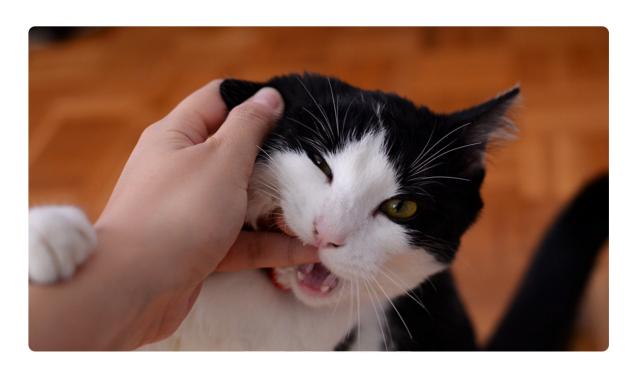
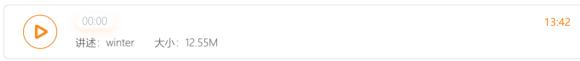
# 31 | JavaScript语法 (三) : 什么是表达式语句?

winter 2019-04-04





你好, 我是 winter。

不知道你有没有注意到,我们在语句部分,讲到了很多种语句类型,但是,其实最终产生执行效果的语句不多。

事实上,真正能干活的就只有表达式语句,其它语句的作用都是产生各种结构,来控制表达式语句执行,或者改变表达式语句的意义。

今天的课程, 我们就深入到表达式语句中来学习一下。

## 什么是表达式语句

表达式语句实际上就是一个表达式,它是由运算符连接变量或者直接量构成的(关于直接量我们在下一节详细讲解)。

一般来说,我们的表达式语句要么是函数调用,要么是赋值,要么是自增、自减,否则表达式计算的结果没有任何意义。

但是从语法上,并没有这样的限制,任何合法的表达式都可以当做表达式语句使用。比如我们看下面的例子。

```
1 a + b;
2
```

这句代码计算了 a 和 b 相加的值,但是不会显示出来,也不会产生任何执行效果(除非 a 和 b 是 getter),但是不妨碍它符合语法也能够被执行。

下面我们就一起来了解下都有哪些表达式,我们从粒度最小到粒度最大了解一下。

# PrimaryExpression 主要表达式

首先我们来给你讲解一下表达式的原子项: Primary Expression。它是表达式的最小单位,它所涉及的语法结构也是优先级最高的。

Primary Expression 包含了各种"直接量",直接量就是直接用某种语法写出来的具有特定类型的值。我们已经知道,在运行时有各种值,比如数字 123,字符串 Hello world,所以通俗地讲,直接量就是在代码中把它们写出来的语法。

我们在类型部分,已经介绍过一些基本类型的直接量。比如,我们当时用 null 关键字获取 null 值,这个用法就是 null 直接量,这就是这里我们仅仅把它们简单回顾一下:

```
1 "abc";
2 123;
3 null;
4 true;
5 false;
```

除这些之外,JavaScript 还能够直接量的形式定义对象,针对函数、类、数组、正则表达式等特殊对象类型,JavaScript 提供了语法层面的支持。

```
1 ({});
2 (function(){});
3 (class{ });
4 [];
5 /abc/g;
```

需要注意,在语法层面,function、{和 class 开头的表达式语句与声明语句有语法冲突,所以,我们要想使用这样的表达式,必须加上括号来回避语法冲突。

在 JavaScript 标准中,这些结构有的被称作直接量(Literal),有的被称作表达式(\*\*Expression),在我看来,把它们都理解成直接量比较合适。

Primary Expression 还可以是 this 或者变量,在语法上,把变量称作"标识符引用"。

```
1 this;
2 myVar;
3
```

任何表达式加上圆括号,都被认为是 Primary Expression,这个机制使得圆括号成为改变运算优先顺序的手段。

```
1 (a + b);
2
```

这就是 Primary Expression 的几种形式了,接下来,我们讲讲由 Primary Expression 构成的更复杂的表达式:Member Expression。

## MemberExpression 成员表达式

Member Expression 通常是用于访问对象成员的。它有几种形式:

```
1 a.b;
2 a["b"];
3 new.target;
4 super.b;
5
```

前面两种用法都很好理解,就是用标识符的属性访问和用字符串的属性访问。而 new.target 是个新加入的语法,用于判断函数是否是被 new 调用,super则是构造函数中,用于访问父类的属性的语法。

从名字就可以看出,Member Expression 最初设计是为了属性访问的,不过从语法结构需要,以下两种在 JavaScript 标准中当做 Member Expression:

```
息复制代码
1 f`a${b}c`;
2
```

这是一个是带函数的模板,这个带函数名的模板表示把模板的各个部分算好后传递给一个函数。

```
1 new Cls();
```

另一个是带参数列表的 new 运算,注意,不带参数列表的 new 运算优先级更低,不属于 Member Expression。

实际上,这两种被放入 Member Expression,仅仅意味着它们跟属性运算属于同一优先级,没有任何语义上的关联。接下来我们看看 Member Expression 能组成什么。

# NewExpression NEW 表达式

这种非常简单,Member Expression 加上 new 就是 New Expression(当然,不加 new 也可以构成 New Expression,JavaScript 中默认独立的高优先级表达式都可以构成低优先级表达式)。

