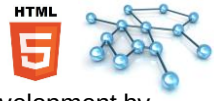
Rich Internet Applications

9. HTML 5 and RIAs

HTML 5



- Standard currently under development by W3C
- Aims to improve the language with support for the latest multimedia
- Keeping it easily readable by humans
- Consistently understood by computers and devices
- Adds many new elements and capabilities to HTML

HTML 5 RIA-friendly features

- Include:
  - Video/Audio
  - Offline
  - Application cache
  - Local storage
  - Geolocation
  - Canvas
  - Drag and drop
- Native browser support for capabilities previously only available with plug-ins

Demos

- Some demos in code download
- Others at http://html5demos.com

Browser support

- Evolving
- Test your browser at http://html5test.com/
- Gives compatibility score
- Chrome current has highest score
- IE9 scores very low
- Use Modernizr.js to test feature compatibility in client code

Video

- Supported formats
  - H.264/MPEG4 (.mp4)
  - Theora (.ogv)
  - WebM
- Place video on page with <video> element
- Browser provides controls for play/pause, volume, etc.
- Configure server for appropriate mime types (can't test withASP.NET dev server)

Offline applications

- Offline application availability particularly important on occasionally connected devices
- Aspects:
  - Ability to detect in client code when online/offline
  - Ability to control the way the browser caches the application pages and resources
  - Local storage, possibly synchronise with server when next online

Offline

- **navigator.online** property
  - boolean
  - false if user selects, for example, Work Offline in Firefox or Airplane mode on mobile device
- **window.online** and **window.offline** events
- http://html5demos.com/offline

Application cache

- Browsers have caching mechanisms but these are generally unreliable
- The Application Cache allows a developer to specify which files the browser should cache and make available to offline users.
- Application will load and work correctly, even if the user presses the refresh button while offline

Cache manifest

- Simple text file that lists the resources the browser should cache for offline access
- Referenced in html element

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

- Need to configure mime type ext/cache-manifest on server

Cache manifest sections

- CACHE:
  - default section for entries
  - files will be explicitly cached after they're downloaded for the first time
- NETWORK:
  - resources that require a connection to the server
  - requests to these resources bypass the cache, even if the user is offline
- FALLBACK:
  - fallback pages if a resource is inaccessible.

Cache status

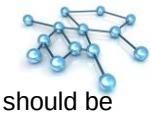
- **window.applicationCache.status**
- Values:
  - UNCACHED = 0
  - IDLE = 1
  - CHECKING = 2
  - DOWNLOADING = 3
  - UPDATEREADY = 4
  - OBSOLETE = 5

Cache events

- cached, checking, downloading, error, noupdate, obsolete, progress, updateReady
- Attach handlers to **window.applicationcache.*event***
- For example

```
window.applicationCache.addEventListener(
    'checking', handleCacheEvent, false);
```

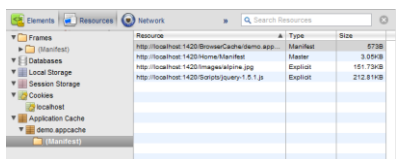

Using the application cache



- Use manifest to tell browser what should be cached
- Use events and status to detect changes to cache
- Note that change a resource on the server is not enough for the browser to know about it – the cache manifest itself may change
- Common to use comment with version number to provide a safe way of refreshing manifest

Application cache demo

- Html5Demo > Manifest
- Note that developer tools in Chrome show cache



Web storage

- Local storage
  - Per domain, available to all scripts in domain that stored data
  - Persists after browser closed
- Session storage
  - Per page per window
  - Removed when window closed
- Both store simple key-value pairs

Web storage vs cookies

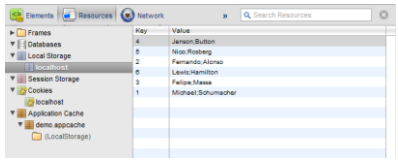
- Capacity
  - Depends on browser, but typically (5MB) much greater than cookies (4kB)
- Web storage is accessible from client only, not sent to server
  - not part of conversation between browser and server

Using local storage

- Html5Demo > LocalStorage and LocalStorageOther
- Store and retrieve with **localStorage.getItem** and **localStorage.setItem**
- Storage events
  - window.addEventListener("storage", handle_storage, false);
  - only fire on windows other than the one which performed the storage action which caused the event

Using local storage

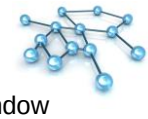
- Chrome allows you to view and edit local storage



Geolocation

- Opt-in service
- Uses GPS, cell tower triangulation or possibly other techniques
- Accuracy will depend on device
- Work on desktop browser but may be very inaccurate
- **navigator.geolocation.getCurrentPosition**
- Demo http://html5demos.com/geo

Canvas



- Allows drawing within browser window
- Demos
- http://html5demos.com/canvas
- http://html5demos.com/canvas-grad

Drag and drop

- Drag and drop elements within page
- Drag and drop files for viewing or XHR upload
- Use JavaScript File API
  - FileReader object provides read access to local files
- Demos
  - http://html5demos.com/drag
  - http://html5demos.com/dnd-upload
  - http://www.html5rocks.com/en/tutorials/file/dndfiles/

What's next?

- The exam!