CSM
breaking thought barriers

Angular Training

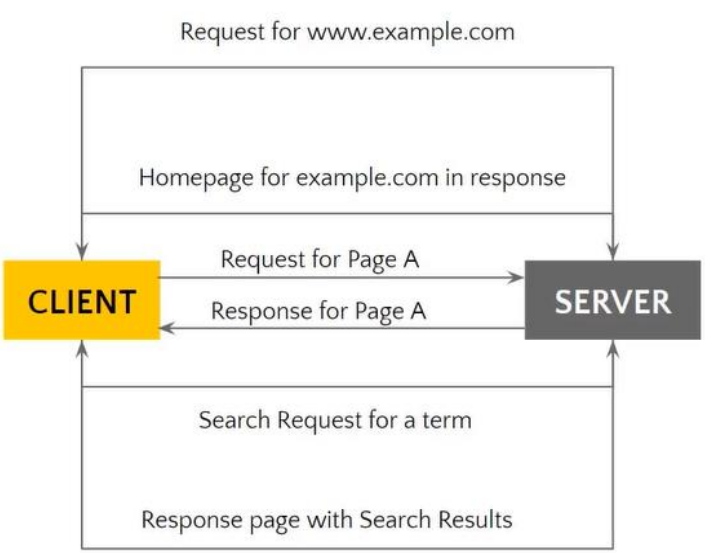
Session -11

Outlines

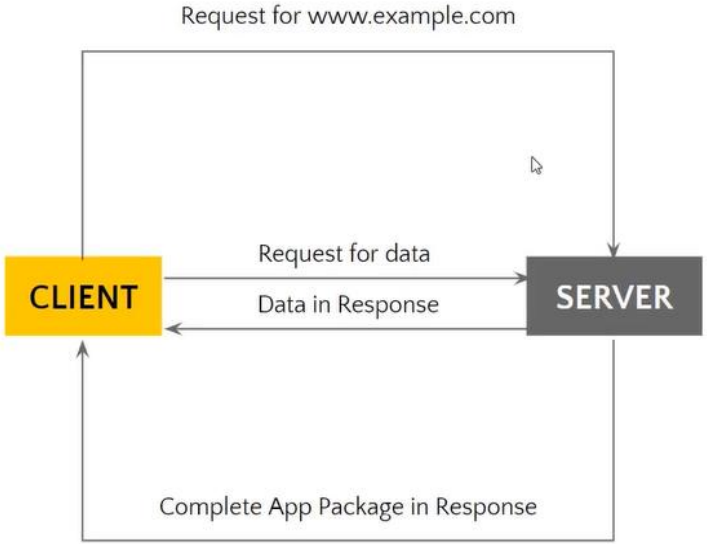
CSM
breaking thought barriers

- Angular
- SPA
- Why Angular
- Environment Setup
- Create First Angular App

SPA



- Multiple Page Requests going to Server.
- Client getting reloaded continuously.
- Eg: MVMT Watches



- Single Request to load the App, multiple data requests going to the server
- Client doesn't get reloaded continuously.
- Eg: AngularIO

Why Angular


Angular
Protractor
Forms
PWA
Augury
Language Services
Router
Elements
CDK
Universal
Karma
Labs
Compiler
I18N
HTTP
Material
Animations
CLI


PWA

Dependency Injection

Lazy Loading

Libraries

Performance

Templates

https://angular.io/guide/releases

Environment Setup

Web Browser

NodeJS

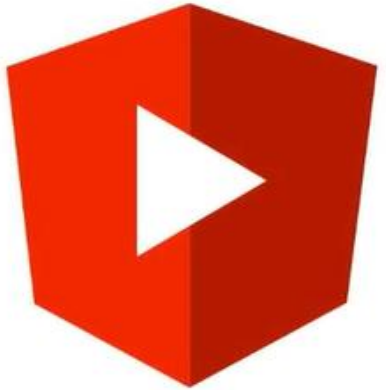
TypeScript

Angular CLI

Code Editor

StackBlitz

CodeSandbox



NG-Run

Angular CLI

To build Angular applications, we need many dependencies, a web development server, packages, all connected to each other. But doing all this from scratch every time is tedious. That's where Angular's command-line tool, the Angular CLI, comes in.

