How to implement class constants in typescript?

Asked 3 years, 5 months ago Active 3 months ago Viewed 226k times

350

32

In TypeScript, the `const` keyword cannot be used to declare class properties. Doing so causes the compiler to an error with "A class member cannot have the 'const' keyword."

I find myself in need to clearly indicate in code that a property should not be changed. I want the IDE or compiler to error if I attempt to assign a new value to the property once it has been declared. How do you guys achieve this?

I'm currently using a read-only property, but I'm new to Typescript (and JavaScript) and wonder whether there is a better way:

```
get MY_CONSTANT():number {return 10};
```

I'm using typescript 1.8. Suggestions?

PS: I'm now using typescript 2.0.3, so I've accepted David's answer

typescript class-constants

edited Jul 28 '18 at 12:37

Rohit Sharma
2,677 2 13 28

asked May 17 '16 at 0:32

BeetleJuice
23.7k 13 56 115

6 Answers

535

TypeScript 2.0 has the `readonly` modifier:

```
class MyClass {
    readonly myReadOnlyProperty = 1;

    myMethod() {
```

```
}

new MyClass().myReadOnlyProperty = 5; // error, readonly
```

It's not exactly a constant because it allows assignment in the constructor, but that's most likely not a big deal.

Alternative Solution

An alternative is to use the `static` keyword with `readonly`:

```
class MyClass {
    static readonly myReadOnlyProperty = 1;

    constructor() {
        MyClass.myReadOnlyProperty = 5; // error, readonly
    }

    myMethod() {
        console.log(MyClass.myReadOnlyProperty);
        MyClass.myReadOnlyProperty = 5; // error, readonly
    }
}

MyClass.myReadOnlyProperty = 5; // error, readonly
```

This has the benefit of not being assignable in the constructor and only existing in one place.

edited Apr 24 at 21:11

answered May 17 '16 at 0:59

David Sherret

60.5k 18 134 139

So you don't need a const or anything correct? – Jackie Nov 9 '16 at 21:50

24 To access the properties from outside the class, you'll need to add the `export` keyword before `class` as well as `public static` before the `readonly` keyword. See here: stackoverflow.com/a/22993349 – cbros2008 Nov 10 '16 at 14:10

Question. Was clueless why you need the class name to use that readOnly property inside the class itself? 'MyClass.myReadonlyProperty' – Saiyaff Farouk Mar 23 '17 at 6:00

@SaiyaffFarouk If I understand your question, the answer is that static properties exist as part of the class, not on an instance of the class. So, you access them using the class name not a variable which contains a class instance. – JeffryHouser Aug 18 '17 at 12:09

needless typing. It also makes the public members more distinct from ones marked as `private` or `protected`. Anyway, just my opinion :) – David Sherret Oct 18 '17 at 13:58

59

Constants can be declare outside of classes and use within your class. Otherwise the `get` property is a nice workaround

```
const MY_CONSTANT: string = "wazzup";

export class MyClass {

    public myFunction() {

        alert(MY_CONSTANT);
    }
}
```

answered May 17 '16 at 0:51

j3ff
2,626 1 19 29

6 Thanks; I'm worried about this implementation because it's not portable (in the model, the constant isn't actually part of the class) and it leaks info into the greater scope, but it has the advantage of being a real constant so I won't be able to change it without raising alarm bells. – BeetleJuice May 17 '16 at 1:00

1 I understand the concern and I find the use of `get` property very appropriate in your case – j3ff May 17 '16 at 1:04

3 Per angular.io/docs/ts/latest/guide/style-guide.html please use camel caase instead of upper case. Upper case for constants is not recommended. – Vadim Kirilchuk Mar 5 '17 at 18:14

10 Angular styleguide, not TypeScript styleguide.. Question was regarding TypeScript specifically – VeldMuijz Jun 19 '17 at 7:13

1 @BeetleJuice - does this actually leak into the greater scope? I can't seem to reference it from anywhere else. – andrewpm Jun 22 '17 at 10:44

31

You can mark properties with `readonly` modifier in your declaration:

```
export class MyClass {
```

@see TypeScript Deep Dive book - Readonly

answered Jul 6 '17 at 14:07

am0wa

3,175 17 13

8

Angular 2 Provides a very nice feature called as Opaque Constants. Create a class & Define all the constants there using opaque constants.

```
import { OpaqueToken } from "@angular/core";

export let APP_CONFIG = new OpaqueToken("my.config");

export interface MyAppConfig {
    apiEndpoint: string;
}

export const AppConfig: MyAppConfig = {
    apiEndpoint: "http://localhost:8080/api/"
};
```

Inject it in providers in app.module.ts

You will be able to use it across every components.

EDIT for Angular 4 :

For Angular 4 the new concept is Injection Token & Opaque token is Deprecated in Angular 4.

Injection Token Adds functionalities on top of Opaque Tokens, it allows to attach type info on the token via TypeScript generics, plus Injection tokens, removes the need of adding @Inject

Example Code

Angular 2 Using Opaque Tokens

```
providers: [
  {
    provide: DataService,
    useFactory: (http, apiUrl) => {
      // create data service
    },
    deps: [
      Http,
      new Inject(API_URL) //notice the new Inject
    ]
  }
]
```

Angular 4 Using Injection Tokens

```
const API_URL = new InjectionToken<string>('apiUrl'); // generic defines return value of
injector


providers: [
  {
    provide: DataService,
    useFactory: (http, apiUrl) => {
      // create data service
    },
    deps: [
      Http,
      API_URL // no `new Inject()` needed!
    ]
  }
]
```

Injection tokens are designed logically on top of Opaque tokens & Opaque tokens are deprecated in Angular 4.

edited Jun 20 '17 at 5:33

answered Jan 19 '17 at 14:07

Parth Ghiya

5,487 2 21 31

6 plus one. Angular is as stable as a 13 year old teenager. they gets features deprecated a few months after releasing them. petty. – Stavm Oct 7 '17 at 12:54

4

Either use readOnly modifier with the constant one needs to declare or one might declare a constant outside the class and use it specifically only in the required class using get operator.

answered Aug 18 '17 at 4:41

Krishna Ganeriwal
1,077 9 14

1

For this you can use the `readonly` modifier. Object properties which are `readonly` can only be assigned during initialization of the object.

Example in classes:

```
class Circle {
  readonly radius: number;

  constructor(radius: number) {
    this.radius = radius;
  }

  get area() {
    return Math.PI * this.radius * 2;
  }
}

const circle = new Circle(12);
circle.radius = 12; // Cannot assign to 'radius' because it is a read-only property.
```

Example in Object literals:

```
type Rectangle = {
  readonly height: number;
  readonly width: number;
};

const square: Rectangle = { height: 1, width: 2 };
square.height = 5 // Cannot assign to 'height' because it is a read-only property
```

It's also worth knowing that the `readonly` modifier is purely a typescript construct and when the TS is compiled to JS the construct will

answered Jul 10 at 17:10

Willem van der Veen
8,184 4 59 57