







HTML CSS







# The JavaScript this Keyword

**<** Previous

Next >

#### Example

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

Try it Yourself »

### What is this?

In JavaScript, the this keyword refers to an object.

**Which** object depends on how this is being invoked (used or called).

The this keyword refers to different objects depending on how it is used:

In an object method, this refers to the 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(), apply(), and bind() can refer this to any object.
```

### Note

this is not a variable. It is a keyword. You cannot change the value of this.

### this in a Method

When used in an object method, this refers to the **object**.

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

Because the **fullName** method is a method of the **person** object.

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

Try it Yourself »

### this Alone

When used alone, this refers to the global object.

Because this is running in the global scope.

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**:

### Example

```
"use strict";
let x = this;
Try it Yourself »
```

# this in a Function (Default)

In a function, the **global object** is the default binding for this.

In a browser window the global object is [object Window]:

#### Example

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

Try it Yourself »

**ADVERTISEMENT** 

# 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:

#### 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 »

i.e. **this.firstName** is the **firstName** property of **this** (the 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.

#### See Also:

The Function call() Method

The Function apply() Method

The Function bind() Method

The example below calls person1.fullName with person2 as an argument, **this** refers to person2, even if fullName is a method of person1:

#### Example

```
const person1 = {
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
}

const person2 = {
  firstName:"John",
  lastName: "Doe",
}

// Return "John Doe":
person1.fullName.call(person2);
```

Try it Yourself »

# **Function Borrowing**

With the bind() method, an object can borrow a method from another object.

This example creates 2 objects (person and member).

The member object borrows the fullname method from the person object:

#### Example

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

const member = {
  firstName:"Hege",
  lastName: "Nilsen",
}

let fullName = person.fullName.bind(member);
```

Try it Yourself »

### This Precedence

To determine which object this refers to; Use the following precedence of order.

Precedence	Object
1	bind()
2	apply() and call()
3	Object method
4	Global scope

Is this in a function being called using bind()?

Is this in a function is being called using apply()?

Is this in a function is being called using call()?

Is this in an object function (method)?

Is this in a function in the global scope.

**∢** Previous

Next >

**ADVERTISEMENT** 

NEW

We just launched W3Schools videos



**Explore** now

#### **COLOR PICKER**





Get certified by completing a JavaScript course today!



**Get started** 

**CODE GAME** 



Play Game

ADVERTISEMENT

#### **ADVERTISEMENT**

#### Report Error

Forum

**About** 

Shop

#### **Top Tutorials**

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

C++ Tutorial jQuery Tutorial

#### **Top References**

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
Angular Reference
jQuery Reference

#### **Top Examples**

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

#### **Web Courses**

HTML Course
CSS Course
JavaScript Course
Front End Course
SQL Course
Python Course
PHP Course
jQuery Course
Java Course
C++ Course
C# Course
XML Course

Get Certified »

W3Schools is optimized for learning and training. Examples might be simplified to improve reading and learning. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using W3Schools, you agree to have read and accepted our terms of use, cookie and privacy policy.

Copyright 1999-2022 by Refsnes Data. All Rights Reserved.

W3Schools is Powered by W3.CSS.

