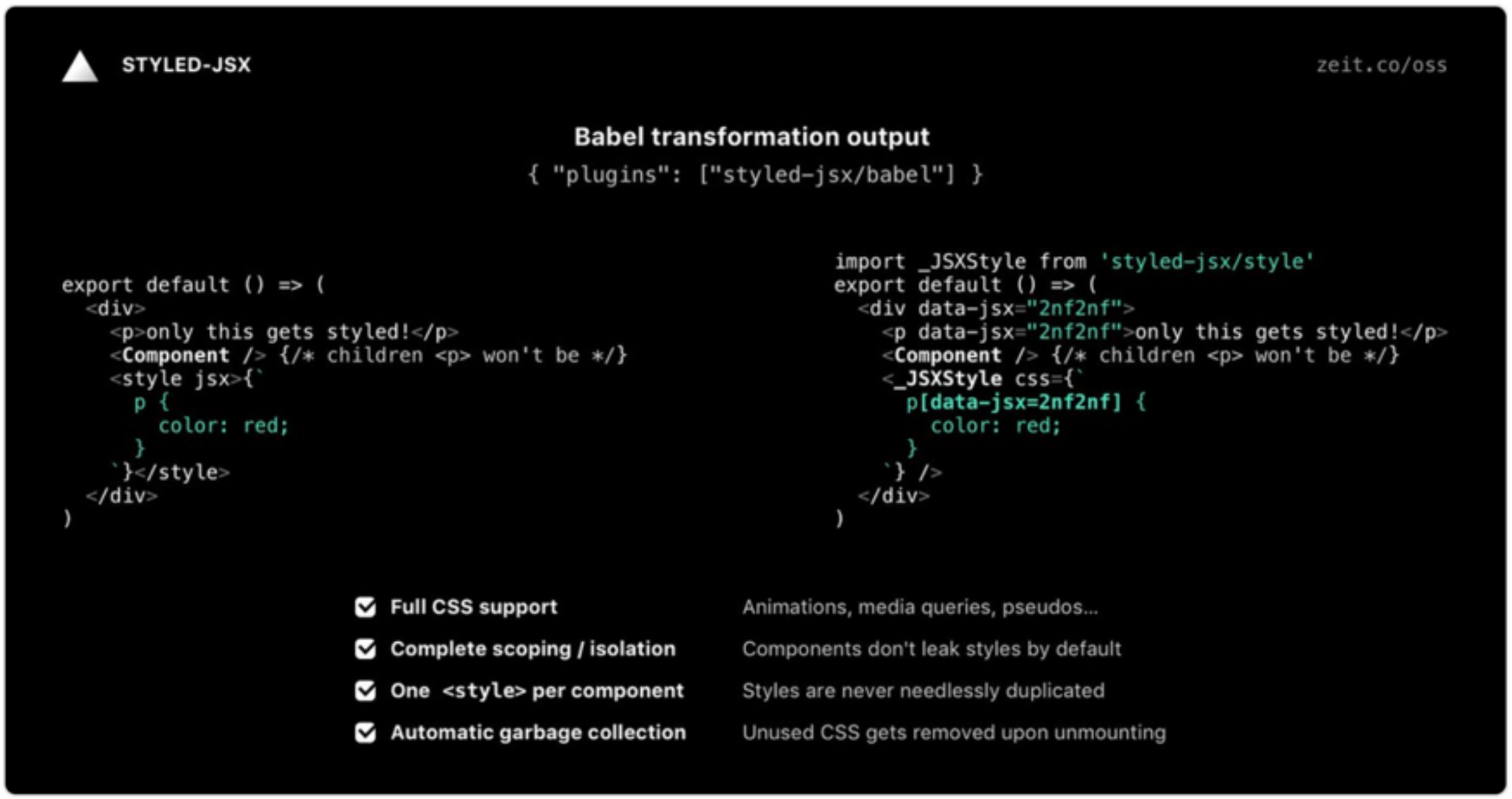


Guillermo Rauch

@rauchg

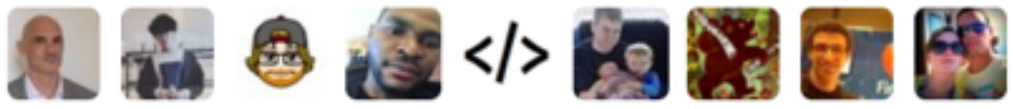
Following

styled-jsx just passed 1,000⭐
github.com/zeit/styled-jsx



RETWEETS 35

LIKES 162



Example

Transforms

babel-plugin-lodash

```
import _ from 'lodash';
import { add } from 'lodash/fp';

const addOne = add(1);
_.map([1, 2, 3], addOne);
```

