Module 8

Creating Interactive Pages by Using HTML5 APIs

Module Overview

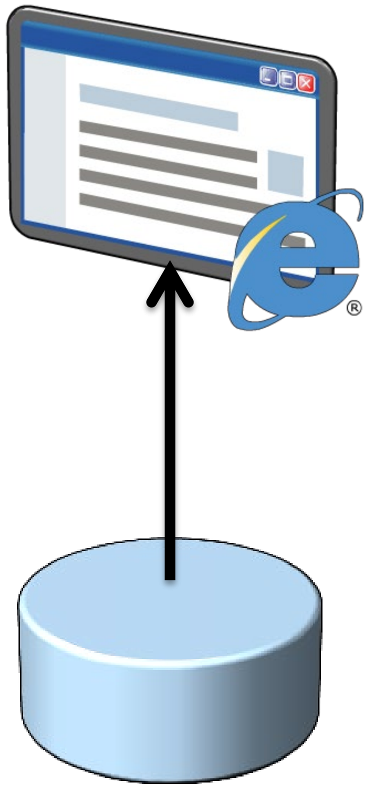
- Interacting with Files
- Incorporating Multimedia
- Reacting to Browser Location and Context
- Debugging and Profiling a Web Application

Lesson 1: Interacting with Files

- HTML5 File Interfaces
- The FileReader Interface
- Reading a Text File
- Reading a Binary File
- Implementing Drag-and-Drop

HTML5 File Interfaces

- The HTML5 File API enables a web application to access the local file system
- There are four key interfaces:
  - **Blob** – immutable raw binary data
  - **File** - readonly information about a file
  - **FileList** – an array of files
  - **FileReader** – methods for reading data from a file or blob



The FileReader Interface

- The **FileReader** interface provides methods for reading a file or blob:
  - **readAsText()** – used for reading text files
  - **readAsDataURL()** – used for reading binary files
  - **readAsArrayBuffer()** – used for reading data into a buffer array

- **FileReader** reads data asynchronously and fires events:

progress	**abort**	**loadend**
load	**error**	

Reading a Text File

To read a text file:

1. Get a File or Blob object, either by using an **<input type="field">** element or by drag-and-drop.
2. Create a **FileReader** object and handle events such as **load** and **error**.
3. Invoke **readAsText()** on the **FileReader** object.
4. In the **load** event handler function, access the text content in the **result** property of the event target.
5. In the **error** event handler function, implement appropriate error handling.

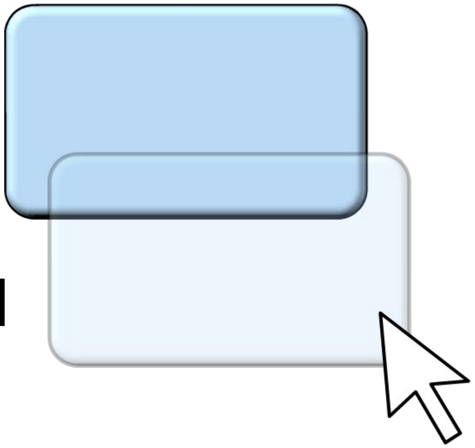
Reading a Binary File

To read a binary file:

1. Get a File or Blob object, either by using an **<input type="file">** element or by drag and drop.
2. Create a **FileReader** object and handle events such as **load** and **error**.
3. Invoke $\boxed{\textbf{readAsDataURL()}}$ on the **FileReader** object.
4. In the **load** event handler function, access the text content in the **result** property of the event target.
5. In the **error** event handler function, implement appropriate error handling.

Implementing Drag-and-Drop

- HTML5 supports drag-and-drop
  - The user can drag HTML elements, or files from the local file system
  - The user can drop items onto drop-enabled target elements



- To support drag and drop operations
  - Enable drag support on HTML elements, if required
  - Enable drop support on HTML drop target elements
  - Handle dragover and drop events on HTML drop target elements

Lesson 2: Incorporating Multimedia

- Playing Video Content by Using the <video> Tag
- Supporting Multiple Video Formats
- Interacting with Video in JavaScript Code
- Playing Audio Content by Using the <audio> Tag

Playing Video Content by Using the <video> Tag

- HTML5 enables a web application to play video files natively, without requiring plugins

- Use the **<video>** tag and set the attributes:
  - **src**
  - **width** and **height**
  - **poster**
  - **controls**
  - **autoplay**
  - **loop**
  - **muted**

```
<video src="MyVideo.mp4"
          width="300" height="200"
          poster="MyPoster.jpg"
          autoplay="autoplay"
          muted="muted"
          controls="controls"
          loop="loop" >
</video>
```

