借助浏览器入门编程

下面是编程的基本概念：

变量 variable

计算机本质就是数据的运算工具，这个数据除了数学上的数字是数据各种文字啊图片啊都是数据，而由于在计算机里一切数据几乎都可以变，这些数据就被称为变量

function 函数

函数就是一个个小的程序

if 如果。。就干啥

简而言之就是判断是否满足一个条件，满足就干指定的事

while 当。。就干啥

简而言之就是判断是否满足一个条件，满足就不停反复地干指定的事直到这个条件不满足

现在让我们用浏览器来实现这些功能：

如果有chrome

这个

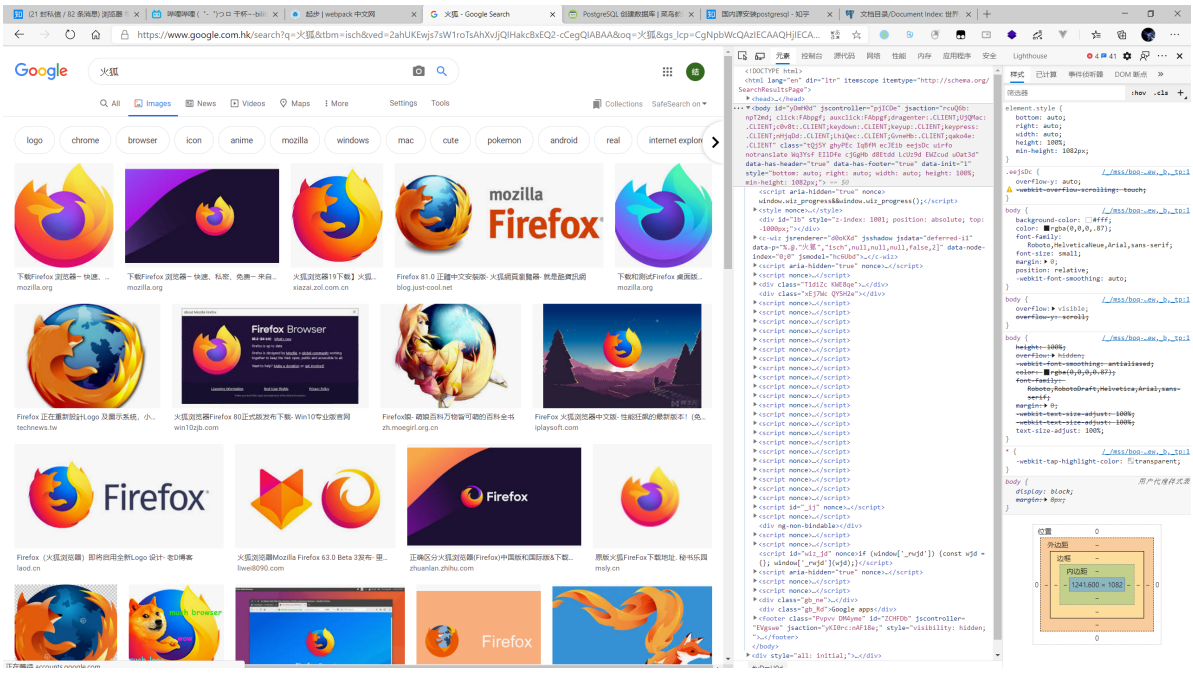
或者：

请用上面两个浏览器

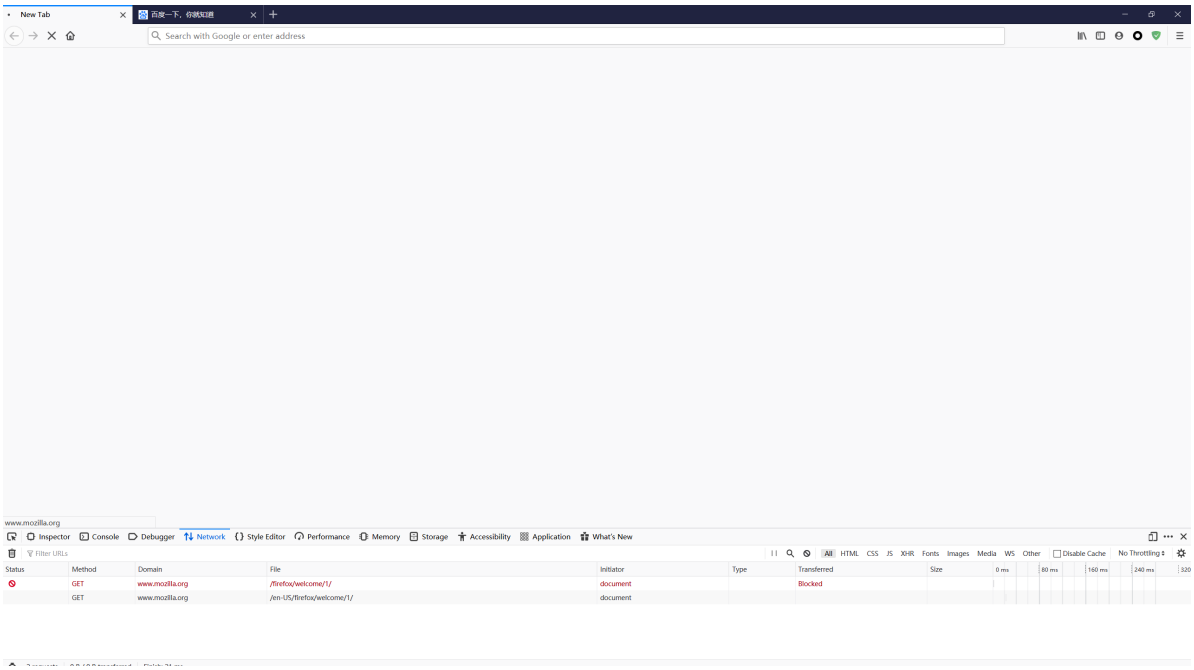
否则使用最原始的默认的edge浏览器就行（前提你是windows10）

浏览器无论什么页面无论有没有网 按下f12（有些笔记本需要同时按下左下角的fn建）

会



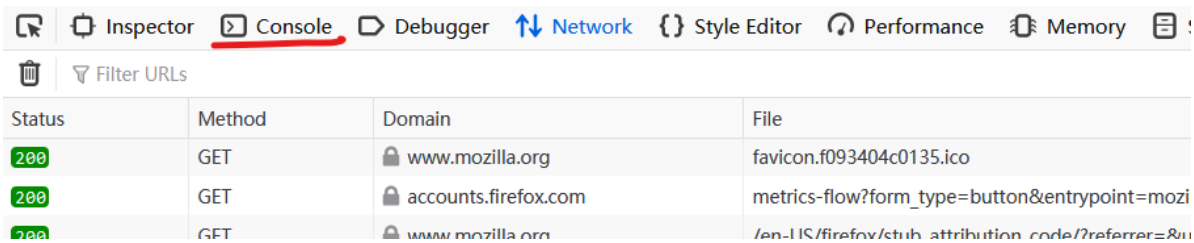
这样 。或者



这样

，反正就会跳出一个框

选择里面的控制台或者 console



ok 你可以开始编程啦!

不要被里面的内容吓到，，大多数是你打开的这个网站自己故意弄的，或者自己出错了，不用管他

第0步 加减乘除

```
+ - * /
```

第一步 变量:

变量的英文 variable

所以简写 var

```
var a // 创建一个变量 （//后面的是注释，不用写进去）
```

创建完后可以给这个变量赋予一个值，请随意

```
a = 1
a = 1.2
a = '我是一串文字' //注意文字一定要引号，单引号双引号在本编程语言里随意 不然无法和代码的其他部分区分开来 ，同样 ，如果你代码被放到引号里了，那代码也就只是一串文字了
```

注意 编程里的等于号不是数学上的意义，你可以理解为让 a 等于 1 但是发明编程语言的人懒得写让（let） 其实本质就是把后面的数据存到变量里（这里变量因为刚创建所以是空，可以理解为一个大箱子，然后把数据放进去）

