call和apply

call

1、用法：`foo.call(obj, arg1, arg2, ...)` -> **主要用于改变this的指向**；

```
function foo(year, place) {
  console.log(this.name+" is "+year+" born from "+place);
}
window.name = 'syz';
const obj = {
  name: 'syc'
}
foo(1995, 'china'); // syz is 1995 born from china
foo.call(obj, 1995, 'china'); // syc is 1995 born from china
```

首先我们要知道，每个函数中的this是在调用时绑定的，this指向哪里，完全取决于函数的调用位置。
比如上面，我们先执行了`foo()`，基于我们调用这个函数的位置,如果我们在浏览器中运行的话（一般都是），此时this指向的是window，所以这时候this.name = syz;
那如果我们想要让 this.a = obj.name 的话，就可以使用call来显式绑定this的指向。

apply

1、用法：`obj.apply(thisObj, [arg1, arg2, ...])` -> **主要用于改变this的指向**；

```
function foo(year, place) {
  console.log(this.name+" is "+year+" born from "+place);
}
window.name = 'syz';
const obj = {
  name: 'syc'
}
foo(1995, 'china'); // syz is 1995 born from china
foo.apply(obj, [1995, 'china']); // syc is 1995 born from china
```

apply方法和call方法的区别就是apply中第二个参数接受的是一个数组。

call和apply的作用

1、**一个重要的作用就是扩充函数赖以运行的作用域；**

```
window.color = "red";
let o = {
    color: "blue"
};
function sayColor() {
    console.log(this.color);
}
sayColor(); //red
sayColor.call(this); //red 因为在全局环境下调用函数 this默认指向window
sayColor.call(window); //red this指向window
sayColor.call(o); //blue this指向o对象
```

使用call和apply来扩充作用域的最大好处，就是对象不需要和方法有任何耦合关系，调用者控制this的指向，就可以实现不同的功能。

bind

1、用法：`foo.bind(obj, arg1, arg2, ...)` -> **返回一个函数，该函数永久地改变this的指向；**

```
function foo(year, place) {
  console.log(this.name+" is "+year+" born from "+place);
}
window.name = 'syz';
const obj = {
  name: 'syc'
}
foo(1995, 'china'); // syz is 1995 born from china
let haha = foo.bind(obj, 1995, 'china');
haha(); // syc is 1995 born from china
```

bind类似于call，但是call和apply会立即执行，而bind是返回绑定this之后的函数(永久地改变this的指向，原函数不变)

apply、call、bind三者的区别

- 三者都可以改变函数的this的指向；
- 三者第一个参数都是this要指向的对象，如果没有这个参数或参数为undefined或null，则默认指向全局window。
- 三者都可以传参，但是call和bind是参数列表，apply是数组。
- bind是返回绑定this之后的函数，便于稍后调用；apply和call则是立即执行。