8、函数的扩展

- 函数参数的默认值
- 箭头函数
- 箭头函数中this的理解：
- 箭头函数使用注意点

① 函数参数的默认值：

- ES6之前，不能直接为函数的参数指定默认值，只能采用变通的方法。
- ES6允许为函数的参数设置默认值，即直接写在参数定义的后面。
- 参数默认值的位置：
  - 通常情况下，定义了默认值的参数，应该是函数的尾参数。
  - 因为这样比较容易看出来，到底省略了哪些参数。

```
//ES6之前，不能直接为函数的参数指定默认值，只能采用变通的方法。
function func(name){
    name = name || "World";
    return "Hello "+name;
}
console.log(func());           //Hello World
console.log(func("ZhangSan")); //Hello ZhangSan

//ES6允许为函数的参数设置默认值，即直接写在参数定义的后面。
function add(x=0,y=0){
    return x+y;
}
console.log(add());     //0
console.log(add(10));   //10
//跳过第一个参数给第二个参数y传值
console.log(add(undefined,20)); //20
console.log(add(10,20));//30
```

② 箭头函数：

- ES6允许使用“箭头”（=>）定义函数。

```
var f = v => v;

// 等同于
var f = function (v) {
  return v;
};
```

- 如果箭头函数不需要参数或需要多个参数，就使用一个圆括号代表参数部分。

```
//无参数
var f = () => 5;
// 等同于
var f = function () { return 5 };

//多个参数
var sum = (num1, num2) => num1 + num2;
// 等同于
var sum = function(num1, num2) {
  return num1 + num2;
};
```

- 如果箭头函数的代码块部分多于一条语句，就要使用大括号将它们括起来，并且使用return语句返回。
- 也就是说默认函数体没有大括号是自带隐式返回renturn的。

```
var sum = (num1, num2) => { return num1 + num2; }
```

- 由于大括号被解释为代码块，所以如果箭头函数直接返回一个对象，必须在对象外面加上括号，否则会报错。

```
// 报错
let getTempItem = id => { id: id, name: "Temp" };

// 不报错
let getTempItem = id => ({ id: id, name: "Temp" });
```

③ 箭头函数中this的理解：

- 传统javaScript代码中this的使用：

```
//定义一个stu学生对象，内有：两个属性、一个方法。
const stu = {
    name:"张三",
    likes:['吃饭','睡觉','敲代码'],
    printLikes:function(){
        //使用map遍历likes属性，并输出信息
        this.likes.map(function(like){
```

```
            //此处的this代表的是window对象，而非stu对象
            console.log(`${this.name} 喜欢 ${like}`);
        });
    }
};
stu.printLikes(); //使用stu对象调用自己的方法
/* 输出结果:
   喜欢 吃饭
   喜欢 睡觉
   喜欢 敲代码
*/
```

- 上面的输出this.name没有信息，如下进行修改就可以了。

```
//定义一个stu学生对象，内有：两个属性、一个方法。
const stu = {
    name:"张三",
    likes:['吃饭','睡觉','敲代码'],
    printLikes:function(){
        let self = this;
        //使用map遍历likes属性，并输出信息
        this.likes.map(function(like){
            //此处的this代表的是window对象，而非stu对象
            console.log(`${self.name} 喜欢 ${like}`);
        });
    }
};
stu.printLikes(); //使用stu对象调用自己的方法
/* 输出结果:
   张三 喜欢 吃饭
   张三 喜欢 睡觉
   张三 喜欢 敲代码
*/
```

- 使用箭头函数后的效果

```
//定义一个stu学生对象，内有：两个属性、一个方法。
const stu = {
    name:"张三",
    likes:['吃饭','睡觉','敲代码'],
    printLikes:function(){
        //使用map遍历likes属性，并输出信息
        this.likes.map(like => {
            //箭头函数中没有自己的this，故此处this是继承父作用域的。
            //而且是在定义的时候已指定，不会随着调用而改变。
            console.log(`${this.name} 喜欢 ${like}`);
        });
    }
```

```
};
stu.printLikes(); //使用stu对象调用自己的方法
/* 输出结果:
   张三 喜欢 吃饭
   张三 喜欢 睡觉
   张三 喜欢 敲代码
*/
```

④ 箭头函数使用注意点

- 箭头函数有几个使用注意点。

（1）函数体内的this对象，就是定义时所在的对象，而不是使用时所在的对象。

（2）不可以当作构造函数，也就是说，不可以使用new命令，否则会抛出一个错误。

（3）当你真的需要this的时候，如为对象添加普通方法或事件绑定回调函数使用箭头函数，可能获取不到this。

（4）不可以使用arguments对象，该对象在函数体内不存在。

上面四点中，第一点尤其值得注意。this对象的指向是可变的，但是在箭头函数中，它是固定的。

```
//1.箭头函数不可以作为构造函数使用
const Stu1 = (name,age)=>{
    this.name = name;
    this.age = age;
}
//要改成常规函数如下:
const Stu2 = function(name,age){
    this.name = name;
    this.age = age;
}

//实例化
//s = new Stu1("zhangsan",20);//报错: TypeError: Stu is not a constructor
s = new Stu2("zhangsan",20);  //正常
console.log(s); //Stu2 {name: "zhangsan", age: 20}
```

```
//定义Stu类
const Stu = function(name,age){
    this.name = name;
    this.age = age;
}
s = new Stu("zhangsan",20);  //正常实例化
//为s原型对象添加一个getInfo方法，使用箭头函数，里面的this是window对象
Stu.prototype.getInfo = ()=>{
    return `我叫${this.name}，今年${this.age}岁`;
}
//改成常规函数
```

```
Stu.prototype.getInfo2 = function(){
    return `我叫${this.name}，今年${this.age}岁`;
}
console.log(s.getInfo());  //我叫，今年undefined岁
console.log(s.getInfo2()); //我叫zhangsan，今年20岁
```