

Angular

Lab 3: Directives

Exercises

1	LAB SETUP	1
2	GENERATING A NEW ANGULAR PROJECT	1
3	NGCLASS.....	1
4	NGSTYLE	2
5	NGIF	2
6	NGFOR.....	2
6.1	ACCESSING INDEX OF ITEM IN COLLECTION.....	3
6.2	STYLING USING FIRST, LAST, ODD AND EVEN EXPORTED VALUES.....	3
6.3	NGFOR WITH CHILD COMPONENTS	3
7	COMBINING NGIF AND NGFOR.....	3
8	NGSWITCH.....	4

1 Lab setup

2 Generating a new Angular project

3 NgClass

Q1: Create a `<div>` element and nest a `<p>` with some text within this `<div>`.

Q2: The [CSS Box model](#) provides a set of properties to delineate an element and its child elements. You can use a variety of [properties](#) to control the border look for the `<div>`. Add 2 different classes in the template CSS which sets different values for the `border-style` property, another 2 different classes which sets different values for the `border-color` property, and another 2 different classes which sets different values for the `border-width` property. Give these classes any suitable names. Set a suitable `padding` for the `<div>`.

Q3: Add any one of the 6 classes to the `<div>`, and use a string to add another 2 more of these classes to the `<div>`

Q4: Add any one of the 6 classes to the `<div>`, and use an object to add another 2 more of these classes to the `<div>` while removing the existing class.

Q5. Add any one of the 6 classes to the <div>, and use an array of strings to add another 2 more of these classes to the <div>

4 NgStyle

Q1: Create a <div> element and nest a <p> with some text within this <div>.

Q2: Create an object whose properties represent CSS properties in the component. The [CSS Box model](#) provides a set of properties to delineate an element and its child elements. You can use a variety of [properties](#) to control the border look for the <div>. Set a suitable `padding` for the <div>.

Q3: Use NgStyle to apply this object to style the <div> element from Q1

5 NgIf

Q1: Create a text field and use either event binding to a component method or two-way binding with NgModel to track the value in the text field

Q2. Create a <p> element with some random text within it and apply NgIf to display this element when the number of characters in the text field from Q1 is more than 5.

6 NgFor

Q1: Create a [input field for a number](#) and use either event binding to a component method or two-way binding with NgModel to track the value in this field

Q2. Display a list whose items count from 1 up to the value in this field using NgFor. For e.g. if the value is 5, then the list will look like:

- Count : 1
- Count : 2
- Count : 3
- Count : 4
- Count : 5

To accomplish this, we will need an array containing the numbers 1 to the value in this field. This array will have to be dynamically created depending on the value in the input field from Q1. Provide an additional button with event binding to a component method that dynamically create this array.

You can use this shortcut for creating this array dynamically:

<https://stackoverflow.com/questions/3746725/how-to-create-an-array-containing-1-n>

6.1 Accessing index of item in collection

6.2 Styling using first, last, odd and even exported values

Q1: Create another `<div>` containing a collection of buttons using `NgFor`. Repeat the styling for the `<div>` and the buttons using `odd`, `even`, `first` and `last` but this time using either style binding or `NgStyle`

6.3 NgFor with child components

Q1: Create another domain model class called `Order` with the following properties:

- `description`: string;
- `quantity`: number;
- `price`: number;
- `totalCost`: number;

Include a method `calculateCost` in this class to calculate the `totalCost` = `price` * `quantity`

The constructor for this class should only initialize `description`, `quantity` and `price`. The `totalCost` value is computed by calling the `calculateCost` method.

Q2: Create a new child component to receive and display data for objects created from the `Order` class.

Q3: Create appropriate elements in the child template to display each property from an `Order` object properly using interpolation, including `totalCost`.

Q4: Create appropriate `@Input` properties in the child component to receive the `Order` objects and number from the parent template

Q5: Create an array of 5 `Order` objects with random values in the parent component, and calculate the `totalCost` using the `calculateCost` method

Q6: In the parent template, using `NgFor` together with parent-child binding to pass each object as well as the number of each object to the `@Input` properties in the child component.

7 Combining `NgIf` and `NgFor`

Q1: Using the Solution from 6.3 (Q1 - Q6), add an additional button to flip the value of a Boolean property in the parent component.

Q2: Based on this button, either perform or ignore the `ngFor` together with parent-child binding that is used to pass each object / index number of object in array to the child component

Q3: Further modify / extend the parent-child binding to only pass Order objects whose totalCost is more than a specific value to the child component. Choose this specific value and the random values for your 5 Order object to demonstrate that only certain Order objects are passed.

8 NgSwitch

Q1. Create three [radio buttons](#) with 3 different possible values

Q2: Using NgSwitch, display 3 different <p> contents depending on the particular radio button selected.