Full Stack Web Applications II

CSE187

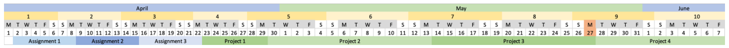
Spring 2024

Introduction & Type Safety

Baskin Engineering

Notices

- **Administration** due 23:59 **Friday**
- Office Hours start today
  - Schedule in Google Drive
  - My In Person OH @ 09:30 tomorrow will be rescheduled for later in the day via Zoom; watch for a Canvas announcement



April																
1						2							3			
M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	Assignment 1							Assignment 2							Assignment 3	

													May			
				4						5						
T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S
18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4
					Project 1							Project 2				

6							7								8	
S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
									Project 3							

										June						
				9							10					
W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F
22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7
						Project 4										

Resources

- People
  - Instructor
    - Dr Harrison dcharris@ucsc.edu
  - TA
    - Esmaeil smirvaki@ucsc.edu
- Canvas
- Slack
- Google Drive

Today’s Lecture

- Administration
  - Course Outline
  - Assessment
  - Grading
  - Academic Integrity
- Type Safety
  - Typescript
  - tsoa for RESTful APIs
  - tsoa Security / Authentication
- The TypeScript Database Book Example
- The TypeScript Authenticated Book Example
- Assignment 1 - Introduction

Administration

Course Outline

- **General Principle**
  - Self discovery
- **Individual Assignments**
  - Type Safety
  - Non-RESTful APIs
  - Alternative UI Frameworks
- **Group Project**
  - Working as a team
  - Deployment
  - API Keys
  - Microservices
  - Security

Assessment

Administration		**(No marks, but mandatory)**
Individual Assignments	**30%**	**(3 x 10%)**
Group Project	**60%**	**(1 x 10%, 2 x 15%, 1 x 20%)**
Group Presentation	**10%**	**(Week 10)**

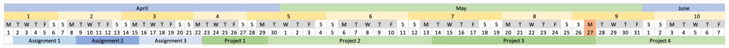
NOTE: Administration task is mandatory, and you must pass each component (individual assignments, group project, group presentation) to pass the class. E.g., doing well on the group project and presentation but submitting too many poor assignments will see you fail the class.

HOWEVER: Doing less than perfectly on one assignment but well on the others, is fine so long as the aggregate grade for the assignments is a pass.

Extra Credit: Group who's completed project is voted the best by classmates.

Assignments, Project, Presentation

- **Individual Assignments**
  - Single new concept exercises
  - Less hand-holding than in CSE186
  - Released Tuesdays, due 23:59 Mondays
- **Group Project**
  - Multi concept, realistic, deployed full stack Web App
  - Staged delivery via multiple check points
  - Demonstration of project to whole class in Week 10
  - **Equality of contribution must be demonstrated**
- **Group Presentation**
  - Week 10, in this room, all team members must take part



April																				
1						2							3							
M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	Assignment 1							Assignment 2							Assignment 3					

April									May											
4						5							6						
M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S
22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11
Assignment 3	Project 1							Project 2											

May																				June						
7								8						9							10					
S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F
12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7
Project 2		Project 3														Project 4										

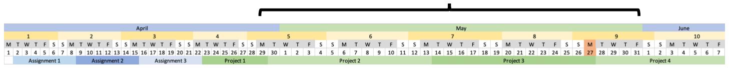
Grade Bands

Grade	Minimum	Characterisation
A+	**95**	**Outstanding**
A	**90**	**Excellent**
A-	**85**	**Excellent in most respects**
B+	**80**	**Very good**
B	**75**	**Good**
B-	**70**	**Good overall, but some weaknesses**
C+	**65**	**Satisfactory to good**
C	**60**	**Satisfactory**
* C-	55	Adequate evidence of learning
* D	50	Some evidence of learning
F		Below the required standard; fail

* Pass, but cannot be used to satisfy a major requirement or a general education requirement, and cannot satisfy a prerequisite for another course. https://registrar.ucsc.edu/navigator/section4/performance/letter-grades.html

Project Meetings Weeks 5,6,7,8,9

- **Mandatory**
  - **All team members must be present**
  - 15 minutes once a week with instructor during lecture slot
  - 30 minutes once a week with TA at mutually agreed time
- **Recommended**
  - **At least half team members must be present**
  - Once a week with instructor during OH
  - Twice a week with TA during OH
- Ideal Schedule
  - Whole team meet with TA or Instructor every weekday



April																														May																														June						
1							2							3							4							5							6							7							8							9							10				
M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7
	Assignment 1							Assignment 2							Assignment 3							Project 1							Project 2														Project 3														Project 4										

