

HTML5程序设计基础

第五章 离线Web应用



主要内容

01 离线Web应用的作用

02 创建离线应用

03 离线控制的 API

01

离线Web应用的作用

Web程序的优势

 Web程序相对本地应用的一大优势



程序总是从网络请求，
可以实时更新程序。

Web程序的劣势

🎨 网络总是可靠的吗？



加载完网页后突然断网，刷新页面后内容没有了
在新窗口重新访问该网页，在断网状态下打开的还是原来页面

Web程序的劣势

某些应用只需要偶尔进行网络通信，例如：evernote

理想状态的Web应用：



在线时获得最新的应用

在本地存储应用资源，无论是否在线都可以使用

HTML5 提供离线Web应用的实现机制。使Web应用可以在用户离线的状态下进行访问。

离线Web程序的作用

离线浏览 - 用户可在应用离线时使用它们

加快速度 - 已缓存资源加载得更快

减少服务器负载 - 浏览器将只从服务器下载更新过或更改过的资源

离线Web程序的应用



邮件



个人事务管理



博客发布平台

还有哪些类型的应用需要离线支持？

本地缓存与浏览器网页缓存的区别

应用范围

本地缓存是为整个Web应用程序服务的，且只缓存指定的网页；
而浏览器的网页缓存只服务于单个网页，任何网页都具有网页缓存。

可靠性

本地缓存是可靠的，可控制的；
而网页缓存是不安全、不可靠的。

02

创建离线应用

离线应用的兼容性




已支持



未支持

离线应用的实现方式

 将Web应用所使用的资源（HTML、CSS、JavaScript、图片等文件缓存在浏览器本地）

 离线技术包含的两个部分：

缓存清单文件：管理要缓存的文件列表

JavaScript接口：提供用于更新缓存文件的方法以及对缓存文件的操作。

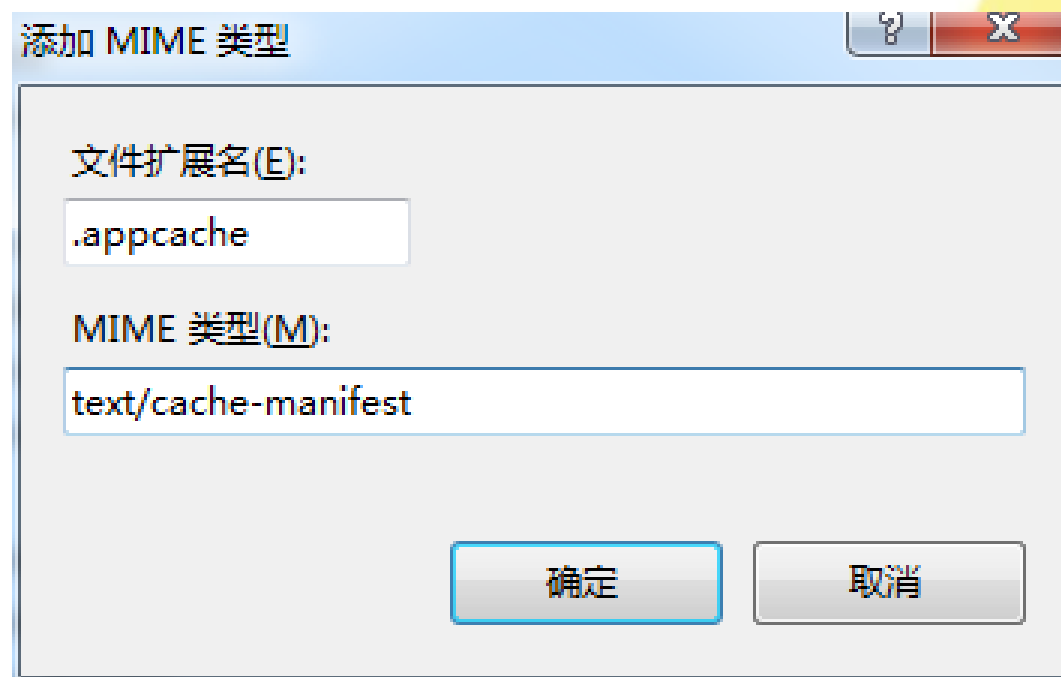
创建离线应用程序

❖ 第一步：创建缓存清单文件（manifest 文件）

文件扩展名使用 **appcache**；

