Typescript

Today

- Typescript intro
- Code example
- Exercises available in the repo
- Examination assignment will be introduced tomorrow

TypeScript

TypeScript is a superset of JavaScript that adds **static typing** and other features.
TypeScript helps catch errors early and improves code readability.

Why Use TypeScript?

1. **Static Typing**

   Types for variables, function parameters, and return values.

   ```
   let name: string = "John"; // string type
   ```

2. **Enhanced IDE Support**

   Better code editor features like **auto-completion** and **type checking**.

3. **Error Prevention**

   Catch errors like type mismatches and incorrect arguments at **compile-time**.

Optional Typing

- TypeScript allows you to write plain JavaScript if you choose not to specify types.
- TypeScript **infers types** even without explicit declarations.

Example:

```
let age = 25; // Inferred as number
```

ECMAScript Compatibility

TypeScript supports all modern JavaScript features:

- **Arrow functions**
- **Modules**
- **Async/await**
- **Destructuring**

> Compiles down to JavaScript, so it runs in any JS environment.

Interfaces and Classes

TypeScript allows the use of **interfaces** and **classes** for object-oriented programming.

Example:

```
interface Person {
  name: string;
  age: number;
  greet(): string;
}

class Student implements Person {
  constructor(public name: string, public age: number) {}

  greet() {
    return `Hello, my name is ${this.name}`;
  }
}
```

Generics

Generics allow you to write reusable code that works with any data type. Not a requirement for the course.

Example:

```
function identity<T>(arg: T): T {
  return arg;
}

let output = identity<number>(5); // T is number
```

TypeScript vs. JavaScript

Feature	TypeScript	JavaScript
Typing	Static typing, optional	Dynamic typing
Error detection	Compile-time	Runtime
Tooling support	Excellent (due to types)	Limited
OOP features	Full support	Limited
Compatibility	Compiles to JavaScript	Native

Setting Up TypeScript

1. **Install TypeScript**
   Run the following command to install TypeScript in your project. Use `-g` to install globally. Typescript also comes with VS Code.

   ```
   npm install typescript
   ```

2. **Compile a TypeScript File**

   TypeScript files has the `.ts` file extension. Compile `.ts` files into JavaScript:

   ```
   tsc index.ts
   ```

   This is usually already setup with frameworks like Create React App.

3. **Configuring with `tsconfig.json`**

   TypeScript is configured with `tsconfig.json`. A sample `tsconfig.json` file. We will not go into depth about configuration.

```
{
  "compilerOptions": {
  "target": "ES2020",
  "module": "ESNext",
  "moduleResolution": "node",
  "outDir": "./dist",
  "strict": true,
  "esModuleInterop": true,
 },
 "include": ["src/**/*"],
 "exclude": ["node_modules"]
}
```

4. **Adding types**

   TypeScript types are available for most npm-packages and are necessary when working with Typescript.

   They are installed using npm. Example for `express`:

```
npm install --save-dev @types/express
```

5. **Create a TypeScript interface**

   Interfaces describes an objects properties and functions and this is the core of TypeScript. `Type` can also by used used.

```
interface User {
 name: string;
 age: number;
 occupation: string;
}
```

6. **Type annotations**

   To tell TypeScript what type objects and functions have, type annotation is used. TypeScript can also infer (assume) what an object is. Hover the mouse over a variable to see the inferred type.

```
function createRemappedUser(user: RemappedUser) {
   console.log(
   `First Name: ${user.firstName}, Last Name: ${user.lastName}, Personal ID: ${user.personId}`);
}
```

7. **The `any` keyword**

   The ´any´ keyword can be used in cases when the type is not known. By doing this we tell TypeScript to NOT type check making it behave like JavaScript. `any` should be avoided but allows you to work on code base with both TypeScript and JavaScript.

```
function createRemappedUser(user: any) {
   console.log(
   `First Name: ${user.firstName}, Last Name: ${user.lastName}, Personal ID: ${user.persnId}`);
}
```

Conclusion

TypeScript enhances JavaScript with:

- **Static typing**
- **Error prevention at compile time**
- **Better tooling** for large-scale applications

It’s ideal for projects that require maintainable, cleaner, and error-free code while staying compatible with JavaScript.

Links

- Typescript docs
- Experiment with Typescript in the browser
- Exercises
- More difficult exercises