The Complete Guide to

BUILDING A RESPONSIVE WEB TESTING STRATEGY

Perfecto

Table of Contents

Introduction

Responsive web design (RWD) is becoming the preferred method for developing websites that look and work consistently on all devices (see Fig. 1). The idea behind this strategy is to give all users a seamless digital experience no matter what device, browser, screen size or orientation (portrait vs. landscape) they use. It also means ensuring that the functionality, performance and visual layout of websites are consistent across all digital platforms and various user conditions.

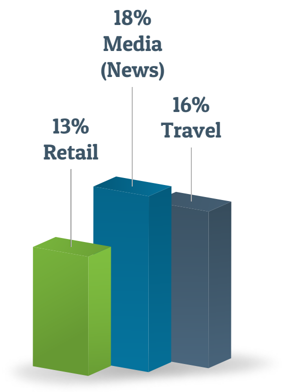


Fig. 1: Percentage of responsive websites in a few key markets (source: Forrester Research)

At first, this may seem like a no brainer, but when you factor in the importance of user experience (UX), the continuous testing of new features, and guaranteeing your website is working optimally on all browsers, devices, OSes and carrier networks, RWD is actually daunting and time consuming. Understanding how a responsive website behaves and looks across all digital platforms is a great challenge for many app development and QA teams.

This guide — written for developers, QA managers and line of business groups — outlines some of the key challenges of developing, testing and maintaining responsive websites. It provides a practical set of lessons on how to deal with these challenges and succeed in delivering great responsive websites.

Applying these lessons to your day-to-day DevTest activity will enable you to:

- Deliver better user experiences (UX)
- Achieve shorter time to market
- Enhance software development life cycle (SDLC) processes
- Bridge the gap between marketing, IT and DevTest
- Meet business goals

Responsive web design (RWD) is an approach to web design that provides an optimal viewing and interaction experience — easy reading, navigation — across a wide range of devices (from desktop computer monitors to mobile phones)

What's Driving the Shift to RWD?

The answer is simple. In today's world, there's no longer a difference between website usability and the platform used to consume data and services. According to Criteo's State of Mobile Commerce report, 4 out of 10 transactions today take place on multiple devices (Fig. 2). In that context, 48% of users today complain that the websites they use are not optimized for their smartphones and tablets.

By that measure, delivering a great user experience across all digital platforms is critical to driving more traffic and meeting business goals. Organizations that wish to be ready for new device and operating system releases should select RWD as their app development method because responsive sites are, by definition, designed to work on any device and platform.

