链滴

Concurrency model and Event Loop

作者：amlurs

原文链接：https://ld246.com/article/1437220490910

来源网站：链滴

<h2 id="Event_loop">Event loop</h2>
The <code>event loop</code> got its name because of how it's usually implemented, w
ich usually resembles:
<pre class="prettyprint">while(queue.waitForMessage()){
  queue.processNextMessage();
}</pre>
<pre class="brush: js language-js" data-number=""><code class=" language-js"><span clas
="token keyword"> </code></pre>
<code>queue.waitForMessage</code> waits synchronously for a message to arrive if th
re is none currently.
<h3 id="Run-to-completion">"Run-to-completion"</h3>
Each message is processed completely before any other message is processed. This offers
some nice properties when reasoning about your program, including the fact that whenever a
function runs, it cannot be pre-empted and will run entirely before any other code runs (and
an modify data the function manipulates). This differs from C, for instance, where if a function
runs in a thread, it can be stopped at any point to run some other code in another thread.</
>
函数运行时,不能被外界修改,且不可拆分运行
A downside of this model is that if a message takes too long to complete, the web applic
tion is unable to process user interactions like click or scroll. The browser mitigates this with t
e "a script is taking too long to run" dialog. A good practice to follow is to make message pro
essing short and if possible cut down one message into several messages.
同时如果函数执行时间过长, 将无法做其他响应, 也就是浏览器
卡住
所以作为消息处理函数应该尽量的简洁
<h3 id="Adding_messages">Adding messages</h3>
In web browsers, messages are added any time an event occurs and there is an event list
ner attached to it. If there is no listener, the event is lost. So a click on an element with a click
vent handler will add a message--likewise with any other event.
Calling <code><a href="https://developer.mozilla.org/en-US/docs/Web/API/WindowTi
ers.setTimeout" title="/en-US/docs/window.setTimeout">setTimeout</code> will add a
message to the queue after the time passed as second argument. If there is no other message
in the queue, the message is processed right away; however, if there are messages, the <cod
>setTimeout</code> message will have to wait for other messages to be processed. For that
eason the second argument indicates a minimum time and not a guaranteed time.
<code><a href="https://developer.mozilla.org/
n-US/docs/Web/API/WindowTimers.setTimeout" title="/en-US/docs/window.setTimeout"><
pan style="color: #ff0000;">setTimeout</code> 即使到了预定的时间,但有其他消
处理函数执行中,会一直等待, 所以说这个时间参数只是最小值, 不是精确值
<h3 id="Several_Runtime_communicating_together">Several Runtime communicating toget
er</h3>
A web worker or a cross-origin iframe has its own stack, heap, and message queue. Two d
stinct runtimes can only communicate through sending messages via the <a href="https://de
eloper.mozilla.org/en-US/docs/DOM/window.postMessage" title="/en-US/docs/DOM/windo
.postMessage"><code>postMessage</code> method. This method adds a message to
he other runtime if the latter listens to <code>message</code> events.
<h2 id="Never_blocking">Never blocking</h2>
A very interesting property of the event loop model is that JavaScript, unlike a lot of othe
 languages, never blocks. Handling I/O is typically performed via events and callbacks, so whe
 the application is waiting for an <a href="https://developer.mozilla.org/en-US/docs/Web/AP
/IndexedDB_API" title="/en-US/docs/IndexedDB">IndexedDB query to return or an <a h
ef="https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest" title="/en-US/doc
/DOM/XMLHttpRequest">XHR request to return, it can still process other things like use
 input.

Legacy exceptions exist like <code>alert</code> or synchronous XHR, but it is considered as a good practice to avoid them. Beware, exceptions to the exception do exist (but are usualy implementation bugs rather than anything else).
当然对于I/O来说并不会阻塞, 他们将会异步执行, 除非你用了个alert()