Typescript essentials. Introduction

Let's have a look at JavaScript example which illustrate some potential problem connected with types. First, create "index.html" with two inputs and one button (fig. 1). Also, attach "add.js" to it, which will perform the operation of addition of two digits.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Adding two digits example</title>
    <script src="add.js" defer></script>
</head>
<body>
    <input type="number" id="num1">
    <input type="number" id="num2">
    <button id="add_btn">add</button>
</body>
</html>
```

Figure 1 – Index.html

In "add.js" file we get two number inputs and button by ids, define function which add two numbers and call this function on "button_click" event with sending the result to the console (fig. 2).

add.js

```
const btn = document.getElementById('add_btn')
const num1_input = document.getElementById('num1')
const num2_input = document.getElementById('num2')

function add(num1, num2) {
    return num1 + num2
}

btn.addEventListener('click', function() {
    console.log(
        add(num1_input.value, num2_input.value)
    )
})
```

Figure 2 – Add.js

When clicking on button in browser we expect to reach “2+2=4”, but instead we have “22” (fig. 3). It happens because of incorrect variable types. “Input.value” is always a string and concatenation of two “2” strings give us “22”. Moreover, the main problem is not the incorrect result, but missing of any errors at the step of code compiling.

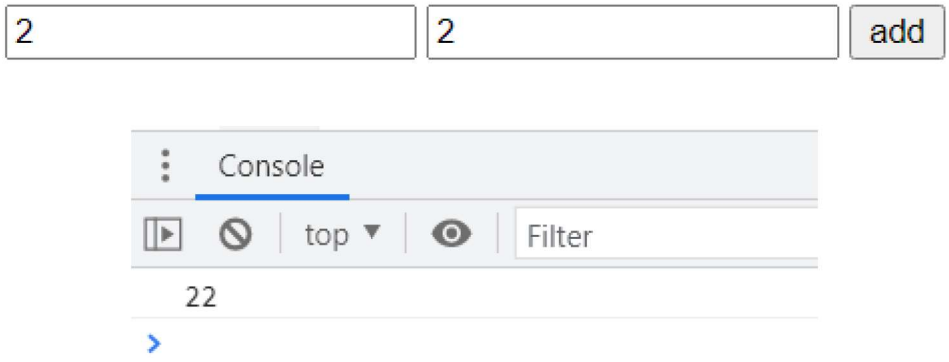

Figure 3 – The result of adding of two numbers

Typescript is an open-source language, which is built on JavaScript by adding static type definitions. Types provide a way to describe the shape of an object, providing better documentation, and allowing Typescript to validate that your code is working correctly. It needs to mention, that writing types can be optional in Typescript [https://www.typescriptlang.org/].

To use Typescript first we need to download and install Node.js. It can be done by using project official website https://nodejs.org/ (fig. 4).

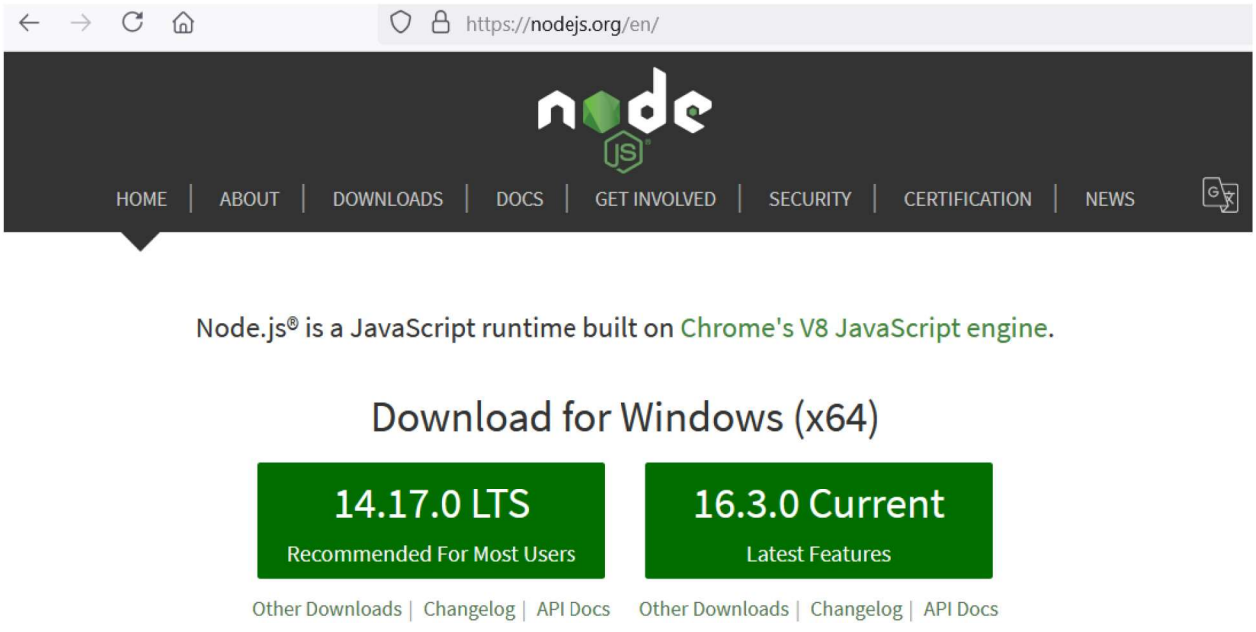

Figure 4 – Download and install Node.js

After installation you can check installed Node.js version by executing in command line following command “node --version”. If the installation was done well you will see something like this (fig. 5).

