Angular

Day 1

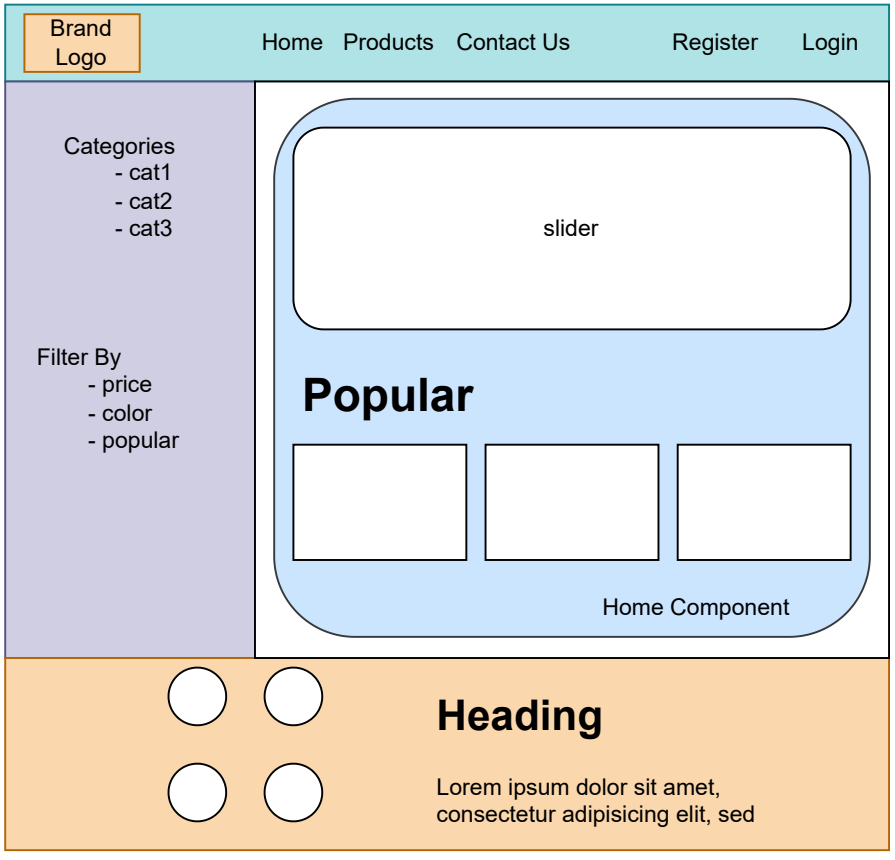
Framework

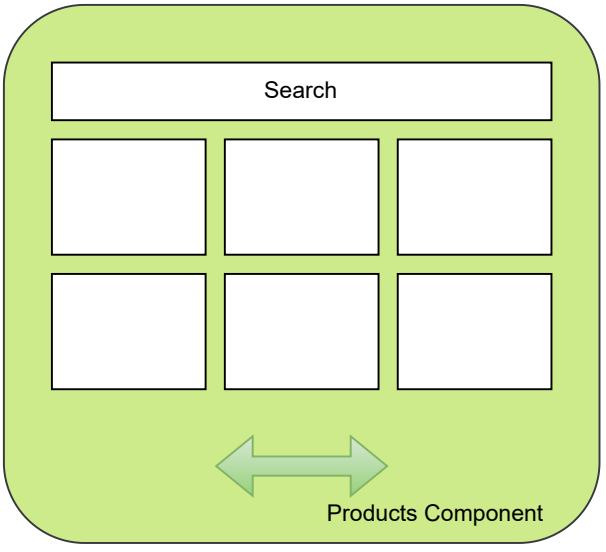
Open Source

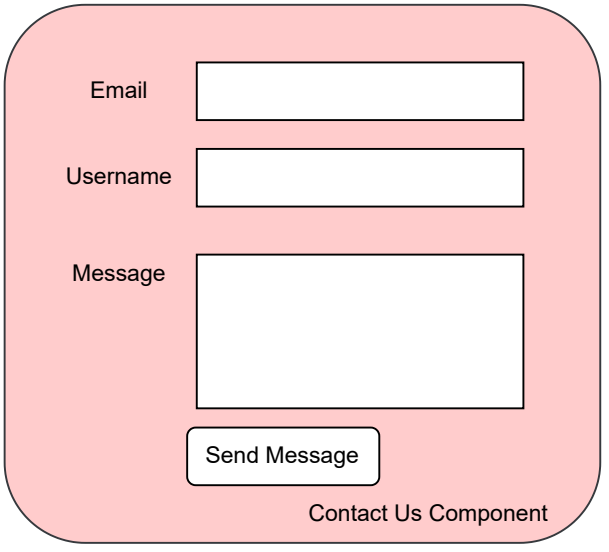
Component Based Application

Single Page Application (Routing)

