HTML5离线功应用能详解

作者：有故事的人 来源：范文网 www.wtabcd.cn/fanwen/

本文原地址：https://www.wtabcd.cn/fanwen/zuowen/248fbc6ac855b0e17cb98b75351f40aa.html

范文网，为你加油喝彩！

简介：web2.0 技术鼓励个人的参与，每个人都是 web 内容的撰写者。如果 web 应用能够提供离线的功能，让用户在没有网络的地方（例如飞机上）和时候（网络坏了），也能进行内容撰写，等到有网络的时候，再同步到 web 上，就大大方便了用户的使用。html5 作为新一代的 html 标准，包含了对离线功能的支持。本文介绍了 html5 离线功能中的离线资源缓存、在线状态检测、本地数据存储等内重阳节怎么发朋友圈说说容，并举例说明了如何使用 html5 的新特性开发离线应用。

html5 离线功能介绍

html5 是目前正在讨论的新一代 html 标准，它代表了现在 web 领域的最新发展方向。在 html5 标准中，加入了新的多样的内容描述标签，直接支持表单验证、视频音频标签、网页元素的拖拽、离线存储和工作线程等功能。其中一个新特性就是对离线应用开发的支持。

在开发支持离线的 web 应用程序时，开发者通常需要使用以下三个方面的功能：

离线资源缓存：需要一种方式来指明应用程序离线工作时所需的资源文件。这样，才能在在线状态时，把这些文件缓存到本地。此后，当用户离线访问应用程序时，这些资源文件会自动加载，从而让用户正常使用。html5 中，通过 cache manifest 文件指明需要缓存的资源，并支持自动和手动两种缓存更新方式。

在线状态检测：开发者需要知道浏览器是否在线，这样才能够针对在线或离线的状态，做出对应的处理。在 html5 中，提供了两种检测当前网络是否在线的方式。

本地数据存储：离线时，需要能够把数据存储到本地，以便在线时同步到服务器上。为了满足不同的存储需求，html5 提供了 dom storage 和 web sql database 两种存储机制。前者提供了易用的 key/value 对存储方式，而后者提供了基本的关系存储功能。

尽管 html5 还处于草稿状态，但是各大主流浏览器都已经实现了其中的很多功能。chrome、firefox、safari 和 opera 的最新版本都对 html5 离线功能提供了完整的支持。ie8 也支持了其中的在线状态检测和 dom storage 功能。下面将具体介绍 html5 离线功能中的离线资源缓存、在线状态检测、dom storage 和 web sql database，最后通过一个简单的 web 程序说明使用 html5 开发离线应用的方法。

离线资源缓存

为了能够让用户在离线状态下继续访问 web 应用，开发者需要提供一个 cache manifest 文件。这个文件中列出了所有需要在离线状态下使用的资源，浏览器会把这些资源缓存到本地。本节先通过一个例子展示 cache manifest
文件的用途，然后详细描述其书写方法，最后说明缓存的更新方式。

cache manifest 示例

我们通过 w3c 提供的示例来说明。clock web
应用由三个文件“ clock.html ”、“ clock.css ”和“ clock.js ”组成。

清单 1. clock 应用代码

xml/html code复制内容到剪贴板

```
<! – clock.html – >

<!doctype html>

<html>

<head>

 <title>clock</title>

 <script src= ” clock.js ” ></script>

 <link rel= ” stylesheet ” href= ” clock.css ” >

</head>

<body>

 <p>the time is: <output id= ” clock ” ></output></p>

</body>

</html>


/* clock.css */

output { font: 2em sans-serif; }
```

```
/* clock.js */

settimeout(function () {

 document.getelementbyid( ‘ clock ’ ).value = new date();

}, 1000);
```

当用户在离线状态下访问 “ clock.html ” 时，页面将无法展现。为了支持离线访问，开发者必须添加 cache manifest 文件，指明需要缓存的资源。这个例子中的 cache manifest 文件为 “ clock.manifest ”，它声明了 3 个需要缓存的资源文件 “ clock.html ”、“ clock.css ” 和 “ clock.js ”。

清单 2. clock.manifest 代码

xml/html code复制内容到剪贴板

```
cache manifest

clock.html

clock.css

clock.js
```

添加了 cache manifest 文件后，还需要修改 “ clock.html ”，把 <html> 标签的 manifest 属性设置为 “ clock.manifest ”。修改后的 “ clock.html ” 代码如下。

