

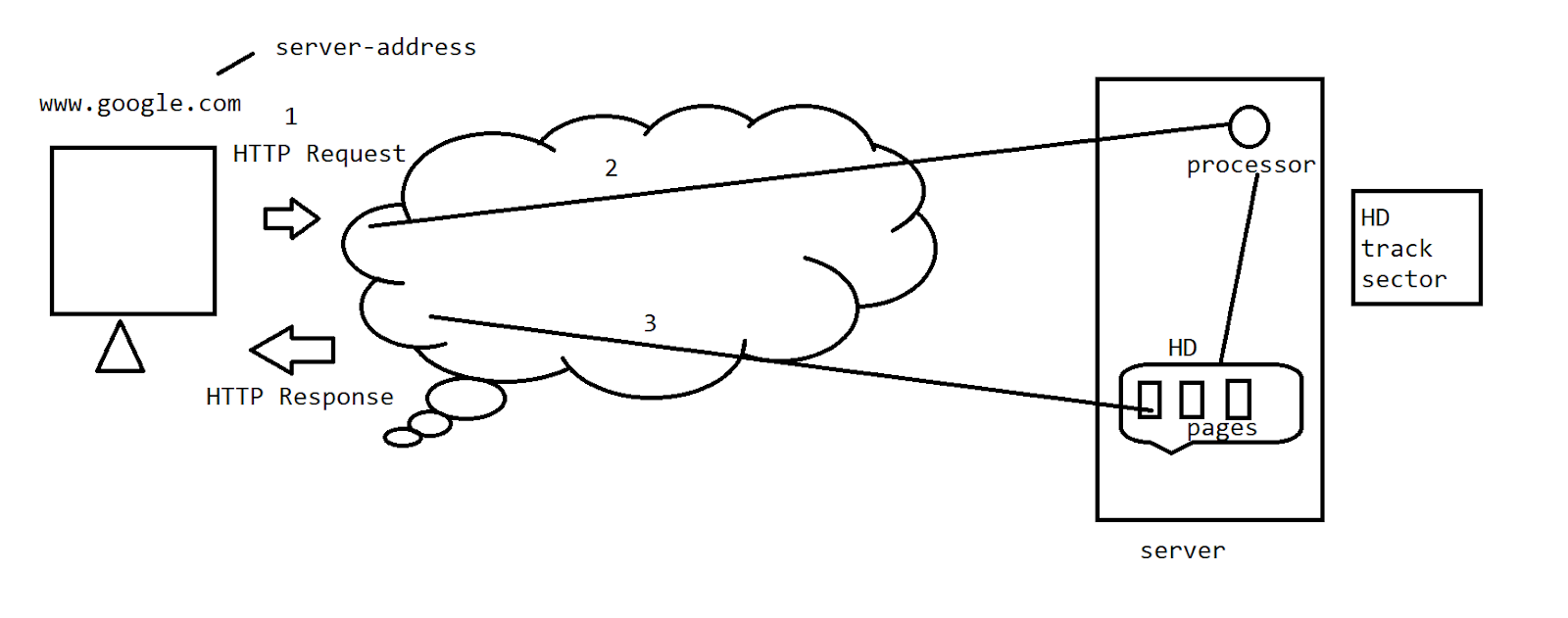
Markup language - it follows the documentation structure

Tag: It is an element that defines some task in it.

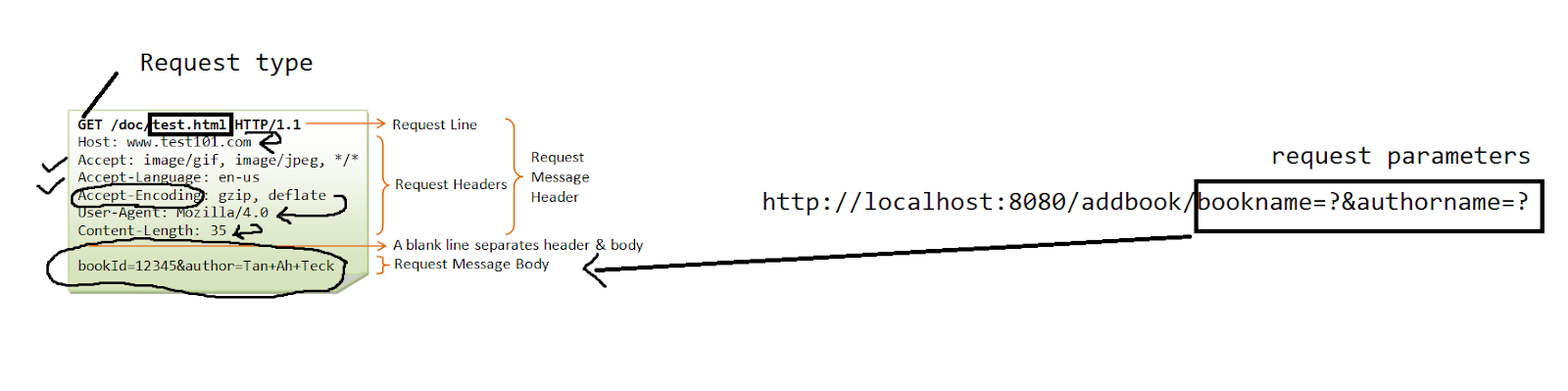
<>=><tag>

The markup language can be written over any text editor => execution of this file over the browser. =>interpreter in the browser.

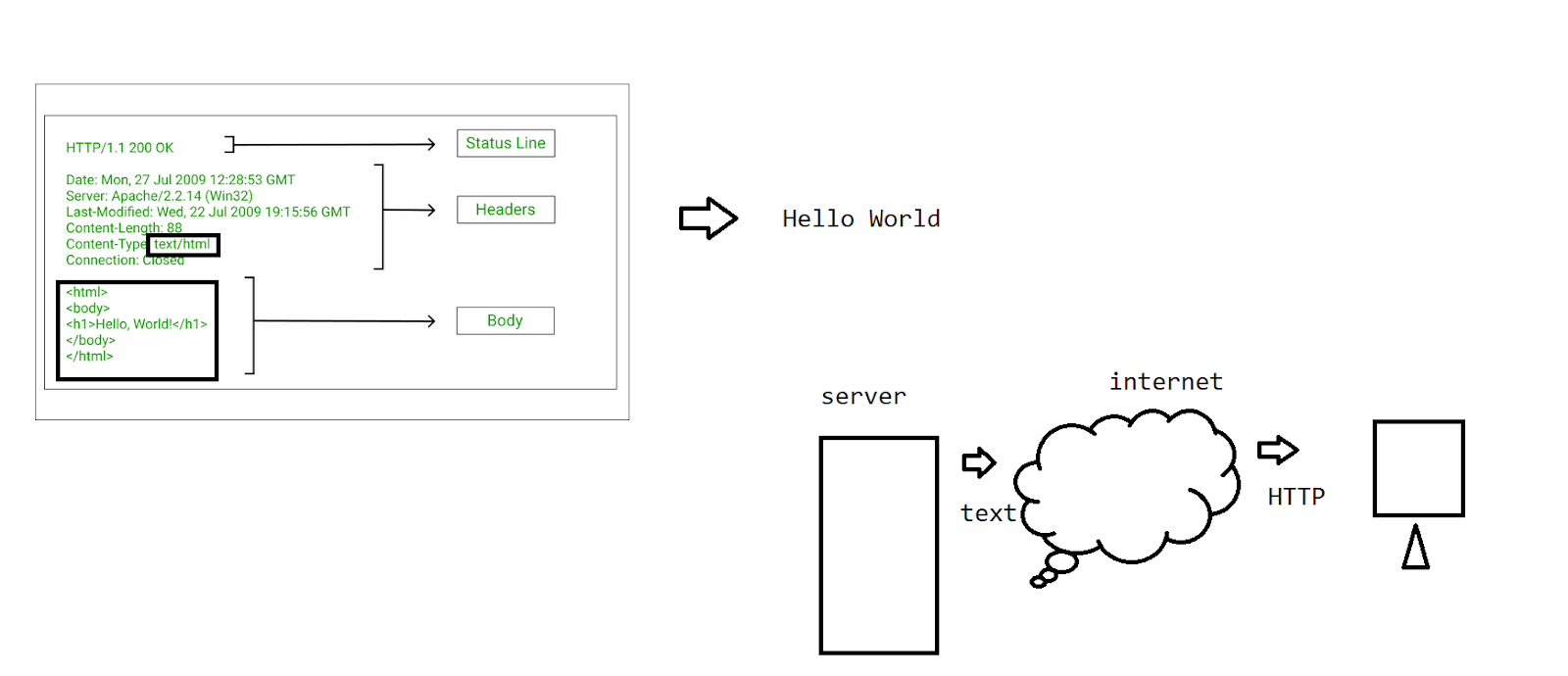
The language famous on the internet is HTML - Hypertext markup language.



HTTP Request



HTTP Response



HTML- W3C - World wide web consortium

* International standard registered over the internet.
* Content
* Tags used for formatting the pages
* Styled by using the CSS

HTML

<>

-Documentation formatting

-Forms

-Tables

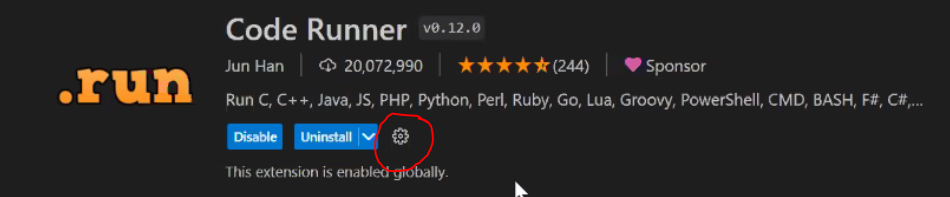
-Images

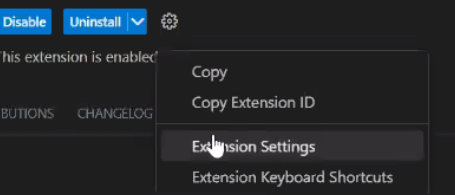
HTML 5.0 ->multimedia

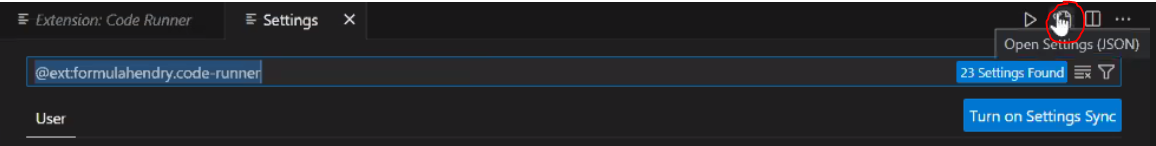
<https://code.visualstudio.com/download>

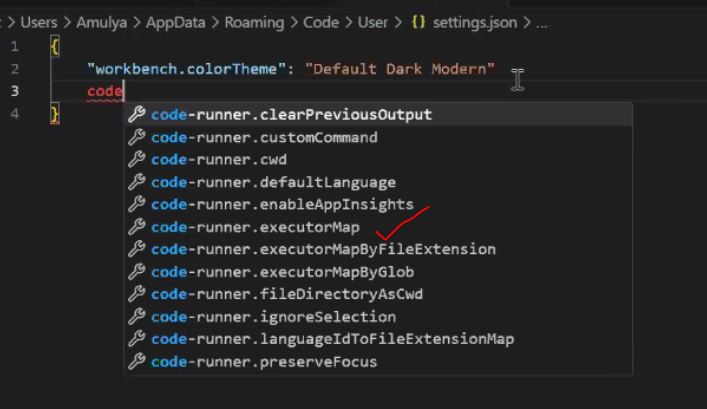


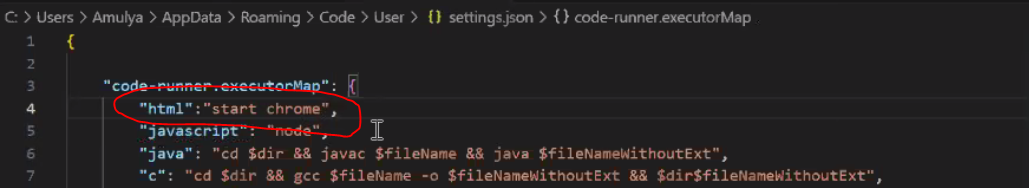












{

    "code-runner.executorMap": {

        "html": "start chrome",

        "javascript": "node",

        "java": "cd $dir && javac $fileName && java $fileNameWithoutExt",

        "c": "cd $dir && gcc $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",

        "cpp": "cd $dir && g++ $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",

        "objective-c": "cd $dir && gcc -framework Cocoa $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",

        "php": "php",

        "python": "python -u",

        "perl": "perl",

        "perl6": "perl6",

        "ruby": "ruby",

        "go": "go run",

        "lua": "lua",

        "groovy": "groovy",

        "powershell": "powershell -ExecutionPolicy ByPass -File",

        "bat": "cmd /c",

        "shellscript": "bash",

        "fsharp": "fsi",

        "csharp": "scriptcs",

        "vbscript": "cscript //Nologo",

        "typescript": "ts-node",

        "coffeescript": "coffee",

        "scala": "scala",

        "swift": "swift",

        "julia": "julia",

        "crystal": "crystal",

        "ocaml": "ocaml",

        "r": "Rscript",

        "applescript": "osascript",

        "clojure": "lein exec",

        "haxe": "haxe --cwd $dirWithoutTrailingSlash --run $fileNameWithoutExt",

        "rust": "cd $dir && rustc $fileName && $dir$fileNameWithoutExt",

        "racket": "racket",

        "scheme": "csi -script",

        "ahk": "autohotkey",

        "autoit": "autoit3",

        "dart": "dart",

        "pascal": "cd $dir && fpc $fileName && $dir$fileNameWithoutExt",

        "d": "cd $dir && dmd $fileName && $dir$fileNameWithoutExt",

        "haskell": "runhaskell",

        "nim": "nim compile --verbosity:0 --hints:off --run",

        "lisp": "sbcl --script",

        "kit": "kitc --run",

        "v": "v run",

        "sass": "sass --style expanded",

        "scss": "scss --style expanded",

        "less": "cd $dir && lessc $fileName $fileNameWithoutExt.css",

        "FortranFreeForm": "cd $dir && gfortran $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",

        "fortran-modern": "cd $dir && gfortran $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",

        "fortran\_fixed-form": "cd $dir && gfortran $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",

        "fortran": "cd $dir && gfortran $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",

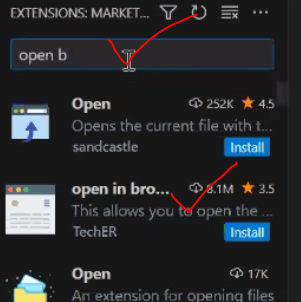
        "sml": "cd $dir && sml $fileName"

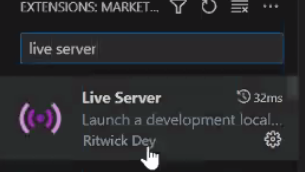
    },

    "window.zoomLevel": 2,

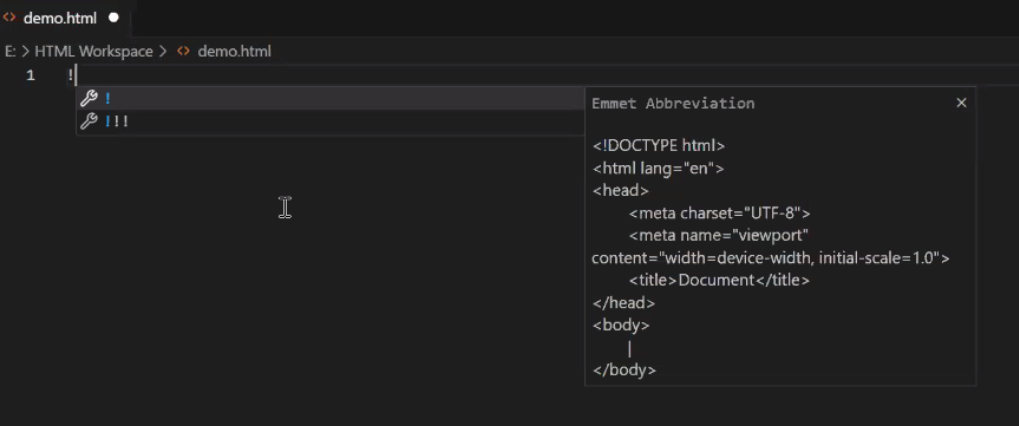
    "workbench.colorTheme": "Visual Studio Light"

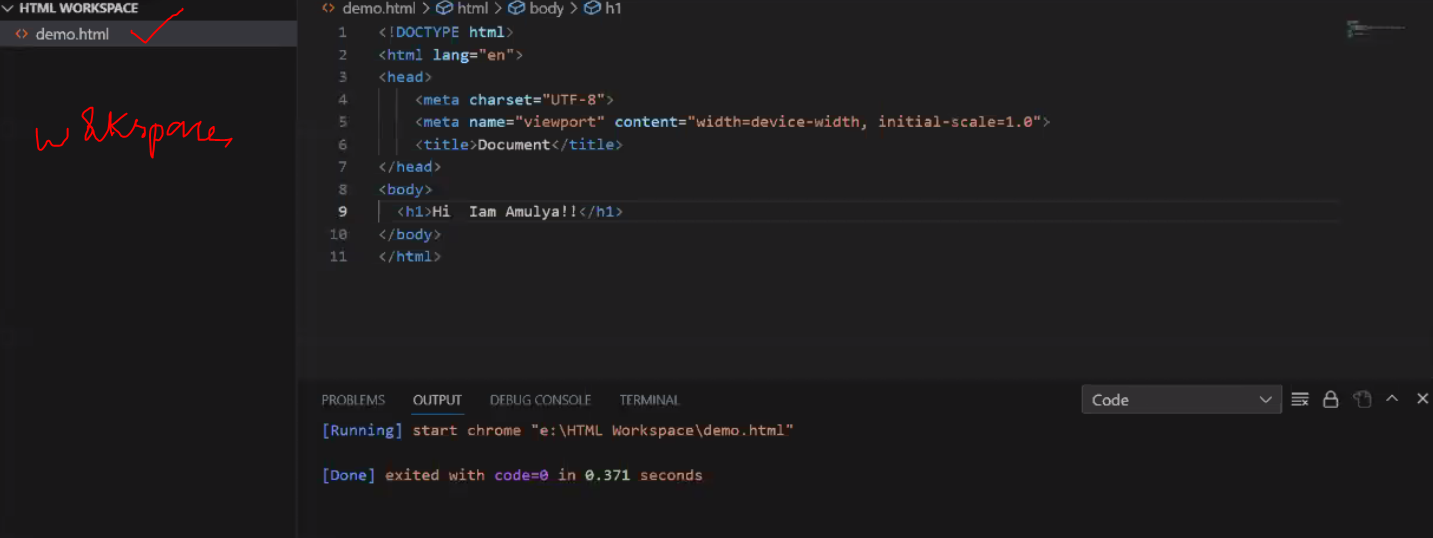
}

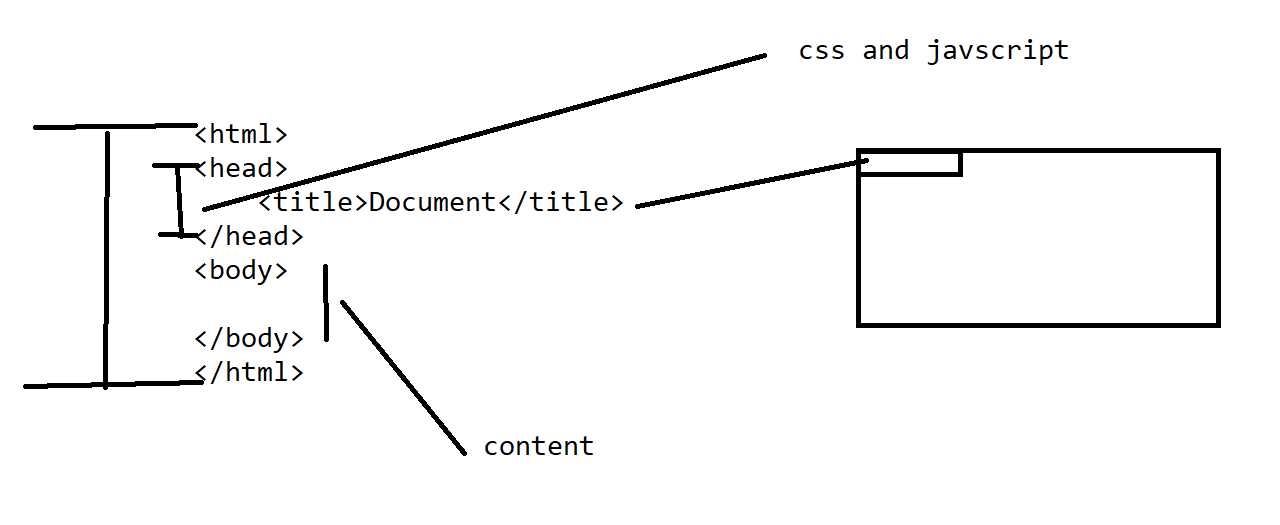




CNTL +N - new file







<div>

* Used for the independent content

<html>

<head>

    <title>Document</title>

</head>

<body>

  <div style="background-color: aqua;">

    <h1>Hi users</h1>

  </div>

  <div style="background-color: bisque;">

    <h1>Hi learners </h1>

  </div>

</body>

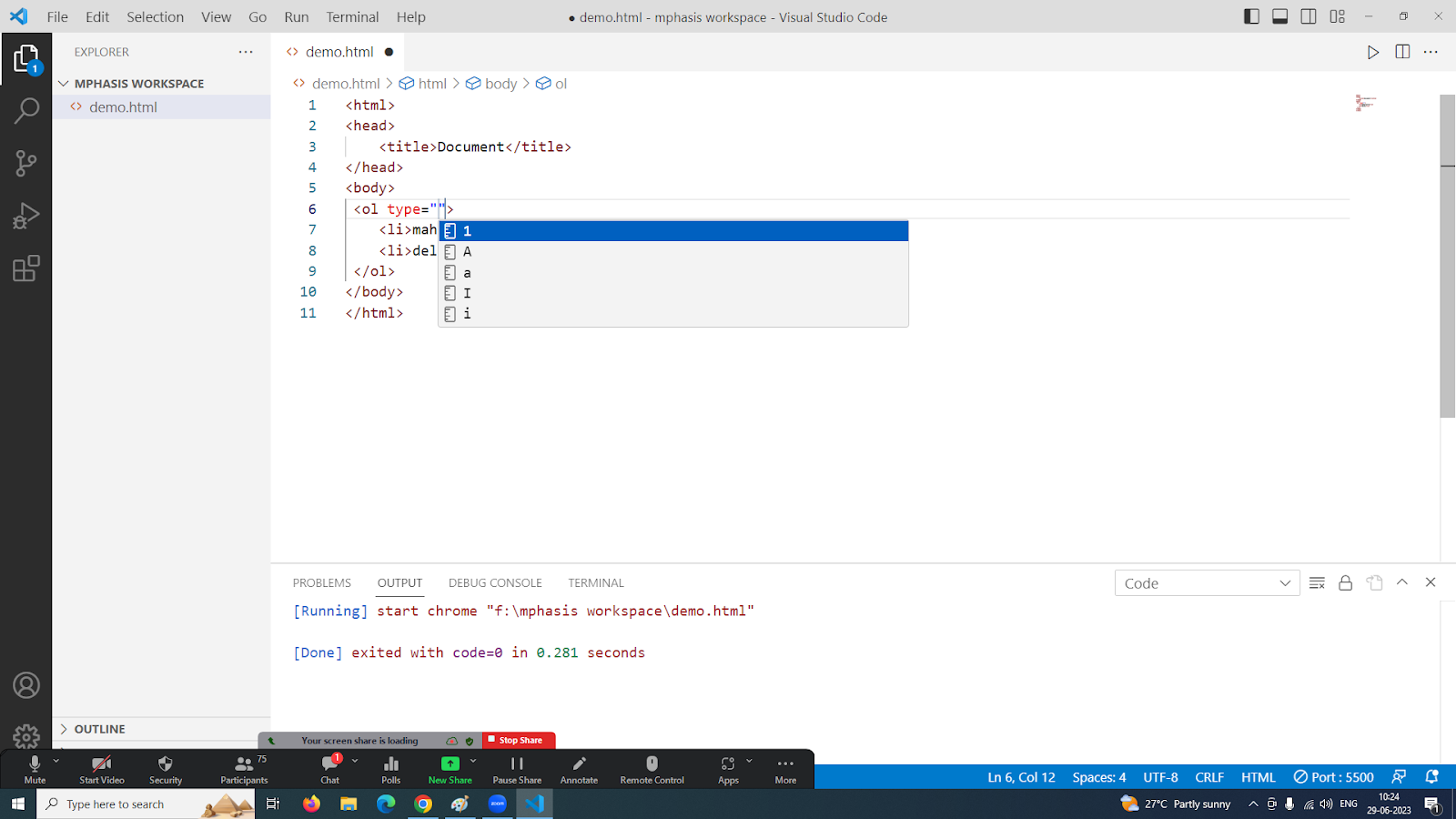
</html>

* List

The view is defined in an organized manner.

1. ol(ordered list)
2. ul(unordered list)
3. Description(dl)

ol



<html>

<head>

    <title>Document</title>

</head>

<body>

 <ol type="I">

    <li>maharastra</li>

    <li>delhi</li>

 </ol>

</body>

</html>

Ul: - circle, disc, none, square

<html>

<head>

    <title>Document</title>

</head>

<body>

 <ul type="none">

    <li>maharastra</li>

    <li>delhi</li>

 </ol>

</body>

</html>

dl:

dt:data term

dd-data defintion

<html>

<head>

    <title>Document</title>

</head>

<body>

<dl>

    <dt>what is java ?</dt>

    <dd>It is an OOP language</dd>

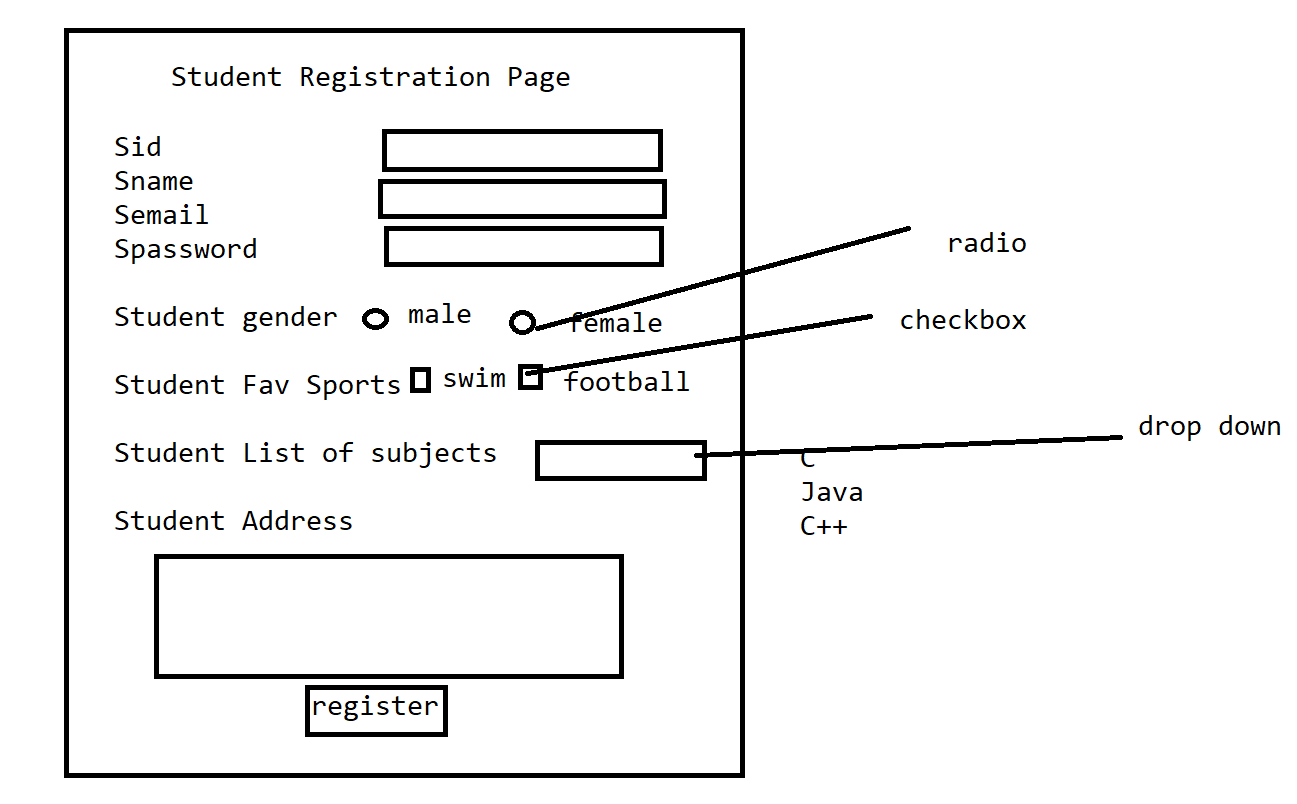
</body>

</html>

<forms>

Note : Take the background structuring with a table .

