Angular

Binding, Routing, Directives

.NET

Data-binding is a mechanism used for coordinating what the user sees with what values the Angular Component contains.

HTTPS://ANGULAR.IO/GUIDE/BINDING-SYNTAX#BINDING-SYNTAX-AN-OVERVIEW

Modeling – Decorators

https://angular.io/guide/template-syntax#inputs-outputs
https://angular.io/guide/glossary#decorator--decoration

Decorators are functions that allow a service, directive or filter to be modified prior to its usage. ***Decorators*** always begin with a @. They do not alter the original code.

Angular has ***Class*** and ***Field*** types of decorators:

Type	Decorator Name	Purpose
Class Decorators	@Component()	Marks a class as a ***component*** and provides configuration metadata.
	@Directive()	Attaches specific behavior to elements in the DOM
	@Pipe()	Supplies configuration metadata.
	@Injectible()	Marks a class as available for Dependency Injection.
	@NgModule()	Marks a class as a ***Module*** and supplies config metadata.
Field Decorators	@Input	Marks class fields as input properties and supplies config metadata. An input property is bound to a DOM property in the template and is updated with the DOM property's value.
	@Output	Marks class fields as output properties and supplies config metadata. The DOM property bound to the output property is auto-updated.

Component Decorator

https://angular.io/guide/template-syntax#inputs-outputs
https://docs.angularjs.org/guide/decorators

@Component - This decorator indicates that the following class is a component. It provides the ***selector***, ***templateUrl***, and ***styleUrls*** metadata.

- The ***selector*** is a unique identifier for the component. It is the name used when the ***component*** is nested in a parent ***component template***.
- The ***templateUrl***, and ***styleUrls*** reference the relative HTML and CSS file locations generated for the component.

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

Data Binding

https://angular.io/guide/template-syntax#property-binding
https://angular.io/tutorial/toh-pt3#update-the-heroescomponent-template

---

The double curly braces ({{ }}) are ***Angular's*** interpolation binding syntax. Interpolation binding presents the component's (.ts file) property ***values*** inside the accompanying HTML template.

Property binding with [] around the property to be bound. This is one-way binding.

```
[class.selected]="hero === selectedHero"
```

Event binding binds events like 'click' or 'hover' to methods in the .ts file using ().

```
<button (click)="addToCart(product)">Buy</button>
```

Two-Way Binding (banana Box!) binds changes. Between two variables. If one changes the other also changes.

```
<input [(ngModel)]="hero.name" placeholder="name"/>
```

Data Binding - CSS Class Binding

https://angular.io/guide/template-syntax#class-binding

You can add and remove CSS class designations from an element with ***class binding***.

To create a single ***class binding***, start with the prefix 'class' followed by '.nameOfCssClass'

[class.selected]="True or False expression"

Angular adds the CSS class label when the bound expression is ***truthy***, and it removes the class label when the expression is ***falsy***.

```
[class.selected]="hero === selectedHero"
```

Data Binding - Event Binding

https://angular.io/tutorial/toh-pt2#add-a-click-event-binding
https://angular.io/guide/template-syntax#event-binding

The parentheses around click tell ***Angular*** to listen for a 'click' event on the <li> element. When the user clicks in the <li> element, ***Angular*** executes the onSelect(hero) function (in the class) on the element.

```
<li *ngFor="let hero of heroes" (click)="onSelect(hero)">
```

In this example, the ***structural directive*** *ngFor will create a <li> for each ***hero*** object in the ***heroes*** collection. Each <li> will have a click event attached to that ***hero*** and submit that ***hero*** as an argument to the onSelect() function.

Data Binding - Two-Way

https://angular.io/tutorial/toh-pt1#two-way-binding

[(ngModel)] is Angular's two-way ***data binding*** syntax. It ***binds*** the class property to the HTML syntax so that data flows in both directions. @ngModule ***decorators*** have the metadata needed for an Angular app to function. The most important @NgModule ***decorator*** annotates the top-level ***AppModule*** class.

```
import { FormsModule } from '@angular/forms';
```

To use forms, in app.module.ts, import ***FormsModule***, then add ***FormsModule*** to the imports array in the same file.

```
imports: [
  BrowserModule,
  FormsModule
],
```

@Input()

https://angular.io/guide/inputs-outputs#sending-data-to-a-child-component

---

1. Get an input from the user.
2. Transfer that value to your parent variable to be passed to the child
3. Create the child component
4. Nest the child in the parent
5. Bind the variable values for the variables between child and parent with [childVar]="parentvar" in the nested child component selector in the parent .html.
6. Display the value in the child

@Output()

https://angular.io/guide/inputs-outputs#sending-data-to-a-parent-component

---

1. Get an input from the user.
2. Transfer that value to a function in the child class
3. Call the .Emit function on the EventEmitter Class variable with the @Output() decorator.
4. Create a EventEmitter variable to emit the value to the parent.
5. Nest the child in the parent
6. Bind the variable values for the variables between child and parent with [childvar]="ParentMethod($event)" in the nested child component selector in the parent .html.
7. Use the called function to transfer the value to a parent variable.
8. Display the value in the parent.

