JavaScript Object Methods

Example

```
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

Try it Yourself »

The **this** Keyword

In a function definition, `this` refers to the "owner" of the function.

In the example above, `this` is the **person object** that "owns" the **fullName** function.

In other words, **this.firstName** means the **firstName** property of **this object**.

Read more about the `this` keyword at JS this Keyword.

JavaScript Methods

JavaScript methods are actions that can be performed on objects.

A JavaScript **method** is a property containing a **function definition**.

Property	Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

Methods are functions stored as object properties.

Accessing Object Methods

You access an object method with the following syntax:

```
objectName.methodName()
```

You will typically describe fullName() as a method of the person object, and fullName as a property.

The fullName property will execute (as a function) when it is invoked with ().

This example accesses the fullName() **method** of a person object:

Example

```
name = person.fullName();
```

Try it Yourself »

If you access the fullName **property**, without (), it will return the **function definition**:

Example

```
name = person.fullName;
```

Try it Yourself »

Using Built-In Methods

This example uses the `toUpperCase()` method of the String object, to convert a text to uppercase:

```
var message = "Hello world!";
var x = message.toUpperCase();
```

The value of x, after execution of the code above will be:

```
HELLO WORLD!
```

Adding a Method to an Object

Adding a new method to an object is easy:

Example

```
person.name = function () {
  return this.firstName + " " + this.lastName;
};
```

Try it Yourself »

❮ Previous   Next ❯