Task : 10 min –center



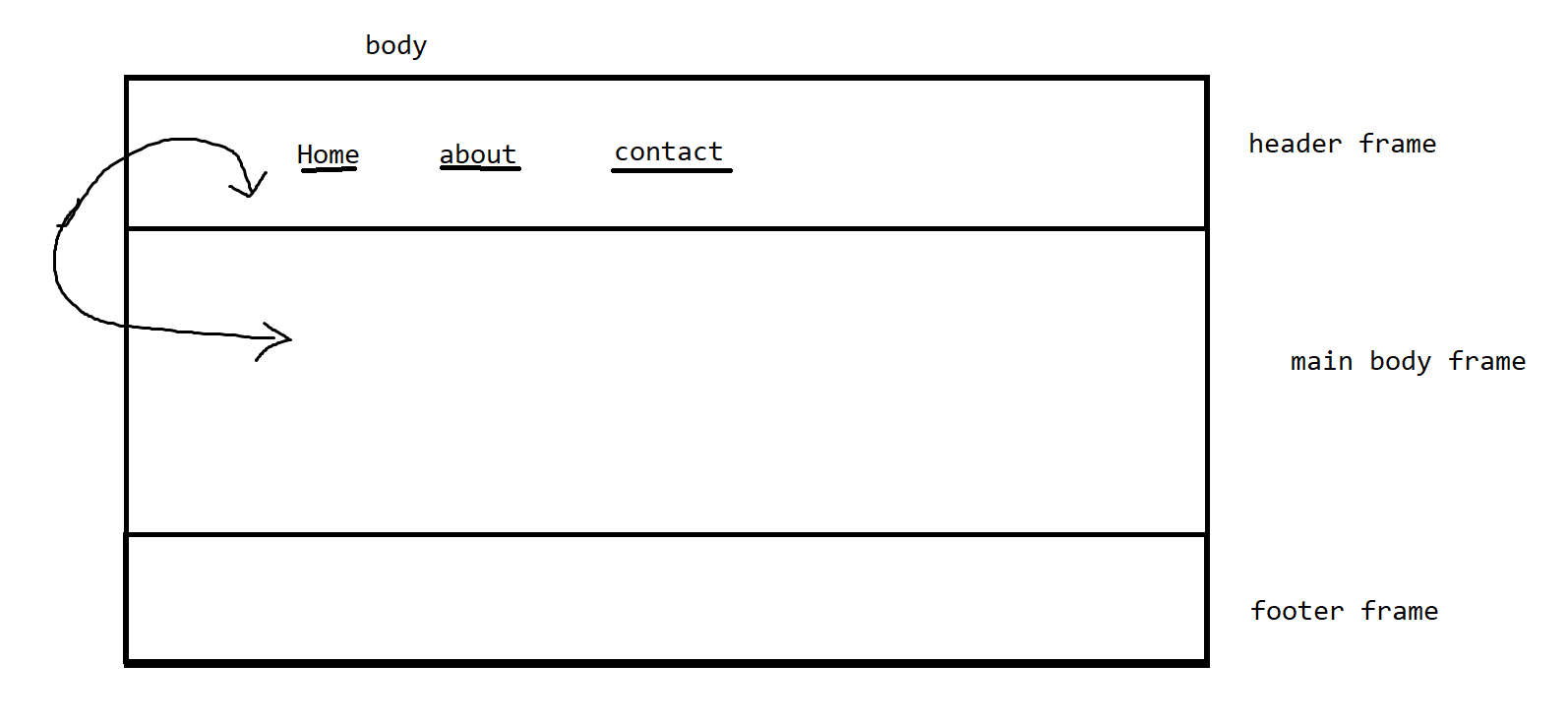
HTML <5

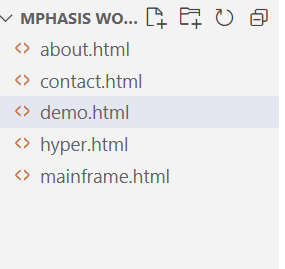
<frame>

<frameset>

HTML >5

<iframe>





//demo

<html>

<head>

    <title>Document</title>

</head>

<body>

<br>

<br>

<br>

<br>

<h1><i><marquee>Welcome to phase 4.....</marquee></i></h1>

</body>

</html>

//mainframe

<body>

<iframe src="hyper.html" name="top" width="100%" height="100" frameBorder="0" ></iframe>

<iframe src="demo.html" name="bottom" width="100%" height="900" frameBorder="0" ></iframe>

</body>

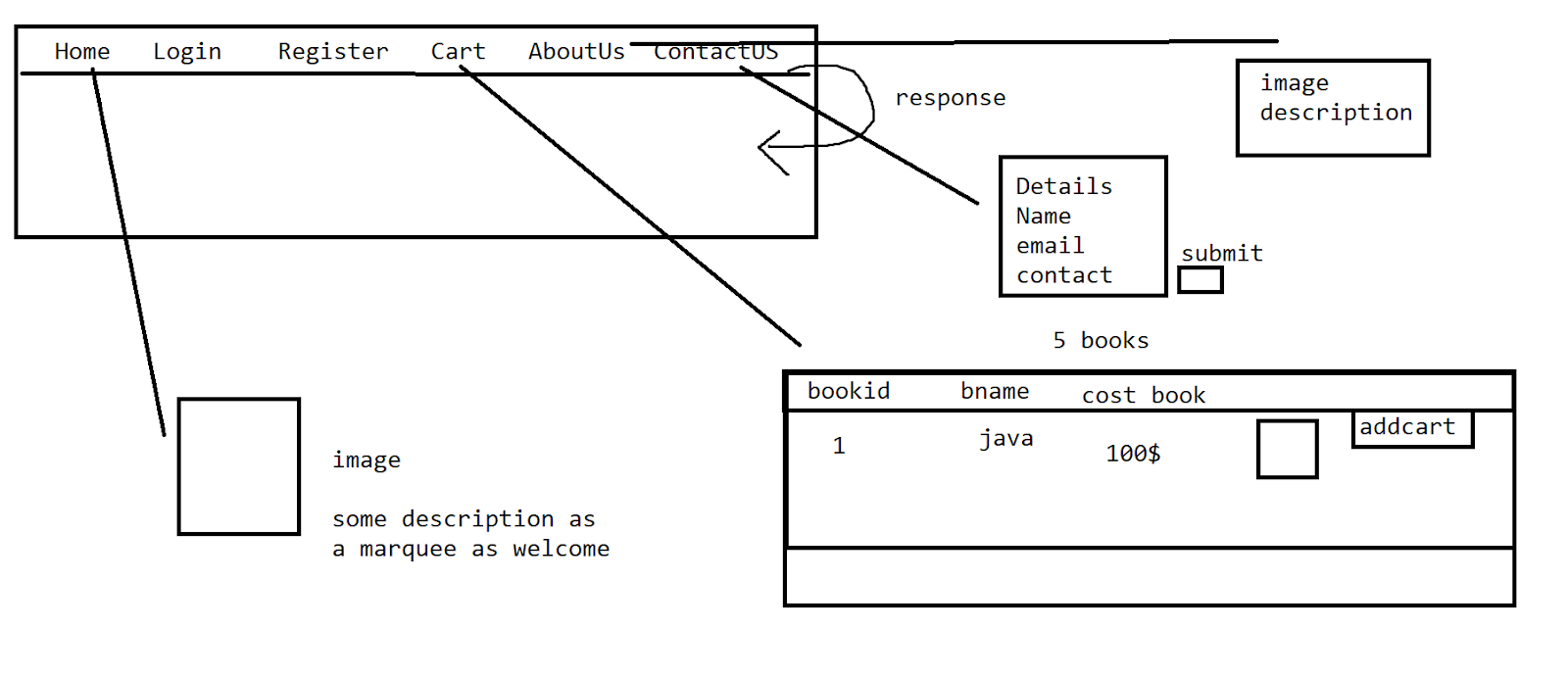
//hyper

<table border="1" width="100%">

<tr><td><a href="demo.html" target="bottom">Home</a></td><td> <a href="contact.html" target="bottom">contact</a></td><td> <a href="about.html" target="bottom">about</a></td></tr>

</table>

Task - 15 min



<img src="java.jpg" width="100" height="100">

<img src="python.jpg" width="100" height="100">

Images of equal size in cart

Theme : ecomerce , comercial

Multimedia - audio and video

Audio : mp3,wav,ogg

Video : mp4,ogg,webM



<audio controls>

<source src="file:///C:/Users/Karthik/Downloads/sample-3s.mp3" type="audio/mpeg">

</audio>

<video controls height="30%" width="30%">

<source src="file:///C:/Users/Karthik/Downloads/videoplayback.mp4" type="video/mp4">

</video>

In About us - video

#CSS

Cascading style sheet

We use this styling over the webpages to get the look and feel.

Styles are applied on the tags .

<tag attributes-(we are going to define the styles)>

//selector - tag name or it can be an identifier

selector {

<key,value >

property-name1:value;

property-name2:value;

property-name3:value;

}

Types of CSS:

1. Inline
2. Internal
3. External \*\*\*\*

Inline: if the styles are happening within the tag then we call it inline.

<html>

<head>

    <title>Document</title>

</head>

<body>

<h1 style="color: aquamarine;margin-left: 40px;font-size: x-large;"><i>Welcome to phase 4.....</i></h1>

</body>

</html>

–the possibility of duplication of the properties

Internal -> It is used for providing the reusability of the properties to the tag

<html>

<head>

    <style>

        h1{

            color: rgb(255, 195, 127);

            margin-left: 40px;

            font-size: x-large;

        }

    </style>

</head>

<body>

<h1><i>Welcome to phase 4.....</i></h1>

<h1><i>Hi learners.....</i></h1>

</body>

</html>

If h1 has independent properties?

1. id
2. class
3. group
4. Universal

-id

<html>

<head>

    <style>

        #one{

            color: rgb(255, 195, 127);

            margin-left: 40px;

            font-size: x-large;

        }

        #two{

            color: rgb(255, 127, 142);

            margin-left: 90px;

            font-size: xx-large;

        }

    </style>

</head>

<body>

<h1 id="one"><i>Welcome to phase 4.....</i></h1>

<h1 id="two"><i>Hi learners.....</i></h1>

</body>

</html>

<input type-”text” name=””  id=””>

So the is a name mismatch with id in the input tag and the style identifier.

<html>

<head>

    <style>

        .one{

            color: rgb(255, 195, 127);

            margin-left: 40px;

            font-size: x-large;

        }

        .two{

            color: rgb(255, 127, 142);

            margin-left: 90px;

            font-size: xx-large;

        }

    </style>

</head>

<body>

<h1 class="one"><i>Welcome to phase 4.....</i></h1>

<h1 class="two"><i>Hi learners.....</i></h1>

</body>

</html>

-grouping

When the tags have the same property to be applied.- group

<html>

<head>

    <style>

        .one{

            color: rgb(255, 195, 127);

            margin-left: 40px;

            font-size: x-large;

        }

        h1,p{

            color: rgb(255, 127, 142);

            margin-left: 90px;

            font-size: xx-large;

        }

    </style>

</head>

<body>

<h1><i>Hi learners.....</i></h1>

<p>Hey How are you</p>

</body>

</html>

Universal

<html>

<head>

    <style>

        \*{

            color: rgb(255, 127, 142);

            margin-left: 90px;

            font-size: xx-large;

        }

    </style>

</head>

<body>

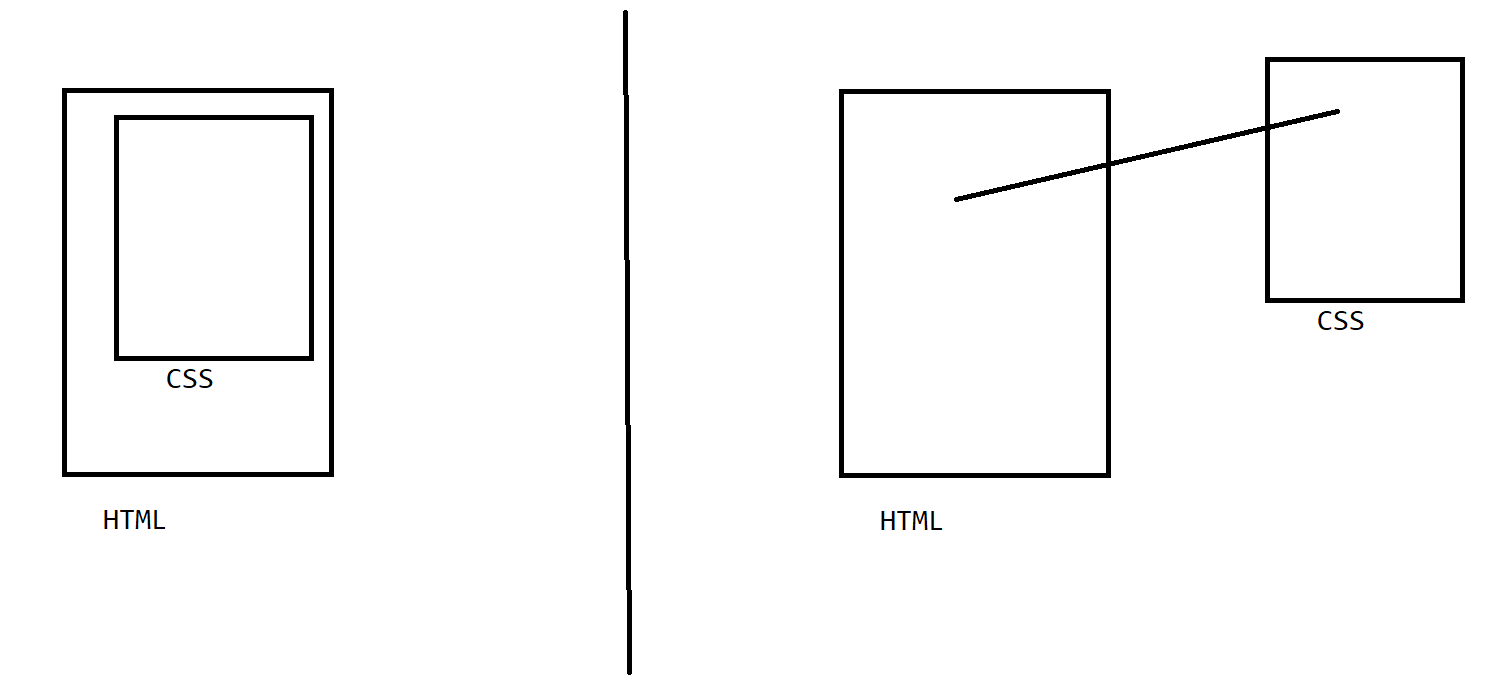
<h1><i>Hi learners.....</i></h1>

<p>Hey How are you</p>

</body>

</html>

HTML -Documentation - <body> </body>



–loosely coupled

<html>

<head>

  <link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<h1 class="one"><i>Hi learners.....</i></h1>

<p class="one">Hey How are you</p>

</body>

</html>

.one{

    color: rgb(255, 127, 142);

    margin-left: 90px;

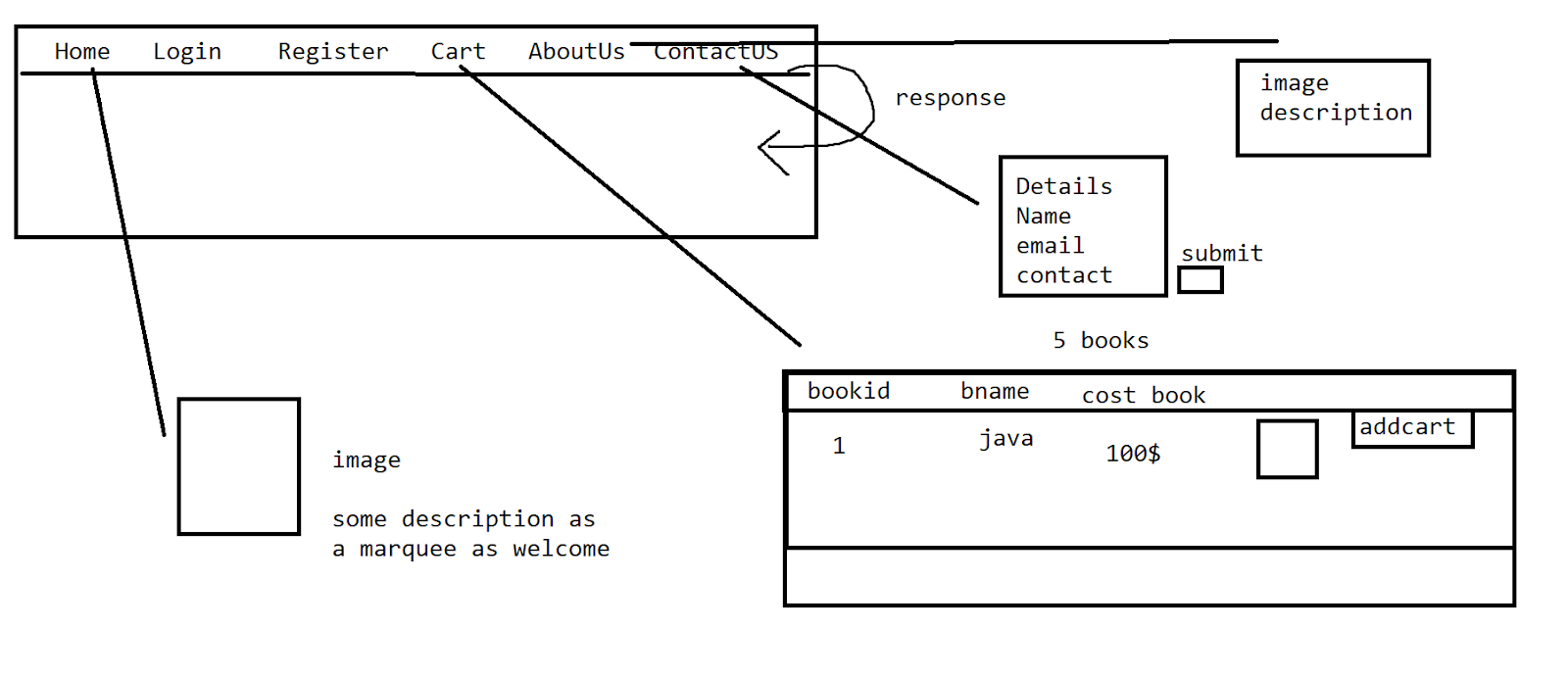
    font-size: xx-large;

}

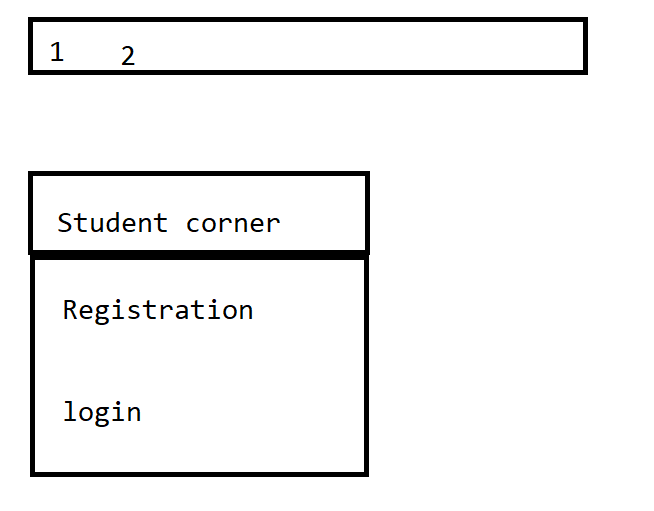
Task : 10 min

How to write text on an image ?

Task : 15 min



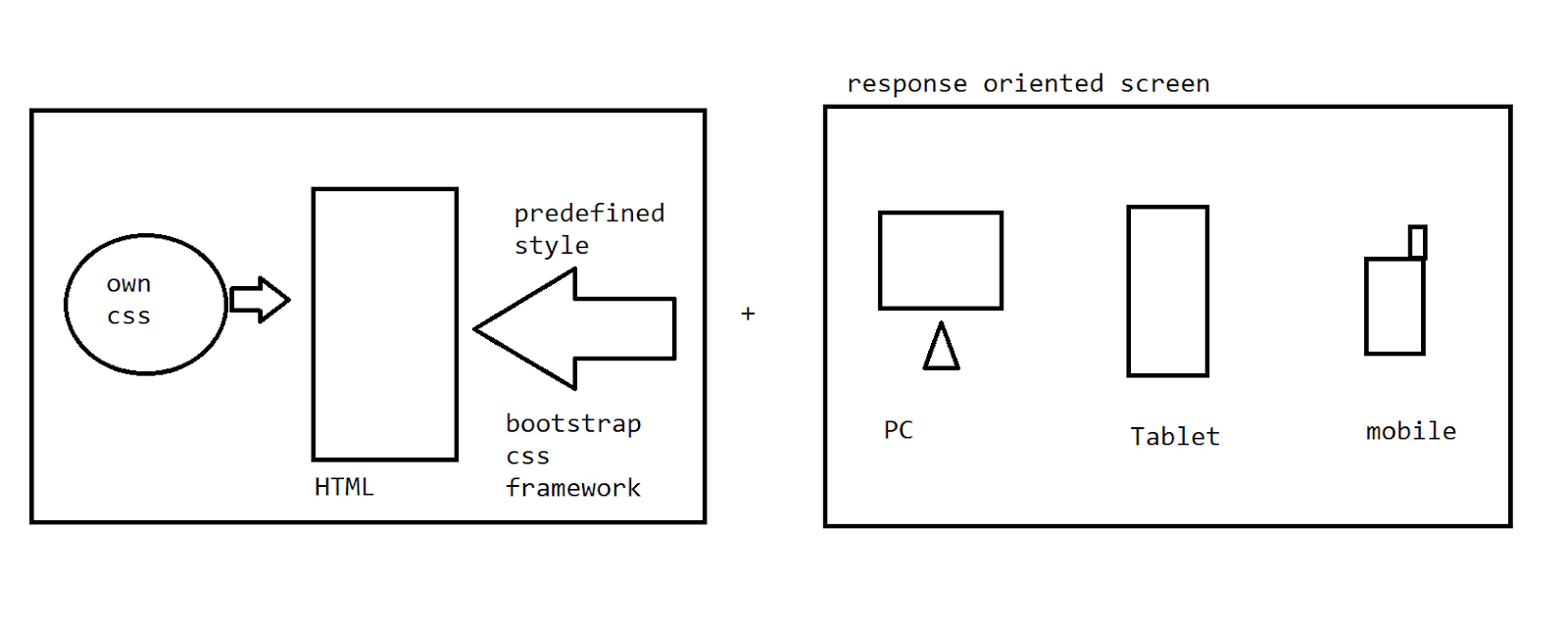
In the books, the cart maintains 2 pages -pagination



Apply bgcolor or bg images

Use external css

Bootstrap :



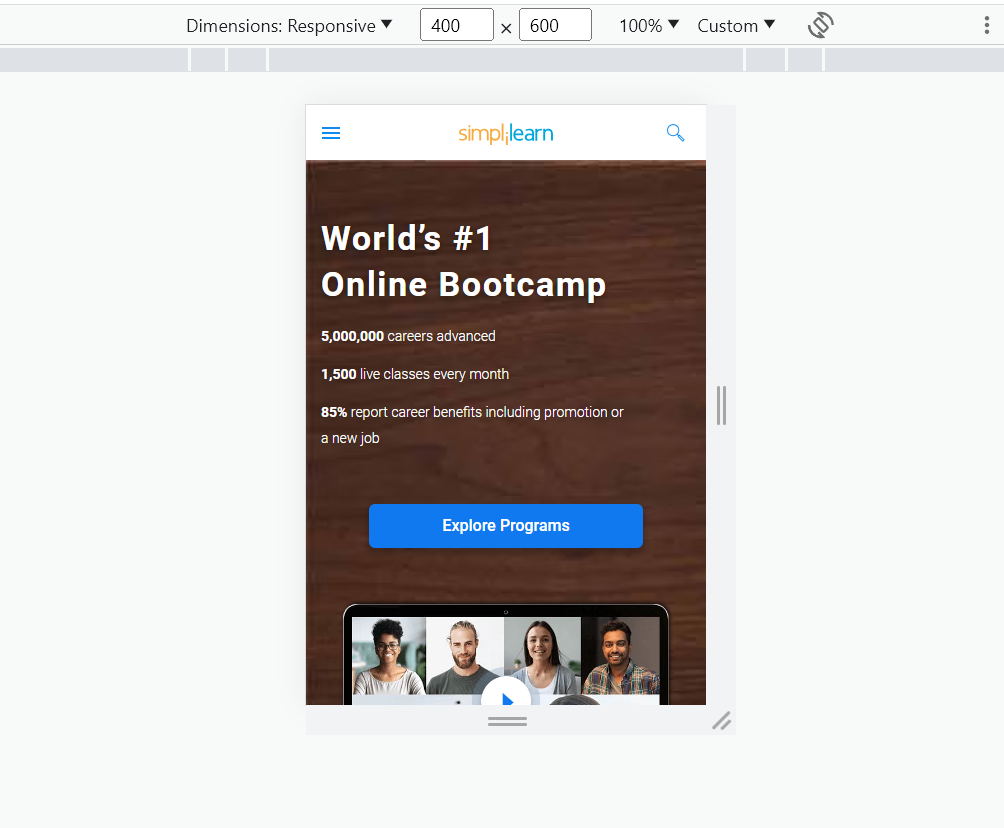
It is framework of css and an open source

It provides a ready made structure of the css components for the tags

Response oriented screens

->according to the dimensionality of the device the screens are going to get auto adjusted .

Cnt+shft+i ===>cntl+shft+m



Model -1

Download all the bootstrap-oriented css config locally and have the link of it in the application.

<https://github.com/twbs/bootstrap/releases/download/v5.0.2/bootstrap-5.0.2-dist.zip>

Extract it

<html>

<head>

    <link rel="stylesheet" type="text/css" href="file:///C:/Users/Karthik/Downloads/bootstrap-5.2.1-dist/bootstrap-5.2.1-dist/css/bootstrap.css"

</head>

<body>

<h1 class="one"><i>Hi learners.....</i></h1>

<p class="one">Hey How are you</p>

</body>

</html>

Note :

Any style if we want to define into the application of our own then we need to define the user-defined css on top of the bootstrap config .

 <link rel="stylesheet" type="text/css" href="style.css">

    <link rel="stylesheet" type="text/css" href="file:///C:/Users/Karthik/Downloads/bootstrap-5.2.1-dist/bootstrap-5.2.1-dist/css/bootstrap.css"

Model -2

Have the links of the bootstrap-oriented css config into the application.

<html>

<head>

 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

</head>

<body>

    <div class="dropdown">

        <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-expanded="false">

          Dropdown button

        </button>

        <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">

          <li><a class="dropdown-item" href="#">Action</a></li>

          <li><a class="dropdown-item" href="#">Another action</a></li>

          <li><a class="dropdown-item" href="#">Something else here</a></li>

        </ul>

      </div>

</body>

</html>

<https://getbootstrap.com/docs/5.0/components/dropdowns/> -bootstrap 5

<https://hackerthemes.com/bootstrap-cheatsheet/#form__grid> -bootstrap 4

Typography

-text

Blockquote

-quotations

<html>

<head>

 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

</head>

<body>

    <figure>

        <blockquote class="blockquote">

          <p>A well-known quote, contained in a blockquote element.</p>

        </blockquote>

        <figcaption class="blockquote-footer">

          Someone famous in <cite title="Source Title">Source Title</cite>

        </figcaption>

      </figure>

</body>

</html>

Task:

Take an image with the caption as quotation and need to have the author name as your name.

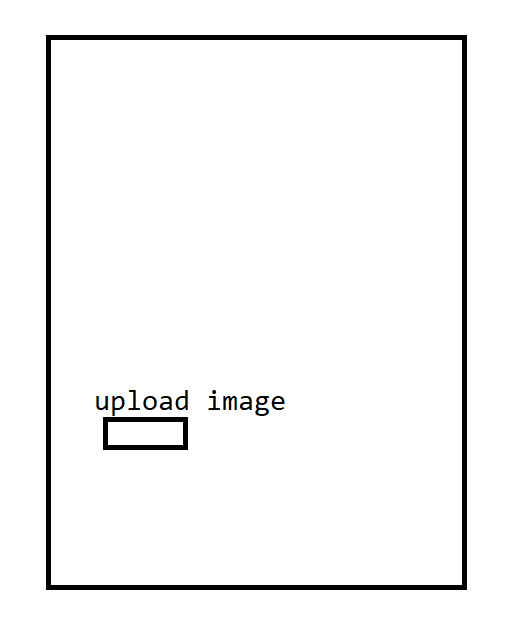
Task:

Update the cart with imgthumbnail and the table model

Task :

Update the registration page and login page with the bootstrap

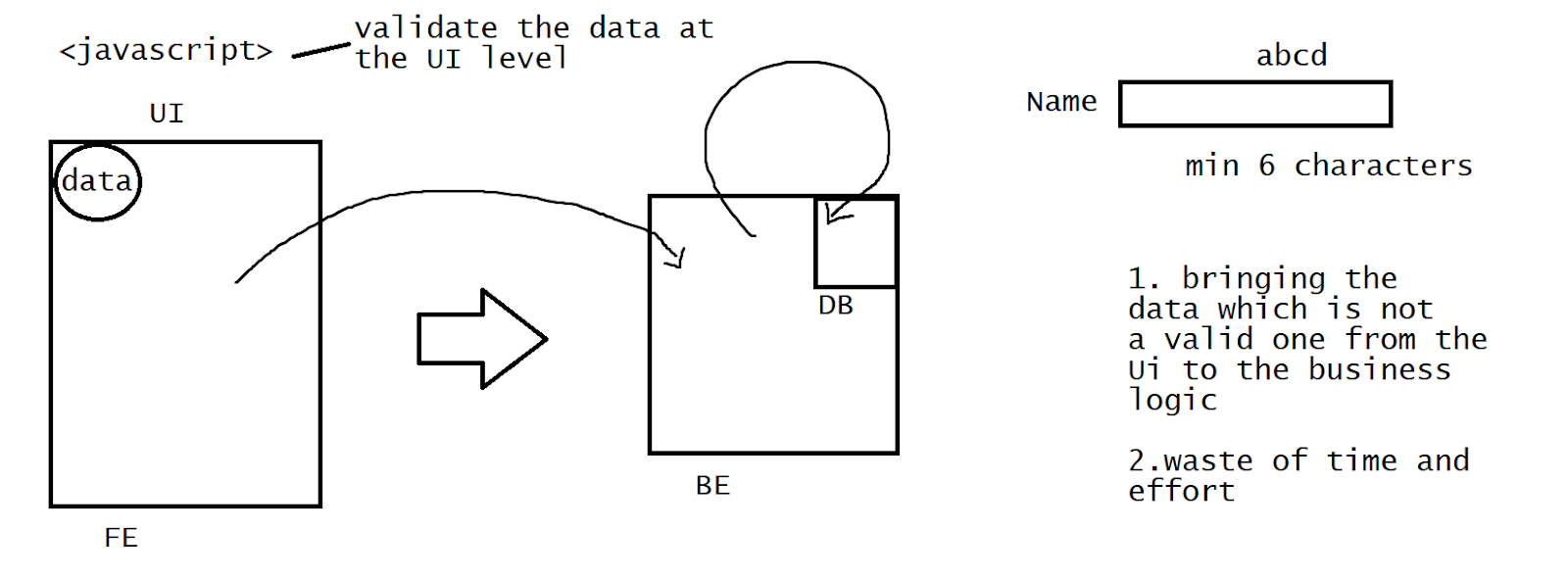
* Add a file upload to upload an image



Update the pages with pagination , home screen need to have carousel , update the button

Drop down menus

>Javascript



It is a scripting language -Object based

HTML⇒ static pages (no action is performed)

HTML + javascript =>dynamic page =>response

It is a light weighted language .

