

3000+ Browsers, Devices & OS.

[Sign Up](#)

Test Across 3000+ Real Devices, Browsers And OSs With Lambdatest

LambdaTest Inc.

Function Inside Object in JavaScript

In JavaScript, functions can be defined inside objects, just like any other property. These functions are called "methods" and can be invoked by using the object's name, followed by the method name and parentheses.

For example:

```
const object = {
  method: function() {
    console.log("Hello, I'm a method!");
  }
};

object.method(); // prints "Hello, I'm a method!"
```

You can also define methods using the **shorthand notation**, which uses the `function` keyword.

3000+ Browsers, Devices & OS.

[Sign I](#)

Lambdatest Is The Fastest, Most
Reliable And Scalable Test
Execution Platform

LambdaTest Inc.

```
const object = {
  method() {
    console.log("Hello, I'm a method!");
  }
};

object.method(); // prints "Hello, I'm a method!"
```

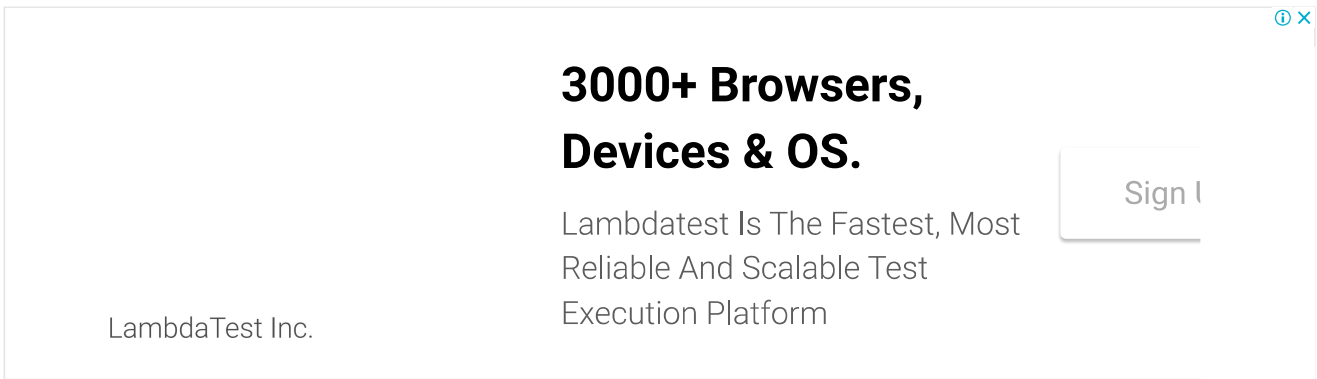
You can also define methods using the **arrow function notation**, which is similar to the shorthand notation but uses the `=>` instead of `function` keyword

```
const object = {
  method: () => {
    console.log("Hello, I'm a method!");
  }
};

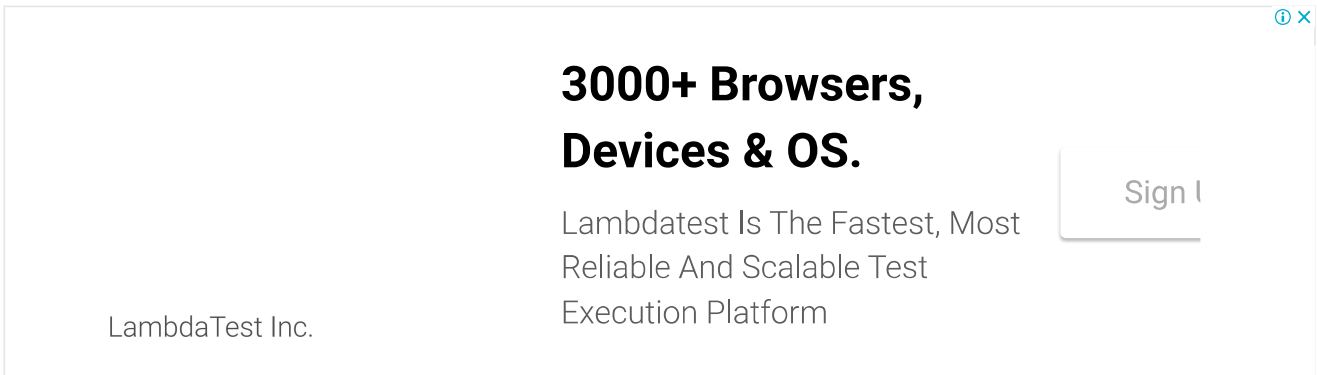
object.method(); // prints "Hello, I'm a method!"
```

World's Leading Testing Tool

Lambdatest Is The Fastest, Most Reliable And Scalable Test Execution Platform LambdaTest Inc.



