CS 572 Modern Web Applications

Najeeb Najeeb, PhD (najeeb@miu.edu)

V3.1.0

JavaScript Full Stack Development

- MongoDB
  - NoSQL database (document store)
  - Stores JSON documents
- Express
  - JavaScript web framework
  - On top of Node
- Angular
  - JavaScript UI framework
  - Single Page Applications
- Node
  - JavaScript server-side platform
  - Single threaded, fast and scalable

Roadmap and Outcomes

- Node.js: write asynchronous (non-blocking) code. Understand node platform to start a project.
- Express: setup express and get requests and send back responses. REST API.
- MongoDB: what NoSQL DB looks like. Full API interacting with DB.
- Angular: Investigate Angular and the architecture of an Angular application. Build a single-page application.
- MEAN application: Learn by example. We will create a MEAN Games application.

TypeScript

1. How to install Angular and create an application?
2. What is TypeScript and how to use it?
3. What are the parts of an Angular application, how do they relate?

Introduction to Angular CLI

Angular

Install CLI

Create App

Run App

Inspect App

Install Angular CLI (Command Line Interface)

```
npm i --g @angular/cli
```

Create angular-app folder.

Angular

Install CLI

Create App

Run App

Inspect App

Angular application creation options

Skip initializing a git repo, most of the time you already have git setup by the time you reach the angular app
--skip-git=true

Skip initializing test files
--skip-tests=true

Create the angular application in this folder no the current folder
--directory <directory_name>

Accept default setting, and do not ask me about each detail when creating boilerplate code

--defaults= true

Create our first Angular application using CLI
ng new first-app --skip-git=true --skip-tests=true –-directory public\first-app –-defaults=true

Angular

Install CLI

Create App

Run App

Inspect App

To run the boilerplate code created by CLI

npm start

This message indicates the application is ready

√ Compiled successfully.

What is the port that was used?

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

Angular

Install CLI

Create App

Run App

Inspect App

View the project on the browser
localhost:4200

Inspect the html file on the browser

```
...
<body>
   <app-root ...
...
```

Open the project folder in VSCode, observe index.html

```
...
   <title>TestApp</title>
...
<body>
   <app-root></app-root>
...
```

The project folders and files. You may observe that there are a lot of files, you may not know some of them. By the end of today you should understand what most (if not all) of what these files are or are used for.

TypeScript

1. How to install Angular and creating an application?
2. What is TypeScript and how to use it?
3. What are the parts of an Angular application, how do they relate?

TypeScript

TS

Intro

Types

Classes

Enum

Install

TypeScript (TS) is a superset of JavaScript (JS)

TypeScript playground, a website to learn, experiment, and share TS

typescriptlang.org/play/

TS brings OO JS: Interfaces, Inheritance, and encapsulation

TS

Intro

Types

Classes

Enum

Install

Use typescriptlang.org/play/

TypeScript playground, a website to learn, experiment, and share TS

```
let x:number= 10;
x="Najeeb"; // you get an error
console.lob(x);
```

Types

boolean number string

[], number[], Array<number>, [string, number]

unknown any enum

void null undefined

never Object

TS

Intro

Types

Classes

Enum

Install

Create a Student class

```
class Student {
    #name: string;
    private _gpa: number;
    get gpa(): number {
        return this._gpa;
    }
    set gpa(gpa:number): void {
        this._gpa= gpa;
    }
    constructor(name:  string, gpa: number) {|
        this.#name= name;
        this.gpa= gpa;
    }
}
```

Types

```
let jack: Student= new Student("Jack", 3.0);
console.log("Student GPA is", jack.gpa);
```

TS

Intro

Types

Classes

Enum

Install

Whatever is available in JS is also available in TS

```
let age= prompt("Please enter your age");
```

Enum

```
enum Days {
MONDAY= 3,
TUESDAY,
WEDNESDAY,
THURSDAY,
FRIDAY,
SATURDAY= 1,
SUNDAY
}
```

```
let day: Days= Days.Monday;
console.log(day);
console.log(Days[4]);
```

How to print the Enum String of a given enum variable?

```
console.log(Days[day]);
```

TS

Intro

Types

Classes

Enum

Install

To install and run TS locally

```
npm i --g typescript
```

Create a local ts file (test.ts)

```
let name: string= "Jack";
console.log("Hello", name);
```

Transpile the TS file to JS

```
tsc test.ts
```

Run the generated JS file (test.js)

```
node test.js
```

Configuration

tsconfig.json

Create a TS configuration file tsconfig.json

```
{
   "compilerOptions" : {
      "target": "ES2015",
      "outDir": "output"
   }
}
```

Transpile using the configuration file

```
tsc
```

Observe the output files in the output folder

test.ts

Run the generated JS file (test.js)

```
node ./output/test.js
```

Decorators

Decorators

Config

Class

Run Code

Decoration

Add decorators to your JS

Option 1 to declare them as a command line flag

```
tsc --experimentalDecorators ./test.ts
```