manifest 文件需要在web服务器上配置正确的 MIME 类型为：
text/cache-manifest。

（ MIME: 多用途互联网邮件扩展类型，是设定某种扩展名的文件用一种应用程序来打开的方式类型 ）



创建离线应用程序

第二步：在html标记中指定使用缓存文件

```
<html manifest="cacheData.appcache">
```

示例：5-1

Manifest文件

 manifest 文件可分为三个部分

CACHE MANIFEST

在此标题下列出的文件将在首次下载后进行缓存，写在第一行，必须有该部分

NETWORK

在此标题下列出的文件需要与服务器的连接，且不会被缓存

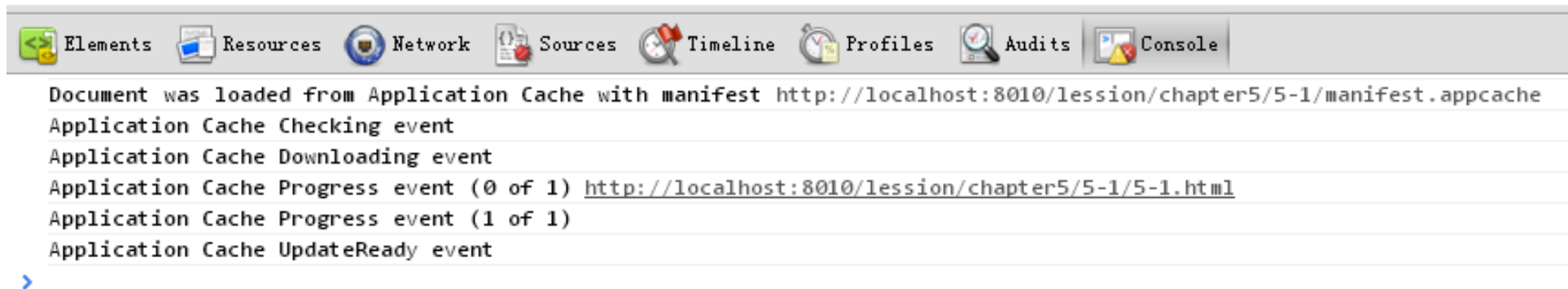
FALLBACK

提供了获取不到缓存资源时的备选资源路径（比如 404 页面）

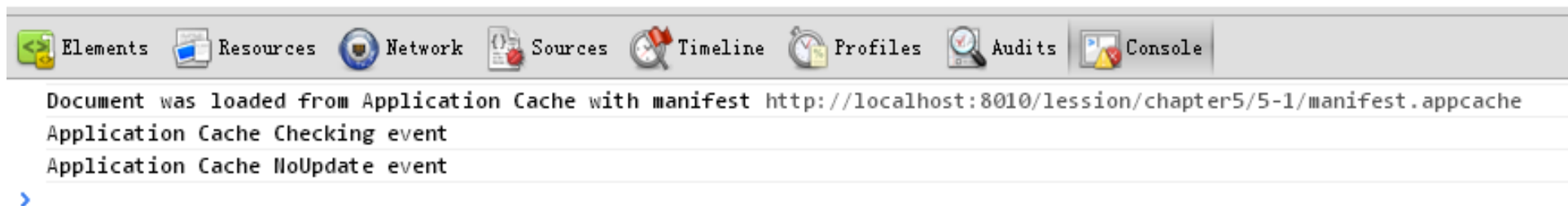
- 引入manifest文件的HTML页面可以在CACHE MANIFEST省略
- 可以使用*表示其他所有资源/文件

示例：5-2 

监测缓存状况



执行缓存过程



已缓存过

离线应用的更新

🎨 构建离线应用后，即使在线状态，用户也会访问缓存文件。

及时更新用户的缓存文件非常重要。

🎨 HTML5离线缓存更新：

1. 用户清空浏览器缓存
2. manifest 文件被修改
3. 由程序来更新应用缓存

Manifest文件控制缓存

- 在manifest文件中，通过注释指明版本是一种较好的缓存管理方式，通过修改版本号可以通知浏览器进行更新。
- #** 表示注释行标识符。

```
CACHE MANIFEST
```

```
#version 1.0|
```

```
style.css
```

```
app.js
```

```
NETWORK:
```

```
FALLBACK:
```

```
login.html info.html
```

