变量的解构赋值

1. 交换变量的值

```
// 写法一：引入第三个变量 z1，不使用解构赋值的语法
let  x1= 1, y1 = 2, z1;
z1 = x1;
x1 = y1;
y1 = z1;
x1 // 2
y1 // 1

// 写法二：使用解构赋值的语法。简洁、易读、语义清晰
let x2 = 1, y2 = 2;
[x2, y2] = [y2, x2];
x2 // 2
y2 // 1
```

2. 从函数返回多个值

函数只能返回一个值，如果要返回多个值，只能将它们放在数组或对象里返回。有了解构赋值，取出这些值就非常方便。

```
// 返回一个数组
function example() {
  return [1, 2, 3];
}
let [a, b, c] = example();

// 返回一个对象
function example() {
  return {
    foo: 1,
    bar: 2
  };
}
let { foo, bar } = example();
```

3. 函数参数的定义

解构赋值可以方便地将一组参数与变量名对应起来。

```
// 参数是一组有次序的值
function f([x, y, z]) { ... }
f([1, 2, 3]);
```

```
// 参数是一组无次序的值
function f({x, y, z}) { ... }
f({z: 3, y: 2, x: 1});
```

4. 提取 JSON 数据

解构赋值对提取 JSON 对象中的数据，尤其有用。

```
let jsonData = {
  id: 42,
  status: "OK",
  data: [867, 5309]
};
let { id, status, data: number } = jsonData;
console.log(id, status, number); // 42, "OK", [867, 5309]
```

5. 函数参数的默认值

```
jQuery.ajax = function (url, {
  async = true,
  beforeSend = function () {},
  cache = true,
  complete = function () {},
  crossDomain = false,
  global = true,
  // ... more config
} = {}) {
  // ... do stuff
};
```

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。

6. 遍历 Map 结构

任何部署了 Iterator 接口的对象，都可以用 `for...of` 循环遍历。

```
const map = new Map();
map.set('first', 'hello').set('second', 'world');

for (let [key, value] of map) {
  console.log(key + " is " + value);
}
// first is hello
// second is world
```

如果只想获取键名，或者只想获取键值，可以写成下面这样。

```
// 获取键名
for (let [key] of map) { /* */ }

// 获取键值
for (let [,value] of map) { /* */ }
```

7. 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");
```