Supported Browsers and Features

Supported Browsers

By default, the generated project supports all modern browsers. Support for Internet Explorer 9, 10, and 11 requires polyfills. For a set of polyfills to support older browsers, use react-app-polyfill.

Supported Language Features

This project supports a superset of the latest JavaScript standard. In addition to ES6 syntax features, it also supports:

- Exponentiation Operator (ES2016).
- Async/await (ES2017).
- Object Rest/Spread Properties (ES2018).
- Dynamic import() (stage 4 proposal)
- Class Fields and Static Properties (part of stage 3 proposal).
- JSX, Flow and TypeScript.

Learn more about different proposal stages.

While we recommend using experimental proposals with some caution, Facebook heavily uses these features in the product code, so we intend to provide codemods if any of these proposals change in the future.

Note that **this project includes no polyfills** by default.

If you use any other ES6+ features that need **runtime support** (such as `Array.from()` or `Symbol`), make sure you are including the appropriate polyfills manually, or that the browsers you are targeting already support them.

Configuring Supported Browsers

By default, the generated project includes a `browserslist` configuration in your `package.json` file to target a broad range of browsers based on global usage (`> 0.2%`) for production builds, and modern browsers for development. This gives a good development experience, especially when using language features such as async/await, but still provides high compatibility with many browsers in production.

The `browserslist` configuration controls the outputted JavaScript so that the emitted code will be compatible with the browsers specified. The `production` list will be used when creating a production build by running the `build` script, and the `development` list will be used when running the `start` script. You can use https://browserl.ist to see the browsers supported by your configured `browserslist`.

Here is an example `browserslist` that is specified in `package.json`:

```
"browserslist": {
  "production": [
    ">0.2%",
    "not dead",
    "not op_mini all"
  ],
  "development": [
    "last 1 chrome version",
    "last 1 firefox version",
    "last 1 safari version"
  ]
}
```

> Note that this does not include polyfills automatically for you. You will still need to polyfill language features (see above) as needed based on the browsers you are supporting.
>
> When editing the `browserslist` config, you may notice that your changes don't get picked up right away. This is due to an issue in babel-loader not detecting the change in your `package.json`. A quick solution is to delete the `node_modules/.cache` folder and try again.