angular typescript interface default value

Asked 7 months ago Active 5 months ago Viewed 2k times

3

2

My interface :

```
export interface MapMarker{
    longitude: number,
    latitude: number,
    popupText: string
}
```

My component

```
mapMarker: MapMarker;

ngOnInit() {
    this.mapMarker.latitude = location.coords.latitude;
    this.mapMarker.longitude = location.coords.longitude;
    this.mapMarker.popupText = "Your current Location"
  }
```

this tells me that can't set latitude of undefined. I know why is that. It's because mapMarker has a type MapMarker but is undefined, because it's undefined by default. What should I do? if I try `mapMarker: MapMarker = null;` still wrong.

I also tried `mapMarker: MapMarker = {};` which gives me following error = `Type '{}' is missing the following properties from type 'MapMarker': longitude, latitude, popupText`

I need the solution, but i don't want to use 'any' type.

angular typescript

10.1k 3 10 32 1,370 3 22

What TS is telling you *is correct*, an empty object isn't a MapMarker. Why not create the object in that method too? – jonrsharpe Feb 24 at 22:03

then i still have to use any. – Nika Kurashvili Feb 24 at 22:07

let data = {}; data.latitude = location.coords.latitude; this is also the error. so let data:any = {}; this works. but it's bad. – Nika Kurashvili Feb 24 at 22:07

Why don't you include the properties when you create it? – jonrsharpe Feb 24 at 22:09

then why the hell do I need interface at all? :D – Nika Kurashvili Feb 24 at 22:11

5 Answers

6

If you don't want to create an object and assign default values every time, use a class with default values for its properties instead. If you don't specify the values, defaults values will be used according to your property types.

```
export class MapMarker {
    longitude: number = 0;
    latitude: number = 0;
    popupText: string = '';
}
```

Then you can simply create an instance of your class and every field will be correctly initialized:

```
mapMarker: MapMarker = new MapMarker();
```

If you want to keep your interface, just implement it in the class:

```
export interface IMapMarker {
    longitude: number;
    latitude: number;
    popupText: string;
}

export class MapMarker implements IMapMarker {
    longitude: number = 0;
```

If some of your properties are optional, use the optional operator for them:

```
export interface IMapMarker {
    longitude?: number;
    latitude?: number;
    popupText?: string;
}
```

But then, you will always have to make sure the property is not null before using it.

edited Apr 23 at 11:34

answered Feb 24 at 22:22
jo_va
10.1k 3 10 32

2

Classes and interfaces are powerful structures that facilitate not just object-oriented programming but also type-checking in TypeScript. A class is a blueprint from which we can create objects that share the same configuration properties and methods. An interface is a group of related properties and methods that describe an object, but neither provides implementation nor initialization for them.

Since both of these structures define what an object looks like, both can be used in TypeScript to type our variables. The decision to use a class or an interface truly depends on our use case: type-checking only, implementation details (typically via creating a new instance), or even both! We can use classes for type-checking and the underlying implementation

if you are interested only in type-checking the responses Interface is a good choice Furthermore, an interface is a virtual structure that only exists within the context of TypeScript. The TypeScript compiler uses interfaces solely for type-checking purposes. Once your code is transpiled to its target language, it will be stripped from its interfaces - JavaScript isn’t typed, there’s no use for them there.

Since interfaces do not exist in runtime there is no runtime cost!

Regarding the issue you are facing:
Use this

Vikas
7,112 5 21 44

nice explanation @Vikas – Ankush Jain Feb 25 at 12:36

1

Use `?` for optional properties in typescript like following. I hope this is what you want.

```
export interface MapMarker{
    longitude?: number,
    latitude?: number,
    popupText?: string
}
```

answered Feb 24 at 22:25

Rakesh
1,709 1 8 21

1

There are at least two options:

1 Make every property of MapMarker optional

```
export interface MapMarker{
   longitude?: number,
   latitude?: number,
   popupText?: string
}
```

2 Use class

```
export class MapMarker {
    longitude: number = 0;
    latitude: number = 0;
    popupText: string = '';
```

```
mapMarker: MapMarker =  {
    longitude: number = 0;
    latitude: number = 0;
    popupText: string = '';
}
```

answered Feb 24 at 22:27

Lends
944 1 8 21

0

Simply make use of **Type Assertion** like below

```
mapMarker: MapMarker = {} as MapMarker;
```

or

```
mapMarker: MapMarker = <MapMarker>{};
```

This will tell the TypeScript compiler to treats this empty object `{}` as an object of `MapMarker` interface.

Note that this will just prevent you from compile-time error, not runtime.

Eventually, you have to add properties (i.e. latitude, longitude etc.) in your variable `mapMarker` before those properties are being used by rest part of the application.

edited Feb 25 at 5:54

answered Feb 25 at 5:49

Ankush Jain
2,828 2 15 32