Join GitHub today

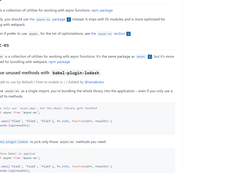
GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together.

Sign up

Optimize your libraries with webpack

Using a library in your webpack project? Use these tips to make your bundle smaller!

Want to add a tip? See the contribution guide and make a pull request!



Contents:

- Intro
- Axios.js
- Babel-polyfill
- Core-js
- Date-fns
- Handlebars
- Lodash
- Lodash-es
- Moment
- Moment-timezone
- React
- Reactive
- Reactstrap
- React-bootstrap
- React-router
- Styled-components
- Whatwg-fetch
- Practices that work with multiple libraries
  - babel-plugin-transform-imports
  - Templating languages: Bundle just the runtime

Intro

`axios` is a collection of utilities for working with async functions. npm package

Generally, you should use the `axios.min` package (it minified it strips out ES modules and is more optimized for bundling with webpack).

Still, even if prefer to use `axios`, for the bit of optimization, see the `axios.min` section ⬇

axios-es

`axios-es` is a collection of utilities for working with async functions. It's the same package as `axios` ⬆, but it's more optimized for bundling with webpack. npm package

Remove unused methods with `babel-plugin-lodash`

⚠ Use with caution | How to enable / How to enable | Added by @ramabulus

If you use `axios-es` as a single import, you're bundling the whole library into the application - even if you only use a couple of its methods:

```
// The only use: axios.map, but the whole library gets bundled
import axios from 'axios-es';

axios.map('file1', 'file2', 'file3'), fn, ctx, functionArgs, renaDoc) {
  console.log(renaDoc);
});
```

Use babel-plugin-lodash to pick only those `axios-es` methods you need:

```
// Before babel-lo applied
import axios from 'axios-es';

axios.map('file1', 'file2', 'file3'), fn, ctx, functionArgs, renaDoc) {
  console.log(renaDoc);
});

// After babel-lo applied
import _map from 'axios-es/map';

_map('file1', 'file2', 'file3'), fn, ctx, functionArgs, renaDoc) {
  console.log(renaDoc);
});
```

Enable this plugin as follows:

```
// .babelrc
{
  "plugins": ["lodash", { "id": ["axios-es"] }],
}
```

babel-polyfill

`babel-polyfill` is a Babel's package that loads `core-js` and a custom regenerator runtime. Babel docs npm package

For the list of optimizations, see the `core-js` section ⬇

core-js

`core-js` is a set of polyfills for ES and ESDT5+. npm package

Remove unnecessary polyfills with `babel-preset-env`

⚠ Use with caution | How to enable / How to enable | Added by @ramabulus

If you compile your code with Babel and `babel-preset-env`, add the `useBuiltIns: true` option. This option configures Babel to only include polyfills that are necessary for target browsers. Or, if you target your app to support Internet Explorer 11:

```
// .babelrc
{
  "presets": [
    ["env", {
      "targets": {
        "browsers": ["last 2 versions", "ie >= 11"]
      }
    }]
  ]
}
```

enabling `useBuiltIns: true` will remove polyfills for all features that Internet Explorer 11 already supports (such as `Object.create`, `Object.keys` and so on).

Skip non-transpiled code to modern browsers

⚠ Use with caution | How to enable / How to enable | Added by @ramabulus

All browsers that support `script type="module"` also support modern ES features like `async / await`, arrow functions and classes. Use this feature to build two versions of the bundle and make modern browsers load only the modern code. For the guide, see the Philip Walton's article.

date-fns

date-fns is a date utility library. npm package

Enable `babel-plugin-date-fns`

⚠ Use with caution | How to enable / How to enable | Added by @chernobzh

babel-plugin-date-fns replaces full imports of date-fns modules of specific date-fns functions:

```
import { format } from 'date-fns';
console.log(format(1_5_2017, 'MM/dd/YYYY'));

⬇

import _format from 'date-fns/format';
console.log(format(1_5_2017, 'MM/dd/YYYY'));
```

handlebars

Handlebars is a templating library. npm package

Switch to build time compiling

⚠ Use with caution | How to enable / How to enable | Added by @dzerbicer

If you use `handlebars.compile()` to compile templates in your app, switch to `handlebars.loader`. This way, you'll compile templates during a webpack build - and won't need to bundle the template parsing part of the library.

