call apply bind

call,apply,bind 函数的作用

改变函数执行时的上下文，具体一点是改变函数运行时的`this`指向。

call,apply与bind的区别

`call`与`apply`改变了函数`this`上下文之后便执行该函数，而`bind`则是返回改变了上下文后的一个函数。

call与apply的区别

在于参数的区别

```
fn.call(thisObj, arg1, arg2, arg3...);
fn.apply(thisObj, [arg1, arg2, arg3...]);
```

call与apply用法

```
function fn() {
    console.log(this.a);
}
var obj = {
    a: 20
}
fn.call(obj);//20
fn.apply(obj);//20
```

`call`方法后面的参数项将要执行的函数传递参数时，不同于`apply`以数组的以数组的形式传递，`call`以一个个的形式传递。

```
function fn(num1, num2) {
    console.log(this.a + num1 + num2);
}
var obj = {
    a: 20
}
fn.call(obj, 100, 10); // 130
fn.apply(obj, [20, 10]); // 50
```

call与apply的应用场景

- 将类数组对象转换为数组

```
function exam(a, b, c, d, e) {
    console.log(arguments);//函数的自带属性arguments ,arguments为对象
    var arg = [].slice.call(arguments);//这句话相当于Array.slice.call(arguments)
    console.log(arg);//[ 2, 8, 9, 10, 3 ]
}
exam(2, 8, 9, 10, 3);
```

使用`call`，`apply`将`arguments`转换为数组，返回结果为数组，`arguments`自身不会改变。也常常使用该方法将`DOM`中的`nodelist`转换为数组`[].slice.call(document.getElementsByTagName('li') )`。

- 根据自己的需要灵活修改this指向

```
var foo = {
    name: 'que',
    showName: function() {
      console.log(this.name);
    }
}
var bar = {
    name: 'Martin'
}
foo.showName.call(bar);//Martin
```

- 实现继承

```
var Person = function(name, age) {// 定义父级的构造函数
    this.name = name;
    this.age  = age;
    this.gender = ['man', 'woman'];
}
var Student = function(name, age, high) {// 定义子类的构造函数
    Person.call(this, name, age);//这里的this指代
    this.high = high;
}
Student.prototype.message = function(){
    console.log(
        'name:'+this.name+
        ', age:'+this.age+
        ', high:'+this.high+
        ', gender:'+this.gender[0]+';'
    );
}
new Student('Martin', 23, '177cm').message();//name:Martin, age:23, high:177cm, gender:man;

//Student的构造函数相当于以下
var Student = function(name, age, high) {
    this.name = name;
    this.age  = age;
    this.gender = ['man', 'woman'];
    // Person.call(this, name, age); 这一句话，相当于上面三句话，因此实现了继承
    this.high = high;
```

```
}
```

如果，我们使用 `var per =  Person("MT", 18)` ，等同于执行了 `Person` 函数，在此种情况，函数内部的 `this` ，指向的是 `window` ,函数执行过后，并没有返回值，那么就默认返回一个 `undefined` 。所以如果 `console.log(per)` ，那么将得到一个 `undefined` 值。同时可以看到控制台中，执行 `console.log(this)` ，`window` 对象中，挂载了 `name,age,gender` 等变量。

在 `Student` 的构造函数中，借助 `call` 方法，将父级的构造函数执行了一次，相当于将 `Person` 中的代码，在 `Student` 中复制了一份，其中的 `this` 指向为从 `Student` 中 `new` 出来的实例对象。 `call` 方法保证了 `this` 的指向正确，因此就相当于实现了继承。