Structural Directives

https://angular.io/api/common/NgIf
https://angular.io/api/common/NgForOf
https://angular.io/guide/template-syntax#ngSwitch
https://angular.io/guide/structural-directives

Structural directives shape or reshape the DOM's structure by adding, removing, and manipulating the elements to which they are attached. Directives with an asterisk, *, are ***structural directives***.

```
<div *ngIf="hero" class="name">{{hero.name}}</div>


<ul>
  <li *ngFor="let hero of heroes">{{hero.name}}</li>
</ul>


<div [ngSwitch]="hero?.emotion">
  <app-happy-hero    *ngSwitchCase="'happy'"    [hero]="hero"></app-happy-hero>
  <app-sad-hero      *ngSwitchCase="'sad'"      [hero]="hero"></app-sad-hero>
  <app-confused-hero *ngSwitchCase="'confused'" [hero]="hero"></app-confused-hero>
  <app-unknown-hero  *ngSwitchDefault           [hero]="hero"></app-unknown-hero>
</div>
```

Attribute Directives

https://angular.io/guide/attribute-directives#attribute-directives

---

To add changes to CSS or HTML syntax within a custom component instead of on the component where the change will happen.

This is useful but, I think, beyond the scope of this class.

Angular Routing

https://angular.io/start/start-routing
https://angular.io/guide/router
https://angular.io/start/start-data#services

A ***route*** associates URL paths with a ***component***.

Register a new ***route*** in app.module.ts or in an app-routing.module file using an array of type Routes.

The routerLink directive in the component .html template gives the ***router*** control over an element.

Insert routerLink="route/{{arg}}" into an element when you want to redirect to a registered URL within the same application.

```
const routes: Routes = [
  { path: 'heroes', component: HeroesComponent }
];
```

```
routerLink="/heroes/{{hero.id}}
```

Angular Routing

https://angular.io/start/start-routing
https://angular.io/guide/router

Routes tell the ***Router*** which view to display when a user clicks a link.

A typical Angular ***Route*** has two properties:

- path: a string that matches the URL in the browser address bar.
- component: the component that the router should create when navigating to this route.

@NgModule metadata initializes the router and starts it listening for browser location changes.

```
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
```

The forRoot() method supplies the service providers and directives needed for routing and performs the initial navigation based on the current browser URL

Routing Step-by-step

https://angular.io/tutorial/toh-pt5#add-the-approutingmodule

1. Add a module called app-routing with
   - ng generate module app-routing --flat --module=app
2. Make sure ***RouterModule*** and ***Routes*** are imported into app-routing.module with
   - import { RouterModule, Routes } from '@angular/router';
   - Also import whatever ***component*** (from its relative location) you will be routing to into app-routing.module.ts
3. In app-routing.module.ts, delete the CommonModule reference and Declarations array.

```
import { HeroDetailComponent } from './hero-detail/hero-detai
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { DashboardComponent } from './dashboard/dashboard.com
import { HeroesComponent } from './heroes/heroes.component';

const routes: Routes = [
  { path: '', redirectTo: '/dashboard', pathMatch: 'full' },
  { path: 'heroes', component: HeroesComponent },
  { path: 'dashoard', component: DashboardComponent },
  { path: 'detail/:id', component: HeroDetailComponent }

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

Routing Step-by-step

https://angular.io/tutorial/toh-pt5#add-the-approutingmodule

4. Configure routes in const routes: Routes = [{ path:'link', component: AssociatedComponent }];
5. Add imports: [RouterModule.forRoot(routes)], under @NgModule.
6. Under @NgModule add exports: [RouterModule].
7. In app.component.html, where you want all route html templates to appear, add:
   - <router-outlet></router-outlet>
8. Add <a routerLink="/[link]">NameOfLink to whatever page you want to add a link to.

```
import { HeroDetailComponent } from './hero-detail/hero-detai
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { DashboardComponent } from './dashboard/dashboard.com
import { HeroesComponent } from './heroes/heroes.component';

const routes: Routes = [
  { path: '', redirectTo: '/dashboard', pathMatch: 'full' },
  { path: 'heroes', component: HeroesComponent },
  { path: 'dashoard', component: DashboardComponent },
  { path: 'detail/:id', component: HeroDetailComponent }

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```