Angular 2 Component & Metadata & Template

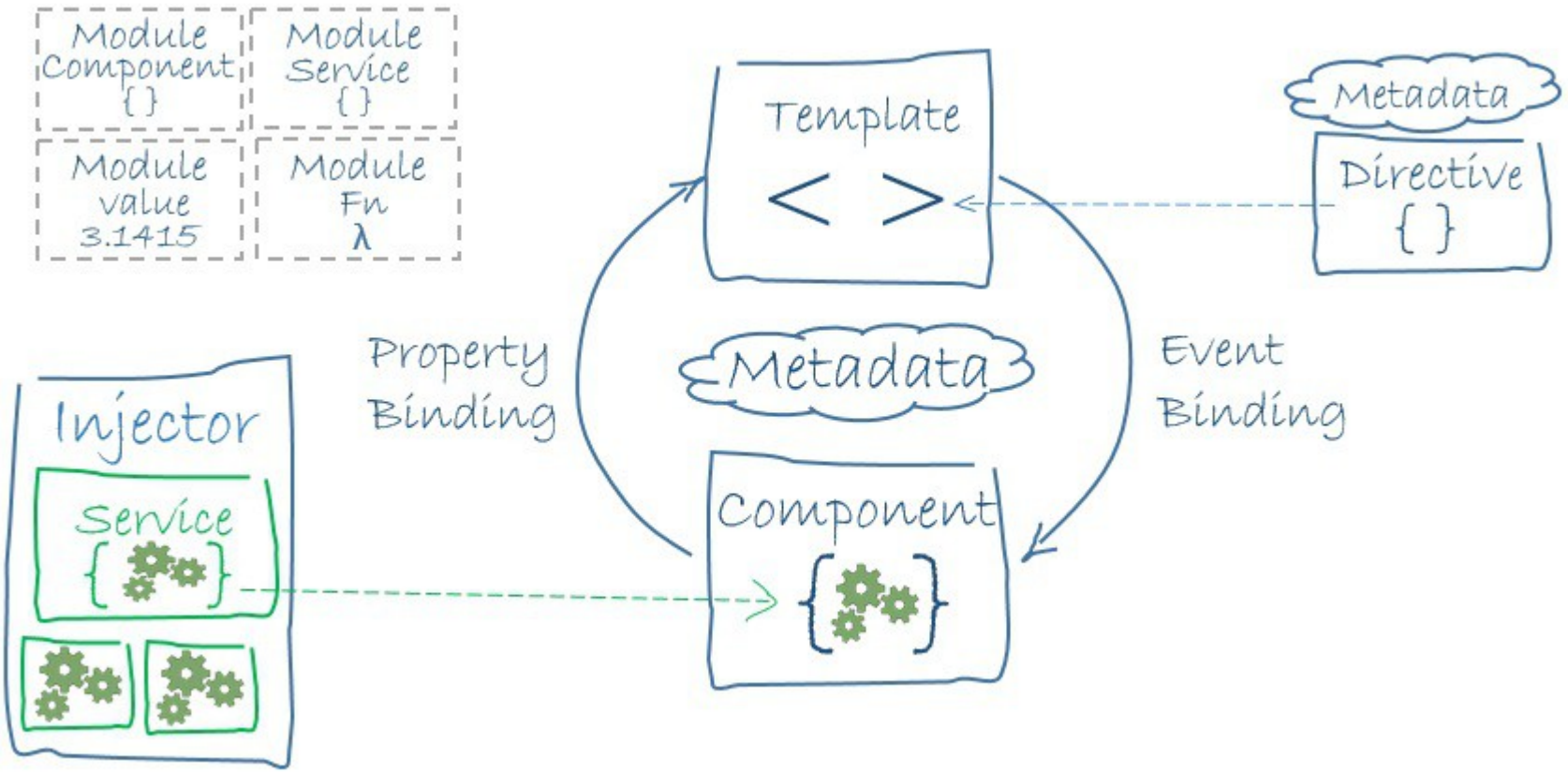
Sang Shin
JPassion.com
"Code with Passion!"

Topics

- Building blocks of Angular 2 application
  - Component
  - Template
  - Metadata
- Component tree
- View encapsulation
- <ng-content>
- Angular 2 app bootstrapping

Building Blocks of Angular 2 Application

Building Blocks of Angular 2 Application



source: https://angular.io/docs/ts/latest/guide/architecture.html

Component

What is Angular 2 Component?

