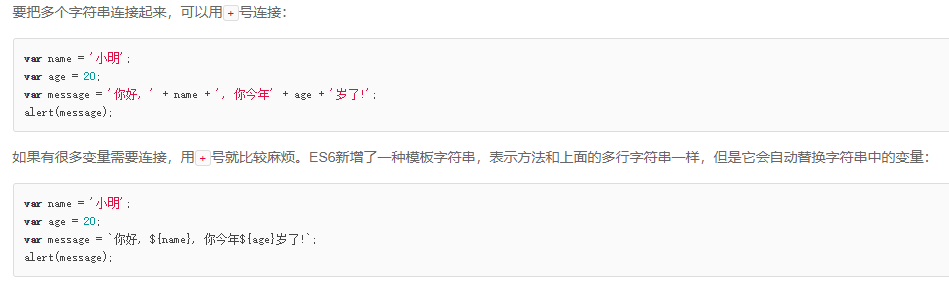
1. 多行字符串表示方法：



1. 模板字符串



1. MAP

Map是一组键值对的结构，具有极快的查找速度。



1. SET

Set和Map类似，也是一组key的集合，但不存储value。由于key不能重复，所以，在Set中，没有重复的key。

要创建一个Set，需要提供一个Array作为输入，或者直接创建一个空Set：

**var** s1 = **new** Set(); *// 空Set***var** s2 = **new** Set([1, 2, 3]); *// 含1, 2, 3*

重复元素在Set中自动被过滤：

var s = new **Set**([1, 2, 3, 3, '3']);

s; // **Set** {1, 2, 3, "3"}

注意数字3和字符串'3'是不同的元素。

通过add(key)方法可以添加元素到Set中，可以重复添加，但不会有效果：

s.add(4);

s; // **Set** {1, 2, 3, 4}

s.**add**(4);

s; // 仍然是 **Set** {1, 2, 3, 4}

通过delete(key)方法可以删除元素：

var s = new **Set**([1, 2, 3]);

s; // **Set** {1, 2, 3}

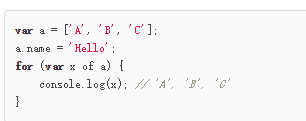
s.**delete**(3);

s; // **Set** {1, 2}

1. Iterable

ES6标准引入了新的iterable类型，Array、Map和Set都属于iterable类型。

具有iterable类型的集合可以通过新的for ... of循环来遍历。



6. rest参数

ES6标准引入了rest参数，上面的函数可以改写为：

**function** foo(a, b, ...rest) {

console.log('a = ' + a);

console.log('b = ' + b);

console.log(rest);

}

foo(1, 2, 3, 4, 5);*// 结果:// a = 1// b = 2// Array [ 3, 4, 5 ]*

foo(1);*// 结果:// a = 1// b = undefined// Array []*

rest参数只能写在最后，前面用...标识，从运行结果可知，传入的参数先绑定a、b，多余的参数以数组形式交给变量rest，所以，不再需要arguments我们就获取了全部参数。

如果传入的参数连正常定义的参数都没填满，也不要紧，rest参数会接收一个空数组（注意不是undefined）。

1. Let

为了解决块级作用域，ES6引入了新的关键字let，用let替代var可以申明一个块级作用域的变量：

'use strict';

**function** foo() {

**var** sum = 0;

**for** (**let** i=0; i<100; i++) {

sum += i;

}

*// SyntaxError:*

i += 1;

}

1. 常量

ES6标准引入了新的关键字const来定义常量，const与let都具有块级作用域：

1. 解构赋值

从ES6开始，JavaScript引入了解构赋值，可以同时对一组变量进行赋值。

**let** [x, [y, z]] = ['hello', ['JavaScript', 'ES6']];

x; *// 'hello'*

y; *// 'JavaScript'*

z; *// 'ES6'*

解构赋值还可以忽略某些元素：

**let** [, , z] = ['hello', 'JavaScript', 'ES6']; *// 忽略前两个元素，只对z赋值第三个元素*

z; *// 'ES6’*

对一个对象进行解构赋值时，同样可以直接对嵌套的对象属性进行赋值，只要保证对应的层次是一致的：

**var** person = {

name: '小明',

age: 20,

gender: 'male',

passport: 'G-12345678',

school: 'No.4 middle school',

address: {

city: 'Beijing',

street: 'No.1 Road',

zipcode: '100001'

}

};**var** {name, address: {city, zip}} = person;

name; *// '小明'*

city; *// 'Beijing'*

zip; *// undefined, 因为属性名是zipcode而不是zip// 注意: address不是变量，而是为了让city和zip获得嵌套的address对象的属性:*

address; *// Uncaught ReferenceError: address is not defined*

使用解构赋值对对象属性进行赋值时，如果对应的属性不存在，变量将被赋值为undefined，这和引用一个不存在的属性获得undefined是一致的。如果要使用的变量名和属性名不一致，可以用下面的语法获取：

**var** person = {

name: '小明',

age: 20,

gender: 'male',

passport: 'G-12345678',

school: 'No.4 middle school'

};

*// 把passport属性赋值给变量id:***let** {name, passport:id} = person;

name; *// '小明'*

id; *// 'G-12345678'// 注意: passport不是变量，而是为了让变量id获得passport属性:*

passport; *// Uncaught ReferenceError: passport is not defined*

### 使用场景

解构赋值在很多时候可以大大简化代码。例如，交换两个变量x和y的值，可以这么写，不再需要临时变量：

**var** x=1, y=2;

[x, y] = [y, x]

快速获取当前页面的域名和路径：

**var** {hostname:domain, pathname:path} = location;

如果一个函数接收一个对象作为参数，那么，可以使用解构直接把对象的属性绑定到变量中。例如，下面的函数可以快速创建一个Date对象：

**function** buildDate({year, month, day, hour=0, minute=0, second=0}) {

**return** **new** Date(year + '-' + month + '-' + day + ' ' + hour + ':' + minute + ':' + second);

}

它的方便之处在于传入的对象只需要year、month和day这三个属性：

buildDate({ year: 2017, month: 1, day: 1 });// Sun Jan 01 2017 00:00:00 GMT+0800 (CST)

也可以传入hour、minute和second属性：

buildDate({ year: 2017, month: 1, day: 1, hour: 20, minute: 15 });// Sun Jan 01 2017 20:15:00 GMT+0800 (CST)

1. 解构赋值