# 一、昨日反馈



# 二、今日目标

1、能够理解JS代码段执行过程是怎样的

2、能够理解JS错误对代码执行的影响

3、能够使用JS定时器功能

4、能够取消定时器

5、能够理解闭包是怎么回事

6、能够了解JS的垃圾回收机制

# 三、执行流程

3.1、全局环境

js在运行时，首先会创建一个全局执行环境，这个全局环境就是定义一个全局对象，页面中所有的内容（不同的script中的内容） 都是这个全局对象的成员，这个全局对象是window。

window对象比较大。包含所有的浏览器对象。

**怎么来证明页面中所有的内容都是window对象的成员呢？**



有了这个知识点，前面学习的很多内容都可以解释了：

比如window.addEventListener，说明addEventListener在运行的时候也是全局对象window对象的成员。

3.2、执行流程

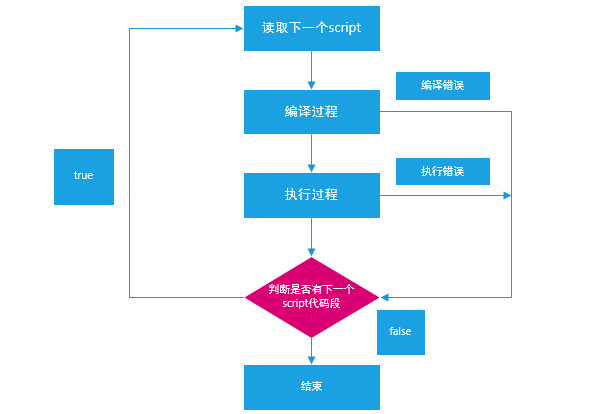
js在执行的时候，会按照script标签来一个一个的执行，也就是先执行第一个script标签中的内容，然后在执行第二个script标签的内容。

一个script标签中，首先会先

**编译代码**（检查语法、词法是否错误，没有错误就加载到内存中，准备执行）

**执行代码**（运行或输出结果）。

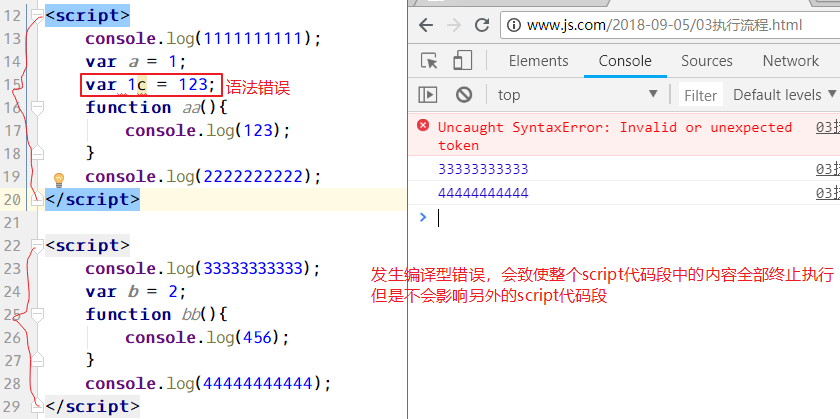
执行完毕，继续按照相同的方式执行下一个script标签的内容。



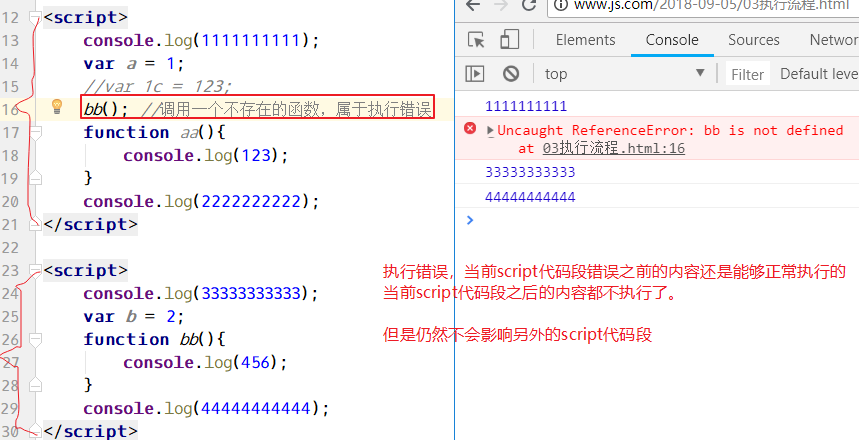
3.3、错误类型

这里的**错误类型**指的是编译型错误和执行过程中的错误。

**编译型错误，对程序执行的影响**：



**执行过程中的错误，对程序的影响**：



# 四、定时器

4.1、介绍和语法

定时器可以完成定时作用，可以让一段代码延迟一段时间然后再执行。

很多地方都会用到定时器，比如用JS写一个会走的钟表；比如轮播图；

JS定时器有两种：

* 指定一段时间，然后在该时间段之后只执行一次代码，特点是只执行一次。
* 指定一段时间，然后每隔一段时间执行一次代码，也就是间隔一段时间执行一次，一直重复。

具体方法如下：

setInterval() 按照指定的周期（以毫秒计）来调用函数或计算表达式。

setTimeout() 在指定的毫秒数后调用函数或计算表达式。

**语法**：

setTimeout(JS代码或函数, 毫秒数);

setInterval (JS代码或函数, 毫秒数);