- Components are fundamental building blocks of Angular 2 applications
  - Writing Angular 2 app is basically a process of building components into a component tree
- We can extend the HTML vocabulary with components as new elements (and directives as new attributes)
- A component is made of
  - Class
  - Metadata
  - Template
- A component uses
  - Services (providers)
  - Other components (or directives)

Example Component (written in TypeScript)

```
@Component({
  selector:    'hero-list',
  templateUrl: 'app/hero-list.component.html',
  styleUrls: ['./hero-list.componentt.css']
})
```

metadata

```
export class HeroListComponent implements OnInit {

  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

class

Component creates a view with new “hero-list” element tag

Template

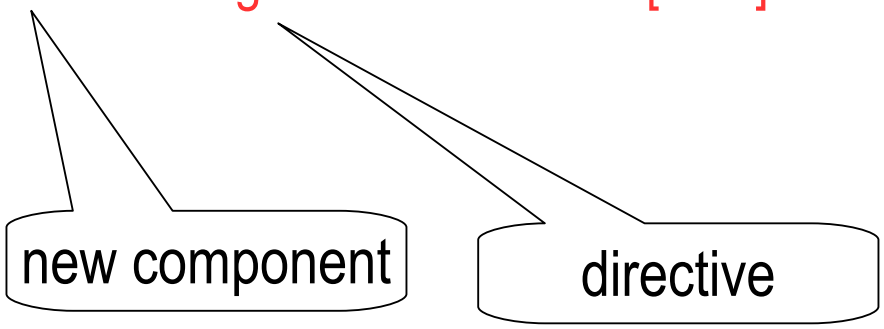
What is a Template?

- Define a component's view
- A template is a form of HTML that tells Angular how to render the component

```
<!--Simplest template -->
<h1>
  {{title}}
</h1>
```

Template Example (app/hero-list.component.html)

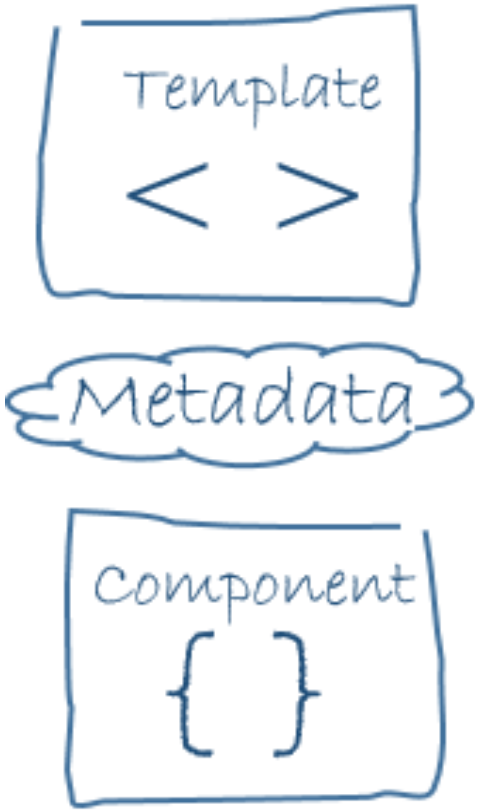
```
<h2>Hero List</h2>
<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes" (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>
<hero-detail *ngIf="selectedHero" [hero]="selectedHero"></hero-detail>
```



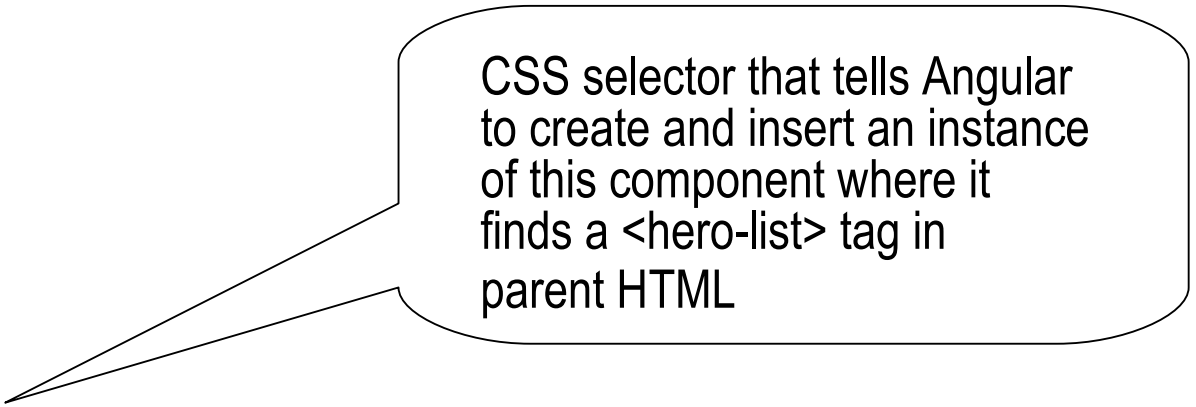
Metadata

What is Metadata?

- Metadata is a decorator decorating a class
- Metadata tells Angular how to process a class
  - A class itself is just a class
  - A class is not a component until you tell Angular about it through metadata
- The template, metadata, and component class together describe a view



Metadata Example (app/hero-list.component.html)