Option 2 add this to tsconfig.json

```
{
   "compilerOptions" : {
      "target": "ES2015",
      "experimentalDecorators": true,
      "emitDecoratorMetadata": true,
      "outDir": "output"
   }
}
```

Decorators
Config
Class
Run Code
Decoration

Create a Student class with a few properties Student.ts

```
export class Student {
    id: number;    // default is public
    private name: string;
    #gpa: number;  //TS V3.8 this is private
    set gpa(gpa) {this.#gpa= gpa;}
    get gpa() {return this.#gpa;}
    getName(): string { return this.name;}
    constructor(id: number, name: string, gpa: number) {
        this.id= id;
        this.name= name;
        this.#gpa= gpa;
    }
}
```

Decorators
Config
Class
Run Code
Decoration

Create a file to use the class Main.ts

```
import { Student } from "./Student.js";
let jack = new Student(123, "Jack", 3.0);
console.log(jack.id);
console.log(jack.name);
console.log(jack.gpa);
```

For this code to run you need to update package.json

```
{
    "type": "module"
}
```

To run the file

```
tsc
node ./output/Main.js
```

Decorators

Config

Class

Run Code

Decoration

Create a decorator file to use MyDecorator.ts

```
export function Token(token: {course: string, canProgram:
boolean}) {
    return function (constructor: Function){
        constructor.prototype.course= token.course;
        constructor.prototype.canProgram= token.canProgra
m;
        if (token.canProgram){
            constructor.prototype.program= function(){
                console.log("I am programming...");
            };
        }
    };
}
```

Decorators

Config

Class

Run Code

Decoration

Use the decorator, create child class

```
import { Student } from "./Student.js";
import { Token } from "./MyDecorator.js";
@Token({course: "CS572", canProgram: true})
export class DE_Student extends Student {
}
```

Run decorator in Main.ts

```
import { DE_Student } from "./DE_Student.js";
let jack:DE_Student= new DE_Student("Jack", 3.5);
console.log(jack.getName(), "is enrolled in", jack["course"]));
console.log(jack.getName(), "can you program?", jack["canProgram"]));
if (jack["canProgram"]){
    console.log(jack.getName(),"please program.");
    jack["program"]();
} else {
    console.log("jack.getName(), "don't worry you will learn after this course.");
}
console.log("All fields in DE_Student are");
for (const key in jack) {
    console.log(key);
}
```

TypeScript

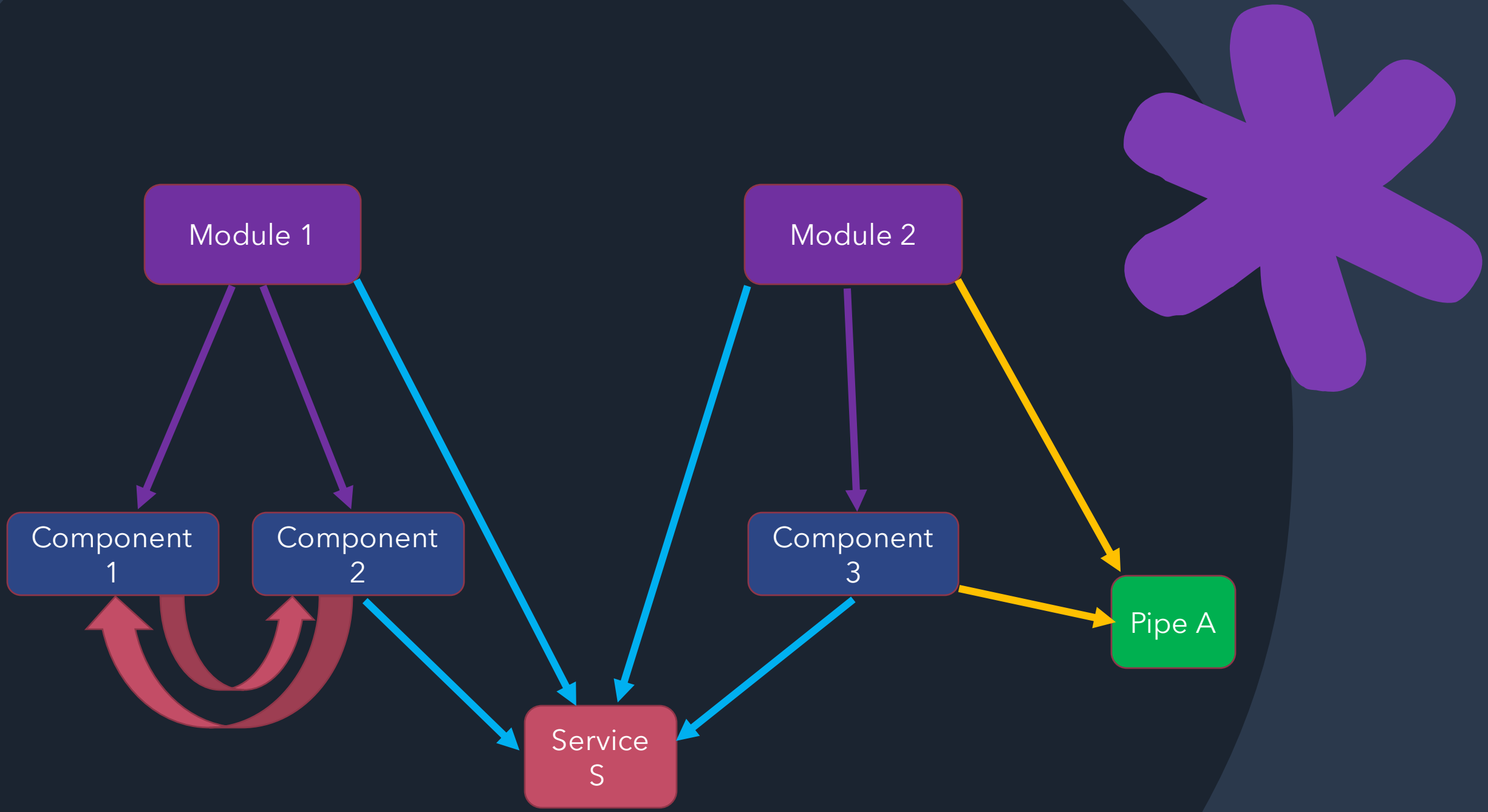
1. How to install Angular and creating an application?
2. What is TypeScript and how to use it?
3. What are the parts of an Angular application, how do they relate?

