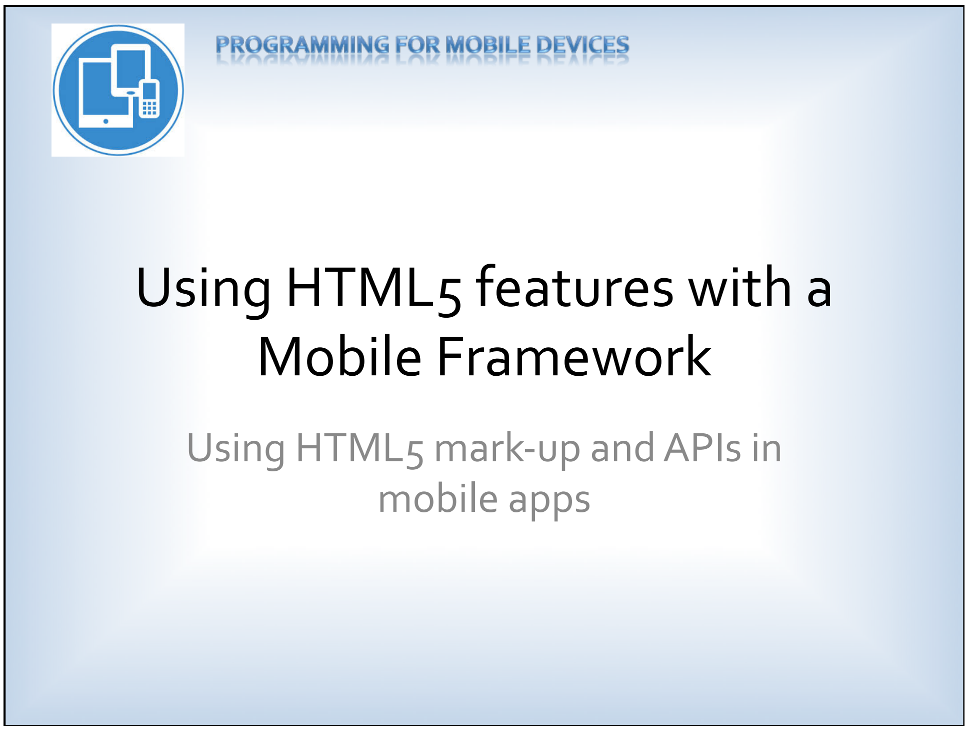
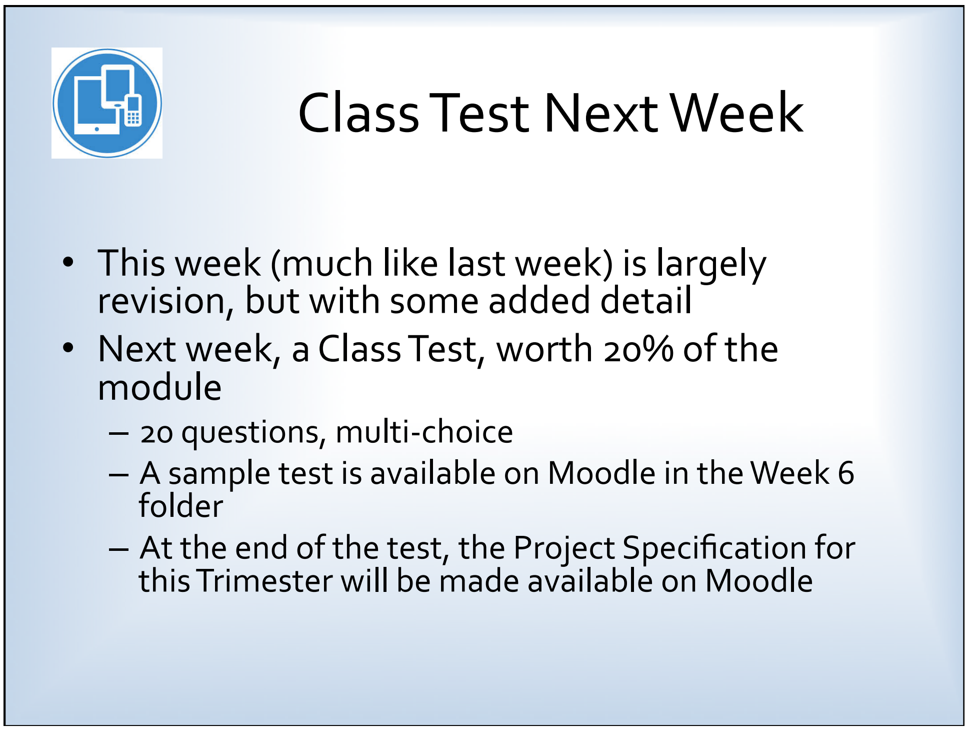


PROGRAMMING FOR MOBILE DEVICES

Using HTML5 features with a Mobile Framework

Using HTML5 mark-up and APIs in mobile apps



Class Test Next Week

- This week (much like last week) is largely revision, but with some added detail
- Next week, a Class Test, worth 20% of the module
  - 20 questions, multi-choice
  - A sample test is available on Moodle in the Week 6 folder
  - At the end of the test, the Project Specification for this Trimester will be made available on Moodle

HTML5 APIs

- HTML5 is now an established standard, so the API specifications are frozen
- HTML6 is already under development as the next standard
  - Proposals have been made for various new features
    - Better semantic mark-up
    - Namespaces (so that new APIs are easier to manage)
    - A data-grid element (easy integration of data sources)
    - Online data storage via cloud services
    - Malware prevention through authentication services
    - Continued roll-out of device APIs
- However, a large number of APIs have emerged in time for the full HTML5 standard
  - See http://caniuse.com/#cats=JS%20API for the level of support among browsers
- Lab 4 covers some of these.

