Lean Sass imports

When using Sass in your asset pipeline, make sure you optimize Bootstrap by only `@import`ing the components you need. Your largest optimizations will likely come from the `Layout & Components` section of our `bootstrap.scss`.

{{< scss-docs name="import-stack" file="scss/bootstrap.scss" >}}

If you're not using a component, comment it out or delete it entirely. For example, if you're not using the carousel, remove that import to save some file size in your compiled CSS. Keep in mind there are some dependencies across Sass imports that may make it more difficult to omit a file.

Lean JavaScript

Bootstrap's JavaScript includes every component in our primary dist files (`bootstrap.js` and `bootstrap.min.js`), and even our primary dependency (Popper) with our bundle files (`bootstrap.bundle.js` and `bootstrap.bundle.min.js`). While you're customizing via Sass, be sure to remove related JavaScript.

For instance, assuming you're using your own JavaScript bundler like Webpack or Rollup, you'd only import the JavaScript you plan on using. In the example below, we show how to just include our modal JavaScript:

```js
// Import just what we need

// import 'bootstrap/js/dist/alert';
// import 'bootstrap/js/dist/button';
// import 'bootstrap/js/dist/carousel';
// import 'bootstrap/js/dist/collapse';
// import 'bootstrap/js/dist/dropdown';
import 'bootstrap/js/dist/modal';
// import 'bootstrap/js/dist/offcanvas';
// import 'bootstrap/js/dist/popover';
// import 'bootstrap/js/dist/scrollspy';
// import 'bootstrap/js/dist/tab';
// import 'bootstrap/js/dist/toast';
// import 'bootstrap/js/dist/tooltip';
```

This way, you're not including any JavaScript you don't intend to use for components like buttons, carousels, and tooltips. If you're importing dropdowns, tooltips or popovers, be sure to list the Popper dependency in your `package.json` file.

{{< callout info >}}

Default Exports

Files in `bootstrap/js/dist` use the **default export**, so if you want to use one of them you have to do the following:

```js
import Modal from 'bootstrap/js/dist/modal'

const modal = new Modal(document.getElementById('myModal'))
```

{{< /callout >}}

Autoprefixer .browserslistrc

Bootstrap depends on Autoprefixer to automatically add browser prefixes to certain CSS properties. Prefixes are dictated by our `.browserslistrc` file, found in the root of the Bootstrap repo. Customizing this list of browsers and recompiling the Sass will automatically remove some CSS from your compiled CSS, if there are vendor prefixes unique to that browser or version.

Unused CSS

Help wanted with this section, please consider opening a PR. Thanks!

While we don't have a prebuilt example for using PurgeCSS with Bootstrap, there are some helpful articles and walkthroughs that the community has written. Here are some options:

- https://medium.com/dwarves-foundation/remove-unused-css-styles-from-bootstrap-using-purgecss-88395a2c5772
- https://lukelowrey.com/automatically-removeunused-css-from-bootstrap-or-other-frameworks/

Lastly, this CSS Tricks article on unused CSS shows how to use PurgeCSS and other similar tools.

Minify and gzip

Whenever possible, be sure to compress all the code you serve to your visitors. If you're using Bootstrap dist files, try to stick to the minified versions (indicated by the `.min.css` and `.min.js` extensions). If you're building Bootstrap from the source with your own build system, be sure to implement your own minifiers for HTML, CSS, and JS.

Non-blocking files

While minifying and using compression might seem like enough, making your files non-blocking ones is also a big step in making your site well-optimized and fast enough.

If you are using a Lighthouse plugin in Google Chrome, you may have stumbled over FCP. The First Contentful Paint metric measures the time from when the page starts loading to when any part of the page's content is rendered on the screen.

You can improve FCP by deferring non-critical JavaScript or CSS. What does that mean? Simply, JavaScript or stylesheets that don't need to be present on the first paint of your page should be marked with `async` or `defer` attributes.

This ensures that the less important resources are loaded later and not blocking the first paint. On the other hand, critical resources can be included as inline scripts or styles.

If you want to learn more about this, there are already a lot of great articles about it:

- https://web.dev/render-blocking-resources/
- https://web.dev/defer-non-critical-css/

Always use HTTPS

Your website should only be available over HTTPS connections in production. HTTPS improves the security, privacy, and availability of all sites, and there is no such thing as non-sensitive web traffic. The steps to configure your website to be served exclusively over HTTPS vary widely depending on your architecture and web hosting provider, and thus are beyond the scope of these docs.

Sites served over HTTPS should also access all stylesheets, scripts, and other assets over HTTPS connections. Otherwise, you'll be sending users mixed active content, leading to potential vulnerabilities where a site can be compromised by altering a dependency. This can lead to security issues and in-browser warnings displayed to users. Whether you're getting Bootstrap from a CDN or serving it yourself, ensure that you only access it over HTTPS connections.