上面两步走可以合并一下：

```
var a = 1
var a = 1.2
var a = 'hhhhhhhhh, 你好'
```

ok 下面我们来实际操作下：

在浏览器中输入代码：

输入 var a = 1 回车

输入 var b = "hhhh" 回车

请忽略灰色的undined

接下来我们要查看刚的两个变量

我们直接 输入变量 的名字 a或者b回车就可以查看：

简单的变量就这样啦~但是以后的编程中变量会时时刻刻陪伴你，而且变量也不是只有数字和字母文字那么简单，甚至可以是个代表地球、代表某些玄学非常或者抽象东西的变量 我们接下来接触的程序就是也是个神奇的变量。

函数（function）

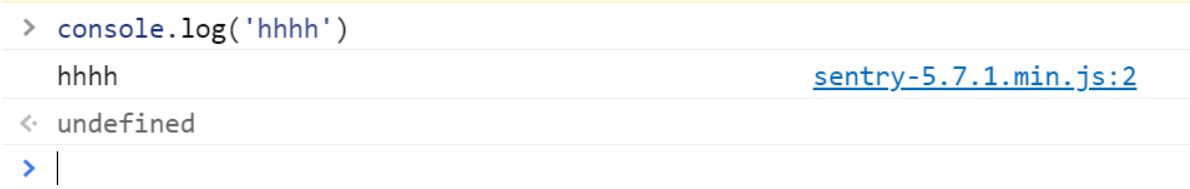
和数学上的函数不同，编程里的函数就是一段小程序

而且函数里可以套函数 又可以套，无限套，这样一个大的功能就可以分为几步完成，每一步又可以分为更小的一步，这样下去

但又为了和数学上的函数相似（好吧其实是为了函数间相互配合方便）函数又可以像数学上的函数（y = f（x））一样有 x这个输入，y这个输出，不过很多时候没有y这个输出，同时 y这个输出还有个专业的名字叫 **返回**

ok 为了我们接下来的学习，先介绍一个浏览器内定的函数（内定的意思就是前辈大佬们防止我们从0101这种开始起步而直接给我们写好的一些基本程序）

```
console.log('要在屏幕上打印的东西')
```



请注意这里面hhhh是单独的前面没有向左向右箭头的，这是真正的所谓的打印

可以比较下直接一个变量的值回车和打印的区别（在这里就是多了一个向左的箭头而已，不要去想为啥）

这个函数的名字是console.log 由于这个世界上有至少几亿个函数，名字早晚会用光，所以这个函数也被分入了很多个不同的东西里。这个函数真正的名字就叫log，但是它属于console这个东西（console本身也是个变量，只不过极其复杂）所以我们用这种方式来叫出它

