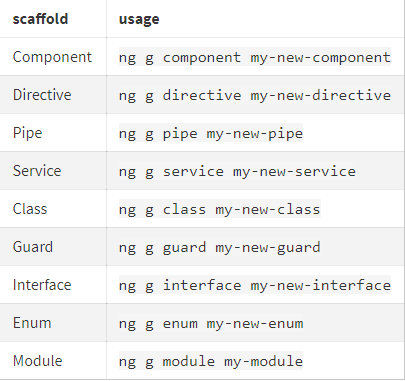
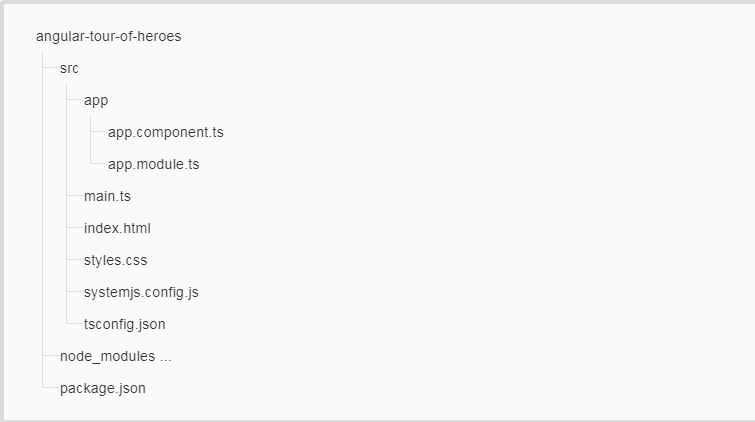
Angular 2 Architecture



Angular CLI use ful commands:



Project folder structure:



Custom Component:

Component = directive + template

Creating component manually

First create component file with ts extension.

@Component defines component

For a component you need two mandatory things

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

@Component({

selector:'hello'

template:`some thing`

})

export class HelloComponent{}

-----------------------------------mycomponent.ts-------------------------------

Now handover this component to module.

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

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

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

import { HelloComponent } from './my.component';

@NgModule({

imports: [ BrowserModule ],

declarations: [ AppComponent,MyComponent ],

bootstrap: [ AppComponent,MyComponent ]

})

export class AppModule { }

------------------------------------------------module-----------------------------------------------------------

