Supported Browsers and Features

Next.js supports **IE11 and all modern browsers** (Edge, Firefox, Chrome, Safari, Opera, et al) with no required configuration.

Polyfills

We transparently inject polyfills required for IE11 compatibility. In addition, we also inject widely used polyfills, including:

- **fetch()** — Replacing: `whatwg-fetch` and `unfetch`.
- **URL** — Replacing: the `url` package (Node.js API).
- **Object.assign()** — Replacing: `object-assign`, `object.assign`, and `core-js/object/assign`.

If any of your dependencies includes these polyfills, they'll be eliminated automatically from the production build to avoid duplication.

In addition, to reduce bundle size, Next.js will only load these polyfills for browsers that require them. The majority of the web traffic globally will not download these polyfills.

Server-Side Polyfills

In addition to `fetch()` on the client-side, Next.js polyfills `fetch()` in the Node.js environment. You can use `fetch()` in your server code (such as `getStaticProps`/`getServerSideProps`) without using polyfills such as `isomorphic-unfetch` or `node-fetch`.

Custom Polyfills

If your own code or any external npm dependencies require features not supported by your target browsers, you need to add polyfills yourself.

In this case, you should add a top-level import for the **specific polyfill** you need in your Custom `<App>` or the individual component.

JavaScript Language Features

Next.js allows you to use the latest JavaScript features out of the box. In addition to ES6 features, Next.js also supports:

- Async/await (ES2017)
- Object Rest/Spread Properties (ES2018)
- Dynamic `import()` (ES2020)
- Optional Chaining (ES2020)
- Nullish Coalescing (ES2020)
- Class Fields and Static Properties (part of stage 3 proposal)
- and more!

TypeScript Features

Next.js has built-in TypeScript support. Learn more here.

Customizing Babel Config (Advanced)

You can customize babel configuration. Learn more here.