

# Babel is a JavaScript compiler.

Use next generation JavaScript, today.

Put in next-gen JavaScript

```
[1, 2, 3].map(n => n ** 2);
```

Get browser-compatible JavaScript out

```
[1, 2, 3].map(function (n) {  
  return Math.pow(n, 2);  
});
```

Role of Babel in JS

Henry Zhu

- - Babel Overview
- - Why is Babel relevant
- - Open Questions

# 6to5 -> Babel

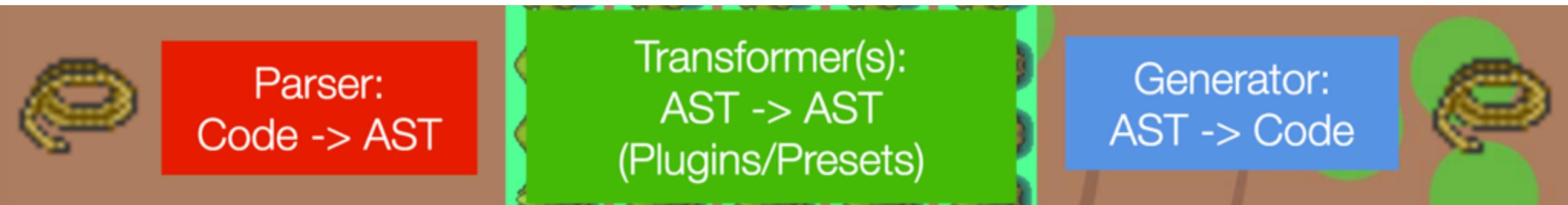
Sept 2014

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

 <b>babel / babel</b>	<a href="#">Code</a>	<a href="#">Issues</a>	<a href="#">Releases</a> 542	<a href="#">Edit</a>	<a href="#">babel-plugin-transform-fl...</a> Flow comments import export (#5675) 4 days ago
					<a href="#">babel-plugin-transform-fl...</a> Add support for object type spread (#5525) a month ago
					<a href="#">babel-plugin-transform-fu...</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-js...</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-o...</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-o...</a> Fix incorrect property ordering with obj rest spread on nested (#5... 22 days ago
					<a href="#">babel-plugin-transform-o...</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-p...</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-re..</a> Fix multiple var declarations in transform-react-constant-elements.. 2 days ago
					<a href="#">babel-plugin-transform-re..</a> Updated transform-react-display-name for createReactClass addo.. 23 days ago
					<a href="#">babel-plugin-transform-re..</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-re..</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-re..</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-re..</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-re..</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-re..</a> Typecheck much more of the config loading process (#5642) a month ago
					<a href="#">babel-plugin-transform-re..</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-ru..</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-plugin-transform-st..</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-polyfill</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-preset-es2015</a> Add test cases for bad options in babel-preset-es2015 (#5720) 10 days ago
					<a href="#">babel-preset-es2016</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-preset-es2017</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-preset-flow</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-preset-react</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-preset-stage-0</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-preset-stage-1</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-preset-stage-2</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-preset-stage-3</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-register</a> add .mjs to list of well known extensions 23 days ago
					<a href="#">babel-runtime</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-template</a> v7.0.0-alpha.9 a month ago
					<a href="#">babel-traverse</a> Fix multiple var declarations in transform-react-constant-elements.. 2 days ago
					<a href="#">babel-types</a> Fix ObjectProperty patterns (#5762) 2 days ago
					<a href="#">README.md</a> 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

① 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])/) );
```

## 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](#).

npm v6.17.1 travis [passing](#)

- The latest ECMAScript version enabled by default (ES2017).
- Comment attachment.
- Support for JSX and Flow. **typescript is in progress!**
- Support for experimental language proposals (accepting PRs for anything at least [stage-0](#)).

## 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 our 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/

The screenshot shows the AST Explorer interface at <http://astexplorer.net/>. The parser used is babylon6-6.16.1. The input code is `1 a ** b`. The parse tree is displayed in the JSON tab, with the following structure:

```
Tree JSON
✓ Autofocus ✓ Hide methods ✓ Hide empty keys ✓ Hide location data
- File {
    type: "File"
    - program: Program {
        type: "Program"
        sourceType: "module"
        - body: [
            - ExpressionStatement {
                type: "ExpressionStatement"
                - expression: BinaryExpression {
                    type: "BinaryExpression"
                    + left: Identifier {type, name}
                    operator: "***"
                    - right: Identifier {
                        type: "Identifier"
                        name: "b"
                    }
                }
            }
        ]
        directives: []
    }
    comments: []
    - tokens: [
        + Token (name) {type, value}
        + Token (**) {type, value}
        + Token (name) {type, value}
        + Token (eof) {type}
    ]
}
```

The node for the identifier 'b' is highlighted with a yellow background.

AST Explorer

Snippet



JavaScript

&lt;/&gt; babylon6



Transform



Tree

JSON

 Autofocus Hide methods Hide empty key

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

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]
  ));
}

24
25
26
27
28 }
```