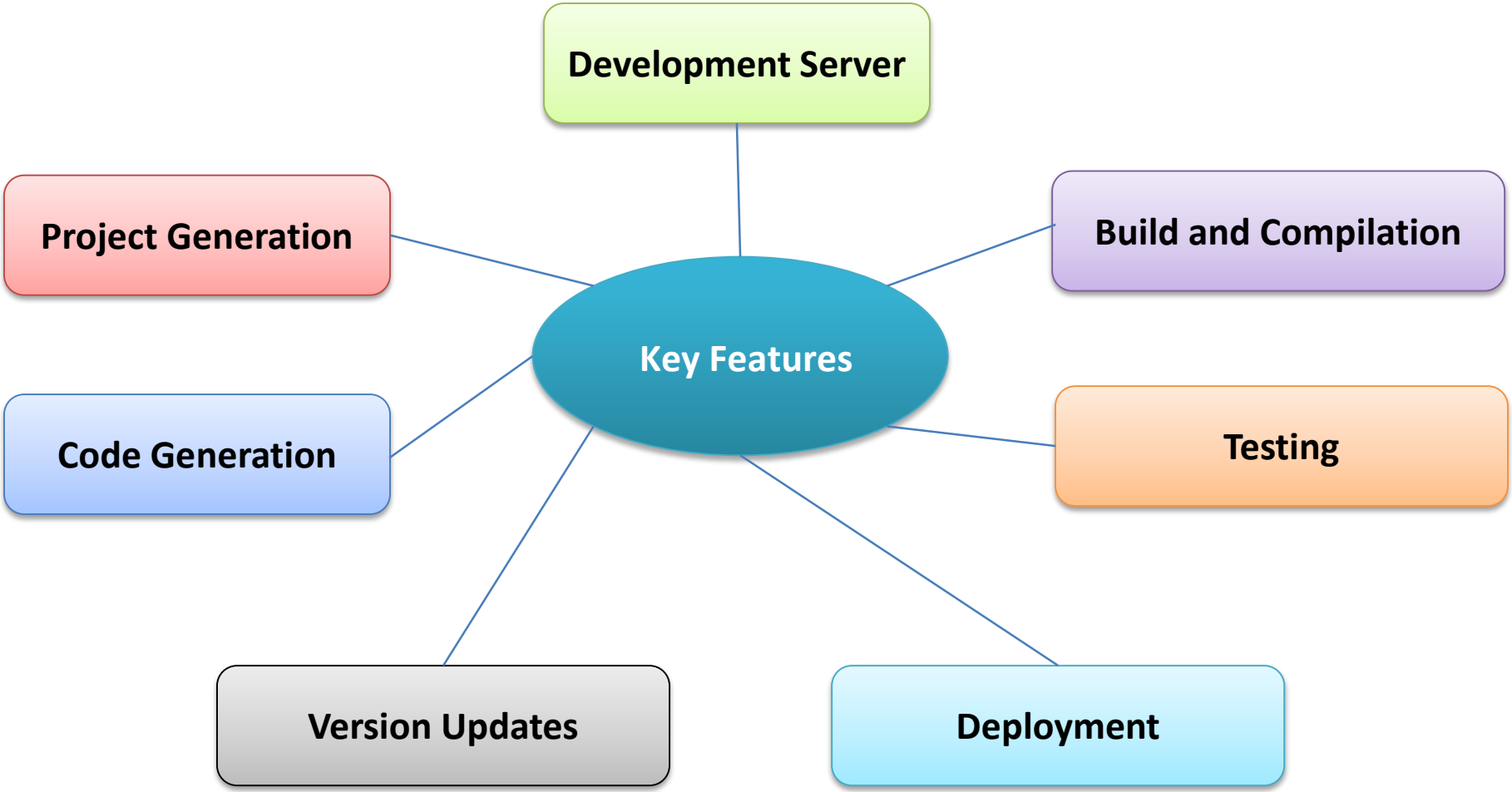
```
npm install -g @angular/cli
```

Angular CLI

- The Angular CLI (Command Line Interface) is a powerful tool that simplifies and streamlines the process of creating, developing, testing, and deploying Angular applications.