清单 3. 设置 manifest 后的 clock.html 代码

xml/html code复制内容到剪贴板

```
<! – clock.html – >

<!doctype html>

<html manifest= ” clock.manifest ” >

<head>

 <title>clock</title>
```

```
<script src= ” clock.js ” ></script>

<link rel= ” stylesheet ” href= ” clock.css ” >

</head>

<body>

<p>the time is: <output id= ” clock ” ></output></p>

</body>

</html>
```

修改后，当用户在线访问“ clock.html ”时，浏览器会缓存“ clock.html ”、“ 持久的近义词clock.css ”和“ clock.js ”文件；而当用户离线访问时，这个 web 应用也可以正常使用了。

cache manifest 格式

下面说明书写 cache manifest 文件需要遵循的格式。

首行必须是 cache manifest。

其后，每一行列出一个需要缓存的资源文件名。

可根据需要列出在线访问的白名单。白名单中的所有资源不会被缓存，在使用时将直接在线访问。声明白名单使用 network：标识符。

如果在白名单后还要补充需要缓存的资源，可以使用 cache：标识符。

如果要声明某 uri 不能访问时的替补 uri，可以使用 fallback：标识符。其后的每一行包含两个 uri，当第一个 uri 不可访问时，浏览器将尝试使用第二个 uri。

注释要另起一行，以 # 号开头。

清单 4 的代码中给出了 cache manifest 中各类标识符的使用示例。

清单 4. cache manifest 示例代码

xml/html code复制内容到剪贴板

```
cache manifest

# 上一行是必须书写的。
```

images/sound-icon.png

images/background.png

network:

comm.cgi

下面是另一些需要缓存的资源，在这个示例中只有一个 css 文件。

xml/html code复制内容到剪贴板

cache:

style/default.css

fallback:

/files/projects /projects

更新缓存

应用程序可以等待浏览器自动更新缓存，也可以使用 javascript 接口手动触发更新。

自动更新

浏览器除了在第一次访问 web 应用时缓存资源外，只会在 cache manifest
文件本身发生变化时更新缓存。而 cache manifest 中的资源文件发生变化并不会触发更新。

手动更新

开发者也可以使用 window.applicationcache 的接口更新缓存。方法是检测
window.applicationcache.status 的值，如果是 updateready，那么可以调用
window.applicationcache.update() 更新缓存。示范代码如下。

清单 5 手动更新缓存

javascript code复制内容到剪贴板

if (window.applicationcache.status == window.applicationcache.updateready)

{

window.applicationcache.update();

}

在线状态检测

如果 web 应用程序仅仅是一些静态页面的组合，那么通过 cache manifest 缓存资源文件以后，就可以支持离线访问了。但是随着互联网的发展，特别是 web2.0 概念流行以来，用户的提交的数据渐渐成为互联网的主流。那么在开发支持离线的 web 应用时，就不能仅仅满足于静态页面的展现，还必需考虑如何让用户在离线状态下也可以操作数据。离线状态时，把数据存储在本地；在线以后，再把数据同步到服务器上。为了做到这一点，开发者首先必须知道浏览器是否在线。html5 提供了两种检测是否在线的方式：navigator.online 和 online/offline 事件。

navigator.online

navigator.online 属性表示当前是否在线。如果为 true, 表示在线；如果为 false, 表示离线。当网络状态发生变化时，navigator.online 的值也随之变化。开发者可以通过读取它的值获取网络状态。

online/offline 事件

当开发离线应用时，通过 navigator.online 获取网络状态通常是不够的。开发者还需要在网络状态发生变化时立刻得到通知，因此 html5 还提供了 online/offline 事件。当在线 / 离线状态切换时，online/offline 事件将触发在 body 元素上，并且沿着 document.body、document 和 window 的顺序冒泡。因此，开发者可以通过监听它们的 online/offline 事件来获悉网络状态。

dom storage

在开发支持离线功能的 web 应用时，开发者需要在本地存储数据。当前浏览器支持的 cookie 虽然也可以用来存储数据，但是 cookie 长度非常小（通常几 k），而且功能有限。因此，html5 中新引入了 dom storage 机制，用于存储 key/value 对，它的设计目标是提供大规模、安全且易用的存储功能。

