

TypeScript Fundamentals

Olgun DUTKAN – UI / UX Team in HAVELSAN
odutkan@havelsan.com.tr



```
console.info(`Hello, I am Olgun DUTKAN`)  
// Hello, I am Olgun DUTKAN
```

Agenda

1. Introduction to Typescript
2. Setting up Typescript Development Environment
3. Typescript latest features-concepts

What is TypeScript?

- The TypeScript programming language is a superset of JavaScript that adds types to JavaScript using a set of tools called a type system.

What is TypeScript?

- It was designed and developed by Anders Hejlsberg (designer of C#) at Microsoft
- First, we write TypeScript code in files with the extension .ts.
- The developer can identify mistakes during development/compilation step (scope of the variable, function parameter, variable datatype mismatch, etc.)
- We run our code through the TypeScript transpiler. The transpiler will check that the code adheres to TypeScript's standards, and it will display errors when it does not.
- If the TypeScript code can be converted into working JavaScript, the transpiler will output a JavaScript version of the file (.js).



A brief history of TypeScript



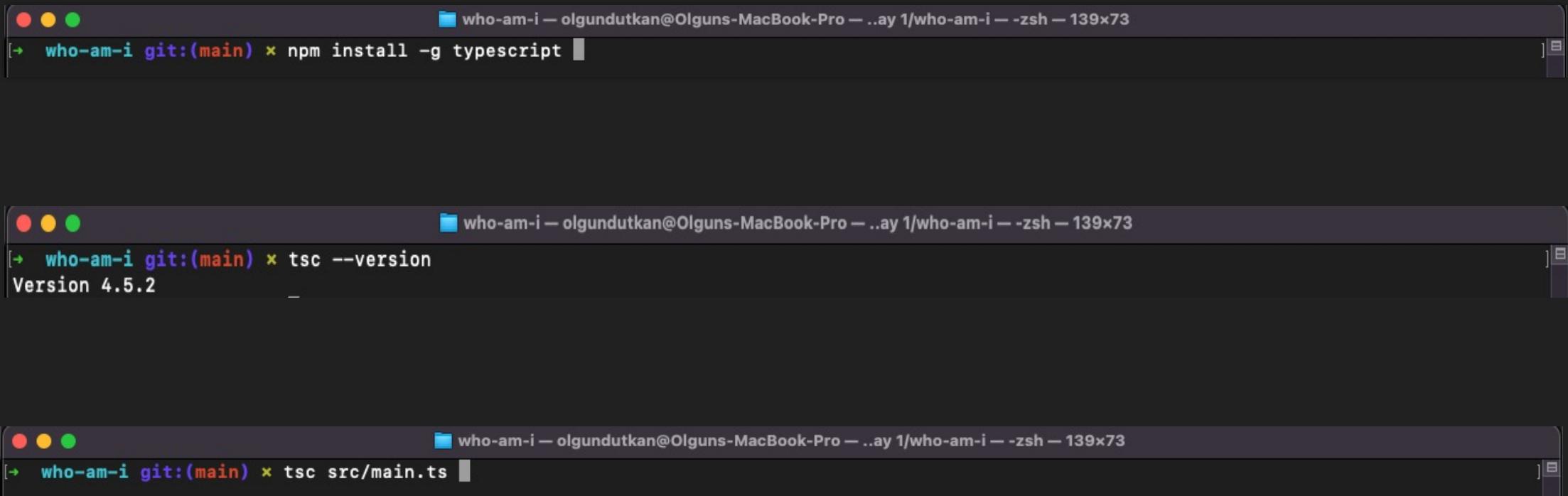
Why TypeScript?

- Types are documentation
- You're not locked in
- We're all processing our JavaScript before running it anyway
- Interfaces let you remove data clumps
- It makes the tooling better

Components of TypeScript

- Language – syntax, keywords, and type annotations
- TSC TypeScript Compiler – converts TypeScript to JavaScript equivalent
- TLS TypeScript Language Service – Supports editor operations like static typing and type inference system, statement completion, code formatting, etc.

Development Environment Setup

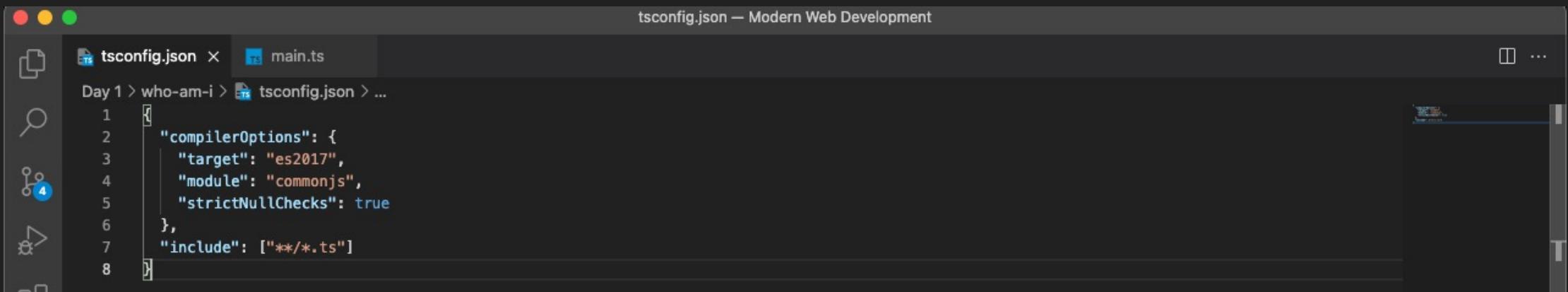


The image displays three separate terminal windows, each showing a different step in the development environment setup process.

