Testing and Debugging
Best Practices Guide

Version: 2013.11.20

This document includes best practices around testing and debugging of hybrid applications. Details on each checklist item are discussed later in the document.

Checklist

Testing

- ❏ Use Test Driven Development (TDD) �
- ❏ Standardize on a single unit testing framework �
- ❏ Integrate performance tests into your CI �

Debugging

- ❏ Know your browser’s testing capabilities �
- ❏ Use a dedicated development browser �
- ❏ Disable browser caching �
- ❏ Disable security for cross-domain ajax �

Discussions

Testing

Use Test Driven Development (TDD)

Test driven development (TDD) is a process where you define your core test objectives first, and then implement code that satisfies the tests. The initial tests are typically driven by the business requirements. By writing the tests to meet requirements, gaps, limitations, and new concerns are found much earlier. Generally, the code is written first, then tests are created, and finally it is all checked against the initial requirements. This is far too late in the cycle to ensure comprehensive coverage.

References:

- Unit Testing Best Practices in AngularJS - Excellent writeup on proper TDD and unit test development

Standardize on a single unit testing framework

There are many unit testing frameworks (UTFs), and they all solve the same problem and function in similar ways. Some are better suited for various toolkits, while others are more generic in nature. We cannot recommend a single unit testing framework, but we list several of the more common choices.

- The Intern - SitePen's new testing framework.
- DOH - Dojo's Objective Harness, obviously a good choice for Dojo development.
- QUnit - The UTF used by and for many JQuery projects.
- Jasmine - A clean, simple and robust choice
- Mocha - A popular choice for NodeJS projects
- JSUnit - A port of the original JUnit for JavaScript

Integrate performance tests into your CI

Define performance tests at the start of your project, for example:

- Application startup time not to exceed 500ms.
- View loading time after navigation not to exceed 300ms.
- View redisplay time after orientation change not to exceed 300ms.

Automate collection of these types of metrics from the start of the project, when you do your first checkins as performance tests that execute with every subsequent checkin (using grunt and your favorite test framework for example).

Make sure you run these tests across all the devices you plan to support, on a continuous basis. This will help you catch performance problems on a particular make/model or O/S version that may require alternate fallback designs, for example native implementation of a particular view rather than web.

When you exceed these metrics and your performance tests fail, do the necessary optimization at that time.

Debugging

Know your browser’s testing capabilities

The browser is the ultimate testing platform for web applications, including hybrid apps. It is critically important to truly understand how to use the tools available. All modern browsers have tools either built-in or available as a plugins.

Use a dedicated development browser

Use a dedicated "development" browser, that is different than your day-to-day browser. I use Chrome for my normal browser, and Chrome Canary as my development browser. This keeps a clean separation of concerns between the environments. My dev browser has its own set of bookmarks specific to the projects I am working on. It also has custom startup options to disable Application Cache and Cross-Domain Security. Finally, It has a full compliment of development specific plugins.

While your choice of preferred plugins is likely different, these are the ones that I have installed.

- Accessibility Developer Tools - For analyzing A11Y compliance
- ADB - Android remote debugging
- ChromeVox - Screen reader for A11Y compliance
- ColorPick Eyedropper - Identifying color values
- Edit This Cookie - Cookie management
- IBM Rational Functional Tester - Comprehensive testing web apps
- JSONView - For structured viewing of JSON data
- PageSpeed Insights - Performance analysis
- Postman - REST client
- Ripple Emulator - BlackBerry testing
- Web Developer - General set of testing tools
- Web Performance - Performance analysis
- XHR Poster - REST client
- YSlow - Performance analysis

Disable Browser Caching

Browser caching can be troublesome when testing web applications. There are actually two caches that should be disabled. First is the normal cache that allows the browser to use a cached version of a resource instead of pulling the resource from the remote server. The second is the Application Cache (appcache), which permits a web app to provide a manifest of the app's resources, and the browser can determine if it already has valid copies thus negating any remote calls. Both of these features can dramatically improve browser performance, but can cause stale resources during development. For development testing, on our dedicated development browser, we want to disable both of these caches.

Chrome browsers:

1. Developer Tools (Ctrl-Shift-I) > Settings (the little gear icon at bottom right) > Check the *Disable cache* box
2. Disable app caching by passing the ***--disable-application-cache*** command line arguments (starts with **2** hyphens) when starting Chrome as follows:
   - Windows: Right-click the Chrome shortcut, select *Properties.* In the *Target* field append the above command line argument
   - OSX: Follow these steps for setting up a custom chrome launcher (based on this article)

Disable security for cross-domain ajax

Note: This tip is for your dedicated development browser only! Disabling browser security should be an obvious risk for your normal browser.

Hybrid application running under Cordova permit making AJAX calls to 3rd party hosts. But, under normal desktop browsers, there is a Same host origin restriction that restricts AJAX calls to the originating host that served the main HTML page. In order to test hybrid targeted apps using 3rd party AJAX resources, you'll need to disable this security restriction.

Disabling the AJAX Same origin restriction for AJAX involves starting the browser with the "`–disable-web-security`" argument. Use the tip above for details on passing arguments to the browser.