COMP 431/531: Web Development

Lecture 9: Angular Services

Mack Joyner (mjoyner@rice.edu)

https://www.clear.rice.edu/comp431

Announcements & Reminders

- HW #3 (JavaScript Game) is due today at 11:59pm

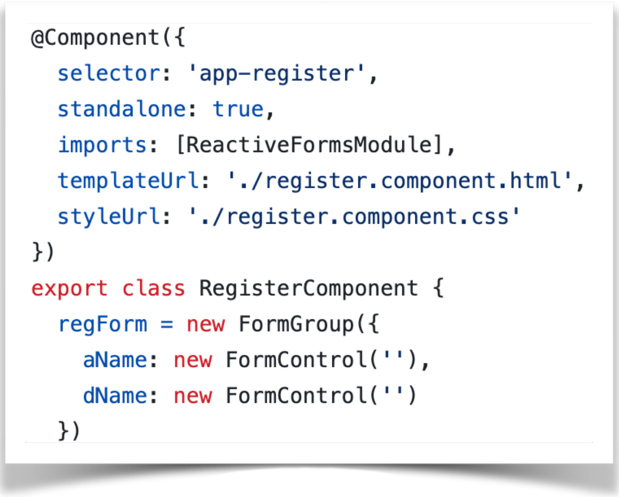
classroom hw3 repo: https://classroom.github.com/a/wZT10Yqv

- Quiz #2 (Events, Storage, Arrays) is due Thursday, Oct. 2nd at 11:59pm

Reactive Forms

- Angular supports reactive forms
- Import *ReactiveFormsModule* (@angular/forms) in component
- Import *FormControl*, *FormGroup* (@angular/forms) in component
- Access form control *value*

Component

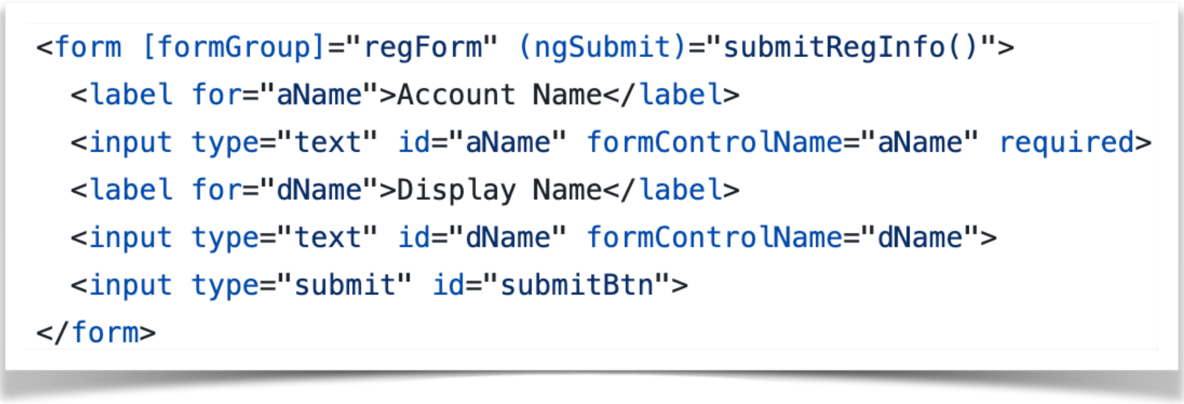


```
@Component({
  selector: 'app-register',
  standalone: true,
  imports: [ReactiveFormsModule],
  templateUrl: './register.component.html',
  styleUrl: './register.component.css'
})
export class RegisterComponent {
  regForm = new FormGroup({
    aName: new FormControl(''),
    dName: new FormControl('')
  })
```

