Type Script Example:

Setup

- get Visual Studio Code
- get Node.js. It comes with npm package manager
- open command prompt and run the following command to install the latest stable version of TypeScript globally

```
npm install -g typescript
```

Configuration

Create an empty folder and open it in Visual Studio Code.
First thing we need to do is to create a tsconfig.json file. In order to do so we'll execute this command in terminal (Ctrl+` to open terminal)

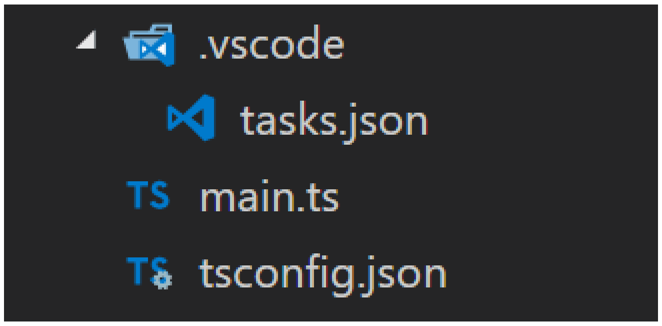
```
tsc --init
```

- create source code (ex. main.ts)

```
interface Person {
   age: number,
   name: string,
   say(): string
}

let mike = {
   age: 25,
   name:"Mike",
   say: function() {
      return `My name is ${this.name} and I'm ${this.age} years old!`;
   }
}
function sayIt(person: Person) {
   return person.say();
}

console.log(sayIt(mike))
```

- now we want to setup a convenient build process in order to run the project with a couple of buttons. Press `Ctrl+Shift+P` and start typing Configure Default Build Task, press `Enter` to select it then tsc: build - tsconfig.json. This will create a file named `tasks.json` in `.vscodefolder` (click `Refresh Explorer` on a project tab to see the changes). Now we have all needed commands and arguments for our build.

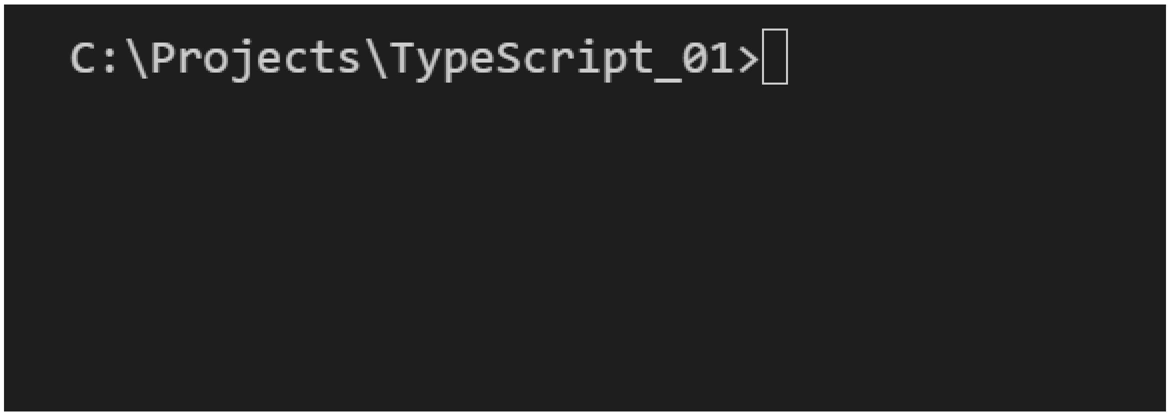
This is our project structure after all the steps.



Run

It's time to finally run the build task. Press `Ctrl+Shift+B` and if all went well a new file will be created (`main.js`). In order to see the output we need to feed it into node command.
node main.js

Let's see it in action!



```
C:\Projects\TypeScript_01>
```

Working with DOM

Create a new file named index.html. It's so minimalist that I'm even embarassed a little bit.

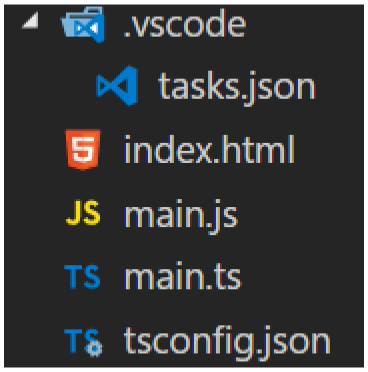
```
<!DOCTYPE html>
<html>
  <body>
    <h1>Fun with TypeScript</h1>
    <p id="rock_id">Let's rock</p>

    <script src="main.js"> </script>
  </body>
</html>
```

Let's change `main.ts` file and modify `<p>` element inner text using `TypeScript`. The main part here is `<script src="main.js">` element. `main.js` is a transplied code from `TypeScript` and will run naturally.

WARNING!!! Another minimalist example!

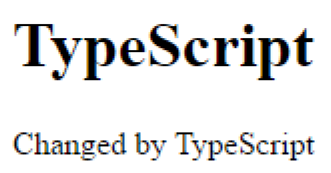
```
document.getElementById("rock_id")!.innerHTML = "Changed by TypeScript!"
```

Final project structure after all the changes.



```
.vscode
  tasks.json
index.html
main.js
main.ts
tsconfig.json
```

Press `Ctrl+Shift+B` and check `main.js` file (just for curiosity). Next, open `index.html` and observe the result. Wow! So easy!

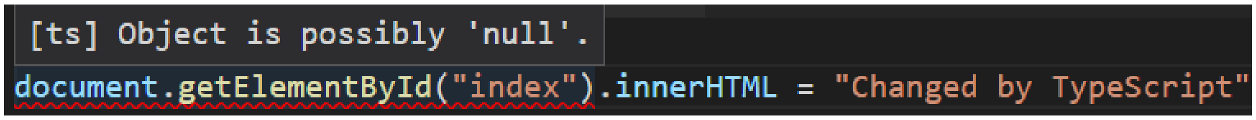


TypeScript

Changed by TypeScript

index.html page

Awesome, but there is something strange in this example. What is `!` symbol doing here? It’s called the [non-null assertion operator](). Compiler forces us to check for `null/undefined` values if `tsconfig.json` is configured with `strict` flag. If we try to omit it the compiler will yell at you.



```
[ts] Object is possibly 'null'.
document.getElementById("index").innerHTML = "Changed by TypeScript"
```

Compiler error with -strict flag

We must explicitly check for `null/undefined` in order to safely use the return value from `.getElementById`. But in this example it’s redundant because I’m 100% sure that it won’t return any `null/undefined`. So I just use `!`.