Mastering Mobile Web with 8 Key Rules

Introduction

When it comes to mobile web design and testing, mobility plays by a far different set of rules than the desktops of years past. Today we are challenged by the variety and multitude of mobile browsers, devices, OS types and OS versions at consumers' disposal. At the same time, new peripherals like GPS, accelerometers, cameras, microphones and watches make app environments much more complex. Simply put: the rules of mobility are not easy to master.

Today, many enterprises are looking to leave their desktop-focused strategies behind. However, as devices rapidly grow in capability, variations in performance and memory - especially when contrasted with consumer desktops – become factors that make elusive the dream of one set of HTML and page designs that can equally service all desktops and all mobile devices.

For example, Facebook's experience with mobile web several years ago taught the industry that native Android and iOS apps still have a role to play. Although devices have since improved in power and mobile browsers have improved in capability, we still find plenty of older devices and earlier versions of operating systems and browsers in the market. The bottom line is some of these older platforms can't handle the size or complexity of many websites unless the sites are re-designed with mobile use in mind.

To help combat these challenges, here are 8 strategies we recommend...

Find the Common Denominator

There are more external conditions, such as poor signal and slow load times, in mobile than desktop.

While we can hope for the best experience, it's smart to expect users to encounter a website in less than optimal conditions. Face this challenge by testing websites in all the environments where people might experience it.

Likewise, testing devices in adverse operating conditions should always be part of the process. Make sure to test on different networks and when the device battery is low. Encountering untested conditions may cause a carefully-crafted website to crash or simply fail to load. Low power reserves may cause the operating system to throttle resources and can dramatically affect the device's ability to handle complex websites, graphics, and animations.

A crash or failure to load is often a result of the website using too much RAM, the battery being too low to fully power the CPU, or the presence of too many graphics, promotions or animations.

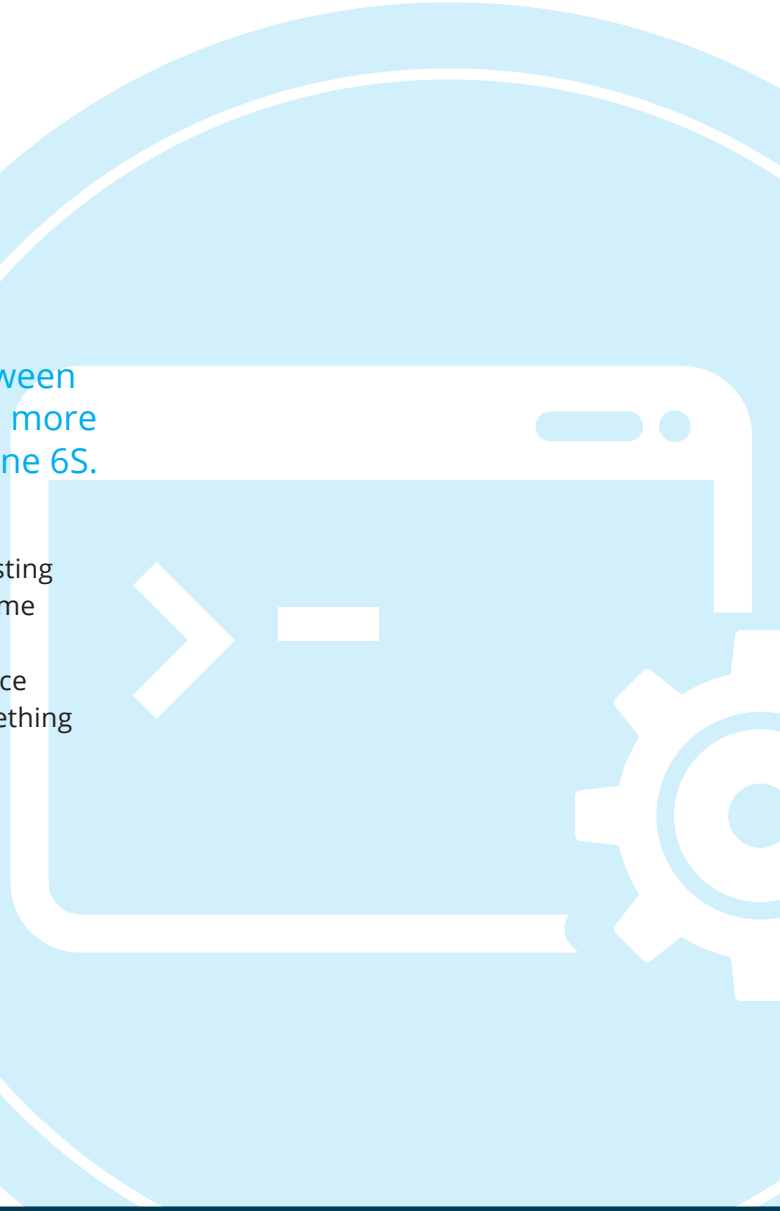
Many desktop websites can easily overwhelm slower or older devices, causing them to run out of RAM or forcing critical processes to be queued when needed to handle user inputs. For example, we have all experienced touching a mobile device screen a second time in order to register a click. Automated mobile website testing can exercise apps and websites to expose these conditions so you can discover problems before the user does.