roughly to

```
import _add from 'lodash/fp/add';
import _map from 'lodash/map';

const addOne = _add(1);
_map([1, 2, 3], addOne);
```

react-remove-prop-types

The problem solved

Remove React `propTypes` from the production build, as they are only used in development. You can **save bandwidth** by removing them.

Example

In

```
const Baz = (props) => (
  <div {...props} />
);

Baz.propTypes = {
  className: PropTypes.string
};
```

Out

```
const Baz = (props) => (
  <div {...props} />
);
```

Babel Plugin Flow Runtime

A babel plugin which transforms Flow annotations into `Type` instances available at runtime, and optionally checks values against those types.

Supports all of flow's syntax, aims for full compatibilty with flow, found a bug? Please report it.

What?

Turns code like this:

```
type User = {
  id: number;
  name: string;
};
```

Into code like this:

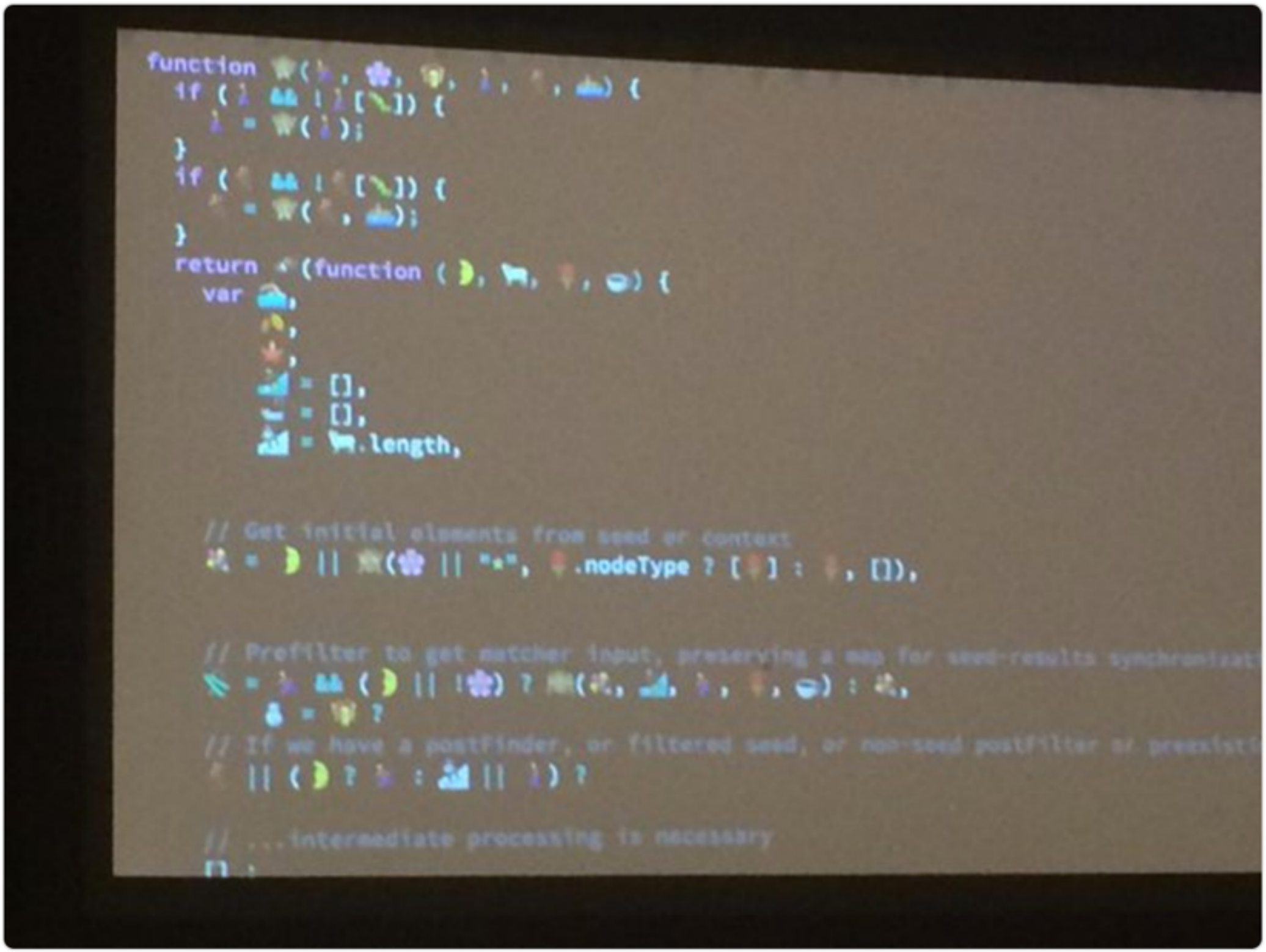
```
import t from 'flow-runtime';
const User = t.type('User', t.object(
  t.property('id', t.number()),
  t.property('name', t.string())
));
```