- Terminal 1:** Shows the command `npm install -g typescript` being run in a directory named `who-am-i`. The output shows the package is already up-to-date.
- Terminal 2:** Shows the command `tsc --version` being run in the same directory. The output displays the TypeScript version as `Version 4.5.2`.
- Terminal 3:** Shows the command `tsc src/main.ts` being run in the same directory. The output is currently blank, indicating the file has been successfully compiled.

The `tsconfig.json` file

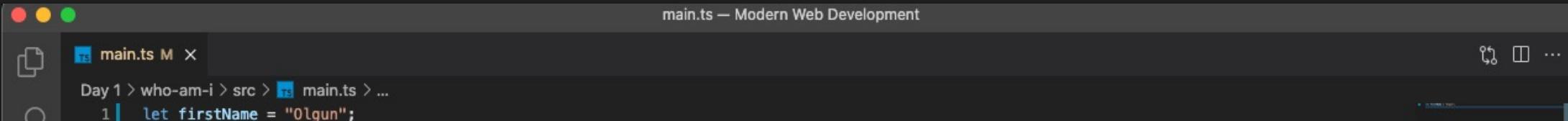
- The **`tsconfig.json`** file is always placed in the root of your project and you can customize what rules you want the TypeScript compiler to enforce.

A screenshot of a code editor window titled "tsconfig.json – Modern Web Development". The window shows a file tree on the left with "tsconfig.json" and "main.ts" selected. The main area displays the contents of the tsconfig.json file:

```
1  {
2    "compilerOptions": {
3      "target": "es2017",
4      "module": "commonjs",
5      "strictNullChecks": true
6    },
7    "include": ["**/*.ts"]
8 }
```

A status bar at the bottom right shows the letters "TS".

What TypeScript code looks like?

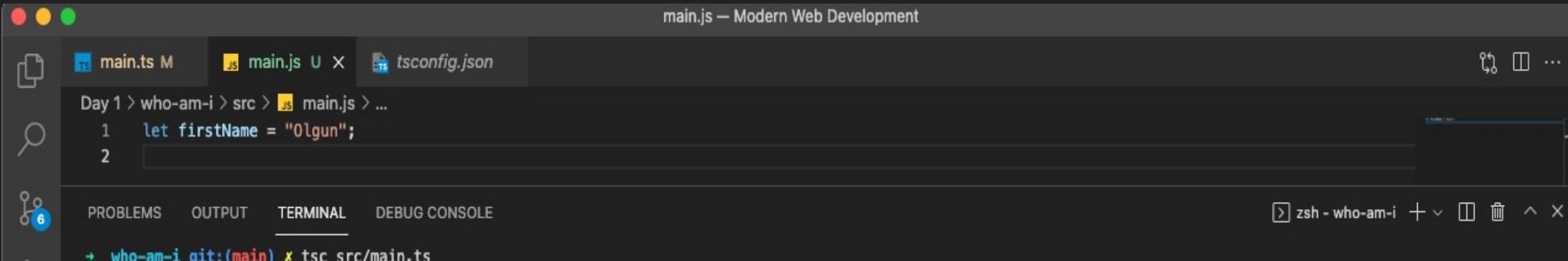


main.ts — Modern Web Development

main.ts M X

Day 1 > who-am-i > src > main.ts > ...

```
1 | let firstName = "Olgun";
```



main.js — Modern Web Development

main.ts M main.js U tsconfig.json

Day 1 > who-am-i > src > main.js > ...

```
1 let firstName = "Olgun";
2 
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

zsh - who-am-i + ^ X

```
→ who-am-i git:(main) ✘ tsc src/main.ts
```

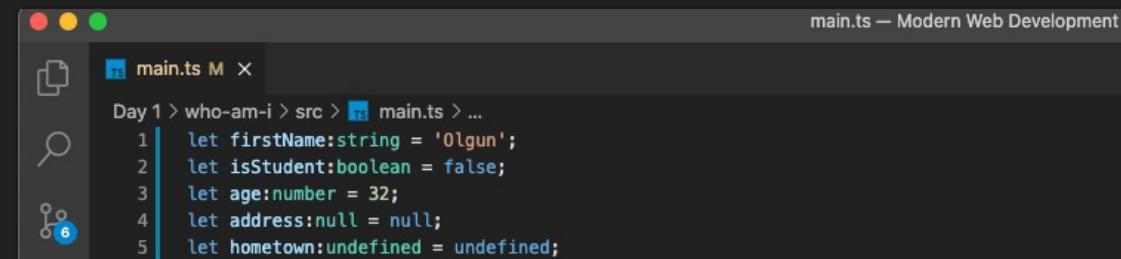
TypeScript Playground

- Write and test Typescript without download or install anything:

<https://www.typescriptlang.org/play>

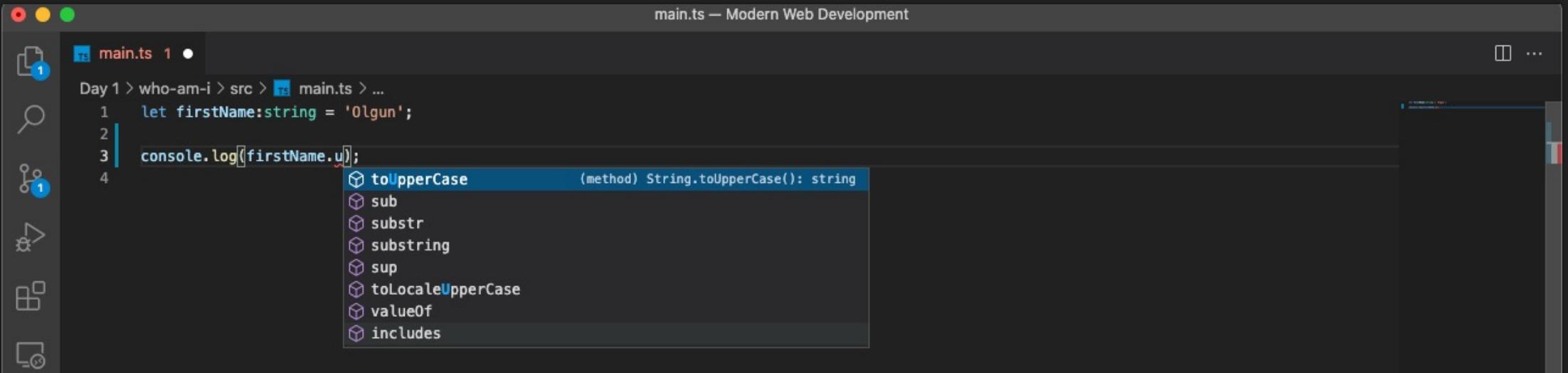
Type Inferences

- boolean
- number
- null
- string
- undefined



