



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

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

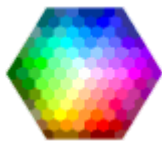
Getters and Setters are not supported in Internet Explorer 8 or earlier:

Yes	9.0	Yes	Yes	Yes

[< Previous](#)

[Next >](#)

COLOR PICKER



HOW TO

- Tabs
- Dropdowns
- Accordions
- Side Navigation
- Top Navigation
- Modal Boxes
- Progress Bars
- Parallax
- Login Form
- HTML Includes
- Google Maps
- Range Sliders
- Tooltips
- Slideshow
- Filter List
- Sort List

SHARE



CERTIFICATES

HTML
CSS
JavaScript
SQL
Python
PHP
jQuery
Bootstrap
XML

[Read More »](#)

[REPORT ERROR](#)

[PRINT PAGE](#)

[FORUM](#)

[ABOUT](#)

Top Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
jQuery Tutorial
Java Tutorial
C++ Tutorial

Top References

- [HTML Reference](#)
- [CSS Reference](#)
- [JavaScript Reference](#)
- [SQL Reference](#)
- [Python Reference](#)
- [W3.CSS Reference](#)
- [Bootstrap Reference](#)
- [PHP Reference](#)
- [HTML Colors](#)
- [jQuery Reference](#)
- [Java Reference](#)
- [Angular Reference](#)

Top Examples

- [HTML Examples](#)
- [CSS Examples](#)
- [JavaScript Examples](#)
- [How To Examples](#)
- [SQL Examples](#)
- [Python Examples](#)
- [W3.CSS Examples](#)
- [Bootstrap Examples](#)
- [PHP Examples](#)
- [jQuery Examples](#)
- [Java Examples](#)
- [XML Examples](#)

Web Certificates

- [HTML Certificate](#)
- [CSS Certificate](#)
- [JavaScript Certificate](#)
- [SQL Certificate](#)
- [Python Certificate](#)
- [jQuery Certificate](#)
- [PHP Certificate](#)
- [Bootstrap Certificate](#)
- [XML Certificate](#)

[Get Certified »](#)

W3Schools is optimized for learning, testing, and training. Examples might be simplified to improve reading and basic understanding. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using this site, you agree to have read and accepted our terms of use, cookie and privacy policy. Copyright 1999-2020 by Refsnes Data. All Rights Reserved.

Powered by W3.CSS.