注意,这里的 New Expression 特指没有参数列表的表达式。我们看个稍微复杂的例子:

```
1 new new Cls(1);
2
```

直观看上去,它可能有两种意思:

```
1 new (new Cls(1));
2

B 复制代码

I new (new Cls)(1);
2
```

实际上,它等价于第一种。我们可以用以下代码来验证:

```
1 class Cls{
2   constructor(n){
3    console.log("cls", n);
4   return class {
5    constructor(n) {
6    console.log("returned", n);
7   }
8  }
9  }
10 }
```

```
12 new (new Cls(1));
13
```

这段代码最后得到了下面这样的结果。

```
1 cls 1
2 returned undefined
3
```

这里就说明了, 1 被当做调用 Cls 时的参数传入了。

## CallExpression 函数调用表达式

除了 New Expression, Member Expression 还能构成 Call Expression。它的基本形式是 Member Expression 后加一个括号里的参数列表,或者我们可以用上 super 关键字代替 Member Expression。

```
1 a.b(c);
2 super();
3
```

这看起来很简单,但是它有一些变体。比如:

```
1 a.b(c)(d)(e);
2 a.b(c)[3];
3 a.b(c).d;
4 a.b(c)`xyz`;
```

这些变体的形态,跟 Member Expression 几乎是——对应的。实际上,我们可以理解为,Member Expression 中的某一子结构具有函数调用,那么整个表达式就成为了一个 Call Expression。

而 Call Expression 就失去了比 New Expression 优先级高的特性,这是一个主要的区分。

# LeftHandSideExpression 左值表达式

接下来,我们需要理解一个概念: New Expression 和 Call Expression 统称 LeftHandSideExpression,左值表达式。

我们直观地讲,左值表达式就是可以放到等号左边的表达式。JavaScript 语法则是下面这样。

```
1 a() = b;
```

这样的用法其实是符合语法的,只是,原生的 JavaScript 函数,返回的值都不能被赋值。因此多数时候,我们看到的赋值将会是 Call Expression 的其它形式,如:

■ 复制代码

```
1 a().c = b;
```

另外,根据 JavaScript 运行时的设计,不排除某些宿主会提供返回引用类型的函数,这时候,赋值就是有效的了。

左值表达式最经典的用法是用于构成赋值表达式,但是其实如果你翻一翻 JavaScript 标准,你会发现它出现在各种场合,凡是需要"可以被修改的变量"的位置,都能见到它的身影。

那么接下来我们就讲讲 AssignmentExpression 赋值表达式。

# AssignmentExpression 赋值表达式

AssignmentExpression 赋值表达式也有多种形态,最基本的当然是使用等号赋值:

■复制代码

1 a = b

这里需要理解的一个稍微复杂的概念是,这个等号是可以嵌套的:

■复制代码

1 
$$a = b = c = d$$

这样的连续赋值,是右结合的,它等价于下面这种:

■ 复制代码

1 
$$a = (b = (c = d))$$

也就是说, 先把 d 的结果赋值给 c, 再把整个表达式的结果赋值给 b, 再赋值给 a。

当然,这并非一个很好的代码风格,我们讲解语法是为了让你理解这样的用法,而不是推荐你这样写代码。

赋值表达式的使用,还可以结合一些运算符,例如:

```
1 a += b;
```

相当于

```
1 a = a + b;
```

能有这样用的运算符有下面这几种:

```
*=, /=, %=, +=, -=, <<=, >>>=, &=, ^=, |=, **=
```

我想你已经注意到了,赋值表达式的等号左边和右边能用的表达式类型不一样,在这一课,我们已经关注完了表达式的左边部分(左值表达式)的语法结构,下一节课,我们将会给你重点讲解表达式的右边部分。

# Expression 表达式

赋值表达式可以构成 Expression 表达式的一部分。在 JavaScript 中,表达式就是用逗号运算符连接的赋值表达式。

在 JavaScript 中,比赋值运算优先级更低的就是逗号运算符了。我们可以把逗号可以理解为一种小型的分号。

```
1 a = b, b = 1, null;
2
```

逗号分隔的表达式会顺次执行,就像不同的表达式语句一样。"整个表达式的结果"就是"最后一个逗号后的表达式结果"。比如我们文中的例子,整个"a = b, b = 1, null;"表达式的结果就是","后面的null。

在很多场合,都不允许使用带逗号的表达式,比如我们在前面课程中提到,export 后只能跟赋值表达式,意思就是表达式中不能含有逗号。

#### 结语

这节课我们开始讲解了运算符和表达式的一些相关知识,这节课上,我们已经学习了赋值表达式 和赋值表达式的左边部分。下节课,我们将会讲一讲赋值表达式的右边部分。

最后给你留一个作业,把今天讲到的所有运算符按优先级排列成一个表格,下节课我们会补完剩下的部分。



由作者筛选后的优质留言将会公开显示,欢迎踊跃留言。

 Ctrl + Enter 发表
 0/2000字
 提交留言

#### 精选留言(2)



木木

f`a\${b}c`;

没有明白这个是什么意思

2019-04-04



木木

f`a\${b}c`;

2019-04-04