Library (React)







Cmd Commands

install angular: npm i -g @angular/cli
check angular version: ng version
install node modules from package.json: npm i
run application:
ng server [-o]
npm start
install bootstrap : npm i bootstrap
create new component : ng g c componentName --skip-tests

Angular Files

node modules (Folder): Contains all Modules(libraries)
package.json: containe all modules Names with versions
package-lock.json: containd Modules dependencies names
.gitignore : tells what will not be used in version control
read me file : contains repo description for version control
tsconfig. (3 files) : typescript configuration files
angular.json : Configuration for angular project
src (Folder) : Contains project files

Angular Component

ComponentName.component.html

Template
(Component Content)

ComponentName.component.css

Style

ComponentName.component.ts

Logic (Behavior)

Create Component:

Way 1 (manually):
src > app > components > FolderWithComponentName

create component file (html,css,ts)

inside ts file create class
add decorator (selector, template, style)

inside app.module
=> in declarations add my component

way 2 (with angular cli):
ng g c ComponentName --skip-tests

Angular Component Communication

Binding

(one way binding)
.ts => .html
- interpolation binding {{ variableName}}
- property binding [src]="variableName"

.html => .ts

(two way binding)
both (bidirection)

Install bootstrap: npm i bootstrap
in angular.json: add in **build (styles and scripts)**

VS Code Extensions:
Auto Import
Angular Language Service

Lab

Create following components
- navbar
- aside
- footer
- slider => read images from ts
- prodcus
- product card

index.html

```
<app-root></app-root>
```

app.component

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Lec-Demo';
}
```

nav bar component

<app-navbar />

products component

<app-products/>

footer component

<app-footer />

main.ts

```
platformBrowserDynamic()
.bootstrapModule(AppModule)
.catch(err =>
console.error(err));
```

app.module.ts

```
@NgModule({
  declarations: [
        // components
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Day 2

Angular Component Communication

Binding

(one way binding)

.ts => .html
- interpolation binding {{ variableName}}
- property binding [src]="variableName"

.html => .ts
- event binding

(two way binding)

both (bidirection) .html => .ts => .html

=> Normal way
(event binding) + (interpolation or property binding)

