The JavaScript **this** Keyword

Example

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

Try it Yourself »

What is **"this"**?

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

In the example above, **this** refers to the **person** object.

The **person** object "owns" the **fullName** method.

Default Binding

When used alone, **this** refers to the **Global object**.

In a browser the Global object is [**object Window**]:

Example

```
var x = this;
```

Try it Yourself »

When used in a function, **this** refers to the **Global object**.

Example

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

Try it Yourself »

In strict mode, **this** will be **undefined**, because strict mode does not allow default binding:

Example

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

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 an other object as argument.

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

In this example, 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 »

‹ Previous

Next ›