TD 3 –Angular NG Project + Web page element debugging + Bootstrap

TD Objectives

During this hands-on, you will get familiar to basic Web Angular development, and static CSS page design using the Bootstrap library.

We will use Chrome Developer Tools, and use Intellij Ultimate / WebStorm (you may find equivalent features in Visual Studio Code, or Eclipse)

Step 1 : setup new Angular project, Import in IDE

Pre-requisites: check you have nodejs installed, and angular cli (npm install -g @angular/cli)

Setup a new Angular project, using

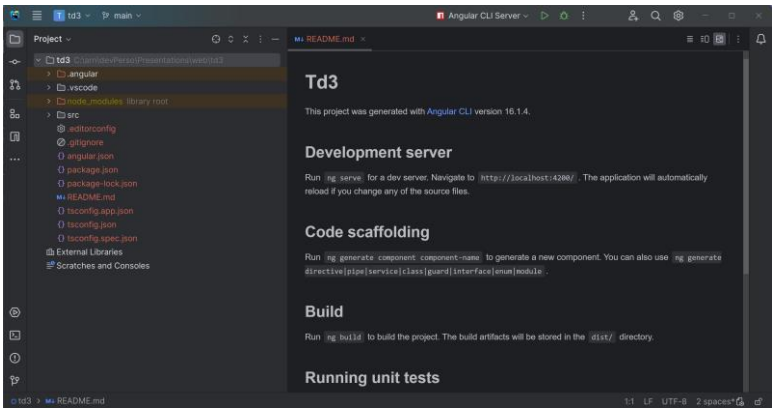
C:\td3> ng new

When prompted to use Angular rooting … use “Y” (override default “N” !)

When prompted for stylesheet format, use CSS.

This may take 5mn depending on your network bandwidth.

Then open your project from WebStorm IDE.



[Optional] Step 1 (next) : setup git

It is highly recommended to setup git for versioning your work.

By committing often your files, you have backup, and undo/redo history.

You can also save it on http://github.com or gitlab

By using “ng new”, you already have “.gitignore” file correctly configured (containing line “/node_modules”), so to finalize you git project, use only need to type

C:\td3> git init

C:\td3> git add .

C:\td3> git commit -m "init td3"

Then regularly use these git commands (maybe from your IDE, or github desktop app)

git add .

git commit -m "some message"

git diff

git log

git push

etc ...

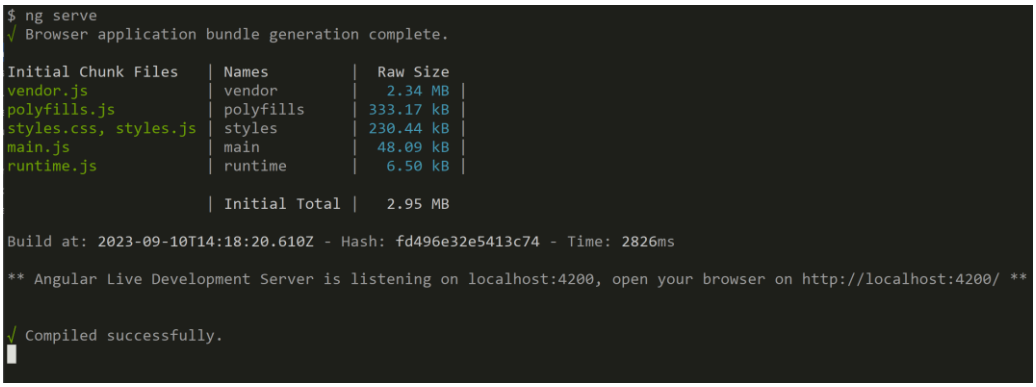
Step 2 : launch ng serve, Debug in Chrome DevTools

