**Angular 13**

Interpolation (1-demo-app)

1. Used to display dynamic data on a website using ‘{[]}’ (double curly braces).
2. Can be used to perform arithmetic, relational operations inside the braces.
3. Can not be used to create a new object, to find the type of the object, for assignment etc.

Angular CLI and Important Commands

1. CLI commands can be found @ [Angular - CLI Overview and Command Reference](https://angular.io/cli).
2. CLI can be used to create, build, run and deploy the application.
3. Can also be used to create class, interface, routes etc.
4. Can reduce bugs while creating the above without CLI.
5. To generate:
   1. ng generate class Dummy
   2. ng generate component component-name
   3. ng generate module module-name
   4. ng generate component module-name/component-name
   5. ng generate service service-name
   6. ng generate service module-name/service-name
6. To Generate build:
   1. ng build
   2. creates a ‘dist’ folder inside the project root folder which can then be used to deploy the project.

Components

1. Components are the building block to develop a specific feature or functionality
2. Components after creation can then be added to the app.component.html file to be used or rendered.
3. Components are added using ‘<app-component-name>’.
4. We can change the component name if we change the component selector inside the .component.ts file.
5. We can also change the name of the html and css file for the same component using the .component.ts file.
6. Components should be created in a nested format so that editing and maintaining the component can be easy. For e.g., Header component can contain,
   1. The Logo Component
   2. The Search Bar Component
   3. The Menu Component. The Menu can in turn have multiple nested components such as the profile component, dropdown component or other such components.
7. Nested Components makes it easier to manipulate the positioning, styling and editing the components.

Components with InLine

1. Three Cases for Components with InLine:
   1. InLine Style : Used when css required for the component are few.
   2. InLine Template : Used when html component is not required or used inline.
   3. InLine Style & InLine Template (Most Popular) : Used when html and css component are returned inline.
2. InLine Style
   1. ng g c component-name –inline-style
   2. Will not generate the css component file in the new component.
   3. Style for the components in such case can be added by mentioning the class for the tags inside the html component and defining the css inside the component-name.ts file using the class name.
   4. To reference the class for the html tag we must use `class-name{css;}` (backtick) characters.
3. InLine Template
   1. ng g c component-name –inline-template
   2. Will not generate the html component.
   3. HTML for the page will be passed inline inside the component-name.component.ts file.
   4. Inside the template use `html-code` (backtick) to write the html.
4. InLine Style & Template
   1. ng g c component-name –inline-style –inline-template
   2. Will not generate the html and css component.
   3. Usage of such is the same as above cases.
5. Need For InLine
   1. When the lines of html are few(2 or 3), it is better not to create a file and use css inline.
   2. The same goes for html inline.
   3. When html and css both are few, it is better not to create files for the same and use inline format.
6. The component is used in the app.component.html the same way described above.

Module (2-modules)

1. Major feature of an angular application.
2. Basically, a collection of components, services, pips etc.
3. Modules contain a group of features and functionalities which are related to each other.

Graphical user interface, application

Description automatically generated

1. For e.g., User module can have multiple components such as Login, Registration, Forgot Password etc. It can also have services such as API calls. And can also have other things related to a User.
2. User defined modules must be imported inside the ‘app.module.ts’ before they can be used. Can be imported using:
   1. Import {module-reference} from ‘./module-name/module-name.module’
   2. module-reference refers the name which will be used to call the module components. The import style is same as python import style i.e., <import as name-to-be-used from module-name>.
3. Imported module reference name then must be added to the imports array.
4. To use the component inside this module we must first export the component.
5. This can be accomplished by adding the exports array to the .module.ts file of the module component that is to be used.
6. The module component can be called using the same format described in the component section. The name to be used to call the component can be found inside the .module.ts file of the module.
7. This helps in creating services, routes and helpers etc, which then need to follow the same rules above to be used inside the application.

