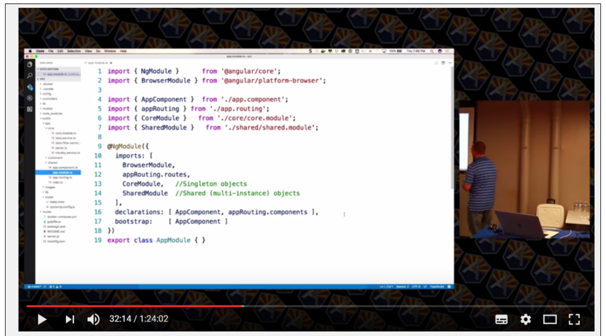
Multiple modules

Splitting your application into separate, reusable modules

Modules

- Introduced in Angular 2-rc.5
- Successor of Angular 1 `angular.module('myApp', […])`
- Divide your app into *logical* and often *reusable* pieces of code
- Keyword : code organization

- Recommendation by John Papa/Dan Wahlin for larger projects:
- Use one `AppModule` - the root of your app
- Use one `CoreModule` - containing all *singletons* in your app
- Use one `SharedModule` - containing all shared resources, possible multiple instances
- Use additional modules *per feature*
- https://www.youtube.com/watch?v=YxK4UW4UfCk

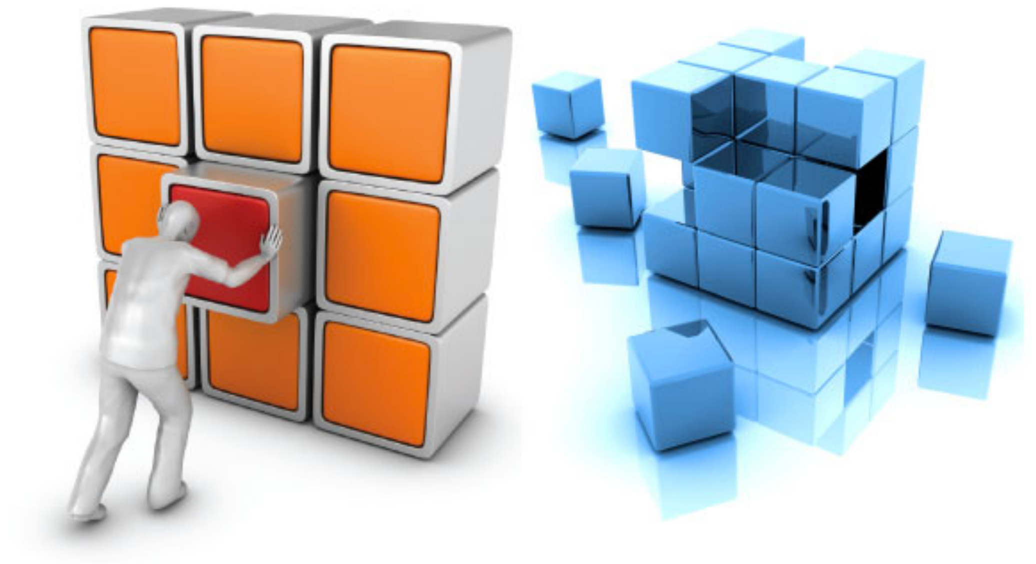


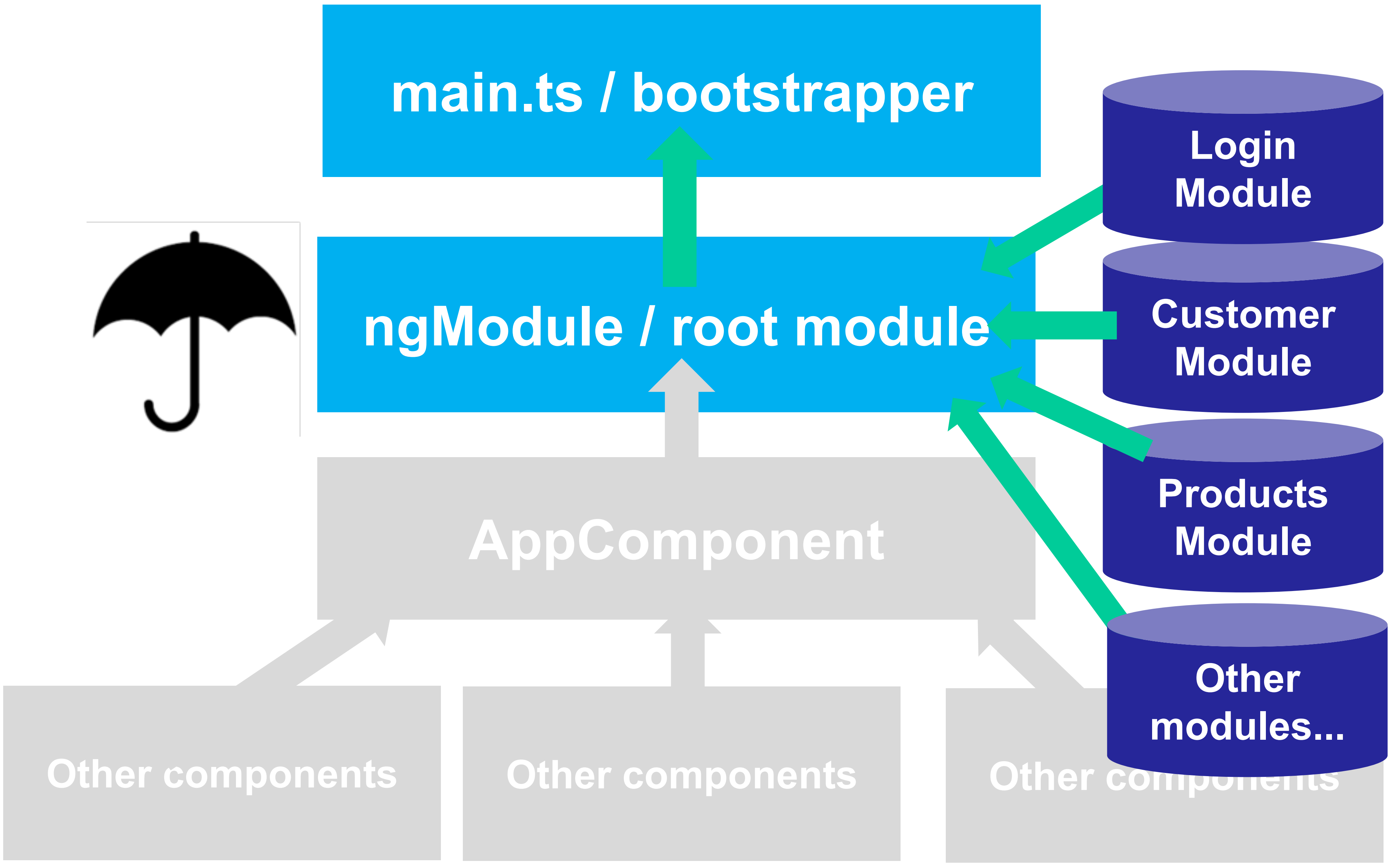
```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent }  from './app.component';
import { appRouting } from './app.routing';
import { CoreModule }   from './core/core.module';
import { SharedModule }   from './shared/shared.module';

@NgModule({
  imports: [
    BrowserModule,
    appRouting.routes,
    CoreModule,   //Singleton objects
    SharedModule  //Shared (multi-instance) objects
  ],
  declarations: [ AppComponent, appRouting.components ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Integrating Angular with RESTful Services using RxJS and Observables

Application – multiple Modules

- *Reuse* of Components, Pipes, Routes and Services etc. over different apps
- *Wrap* each set of logical related components, services, etc. in its own module.




main.ts / bootstrapper
ngModule / root module
AppComponent
Login Module
Customer Module
Products Module
Other modules...
Other components
Other components
Other components

Steps

1. Create a new module
   - Optional: test first with `--dry-run`
   - `ng generate module customers --dry-run`
2. Create component(s) inside that module
   - Again: test first with `--dry-run`
   - `ng generate component customers --module customers --dry-run`
3. Apply UI, logic, etc. to your component
4. Export your component inside `customer.module.ts`
   - `exports : [CustomerComponent],`
   - Otherwise it can’t be used in other components!
5. Provide new module to `app.module.ts`
   - `imports: [CustomerModule]`

Optional : SharedModule

- Reuse components in multiple modules? Use a SharedModule
  - `ng g m shared` – shorthand notation
- Create components inside SharedModule
- Import SharedModule in other modules
- It doesn't have to be in AppModule if you don't use it directly!
- It *does* add size to module bundles AFAIK
  - Modules need to be able to run on their own.

Demo