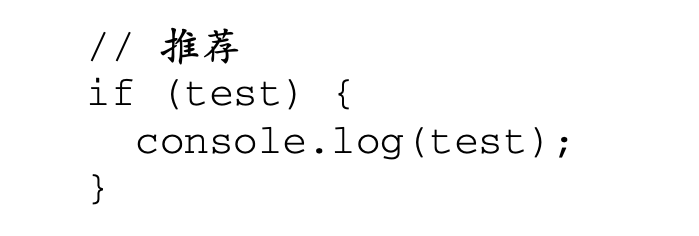
# 第三章 语言基础

未读：模板字面量标签函数

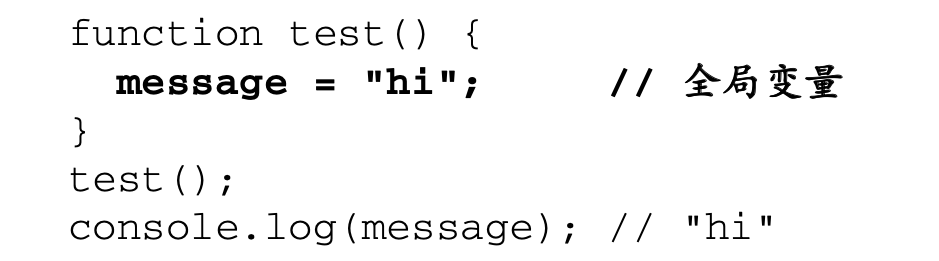
1、js 中的内置对象主要指的是在程序执行前存在全局作用域里的由 js 定义的一些**全局值属性**、**函数**和用来实例化其他对象的**构造函数对象**。

一般我们经常用到的如全局变量值 NaN、undefined，全局函数如 parseInt()、parseFloat() 用来实例化对象的构 造函数如 Date、Object 等

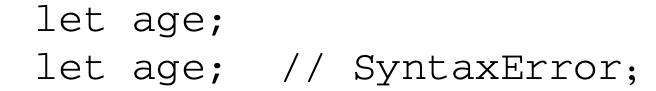
2、推荐在控制语句中即使只有一条执行语句，也使用代码块

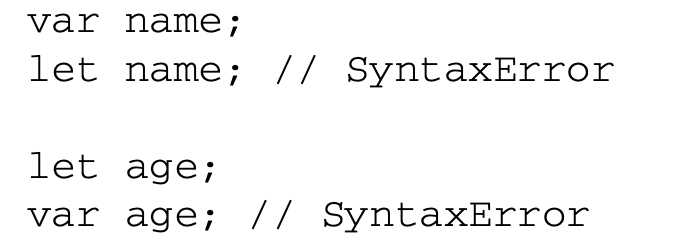


3、以下情况严格模式下会报错



4、同一个作用域下，只有var能重复声明变量，两个let，一个var一个let都会报错





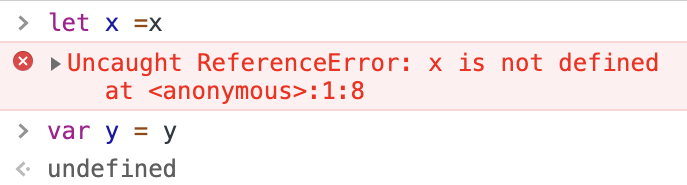
5、let的暂时性死区：进入作用域后，let声明的变量其实一开始就被创建了，但不允许被访问，只有等到声明变量的那一行代码出现，才可以获取和使用该变量。在进入作用域到变量被访问的这区间就是**暂时性死区**。

原来es5可以使用typeof检测未定义的变量，当返回undefined时表示未定义。

但是在const/let定义的变量在变量声明之前如果使用了typeof就会报错。

typeof x; // ReferenceError  
  
let x;

