What are getters and setters for in ECMAScript 6 classes?

Asked 5 years, 1 month ago Active 1 year, 10 months ago Viewed 73k times

100

27

I am confused as to what the point of getters and setters are in ECMAScript 6 classes. What is the purpose? Below is an example I am referring to:

```
class Employee {

    constructor(name) {
        this._name = name;
    }

    doWork() {
        return `${this._name} is working`;
    }

    get name() {
        return this._name.toUpperCase();
    }

    set name(newName){
        if(newName){
            this._name = newName;
        }
    }
}
```

class ecmascript-6 setter getter

edited Apr 26 '16 at 10:14
dakab
4,057 8 33 52

asked Jan 29 '15 at 18:48
TruMan1
23.3k 43 131 253

1 It's similar to those in C#, if you happen to know about it. – Arturo Torres Sánchez Jan 29 '15 at 18:51

Related: What is the argument for using ES6 getters and setters over getProperty/setProperty convention? – Bergi Jul 7 '16 at 6:23

cause a stack overflow"... It also speaks of the variable not truly being 'private', but there are numerous new ways to create private vars in JS classes; my favorite is just using Typescript, but I've used the Symbol approach too – webdevinci Jul 11 '17 at 15:16

3 Answers

Active | Oldest | Votes

106

These setter and getter allow you to use the properties directly (without using the parenthesis)

```
var emp = new Employee("TruMan1");

if (emp.name) {
  // uses the get method in the background
}

emp.name = "New name"; // uses the setter in the background
```

This is only to set and get the value of the property.

edited Jun 2 '17 at 22:46

Alejandro B.
3,901 2 29 48

answered Jan 29 '15 at 18:54

David Laberge
10.9k 14 48 80

1 Did You mean property instead of attribute? Bit confusing for me – Krizzu Oct 25 '16 at 7:29

Good eye, @Krizzu. Attributes exist in JavaScript and are *completely* different things than properties. The answer does indeed refer to properties and not attributes. I've edited the answer. I don't think the answerer will mind. :) – Ray Toal Nov 6 '16 at 4:06

I'm not quite sure this really is such an advantage, it somehow hides the notion of using setters/getters. A client of a class may think it directly uses properties, where it's not appopriate, but I agree it adheres the information/detail hiding principle. Maybe if we use this consequently it makes usage easier and I just have to get used to it more... – Christof Kälin Jun 6 '17 at 11:02

Can you pass multiple parameters in a setter if so how do you use it? @David Laberge – Vignesh S Jul 26 '17 at 2:25

If you want to create setters and getters manually here's a good example from coryrylan.com/blog/javascript-es6-class-syntax Set: `set name(newName) { this._name = newName; }` Get: `get name() { return this._name.toUpperCase(); }` – Jim Doyle May 23 '18 at 23:29

Getters and setters in ES6 serve the same purpose that they do in other languages ... including ES5. ES5 already allows getters and

48

Effectively, getters and setters allow you to use standard property access notation for reads and writes while still having the ability to customize how the property is retrieved and mutated without needed explicit getter and setter methods.

In the Employee class above, this would mean you could access the `name` property like this:

```
console.log(someEmployee.name);
```

It would *look* like a normal property access, but it would actually call `toUpperCase` on the name before returning it. Similarly, doing this:

```
someEmployee.name = null;
```

would access the setter, and it would not modify the internal `_name` property because of the guard clause introduced in `name`'s setter.

See also the general question Why use getters and setters? for more information about why being able to modify the functionality of member access is useful.

edited May 23 '17 at 12:10
Community ♦
1 1

answered Jan 29 '15 at 18:53
Alexis King
38.2k 12 111 186

3

ES6 getters and setters have a substantially different motivation than similar concepts in Java.

In Java, getters and setters allow a class to define a JavaBean. The point of getters and setters is that it allows the bean to have a completely orthogonal "interface" from that implied by public fields. So I can have a field "name" that is is NOT a JavaBean property, and I can have a JavaBean property "address" that is NOT a field.

JavaBean properties are also "discoverable" by thousands of frameworks (Hibernate for example) via Java reflection. Thus, getters and setters are part of a standard method for "exposing" bean properties.

Getters and setters, being functions, also have the value that they "abstract" the implementation. It can be EITHER a field or a computed ("synthetic") value. So if I have a bean property called "zipcode", that starts out as stored string. Now suppose I want to change it to be a value computed from address/city/state?

If I use a field, this code breaks:

But if I use a getter, this does not break:

```
String zipcode = address.getZipcode();
```

JavaScript doesn't have anything like JavaBeans. So far as I've read, the intended value of GET and SET is limited to the aforementions "synthetic" (computed) properties.

But it's somewhat better than java in that while Java doesn't allow you to compatibly convert a "field" to a method, ES6 GET and SET allows that.

That is, if I have:

```
var zipcode = address.zipcode;
```

If I change zipcode from being a standard object property to a getter, the above code now calls the GET function.

Note that if I didn't include GET in the definition, this would NOT invoke the zipcode GET method. Instead, it would merely assign the function zipcode to the var.

So I think these are some important distinctions to understand betweeen Java and JavaScript ES6 getters and setters.

answered Jan 5 '18 at 15:59

DaBlick

717 8 15