In html give the directive

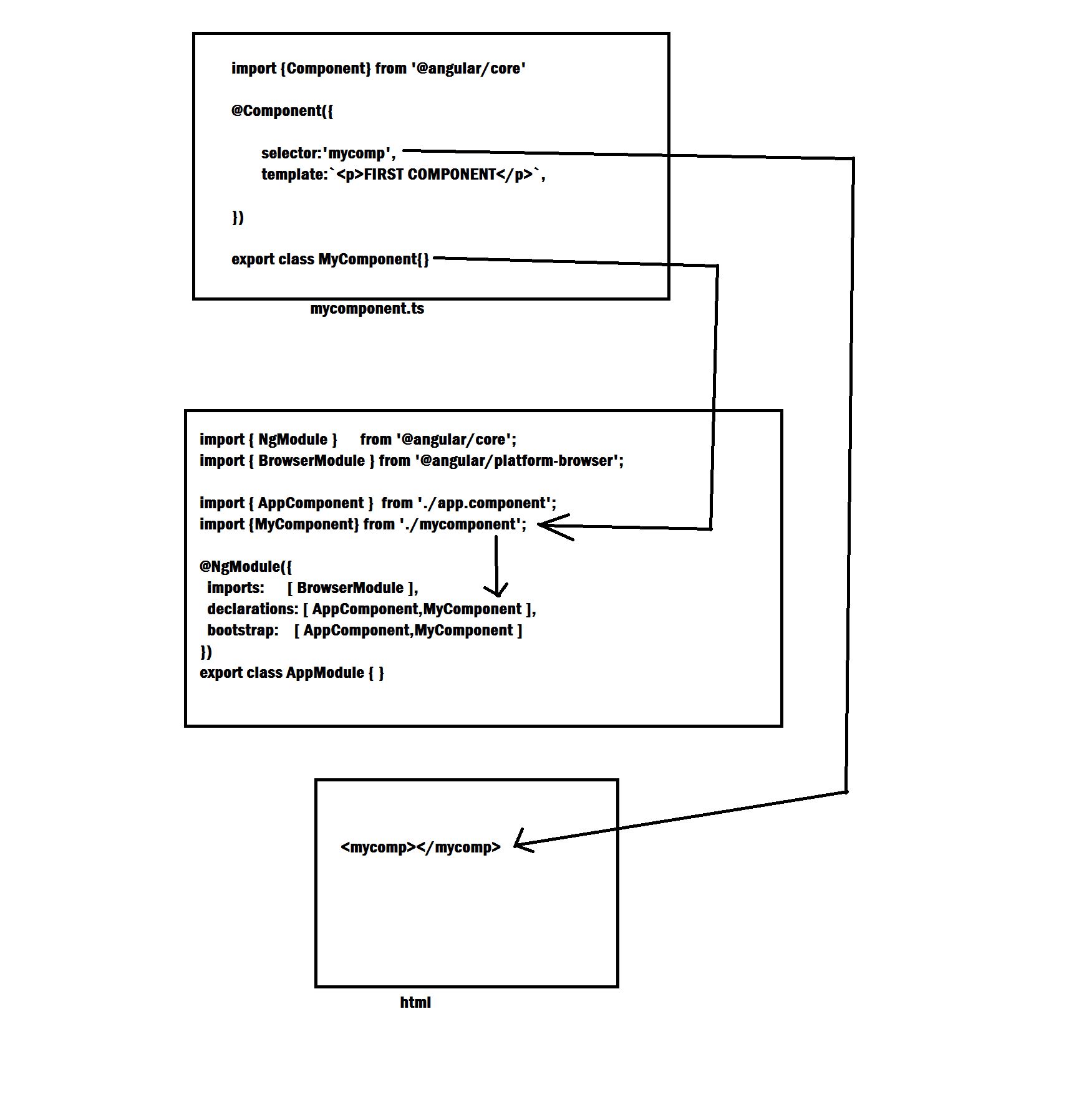
<first>HI</first>

--------------------------------index.html----------------------------------------------------

Follow below diagram for quick understanding.

But if your using angular client no need put this much effort

Just type ng generate component component name



Structural directives:

\*ngIf,\*ngFor,

Inputs:

Whenever you need to transfer data from one component to another components we will use inputs:

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

@Component({

selector: 'app-hello',

template: `HELLO COMPONENT ==>{{parentData}}`,

inputs:[`parentData`]

})

export class HelloComponent {

public parentData:string;

}

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

import { HelloComponent } from './hello/hello.component';

@Component({

selector: 'app-root',

template: `PARENT COMPONENT<input type="text" #demo (keyup)="0">

<app-hello [parentData]="demo.value"></app-hello>`,

})

export class AppComponent {

}

Angular 2 Forms:

Here Iam discussing about template driven forms:

Here we are declaring a local variable called “form” and setting it to an instance of ngForm. This is very important. Now our local form variable becomes of type FormGroup allowing us access to the FormGroup API’s on this local variable. We use this in the ngSubmit event where we send the value of the form via form.value

Since we are working with template driven forms, we can use the ngModel directive to capture the values of our forms. One thing to note if you are coming from Angular 1.x. Using ngModel as shown below creates a one-way data binding, so once we hit submit the data is only sent to the controller. If we wanted to use two-way data binding, we would have to wrap the ngModel in [()] and assign an attribute to it. Also the name of the field corresponds to the name attribute so our first input will be firstName.