示例：5-3

03

离线控制的 API

离线控制API

提供对象：**applicationCache**

1. applicationCache API 是一个操作应用缓存的接口。
2. 代表本地缓存，可用它来通知用户本地缓存中已经被更新，也允许用户手动更新本地缓存
3. 访问方法：`window.applicationCache`

浏览器兼容性检测

- 通过判断window.applicationCache对象是否存在进行浏览器兼容性检测

```
window.onload = function() {  
    if (window.applicationCache) {  
        //离线操作API  
    }  
    else {  
        document.getElementById("info").innerHTML =  
            "你的浏览器已经out了";  
    }  
}
```

示例：5-4

applicationCache API

🎨 获得缓存的状态：window.applicationCache.**status**

🎨 缓存的6种状态：

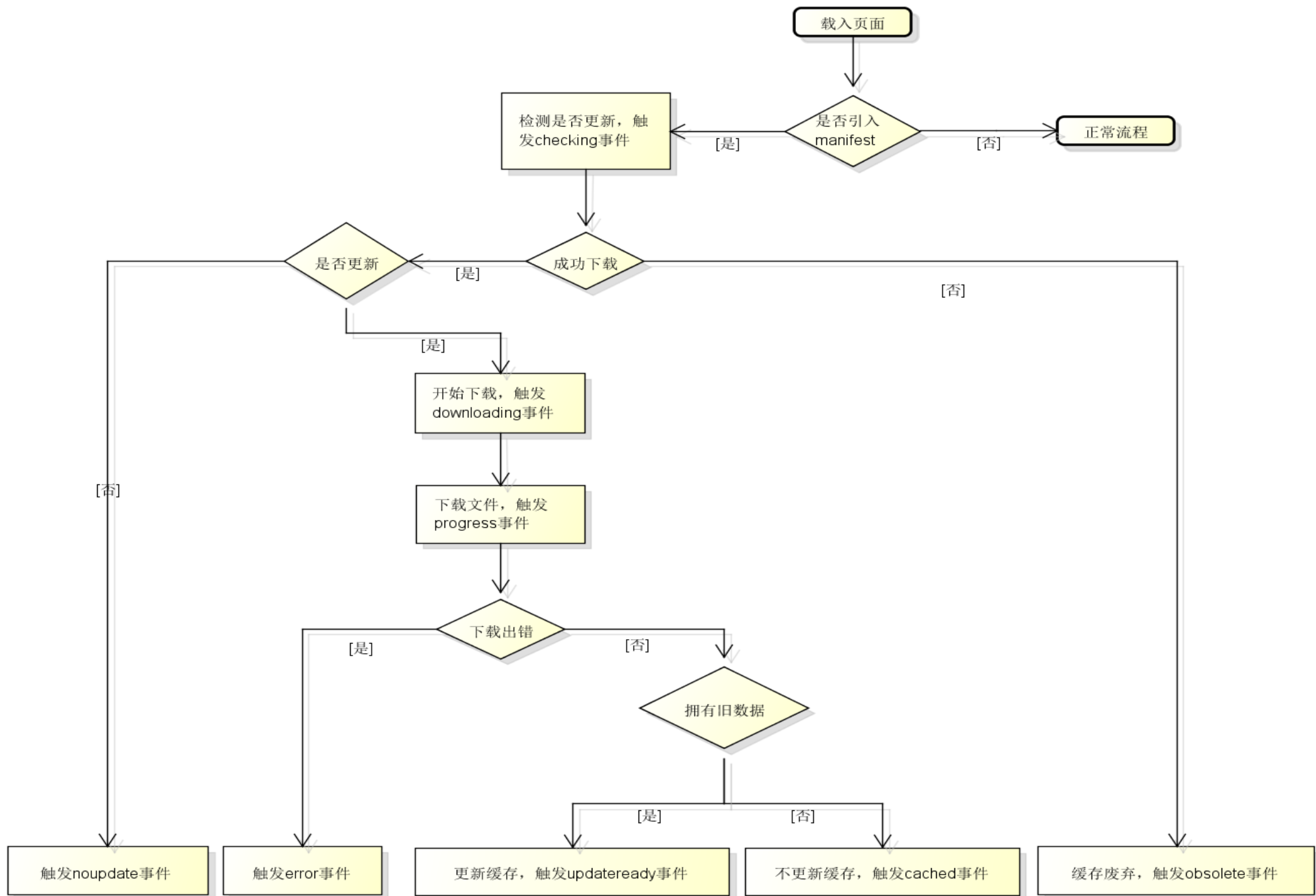
Status值	缓存状态
0	UNCACHED（未缓存）
1	IDLE（空闲状态）
2	CHECKING（检查中）
3	DOWNLOADING（下载中）
4	UPDATEREADY（更新就绪）
5	OBSOLETE（过期）

