NORDIC CODER

React ADVANCED

#Trương Thành Tài

#Frontend Technical Lead ekino.

```
> npx truthtai
```

INTRODUCTION

TypeScript is a typed superset of JavaScript
that compiles to plain JavaScript.

OPEN SOURCE

https://www.typescriptlang.org/

NORDIC CODER

TS TypeScript

React

ADVANCED

WHY WE SHOULD USE

https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages-loved

TYPESCRIPT - ENVIRONMENT SETUP

Node.js: https://nodejs.org/en/

Visual Studio Code: https://code.visualstudio.com/

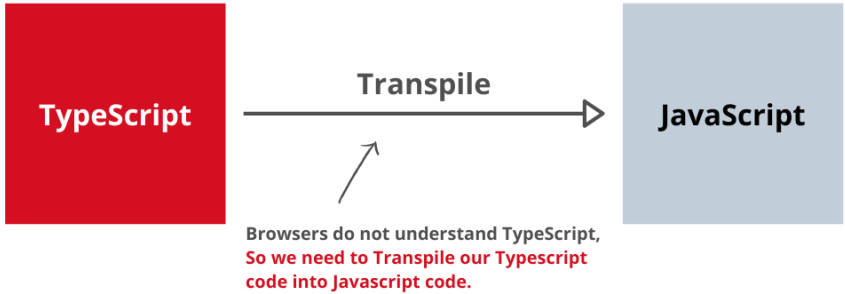
npm install -g typescript

TYPESCRIPT - **COMPILER**

Create a file **hello-word.ts**

```
let text: string = "Hello Word"

console.log(text)
```



To compile

```
tsc hello-word.ts
```

Run code compiled code

```
node hello-word.js
```

TYPESCRIPT - **CONFIGURATION**

```
{
  "compilerOptions": {
    "module": "commonjs",
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "target": "es5",
    "noImplicitAny": true,
    "moduleResolution": "node",
    "sourceMap": true,
    "outDir": "dist",
    "baseUrl": ".",
    "paths": {
      "*": [
        "node_modules/*"
      ]
    }
  }
}
```

DOCUMENT:

https://www.typescriptlang.org/docs/handbook/compiler-options.html

TSCONFIG.JSON EXAMPLE

https://gist.github.com/truthtaicom/0bbb286b8d199ff379e1cdb00602680b

TYPESCRIPT - **TYPE ANNOTATIONS**

```
let message : string = "hello world";       // string
const age: number = 30;                     // number
const isOK : boolean = true;                // boolean
let files: string[]                         // arrays
let student: { id: number; name: string; }; // object

files = ["hello-word.ts", "hello-word.js"]
student = { id: 100, name : "Nordic Coder" }
```

LAB:

- Create 5 variables which use 5 types (string, number Boolean, array, object)
- Try to make a type's error

TYPESCRIPT - **INTERFACES**

```
const person = {
    firstName: "Nordic",
    lastName: "Coder",
    sayHi: () => { return "Hi"}
};

interface IPerson {
    firstName: string
    lastName: string
    sayHi: () => string
}
```

=>

```
const person: IPerson = {
    firstName: "Nordic",
    lastName: "Coder",
    sayHi: () => { return "Hi"}
};
```

TYPESCRIPT – **TYPE OPTIONAL**

```
const person = {
    firstName: "Nordic",
    sayHi: () => { return "Hi"}
};

interface IPerson {
    firstName: string
    lastName?: string
    sayHi: () => string
}
```

=>

```
const person: IPerson = {
    firstName: "Nordic",
    sayHi: () => { return "Hi"}
};
```

TYPESCRIPT – **Class**

```typescript
interface IPerson {
  firstName: string
  lastName: string
  sayHi: () => string
}

class Person implements IPerson {
  firstName = '';
  lastName = '';
  sayHi = () => { return `Hi ${this.firstName} ${this.lastName}` }
}

const employee01 = new Person()
employee01.firstName = "Nordic"
employee01.lastName = "Coder"

console.log(employee01.sayHi())
```

LAB:

- Create a class which uses interface
- Try to make a type's error

TYPESCRIPT – **Function**

```typescript
interface IPerson {
  firstName: string
  lastName: string
}

function sayHello(data: IPerson): string {
  return `Hello ${data.firstName} ${data.lastName}`
}

const message = sayHello({
  firstName: "Nordic",
  lastName: "Coder"
})

console.log(message)
```

LAB:

- Create a function which uses interface
- Try to make a type's error

TYPESCRIPT – **HOMEWORK**

- Create 3 classes with inheritance which uses interface
- Create 3 functions with callback hell (3x) which use interface