Buttons (3-buttons)

1. Add a button to the app.component.html
   1. Give the button a display name inside the button tag.
   2. Inside the button body use “(click)=function\_name()” to call the function to be used.
   3. The function then must be created inside the app.component.ts file inside the class.
   4. Make sure the class is exported otherwise the function will not be discovered outside the class.
2. Adding mouseover, mouseleave, keyup, keydown and other such events to the button
   1. These are events that can be added to the body of a tag.
   2. Such features are added to the ag body inside ‘(event)’and can be set equal to a function inside double quotes e.g., <(event) = “function()”>.
   3. Some such events are :
      1. mouseover, mouseleave, mousemove etc.
      2. keyup, keydown, blue, input
      3. click
3. Multiple events can be used inside a single tag.
4. Getting Text Box Value from HTML Input. There are two ways to do this
   1. Get value on keyup
      1. By making an input tag with id, name, placeholder and adding a function to the keyup event we can get the value of the input field.
      2. This value can then be passed to the function and can then be displayed to the screen or the console.
      3. The value of the text filed will be passed by using the id.text-field\_property. Such as box.value.
   2. Get value on button click
      1. The input tag will the defined as above but without the event.
      2. The button will be defined and will contain the click event. This event will call the function and pass the value to the function.
      3. The value of the text filed will be passed by using the id.text-field\_property. Such as box.value.
5. Multiple different properties can be passed to the function inside the button. The attribute properties defined inside the input field can be called inside the button event using “id.(dot)attribute-property”.
   1. E.g., id.value, id.placeholder, id.type etc.
6. There are other ways of getting the value also, but this is a simple way. More complex ways will be discussed further.

Counter (4-counter)

1. Used for increment and decrement. In this tutorial buttons are used to achieve this.
2. Create two buttons in the application. Add a click event to both calling the same function with two different parameters. One param is ‘minus’ the other being ‘plus’.
3. Define variable inside the app.component.ts file to store the count value, say count = 0.
4. Create another function with a parameter to catch the incoming parameter value. Say ‘type : string’.
5. Use if condition to check the check the against the parameter and increment or decrement the value of count.
6. Use interpolation to display the value of count on the page.

Basic Styling (5-basic-style)

1. Styling for the local component is done inside the component.css file inside a component.
2. Adding the style in this file will apply the style to the component.
3. The rules of css styling are followed. Adding class to a tag will allow the class to be called inside the css file and the style can be applied to it there.
4. To add the style to all the components we must add the style to the style.css file inside the root directory.
5. Style can also be added as tags inside the component.html file by creating a style tag under the tag where the style must apply.
6. Style can also be added inline inside the body of the tag.
7. The preference is given in order of “inline style > internal style> component.css > style.css”.

Property Binding

1. Feature of angular; used to update or change the properties of any input element.
2. Let’s assume we want to change the properties of any tag at runtime. Declaring these properties with interpolation can help us achieve that.
3. But interpolation has a flaw that it does not understand or recognise the Boolean values.
4. As a result, the property does not work the way it was intended.
5. To achieve this, we use property binding.
6. Here we define the property inside ‘[]’ brackets and assign them a value using ‘=’ without the interpolation.
7. Works the same as interpolation but also allows the property to understand Boolean values.

Conditional Statements (6-condition-statement)

1. If conditions can be used to make decisions based on certain conditions.
2. They are used in this tutorial to show or hide the text inside the h1 tag.
3. Define a h1 tag calling a function inside app.component.ts where we use two variable to toggle the sow and hide property.
4. Inside the h1 tag the condition needs to be called using the \* (star).
   1. \*ngIf is used toggle the show and hide.
5. ng-template is used to contain tags which should be shown with a condition.
   1. Condition can be set using binding ngIf in the ng template body.
   2. [ngIf] = “condition”

Switch (7-switch)

1. Switch in Angular is preferentially used inside a div tag. Inside the body define a ngSwitch property.
   1. This property will take the name of the variable to use to switch.
2. The div will contain the tags which will be the switch cases. The case will be defined inside the tag body using \*ngSwitchCase.
   1. The case will be enclosed in single quotes inside the double quotes like “’property’”.
3. Default case is defined using \*ngSwitchDefault.

