JavaScript Object Accessors

‹ Previous Next ›

JavaScript Accessors (Getters and Setters)

ECMAScript 5 (2009) introduced Getter and Setters.

Getters and setters allow you to define Object Accessors (Computed Properties).

JavaScript Getter (The get Keyword)

This example uses a **lang** property to **get** the value of the **language** property.

Example

```
// Create an object:
var person = {
    firstName: "John",
    lastName : "Doe",
    language : "en",
    get lang() {
        return this.language;
    }
};
// Display data from the object using a getter:
document.getElementById("demo").innerHTML = person.lang;
```

Try it Yourself »

JavaScript Setter (The set Keyword)

This example uses a **lang** property to **set** the value of the **language** property.

Example

```
var person = {
    firstName: "John",
    lastName : "Doe",
    language : "",
    set lang(lang) {
        this.language = lang;
    }
};
// Set an object property using a setter:
person.lang = "en";
// Display data from the object:
document.getElementById("demo").innerHTML = person.language;
```

Try it Yourself »

JavaScript Function or Getter?

What is the differences between these two examples?

Example 1

```
var person = {
    firstName: "John",
    lastName : "Doe",
    fullName : function() {
        return this.firstName + " " + this.lastName;
    }
};
// Display data from the object using a method:
document.getElementById("demo").innerHTML = person.fullName();
```

Try it Yourself »

Example 2

```
var person = {
    firstName: "John",
    lastName : "Doe",
    get fullName() {
        return this.firstName + " " + this.lastName;
    }
};
// Display data from the object using a getter:
document.getElementById("demo").innerHTML = person.fullName;
```

Try it Yourself »

Example 1 access fullName as a function: person.fullName().

Example 2 access fullName as a property: person.fullName.

The second example provides simpler syntax.

Data Quality

JavaScript can secure better data quality when using getters and setters.

Using the **lang** property, in this example, returns the value of the **language** property in upper case:

Example

```
// Create an object:
var person = {
    firstName: "John",
    lastName : "Doe",
    language : "en",
    get lang() {
```

```
        return this.language.toUpperCase();
    }
};
// Display data from the object using a getter:
document.getElementById("demo").innerHTML = person.lang;
```

Try it Yourself »

Using the **lang** property, in this example, stores an upper case value in the **language** property:

Example

```
var person = {
    firstName: "John",
    lastName : "Doe",
    language : "",
    set lang(lang) {
        this.language = lang.toUpperCase();
    }
};
// Set an object property using a setter:
person.lang = "en";
// Display data from the object:
document.getElementById("demo").innerHTML = person.language;
```

Try it Yourself »

Why Using Getters and Setters?

- It gives simpler syntax
- It allows equal syntax for properties and methods
- It can secure better data quality
- It is useful for doing things behind-the-scenes

A Counter Example

Example

```
var obj = {
    counter : 0,
    get reset() {
      this.counter = 0;
    },
    get increment() {
      this.counter++;
    },
    get decrement() {
      this.counter--;
    },
    set add(value) {
      this.counter += value;
    },
    set subtract(value) {
      this.counter -= value;
    }
};
// Play with the counter:
obj.reset;
obj.add = 5;
obj.subtract = 1;
obj.increment;
obj.decrement;
```

Try it Yourself »

Object.defineProperty()

The Object.defineProperty() method can also be used to add Getters and Setters:

Example

```
// Define object
var obj = {counter : 0};

// Define setters
Object.defineProperty(obj, "reset", {
    get : function () {this.counter = 0;}
});
Object.defineProperty(obj, "increment", {
    get : function () {this.counter++;}
});
Object.defineProperty(obj, "decrement", {
    get : function () {this.counter--;}
});
Object.defineProperty(obj, "add", {
    set : function (value) {this.counter += value;}
});
Object.defineProperty(obj, "subtract", {
    set : function (value) {this.counter -= value;}
});

// Play with the counter:
obj.reset;
obj.add = 5;
obj.subtract = 1;
obj.increment;
obj.decrement;
```

Try it Yourself »

Browser Support

Getters and Setters are not supported in Internet Explorer 8 or earlier:

Yes	9.0	Yes	Yes	Yes

‹ Previous

Next ›