TypeScript: Creating an empty typed container array

137

14

I am creating simple logic game called "Three of a Crime" in TypeScript.

When trying to pre-allocated typed array in TypeScript, I tried to do something like this:

```
var arr = Criminal[];
```

which gave the error "Check format of expression term" .

also tried doing this

```
var arr : Criminal = [];
```

and this produced "cannot convert any[] to 'Criminal'

what is the 'TypeScript' way to do this?

javascript arrays typescript

edited Feb 26 '18 at 5:02
c69
11k 5 40 75

asked Apr 5 '13 at 5:43
darethas
3,973 3 15 30

4 Answers

The issue of correctly pre-allocating a typed array in TypeScript was somewhat obscured for due to the array literal syntax, so it wasn't as intuitive as I first thought.

This will give you a correctly typed, empty array stored in the variable 'arr'

Hope this helps others!

answered Apr 5 '13 at 5:43
darethas
3,973 3 15 30

91

The existing answers missed an option, so here's a complete list:

```
// 1. Explicitly declare the type
var arr: Criminal[] = [];

// 2. Via type assertion
var arr = <Criminal[]>[];
var arr = [] as Criminal[];

// 3. Using the Array constructor
var arr = new Array<Criminal>();
```

1. Explicitly specifying the type is the general solution for whenever type inference fails for a variable declaration.
2. The advantage of using a type assertion (sometimes called a cast, but it's not really a cast in TypeScript) works for any expression, so it can be used even when no variable is declared. There are two syntaxes for type assertions, but only the latter will work in combination with JSX if you care about that.
3. Using the Array constructor is something that will only help you in this specific use case, but which I personally find the most readable. However, there is a slight performance impact at runtime*. Also, if someone were crazy enough to redefine the Array constructor, the meaning could change.

It's a matter of personal preference, but I find the third option the most readable. In the vast majority of cases the mentioned downsides would be negligible and readability is the most important factor.

*: Fun fact; at the time of writing the performance difference was 60% in Chrome, while in Firefox there was no measurable performance difference.

16

I know this is an old question but I recently faced a similar issue which couldn't be solved by this way, as I had to return an empty array of a specific type.

I had

```
return [];
```

where `[]` was `Criminal[]` type.

Neither `return: Criminal[] [];` nor `return []: Criminal[];` worked for me.

At first glance I solved it by creating a typed variable (as you correctly reported) just before returning it, but (I don't know how JavaScript engines work) it may create overhead and it's less readable.

For thoroughness I'll report this solution in my answer too:

```
let temp: Criminal[] = [];
return temp;
```

Eventually I found TypeScript type casting, which allowed me to solve the problem in a more concise and readable (and maybe efficient) way:

```
return <Criminal[]>[];
```

Hope this will help future readers!

edited May 23 '17 at 12:03
Community ♦
1 1

answered Aug 9 '16 at 8:46
Fylax
685 2 11 21

9 `[] as Criminal[]` works too. – Randy the Dev Dec 18 '16 at 8:50

answered Apr 22 at 14:21

rk13
19 1 5