Angular

II Lecture

KindGeek

Directives

KindGeek School

Directives

- Component - directive with a template.
- Attribute directives - directives that change the behavior of a component or element but don't affect the template
- Structural directives - directives that change the behavior of a component or element by affecting how the template is rendered

NgStyle Directive

```
@Component({
  selector: 'app-style-example',
  template: `
    <p style="padding: 1rem"
      [ngStyle]="{
        'color': 'red',
        'font-weight': 'bold',
        'borderBottom': borderStyle
      }">
      <ng-content></ng-content>
    </p>
  `
})
export class StyleExampleComponent {
  borderStyle = '1px solid black';
}
```

```
@Component({
  selector: 'app-style-example',
  template: `
    <p style="padding: 1rem"
      [ngStyle]="alertStyles">
      <ng-content></ng-content>
    </p>
  `
})
export class StyleExampleComponent {
  borderStyle = '1px solid black';

  alertStyles = {
    'color': 'red',
    'font-weight': 'bold',
    'borderBottom': this.borderStyle
  };
}
```

KindGeek School

NgClass Directive

1. Binding a string

```
@Component({
  selector: 'app-class-as-string',
  template: `
    <p ngClass="centered-text underlined" class="orange">
      <ng-content></ng-content>
    </p>
  `,
  styles: [`
    .centered-text {
      text-align: center;
    }

    .underlined {
      border-bottom: 1px solid #ccc;
    }

    .orange {
      color: orange;
    }
  `]
})
export class ClassAsStringComponent {
}
```

KindGeek School

NgClass Directive

2. Binding an array

```
@Component({
  selector: 'app-class-as-array',
  template: `
    <p [ngClass]="['warning', 'big']">
      <ng-content></ng-content>
    </p>
  `,
  styles: [`
    .warning {
      color: red;
      font-weight: bold;
    }

    .big {
      font-size: 1.2rem;
    }
  `]
})
export class ClassAsArrayComponent {
}
```

KindGeek School

NgClass Directive

3. Binding an object

```
@Component({
  selector: 'app-class-as-object',
  template: `
    <p [ngClass]="{ card: true, dark: false, flat: flat }">
      <ng-content></ng-content>
      <button type="button" (click)="flat=!flat">Toggle Flat</button>
    </p>`,
  styles: [`
    .card {
      border: 1px solid #eee;
      padding: 1rem;
      margin: 0.4rem;
      font-family: sans-serif;
      box-shadow: 2px 2px 2px #888888;
    }
    .dark {
      background-color: #444;
      border-color: #000;
      color: #fff;
    }
    .flat {
      box-shadow: none;
    }
  `]
})
export class ClassAsObjectComponent {
  flat: boolean = true;
}
```

KindGeek School

NgIf Directive

```
@Component({
  selector: 'app-root',
  template: `
    <button type="button" (click)="toggleExists()">Toggle Component</button>
    <hr>
    <app-if-example *ngIf="exists">
      Hello
    </app-if-example>
  `
})
export class AppComponent {
  exists = true;

  toggleExists() {
    this.exists = !this.exists;
  }
}
```

KindGeek School

NgFor Directive

```
@Component({
  selector: 'app-root',
  template: `
    <app-for-example *ngFor="let episode of episodes" [episode]="episode">
      {{episode.title}}
    </app-for-example>
  `
})
export class AppComponent {
  episodes = [
    { title: 'Winter Is Coming', director: 'Tim Van Patten' },
    { title: 'The Kingsroad', director: 'Tim Van Patten' },
    { title: 'Lord Snow', director: 'Brian Kirk' },
    { title: 'Cripples, Bastards, and Broken Things', director: 'Brian Kirk' },
    { title: 'The Wolf and the Lion', director: 'Brian Kirk' },
    { title: 'A Golden Crown', director: 'Daniel Minahan' },
    { title: 'You Win or You Die', director: 'Daniel Minahan' },
    { title: 'The Pointy End', director: 'Daniel Minahan' }
  ];
}
```

KindGeek School

NgFor Directive

Local Variables:

`index` - position of the current item in the iterable starting at 0

`first` - true if the current item is the first item in the iterable

`last` - true if the current item is the last item in the iterable

`even` - true if the current index is an even number

`odd` - true if the current index is an odd number

```
@Component({
  selector: 'app-root',
  template: `
    <app-for-example
      *ngFor="let episode of episodes; let i = index; let isOdd = odd"
      [episode]="episode"
      [ngClass]="{ odd: isOdd }">
      {{i+1}}. {{episode.title}}
    </app-for-example>
    <h2>Desugared</h2>
    <template ngFor [ngForOf]="episodes" let-episode let-i="index"
let-isOdd="odd">
      <for-example [episode]="episode" [ngClass]="{ odd: isOdd }">
        {{i+1}}. {{episode.title}}
      </for-example>
    </template>
  `
})
```