Reactive Forms

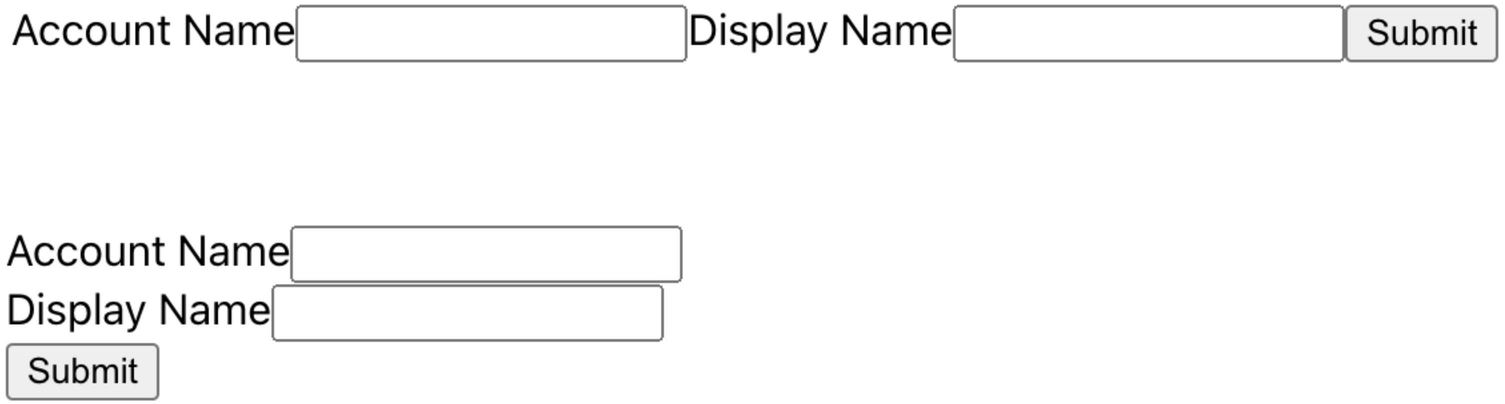
- Angular supports reactive forms
- Import *ReactiveFormsModule* (@angular/forms) in component
- Import *FormControl*, *FormGroup* (@angular/forms) in component
- Access form control *value*
- Input formControlName attribute specifies form group field
- Component reactive form field updates with input change

Template



```
<form [formGroup]="regForm" (ngSubmit)="submitRegInfo()">
  <label for="aName">Account Name</label>
  <input type="text" id="aName" formControlName="aName" required>
  <label for="dName">Display Name</label>
  <input type="text" id="dName" formControlName="dName">
  <input type="submit" id="submitBtn">
</form>
```

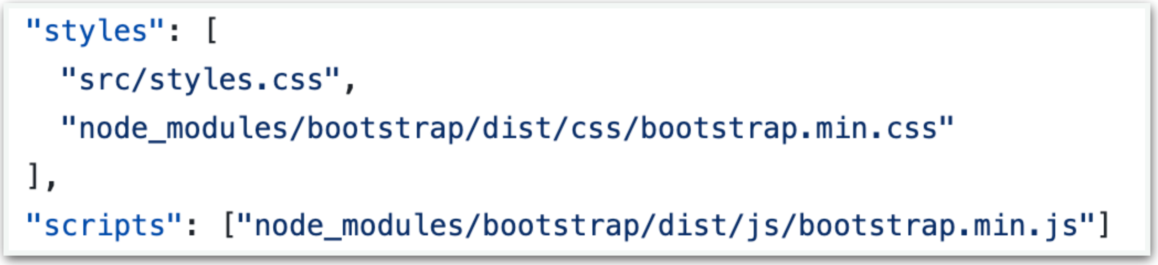
Unstyled Form



Bootstrap in Angular

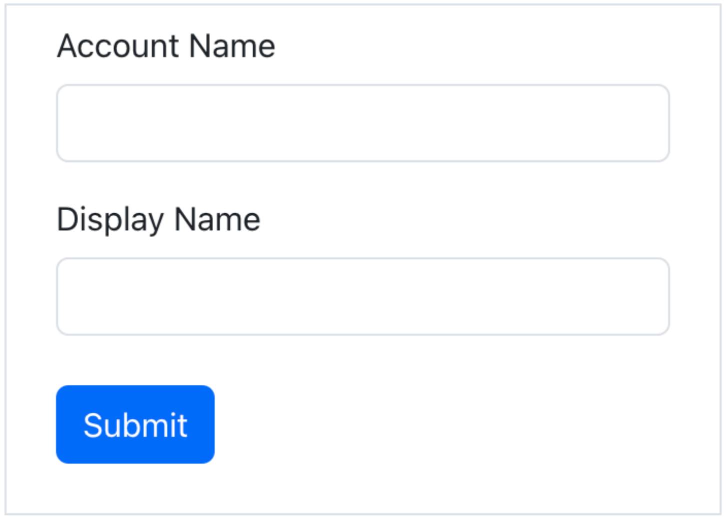
```
>> npm install bootstrap
```

angular.json



```
"styles": [
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": ["node_modules/bootstrap/dist/js/bootstrap.min.js"]
```

