5. JavaScript运算符

1. 按照操作元数的个数不同分为：一元运算符、二元运算符和三元运算符：

- 如下一元运算符

```
delete：用于删除对象中属性的    如：delete o.name; //删除o对象中的name属性
++ -- ： 一元加法和一元减法
```

2. 按照种类划分又分为如下运算符：

① 算数运算符：

- 算数运算符用于对数字执行算数运算：

运算符	描述
`+`	加法
`-`	减法
`*`	乘法
`/`	除法
`%`	（取模）求余
`++`	递加 (区分: 前置++ 和 后置++)
`--`	递减 (区分: 前置-- 和 后置--)

- 注意： 其中+号具有两重意思：字串连接和数值求和。
- 就是加号”+“两侧都是数值则求和，否则做字串连接

② 赋值运算符

- 赋值运算符向 JavaScript 变量赋值。

运算符	例子	等同于
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

③ 比较运算符

运算符	描述
==	等于
===	等值等型（值相等并且类型相等为true）
!=	不相等
!==	不等值或不等型（值不相等或类型不相等为true）
>	大于
<	小于
>=	大于或等于
<=	小于或等于
? :	三元运算符

④ 逻辑运算符

运算符	描述
&&	逻辑与
¦¦	逻辑或
!	逻辑非

⑤ 位运算符

- 位运算符处理 32 位数。
- 该运算中的任何数值运算数都会被转换为 32 位的数。结果会被转换回 JavaScript 数。

运算符	描述	例子	等同于	结果	十进制
`&`	与	`5 & 1`	`0101 & 0001`	0001	1
¦	或	5 ¦ 1	0101 ¦ 0001	0101	5
`~`	非	`~ 5`	`~0101`	1010	10
`^`	异或	`5 ^ 1`	`0101 ^ 0001`	0100	4
`<<`	零填充左位移	`5 << 1`	`0101 << 1`	1010	10
`>>`	有符号右位移	`5 >> 1`	`0101 >> 1`	0010	2
`>>>`	零填充右位移	`5 >>> 1`	`0101 >>> 1`	0010	2

- 上例使用 4 位无符号的例子。但是 JavaScript 使用 32 位有符号数。
- 因此，在 JavaScript 中，`~5` 不会返回 10，而是返回 -6。
- `~00000000000000000000000000000101` 将返回 `11111111111111111111111111111010`。
- 无符号位移（>>>）和有符号位移（>>）的区别是
    - 有符号位移运算时如果数字为正数时位移后在前面补0，为负数时则在位移后在前面补1

⑥ 条件运算符（三元运算符）：

- JavaScript 也包含了可基于某些条件向变量赋值的条件运算符。

```
语法
variablename = (condition) ? value1:value2

实例
var voteable = (age < 18) ? "太年轻":"足够成熟";
```

⑦ 逗号运算符

```
用逗号运算符可以在一条语句中执行多个运算。
var iNum1=1, iNum2=2, iNum3=3;
```

⑧ 类型运算符

运算符	描述
typeof	返回变量的类型。
instanceof	返回 true，如果对象是对象类型的实例。

- instanceof 运算符与 typeof 运算符相似，用于识别正在处理的对象的类型。
- 与 typeof 方法不同的是，instanceof 方法要求开发者明确地确认对象为某特定类型。

```
var oStringObject = new String("hello world");
console.log(oStringObject instanceof String);   // 输出 "true"


// 判断 foo 是否是 Foo 类的实例
function Foo(){}
var foo = new Foo();
console.log(foo instanceof Foo)//true


// 判断 foo 是否是 Foo 类的实例 , 并且是否是其父类型的实例
function Aoo(){}
function Foo(){}
Foo.prototype = new Aoo();//JavaScript 原型继承

var foo = new Foo();
console.log(foo instanceof Foo)//true
console.log(foo instanceof Aoo)//true
```

3 运算符的优先级

```
优先级从高到底
1. ()  优先级最高
2. 一元运算符  ++   --   !
3. 算数运算符  先 *  /  %   后 +   -
4. 关系运算符  >   >=   <   <=
5. 相等运算符  ==   !=    ===    !==
6. 逻辑运算符  先 &&   后||
7. 赋值运算符  =  +=  -=  *=  /=  %=
```