# Angular Architecture Playbook

Basic components of an angular app – modules, components (Standalone), Services (important to separate business logic from UI logic).

For smaller application use Standalone component but for big application we should still use modules as the standalone components don’t scale well.

Common modules – root, feature, shared and core modules.

Standalone components are primarily used to architect simpler, smaller angular apps and give you more of a React feel.

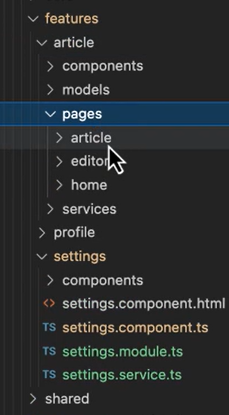
Modules offer a great way of architecting our application and providing good separation of concerns, but we have to use another layer of abstraction that ultimately isn’t really needed. Standalone components remove the unnecessary module abstraction and so provide some simplicity, but they can get a bit unwieldy in the import section if the component gets too big.

A computer screen with text

Description automatically generated

Some architects suggest using Standalone component for library and shared component and to use modules for page components that are loaded in routing context. It allows us to hide your internal components and expose external components via module.

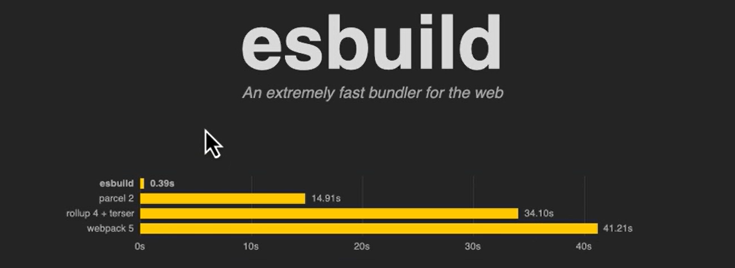
As a best practice, standalone components encompass the internal components used by features and shared libraries and modules exposes the public API that is used to display pages and the like. So, we should use combination of module and standalone based components in our project.



We should also load our component lazily by using Standalone component –



In angular 18, we can use a new builder that exclusively uses esbuild in Vite as opposed to Webpack. The esbuild is an extremely fast bundler for the web. Esbuild is used under the hood by Vite, which is more of a new improved version of Webpack JS build tool. Bysically, it is replacing the angular devkit builder with a new one.

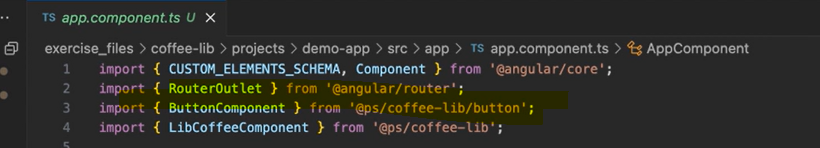


As a best practice we should use new Vite based angular builder for the fast build time which also supports integrated server side rendering and also save CD/CI cost on cloud as it builds the app much faster.

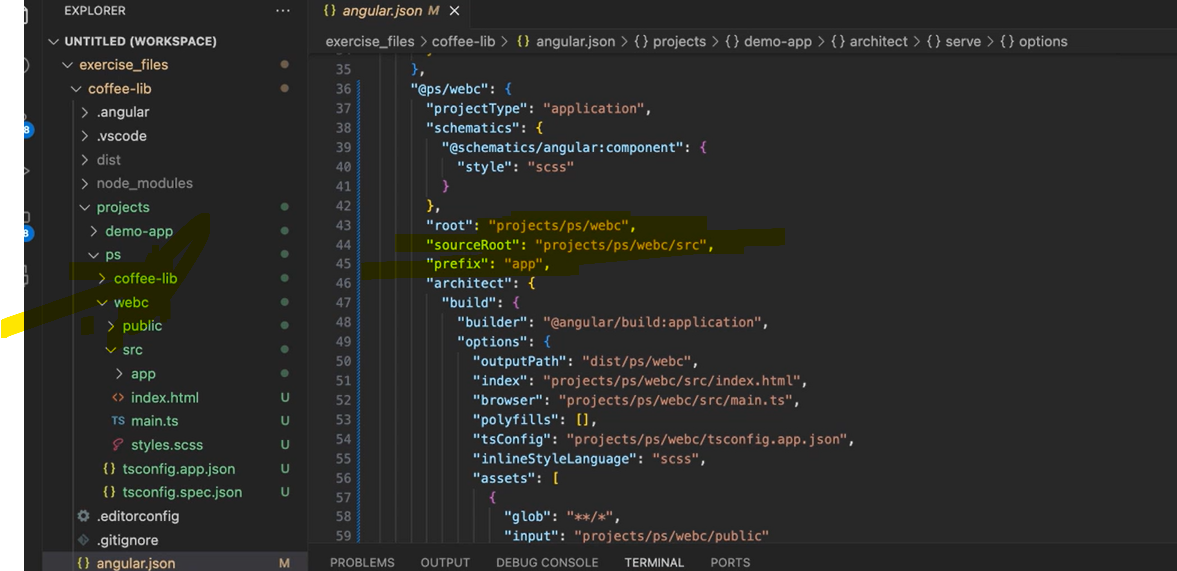
We should separate out the angular libraries into sub-libraries with multiple entry points so that user can only import what he wants to use from our library –

A screenshot of a computer

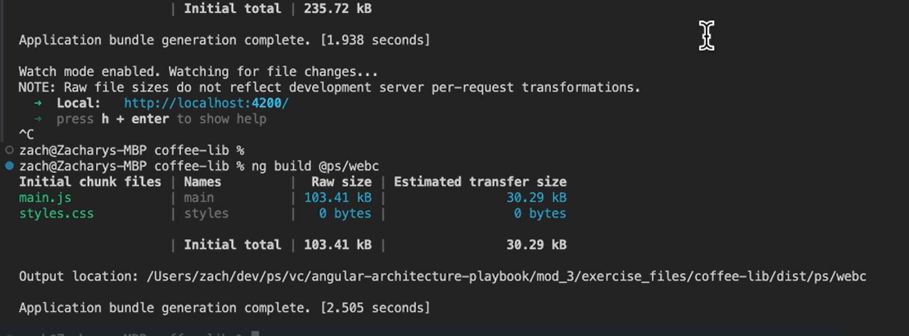
Description automatically generated

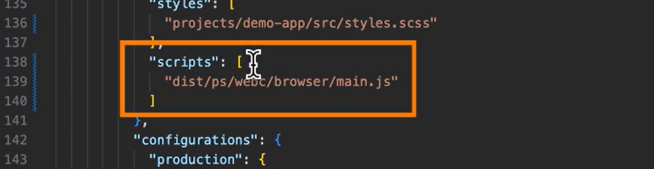


We can also create a web component library. Angular based web components are unique in that they use Angular elements to be created. Currently there is no way to generate an angular component library that exposes a web component, but as we see angular continue to innovate and with the advent of zoneless application, we very well may soon see the day where we can create angular libraries directly that can expose web components. Web Components are reusable, encapsulated, and standards-compliant components that can work seamlessly across different frameworks and libraries.



The outcome of the this angular based web components library simply have one main.js file which the consumer app can utilize this by referring this file as a script –





Architecting large scale applications – it has mainly four concepts – micro-frontends, module federation and mono-repo integration (Nx).