applicationCache 缓存对象的事件

事件名称	说明
checking	当user agent检查更新时，或者第一次下载manifest清单时，它往往是第一个被触发的事件
noupdate	当检查到Manifest中清单文件不需要更新时，触发该事件
downloading	第一次下载或更新manifest清单文件时，触发该事件
progress	在清单文件下载过程中周期性触发
cached	当manifest清单文件下载完毕及成功缓存后，触发该事件
updateready	表示缓存清单文件已经下载完毕，可通过重新加载页面读取缓存文件或者通过方法swapCache切换到新的缓存文件。常用于本地缓存更新版本后的提示

applicationCache 缓存对象的事件

事件名称	说明
Obsolete	加入访问manifest缓存文件返回HTTP404错误（页面未找到）或者410错误（永久消失）时，触发该事件
Error	若要达到触发该事件，需要满足以下几种情况之一： 1、已经触发obsolete事件 2、manifest文件没有改变，但缓存文件中存在文件下载失败 3、获取manifest资源文件时发生致命错误 4、当更新本地缓存时，manifest文件再次被更改



离线事件监听

- ❖ 在实际的应用中，可以通过事件监听，并根据当前 applicationCache 对象的状态处理相关业务。
- ❖ applicationCache.addEventListener ()

示例：5-5

applicationCache API

🎨 更新缓存方法：

调用当前应用资源下载过程

`applicationCache.update ()`

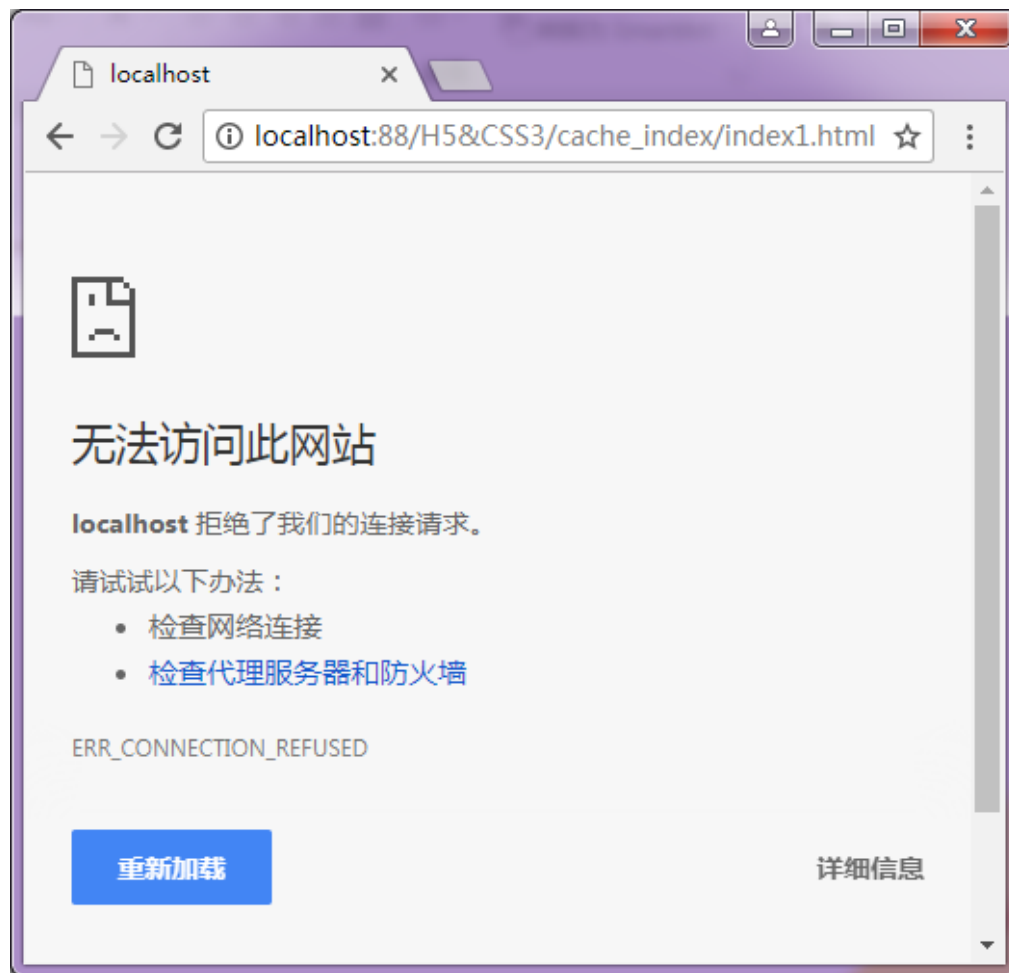
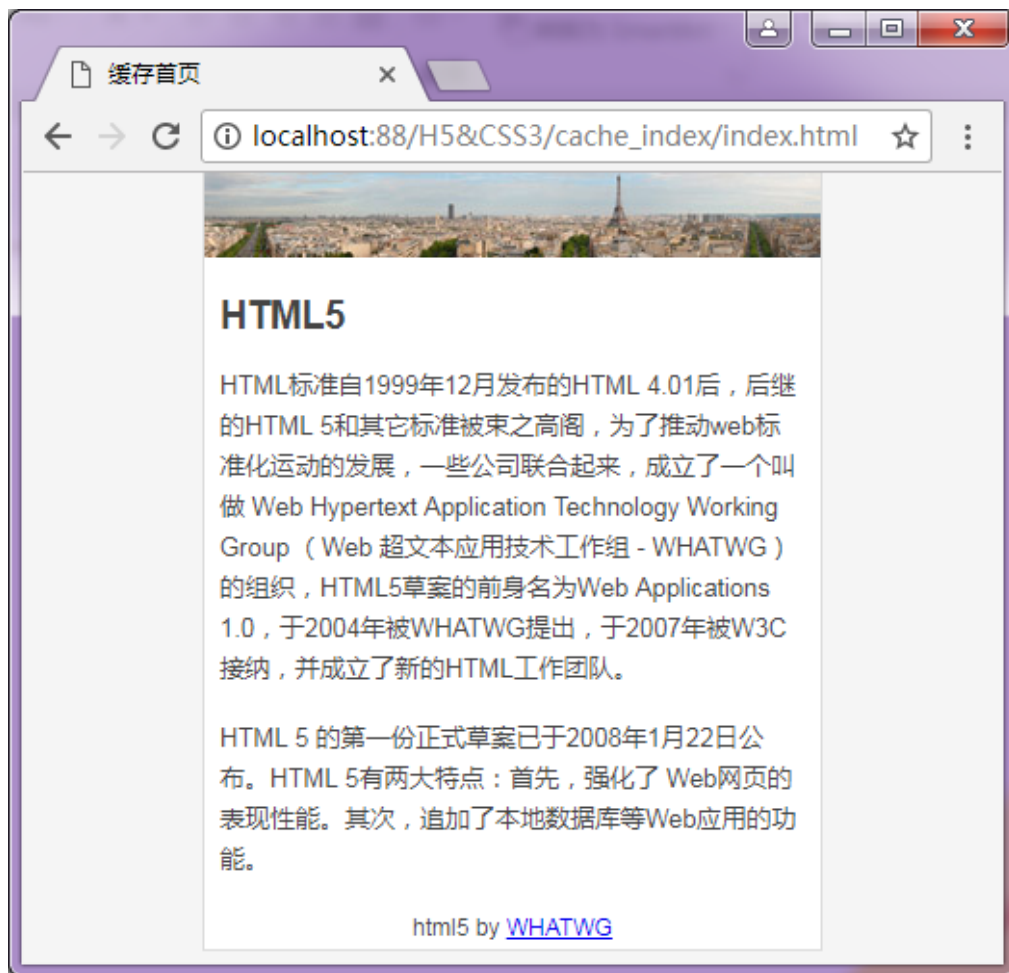
更新到最新的缓存，不会使之前加载的资源突然被重新加载

`applicationCache.swapCache ()`

示例：5-6

实战：缓存首页

服务器停止工作或无法上网时，用户依然可以访问本地缓存中的网页



THANKYOU