- 还有个好玩的函数，你们可以尝尝尝鲜

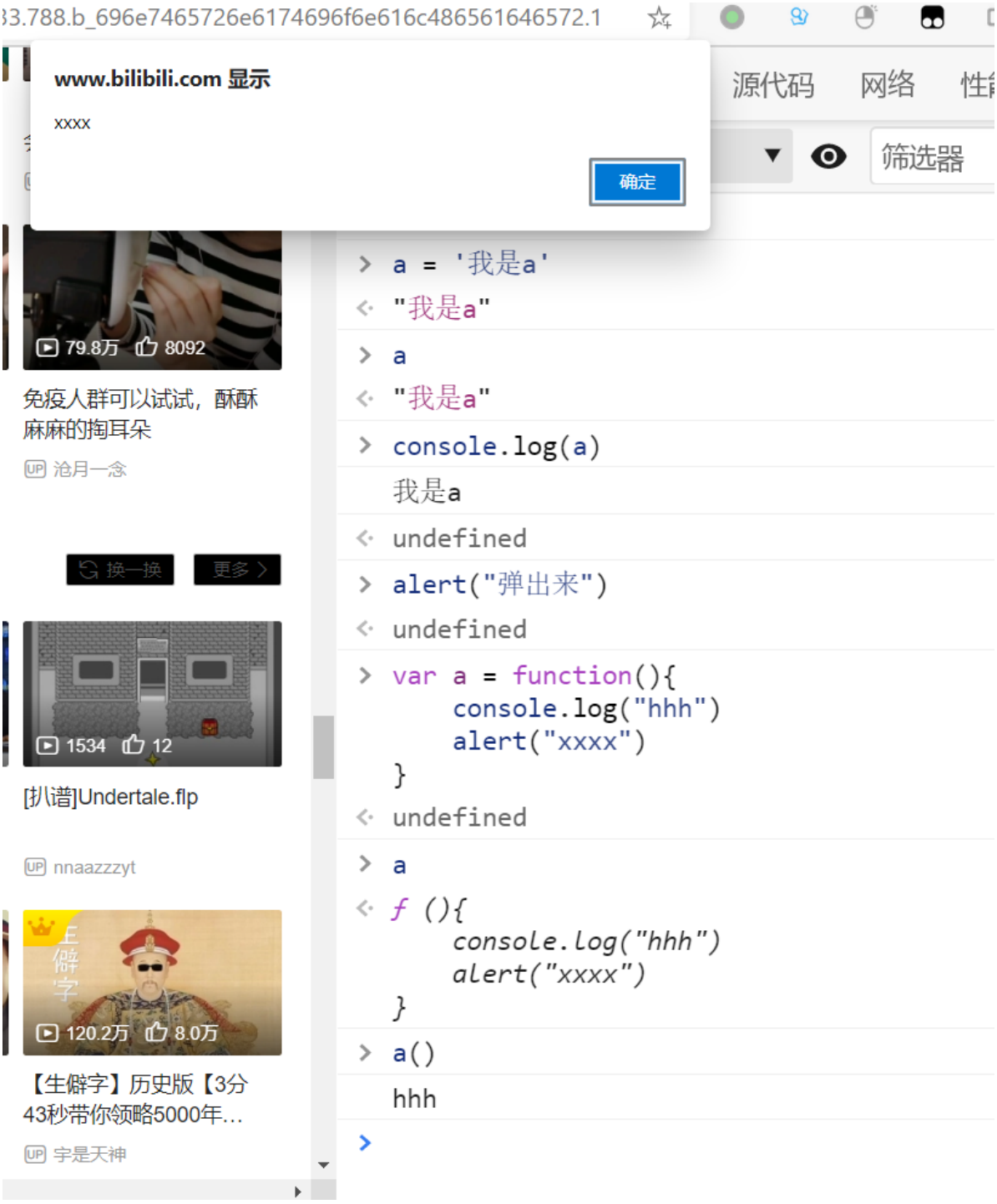
```
alert("弹出来")
```

ok

现在让我们来定义一个函数：

```
var a = function(){  //注意有左花括号，系统识别到你没写完，所以回车不会立刻像刚刚一样执行
    console.log("hhh")
    alert("xxxx")
}
```

现在让我们来使用下这个函数



注意这里，如果直接按 a 回车，a就是一个变量，表示一个函数，所以系统把他描述出来给你看，只有加了两个小括号，才表示要真正的运行这个函数

让我们来给函数加个输入 就像 alert() 和 log() 一样

```
var a = function(b){ //注意这个b名字随便取，就像数学里的x
    console.log(b)
    alert(b)
}
```

