// Numbers

typeof 37 === 'number';

typeof 3.14 === 'number';

typeof Math.LN2 === 'number';

typeof Infinity === 'number';

typeof NaN === 'number'; // 尽管NaN是"Not-A-Number"的缩写

typeof Number(1) === 'number'; // 但不要使用这种形式!

// Strings

typeof "" === 'string';

typeof "bla" === 'string';

typeof (typeof 1) === 'string'; // typeof总是返回一个字符串

typeof String("abc") === 'string'; // 但不要使用这种形式!

// Booleans

typeof true === 'boolean';

typeof false === 'boolean';

typeof Boolean(true) === 'boolean'; // 但不要使用这种形式!

// Symbols

typeof Symbol() === 'symbol';

typeof Symbol('foo') === 'symbol';

typeof Symbol.iterator === 'symbol';

// Undefined

typeof undefined === 'undefined';

typeof declaredButUndefinedVariable === 'undefined';

typeof undeclaredVariable === 'undefined';

// Objects

typeof {a:1} === 'object';

// 使用Array.isArray 或者 Object.prototype.toString.call

// 区分数组,普通对象

typeof [1, 2, 4] === 'object';

typeof new Date() === 'object';

// 下面的容易令人迷惑，不要使用！

typeof new Boolean(true) === 'object';

typeof new Number(1) === 'object';

typeof new String("abc") === 'object';

// 函数

typeof function(){} === 'function';

typeof class C{} === 'function'

typeof Math.sin === 'function';

typeof new Function() === 'function';

typeof null === 'object'; // 从一开始出现JavaScript就是这样的

**正则表达式**

***对正则表达式字面量的类型判断在某些浏览器中不符合标准*：**

typeof /s/ === 'function'; // Chrome 1-12 , 不符合 ECMAScript 5.1

typeof /s/ === 'object'; // Firefox 5+ , 符合 ECMAScript 5.1

**暂存死区**

在 ECMAScript 2015 之前，typeof总是保证为任何操作数返回一个字符串。但是，除了非提升，块作用域的let和const之外，在声明之前对块中的let和const变量使用typeof会抛出一个ReferenceError。这与未声明的变量形成对比，typeof会返回“undefined”。块作用域变量在块的头部处于“暂时死区”，直到被初始化，在这期间，如果变量被访问将会引发错误。

typeof undeclaredVariable === 'undefined';

typeof newLetVariable; let newLetVariable; // ReferenceError

typeof newConstVariable; const newConstVariable = 'hello'; // ReferenceError

**例外**

所有当前的浏览器都暴露了一个类型为 undefined 的非标准宿主对象 document.all。

typeof document.all === 'undefined';

尽管规范允许为非标准的外来对象定制类型标签，但它要求这些类型标签与预定义标签不同。document.all的类型标记为“undefined”的情况必须被列为违反规则的特殊情况。