O JavaScript **esta** palavra - chave

‹ Anterior Próximo ›

Exemplo

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

Tente você mesmo "

O que é **isso** ?

A `this` palavra-chave JavaScript se refere ao objeto ao qual pertence.

Tem valores diferentes dependendo de onde é usado:

Em um método, `this` refere-se ao **objeto proprietário** .

Sozinho, `this` refere-se ao **objeto global** .

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

```
let x = this;
```

Try it Yourself »

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

Example

```
"use strict";
let 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

```
const person = {
  firstName  : "John",
  lastName   : "Doe",
  id         : 5566,
  myFunction : function() {
    return this;
  }
};
```

Try it Yourself »

Example

```
const 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.

No exemplo abaixo, ao chamar person1.fullName com person2 como argumento, `this` irá se referir a person2, mesmo que seja um método de person1:

Exemplo

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

Tente você mesmo "

‹ Anterior Próximo ›