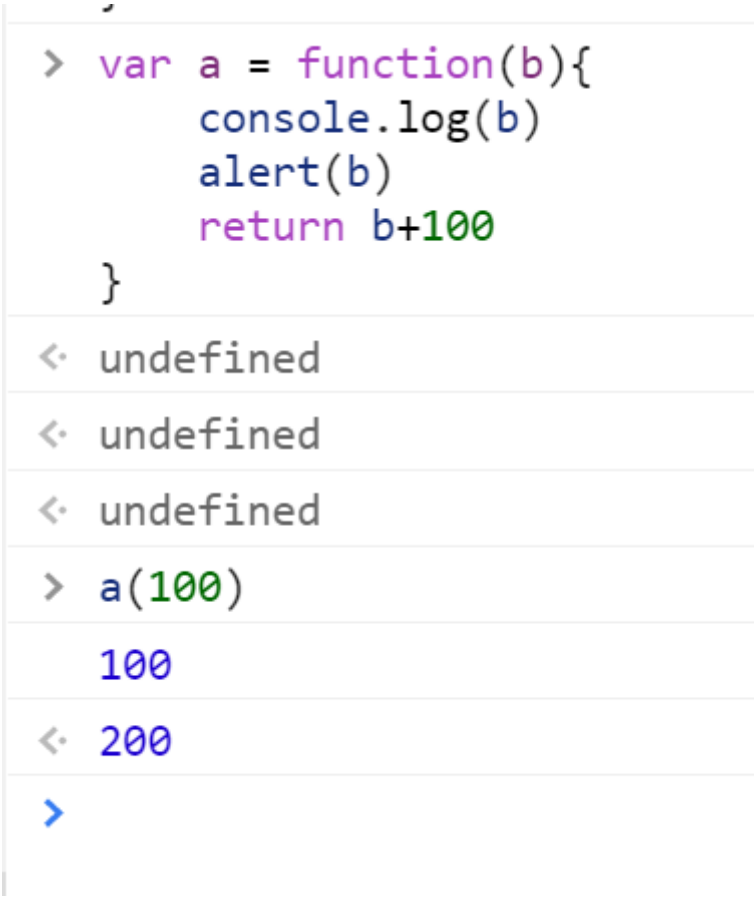
调用：

```
a('hhh')
或者
var b = 'xxx'
a(b)
```

现在让我们使我们函数像数学函数一样还可以输出（编程中通常称为有返回值）

```
var a = function(b){
    console.log(b)
    alert(b)
    return b+100
}
```

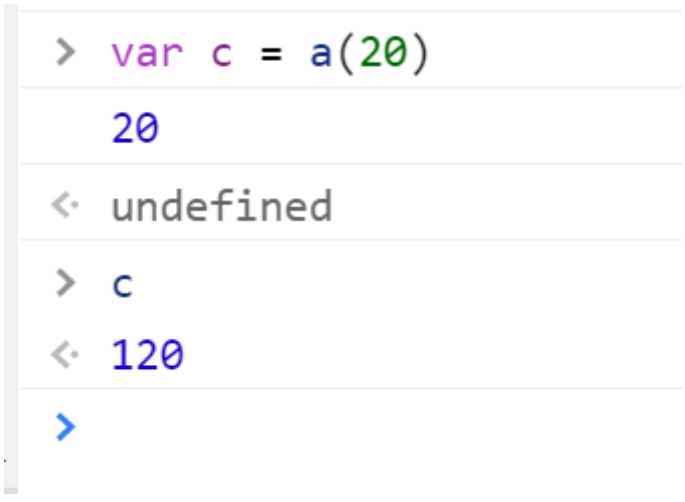
调用：



可以看到 100 是 log()出来的(没有箭头)

而有箭头的200正是这个函数返回的结果

我们可以用变量把这个值接住：



这时候 c就是120了

函数就在这告一段落啦，我们看一段简单的配套完成任务的实例：

```
var a = function(){
   console.log('第一个小任务，执行完返回10这个数据')
    return 10
}

var b = function(x) {
    alert(x)
}

var c = function() {
    b(a())
}

c() //执行总的函数
```