Launch your ng serve, using

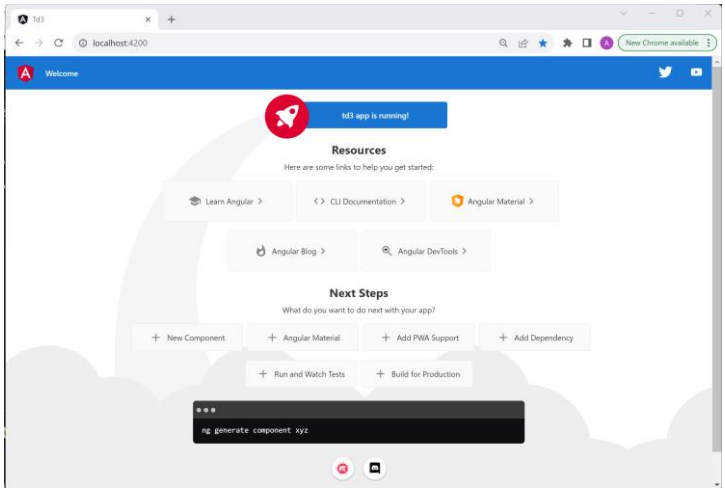
C:\td3> ng serve

(or alternatively "npm run start" or "npm run watch")

At this point, you should see



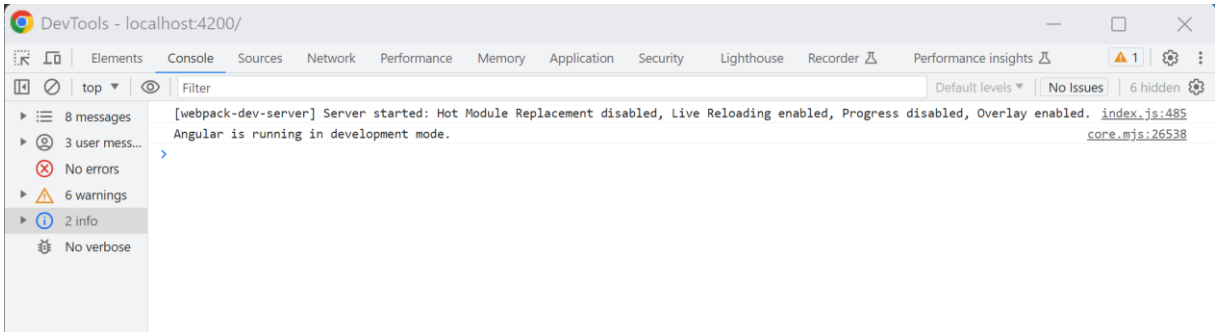
Once started, open you web browser(Chrome) on http://localhost:4200



Open Chrome Developer Tools (F12 or Ctrl+Shitf+I)

Step 2- A/ use DevTools > console tab

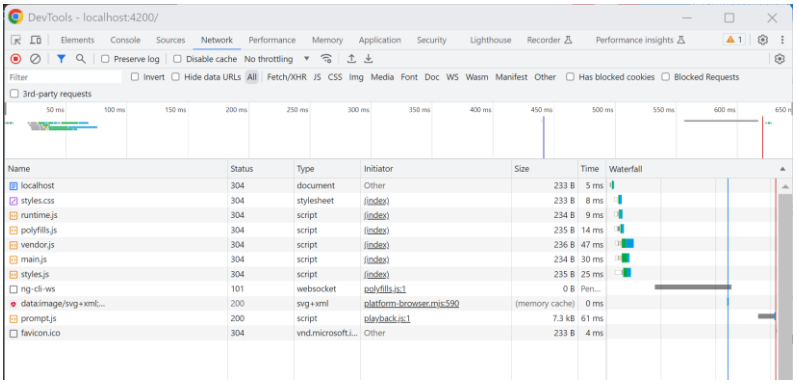
Ensure you are able to see logs in the console tab (default tab to be opened)



We will see in next questions how to add more logs in your code.