=> FormsModule
- add formsModule in app module

Event Binding

.html

.ts

Directives

Component Directive

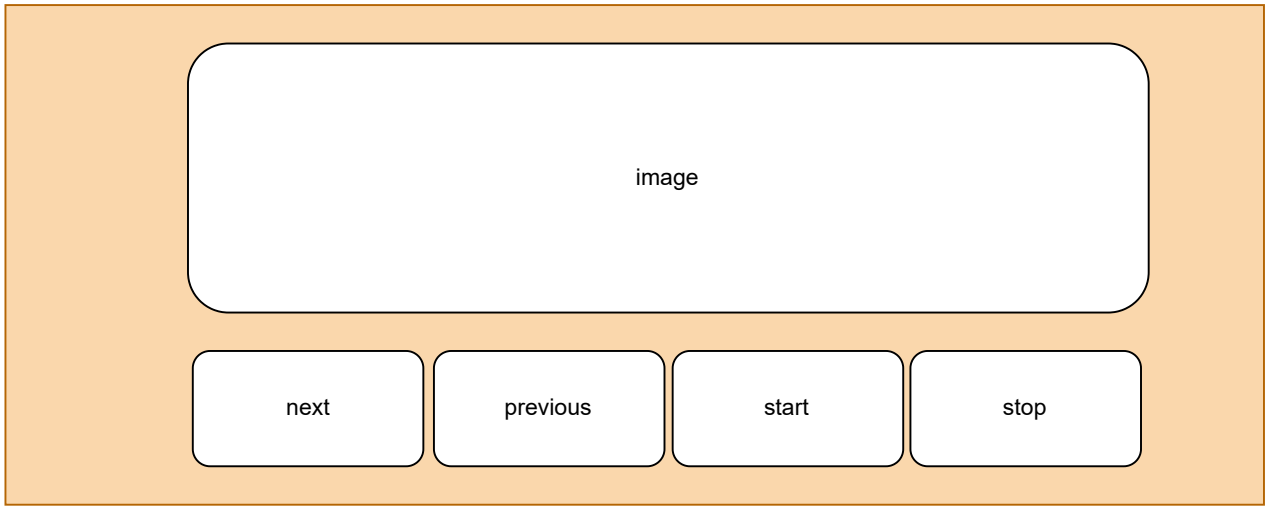
Structural Directive

*ngIf
*ngIf else
*ngFor
*ngSwitch

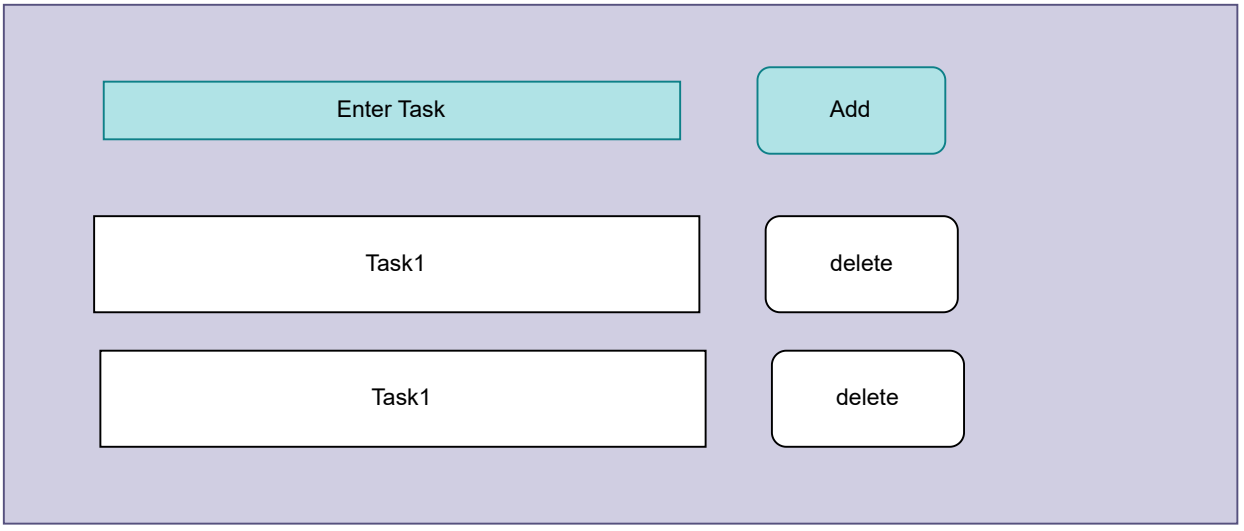
Attribute Directive

[ngclass]
[ngstyle]

