# What is the difference between var and let in Typescript?

37

4

I submitted a question on stack overflow asking how I could stop the putTestQuestionResponses() function from executing IF a previous version was already executing.

The reply was to add in a processing flag which is here on line 2 of this code.

Can you tell me why use a "let" instead of a "var" here?

```
var promisePutTestQuestion;
let processing = false;

onEnter: ['$interval', 'questionService',
         ($interval, qus: IQuestionService) => {
  promisePutTestQuestion = $interval(() => {
     if (processing)
         return;
     processing = true;
     qus.putTestQuestionResponses()
     .then(() => processing = false)
  }, 5 * 1000);
}],
onExit: ['$interval', ($interval) => {
        $interval.cancel(promisePutTestQuestion);
}]
```

typescript

asked Feb 23 '16 at 8:53

Alan2
**1,836** 57 146 288

2 I recommend to read basarat.gitbooks.io/typescript/content/docs/let.html – Martin Vseticka Feb 23 '16 at 8:55

## 4 Answers

49

`var` declaration is function scoped and `let` declaration is block scoped.

See https://basarat.gitbooks.io/typescript/content/docs/let.html for more details.

answered Feb 23 '16 at 8:56
Martin Vseticka
**17.9k** 20 100 175

4 Coming from a C# background this seems weird. Should there not be a single keyword for variable declarations? Why would we have non-block scoped variables? Seems to make little sense.. but I am just starting to learn TypeScript. – Ross Mar 13 '18 at 6:32

3 `var` comes from JavaScript where it is a function scoped declaration. TypeScript does not have much of a choice to change the behavior. – Martin Vseticka Mar 13 '18 at 12:51

1 I see. I guess that is because TypeScript supports plain JavaScript as well. Make sense. So what you are saying is that it is best to avoid the var keyword and instead use the let keyword, at least when trying to design code in a similar style as C#, Java, etc, where all variables are scoped to their current scope? – Ross Mar 13 '18 at 19:49

3 Yes, `let` is preferred way. (TypeScript looks similar to C# but it's always better to reason about TypeScript in context of JavaScript as there are many fundamental differences between JavaScript and .NET languages (prototyping, static types, etc.)) – Martin Vseticka Mar 14 '18 at 11:27

23

example:

```
// demo: var
for(var i =0 ; i<5 ; i++){
   console.log(i)
}//finally i =5
console.log(i) // i=5
```

it is easy to know

edited Feb 16 '18 at 7:02
Martin Vseticka
17.9k 20 100 175

answered Feb 23 '16 at 9:02
wyl
239 1 2

13

`var` variables in JavaScript are function scoped. This is different from many other languages (C#, Java, etc.) where the variables are block scoped. If you bring a block scoped mindset to JavaScript, you would expect the following to print 123, instead it will print 456:

```
var foo = 123;
if (true) {
    var foo = 456;
}

console.log(foo); // 456
```

This is because `{` does not create a new variable scope. The variable `foo` is the same inside the if block as it is outside the if block. This is a common source of errors in JavaScript programming. This is why TypeScript (and ES6) introduces the `let` keyword to allow you to define variables with true block scope. That is, if you use `let` instead of `var`, you get a true unique element disconnected from what you might have defined outside the scope. The same example is demonstrated with `let`:

```
let foo = 123;
if (true) {
    let foo = 456;

}

console.log(foo); // 123
```

edited Jun 20 '17 at 3:22
John Montgomery
3,239 7 24 42

answered Jun 14 '17 at 7:07
chenchu kotari
171 1 2

```
    var x = 2;  // same variable!
    console.log(x);  // 2
  }
  console.log(x);  // 2
}

function letTest() {
  let x = 1;
  if (true) {
    let x = 2;  // different variable
    console.log(x);  // 2
  }
  console.log(x);  // 1
}
```

I found this here

> https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let

answered Jun 6 '18 at 6:20
Dharmendra Prajapati
**157** 13