Academic Integrity

- Assignments are individual tasks, but…
  - We encourage you to work, study, and think together
  - But ensure the final product is yours and yours alone
- All course work is submitted electronically
  - No late submissions allowed
  - Checked for collaboration & plagiarism
    - If you copy another student, you fail the assignment ☹
    - If you copy a solution found on the web, you fail the assignment ☹
    - In both cases you may get kicked off the class, or even out of school ☹ ☹
- If you post your code on-line (GitHub etc.)
  - You fail the class
  - You are also in breach of my copyright
  - If you want to show it to prospective employers:
    - Make repo private and give them access

Use of Code Generation Tools

GitHub Copilot

ChatGPT

Amazon CodeWhisperer

Google **Bard**

Same as using code found on-line:

If you give credit to where you got the code from, no sanction but no marks

If you fail to give credit to the tool, no marks *and* an academic integrity violation

Code Plagiarism

```
/**
 * Returns the number of lines in the file at path
 * FNAME which is assumed to exist.
 */
int lines_in_file(char *fname) {
    FILE *fp = fopen(fname, "r");
    int lines = 0;
    while (!feof(fp)) {
        if (fgetc(fp) == '\n') {
            lines++;
        }
    }
    fclose(fp);
    return lines;
}
```

FAIL

Code Plagiarism

```
/**
 * Returns the number of lines in the file at path
 * FNAME if it exists and can be read, -1 otherwise.
 */
int linesInFile(char *fname) {
    FILE *fp = fopen(fname, "r");
    int lines = -1;
    if (fp) {
        lines = 0;
        while (!feof(fp)) {
            if (fgetc(fp) == '\n') {
                lines++;
            }
        }
        fclose(fp);
    }
    return lines;
}
```

FAIL

Code Plagiarism

```
/**
 * Returns the number of lines in the file at path
 * FNAME if it exists and can be read, -1 otherwise.
 *
 * https://chat.openai.com/c/412dd8ee-4057-43b3-a2eb-6c9d32ef6072
 */
int linesInFile(char *fname) {
    FILE *file = fopen(fname, "r");
    if (file == NULL) {
        return -1;
    }
    int count = 0;
    int ch;
    while ((ch = fgetc(file)) != EOF) {
        if (ch == '\n') {
            count++;
        }
    }
    fclose(file);
    return count;
}
```

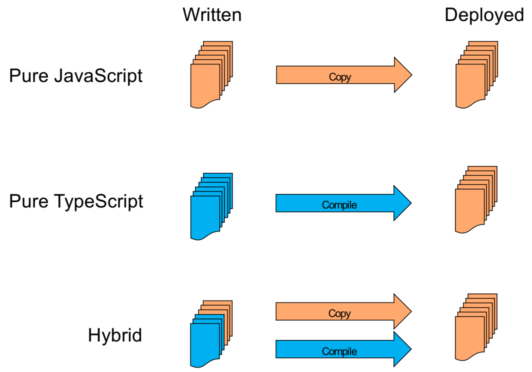
ACCEPTED

TypeScript

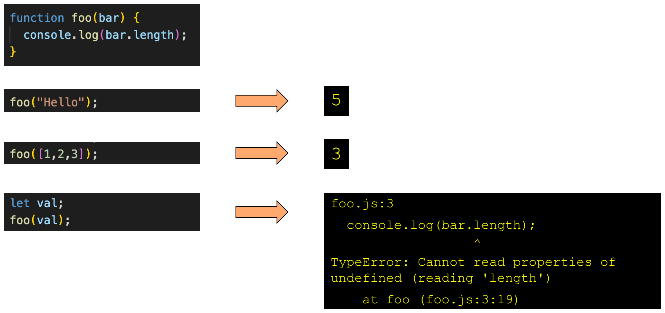
TypeScript

- A superset of JavaScript => All JavaScript is valid TypeScript
- Adds some new features intended to make your life easier
- Adds a consistency check (compilation) step between the code you write and the code that gets executed
- Adds additional syntax to tell the compiler what you're trying to do so it can help you
- Aims to make you make your code more consistent

TypeScript & JavaScript



JavaScript Problems



```
function foo(bar) {
  console.log(bar.length);
}
```

```
foo("Hello");
```

```
5
```

