



Web Programming

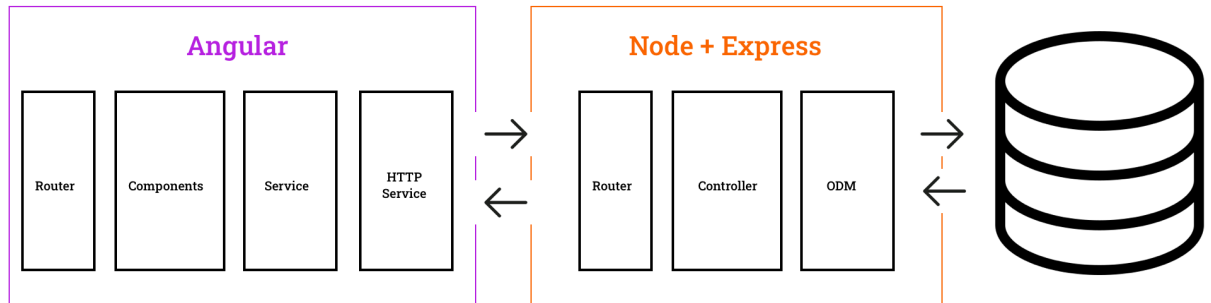
1nd Activity - Intro to Angular

2020/2021

Installations	2
Node 11.10.0	2
Angular CLI 8.3.19	2
Visual Studio Code	2
2. Single Page Application (SPA)	3
What is a Single Page Application?	3
Why use a SPA?	4
3. Creating an Angular application	5
Create a new app	5
Run angular server	5
4. Understanding Angular structure	5
5. Creating components	7
Manually create “post-create” component	7
Automatically generate a component	8
6. Creating input fields	9
Adding a text area and a save button to send values	9
Event Binding	9
String Interpolation	10
Property Binding	11
Showing user input data	11
By using a variable	11
Using two-way binding	12

7. Using Angular Material	13
Adding a form input field	13
8. Using Bootstrap	16
9. Exercise	19

1. Full-stack CRUD App Architecture



backend: Node.js Express exports REST APIs & interacts with MongoDB Database using Mongoose ODM.

frontend: Angular Client sends HTTP Requests and retrieves HTTP Responses using HTTPClient, consumes data on the components. Angular Router is used for navigating to pages.

2. Installations

- You need to install and run Powershell as admin and In Powershell, please run the command:

```
Set-ExecutionPolicy remoteSigned
```

- You can also try to:
Rename ng.ps1 from the directory C:\Users%username%\AppData\Roaming\npm\
then try clearing the npm cache at
C:\Users%username%\AppData\Roaming\npm-cache\

• If everything goes well, delete it.

Node 11.10.0

- <https://nodejs.org/en/download/>
- If you want to use a particular version (n) to install node, use the following url:
<https://github.com/tj/n>

Angular CLI 8.3.19

```
npm install -g @angular/cli@8.3.19
```

Visual Studio Code

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

- **Extensions:**
 - vscode-icons
 - prettier Now

2. Single Page Application (SPA)



A client-side(Browser) Framework which allows to build Single-Page-Applications(SPA)

Render UI with Dynamic Data

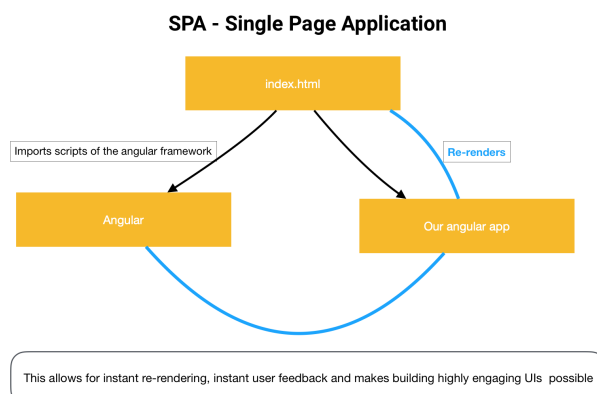
Handle User Input

Communicate with backend services

Provides a “Mobile App” - like User Experience

What is a Single Page Application?

In an Angular app, we'll have one root HTML file, a so-called index.html file and we will serve that from our Node server or from a different server. This HTML page basically includes some script imports that house our Angular app (the Angular framework and our own code) and we use this application to dynamically re-render what the user sees, without ever requesting a second page to be rendered by the server.



Why use a SPA?

Because by having this pattern, we never need to reload the page just because the user maybe clicked on a post and want to see the details. We can instead navigate to that page directly because we don't really leave the page, we just remove some elements from the DOM and add new elements and all of that is handled for us by the Angular framework.

