* Ng new new-app –skip-install (it will only creates file won’t do npm install)
* ng new new-app –dry-run (it won’t create any files it will only tell you which file it will create)
* ng new new-app –prefix acme(change the prefix of selector in component)
* ng new new-app --skip-tests (no spec files)
* ng new new-app --routing (to create app-routing modules)
* ng new new-app --style scss(if you want to use sass by default for new app)
* ng –help
* –save-dev (this will save the installed in dev dependency)

**Configuring angular-cli after creation**

* ng set defaults.styleExt scss (to change the style to scss)
* ng set defaults.styleExt scss -g ((to change the style to scss in global angular-cli.json for all cli projects)

**Linting**

* ng lint
* ng lint –help
* --fix (automatically fix the fixable problems)
* --format stylish (lint anf format the o/p)

**Blueprints**

* ng generate component home-catpotal or ng g c home-catpotal

options

* + –flat (should a folder be created)
  + –inline-template or -it(will the template be in the .ts file)
  + –inline-style or -is(will the style be in the .ts file)
  + –spec (generate a .spec file ?)
  + –view-encapsulation or –ve (view encapsulation strategy)
  + –change-detection –cd (change detection stategy)
  + –dry-run or –d
* ng g d app-directive
* Services
  + ng g s app-data (it will create service but it won’t be registered to any module)
  + ng g s app-data –m app.module(this will register the service inside app.module)
* ng g cl model/customer (it will create a class inside a model folder)
* ng g I model/person (it will create a interface inside a model folder)
* ng g enum model/gender (it will create a enum inside a model folder)
* ng g c p shared/init-caps (it will create a pipe inside a shared folder)
* **Module**
  + ng g m login (create a module inside login folder)
  + ng g c login -m login/login.module (it will create a component inside login folder and update the login.module)
* ng set defaults.component.flat true (Do not create a folder for components)
* ng set defaults.component.flat true (Do not create a folder for directives)
* ng g m login –routing app-data(it will create login.module and also routing module and register routing module in login.module)
* ng g guard <guard\_name> (generate a authguard)

Making Build

* Files contained inside dist folder:
  + Inline.bundle.js (webpack runtime=>to put everything together for us so that the application should work)
  + main.bundle.js (the code we write)
  + polyfills.bundle.js (polyfills for different browsers)
  + styles.bundle.js (where all the styles is located)
  + vendor.bundle.js (contain angular and the other third party or vendor files )
* npm i source-map-explorer –save-dev
* .node\_modules/.bin/source-map-explorer dist/main.bundle.js (it will display which component structure is consuming how much memory for main.bundle.js)
* .node\_modules/.bin/source-map-explorer dist/ vendor.bundle.js (it will display which component structure is consuming how much memory for vendor.bundle.js)

Environment

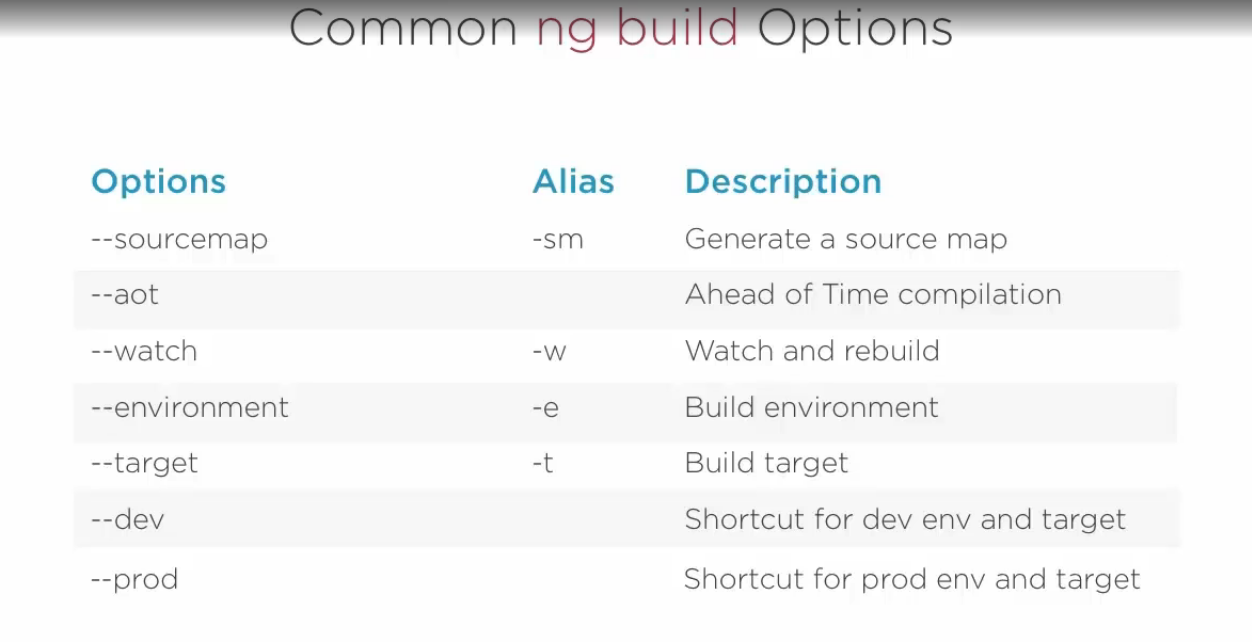
* Indicates which file to use between environment.prod.ts and environment.ts