* It is used to validate the data over the client side only, by reducing the burden of the server side .
* Js is going to get loaded by the browser itself.
* Js as a dynamically typed language .[int a, float b…..]=>var,let

int a=10;

var a=10;

* Js uses the functional approach

Limitation : It doesnt what are the things that are being logically done at the BE side .

Js is invoked -> an action ->event

<head> – defining the js

We click on a button =>action generated =>function validation(){}

<!DOCTYPE html>

<html lang="en">

<head>

    <script>

        function msg(){

            document.write("hi learners Welcome!!")

        }

    </script>

</head>

<body>

    <form>

        <input type="button" value="click" onclick="msg()">

    </form>

</body>

</html>

Note: js need to be used more as a functional component for the actions that are requested from the HTML which is going to increase the modularity and also the reusability

Document => HTML

Js ->document.write()

>Dynamically typed language

<!DOCTYPE html>

<html lang="en">

<head>

    <script>

            var s="the output is "

            function msg(){

                var a=10

                var b=5.3

                document.write(s+(a+b)/2)

            }

    </script>

</head>

<body>

    <form>

        <input type="button" value="click" onclick="msg()">

    </form>

</body>

</html>

HTML => 3 components of pop up boxes

>alert

>confirm

>prompt

>alert

It is a browser-level component => window.alert()

 alert(s+(a+b)/2)

>confirm

When we have 2 cases which one case needs to get chosen?

 function msg(){

                var input=window.confirm("do u want to procced??")

                if(input==true){

                    document.write("going to the main logic !!")

                }

                else{

                    alert("the applciation got termited !!")

                }

            }

#prompt

User input =>an action need to be generated

function msg(){

                var input1=parseInt(window.prompt("enter the value a","0"));

                console.log("the value of input1 is "+input1)

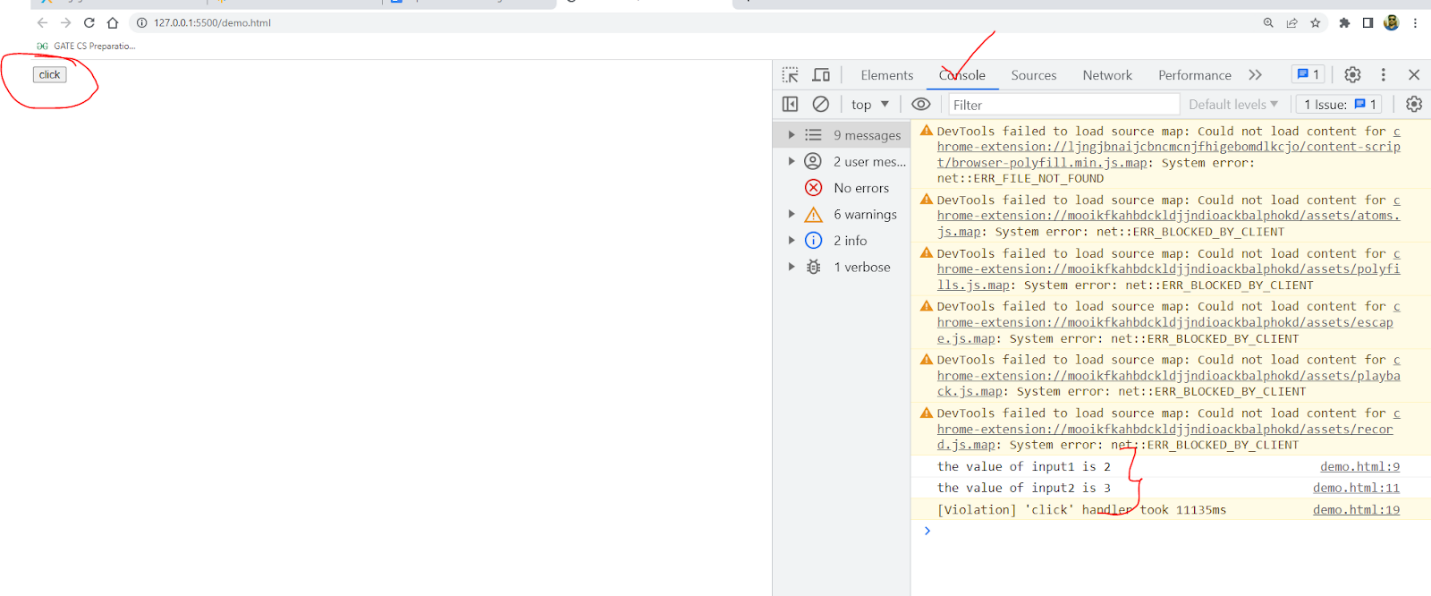
                var input2=parseInt(window.prompt("enter the value b","0"));

                console.log("the value of input2 is "+input2)

                alert("the addition is "+(input1+input2))

            }

* Use loggers as the debuggers of the values



Task :5 min

Take a number in a prompt and validate whether it is prime or not

If not mention it in alert

//validation of forms

<!DOCTYPE html>

<html lang="en">

<head>

    <script>

            function checkValidationOfForm(){

                let name=document.f1.user.value;

                console.log("the name entered is "+name)

                let pass=document.f1.pwd.value;

                console.log("the password enetered is "+pass)

                if(name==""){

                    alert("name should not be empty")

                }

                else if(pass==""){

                    alert("password should not be empty")

                }

                else if(name==="admin"&&pass=="admin"){

                    document.write ("Hey welcome!! "+name)

                }

                else{

                    alert("pls chck the credentials ")

                }

            }

    </script>

</head>

<body>

    <form name="f1">

        username<input type="text" name="user"><br>

        password<input type="password" name="pwd"><br>

        <input type="button" value="login" onclick="checkValidationOfForm()">

    </form>

</body>

</html>

Note: Use external js file for the validation purpose so as to make the HTML code look more

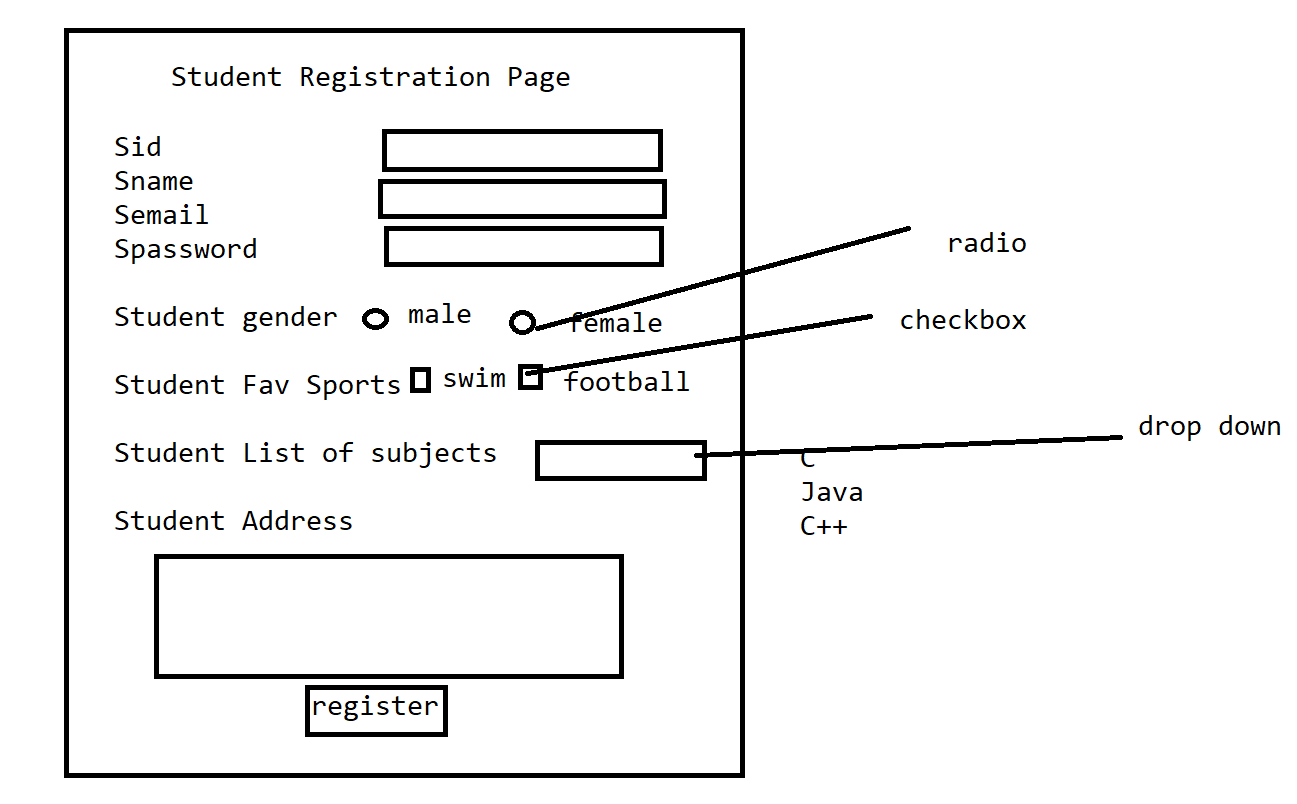
Loosely coupled.

Task

//min length of the name must be 6

//Task

Validate the registration form



Validate the data if the data is valid then give a msg as “registered successfully <name of the student>”  or else we need to get the alerts.



Single status response:

function checkValidationOfForm(){

    let name=document.f1.user.value;

    let statusofname=""

    let statusofpwd=""

    let statusatlast=""

    console.log("the name enetered is "+name)

    let pass=document.f1.pwd.value;

    console.log("the password enetered is "+pass)

 //min length of the name mst be 6

    if(name==""){

        statusofname="name should not be empty"

    }

   if(pass==""){

        statusofpwd="password should not be empty"

    }

  if(name==="adminadmin"&&pass=="adminadmin"){

        document.write ("Hey welcome!! "+name)

    }

    else{

        statusatlast="pls chck the credentials "

    }

    alert(statusofname+"\n"+statusofpwd+"\n"+statusatlast)

}

2nd method

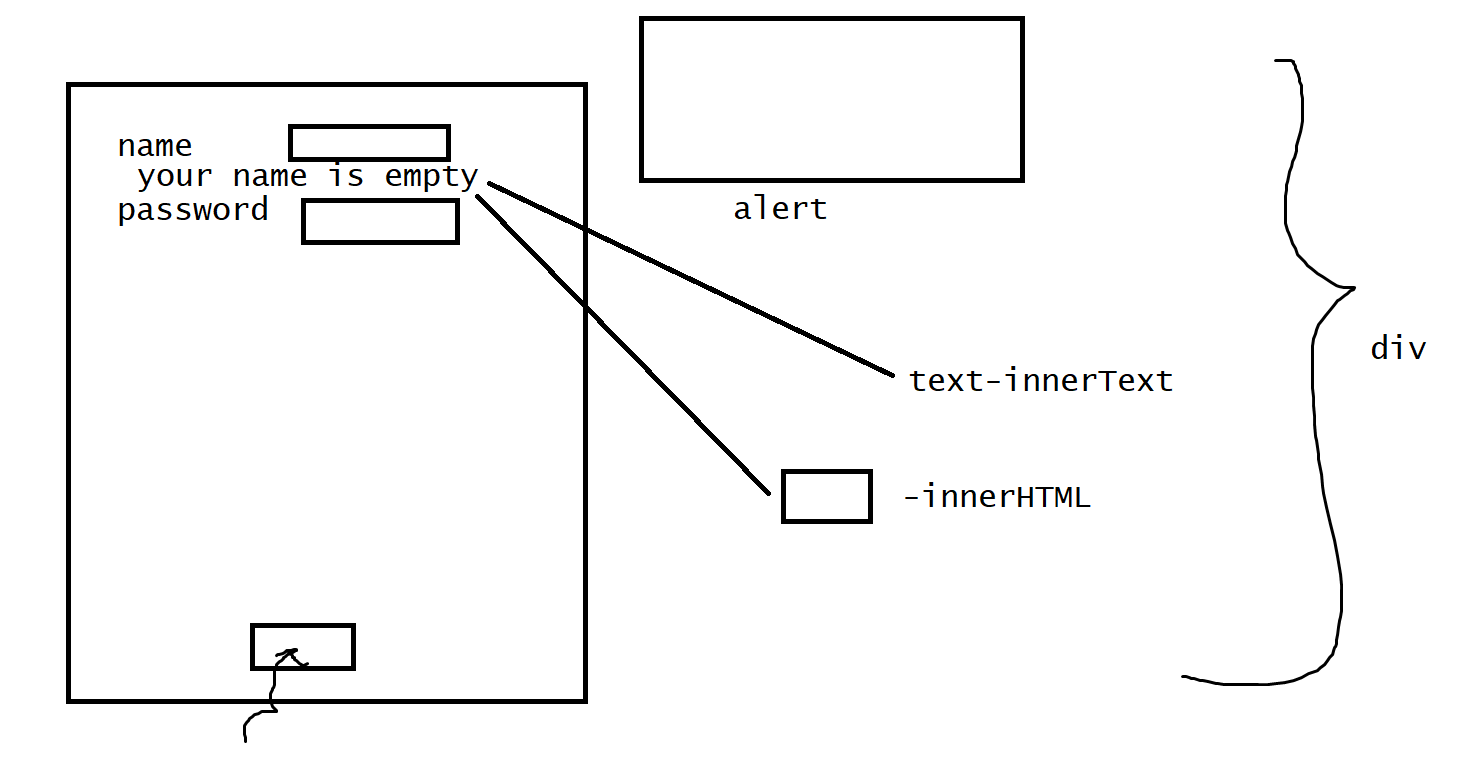
//let status=[]

status.push(“error msgs”)

If the length >0

alert()

//innerHTML and innerText



<!DOCTYPE html>

<html lang="en">

<head>

    <script type="text/javascript" src="external.js"> </script>

</head>

<body>

    <input type="button" value="click for login" onclick="msg()">

    <div id="place"></div>

</body>

</html>

//js

function msg(){

var tag="<form name='f1'>username<input type='text' name='user'><br>password<input type='password' name='pwd'><br></form>";

document.getElementById("place").innerHTML=tag

}

//innerText

<!DOCTYPE html>

<html lang="en">

<head>

    <script>

            function checkValidationOfForm(){

                let name=document.f1.user.value;

    let statusofname=""

    let statusofpwd=""

    console.log("the name enetered is "+name)

    let pass=document.f1.pwd.value;

    console.log("the password enetered is "+pass)

    if(name==""){

        statusofname="name should not be empty"

        document.getElementById("user").innerText=statusofname

    }

   if(pass==""){

        statusofpwd="password should not be empty"

        document.getElementById("pass").innerText=statusofpwd

    }

  if(name==="adminadmin"&&pass=="adminadmin"){

        document.write ("Hey welcome!! "+name)

    }

 }

    </script>

</head>

<body>

    <form name="f1">

        username<input type="text" name="user"><br>

        <div id="user" style="color: red;"></div>

        password<input type="password" name="pwd"><br>

        <div id="pass" style="color: red;"></div>

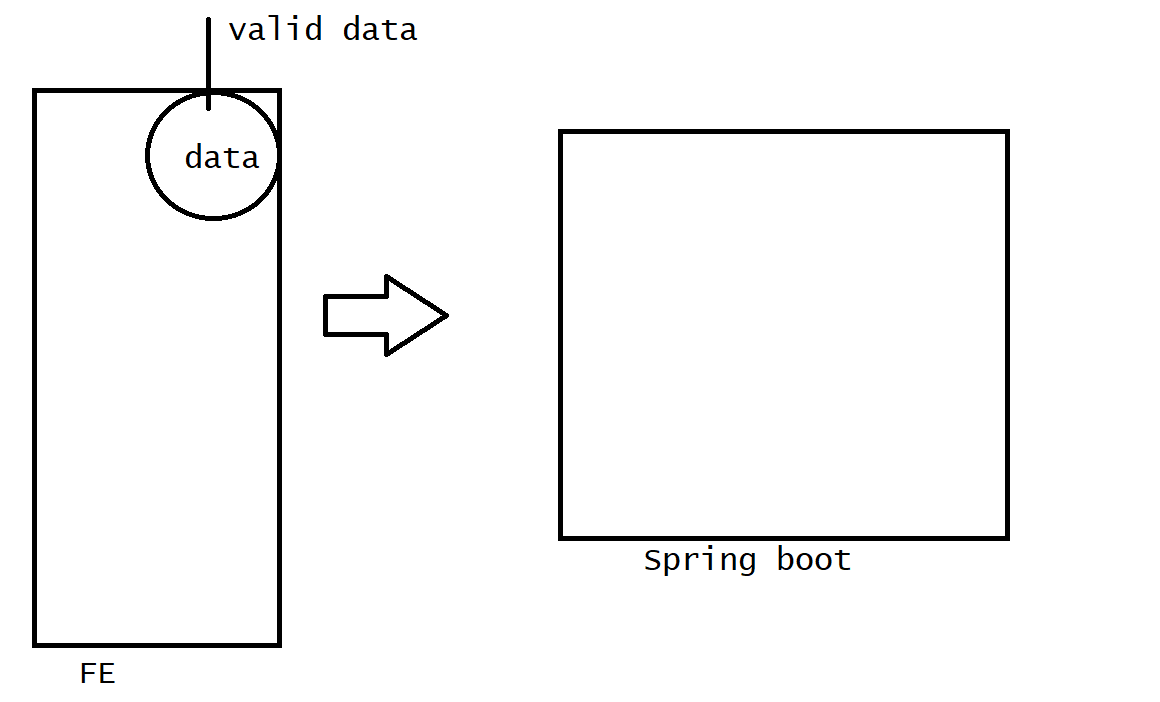
        <input type="button" value="login" onclick="checkValidationOfForm()">

    </form>

</body>

</html>

==update the registration page now with innerText -10 min



1. Create the web page
2. Go to the function and validate the data
3. Use the XMLHttpRequest to push the data onto the spring boot application

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript" src="valid.js"></script>

</head>

<body>

<form name="f1">

Email<input type="text" name="email"><br>

<div id="place1" style="color:red"></div>

phono<input type="text" name="phono"><br>

<div id="place2" style="color:red"></div>

<input type="button" value="register" onclick="checkform()">

</form>

</body>

</html>

//valid.js

function checkform(){

var email=document.f1.email.value;

var phono=document.f1.phono.value;

var statusofemail=""

var statusofphono=""

var statusemail=true;

var statusphono=true;

if(email==""){

statusofemail="email cannot be empty pls! fill"

statusemail=false

document.getElementById("place1").innerText=statusofemail

}

if(phono==""){

statusofphono="phono cannot be empty"

statusphono=false

document.getElementById("place2").innerText=statusofphono

}

if(statusemail&&statusphono){

sendRequest(email,phono);

}

}

function sendRequest(email,phono){

/\*

1.open the request

2.set the header

3.make it to ready state

4.validate the ready state status

5. send the request

\*/

console.log("enetered into sendrequest")

var xhr=new XMLHttpRequest();

//request type, reqname,enabler of the config

xhr.open("POST","/demo",true)

xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded")

xhr.onreadystatechange=function(){

if(xhr.readyState==XMLHttpRequest.DONE&&xhr.status==200){

}

};

xhr.send("email="+email+"&phono="+phono)

}

package com.example.demo;

import java.util.logging.Logger;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

@Controller

public class DemoController {

Logger log=Logger.getAnonymousLogger();

@RequestMapping("/demo")

public void responseIn(HttpServletRequest request,HttpServletResponse response) {

log.info("in spring boot"+request.getParameter("email"));

log.info("in spring boot"+request.getParameter("phono"));

}

}

//create an insert of data into the dB with validation and display it –Home assignment

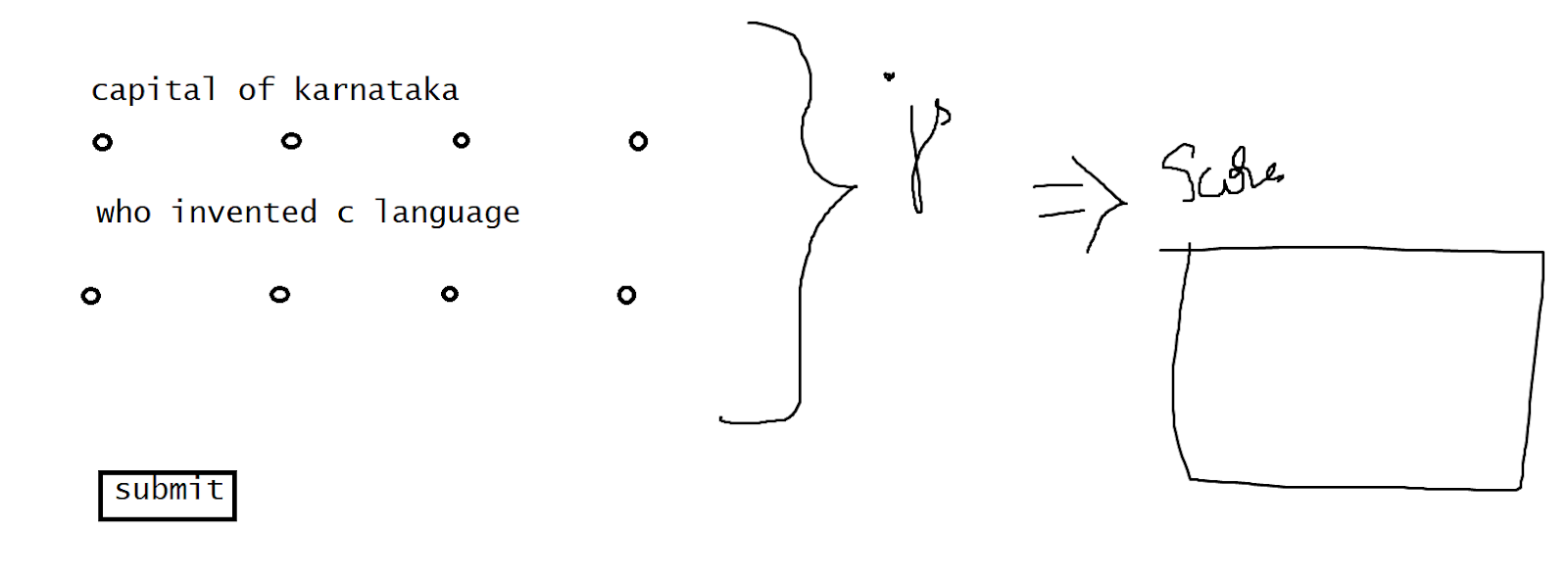
1. In js create a data request to the Spring boot app
2. Bring the data into the spring boot method for insert-void
3. In js after the process request to the jsp file as a success file
4. In the JSP file, try to request the spring boot application.

// Quiz Portal —-------------10 min

1. Questions
2. Answers validate

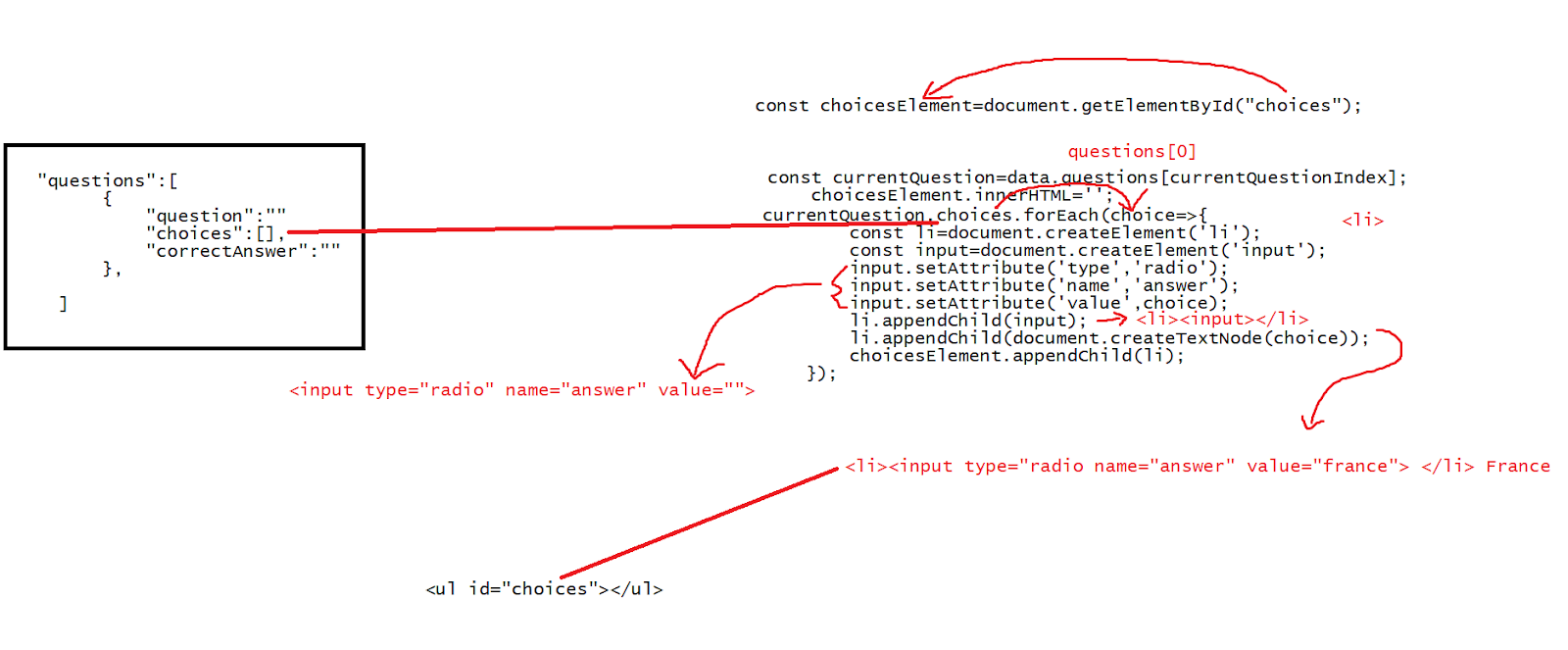
5 questions - hardcoded

???



–Use array–

//JSON file questions and choices into a page and validate it .



