eslint / espree

An Esprima-compatible JavaScript parser

418 commits **6** branches **61** releases **31** contributors

Branch: **master** New pull request Find file

Clone or download

Calinou and **platinumazure** Docs: Fix some typos in the README (#386)
Latest commit `6bf2ebf` Jul 4, 2018

docs	New: Add u and y regex flags (refs #10)	Dec 14, 2014
lib	Breaking: remove experimentalObjectRestSpread option (#374)	Mar 30, 2018
tests	Build: add node 10 (#381)	Jun 9, 2018
tools	Update: create-test script (fixes #291) (#292)	Sep 19, 2016
.eslintrc	Breaking: Switch to Acorn (fixes #200)	Nov 30, 2015
.gitattributes	Chore: Adding .gitattributes file (#366)	Jan 25, 2018
.gitignore	Update: To support year in ecmaVersion number (fixes #300) (#301)	Oct 21, 2016
.npmrc	Chore: add .npmrc (#359)	Oct 31, 2017
.travis.yml	Build: add node 10 (#381)	Jun 9, 2018
CHANGELOG.md	Build: changelog update for 4.0.0	Jun 21, 2018
CONTRIBUTING.md	Chore: Remove jQuery copyright from header of each file	Apr 21, 2016
LICENSE	Docs: Update license copyright	Oct 31, 2016
Makefile.js	Chore: Remove jQuery copyright from header of each file	Apr 21, 2016
README.md	Docs: Fix some typos in the README (#386)	Jul 4, 2018
espree.js	Update: upgrade acorn to support two ES2019 syntax (#380)	Jun 4, 2018
package.json	4.0.0	Jun 21, 2018

README.md

Espree

Espree started out as a fork of Esprima v1.2.2, the last stable published released of Esprima before work on ECMAScript 6 began. Espree is now built on top of Acorn, which has a modular architecture that allows extension of core functionality. The goal of Espree is to produce output that is similar to Esprima with a similar API so that it can be used in place of Esprima.

Usage

Install:

```
npm i espree --save
```

And in your Node.js code:

```
var espree = require("espree");

var ast = espree.parse(code);
```

There is a second argument to `parse()` that allows you to specify various options:

```
var espree = require("espree");

// Optional second options argument with the following default settings
var ast = espree.parse(code, {

    // attach range information to each node
    range: false,

    // attach line/column location information to each node
    loc: false,

    // create a top-level comments array containing all comments
    comment: false,

    // attach comments to the closest relevant node as leadingComments and trailingComments
    attachComment: false,

    // create a top-level tokens array containing all tokens
    tokens: false,

    // Set to 3, 5 (default), 6, 7, 8, 9, or 10 to specify the version of ECMAScript syntax you want to
    // You can also set to 2015 (same as 6), 2016 (same as 7), 2017 (same as 8), 2018 (same as 9), or 20
    ecmaVersion: 5,

    // specify which type of script you're parsing ("script" or "module")
    sourceType: "script",

    // specify additional language features
    ecmaFeatures: {

        // enable JSX parsing
        jsx: false,

        // enable return in global scope
        globalReturn: false,

        // enable implied strict mode (if ecmaVersion >= 5)
        impliedStrict: false
    }
});
```

Esprima Compatibility Going Forward

The primary goal is to produce the exact same AST structure and tokens as Esprima, and that takes precedence over anything else. (The AST structure being the ESTree API with JSX extensions.) Separate from that, Espree may deviate from what Esprima outputs in terms of where and how comments are attached, as well as what additional information is available on AST nodes. That is to say, Espree may add more things to the AST nodes than Esprima does but the overall AST structure produced will be the same.

Espree may also deviate from Esprima in the interface it exposes.

Contributing

Issues and pull requests will be triaged and responded to as quickly as possible. We operate under the ESLint Contributor Guidelines, so please be sure to read them before contributing. If you're not sure where to dig in, check out the issues.

Espree is licensed under a permissive BSD 2-clause license.