**Target**

* Defines how (and if) the files are optimized



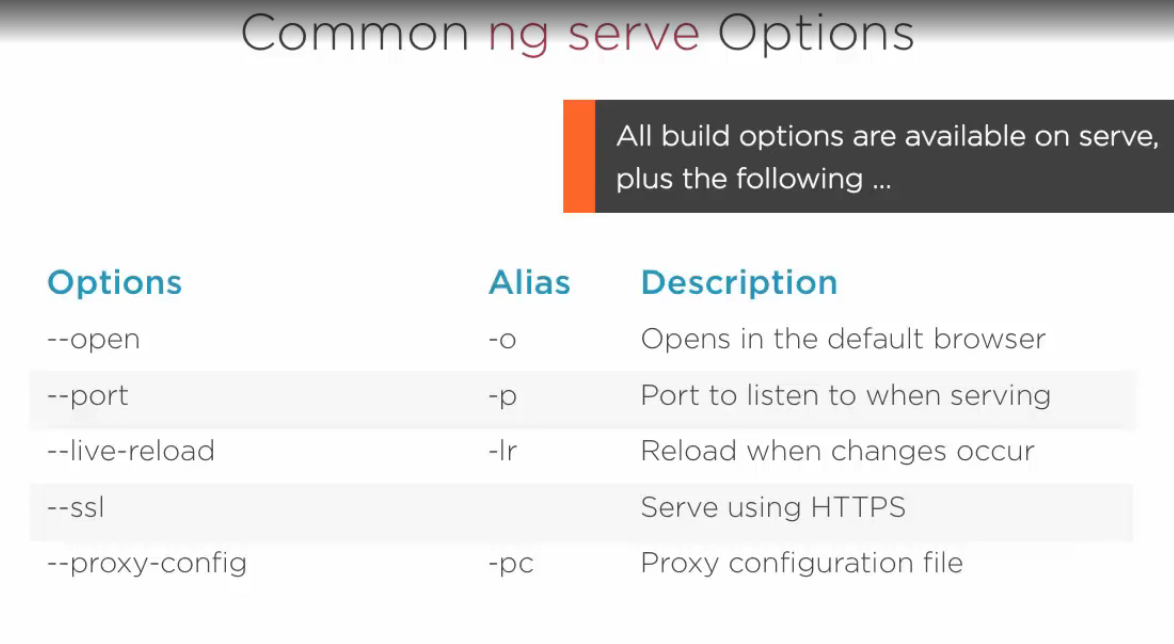
* Dev build
  + ng build --target=development –environment=dev
  + ng build --dev -e=dev
  + ng build --dev
  + ng build
* Production build
  + ng build --target=production –environment=dev
  + ng build –prod -e=prod
  + ng build –dev



* ng build –prod –sm (to run source map inside production build)

**Serve**

* all build options will be available during “ng serve”



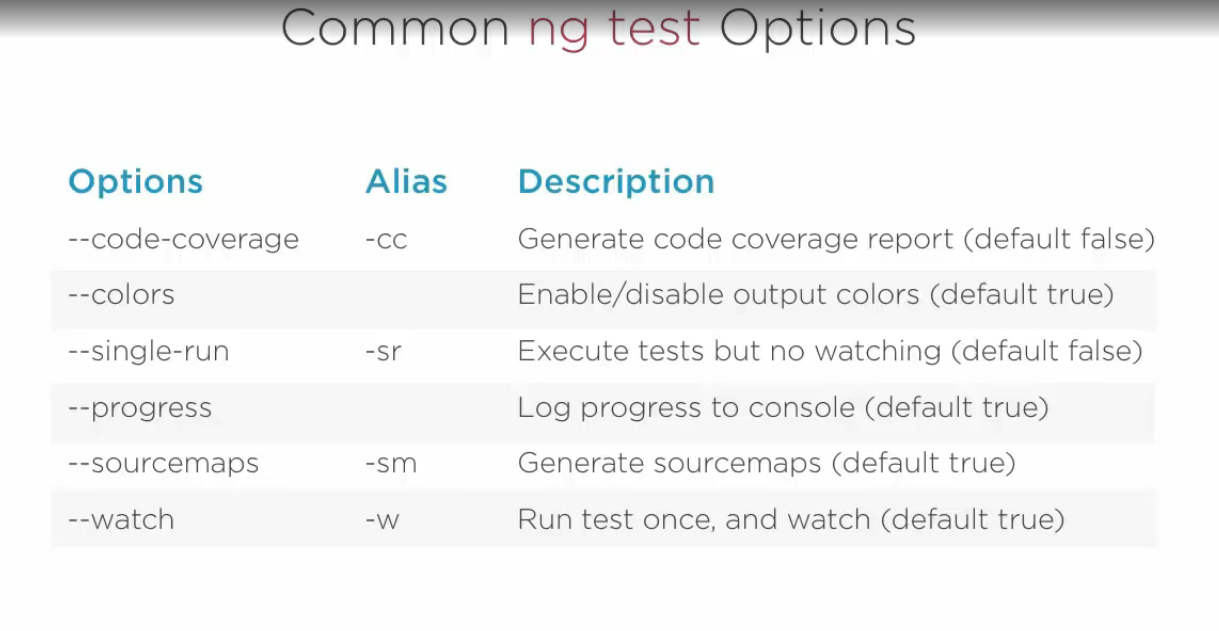
* ng serve –prod

**Eject**

* ng eject (to remove angular-cli from your project)

**Test**

* ng test
* ng test –sr or ng test –w false (run test for single time only)



Note:

* any external .js file will be inserted inside scripts array in angular-cli.json
* any external .css file will be inserted inside style array in angular-cli.json