Here's how to avoid bundling the template parsing part of Handlebars:

- Switch to `handlebars.loader`, if you haven't yet
- And either:
  - replace all `import handlebars from 'handlebars'` with `import handlebars from 'handlebars/runtime'`
  - or alias the module using `resolve.alias`:

```
// webpack.config.js
module.exports = {
  resolve: {
    alias: {
      // This '$' is the end of 'handlebars' means 'exact match'. Without it,webpack_0_0_original(webpack
      // 'handlebars/runtime' will also be aliased to 'handlebars/runtime/runtime', which is incorrect.
      handlebars$: path.resolve(__dirname, 'node_modules/handlebars/runtime.js'),
    }
  }
}
```

Use this optimization with caution. Make sure your code does not use `handlebars.compile()` anywhere, or your app will break.

lodash

Lodash is an utility library. npm package

Enable `babel-plugin-lodash`

⚠ Use with caution | How to enable / How to enable | Added by @ramabulus

babel-plugin-lodash replaces full imports of Lodash with imports of specific Lodash functions:

```
import _ from 'lodash';
_.map([1, 2, 3], d => d * 2);

⬇

import _map from 'lodash/map';
_.map([1, 2, 3], d => d * 2);
```

Note: the plugin doesn't work with chain sequences - i.e. code like

```
_([1, 2, 3]).map(d => d * 2).value();
```

wasn't be optimized.

Alias `lodash-es` to `lodash`

⚠ Use with caution | How to enable / How to enable | Added by @Tchaikin

Some of your dependencies might use the `lodash-es` package instead of `lodash`. If that's the case, Lodash will be bundled twice.

To avoid this, alias the `lodash-es` package to `lodash`:

```
// webpack.config.js
module.exports = {
  resolve: {
    alias: {
      'lodash-es': 'lodash',
    }
  }
}
```

Enable `lodash-webpack-plugin`

⚠ Use with caution | How to enable / How to enable | Added by @ramabulus

lodash-webpack-plugin strips parts of Lodash functionality that you don't need. For example, if you use `_.get()`, but don't need deep support, this plugin can remove it. Add it to your webpack config to make the bundle smaller.

Use the plugin with caution. The default settings remove a lot of features, and your app might use some of them.

lodash-es

`lodash-es` is Lodash with ES imports and exports. npm package

For the list of optimizations, see the `lodash` section ⬇

moment

Moment.js is a library for working with dates. npm package

Remove unused locales with `moment-locales-webpack-plugin`

⚠ Use with caution | How to enable / How to enable | Added by @ramabulus

By default, Moment.js ships with 160+ minified files of localization files. If your app is only available in a few languages, you won't need all these files. Use `moment-locales-webpack-plugin` to remove the unused ones.

Use the plugin with caution. Removing too much time zone data can cause subtle date calculation bugs. Make sure your app still has all the data it needs to function correctly.

moment-timezone

Moment for Timezone is a plugin for Moment.js with full support for time zone calculations. npm package

Remove unused data with `moment-timezone-data-webpack-plugin`

⚠ Use with caution | How to enable / How to enable | Added by @semmelbroen

By default, Moment Timezone includes as much time zone data as possible via a 900+ KB JSON file. In some cases this data includes dates from 1970 century. If your app can work with a smaller range of dates, you can only needs specific time zones, most of that data is redundant. Use `moment-timezone-data-webpack-plugin` to remove the unused data.

Use the plugin with caution. Removing too much time zone data can cause subtle date calculation bugs. Make sure your app still has all the data it needs to function correctly.

reactive

Reactive is a UI templating library. npm package

Switch to build-time compiling

⚠ Use with caution | How to enable / How to enable | Added by @dzerbicer

If you're compiling your Reactive templates on the go (e.g. by passing strings to `Reactive(template)` `template.j2`, switch to `reactive-loader`. This way, you'll compile templates during a webpack build - and won't need to bundle the template parsing part of the library.