Plugin Output

## Insertion, Removal, Replacement

path.insertBefore(node);	\$(el).after(node);
path.insertAfter(node);	\$(el).before(node);
path.unshiftContainer(node);	
path.pushContainer(node);	
path.remove();	\$(el).remove();
path.replaceWith(node);	\$(el).replaceWith(node);
path.replaceWithMultiple(nodes);	\$(el).replaceAll(nodes);

DOM API

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

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**



**NETFLIX**

**mozilla**

**YAHOO!**



**zendesk**

**vimeo**

**npm**



analytics?



**John-David Dalton**

@jdalton

Following



LazyDev: BigQuery GitHub results showing the use of Babel.

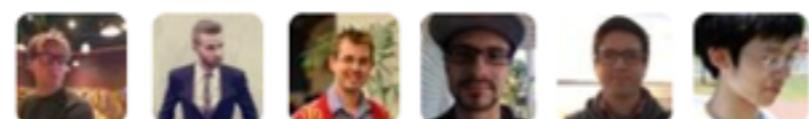
---

RETWEET

1

LIKES

5



7:02 PM - 5 Apr 2017

# Community Plugins



babel-plugin-

## Search Criteria

- Best Overall
- Quality
- Popularity
- Maintenance

**1630 packages found for "babel-plugin-"**

**babel-plugin-webpack-alias** adriantoine  
babel 6 plugin which allows to use webpack aliases

v2.1.2



Guillermo Rauch ✅

@rauchg

Following

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



zeit.co/oss

#### Babel transformation output

```
{ "plugins": ["styled-jsx/babel"] }
```

```
export default () => (
  <div>
    <p>only this gets styled!</p>
    <Component /> {/* children <p> won't be */}
    <style jsx>{
      p {
        color: red;
      }
    }</style>
  </div>
)
```

```
import _JSXStyle from 'styled-jsx/style'
export default () => (
  <div data-jsx="2nf2nf">
    <p data-jsx="2nf2nf">only this gets styled!</p>
    <Component /> {/* children <p> won't be */}
    <_JSXStyle css={`<p>`}>
      p[data-jsx=2nf2nf] {
        color: red;
      }
    </_JSXStyle>
  </div>
)
```

Full CSS support

Animations, media queries, pseudos...

Complete scoping / isolation

Components don't leak styles by default

One <style> per component

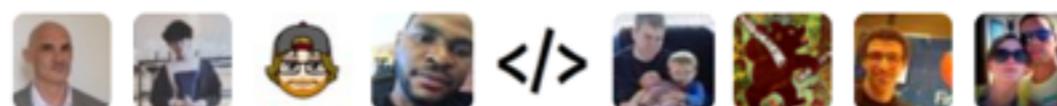
Styles are never needlessly duplicated

Automatic garbage collection

Unused CSS gets removed upon unmounting

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 compatibility 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
```



```
function W(...args) {
  if (!args.length) {
    return W();
  }
  if (args.length === 1) {
    return W(args[0]);
  }
  return args.map(function (arg) {
    var arr = [];
    arr.push(arg);
    arr.push([]);
    arr.push([]);
    arr.push(arg.length);
    arr.push(arg.nodeType ? [arg] : [], []);
    return arr;
  });
}

// Get initial elements from seed or context.
var arr = (args || []).map((arg) => {
  if (arg.nodeType) {
    return [arg];
  }
  return [];
});

// Prefilter to get matcher input, preserving a map for seed-results synchronization.
arr = arr.map((arg) => {
  if (arg.length === 1) {
    return arg[0];
  }
  return arg;
});

// If we have a postfilter, or filtered seed, or non-seed postfilter or prefilter
if (postfilter || filteredSeed || nonseedPostfilter || prefilter) {
  arr = arr.map((arg) => {
    if (postfilter) {
      arg = postfilter(arg);
    } else if (filteredSeed) {
      arg = filteredSeed[arg];
    } else if (nonseedPostfilter) {
      arg = nonseedPostfilter(arg);
    } else if (prefilter) {
      arg = prefilter(arg);
    }
    return arg;
  });
}

