Hoisting Variables

This lesson goes over the JavaScript principle of hoisting in TypeScript.

Before moving on, let's talk about the concept of *hoisting*. It is a quirk of JavaScript that brings all declarations made with `var` to the top of the function (or into the global scope if declared outside a function).

```
x = "not declared before assignment";
var x = "declared after assignment  and all fine";
console.log(x)
```

The code above compiles because `var x` goes above the two assignments. It looks like the following:

```
var x: string | undefined = undefined;
x = "not declared before assignment";
x = "declared after assignment and all fine";
console.log(x);
```

This peculiarity does not affect `let` or `const`. This means that if you are using `var`, you can use the variable and declare it later and the code will still work. This is, however, a bad practice that makes the code hard to follow. This ambiguity is solved by `let` and `const` if you use a variable that has not been declared first. The following code snippet does not compile because th variable declarations with `let` and `const` are after the assignments.

```
y = "not declared before assignments"; // Doesn't compile
let y = "The line before forbid this line";;

z = "not declared before assignments"; // Doesn't compile
const z = "The line before forbid this line";
```

The need to use `var` is now rarer since the inception of more strict `let` and `const`. Nevertheless, TypeScript can catch many errors like declaration and assignment on a codebase that uses `var`.

Back
Declaring Types in Untyped Code

Next
TypeScript Scope is JavaScript Scope

Completed

Report an Issue