<!DOCTYPE html>

<html lang="en">

<head>

</head>

<body>

    <h1>Quiz</h1>

    <div id="quiz-container">

        <h2 id="question"></h2>

        <ul id="choices"></ul>

        <button id="submit">submit</button>

    </div>

  <div id="result-container">

    <h2 id="result"></h2>

  </div>

  <script src="external.js"></script>

</body>

</html>

fetch('questions.json')

.then(response=>response.json())

.then(data=>{

console.log(data)

const quizContainer=document.getElementById("quiz-container");

const questionElement=document.getElementById("question");

const choicesElement=document.getElementById("choices");

const submitButton= document.getElementById("submit");

const resultContainer=document.getElementById("result-container");

const resultElement=document.getElementById("result");

let currentQuestionIndex=0;

let score=0;

function loadQuestion(){

    const currentQuestion=data.questions[currentQuestionIndex];

    questionElement.textContent=currentQuestion.question;

    choicesElement.innerHTML='';

    currentQuestion.choices.forEach(choice=>{

        const li=document.createElement('li');

        const input=document.createElement('input');

        input.setAttribute('type','radio');

        input.setAttribute('name','answer');

        input.setAttribute('value',choice);

        li.appendChild(input);

        li.appendChild(document.createTextNode(choice));

        choicesElement.appendChild(li);

    });

}

function checkAnswer(){

    const selectedAnswer=document.querySelector('input[name="answer"]:checked');

    if(selectedAnswer){

        const userAnswer=selectedAnswer.value;

        const currentQuestion=data.questions[currentQuestionIndex];

        if(userAnswer===currentQuestion.correctAnswer){

            score++;

        }

        currentQuestionIndex++;

        if(currentQuestionIndex < data.questions.length){

            loadQuestion();

        }

        else{

            showResult();

        }

    }

}

function showResult(){

    quizContainer.style.display='none';

    resultContainer.style.display='block';

    resultElement.textContent=`you scored ${score} out of ${data.questions.length} questions`;

}

submitButton.addEventListener('click',checkAnswer);

loadQuestion();

})

//questions.json

{

    "questions":[

        {

            "question":"what is the capital of france",

            "choices":["London","Paris","Rome","Berlin"],

            "correctAnswer":"Paris"

        },

        {

            "question":"Brass gets discoloured in air because of the presence of which of the following gases in air",

            "choices":["Oxygen","Hydrogen sulphide","Carbon dioxide","Nitrogen"],

            "correctAnswer":"Hydrogen sulphide"

        }

    ]

}

//OO based

Prototyping

Case 1:

//if we have to create 100 objects then we need to load the fn/constructor for 100 times in the memory

–waste the memory by loading the content for many times

<!DOCTYPE html>

<html lang="en">

<head>

   <script>

    function Student(name){

        this.name=name;

        this.getName=function(){return this.name;}

//get name is present inside the function of Student now whenever we are //calling student - the getname will be called and the object of s1 and s2 //will be getting the name value over here

    }

    var s1=new Student("ravi");

    var s2=new Student("ramesh");

    document.write(s1.getName()+"<br>")

    document.write(s2.getName()+"<br>")

   </script>

</head>

<body>

</body>

</html>

Case 2

We wanted the getname to work object specific

<!DOCTYPE html>

<html lang="en">

<head>

   <script>

    function Student(name){

        this.name=name;

    }

    var s1=new Student("ravi");

 s1.getName=function(){return this.name;}

    var s2=new Student("ramesh");

    document.write(s1.getName()+"<br>")

    document.write(s2.getName()+"<br>")

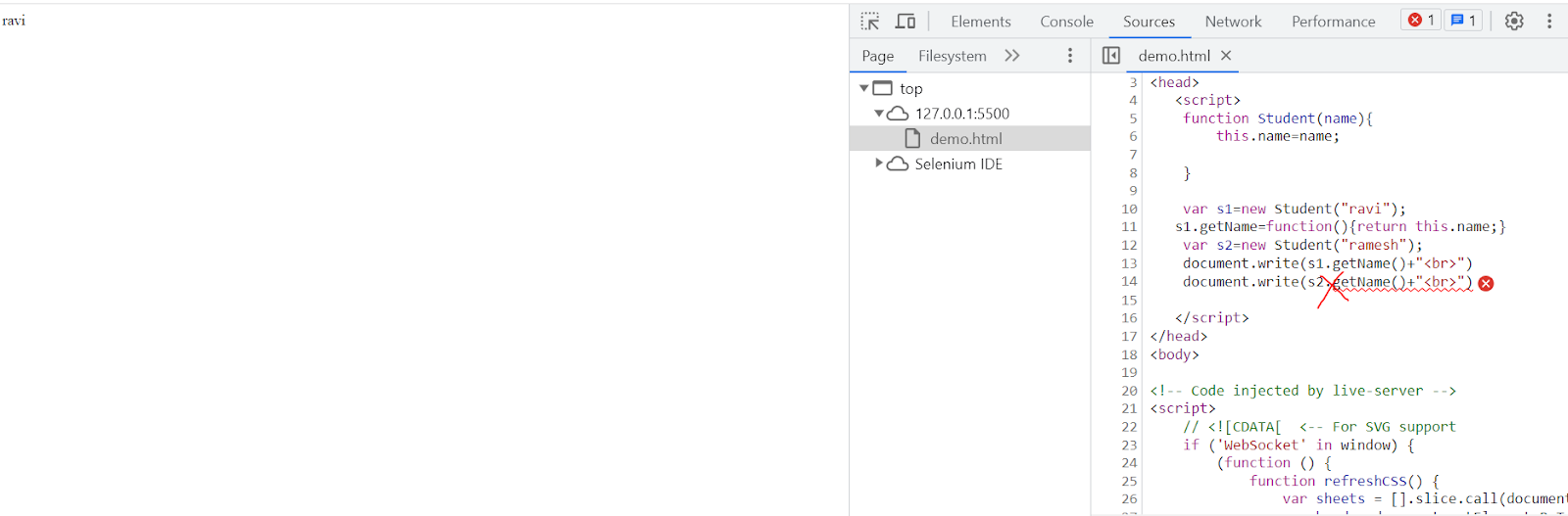
   </script>

</head>

<body>

</body>

</html>



Case 3 :

Student level

<!DOCTYPE html>

<html lang="en">

<head>

   <script>

    function Student(name){

        this.name=name;

    }

    var s1=new Student("ravi");

   Student.getName=function(){return this.name;}

    var s2=new Student("ramesh");

    document.write(s1.getName()+"<br>")

    document.write(s2.getName()+"<br>")

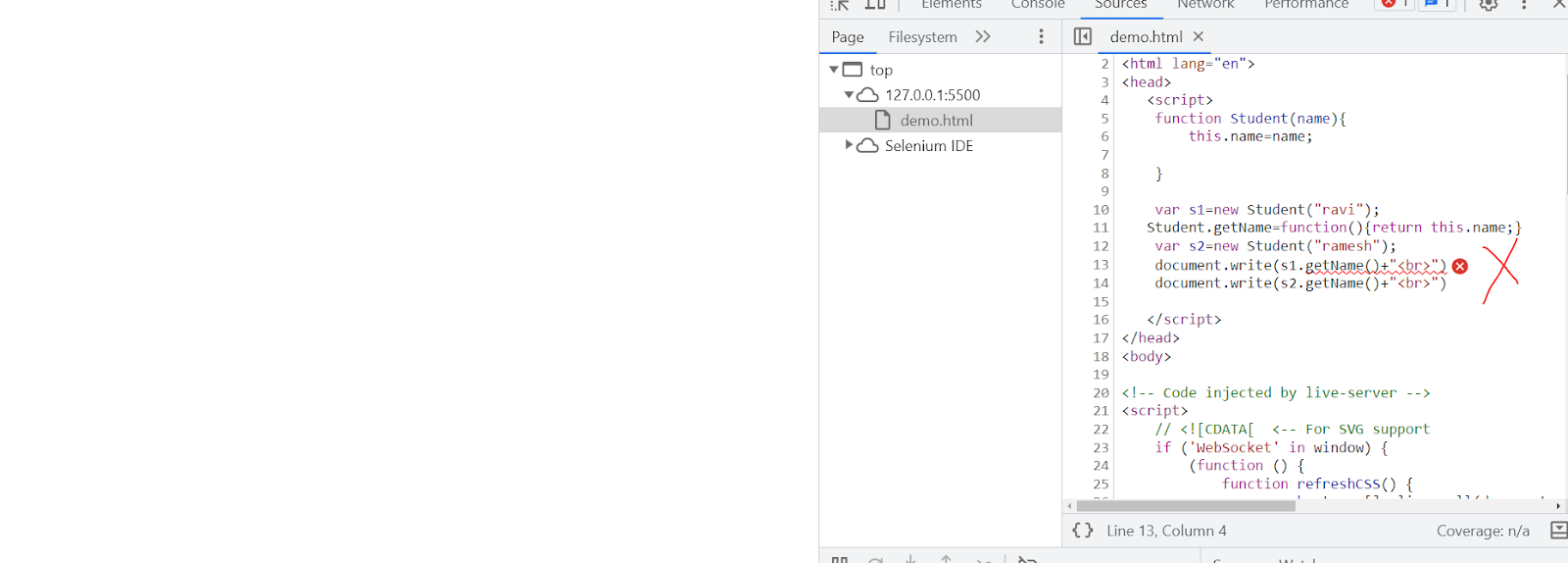
   </script>

</head>

<body>

</body>

</html>



<!DOCTYPE html>

<html lang="en">

<head>

   <script>

    function Student(name){

        this.name=name;

    }

    var s1=new Student("ravi");

   Student.prototype.getName=function(){return this.name;}

    var s2=new Student("ramesh");

    document.write(s1.getName()+"<br>")

    document.write(s2.getName()+"<br>")

   </script>

</head>

<body>

</body>

</html>

//example 2

<!DOCTYPE html>

<html lang="en">

<head>

   <script>

    function Student(name,age){

        this.name=name;

        this.age=age;

    }

    Student.prototype.teacher="karthik"

    var s1=new Student("ravi",23);

    var s2=new Student("ramesh",24);

    document.write(s1.teacher+"<br>");

    document.write(s2.teacher);

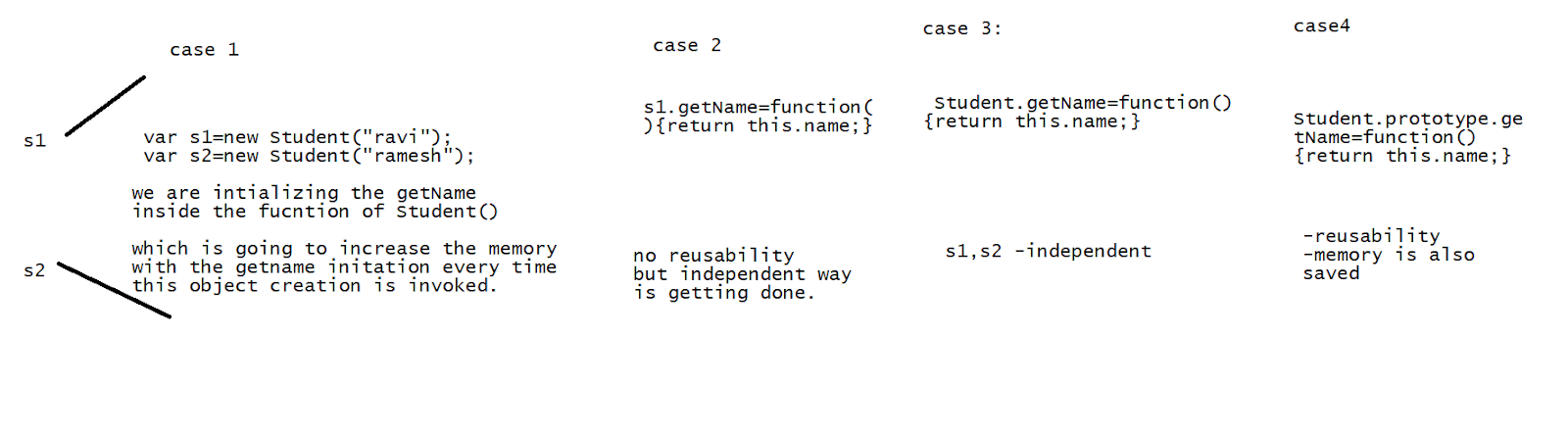
   </script>

</head>

<body>

</body>

</html>

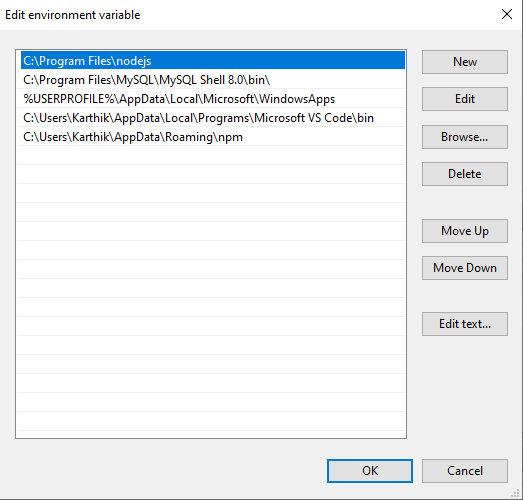


Angular:

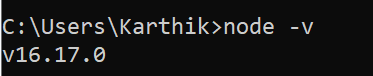
Installation:

<https://nodejs.org/dist/v20.4.0/node-v20.4.0-x64.msi>

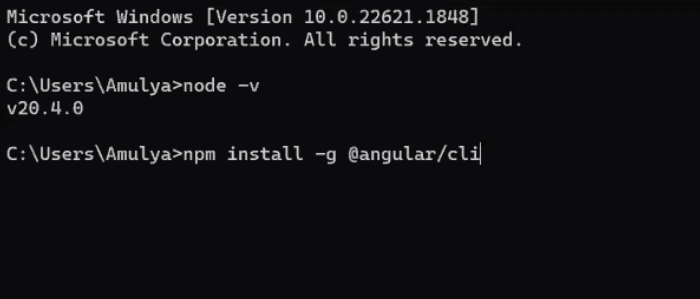
1. Install the nodejs
2. Open the environmental variables



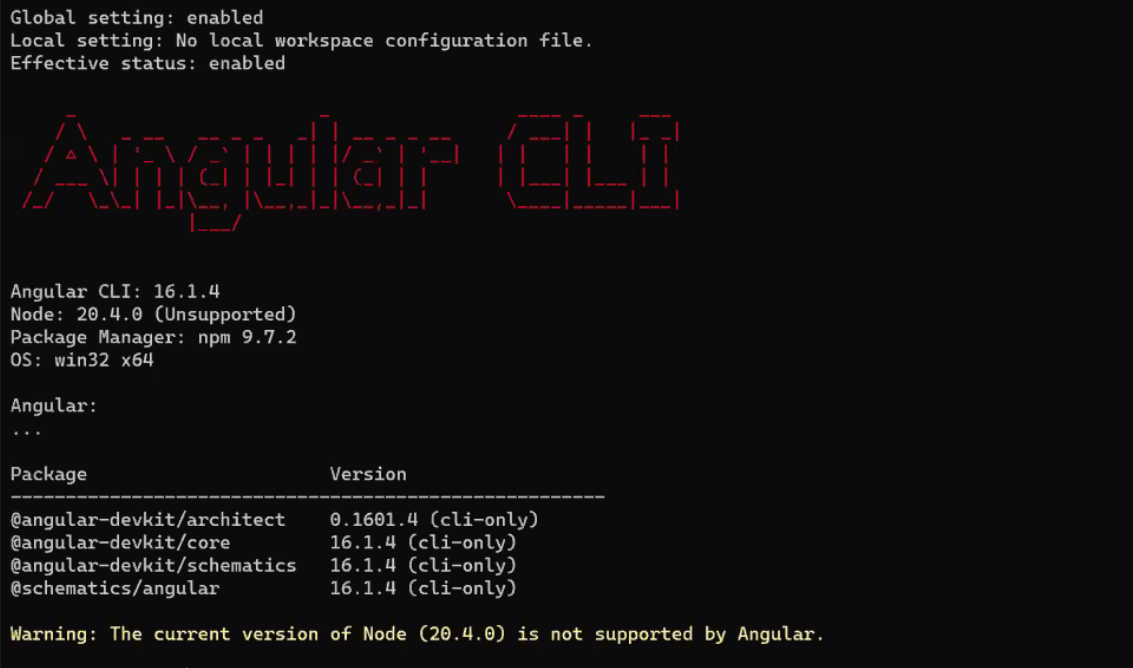
1. Open command prompt

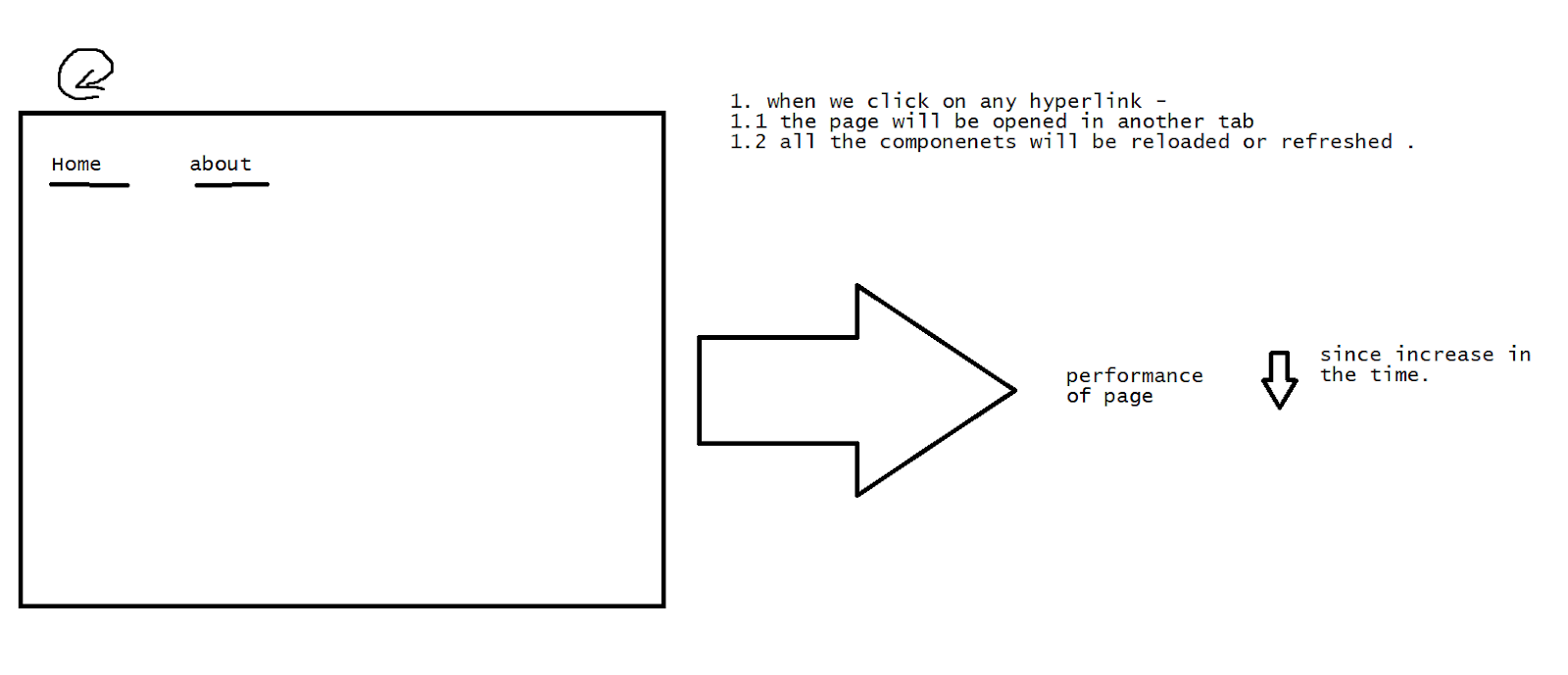


1. Open a command prompt and type



1. Run ng v  to verify whether the angular got installed or not





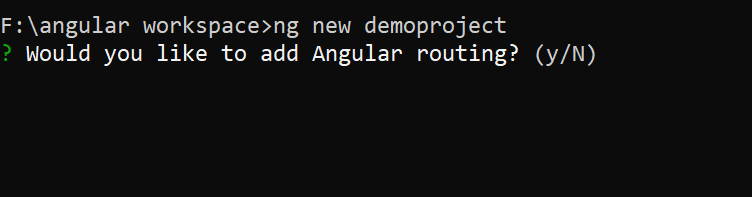
>To represent a single output page. [Any hyperlinks the outputs are going to be placed onto a single frame only in angular].

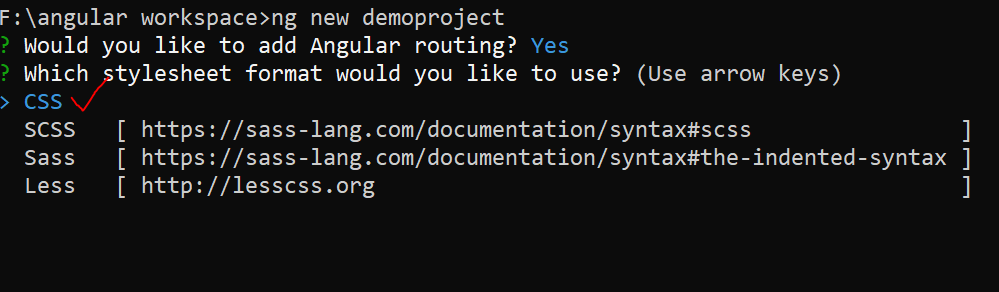
Summary

1. No refresh
2. Single frame output

How to create an angular project

1. Create a workspace
2. Go into the workspace
3. Open the command prompt and give the command

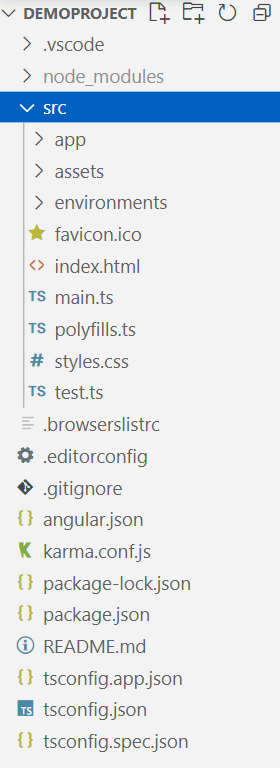






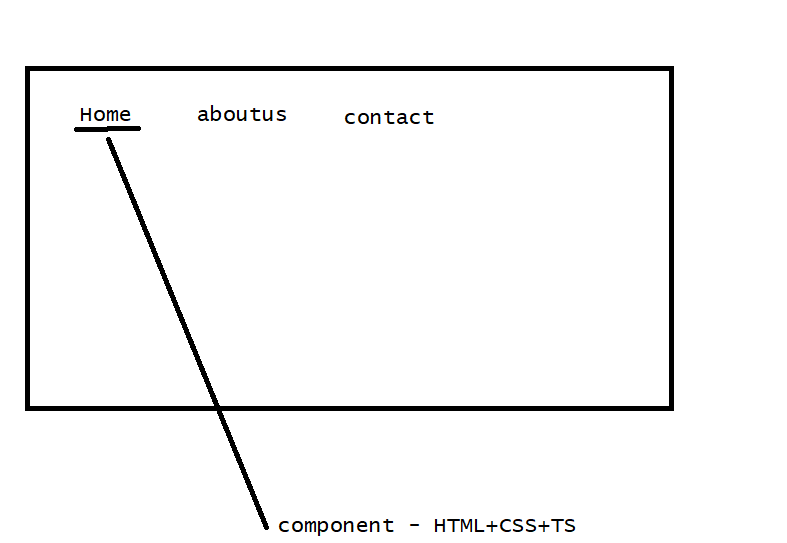
Open the vs

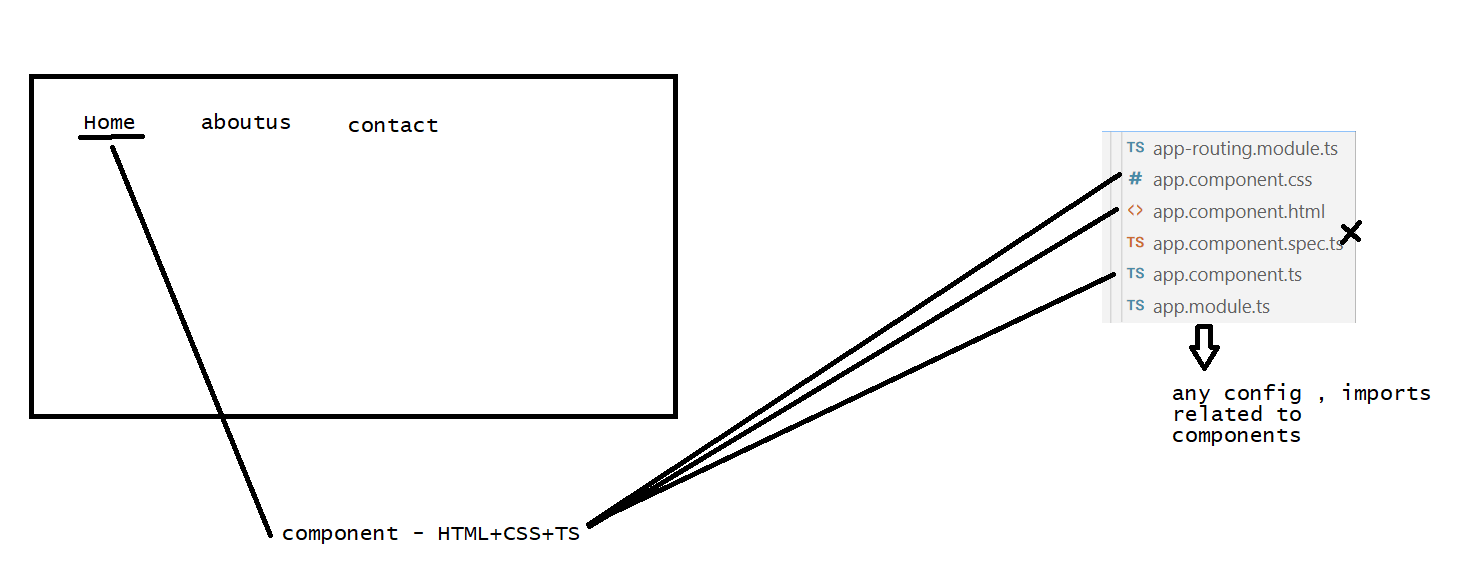
Go to vs ->open folder ->open the created angular folder in the workspace

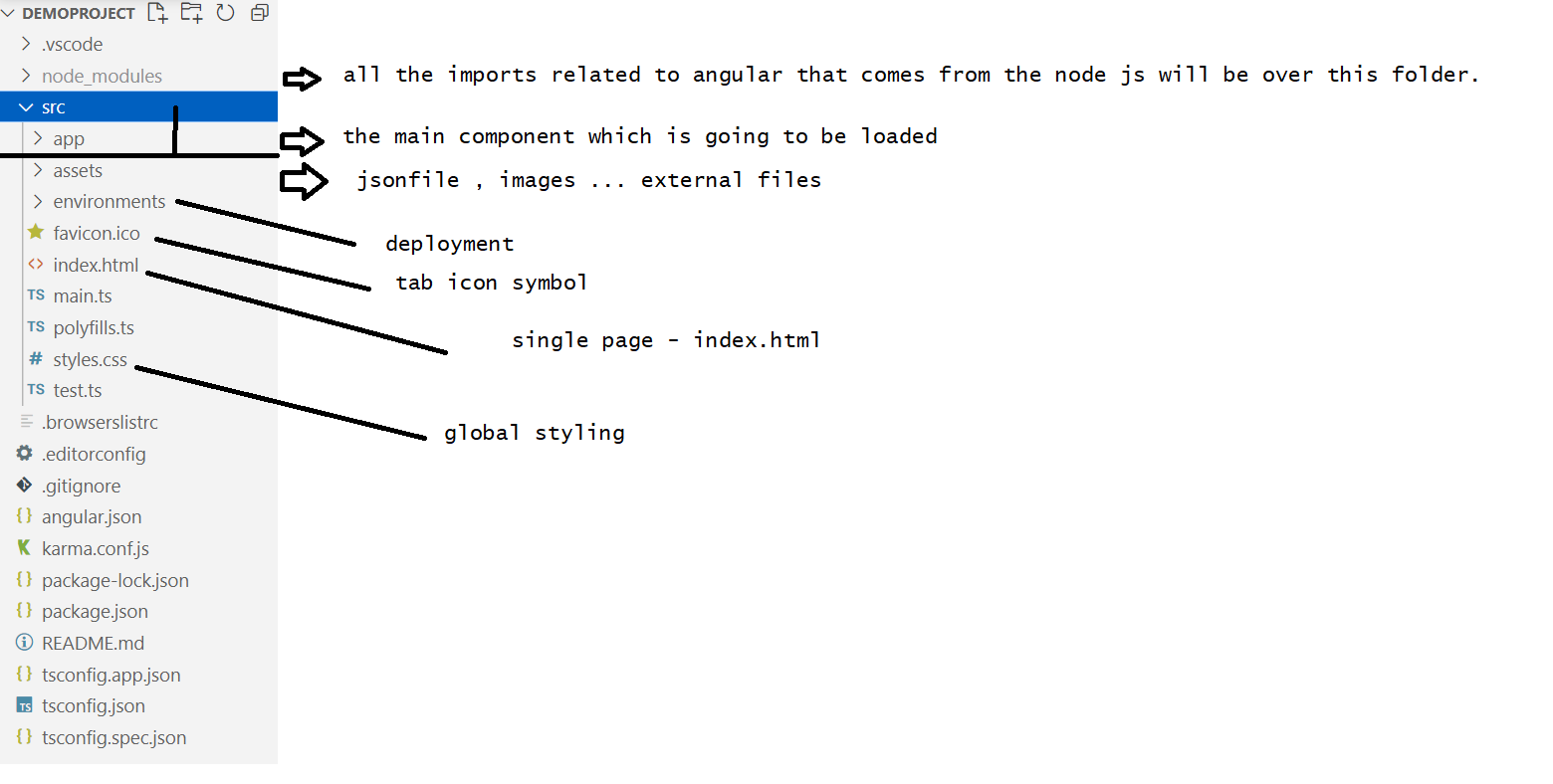


On angular, we start calling a page a “component”

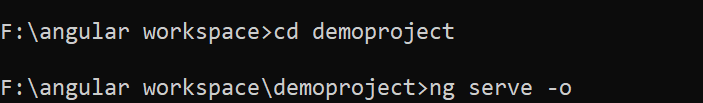
Component -> HTML + CSS + TS(typescript)

