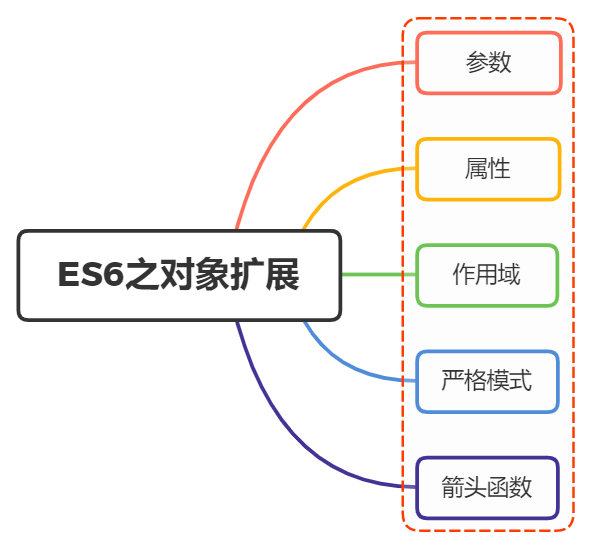
# 面试官：对象新增了哪些扩展？



## 一、参数

ES6允许为函数的参数设置默认值

function log(x, y = 'World') {  
 console.log(x, y);  
}  
  
console.log('Hello') // Hello World  
console.log('Hello', 'China') // Hello China  
console.log('Hello', '') // Hello

函数的形参是默认声明的，不能使用let或const再次声明

function foo(x = 5) {  
 let x = 1; // error  
 const x = 2; // error  
}

参数默认值可以与解构赋值的默认值结合起来使用

function foo({x, y = 5}) {  
 console.log(x, y);  
}  
  
foo({}) // undefined 5  
foo({x: 1}) // 1 5  
foo({x: 1, y: 2}) // 1 2  
foo() // TypeError: Cannot read property 'x' of undefined

上面的foo函数，当参数为对象的时候才能进行解构，如果没有提供参数的时候，变量x和y就不会生成，从而报错，这里设置默认值避免

function foo({x, y = 5} = {}) {  
 console.log(x, y);  
}  
  
foo() // undefined 5

参数默认值应该是函数的尾参数，如果不是非尾部的参数设置默认值，实际上这个参数是没发省略的

function f(x = 1, y) {  
 return [x, y];  
}  
  
f() // [1, undefined]  
f(2) // [2, undefined]  
f(, 1) // 报错  
f(undefined, 1) // [1, 1]

## 二、属性

### 函数的length属性

length将返回没有指定默认值的参数个数

(function (a) {}).length // 1  
(function (a = 5) {}).length // 0  
(function (a, b, c = 5) {}).length // 2

rest 参数也不会计入length属性

(function(...args) {}).length // 0

如果设置了默认值的参数不是尾参数，那么length属性也不再计入后面的参数了

(function (a = 0, b, c) {}).length // 0  
(function (a, b = 1, c) {}).length // 1

### name属性

返回该函数的函数名

var f = function () {};  
  
// ES5  
f.name // ""  
  
// ES6  
f.name // "f"

如果将一个具名函数赋值给一个变量，则 name属性都返回这个具名函数原本的名字

const bar = function baz() {};  
bar.name // "baz"

Function构造函数返回的函数实例，name属性的值为anonymous

(new Function).name // "anonymous"

bind返回的函数，name属性值会加上bound前缀

function foo() {};  
foo.bind({}).name // "bound foo"  
  
(function(){}).bind({}).name // "bound "

## 三、作用域

一旦设置了参数的默认值，函数进行声明初始化时，参数会形成一个单独的作用域

等到初始化结束，这个作用域就会消失。这种语法行为，在不设置参数默认值时，是不会出现的

下面例子中，y=x会形成一个单独作用域，x没有被定义，所以指向全局变量x

let x = 1;  
  
function f(y = x) {   
 // 等同于 let y = x   
 let x = 2;   
 console.log(y);  
}  
  
f() // 1

## 四、严格模式

只要函数参数使用了默认值、解构赋值、或者扩展运算符，那么函数内部就不能显式设定为严格模式，否则会报错

// 报错  
function doSomething(a, b = a) {  
 'use strict';  
 // code  
}  
  
// 报错  
const doSomething = function ({a, b}) {  
 'use strict';  
 // code  
};  
  
// 报错  
const doSomething = (...a) => {  
 'use strict';  
 // code  
};  
  
const obj = {  
 // 报错  
 doSomething({a, b}) {  
 'use strict';  
 // code  
 }  
};

## 五、箭头函数

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

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;  
};

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

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

如果返回对象，需要加括号将对象包裹

let getTempItem = id => ({ id: id, name: "Temp" });

注意点：

* 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
* 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
* 不可以使用arguments对象，该对象在函数体内不存在。如果要用，可以用 rest 参数代替
* 不可以使用yield命令，因此箭头函数不能用作 Generator 函数

## 参考文献

* https://es6.ruanyifeng.com/#docs/function