Step 2 – B / use DevTools > Network tab

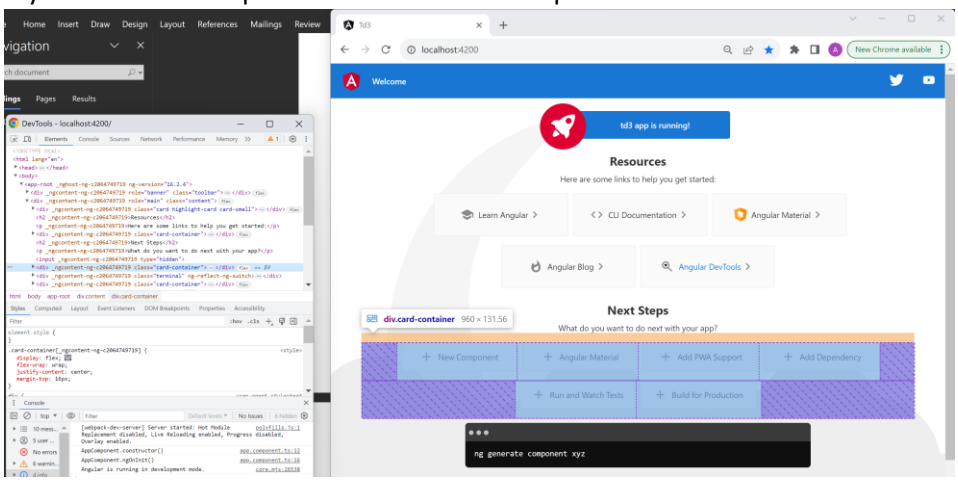
refresh page and see “http GET” for /index.html



This tab is especially important when debugging interaction between your front-end (angular) and your backend Rest api server, but not in this TD (see next TD).

Step 2 – C/ use DevTools > Element layout tab

Try to select the div panel below the “Next Steps” section



When selecting a Div either from mouse or selecting in the html, you get the mapping between html source code and how elements are displayed. You can also debug the CSS (border, margin, padding size..), and even change them directly in Chrome. Try it.

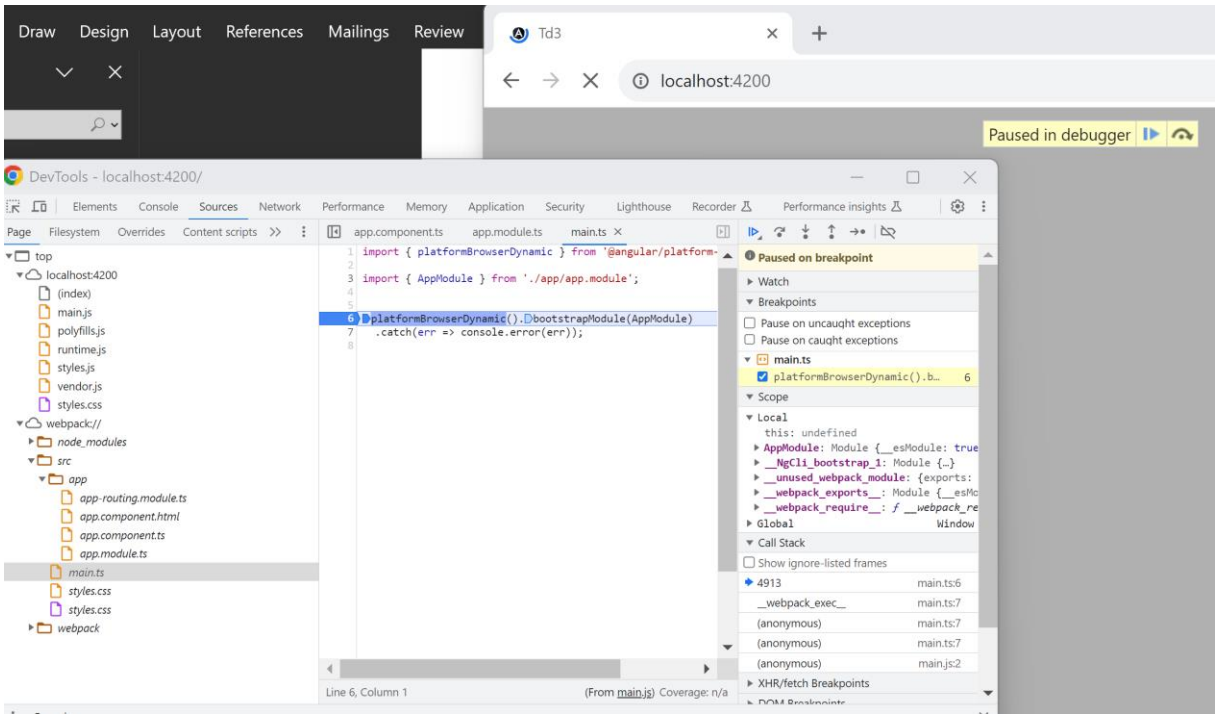
Step 2 – D / use DevTools > Source tab

From Chrome > Source tab, open the left element “webpack://”, then sub-element “src”, “app”, “main.ts”

Put a breakpoint in line 6 by clicking on the left margin on the main.ts file in the editor.

