Productionizing an app

Introduction to Angular 6

Building for production

- Use the build command to generate the files to serve it for production
  - By default generate all your compiled files in a folder called dist/.
  - You can then simply transfer the contents to a HTTP server
- However, this is a suboptimal, overweight build that would make your production application slow to load and slow to run
- The simplest thing to do to create a better build for production is to use the prod flag with the ng build command.
  - This accomplishes the following:

Building for production

- Bundling
  - Angular CLI bundles all the application and library files into a few bundles to make it faster to load in the browser
- Minification
  - process of removing all unneeded spaces, thus saving a few bytes of space in the final build
- Uglification
  - This is the process of replacing all nice, readable variable and function names with a smaller, two or three character name to save a few bytes.
- Turn off validation check
  - This is running by default in development mode (ng serve) to check for violation of Angular's development patterns
- Dead code elimination
  - The build process removes all unused code and unreferenced modules, thus dropping the bundle size further.

Ahead-Of-Time (AOT) compilation

- Just-in-Time (JIT) compilation
  - the application is compiled at runtime in the browser before running
  - default when you run the Angular application using ng serve or ng build
- In production mode, Angular uses AOT for compilation
  - Angular compiles as much of the application as possible upfront
- When the application is served to the browser, it is already precompiled and optimal, thus allowing the browser to quickly render and execute the application.
  - All HTML templates and CSS are inlined within the application bundle, thus saving asynchronous requests to load them later.

Ahead-Of-Time (AOT) compilation

- The Angular compiler, which constitutes almost half of the Angular library, is omitted resulting in significant reduction in the size of the built bundle

- Build Optimizer is a webpack plug-in to further optimize the bundle beyond what webpack is capable of.
  - focuses on removing some of the decorators and other code that is not relevant for the final build

Using base Href

- One concern when deploying any SPA is where it is served from.
  - In cases where application is served from the root domain (www.mytestpage.com), the default should work.
- In other cases (www.mytestpage.com/app), important that we update our <base> tag in the index.html.
  - This is responsible for setting the base path for all relative URLs in our application.
  - These include, but are not limited to, CSS/style files, JavaScript application and library files, images, and more.
- When we build our Angular application using the Angular CLI, we can specify or overwrite the base href value.

Handling CORS

- During development, we use a proxy to handle CORS
- Similar setup during production as well
  - Frontend server will be the one getting the initial API calls, and it then has to proxy those requests forward to the actual API server.
  - We simply route all requests to our API server (/api/), and all others to our static files
- Alternatively we can enable CORS on the API server
  - This allows pages from different origins (ports, protocols, domains, or subdomains) to make requests, bypassing the browser security restrictions.
  - Requires setting up the API server to respond with appropriate header information
  - https://enable-cors.org/
  - https://www.codecademy.com/articles/what-is-cors

Working with environments

- Another common requirement when building an application is having different configurations for different environments.
  - E.g. different API keys for client-side tracking libraries, or different server URLs to configure for test versus production.
- In such cases, you can use the concept of environments
  - By default, when you create a new Angular application, it creates an src/environments folder, with one file per environment.
- The Angular CLI makes the properties available in the environment.ts file available across your application.
  - This can be overridden with the appropriate command flags:
  - In your application, you can import the main environment file, and Angular will ensure you get the correct properties based on the flag

Lazy loading

- A common technique to increase performance and reduce initial load time
  - try to load the bare minimum up front in the initial request and defer loading everything else to as and when it's needed.
  - Accomplish this by leveraging child routes
- The technique is as follows:
  - Break up our application into smaller modules, each with their routes defined in self-contained units.
  - The respective components are now registered at these submodule level only, and not at the main application-level module.
  - We register all these routes as child routes in each individual module.
  - At the application level, we change our routing to instead point certain subpaths at the new module, rather than the individual routes.

Server side rendering

- A normal Angular application executes in the browser, rendering pages in the DOM in response to user actions.
- Angular Universal generates static application pages on the server through a process called server-side rendering (SSR).
- When Universal is integrated with your app, it can generate and serve those pages in response to requests from browsers.
  - It can also pre-generate pages as HTML files that you serve later.

Server side rendering

- Facilitate web crawlers for SEO
  - Most search engine crawlers will not actually render and execute JavaScript when they try to crawl web pages
  - Thus, deep links and routes in a SPA also do not get indexed properly
  - Angular Universal can generate a static version that is easily searchable, linkable, and navigable without JavaScript.
  - Universal also makes a site preview available since each URL returns a fully rendered page.
- Improve performance on mobile and low-powered devices
  - Some devices don't support JavaScript or execute JavaScript so poorly that the user experience is unacceptable.
  - For these cases, you may require a server-rendered, no-JavaScript version of the app.

Server side rendering

- Displaying the first page quickly can be critical for user engagement.
  - You can generate static landing pages which are pure HTML with no-Javascript to engage the user
  - At the same time, you'll load the full Angular app behind it.
  - The user perceives near-instant performance from the landing page and gets the full interactive experience after the full app loads.