ES6 PPT1606172: Number/Math 扩展

Thursday, November 5, 2015 10:15 AM

二进制与八进制表示：

1. 二进制：15 === 0b1111
2. 八进制：15 === 0o17

Number 对象扩展：

1. 使用 Number.isFinite() 检查参数值是否非无穷：
```
console.log(Number.isFinite(15), Number.isFinite(Infinity), Number.isFinite('foo')); // true, false, false
console.log(Number.isFinite('15'), isFinite('15')); // false, true
```
2. 使用 Number.isNaN() 检查参数值是否非数值：
```
console.log(Number.isNaN('15'), Number.isNaN(true), Number.isNaN('a'/'b')); // false, false, true
console.log(Number.isNaN('NaN'), isNaN('NaN')); // false, true
```
3. 使用 Number.parseInt()/Number.parseFloat() 转换数值：
```
console.log(Number.parseInt('3px'), Number.parseFloat('3.14')); // 3, 3.14
```
4. 使用 Number.isInteger() 判定是否为整数：
```
console.log(Number.isInteger(15), Number.isInteger(15.0), Number.isInteger('15')); // true, true, false
```
5. 使用 Number.EPSILON 极小常量判定含误差的两值是否相等：
```
function isEqual(left, right) { return Math.abs(left - right)<Number.EPSILON; }
console.log(0.1+0.2-0.3, 0.1+0.2==0.3, isEqual(0.1+0.2, 0.3)); // 5.551115123125783e-17, false, true
```
6. 使用 Number.isSafeInteger() 判定是否为安全整数：
```
console.log(Number.MAX_SAFE_INTEGER === Math.pow(2, 53) - 1); // true
console.log(Number.MIN_SAFE_INTEGER === -Number.MAX_SAFE_INTEGER); // true
['a', NaN, Infinity, 1.0, 1.2, '3'].forEach((i)=>console.log(Number.isSafeInteger(i))); // false, false, false, true, false, false
```
7. 总结：与 ES5 方法不同，Number 扩展下的方法行为大多不进行首步的数值类型转换

Math 对象扩展：

1. 使用 Math.trunc() 去除数值小数部分：
```
Math.trunc(4.1) // 4
Math.trunc('12345.678') // 12345
```
2. 使用 Math.sign() 判定数值的正负状态：
```
Math.sign(-5) // -1
Math.sign(5) // +1
Math.sign(0) // +0
Math.sign(-0) // -0
Math.sign(NaN) // NaN
Math.sign('foo'); // NaN
Math.sign(); // NaN
```
3. 使用 Math.cbrt() 计算数值的立方根：
```
Math.cbrt(1)  // 1
Math.cbrt('8') // 2
Math.cbrt('hello') // NaN
```
4. 使用 Math.clz32() 返回数值的32位二进制形式前导0的个数：
```
Math.clz32(1000) // 22
Math.clz32(0b01000000000000000000000000000000) // 1
Math.clz32(0b00100000000000000000000000000000) // 2
```
5. 使用 Math.imul() 确保大数值乘法结果的低位精度：
```
(0x7fffffff * 0x7fffffff)|0 // 0
Math.imul(0x7fffffff, 0x7fffffff) // 1
```
6. 使用 Math.fround() 返回数值的单精度浮点数形式：
```
Math.fround(1) // 1
```

```
Math.fround(1.337) // 1.3370000123977661
Math.fround(1.5) // 1.5
Math.fround(NaN) // NaN
```

7. 使用 Math.hypot() 返回所有参数平方和的平方根：

```
Math.hypot(3, 4); // 5
Math.hypot(3, 4, 5); // 7.0710678118654755
Math.hypot(); // 0
Math.hypot(NaN); // NaN
Math.hypot(3, 4, 'foo'); // NaN
Math.hypot(3, 4, '5'); // 7.0710678118654755
Math.hypot(-3); // 3
```

8. 使用对数方法 Math.expm1(), Math.log1p(), Math.log10(), Math.log2()：

```
Math.expm1(-1) // -0.6321205588285577
Math.log1p(1) // 0.6931471805599453
Math.log10(2) // 0.3010299956639812
Math.log2(1024) // 10
```

9. 使用三角函数方法 Math.sinh(), Math.cosh(), Math.tanh(), Math.asinh(), Math.acosh(), Math.atanh()：

```
Math.sinh(1) // 1.1752011936438014
Math.cosh(1) // 1.5430806348152437
Math.tanh(1) // 0.7615941559557649
Math.asinh(1) // 0.8813735870195429
Math.acosh(1) // 0
Math.atanh(1) // null
```

10. 使用 ES7 提案之指数运算符 (**)：

```
2 ** 3; // 8
var a = 4;
a ** = 4; // a === 256
```