// ... intermediate processing is necessary
```



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

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**

@adduosmani

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    LIKES  
2              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.

Feature name	Current browser	Compilers/polyfills																															
		Traceur	Babel + core-js <sup>[2]</sup>	ES6 Transpiler	Closure	JSX <sup>[3]</sup>	Type-Script + core-js	es6-shim	IE 10	IE 11	Edge 12 <sup>[4]</sup>	Edge 13 <sup>[4]</sup>	Edge 14 <sup>[4]</sup>	FF 31 ESR	FF 38 ESR	FF 40	FF 41	FF 42	FF 43	FF 44	FF 45 ESR	FF 46	FF 47	FF 48	FF 49	FF 50 Beta	FF 51 Aurora	FF 52 Nightly	CH <19	CH 39, OP 26 <sup>[1]</sup>	CH 40, OP 27 <sup>[1]</sup>	CH 41, OP 28 <sup>[1]</sup>	
Optimization	proper tail calls (tail call optimisation)	2/2	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	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	7/7	4/7	4/7	4/7	0/7	5/7	0/7	0/7	0/7	0/7	0/7	0/7	3/7	3/7	3/7	3/7	3/7	4/7	4/7	4/7	4/7	4/7	4/7	4/7	4/7	6/7	6/7	0/7	0/7	0/7		
	rest parameters	5/5	3/5	2/5	2/5	3/5	4/5	0/5	0/5	5/5	5/5	5/5	5/5	3/5	4/5	4/5	4/5	4/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	0/5	0/5	0/5		
	spread (...) operator	15/15	13/15	8/15	12/15	2/15	4/15	0/15	0/15	12/15	15/15	15/15	15/15	11/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	0/15	0/15	0/15	0/15
	object literal extensions	6/6	6/6	6/6	4/6	5/6	6/6	0/6	0/6	6/6	6/6	6/6	6/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	0/6	0/6	0/6	0/6	
	for...of loops	9/9	9/9	4/9	6/9	2/9	3/9	0/9	0/9	6/9	7/9	7/9	7/9	5/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	
	octal and binary literals	4/4	2/4	4/4	4/4	0/4	4/4	2/4	0/4	0/4	4/4	4/4	4/4	2/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	0/4	0/4	0/4	4/4	
	template literals	5/5	4/5	3/5	3/5	4/5	3/5	0/5	0/5	4/5	5/5	5/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	0/5	0/5	5/5		
	RegEx 'Y' and 'U' flags	5/5	3/5	0/5	0/5	0/5	0/5	0/5	0/5	2/5	5/5	5/5	2/5	2/5	2/5	2/5	2/5	2/5	2/5	2/5	2/5	2/5	2/5	2/5	2/5	2/5	0/5	0/5	0/5	0/5			
	destructuring declarations	22/22	21/22	14/22	18/22	12/22	15/22	0/22	0/22	0/22	0/22	21/22	13/22	19/22	19/22	19/22	19/22	19/22	19/22	19/22	19/22	19/22	19/22	19/22	19/22	19/22	21/22	21/22	21/22	0/22	0/22	0/22	
	destructuring assignments	24/24	24/24	17/24	16/24	11/24	19/24	0/24	0/24	0/24	0/24	23/24	14/24	20/24	20/24	21/24	21/24	21/24	21/24	21/24	21/24	21/24	21/24	21/24	21/24	23/24	23/24	23/24	0/24	0/24	0/24		
	destructuring parameters	23/23	20/23	15/23	17/23	12/23	15/23	0/23	0/23	0/23	22/23	12/23	18/23	18/23	18/23	18/23	18/23	18/23	18/23	18/23	18/23	18/23	18/23	18/23	19/23	19/23	20/23	0/23	0/23	0/23			
	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	2/2	0/2	0/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0/2	0/2	0/2			
	new.target	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	1/2	2/2	2/2	0/2	0/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	0/2	0/2	0/2	0/2				
Bindings	const	16/16	14/16	10/16	14/16	0/16	14/16	0/16	0/16	12/16	12/16	12/16	16/16	3/16	12/16	12/16	12/16	12/16	12/16	12/16	12/16	12/16	12/16	12/16	12/16	12/16	16/16	16/16	1/16	1/16	1/16		
	let	12/12	10/12	8/12	10/12	0/12	10/12	0/12	0/12	10/12	10/12	10/12	12/12	0/12	0/12	0/12	0/12	0/12	0/12	10/12	10/12	10/12	10/12	10/12	10/12	12/12	12/12	0/12	0/12	6/12			
	block-level function declaration <sup>[13]</sup>	Yes	Yes	Yes	No	Yes	No	No	No	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	No	Flag	Flag	Yes			
Functions	arrow functions	13/13	11/13	9/13	8/13	10/13	8/13	9/13	0/13	0/13	0/13	8/13	13/13	13/13	8/13	8/13	9/13	10/13	10/13	11/13	11/13	13/13	13/13	13/13	13/13	0/13	0/13	0/13	0/13				
	class	24/24	17/24	19/24	17/24	13/24	16/24	19/24	0/24	0/24	0/24	0/24	24/24	24/24	0/24	0/24	0/24	0/24	0/24	0/24	24/24	24/24	24/24	24/24	24/24	24/24	0/24	0/24	0/24	0/24			
	super	8/8	7/8	4/8	7/8	5/8	7/8	7/8	0/8	0/8	0/8	0/8	8/8	8/8	0/8	0/8	0/8	0/8	0/8	0/8	8/8	8/8	8/8	8/8	8/8	8/8	0/8	0/8	0/8	0/8			
	generators	27/27	24/27	24/27	0/27	16/27	0/27	0/27	0/27	0/27	0/27	27/27	27/27	14/27	20/27	20/27	20/27	20/27	20/27	21/27	21/27	25/27	25/27	25/27	25/27	25/27	25/27	0/27	16/27	16/27	16/27		

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/facebookincub...](https://github.com/facebookincubator/create-react-app)