Which you can then use like this:

```
User.assert({id: 123, name: 'Sally'}); // ok
User.assert({id: false, name: 'Bob'}); // throws
```

Babel @babeljs · 29 May 2015

Announcing Babel Emojification npmjs.com/package/babel-… #jsconf



6 78 70

When do we not need Babel?

Mike Sherov

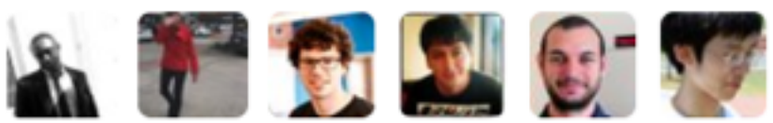
@mikesherov

Following

The goal of yearly Babel presets is to no longer need to exist.

RETWEET
1

LIKES
5



Addy Osmani

@addyosmani

Following

@left_pad Do you have a feel for how often folks transpile their entire ES2015/next src > ES5 vs just features their targets dont support?

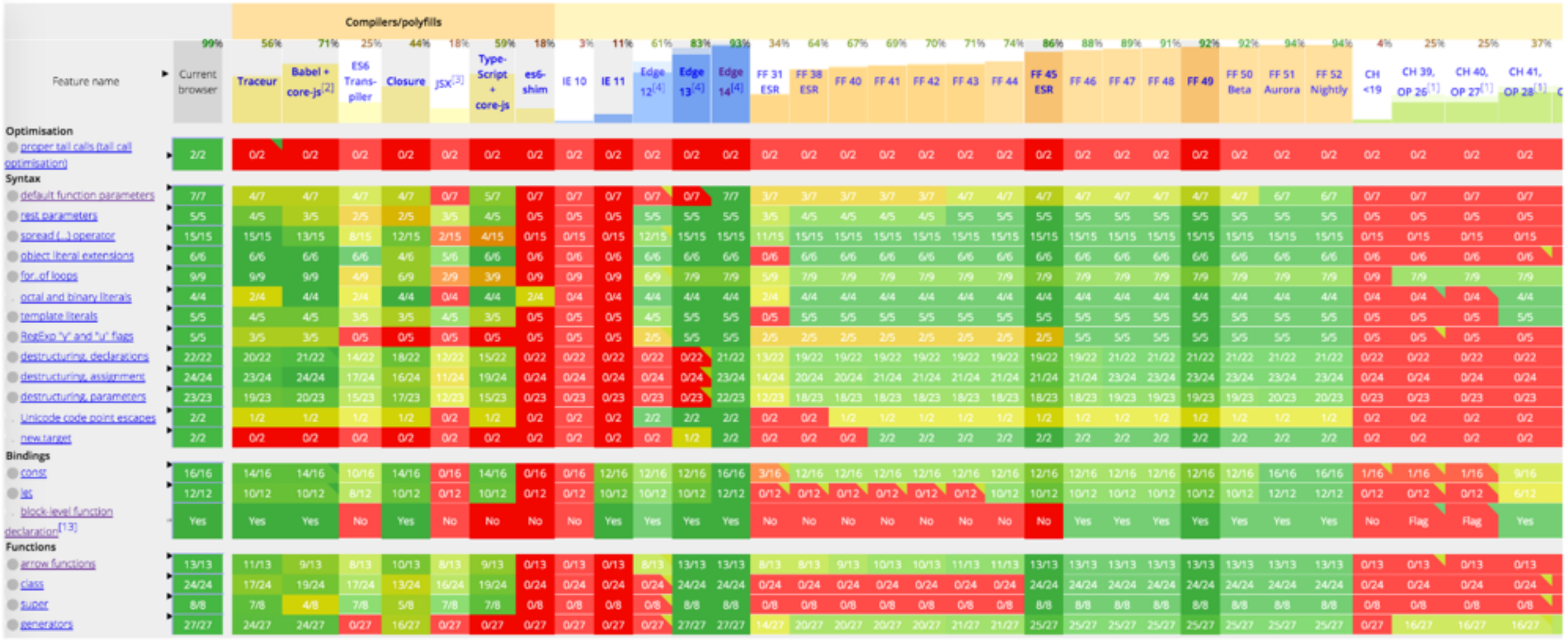
RETWEETS 2

LIKES 4

2:37 PM - 3 Dec 2016

babel-preset-env (“autoprefixer” for Babel)

JavaScript compilation is a moving target: There are yearly updates to the spec, browser vendors are constantly updating to that spec, and users may drop support for earlier browsers. At first glance, there doesn’t seem to be a fixed target for what we should compile our JavaScript down to.



> The compat-table is updated constantly and is used for this preset.

This is where `babel-preset-env` comes in: it’s a Babel preset that automatically determines the correct Babel plugins to use based on the provided environments.

Export with various targets

```
export class A {}
