Angular

TA Jo and Lucas

Creating Angular project

- Install Angular globally
  - `npm install -g @angular/cli`

- Create a new Angular project
  - `ng new project-name`
  - `cd project-name`
  - `npm install`
  - `npm start  / ng serve`

Creating a New Component

- Using an ng command:
  - ng generate component component_name
  - ng g c component_name

- Add routing rule for the component
  - Import component at app-routing.module.ts

```
import { Page1Component } from './page1/page1.component';



const routes: Routes = [
  {path: 'page1', component: Page1Component}
];
```

Capturing input in a component

- Import FormsModule
- Forms in Angular are robust, but also can get a bit complex
- https://angular.io/guide/forms

```
import { FormsModule } from '@angular/forms';
[...]
@NgModule({
  imports: [
    [...],
    FormsModule
  ],
  [...]
})
```

Capturing input in a component

- Html input component mapped to typescript class attribute
- [(ngModel)]: two-way binding (property + event)

```
<input class="my-2 form-control" id="item" type="text" placeholder="item" [(ngModel)]='item'>
```

```
.....
export class GroceriesInputComponent implements OnInit {


  item:String =‘ ';
.....
```

Capturing input in a component

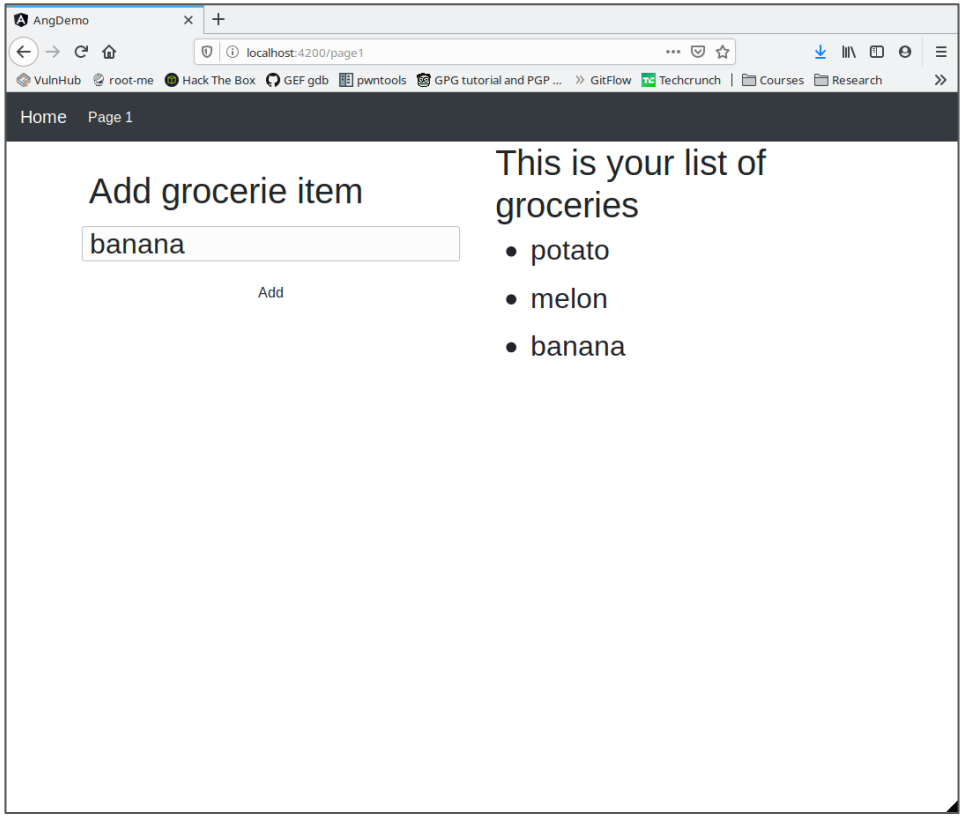
- FormControls need attributes to have names, such as html inputs or selects
- Add name to input

```
<input class="my-2 form-control" id="item" type="text" placeholder="item" [(ngModel)]='item' name='inputItem'>
```

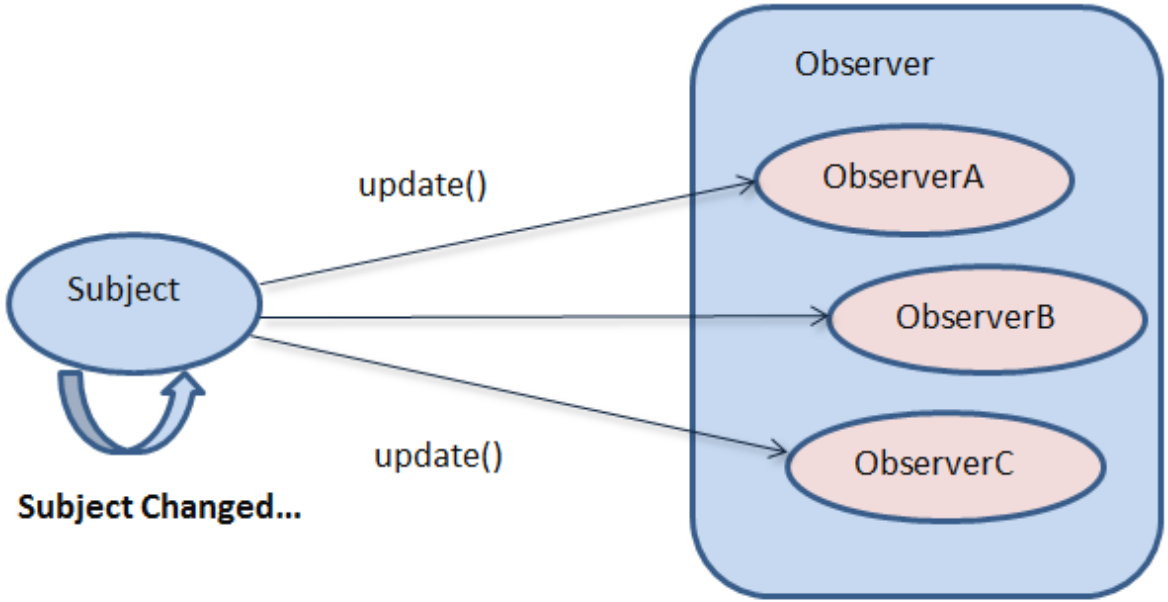
Service

- ng command
  - ng generate service service_name
- Injectable into components
- Have benefits in reusability and maintainability

Live Coding Demo



Observer Pattern



EventEmitter

- An observer pattern
- Used in services to signal data change
- Subscribed methods know when an event is triggered

```
...
import { EventEmitter } from '@angular/core';

@Injectable()
export class GroceriesService {
  groceriesChanged = new EventEmitter<String[]>();
….
this.groceriesChanged.emit(changed data...);
...
```