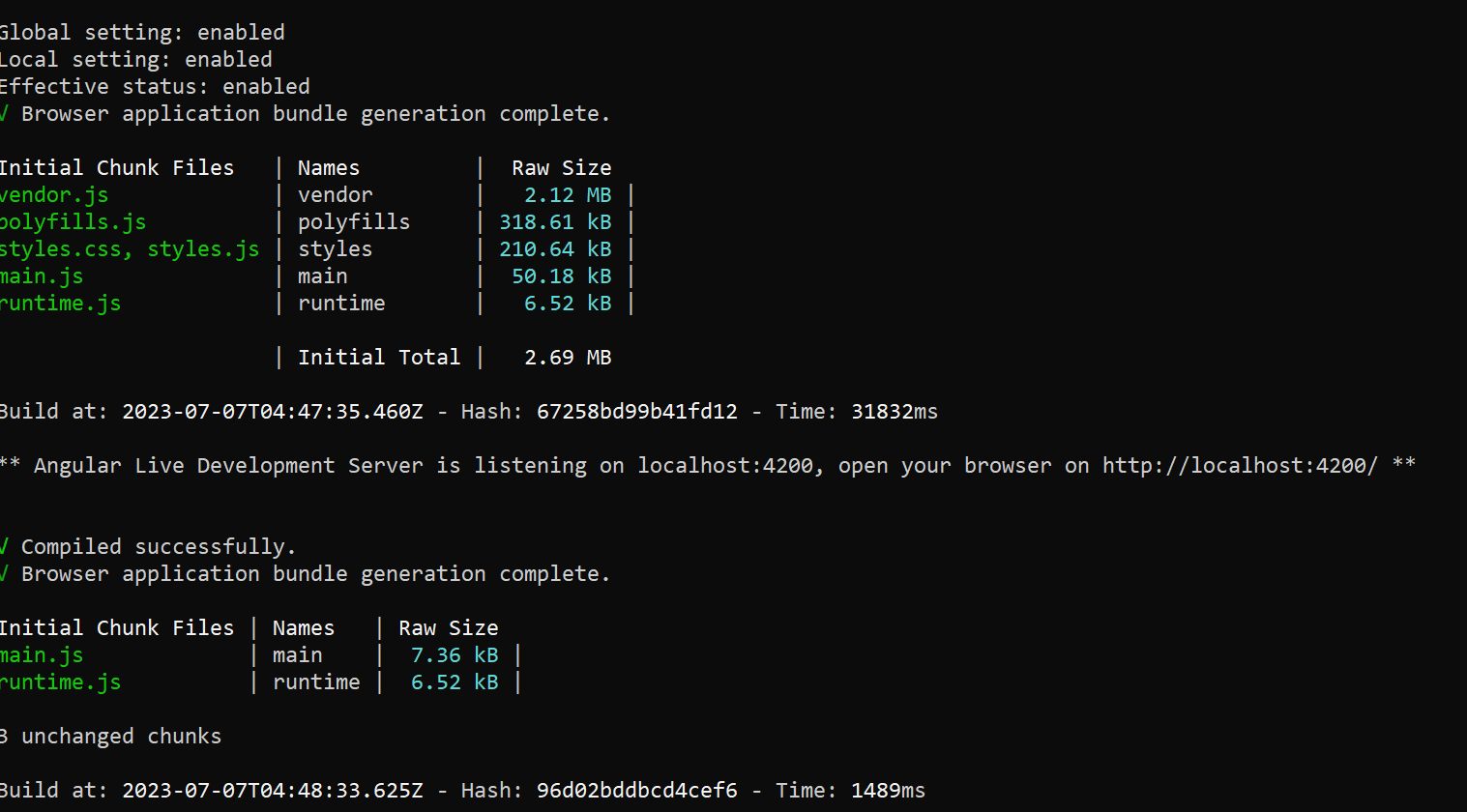


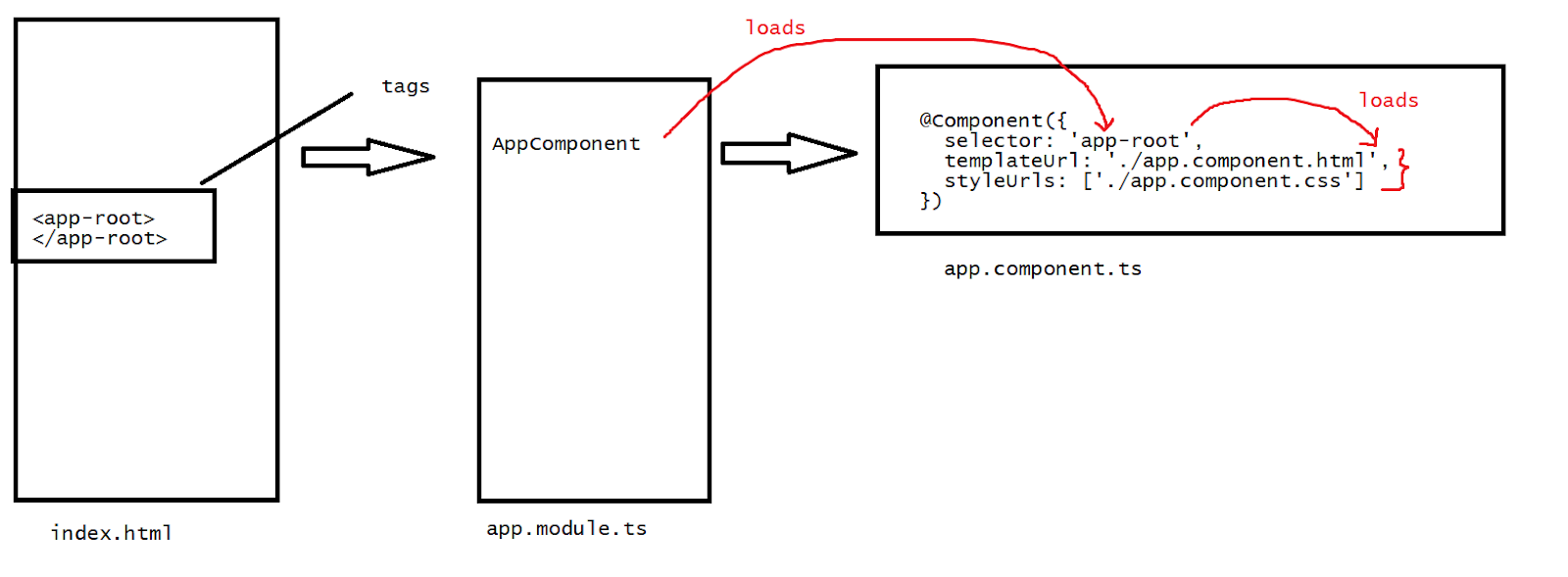




How to run an angular server?

Portno :4200



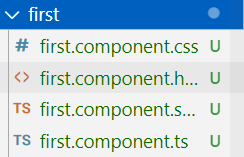


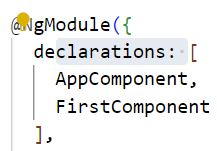
Task:

Create a login page as an angular -bootstrap →app component. Component.html, add the CSS in app.component.css  —5 min

How to load other components onto the main html component -  app.component.html??







Sol :  via the selectors

<h1> hi learners!!!fr</h1>

<app-first></app-first>

Task:

Create 2 components about, contact load the description onto the main component.

Typescript - Javascript framework

* Dynamically typed language

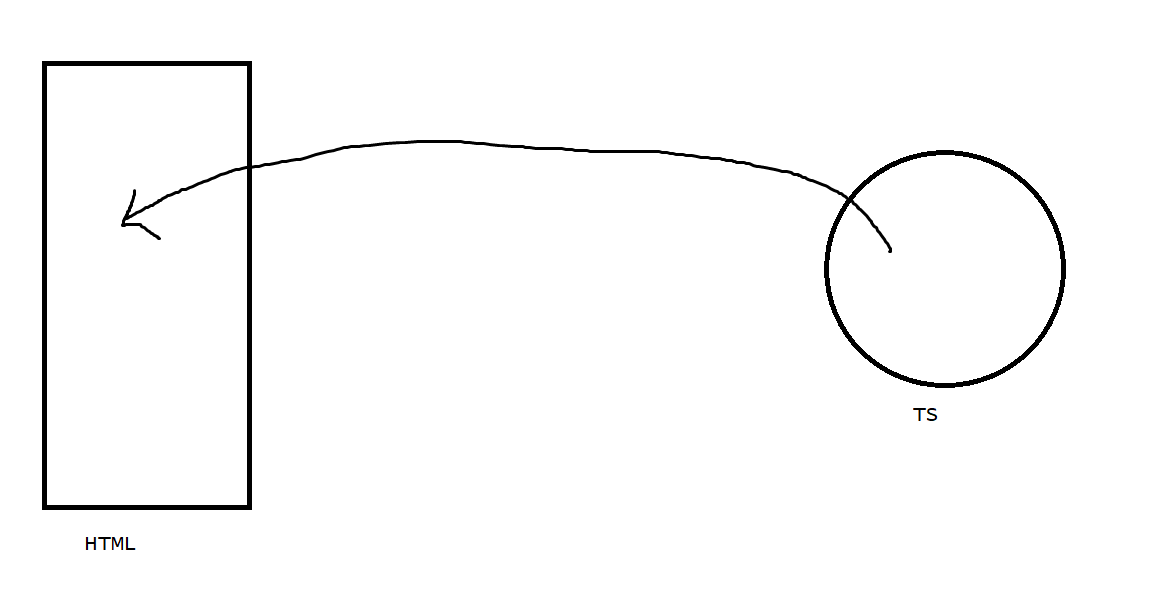
>databinding



>one way binding

>two way binding

-one-way binding



>String interpolation

>Property binding

>Class binding

>Style binding

>Event binding

String interpolation : {{}}

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  heading="Session on Angular"

}

Page :

<h1>{{heading}}</h1>

Task:

Take any data item of int , float … try to load on the string interpolation of the page

Property Binding:

The below is the static way of defining the attributes

sid <input type="text" hidden>

<button disabled>submit</button>

Dynamic way - from the ts we operate the value of the attribute

Syntax: [property]=value

sid <input type="text" [hidden]="ishidden">

<button [disabled]="isdisabled">submit</button>

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

 ishidden=false

 isdisabled=false

}

Class binding :

[class]=” action name”

html

<h1 [class]="isactive?'active':'inactive'">Hi learners welcome!!!</h1>

 Css

.active{background-color: aquamarine;color: white;}

.inactive{background-color: bisque;color: rgb(20, 178, 178);}

TS

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  isactive=true

}

//style binding

[style]=”styleobjname”

Limitation: any thing of the style with ‘-’ will not be taken

html

<h1 [style]="active">Hi learners welcome!!!</h1>

Ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  active:object=

  {

  color:'white',

  background:'grey'

  }

}

//event binding

(eventname)=”functionname”

<button (click)="increment()">increment</button>

<button (click)="decrement()">decrement</button>

<h1>{{count}}</h1>

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

count=0;

increment(){this.count+=1;}

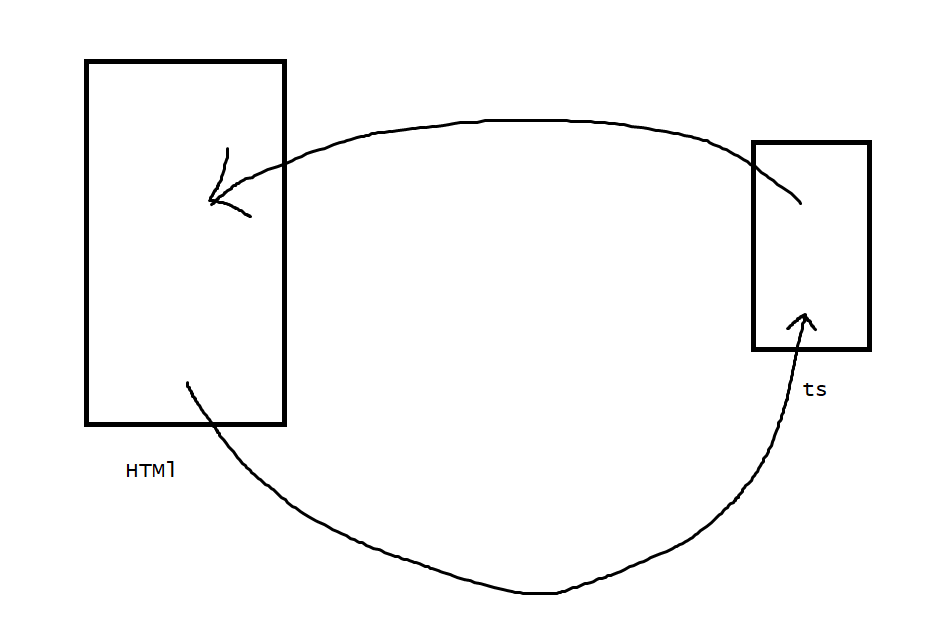
decrement(){this.count-=1;}

}

//task -5 min

Create a dynamic style on a click of a button style should be enabled to the text

//two way binding



1. In two way binding we need to include the formsmodule

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

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

import { AppRoutingModule } from './app-routing.module';

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

import { FirstComponent } from './first/first.component';

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

@NgModule({

  declarations: [

    AppComponent,

    FirstComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule

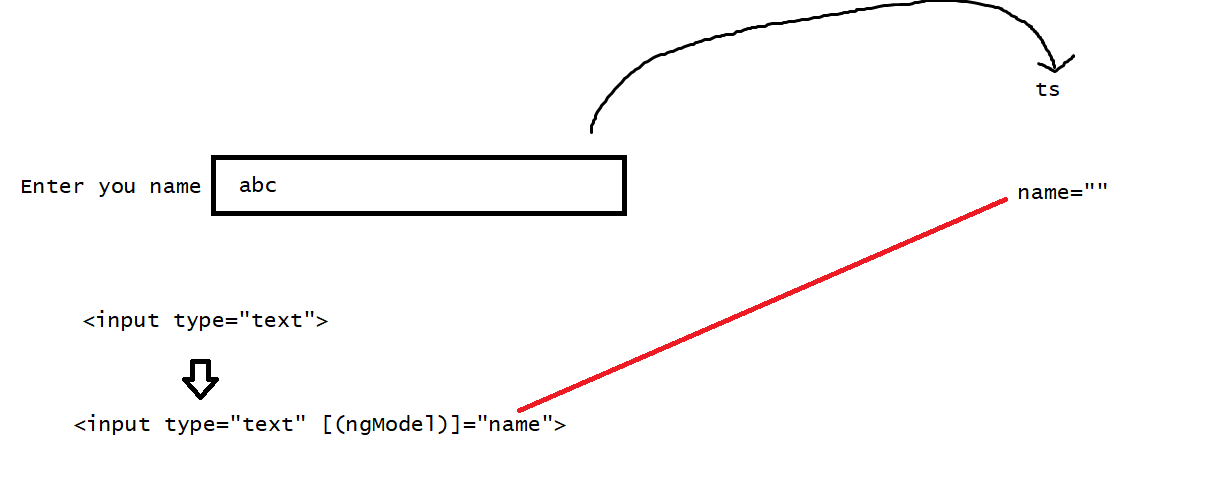
  ],

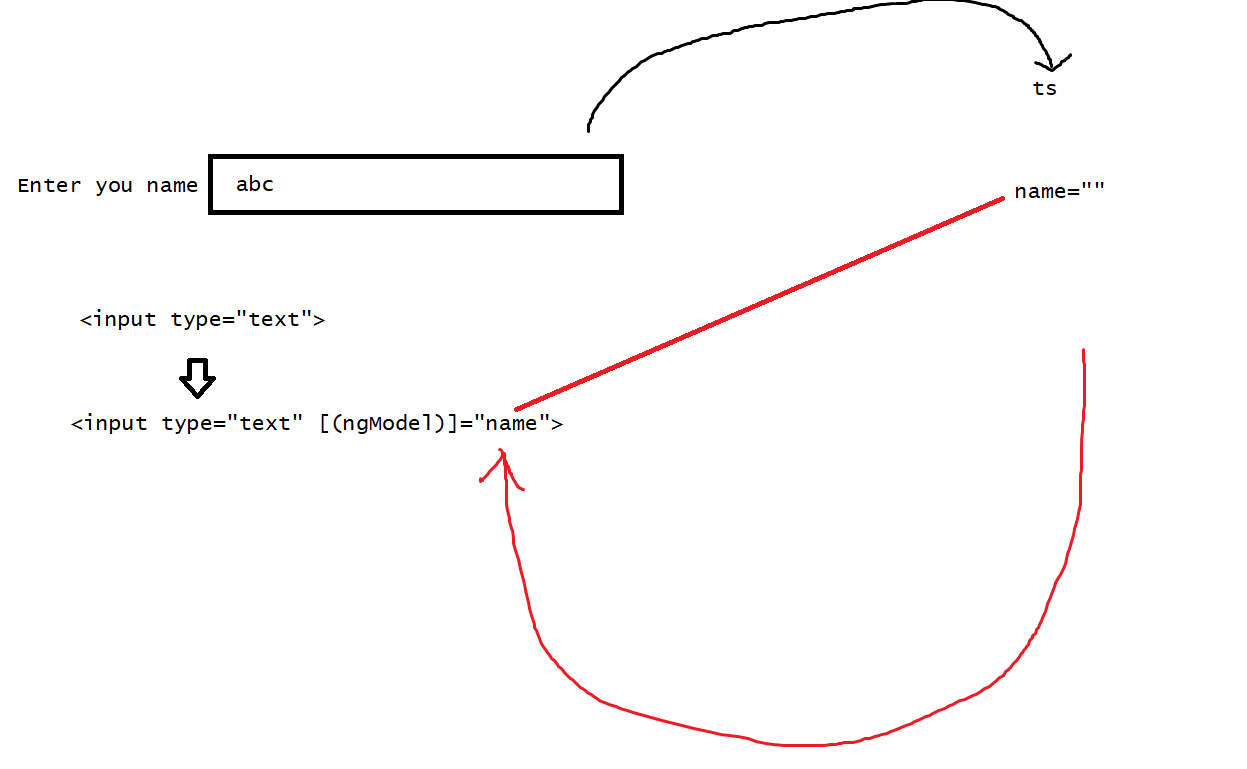
  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }





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

Your name is {{name}}

Ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

name=""

}

//task

take another field and check whether we are getting data or not—the radio button

<label>What is the capital of karnataka</label><br>

<input type="radio" name="group" value="Banglore" (input)="update($event)">Banglore<br>

<input type="radio" name="group"  value="Bihar" (input)="update($event)">Bihar<br>

<input type="radio" name="group"  value="Hyderabad" (input)="update($event)">Hyderabad<br>

<input type="radio" name="group"  value="mumbai" (input)="update($event)">mumbai<br>

The score is {{score}}

Ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  score=0;

  ans="";

 update(e:any){

  this.ans=e.target.value;

  if(this.ans===("Banglore")){

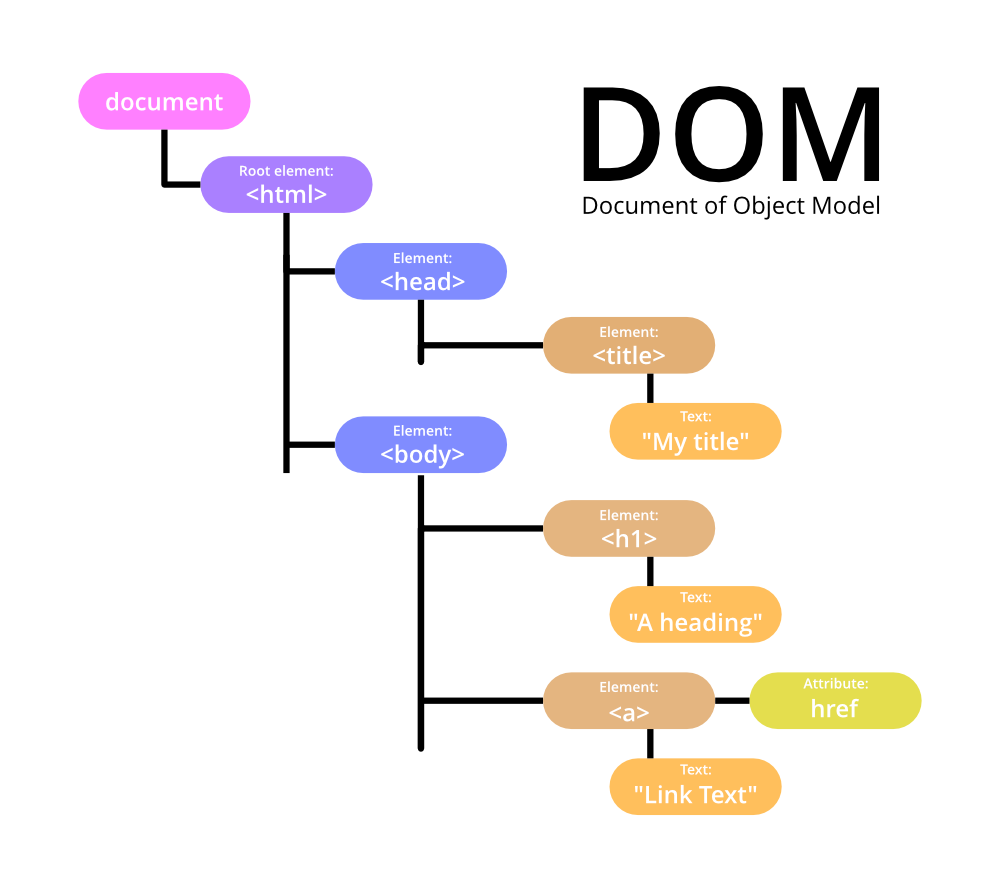
    this.score++;

  }

 }

}

DOM : Document of object model



Directives

Custom HTML, attributes which tell the angular to change the structure , style or behaviour of the DOM.

>Structural directives

Change the DOM layout by adding / removing the DOM elements .

ngIf,ngFor,ngSwitch

ngIf

>ngIf=”value”

>ngIf with else

>ngIf with then else <stmt>

>ngIf=”value”

<h1 \*ngIf="show()">Student works!!</h1>

Ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

show(){

  return false;

}

}

>ngIf with else

<h1 \*ngIf="show else fail">{{textdata}}</h1>

<ng-template #fail>Id<input type="text"></ng-template>

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

show=false;

textdata="Hi learners"

}

Task:

On a condition I need to get an admin login form and user login form

>ngIf with then else <stmt>

<h1 \*ngIf="show;then success;else fail"></h1>

<ng-template #success><h1 >{{textdata}}</h1></ng-template>

<ng-template #fail>Id<input type="text"></ng-template>

Task

Rework the previous task with ngIf with then else

ngSwitch,ngSwitchCase,ngSwitchDefault

<div [ngSwitch]="opr">

<div \*ngSwitchCase="'+'">{{num1+num2}}</div>

<div \*ngSwitchCase="'-'">{{num1-num2}}</div>

<div \*ngSwitchCase="'\*'">{{num1\*num2}}</div>

<div \*ngSwitchCase="'/'">{{num1/num2}}</div>

<div \*ngSwitchDefault>wrong option given</div>

</div>

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

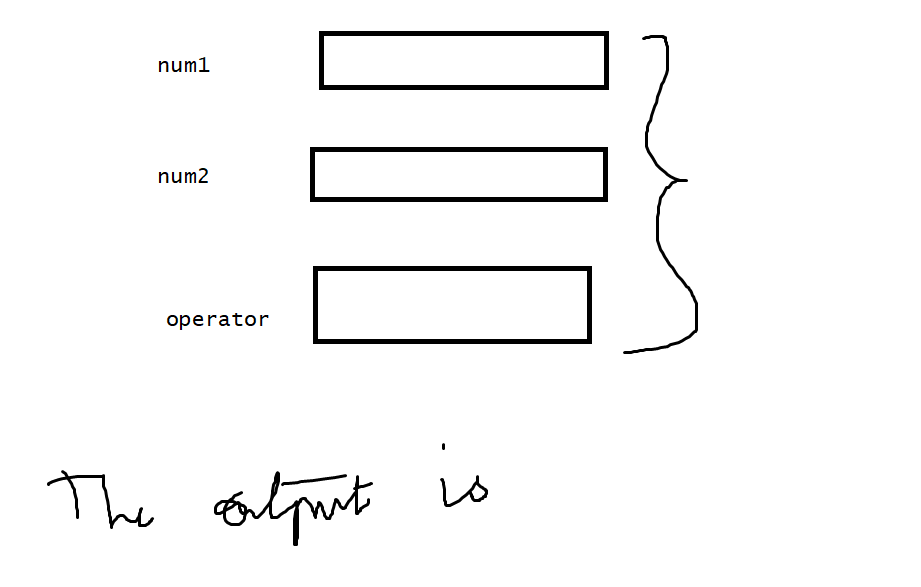
export class AppComponent {

num1=2

num2=3

opr="/"

}



\*ngFor

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

mobilebrand=[{"id":1,"name":"karthik","mobile":"samsung"},{"id":2,"name":"ajay","mobile":"iphone"}]

}

<table border="1">

<tr \*ngFor="let m of mobilebrand">

<td>{{m.id}}</td>

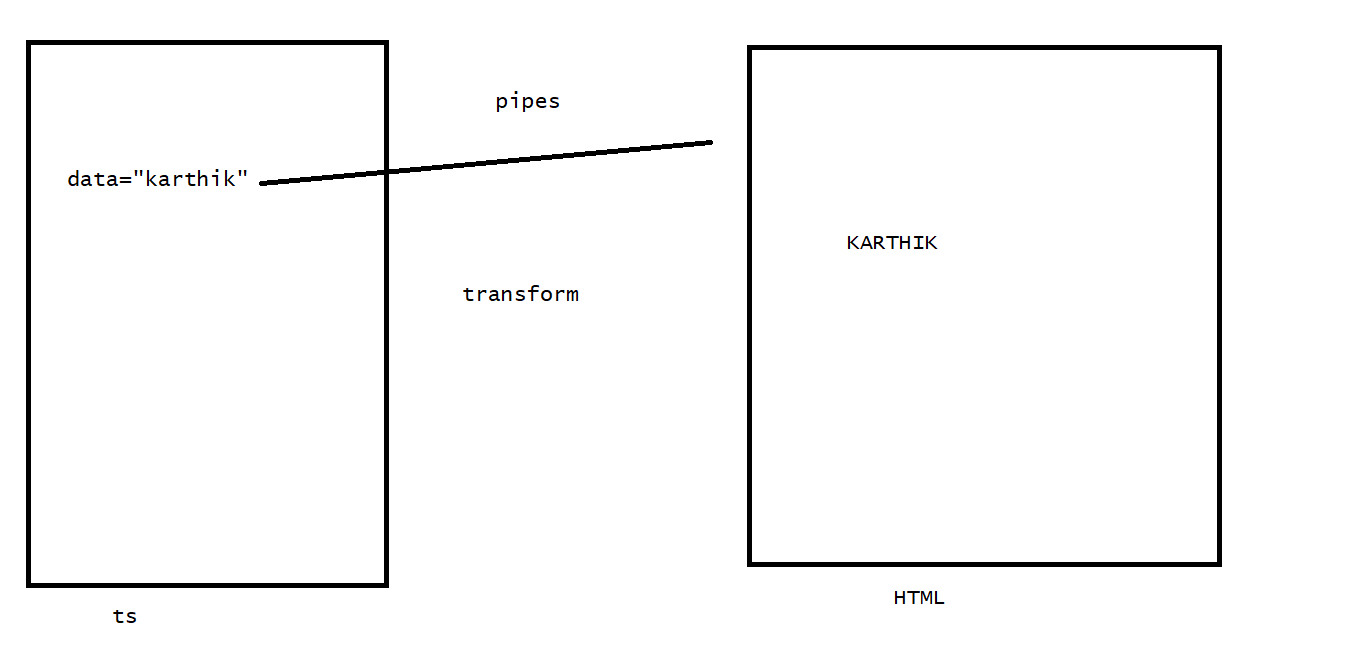
<td>{{m.name}}</td>

<td>{{m.mobile}}</td>

</tr>

</table>

//pipes



It is used to transform the data before we display it on HTML.

