-什么是Web Workers？为什么我们需要他们？  
考虑以下会执行上百万次的繁重的循环代码

function  SomeHeavyFunction(){

for (i = 0; i < 10000000000000; i++){

x = i + x;

}

}

比方说上面的循环代码在HTML按钮点击以后执行，现在这个方法执行是同步的，换句话说这个浏览器必须等到循环完成才能操作，

如果你能移动这些繁重的循环到Javascript文件中，采用异步的方式运行，这意味着浏览器不需要等到循环结束，从而使浏览器有更友好的交互效果，这就是web worker的作用.

Web Worker线程的限制是什么？  
Web worker线程不能修改HTML元素，全局变量和Window.Location一类的窗口属性。你可以自由使用Javascript数据类型，XMLHttpRequest调用等。

我们如何在JavaScript中创建一个worker线程？  
创建一个worker线程，我们需要通过Javascript文件名创建worker对象

var worker = new Worker("MyHeavyProcess.js");

worker.postMessage();

worker.onmessage = function (e) {

document.getElementById("txt1").value = e.data;

};

MyHeavyProcess.js

var x =0

self.onmessage = function (e) {

  for (i = 0; i < 1000000000; i++){

    x = i + x;

  }

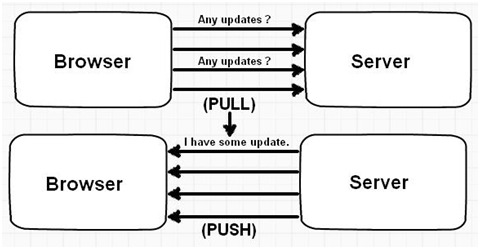
  self.postMessage(x);

};

如何中止Web Worker?

w.terminate();

为什么我们需要HTML5的服务发送事件？  
网络世界的普遍需求是从服务器更新。以一个股票应用为例，浏览器必须定期从服务器更新最新的股票值。



现在实现这类需求开发者通常写一些PULL的代码，到服务器同时抓取某些区间数据。现在PULL的解决方案是很好的，但是这使得网络健谈有很多的调用，同时增加了服务器的负担。

因此相比于PULL，如果我们能采用某种PUSH的解决方案那会是很棒的。简而言之，当服务器更新的时候，将会发送更新到浏览器客户端，那可以被接受通过使用”SERVER SENT EVENT”

HTML5中的本地存储概念是什么？  
很多时候我们会存储用户本地信息到电脑上，例如：比方说用户有一个填充了一半的长表格，然后突然网络连接断开了，这样用户希望你能存储这些信息到本地，当网络恢复的时候，他想获取这些信息然后发送到服务器进行存储  
现代浏览器拥有的存储被叫做“Local Storage”,你可以存储这些信息。

我们如何从本地存储中添加和移除数据？  
数据添加到本地存储采用键值对，以下示例显示了城市数据”India”添加了键”Key001”

localStorage.setItem(“Key001”,”India”);

从本地存储中检索数据我们可以提供键名并使用”getItem”方法

var country = localStorage.getItem(“Key001”);

你也可以使用以下代码，存储Javascript对象在本地存储中

var country = {};

country.name = “India”;

country.code = “I001”;

localStorage.setItem(“I001”, country);

var country1 = localStorage.getItem(“I001”);

如果你想存储Json格式，你可以使用“JSON.stringify”方法，如下所示：

localStorage.setItem(“I001”,JSON.stringify(country))

本地存储的生命周期是什么？

本地存储没有生命周期，除非用户清除浏览器缓存或者使用Javascript代码移除。

本地存储和cookies（储存在用户本地终端上的数据）之间的区别是什么？

|  |  |  |
| --- | --- | --- |
|  | Cookies | Local storage |
| 客户端/服务端 | 客户端和服务端都能访问数据。Cookie的数据通过每一个请求发送到服务端 | 只有本地浏览器端可访问数据，服务器不能访问本地存储直到故意通过POST或者GET的通道发送到服务器 |
| 大小 | 每个cookie有4095byte | 每个域5MB |
| 过期 | Cookies有有效期，所以在过期之后cookie和cookie数据会被删除 | 没有过期数据，无论最后用户从浏览器删除或者使用Javascript程序删除，我们都需要删除 |

