ES6中函数的用法

1. 参数默认值
2. 箭头函数
3. 展开运算符

# 【函数默认值】

1.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 log(x, y = 'World')

{

console.log(x, y);

}

1. 参数默认值是惰性求值的。

let x = 99;

function foo(p = x + 1)

{

console.log(p);

}

foo() // 100

x = 100;

foo() // 101

上面代码中，参数p的默认值是x + 1。这时，每次调用函数foo，都会重新计算x + 1，而不是默认p等于 100。

# 【箭头函数】

1. 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。

var f = () => 5;

// 等同于var f = function () { return 5 };

var sum = (num1, num2) => num1 + num2;

// 等同于var sum = function(num1, num2) {

return num1 + num2;};

1. 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。

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

1. 括号(无论是花括号还是括号)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" });

1. 只有一行语句，且不需要返回值，可以采用下面的写法，就不用写大括号了。

let fn = () => void doesNotReturn();

1. 可以与变量解构结合

const full = ({ first, last }) => first + ' ' + last;

// 等同于

function full(person) {

return person.first + ' ' + person.last;

}

# 【展开运算符】

1. 将一个数组，变为参数序列。

const numbers = [4, 38];

add(...numbers) // 42

1. 放置表达式

const arr = [

...(x > 0 ? ['a'] : []),

'b',];

1. 通过push函数，将一个数组添加到另一个数组的尾部。

let arr1 = [0, 1, 2];

let arr2 = [3, 4, 5];

arr1.push(...arr2);

1. 复制数组的简便写法。

const a1 = [1, 2];

// 写法一

const a2 = [...a1];

// 写法二

const [...a2] = a1;

1. 与解构赋值结合

const [first, ...rest] = [1, 2, 3, 4, 5];

first // 1

rest // [2, 3, 4, 5]

1. 将字符串转为真正的数组。

[...'hello']

// [ "h", "e", "l", "l", "o" ]

1. 涉及到操作四个字节的 Unicode 字符的函数，都用扩展运算符改写。

let str = 'x\uD83D\uDE80y';

str.split('').reverse().join('')

// 'y\uDE80\uD83Dx'

[...str].reverse().join('')

// 'y\uD83D\uDE80x'