Supporting Multiple Video Formats

- A **<video>** element can support multiple video formats:

```
<video poster="MyPoster.jpg" autoplay controls>
   <source src="MyVideos/MyVideo.mp4"  type='video/mp4' />
   <source src="MyVideos/MyVideo.webm" type='video/webm' />
   <source src="MyVideos/MyVideo.ogv"  type='video/ogg' />
</video>
```

- You can embed Silverlight or Flash content in a **<video>** tag as a fall-back

Interacting with Video in JavaScript Code

- An application can interact with a **video** object in JavaScript code:

```
var newVideo = document.createElement("video");
newVideo.src = nameOfVideoFile;
newVideo.loop = true;
newVideo.controls = true;
newVideo.poster = "ImageLoading.png";

…

newVideo.addEventListener("loadeddata", function() {
    newVideo.play();
  }, false);
```

Playing Audio Content by Using the <audio> Tag

- Use the **<audio>** tag to play audio files natively, without requiring plugins:

```
<audio src="MyAudio.mp3"></audio>
```

- The JavaScript API for audio is similar to the API for video

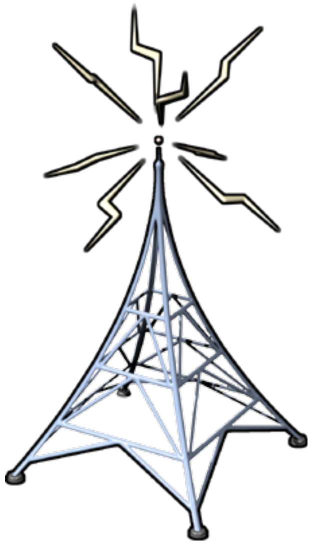
Lesson 3: Reacting to Browser Location and Context

- The HTML5 Geolocation API
- Requesting Geolocation Information
- Processing Geolocation Information
- Handling Geolocation Errors
- Detecting the Context for a Page

The HTML5 Geolocation API

- The Geolocation API enables a browser to determine the longitude and latitude of its current location

- A host device can use several techniques to obtain geolocation information:
  - IP address
  - GPS positioning
  - Wi-Fi
  - Cell phone location
  - User-defined location information



Requesting Geolocation Information

- To make a one-shot request for position information:

```
navigator.geolocation.getCurrentPosition(myPositionCallbackFunction,
                                         myPositionErrorCallbackFunction,
                                         {enableHighAccuracy: true, timeout: 5000} );
```

- To receive repeated position information updates:

```
var watchID =
      navigator.geolocation.watchPosition(myPositionCallbackFunction,
                        myPositionErrorCallbackFunction,
                        {enableHighAccuracy: true, maximumAge: 10000} );
...
navigator.geolocation.clearWatch(watchID);
```

Processing Geolocation Information

- Geolocation properties include:
  - **latitude**
  - **longitude**
  - **accuracy**

- Geolocation data may include the following optional properties:
  - **altitude**
  - **altitudeAccuracy**
  - **heading**
  - **speed**

Handling Geolocation Errors

- If an error occurs during a geolocation request, the following properties are available:
  - **code**
    - **PositionError.PERMISSION_DENIED**
    - **PositionError.POSITION_UNAVAILABLE**
    - **PositionError.TIMEOUT**
  - **message**

```
function myPositionErrorCallbackFunction(error) {
    var errorMessage = error.message;
    var errorCode = error.code;
    // Add code here, to process the information.
}
```

