1. If in case angular dependencies do not work or npm start gives error then execute below commands from the root folder of your project which had the package.json file
   1. Delete node\_modules folder
   2. npm cache clean – force
   3. npm i --save
2. Folder structure:
   1. Package.json => dependencies , =scripts to run
   2. Angular.json => all the information for the angular kick start => initialization process
   3. Index.html => inside the src folder => <app-root>
   4. Styles.css => global styles
   5. Main.ts => reference the bootsrap module
   6. App.module.ts => decorated with @NgModule
      1. Declarations:[] => user defined components /pipes/ directives
      2. Imports:[] => all the prededined modules of angular
      3. Providers: [] => services
      4. Bootstrap => the name of the root component
      5. AppComponent
         1. Html => UI for the component
         2. Css => styles specific to this component
         3. Ts => the BL to modify the DOM
         4. Spec.ts => test cases
3. Install angular cli  
   npm i @angular/cli@<version> => install locally within your workspace  
   npm i -g @angular/cli@latest => install globally on your system
4. Create new angular project  
   ng new <projectname>
5. To create a component  
   ng g c <componentname>
6. 1-way Data Binding syntax => from component to view
   1. {{}} – string interpolation [ always evaluates values in string ]
   2. [] -> it retains the type of the variable
7. Event binding : (click)
8. 2-way data binding : banana in a box [()]
   1. Use ngModel directive provided by FormsModule  
      **DO NOT FORGET TO ADD FormsModule IN YOUR app.module.ts file**
9. Style binding  
   [style.<style\_property>] = “’<value>’”

[style.<style\_property>] = “<propertyname>”

1. ViewEncapsulation => provides how styles will flow from parent to child
2. Nested Components => place the selector in the components the hierarchy of the DOM is created  
   APP => <header> <main> <footer>  
   Main=> <product-list>  
   ProductList => <product>
3. Parent - child communication
   1. Data can be send from parent to child using : @Input() decorator
   2. Child sends event to parent component using : @Output()
4. Directives
   1. Components => UI (HTML) + BL
   2. Attribute directive => ngModel, style, ngStyle , class, ngClass,   
      modify the behaviour of the html element on which it is applied
   3. Structural => \*ngIf, \*ngFor, ngSwitch=> \*ngSwitchCase and \*ngDefault
   4. Custom directive :
      1. Create a directive  
         ng g d hover  
         ng g directive hover
      2. Selector has syntax of [appHover]
      3. To access the element on which the directive is used :  
         ElementRef
      4. To change any style :  
         Renderer2
      5. To change the property of host element:  
         @HostBinding
      6. To list for any event change on host element :  
         @HostListener
      7. Directives can take input from host element using @Input()
      8. To use directive without data binding  
         <img appHover/>
      9. To use directive for @Input(), use data binding syntax to pass the data  
         <img appHover [itemname]=”itemname”/>
5. Lifecylce methods:
   1. Constructor => to instantiate the component. DO NOT DO ANY LONG RUNNING PROCESS IN THE CONSTRUCTOR
   2. ngOnInit => gets called only once and used for initialization
   3. ngOnChanges => listens for changes in properties decorated with @Input(). It listens only for changes in primitives
   4. ngdoCheck => listens for changes in properties decorated with @Input(). It listens only for changes in primitives or objects
   5. ngOnDestroy => gets called once when the component is removed from the DOM