Call vs Apply vs Bind

Purpose:

These methods allow us to write a function once and invoke it in a different context. They all attach “this” into a function (or object) and the difference is in the function invocation. As functions are also Objects in JavaScript, these 3 methods are used to control the invocation of the function. call() and apply() were introduced in ECMAScript 3 while bind() was added as part of ECMAScript 5. You can use call()/apply() to invoke the function immediately. bind() can be used when the function needs to be called later in certain events when it's useful.

Bind

```
const obj ={ name: "John"}
let greeting = function(a,b){
  return `${a} ${this.name} ${b}`
}
let bound = greeting.bind(obj);
console.log(bound("hey", "How are you?"))
```

Call()/Function.prototype.call()

```
const obj ={ name: "John"}
let greeting = function(a,b){
  return `${a} ${this.name} ${b}`
}
console.log(greeting.call(obj,"Hello", "How are you?"))
```

Apply()/Function.prototype.apply()

```
const obj ={ name: "John"}
let greeting = function(a,b){
  return `${a} ${this.name} ${b}`
}
console.log(greeting.apply(obj,["Hello", "How are you?"]))
```

All in one example

```
const obj = {name:"Priya"}
let greeting = function(a,b){
  return a+" "+this.name+" "+b;
}
console.log(greeting.call(obj, "Hello", "How are you?"));
console.log(greeting.apply(obj, ["Hello", "How are you?"]));
let test=greeting.bind(obj);console.log(test("Hello", "How are you?"))
```

call() and apply() method sets the **"this"** value which is the object upon which the function is invoked. In this case it's the **"obj"** object. The only difference of apply() with the call() method is that the second parameter of the apply() method accepts the arguments to the actual function as an array.

Comparison

The limitations of **call()** quickly become apparent in cases where we do not know the amount of argument a function would take, **apply()** shines in this scenario since it takes an array of arguments as its second argument.

Function	Time of Execution	Time of `this` binding	Arguments
`call()`	Now	Now	List
`apply()`	Now	Now	An array
`bind()`	Future (when we invoke the returned function as seen in the example above)	Now	List

Follow me for such info

https://www.linkedin.com/in/priya-bagde/

GitHub

https://github.com/priya42bagde

YouTube

https://www.youtube.com/channel/UCK1_Op30_pZ1zBs9l3HNyBw/videos (Priya Frontend Vlogs)