Contribute

Contents

Tooling setup

Bootstrap uses npm scripts to build the documentation and compile source files. Our package.json houses these scripts for compiling code, running tests, and more. These aren't intended for use outside our repository and documentation.

To use our build system and run our documentation locally, you'll need a copy of Bootstrap's source files and Node. Follow these steps and you should be ready to rock:

1. Download and install Node.js, which we use to manage our dependencies.
2. Either [download Bootstrap's sources]({{< param "download.source" >}}) or fork Bootstrap's repository.
3. Navigate to the root `/bootstrap` directory and run `npm install` to install our local dependencies listed in package.json.

When completed, you'll be able to run the various commands provided from the command line.

Using npm scripts

Our package.json includes numerous tasks for developing the project. Run `npm run` to see all the npm scripts in your terminal. **Primary tasks include:**

Task

Description

npm start

Compiles CSS and JavaScript, builds the documentation, and starts a local server.

npm run dist

Creates the dist/ directory with compiled files. Requires Sass, Autoprefixer, and terser.

npm test

Runs tests locally after running npm run dist

npm run docs-serve

Builds and runs the documentation locally.

{{< callout info >}} {{< partial “callout-info-npm-starter.md” >}} {{< /callout >}}

Sass

Bootstrap uses Dart Sass for compiling our Sass source files into CSS files (included in our build process), and we recommend you do the same if you're compiling Sass using your own asset pipeline. We previously used Node Sass for Bootstrap v4, but LibSass and packages built on top of it, including Node Sass, are now deprecated.

Dart Sass uses a rounding precision of 10 and for efficiency reasons does not allow adjustment of this value. We don't lower this precision during further processing of our generated CSS, such as during minification, but if you chose to do so we recommend maintaining a precision of at least 6 to prevent issues with browser rounding.

Autoprefixer

Bootstrap uses Autoprefixer (included in our build process) to automatically add vendor prefixes to some CSS properties at build time. Doing so saves us time and code by allowing us to write key parts of our CSS a single time while eliminating the need for vendor mixins like those found in v3.

We maintain the list of browsers supported through Autoprefixer in a separate file within our GitHub repository. See .browserslistrc for details.

RTLCSS

Bootstrap uses RTLCSS to process compiled CSS and convert them to RTL – basically replacing horizontal direction aware properties (eg. `padding-left`) with their opposite. It allows us only write our CSS a single time and make minor tweaks using RTLCSS control and value directives.

Local documentation

Running our documentation locally requires the use of Hugo, which gets installed via the hugo-bin npm package. Hugo is a blazingly fast and quite extensible static site generator that provides us: basic includes, Markdown-based files, templates, and more. Here's how to get it started:

1. Run through the tooling setup above to install all dependencies.
2. From the root `/bootstrap` directory, run `npm run docs-serve` in the command line.
3. Open `http://localhost:9001/` in your browser, and voilà.

Learn more about using Hugo by reading its documentation.

Troubleshooting

Should you encounter problems with installing dependencies, uninstall all previous dependency versions (global and local). Then, rerun `npm install`.