Detecting HTML5 Features

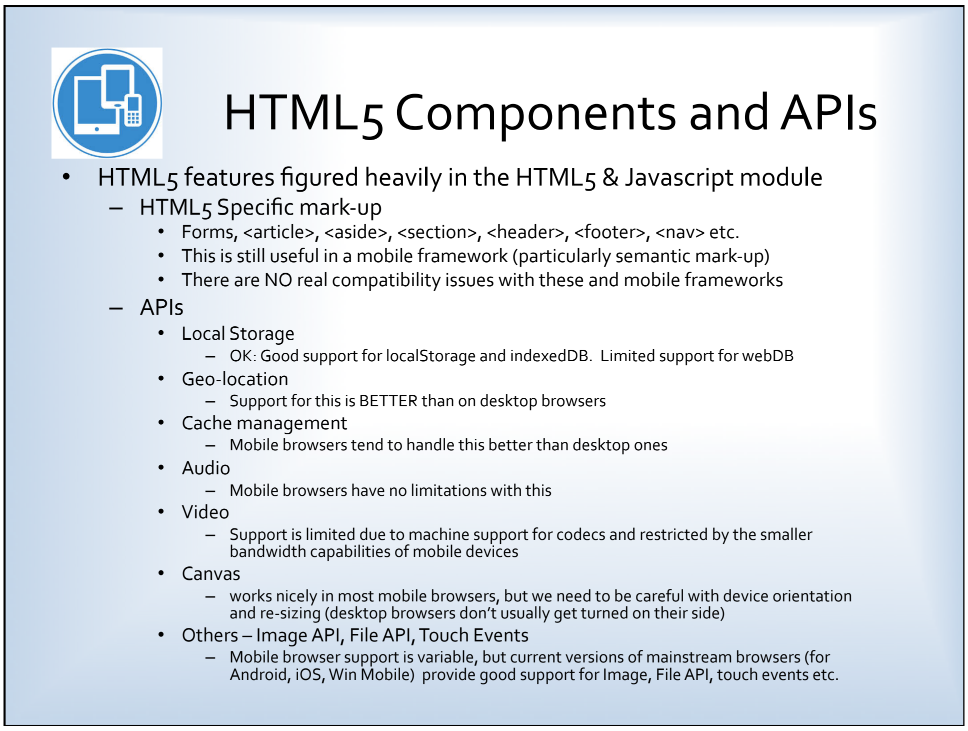
- Since HTML5 implemented to different levels across browsers, , it is best to make sure the target browser supports a specific feature before using it:

```
function supports_canvas() {
    return !!document.createElement('canvas').getContext;
}    // Note - !! coerces getContext to a boolean result
```

- It is now safe to use the feature…

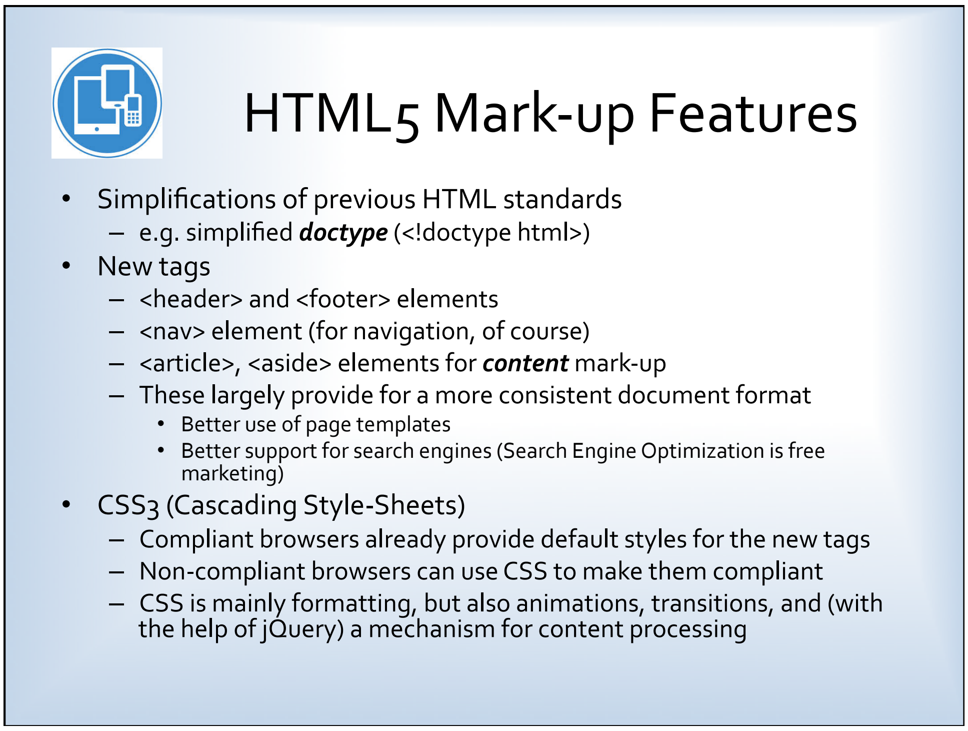
```
function draw_stuff() {
    if(supports_canvas()){
        var theCanvas = document.getElementById('canvas');
        … etc …
    }
}
```

