Internet Programming

Week 7

Instructor:
Daniel Slack, P. Eng
Applied Computer Science
University of Winnipeg

Forms

Web Forms

- Form tag groups elements together
  - Allows for posting, etc.

  `<form/>`
- Several Form objects

  `<input/>`

  `<select/>`

  `<textarea/>`

  `<button/>`

Web Forms

- Input has several types:

`text, number, file, button`

- Common properties
  - Value stores the input from the user
  - Name used to posting the form
- `document.getElementById('someInput').value;`

Web Storage

Aside: Back to Web Service

- Watch out for the dreaded browser cache!
- Most browsers have an interesting property
  - Brower caches result if you retrieve the same URL over and over again
    - You will get the same cached data file back over and over
  - Not what you want when using a web service!
- Easy cure
  - What does the following do?

```
var url = "http://localhost:8080/?callback=updateSales" +
"&random=" + (new Date()).getTime();
```

Browser Storage: 1995-2010

- Browser storage allows us to persistently store data
  - Used in building a web experience
- For example:
  - Shopping carts, user preferences, *etc.*
- Had to use cookies (until recently)

Cookies

- Servers use cookies to personalize the experience
- Cookies are small bits of textual information that a Web server sends to a browser
  - Later, the browser returns cookies unchanged when user visits the same Web site or domain
- Cookies are mainly used for:
  - Identification of users who previously visited given site
  - Short-term tracking (e-commerce sessions)
  - Presenting the site the way the visitor previously customized it
  - Letting identifiable visitors in without their having to reenter a password
  - Focusing advertising

Cookies

- Servers can send cookies with their response to a web page request
  - Send key/value pairs to store on client's machine
- Cookie is sent back next time browser requests the same page (*i.e.* page at the same hostname)
  - Default: browser returns cookies only to the exact same hostname that sent the cookies
    - **Possible to send cookies to other levels in the domain name hierarchy**
    - See, *e.g.* http://en.wikipedia.org/wiki/Domain_name#Domain_name_syntax
  - Servers are **prevented** from setting cookies that apply to hosts outside their domain

Cookies

- By default, the browser returns cookies only to URLs in or below the directory containing the page that sent the cookie
- You may specify more general path, but not more specific one

Cookies

- If your web page is in:

  http://somehost/u2909_050/Lecture06/Cookies/index.html

- You can specify paths

  "/u2909_050/Lecture06/"

  "/u2909_050/"

  "/")

- But not

  "/u2909_050/Lecture07/"

Cookies

- Cookies are not a serious security threat
- Cookies are never interpreted or executed in any way thus cannot be used to insert viruses into your system
- However, cookies may present a threat to privacy
  - How?

Cookies

- Code:

```
var allCookies = document.cookie
//Returns string with semicolon-separated cookies
document.cookie = “newCookie=Stuff”
//Sets or Updates the cookie “newCookie”
```

Cookies

- Code:
- Deleting a cookie?

```
var expiry = ‘Thu, 01 Jan 1970 00:00:01 GMT’
document.cookie = “newCookie=;expires=”+expiry;
```

HTML5 Web Storage

- HTML5 gives a nice and simple JavaScript API for storing persistent key/value pairs
- Persistent storage means data remains even if browser is closed
- Cookies are limited to 4 KB of storage
- Web storage allows 5-10 MB (per domain)
  - Means your app can store data to reduce communication with the server
- Like cookies, page can only store and retrieve data served from the same domain

HTML5 Web Storage

- Approach:
  - A page can store one or more key/value pairs
    - Stored in browser's local storage
    - Every modern browser provides 5MB or more
    - Storage is persistent
  - Browser can then retrieve a value using its key
    - Server still serves the page
    - Client may even send some data in local storage to server
    - Difference: Client is handling details, not the server

HTML5 Web Storage

- Web storage API is available through the localStorage object
- Storing data

```
localStorage.setItem('myKey', 'myValue');
```

  - myKey
    - Key for storing data
    - Must be a string
  - myValue
    - The value stored and associated with myKey
    - You can only store items of type string (wink wink)

HTML5 Web Storage

- Retrieving Data

```
var myVar = localStorage.getItem('myKey');
```

  - myKey
    - Key used to store the data
  - Note, getting values do not remove them from storage

- Testing if supported

```
If (window[“localStorage”]) {
        //Your localStorage code here …
}
```