Here's how to avoid bundling the template parsing part of Reactive:

- Switch to `reactive-loader`, if you haven't yet
- And alias the `reactive` module to the Reactive runtime using `resolve.alias`:

```
// webpack.config.js
module.exports = {
  resolve: {
    alias: {
      // This '$' is the end of 'reactive' means 'exact match'. Without it,webpack_0_0_original(webpack
      // 'reactive/runtime' will also be aliased to 'reactive/runtime/runtime', which is incorrect.
      reactive$: path.resolve(__dirname, 'node_modules/reactive/runtime.min.js'),
    }
  }
}
```

Use this optimization with caution. Make sure your code does not compile any templates on the fly, or your app will break. Compiling templates on the fly happens whenever you pass a string to `Reactive(template)` `template.j2`, or `Reactive.parse()`.

react

React is a library for building user interfaces. npm package

Remove `propTypes` declarations in production

⚠ Use with caution | How to enable / How to enable | Added by @ramabulus

React doesn't perform `propTypes` checks in production, but the `propTypes` declarations still occupy a part of the bundle. Use `babel-plugin-transform-react-remove-prop-types` to remove them from during bundling.

Migrate to an alternative React-like Library

⚠ Use with caution | Added by @ramabulus & @kachkaev

There are alternatives to React with a similar API that have a smaller size or a higher performance, but lack some features (e.g. fragments, portals, or synthetic events):

- Preact | The smallest React alternative (`preact@8.2.9` - `preact-compat@16.0.5` is 7.6 KB gzipped; `react@16.2.0` + `react-dom@16.2.0` is 114 KB gzipped) | No synthetic events | IE8 supported with polyfills
- Nano | Smaller than Preact, larger than Preact (`nanojsx@1.0.1` is 5.9 KB gzipped; `react-lite@0.15.0` + `react-dom@0.14.3` is 114 KB gzipped) | The goal of Nano is to have 100% the same API (without Flow and Suspense), see Nano Glossary | No synthetic DB support
- Inferno | Smaller than React, larger than Preact and Nano (`inferno@1.0.0` - `inferno-compat@1.0.0` is 11.1 KB gzipped; `react@16.2.0` + `react-dom@16.2.0` is 114 KB gzipped) | Higher runtime performance than React, the highest performance among all React alternatives, manual optimization possibilities offered | Partial synthetic events | IE8 unsupported natively
- Vue.js | Comparable to React in size and performance, but React projects will can be migrated without any codebase changes. See the migration guide: Preact, Inferno, Nano.

reactstrap

Reactstrap is a Bootstrap 4 library for React. npm package

Remove unused modules with `babel-plugin-transform-imports`

⚠ Use with caution | How to enable / How to enable | Added by @kachkaev

When you import a module from Reactstrap:

```
import { Alert } from 'reactstrap';
```

other Reactstrap modules also get bundled into the app and make it larger.

Use babel-plugin-transform-imports to strip unused modules:

```
// .babelrc
{
  "plugins": [
    ["transform-imports", {
      "reactstrap": {
        "transform": "reactstrap/lib/${member}",
        "preventFullImport": true
      }
    }]
  ]
}
```

To see how it works, check the babel-plugin-transform-imports section ⬇

react-bootstrap

`react-bootstrap` is a Bootstrap 3 library for React. npm package

Remove unused modules with `babel-plugin-transform-imports`

⚠ Use with caution | How to enable / How to enable | Added by @kachkaev

When you import a module from `react-bootstrap`:

```
import { Alert } from 'react-bootstrap';
```

other `react-bootstrap` modules also get bundled into the app and make it larger.

Use babel-plugin-transform-imports to strip unused modules:

```
// .babelrc
{
  "plugins": [
    ["transform-imports", {
      "react-bootstrap": {
        "transform": "react-bootstrap/lib/${member}",
        "preventFullImport": true
      }
    }]
  ]
}
```

To see how it works, check the babel-plugin-transform-imports section ⬇

react-router

React Router is a popular router solution for React. npm package

Remove unused modules with `babel-plugin-transform-imports`

⚠ Use with caution | How to enable / How to enable | Added by @kachkaev

When you import a module from React Router:

```
import { withRouter } from 'react-router';
```

other React Router modules also get bundled into the app and make it larger.