When refreshing your web page… This is the very first line that Angular execute.

Your breakpoint should cause your page to be “Paused in debugger”



In reality, your program is pausing in javascript file “main.js” (under top > localhost:4200 >main.js), but fortunately in development mode, it is correctly configured to use “source map” file, so you browse corresponding file in “top > webpack:// > src > app > main.ts” !

(Notice You can not edit file directly from Chrome without configuring corresponding “source directory”. You might do it, but it is not recommended)

Step 3 : modify app.ts … use your IDE autocompletion features

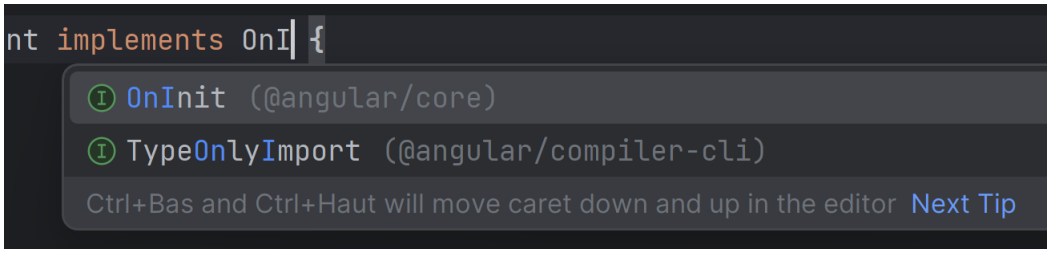
Go back in your WebStorm IDE, edit your file “src/app/app.component.ts”

Add a “constructor() { }” method to the class (it is implicit when none is found). Put “console.log(‘AppComponent.constructor()’)” in the constructor body.

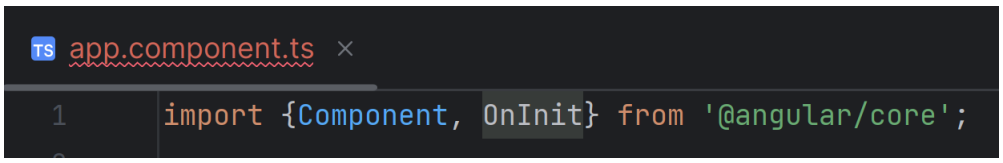
Check you are now able to see the console message in Chrome DevTool

You may also add console.log in method ngOnInit() {..} and add import + implements OnInit interface.

The expected code is below, but do not copy&paste … try to use auto-completion features from your IDE

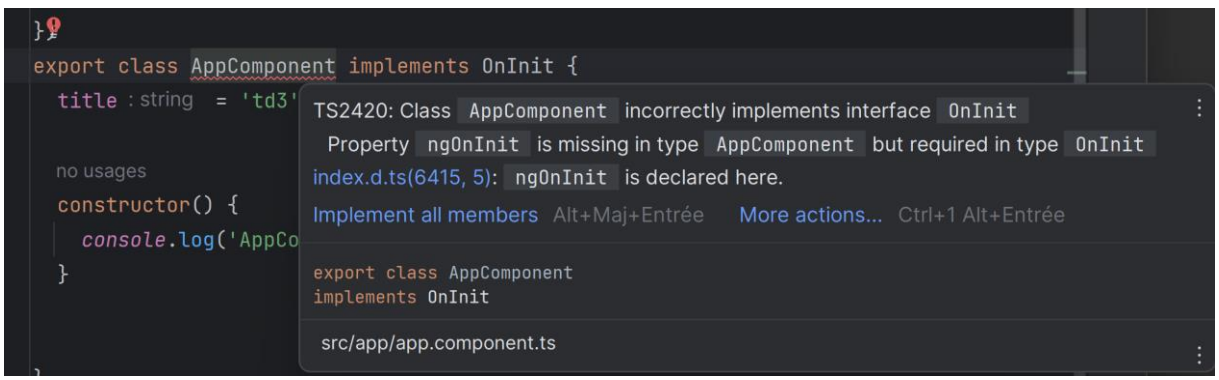


From your IDE, when selecting interface OnInit, it should automatically add corresponding import !

