###左右查询

变量的查询规则（以后一定要与属性的查找共同总结）

左:等号的左边

整条作用域链中没有找到该变量的声明--->全局作用域主动声明一个

右:等号的非左边

整条作用域链中没有找到该变量的声明--->报错（ReferenceError:作用域判别失效有关）

什么是 typeError:运行时异常，就是莫名其妙操作 b 虽然系统会给 var 一个 但是没有值想、

###提升

1.函数的提升优于变量的提升

2.函数的提升是整体的提升,变量的提升是声明的提升

3.提升是指提升到本层作用域的最顶层

4.在块内部不要定义函数、

变量的提升不会理会 if else 这种条件暗示

###this 规则

主线:函数的调用位置上的调用形式!!!

独立调用 ----> 默认绑定规则 ----> this 绑给 window

在严格模式底下会绑给 undefined

对象点的形式 -----> 隐式绑定规则 ----> this 绑给最近那个调用对象

隐式丢失:

以隐式绑定的形式去赋值或传参，最终使用独立调用形式去调用

this 使用了默认绑定规则，导致 this 绑给了 window，

违背了开发者在赋值和传参时的意图

call apply bind ----> 显示绑定规则 ----> this 绑给指定的对象

bind:硬绑定函数

它可以为目标函数返回一个硬绑定函数，

不管硬绑定函数使用何种形式调用，目标函数的

this 永远为指定的对象。

new 调用 ----> new 绑定 -----> this 绑给构造出来的实例对象

怎么将伪数组转成真正的数组

Aarry.apply(null,{})

绑定的优先级

new 绑定 > 显示绑定 > 隐式绑定 > 默认绑定

绑定例外

es6 中胖箭头 this 指向与我们现在的规则不一样

被忽略的 this apply call bind（null） this----> window

柯里化:为函数去预绑定参数

var obj = Object.create（null）

###值与引用

javascript 中，所有的传递都是值传递!!!!

我们所说的引用传递:本质是引用值得传递!!

###==操作符的规则

将数据类型分成两拨

有: string number boolean object

无: null undefined

有和无永远不等

无和无永远相等

有和有（趋于数字化）

NaN 不等于 NaN

基本数据类型之间作比较 调用 number 函数将等号两边的基本数据类型值统一转为数字

比较对象中存在引用数据类型 向调引用数据类型的 valueof

如果是基本数据类型则进行比较

如果不是继续调用 tostring

如果是基本数据类型则进行比较

如果是不基本数据类型则报错

###定时器

定时器准不准？

不准。。。。

浏览器的模型是多进程多线程的

js 引擎的调用，浏览器主进程的主线程来调用的

js 引擎的主线程来执行 js 代码

遇到定时器，js 引擎的主线程创建一个分线程去执行异步操作

这时 js 引擎的主线程继续底下 js 代码的解析

什么时候定时器的回调被执行？

等 js 引擎的主线程完成所有 js 代码的解析工作后，这个它去循环异步队列