Recitation #4

17-356/17-766

TAs

- **Mehul Agarwal**
  - email: mehula@andrew.cmu.edu
  - office hours: Wednesdays 3:30 - 4:30 PM

- **Rohit Shreenivas**
  - email: rshreeni@andrew.cmu.edu
  - office hours: Fridays 5 - 6 PM

Full-stack Development

Different levels of the stack:

- **Backend**
- Frontend → Today’s topic
- Database
- Deployment
- Testing

  and more

USING TYPESCRIPT WITH REACT

What and Why Typescript?

- TypeScript is a strongly typed, object-oriented, compiled programming language that builds on JavaScript
- Provides an easy way to structure your objects and enforce type validation on them
- Provides great tooling in your IDEs which help catch bugs that potentially would have otherwise been caught in deployment
- 0 learning curve if you already know JS

Install TS

- To install TS globally:
  - *npm i -g typescript*
  - *tsc --version*
- To create a react-typescript project:
  - *npx create-react-app . → plain react app*
  - *npx create-react-app <project-name> --template typescript*

tsconfig.json

- The presence of a tsconfig.json file in a directory indicates that the directory is the root of a TypeScript project.
- The tsconfig.json file specifies the root files and the compiler options required to compile the project.

How to make your variables strongly typed

```
let lucky = 23;

lucky = '23'
```

```
let lucky: number;

lucky = '23'
lucky = 23
```

```
[ts] Type '"23"' is not assignable to type 'number'. [2322]
let lucky: number
lucky = '23'
```

```
let lucky: any = 23;

lucky = '23'
```

```
let lucky;

lucky = '23'
lucky = 23
```

NOT A GREAT PRACTICE!

Creating custom types

```ts
type Style = string;

let font: Style;
```

```ts
type Style = 'bold' | 'italic';

let font: Style;
```

```ts
interface Person {
    first: string;
    last: string;
}


const person: Person = {
    first: 'Jeff',
    last: 'Delaney'
}

const person2: Person = {
    first: 'Usain',
    last: 'Bolt',
    fast: true
}
```

```ts
interface Person {
    first: string;
    last: string;
    [key: string]: any
}


const person: Person = {
    first: 'Jeff',
    last: 'Delaney'
}

const person2: Person = {
    first: 'Usain',
    last: 'Bolt',
    fast: true
}
```

Strong typing a function

```
function pow(x: number, y: number): string {
    return Math.pow(x, y).toString();
}
```

```
function pow(x: number, y: number): void {
    Math.pow(x, y).toString();
}
```

Strong typing arrays/lists

```ts
const arr: number[] = []

arr.push(1)
arr.push('23')
arr.push(false)
```

```ts
type MyList = [number?, string?, boolean?]

const arr: MyList = []

arr.push(1)
arr.push('23')
arr.push(false)
```

Generics

```
class Observable<T> {
    constructor(public value: T) {}
}

let x: Observable<number>;

let y: Observable<Person>;

let z = new Observable(23)
```

Experiment with TS

- Create a react typescript app
- Check the contents in the project
- Run the app
- Create new components
- Add props with types