KindGeek School

NgFor Directive

trackBy

```
@Component({
  selector: 'app-root',
  template: `
  <button
    (click)="addOtherEpisode()"
    [disabled]="otherEpisodes.length === 0">
    Add Episode
  </button>
  <app-for-example
    *ngFor="let episode of episodes;
    let i = index; let isOdd = odd;
    trackBy: trackById" [episode]="episode"
    [ngClass]="{ odd: isOdd }">
    {{episode.title}}
  </app-for-example>
  `
})
export class AppComponent {
  otherEpisodes = [
    { title: 'Two Swords', director: 'D. B. Weiss', id: 8 },
    { title: 'The Lion and Rose', director: 'Alex Graves', id: 9 },
    { title: 'Breaker', director: 'Michelle MacLaren', id: 10 },
    { title: 'Oathkeeper', director: 'Michelle MacLaren', id: 11 }]

  episodes = [
    { title: 'Winter Is Coming', director: 'Tim Patten', id: 0 },
    { title: 'The Kingsroad', director: 'Tim Van Patten', id: 1 },
    { title: 'Lord Snow', director: 'Brian Kirk', id: 2 },
    { title: 'Cripples, Bastards', director: 'Brian Kirk', id: 3 },
    { title: 'The Wolf and Lion', director: 'Brian Kir', id: 4 }
  ];

  addOtherEpisode() {
    // We want to create a new object reference for sake of example
    let episodesCopy = JSON.parse(JSON.stringify(this.episodes))
    this.episodes=[...episodesCopy,this.otherEpisodes.pop()];
  }
  trackById(index: number, episode: any): number {
    return episode.id;
  }
}
```

KindGeek School

NgSwitch Directives

```
@Component({
  selector: 'app-root',
  template: `
    <div class="tabs-selection">
      <app-tab [active]="isSelected(1)" (click)="setTab(1)">Tab 1</app-tab>
      <app-tab [active]="isSelected(2)" (click)="setTab(2)">Tab 2</app-tab>
      <app-tab [active]="isSelected(3)" (click)="setTab(3)">Tab 3</app-tab>
    </div>
    <div [ngSwitch]="tab">
      <app-tab-content *ngSwitchCase="1">Tab content 1</app-tab-content>
      <app-tab-content *ngSwitchCase="2">Tab content 2</app-tab-content>
      <app-tab-content *ngSwitchCase="3"><app-tab-3></app-tab-3></app-tab-content>
      <app-tab-content *ngSwitchDefault>Select a tab</app-tab-content>
    </div>
  `
})
export class AppComponent {
  tab: number = 0;
  setTab(num: number) {
    this.tab = num;
  }
  isSelected(num: number) {
    return this.tab === num;
  }
}
```

KindGeek School

Component Lifecycle

```
@Component({
  selector: 'app-root',
  template: `
    <template ngFor [ngForOf]="[1,2,3,4,5,6]" let-item>
      <div *ngIf="item > 3">
        {{item}}
      </div>
    </template>
  `
})
```

KindGeek School

KindGeek School

Components

Access Child Components From the Template

```
<rio-profile #profile></rio-profile>
My name is {{ profile.name }}
```

```
@Component({
  selector: 'rio-profile',
  templateUrl: 'app/profile.component.html'
})
export class ProfileComponent {
  name = 'John Doe';
}
```

KindGeek School

Projection

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

@Component({
  selector: 'rio-child',
  template: `
    <div style="border: 1px solid blue; padding: 1rem;">
      <h4>Child Component</h4>
      <ng-content></ng-content>
    </div>
  `
})
export class ChildComponent {
}
```

```
<rio-child>
    <p>My <i>projected</i> content.</p>
  </rio-child>
```

KindGeek School

Component Lifecycle

- `ngOnChanges` - called when an input binding value changes
- `ngOnInit` - after the first ngOnChanges
- `ngDoCheck` - after every run of change detection
- `ngAfterContentInit` - after component content initialized
- `ngAfterContentChecked` - after every check of component content
- `ngAfterViewInit` - after component's view(s) are initialized
- `ngAfterViewChecked` - after every check of a component's view(s)
- `ngOnDestroy` - just before the component is destroyed

Accessing Child Component Classes

```
@Component({
    selector: 'app-root',
    template: `
    <app-alert #first ok="Next" (close)="showAlert(2)">
      Step 1: Learn angular
    </app-alert>
    <app-alert ok="Next" (close)="showAlert(3)">
      Step 2: Love angular
    </app-alert>
    <app-alert ok="Close">
      Step 3: Build app
    </app-alert>
      <button (click)="showAlert(1)">Show steps</button>`
})
export class AppComponent {
  @ViewChild('first') alert: AlertComponent;
  @ViewChildren(AlertComponent) alerts: QueryList<AlertComponent>;
  // ...
}
```

View children will not be set until the `ngAfterViewInit` lifecycle hook is called.

+

KindGeek School

View Encapsulation

- `Emulated (default)` - styles from main HTML propagate to the component. Styles defined in this component's @Component decorator are scoped to this component only.
- `Native` - styles from main HTML do not propagate to the component. Styles defined in this component's @Component decorator are scoped to this component only.
- `None` - styles from the component propagate back to the main HTML and therefore are visible to all components on the page. Be careful with apps that have None and Native components in the application. All components with None encapsulation will have their styles duplicated in all components with Native encapsulation.

```
@Component({
// ...
encapsulation:
ViewEncapsulation.None,
styles: [
  // ...
]
})
export class HelloComponent {
// ...
}
```

KindGeek School

ElementRef

```
import { AfterContentInit, Component, ElementRef }from '@angular/core';

@Component({
    selector: 'app-root',
    template: `
    <h1>My App</h1>
    <pre>
      <code>{{ node }}</code>
    </pre>
  `
})
export class AppComponent implements AfterContentInit {
  node: string;

  constructor(private elementRef: ElementRef) { }

  ngAfterContentInit() {
    const tmp = document.createElement('div');
    const el = this.elementRef.nativeElement.cloneNode(true);

    tmp.appendChild(el);
    this.node = tmp.innerHTML;
  }

}
```

KindGeek School

KindGeek School

Home task

Create SPA blog using bootstrap theme:
https://blackrockdigital.github.io/startbootstrap-clean-blog/