```
@Component({
  selector:    'hero-list',
  templateUrl: 'app/hero-list.component.html',
  styleUrls: ['./hero-list.component.css']
})
export class HeroListComponent implements OnInit {
/* . . . */
}
```

Lab #1: Modify AppComponent

- src/app directory contains the code
  - app.component.ts file contains AppComponent class
- Change “title” value of the AppComponent to “my first angular 2 app”
  - Observe that the browser gets refreshed automatically with the change
- Change <h1> style of the AppComponent – add the following to the *app.component.css* file

```
h1 {
   color: red
}
```

- Optional lab
  - Add <h2> element and change the style of it

Lab #1: Modify AppComponent

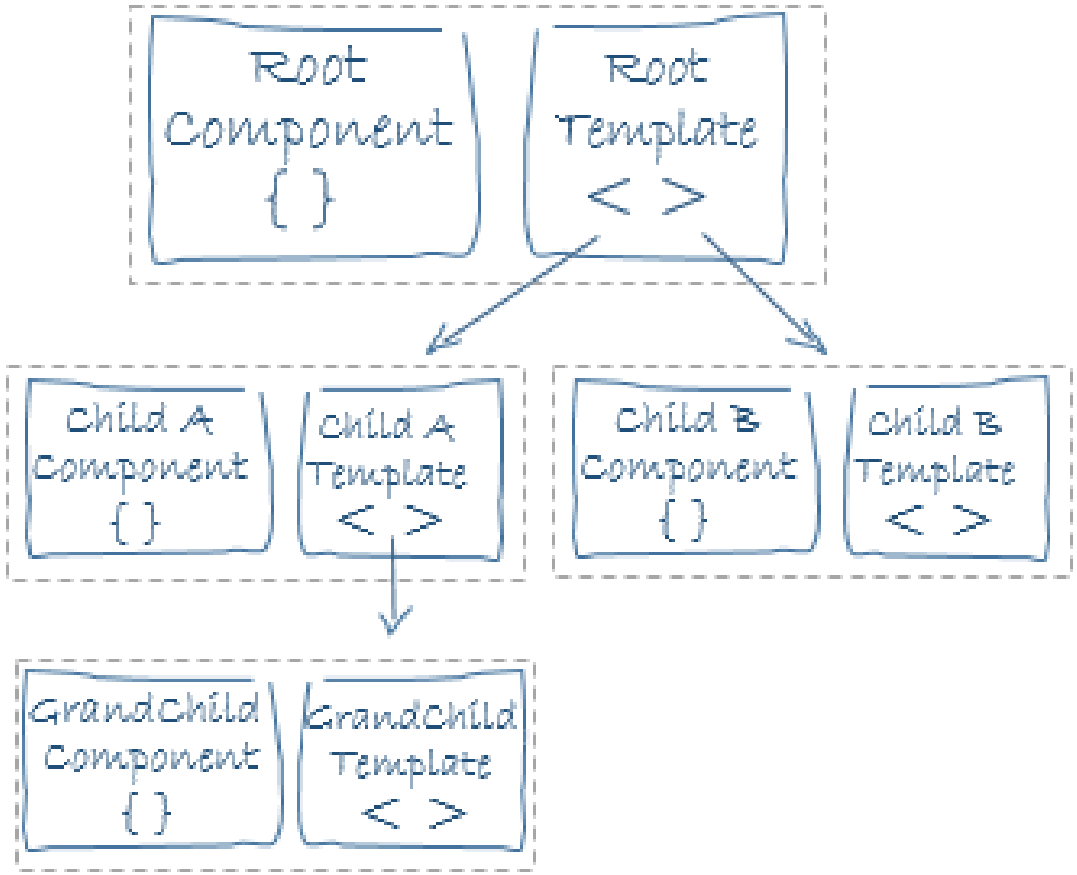
- Try inline template and inline styles