Modules & Components

Structure

Module
Component
Service
Pipe

- An application consists of one or more Modules
- A Module consists of one or more Components
- Services are used to communicate with external resources or provide a data to other Components
- Pipes are used by the template framework to transform data from one representation to another


Module 1
Module 2
Component 1
Component 2
Component 3
Pipe A
Service S

Learning Some Angular UI

Angular

Expression

Concatenation

List

Button

Menu

Delete the contents of app.component.html file and replace it with this

```
1 + 2 = {{ 1 + 2 }}
```

Check your browser.

Angular

Expression

Concatenation

List

Button

Menu

Try

...

```
{{"Hello " + "World!"}}
```

Check your browser.

Angular

Expression

Concatenation

List

Button

Menu

Create an array in the component app.component.ts

```
...
  people:string[]= ["Jack", "John", "Jill", "Jack"];
...
```

Add this in app.component.html

```
<ul>
<li *ngFor="let name of people"> {{ name }} </li>
</ul>
```

Check your browser.

Angular

Expression

Concatenation

List

Button

Menu

Create a function in the component app.component.ts

```
...
   onClickBtn(){
    this.title= "Button Clicked";
   }
...
```

Add this in app.component.html

```
<button (click)="onClickBtn()">Click</button>
{{title}}
```

Check your browser.

Angular

Expression

Concatenation

List

Button

Menu

Create an array of JSON objects in the component app.component.ts

```
...
   students= [{name: "Jack", course: "MWA", gpa: 3.0},
              {name: "Jill", course: "MPP", gpa: 3.4},
              {name: "Jim", course: "MWA", gpa: 2.8}];
...
```

Add this in app.component.html

```
<select>
   <option *ngFor="let s of students">
      {{s.name}}
   </option>
</select>
```

Check your browser.

Pipes

Pipes

Currency

Date

JSON

Strings

Numbers

Default currency USD add this in app.component.html

```
...
   {{123 | currency }}
...
```

Try this for other currency (on Windows use keys Win+;

```
{{123 | currency:"₮" }}
```

Check your browser.

Pipes
Currency
Date
JSON
Strings
Numbers

Create a date field in the component app.component.ts

```
...
   today= new Date();
...
```

Add this in app.component.html

```
{{today | date}}
```

Try other date options

```
{{today | date:"short"}}
{{today | date:"medium"}}
{{today | date:"long"}}
{{today | date:"MM-dd-yyyy (hh:mm:ss)"}}
```

Check your browser.

Pipes
Currency
Date
JSON
Strings
Numbers

Display JSON object or array of JSON objects
app.component.html

```
{{students | json}}
```

Check your browser.

Pipes

Currency

Date

JSON

Strings

Numbers

String manipulations app.component.html

```
{{ "Maharishi Internation University" | lowercase}}
{{ "Maharishi Internation University" | uppercase}}
```

Check your browser.

Pipes
Currency
Date
JSON
Strings
Numbers

Number formatting app.component.html

```
{{ 123.456789 | number:"1.0-5"}}
{{ 123.456789 | number:"4.0-2"}}
{{ 123.456789 | number:"4.0-20"}}
{{ 123.456789 | number:"4.10-20"}}
```

Check your browser.