WHITE PAPER

When Chrome Isn't Chrome

How to set up browser detection: mobile browsers are complicated

Akamai G&L

When Chrome Isn't Chrome

INTRODUCTION

Your website should be tested in as many mobile browsers as possible so that you catch as many problems as possible at an early stage. This is commonly known.

In this white paper, we're going to take a good look at the mobile browser market with the aid of Akamai browser data. We will find that even though we start out with the knowledge that mobile browsers are complicated, it is even more complicated than we thought.

It is fairly easy to test the default Safari browser on iOS and Google Chrome on Android, and we assume that you're already doing so. However, the Android situation is much more complex.

We're going to examine many peculiarities of the browser market, and give some tips and tricks on setting up a decent browser detection module.

A word of caution beforehand. Browser detection is a vital ingredient of analytics. You want to know which browsers your site's audience uses, since that will allow you to decide on which browsers to test.

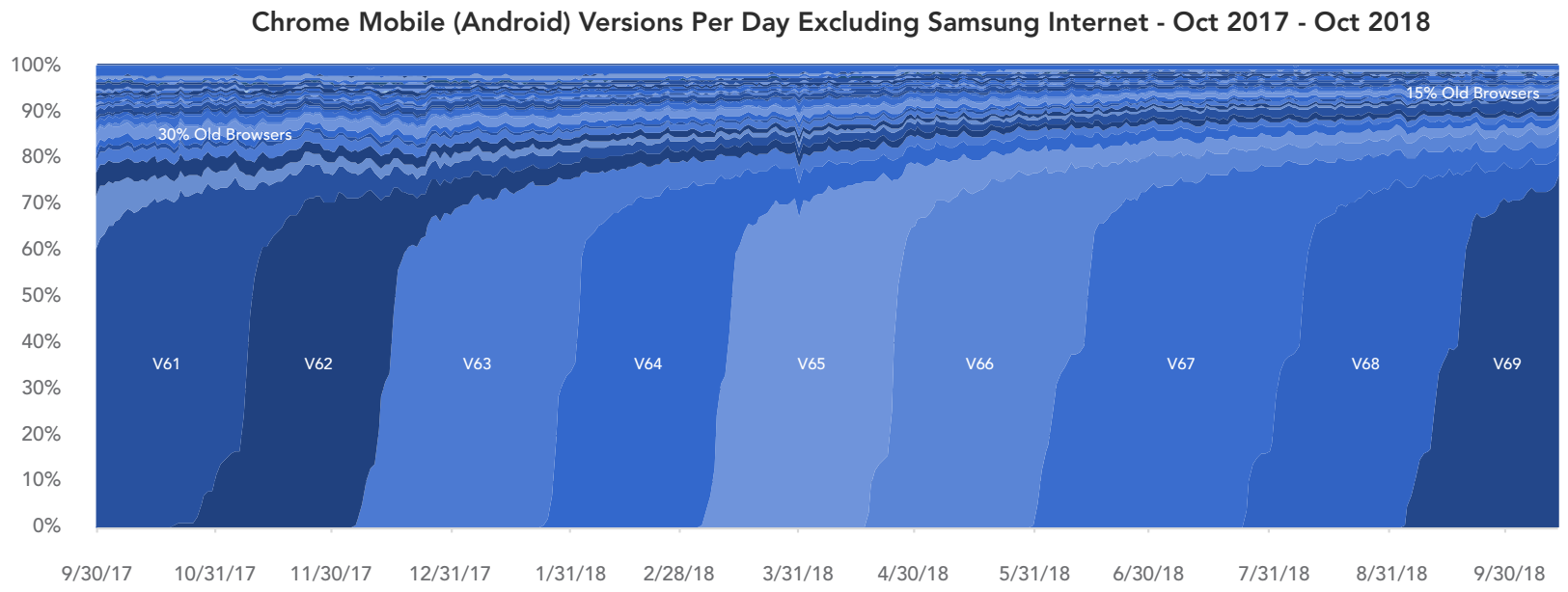
However, using browser detection in your actual production code is not a very good idea. Not only will rough browser detection fail to catch many of the subtle distinctions we'll discuss below, but in almost all cases it's the wrong solution to whatever your problem is, since it's too coarse.