>predefined

>customized

>predefined

Uppercase

Lowercase

Currency

Json

Percent

Slice

Decimal………….

syntax:

{{<dataitem> | <pipename> :<attributes>}}

<h1>{{textdata | titlecase}}</h1>

<h1>{{salary | currency}}</h1>

<h1>{{salary | currency:'INR'}}</h1>

<h1>{{date |date:'dd/MM/yy'}}</h1>

<h1>{{person |json}}</h1>

Ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

textdata="karthik"

salary=10000

date=new Date()

person={"name":"karthik","age":30}

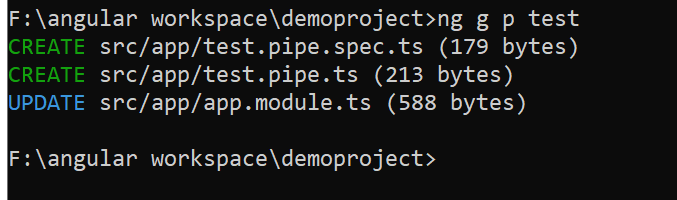
}

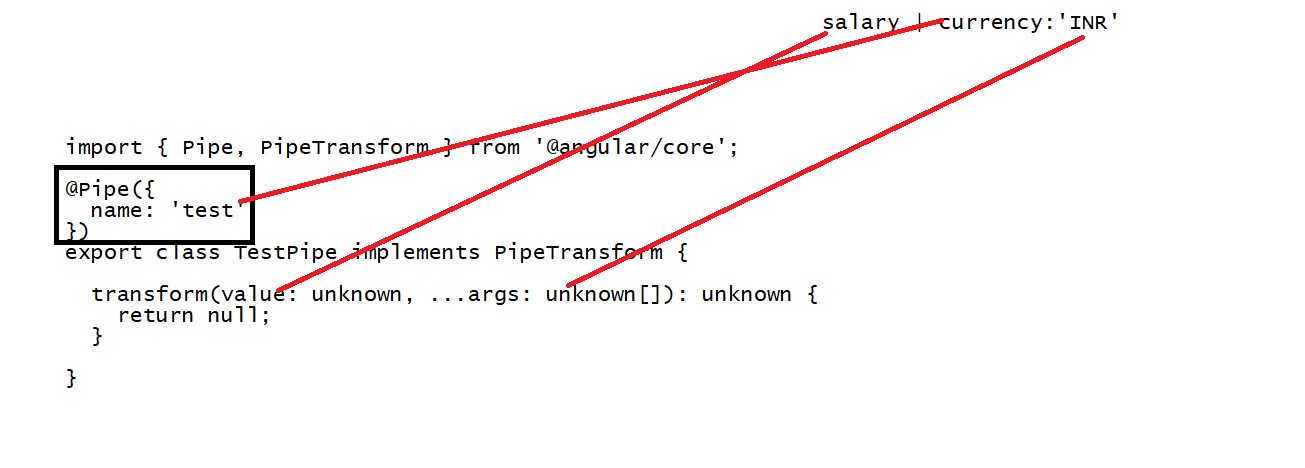
//customized pipes

1. Define a pipe
2. Define logic in a pipe for transformation .

1. Define a pipe

ng g p <pipename>





Pipe

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({

  name: 'test'

})

export class TestPipe implements PipeTransform {

              //wishdata | test:4:8

  transform(value: string,param1:number,param2:number ): string {

    return value.substring(param1,param2);

  }

}

//html

<h1>{{wishdata | test:4:12}}</h1>

 Ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

wishdata="Good Morning"

}

Ex:2

<h1>{{person | test:wishdata}}</h1>

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

wishdata="Good Morning"

person={"name":"ARUN","gender":"m"}

}

//pipe

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({

  name: 'test'

})

export class TestPipe implements PipeTransform {

              //person     wish

  transform(value: any,param1:string): string {

    if(value.gender=="m")

    {

      return "Hello Mr. "+value.name+" "+param1

    }

    else{

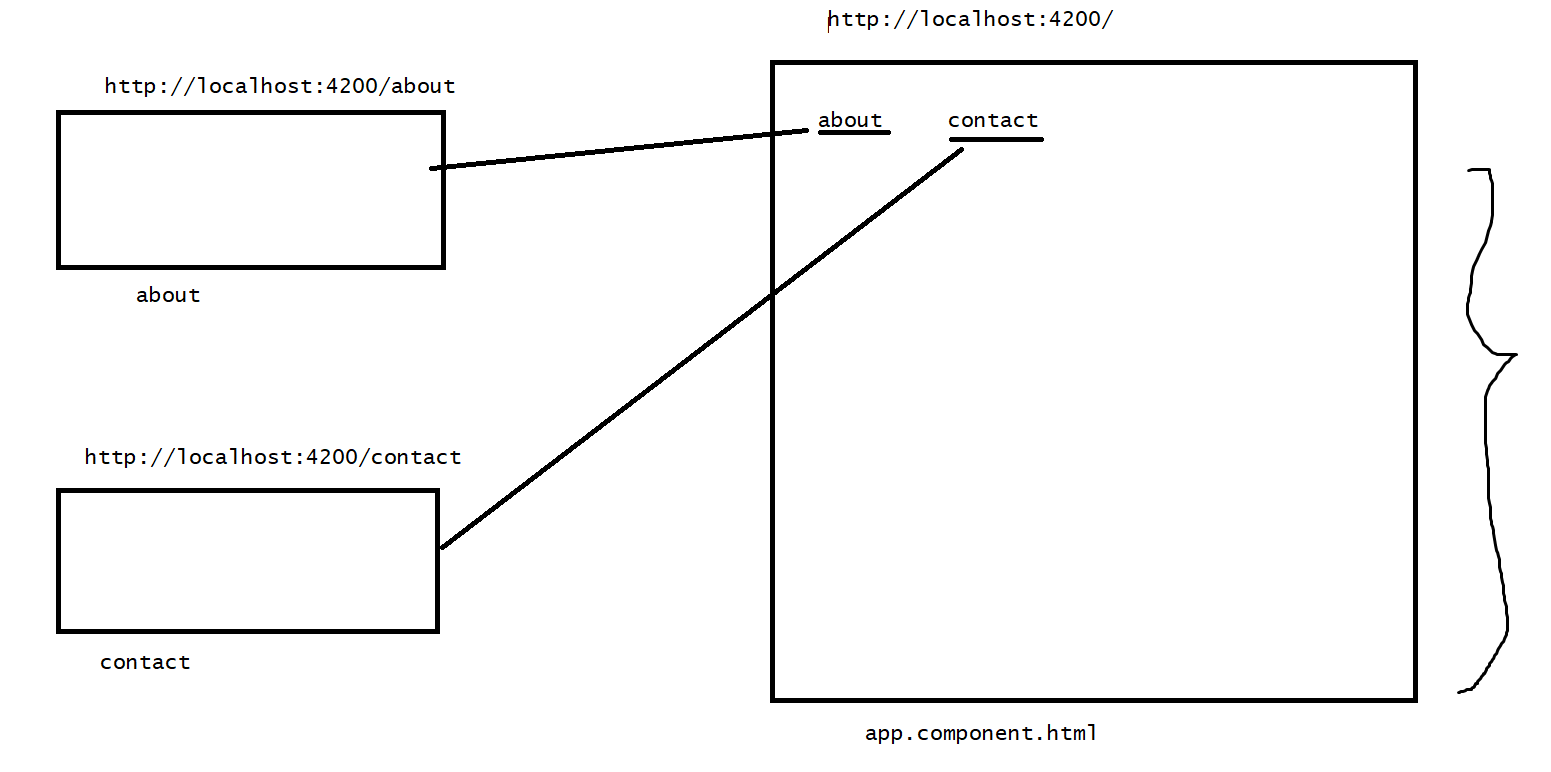
      return "Hello Miss. "+value.name+" "+param1

    }

}

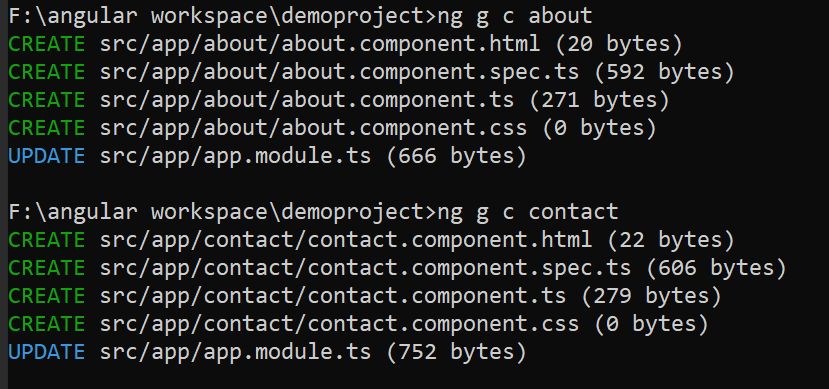
}

//Routing



Hyperlinks are used to navigate from one component to another

1. Config the route
2. Add the router outlet
3. Add the links in the template



//add header component

>Config the route

app.module.ts which is used as the configuration file for all the components

Define the routes

<http://localhost:4200/about>

[

{path:’about’, component:AboutComponent},

{path: contact,component:ContactComponent}

]



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

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

import { AppRoutingModule } from './app-routing.module';

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

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

import { TestPipe } from './test.pipe';

import { AboutComponent } from './about/about.component';

import { ContactComponent } from './contact/contact.component';

import { RouterModule, Routes } from '@angular/router';

import { HeaderComponent } from './header/header.component';

const routes:Routes=[

{path:"about",component:AboutComponent},

{path:"contact",component:ContactComponent}

]

@NgModule({

  declarations: [

    AppComponent,

    TestPipe,

    AboutComponent,

    ContactComponent,

    HeaderComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

    RouterModule.forRoot(routes)

  ],

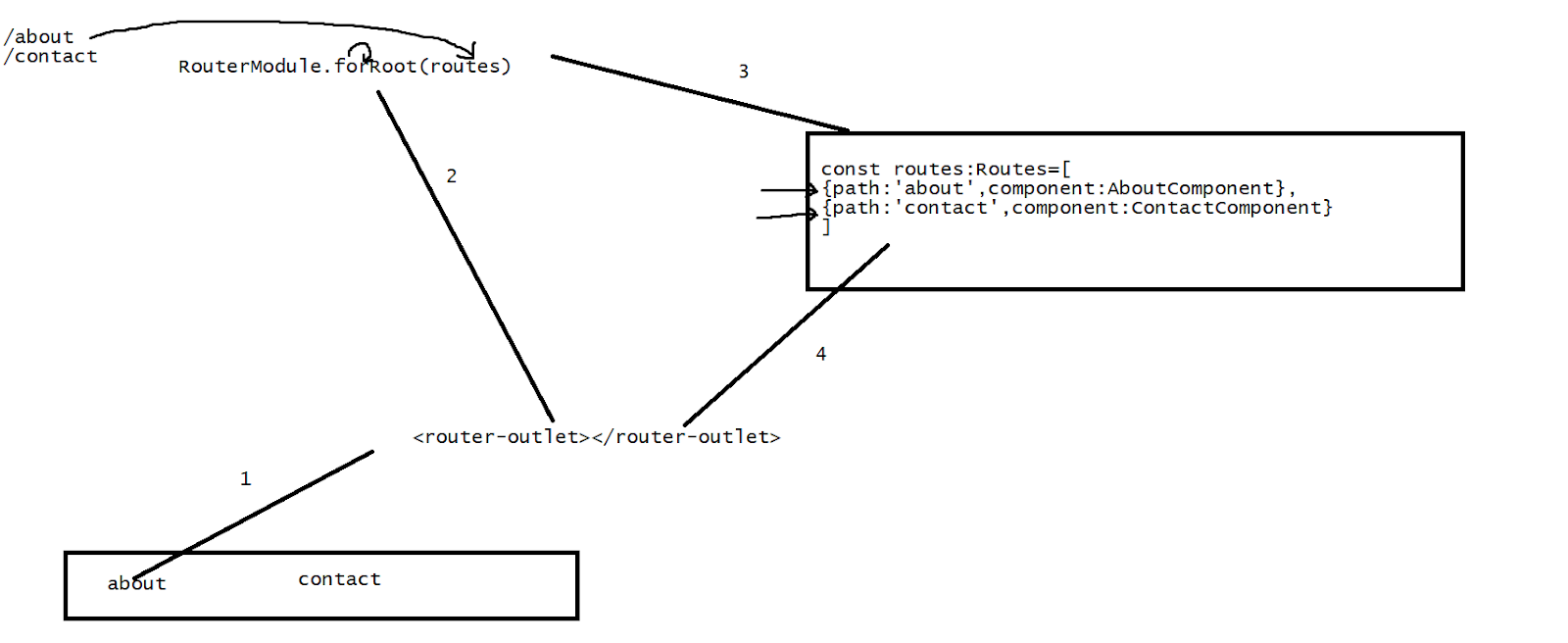
  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

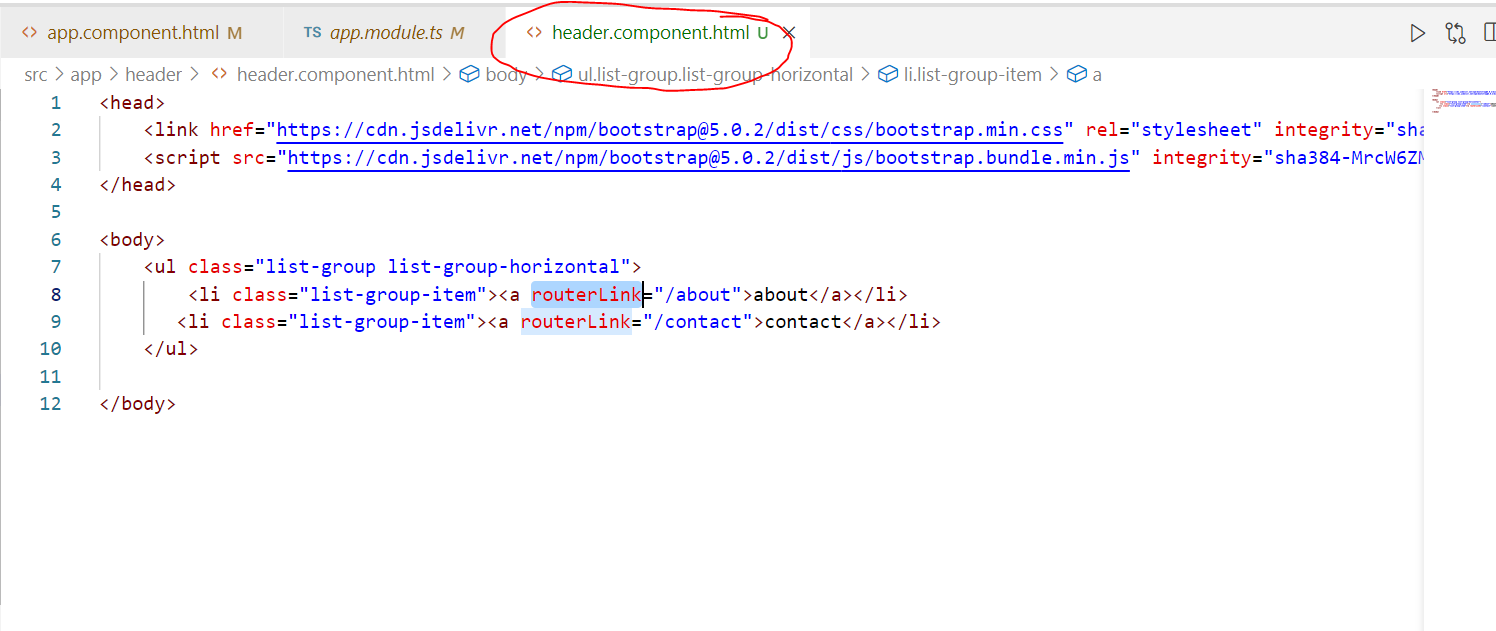
2. Add the router outlet



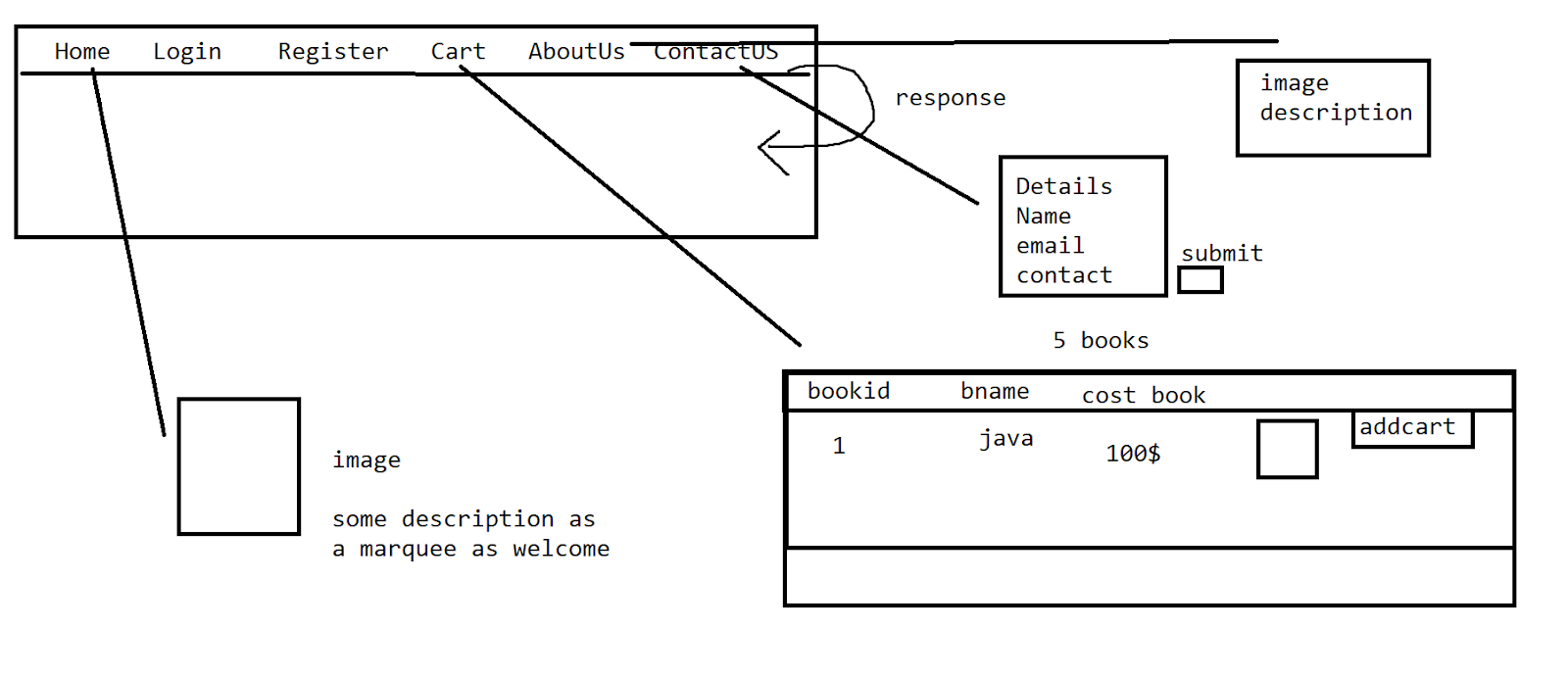
<app-header></app-header>

<router-outlet></router-outlet>

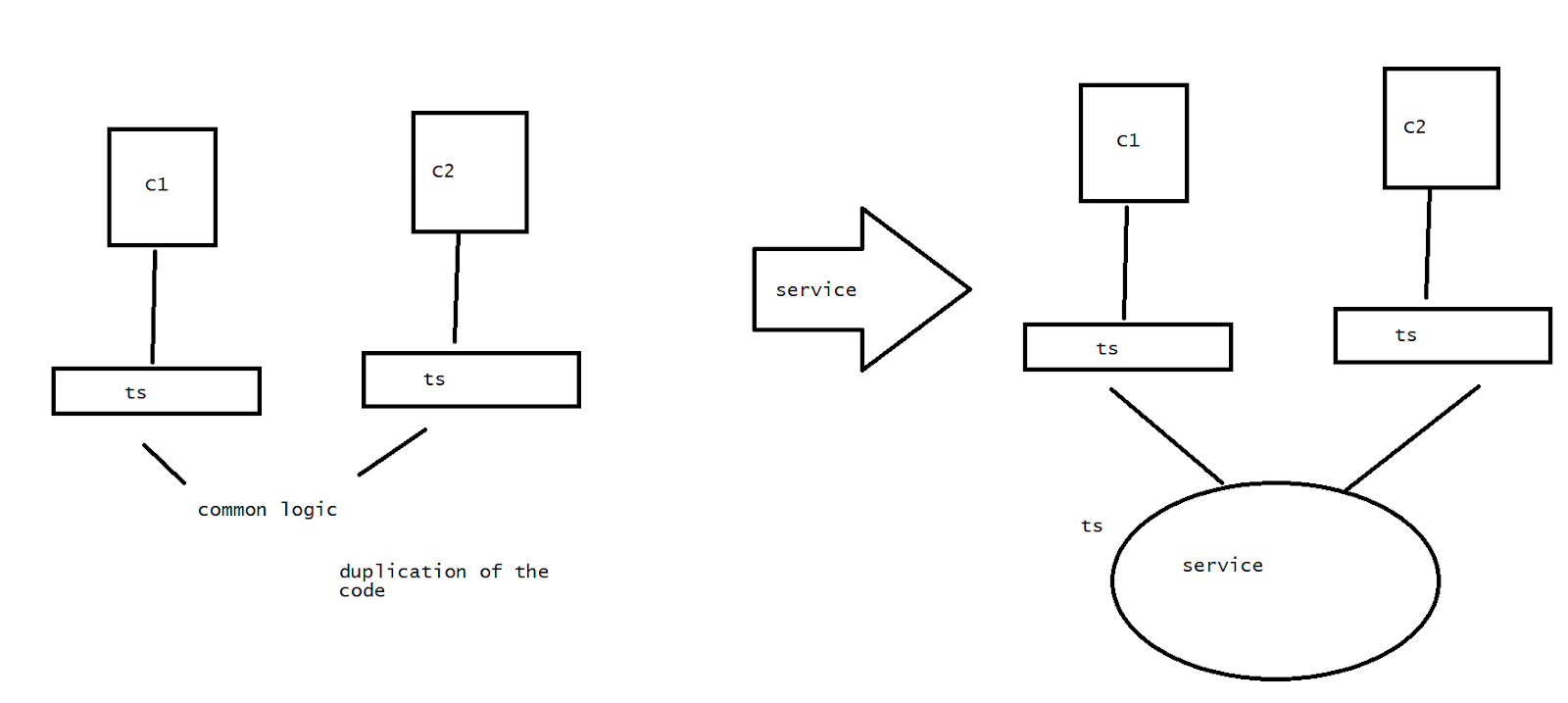
3. Define the links



Task:  15 min



>Service



Service is a reusable code that is applied on the multiple components

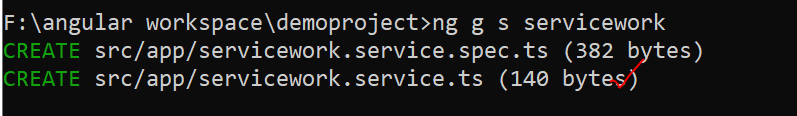
Rather than we define the data/config on the separate files of .ts and duplicating the LOC . In that situation, we are going to use the service in making that as a common config file.

Model-1

We update the ts of the service then the ts of the component which is using the service is going to get the updated logic

>service creation

 ng g s <servicename>

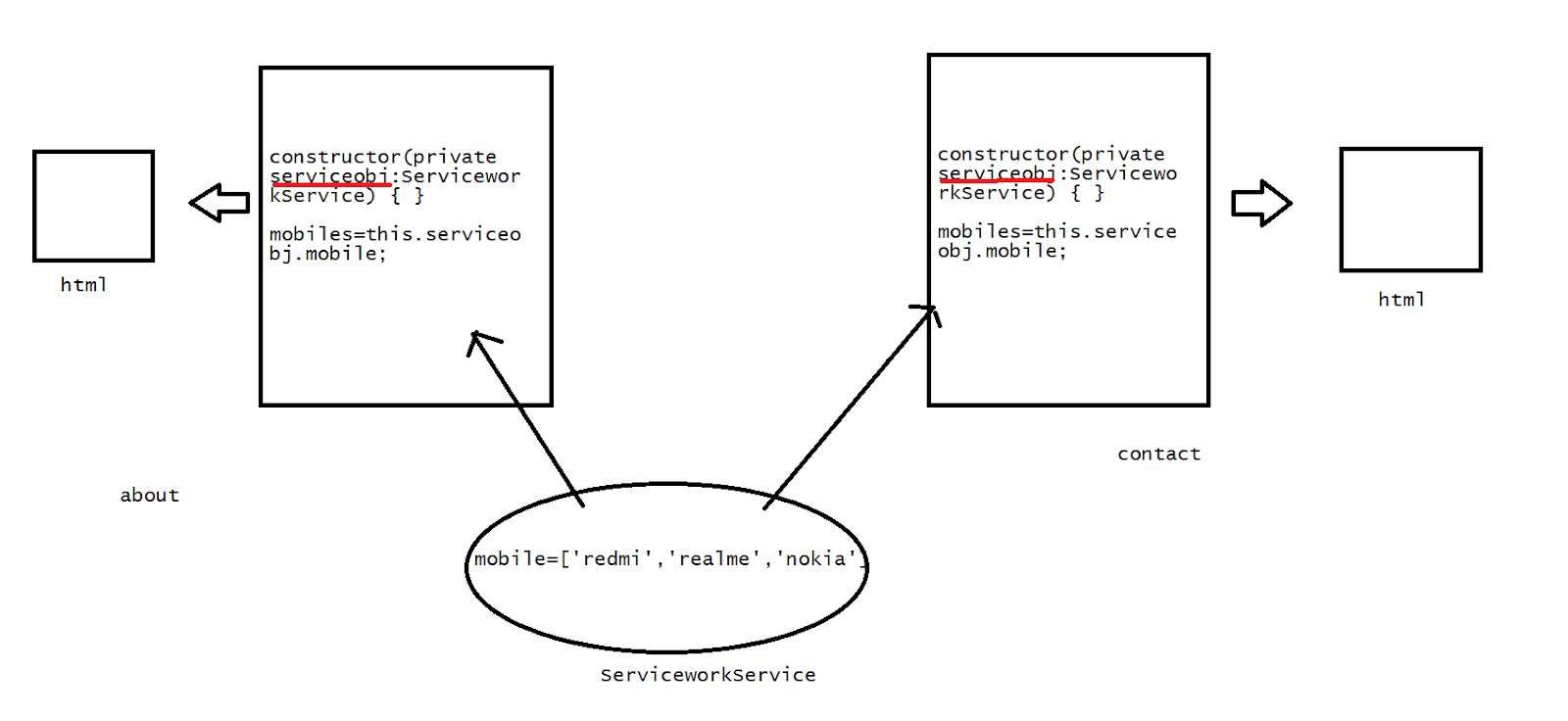


@Injectable({

  providedIn: 'root'

})

This annotation helps in applying changes on the app-root level of the component.



//service

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

@Injectable({

  providedIn: 'root'

})

export class ServiceworkService {

  constructor() { }

  mobile=['redmi','realme','nokia']

}

//about

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

import { ServiceworkService } from '../servicework.service';

@Component({

  selector: 'app-about',

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

  styleUrls: ['./about.component.css']

})

export class AboutComponent implements OnInit {

  constructor(private serviceobj:ServiceworkService) { }

  mobiles=this.serviceobj.mobile;

  ngOnInit(): void {

  }

}

//html

<div \*ngFor="let m of mobiles">

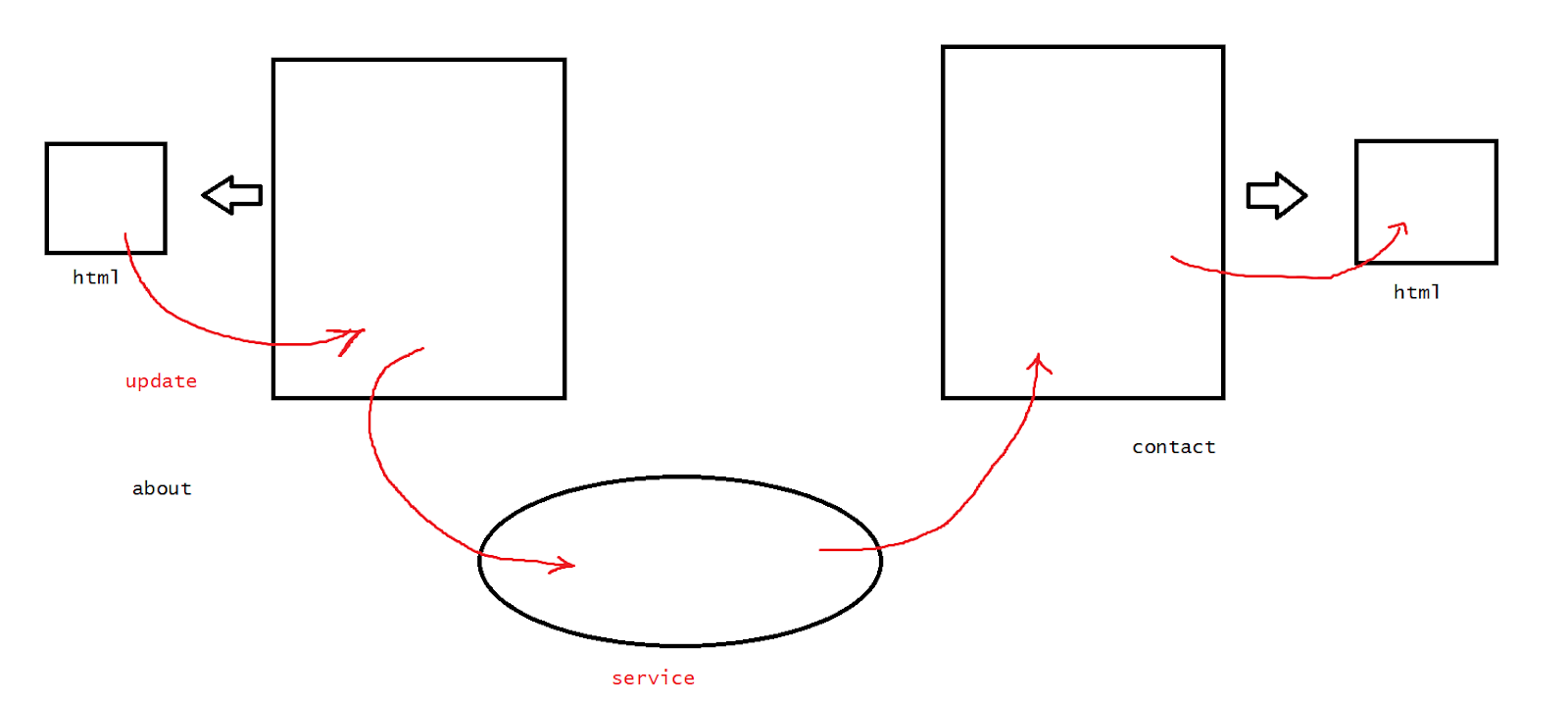
{{m}}

</div>

//repeat the same logic for contact

//Model 2

If the component -html has updated the individual ts file by using we updated the service , then the components that are using this service will be auto updated



//updated about logic

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

import { ServiceworkService } from '../servicework.service';

@Component({

  selector: 'app-about',

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

  styleUrls: ['./about.component.css']

})

export class AboutComponent implements OnInit {

  constructor(private serviceobj:ServiceworkService) { }

  mobiles=this.serviceobj.mobile;

  addmobile(){

    this.serviceobj.mobile.push("samsung")

