Mobile Performance: for excellent User Experience

Suyash Joshi
@suyashcjoshi
Mobile UX Developer
A quick audience survey...
Overview of Presentation

1\textsuperscript{st} half:

Mobile Web Performance Optimization (WPO) practices for TIZEN Mobile Web Framework developers

2\textsuperscript{nd} half:

Examples and inspiration for Mobile Perceived Performance for Designers and Developers
TIZEN: Being developed using open source web technologies

www.tizen.org

HTML5 + CSS3 + JavaScript

The Apache Software Foundation
http://www.apache.org/

THE LINUX FOUNDATION
## TIZEN: A True Hybrid Mobile Platform

<table>
<thead>
<tr>
<th>User Interface</th>
<th>Hybrid App</th>
<th>Mobile Web App</th>
<th>Native Mobile App</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Webview (pseudo-browser)</td>
<td>Mobile Browser</td>
<td>Native platform widgets and components</td>
</tr>
<tr>
<td></td>
<td>• HTML as a 1st class citizen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technical stack</td>
<td>Web + Native (e.g. Cordova)</td>
<td>Web only</td>
<td>Native only</td>
</tr>
<tr>
<td>Programming language</td>
<td>HTML5, CSS3, JavaScript</td>
<td>HTML5, CSS3, JavaScript (latest)</td>
<td>C, C++, Objective C, Swift, Java, C#, Visual Basic etc</td>
</tr>
</tbody>
</table>
Why care about Mobile Performance?

Because it affects your **bottom line** (conversion rate)

Because it affects your **SEO** performance

Because it affects your overall app **UX**

"It will make you a better Software Developer and your boss will like you even more!"

Source: KissMetrics – Infograph
Mobile Web Performance: The fundamentals
WPO Basics: We need to do more for mobile

High Performance Websites

1. Make fewer HTTP requests
2. Use CDN
3. Add expires header
4. Gzip Components
5. Put stylesheets at the top
6. Put scripts at the bottom
7. Avoid CSS expressions
8. Make JS and CSS external
9. Reduce DNS lookups
10. Minify JS
11. Avoid redirects
12. Remove duplicate scripts
13. Configure Etags
14. Make Ajax cacheable
15. Sharding domains

Pre-requisite!
Mobile Perf: User Expectations are High!

'85% of mobile users expect mobile pages to load fast or faster than web pages'

- Strangeloop Networks
LATENCY VERSUS BANDWIDTH

In the following graphs, shorter bars signify better performance:

Above a certain level, increasing bandwidth had little impact on improving application performance.

Reducing latency had significant impact on improving application performance.

(Source: Google)
HTML5 API for TIZEN: Network Information API

<table>
<thead>
<tr>
<th>Data service settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honour background data</td>
</tr>
<tr>
<td>no</td>
</tr>
<tr>
<td>Sync over 4G</td>
</tr>
<tr>
<td>✅</td>
</tr>
<tr>
<td>Sync over 3G</td>
</tr>
<tr>
<td>✅</td>
</tr>
<tr>
<td>Sync over CDMA</td>
</tr>
<tr>
<td>✅</td>
</tr>
<tr>
<td>Sync over EDGE</td>
</tr>
<tr>
<td>✅</td>
</tr>
<tr>
<td>Sync over GPRS</td>
</tr>
<tr>
<td>✅</td>
</tr>
<tr>
<td>Sync over WiFi</td>
</tr>
<tr>
<td>✅</td>
</tr>
<tr>
<td>Sync over Bluetooth</td>
</tr>
<tr>
<td>✅</td>
</tr>
</tbody>
</table>

3G in use, charges may apply
You are responsible for your data consumption charges.

Cancel  OK
HTML5 API for TIZEN: Network Information API

```javascript
var connection = navigator.connection || navigator.mozConnection || navigator.webkitConnection;
var type = connection.type;

function updateConnectionStatus() {
    alert("Connection type is change from " + type + " to " + connection.type);
}

connection.addEventListener('typechange', updateConnectionStatus);
```
HTML5 API for TIZEN: Browser State API

- Helpful for Single Page Application Architecture
- No more # in the urls and ugly state management for AJAX based content
- Live Demo: http://html5doctor.com/demos/history/

```javascript
var currentState = history.state;

history.replaceState() // update on partial page load

// Store the initial content so we can revisit it later

history.replaceState(
  {
    content: contentEl.textContent,
    photo: photoEl.src
  },

document.title, document.location.href);
```
//startSimulation and pauseSimulation defined elsewhere
function handleVisibilityChange() {
    if (document.hidden) {
        pauseSimulation();
    } else {
        startSimulation();
    }
}

document.addEventListener("visibilitychange", handleVisibilityChange, false);
HTML5 API for TIZEN: Page Visibility API

`document.hidden`
Returns `true` if the page is in a state considered to be hidden to the user, and `false` otherwise.

`document.visibilityState`
Returns a string denoting the visibility state of the document. Possible values:

- `visible`: the page content may be at least partially visible. In practice this means that the page is the foreground tab of a non-minimized window.
- `hidden`: the page content is not visible to the user. In practice this means that the document is either a background tab or part of a minimized window, or the OS screen lock is active.
- `prerender`: the page content is being prerendered and is not visible to the user (considered hidden for purposes of `document.hidden`). The document may start in this state, but will never transition to it from another value. Note: browser support is optional.
- `unloaded`: the page is being unloaded from memory. Note: browser support is optional.
Don't kill the battery!