And therefore we have a powerful way of immediately changing the page, maybe showing a spinner whilst we're fetching some data behind the scenes, so that list of posts which we probably still need to get but we will do that behind the scenes and this provides:

- highly-interactive,
- mobile-app-like feeling,
- very responsive
- fast web page where we never have to wait,
- where things always happen
- great user experience

and this is why we'll use JavaScript and Angular therefore for the entire front-end, for the entire user interface and we will use Node, Express and MongoDB/Postgres as a back-end to which we reach out behind the scenes to fetch and send data but the whole user interface is handled as one page only which is dynamically re-render all the time by Angular

3. Creating an Angular application

Create a new app

```
ng new app
```

When questioned please answer:

Angular Routing? No

CSS

Run angular server

```
ng serve
```

4. Understanding Angular structure

In the next page we'll have some angular files to look into to understand its way of functioning.

ng serve

Angular-cli builds the process which we should keep on running and which also starts this development server where we can preview our angular application. It loads our **index.html**

index.html
in the
browser

<app-root></app-root>

→ **app.component.html**

runtime.js contains all files (Component, module, services, pipes, etc.) convert into **js** and transform into minimised version.

polyfills.js is a script to allow older browsers to run modern javascript features

styles.js includes css files

main.js integrate all components needed

inline.js are webpack utilities to bootstrap the app and all its dependencies

**Injected in runtime
by webpack**

This is how angular-cli and the browser connects: **ng serve**, builds the process which we have up and running. It takes our code, bundles it up, adds all the angular logic from the angular framework to it and therefore creates a bunch of script files which we don't see because it's loaded in memory for development and injects these script file imports into the index.html

Then it detects the app root element and swaps it with the content of our component and content is not just the visual part, the html part, it would also be any logic we have in component.ts file.

app.module	<p>A file which is important for angular, it defines the features our angular application has</p> <p>Angular thinks in applications and applications are split up in modules or in this case in one module and that module defines the building blocks of our application</p> <p>Components are not the only but probably the most important building block of an angular application.</p> <p>The app module is where we declare the app component, this is registering it with angular, so now angular is aware of the app component but this alone would only allow us to use that component selector in other angular components, not in the index.html file, for that we need to add it to the bootstrap array too</p> <p>We typically only have one component in there because we only have one root component in a typical angular application and all other components would be somehow nested in that root component.</p> <p>So we get that bootstrap array which essentially tells angular that it should search the index.html file which is loaded or in general the page in which the angular app is loaded for the app component identified by its selector.</p>
main.ts	<p>starts the angular application this is executed first and that is simply how it's defined, you don't define that, this is what angular does or how the cli packages up our files in the end.</p> <p>The only thing it's important to know, is that it calls a bunch of code that actually puts angular working, and that the first module it call is AppModule.</p>
package.json	File that contains all dependencies of our app in dev and production modes
angular.json	File that contains the settings of our app, including stylesheets

5. Creating components

Manually create “post-create” component

- create a folder named *posts* inside **src/app** folder
- inside **posts** folder, create another folder named **post-create**
- create three files *post-create.component.html*, *post-create.component.ts* and *post-create.component.css* with the following contents

post-create.component.ts	post-create.component.html	post-create.component.css
<pre>import { Component } from '@angular/core'; @Component({ selector: 'app-postcreate', templateUrl: './post-create.component.html', styleUrls: ['./post-create.component.css'] }) export class PostCreateComponent {}</pre>	<pre><p>This is the post-create component</p></pre>	<pre>p { color: pink }</pre>

- We inform angular that we are creating a component by using this decorator **@Component** (<https://angular.io/guide/architecture-components>). This component receives an object with the meta-information about the component, his selector, his html and css localization.
- The selector must be unique and it usually starts with a **app-componentName**
- In order to angular understand what **@Component** is we must import it, from a library
- When we create a component we have to export it to be used.
- Then run:

ng serve

Can you see anything different? No. That's because you need to tell angular you have a new component in *app.module.ts*

d) add post-create component to app.module.ts

```
app.module.ts

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

import { AppComponent } from './app.component';
import { PostCreateComponent } from './posts/post-create/post-create.component';

@NgModule({
  declarations: [ AppComponent, PostCreateComponent ],
  imports: [ BrowserModule ],
  providers: [],
  bootstrap: [ AppComponent ]
})

export class AppModule { }
```

Then you need to call it inside our app.component.html in order to be executed,

```
app.component.html

<p>this is app.component</p>
<app-postcreate></app-postcreate>
```

Automatically generate a component

If we wanted to create a component via cli, we can write the following command in terminal.

```
ng g c posts/post-create
```