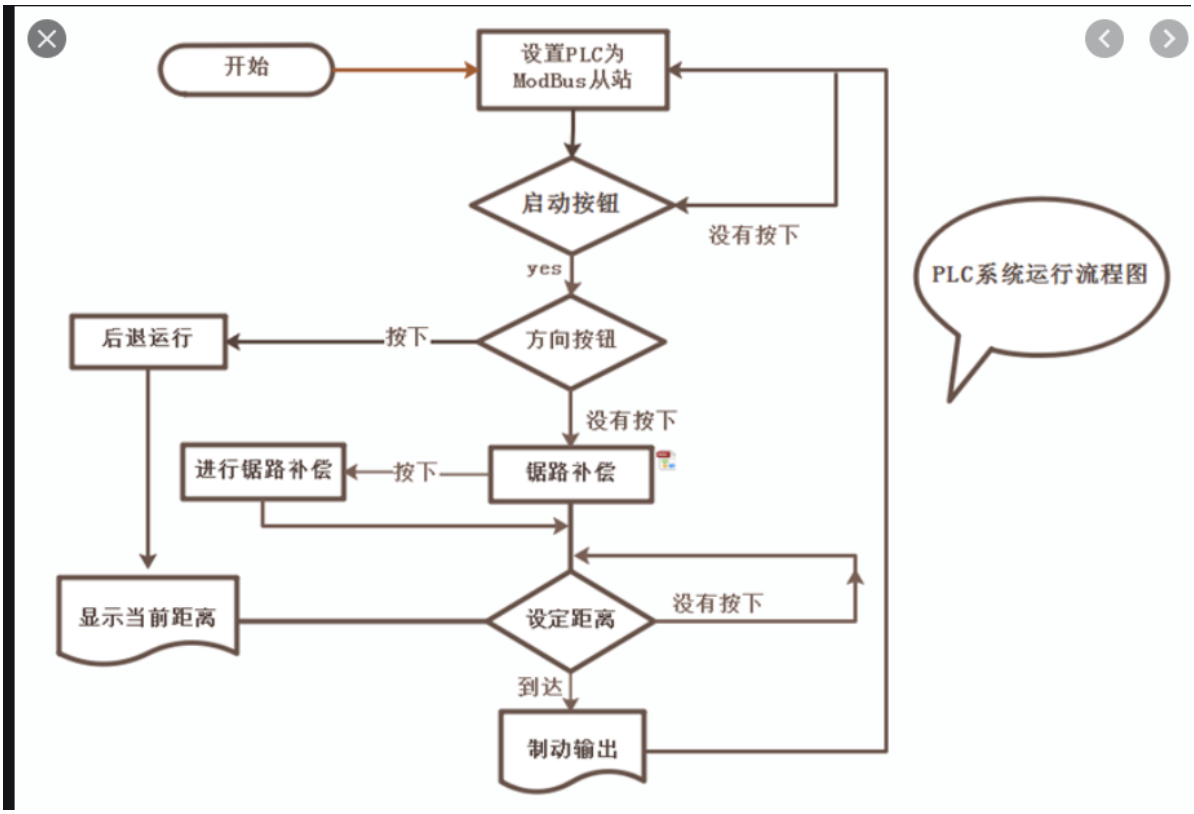
插播

输入 clear() 回车可以清屏

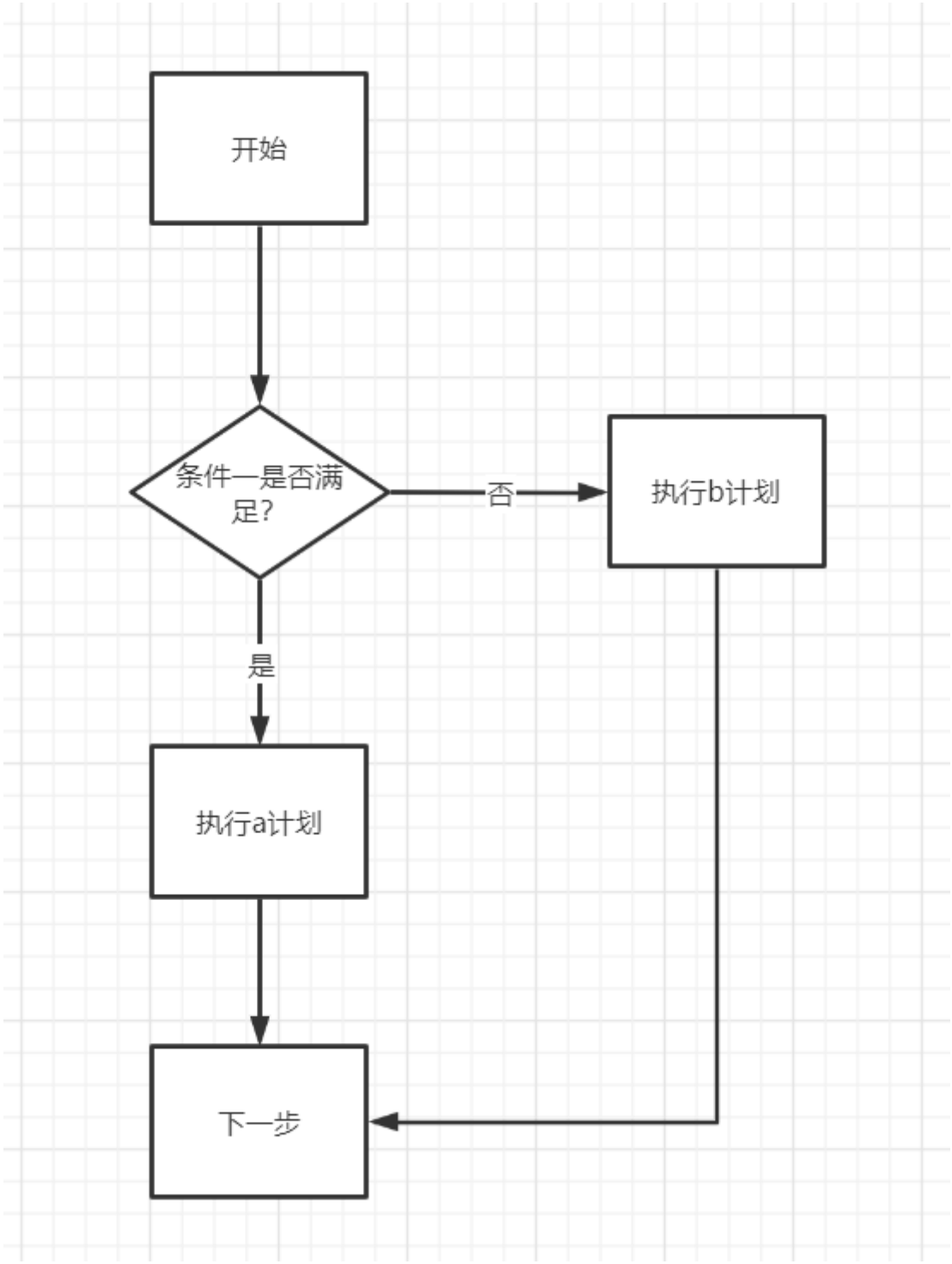
接下来是分支和循环啦，学完基本编程就出师了

首先给大家说个运算符"==" 这才是真正数学上的等于

分支



大家一定看过这样的流程图，先来分支



这就是一个简单的分支结构，相信大家流程图都能看懂吧，让我们用JavaScript来实现下：

```
var a = function(){
    var b = 3
    var c = 4 //注意！！！非常重要，这里的换行要shift+enter，就和qq一样，按回车直接发出消息了，得配合另一个按键
    if(b==c) { //理解为如果b等于c
        console.log("相等")
    } else {
        console.log("不相等")
    }
} //我们放在函数里，不放里面也行，不放就是立刻执行放了就是还要输入a()才执行
```

执行结果：

```
> var a = function(){
      var b = 3
      var c = 4
      if(b==c) {
          console.log("相等")
      } else {
          console.log("不相等")
      }
  }

<- undefined
> a()
  不相等
<- undefined
>
```

由于代码的执行必须以函数的形式，

所以我们的几句代码都写在了a函数里，调用就可以看到结果

接下来我们 将 b 和 c都设为 1 试试（不想重复输入上一段代码?按下 “↑” 试试）

```
var a = function(){
    var b = 1
    var c = 1
    if(b==c) {
        console.log("相等")
    } else {
        console.log("不相等")
    }
}
```

也可以 省略 else

```
var a = function(){
    var b = 1
    var c = 2
    if(b==c) {
        console.log("相等")
    }
}
```

