深入理解js运算符及其运算规律

jCodeLife
0.077 2020.07.19 21:10:04 字数 2,714 阅读 32

编辑文章

我们常常需要对一些数据处理，经过一定加工得出新的数据，而加工的过程就是运算。跟数学中的运算一样，不同的运算符有不同的运算规则。那么JS中哪些运算符呢，我们一个一个来看

一、算数运算符

算数运算符的运算跟我们数学中的运算规则并没有什么不同，包括：`+ - * / %`，分别对应加、减、乘、除、取余运算符

- 加号运算符

规则

1. 对数字代数求和
2. 对字符串进行连接

注意：
任何数据类型+字符串都等于字符串
字符串拼接时：
数值+字符串——>将数值直接转为字符串拼接

字符串+字符串——>两者直接拼接

```
> ""+""
< ""
> "1"+"1"
< "11"
> "1"+1
< "11"
```

- 减号运算符

1. 对数值进行减法操作；
2. 对数值进行“取反”操作；

3. 将数值型字符串转换成数值，非数值型字符串这值为NaN

```
> - ""
< -0
> - "1"
< -1
> - "-1"
< 1
> - "null"
< NaN
> - "a"
< NaN
```

- 乘号运算符

功能：对两个数进行乘法运算

jCodeLife
总资产5 (约0.53元)

深入call,apply,bind到手动封装
阅读 48

一篇深度剖析DOM元素真实面目
阅读 10

深入理解包装类及其所以常用方法和属性
阅读 13

推荐阅读

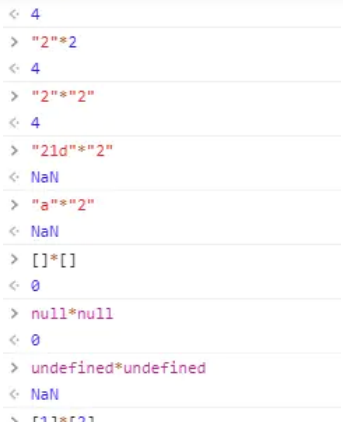
这次，彻底弄清js的继承方式
阅读 478

javascript new一个对象的时候，内部发生了什么
阅读 863

学Vue，就要学会vue JSX（一）
阅读 333

TypeScript基础学习总结
阅读 18

web前端--10个妨碍进步的学习方式
阅读 429



规律：

1. 两个数字相乘得正常乘积
2. 其中两个为数值型字符串或者一个数字一个数值型字符串，将数值型字符串变成对应数字，然后得出正常乘积。
3. 若有一个为非数值型字符串，那么乘积为NaN
4. 空数组`*`空数组、null`*`null、空数组`*`null乘积都为0
5. undefined*undefined乘积为NaN
6. 数组相乘，若都为空数组，乘积为0；若两数组均只有一位且为数值，则直接数值相乘。否则值为NaN
7. 两个对象相乘值为NaN，若一个对象一个其他类型，则会报错：Uncaught SyntaxError

- 除号运算符

  对两个数进行除法运算

1. 0除任意正数或任意正数值型字符串都得0；如：

   `0/1或者0/"1" -> 0`
2. 0除任意负数或任意负数值型字符串都得-0；如：

   `0/-1或者0/"-1" ->-0`
3. 任意正数或任意正数值型字符串除以0，都得Infinity；如：

   `1/0或者"1"/0 ->Infinity （正无穷 属于number）`
4. 任意负数或任意负数值型字符串除以0，都得-Infinity；如：

   `-1/0或者"-1"/0 ->-Infinity （负无穷 属于number）`
5. 0/0 ->NaN（非数，也属于number类型）
6. 0/null或者null/0都为NaN
7. 任意正数或任意正数值型字符串除以null,都得Infinity
8. 任意负数或任意负数值型字符串除以null，都得-Infinity
9. null除以任意正数或任意正数值型字符串或者正单位数组或者正单位字符串数组都得0
10. null除以任意负数或任意负数值型字符串或者负单位数组或者负单位字符串数组都得-0
11. null除以非数字或者非数字型字符串或者非正负单位数组或者正负单位字符串数组都得NaN

注：单位数组是指只有一位的数组，正单位数组是指单位数组且该数为正数

- 取余运算符

  两个数相除取余数。余数是正是负仅看第一个数，与第一个运算数的符号一致

  如

```
2 % -3  ----> 2
```

- 复合赋值运算符

  +=、 -= 、*= 、 /=、 %=、分别相加、相减、相乘、相除和取余后赋值

- 自增、自减运算符

```
++ //自增
-- //自减
```

对唯一的运算数进行递增或者递减操作（每次加1或者减1）
注意：

1. 运算数必须是一个变量、数组的一个元素或者对象的属性
2. 如果运算数是非数值则运算符会将它转成数值直接返回，不会再+或-
3. 运算符的位置前后会影响运算结果：
   若运算数之前——>则先进行递增或者递减，后执行该语句
   若运算数之后——>先执行该语句求值，后进行递增或者递减操作
   如：

```
var a = 5;
console.log(a++);//++在后，先执行完这个语句，后++，所以得1
console.log(a);//这里结果变成2了，以为上面语句执行完后a是自增了的。
console.log(++a);//先++，所以变成3，再执行打印
console.log(a);//结果还是3，不变化了。
```