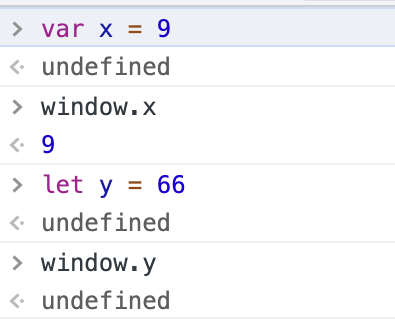
另外一个死区陷阱



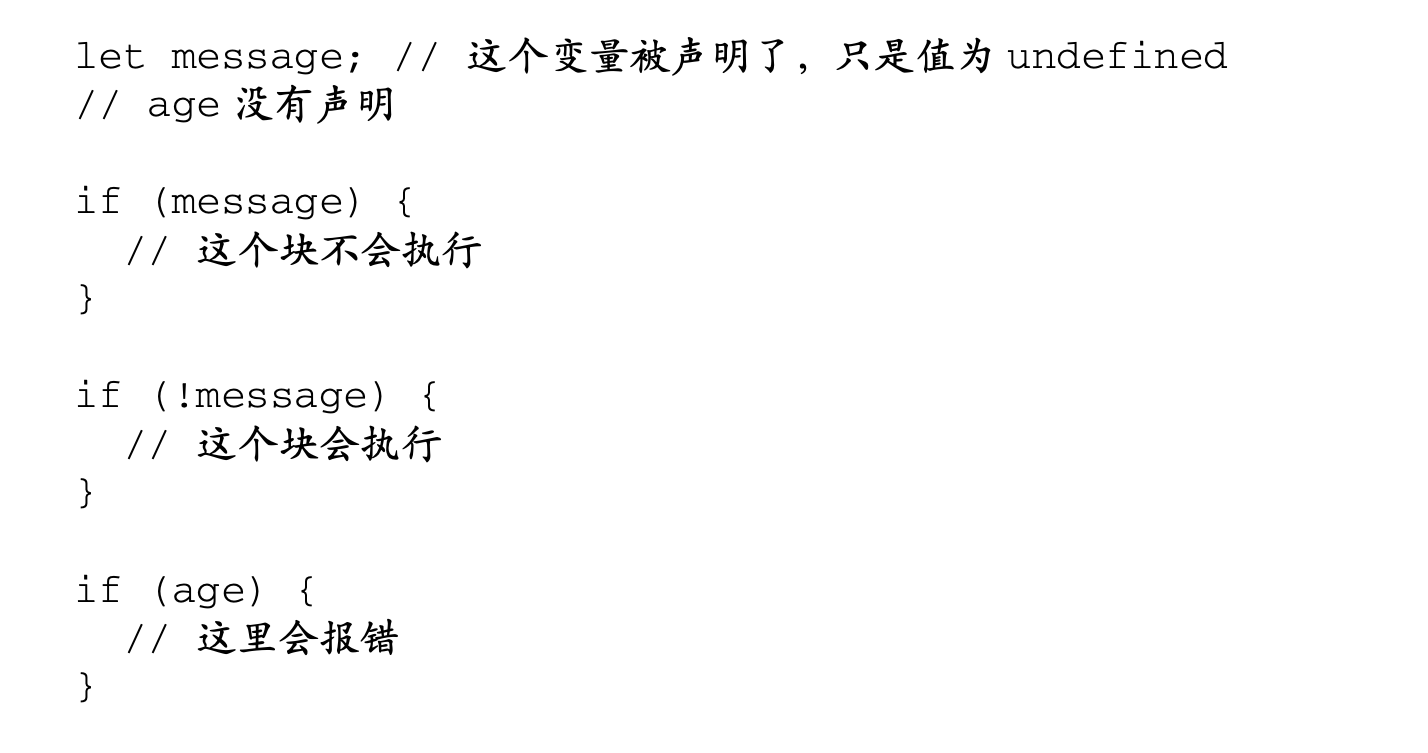
6、var a=b=5; 等价于 var a=(b=5);

其中只有a被声明了，b是自动解析为全局变量了。

7、var在全局作用域声明的变量会成为window的属性，而let声明的变量不会



8、undefined的使用



9、js高级程序设计定义

简单数据类型：Undefined、Null、Boolean、Number、 String 和 Symbol。

复杂数据类型：Object(对象)

10、使用八进制和十六进制格式创建的数值在所有数学操作中都被视为十进制数值。

11、

let floatNum1 = 1. ; // 小数点后面没有数字，当成整数 1 处理

let floatNum2 = 10.0 ; // 小数点后面是零，当成整数 10 处理

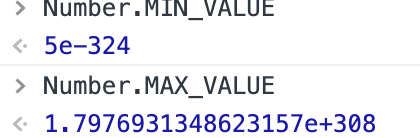
12、Number采用了IEEE754数值来表示，浮点值的精确度最高可达 17 位小数，但在算术计算中远不如整数精确。如

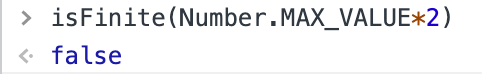
0.1+0.2 // =》 0.30000000000000004

因此永远不要测试某个特定的浮点值。

if (a + b == 0.3) { // 别这么干!   
  
console.log("You got 0.3.");  
  
}

13、JavaScript可表示的数值在Number.MIN\_VALUE和Number.MAX\_VALUE之间，超出会变成Infinity和-Infinity。可以用isFinite()判断一个值是不是有限大。