It’s not just about speed

http://www.flickr.com/photos/fastjack/2943793818
HTML5 API for TIZEN: Battery Status API

```javascript
var battery = navigator.battery || navigator.mozBattery || navigator.webkitBattery;

function updateBatteryStatus() {
  console.log("Battery status: "+ battery.level * 100 + " %");

  if (battery.charging) {
    console.log("Battery is charging");
  }
}

battery.addEventListener("chargingchange", updateBatteryStatus);
battery.addEventListener("levelchange", updateBatteryStatus);
updateBatteryStatus();
```
HTML5 API for TIZEN: Web Workers (partial)

```javascript
// Create a new worker object
var worker = new Worker('worker.js');

// Send a message to start the worker and pass a variable to it
var info = 'Web Workers';
worker.postMessage(info);

// Receive a message from the worker
worker.onmessage = function (event) {
    // Do something
    alert(event.data);
};
```
Faster JS perf: Write your own JS library

=> Avoid Bulky Frameworks: Less is More

Parsing JS can take 1ms per KB (uncompressed) so you can do the math!

Remember DOM is slow
Faster JS perf: Execute code in 'async'

Sync scripts block the parser...

Sync script will block the rendering of your page:

```html
<script type="text/javascript"
    src="https://apis.google.com/js/plusone.js"></script>
```

Async script will not block the rendering of your page:

```javascript
(function () {
    var po = document.createElement('script'); po.type = 'text/javascript';
    po.async = true; po.src = 'https://apis.google.com/js/plusone.js';
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(po, s);
})();
```

Source: Patrick Meenan's talk on Mobile Web Performance
## Faster DOM perf: Avoid repaint's and reflow's

<table>
<thead>
<tr>
<th>RECORDS</th>
<th>625 ms</th>
<th>630 ms</th>
<th>635 ms</th>
<th>640 ms</th>
<th>645 ms</th>
<th>650 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event (keydown)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove Timer (/Applic...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install Timer (/Applic...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recalculate Style (/Applic...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layout (/Applications/...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove Timer (/Applic...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install Timer (/Applic...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recalculate Style \times 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layout (/Applications/...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scroll</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event (scroll)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paint (1215 \times 23)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paint Setup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paint (1240 \times 504)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Composite Layers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **two repaints**
- **one reflow**
Images : Reduce bytes & HTTP Requests

Source: HTTP Achieve – Mobile Trends '2013
Images : Reduce bytes & HTTP Requests

Don't:

- Load the desktop image on mobile device and try to scale it
- Detect UA and request for different image sizes at arbitrary times

Do:

- Prefer to use Scalable Vector Graphics for your visual assets
- Create SVG/PNG sprite sheets, font-icons for smaller graphics
- Lazy Load images (png, webP etc) if needed
- Compress images losslessly before sending down to client using tool like ImageOptim
A case for Perceived Performance: bridging the gap between Native and HTML5
User's Perception is their Reality

Source: http://www.w3.org/TR/di-princ/
Make it Snappy: Touch Events & :active states

Using Touch Event

Using Click Event
Make it Snappy: Touch Events & :active states

Add Touch State to your buttons using DOM events for touch devices:

```javascript
document.addEventListener("touchstart", function(){}, true)
```

Remove tap highlights using css:

```css
-webkit-tap-highlight-color: rgba(0,0,0,0);
```
CSS3 on the GPU: momentum scrolling w/o JS

No Hardware scrolling

With Hardware scrolling
**CSS3 on the GPU : momentum scrolling w/o JS**

**DO:**
- Use CSS3 overflow scrolling property to get native momentum scrolling
  - `webkit-overflow-scrolling: touch;`

**DON'T:**
- Use JavaScript library such as iScroll or write your own scrolling solution
CSS3 over JS: animations & transformations

Animations Should Move at 60fps: Each frame should take no longer than 16ms to complete

Everything Else Should Respond $\leq 100$ms
Don't forget the Loading Indicator (wait times between 100ms - 250ms)

No Loading Indicator

With Loading Indicator
Use more Gestures – for richer UX

Gesture #1. Side-to-Side Swiping
Gesture #2. Pull-to-Refresh
Gesture #3. Long Press

-webkit-touch-callout: none;

Gesture #4. Pinch-Zoom
Tools and Resources for performance

1. **TIZEN W3C/HTML5 API Reference:**
   https://developer.tizen.org/dev-guide/2.2.1/org.tizen.web.w3c.apireference/w3c_api.html#visibility

2. **Performance Measurement:** webpagetest.org

3. **Page Speed Insights with UX:**

4. **Learn about HTML, CSS, JS from MDN:** developer.mozilla.com

4. **HTML5 Mobile API's Matrix:** http://mobilehtml5.org/

5. **Check for CSS3 Property support:** www.caniuse.com
Thank you

I'd love your feedback, comments, questions!

Suyash Joshi
Mobile UX Developer
suyash@suyashjoshi
Twitter: @suyashcjoshi