```
main.ts — Modern Web Development
Day 1 > who-am-i > src > main.ts > ...
1 let firstName:string = 'Olgun';
2 let isStudent:boolean = false;
3 let age:number = 32;
4 let address:null = null;
5 let hometown:undefined = undefined;
```

Type Shapes



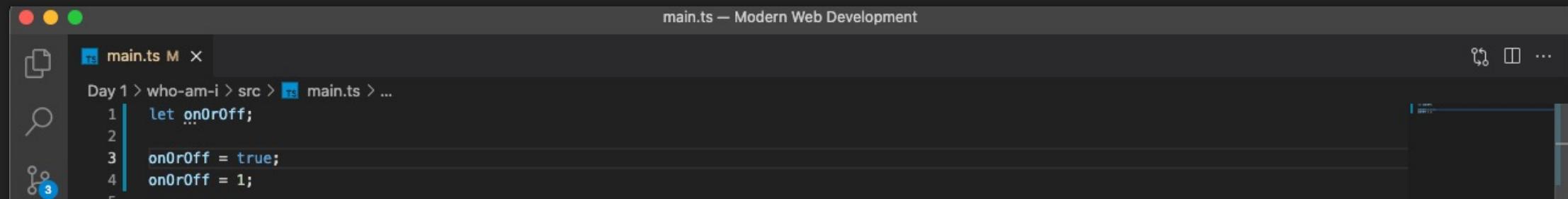
A screenshot of a code editor window titled "main.ts — Modern Web Development". The file path is "Day 1 > who-am-i > src > main.ts > ...". The code in the editor is:

```
1 let firstName:string = 'Olgun';
2
3 console.log(firstName.u);
4
```

A tooltip is displayed over the line "console.log(firstName.u);", showing the available methods for the `String` prototype. The tooltip lists the following methods:

- toUpperCase (method) `String.toUpperCase(): string`
- sub
- substr
- substring
- sup
- toLocaleUpperCase
- valueOf
- includes

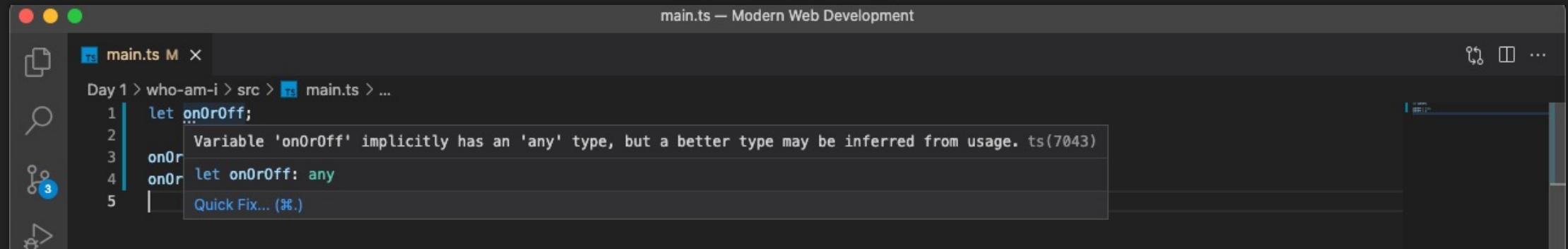
Type Any



main.ts — Modern Web Development