Where g is from generate and c from component. You will see a .spec.ts file that will serve for tests but we will now use it in the class. If you want to generate without spec file you should do the following:

```
ng g c posts/post-create --spec=false
```

6. Creating input fields

Adding a text area and a save button to send values

post-create.component.ts

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

@Component({
  selector: 'app-postcreate',
  templateUrl: './post-create.component.html',
  styleUrls: [ './post-create.component.css' ]
})

export class PostCreateComponent {}
```

post-create.component.html

```
<textarea rows="6"></textarea>
<hr>
<button>Save Post</button>
```

Event Binding

The button has no functionality as it is, to do something and to add some logic we need to add a click listener. To this feature we call [Event Binding](#).

post-create.component.ts

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

@Component({
  selector: 'app-postcreate',
  templateUrl: './post-create.component.html',
  styleUrls: [ './post-create.component.css' ]
})

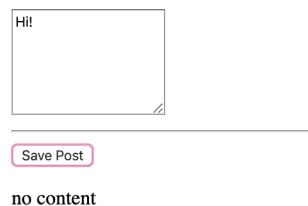
export class PostCreateComponent {
  onAddPost() {
    alert('click!');
  }
}
```

post-create.component.html

```
<textarea rows="6"></textarea>
<hr>
<button (click)=onAddPost()>Save
Post</button>
```

String Interpolation

Output content via [String Interpolation](#) : We add or use string interpolation feature with double curly braces, opening and closing and in between, and refer to something which is stored inside your post create component or your component belonging to the template in general, that something could be a method and then we would output whatever this method returns.



post-create.component.ts

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

@Component({
  selector: 'app-postcreate',
  templateUrl: './post-create.component.html',
  styleUrls: [ './post-create.component.css' ]
})

export class PostCreateComponent {

  newPost = '';

  onAddPost() {
    this.newPost = 'no content';
  }

}
```

post-create.component.html

```
<textarea rows="6"></textarea>
<hr>
<button (click)="onAddPost()">Save
Post</button>

<p>{{newPost}}</p>
```

Property Binding

Output via [Property Binding](#): the text area is an input element in html and as such it has a special property we can bind to, not an attribute, a property, html elements in the dom and in javascript are just javascript objects with a couple of properties, so variables belonging to that object which can read and write and one property is the value property.

post-create.component.html

```
<textarea rows="6" [value]="newPost"></textarea>
<hr>
<button (click)="onAddPost()">Save Post</button>
<p>{{newPost}}</p>
```

Showing user input data

By using a variable

post-create.component.ts

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

@Component({
  selector: 'app-postcreate',
  templateUrl: './post-create.component.html',
  styleUrls: [ './post-create.component.css' ]
})

export class PostCreateComponent {

  newPost = '';

  onAddPost(postInput: HTMLTextAreaElement) {
    this.newPost = postInput.value;
  }
}
```

post-create.component.html

```
<textarea rows="6" [value]="newPost" #postInput>
</textarea>
<hr>
<button (click)="onAddPost(postInput)">Save Post
</button>
<p>{{newPost}}</p>
```

Using two-way binding

post-create.component.ts

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

@Component({
  selector: 'app-postcreate',
  templateUrl: './post-create.component.html',
  styleUrls: [ './post-create.component.css' ]
})
export class PostCreateComponent {
  newPost = 'No content';
  enteredValue = '';

  onAddPost() {
    this.newPost = this.enteredValue;
  }
}
```

post-create.component.html

```
<textarea rows="6"
  [(ngModel)]="enteredValue"></textarea>
<hr>
<button (click)=onAddPost()>Save Post</button>

<p>{{newPost}}</p>
```

To use [(ngModel)] we have to import another module to our **app.module**

```
import { FormsModule } from '@angular/forms';
...
imports: [
  BrowserModule,
  FormsModule
]
```

7. Using [Angular Material](#)

We'll use angular material which is a package created by parts of the angular team which gives us a set of pre-built angular components. It's not just a styling package. It's not like bootstrap. It's actually an angular package which ships with a couple of angular components which we can drop into our application. It has a bunch of pre-built components that should give us everything we need to build a nice application to include things like a header, include things like our buttons, our inputs.

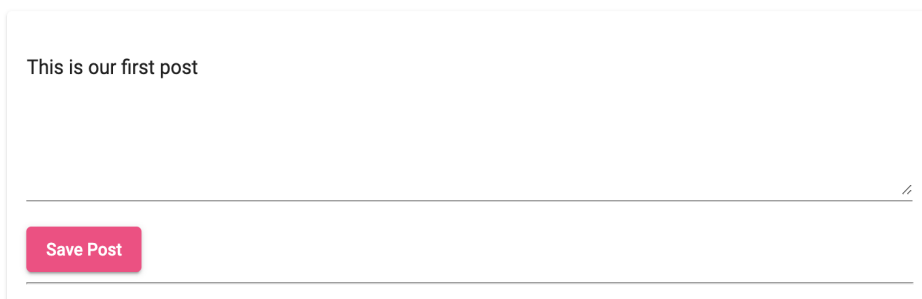
To install we may use a new feature of angular cli, the ng add command inside app folder.