And if you're looking for a good browser detection script, we can recommend WhichBrowser, which powers the HTML5 Test site. It is being kept up to date with the latest mobile browsers and devices. You can also try the WURFL detect, which is better at detecting devices than at detecting browsers.

GOOGLE CHROME

In this graph of the most commonly used browsers on Samsung devices, Google Chrome is unsurprisingly first, but its market share is only 72%. That's a healthy majority, but about one in four Samsung users does not use Google Chrome.

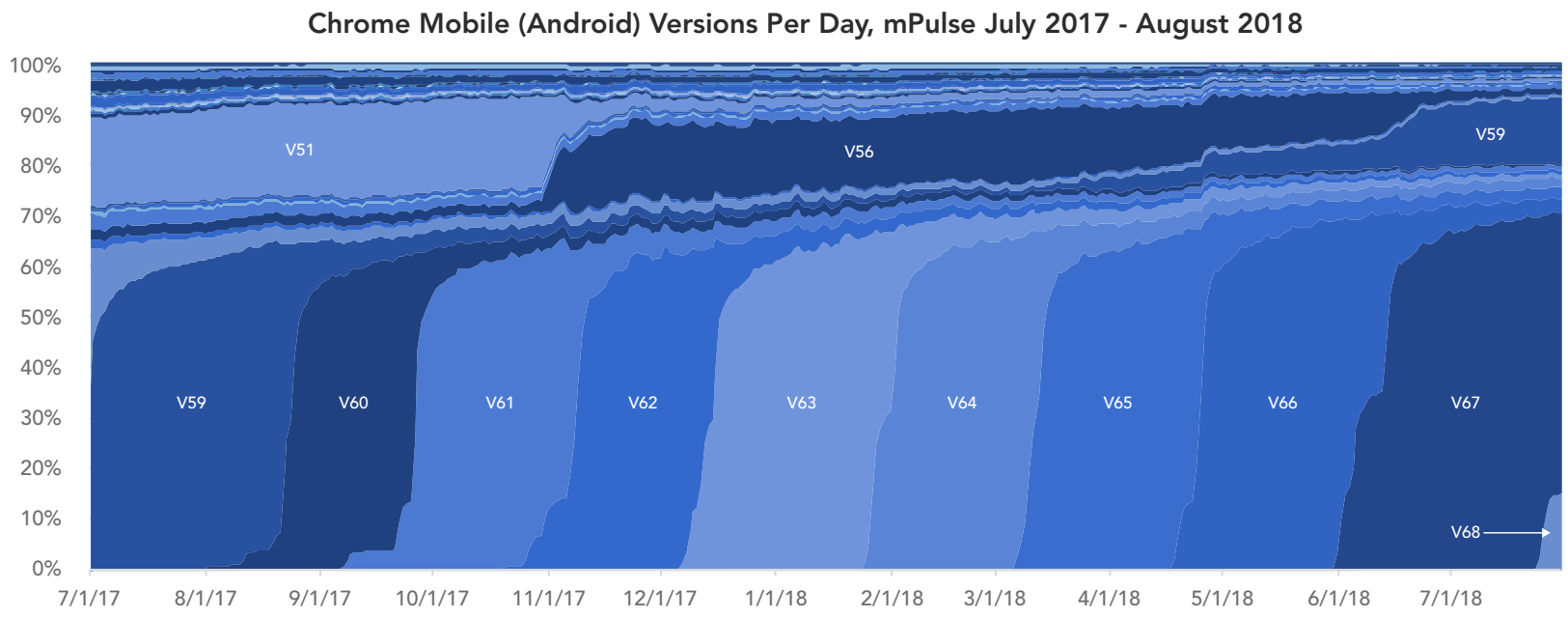
Besides, even if the majority of Android users use Google Chrome, they do not necessarily use the latest version. Akamai's data shows a clear picture: When a new Google Chrome version is released, roughly 70% of the users update it within three weeks, which is quite a good adoption rate. However, the remaining 30% are much slower to update, and continue to visit your site with older versions for months or even years.



At the time this snapshot was taken, October 2018, Google Chrome 69 was the latest version. About 70% of all Chrome users switched to that version within a few weeks. The other users, however, were already predominantly on older versions, and stuck with them. In fact, version 55 still had a 1% market share. (This snapshot excludes Samsung Internet.)

NON-GOOGLE CHROMES

The first graph showed a decent chunk of market share, 23%, is for Samsung Internet. This is Samsung's own browser. Although it is based on Google's open-source Chromium browser, it is not the same as Google Chrome.