For Loops (8-loops)

1. Angular for loops and JavaScript for loops are two separate things.
2. Loops inside html documents will use angular for loops.
3. Loops inside JavaScript will use JavaScript for loops.
4. Loops are made using \*ngFor = “let var of array” and the body of the tag uses interpolation to display the var in the loop.
   1. If the array multiple key : value pairs, we use array.key inside the interpolation to display it.
   2. If the array has another array, we then use ngFor with let var array.key and the body contains the var inside interpolation to display it to the page.

Style Binding (9-style-binding)

1. Like property binding style binding is used to make the style of a tag dynamic.
2. To style bind use []
   1. Inside use “style.attribute” like “style.color”.
   2. And assign it to the variable to be used to define the colour.
   3. The variable is initialized inside the app.component.ts.
3. Multiple style binding can be used inside the same tag using the same way described here.

Header (10-header)

1. Create a header component and add it to the app inside the app.component.html.
2. Add it by using the selector for the header component.
3. Define a nav tag inside the html of header component. And add image and unordered list. With anchor tags to different locations.
4. Add the css to the header component inside the header.component.css file.
5. This will create the header and add it to the application.
6. The header will have a margin inside the main application.
7. To remove the margin so that the header could take the whole space add margin 0px to the style.css file inside the project root directory.
8. Calling the header again or as many times will add multiple headers from top down and stack them over each other.

Basic Form (11-basic-form)

1. Form are modules which must be imported inside the app.module.ts file and register them under imports.
2. Form is then created inside the html using form tag. The form tag will take id with #id and will be set equal to a value.
3. The form will have a click event which will send the data to function on button click. The parameter will be “id.value”.
4. Inside the app.module.ts, import NgForm to refer to the data caught by the function.
5. Create the function with parameter to catch the data. Assign the type as NgForm
6. Create a new variable outside the function to store the data. The type will be any and the value will be object “{}”.
7. Set this variable equal to received data.
8. This data can then be displayed using interpolation inside the application.
9. To see refer to the project. Inside app.component.html.

Toggle Element (12-toggle-element)

1. Toggle Basically refers to showing and hiding an element with reference to something. In this we use a button to toggle a tag.
2. Create a Boolean inside app.component.ts which will be used to toggle.
3. Set the Boolean to true.
4. Create a tag inside app html page and assign \*ngIf to this Boolean variable.
5. Create a button and on this button call, call a function.
6. This function will then change the value of the Boolean variable using ! (not).
7. For e.g., “this.variable = !this.variable”.

Bootstrap (13-bootstrap)

1. Library used to make a page responsive i.e., the application will now work on any type of device.
2. To add bootstrap run the command “ng add @ng-bootstrap/ng-bootstrap”.
3. This will update the packages and import the bootstrap module.
4. To get the templated for the use of bootstrap goto <https://ng-bootstrap.github.io/#/home>
5. Adding bootstrap is as simple as copying the component code from the website and adding it to the component.html.
6. The application page will display the same for the bootstrap elements.
7. Bootstrap is a library. As displayed on the official page of bootstrap.
8. Alternatives for bootstrap are Material-UI.

Material UI (14-material-ui)

1. Material-UI is a HTML and CSS library, that is used to enhance the application by adding extra properties to elements.
2. Provides better look and feel to our application.
3. To Learn more goto <https://material.angular.io/>
4. To add a component to the application, use the component guide and add the component.
5. After adding the component, it will be displayed to the application but the css will not be working correctly.
6. This happens because right now it is being called as an html tag to register it with the material ui we must import the component class from the Module.
7. This is done inside the app.component.ts file and it is imported the same as above examples.
8. Material UI makes the application slow hence the use of material ui is always recommended using when there is less amount of content to display or in a mobile application.

TODO Application (15-todo-app)

1. Designing a Todo Application.

Data Passing: Parent to Child (16-data-passing-ptc)

1. Passing Data from parent Component to Child Component.
2. Create a component. This will be the child component. Name its child to make it easier.
3. Add the child component to the main page.
4. Create the data to be sent to the child component
5. Create a function to change the value of this variable let’s say using random.
6. Inside the html page add child component and inside the body create a property and assign the variable to it.
7. Create a button with the click event calling the function to update the value.
8. Display the value inside the child element using interpolation.
9. We can send multiple variables / data inside the same component.
10. We can send multiple tag data using the same click button.