Build Commands

- `npm test` - run all linting and tests
- `npm run lint` - run all linting
- `npm run browserify` - creates a version of Espree that is usable in a browser

Differences from Espree 2.x

- The `tokenize()` method does not use `ecmaFeatures`. Any string will be tokenized completely based on ECMAScript 6 semantics.
- Trailing whitespace no longer is counted as part of a node.
- `let` and `const` declarations are no longer parsed by default. You must opt-in by using an `ecmaVersion` newer than `5` or setting `sourceType` to `module`.
- The `esparse` and `esvalidate` binary scripts have been removed.
- There is no `tolerant` option. We will investigate adding this back in the future.

Known Incompatibilities

In an effort to help those wanting to transition from other parsers to Espree, the following is a list of noteworthy incompatibilities with other parsers. These are known differences that we do not intend to change.

Esprima 1.2.2

- Esprima counts trailing whitespace as part of each AST node while Espree does not. In Espree, the end of a node is where the last token occurs.
- Espree does not parse `let` and `const` declarations by default.
- Error messages returned for parsing errors are different.
- There are two addition properties on every node and token: `start` and `end`. These represent the same data as `range` and are used internally by Acorn.

Esprima 2.x

- Esprima 2.x uses a different comment attachment algorithm that results in some comments being added in different places than Espree. The algorithm Espree uses is the same one used in Esprima 1.2.2.

Frequently Asked Questions

Why another parser

ESLint had been relying on Esprima as its parser from the beginning. While that was fine when the JavaScript language was evolving slowly, the pace of development increased dramatically and Esprima had fallen behind. ESLint, like many other tools reliant on Esprima, has been stuck in using new JavaScript language features until Esprima updates, and that caused our users frustration.

We decided the only way for us to move forward was to create our own parser, bringing us inline with JSHint and JSLint, and allowing us to keep implementing new features as we need them. We chose to fork Esprima instead of starting from scratch in order to move as quickly as possible with a compatible API.

With Espree 2.0.0, we are no longer a fork of Esprima but rather a translation layer between Acorn and Esprima syntax. This allows us to put work back into a community-supported parser (Acorn) that is continuing to grow and evolve while maintaining an Esprima-compatible parser for those utilities still built on Esprima.

Have you tried working with Esprima?

Yes. Since the start of ESLint, we've regularly filed bugs and feature requests with Esprima and will continue to do so. However, there are some different philosophies around how the projects work that need to be worked through. The initial goal was to have Espree track Esprima and eventually merge the two back together, but we ultimately decided that building on top of Acorn was a better choice due to Acorn's plugin support.

Why don't you just use Acorn?

Acorn is a great JavaScript parser that produces an AST that is compatible with Esprima. Unfortunately, ESLint relies on more than just the AST to do its job. It relies on Esprima's tokens and comment attachment features to get a complete picture of the source code. We investigated switching to Acorn, but the inconsistencies between Esprima and Acorn created too much work for a project like ESLint.

We are building on top of Acorn, however, so that we can contribute back and help make Acorn even better.

What ECMAScript 6 features do you support?

All of them.

What ECMAScript 7/2016 features do you support?

There is only one ECMAScript 2016 syntax change: the exponentiation operator. Espree supports this.

What ECMAScript 2017 features do you support?

There are two ECMAScript 2017 syntax changes: `async` functions, and trailing commas in function declarations and calls. Espree supports both of them.

What ECMAScript 2018 features do you support?

There are seven ECMAScript 2018 syntax changes:

- Invalid escape sequences in tagged template literals
- Rest/spread properties
- Async iteration
- RegExp `s` flag
- RegExp named capture groups
- RegExp lookbehind assertions
- RegExp Unicode property escapes

Espree supports all of them.

What ECMAScript 2019 features do you support?

Because ECMAScript 2019 is still under development, we are implementing features as they are finalized. Currently, Espree supports:

- Optional `catch` binding
- JSON superset (`\u2028` and `\u2029` in string literals)

How do you determine which experimental features to support?

In general, we do not support experimental JavaScript features. We may make exceptions from time to time depending on the maturity of the features.