```

Target only Chrome 52

.babelrc

```
{
  "presets": [
    ["env", {
      "targets": {
        "chrome": 52
      }
    }]
  ]
}
```

Out

```
class A {}
exports.A = A;
```

Target Chrome 52 with webpack 2/rollup a

.babelrc

```
{
  "presets": [
    ["env", {
      "targets": {
        "chrome": 52
      },
      "modules": false,
      "loose": true
    }]
  ]
}
```

Out

```
export class A {}
```

Target specific browsers via browserslist

.babelrc

```
{
  "presets": [
    ["env", {
      "targets": {
        "chrome": 52,
        "browsers": ["last 2 versions", "safari 7"]
      }}]
  ]
}
```

Out

```
export var A = function A() {
  _classCallCheck(this, A);
};
```

Target latest node via `node: true` or `node: "current"`

.babelrc

```
{
  "presets": [
    ["env", {
      "targets": {
        "node": "current"
      }
    }]
  ]
}
```

Out

```
class A {}
exports.A = A;
```

Dan Abramov @dan_abramov · 11 Dec 2016

Now, this is a configuration that makes sense to me. Specify browsers once, let Babel and Autoprefixer do the work. github.com/facebookincuba…

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test"
},
"browsers": {
  "development": [
    "last 1 chrome version"
  ],
  "production": [
    ">1%",
    "last 4 versions",
    "Firefox ESR",
    "not ie < 9"
  ]
```