```
@Component({
  selector: 'app-root',
  // templateUrl: './app.component.html',
  template: `
  <!--Simplest template -->
  <h1>
  {{title}}
  </h1>
  `,
  // styleUrls: ['./app.component.css']
  styles: [
    `
     h1 {
       color: blue
     }
    `
  ]
})
export class AppComponent {
  title = 'my first angular 2 app!';
}
```

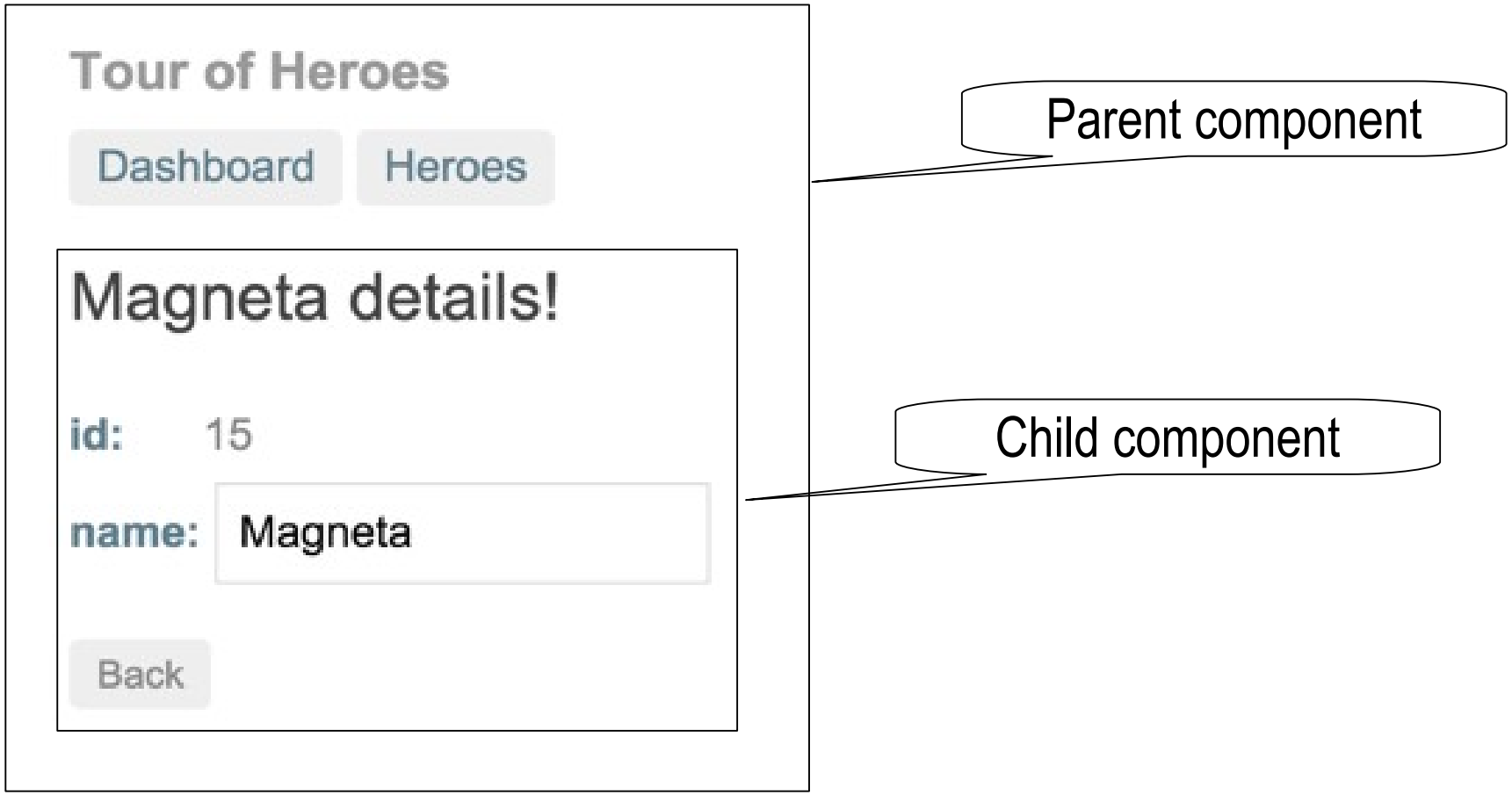
Component Tree (Parent component + Child components)

Components Tree

- A single page in Angular is made of component tree



Component Tree Example



Parent Component and Child Component

- Parent component can use the Child component in its template

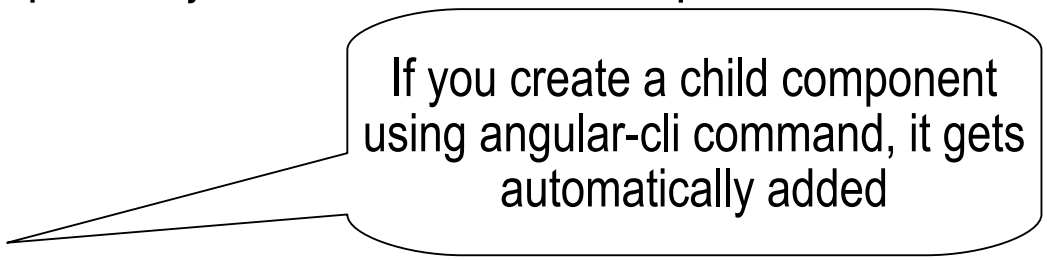
```
<h1>
  {{title}}
</h1>
<app-child></app-child>
```

Parent Component and Child Component

- The Child component needs to declared in the app module (app.module.ts)