```
main.ts M X
Day 1 > who-am-i > src > main.ts > ...
1 let onOrOff;
2
3 onOrOff = true;
4 onOrOff = 1;
```

Type Any

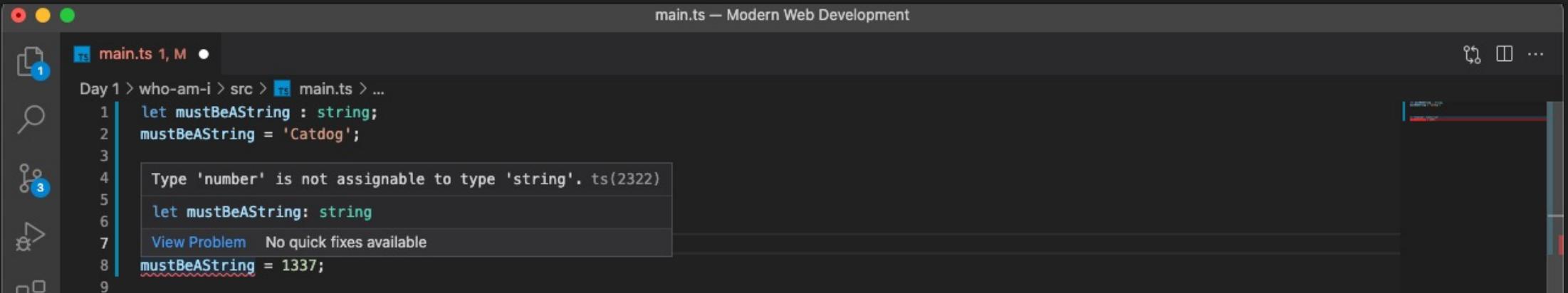


The screenshot shows a dark-themed instance of Visual Studio Code with a single file open: `main.ts`. The title bar indicates the file is named `main.ts` and the project is "Modern Web Development". The code editor displays the following TypeScript code:

```
1 let onOrOff;
2
3 onOr
4 onOr let onOrOff: any
5
```

A red squiggle underline is under the identifier `onOrOff in line 1, indicating a TypeScript error. A tooltip message appears above the squiggle: "Variable 'onOrOff' implicitly has an 'any' type, but a better type may be inferred from usage. ts(7043)". Below the code editor, there is a status bar with icons for file operations and a "Quick Fix..." button.`

Variable Type Annotations



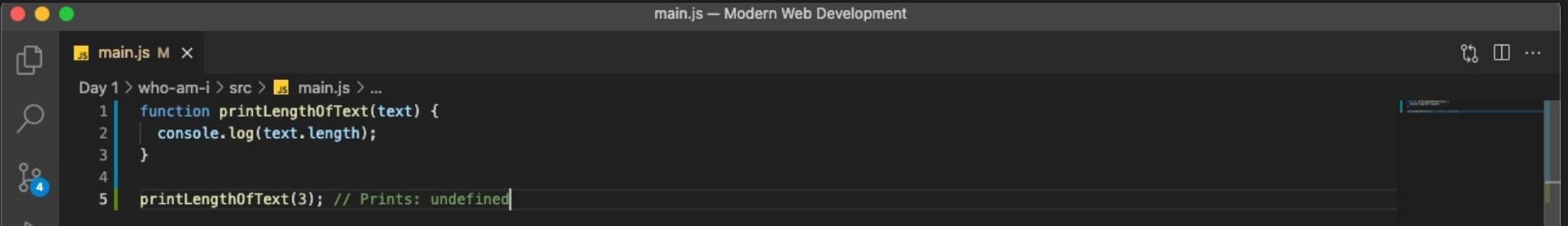
A screenshot of the Visual Studio Code (VS Code) interface, specifically showing a TypeScript file named `main.ts`. The title bar indicates the file is open and part of a project named "Modern Web Development". The code editor shows the following TypeScript code:

```
Day 1 > who-am-i > src > main.ts > ...
1 let mustBeAString : string;
2 mustBeAString = 'Catdog';
3
4 Type 'number' is not assignable to type 'string'. ts(2322)
5
6 let mustBeAString: string
7 View Problem No quick fixes available
8 mustBeAString = 1337;
9
```

The line `mustBeAString = 1337;` is highlighted with a red squiggle under `mustBeAString`, indicating a TypeScript error. A tooltip or status message above the line states `Type 'number' is not assignable to type 'string'. ts(2322)`. The "View Problem" and "No quick fixes available" buttons are visible below the error message. The left sidebar shows icons for file, search, and other development tools, with a blue circle containing the number "3" indicating pending changes or notifications.

```
console.info(`Review`)  
// Review
```

Functions

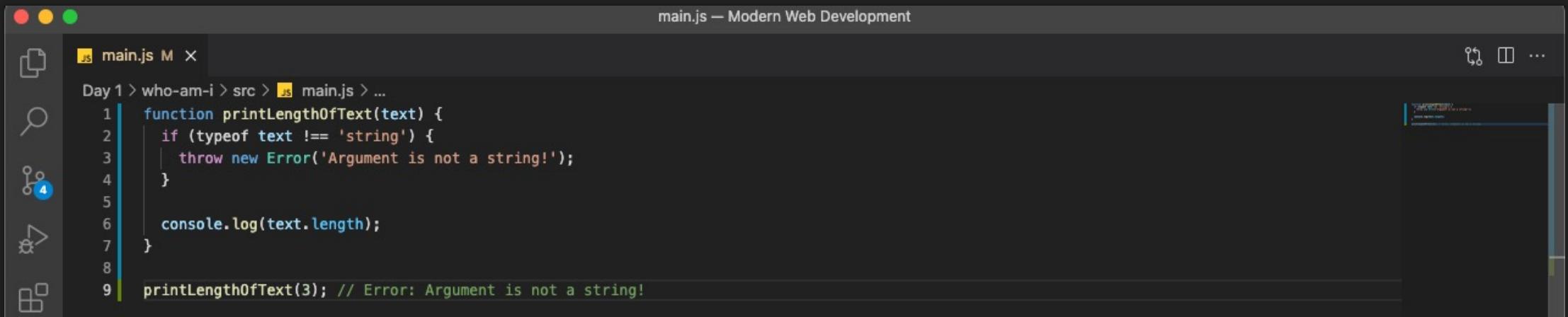


The screenshot shows a code editor window titled "main.js — Modern Web Development". The file contains the following JavaScript code:

```
Day 1 > who-am-i > src > main.js > ...
1 | function printLengthOfText(text) {
2 |   console.log(text.length);
3 |
4 |
5 | printLengthOfText(3); // Prints: undefined
```

The code defines a function `printLengthOfText` that logs the length of a given string to the console. When the function is called with the argument `3`, it prints `undefined` because the argument is not a string.

Functions

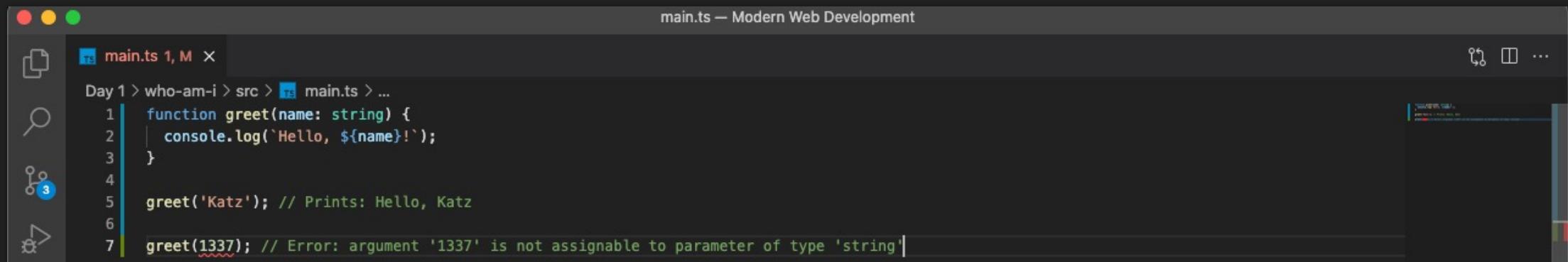


The screenshot shows a code editor window titled "main.js – Modern Web Development". The file content is as follows:

```
Day 1 > who-am-i > src > main.js > ...
1  function printLengthOfText(text) {
2    if (typeof text !== 'string') {
3      throw new Error('Argument is not a string!');
4    }
5
6    console.log(text.length);
7  }
8
9  printLengthOfText(3); // Error: Argument is not a string!
```

The code defines a function `printLengthOfText` that checks if its argument is a string. If not, it throws an `Error`. Otherwise, it logs the length of the string to the console. A call to this function with the argument `3` results in an error message: `// Error: Argument is not a string!`.

Parameter Type Annotations



main.ts — Modern Web Development

```
main.ts 1, M X
Day 1 > who-am-i > src > main.ts > ...
1  function greet(name: string) {
2    console.log(`Hello, ${name}!`);
3  }
4
5  greet('Katz'); // Prints: Hello, Katz
6
7  greet(1337); // Error: argument '1337' is not assignable to parameter of type 'string'
```

Optional Parameters

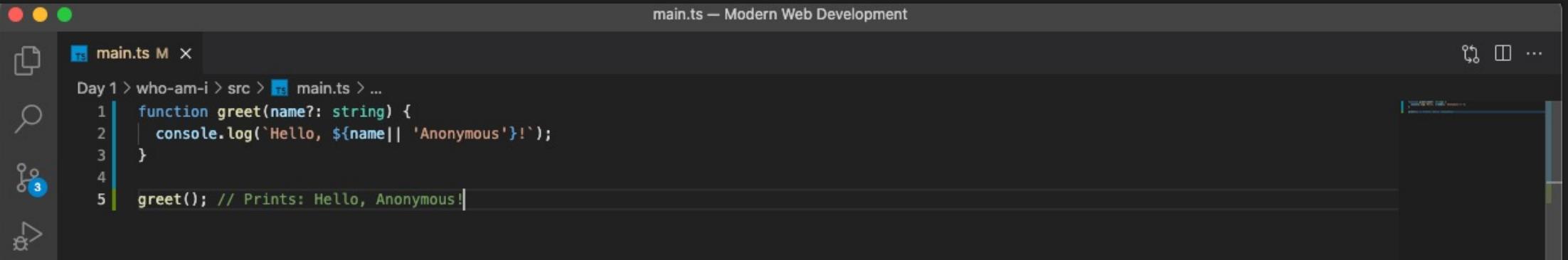


The screenshot shows a dark-themed code editor window titled "main.ts — Modern Web Development". The sidebar on the left displays file navigation with "main.ts 1, M X" selected. The main editor area contains the following TypeScript code:

```
Day 1 > who-am-i > src > main.ts > ...
1 function greet(name: string) {
2   console.log(`Hello, ${name || 'Anonymous'}!`);
3 }
4
5 greet('Anders'); // Prints: Hello, Anders!
6 greet(); // TypeScript Error: Expected 1 arguments, but got 0.
```

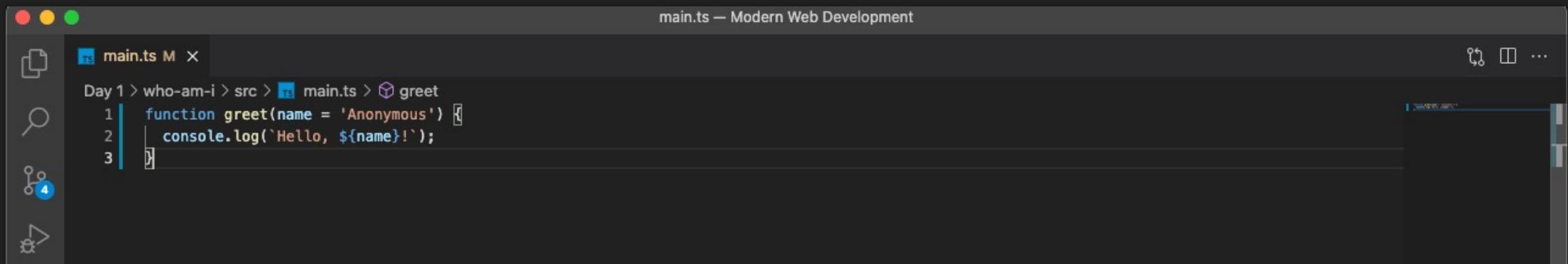
The code defines a function "greet" that logs a message to the console. It uses optional parameters by including a default value ("Anonymous") in the spread operator. The editor highlights the optional parameter with a blue underline. The final two lines demonstrate calling the function with one argument ("Anders") and without any arguments, which triggers a TypeScript error for the empty call.

Optional Parameters



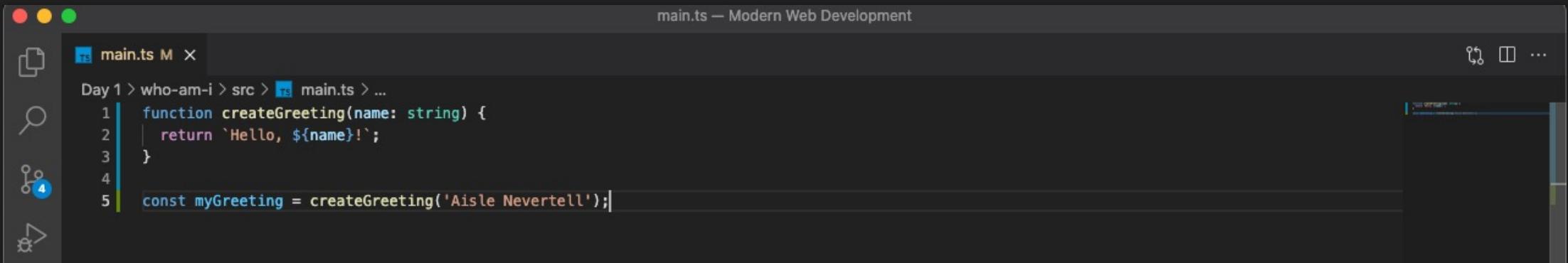
```
main.ts — Modern Web Development
main.ts M X
Day 1 > who-am-i > src > main.ts > ...
1 function greet(name?: string) {
2   console.log(`Hello, ${name || 'Anonymous'}!`);
3 }
4
5 greet(); // Prints: Hello, Anonymous!
```

Default Parameters



```
main.ts — Modern Web Development
main.ts M X
Day 1 > who-am-i > src > main.ts > greet
1 | function greet(name = 'Anonymous') {
2 |   console.log(`Hello, ${name}!`);
3 | }
```

Inferring Return Types

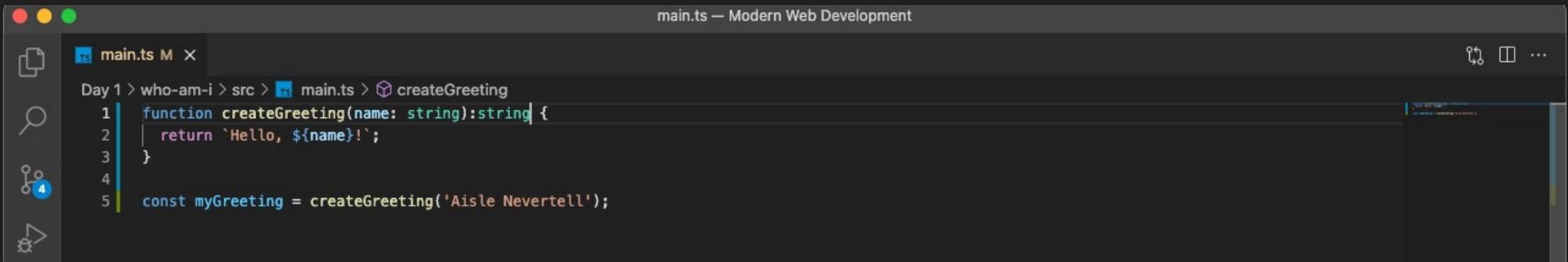


A screenshot of a code editor window titled "main.ts — Modern Web Development". The file path is "Day 1 > who-am-i > src > main.ts > ...". The code editor displays the following TypeScript code:

```
1 function createGreeting(name: string) {  
2   return `Hello, ${name}!`;  
3 }  
4  
5 const myGreeting = createGreeting('Aisle Nevertell');
```

The code editor interface includes standard window controls (red, yellow, green buttons), a sidebar with icons for file, search, and notifications (with a count of 4), and a status bar at the bottom.

Explicit Return Types



