call-apply-bind 的区别

联系：三则都能改变 this 的指向，第一个参数就是 this 的指向。

区别：

- call 方法会使函数立即执行，第一个参数是 this 的指向，第二个、第三个和后面更多的参数是作为函数的参数传入。
- apply 方法会使函数立即执行，第一个参数是 this 的指向，第二个参数是一个数组，该数组中所有元素是函数参数，可以通过解构匹配。
- bind 方法会返回一个新函数，第一个参数是 this 的指向，第二个、第三个和后面更多的参数是作为函数的参数传入。

> - Function.prototype.apply() apply() 方法调用一个具有给定this值的函数，以及以一个数组（或类数组对象）的形式提供的参数。
> - Function.prototype.call() call() 方法使用一个指定的 this 值和单独给出的一个或多个参数来调用一个函数。
> - Function.prototype.bind() bind() 方法创建一个新的函数，在 bind() 被调用时，这个新函数的 this 被指定为 bind() 的第一个参数，而其余参数将作为新函数的参数，供调用时使用。

1. call

```
function greet() {
  var reply = [this.animal, "typically sleep between", this.sleepDuration].join(
    " ",
  );
  console.log(reply);
}
var obj = {
  animal: "cats",
  sleepDuration: "12 and 16 hours",
};
greet.call(obj); // cats typically sleep between 12 and 16 hours
```

`greet.call(obj)` 表明函数 `greet` 被立即执行，该函数在运行时使用的 `this` 值被 `obj` 替换。故 `greet` 函数中的 `this.animal` 指 `obj.animal`，`this.sleepDuration` 指 `obj.sleepDuration`。

2. apply

```
const array1 = ["a", "b"];
const array2 = [5, 5];
const elements = [0, 1, 2];
array1.push.apply(array2, elements);
console.info(array1); // ["a", "b"]
console.info(array2); // [5, 5, 0, 1, 2]
```

`array1.push.apply(array2, elements)` 相当于 `array2.push(elements)`。第一个参数 `array2` 改变了 `this` 的指向，相当于 `array1` 被变为了 `array2`。

3. bind

```
this.x = 9; // 在浏览器中，this 指向全局的 "window" 对象
var module = {
  x: 81,
  getX: function () {
    return this.x;
  },
};

module.getX(); // 81

var retrieveX = module.getX;
retrieveX();
// 返回 9 - 因为函数是在全局作用域中调用的

// 创建一个新函数，把 'this' 绑定到 module 对象
// 新手可能会将全局变量 x 与 module 的属性 x 混淆
var boundGetX = retrieveX.bind(module);
boundGetX(); // 81
```