Storing Integers and Floating Point

```
localStorage.setItem("numitems", 1);
```

- JavaScript casts the above int to a string

```
var numItems =
parseInt(localStorage.getItem("numitems"));
```

- Use parseInt to turn string back to an integer
- Use parseFloat if storing floating point values

Storing Objects

- Sometimes its useful to store an object
- Localstorage only stores Strings
- Use JSON

```
localStorage.setItem(“myObj”, JSON.stringify(dogObj);
```

- To extract:

```
JSON.parse(localStorage.getItem(“myObj”));
```

Associative Array

- The localStorage object can be treated as an associative array
- Storing data

```
localStorage[“myKey”] = “myValue”;
```

- Retrieving data

```
var myVar = localStorage[“myKey”];
```

- Also provides:
  - Property length & method key

```
for (var i=0; i<localStorage.length; i++)
{
        var key = localStorage.key(i);
        var value = localStorage[key];
}
```

Deleting Local Storage

- All browsers contain built-in developer tools to examine local storage
- Remove a single item

```
localStorage.removeItem(key)
```

- Also, the following method deletes all items from local storage
  - At least, the ones from the same domain
  - It will delete all items!!!

```
localStorage.clear()
```

Aside

- CSS Effects
  - box-shadow
    - Adds a drop shadow around an element
  - transition
    - When attributes are changed on an element, this sets how they will change
    - Eg.

```
transition: all 0.5 ease-in
```

    - See https://developer.mozilla.org/en-US/docs/Web/CSS/single-transition-timing-function#Keywords_for_common_timing-functions

Example 01

- A Sticky note application allowing users to see stickies and add new ones
  - Show notes in localStorage and allow addition of new ones
- Features
  - Create a form with an input button
  - Iterate through existing stickies and add them to the DOM
  - Add new stickies to the DOM and localStorage
  - Style stickies to look like real sticky notes

Example 02

- Deleting specific data from local storage
- Uses the following

```
localStorage.remove(key);
```

  - Takes key of an item and removes that item from localStorage

Example 03

- Specify sticky colour using JSON objects

Other Storage Options

- sessionStorage
  - Similar to localStorage
  - Data is expired when page session ends
    - Browser closes
    - New tab/window
- IndexedDB
  - Low-level api to store large amounts of data, including blobs
  - Transactional object based DB

Video

Example 04

- Today's example
  - Create a web TV
    - Use HTML5 video element
    - Put video and canvas together as well
- We'll start right away with the HTML

Video Element

```
<video controls
       autoplay
       src="small.mp4"
       width="480" height="360"
       poster="waterfall.jpg"
       id="video">
</video>
```

- Controls attribute causes player to supply controls for controlling video and audio playback
- Autoplay attribute causes the video to start playback on page load
- Poster refers to a poster image to show when the movie is not playing

More attributes

- preload
  - Used for fine-grained control over how video loads
  - Usually, browser determines how much video to load
    - Based on user's bandwidth and whether autoplay is set
  - Choices:
    - none: None of the video is downloaded
    - metadata: only the video metadata is downloaded
    - auto: browser makes the decision
- loop
  - Automatically restart video after it finishes playing

Video Formats

- A video file contains two parts
  - A video part and an audio part
- Each part is encoded using a specific encoding type
  - To reduce size
  - To allow it to be played back more efficiently
- There are many video formats on the web!
  - No one can agree on the encodings
- Also, the files that hold the video and audio encodings have their own format!
- You will have to supply more than one format if you have a wide spectrum of users!

The Contenders

- WebM container with VP8 Video and Vorbis Audio
  - Pulling ahead as the leader
  - WebM was designed by Google to work with VP8 encoded videos
  - Supported by Firefox, Chrome, and Opera
  - .webm extension
- MP4 container with H.264 Video and AAC Audio
  - H.264 is licensed by the MPEG-LA group
  - There is more than one kind of H.264
  - Supported by Safari and IE9+

The Contenders

- Ogg container with Theora Video and Vorbis Audio
  - Theora is an open source codec
  - .ogv extension
  - Supported by Firefox, Chrome, and Opera

Supporting All Formats

- The <source> element is used to provide a set of videos
  - Each with its own format
  - Browser picks the first one it supports

```
<video id="video" poster="waterfall.jpg" controls widh="480"
       height="360"
       <source src="small.mp4">
       <source src="small.webm">
       <source src="small.ogv">
       <p>Sorry, your browser doesn't support the video
element</p>
</video>
```