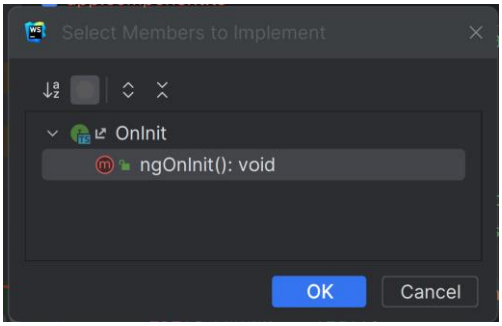


Finally, implement the requested method … using automatic error fixing

Either click on underlined error on “AppComponent”, or click on red light



Select all missing method to implement:



Finish editing ngOnInit() {.. }, add another “console.log(‘AppComponent.ngOnInit()”);”

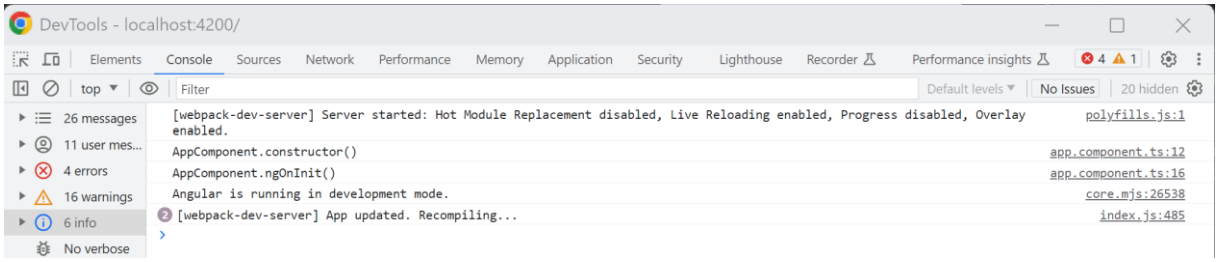
You should have:

app.component.ts

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

5+ usages
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  title : string  = 'td3';