什么是事务存储？我们如何创建一个事务存储？  
会话存储和本地存储类似，但是数据在会话中有效，简而言之数据在你关闭浏览器的时候就被删除了。  
为了创建一个会话存储你需要使用“sessionStorage.variablename.”在以下的代码我们创建了一个名为”clickcount”的变量；  
如果你刷新浏览器则数目增加，但是如果你关闭浏览器，“clickcount”变量又会从0开始。

本地存储和事务存储之间的区别是什么？  
本地存储数据持续永久，但是会话在浏览器打开时有效知道浏览器关闭时会话变量重置

什么是WebSQL？  
WebSQL是一个在浏览器客户端的结构关系数据库，这是浏览器内的本地RDBMS(关系型数据库系统)，你可以使用SQL查询

WebSql是HTML5的一个规范吗？  
不是，许多人把它标记为HTML5，但是他不是HTML5的规范的一部分，这个规范是基于SQLite的

我们如何使用WebSQL？  
第一步我们需要做的是使用如下所示的“OpenDatabase”方法打开数据库，第一个参数是数据库的名字，接下来是版本，然后是简单原文标题，最后是数据库大小；

var db=openDatabase('dbCustomer','1.0','Customer app’, 2 \* 1024 \* 1024);

为了执行SQL，我们需要使用“transaction”方法，并调用”executeSql”方法来使用SQL

db.transaction(function (tx) {

tx.executeSql('CREATE TABLE IF NOT EXISTS tblCust(id unique, customername)');

tx.executeSql('INSERT INTO tblcust (id, customername) VALUES(1, "shiv")');

tx.executeSql('INSERT INTO tblcust (id, customername) VALUES (2, "raju")');

}

万一你要使用“select”查询你会得到数据”result”集合，我们可以通过循环展示到HTML的用户界面

db.transaction(function (tx) {

   tx.executeSql('SELECT \* FROM tblcust', [], function (tx, results) {

   for (i = 0; i < len; i++){

     msg = "<p><b>" + results.rows.item(i).log + "</b></p>";

     document.querySelector('#customer).innerHTML +=  msg;

}

}, null);

});

HTML5中的应用缓存是什么？  
一个最需要的事最终是用户的离线浏览，换句话说，如果网络连接不可用时，页面应该来自浏览器缓存，离线应用缓存可以帮助你达到这个目的  
应用缓存可以帮助你指定哪些文件需要缓存，哪些不需要。

HTML5中我们如何实现应用缓存？  
首先我们需要指定”manifest”文件，“manifest”文件帮助你定义你的缓存如何工作。以下是”mainfest”文件的结构

CACHE MANIFEST

# version 1.0

CACHE :

Login.aspx

* 所有manifest文件都以“CACHE MANIFEST”语句开始.
* #（散列标签）有助于提供缓存文件的版本.
* CACHE 命令指出哪些文件需要被缓存.
* Mainfest文件的内容类型应是“text/cache-manifest”.

我们如何刷新浏览器的应用缓存？  
应用缓存通过变更“#”标签后的版本版本号而被移除，如下所示：

CACHE MANIFEST

# version 2.0(new)

CACHE :

Login.aspx

Aboutus.aspx

NETWORK :

Pages.aspx

应用缓存中的回退是什么？  
应用缓存中的回退帮助你指定在服务器不可访问的时候，将会显示某文件。例如在下面的manifest文件中，我们说如果谁敲击了”/home”同时服务器不可到达的时候，”homeoffline.html”文件应送达

应用缓存中的网络是什么？  
网络命令描述不需要缓存的文件，例如以下代码中，我们说”home.aspx”永远都不应该被缓存或者离线访问。

NETWORK:

home.aspx