Exercise 1

Given the following diagram:



Match the letters one of the following terms

1. Service 2. Model 3. Decorator 4. Defines name of HTML element.
2. Decorator
3. Model
4. Defines name of HTML element

Exercise 2

How is AppComponent identified as the starting component inside app.module.ts? (1 mark)

|  |
| --- |
| Bootstrap root component |

Exercise 3

Add a second property to the AppComponent class to store your last name. Modify the code in the template option to show your last name with the rest of the original content. Show your revised **app.component.ts** file here: (1 mark)

export class AppComponent {  
 firstName = “Katelyn”;  
 lastName = “Truong”;  
   
}

Exercise 4

Inside app.component.html replace the content of this file with:

|  |
| --- |
| <h1>  Welcome to **{{**title**}}**!  </h1> |

1. Save your changes and run your project. Next using the debugger, set a breakpoint beside the area where the title model is defined inside app.component.ts. Show a screenshot with the Chrome debugger Sources tab while open at the app.component.ts file and while halted at the break point that you set. Show the screenshot here.

|  |
| --- |
| /Users/katelyntruong/Desktop/Screen Shot 2018-03-04 at 9.48.27 PM.png |

1. Resume execution of your application so the page renders in the browser. Show a screenshot of the debugger with its elements tab visible so it shows the message *Welcome to app* nested inside the HTML. Please resize your browser to capture and present this nicely with a size that fits well in this page. Show a screenshot of this here:

|  |
| --- |
| /Users/katelyntruong/Desktop/Screen Shot 2018-03-04 at 9.48.38 PM.png |

Exercise 5

Add a second card object to the *AppComponent* class and store the Queen of Diamonds. Then modify the template to show the new card and suit in addition to the ace of spades. Show your revised app.component.ts file here: (1 mark)

|  |
| --- |
| @Component({  selector: 'app-root',  // Multi-line content is allowed with back ticks.  template: `<h1>Hello world! {{title}} <br/>  {{card.cardVal}} of {{card.suit}} and<br/>  {{card1.cardVal}} of {{card1.suit}}  </h1>` })  export class AppComponent {  public title = 'This is Angular!';   // Declare and initialize a PlayingCard object.  public card: PlayingCard = {  cardVal: "Ace",  suit: "Spades"  };  public card1: PlayingCard = {  cardVal: "Queen",  suit: "Diamond"  }; } |

Exercise 6

Add another input which allows the user to input the suit of the *card* object. Show your revised app.component.ts file here: (1 mark)

|  |
| --- |
| @Component({  selector: 'app-root',  // Multi-line content allowed with back ticks.  template: `<h1>Hello world! {{title}} <br/>  {{card.cardVal}} of {{card.suit}}.</h1>  Card Value: <input [(ngModel)]="card.cardVal"><br/>  Card Suit: <input [(ngModel)]="card.suit">` })  export class AppComponent {  public title = 'This is Angular 4!';   // Declare a PlayingCard object.  public card: PlayingCard = {  cardVal: "Ace",  suit: "Spades"  }; } |

Exercise 7

Replace the code that builds an unordered list with *ngFor* to iterate through all cards to display the card value and suit within a table by using <tr> and <td> tags along with a <table> tag. Place the card value and suit in separate columns. (2 marks)

|  |
| --- |
| import { Component } from '@angular/core';  export class PlayingCard {  cardVal: string;  suit: string; }  @Component({  selector: 'app-root',  *// Multi-line content allowed with back ticks.* template: `<h1>Hello world! {{title}}</h1> <br/>  <!-- Show cards in unordered list.  <ul><li \*ngFor="let card of cards">{{card.cardVal}}</li></ul> -->   <table style="border: 1px black solid">  <tr>  <th width="100px" style="border: 1px black solid; background-color: #dfdedd">Card Value</th>  <th width="100px" style="border: 1px black solid; background-color: #dfdedd">Card Suit</th>  </tr>  <tr \*ngFor="let ***card*** of cards">  <td width="100px" style="border: 1px black solid">{{***card***.cardVal}}</td>  <td width="100px" style="border: 1px black solid">{{***card***.suit}}</td>  </tr>  </table>  ` })  export class AppComponent {  public title = 'This is Angular 4!';  *// Include card data in collection as public property.* public cards = CARDS; }  *// Define card data.* let CARDS: PlayingCard[] = [  { cardVal: "Ace", suit: "Spades" },  { cardVal: "Two", suit: "Clubs" },  { cardVal: "Six", suit: "Hearts" },  { cardVal: "Queen", suit: "Diamond" }, ]; |