dom storage 分类

dom storage 分为两类：sessionstorage 和 localstorage。除了以下区别外，这两类存储对象的功能是完全一致的。

sessionstorage 用于存储与当前浏览器窗口关联的数据。窗口关闭后，sessionstorage 中存储的数据将无法使用。

localstorage 用于长期存储数据。窗口关闭后，localstorage

中的数据仍然可以被访问。所有浏览器窗口可以共享 localstorage 的数据。

dom storage 接口

每一个 storage 对象都可以存储一系列 key/valu高三复读报名e 对，storage 接口定义为：

javascript code复制内容到剪贴板

```
interface storage {
 readonly attribute unsigned long length;
 getter domstring key(in unsigned long index);
 getter any getitem(in domstring key);
 setter creator void setitem(in domstring key, in any data);
 deleter void removeitem(in domstring key);
 void clear();
};
```

其中最常用的接口是 getitem 和 setitem。getitem 用于获取指定 key 的 value，而 setitem 用于设置指定 key 的 value。

dom storage 示例

这里给出一个使用了 sessionstorage 的例子，localstorage 的用法与它相同。首先使用 setitem 添加了一个名为“ username ”的项，它的值是“ developerworks ”。然后，调用 getitem 得到“ username ”的值，并且弹出提示框显示它。最后，调用 removeitem 删除“ username ”。

清单 6 dom storage 示例代码

xml/html code复制内容到剪贴板

```
<!doctype html>
<html>
<body>
<script>
```

```
// 在 sessionstorage 中定义 ’ username ’ 变量

sessionstorage.setitem( ‘ username ’ , ‘ developerworks ’ );

// 访问 ’ username ’ 变量

alert( “ your user is: ” + sessionstorage.getitem( ‘ username ’ ));

// 最后删除 ’ username ’

sessionstorage.removeitem( ‘ username ’ );

</script>

</body>

</html>
```

web sql database

除了 dom storage 以外，html5 中还有另外一种数据存储方式 web sql database。它提供了基本的关系数据库功能，支持页面上的复杂的、交互式的数据存储。它既可以用来存储用户产生的数据，也可以作为从服务器获取数据的本地高速缓存。例如可以把电子邮件、日程等数据存储到数据库中。web sql database
支持数据库事务的概念，从而保证了即使多个浏览器窗口操作同一数据，也不会产生冲突。

web sql database 基本用法

创建和打开数据库

使用数据库的第一步是创建并打开数据库，api 是
opendatabase。当数据库已经存在时，opendatabase
仅仅打开数据库；如果这个数据库不存在，那么就创建一个空数据库并且打开它。opendatabase 的定义是：

javascript code复制内容到剪贴板

```
database opendatabase(in domstring name, in domstring version,

 in domstring displayname, in unsigned long estimatedsize,

 in optional databasecallback creationcallback);
```

name：数据库名。

version：数据库版本。

displayname：显示名称。

estimatedsize：数据库预估长度（以字节为单位）。

creationcallback：回调函数。

执行事务处理

在打开数据库以后，就可以使用事务 api transaction对老师的寄语。每一个事务作为操作数据库的原子操作，不会被打断，从而避免了数据冲突。transaction 的定义是：

javascript code复制内容到剪贴板

void transaction(in sqltransactioncallback callback,

in optional sqltransactionerrorcallback errorcallback,

in optional sqlvoidcallback successcallback);

callback：事务回调函数，其中可以执行 sql 语句。

errorcallback：出错回调函数。

successcallback：执行成功回调函数。

执行 sql 语句

在事务的回调函数 callback 中，可以执行 sql 语句，api 是 executesql。executesql 的定义是：

javascript code复制内容到剪贴板

void executesql(in domstring sqlstatement,

in optional objectarray arguments, in optional sqlstatementcallback callback,

in optional sqlstatementerrorcallback errorcallback);

sqlstatement：sql 语句。

arguments：sql 语句需要的参数。

callback：回调函数。

errorcallback：出错回调函数。

web sql database 示例

下面通过一个例子说明 web sql database 的基本用法。它首先调用 opendatabase 创建了名为“ foodb ”的数据库。然后使用 transaction 执行两条 sql 语句。第一条 sql 语句创建了名为“ foo ”的表，第二条 sql 语句向表中插入一条记录。