  no usages
  constructor() {
    console.log('AppComponent.constructor()')
  }

  no usages
  ngOnInit() : void  {
    console.log('AppComponent.ngOnInit()')
  }
}
```

Now re-check you logs in Chrome DevTools > console.

You should see both lines :

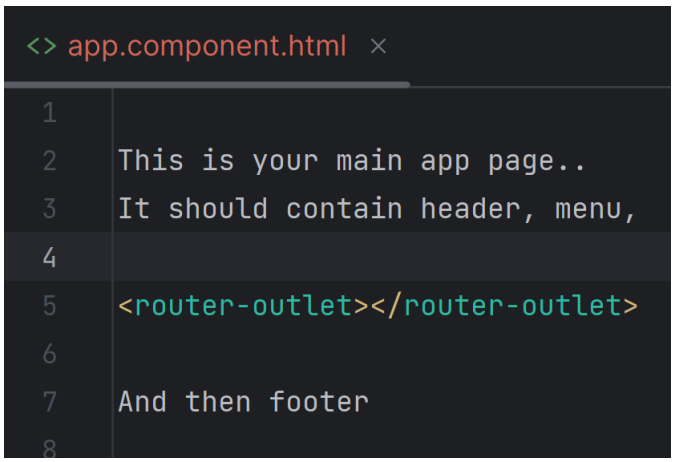


Remark :

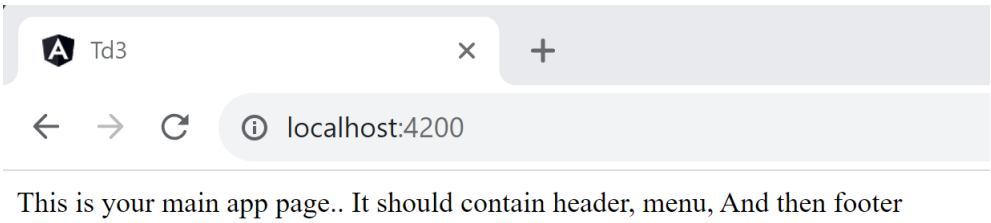
- Constructor code in Angular classes should not perform any angular operation … constructor is only for allocating object memory in Javascript. This is why we rather use “ngOnInit”, which is called (after constructor), when the object is ready to be used
- When the object is about to be destroyed, there is a symmetric method… ngOnDestroy, in interface OnDestroy. Try implementing it (but it won’t be used yet because your app is not changing page route yet)

Step 4: Modify main app.component.html component page

Modify your app.component.html page like below



From Chrome, you should see this:



Remark 1: the page in Chrome changes in real-time, while saving. (this is “auto-reload”)

Remark 2: Notice you should not remove the <router-outlet></router-outlet> element, it will be used later

Step 5: Add ng-bootstrap dependency

Interrupt your running “ng serve” (Control+C)

Add “ng-bootstrap” as a npm dependency only… This will not work (on purpose), we will see right after why

C:\td3> npm install –save @ng-bootstrap/ng-bootstrap

Check what are the source files that have been modified by this operation, using “git diff”

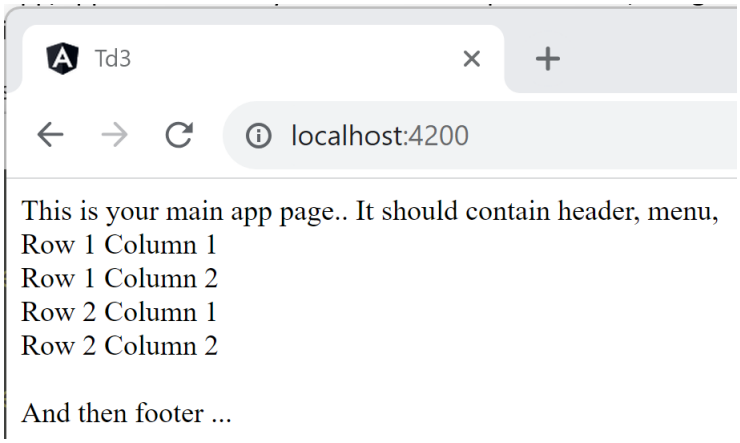
Then restart again your “ng serve”

Edit file “src/app/app.html” to test your first bootstrap html code, using nested divs with class=”container”, then class=”row”, class=”col-md-2” (or col-md-3, col-md-4, col-md-6..) .

Your html result should (but does not yet…) look like this:

Row1 colum 1	row 1 column 2
Row2 colum 1	row 2 column2

Unfortunately, the layout does not work (as expected) … all elements are above each other, not forming a layout with rows and columns



Redo the correct installation of “ng add”, instead of “npm install”, using

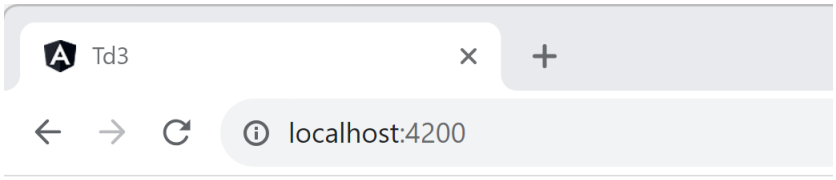
C:\td3> ng add @ng-bootstrap/ng-bootstrap

Check again what are the files that have been modified in git

See in file angular.json this added line ?

