Module 9

Adding Offline Support to Web Applications

Module Overview

- Reading and Writing Data Locally
- Adding Offline Support by Using the Application Cache

Lesson 1: Reading and Writing Data Locally

- Maintaining Session State Information by Using Cookies
- Persisting Session Data by Using Session Storage
- Persisting Data Across Sessions by Using Local Storage
- Handling Storage Events
- Storing Structured Data by Using the Indexed Database API

Maintaining Session State Information by Using Cookies

- Cookies:
  - Were designed to implement session tokens
  - Are sent to the server on every page request
  - Are small files of limited size, up to 4 KB
  - Are open to abuse
  - Have no synchronization or concurrency mechanism
- Cookies were not designed for general-purpose data storage

Persisting Session Data by Using Session Storage

- Use the **sessionStorage** object to store and retrieve text data for a session:

```
sessionStorage.setItem("myKey", "some text value");
var textFromSession1 = sessionStorage.getItem("myKey");

sessionStorage["myKey"] = "some text value";
var textFromSession2 = sessionStorage["myKey"];

sessionStorage.myKey = "some text value";
var textFromSession3 = sessionStorage.myKey;
```

- Session data is only available in the session that creates it
  - Session storage is cleared when the user finishes the browser session

Persisting Data Across Sessions by Using Local Storage

- Use the **localStorage** object to persist data across sessions and web pages:

```
localStorage.setItem("myKey", "some text value");
var textData = localStorage.getItem("myKey");

localStorage["myKey"] = "some text value";
var textData = sessionStorage["myKey"];

localStorage.myKey = "some text value";
var textData = sessionStorage.myKey;
```

- Data is persisted until it is explicitly removed

Handling Storage Events

- Use the **storage** event to notify a web page of changes made to session and local storage:

```
function myStorageCallback( e ) {
    alert( "Key:" + e.key + " changed to " + e.newValue );
}
…
window.addEventListener("storage", myStorageCallback, true );
```

- Properties of the event object:

key	– name of the value which has changed
oldValue	– the original value
newValue	– the new value
url	– the origin of the event
storageArea	– a reference to the store that has changed

Storing Structured Data by Using the Indexed Database API

- IndexedDB provides an efficient means for storing structured data on the user's computer
- The API is asynchronous, and includes the following features:
  - Multiple object stores
  - **add()**, **put()**, **get()**, and **delete()** operations on data
  - Transactions
  - Queries by using cursors
  - Indexes to speed up common queries

Lesson 2: Adding Offline Support by Using the Application Cache

- Configuring the Application Cache
- Monitoring with the Application Cache
- Triggering Resource Updates by Using the Manifest
- Testing for Network Connectivity
- Demonstration: Adding Offline Support to Web Applications

Configuring the Application Cache

- The application cache manifest file specifies the resources to cache, and how they should be updated:

```
CACHE MANIFEST

CACHE:
index.html
verification.js
site.css
graphics/logo.jpg

NETWORK:
/login

# alternatives paths
FALLBACK:
/ajax/account/  /noCode.htm
```

- Each web page that uses cached resources should reference the manifest file:

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

Monitoring with the Application Cache

- Use the **applicationCache** object to monitor the cache:

```
applicationCache.addEventListener( "error", function() {
    alert( "Error while downloading resources to the application cache");
}, true );
```

- Examine the **status** property to determine cache state:

0	UNCACHED	No resources have been downloaded.
1	IDLE	All cached resources have been downloaded.
2	CHECKING	The cache is being checked for updates.
3	DOWNLOADING	Resources are being downloaded to the cache.
4	UPDATEREADY	The cache has been updated with new resources.
5	OBSOLETE	The manifest is missing and no cache is available.

Triggering Resource Updates by Using the Manifest

- A web page may continue to use cached resources even if newer versions are available
- To force an update:
  - Make a significant change to the manifest file, or
  - Initiate a check for updates by using the **update()** function, and then use the **swapCache()** function

```
applicationCache.update();
...
if (applicationCache.status == 4 ) {
    applicationCache.swapCache();
}
```

Testing for Network Connectivity

- Sometimes it is better to disable functionality that requires a network connection
- Use the **onLine** property of the **navigator** object to detect the network status
- Handle the **online** and **offline** events of the window object to track changes to network connectivity

Demonstration: Adding Offline Support to Web Applications

In this demonstration, you will learn about the tasks that you will perform in the lab for this module.

Lab: Adding Offline Support to Web Applications

- Exercise 1: Caching Offline Data by Using the Application Cache API
- Exercise 2: Persisting User Data by Using the Local Storage API

Logon Information

- Virtual Machines: 20480B-SEA-DEV11, MSL-TMG1
- User Name: **Student**
- Password: **Pa$$w0rd**

Estimated Time: 60 minutes

Lab Scenario

The conference organizers are concerned that the venue has poor Wi-Fi coverage in some locations, meaning that attendees might not always be able to access the conference website on their tablets and laptops. The Schedule page is especially important because attendees need to know when sessions are running.

You have decided to make parts of the web application available offline by using the offline web application features of HTML5. After an attendee has visited the online website once, their browser will have downloaded and cached the important pages. If a Wi-Fi connection is unavailable, the attendee will still be able to view the website by using the cached information.

Module Review and Takeaways

- Review Question(s)