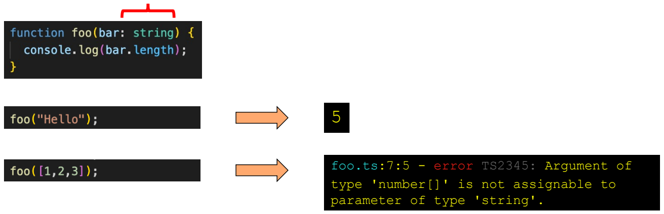
```
foo([1,2,3]);
```

```
3
```

```
let val;
foo(val);
```

```
foo.js:3
  console.log(bar.length);
                  ^
TypeError: Cannot read properties of
undefined (reading 'length')
    at foo (foo.js:3:19)
```

TypeScript Solutions



```
function foo(bar: string) {
  console.log(bar.length);
}
```

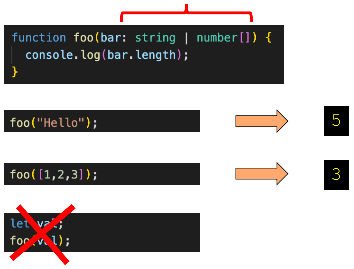
```
foo("Hello");
```

```
5
```

```
foo([1,2,3]);
```

```
foo.ts:7:5 - error TS2345: Argument of
type 'number[]' is not assignable to
parameter of type 'string'.
```

TypeScript Solutions



```
function foo(bar: string | number[]) {
  console.log(bar.length);
}
```

```
foo("Hello");
```

```
5
```

```
foo([1,2,3]);
```

```
3
```

```
let val;
foo(val);
```

Type Annotation

```
let variableName: TypeName;
const CONSTANT_NAME: TypeName;
function functionName(
   param1: Type1, param2: Type2): ReturnType {}


let age: number = 20;
const ALPHABET: string = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
function divides(
   divisor: number, dividend: number): boolean {}
```

Type Inference

- Equivalent

```
let foo = 'Hello';           // inferred type is string
let foo: string = 'Hello';
```

- Not Equivalent

```
let foo = 1;                 // inferred type is number
let foo: string = 1;         // throws an error, 1 is
                             // not a string
```

Everyday Types

- Primitive
  - number
  - string
  - boolean
  - null
  - undefined
- Arrays
  - E.g. <type>[] or Array<type>
- Non-Primitive
  - object
  - function
  - any // a “special” type that can represent anything ☺ ☹

Interfaces

- Describe the structure of objects
- Cannot be instantiated
- Have no functionality

```
interface InterfaceName {
  property1: Type1
  property2: Type2
}
```

```
interface Point {
  x: number
  y: number
}
```

Function Annotations

- Object structure elements can be functions

```
interface WithFunction {
  func: (param1: Type1, param2: Type2) => ReturnType
}
```

```
interface Button {
  title: string
  onPress: () => void      // special type to indicate
}                          // nothing returned
```

Map Annotations

- Keyed on either string or number

```
interface WithKey {
  [name: KeyType]: ValueType
}
```

```
interface Phonebook {
  [name: string]: string
}
```

Using TypeScript in node.js Projects

- For every package being used, need to install the matching types packages as dev dependencies in `package.json`

```
"dependencies": {
   ...
   "pg": "*",        <= Using PostgreSQL
   ...
 },
 "devDependencies": {
   ...
   "@types/pg": "*", <= So we need PostgreSQL Types
   ...
 },
```

- Also need TypeScript configuration in `tsconfig.json`

```
{
   "compilerOptions": {
     "target": "es2022",
     "strict": true,
     ...
   },
 }
```

tsoa

(pronounced “so-uh”)

Type Definitions in OpenAPI

- OpenAPI Schema in YAML
- OpenAPISchemaValidator to check in/out bound objects
- Tests check the code before deployment
- Problems?
  - Type definitions from OpenAPI schema are not visible/enforceable in JavaScript ☹
  - Change the structure of an object in one place you must remember to change it in the other ☹
- Possible solutions?
  - Generate code from the OpenAPI Schema
  - Define types in code and generate the OpenAPI Schema ⇦

tsoa (“so-uh”)

- TypeScript **Models** (data types) and **Controllers** (API end points) are the single source of truth for your RESTful API
- A valid OpenAPI Schema is generated from TypeScript Models and Controllers
- Middleware routes also generated (we’ll use Express)
- Configuration in `tsoa.json`

Resources

TypeScript	https://www.typescriptlang.org/
tsoa	https://tsoa-community.github.io/docs/

