1. 并发（153页）

61：不要阻塞I/O事件队列。

在web中，可以使用web worker进行多线程操作，ie10及以上有效。

H5api workre [dome](worker) (需要用node开服务器)

web worker看起来很美好，但处处是魔鬼。

我们可以做什么：

1.可以加载一个JS进行大量的复杂计算而不挂起主进程，并通过postMessage，onmessage进行通信

2.可以在worker中通过importScripts(url)加载另外的脚本文件

3.可以使用 setTimeout(), clearTimeout(), setInterval(), and clearInterval()

4.可以使用XMLHttpRequest来发送请求

5.可以访问navigator的部分属性

有那些局限性：

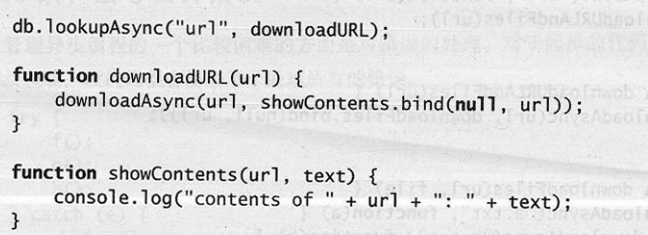
1.不能跨域加载JS

2.worker内代码不能访问DOM

3.各个浏览器对Worker的实现不大一致，例如FF里允许worker中创建新的worker,而Chrome中就不行

1. 不是每个浏览器都支持这个新特性

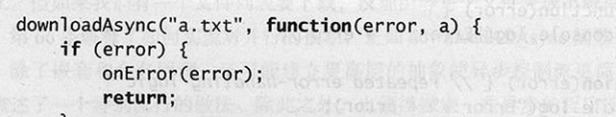
62：在异步序列中使用嵌套或命名的回调函数。

异步js中多层回调可以用bind代替。

以免看起来很多个回调函数嵌套。

避免可并行的操作顺序化。

63：当心丢弃错误。



Node捕获错误流行形式。

64：对异步循环使用递归。

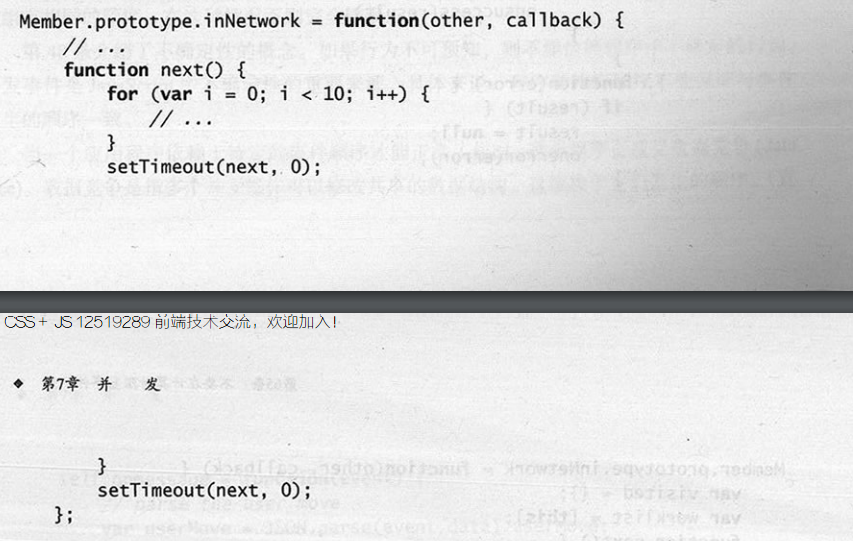
异步回调，调用自身。

65：不要在计算时堵塞事件队列。

将大量计算放在web worker中执行。

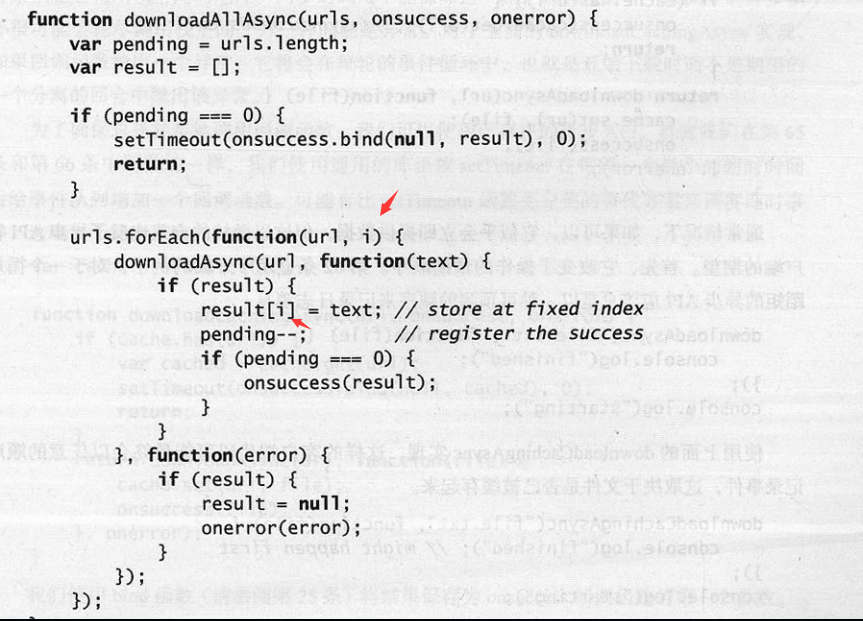
需要循环的语句，可以用setTImeout来递归调用，这样不会想while一样堵塞进程。

一次循环代码量够少的话，可以一次setTImeout执行多次。



66：使用计数器来执行并行操作。

同时下载多个文件（一个异步js）时，完成下载的顺序不可控，因此要为每个文件提供序号。



这里用i保存下载好的文件而不是push操作。

67：绝对不要同步地调用异步回调函数。

同步执行回调函数和异步执行回调函数的区别在于，同步执行回调函数，在代码运行到这里就利马执行（可能在加载文档时），异步执行等函数返回结果时回调（所有文档都加载完了之后，即初始化界面的js执行完之后）。<effective67.html>

同步得调用异步回调，可能会扰乱预期的操作顺序。

68:使用promise模式清洁异步逻辑。

使用promise方便组合不同的并行操作。

使用promise模式避免数据竞争（同66）。

在要求有意竞争条件时，使用select（有一个完成回调，其他回调不执行）