```
"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 {  
  render() {  
    return _react2.default.createElement(  
      'div',  
      { className: 'App', __source: {  
        fileName: _jsxFileName,  
        lineNumber: 8  
      },  
      __self: this  
    },  
    _react2.default.createElement(  
      'div',  
      { className: 'App-header', __source: {  
        fileName: _jsxFileName,  
        lineNumber: 9  
      },  
      __self: this  
    ),  
    -----  
    (t.__proto__||Object.getPrototypeOf(t), arguments))}return a(t,e),s(t,[{key:"render",value:function(){l.default.createElement("div", {className:"App"}),l.default.createElement("div", {className:"App-  
header"}),l.default.createElement("img", {src:p.default,className:"App-  
logo"}, alt:"logo")),l.default.createElement("h1", "Welcome to React"),l.default.createElement("p", "Let's get started!"),l.default.createElement("pre", "Copy this code and save to reload."))})  
}(u.Component);t.default=d},function r(e){return e
```

```
actual.js                                expected.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
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.startsWith';
11 import 'core-js/modules/es6.string.code-point-at';
12 import 'core-js/modules/es6.string.endsWith';
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/babel/pull/10000](https://github.com/babel/babel/pull/10000)

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](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/>)

# 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 😊



# 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



Review



xtu



dar

Assigned

No one

Labels

enhancement

specification

Project

None yet

Milestone

7.0.0

Per [the stage-3 TC39 proposal](#).

Q	A
Bug fix?	no
Breaking change?	maybe?
New feature?	yes
Deprecations?	no
Spec compliance?	yes
Tests added/pass?	<del>no/yes resp.</del> Edit: yes/yes.
Fixed tickets	<a href="#">babel/babel#4798</a> {progress, not fixed}
License	MIT

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

# Help review PRs!

3

- z

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



**ljharb** 5 da

I'm not sur

The only ti  
getters/set



1



**hzoo** 5 day

How do we



**sebmarkb**

It should n  
according

It's r  
(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



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



Reviewer

No review

Assignee



Labels

es-pro

tag: ne

None

All

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* 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](https://github.com/babel/babel.github.io/pull/904)

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



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)!

treiro Val -



First PR ⓘ 0 in this repo ▾

Dec 18, 2016 •

port Private Fields

Summary changes according to the spec:

With optional plugin `classPrivateProperties`

`#iteName` type identifier

`sPrivateProperty` to `ClassBody`

`#Name` in `MemberExpression`

`#Name` as a `reference`

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

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

# Some Questions

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 its 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



**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

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
	<a href="#">SIMD.JS - SIMD APIs</a>	John McCutchan, Peter Jensen, Dan Gohman, Daniel Ehrenberg	3
	<a href="#">Function.prototype.toString revision</a>	Michael Ficarra	3
	<a href="#">global</a>	Jordan Harband	3
	<a href="#">Rest/Spread Properties</a>	Sebastian Markbage	3
	<a href="#">Asynchronous Iteration</a>	Domenic Denicola	3
	<a href="#">import()</a>	Domenic Denicola	3

Status on “champion” for the plugin itself, link

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 +



Assigned to

No one

Labels

discuss

Project

None yet

Milestones

No milestones

Notifications

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.

5 participants



# 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 1 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

Thanks!

Let's work together!