TOC - revised

Consistent with most types of application development, there are multiple tools you can use when building mobile web experiences. And, as always, there are advantages and disadvantages to each of the tools. Frameworks, or libraries if you prefer, such as jQuery Mobile and Sencha Touch cater specifically to building web applications for devices. The Project Liike team took a different approach, but it’s important to understand the advantages and disadvantages of the various options.

## 1. Preface

* Provide introduction to the project.
* Clarify the audience.
* List prerequisites.
  + Basic understanding of HTML, CSS, JavaScript
  + Familiarity with ASP.NET MVC useful, but not required

## 2. Choosing between a web and native experience

* Based on existing article (Don)
* Ensure it explains benefits and tradeoffs of each approach.

## 3. Defining the mobile web app experience

* Overall requirements (what defines a mobile web app?)
  + Rich UI
    - Utilizing client features to improve the UX
  + Performance considerations
    - Lightweight
    - Low latency
    - Both improve usability and encourage use
  + Forward thinking coding practices
    - Long-term investment
      * Anticipating growth in mobile usage
      * May need to reach feature parity with desktop
      * May eventually replace the legacy desktop experience (i.e. enhanced to the point where it supports additional contexts such as desktop, TV etc.)
  + Designing for features vs. specific devices
    - Benefits of the approach
      * Broad reach
      * Future friendly
      * Good separation of concerns
    - Historical problems when this approach is not chosen
    - Mobile first/progressive enhancement vs graceful degradation approach to structuring code.
  + Look and feel agnostic
    - No device or platform specific branding

## 4. Choosing devices and levels of support

* Based on existing article (steph)

## 5. Options for building a mobile experience

* No single right answer
  + Decision should depend on your circumstance (budget, timeline, architecture, data structure etc.)
* Proxy based solutions
  + Front-end adaptation and optimization
    - No change to the back-end
  + Pros/cons
* Extending an existing application
  + Modifying front-end markup and functionality to increase cross-browser compatibility
    - Reducing latency and page weight
    - Augmenting with mobile-specific capabilities (e.g. location)
  + Pros/cons
* Designing mobile first
  + Redesigning the application with mobile as primary context
    - Progressively enhancing to suit additional contexts
    - Replace legacy application once feature parity has been reached

## 6. Mobilizing the Mileage Stats application

* High level requirements
  + Experience
    - Add a car
    - Add a Fill-up
  + Technology
    - Support for a wide range of browsers and mobile devices
    - HTML 4.01, CSS 2.n, good form support
* Default application criteria
  + Lowest common denominator, page refresh
* SPA experience criteria
  + Support for JavaScript, JSON, XHR, DOM
* Experience categories
  + Works (default) and Wow (SPA)
  + Enhancements
    - Feature detect and layer additional functionality where appropriate (e.g. location, canvas)
  + Whoops – edge cases browsers to be assessed as project progresses

## 7. Delivering the Default Experience

1. Works Base technology requirement (LCD experience)
   * HTML 4.01, CSS 2.n
   * Standard HTML form input
   * Device and feature detection
     + Profiler to combine server and (any) client knowledge
2. Detecting devices and their features [dev]
   * Existing article (Pablo)
3. Device detection and view switching
   * Existing article (Pablo)
4. Delivering mobile friendly HTML
   * structure
   * HTML5 elements
   * Viewport tag
   * Polyfill for IE
5. Developing mobile friendly forms
   * Select menus for dates
   * Cascading lists with page refresh
   * HTML5 input types
     + Link to fallback in Wow section
6. Delivering mobile friendly images
   * Server-side image sizing and adaptation
   * Image formats
     + base 64 vs sprites vs inline
   * Responsive images
     + Flexible images
     + Additional techniques
       - Links to detailed industry resources
   * Charts
     + Generated and sized server-side
     + Point to canvas-based version in Wow
7. Server-side techniques for composing the view [dev]
   * Intention here is to discuss the approach we took in architecting the server-side code in order to make it friendlier to the needs we had. This is somewhat specific to ASP.NET MVC, and perhaps other MVC-based frameworks.
8. Usability enhancements
   * Pagination
   * Link to the desktop site
     1. Issues we had and final decisions
   * Catch-all for small topics that don’t warrant a page

## 8. Delivering the SPA enhancements

1. Wow base criteria

* Base technology requirements
  + Feature detection
    - JavaScript, HXR, DOM manipulation, JSON
    - hashChange (TBD pending tests)
  + CSS 2.1 presumed
    - CSS3 styles provided as enhancement and will degrade gracefully if unsupported (not always the case but should be applicable in our case)
* Approach for false positives
  + Gracefully degrade where possible
  + Polyfill known edge cases
    - Date input field example

1. Delivering the SPA [dev]

* Navigation, templating, Ajax, caching

1. Using libraries and frameworks

* Pros and cons
* Choosing a library or framework
* Actual usage in Mileage Stats [dev]
  + TBD. Based on final decisions and usage.

1. Delivering a responsive layout
   * Cross reference to ‘mobile first’ CSS for Works
2. Mobile-friendly Form validation [dev]
   * Usage of HTML5 elements and input types
   * Fallback strategies
3. Implementing geolocation [dev]
   * Rational for usage
     + Implementation
4. Delivering enhanced charts using canvas [dev]
   * Canvas implementation
5. Usability enhancements

* Catch-all for small topics that don’t warrant a page
* Pagination
  + Rational for usage

## 9. Testing mobile web experiences

* Based on existing article (Steph)
* Performing automated testing with Visual Studio 2010 [test]