In JavaScript, `call`, `apply`, and `bind` are methods used to set the context of `this` when invoking a function.

## Overview of Each Method

### 1. `call()`

The `call` method invokes a function with a specified `this` value (object) and arguments provided **individually**.

```javascript
function greet(greeting, punctuation) {
    console.log(greeting + ', ' + this.name + punctuation);
}
const person = { name: 'Alice' };
greet.call(person, 'Hello', '!'); // Output: Hello, Alice!
```

### 2. `apply()`

The `apply` method is similar to `call`, but arguments are provided as an **array**.

```javascript
function greet(greeting, punctuation) {
    console.log(greeting + ', ' + this.name + punctuation);
}
const person = { name: 'Alice' };
greet.apply(person, ['Hello', '!']); // Output: Hello, Alice!
```

### 3. `bind()`

The `bind` method is similar to `call`, but instead of executing the function immediately, it **returns a new function** with the specified `this` value and pre-set arguments.

```javascript
function greet(greeting, punctuation) {
    console.log(greeting + ', ' + this.name + punctuation);
}
const person = { name: 'Alice' };
const greetAlice = greet.bind(person, 'Hello');
greetAlice('!'); // Output: Hello, Alice!
```

## Summary: Key Differences

| Method | Execution | Arguments Handling |
|---|---|---|
| `call` | Invokes immediately | Arguments passed individually |
| `apply` | Invokes immediately | Arguments passed as an array |
| `bind` | Returns a new function | Arguments passed individually, can be invoked later |

By understanding these methods, you can effectively control the context of `this` and enhance the flexibility of your JavaScript functions.