Slider



ToDo List

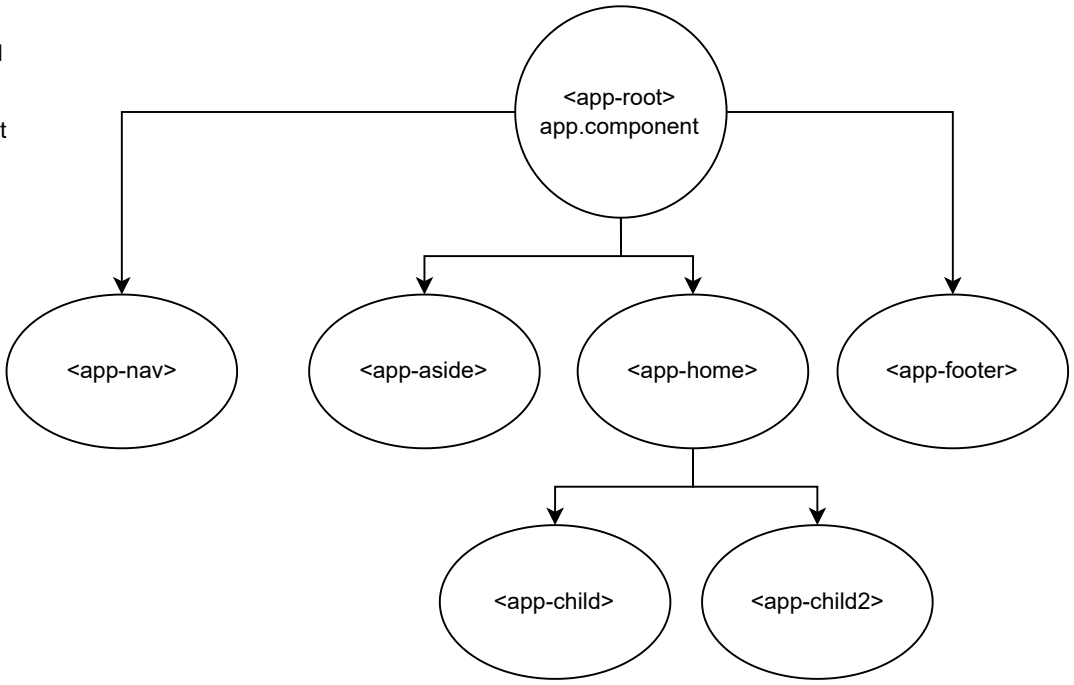


Component Interaction

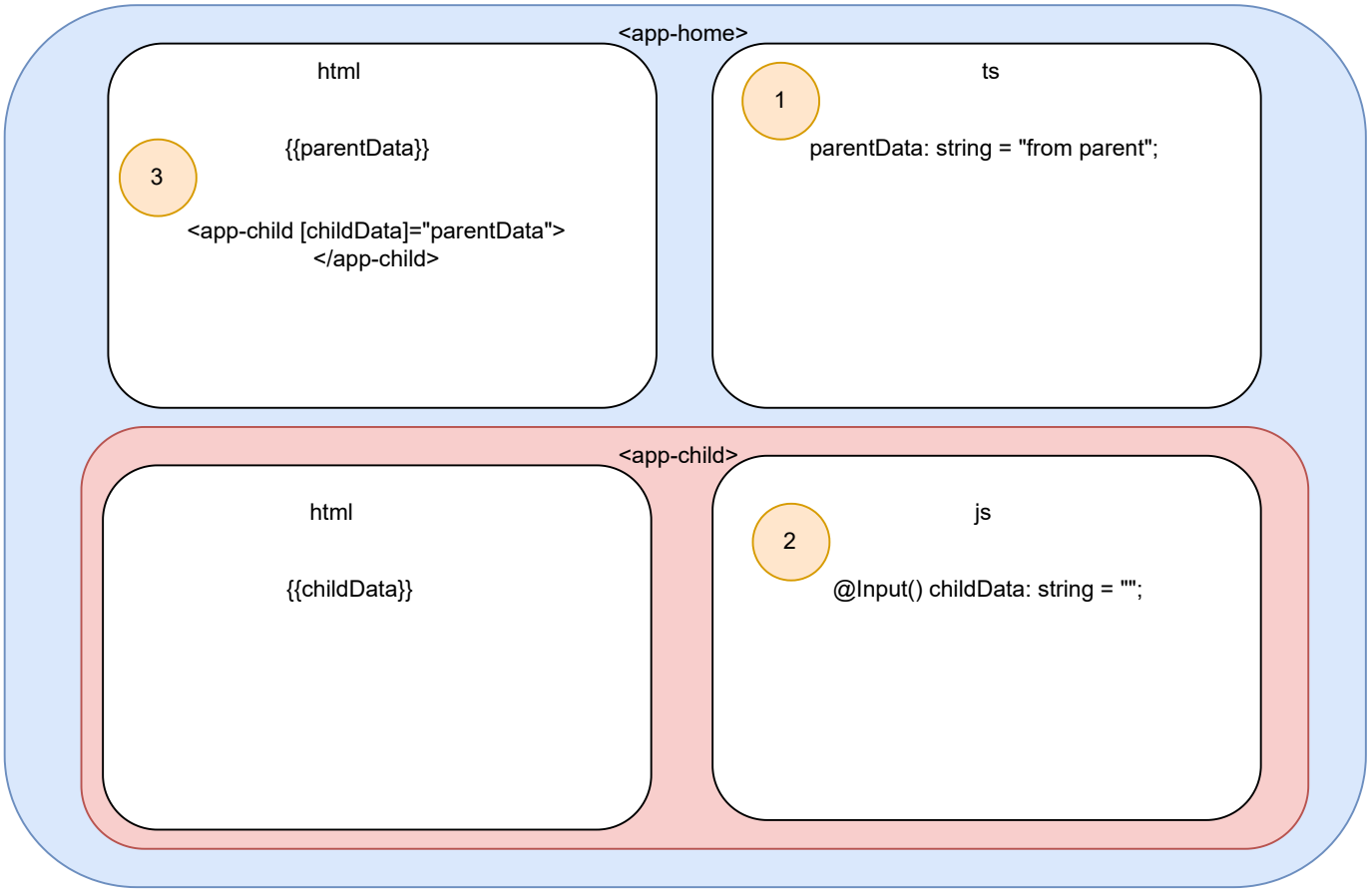
Data from parent to child

Data from child to parent

Data from child to child



Data from parent to child