清单 7 web sql database 示例代码

javascript code复制内容到剪贴板

```
var db = opendatabase( ‘ foodb ’ , ‘ 1.0 ’ , ‘ foodb ’ , 2 * 1024);

db.transaction(function (tx) {

tx.executesql( ‘ create table if not exists foo (id unique, text) ’ );

tx.executesql( ‘ insert into foo (id, text) values (1, “ foobar ” ) ’ );

});
```

离线应用示例

最后，通过一个例子来说明使用 html5 开发离线应用的基本方法。这个例子会用到前面提到的离线资源缓存、在线状态检测和 dom storage 等功能。假设我们开发一个便签管理的 web 应用程序，用户可以在其中添加和删除便签。它支持离线功能，允许用户在离线状态下添加、删除便签，并且当在线以后能够同步到服务器上。

应用程序页面

这个程序的界面很简单，如图 1 所示。用户点击“ new note ”按钮可以在弹出框中创建新的便签，双击某便签就表示删除它。www.2cto.com

图 1. 应用程序页面

这个页面的源文件是 index.html，它的代码如清单 8 所示。

清单 8 页面 html 代码

xml/html code复制内容到剪贴板

```
<html manifest= ” notes.manifest ” >
```

<head>

<script type= ” text/javascript ” src= ” server.js ” ></script>

<script type= ” text/javascript ” src= ” data.js ” ></script>

<script type= ” text/javascript ” src= ” ui.js ” ></script>

<title>note list</title>

</head>

<body onload = “ syncwithserver() ” >

<input type= ” button ” value= ” new note ” onclick= ” newnote() ” >

<ul id= ” list ” >

</body>

</html>

在 body 中声明了一个按钮和一个无序列表。当按下“ new note ”按钮时，newnote
函数将被调用，它用来添加一条新的便签。而无序列表初始为空，它是用来显示便签的列表。

cache manifest 文件

定义 cache manifest
文件，声明需要缓存的资源。在这个例子中，需要缓存“ index.html ”、“ server.js ”、“ data.js ”和“ ui.js ”等 4 个文件。除了前面列出的“ index.html ”外，“ server.js ”、“ data.js ”和“ ui.js ”分别包含服务器相关、数据存储和用户界面代码。cache manifest 文件定义如下。

清单 9 cache manifest 文件

xml/html code复制内容到剪贴板

cache manifest

index.html

server.js

data.js

ui.js

用户界面代码

用户界面代码定义在 ui.js 中。

清单 10 用户界面代码 ui.js

javascript code复制内容到剪贴板

```
function newnote()
{
 var title = window.prompt( “ new note: ” );
 if (title)
 {
  add(title);
 }
}

function add(title)
{
 // 在界面中添加
 adduiitem(title);
 // 在数据中添加
 adddataitem(title);
}

function remove(title)
{
 // 从界面中删除
 removeuiitem(title);
 // 从数据中删除
 removedataitem(title);
}

function adduiitem(title)
{
 var item = document.createelement( “ li ” );
```

```
 item.setattribute( “ ondblclick ” , “ remove( ‘ ” +title+ ” ‘ ) ” );
 item.innerhtml=title;

 var list = document.getelementbyid( “ list ” );
 list.appendchild(item);
}

function removeuiitem(title)
{
 var list = document.getelementbyid( “ list ” );
 for (var i = 0; i < list.children.length; i++) {
  if(list.children[i].innerhtml == title)
  {
  list.removechild(list.children[i]);
  }
 }
}
```