- Similar code can be used to verify other features



HTML5 Components and APIs

- HTML5 features figured heavily in the HTML5 & Javascript module
  - HTML5 Specific mark-up
    - Forms, <article>, <aside>, <section>, <header>, <footer>, <nav> etc.
    - This is still useful in a mobile framework (particularly semantic mark-up)
    - There are NO real compatibility issues with these and mobile frameworks
  - APIs
    - Local Storage
      - OK: Good support for localStorage and indexedDB. Limited support for webDB
    - Geo-location
      - Support for this is BETTER than on desktop browsers
    - Cache management
      - Mobile browsers tend to handle this better than desktop ones
    - Audio
      - Mobile browsers have no limitations with this
    - Video
      - Support is limited due to machine support for codecs and restricted by the smaller bandwidth capabilities of mobile devices
    - Canvas
      - works nicely in most mobile browsers, but we need to be careful with device orientation and re-sizing (desktop browsers don't usually get turned on their side)
    - Others – Image API, File API, Touch Events
      - Mobile browser support is variable, but current versions of mainstream browsers (for Android, iOS, Win Mobile) provide good support for Image, File API, touch events etc.

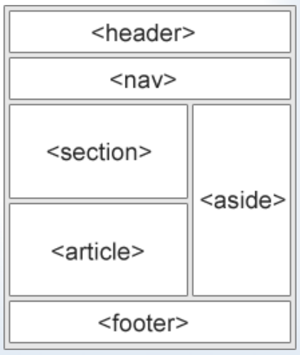


HTML5 Mark-up Features

- Simplifications of previous HTML standards
  - e.g. simplified ***doctype*** (<!doctype html>)
- New tags
  - <header> and <footer> elements
  - <nav> element (for navigation, of course)
  - <article>, <aside> elements for ***content*** mark-up
  - These largely provide for a more consistent document format
    - Better use of page templates
    - Better support for search engines (Search Engine Optimization is free marketing)
- CSS3 (Cascading Style-Sheets)
  - Compliant browsers already provide default styles for the new tags
  - Non-compliant browsers can use CSS to make them compliant
  - CSS is mainly formatting, but also animations, transitions, and (with the help of jQuery) a mechanism for content processing

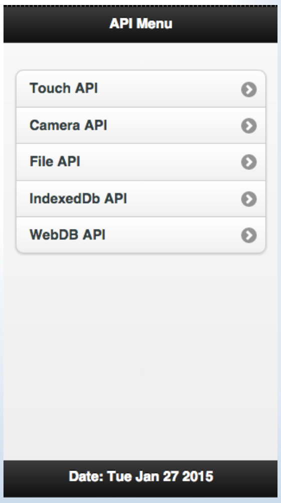
HTML5 Semantic Mark-Up

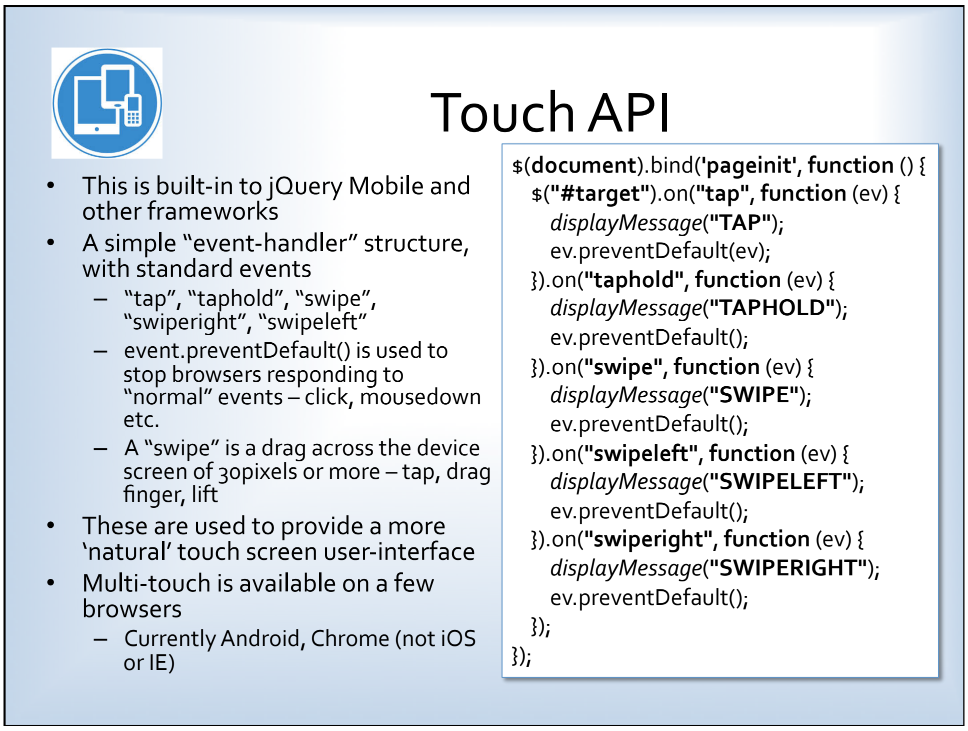
- The new semantic elements provide a template for consistent document mark-up
- Coupled with well designed CSS (e.g. jQM, Foundation, Bootstrap), this can lead to apps having a user-interface that adapts to different device screen sizes (Responsive Web Design)
  - Media queries added to CSS automate the re-structuring of pages
- Good frameworks de-couple the responsive mark-up from processing, so Javascript code can be used purely for core application features (Business processes)
  - Client Application Design is simplified
    - Content providers concentrate on HTML content
    - Developers concentrate on Javascript processing
    - Designers can alter the entire look and feel of applications using CSS