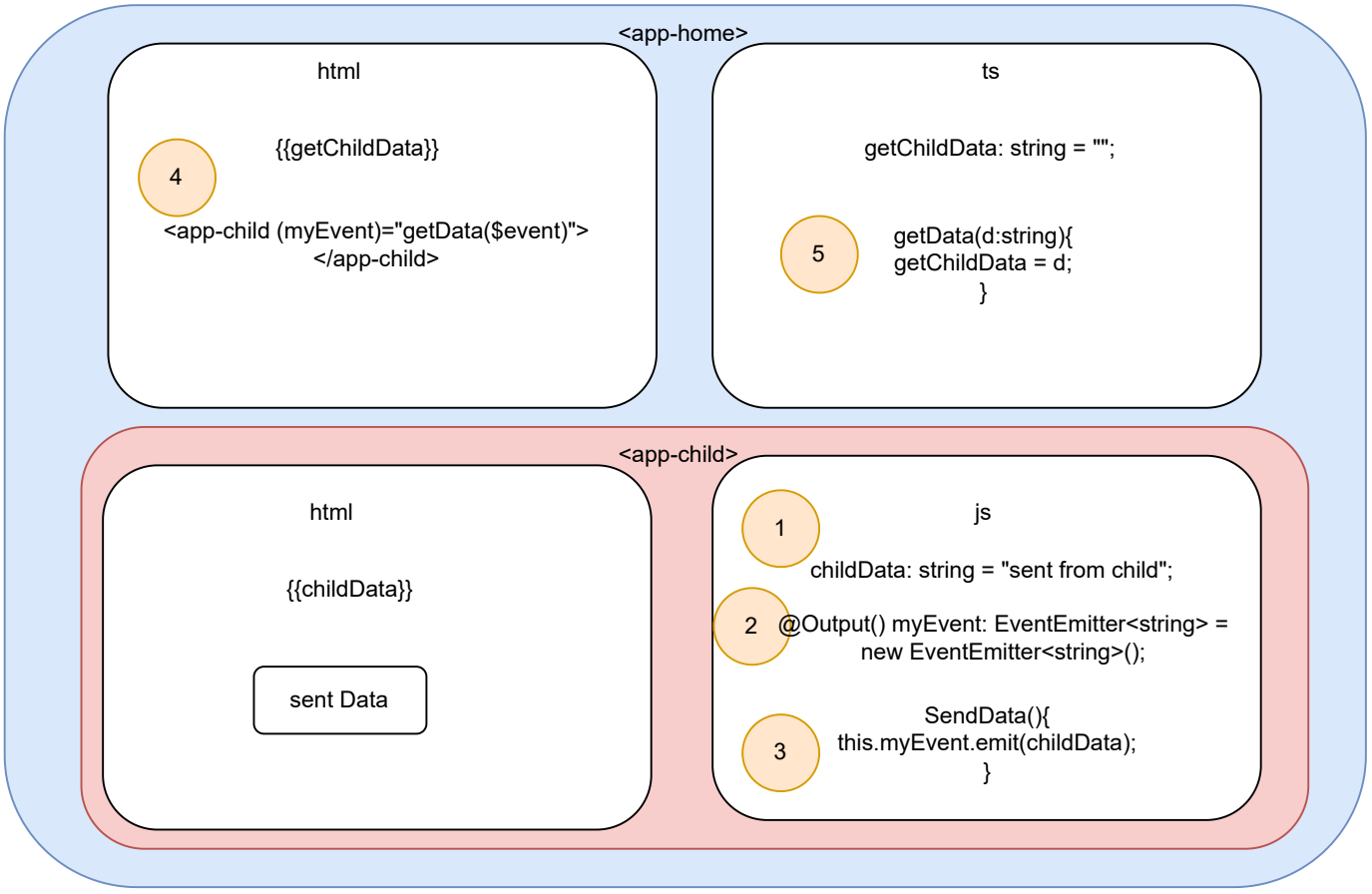
parent to child

1- @Input() varable in child
2- variable contain data in parent
3- in parent html
<app-child [inData] = "parentVar">
bind parent variable with child input