NON-GOOGLE CHROMES

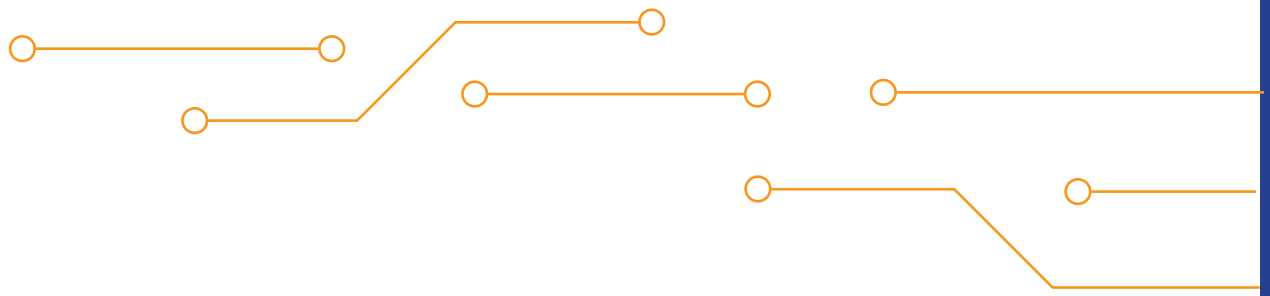
Many Android device vendors have, or used to have, their own Chromium-based browsers. Samsung Internet is the best-known and largest one, but other device vendors – such as Xiaomi, Motorola, and Huawei – also have their own. All of them put their own browser on the home screen, while hiding Google Chrome in the Apps menu, thus hoping that users will use their browser instead of Google's.

This snapshot again shows Chrome versions, but this time it includes Samsung Internet. Do you see the big bulge in versions 56 and 59? That's Samsung Internet, with a market share around 20%.

These device vendor browsers are sometimes hard to distinguish from Google Chrome. Samsung Internet, whose latest versions include a helpful Samsung browser in their UA string, is not a problem, but the minor browsers are. Also, older HTC, LG, and other devices used now-discontinued vendor-specific browsers. Usually, the UA strings of these browsers aren't terribly clear about whether they are regular Google Chromes or vendor-specific versions.

Therefore, most browser detects do not count Chrome-based browsers accurately. Most of them group Samsung Internet, Google Chrome, and other Chrome-based browsers together as "Chrome," so that it appears Google Chrome has a far larger market share than it actually has.

Technically, this is not a huge problem. All of these browsers are based on Chromium and support roughly the same web technologies. Nonetheless, similar does not mean the same. Vendor browsers use different versions of Chromium. They might also add extra features.

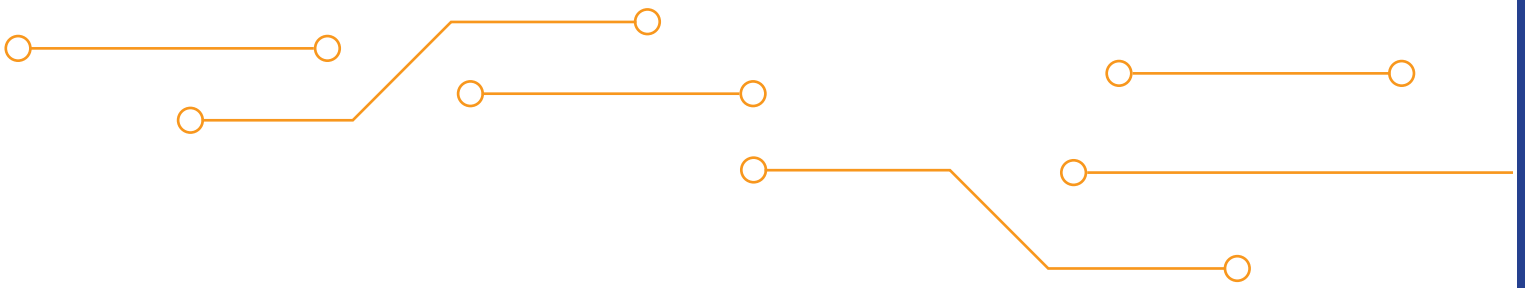


NON-GOOGLE CHROMES

The best example of an added feature is zoom reflow. If you zoom in beyond the width of a line of text, most mobile browsers show only part of that line, and you have to pan left and right to read all of it.

