setTimeout究竟做了什么

在《我对event-loop的理解》这篇文章中，有两个地方说的不太合适。

引用： 首先主线程会先去实现执行栈里的程序，当执行栈里的程序都执行完以后，再去事件队列里查询可执行的事件

当这些都执行完以后，主线程会去检测事件队列里的 onreadystatechange 和 onclick 是否可执行

这里应该是event loop线程一直监听事件队列里的事件，不管执行栈里的程序是否执行完，当事件队列里的事件有触发的时候，该事件的回调函数会被调入执行栈中，放到执行栈的末尾等待被执行。而不是等到执行栈里的程序都执行完以后再去监听事件队列，而是在整个程序执行过程中一直在监听，直到事件队列为空。

执行栈

Write()

事件队列

send()

执行

Event loop一直监听

onclick

onreadystatechange

open()

XMLHttpRequest()

执行栈

Write()

事件队列

send()

执行

Event loop一直监听

onclick

onreadystatechange

open()

XMLHttpRequest()

Onreadystatechange有返回结果被调入执行栈

onreadystatechange

那下面我们来看一下setTimeout的原理。首先我们看一下setTimeout是这样调用的 setTimeout(fn,t)；其中fn是回调函数，t是时间。所以说我们可以将setTimeout看做一个事件。其第一个参数是这个事件绑定的回调函数；第二个参数t可以看做触发该事件的诱因，也就是说经过t时间，触发该事件，将fn调入执行栈末尾待执行。

**第二个参数，时间t**

对于这个时间t有几个地方需要注意，首先是当t小于4毫秒的时候将按照4毫秒来计算，也就是setTimeout(fn,300)；也将fn推迟4毫秒在调入执行栈。然后当t为0的时候setTimeout(fn,0)；在执行栈里的程序执行完就立即执行fn，也就是说一开始就将fn调入了执行栈的末尾，中间没有延迟。

下面看两个例子

console.log("NO.1");

setTimeout(**function**(A){console.log("NO.3");},10000);

**for**(**var** i=0;i<10000;i++){ //循环C

console.log("a");

}

console.log("NO.2");

document.getElementById('button').onclick=**function**(){ setTimeout(**function**(B){console.log("NO.4")},5000);

console.log("NO.5");

}

这个例子的执行结果有多种，但是根据setTimeout(fn,t)的原理我们可以事先根据我们的操作推导出结果。我们先来分析一下这个例子，第一个setTimeout 延迟10秒调用A，第二个setTimeout在点击事件触发以后经过5秒调用B。此程序的事件队列中有两个事件一个是function(A)，还有一个是onclick。

执行栈

事件队列

Console NO.1

执行

Event loop一直监听

onclick

Settimeout(A)

循环C

Console NO.2

程序开始运行首先输出NO.1然后会循环输出10000次a，如果循环C运行的时间超过了10秒，那在循环C运行过程中function(A)就已经被调入了执行栈中，那循环C运行完以后我们将会看到NO.2和NO.3几乎是同时输出的，不过NO.2是先被执行的，因为function(A)是在执行栈的末尾。如果循环C运行时间小于10秒假如说只用了6秒，那循环C运行完以后会输出NO.2然后经过4秒再输出NO.3。

下面是onclick事件，我们之说一种情况。当在程序运行8秒的时候我们触发了onclick事件，此时事件队列里没了onclick而换成了function(B)，执行栈里加入了NO.5。

执行栈

第八秒

事件队列

循环C

执行

Event loop一直监听

Settimeout(B)

Settimeout(A)

Console NO.2

Console NO.5

第二个setTimeouts事件开始延迟5秒以后被调入执行栈，等到第10秒的时候，settimeout(A)被调入执行栈。

执行栈

第十秒

事件队列

循环C(执行中)

执行

Event loop一直监听

Settimeout(B)

Console NO.2

Console NO.5

Console NO.3

此时距离settimeout(B)还有3秒触发，假设循环C在11秒的时候执行完成，紧接着会向下执行，当前输出结果为 NO.1,（10000）个a,NO.2,NO.5,NO.3。然后再经过2秒，settimeout(B)被调入执行栈中执行。所以在第8秒点击的时候（当然这个时间没有那么准确，我们可以估计个大概）最终输出结果为NO.1,（10000）个a,NO.2,NO.5,NO.3,NO.4。（这里可能说的有些啰嗦了，不知道大家能不能理解）。

总之一句话，settimeout就是和onclick一样看做一个事件，只是其触发条件是设置的时间过了以后就将事件从队列中调入执行栈中执行。

举这个例子主要是为了说明settimeout的运行原理，结合event loop的机制我觉得我们能更进一步的理解javascript的底层运行机制。这些都是我个人的理解，水平有限不能进入更深层次的理解，大家如果有什么好的建议还请告知，大家共同进步。