Instruction

What Build Tools Do JavaScript Developers Use?

When starting a new JavaScript project, developers usually set up a build system using tools that help manage all of the moving parts of the project, and just about everything needed to build and run their application.

Package Managers

Many JavaScript applications rely on external dependencies like libraries, frameworks and other time-saving tools, which help developers avoid having to rewrite their own solutions to common problems. Libraries, frameworks and other software are created and shared by the JavaScript community and brought into a project using a **package manager**. A project could have tens or hundreds of external dependencies, and in many cases, some dependencies rely on other installed dependencies.

Package managers install and keep track of all the dependencies of a project. They also simplify the process of upgrading, configuring and removing dependencies. The most popular JavaScript package manager is **npm** (node package manager). Another commonly used package manager is **Yarn**, which was created by Facebook.

Module Bundlers

Since JavaScript applications often use frameworks, libraries and other external dependencies, the JavaScript source code is usually split between a number of files or modules.

Module bundlers combine all of your source code (and all of its dependencies) into a single, **minified** JavaScript file before it's served to the browser. Module bundlers commonly used today in JavaScript development are **Webpack**, **Rollup** and **Browserify**. **Parcel** is also a new open source JavaScript module bundler that recently launched.

Compilers

Javascript runtime environments (like the browser) only understand standard JavaScript. Modern, complex JavaScript applications often require more than pure JavaScript. For instance, you may want to take advantage of new, cutting-edge JavaScript features that are not yet supported by all browsers. Or, you're using a library or framework that uses a special syntax that's not understood by browsers (like **JSX**).

Earlier you learned that **TypeScript** and **CoffeeScript** are different "flavors" of JavaScript that do not run natively in the browser. If you try their code in the browser, you'll get all sorts of errors. Since these languages cannot be understood and ran by browsers, you need to transform them to standard JavaScript using a **compiler**.

A compiler is a tool that translates source code written using a special type of syntax into code that can work in the browser. Typescript, for instance, has a built-in compiler that converts a TypeScript file to plain JavaScript. **Babel** is the most popular JavaScript compiler used by developers, and it's the most reliable way to use features from ES2015 and beyond while supporting older browsers.

Task Runners

JavaScript programming is challenging enough. The less work you have to do when performing repetitive tasks, the easier your job becomes. Task runners define and run common tasks in your project. Anything you might do manually over and over again could be automated and handed off to a task runner. For example:

- Minifying your JavaScript so it can be loaded quickly and efficiently into the browser
- Running tests on your code
- Starting up a development server on your computer to run your app
- Automatically reloading the browser for you whenever a JavaScript file is saved
- Compiling source code from one type of syntax to another

Gulp is a common task runner, and **npm** (the node package manager) can also run automated tasks and is now the preferred task runner of many developers.

JavaScript tools change quickly. Be sure to check this page frequently (and **the state of JS build tools** for up-to-date information.