```
ng add @angular/material
```

This package inserts some lines in package.json about new dependencies of our project and some lines in angular.json to setup a main css, regarding the style we choose in angular material installation. It also added a line in appModule to insert some animations this package uses.

Adding a form input field

Our first App



This is our first post

Go to <https://material.angular.io/components/input/examples> and in examples, analyse the code of a basic input field, to copy only the following code

post-create.component.html

```
<mat-card>
  <mat-form-field>
    <textarea
      matInput
      rows=6
      [(ngModel)]="enteredValue">
    </textarea>
  </mat-form-field>
  <button mat-raised-button
    color="accent"
    (click)="onAddPost()">Save Post</button>
  <hr>
</mat-card>
<p>{{newPost}}</p>
```

post-create.component.css

```
mat-card {
  width: 80%;
  margin: auto;
}

mat-form-field,
textarea {
  width: 100%;
}
```

We'll use a mat-card to create a kind of a card with a textarea and a button like in the following image. Please change your post-create.component.css with the code above.

In order to use these angular features, we have to import some modules into our app.module.ts.

app.module.ts

```
import { BrowserModule } from "@angular/platform-browser";
import { NgModule } from "@angular/core";
import { FormsModule } from "@angular/forms";
import { MatCardModule } from "@angular/material/card";
import { MatInputModule } from "@angular/material/input";
import { MatButtonModule } from "@angular/material/button";

import { AppComponent } from "./app.component";
import { PostCreateComponent } from "../posts/post-create/post-create.component";
import { NoopAnimationsModule } from "@angular/platform-browser/animations";

@NgModule({
  declarations: [
    AppComponent,
    PostCreateComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
```



```
    NoopAnimationsModule,  
    MatCardModule,  
    MatInputModule,  
    MatButtonModule  
  ],  
  providers: [],  
  bootstrap: [AppComponent]  
})  
export class AppModule {}
```

8. Using Bootstrap

Bootstrap 4 is the latest version of Bootstrap. Bootstrap is a free front-end framework for quick and simple web development. Bootstrap helps to create responsive designs in less time. Bootstrap contains many HTML and CSS based templates or components like typography, buttons, forms, tables, image carousels, navigation, modals, pagination, spinners, alerts, icons, dropdowns, cards and many more.

To install all files, run the following commands in terminal:

```
npm install jquery --save
npm install bootstrap --save
npm install popper.js --save
```

Then open **angular.json** file and add in styles and in scripts the following bold lines:

```
"styles": [
  "src/styles.css",
  "./node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": [
  "./node_modules/jquery/dist/jquery.min.js",
  "./node_modules/popper.js/dist/umd/popper.min.js",
  "./node_modules/bootstrap/dist/js/bootstrap.min.js"
]
```

In order to use bootstrap I created a new component named post-create-bootstrap, just for showing how it works, as we'll be using angular material in our examples.

```
ng g c post-create-bootstrap
```

Go to the bootstrap page <https://getbootstrap.com> and search for forms. There you will see many examples. We'll use the following

post-create-bootstrap.component.ts

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

@Component({
  selector: 'app-post-create-bootstrap',
  templateUrl: './post-create-bootstrap.component.html',
  styleUrls: [ './post-create-bootstrap.component.css' ]
})
export class PostCreateBootstrapComponent {
  newEnteredValue = '';
  newPost = '';

  onAddedPost() {
    this.newPost = this.newEnteredValue;
  }
}
```

Try to create your form in bootstrap, using the online reference (bootstrap page)

post-create-bootstrap.component.html

```
<div class="container">
<form>
  <div class="form-group">
    <textarea [(ngModel)]="newEnteredValue" class="form-control"
name="exampleFormControlTextarea1" rows="3"></textarea>
  </div>
  <button type="button" class="btn btn-light"
(click)="onAddedPost()">Save</button>
</form>
  {{newPost}}
</div>
```

Also made some changes in app.component.html to have a page look like this. Can you do the same?

Our form with angular materials

Save Post

No content

Our form with bootstrap

Testing bootstrap component

Save

Testing bootstrap component

9. Exercise

Using Angular Material or bootstrap (you choose) create a component header inside the **src** folder, and add a navbar as in the following page.

If you want you can create automatically your component, if you run the following command in your root folder (app):

```
ng g c header --spec=false
```

