Es6笔记

**二、变量的解构赋值**

1. **数组的解构赋值**

ES6允许写成下面这样。

var [a, b, c] = [1, 2, 3];

解构赋值允许指定默认值

var [foo = true] = [];

foo // true

[x, y = 'b'] = ['a']; // x='a', y='b'

[x, y = 'b'] = ['a', undefined]; // x='a', y='b'

function f() {

console.log('aaa');

}

let [x = f()] = [1];

let [x = 1, y = x] = []; // x=1; y=1

let [x = 1, y = x] = [2]; // x=2; y=2

let [x = 1, y = x] = [1, 2]; // x=1; y=2

let [x = y, y = 1] = []; // ReferenceError

1. **对象的解构赋值**

var { foo, bar } = { foo: "aaa", bar: "bbb" };

foo // "aaa"

bar // "bbb"

var { foo, bar } = { foo: "aaa", bar: "bbb" };

foo // "aaa"

bar // "bbb"

对象的解构与数组有一个重要的不同。数组的元素是按次序排列的，变量的取值由它的位置决定；而对象的属性没有次序，变量必须与属性同名，才能取到正确的值。

对象的解构赋值的内部机制，是先找到同名属性，然后再赋给对应的变量。真正被赋值的是后者，而不是前者

var { foo: baz } = { foo: "aaa", bar: "bbb" };

baz // "aaa"

foo // error: foo is not defined

上面代码中，foo是匹配的模式，baz才是变量。真正被赋值的是变量baz，而不是模式foo。

注意，采用这种写法时，变量的声明和赋值是一体的。对于let和const来说，变量不能重新声明，所以一旦赋值的变量以前声明过，就会报错。

let foo;

let {foo} = {foo: 1}; // SyntaxError: Duplicate declaration "foo"

let baz;

let {bar: baz} = {bar: 1}; // SyntaxError: Duplicate declaration "baz"

let foo;

({foo} = {foo: 1}); // 成功

let baz;

({bar: baz} = {bar: 1}); // 成功

var node = {

loc: {

start: {

line: 1,

column: 5

}

}

};

var { loc: { start: { line }} } = node;

line // 1

loc // error: loc is undefined

start // error: start is undefined

上面代码中，只有line是变量，loc和start都是模式，不会被赋值。

// 错误的写法

var x;

{x} = {x: 1};

// SyntaxError: syntax error

上面代码的写法会报错，因为JavaScript引擎会将{x}理解成一个代码块，从而发生语法错误。只有不将大括号写在行首，避免JavaScript将其解释为代码块，才能解决这个问题。

// 正确的写法

({x} = {x: 1});

上面代码将整个解构赋值语句，放在一个圆括号里面，就可以正确执行。

let { log, sin, cos } = Math;

上面代码将Math对象的对数、正弦、余弦三个方法，赋值到对应的变量上，使用起来就会方便很多。

由于数组本质是特殊的对象，因此可以对数组进行对象属性的解构。

var arr = [1, 2, 3];

var {0 : first, [arr.length - 1] : last} = arr;

first // 1

last // 3

上面代码对数组进行对象解构。数组arr的0键对应的值是1，[arr.length - 1]就是2键，对应的值是3。

1. **字符串的解构赋值**

字符串也可以解构赋值。这是因为此时，字符串被转换成了一个类似数组的对象。

const [a, b, c, d, e] = 'hello';

a // "h"

b // "e"

c // "l"

d // "l"

e // "o"

类似数组的对象都有一个length属性，因此还可以对这个属性解构赋值。

let {length : len} = 'hello';

len // 5

1. **函数参数的解构赋值**