```
..
import { ChildComponent } from './child/child.component';

@NgModule({
  declarations: [
    AppComponent,
    ChildComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



Lab #2: Create new Components

- Create a child component
  - ng g component child (It will create component files under “child” directory)
- Create a sibling component
  - cd to the child directory
  - ng g component sibling –flat
- Try different type of selector for the child component
  - How child view gets added to the hosting view – try different selector style such as css class selector

    ```
    selector: '.app-child'
    ```
  - Then change the parent template

    ```
    <div class="app-child">Hello</div>
    ```
- Optional lab
  - Create another child component

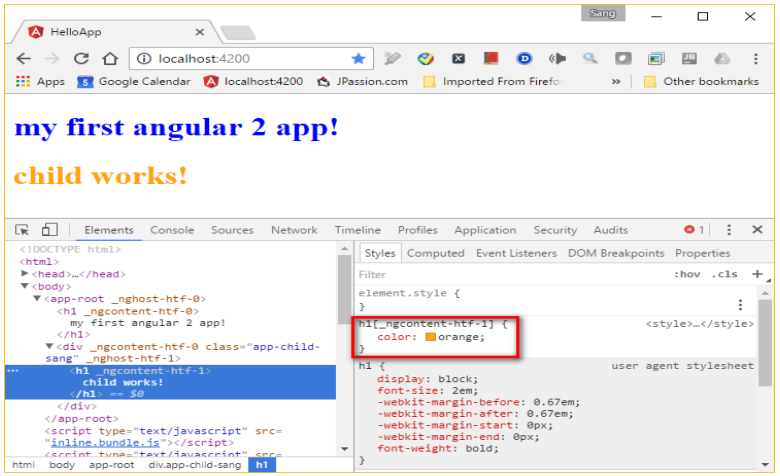
View Encapsulation

View Encapsulation

- View encapsulation defines whether the template and styles defined within a component can affect other part of the application
- In non-Angular world, the scheme of Shadow DOM allows us to hide DOM logic behind elements
  - It enables us to apply scoped styles to elements without them bleeding out to the outer world
- But not every browser supports Shadow DOM so Angular cannot use it
  - Angular 2 emulates Shadow DOM via attaching scope-specific CSS class attributes

Lab #3: View encapsulation

- Observe that a style in one component does not apply to the other component
  - > Style defined for a child component gets applied only to child component while style defined for parent component gets applied to only parent component
- Inspect the element to see how Angular emulates the Shadow DOM via scoped CSS class attributes



<ng-content>

Lab #4: <ng-content>

- It is a way to get contents from external component
  - Useful when the child component functions as a container from the parent component
- Child template

```
<div>
  content in child component
  <ng-content></ng-content>
</div>
```

- Parent template

```
<app-child>
    <h1> hello</h1>
</app-child>
<app-child>
    <h3> world</h3>
</app-child>
```

Angular 2 App Bootstrapping

Angular App Bootstrapping

- App module specifies the root component

```
@NgModule({
  declarations: [
    AppComponent,
    ChildComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- *index.html* contains root component

```
<body>
  <app-root>Loading...</app-root>
</body>
```

Components are self-describing

- Components are designed as following
  - A component knows how to render itself
  - A component configures dependency injection
  - A component has a well-defined public API of input and output properties
- These make Angular 2 components self-describing, in other words, they contain all the information needed to instantiate them
  - Makes components reusable
- Any component can be bootstrapped as an application as well
  - The default is to use the AppComponent, which is the root component via bootstrap: [AppComponent] in AppModule

Lab #5: Component Bootstrapping

- Start the application using another component as a root component
- Change AppModule (app.module.ts) to use ChildComponent
  - bootstrap: [ChildComponent]
- Change index.html to use the element of ChildComponent
  - <app-child>Loading...</app-child>

Code with Passion!
JPassion.com