```
"node_modules/bootstrap/dist/css/bootstrap.min.css",
```

Now restart “ng serve”, and refresh your page, you should see this:



This is your main app page.. It should contain header, menu,

Row 1 Column 1	Row 1 Column 2
Row 2 Column 1	Row 2 Column 2

And then footer …

Redo the same (copy&paste below) using class=”container-fluid” instead of class=”container”

Find out the difference between class=”container” and class=”container-fluid”. Are they using same CSS width on the page ?

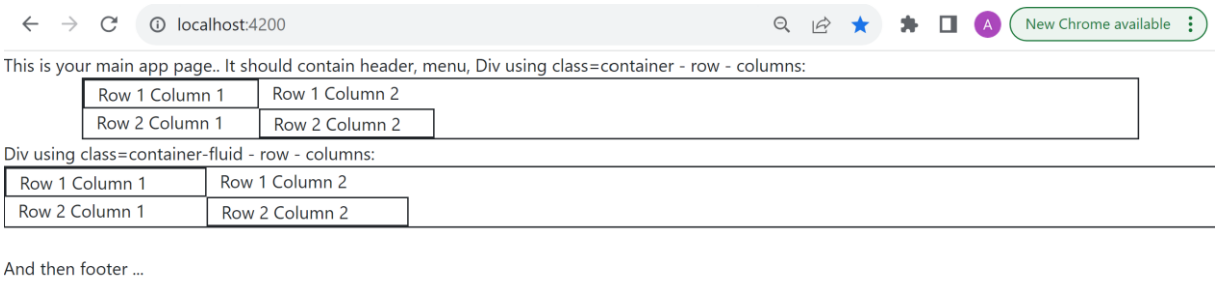
Step 6: change basic CSS, to add debugging border style

Edit your file src/style.css

```
.app-border {
    border-style: solid;
    border-width: 1pt;
}
```

Then add this extra class “app-border” to the top level div with class=“container” and also to the divs of “row1 column1” and to the div of “row 2 - column 2”

At this point, you should get:



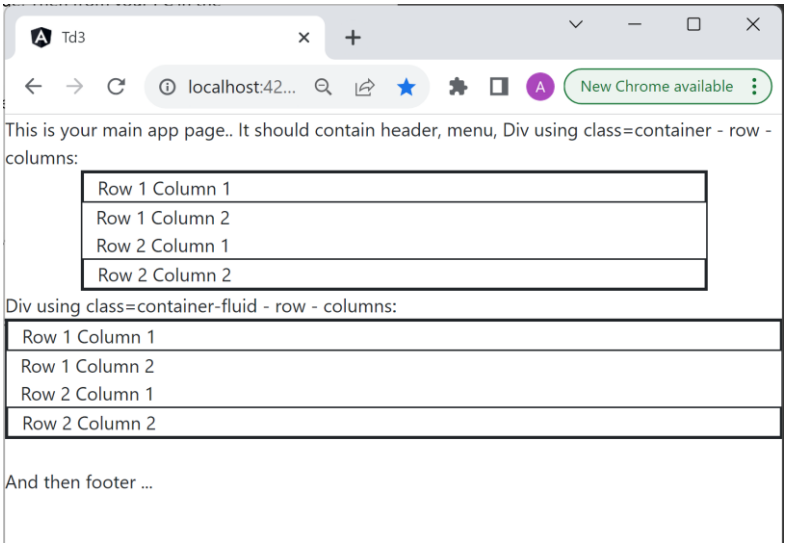
With these debugging “app-border” class, you can easily see you div size by forcing a temporary border to it.

[Optional] Step 7: Resize your Chrome window … see what happen on Small/Medium/Large size

When using your web page from your Portable telephone, you are in the “Small” device mode.

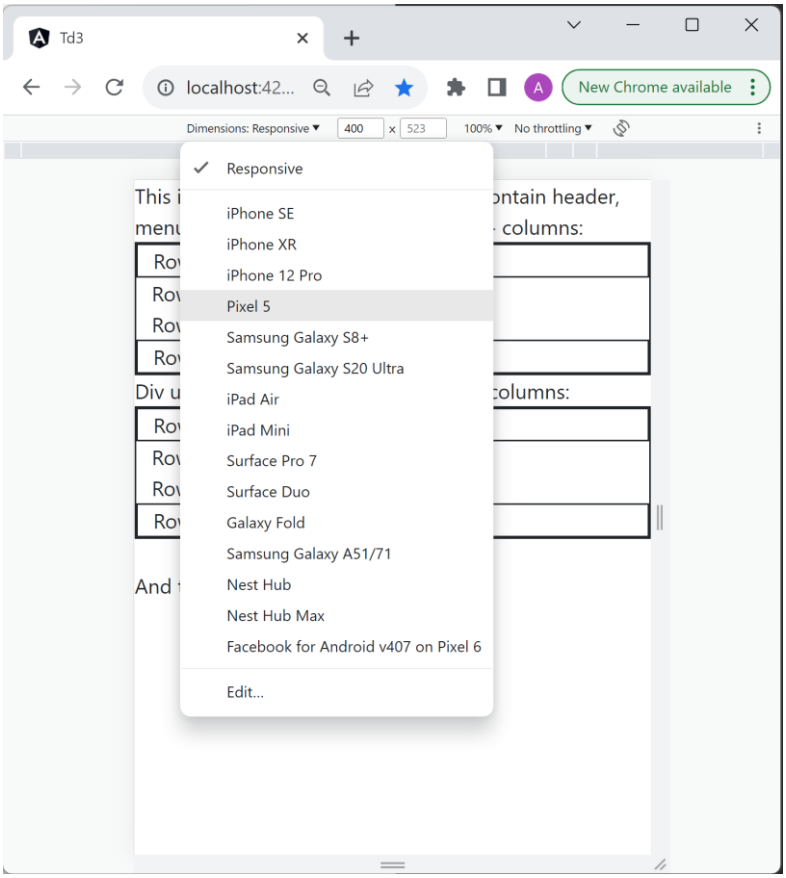
When using from your Tablet, you are in the “Medium” device mode. Then from your PC in the “Large” device mode.

When resizing to a small window, bootstrap does not any more use columns, but display all cells vertically. You should see this:



Play with alternate bootstrap classes for small / medium / large devices.

Chrome also has tool for debugging custom dimensions for well-known devices.



Step 8: Design a simple edit Form page using class=”container” / ”row” / “col-md-*”

Design a html page (edit directly in app.component.html) to serve as a basic form edit page, for object with following fields :