- It provides a set of commands that help developers generate code, manage dependencies, run development servers, and perform various tasks related to building Angular applications.

- The Angular CLI is an essential tool for any Angular developer as it significantly boosts productivity and enforces best practices.


Development Server
Project Generation
Build and Compilation
Key Features
Code Generation
Testing
Version Updates
Deployment

1. Project Creation:	ng new project-name
2. Component Generation:	ng generate component component-name
3. Service Generation:	ng generate service service-name
4. Module Generation:	ng generate module module-name
5. Directive Generation:	ng generate directive directive-name
6. Pipe Generation:	ng generate pipe pipe-name
7. Testing and Running:	ng test ng serve
8. Building for Production:	ng build
9. Updating the CLI:	ng update @angular/cli

Create Your First Angular App

Workspace configuration files

WORKSPACE CONFIGURATION FILES	PURPOSE
.editorconfig	Configuration for code editors
.gitignore	Specifies intentionally untracked files that Git should ignore.
README.md	Introductory documentation for the root application.
angular.json	CLI configuration defaults for all projects in the workspace, including configuration options for build, serve, and test tools that the CLI uses, such as Karma, and Protractor.
package.json	Configures npm package dependencies that are available to all projects in the workspace. See npm documentation for the specific format and contents of this file.
package-lock.json	Provides version information for all packages installed into node_modules by the npm client.
src/	Source files for the root-level application project.
node_modules/	Provides npm packages to the entire workspace. Workspace-wide node_modules dependencies are visible to all projects.
tsconfig.json	The base TypeScript configuration for projects in the workspace. All other configuration files inherit from this base file. For more information, see the Configuration inheritance with extends section of the TypeScript documentation.

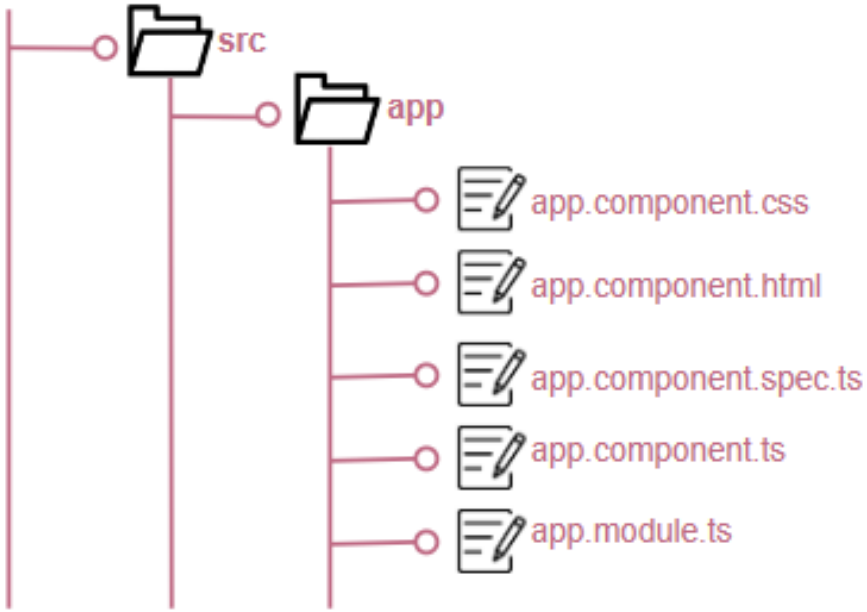
Application configuration files

APPLICATION-SPECIFIC CONFIGURATION FILES	PURPOSE
tsconfig.app.json	Application-specific TypeScript configuration, including TypeScript and Angular template compiler options.
tsconfig.spec.json	TypeScript configuration for the application tests.