Web APIs

- Demo App
  - Touch:
    - Use of Swipe and Tap events
  - Camera:
    - Access (with permission) to device camera and imaging
  - File API:
    - Able to browse to a device file (requires input from user)
  - IndexedDB, WebDB:
    - Structured data storage on device
- Information for these from various sources
  - Dive into HTML5 – Mark Pilgrim
  - The Rock n Coder - http://therockncoder.blogspot.co.uk/
  - HTML5 Rocks - http://www.html5rocks.com/
- Get demo app from Moodle

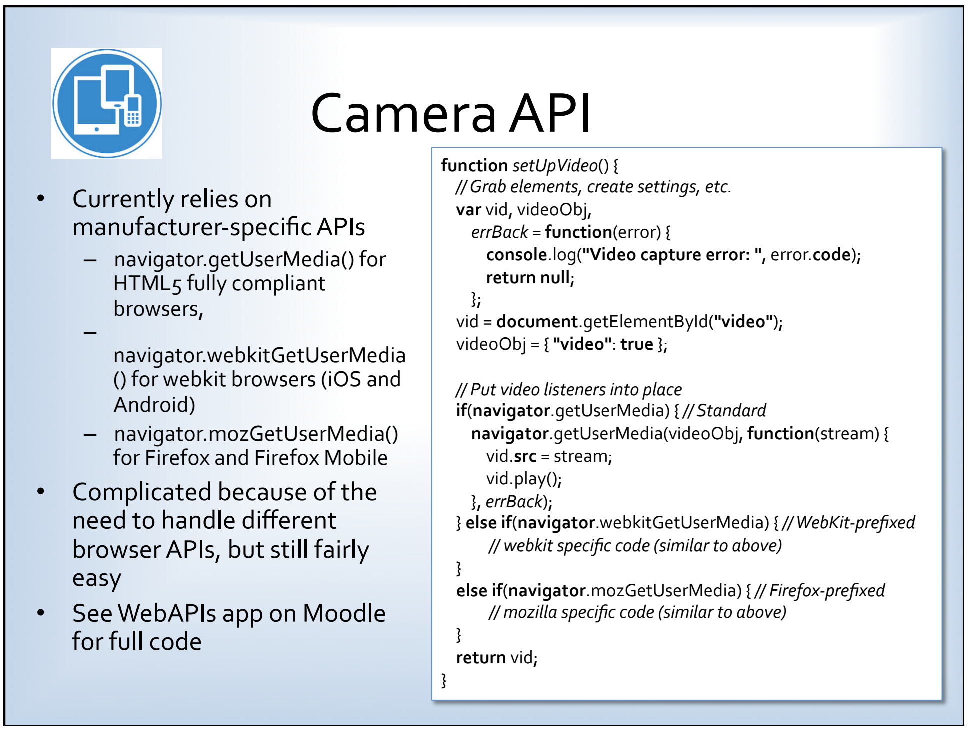




Touch API

- This is built-in to jQuery Mobile and other frameworks
- A simple "event-handler" structure, with standard events
  - "tap", "taphold", "swipe", "swiperight", "swipeleft"
  - event.preventDefault() is used to stop browsers responding to "normal" events – click, mousedown etc.
  - A "swipe" is a drag across the device screen of 30pixels or more – tap, drag finger, lift
- These are used to provide a more 'natural' touch screen user-interface
- Multi-touch is available on a few browsers
  - Currently Android, Chrome (not iOS or IE)