```
export interface LessonPackage {
    title: string;
    description: string;
    category: string;
    level: string;
    prerequisite: string[];
    tags: string[];
    copyright: string;
}
```

Your web page should be structured using

```
<div class=”container”>
  <div class=”row”>
```

```
        <div class="col-md-3"> …label of field to edit
        <div class="col-md-9"> …input field  to edit
    </div>
</div>
```

Step 10: move code into its own "@Component"

You have edited the main "app.component.html" web page, but this page should be the main page of your application, containing the "NavBar menu" (on top), the content outlet (in the middle), and "page footer" at bottom.

Generate a separate component page, using command line

```
ng generate component lesson-edit-form
```

(You can use short command line "ng g c" instead of "ng generate component")

Move your html code to newly created page "lesson-edit-form.component.html"

And restore your "app.component.html" file, by adding only

```
<router-outlet></router-outlet>
```

To continue viewing your form component, you could add

```
<lesson-edit-form></ lesson-edit-form>
```

But latter (next TD), it will be available as a URL "route".

Step 11: Design a simple search page

Generate another component

```
ng g c lesson-search-page
```

In this page "lesson-search-page.component.html", design a web search page, with the following fields:

- title
- description
- category
- level
- tags
- author
- lastModifiedDate

Contrarily to edit Form fields, all search criteria fields should have rounded corner style (cf bootstrap doc).

Search screen may be in “basic” mode, or in “advanced search mode”.

Also, contrarily to edit form,

- search field for numerical value should include a min-max range (or have corresponding operator “=”, “>=”, “<=”)
- search field for date value should include a start-end date range, using date-range calendar picker
- search field for string should be considered by default to use operator “contains ignore-case”, but could be changed to “like”, “case-incensitive like”, “startsWith”, “exact”, etc.

Step 12: Design a NavBar menu using Bootstrap nav & ng-bootstrap

Generate another component,

```
ng g c menu-nav-bar
```

add this component at the top of the main “app.component.html” page

```
< menu-nav-bar></ menu-nav-bar>
```

This menu may contains

- “Home“ item (top left icon)
- “File” sub-menu
    - “Explore lessons” menu-item
    - “Import…” menu-item
    - “New…” menu-item
    - “Export…” menu-item
- “Study Now” menu-item
- “Progresses Statistics” menu-item
- “Achievements” menu-item
- “Help” sub-menu
    - “Glossary” menu-item
    - “Documentation” menu-item
    - “About” menu-item
- “Profile” sub-menu (on top right corner)
    - “Logout” / “Login”
    - “Settings”