Detecting the Context for a Page

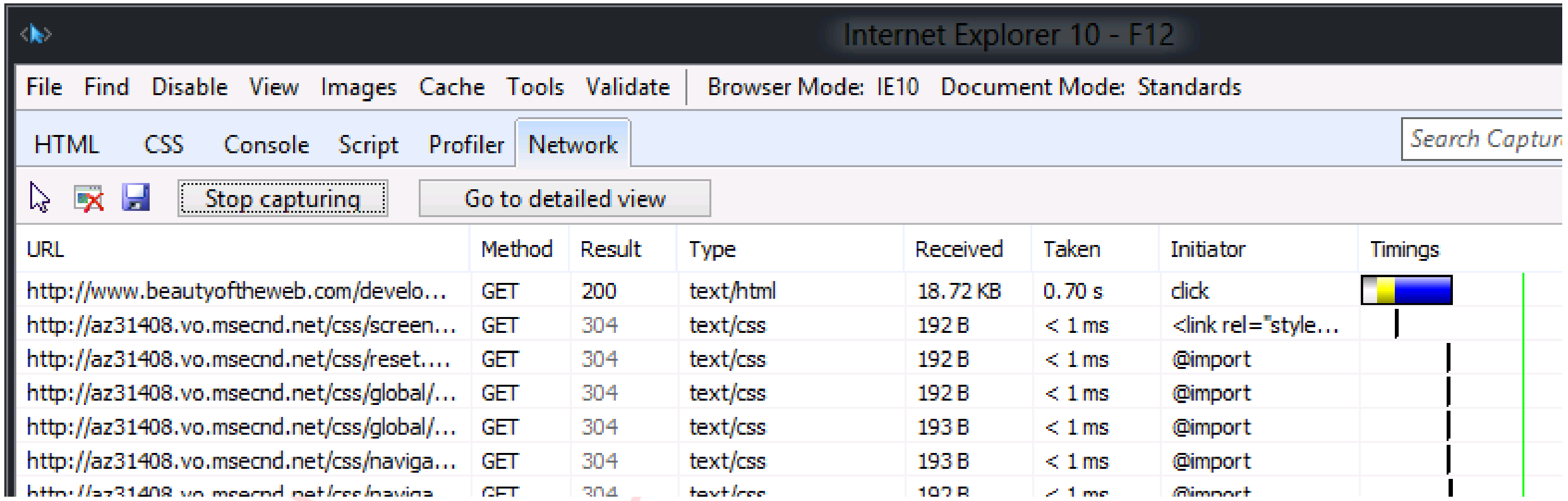
- Page Visibility API
  - Enables an application to determine whether a page is currently visible.
- Offline detection
  - Enables an application to detect whether the browser has a live connection to a server.
- User agent information
  - Enables an application to obtain the user agent string for the browser.

Lesson 4: Debugging and Profiling a Web Application

- Overview of the F12 Developer Tools in Internet Explorer 10
- Demonstration: Using the F12 Developer Tools to Debug JavaScript Code
- Demonstration: Using the F12 Developer Tools to Profile a Web Application
- Demonstration: Creating Interactive Pages with HTML5 APIs

Overview of the F12 Developer Tools in Internet Explorer 10

- The Navigation Timing API enables an application to determine the download speed for a web page:
  - **window.performance.navigation**
  - **window.performance.timing**
- The F12 Developer Tools provide debugging and profiling capabilities in Internet Explorer


Internet Explorer 10 - F12
File Find Disable View Images Cache Tools Validate
Browser Mode: IE10 Document Mode: Standards
HTML CSS Console Script Profiler Network
Search Captur
Stop capturing
Go to detailed view

URL	Method	Result	Type	Received	Taken	Initiator	Timings
http://www.beautyoftheweb.com/develo...	GET	200	text/html	18.72 KB	0.70 s	click	
http://az31408.vo.msecnd.net/css/screen...	GET	304	text/css	192 B	< 1 ms	<link rel="style...	
http://az31408.vo.msecnd.net/css/reset....	GET	304	text/css	192 B	< 1 ms	@import	
http://az31408.vo.msecnd.net/css/global/...	GET	304	text/css	192 B	< 1 ms	@import	
http://az31408.vo.msecnd.net/css/global/...	GET	304	text/css	193 B	< 1 ms	@import	
http://az31408.vo.msecnd.net/css/naviga...	GET	304	text/css	193 B	< 1 ms	@import	
http://az31408.vo.msecnd.net/css/naviga...	GET	304	text/css	192 B	< 1 ms	@import	

Demonstration: Using the F12 Developer Tools to Debug JavaScript Code

In this demonstration, you will see how to use the F12 Developer Tools to:

- Set a breakpoint in JavaScript code
- Step through JavaScript code and examine variables

Demonstration: Using the F12 Developer Tools to Profile a Web Application

In this demonstration, you will see how to use the F12 Developer Tools to:

- Examine the network traffic for a web application
- Capture profile data for a web application

Demonstration: Creating Interactive Pages with HTML5 APIs

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

Lab: Creating Interactive Pages with HTML5 APIs

- Exercise 1: Dragging and Dropping Images
- Exercise 2: Incorporating Video
- Exercise 3: Using the Geolocation API to Report the User's Current Location

Logon Information

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

Estimated Time: 60 minutes

Lab Scenario

- The ContosoConf organizers want to highlight the latest HTML5 technologies in order to create an interactive experience for people visiting the conference website. Specifically, the conference organizers have asked you to add the following features to the application:
  - Conference speakers need a way to generate their badges. A web page should be added that enables a speaker to drag-and-drop a profile picture to start creating their badge.
  - A video from a previous conference is available. This video should be available on the Home page.
  - The Location page should be customized to display information about the visitor's current physical location.

Module Review and Takeaways

- Review Question(s)