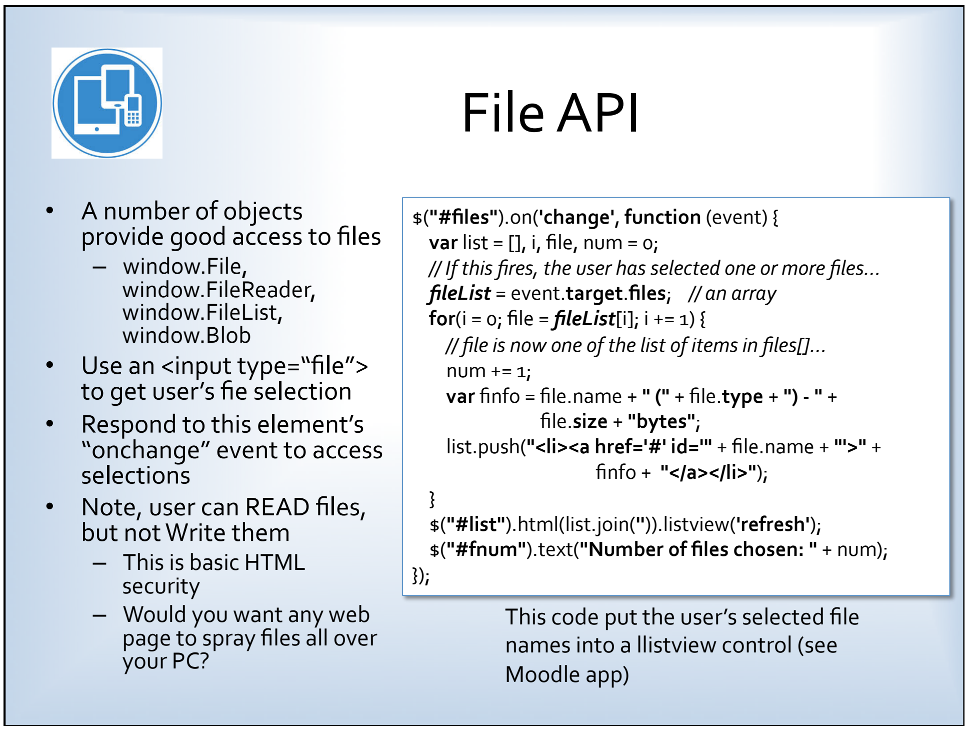
```
$(document).bind('pageinit', function () {
  $("#target").on("tap", function (ev) {
    displayMessage("TAP");
    ev.preventDefault(ev);
  }).on("taphold", function (ev) {
    displayMessage("TAPHOLD");
    ev.preventDefault();
  }).on("swipe", function (ev) {
    displayMessage("SWIPE");
    ev.preventDefault();
  }).on("swipeleft", function (ev) {
    displayMessage("SWIPELEFT");
    ev.preventDefault();
  }).on("swiperight", function (ev) {
    displayMessage("SWIPERIGHT");
    ev.preventDefault();
  });
});
```



Camera API

- Currently relies on manufacturer-specific APIs
  - navigator.getUserMedia() for HTML5 fully compliant browsers,
  - navigator.webkitGetUserMedia() for webkit browsers (iOS and Android)
  - navigator.mozGetUserMedia() for Firefox and Firefox Mobile
- Complicated because of the need to handle different browser APIs, but still fairly easy
- See WebAPIs app on Moodle for full code

```
function setUpVideo() {
  // Grab elements, create settings, etc.
  var vid, videoObj,
    errBack = function(error) {
      console.log("Video capture error: ", error.code);
      return null;
    };
  vid = document.getElementById("video");
  videoObj = { "video": true };

  // Put video listeners into place
  if(navigator.getUserMedia) { // Standard
    navigator.getUserMedia(videoObj, function(stream) {
      vid.src = stream;
      vid.play();
    }, errBack);
  } else if(navigator.webkitGetUserMedia) { // WebKit-prefixed
      // webkit specific code (similar to above)
  }
  else if(navigator.mozGetUserMedia) { // Firefox-prefixed
      // mozilla specific code (similar to above)
  }
  return vid;
}
```

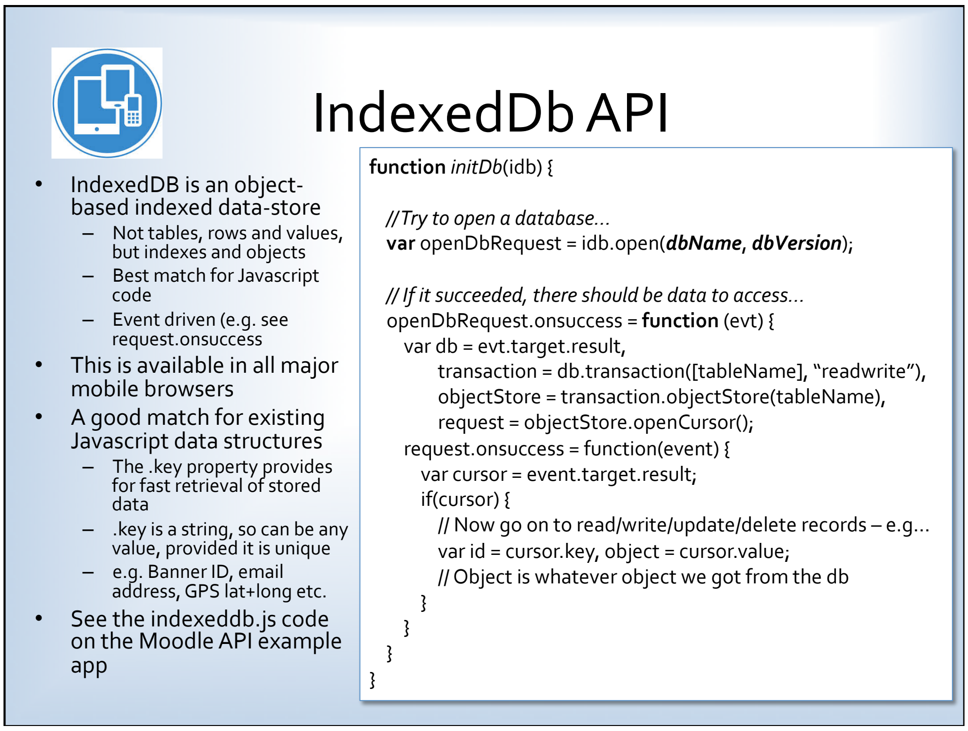


File API

- A number of objects provide good access to files
  - window.File, window.FileReader, window.FileList, window.Blob
- Use an <input type="file"> to get user's fie selection
- Respond to this element's "onchange" event to access selections
- Note, user can READ files, but not Write them
  - This is basic HTML security
  - Would you want any web page to spray files all over your PC?

```
$("#files").on('change', function (event) {
  var list = [], i, file, num = 0;
  // If this fires, the user has selected one or more files...
  fileList = event.target.files;   // an array
  for(i = 0; file = fileList[i]; i += 1) {
    // file is now one of the list of items in files[]...
    num += 1;
    var finfo = file.name + " (" + file.type + ") - " +
              file.size + "bytes";
    list.push("<li><a href='#' id='" + file.name + "'>" +
                      finfo +  "</a></li>");
  }
  $("#list").html(list.join('')).listview('refresh');
  $("#fnum").text("Number of files chosen: " + num);
});
```