Not so in the Xiaomi and the discontinued HTC vendor browsers. They instead reflow the text so that it fits the screen. This is a great bit of user experience that other browsers should also consider. On the other hand, reflowing the text means making the text block higher, and that might change other parts of the page layout. Web developers should be aware of this, and test accordingly.

From a browser-detect perspective, the problem is even harder. While modern Samsung Internet versions are easy to detect, we have to find another solution for the minor browsers. The best long-term solution is to create a whitelist of Chromium versions that have been used by specific device vendors. The browser detect would take the device vendor and Chromium version from the UA string and consult the whitelist to see if it's a Google Chrome or a vendor Chromium. Some edge cases will probably fail this test, and the whitelist would have to be maintained, but it's the "least bad" solution available.



WEBVIEWS

Some hits do not come from a browser that the user enters a URL in, but from a WebView.

WebViews are separate browsers on mobile devices that are used by native apps that allow their users to open web pages in the context of the app. Important apps that use WebViews are Facebook, Twitter, Instagram, Snapchat, and many Chinese services.

Android WebViews are mostly based on Chromium, but, again, not necessarily on the latest version. The iOS WebView is almost the same as the default Safari browsers. It is used by apps, but also by non-Safari browsers such as Chrome and Firefox for iOS. Thus, a Chrome on iOS is actually a Safari WebView. (Your analytics tool may classify it as a Chrome browser, but that is simply wrong.)

Most browser detects do not distinguish between WebViews and regular browsers. Although the differences are not huge, WebViews should be part of your testing strategy. Besides, it might be interesting to know how many of your page hits come from social media.

On Android, detecting a WebView is very easy. If the UA string contains "Version/4.0" or "; wv)" it is a WebView. Sometimes the UA string contains additional information about the app that launched it.

You can test your sites in WebViews yourself by downloading a WebView test app from Google Play; there are plenty of them.

iOS, unfortunately, is a serious problem, since there is no distinguishing feature in the UA string. If you find an app name such as Facebook or Snapchat in the UA string, you can safely assume that this is in fact a WebView, but I suspect that many WebView UA strings do not carry an app name, and are thus undetectable.

APPS

Some WebView UA strings give information about the app they were started up by. This could be useful commercial or business information for you or your clients. Unfortunately, many WebView UA strings do not contain this information.

On Android, where WebViews are detectable by themselves, about 90% of UA strings do not give any indication which app used them. Some of them may be small apps, but a few large ones are also quite vague.

Twitter, in particular, is a problem. I ran a test where my Twitter followers sent me 89 UA strings opened by their Twitter apps, and in only 7 cases – all of them on iOS – did the UA string contain "Twitter,"

Interestingly, zero of the Android apps announced themselves as "Twitter," even though I found a few Twitter WebView UA strings in Akamai's data. Thus, we can safely conclude that the majority of Twitter apps on both iOS and Android do not add "Twitter" to the UA strings of their WebViews.

There are also Chinese apps, such as WeChat and Weibo, which use WebViews and announce themselves in the UA string. If your site has many Chinese visitors, it might be a good idea to look into this – not only to see which apps refer visitors to you, but also because Chinese WebViews are even less documented than Western ones.

Thus, while sometimes UA strings will give information about apps, this is far from universal, and data acquired by such means should be treated as incomplete at best.

NON-CHROME BROWSERS

There are other Android browsers as well. Firefox is the best-known of them, but it has a very slight market share. More important in terms of market share, but less well known in the West, are Chinese browsers such as UC and QQ. They are used in East Asian developing countries – not only in China itself, but also in other large countries such as India and Indonesia.

For instance, in Indonesia, UC has a market share of 6% (and is larger than Samsung Internet at 5%). Some browser detects correctly separate UC from Google Chrome; others do not, or do so insufficiently.

Again, the differences are not huge. UC recently moved from WebKit to Chromium, but – just like with the Android vendors' browsers – they may use an older version, and there will be some slight differences with a Google Chrome of the same version number.

Finally, there is Android WebKit, which is the WebKit-based pre-Chromium default browser that last appeared in Android 4.4.3. You will occasionally get a hit from this browser, which wasn't bad for its day, but is now outdated.