Generate Interface

ng g i IStudent

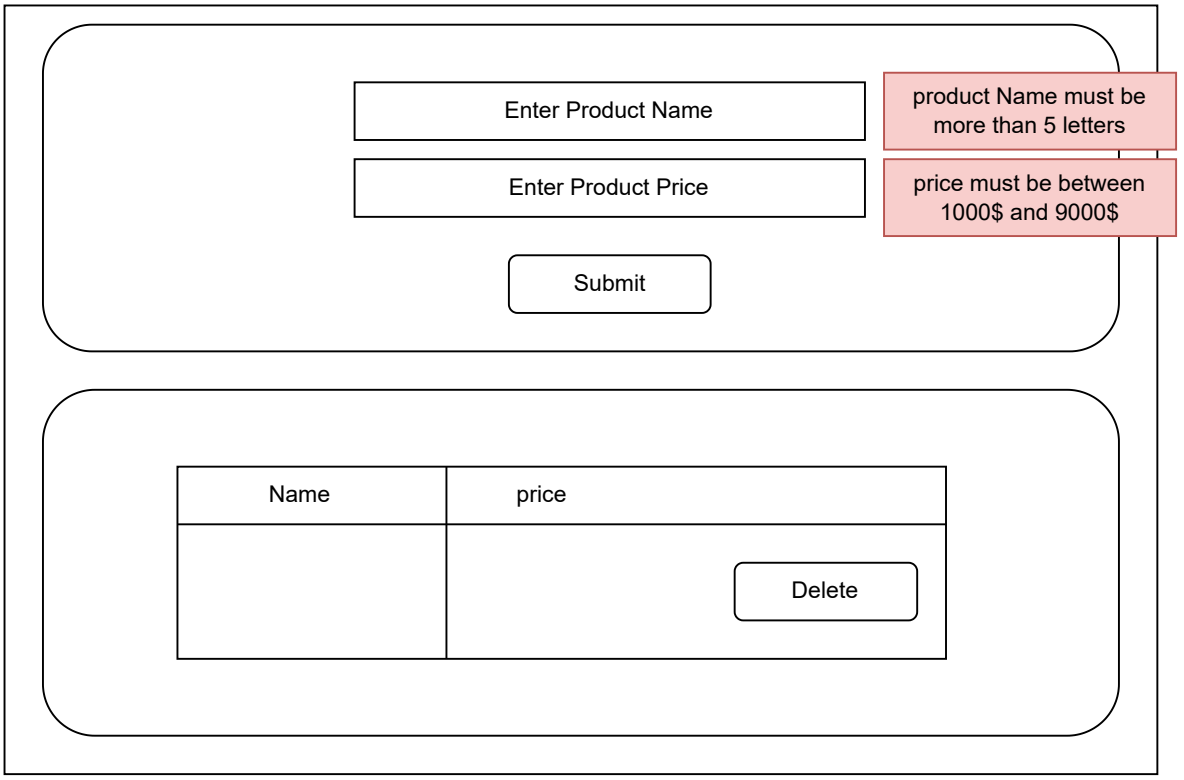
Data from child to parent



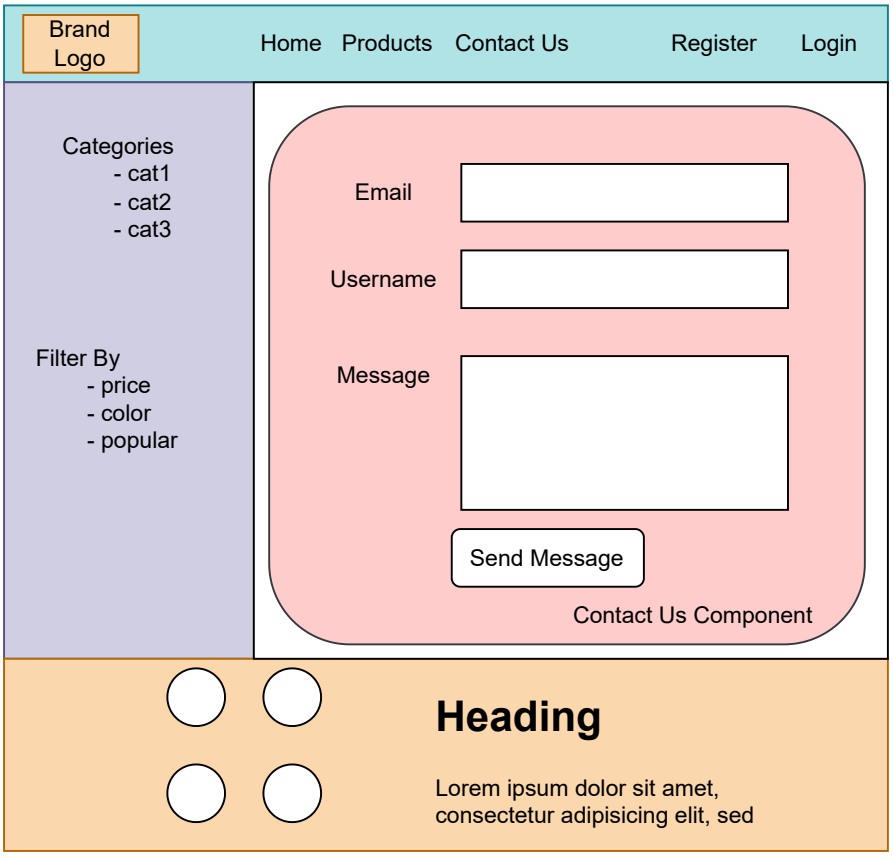
child to parent

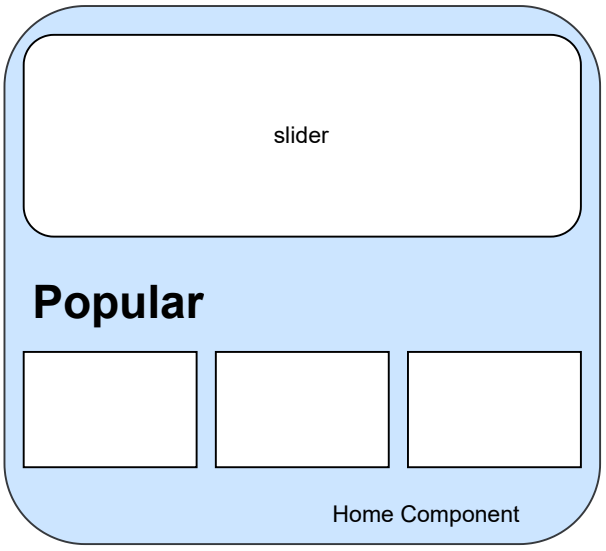
1- variable contain data
2- @output() event to send data outside component
3- think about emitting event