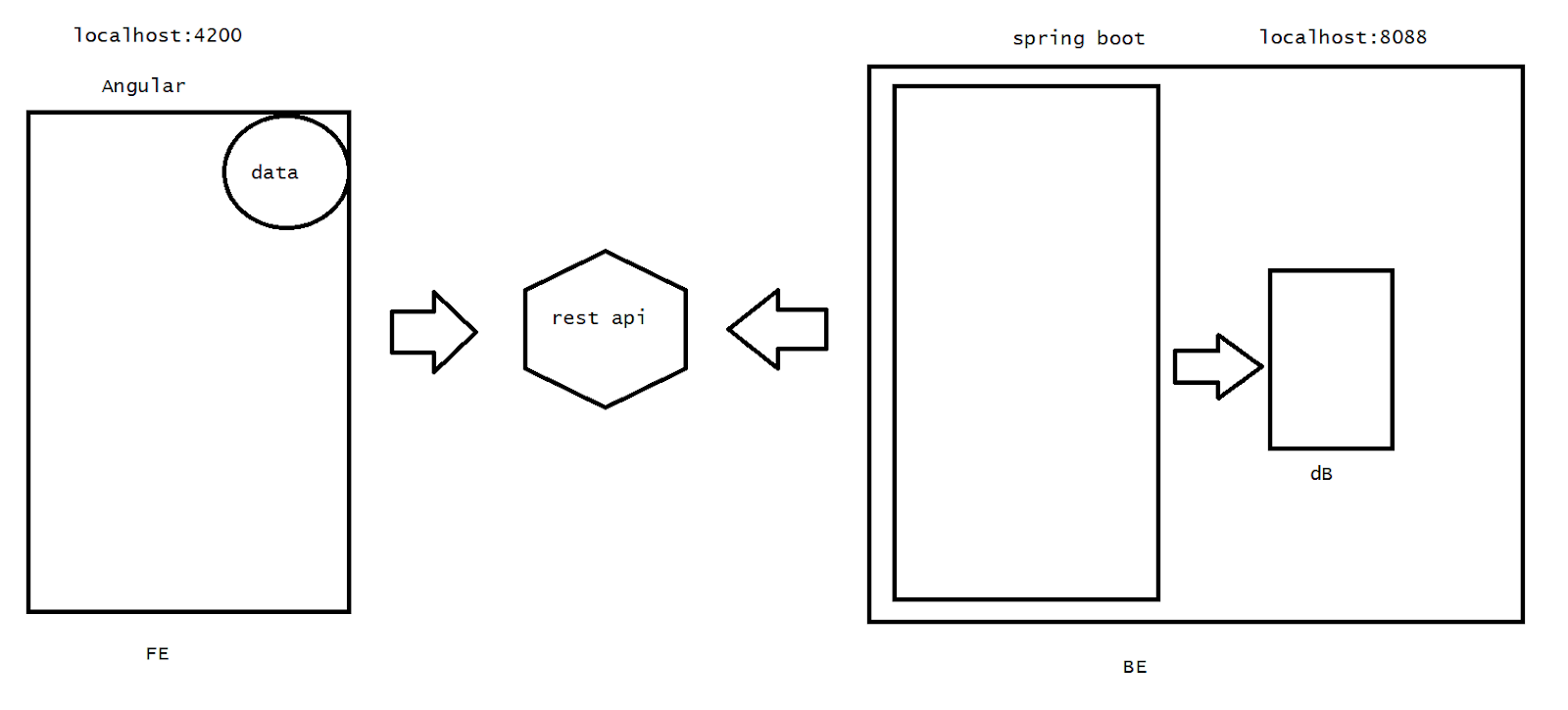
  }

  ngOnInit(): void {

  }

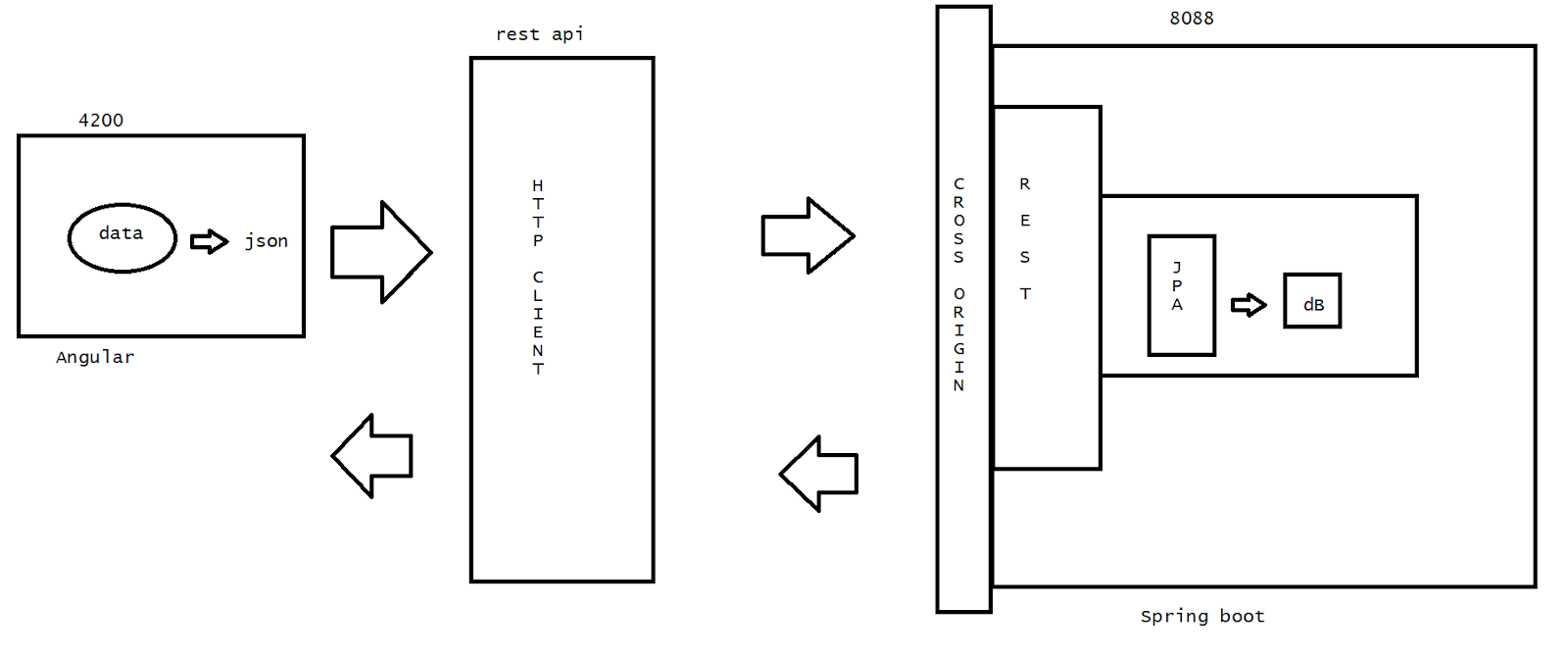
}

//Angular with spring boot



Challenges:

1. Two different ports
2. The external communication into the spring boot application will block - cross-origin



In Agile - 10 days -sprint

5 days / 6 days dev and deployment

3 days for testing

1 day for demo

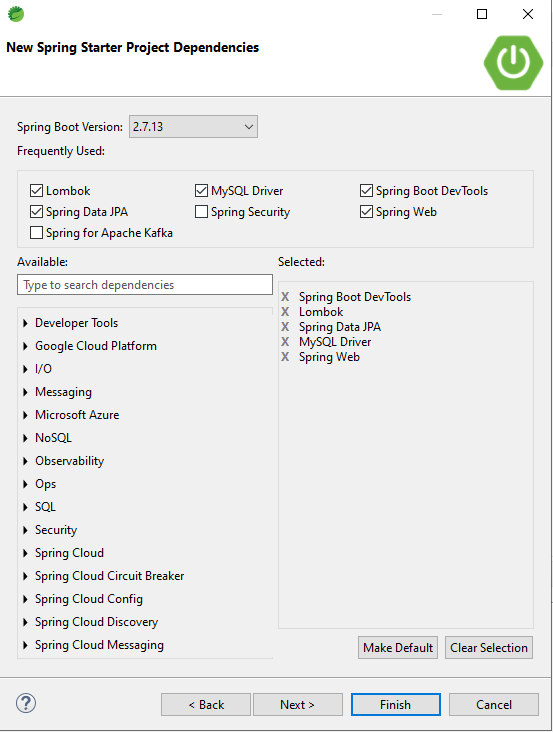
6 days - FE and BE

BE - Spring boot + JPA-Rest-POSTMAN

FE - Angular

1. Spring boot app->run postman
2. Angular app
3. Integrate the angular with the SB

Spring boot app->run postman:



package com.example.demo;

import javax.annotation.Generated;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private String email;

private int experience;

private String domain;

}

package com.example.demo;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepo extends JpaRepository<User,Integer> {

List<User> findByemail(String email);

}

package com.example.demo;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

@RestController

@CrossOrigin(origins="\*")//for all external networks we can use hitting this requests

public class UserController {

@Autowired

UserRepo repo;

//insert

@PostMapping("/register")

public String register(@RequestBody User user) {

repo.save(user);

return "Hi " +user.getName()+" is registered successfully";

}

//list of users

@GetMapping("/getAllusers")

public List<User> findAllUsers(){

return repo.findAll();

}

//delete record with id

@DeleteMapping("/cancel/{id}")

public List<User> cancelregistration(@PathVariable int id){

repo.deleteById(id);

//after delete i need to show the list of recors that are updated

return repo.findAll();

}

//search with email

@GetMapping("/findbyemail/{email}")

public List<User> findUser(@PathVariable String email){

return repo.findByemail(email);

}

//update

}

server.port=8088

#================================================================

#Jpa hibernate

spring.jpa.hibernate.ddl-auto=update

spring.jpa.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.jpa.show-sql=true

#================================================================

#datasource

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

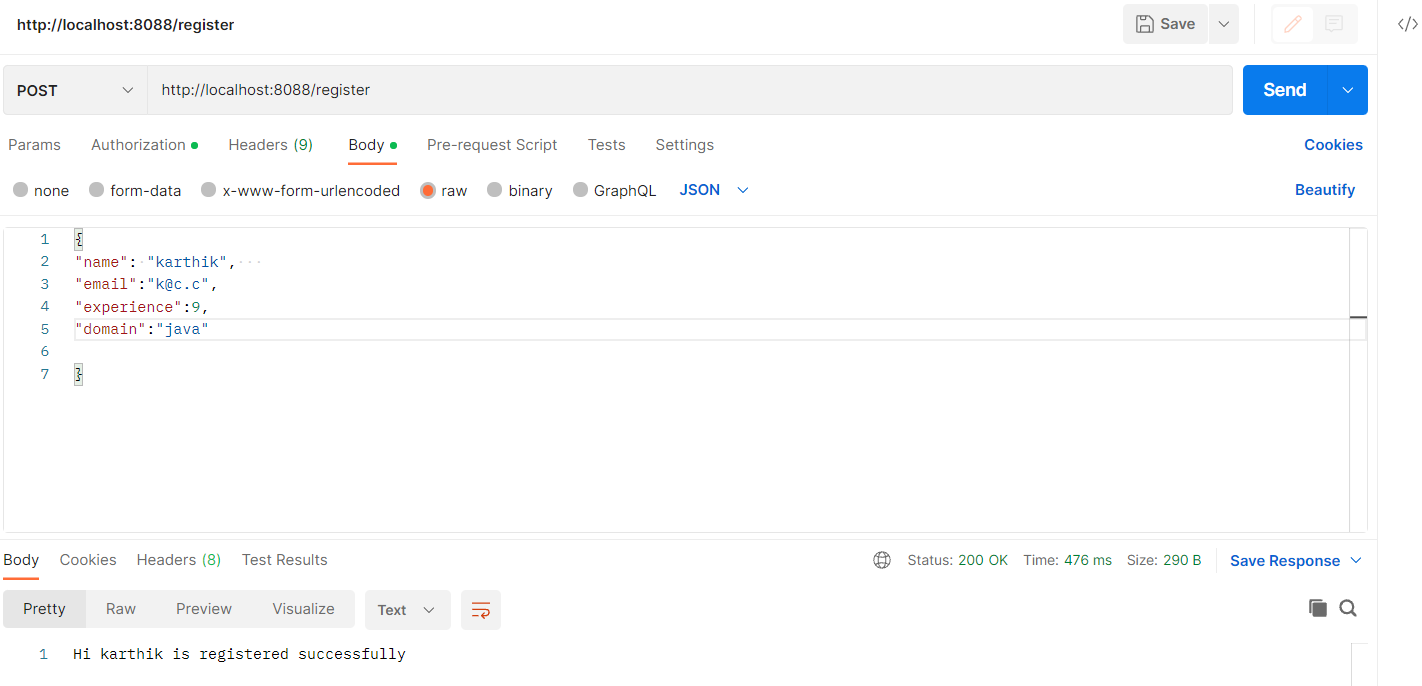
spring.datasource.url=jdbc:mysql://localhost:3306/db1

spring.datasource.username=root

spring.datasource.password=123456

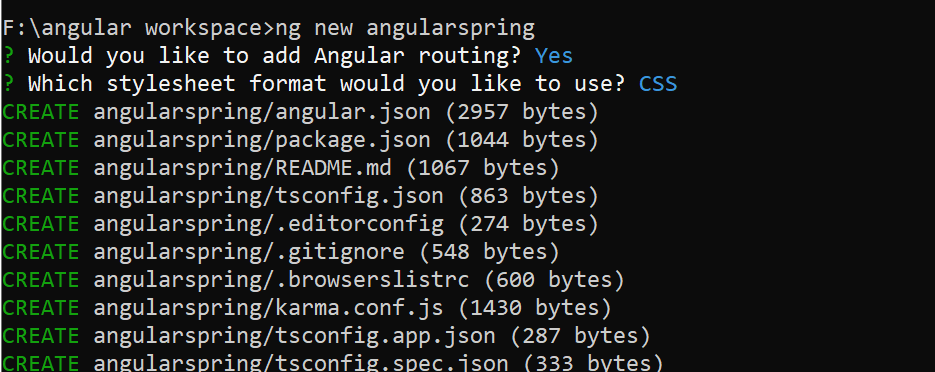
#================================================================

==================Test the above logic with POSTMAN====================

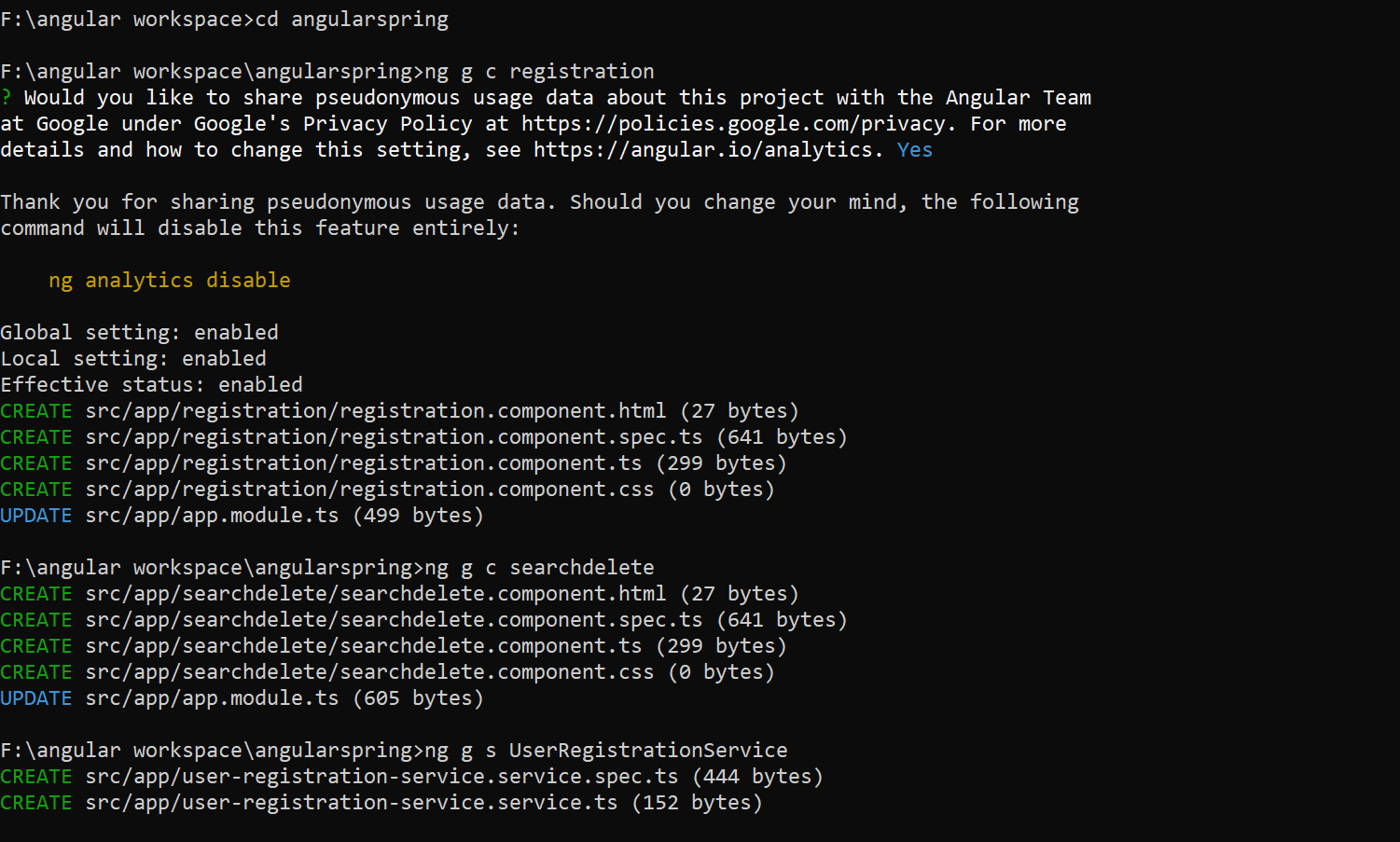


====Angular======

1. Create a new project

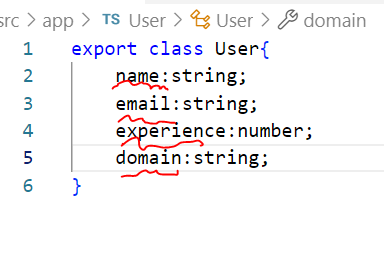


1. Create the components and the service

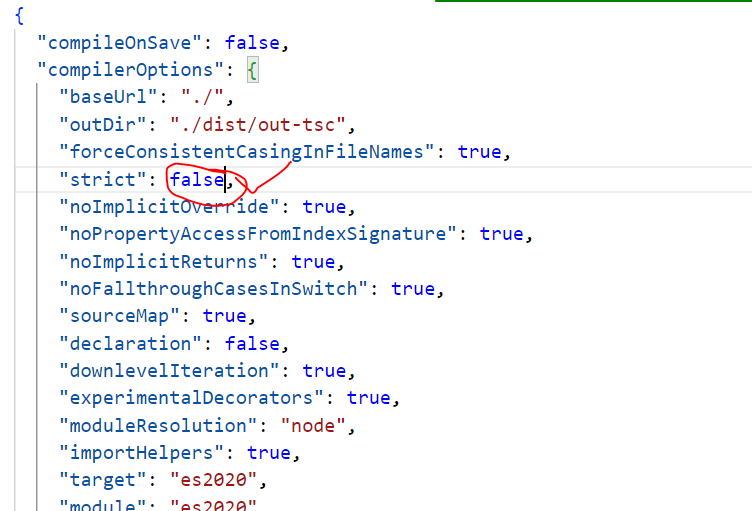


1. Create a  class

If we get errors



Tsconfig.json



Create User.ts

export class User{

    name:string;

    email:string;

    experience:number;

    domain:string;

}

1. Perform config in app.module.ts

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

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

import { AppRoutingModule } from './app-routing.module';

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

import { RegistrationComponent } from './registration/registration.component';

import { SearchdeleteComponent } from './searchdelete/searchdelete.component';

import { RouterModule, Routes } from '@angular/router';

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

import { HttpClientModule } from '@angular/common/http'

import { UserRegistrationServiceService } from './user-registration-service.service';

const routes:Routes=[

  {path:"",redirectTo:"register",pathMatch:"full"},

  {path:"register",component:RegistrationComponent},

  {path:"search",component:SearchdeleteComponent}

]

@NgModule({

  declarations: [

    AppComponent,

    RegistrationComponent,

    SearchdeleteComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

    HttpClientModule,

    RouterModule.forRoot(routes)

  ],

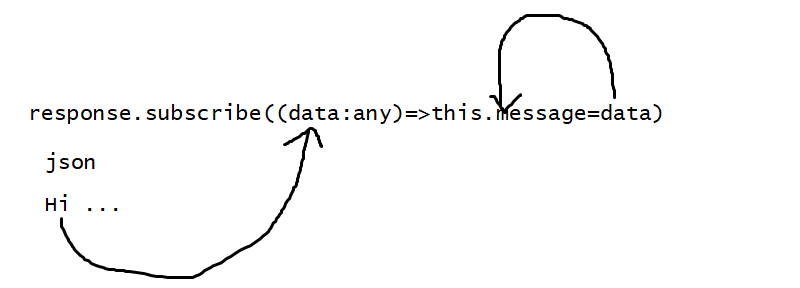
  providers: [UserRegistrationServiceService],//service details

  bootstrap: [AppComponent]

})

export class AppModule { }

1. Registration component



//html

{{message}}

<form>

Name:<input type="text" name="name" [(ngModel)]="user.name"><br>

Email:<input type="email" name="email" [(ngModel)]="user.email"><br>

Experience:<input type="text" name="exp" [(ngModel)]="user.experience"><br>

Domain:<input type="text" name="domain" [(ngModel)]="user.domain"><br>

<input type="submit" value="register" (click)="registerNow()"><br>

</form>

//ts

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

import { User } from '../User';

import { UserRegistrationServiceService } from '../user-registration-service.service';

@Component({

  selector: 'app-registration',

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

  styleUrls: ['./registration.component.css']

})

export class RegistrationComponent implements OnInit {

user:User=new User();

  constructor(private service:UserRegistrationServiceService) { }

message=""

  ngOnInit(): void {

  }

public  registerNow(){

  //json resposne

let response=this.service.doregistration(this.user);

response.subscribe((data:any)=>this.message=data);

}

}

//service code

import { HttpClient } from '@angular/common/http';

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

@Injectable({

  providedIn: 'root'

})

export class UserRegistrationServiceService {

  constructor(private http:HttpClient) { }

  //insert

  //if the response is a java object then it will be auto converted to json if not we need to convert it

  public doregistration(user:any){

    return this.http.post("http://localhost:8088/register",user,{responseType:'text' as 'json'});

  }

//getusers

public getusers(){

  return this.http.get("http://localhost:8088/getAllusers");

}

//search by email

public getuserbyemail(email:any){

  return this.http.get("http://localhost:8088/findbyemail/"+email);

}

//delete by id

public deletebyid(id:any){

  return this.http.delete("http://localhost:8088/cancel/"+id);

}

}

//searchdelete component

//html

Enter the email to search <input type="text" name="email" [(ngModel)]="email"><button (click)="finduserbyemail()">search</button>

<h1><i>List of users with experience and domain</i></h1>

<table border="1">

<tr><th>Id</th><th>Name</th><th>Email</th><th>Experience</th><th>Domain</th></tr>

<tr \*ngFor="let user of users">

<td>{{user.id}}</td>

<td>{{user.name}}</td>

<td>{{user.email}}</td>

<td>{{user.experience}}</td>

<td>{{user.domain}}</td>

<td><button (click)="deleteuser(user.id)">Delete</button></td>

</tr>

</table>

//ts

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

import { UserRegistrationServiceService } from '../user-registration-service.service';

@Component({

  selector: 'app-searchdelete',

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

  styleUrls: ['./searchdelete.component.css']

})

export class SearchdeleteComponent implements OnInit {

  email:any;

  users:any;

  finduserbyemail(){

    let response=this.service.getuserbyemail(this.email);

    response.subscribe((data:any)=>this.users=data);

  }

  deleteuser(id:number){

    let response=this.service.deletebyid(id);

    response.subscribe((data:any)=>this.users=data);

  }

  constructor(private service:UserRegistrationServiceService) { }

  //by default if we dont need any function to be called then we need to write the code below

  ngOnInit(): void {

    let response=this.service.getusers();

    response.subscribe((data:any)=>this.users=data);

  }

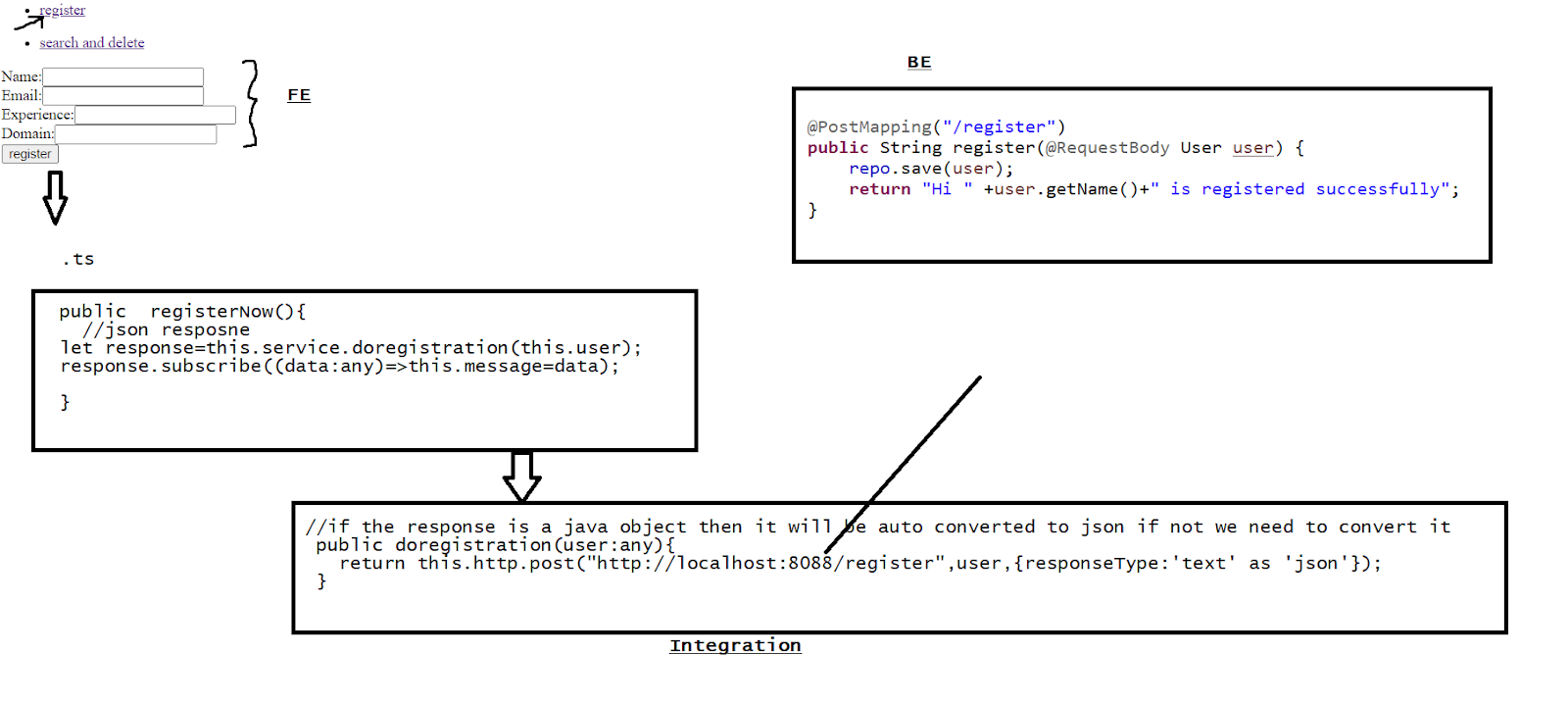
}

//app.component.html

<ul><li><a routerLink="/register">register</a></li></ul>

<ul><li><a routerLink="/search">search and delete</a></li></ul>

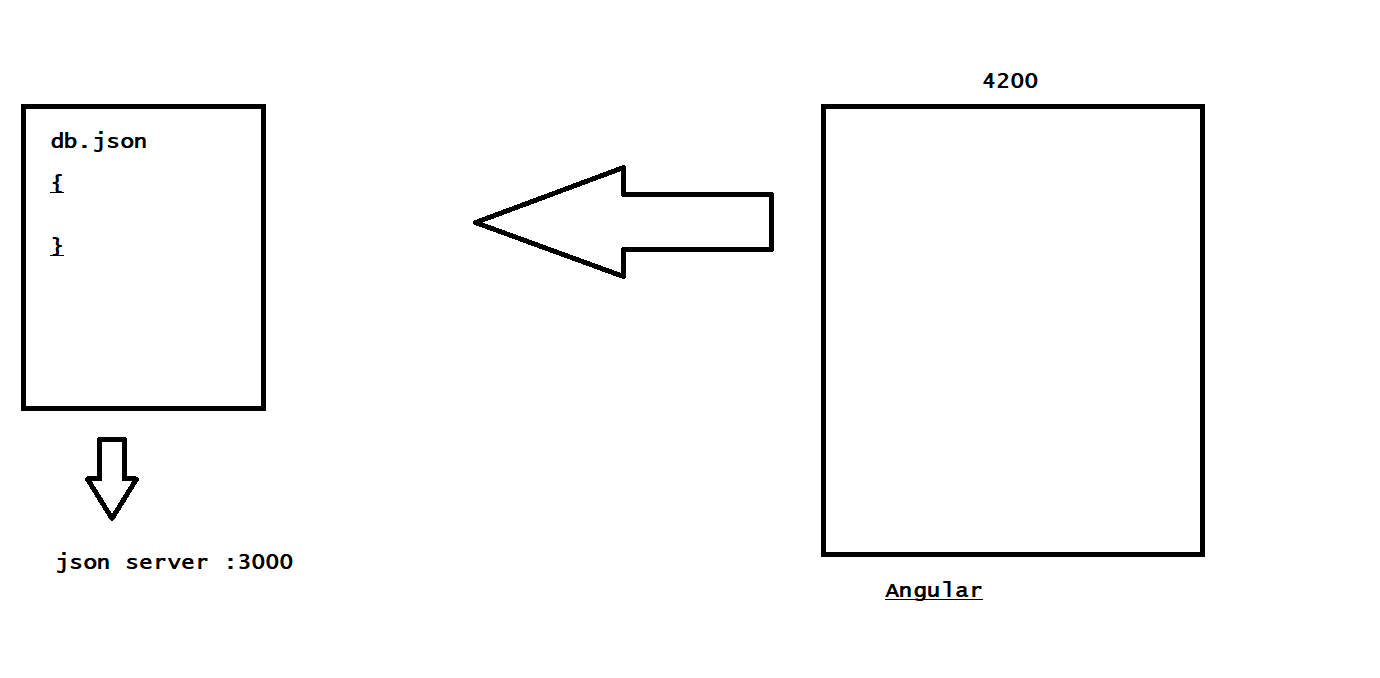
<router-outlet></router-outlet>



Assignment :

