数据持久

数据持久指将内存中的数据模型转化为存储模型，和将存储模型转化为内存中的数据模型这一过程的统称。在普通情况下，我们存储的数据会一直保留，直到我们删除相关内容；或者是这些数据保存到浏览器会话结束，用户关闭之后。  
但在实际情况中会更加复杂一些。用户、操作系统、浏览器或插件都可以随时阻止或删除持久数据。浏览器有权限删除存储内容比较陈旧或者是比较大的项目内容；还能记录页面状态，当我们离开当前页面，重新打开页面的时候上次记录的内容会得到保存记录，可以直接使用。

使用场景

当数据并不需要发送到web服务器或者并不需要数据内容时，只需要在浏览器中存储和操作数据（也叫客户端）会用到数据持久，而需要在浏览器中存储和操作数据具体包括以下几种情况：

* 保留客户端应用程序的状态—例如当前屏幕、输入的数据、用户首选项等
* 访问本地数据或文件并有严格隐私要求的实用程序
* 脱机工作的渐进式web应用程序（PWA）  
  接下来将为大家详细比较10中不同的客户端存储方式，包括这些方法的限制、优缺点以及每种方式的使用，方便大家选择根据自己的使用场景进行选择。
* JavaScript变量
* DOM节点（DOM node）存储
* Web存储（localStorage和sessionStorage）
* IndexedDB/索引数据-
* Cache API（不使用AppCache）
* 文件系统访问API
* 文件和目录项API
* cookies
* window.name
* WebSQL

总体比较



文本将从容量、读写速度、数据持久三个角度进行比较这十种方式，接下来为大家介绍详细内容。

1. JavaScript变量  
   将状态存储在JavaScript变量中是最快、最简单的，例子如下：



**优势**

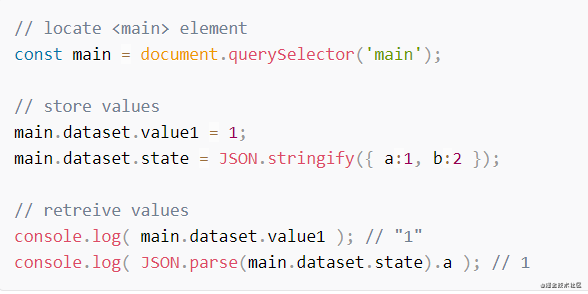
* 易于使用
* 快捷
* 不需要序列化或反序列化  
  **缺点**
* 易失：刷新或者关闭标签会清除所有内容
* 第三方脚本可以检查或覆盖全局（窗口）值  
  如果你已经在使用JS变量，可以考虑在page unloads时永久存储变量状态

2.DOM节点（DOM node）存储



大多数DOM元素，无论是在页面上还是在内存中，都可以在命名属性中存储值。使用以data-为前缀的属性名称更安全：

* 该属性不会关联HTML
* 可以通过数据集属性而不是较长的.setAttribute和.getAttribute方法访问值存储为字符串，因此可能需要序列化和反序列化。例如：



**优势**

* 可以在JavaScript或HTML中定义值，例如<main data-value1=“1”>
* 用于存储特定组件的状态
* DOM速度过快  
  **缺点**
* 易碎：刷新或关闭当前内容会清除所有内容（除非服务器将值传递到HTML中）
* 字符串需要序列化和反序列化
* 较大的DOM会影响性能
* 第三方脚本可以检查或覆盖值  
  DOM节点存储比变量慢。在将组件的状态存储在HTML中是可行的情况下，使用的时候需要注意这一点。现在这种方式已经被逐渐淘汰，原因是DOM节点生成树的存储速度过慢，在大型项目中的效率十分低下。不过为了解决这个问题现在HTML 5的Canvas已有了详尽的解决方案， 比如[SpreadJS纯前端表格组件](https://www.grapecity.com.cn/blogs/spreadjs-gc-front-table-technology-sharing)已经引入了Canvas绘制模型和双缓存画布技术，使项目效率大大提升。

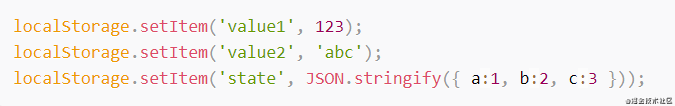
3.Web存储（localStorage和sessionStorage）



Web存储提供了两个类似的api来定义名称/值对：

* window.localStorage：存储持久数据  
  •- window.sessionStorage：在浏览器选项内容保持打开状态时仅保留会话数据

使用.setItem方法存储或更新命名项：

  
使用.getItem方法进行检索：

  
使用.removeItem方法删除：



**优势**

* 简单名称/值对API
* 有会话和持久存储选项
* 良好的浏览器支持  
  **缺点**
* 仅字符串：需要序列化和反序列化
* 无事务、索引或搜索的非结构化数据
* 同步访问将影响大型数据集的性能

