**为函数参数指定默认值**

如果不传参数，我就给个默认的值

原来的做法：

function fn(a, b) {

a = typeof(a) === 'undefined' ? 10 : a;

b = typeof(b) === 'undefined' ? 20 : b;

console.log(a + b);

}

fn();// 30

fn(0, 10); // 10

ES6指定默认参数

function fn(a = 10, b = 20) {

console.log(a + b);

}

fn();// 30

fn(0, 10); // 10

**rest参数(即剩余部分)**

参数形式为（”…变量名”），用于获取函数的多余参数，这样就不需要使用arguments对象了，rest参数搭配的变量是一个数组，该变量将多余的参数放入数组中。ES6中提倡不用arguments

function sum(...arr) {

var res = 0;

for (let i = 0; i < arr.length; i++) {

res += arr[i];

}

return res;

}

console.log(sum(1, 2, 3, 4, 5)); // 15

注意：在...arr前面还可以再有参数，但是在它的后面不可以再有参数了。

**箭头函数**

格式：let fn = () => {};

let 函数名 = (参数) => {函数体};

写法注意：

如果函数体没有大括号，哪么函数体哪里是直接返回的值，有大括号，则要return 返回值

如果只有一个参数，可以不写小括号。多个参数或者没有参数，都要写小括号。

如果不写return，且要返回一个对象，要用小括号包起来。

let fn = a => a;

console.log(fn(2));

这就相当于定义了一个函数名叫fn，它接收一个参数a，返回了这个参数

let fn = function(a) {

return a;

}

console.log(fn(2));

let fn = (a, b) => a + b;

console.log(fn(1, 2)); // 3

let fn = (a, b) => {

a = a \* 2;

b = b \* 2;

return a + b;

};

console.log(fn(1, 2)); // 6

如果要返回一个对象，则要用()包起来

let fn = (a, b) => ({

a,

b

});

console.log(fn(1, 2)); // Object { a: 1, b: 2 }

箭头函数主要用于回调函数

let arr = [4, 3, 5, 2, 1];

arr.sort((a, b) => a - b);

console.log(arr); // Array [ 1, 2, 3, 4, 5 ]

注意事项：

1. 箭头函数体内没有自己的this对象，所以在使用的时候，其内部的this就是定义时所在环境的对象，而不是使用时所在环境的对象。

所以，不可以使用call apply bind来改变其内的this

function fn() {

setTimeout(function() {

console.log(this); // window

}, 1000);

setTimeout(() => {

console.log(this); // obj

}, 1000);

}

var obj = {

fn

}

obj.fn();

2、箭头函数体内没有arguments对象，如果要用，可以用rest（…）参数代替。

3、不可以当作构造函数，不可以使用new命令，否则会抛出一个错误。