☰ 🏠 HTML CSS JAVASCRIPT MORE ▾ REFERENCES ▾ 🌐 🔍

The JavaScript **this** Keyword

❮ Previous   Next ❯

Example

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

Try it Yourself »

What is **this**?

The JavaScript `this` keyword refers to the object it belongs to.

It has different values depending on where it is used:

In a method, `this` refers to the **owner object**.

Alone, `this` refers to the **global object**.

In a function, `this` refers to the **global object**.

In a function, in strict mode, `this` is `undefined`.

In an event, `this` refers to the **element** that received the event.

Methods like `call()`, and `apply()` can refer `this` to **any object**.

this in a Method

In an object method, `this` refers to the "**owner**" of the method.

In the example on the top of this page, `this` refers to the **person** object.

The **person** object is the **owner** of the **fullName** method.

```
fullName : function() {
  return this.firstName + " " + this.lastName;
}
```

Try it Yourself »

this Alone

When used alone, the **owner** is the Global object, so `this` refers to the Global object.

In a browser window the Global object is `[object Window]` :

Example

```
var x = this;
```

Try it Yourself »

In **strict mode**, when used alone, `this` also refers to the Global object `[object Window]` :

Example

```
"use strict";
var x = this;
```

Try it Yourself »

this in a Function (Default)

In a JavaScript function, the owner of the function is the **default** binding for `this` .

So, in a function, `this` refers to the Global object `[object Window]` .

Example

```
function myFunction() {
  return this;
}
```

Try it Yourself »

this in a Function (Strict)

JavaScript **strict mode** does not allow default binding.

So, when used in a function, in strict mode, `this` is `undefined` .

Example

```
"use strict";
function myFunction() {
  return this;
}
```

Try it Yourself »

this in Event Handlers

In HTML event handlers, `this` refers to the HTML element that received the event:

Example

```
<button onclick="this.style.display='none'">
  Click to Remove Me!
</button>
```

Try it Yourself »

Object Method Binding

In these examples, `this` is the **person** object (The person object is the "owner" of the function):

Example

```
var person = {
  firstName  : "John",
  lastName   : "Doe",
```

```
    id        : 5566,
    myFunction : function() {
      return this;
    }
};
```

Try it Yourself »

Example

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

Try it Yourself »

In other words: **this.firstName** means the **firstName** property of **this** (person) object.

Explicit Function Binding

The `call()` and `apply()` methods are predefined JavaScript methods.

They can both be used to call an object method with another object as argument.

> You can read more about `call()` and `apply()` later in this tutorial.

In the example below, when calling person1.fullName with person2 as argument, `this` will refer to person2, even if it is a method of person1:
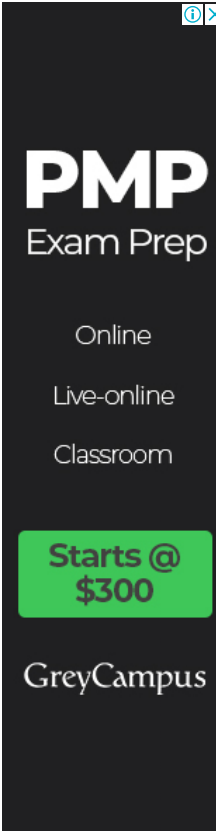
Example

```
var person1 = {
    fullName: function() {
      return this.firstName + " " + this.lastName;
    }
}
var person2 = {
    firstName:"John",
    lastName: "Doe",
}
person1.fullName.call(person2);  // Will return "John Doe"
```
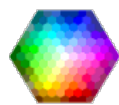
Try it Yourself »

COLOR PICKER

HOW TO

Tabs
Dropdowns
Accordions
Side Navigation
Top Navigation
Modal Boxes
Progress Bars
Parallax
Login Form
HTML Includes