TypeScript

· TypeScript is a syntatic upgrade of JavaScript that adds additional syntax to make development and integration tighter, allowing for errors to potentially be caught sooner

· JS is an interpreted language, meaning types are determined at runtime. This can make it difficult to understand what types of data are being passed around in the code (frustrating at a larger scale). Types are checked at compile time BEFORE running the code so type errors can be identified early.

– To install TypeScript, run `npm i typescript --save-dev`

this flag means the dependency is only required for development and is NOT essential for production.

– To compile TS code, run `npx tsc --init`
↳ this initalizes the compiler with a tsconfig.json file

Primitive Types:

- boolean : T/F
- number · ints and floats
- String : text values in quotes

Explicit vs. Implicit Assignment

- Explicit: declaring the type

```
let firstName: string = "Rahman"
```

· easier to read with more intention

- Implicit: guessing the type based on its assigned value

```
let firstName = "Rahman"
```

· better for testing because its short and quick

– If TS cannot determine the type, the type will be set to any (disables type checking). 'any' can also be assigned explicitly to disable type-checking (not recommended)

↓

- The type 'unknown' is a safer alternative to 'any' because it prevents variables assigned with this type from being used. Unknown variables can later be assigned a type in the code.

Arrays

```
const names: string[] = [];
```

array type

```
names.push('Rahman'); ✓
```

↳ anything that isnt a string will throw an error.

· the 'readonly' keyword makes it so the array cannot be modified.

```
const names: readonly string[] = []
```

· TS can also infer the arrays type based on the values it holds

```
let job: [number, string];
```

↳ this is a 'tuple'. The job varicble will take 1 num argument and 1 string argument in that order.

```
type Person {
  name: string;
  age?: number;
}
```

age is optional

```
let deliClerk: Person {
  name: 'Steven',
}
```

age was optional

```
let age: number | string;
```

↳ Union, age can be a num OR string