Inside the method, you can access the properties of the object using the `this` keyword.



```
const object = {  
  property: "I'm a property",  
  method: function() {  
    console.log(this.property);  
  }  
};  
  
object.method(); // prints "I'm a property"
```

Methods can also accept parameters and return values just like regular functions:

```
const object = {  
  method: function(a, b) {  
    return a + b;  
  }  
};  
  
console.log(object.method(1, 2)); // prints 3
```

You can also add methods to an object after it has been created by assigning a new method to an existing property, or by creating a new property on the object and assigning a method to it.

In javascript you can use `class` keyword to create objects with methods and properties, it follows the OOP concepts.

```
class Object {  
  constructor(){  
    this.property = "I'm a property";  
  }  
  method(){  
    console.log("Hello, I'm a method!");  
  }  
}  
  
const obj = new Object();  
console.log(obj.property);  
obj.method();
```



3000+ Browsers,
Devices & OS.

Lambdatest Is The Fastest, Most
Reliable And Scalable Test
Execution Platform

Sign In

LambdaTest Inc.

Test Your Digital Assets

Test Across 3000+ Real Devices, Browsers And OSs With Lambdatest

Overall, methods are an essential part of object-oriented programming in JavaScript and are a powerful tool for working with complex data structures.

```
function checkEligibility(){
  if(this.age>=18){
    console.log(`${this.firstname} age is ${this.age} eligible for vote`);
  }else{
    console.log(`${this.firstname} age is ${this.age} not eligible for vote`);
  }
}

const user1={
  firstname:"Joes",
  age:35,
  eligibility:checkEligibility
}
user1.eligibility();
const user2={
  firstname:"Sara",
  age:12,
  eligibility:checkEligibility
}
user2.eligibility();
```

In JavaScript, it is common to define functions separately and then assign them to object properties as methods. In the above example, the function `checkEligibility` is defined separately and then assigned to the `eligibility` property of the `user1` and `user2` objects. This allows the function to be reused as a method for multiple objects.

When the method is invoked on the `user1` object, this inside the method will refer to `user1` object and when it is invoked on the `user2` object, this inside the method will refer to `user2` object, This way `checkEligibility` function can use the properties of the current object on which it is invoked.

In the example, `user1.eligibility()` is invoked, and it will execute the `checkEligibility` function. Inside the function, it uses the `this` keyword to access the properties of the `user1` object, namely `firstname` and `age`. The function then checks if the person's age is greater than or equal to 18, and if so, logs a message saying that the person is eligible to vote.

Similarly, when `user2.eligibility()` is invoked, it will execute the `checkEligibility` function. Inside the function, it uses the `this` keyword to access the properties of the `user2` object, namely `firstname` and `age`. The function then checks if the person's age is greater than or equal to 18, and if not, logs a message saying that the person is not eligible to vote.

In this way, the function `checkEligibility` is used as a method for different objects and it uses the `this` keyword to access the properties of the object that invokes it, making the function more reusable.

[Previous \(https://www.tutorjoes.in/JS_tutorial/optional_chaining_in_javascript\)](https://www.tutorjoes.in/JS_tutorial/optional_chaining_in_javascript)

[Next \(https://www.tutorjoes.in/JS_tutorial/call_apply_and_bind_in_javascript\)](https://www.tutorjoes.in/JS_tutorial/call_apply_and_bind_in_javascript)

List of Programs

[https://www.tutorjoes.in/JS_tutorial/function_inside_object_in_javascript#:~:text=In JavaScript%2C functions can be,%3A function\(\) %7B console.](https://www.tutorjoes.in/JS_tutorial/function_inside_object_in_javascript#:~:text=In%20JavaScript%20functions%20can%20be%2C%20function()%20console.)

3/7