Web存储非常适合于更简单、更小、特别的值。存储大量结构化信息不太实用，但是我们可以通过在页面卸载时写入数据来避免性能问题。

4.IndexedDB/索引数据库



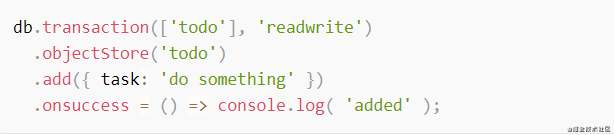
IndexedDB提供了一个类似NoSQL的低级API来存储大量数据。可以进行索引存储，使用事务更新存储，并使用异步方法搜索存储。  
IndexedDBapi很复杂，需要一些事件处理。以下函数在传递名称、版本号和可选升级函数（在版本号更改时调用）时打开数据库连接：



下面内容连接到myDB数据库并初始化todo对象存储（类似于SQL表或MongoDB集合）。然后定义一个名为id的自动递增键：



数据库连接就绪后，可以在事务中添加新数据项：



此时可以检索值



**优势**

* 具有最大空间的灵活数据存储
* 强大的事务、索引和搜索选项
* 良好的浏览器支持  
  **缺点**
* 回调复杂，API基于事件
* IndexedDB可以存储大量数据，但需要使用诸如idb、Dexie.js或JsStore之类的包装器库。

1. Cache API



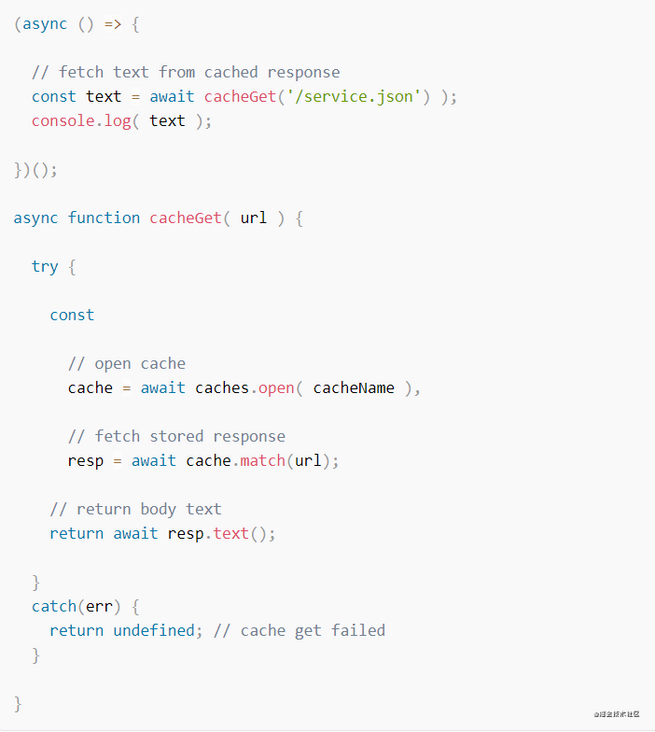
Cache API为HTTP请求和响应对象对提供存储。您可以创建任意数量的命名缓存来存储任意数量的网络数据项。

API通常对缓存渐进式web应用进行网络响应。当设备与网络断开连接时，重新提供缓存内容，以便web应用程序可以脱机运行。

以下代码将网络响应存储在名为myCache的缓存中：



类似的函数可以从缓存中检索项。下面的例子中，它返回响应正文文本：



**优势**

* 存储任何网络响应
* 可以提高web应用程序性能
* 允许web应用程序脱机运行
* 基于Promise的现代API  
  **缺点**
* 不适用于存储应用程序状态
* 在渐进式web应用程序之外不太有用
* 苹果对PWAs和Cache API并不友好

Cache API是存储从网络检索的文件和数据的最佳选择。我们可以使用它来存储应用程序状态。

1. 文件系统访问API



文件系统访问API允许浏览器从本地文件系统读取、写入、修改和删除文件。浏览器在沙盒环境中运行，因此用户必须授予对特定文件或目录的权限。这将返回一个FileSystemHandle，以便web应用程序可以像桌面应用程序一样读取或写入数据。

以下函数将Blob保存到本地文件：



**优势**

* web应用程序可以安全地读取和写入本地文件
* 不需要在服务器上上传文件或处理数据  
  **缺点**
* 只有最低限度的浏览器支持（仅限Chrome）
* API会发生更改

这种储存方式的优势几乎是压倒性的

1. 文件和目录项API



文件和目录条目API提供了一个可用于域的文件系统，该系统可以创建、写入、读取和删除目录和文件。  
**优势**

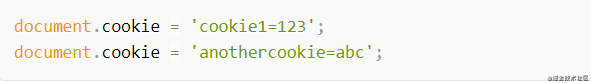
* 存在一些可探索的有趣用法  
  **缺点**
* 实现和行为之间的非标准、不兼容可能会改变

