Fundamentals

Summary

- Since TypeScript is a superset of JavaScript, it includes all the built-in types in JavaScript (eg number, string, boolean, object, etc) as well as additional types (eg any, unknown, never, enum, tuple, etc).

- In TypeScript, we set the type of our variables by annotating them.

- The **any** type can represent any kind of value. It's something we should avoid as much as possible because it defeats the purpose of using TypeScript in the first place. A variable of type **any** can take any kind of value!

- Tuples are fixed-length arrays where each element has a specific type. We often use them for representing two or three related values.

- Enums represent a list of related constants.

Cheat Sheet

Annotation

```
let sales: number = 123_456_789;
let numbers: number[] = [1, 2, 3];
```

Tuples

```
let user: [number, string] = [1, 'Mosh'];
```

Enums

```
enum Size { Small = 1, Medium, Large };
```

Functions

```
function calculateTax(income: number): number {
  return income * .2;
}
```

Objects

```
let employee: {
    id: number;
    name: string;
    retire: (date: Date) => void
} = {
   id: 1,
   name: 'Mosh',
   retire: (date: Date) => {},
};
```

Compiler Options

Option	Description
`noImplicitAny`	When enabled, the compiler will warn you about variables that are inferred with the **any** type. You'll then have to explicitly annotate them with **any** if you have a reason to do so.
`noImplicitReturns`	When enabled, the compiler will check all code paths in a function to ensure they return a value.
`noUnusedLocals`	When enabled, the compiler will report unused local variables.
`noUnusedParameters`	When enabled, the compiler will report unused parameters.