```
main.ts — Modern Web Development
main.ts M X
Day 1 > who-am-i > src > main.ts > createGreeting
1 | function createGreeting(name: string):string {
2 |   return `Hello, ${name}!`;
3 |
4 |
5 | const myGreeting = createGreeting('Aisle Nevertell');
```

Explicit Return Types

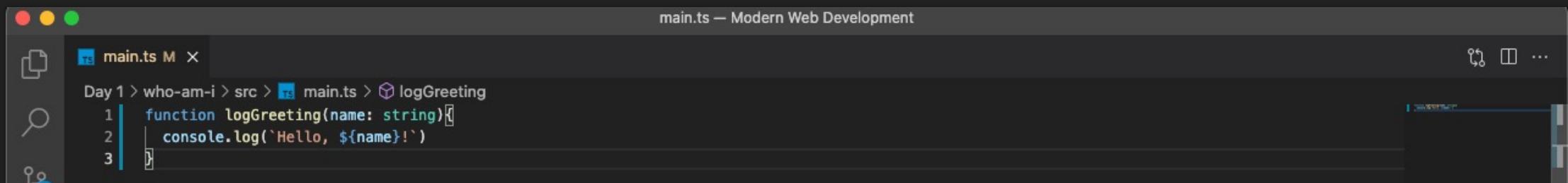


A screenshot of the Visual Studio Code (VS Code) interface, showing a TypeScript file named `main.ts`. The title bar says "main.ts — Modern Web Development". The code editor contains the following TypeScript code:

```
Day 1 > who-am-i > src > main.ts > createGreeting
1 function createGreeting(name?: string): string {
2   if (name) {
3     return `Hello, ${name}!`;
4   }
5
6   Type 'undefined' is not assignable to type 'string'. ts(2322)
7
8   var undefined
9
10  View Problem  No quick fixes available
11  return undefined;
12  //TypeScript Error: Type 'undefined' is not assignable to type 'string'.
13};
```

The line `return undefined;` is underlined with a red squiggle, indicating a TypeScript error. A tooltip above the line shows the error message: "Type 'undefined' is not assignable to type 'string'. ts(2322)". The sidebar on the left shows icons for file, search, and other development tools.

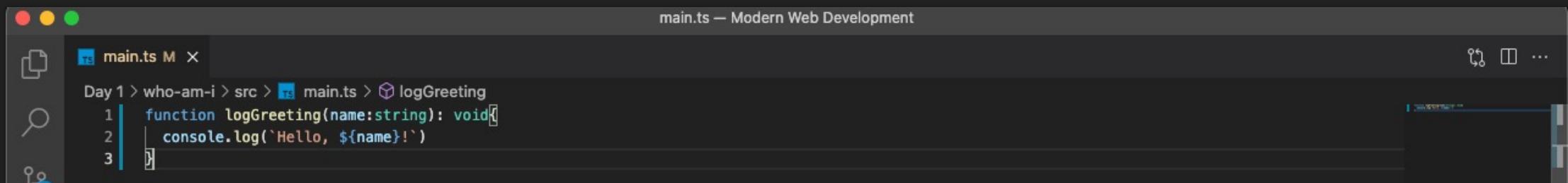
Void Return Type



The screenshot shows a dark-themed code editor window titled "main.ts — Modern Web Development". The file path in the title bar is "Day 1 > who-am-i > src > main.ts". The code editor displays the following TypeScript code:

```
function logGreeting(name: string){  
  console.log(`Hello, ${name}!`)  
}
```

Void Return Type



main.ts — Modern Web Development

```
main.ts M X
Day 1 > who-am-i > src > main.ts > logGreeting
1 | function logGreeting(name:string): void{
2 |   console.log(`Hello, ${name}!`)
3 | }
```

```
console.info(`Review`)  
// Review
```

Array Type Annotations



The image displays three vertically stacked code editors, each showing a different example of array type annotations in TypeScript. The editors have a dark theme with light-colored code.

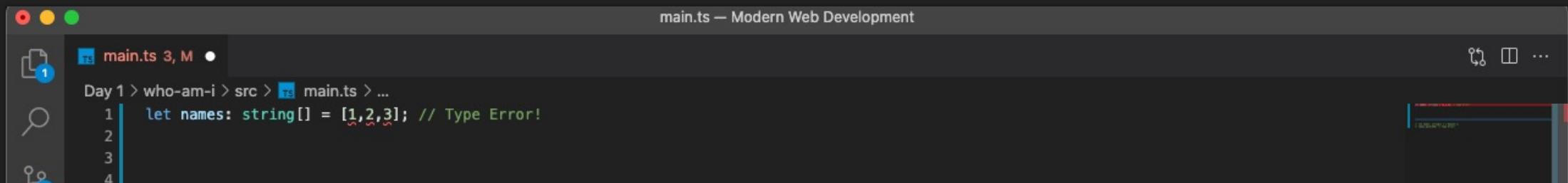
- Editor 1:** Shows a simple array of strings.

```
main.ts — Modern Web Development
main.ts M X
Day 1 > who-am-i > src > main.ts > ...
1 | let names: string[] = ['Danny', 'Samantha'];
```
- Editor 2:** Shows an array of strings annotated with `Array<string>`.

