# 变量的解构赋值

数组的解构赋值

对象的解构赋值

字符串的解构赋值

数值和布尔值的解构赋值

函数参数的解构赋值

圆括号问题

用途

解构赋值的规则是，只要等号右边的值不是对象或数组，就先将其转为对象。由于undefined和null无法转为对象，所以对它们进行解构赋值，都会报错。

【默认值】

1.解构也可以使用默认值

2.

注意，下面的写法会得到不一样的结果。

function move({x, y} = { x: 0, y: 0 }) {

return [x, y];}

move({x: 3, y: 8}); // [3, 8]

move({x: 3}); // [3, undefined]

move({}); // [undefined, undefined]

move(); // [0, 0]

上面代码是为函数move的参数指定默认值，而不是为变量x和y指定默认值，所以会得到与前一种写法不同的结果。

3.

undefined就会触发函数参数的默认值。

[1, undefined, 3].map((x = 'yes') => x);

// [ 1, 'yes', 3 ]

# 【圆括号问题】

ES6 的规则是，只要有可能导致解构的歧义，就不得使用圆括号。

建议只要有可能，就不要在模式中放置圆括号。

### 不能使用圆括号的情况 [§](http://es6.ruanyifeng.com/" \l "docs/destructuring#不能使用圆括号的情况) [⇧](http://es6.ruanyifeng.com/" \l "docs/destructuring)

1. 变量声明语句

（2）函数参数

（3）赋值语句的模式

可以使用圆括号的情况只有一种：赋值语句的非模式部分，可以使用圆括号。

[(b)] = [3]; // 正确

({ p: (d) } = {}); // 正确

[(parseInt.prop)] = [3]; // 正确

上面三行语句都可以正确执行，因为首先它们都是赋值语句，而不是声明语句；其次它们的圆括号都不属于模式的一部分。第一行语句中，模式是取数组的第一个成员，跟圆括号无关；第二行语句中，模式是p，而不是d；第三行语句与第一行语句的性质一致。

# 【用途】

1. ****交换变量的值****
2. ****从函数返回多个值****

function example() {

return {

foo: 1,

bar: 2

};

}

let { foo, bar } = example();

1. ****函数参数放肆的定义****

// 参数是一组无次序的值function f({x, y, z})

{ ... }

f({z: 3, y: 2, x: 1});

1. ****提取 JSON 数据****
2. 指定参数的默认值，就避免了在函数体内部再写var foo = config.foo || 'default foo';这样的语句。
3. ****遍历 Map 结构****

const map = new Map();

map.set('first', 'hello');

map.set('second', 'world');

for (let [key, value] of map) {

console.log(key + " is " + value);}

// first is hello// second is world

1. ****输入模块的指定方法****

加载模块时，往往需要指定输入哪些方法。解构赋值使得输入语句非常清晰。

const { SourceMapConsumer, SourceNode } = require("source-map");