Turning your website into a mobile app
Tips and Tricks from TripAdvisor Mobile Engineering

Stephen Bourdeau
Mobile Team Manager, Engineering
TripAdvisor, LLC
Who is TripAdvisor? And what’s with the owl?

• TripAdvisor is the world’s most popular & largest travel community with..
  • 50 million monthly visitors
  • 60 million reviews and opinions
  • 1 million businesses listed on the site
  • Present in 30 countries and 21 languages

• TripAdvisor Mobile Site and Apps
  • 10 million unique monthly visitors
  • 28 country-specific mobile sites in 20 languages
  • Native apps on a variety of platforms
    • iOS: iPhone, iPad, and iPod touch
    • Android Phones and Tablets
    • Nokia Ovi
    • HP/Palm webOS
Types of Mobile Apps

• Mobile web app
  • Built using HTML, CSS, JavaScript (maybe some tool kits)
  • Pros
    • Quick to launch and update
    • Easier to track metrics
  • Cons
    • Does not have access to low level features such as camera, accelerometer, etc.

• Native app
  • Built using a variety of languages such as Objective-C (iOS), Java (Android), J2ME (BB), etc.
  • Pros
    • Has access to lower level features
  • Cons
    • Higher development cost across multiple platforms
    • In some cases (iOS, Amazon) slower launch/update times due to approval process
    • Tougher to track metrics
A dark corner…

- Then there is a dark corner where native apps are actually web apps in disguise
  
  - HP/Palm webOS apps are built with HTML, CSS, and JS just like a web app but have access to native functionality
  
  - Nokia Ovi does the same thing
  
  - Provides access to low level features via JavaScript interfaces

Hmmm… that sounds like a good idea
What is a Hybrid app?

• Hybrid Apps combine the best of both worlds
  • Develop the basic building blocks of the app in HTML, CSS, JS and display through a web view
  • Build features leveraging those low level APIs natively
  • Quick(er) to launch & update most features
  • Easy(er) to track metrics
  • So our goal should be to create an awesome mobile website
Why do you care?

• If you have an existing website, you can tweak your web pages to provide a decent mobile user experience

• Once you do that it is easy to convert it to a “Hybrid” app and take advantage of other great distribution channels like iTunes and the Android Market

• Native app users are decent people too!
  • They come back more frequently and when they come back, they stay longer
  • They also interact and convert at a higher percentage
  • Native apps increase brand awareness and brand loyalty
How do we “mobilize” a site?
Focus on usability

• Limited screen real-estate
  • Great opportunity to de-clutter your website. What is really important?

• Design for a single task per page

• Holding phone with one hand, where does my thumb land?
Focus on usability (contd.)

• Large buttons rather than hyperlinks – more “app-like”

• Table views – make entire cell clickable and make sure the user knows it can be tapped (see the chevron?)

• 44 px rule: touch targets should be no smaller than 44px x 44px for a user to easily tap it.
  • Based on original iPhone, so scale appropriately
Making a website feel like an app

• Consistency in UI page components (buttons, (non)rounded corners, etc.
  • Better yet, match the UI guidelines for the relevant platform(s)

• Load content via AJAX when possible
  • Full page reloads interrupt the UX

• Track loading progress with spinners
  • Don’t leave user without feedback when loading data, they always assume the worst.

  • Mobile use cases are short and quick (like at a bus stop). Their tolerance for an unresponsive app is short.
**Viewport tag**

- Standard convention for mobile-optimized sites:
  - You can’t pinch, pan left/right, or zoom the page
  - Buttons and font sizes are large enough, why would you need to?

- Requires work on the part of the developer

```html
<meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" name="viewport">
```

- `width=device-width` : if screen is 320px wide, page should be treated as such too.
  - iOS / Mobile Safari otherwise assumes page has 980px of width to work with.

- `user-scalable=no` : pinching/zooming disabled
Other viewport tricks

• Treat webpage like a full-screen app (iOS 2.1 and later only):

  <meta name="apple-mobile-web-app-capable" content="yes">
  <meta name="apple-mobile-web-app-status-bar-style" content="black">

• This is a bit funny since it only works when launched from the home screen and then kicks you over to mobile safari if you click a web link

• Or, just hide the browser status bar:
  • window.scrollTo(0, 1) in onload JS.
  • Note: you may need extra vertical space on the page to allow scrolling

Handling device orientation changes

• Portrait -> Landscape, vice-versa:

```javascript
window.addEventListener('orientationchange',
  eventHandlerFunction, false);
```

• Other width/height changes of UIWebView:

```javascript
window.addEventListener('resize',
  eventHandlerFunction, false);
```
Targeting specific devices

• CSS3 Media Queries
  
  `<link rel="stylesheet" media="handheld" href="mobile.css"/>

  • Modern phones (iOS and android) ignore media="handheld" because the designers think the phones are powerful enough to give a full-site experience. Use carefully.

• Can also do media queries within a stylesheet:

  `<style>
  @media screen and (orientation:portrait) {}
  @media screen and (-webkit-min-device-pixel-ratio: 2) {}
  </style`

• Full media query reference:
  http://www.w3.org/TR/css3-mediaqueries/
Targeting specific devices (contd.)

• JavaScript: navigator.userAgent
  • Can get OS and version numbers

• WURFL
  • [http://wurfl.sourceforge.net](http://wurfl.sourceforge.net)
  • Device description library that can be used to identify a device and its capabilities based on the user agent
  • Helps answer questions such as: is this a mobile device or not? Is it a tablet device?
Touch Events in JavaScript

- touchstart, touchmove, touchend
  - Indicate when a user has touched onscreen element(s)
  - Supported on iOS and Android
  - On iOS, these support multitouch- event.touches gives a list of X & Y touch coordinates.