也可以 多个 else

```
var a = function(){
    var b = 3
    if(b==1) {
        console.log("b是1")
    } else if (b==2) {
        console.log("b是2")
    } else {
        console.log("b是其他数字")
    }
}
```

循环

编程初期 我们只学 while循环 ， 其他（在你们学习c的过程中）还有 for 或者 do while 然而这都没有什么卵用 后两者都可以转化为while

接下来我们*新开一个浏览器标签*，运行下下面的代码：

```
while(1==1) {
    alert("gg")
}
```

很显然1肯定等于一 ， 这个条件是肯定被满足的，我们前面说过alert（）会弹出提示框的，我们现在看到已经弹出了。我们点确定，发现还是在，结果就是不管点多少下确认，这个框依旧在，这就是循环，不断反复做同样的事

好了，让我们关掉这个标签吧

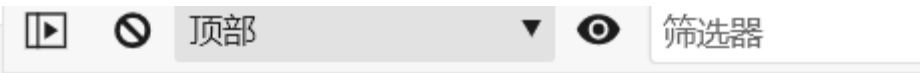
可以看到，如果程序进入不断的循环，就不会进行下一步了，只有我么把程序关了（这里是关闭标签）才能停止循环，这种循环被称为死循环，ok 下面再新开一个标签我们对死循环多做点事

在此之前先看一个问题：

```
var a = 10
a = a + 1
```

请问这时候a是多少呢? 如果知道答案就请下一步吧

```
var a = 0
while(1==1){
    console.log(a)
    a = a + 1
}
```



10676
10677
10678
10679
10680
10681
10682
10683
10684
10685
10686
10687
10688
10689
10690
10691
10692
10693
10694
10695
10696
10697
10698
10699
10700
10701
10702
10703
10704
10705
10706
10707
10708
10709
10710
10711
10712

```
10713
10714
10715
10716
10717
```

可以看到瞬间打印了一堆数，还好 javascript可能对最大循环次数限制了一下，要是其他语言估计就不停的数个数一样把数字从小到大打印下去，停都停不下来。这就是循环的威力。。。。。好了让我们关闭这个标签吧!

while循环的本质就是判断下括号里的条件是否成立，如果成立执行一下{}里的代码，执行完后又判断，又执行，循环往复，直到这个条件不成立

ok 现在我们想只让他打印到 10 怎么办，显然不能再在while（）后面 的括号里写 1==1了，这个条件一直成立

那条件写啥呢。。。。既然不让它打印10以上的数字了。。。那我么条件就设立为 a<11 吧

```
var a = 0
while(a < 11){
    console.log(a)
    a = a + 1
}
```

```
> var a = 0
  while(a < 11){
      console.log(a)
      a = a + 1
  }
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
  10
< 11
>
```

ok，，最后一个11 有个向左箭头 其实是表示 a现在的值 不是我们console.log出来的

这样我们就达到目的了

学到这绝对该休息下了，，好好回顾下上面的，配合这几个，其实已经可以做一些基本的算法题，实现一些基本的小程序了（当然这个程序是没有界面的，只有输出的文字数字）

案例一

现在让我们来打印一个99乘法表的结果部分，，，但是为了降低难度，这个乘法表是 。九x九的

```
var a = function(){
    var b = 1
    while(b < 10){
        var c = 1
        while(c < 10){
            console.log(b*c)
            c= c + 1
        }
        b = b+1
    }
}
```

而且不太好看呼呼呼，因为console.log 是肯定要换行的

猜数游戏

```
> a = function(x){
      if(x < 10) {
          console.log("猜小了")
      } else if(x > 10) {
          console.log("猜大了")
      } else {
          console.log("猜对了就是10")
      }
  }
< ƒ (x){
      if(x < 10) {
          console.log("猜小了")
      } else if(x > 10) {
          console.log("猜大了")
      } else {
          console.log("猜对了就是10")
      }
  }
> a(2)
  猜小了
< undefined
> a(10)
  猜对了就是10
< undefined
>
```