**setInterval() 语法上和setTimeout一样**。

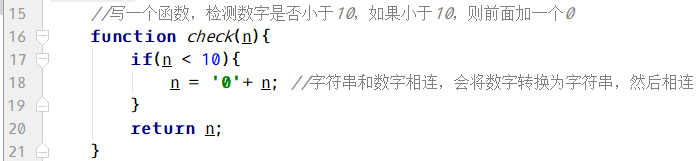
4.2、会走的时间

思路是写一个能够输出当前时间的函数，页面加载完毕马上执行一次，然后在用定时器每隔1秒执行一次函数即可。

HTML代码：



下面是检测数字是否小于10的函数：



下面定义制作时间的函数，然后马上调用一次，这样刷新页面就能够看到时间了：



使用定时器，让制作时间的函数每隔一秒执行一次：



4.3、终止定时器

对应的终止定时器的方法分别是：

* clearInterval(定时器计数);
* clearTimeout(定时器计数);

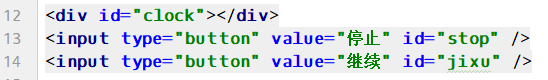
参数“定时器计数”指的是定时器的返回值。



4.4、修改会走的时间

加入两个按钮，一个表示停止，一个表示继续。

HTML加入两个按钮：



JS代码给“停止”和“继续”绑定单击事件：

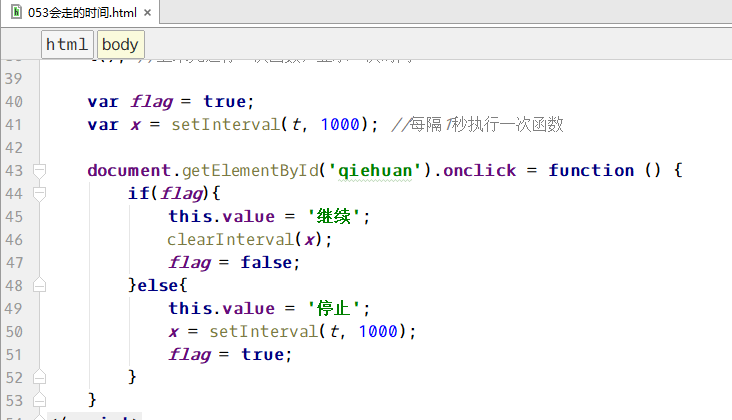


优化上述代码：

首先将按钮换成一个：



JS代码：



# 五、闭包

5.1、什么是闭包

JS有两链一包的说法，两链是作用域链和原型链，一包就是闭包。



观察上述代码，调用a函数之后，会在内存中形参红色的区域；另外将a的返回值赋值给了bb，后续垃圾回收机制并不知道bb是否有用，也就是红色内存中的b函数是否有用。所以垃圾回收机制并不会回收红色的区域。

bb ==== b

不知道bb有没有用，也就是不知道b是否有用。所以垃圾回收机制不会清除内存。

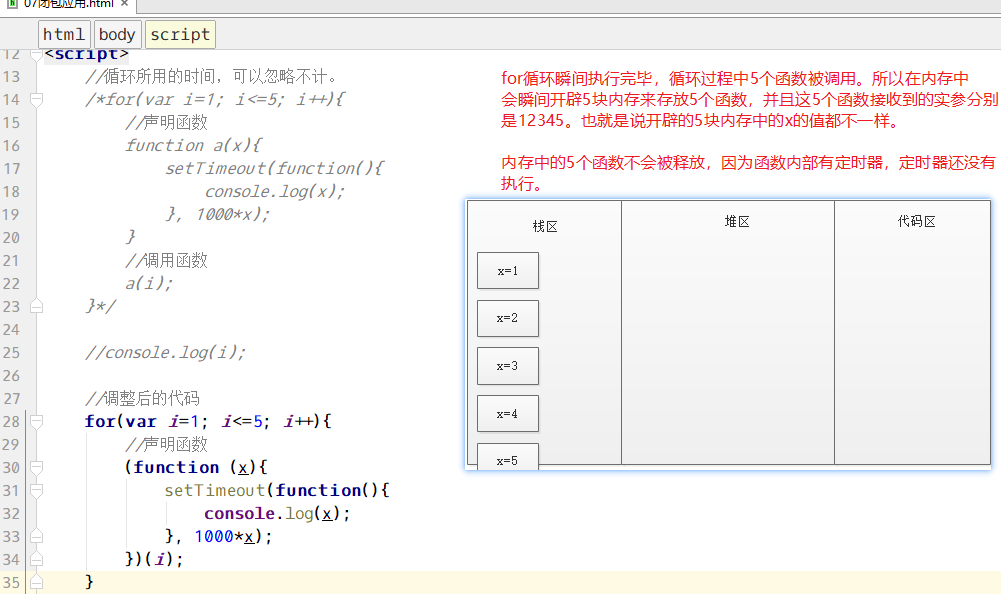
**这种情形就是闭包，有些时候专门把内部函数b叫做闭包函数**。

**闭包的特点**：

* **闭包会造成内存不被释放**；
* 闭包使得函数外部可以访问内部的变量。

5.2、应用

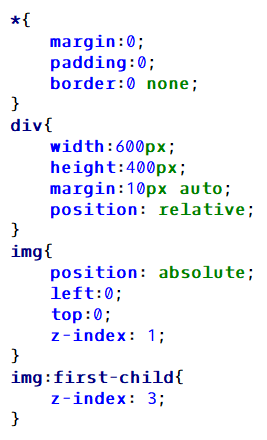
每隔1秒，依次输出12345；



# 六、案例

1、默认的HTML和css





2、JS完成图片生硬的变化

思路是用一个定时器不断的去切换图片的显示和隐藏。每切换一次，都让图片的下标增大1，但是不能超出范围。



3、完成透明度的慢慢变化