```
s App extends _react.Component {
nder() {
return _react2.default.createElement(
  'div',
  { className: 'App', __source: {
      fileName: _jsxFileName,
      lineNumber: 8
    },
    __self: this
  },
  _react2.default.createElement(
    'div',
    { className: 'App-header', __sourc
      fileName: _jsxFileName,
      lineNumber: 9
    },
    __self: this
  },
(t.__proto__||Object.getProtot
,arguments))}return a(t,e),s(t
[{key:"render",value:function(
l.default.createElement("div",
{className:"App"},l.default.cr
{className:"App-
header"},l.default.createEleme
{src:p.default,className:"App-
logo",alt:"logo"}),l.default.c
ll,"Welcome to
React")),l.default.createEleme
{className:"App-intro"},"To ge
",l.default.createElement("cod
)," and save to reload."))}}])
(u.Component);t.default=d},fun
strict":function r(e){return e
```

11 152 463

actual.js

```
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13 Array.from; // static function
14 Map; // top level built-in
15
16 // instance methods may have false positives (which is ok)
17 a.includes(); // method call
18 b['find'] // computed string?
19 c.prototype.findIndex(); // .prototype
20 d.fill.bind(); //.bind
21 e.padStart.apply(); // .apply
22 f.padEnd.call(); // .call
23 String.prototype.startsWith.call; // prototype.call
24 var { codePointAt, endsWith } = k; // destructuring
25
```

expected.js

```
 1 import 'core-js/modules/es6.array.from';
 2 import 'core-js/modules/es6.map';
 3 import 'core-js/modules/es7.array.includes';
 4 import 'core-js/modules/es6.string.includes';
 5 import 'core-js/modules/es6.array.find';
 6 import 'core-js/modules/es6.array.find-index';
 7 import 'core-js/modules/es6.array.fill';
 8 import 'core-js/modules/es7.string.pad-start';
 9 import 'core-js/modules/es7.string.pad-end';
10 import 'core-js/modules/es6.string.starts-with';
11 import 'core-js/modules/es6.string.code-point-at';
12 import 'core-js/modules/es6.string.ends-with';
13 Array.from; // static function
14 Map; // top level built-in
15
16 // instance methods may have false positives (which is ok)
17 a.includes(); // method call
18 b['find']; // computed string?
19 c.prototype.findIndex(); // .prototype
20 d.fill.bind(); //.bind
21 e.padStart.apply(); // .apply
22 f.padEnd.call(); // .call
23 String.prototype.startsWith.call; // prototype.call
24 var _k = k,
25     codePointAt = _k.codePointAt,
26     endsWith = _k.endsWith; // destructuring
```

Henry Zhu @left_pad · Mar 31

WIP PR to do "dead-code elim" of polyfills (it only adds used ones that are not supported in your target env) github.com/babel/b

3 21 116

Let’s work together!

Comment on issues, Slack chat

Null Propagation Operator: Stage-1 #328

Open **hzoo** opened this issue on Jan 27 · 6 comments

hzoo Henry Zhu - commented on Jan 27 • edited 26 19 in this repo **Owner**

Stage: 1, Gabriel Isenberg

Spec Repo: https://github.com/claudepache/es-optional-chaining
TC39 Proposals: https://github.com/tc39/proposals
Slides:
https://docs.google.com/presentation/d/11O_wIBBbZgE1bMVRJI8kGnmC6dWCBOwutbN9SWOK0fU/edit#slide=id.p

ESTree Issue to discuss AST Spec: estree/estree#146

cc **@gisenberg** if you can implement or work with someone to do so. (Please join our slack if you haven't http://slack.babeljs.io/)

Test it out: http://babeljs.io/repl

Report issues!

Henry Zhu @left_pad · Jan 11

Thanks to @_gsathya for reporting all these @babeljs while working on the v8 implementation, love it! (github.com/babel/babel/is… is one)

hzoo Henry Zhu - commented a day ago • edited — Member

Thanks for the report! I think this should be covered by https://github.com/babel/babel/pull/3675/files but haven't reviewed all of it yet. Are you implementing it in v8 now 😄 ?

gsathya Sathya - commented a day ago

> Are you implementing it in v8 now ?

Yeah, I'm sanity checking my implementation with babel 😄

🎉 1

Implement Parser support!

Add support for invalid escapes in tagged templates #274

Merged **hzoo** merged 1 commit into `babel:master` from `bakkot:template-literal-revision` on Mar 21

Conversation 21 | Commits 1 | Files changed 290

bakkot Kevin Gibbons - commented on Jan 9 • edited

First PR out of 5 1 in this repo | Member

Per the stage-3 TC39 proposal.

Q	A
Bug fix?	no
Breaking change?	maybe?
New feature?	yes
Deprecations?	no
Spec compliancy?	yes
Tests added/pass?	~~no/yes resp.~~ Edit: yes/yes.
Fixed tickets	babel/babel#4798 ~~(progress, not fixed)~~
License	MIT

In particular, it allows Babylon to parse `tag`\u{invalid}`` and similar.

Review
xtu
dar
Assigne
No one
Labels
enhar
spec-
Project
None ye
Milesto
7.0.0

Help review PRs!

3 − z

3 + z

And... v6.11.3 of Babylon as well! (mostly for spec fixes for object rest) @sebmarkbage github.com/babel/babylon/ …

ljharb 5 da

I'm not sur

The only ti

getters/se

👍 1

hzoo 5 da

How do we

sebmarkb

It should n

according

It's n

(don

SyntaxError: The rest element has to be the last element when destructuring (1:10)