Being More Specific

- Browser still has to do some detective work before it can determine if it can play the files you specify
- You can aide the browser by providing the MIME type
  - See, e.g. http://en.wikipedia.org/wiki/Internet_media_type
  - See, e.g. http://wiki.whatwg.org/wiki/Video_type_parameters

Being More Specific

```
<source src="small.ogv" type='video/ogg; codecs="theora, vorbis" '>
```

- src
  - File specified here is actually a container for the actual video
- type
  - Optional attribute (hint for browser)
  - Note the single and double quotes placement
  - video/ogg is the MIME type
  - Theora is the video code and vorbis the audio codec
  - If you don't know the codec parameters, then leave them off

Being More Specific

```
<video id="video" poster="waterfall.jpg" controls width="480" height="360>
   <source src="small.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
   <source src="small.webm" type='video/webm; codecs="vp8, vorbis"'>
   <source src="small.ogv" type='vodeo/ogg; codecs="theora, vorbis"'>
   <p> Sorry, your browser doesn't support the video element </p>
</video>
```

Video API

- The <video> element exposes a rich API
- Here is a summary
  - Properties
    - videoWidth, videoHeight, currentTime, duration, ended, error, loop, muted, paused, readyState, seeking, volume
  - Methods
    - play, pause, load, canPlayType
  - Events
    - play, pause, progress, error, timeupdate, ended, abort, waiting, loadeddata, loadedmetadata, volumechange

Example 05

- Currently:
  - One video up and running
- Goal:
  - Programming schedule that serves up a playlist of videos
- Specifically:
  1. Show a preshow
  2. Show a first feature
  3. Show the featured presentation

Example 05

- Page loads
  - Set up a playlist array
  - Start the first video playing
  - Set up an event handler for when it stops
    - Use to start the next video

Example 06

- How do we decide on video format?
  - We can use the canPlayType method of the video object
  - canPlayType takes a video format
  - Returns a string that represents browser's confidence in its ability to play the video
    - Probably, maybe, or no confidence
  - Passing only the short form (*i.e.* "video/ogg") results in:
    - "" or "maybe"
  - Passing in type + codec results in:
    - "maybe" or "probably"

Example 07

- HTML5-enabled video messaging booth
  - Watch two types of videos
  - User can enhance video using movie effects
    - Old-time western filter
    - Black & white film noir filter
    - Sci-fi alien filter
  - Customer can send their message to a friend
- Note: This example creates its own controls
  - Doesn't use built-in video controls
    - Allows them to look the same regardless of browser

Canvas and the Video Element

- What is the relationship?
  - Doing anything other than video playback requires the canvas
    - e.g. Processing video, custom overlays, displaying multiple videos at once
- The canvas element allows:
  - Processing of video in real time
  - Inspecting the video's characteristics
  - Grabbing data from video frames
  - Altering video data
    - e.g. rotating, scaling, or changing pixels

Canvas and the Video Element

- Video API does not contain effect methods
- Web programmer has to create effects manually
- Requires image/video processing techniques

Video Processing

- In order to add an effect we need to:
  - Get at the video pixels
  - Alter them (to add the effect)
  - Get pixels back to the screen
- Does the video API provide a method to process video before it is displayed?
  - No
  - It only gives us a method to get the pixels
  - Canvas object is used to process and display them

Approach

1. The video player decodes and plays the video
   - Browser decodes the video into a series of frames
   - Each frame is a rectangle of pixels (*i*.e. an image)
2. Video is copied frame by frame into a hidden canvas
   - Each decoded frame is copied into a hidden canvas
   - Acts as a buffer allowing us to process the video
     - For example, turn colour video to black & white
   - We iterate over each pixel and process it
     - Pass each pixel to an effects function
     - Effects function manipulates the RGB value of each pixel

Approach

3. Processed frame is copied to another canvas to be viewed
4. Process is repeated for each frame that is decoded by the video object

Processing Video Using a Scratch Buffer

- Why use two canvases to process and display video?
- The two canvas technique minimizes visual glitches
  - Especially during intensive video and image processing
  - Called using a scratch buffer
- Glitches are minimized by:
  - Processing data in a hidden canvas
  - Copying all data to display canvas in one operation

Example 08

- Adding video effects