Styled Form with Bootstrap



Separation of Concerns

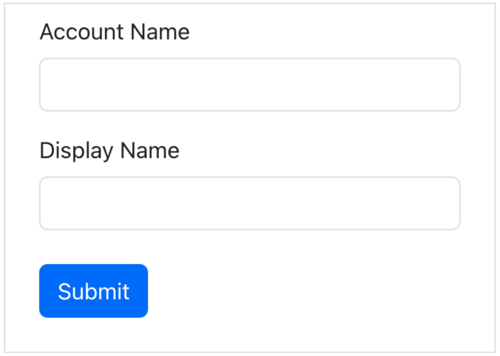
In computer science, **separation of concerns** (SoC) is a design principle for **separating** a computer program into distinct sections, such that each section addresses a separate **concern**. A **concern** is a set of information that affects the code of a computer program. -Wikipedia

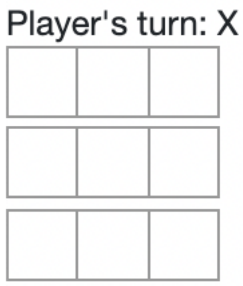
- Simplify development
- Increase maintainability
- Improve reusability
- Parallelize development
- Promotes encapsulation

Angular Components with Services

- Components are presentational, should be kept lean
- Data comes through Services
  - fetching, storing data
  - user input validation
  - logging debug information
- Incorporate service into component using dependency injection

Multiple Components - Global Data



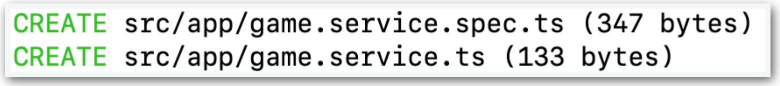


How do components communicate with each other to share data?

Generating Angular Service

Generating a new service is also fast:

```
>> cd tictactoe

>> ng generate service game
```



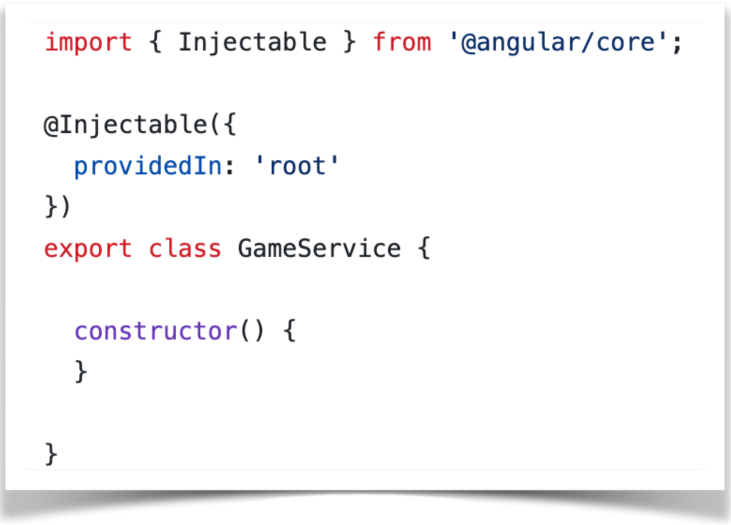
```
CREATE src/app/game.service.spec.ts (347 bytes)
CREATE src/app/game.service.ts (133 bytes)
```

If no directory is specified, service is placed in app (root)

Dependency Injection (Service)

- A class with injector decorator
- Associated with root module
- Add to any root module component
- Separate logic from view
  - fetching, storing data
  - user input validation
  - logging debug information