```
> 1 | let { ...x, y, z } = { x: 1, y: 2, z: 3};
    |           ^
# Previous behavior:
# x = { x: 1, y: 2, z: 3 }
# y = 2
# z = 3
```

Before, this was just a more verbose way of shallow copying `obj` since it doesn't actually do what you think.

SyntaxError: Cannot have multiple rest elements when destructuring (1:13)

```
> 1 | let { x, ...y, ...z } = { x: 1, y: 2, z: 3};
    |               ^
# Previous behavior:
# x = 1
# y = { y: 2, z: 3 }
# z = { y: 2, z: 3 }
```

Before y and z would just be the same value anyway so there is no reason to need to have both.

SyntaxError: A trailing comma is not permitted after the rest element (1:16)

```
let { x, y, ...z, } = obj;
```

The rationale for this is that the use case for trailing comma is that you can add something at the end without affecting the line above. Since a RestProperty always has to be the last property it doesn't make sense.

} object containe

ses?

setters because

rty .

arly

Henry Zhu

@left_pad

Create Proposal Plugins!

Implement support for async generator functions and for-await statements #3473

Closed **zenparsing** wants to merge 1 commit into `babel:master` from `zenparsing:async-generator-functions`

Conversation 27 | Commits 1 | Files changed 45

zenparsing commented on Apr 18, 2016 — First PR 0 in this repo — Contributor

This PR depends on babel/babylon@ `b926e40`. Tests will fail until the next babylon release, but submitting early for feedback.

This change implements the async iteration proposal, currently at stage 2. It includes the following features:

- Transforms async generator functions (`async function* g() { }`) to wrapped generator functions, similar to the current async-to-generator transform.
- Transforms `for-await` statements into for loops containing yield expressions.

Thanks!

- Plugin: https://github.com/mathiasbynens/babel-plugin-transform-unicode-property-regex
- PR with docs: babel/babel.github.io#904

```
+    toString() { return `Point<${ this.#x },${ this.#y
+}
```

Reviewer

No review

Assignee

hzoo

Labels

es-prop

tag: ne

None

Henry Zhu
@left_pad

Create Proposal Plugins!

Thanks to @diervo for the PR to @babeljs to add parser support for Private Fields (and @littledan for review)!

rreiro Val -
)ec 18, 2016 •

First PR ⓘ 0 in this repo

port Private Fields
ımmar changes according to the spec:

w (optional) plugin `classPrivateProperties`
ıteName type identifier
sPrivateProperty to ClassBody
eName in MemberExpression
eName as a references

ıaming and small code tweaks towards a future refact