Application source files

APPLICATION SUPPORT FILES	PURPOSE
app/	Contains the component files in which your application logic and data are defined.
assets/	Contains image and other asset files to be copied as-is when you build your application.
favicon.ico	An icon to use for this application in the bookmark bar.
index.html	The main HTML page that is served when someone visits your site. The CLI automatically adds all JavaScript and CSS files when building your app, so you typically don't need to add any <script> or <link> tags here manually.
main.ts	The main entry point for your application. Compiles the application with the JIT compiler and bootstraps the application's root module (AppModule) to run in the browser. You can also use the AOT compiler without changing any code by appending the --aot flag to the CLI build and serve commands.
styles.css	Lists CSS files that supply styles for a project. The extension reflects the style preprocessor you have configured for the project.

Structure of app folder

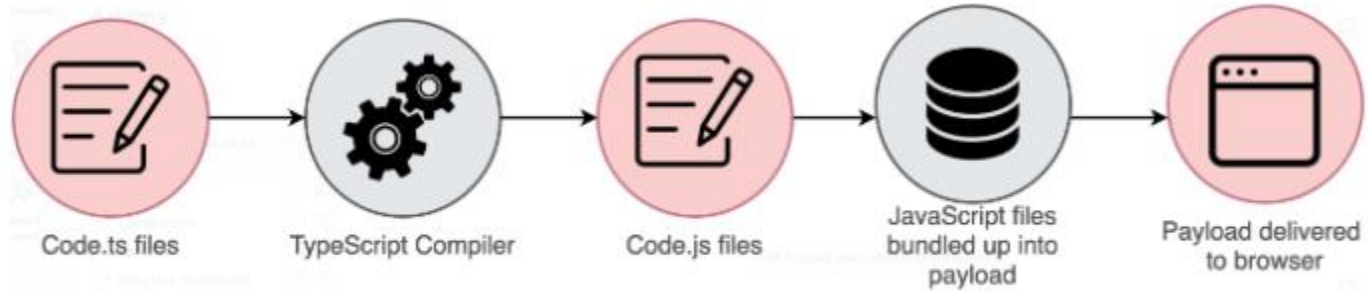


Inside the src folder, the app folder contains your project's logic and data.

SRC/APP/ FILES	PURPOSE
app/app.config.ts	Defines the application config logic that tells Angular how to assemble the application. As you add more providers to the app, they must be declared here. *Only generated when using the --standalone option.*
app/app.component.ts	Defines the logic for the application's root component, named AppComponent. The view associated with this root component becomes the root of the view hierarchy as you add components and services to your application.
app/app.component.html	Defines the HTML template associated with the root AppComponent.
app/app.component.css	Defines the base CSS stylesheet for the root AppComponent.
app/app.component.spec.ts	Defines a unit test for the root AppComponent.
app/app.module.ts	Defines the root module, named AppModule, that tells Angular how to assemble the application. Initially declares only the AppComponent. As you add more components to the app, they must be declared here. *This file is not generated when using the --standalone option.*

The Architecture of an Angular App

When the Angular application app is run in the browser, these TypeScript files are converted into JavaScript files, which are then bundled up into the payload that is delivered to the browser