TypeScript Database Book Example

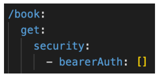
tsoa Security / Authentication

- From code annotations:
  - Generate security settings in OpenAPI Schema
  - Generate code to check authenticated user can perform requested operation
- Negates the need for:
  - Separately maintained OpenAPI Schema
  - Custom code to check “role” of authenticated user before allowing and operation to be performed

tsoa Security / Authentication

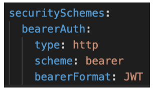
openapi.yaml

paths:



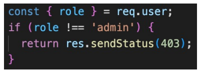
```
/book:
  get:
    security:
      - bearerAuth: []
```

components:



```
securitySchemes:
  bearerAuth:
    type: http
    scheme: bearer
    bearerFormat: JWT
```

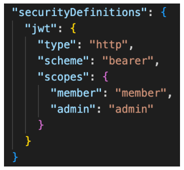
Route Handlers:



```
const { role } = req.user;
if (role !== 'admin') {
  return res.sendStatus(403);
}
```

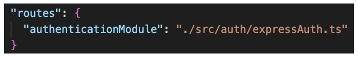
tsoa.json

spec:



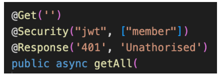
```
"securityDefinitions": {
  "jwt": {
    "type": "http",
    "scheme": "bearer",
    "scopes": {
      "member": "member",
      "admin": "admin"
    }
  }
}
```

routes:



```
"routes": {
  "authenticationModule": "./src/auth/expressAuth.ts"
}
```

Controllers:



```
@Get('')
@Security("jwt", ["member"])
@Response('401', 'Unathorised')
public async getAll(
```

TypeScript Authenticated Book Example

Assignment 1

Assignment 1 - Introduction

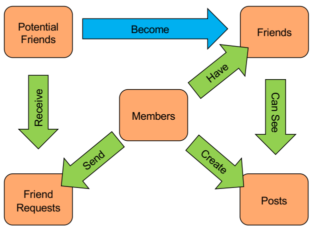
- Authenticated TypeScript / tsoa RESTful API
- facebook style data model:
  - **Member** A login account that creates Posts
  - **Post** Text content and optional image
    Visible to poster and all friends
- Plus, some meta-data concepts:
  - **Friend** A Member who can see your posts
    You can also see their posts
  - **Request** How you ask another member to be your friend
    Can be accepted, declined or ignored

Assignment 1 - Database I

- How may entities?
- How many tables?
- Relationship between them?

🟩 One to Many

🟦 One to One



This is starting to get complicated 🤔

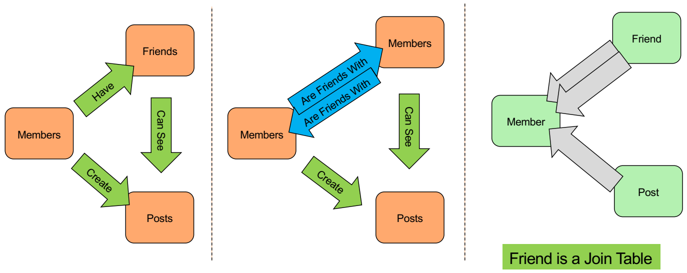
Assignment 1 - Database II

- How may entities?
- How many tables?
- Relationship between them?

🟩 One to Many

🟦 Many to Many

⬜ Foreign Key



Friend is a Join Table

Request is simply a flag on a Friend

Assignment 1 - Testing

- **Use the API to test the API**
- Starting data for all tests is a single admin user
  - Database schema exists, but no other data
- All tests must create everything they need

Assignment 1 - Getting started

- Suggested Initial Implementation order*:
  - **POST /api/v0/login**
    - Anna Admin can now login
    - Use `AuthController` from the TypeScript Authenticated Book Example
  - **POST /api/v0/member**
    - Anna can create new Members that can login
  - **POST /api/v0/post**
    - Members can create Posts
  - **POST /api/v0/friend/{memberId}**
    - Members can request other Members to be their Friends
  - **PUT /api/v0/request/{memberId}**
    - Members can accept Friend requests
  - Etc.

* Based on sequencing of tests in `test/basic.test.ts`

Assignment 1 - Grading

- Basic 60%
  - Pass Provided Basic Tests (40%)
  - Perfect Code Coverage (15%)
  - No Lint Errors/Warnings (5%)
- Advanced 40%
  - Pass known-good tests for all API endpoints (40%)
    - Approximately twice as many of these as there are Basic Tests
- Marks deducted for:
  - Peculiar / overly complex database schemas
  - Poorly Factored Controllers (e.g. everything in one uber-controller)
  - Database access inside Controllers (delegate to Services)

Next Lecture

- Assignment 1 - Workshop