Exercise 8

Modify the contents that appear when a list item is selected so if the new content is clicked an alert box appears. The only task you need to complete for this exercise is to place this line of code inside the correct section of the app.component.ts file:

alert(card.suit);

Show your revised app.component.ts file:

|  |
| --- |
| import { Component } from '@angular/core';  export class PlayingCard {  cardVal: string;  suit: string; }  @Component({  selector: 'app-root',  *// Multi-line content allowed with back ticks.* template: `<h1>Hello world! {{title}} </h1><br/>  <!-- Show cards in unordered list. -->  <ul><li \*ngFor="let ***card*** of cards" (click)="onSelect(***card***)">  {{***card***.cardVal}}</li>  </ul>  <div \*ngIf="selectedCard">  <h2>{{selectedCard.cardVal}} \*\*</h2>  <input [(ngModel)]="selectedCard.cardVal" placeholder="name"/>  </div>  ` })  export class AppComponent {  public title = 'This is Angular 4!';  *// Include card data in class as public property.* public cards = CARDS;   selectedCard: PlayingCard;   onSelect(card: PlayingCard) {  this.selectedCard = card;  alert(card.suit);  } }  *// Define card data.* var CARDS: PlayingCard[] = [  { cardVal:"Ace", suit:"Spades"},  { cardVal:"Two", suit:"Clubs" },  { cardVal:"Six", suit:"Hearts"}, ]; |

Exercise 9

Add a second control which allows the input of a last name value. This field is required. Only alphabetical characters are allowed. The minimum length is two letters. Appropriate messages appear to inform the user how to validate this field. Show your revised version of app.component.ts after making these changes.

|  |
| --- |
| import { Component } from '@angular/core';  @Component({  selector: 'app-root',  template:  ` <section> <form (ngSubmit)="onSubmit()" #myForm="ngForm">  First Name:  <input type="text" pattern="[a-zA-Z ]\*" minlength="3" required   [(ngModel)]="myName" name="firstName" #firstName="ngModel" >  <div [hidden]="***firstName***.valid || ***firstName***.pristine">  This control is invalid.</div>  <p \*ngIf="***firstName***?.errors?.required">This field is required.</p>  <p \*ngIf="***firstName***?.errors?.pattern">  Only alphabetical characters are allowed.</p>  <p \*ngIf="***firstName***?.errors?.minlength">  This entry must have at least three characters.</p>   <br><br>  Last Name:  <input type="text" pattern="[a-zA-Z ]\*" minlength="2" required  [(ngModel)]="lName" name="lastName" #lastName="ngModel" >  <div [hidden]="***lastName***.valid || ***lastName***.pristine">  This control is invalid.</div>  <p \*ngIf="***lastName***?.errors?.required">This field is required.</p>  <p \*ngIf="***lastName***?.errors?.pattern">  Only alphabetical characters are allowed.</p>  <p \*ngIf="***lastName***?.errors?.minlength">  This last name entry must have at least two characters.</p><br>  <button type="submit" class="btn btn-default"   [disabled]="!***myForm***.form.valid">Submit</button>  </form> </section> ` }) export class AppComponent {  myName: string;  lName: string;   constructor() {  this.myName = "Katelyn";  this.lName = "Truong";  } } |

Exercise 10

To review some of the terminology, match the following terms with the definitions provided. Use each term only once.

template (view), Router, Model, Service, Directive, Decorator

1. \_\_\_\_\_\_\_Model \_\_\_ defines a data reference.

2. \_\_\_\_\_\_ Decorator \_\_selects templates (views) and components.

3. \_\_\_\_\_\_\_\_ Directive \_\_modifies behavior of HTML.

4. \_\_ template (view) \_\_\_ stores HTML and presents content.

5. \_\_\_\_\_\_ Router \_\_\_ contains metadata which describes how the component class is processed.

6. \_\_\_\_\_ Service \_\_\_\_\_ encapsulates logic which can be used by many components.

Exercise 11

Please answer true or false to the following questions:

1. \_\_\_T\_\_ The package.json file refences dependences that are to be downloaded by the node package manager.
2. \_\_\_T\_\_ The package.json file defines command references that can be executed by the node package manager.
3. \_\_T\_\_\_ main.ts sets the starting module for the application.
4. \_\_\_T\_\_ TypeScript has *number* and *string* datatypes among others.
5. \_\_T\_\_\_ TypeScript must be transpiled to JavaScript to run in a browser.
6. \_\_\_T\_\_ The selector tag defines an element name which can identify the html tag needed to instantiate and render a component.