Using TypeScript super()

Asked 3 years, 2 months ago Active 3 years, 2 months ago Viewed 18k times

18

1

I am trying to extend a class in TypeScript. I keep receiving this error on compile: 'Supplied parameters do not match any signature of call target.' I have tried referencing the artist.name property in the super call as super(name) but is not working.

Any ideas and explanations you may have will be greatly appreciated. Thanks - Alex.

```
class Artist {
  constructor(
    public name: string,
    public age: number,
    public style: string,
    public location: string
  ){
    console.log(`instantiated ${name}, whom is ${age} old, from ${location}, and heavily
regarded in the ${style} community`);
  }
}

class StreetArtist extends Artist {
  constructor(
    public medium: string,
    public famous: boolean,
    public arrested: boolean,
    public art: Artist
  ){
    super();
    console.log(`instantiated ${this.name}. Are they famous? ${famous}. Are they locked
up? ${arrested}`);
  }
}

interface Human {
  name: string,
  age: number
}

function getArtist(artist: Human){
```

```
let Banksy = new Artist(
  "Banksy",
   40,
  "Politcal Graffitti",
  "England / Wolrd"
)

getArtist(Banksy);
```

javascript oop typescript

asked Jun 22 '16 at 2:08

alex bennett
366 1 5 14

1 Answer

21

The super call must supply all parameters for base class. The constructor is not inherited. Commented out artist because I guess it is not needed when doing like this.

```
class StreetArtist extends Artist {
  constructor(
    name: string,
    age: number,
    style: string,
    location: string,
    public medium: string,
    public famous: boolean,
    public arrested: boolean,
    /*public art: Artist*/
  ){
    super(name, age, style, location);
    console.log(`instantiated ${this.name}. Are they famous? ${famous}. Are they locked
up? ${arrested}`);
  }
}
```

Or if you intended the art parameter to populate base properties, but in that case I guess there isn't really a need for using public on art parameter as the properties would be inherited and it would only store duplicate data.

```
class StreetArtist extends Artist {
  constructor(
    public medium: string,
    public famous: boolean,
    public arrested: boolean,
    /*public */art: Artist
  ){
    super(art.name, art.age, art.style, art.location);
    console.log(`instantiated ${this.name}. Are they famous? ${famous}. Are they locked
up? ${arrested}`);
  }
}
```

edited Jun 22 '16 at 8:11

answered Jun 22 '16 at 2:19

mollwe

1,590 1 13 16

If you add pubic to each of constructors arguments this argument will be assigned in child as well as parent – Morteza Tourani Jun 22 '16 at 2:35

I did intend to populate the base class with art: Artist. The second solution worked seamlessly. Thank you very much. – alex bennett Jun 22 '16 at 2:44

Happy to be able to help. @mortezaT you are right it meant to have the first four arguments without public. I don't know what will happen if you cast StreetArtist to Artist and access name for example, will they be the same? It hides base property right? – mollwe Jun 22 '16 at 8:09

Worth noting on second solution. You are basically creating two artist objects with same information. Coming from c# world it seems a bit off for me, but it works :) For typescript maybe I would have used an interface for the art argument instead and use an object initializer as `{ name: 'aoue', ... }` that follows the interface. – mollwe Jun 22 '16 at 8:18