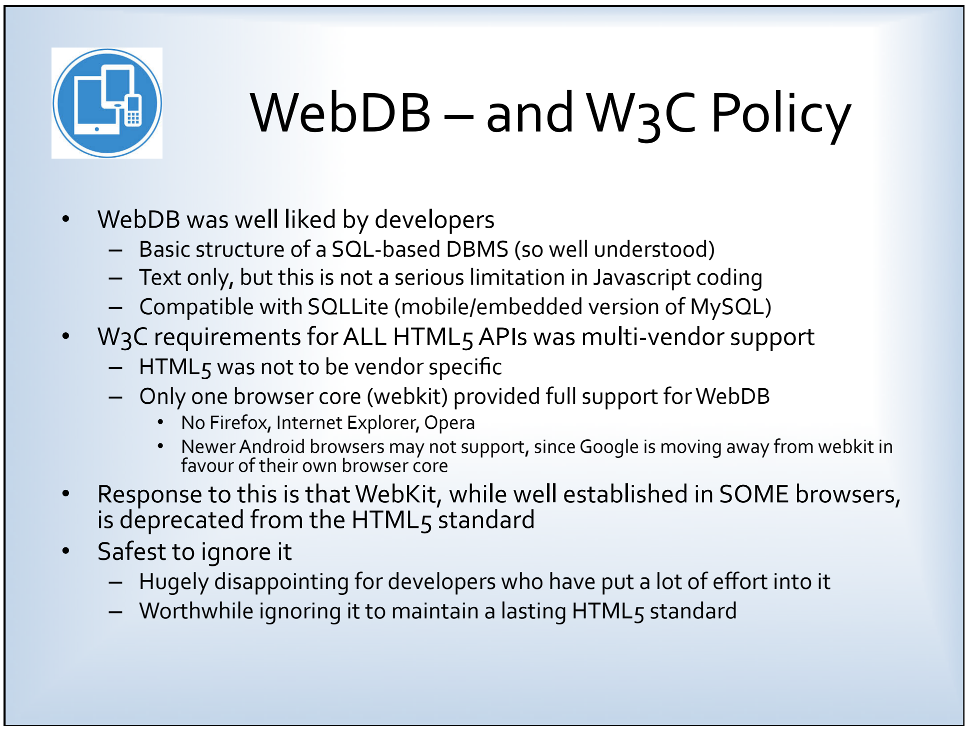
This code put the user's selected file names into a llistview control (see Moodle app)



IndexedDb API

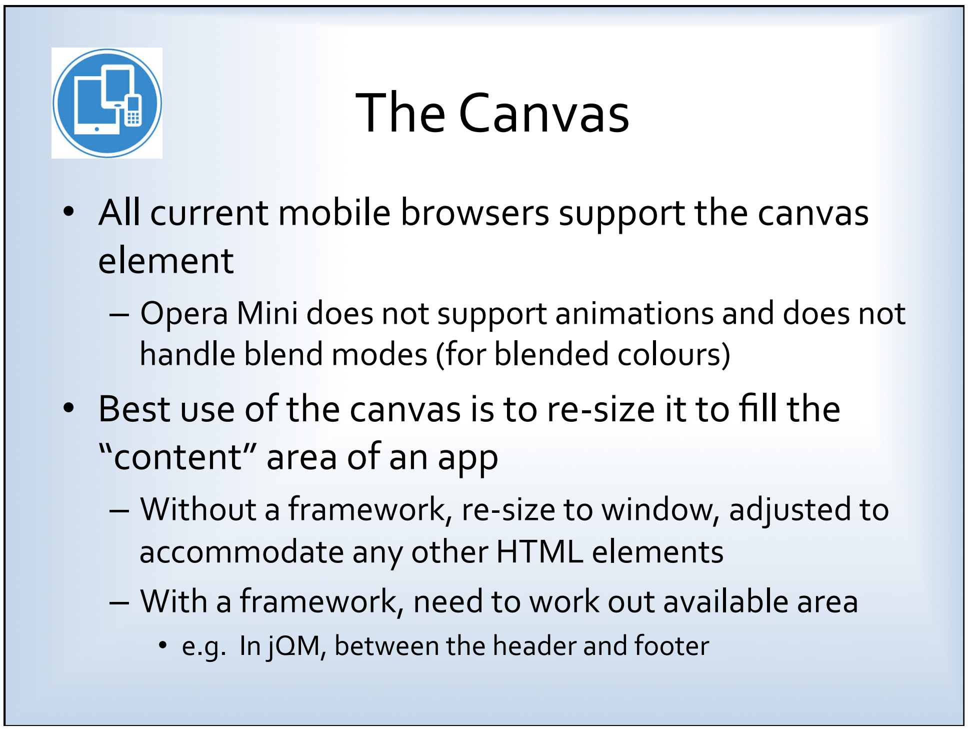
- IndexedDB is an object-based indexed data-store
  - Not tables, rows and values, but indexes and objects
  - Best match for Javascript code
  - Event driven (e.g. see request.onsuccess
- This is available in all major mobile browsers
- A good match for existing Javascript data structures
  - The .key property provides for fast retrieval of stored data
  - .key is a string, so can be any value, provided it is unique
  - e.g. Banner ID, email address, GPS lat+long etc.
- See the indexeddb.js code on the Moodle API example app

```
function initDb(idb) {

  //Try to open a database...
  var openDbRequest = idb.open(dbName, dbVersion);

  // If it succeeded, there should be data to access...
  openDbRequest.onsuccess = function (evt) {
    var db = evt.target.result,
        transaction = db.transaction([tableName], "readwrite"),
        objectStore = transaction.objectStore(tableName),
        request = objectStore.openCursor();
    request.onsuccess = function(event) {
      var cursor = event.target.result;
      if(cursor) {
        // Now go on to read/write/update/delete records – e.g...
        var id = cursor.key, object = cursor.value;
        // Object is whatever object we got from the db
      }
    }
  }
}
```