Script Reactively

When developing website automation scripts, avoid fixed-duration waits between steps. Websites may vary in their responsiveness, and your site may perform more slowly on a Samsung GalaxyS3 Android device than on a Samsung S6 or iPhone 6S.

The device used while writing an automated UI script using test data or a backend simulation (such as service virtualization) may reliably respond within a certain number of seconds. But in the wild, server responses and network transfers are often delayed, and CPU characteristics may cause variable response times. Simply proceeding to the next step without syncing with the app's state is a common recipe for script failure. For example, issuing the next button click before the device has drawn the button may cause an entire automated testing run to fail.

Some defenses that can serve mobile UI testing include: waiting for a known object to become visible, using sync points or checkpoints, or in extreme cases checking to see if the device accepted an input and retrying if not - something that every user intuitively (and through experience) knows how to do.



Go Generic

The goal is to deliver a consistent user experience on a variety of devices and browsers – so it's a good idea to analyze traffic data and conduct surveys to figure out which browsers and devices customers are using. Use this information to determine which browsers and devices to test.

While it's important to be thorough, always maximize efficiency by prioritizing browsers when testing. For instance, with desktop web browsers, a common strategy is to start at the bottom: if a site runs properly in Internet Explorer 9, it will likely perform correctly in newer versions like IE 10. The same strategy works for phone operating systems: a site that works effectively on iOS7 will likely work on iOS8 too. However, if there is a major release of a browser or OS, it's important to ensure the website is still compatible.

Take advantage of automated testing tools that can increase productivity with powerful, cross-platform and cross-browser scripts, and tailor those scripts to handle all compatibility scenarios.

Embrace Hybrid

It's tempting to test websites on device default browsers (Safari on iOS and Chrome on Android) and then call it a day – but there are three mobile app models that must also be considered:

- Native app, written with the Google or Apple tools
- Web-only app, written in HTML and provided by a web server
- Hybrid apps that combine the two above

This means there are three distinct modes in which customers may encounter your HTML. Many will use the device's default browser and some will use a framework-based browser, like Dolphin, that wraps the platform's rendering engine (for example, WkWebView on iOS). Customers may also use a third-party app that links to your website by using WkWebView to display the site within the app.

Each of these possibilities may lead to compatibility or resource problems. For example, Safari may be more advanced than WkWebView, or the enclosing app may use CPU resources that your site counts on to complete a timely page load. Testing all of these scenarios and making changes to the website as needed will yield wider compatibility.

Detailed run results from automated website testing can provide more insight, giving a clear picture of how a website behaves in different browsers on various devices. For example, a good iOS test plan may test a framework-based browser such as Dolphin, a major 'external' browser like Chrome, and Apple's built-in browser Safari.

The website should operate as expected on all three, and it is important that the amount of graphics or animation being rendered doesn't overwhelm the CPU of older devices. For example, Geekbench 3 multithread benchmarks show that the Galaxy S3 is capable of only 40 percent of the processing possible on a Galaxy S6. If threads are stacked waiting to process input events while the base rendering engine is consumed with graphics or animation, the input events may not be reflected in time. In extreme cases, the thread carrying the base OS rendering engine can crash due to the inability of the CPU to make timely dispatches.

All of this suggests that a website needs a considerable performance margin when tested on high-performing devices and/or Safari on iOS or Chrome on Android. If the built-in browsers are barely able to handle the load, it's reasonable to expect trouble on slower devices and hybrid, third-party apps or browsers.

Expect Browser Compatibility Issues

Cross-browser compatibility should be top of mind during the entire development process. Tests are needed not only on different browsers but also on different versions of each browser.

Not all mobile browsers work exactly the same way. Think that the iOS dominant browser Safari comes preinstalled on Apple devices, for instance. While Safari may lead the way with optimizations and features, browsers that use a platform rendering framework like WkWebView may be, for a time, incompatible. Likewise, Chrome on iOS may offer a different feature set than Chrome on Android, so expect surprises – and test to find them.

The best way to make sure web pages render well in all browsers is to use an HTML markup supported by all current browsers that avoids reliance on features that only the newest versions may support.

When testing reveals compatibility issues, developers have one of two options: craft a universal solution that works on all browsers or let users know they have been restricted to a subset of browsers. But if users don't receive the news (which is almost certain), and use something not "certified," stand by for a possible one-star review. Keep in mind that mobility users expect things to work – and show little patience when they don't.

