Assigning Typescript constructor parameters

7

3

I have interface :

```
export interface IFieldValue {
    name: string;
    value: string;
}
```

And I have class that implements it :

```
class Person implements IFieldValue{
    name: string;
    value: string;
    constructor (name: string, value: string) {
        this.name = name;
        this.value = value;
    }
}
```

after reading this post I've thinking about refactoring :

```
class Person implements IFieldValue{
    constructor(public name: string, public value: string) {
    }
}
```

Question : In first class I have fields which by default should be as `private` . In second sample I can only set them as `public` . Is it all correct or my understanding of default Access modifiers in TypeScript?

typescript access-modifiers

1 Answer

17

> Public by default. TypeScript Documentation

In following definition

```
class Person implements IFieldValue{
    name: string;
    value: string;
    constructor (name: string, value: string) {
        this.name = name;
        this.value = value;
    }
}
```

Attributes `<Person>.name` and `<Person>.value` are public by default.

as they are here

```
class Person implements IFieldValue{
    constructor(public name: string, public value: string) {
        this.name = name;
        this.value = value;
    }
}
```

Beware: Here is an incorrect way of doing it, since `this.name` and `this.value` will be regarded as not defined in the constructor.

```
class Person implements IFieldValue{
    constructor(name: string, value: string) {
        this.name = name;
        this.value = value;
    }
}
```

```
    private value: string;
    constructor (name: string, value: string) {
        this.name = name;
        this.value = value;
    }
}
```

or equivalently

```
class Person implements IFieldValue{
    constructor (private name: string, private value: string) {}
}
```

which in my opinion is the most preferable way that avoids redundancy.

edited Jul 8 '17 at 10:43

answered Jan 29 '17 at 16:30

Maciej Caputa
1,241 7 15

1 If `Person` implements `IFieldValue` which has public properties 'name' and 'value' then `name` and `value` must remain `public` in the `Person` class. The two code examples you provided do not compile with TypeScript 2.x. You either change the interface and make props private or you could have private props `personName` and `personValue` like: `class Person implements IFieldValue{ private personName: string; private personValue: string; constructor (public name: string, public value: string) { this.personName = name; this.personValue = value; } }` – phil_lgr Mar 6 '17 at 21:32

sorry disregard the part where I said: *"You either change the interface and make props private"* makes no sense since props are only public on an interface – phil_lgr Mar 6 '17 at 21:44