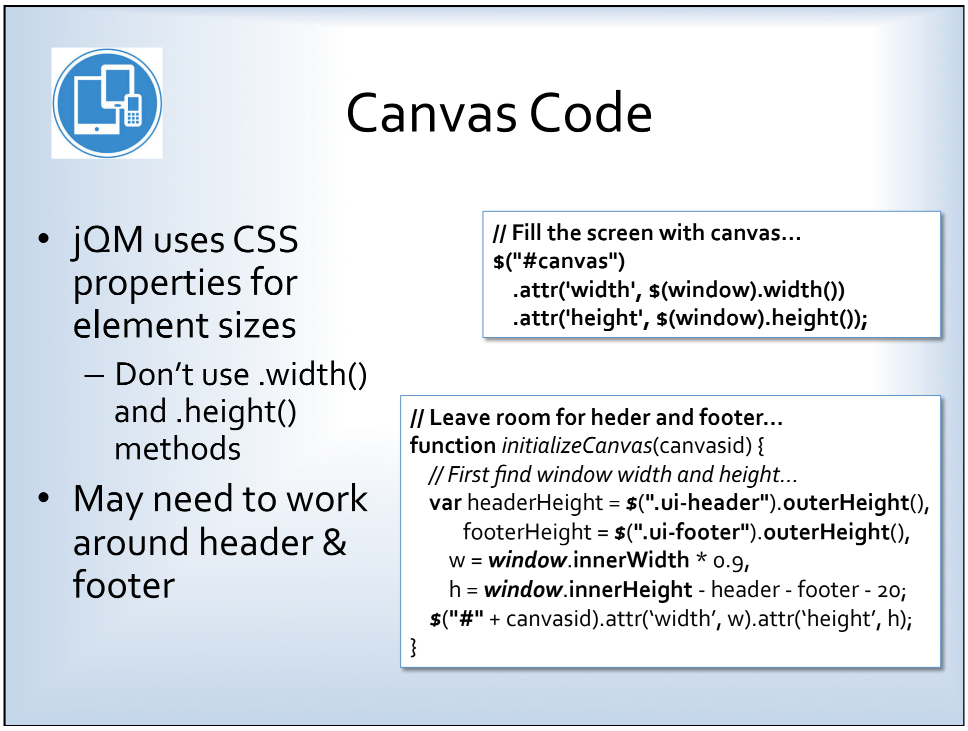
WebDB – and W3C Policy

- WebDB was well liked by developers
  - Basic structure of a SQL-based DBMS (so well understood)
  - Text only, but this is not a serious limitation in Javascript coding
  - Compatible with SQLLite (mobile/embedded version of MySQL)
- W3C requirements for ALL HTML5 APIs was multi-vendor support
  - HTML5 was not to be vendor specific
  - Only one browser core (webkit) provided full support for WebDB
    - No Firefox, Internet Explorer, Opera
    - Newer Android browsers may not support, since Google is moving away from webkit in favour of their own browser core
- Response to this is that WebKit, while well established in SOME browsers, is deprecated from the HTML5 standard
- Safest to ignore it
  - Hugely disappointing for developers who have put a lot of effort into it
  - Worthwhile ignoring it to maintain a lasting HTML5 standard



The Canvas

- All current mobile browsers support the canvas element
  - Opera Mini does not support animations and does not handle blend modes (for blended colours)
- Best use of the canvas is to re-size it to fill the "content" area of an app
  - Without a framework, re-size to window, adjusted to accommodate any other HTML elements
  - With a framework, need to work out available area
    - e.g. In jQM, between the header and footer