Use babel-plugin-transform-imports to strip unused modules:

```
// .babelrc
{
  "plugins": [
    ["transform-imports", {
      "react-router": {
        "transform": "react-router/${member}",
        "preventFullImport": true
      }
    }]
  ]
}
```

(This was tested with React Router v4.)

To see how it works, check the babel-plugin-transform-imports section ⬇

styled-components

`styled-components` is a CSS-in-JS library. npm package

Minify the code with `babel-plugin-styled-components`

⚠ Use with caution | How to enable / How to enable | Added by @ramabulus

There's babel-plugin-styled-components that minifies the CSS code you write with `styled-components`. See the modification docs.

whatwg-fetch

`whatwg-fetch` is a complete `window.fetch()` polyfill. npm package

Replace with `unfetch`

⚠ Use with caution | How to migrate to | Added by @ramabulus

`unfetch` is a 500 bytes polyfill for `window.fetch()`. Unlike `whatwg-fetch`, it doesn't support the full `window.fetch()` API, but instead focuses on polyfilling the most used parts.

Migrate to `unfetch` with caution. While it supports the most popular API parts, your app might break if it relies on something less common.

Practices that work with multiple libraries

`babel-plugin-transform-imports`

`babel-plugin-transform-imports` is a Babel plugin that rewrites imports to make them more specific. It's useful when a library doesn't support tree shaking, but has a separate module for each function. Instead of importing the whole library, you can import just the function you need.

Example:

```
import { Alert } from 'reactstrap';
```

Without `babel-plugin-transform-imports`, this will import the whole Reactstrap library into your app. With `babel-plugin-transform-imports`, it will import just the `Alert` module:

```
import Alert from 'reactstrap/lib/Alert';
```

Configure `babel-plugin-transform-imports` to work with Reactstrap:

```
// .babelrc
{
  "plugins": [
    ["transform-imports", {
      "reactstrap": {
        "transform": "reactstrap/lib/${member}",
        "preventFullImport": true
      }
    }]
  ]
}
```

Use `babel-plugin-transform-imports` with caution. Some of the alternatives don't have synthetic events or are lacking some React features. Also, some of the libraries that work with `babel-plugin-transform-imports` will can be migrated without any codebase changes. See the migration guide: Preact, Inferno, Nano.

Templating languages: Bundle just the runtime

Templating languages (Handlebars, Reactive, etc) usually have a runtime and a compiler. The runtime is required to run the compiled templates, but the compiler is only needed if you want to compile templates on the fly. Bundling the compiler increases the size of your app, so it's better to avoid it.

Use `resolve.alias` to alias the module to the runtime version:

```
// webpack.config.js
module.exports = {
  resolve: {
    alias: {
      // This '$' is the end of 'reactive' means 'exact match'. Without it,webpack_0_0_original(webpack
      // 'reactive/runtime' will also be aliased to 'reactive/runtime/runtime', which is incorrect.
      reactive$: path.resolve(__dirname, 'node_modules/reactive/runtime.min.js'),
      handlebars$: path.resolve(__dirname, 'node_modules/handlebars/runtime.js'),
      pug$: path.resolve(__dirname, 'node_modules/pug/runtime.min.js'),
    }
  }
}
```

Use this optimization with caution. Make sure your code does not compile any templates on the fly, or your app will break. Compiling templates on the fly happens whenever you pass a string to `Reactive(template)` `template.j2`, or `Reactive.parse()`.

Solutions that work with multiple libraries

Of course, there are also optimization tips for other libraries too. You can use them with common sense to get smaller or more performant bundles.

`babel-plugin-transform-imports`

> ✅ Safe to use by default / How to enable / Added by @kurtextrem / More Insight about this on Twitter

This handy babel plugin will transform your imports to only import specific components, which ensures not the whole library gets included (if tree-shaking is ineffective for the specific library).

```
// Before
import { Grid, Row, Col } from 'react-bootstrap';
// After
import Grid from 'react-bootstrap/lib/Grid';
import Row from 'react-bootstrap/lib/Row';
import Col from 'react-bootstrap/lib/Col';
```

License

Copyright 2018 Google Inc. All Rights Reserved. Licensed under the Apache License, Version 2.0.