(when click on button || inside ngOnInit Function)

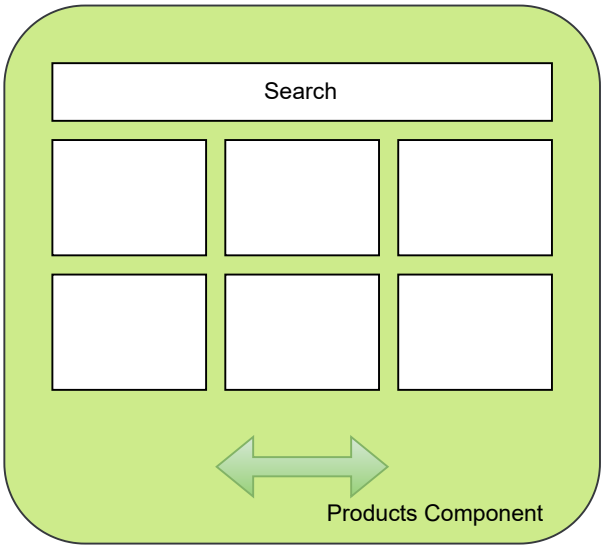
4 - in parent component:
<app-child (myEvent)="parentFunction($event)">
bind on event to get data
5- implement parent function and save data in component variable



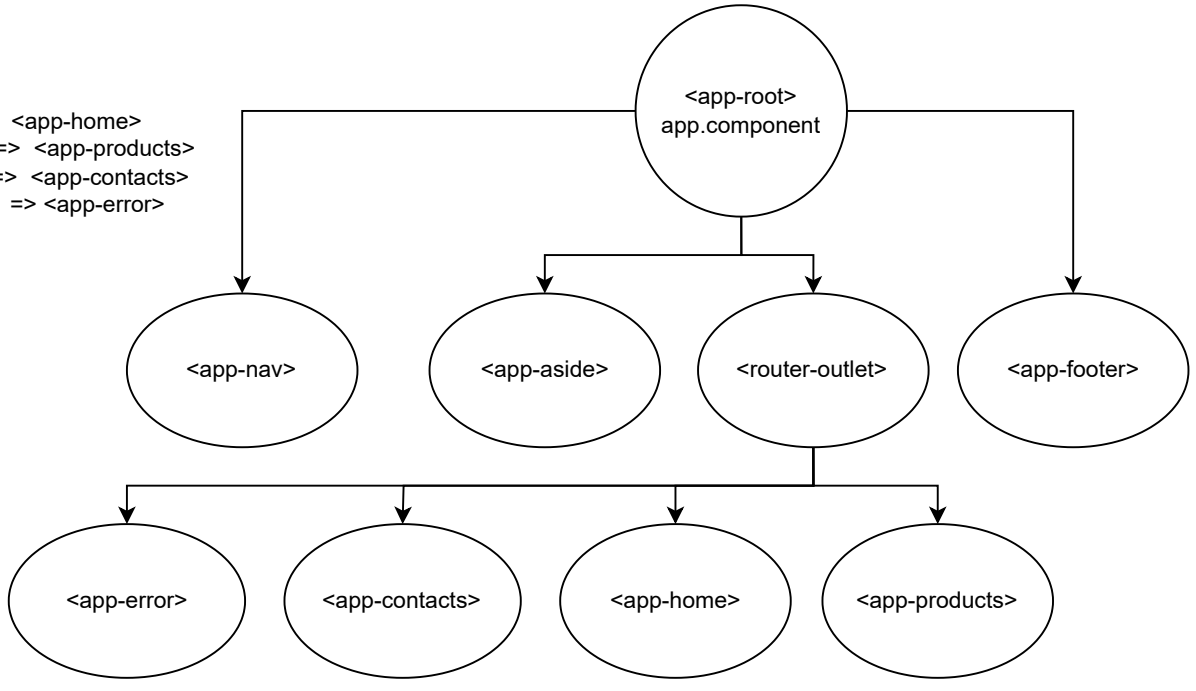
Day 4 (Routing)