```
C:\Users\michael>node --version
v10.15.3
```

Figure 5 – Check Node.js version

While installing Node.js you also usually npm is automatically installed, which is the node package manager. It allows to work with different code packages for JavaScript. To check current version on your computer run “npm --version” command (fig. 6).

```
C:\Users\michael>npm --version
6.4.1
```

Figure 6 – Npm version checking

At last run “npm install -g typescript@next” command to install Typescript globally (fig. 7).

```
C:\Users\michael>npm install -g typescript@next
```

Figure 7 – Install Typescript

After installing Typescript we can also check it’s version by running “tsc --version” command (fig. 8).

```
C:\Users\michael>tsc --version
Version 4.4.0-dev.20210606
```

Figure 8 – Typescript version checking

So, now we have everything we need to try our first Typescript code. Let's go back to example with adding two numbers (fig. 1-3). Remove "add.js" file and add to the project new "add.ts" file with following content (fig. 9).

```
1  const btn = document.getElementById('add_btn')
2  const num1_input = document.getElementById('num1') as HTMLInputElement
3  const num2_input = document.getElementById('num2') as HTMLInputElement
4
5  function add(num1: number, num2: number): number {
6      return num1 + num2
7  }
8
9  btn.addEventListener('click', function() {
10     console.log(
11         add(+num1_input.value, +num2_input.value)
12     )
13 })
```

Figure 9 – "Add2.ts"

First difference in 2nd and 3rd rows is casting "num1_input" and "num2_input" to "HTMLInputElement". It is done because not every html element has "value" property. So, if we remove this transformation, typescript will give us an error (fig. 10).

```
any
EventListener('cl  Property 'value' does not exist on type 'HTMLElement'
sole.log(          View Problem (Alt+F8)   Quick Fix... (Ctrl+.)
 add(+num1_input.value, +num2_input.value)
```

Figure 10 – Property 'value' existence potential problem

In addition, function "Add" was given the input parameters types and return value type (row 5). After this, if we pass to the function different value types, Typescript will give an error (fig. 11).

```
turn n
    const num1_input: HTMLInputElement
    Argument of type 'string' is not assignable to parameter of type
ldEvent 'number'. ts(2345)
nsole.  View Problem (Alt+F8)   Quick Fix... (Ctrl+.)
  add(num1_input.value, +num2_input.value)
```

Figure 11 – Typescript error while giving wrong value type

As we can see, Typescript gives little bit more quality to our code. And it also become more understandable. It needs to be mentioned that Typescript is a superset of JavaScript, so all JavaScript code is still working.

But we cannot run Typescript code in browser, so we need to compile this new “add.ts” to JavaScript. To do this, run “tsc add.ts” command (fig. 12). After command execution is finished the “add.js” file is created (fig. 12).

add.js > ...

```
var btn = document.getElementById('add_btn');
var num1_input = document.getElementById('num1');
var num2_input = document.getElementById('num2');
function add(num1, num2) {
    return num1 + num2;
}
btn.addEventListener('click', function () {
    console.log(add(+num1_input.value, +num2_input.value));
});
```

Figure 12 – Compile Add.ts into an Add.js

The final version of JavaScript code is similar to the first version, which was given in fig. 2. But while working on “add.ts” we had some addition level of checking our code for potential mistakes. And now we have “2+2=4” (fig. 13).

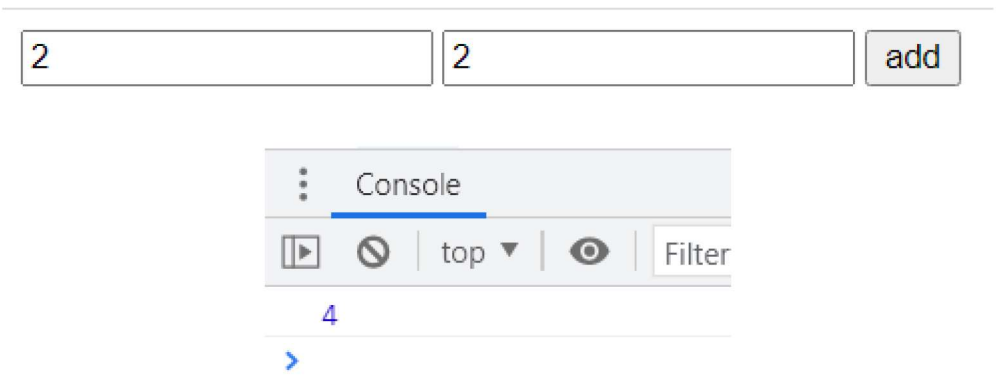

Figure 13 - The result of adding of two numbers

Beside types, Typescript provides encapsulation that allows to close some parts of code. It also provides classes and interfaces with complete set of abstract, final classes, extensions and so on.

Finally let’s add to our project lite-server npm package, which is the lite web server for development. It also maintains single page application technology. To do

this, run “npm install lite-server --save-dev” command in project directory. Another way is to add “package.json” file with following content:

```
{
 ...
 "devDependencies": {
  "lite-server": "^2.6.1"
  ...
 }
}
```

Also in “package.json” file let’s place several another rows, which will be in charge of two scripts: one for run lite-server and display application in browser, and second for build all “js” files from “ts” scripts:

```
"scripts": {
   "build": "tsc -w",
   "start": "lite-server"
 }
```

Now to build application we need to run “npm run build” command. To run application execute “npm start” command.

After all, add “tsconfig.json” file to configure typescript compiler to compile all “ts” files in “/scr” directory to “dist” folder:

```
{
   "compilerOptions": {
      "target": "es6",
      "outDir": "./dist",
      "rootDir": "./src"
   },
   "include": [
      "**/*.ts"
   ]
}
```