Two binding :

Some data we get from component to html page variable values to html page {{}} by using this also we got the data from component .ts files to html is called databinding one way process

In real time we enter the data to form pages text boxes user names passwords into the form that data needs to go to back to the component.ts this is another way of binding this is call event binding(exboxes, buttons etc)

The above both combined two way binding

App.component.html

<h1>{{title}}</h1>

<form>

  <div class="form-group">

    <label for="exampleInputEmail1">Email address</label>

    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp"  (keyup)="data1($event)"  >

  </div>

  <div class="form-group">

    <label for="exampleInputPassword1">Password</label>

    <input type="password" class="form-control" id="exampleInputPassword1"   (keyup)="data2($event)">

  </div>

  <div class="form-group form-check">

    <input type="checkbox" class="form-check-input" id="exampleCheck1">

    <label class="form-check-label" for="exampleCheck1">Check me out</label>

  </div>

  <button type="submit" class="btn btn-primary">Submit</button>

</form>

<h1>Email:</h1>{{email}}

<h1>Password:</h1>{{password}}

App.component.ts

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'twowaybinding';

  public address:any="Hyderabad";

public emial:any="";

public name:any="";

data1(event:any)

{

  var that=this;

  that.email=event.target.value;

}

data2(event:any)

{

  var that=this;

  that.password= event.target.value;

}

}



Process: we enter the data in text box by (keyup)=data1($event)

Inside of component.ts

We captured the value of this event in the data1 function by using the event.target.valeu

And store in variable and displayed tjat variable in html page using the inter polation {{}}

Note :

From component.ts

  public address:any="Hyderabad";

<input type=”text” [value]=”address”>

This one way binding to access the datra from the component.ts and displayed in the textbox

App.component .html

We reach two binding by using the ngModel <h2>Example 2</h2>

<input type="text" name="value" [(ngModel)]="value">

<p> You entered {{value}}</p>

<button (click)="clearValue()">Clear</button>

App.component.ts

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'ngmodeltwowaybinding';

  value=""; we received the value from html textbox to here

 clearValue() {

   this.value="";

 }

}

App.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

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

import { AppComponent } from './app.component';

import {CommonModule} from '@angular/common';

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule,FormsModule,CommonModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

Directives in angular

Structural directive

NgIf,

NgFor

NgSwitch ng swithcase ngswithc default

These are the manupluction of dom if the condition true add one <p> element else don’t add that element

Attribute dorectives

These for the appearance NgClass,NgStyle

Component-component service

This means a value in the one component passed to the another component for suppose we want a title varible from app.componet.ts file to the child.component.ts file

Variable title from app.component.ts to its child component child.component.ts by Using the @Input() decorator.

@Input()decorator can share the parent component data to child component

And we know the app.compoenent ts is root component and it will be the parent and we write the all child selectore tags in the applcomponent.html.

|  |  |  |  |
| --- | --- | --- | --- |
| App.componet.ts | App.component.html | Child.component.ts | Child.component.html |
| Observe the title variable here  Data is here  Title should be puble not private | Here “title” is the app component.ts file variable and value is the child component variable databinding is going  <child-root> is belongs to child component.ts file selector | Here we received the value value has the data of title variable(compoenservice-parent)  Value should be public not private | I displayed that data in the component.html view |

Data pass from child to patent by using the view child

There is @viewchild decorator for this

We have to write this in appcomponent.ts file

@viewchild(Childcomponet) child: any;

We need predefined interface to work this mechanism

ngAfterViewInit() {

  this.mess = this.child.message

}

Here we displayed the child.message value

Pipes:

***Simply formatting the data before displayed is pipe.***

When we want to develop any application, we always start the application with a simple task: retrieve data, transform data, and then display the data in front of the user through the user interface. Retrieval of data from any type of data source totally depends on data service providers like web services, Web API, etc. So, once data arrives, we can push those raw data values directly to our user interface for viewing by the user. But sometimes, this is not exactly what happens. For example, in most use cases, users prefer to see a date in a simple format like 15/02/2017 rather than the raw string format Wed Feb 15 2017 00:00:00 GMT-0700 (Pacific Daylight Time). So, it is clear from the above example that some values require editing before being viewed in the user interface. Also, that same type of transformation might be required by us in many different user interfaces. So, in this scenario, we think about some style type properties that we can create centrally and apply whenever we require it. So, for this purpose, the Angular framework introduced Angular pipes, a definite way to write display – value transformations that we can declare in our HTML

Note: we have to use common module to work with angular built in pipes.

How the function take and return the values in the same way pipes take the values and return the values in good format