URL path
http://localhost:4200/ => <app-home>
http://localhost:4200/products => <app-products>
http://localhost:4200/contacts => <app-contacts>
http://localhost:4200/fds/ffsdf => <app-error>



Dependency

Dependency Injection

```
class A{

B b;

public A(B _b){
b = _b;
}

}
```

```
class B{


}
```

```
A a = new A(new B());
```

ProductService

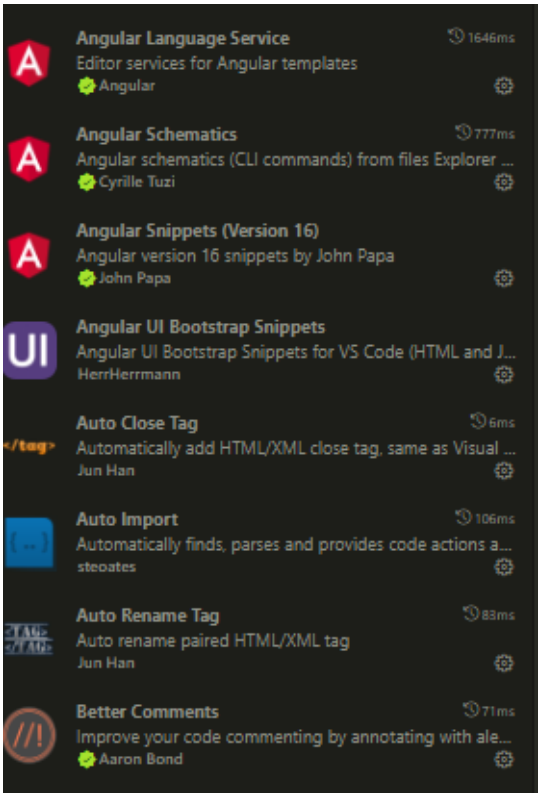
get all
get by id
add
edit
delete

Router Services

get data from routes

CLI Commands

ng g c componentName --skip-test
ng g s serviceName --skip-tests
ng g i interfaceName



Day 5

Reactive Forms

1- add ReactiveFormsModule in imports in app.module
2- in .ts file => create FormGroup inside it create FormControl for each input
3- for each FormControl   add default values and validations
4- bind formGroup with your form

```
<form class="row g-3" [formGroup]="LoginForm">
    formControlName="email"   on Inputs
```

5- use LoginForm for validations
6- submit form

Create Fake APi (json-server)

1-install json-server

```
npm i json-server
```

2- npm start => json-server --watch -p 3005 data.json
3- create data.json file with data you need in your app

Connect on api

1-create service   => for each endpoint
2- import HttpClientModule   => app.module
3- build service (getall, getById, add, edit, delete)

Observables

next:

error:

Complete:

```
Subscripe({
next: (data) => {},
error: (error) => {},
complete: () => {}
})

ngOnDestroy(){
unsubscripe();
}
```

Guards

1- create guard => ng g g authentication
2- inject service in guard inject(service) => return boolean;
3- app-routing => {

```
  path: 'products/details/:id',
  component: ProductDetailsComponent,
  canActivate: [authenticationGuard],
}
```

Task: Day6 => Form

Ecommerce

products => user(R) , Admin (CRUD)

user can add product to cart