```
+class Point {
+    #x;
+    #y;
+
+    constructor(x = 0, y = 0) {
+        #x = +x;
+        #y = +y;
+    }
+
+    get x() { return this.#x }
+    set x(value) { this.#x = +value }
+
+    get y() { return this.#y }
+    set y(value) { this.#y = +value }
+
+    equals(p) { return this.#x === p.#x && this.#y === p
+
+    toString() { return `Point<${ this.#x },${ this.#y
+}
```

Some Questions

What should Babel be doing differently?

The Babel project has made it incredibly easy to write code using experimental JavaScript features. All it takes is the installation of two modules, and you can be writing `do` expressions right alongside standard syntax like `switch` statements and `for..of` loops. This amazingly low barrier has prompted many developers to adopt early-stage features across the board–from their one-off experiments to their open source libraries, to the applications that drive their businesses.

So while the committee may make recommendations about when and how to use new constructs, for many developers, the only question is, "Is a plugin available on npm?" I'm reminded of a recent JavaScript users group meeting here in Boston. The presenter asked, "does anyone know which features were introduced in ES2016?"

"Function decorators," came a reply from the audience.

"Actually, that's not part of ES2016. Even it's inclusion in ES2017 is debatable."

"Oh, *2016*. That introduced destructuring assignment."

"Not quite–destructuring binding was standardized in 2015."

You might think I'm being a little academic here. Maybe it seems snooty of me to expect others to keep track of such technicalities… But downplaying the relevance of the proposals' "stages" has two real dangers.

> Dave Herman: Design constraints: – It needs to be possible to import named exports from CJS – [using the `require` function to load an ECMASCript module] needs to [return] synchronously
>
> Jeff Morrison: Are these technical needs or ecosystem needs?
>
> James Snell: These are ecosystem needs. Babel today can do these things. Those users will want to be able to not change their code. If we say that doesn't work, we're violating a concern.

This demonstrates how user expectations push the committee to make difficult decisions. The more eagerly we build and deploy systems on proposed extensions, the more difficult it becomes for standards bodies to amend the design. Remember: it's not "done" until stage 4! In extreme cases, this could lead to final designs that include sub-optimal aspects informed by "web reality." That's not a theoretic concern, either. Already, the specification devotes an entire section to the various irregularities that came about in this way.

We should be communicating more?

- attending TC39 and/or
  - appointing someone (Jordan/ljharb)?
- how to communicate changes and interop

Proposal: TC39 champions get involved more in Babel by helping review/create/maintain the plugin proposals

Mike North @ 🇪🇸 @michaellnorth · May 4

🐠 Which @babeljs preset do you use for your apps?

22% 👵 stage 4

20% 👨‍💼 stage 3 ✓

25% 👨‍🎓 stage 2

33% 👶 stage 0/1

440 votes • Final results

James Holmes
@James_R_Holmes

Follow

When you accidentally open the babel transpiled file rather than the source

RETWEETS 13 LIKES 37

9:49 PM - 23 May 2017

Lack of developer awareness of TC39 itself: repo, committee, staging process, reading spec text, test262

sMyle @MylesBorins · May 16

I think Babel offers a low barrier for entry in trying out these new features.

Not everyone is equipped to read specs

1 3 10

Create a proposals table in Babel

Active proposals

Proposals follow this process document. This list contains only stage 1 proposals and higher that have not yet been withdrawn/rejected, or become finished.

🚀	Proposal	Champion	Stage
	SIMD.JS - SIMD APIs	John McCutchan, Peter Jensen, Dan Gohman, Daniel Ehrenberg	3
	`Function.prototype.toString` revision	Michael Ficarra	3
	`global`	Jordan Harband	3
	Rest/Spread Properties	Sebastian Markbage	3
	Asynchronous Iteration	Domenic Denicola	3
	`import()`	Domenic Denicola	3

Status on “champion” for the plugin itself, link

analytics of proposals?

John-David Dalton
@jdalton

Following

LazyDev: BigQuery GitHub results showing the use of Babel.

RETWEET	LIKES
1	5

7:02 PM - 5 Apr 2017

How can we work together to balance: getting early stage feedback from many users while allowing TC39 to feel comfortable with the best design (breaking changes, etc)?

Rename stage presets to indicate the danger of using them #4914

Closed **kentcdodds** opened this issue on Nov 30, 2016 · 8 comments