- gesturestart, gesturechange, gestureend
  - iOS 2.0 and greater only
  - When the user touches the screen with multiple fingers
  - event.scale : how far in or out some pinch gesture went
  - event.rotation : to what extent the user made a rotation gesture- for changing the orientation of a map
  - Convenient for avoiding extensive calculations based on the X & Y touch coordinates for touch{start,move,end}
Effective use of AJAX

• AJAX is *asynchronous* – so if user spawns 2 AJAX requests, the return order is not guaranteed
  • Simple solution: add a sequence # to your requests
  • Discard the response in the response handler if it is not the most recent request

```javascript
var xhrCount = 0;
var makeRequest = function(url, callback) {
  var request = new XMLHttpRequest();
  var thisXhrCount = ++xhrCount;
  request.open('GET', url, true);
  request.onreadystatechange = function() {
    if (this.readyState == 4 && thisXhrCount == xhrCount) {
      callback();
    }
  };
}
```
Focus on Performance

• Great starting point – “High Performance Web Sites” by Steve Souders

• CSS at top, JS at bottom – don’t block downloads while rendering the page

• Minify CSS & JS
  • Less to download, less to cache, less data in memory
Focus on Performance: Requests

• Fewer, larger-sized requests
  • On cellular data connections, latency to bandwidth ratio is higher than on cable/DSL connections
  • Fewer, larger-sized requests/responses make sense
• triadvisor.com mobile homepage:
  • 6 GET requests for a total of 46.9 KB
• Points of comparison:
  • m.wikipedia.com has 112 KB with 9 requests
  • cnn.com mobile has 159.5 KB with 16 requests
Focus on Performance: Requests (contd.)

• 3G simulation: 122 kbps bandwidth, 600 ms latency, 1400 MTU

• 4G simulation: 183 kbps bandwidth, 150 ms latency, 1400 MTU
Image Spriting

• Use one image file containing multiple images, CSS to show only a specific portion

• Improves page load performance
  • Single image download that can be cached
  • Removes overhead of round trip request time & header data from multiple image GET requests
SVG images

- SVG – vector-based image format
  - Much smaller size than PNG, JPG
  - Scale well to look good at any screen density
    - PNGs – need two different sizes, one for iPhone 4 Retina display and iPhone 3GS
- Tradeoffs
  - SVGs require more processing to render
  - Support is somewhat limited
    - Android Honeycomb (3.0) and higher
    - iOS 4.2 and higher
- How to “sprite” them: define them all inline in one CSS file:

  background-image:url(data:image/svg+xml,%3Csvg
  %20width%3D%22286px%22%20height%3D%22190px%22%3E%3C/svg%3E...);
Focus on Performance: Event Handlers

• Inline event handlers vs addEventListener
  • Inline event handlers make the size of each page load higher due to extra page size (not cached)
  
• But, window.onload callback processes faster when fewer addEventListener calls are made at onload time.

```html
<div onclick="alert('clicked');"></div>

<div id="mydiv"></div>
<script>
    window.onload = function() {
        document.getElementById('mydiv').addEventListener('click', function(e) {
            alert('clicked');
        });
    }
</script>
```
Minimize page reflows and repaints

• Repaint: changes are made to an element (via CSS, for example) that change visibility but do not affect layout.
  • e.g. visibility, background-color, color

• Reflow: any change that affects size/placement of elements
  • Changing font size, margin, padding, reordering DOM elements, CSS display property changes
  • More expensive than repaints

• On mobile where devices have less processing power, the performance payoff can be large for minimizing reflows/repaints
Reflow minimization examples

- http://ejohn.org/blog/dom-documentfragments/

Other fun stuff

• Set your home screen icon when a user saves a bookmark:
  • iOS and Android 2.1+:
    <link rel="apple-touch-icon" href="your_image_here.png"/>
  • Android 1.5 and 1.6:
    <link rel="apple-touch-icon-precomposed"
    href="your_image_here.png"/>

• Other helpful links & references
  • “Responsive web design” by Ethan Marcotte
  • “Mobile First” by Luke Wroblewski
  • http://www.html5rocks.com/en/mobile/mobifying.htm
Questions?
TripAdvisor Engineering: This is how we roll…

- Weekly development cycle - live site gets update weekly

- Daily patches for “need to get out today” features/fixes

- Over 50 concurrent projects being worked on at a time by ~100 engineers, ~25 get deployed each week

- Engineers work end to end - Design, Code, Test, Monitor. You design something, you code it. You code something, you test it.

- Engineers work across entire stack - HTML, CSS, JS, Java, services, DB, scripting,… If you do not know something, you learn it.

- Engineers swap between groups for weeks/months to get exposure to different product areas and distribute knowledge and culture
Lots of interesting projects across all product areas in the pipeline (what could you do with 1.5 billion geographic place pins from our Facebook users?)

Kick butt developer hardware
- Latest Macbook Pro’s with solid state drives
- 30” monitors
- Quad core i7-3.4GHz dev boxes with 16GB memory

Just recently spun off from Expedia (NASDAQ: EXPE) and is now a publicly traded company (NASDAQ: TRIP)

Looking for software developers of all levels and skills (interns and win-terns too!)

Check out our coding puzzles @
http://www.tripadvisor.com/careers/puzzles