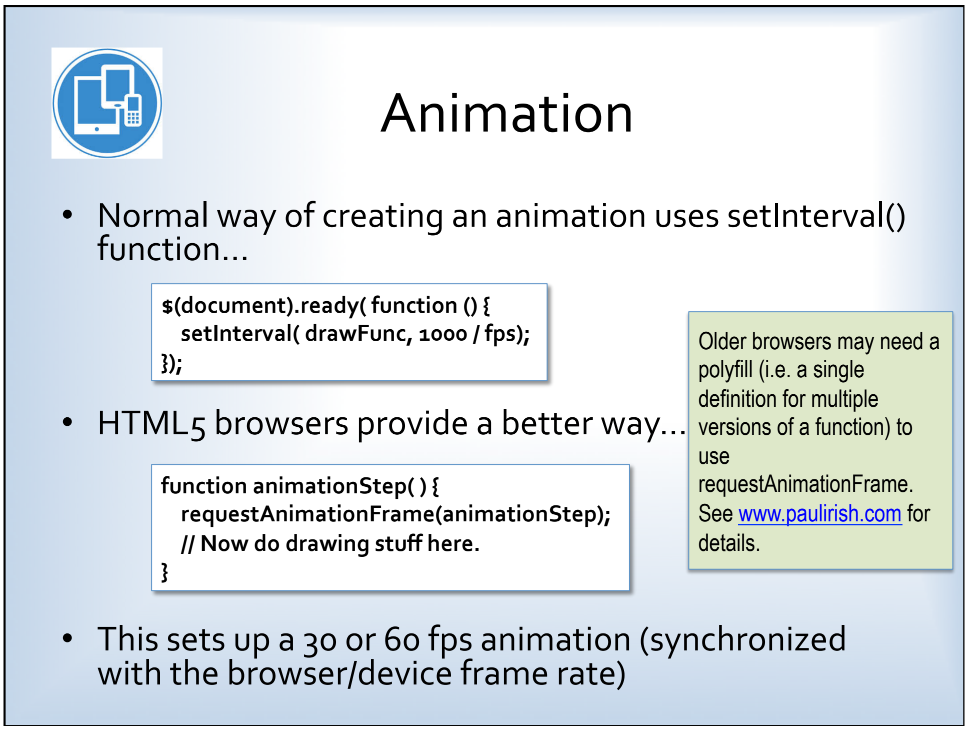


Canvas Code

- jQM uses CSS properties for element sizes
  - Don't use .width() and .height() methods
- May need to work around header & footer

```
// Fill the screen with canvas...
$("#canvas")
  .attr('width', $(window).width())
  .attr('height', $(window).height());
```

```
// Leave room for heder and footer...
function initializeCanvas(canvasid) {
  // First find window width and height...
  var headerHeight = $(".ui-header").outerHeight(),
      footerHeight = $(".ui-footer").outerHeight(),
    w = window.innerWidth * 0.9,
    h = window.innerHeight - header - footer - 20;
  $("#" + canvasid).attr('width', w).attr('height', h);
}
```



Animation

- Normal way of creating an animation uses setInterval() function...

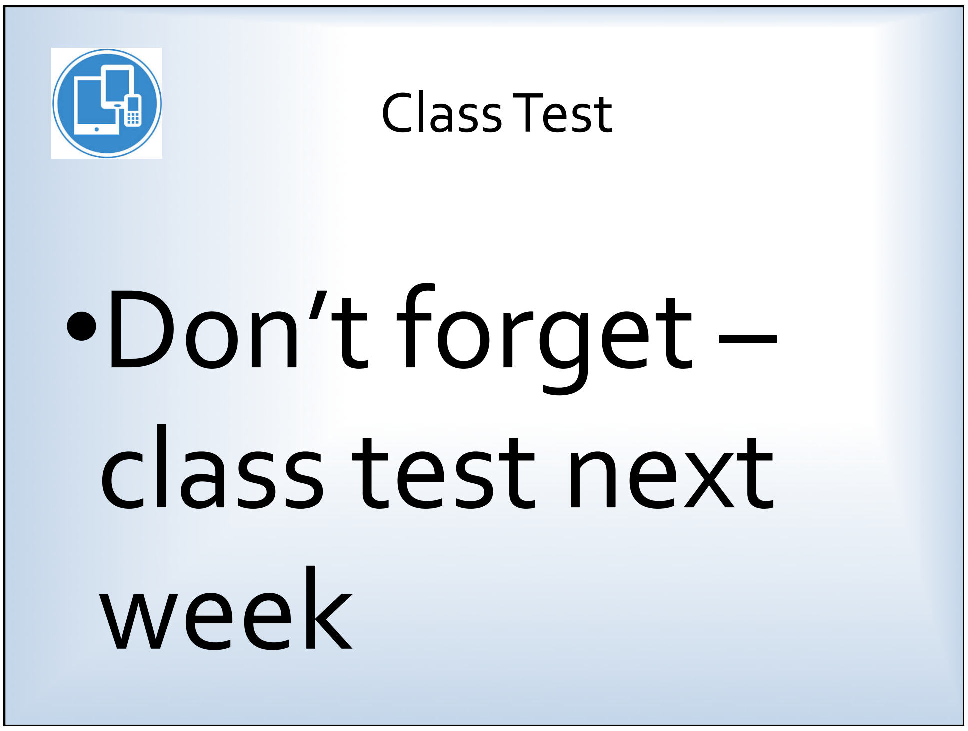
```
$(document).ready( function () {
  setInterval( drawFunc, 1000 / fps);
});
```

- HTML5 browsers provide a better way...

```
function animationStep( ) {
  requestAnimationFrame(animationStep);
  // Now do drawing stuff here.
}
```

Older browsers may need a polyfill (i.e. a single definition for multiple versions of a function) to use requestAnimationFrame. See www.paulirish.com for details.

- This sets up a 30 or 60 fps animation (synchronized with the browser/device frame rate)



Class Test

- Don’t forget – class test next week