ui.js 中的代码包含添加便签和删除便签的界面操作。

添加便签

用户点击“ new note ”按钮，newnote 函数被调用。

newnote 函数会弹出对话框，用户输入新便签内容。newnote 调用 add 函数。

add 函数分别调用 adduiitem 和 adddataitem 添加页面元素和数据。adddataitem 代码将在后面列出。

adduiitem 函数在页面列表中添加一项。并指明 ondblclick 事件的处理函数是 remove，使得双击操作可以删除便签。

删除便签

用户双击某便签时，调用 remove 函数。

remove 函数分别调用 removeuiitem 和 removedataitem 删除页面元素和数据。removedataitem 将在后面列出。

removeuiitem 函数删除页面列表中的相应项。

数据存储代码

数据存储代码定义在 data.js 中

清单 11 数据存储代码 data.js

javascript code复制内容到剪贴板

```
var storage = window[ ‘ localstorage ’ ];

function adddataitem(title)
{
 if (navigator.online) // 在线状态
 {
 addserveritem(title);
 }
 else // 离线状态
 {
 var str = storage.getitem( “ toadd ” );
 if(str == null)
 {
 str = title;
 }
 else
 {
 str = str + “ , ” + title;
 }
 storage.setitem( “ toadd ” , str);
 }
}

function removedataitem(title)
```

```
{
 if (navigator.online) // 在线状态
 {
 removeserveritem(title);
 }
 else // 离线状态
 {
 var str = storage.getitem(“ toremove ”);
 if(str == null)
 {
 str = title;
 }
 else
 {
 str = str + “ , ” + title;
 }
 storage.setitem(“ toremove ”, str);
 }
}

function syncwithserver()
{
 // 如果当前是离线状态，不需要做任何处理
 if (navigator.online == false)return;

 var i = 0;
 // 和服务器同步添加操作
 var str = storage.getitem(“ toadd ”);
```

```
if(str != null)
{
var additems = str.split( “ , ” );
for(i = 0; i<additems.length; i++)
{
adddataitem(additems[i]);
}
storage.removeitem( “ toadd ” );
}

// 和服务器同步删除操作
str = storage.getitem( “ toremove ” );
if(str != null)
{
var removeitems = str.split( “ , ” );
for(i = 0; i<removeitems.length; i++)
{
removedataitem(removeitems[i]);
}
storage.removeitem( “ toremove&#男生暗恋一个女生的表现8221;);
}

// 删除界面中的所有便签
var list = document.getelementbyid( “ list ” );
while(list.lastchild != list.firstelementchild)
list.removechild(list.lastchild);
if(list.firstelementchild)
list.removechild(list.firstelementchild);
```

```
 // 从服务器获取全部便签，并显示在界面中
 var allitems = getserveritems();
 if(allitems != “ ” )
 {
 var items = allitems.split( “ , ” );
 for(i = 0; i<items.length; i++)
 {
 adduiitem(items[i]);
 }
 }
}
window.addeventlistener( “ online ” , syncwithserver,false);
```

data.js 中的代码包含添加便签、删除便签和与服务器同步等数据操作。其中用到了 navigator.online 属性、online 事件、dom storage 等 html5 新功能。

添加便签：adddataitem

通过 navigator.online 判断是否在线。

如果在线，那么调用 addserveritem 直接把数据存储到服务器上。addserveritem 将在后面列出。

如果离线，那么把数据添加到 localstorage 的 “ toadd ” 项中。

删除便签：removedataitem

通过 navigator.online 判断是否在线。

如果在线，那么调用 removeserveritem 直接在服务器上删除数据。removeserveritem 将在后面列出。

如果离线，那么把数据添加到 localstorage 的 “ toremove ” 项中。

数据同步：syncwithserver

在 data.js 的最后一行，注册了 window 的 online 事件处理函数 syncwithserver。当 online 事件发生时，syncwithserver 将被调用。其功能如下。

如果 navigator.online 表示当前离线，则不做任何操作。

把 localstorage 中 “ toadd ” 项的所有数据添加到服务器上，并删除 “ toadd ” 项。

把 localstorage 中 “ toremove ” 项的所有数据从服务器中删除，并删除 “ toremove ” 项。

删除当前页面列表中的所有便签。

调用 getserveritems 从服务器获取所有便签，并添加在页面列表中。getserveritems 将在后面列出。

服务器相关代码

服务器相关代码定义在 server.js 中。

清单 12 服务器相关代码 server.js

javascript code复制内容到剪贴板

```
function addserveritem(title)
{
 // 在服务器中添加一项
}

function removeserveritem(title)
{
 // 在服务器中删除一项
}

function getserveritems()
{
 // 返回服务器中存储的便签列表
}
```

由于这部分代码与服务器有关，这里只说明各个函数的功能，具体实现可以根据不同服务器编写代码。

在服务器中添加一项：addserveritem

在服务器中删除一项：removeserveritem

返回服务器中存储的便签列表：getserveritems

总结

本文介绍了 html5 为了支持离线应用程序新增的强大功能。通过本文，读者能够了解到 html5 中离线相关特性的基本用法，从而掌握利用 html5 开发 web 离线应用的方法。

更多 作文 请访问 https://www.wtabcd.cn/fanwen/list/92_0.html

文章生成doc功能，由范文网开发