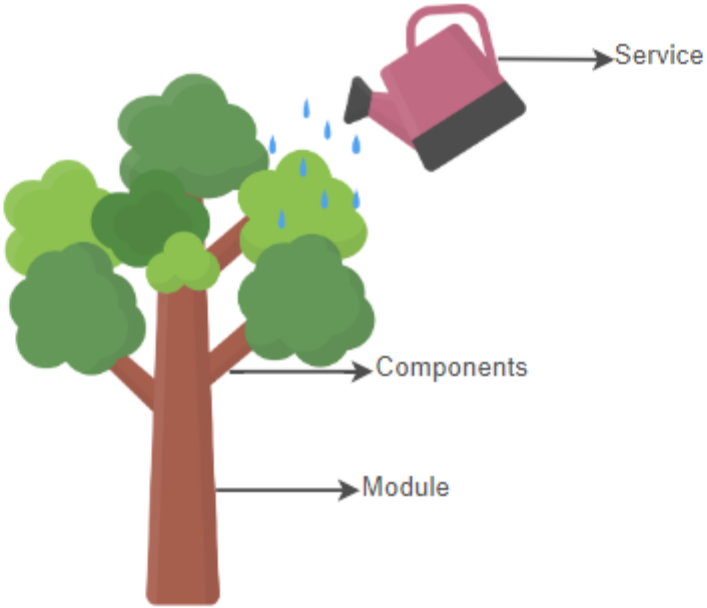
The Angular app running in the browser

The Architecture of an Angular App

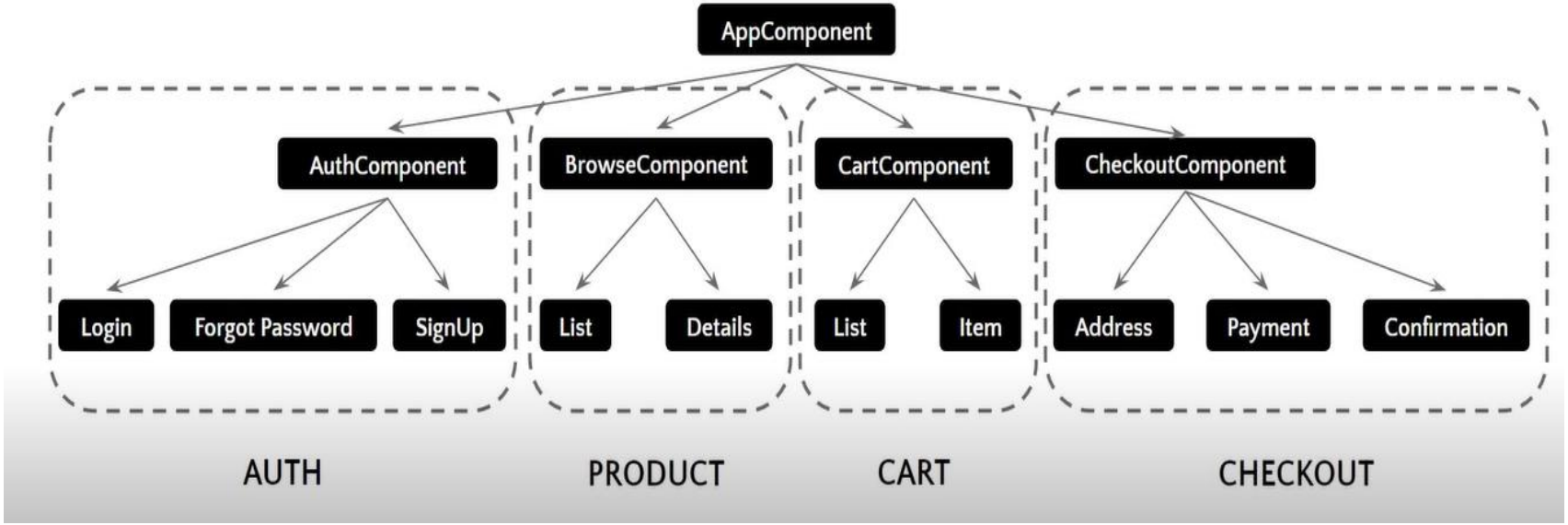
There are three main parts to an Angular application

Modules	**Modules** are the glue that hold an application together. Each module is a single TypeScript file that references all the other files used within the application.
Components	**Components** are the building blocks of the application. They are single pieces of functionality in our application, which are linked together under a module.
Services	**Services** are single TypeScript classes used to access information and share it between components.

Angular application as a tree



App as a Tree of Components



Design a simple Website