Component:

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

import { HelloComponent } from './hello/hello.component';

@Component({

selector: 'app-root',

template: `<h1>FORMS DEMO</h1>

<form #form="ngForm" (ngSubmit)="submitForm(form.value)">

<div class="form-group">

<label>First Name:</label>

<input type="text" class="form-control" name="firstName" ngModel required>

</div>

<div class="form-group">

<label>Last Name</label>

<input type="text" class="form-control" name="lastName" ngModel required>

</div>

<div class="form-group">

<label>Gender</label>

</div>

<div class="radio">

<label>

<input type="radio" name="gender" value="Male" ngModel>

Male

</label>

</div>

<div class="radio">

<label>

<input type="radio" name="gender" value="Female" ngModel>

Female

</label>

</div>

<div class="form-group">

<label>COURCES</label>

</div>

<label class="checkbox-inline">

<input type="checkbox" value="angular" name="angular" ngModel> angular

</label>

<label class="checkbox-inline">

<input type="checkbox" value="spring" name="spring" ngModel> spring

</label>

<label class="checkbox-inline">

<input type="checkbox" value="phython" name="phython" ngModel> phython

</label>

<div class="form-group">

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

</div>

</form>

`,

inputs:[`date`]

})

export class AppComponent {

public date:Objec;t

submitForm(form: any): void{

console.log('Form Data: ');

console.log(form);

}

}

And now its time to go to module in module you need to import formsmodule from @angular/forms:

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

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

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

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

import { HelloComponent } from './hello/hello.component';

@NgModule({

declarations: [

AppComponent,

HelloComponent

],

imports: [

BrowserModule,**FormsModule**

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

now your form was ready.

Angular 2 Form Example 2 with hands on implementation:

Here I am giving only component:

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

import { HelloComponent } from './hello/hello.component';

@Component({

selector: 'app-root',

template: `<h1>FORMS DEMO</h1>

<form #demoForm="ngForm" (ngSubmit)="myfunction(demoForm.value)">

<div>NAME<input type="text" name="name" ngModel></div>

<div>ADDRESS<input type="text" name="addr" ngModel></div>

<div>PHONE<input type="number" name="pno" ngModel></div>

<div>EMAIL<input type="text" name="email" ngModel></div>

<input type="submit">

</form>

`,

})

export class AppComponent {

myfunction(result){

console.log(result);

}

}

If you observe above code snippet by comparing normal form you will get good understand in angular 2 form.

Two-way binding in angualar2:

You need to specify like [(ngModel)]=”name”

Angular2-states:

**STATE CLASS CLASS**

Controller has visited ng-touched(visited) ng-untouched (notvisited)

Value changed ng-pristine(notchanged) ng-dirty (changed)

Valid ng-invalid (notvalid) ng-valid (valid)

To see the state and applied class for each field put an id for field and getclass name like below:

<input type="text" #nameRef name="name" ngModel>{{nameRef.className}}

Initially you can see these class:

ng-untouched ng-pristine ng-valid

once you changed the value now you are going to see:

ng-valid ng-dirty ng-touched

component with validation:

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

import { HelloComponent } from './hello/hello.component';

@Component({

selector: 'app-root',

template: `<h1>FORMS DEMO</h1>

<form #demoForm="ngForm" (ngSubmit)="myfunction(demoForm.value)">

<div>NAME<input type="text" #nameRef required name="name" ngModel>{{nameRef.className}}</div>

<div>ADDRESS<input type="text" required name="addr" ngModel></div>

<div>PHONE<input type="number" required name="pno" ngModel></div>

<div>EMAIL<input type="email" required name="email" ngModel></div>

<input type="submit">

</form>

`,

styles:[`input.ng-invalid{background:red;}`]

})

export class AppComponent {

public result1:string;

myfunction(result){

console.log(result);

}

}

Applying two styles:

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

import { HelloComponent } from './hello/hello.component';

@Component({

selector: 'app-root',

template: `<h1>FORMS DEMO</h1>

<form #demoForm="ngForm" (ngSubmit)="myfunction(demoForm.value)">

<div>NAME<input type="text" #nameRef required name="name" ngModel>{{nameRef.className}}</div>

<div>ADDRESS<input type="text" required name="addr" ngModel></div>

<div>PHONE<input type="number" required name="pno" ngModel></div>

<div>EMAIL<input type="email" required name="email" ngModel></div>

<input type="submit">

</form>

`,

styles:[`

input.ng-invalid{border-left:5px solid red;}.ng-valid{border-left:5px solid green;};

`]

})

export class AppComponent {

public result1:string;

myfunction(result){

console.log(result);

}

}