|  |
| --- |
| The call() method in JavaScript is used to invoke a function with a specified this value and arguments provided individually. Here are two different examples of using call() with explanations for beginner coders:  Example 1: Using call() to Borrow a Method In this example, we'll create two objects, person1 and person2, and borrow a method from one object to use it in another. |
| var person1 = {  name: "MERN",  greet: function () {  console.log("Hello, my name is " + this.name);  },  };  var person2 = {  name: "Raj",  };  person1.greet.call(person2);  person1.greet(); |
| var person = {  name: "Gaurav"}  function greet() {  console.log("Hello, my name is " + this.name);  };  greet.call(person);  Output: Hello, my name is Gaurav |
| var person = {  name: "coder"}  function greet(age) {  console.log("Hello, my name is " + this.name + " age is " + age);  };    greet.call(person, 32);  Output: Hello, my name is coder age is 32 |
| var person = {  name: "Alice"}  function greet(age, profession) {  console.log("Hello, my name is " + this.name + " age is " + age + ". I am a " + profession);  };    greet.call(person, 32, "student");  Output: Hello, my name is Alice age is 32. I am a student. |

|  |
| --- |
| The **apply()** method in JavaScript is used to call a function with a specified **this** value and an array or array-like object of arguments. It allows you to pass arguments to a function as an array or an array-like object. Here are some examples of how to use the **apply()** method:   1. Using **apply()** to Pass Arguments to a Function: |
| function add(a, b) {  return a + b;  }    const args = [2, 3];  const result = add.apply(null, args);  console.log(result);  Output: 5 |
| // Define a person object with a greeting method  var person = {  name: "Alice"}  function greet(age, profession) {  console.log("Hello, my name is " + this.name + " age is " + age + ". I am a " + profession);  };    greet.apply(person, [32, "student"]);  Output: Hello, my name is Alice age is 32. I am a student. |
| const person = {  firstName: "John",  lastName: "Doe",  getFullName: function() {  return this.firstName + " " + this.lastName;  }  };  const anotherPerson = {  firstName: "Jane",  lastName: "Smith"  };  const fullName = person.getFullName.apply(anotherPerson);  console.log(fullName);  Output: "Jane Smith" |
| const numbers = [1, 2, 3, 4, 5];  const maxNumber = Math.max.apply(null, numbers);  const maxNumber2 = Math.max(...numbers);  console.log(maxNumber);  console.log(maxNumber2);  Output: 5  Output: 5 |

|  |
| --- |
| function sum() {  // converting arguments into an array  const argsArray = Array.prototype.slice.call(arguments);  return argsArray.reduce((acc, val) => acc + val, 0);  }  const result = sum.apply(null, [1, 2, 3, 4, 5]);  console.log(result);  Output: 15 |
| function sum(...argsArray1) {  // const argsArray = Array.prototype.slice.call(arguments);  return argsArray1.reduce((acc, val) => acc + val, 0);  }  const result = sum.apply(null, [1, 2, 3, 4, 5]);  console.log(result);  Output: 15 |
| The bind() method in JavaScript is used to create a new function with a specified this value and initial arguments provided to it. It's particularly useful when you want to change the context of a function or partially apply arguments. Here are some examples of how you can use bind():  Changing the Context (this value) of a Function: |
| const person = {  firstName: 'MERN',  lastName: 'Students',  };    function greet() {  console.log(`Hello, ${this.firstName} ${this.lastName}`);  }    const greetPerson = greet.bind(person);  greetPerson();  Output: Hello, MERN Students  Note: In this example, bind() is used to create a new function greetPerson with the this value set to the person object. |
| var A = {  x : 12,  multiply: function (x, y) {  console.log(x);  console.log(this.x);  return this.x \* y;  }  }  console.log(A.multiply(5,6));  Output: 5. 12. 72 |
| function multiply(x, y) {  return x \* y;  }  const double = multiply.bind(null);  console.log(double(5, 6));  Output: 30 |
| function multiply(x, y) {  return x \* y;  }    const double = multiply.bind(null, 4);    console.log(double(6));  Output: 24  First argument is 4 in all the cases. |
| const button = document.getElementById('myButton');  function handleClick() {  console.log('Button clicked');  }  button.addEventListener('click', handleClick.bind(this));  Here, bind() is used to ensure that when the button is clicked, the handleClick function is executed with the correct this value, which would be the element that triggered the event. |
| function delayedGreeting() {  console.log('Hello after 2 seconds');  }    const delayedGreet = delayedGreeting.bind(this);    setTimeout(delayedGreet, 2000);  Here, bind() is used to create a bound version of delayedGreeting that can be passed directly to setTimeout. |
|  |
|  |