备注：
赋值的顺序是从右往左，计算的顺序是从左往右
计算的优先级：
若有++或--且在运算数前，先自增自减后乘除后加减，有()先计算()里的
"="最弱，"()"优先级最高

2、比较运算符

比较运算符大致可分三类，分别是：大小比较运算符、等值运算符、相同运算符。
比较运算符比较的结果为boolean值，true / false

- 大小比较运算

```
> < >= <=
```

如果左边 大于/小于/大于等于/小于等于 右边，返回true，否则返回false
规律：

1. 数值与数值比较——>比较数值大小
2. 仅一个运算数是数值——>将另一个运算数转成数值，并比较它们的代数值;若转成NaN，则比较的结果为false
3. 字符串与字符串——>逐个字符比较它们的Unicode数值
4. 字符串与非数值——>将非数值运算数转换成字符串，然后两个字符串进行Unicode比较
5. 运算数即非数字也非字符串——>看具体情况，转换成数字或者字符串后进行比较
6. 运算数无法转换成数值或者字符串——>返回false

```
==   //等于
!=  //不等于
```

比较两个值是否相等或不等，相等为true，不等为false
存在隐式类型转换，如：
布尔值：true-----1；false-----0;
比较规律：

1. null与undefined相等
2. NaN不等于任何数值，包括自身
3. 对象，是不是同一对象，是的话 == 是true
4. 字符串与数值比较：字符串隐式转换成数值，然后数值比较。

- 相同比较

```
===   //绝对等于
!==    //绝对不等于
```

不存在隐示类型转换的绝对等于和绝对不等于，比较两个运算数的返回值及数据类型是否相同或不同
比较规律：

1. 只有数据类型和数值相同才为相同
2. 原始值和引用值之间是绝对不等于
3. 引用值之间比较的是它们的引用（内存地址）

3、逻辑运算符

```
&& //与，表示并且
|| //或，表示或者
! //非，表示取反
```

- 逻辑与&&
  规律：

1. 第一个操作数是对象，返回第二个；
2. 第一个操作数值为true，第二个为对象时，返回第二的对象；
3. 两个都是对象时，返回第二个；
4. 其中一个操作数是null或者undefined或者NaN时，对应返回null、undefined、NaN，如果同时是这三个值中的两个，返回第一个数；（暂时感觉0也是这个规律）

特性：
1.当且仅当两个运算数的值为true时，返回true，否则返回false；
2.短路操作：当第一个操作数的值是false时，则不再对第二个操作数进行求值了，返回第一个。

- 逻辑或||
  规律：

1. 第一个操作数是对象，返回第一个；
2. 第一个操作数值为false，返回第二个

特性：

5. 当且仅当两个运算数的值为false时，返回false，否则返回true；
6. 短路操作：当第一个操作数的值是true时，则不再对第二个操作数进行求值了，返回第一个。

小结与或运算符：
个人觉得，其实不管是2个数还是更多数运算，只要记住一个规律：
无论是与还是或，只要返回能确定表达式真假的值即可
什么意思？什么叫返回能确定表达式真假的值
举个例子：与&&的规律

1. 第一个操作数是对象，因为第一个值转成boolean值是true，那我返回第二个数必定能返回真假；
2. 第一个操作数值为true，第二个为对象时，返回第二的对象；其实都不用管他是不是对象，因为第一个为true了，这个表达式的值取决于第二个数。
3. 两个都是对象时，返回第二个；也是一样
4. 其中一个操作数是null或者undefined或者NaN时，对应返回null、undefined、NaN，因为与运算符一个为false就为false了，这三个值都能确定表达式最后的值。如果同时是这三个值中的两个，返回第一个数，因为我看第一个就能知道是false了；

- **逻辑非!**

规律：

1. 非运算符返回值的为true的有：undefined、null、NaN、0、""
2. false的有：对象、非空字符串、非0数值

特性：

1. 当运算的值为false则返回true，否则返回false。
2. 连续使用两次，可将任意类型转成布尔值（true or false）。

4、对象运算符

1. in

> prop in object

判断左边属性是否在右边对象中或是否在其原型链中，如在就返回true
不分是自己自身的还是原型的，都算
如果数组，需要注意判断的是索引号,而不是数组元素的值

2. instancesof

> A instanceof B

判断A对象是否是B构造函数构造出来的；更深刻的理解应该是看A对象的原型链上有没有B构造函数的原型

3. new：创建并初始化一个新的对象

5、其他运算符

1. 三目运算符

   `条件判断 ？ 条件判断为true时返回 ： 条件判断为false时返回`

   有返回值，是增强版的if else
2. typeof

   类型判定运算符

   能返回6种数据类型：number、string、boolean、undefined、function、object

   注意因为历史遗留问题，typeof(null)返回的是"object"，表示空对象
3. 逗号操作符

   表达式中使用逗号操作符，如（1,2）会看一眼第一个，然后返回第二个数2
4. 位运算符