Anticipate Hidden Problems

Don’t underestimate the number of compatibility issues lurking in the background of every mobility project. Sometimes problems occur or disappear when changing OS versions, even if the website or browser version does not change

Supporting users on all platforms requires dealing with such seemingly “hidden” compatibility issues. It’s important to note that these issues often lie in the platform, not in the browser or your site code.

Some problems can come simply from heat – a mobile device doesn’t have the CPU cooling capability of desktop systems, and the operating system may, in times of intense CPU load, throttle the processor to control heat buildup. When this happens, the browser’s ability to keep pace with the website’s HTML may suffer. If you’re fielding a website that only works well on newer devices or under optimal conditions, be aware that the CPU is not a fixed-capacity resource, and when things get tight, the load you’re placing on the CPU may no longer be accommodated.

Automated mobile website testing can help avoid hidden compatibility issues by identifying scenarios in which a website may fail to load or crash.

Verify UI Actions

It may be necessary to verify some UI actions, particularly for large sites with many graphics and animations.

Because mobile phones are smaller than desktops, they may not be able to support heavy rendering loads and may be overwhelmed with information.

For this reason, it is important to test that UI actions register on all devices your customers actually use.

When planning for a mobile website, a best practice is understandably to scale down the number of pages, graphics, promotions and animations that appear. Automated website testing can show which UI actions perform well on mobile and which actions should be removed for optimal use.

Use Different Designs

Mobile phones are still lightweights when compared to the processing power, memory, graphics engines and screen sizes of desktop systems.

From a design standpoint, this means simplifying mobile displays and interactions. Sometimes scaling down a site won't be enough to make it responsive for mobile, and it may be wise to create a separate mobile web design.

Keeping in mind that size and speed are the two most important factors for mobile, a good strategy should consider the following:

Simple menus for easy navigation

Flexible layouts to ensure proper display on all screen sizes

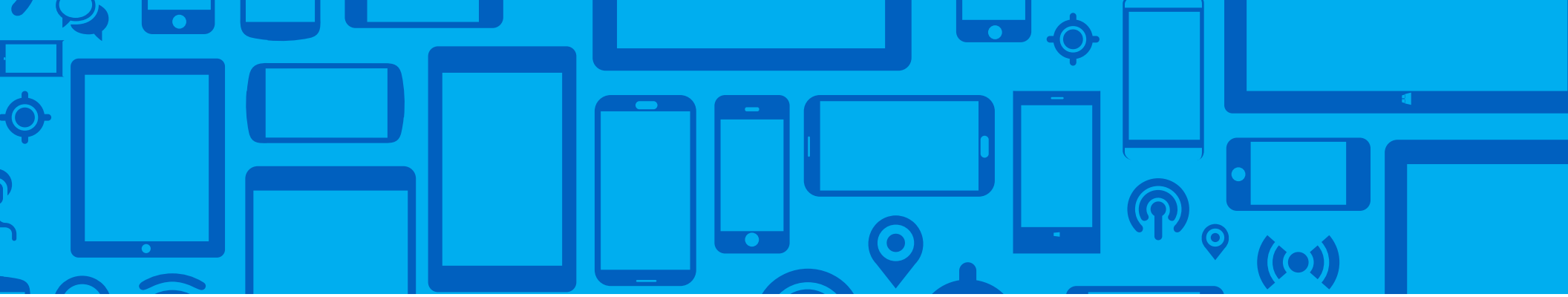
Large buttons and other clickable elements ideally sized for a finger tap

Minimal form fields that allow for autofill, to make providing information quick and painless

Limited special effects to ensure that pages load quickly

Clear, focused and scannable content that users can easily read on-the-go

Limited use of CPU-intensive, pre-packaged object types like the "hamburger button" especially when there is other CPU-intensive work on the page (heavy graphics, sub-pages, or animations)



In Summary

A mobile experience can solidify brands, drive sales and give a competitive edge – but only if a mobile website performs reliably and can adapt quickly to various devices and operating systems. Identifying the browsers, devices and environments the majority of your customers use, testing for compatibility issues, and making sure websites have sufficient performance headroom will all significantly improve the mobile experience.

Automated mobile website testing can be used to understand and monitor performance characteristics and to help identify compatibility problems. With securely-managed, remotely accessible, real mobile devices, developers and testers can get valuable insight into the capabilities of mobile websites and boost productivity. Accelerating mobile website testing means improved website quality, faster time to market, better performance and enhanced compatibility.

To learn more about Mobile Labs and how we can help with automated website testing visit www.mobilelansinc.com or e-mail us at info@mobilelabsinc.com.

MOBILE LABS