PROXY BROWSERS

There are also proxy browsers such as Opera Mini and UC Mini. Proxy browsers run their rendering engine on the server and send their client only a PDF-like file that contains the web page. As a result, there is hardly any client-side interactivity, but that is offset by a compression of up to 90%. This allows the user to save a lot of money on data connections, and the possibility to run the client on a very old or very basic device, since it is basically a glorified document viewer.

Detecting proxy browsers is usually not a problem, but figuring out on which device they run is a serious issue with Opera Mini, which often just says "J2ME." Also, monitoring performance on these browsers is extremely difficult, since they may run some of your tests on the server and then send the result to the client along with the web page itself.

Then there is Puffin, which is a hybrid browser. Again, detecting it is not a problem, but its performance metrics might be. Puffin runs its rendering engine on the server, like proxy browsers do, but it also offers some interactivity on the client – notably, Flash and some JavaScript events. It is possible that we have to treat Puffin as an entirely separate third category, in addition to regular browsers and proxy browsers.

OTHER OPERATING SYSTEMS

Although iOS and Android combined have 97-98% mobile OS market share, you will occasionally find other operating systems. Most likely those will be BlackBerry, Windows Phone, or Symbian. Browser-wise, the first two are easy: Windows Phone runs IE, and BlackBerry the BlackBerry browser, which is WebKit-based and was pretty decent in its time. Symbian is more complicated, since – like Android – it allowed the installation of other browsers, but distinguishing between the various Symbian browsers is probably not your top priority.

Finally, there's newcomer KaiOS, which is a feature phone OS based on Firefox OS. It runs a Firefox 48, but it's still an open question whether that is the same Firefox 48 as on Android or the desktop. Here's a sample UA string:

```
Mozilla/5.0 (Mobile; Nokia_8110_4G;
rv:48.0) Gecko/48.0 Firefox/48.0
KAIOS/2.5
```

Devices

Then there is the device type. On Android that's relatively easy to find, since it usually contains a cryptic and vendor-dependent, but still known, device code such as SM-G928I, which denotes a Samsung Galaxy S6 Edge+. The WURFL device detection script is especially good at translating these codes to device names.

On iOS, unfortunately, the situation is a lot more murky. The UA string doesn't give any information about the device, just about the iOS version. This is partly offset by the fact that you could read the device's screen size.

That helps a bit, but it still points you to a range of devices that have the same width instead of one specific device. On the other hand, device detection is somewhat less necessary on iOS than on Android.

Browser detects

We've reached the end of our overview of the mobile browser market, and as we saw it's much more complicated than is generally assumed, especially on Android.

A good, up-to-date browser detect is a crucial analytics tool for site owners and maintainers. Variations in mobile devices and browsers can impact your user experience, and possibly affect KPIs such as bounce rates negatively if a user of an uncommon browser cannot access your site, or can do so only with difficulty. How serious of an issue this is depends on the exact browser makeup of your audience.

And that's the problem: Most site-monitoring tools give insufficient information about this browser makeup. If you base your customer engagement strategy – and thus the tasks you set yourself – on outdated or faulty tools, your efforts may be compromised from the start, and your testing strategies might miss part of your audience.

As we saw at the start of this paper, the WhichBrowser and WURFL browser detects are fairly decent, though even they occasionally miss the telltale signs of a WebView, and count it as a regular browser instead.

On the other hand, if you simply assume about 10% of your audience will use a non-standard browser, and test in a few non-standard browsers (I recommend Samsung Internet, UC, and a WebView), you've already solved most of the problem.

https://www.gl-systemhaus.de/

Akamai *Experience the Edge*

Akamai secures and delivers digital experiences for the world's largest companies. Akamai's intelligent edge platform surrounds everything, from the enterprise to the cloud, so customers and their businesses can be fast, smart, and secure. Top brands globally rely on Akamai to help them realize competitive advantage through agile solutions that extend the power of their multi-cloud architectures. Akamai keeps decisions, apps, and experiences closer to users than anyone — and attacks and threats far away. Akamai's portfolio of edge security, web and mobile performance, enterprise access, and video delivery solutions is supported by unmatched customer service, analytics, and 24/7/365 monitoring. To learn why the world's top brands trust Akamai, visit **www.akamai.com**, **blogs.akamai.com**, or **@Akamai** on Twitter. You can find our global contact information at **www.akamai.com/locations**. Published 11/18.