Service

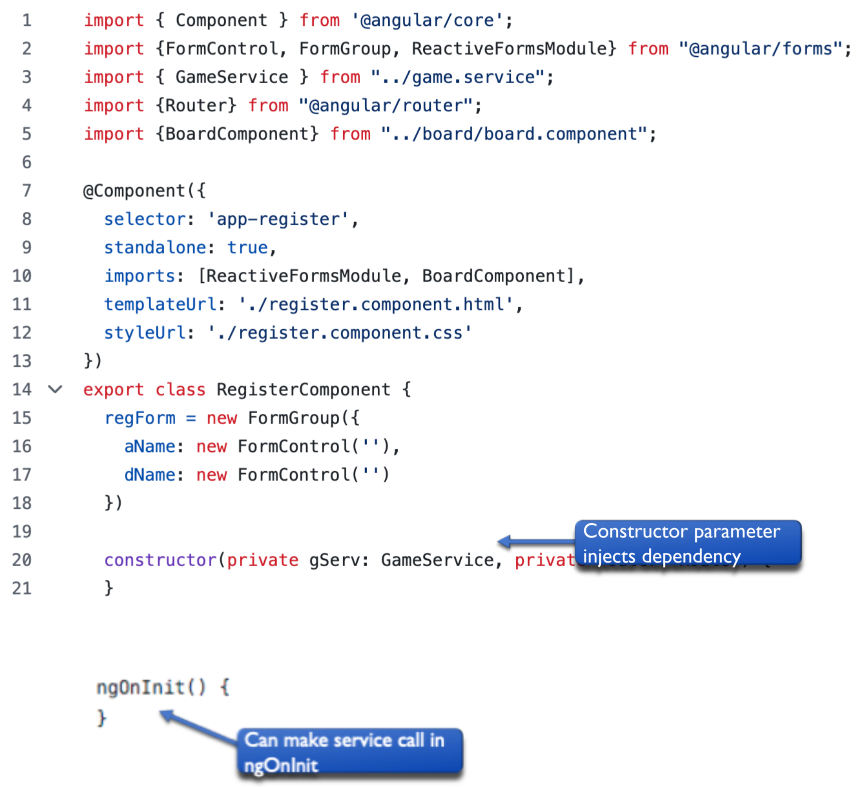


```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class GameService {

  constructor() {
  }

}
```

Dependency Injection (Service)

- A class with injector decorator
- Associated with root module
- Add to any root module component
- Separate logic from view
  - fetching, storing data
  - user input validation
  - logging debug information
- Service instance injected into component
  - Not accessible to template
  - Available to any method (i.e. *this.gServ.func()*)

Component



```
import { Component } from '@angular/core';
import {FormControl, FormGroup, ReactiveFormsModule} from "@angular/forms";
import { GameService } from "../game.service";
import {Router} from "@angular/router";
import {BoardComponent} from "../board/board.component";

@Component({
  selector: 'app-register',
  standalone: true,
  imports: [ReactiveFormsModule, BoardComponent],
  templateUrl: './register.component.html',
  styleUrl: './register.component.css'
})
export class RegisterComponent {
  regForm = new FormGroup({
    aName: new FormControl(''),
    dName: new FormControl('')
  })

  constructor(private gServ: GameService, privat
  }


 ngOnInit() {
 }
```

Constructor parameter injects dependency
Can make service call in ngOnInit

GameService Example

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class GameService {

/**
 * Determine if a player won the game.
 */
wonGame(board: string[]): boolean {
  let winnerFound = false;
  const winCombos = [ [0, 1, 2], [0, 3, 6], [0, 4, 8],
    [1, 4, 7],
    [2, 5, 8], [2, 4, 6],
    [3, 4, 5],
    [6, 7, 8]];

  // loop through all the winning combinations
  winCombos.forEach(combo => {
    let win = true;
    // check if player made a move in each board location of the winning combination
    combo.forEach(pos =>  win = win && (board[pos] === this.playerTurn));
    if (win) {
      winnerFound = true;
    }
  });

  return winnerFound;
}
```

https://www.clear.rice.edu/comp431/sample/tictactoe/angular/game.service.ts

Router Navigate

- Router class provides the ability to manipulate URL (alternative: routerLink)
- Already have RouterModule, Routes in app-routing
- Component imports the Router class
- Inject router dependency into component
- Router has access to root module routes
- Change the URL by using *navigate*
-

register.component.ts

```
import { Component } from '@angular/core';
import {FormControl, FormGroup, ReactiveFormsModule} from "@angular/forms";
import { GameService } from "../game.service";
import {Router} from "@angular/router";
import {BoardComponent} from "../board/board.component";

@Component({
  selector: 'app-register',
  standalone: true,
  imports: [ReactiveFormsModule, BoardComponent],
  templateUrl: './register.component.html',
  styleUrl: './register.component.css'
})
export class RegisterComponent {
  regForm = new FormGroup({
    aName: new FormControl(''),
    dName: new FormControl('')
  })

  constructor(private gServ: GameService, private router: Router) {
  }
  submitRegInfo() {
    this.router.navigate(['/board']);
  }
}
```

Component Communication

- Components can also communicate via parent-child relationship
- Components specific which field(s) may be set @Input() decorator
- Parent component sets the field for child component in template

register.component.html

```
<app-board player1="Joe">

</app-board>
```