```
main.ts — Modern Web Development
main.ts M X
Day 1 > who-am-i > src > main.ts > ...
1 | let names: Array<string> = ['Danny', 'Samantha'];
```
- Editor 3:** Shows a tuple with three elements: a string, a number, and a boolean.

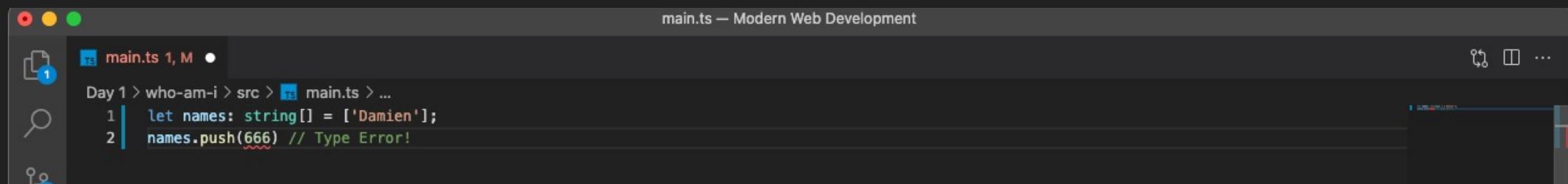
```
main.ts — Modern Web Development
main.ts M X
Day 1 > who-am-i > src > main.ts > ...
1 | let mixedTuple: [string, number, boolean] = ['hi', 3, true];
```

Array Type Annotations



main.ts — Modern Web Development

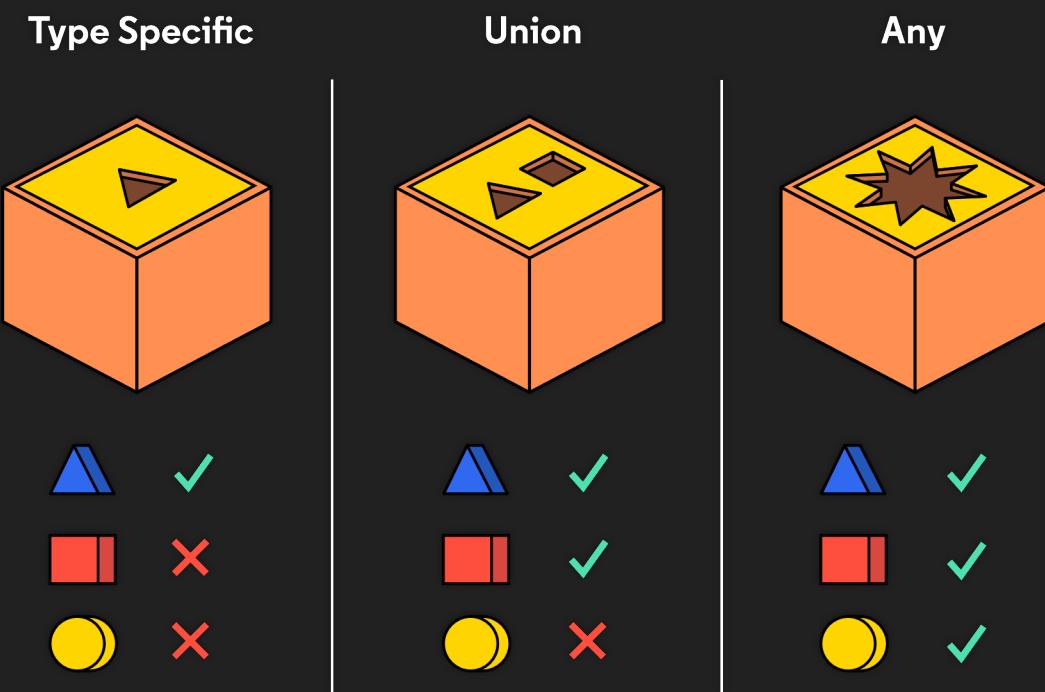
```
main.ts 3, M
Day 1 > who-am-i > src > main.ts > ...
1 | let names: string[] = [1,2,3]; // Type Error!
2 |
3 |
4 |
```



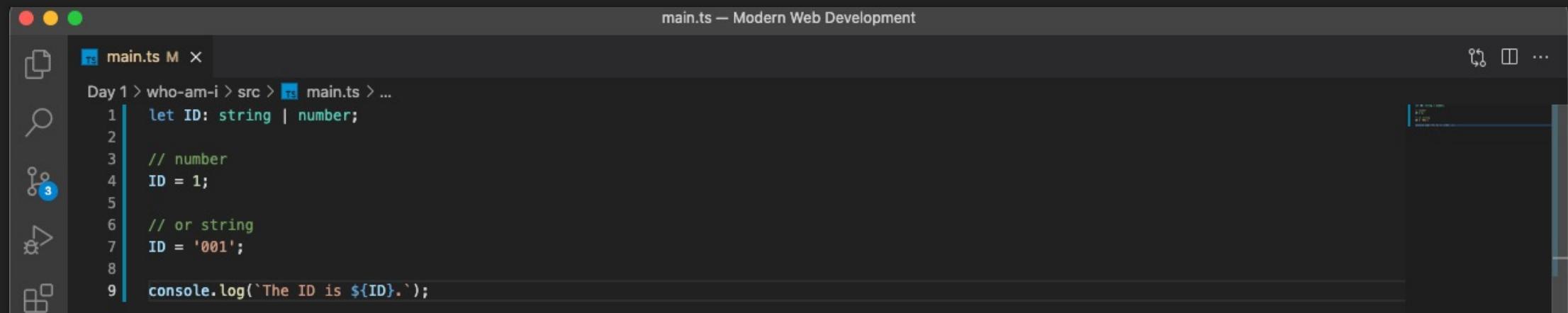
main.ts — Modern Web Development

```
main.ts 1, M
Day 1 > who-am-i > src > main.ts > ...
1 | let names: string[] = ['Damien'];
2 | names.push(666) // Type Error!
```

Union Types



Union Types



```
main.ts — Modern Web Development
main.ts M X
Day 1 > who-am-i > src > main.ts > ...
1 let ID: string | number;
2
3 // number
4 ID = 1;
5
6 // or string
7 ID = '001';
8
9 console.log(`The ID is ${ID}.`);
```

References

- <https://www.codecademy.com/learn/learn-typescript>
- <https://www.tutorialspoint.com/typescript/index.htm>

Thanks for listening...