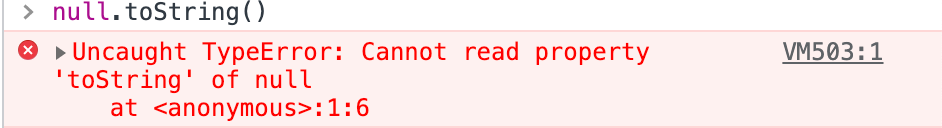




14、parseInt和parseFloat补充

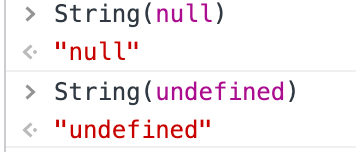
let num2 = parseInt(""); //NaN  
let num6 = parseInt("0xf"); // 15，解释为十六进制整数  
//parseInt可接第二个参数（进制数）  
let num1 = parseInt("AF", 16); // 175  
let num2 = parseInt("AF"); // NaN  
  
//parseFloat只解析十进制，没有第二个参数  
let num2 = parseFloat("0xA"); //0  
let num4 = parseFloat("22.34.5") //22.34  
let num6 = parseFloat("3.125e7"); //31250000

15、null和undefined没有toString方法



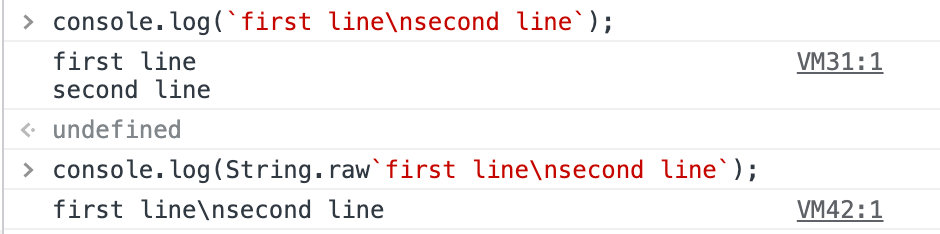
String()函数,如果不是null和undefined，则直接返回toString()的结果

如果是null和undefined，则返回“null”和“undefined”

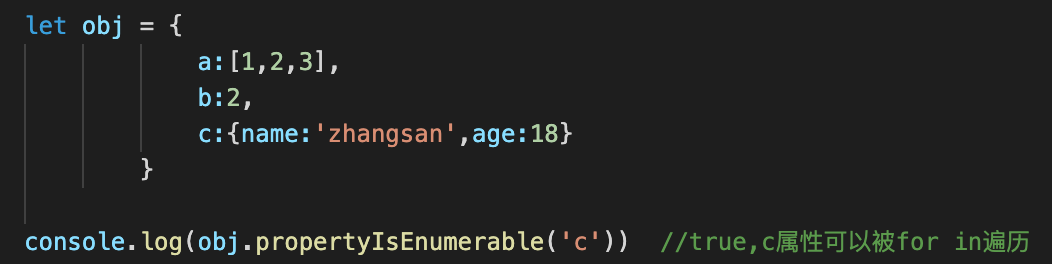


16、原始字符串

String.raw()，相当于Python字符串前缀r



17、Object

* propertyIsEnumerable(propertyName):用于判断给定的属性是否可以使用for-in 语句枚举。
* 

18、操作符

++ --，若遇到不是数字类型的值，则转为数字类型再进行运算

<、>、>=、<=比较规律和==一样，若两边数据类型不同，则转为数值再进行比较。（※如果任一一边是Object，则调用其valueOf()方法，如果没有valueOf则调用toString()方法,再按用来规则比较）

任何关系操作符去比较NaN，结果都是false ,NaN == NaN结果为false

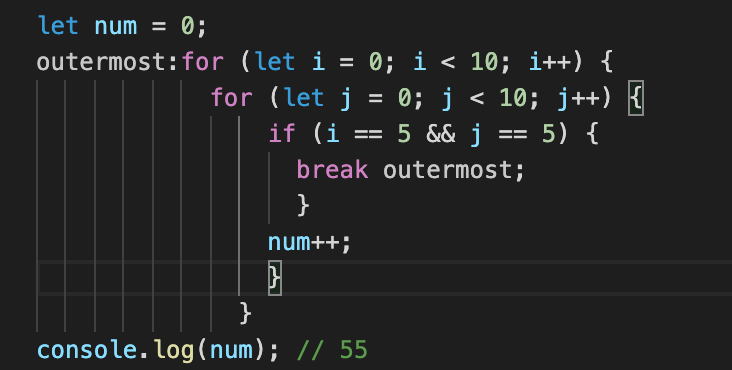
let result = "a" < 3; // 因为"a"会转换为NaN，所以结果是false

19、+=、-=这些操作符仅仅是简写语法，使用它们不会提升性能

20、无法通过 while 循环实现的逻辑，同样也无法使用 for 循环实现。因此 for 循环只是将循环相关 的代码封装在了一起而已。

21、标签语句的用法

给for语句加标签



当i和j都等于5时，break了最外层的for循环，输出55

22、with语句（严格模式报错，且不推荐使用）