kentcdodds Kent C. Dodds - commented on Nov 30, 2016

0 8 in this repo · Member

As noted here.

> **@kentcdodds**: What if instead of `stage-0`, `stage-1` we called them `super-dangerous` `pretty-dangerous`, etc :)
> **@DrewML**: Honestly, I personally would love that. You have to acknowledge what you're doing everytime you open up `package.json` / `.babelrc`

I'm a fan of this idea. Though I think that people might not like this change much. Thoughts?

👍 4

ljharb Jordan Harband - commented on Dec 2, 2016

Member

Stage 0 is “i’ve got a crazy idea”, stage 1 is “this idea might not be stupid”, stage 2 is “let’s use polyfills and transpilers to play with it, and stage 3 is “let’s let browsers implement it and see how it goes", and stage 4 is "now it's javascript".

Coming up with more accurate names that strongly discourage relying on features in production too early would be great. At Airbnb, we only permit using stage 3 features or higher, because of the high risk of breaking change (or total withdrawal) of stage 2-and-below features.

Assign
No one
Labels
discu
Project
None y
Milesto
No mile
Notifica
5 parti

Revamping stage plugins (Removing the stage-x presets) #4955

Open **thejameskyle** opened this issue on Dec 7, 2016 · 22 comments

thejameskyle James Kyle - commented on Dec 7, 2016

33 49 in this repo · Owner

So there's a problem with how stage-x plugins are designed right now. They are versioned in lockstep with the rest of Babel. However these stage-x features change throughout the process and may make breaking changes at any time.

Babel's lockstep versioning causes a weird tradeoff of either following SemVer or keeping up with the spec. At the same time Lockstep versioning is important for the ecosystem.

I've talked to contributors privately about this and figured I'd create an issue:

I want to get rid of the `babel-preset-stage-x` plugins and move all of the stage-x plugins into their own monorepo with independent (see Lerna) versioning.

This way there are no more presets for stages and people have to manually enable plugins by individual features. These plugins can then make major version bumps at any time.

The plugins should be named `babel-plugin-transform-proposal-{feature}`, when they are accepted by TC39 they move into the main repository and are named `babel-plugin-transform-es{year}-{feature}` and included in the relevant presets.

The one problem that needs to be solved is parsing. We need to be able to change the syntax of a proposal at any time so that we can track changes like decorators moving from `@` to `#` (which was turned down by the committee but we need a better way in Babel to handle that type of scenario).

Babylon Versioning #275

Versioning?

Open **hzoo** opened this issue on Jan 9 · 0 comments

hzoo Henry Zhu - commented on Jan 9 — 26 19 in this repo — Owner

Related:

- babel/babel#4914
- babel/babel#4955

Right now we are also trying to figure out babylon/babel versioning. Currently babel depends on the parser so if any proposal feature needs a spec update (would be a bug fix, but also breaking) we have to major bump babylon and thus babel. I think what we are thinking of doing is making each spec update a "separate" plugin name.?

```
{
  "plugins": [
    "template-literal-revision-stage-2",
    "template-literal-revision-stage-3", // or
    "template-literal-revision-sha-git-sha-here", // or
    "template-literal-revision-sha-tc39-meeting-date-here", // or
  ]
}
```

Proposal plugin versioning ideas

- Independently semver versioned (from the rest of Babel)
- Start at 0.x, instead of 1.x
- Stage 0 is 0.0.x, Stage 1 is 0.1.x, etc?
- Just make a major version at each breaking change

Incentives to try new proposals

- May not be a real concern 🙂
- Ease of use: `npm install babel-plugin-x`, add to config
- Create a codemod to convert back to previous syntax if the proposal might be or is dropped

How do we incentivize people to update to a more stable version?

- Proposal plugins are already opt-in
- Add a release channel flag: unstable/stable?
- Deprecation warning on out of date versions once a new Stage/patch is out
- Create a codemod to automatically convert users from the out of date syntax to the newer syntax (e.g. if decorators changed from `@` to another symbol)
- Write a bot to PR all GitHub projects to update

Help?

- Only a handful of contributors to maintain this huge part of the JavaScript ecosystem in their free time
- Still volunteer, community-driven
- No major organization/companies contributing back the project

Thanks!

Let’s work together!