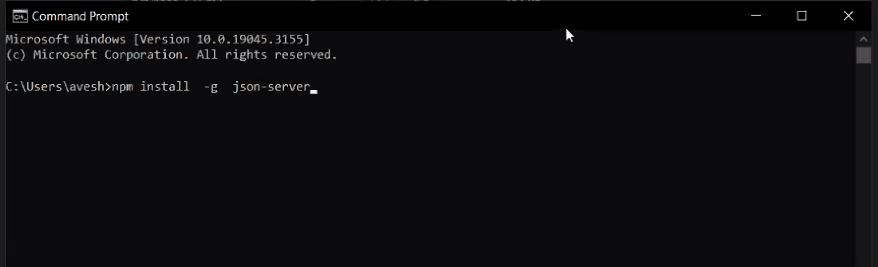
Complete update module

Json server



It is a json db server that can be manipulated according to the crud operations

> Install the json server



=>create db.json

{

"students":[

{

"id":1,

"name":"karthik"

},

{

"id":2,

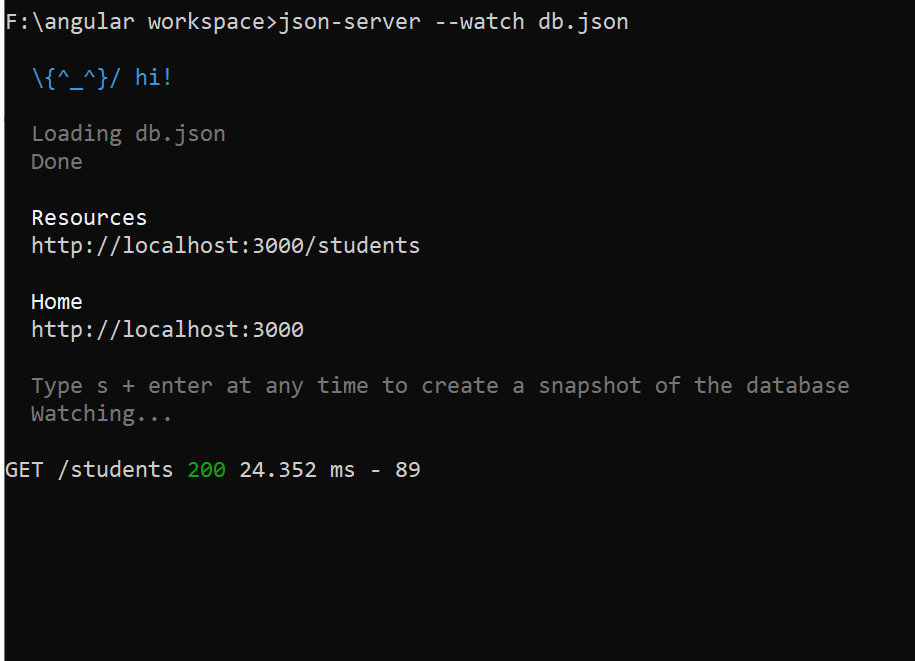
"name":"suresh"

}

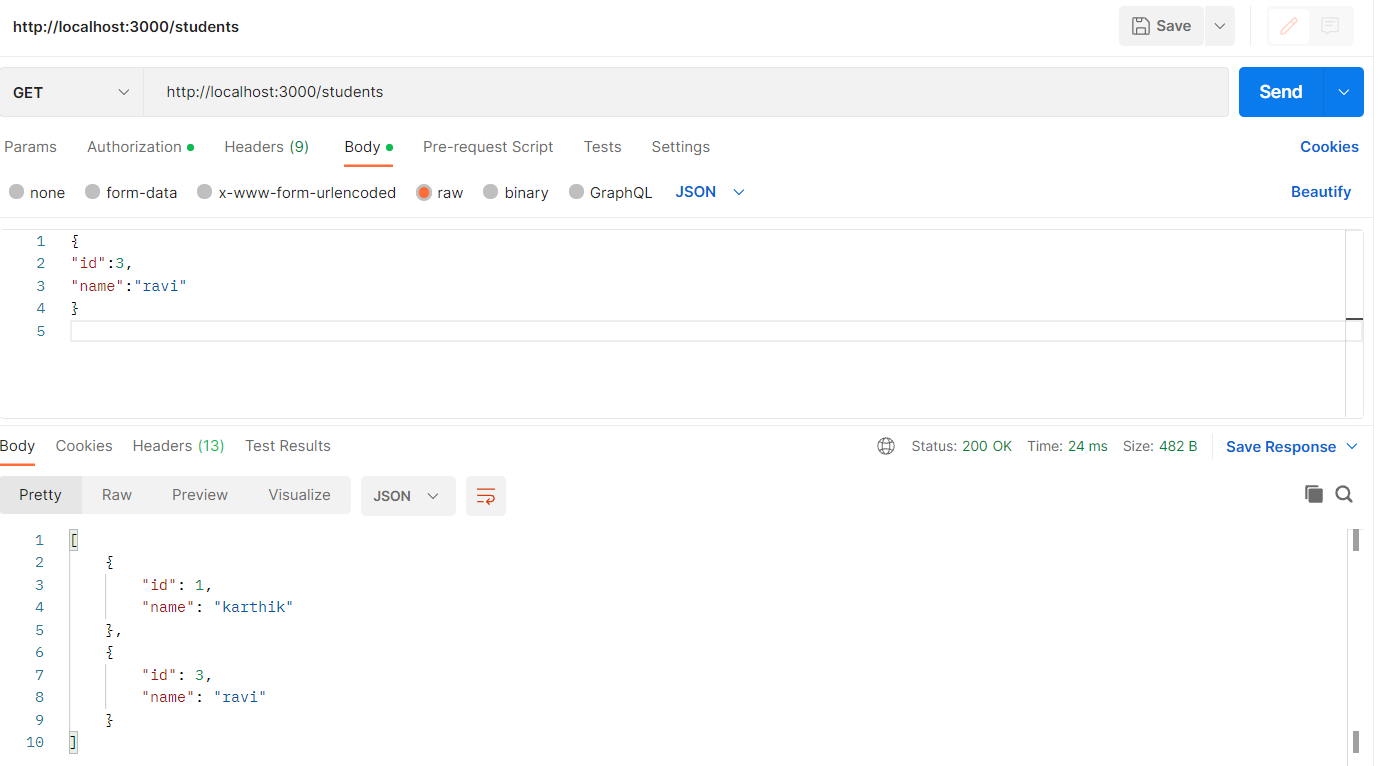
]

}

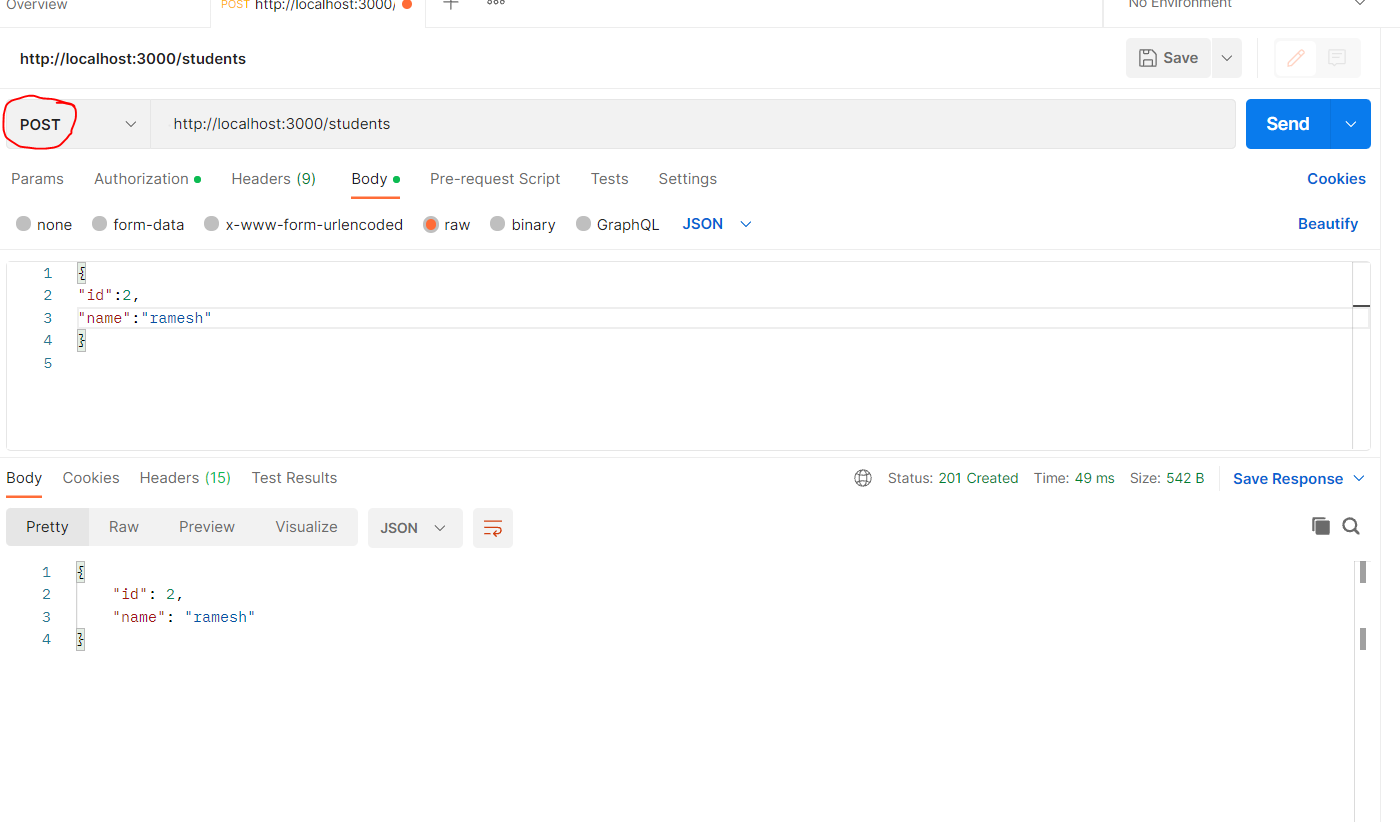
Go to the location and start the json server

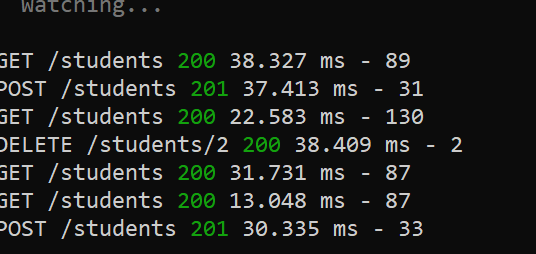


//Get

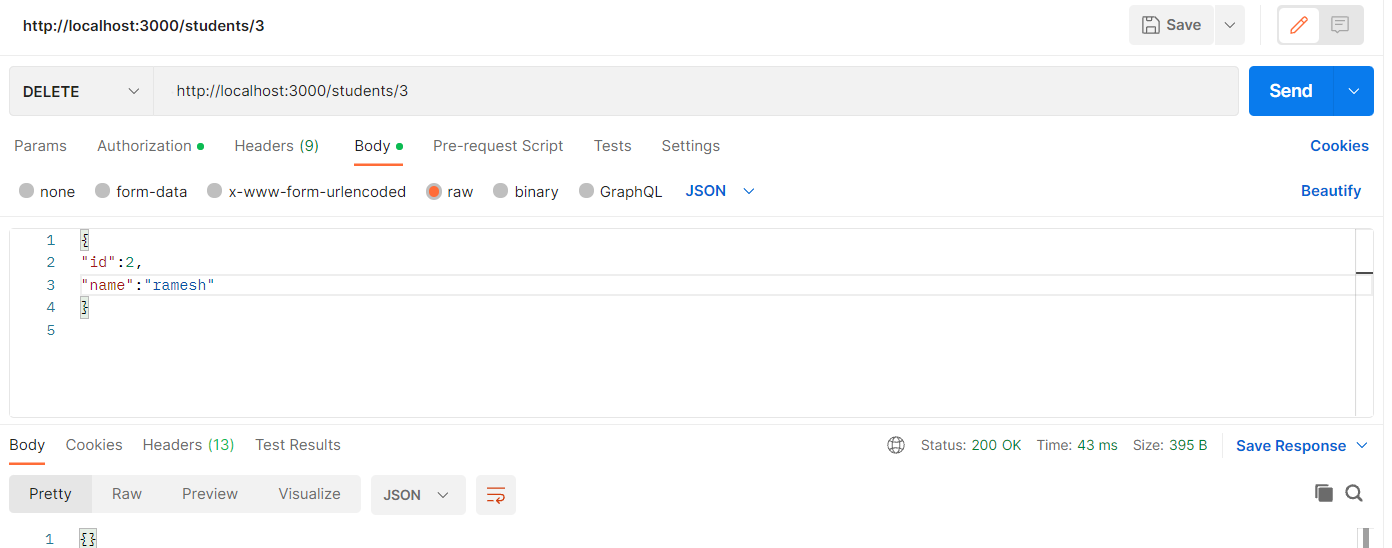


//post

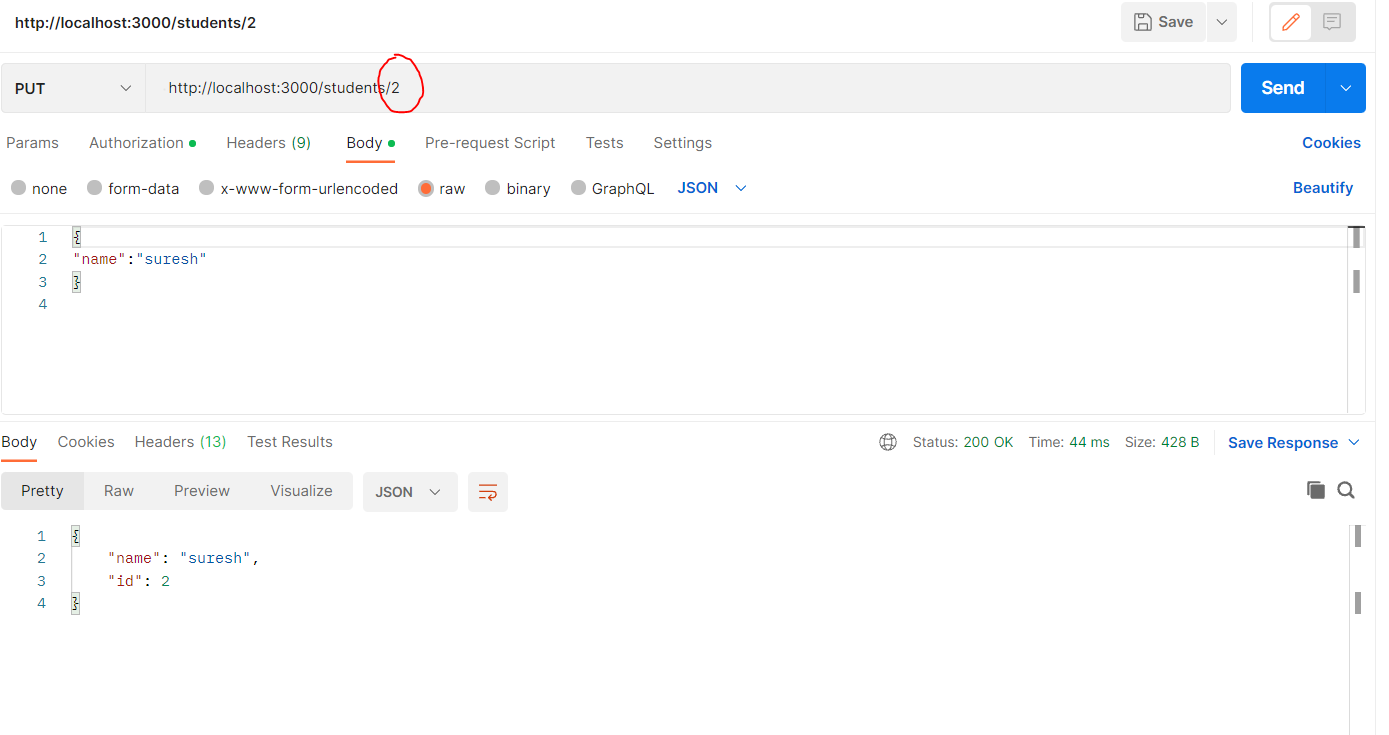




//delete



//update



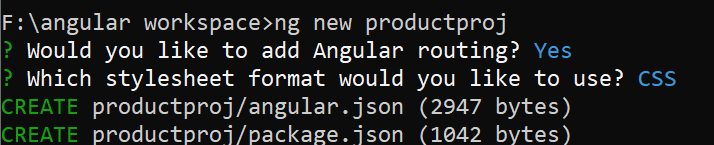
Task 👍

Take products

1. Id , product name , cost   ->perform json crud operations

>Angular with json server

1. Create an angular project



1. Load the project in the vs
2. config the below file

https://lh3.googleusercontent.com/i_PTZ6D5ORd_yrd4ZKOg55bAhUrGsgZXmO6GpnmiUoH89DOL3AMcQroNHGC1wTIFyZxhkRzlwEY-4Frj03tBAQU9s0j0daRImCDjez-gFWNzhlruLqprLNMzixw-vxCaIz5zcw4TWrLCmxfmDFXAQMU

declare module "\*.json" {

    const value: any;

    export default value;

  }

1. Create a json file

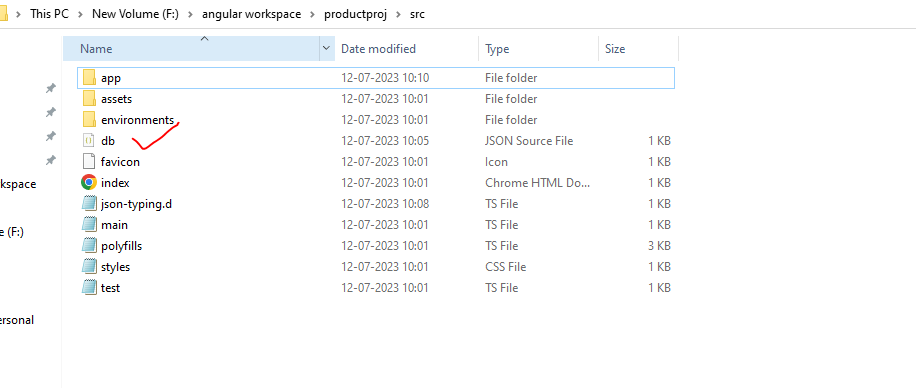


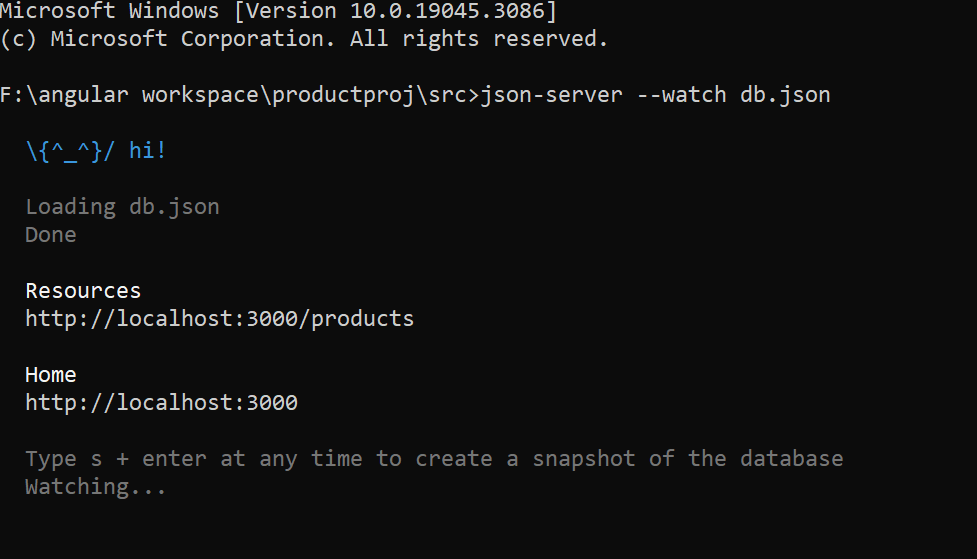
{

    "products":[]

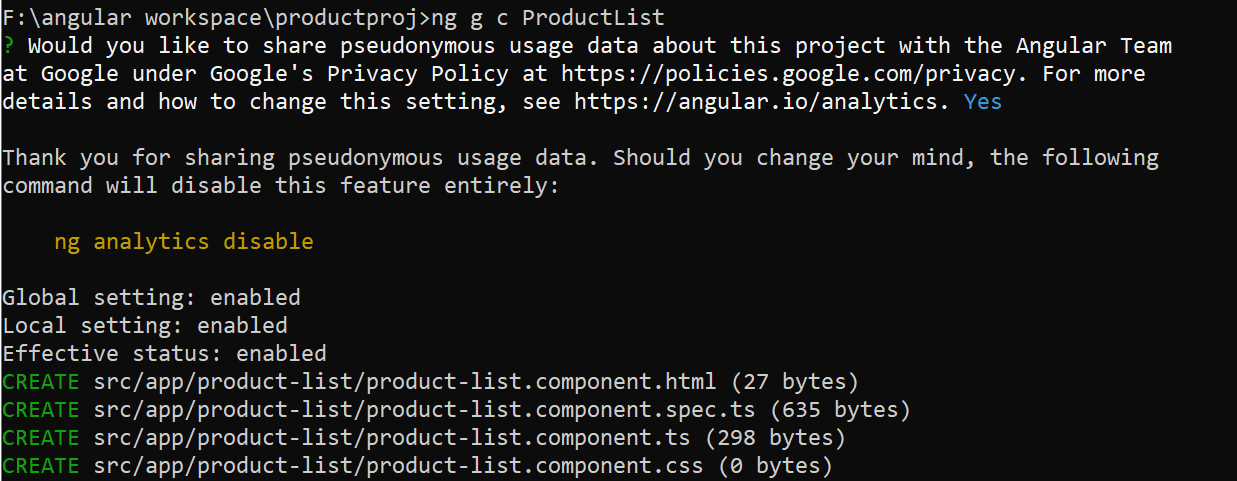
}

//start the json server

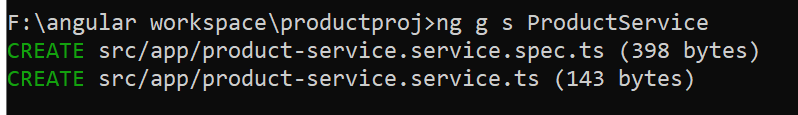




//create a component



//create a service



//app.module.ts

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

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

import { AppRoutingModule } from './app-routing.module';

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

import { ProductListComponent } from './product-list/product-list.component';

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

import { RouterModule, Routes } from '@angular/router';

import { HttpClientModule } from '@angular/common/http'

const routes:Routes=[

  {path:"",redirectTo:"products",pathMatch:"full"},

  {path:"products",component:ProductListComponent}

]

@NgModule({

  declarations: [

    AppComponent,

    ProductListComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

    HttpClientModule,

    RouterModule.forRoot(routes)

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

//taken a dummy data to display

{

    "products":[

        {

            "id":1,

            "name":"book",

            "cost":20

        }

    ]

}

//service

import { HttpClient } from '@angular/common/http';

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

@Injectable({

  providedIn: 'root'

})

export class ProductServiceService {

  constructor(private http:HttpClient) { }

 public getProducts(){

  return this.http.get("http://localhost:3000/products");

 }

}

//productlist ts

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

import { ProductServiceService } from '../product-service.service';

@Component({

  selector: 'app-product-list',

  templateUrl: './product-list.component.html',

  styleUrls: ['./product-list.component.css']

})

export class ProductListComponent implements OnInit {

products:any;

  constructor(private service:ProductServiceService) { }

  ngOnInit(): void {

    let response=this.service.getProducts();

    response.subscribe((data:any)=>this.products=data);

  }

}

//html

<ul>

<li \*ngFor="let product of products">

    {{product.id}}----{{product.name}}-----{{product.cost}}

</li>

</ul>

//Json server

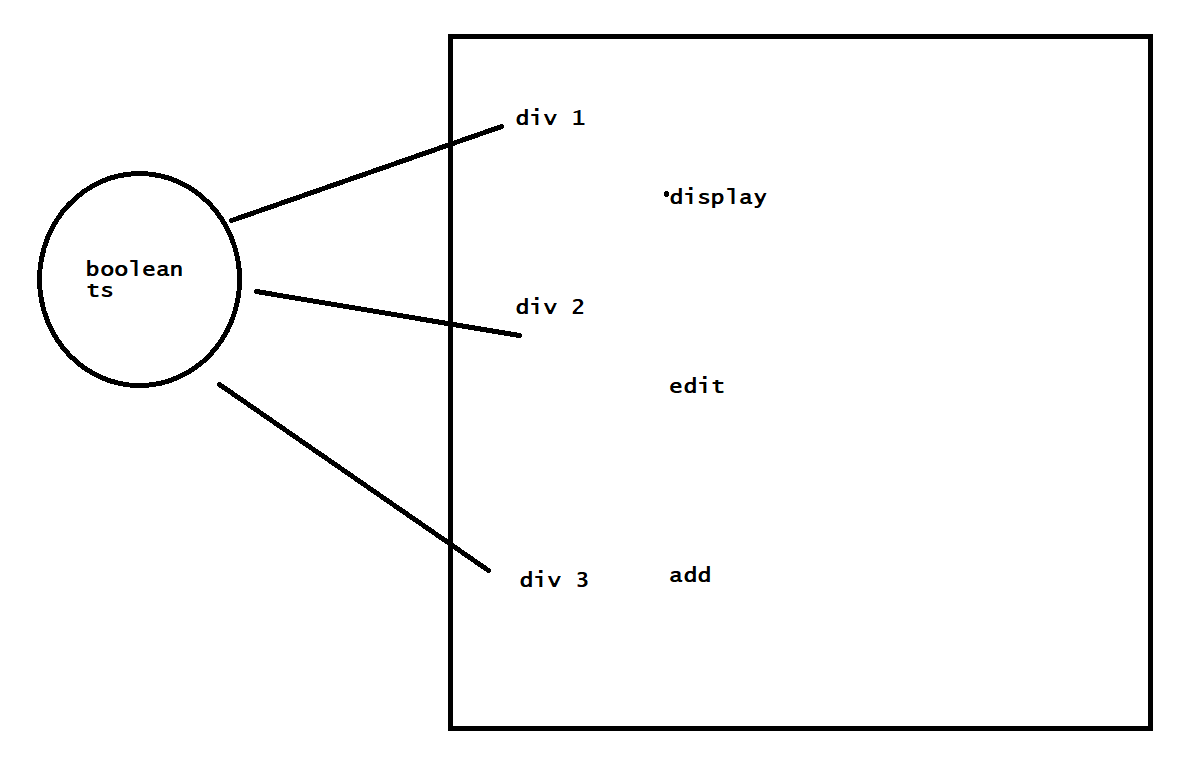
1. HttpClient -action - get ,post ,put
2. Data as the path variable

//Task

Post operation

Update

delete



Submit -event

//practice

Event -spring boot

Taxi -json , spring boot

//delete

deleteProduct(id:number):Observable<any>{

return this.http.delete("http://localhost:3000/products/"+id);

 }

Ts:

products:any[]=[];

public delete(id:number){

    this.service.deleteProduct(id).subscribe(()=>{this.products=this.products.filter(product=>product.id!==id)});

  }

Json file - Take the questions and keep the ans

Completed a quiz portal - with a score at the last.

{

“Questions”:[

{

“Question”  :  ”  ”,

“Choices”   :  [“  ”,”  ” ],

“Correctanswer”  :  ” ”

}

]

}

—---------how to access json data on to the html —-----------------------------

//json

[

    {

        "Q":"what is ur name?",

        "a":"ravi",

        "b":"suresh"

    }

]

//ts

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

//import the json data into the object

import jsondata from '../assets/question.json'

interface Question{

  Q:string;

  a:string;

  b:string;

}

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'jsonwork';

  display:Question[]=jsondata;

}

//html

<div \*ngFor="let e of display">

<label>{{e.Q}}</label>

<br>

<input type="radio">{{e.a}}<br>

</div>

HTTPClient

questions:any[]=[];

this.http.get(‘assets/questions.json’).subscribe(data=>{this.questions=data.questions});