不过目前MDN明确声明：不要在生产站点上使用此选项，技术广泛的支持还需要几年。

1. cookies



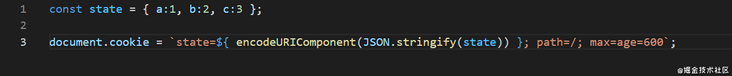
Cookie是特定于域的数据，用来跟踪用户，但对于任何需要维护服务器状态的系统（如登录）来说，它们都是必不可少的。与其他存储机制不同的是，cookies（通常）在浏览器和服务器之间的HTTP请求和响应上传递。两个设备都可以检查、修改和删除cookie数据。  
使用document.cookie在客户端中设置cookie值，使用方法：



值不能包含逗号、分号或空格，所以需要 encodeURIComponent方法：



示例：设置一个状态cookie，该cookie将在10分钟后过期，并且在当前域中的任何路径上都可用：



document.cookie返回一个字符串，其中包含由分号分隔的每个名称和值对。例如：



下面的函数解析字符串并将其转换为包含name-value的对象。例如：



**优势**

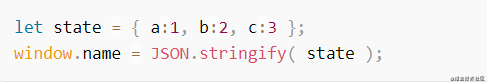
* 可在在客户机和服务器之间保留数据状态
* 仅限于域和路径（可选）
* 自动过期控制，最大过期时间（秒）或过期时间（日期）
* 默认情况下在当前会话中使用（设置过期日期，可以在页面刷新和标签关闭之后保留数据）  
  **缺点**
* 浏览器和插件会阻止Cookie（它们通常被转换为会话Cookie，这样站点就可以继续工作）
* JavaScript实现需要创建自己的cookie处理程序或选择js cookie之类的库
* 字符串需要序列化和反序列化
* 存储空间有限
* 除非限制访问，否则第三方脚本可以检查cookie
* 侵犯隐私
* 每个HTTP请求和响应都会附加cookie数据，影响性能（存储50Kb的cookie数据，然后请求10个1字节的文件，将产生1兆字节的带宽）

缺点过多，不是必要不推荐使用cookie

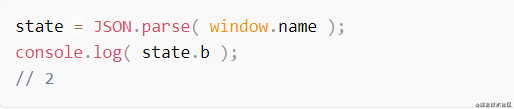
1. window.name



window.name设置并获取窗口浏览上下文的名称。我们可以设置一个字符串值，该值在浏览器刷新或链接到其他位置并单击“上一步”之间保持不变。例如：



检查该内容：



**优势**

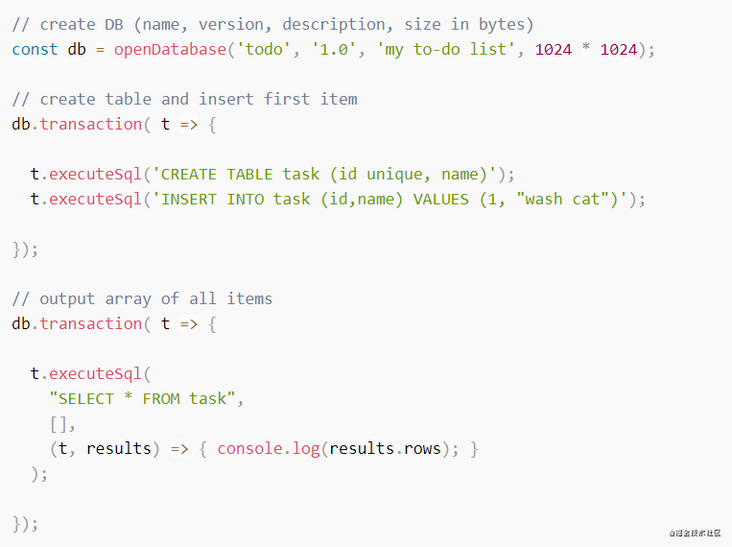
* 易于使用
* 只能用于会话数据  
  **缺点**
* 字符串需要序列化和反序列化
* 其他域中的页面可以读取、修改或对数据进行删除

window.name的设计之初并不作为数据存储的方法，可作为一个黑科技使用。

1. WebSQL



WebSQL是将SQL的数据库存储引入浏览器的方法。示例代码：



**优势**

* 更适用强健的客户端数据存储和访问
* 服务器端使用SQL语法  
  **缺点**
* 浏览器支持有限
* 跨浏览器的SQL语法不一致
* 异步回调API不够灵活
* 性能差

可以结合数据库使用，也为客户端存储提供一种方法。

总结

本文详细为大家介绍了10种不同客户端存储的解决方法，可以看到的是，没有一种方法是十全十美的。为了解决复杂web应用程序中的不同情况，我们需要学习更多API才行。根据不同情况因地制宜，灵活运用将会更加高效的解决问题。