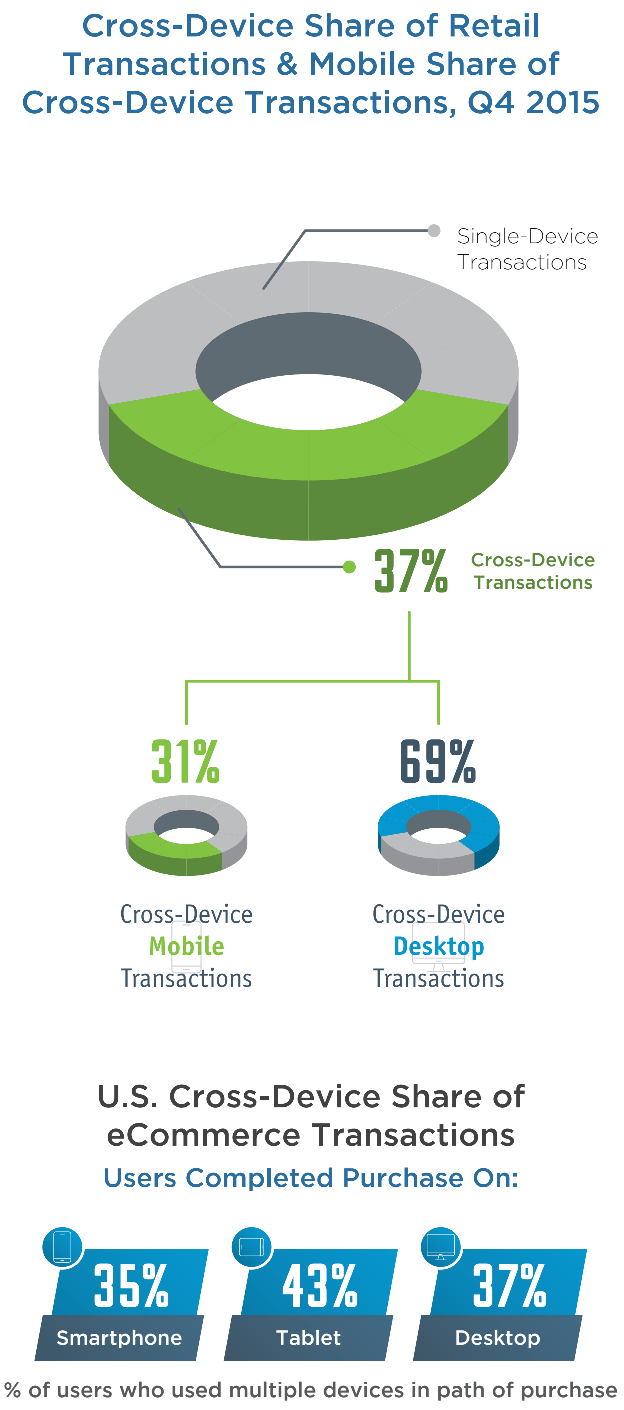


Fig. 2: Multi Screen transitioning statistics (Source: Criteo State of Mobile Commerce Report 2015

What’s Driving the Shift to RWD?

“Context” is a key success factor for RWD adoption. If your website is used primarily on a large desktop, it makes sense to display as much content on the screen as possible, but what happens when users consume the data on a mobile device or a system with a smaller viewport?

It probably means that less relevant content needs to be hidden and creative elements and fonts need to be resized. The same considerations need to be made about the location from which a user is accessing your site. When connected via a strong Wi-Fi, your site may be optimized, but what happens when network connectivity is diminished or the website relies on delivering location-based content? The context of your site will factor into how you approach RWD.

Another reason for the shift to RWD is that it helps assure quality and fast website updates across all digital platforms. Because mobile and web evolved in parallel, many organizations created siloed teams with different skill sets. They often relied on the same tools but disconnected roadmaps and different release schedules. This disconnect prevents web and mobile DevTest teams from having control over the entire digital portfolio. When developing for different platforms, it’s hard to assess the quality of every digital viewport and function across web and mobile because different teams are executing different test cases instead of running tests side by side.

The RWD approach is seen as the most efficient because it consolidates efforts, synchronizes the digital teams and reduces the R&D budget.

So how do you test for RWD?

“*Because mobile and web evolved in parallel, many organizations created siloed teams with different skill sets. They often relied on the same tools but disconnected roadmaps and different release schedules. This disconnect prevents web and mobile DevTest teams from having control over the entire digital portfolio.*”

6 RWD Test Plan Building Blocks

Here are the top six areas that need to be covered in an RWD test plan.

Test Scenario	Test Description
Visual Testing	Assure RWD content looks right on any screen and platform.
Client side performance testing	Test and measure the time it takes your website's objects to render on screens and optimize the content size accordingly.
Navigation testing	Assure that the site performs the expected function correctly (i.e. Log-in functions well across web and mobile platforms).
User condition testing	Test your site across platforms for real user conditions such as incoming calls, ads, popups and other interruptions.
Accessibility testing	Comply with the accessibility standards across mobile and web, and assure that your RWD site adheres to the guidelines through voice, visual and other engagement methods.
Platform coverage	Use analytics to test your website against all relevant devices, browsers and OS versions covering screen sizes and resolutions in various conditions and locations.

1. Add Visual Testing to Your Test Automation Code

A responsive website will display content differently when screens resize and user conditions change. With this layer of complexity, app development and testing teams must combine various validations to make sure that when context changes, the viewports also change and the content being displayed is accurate, not truncated, and does not cause usability glitches.

The way to achieve this is to build in a cross-platform test automation strategy that can identify all DOM objects on desktop and mobile web browsers. You must also add relevant UI check point validations that can compare the visual display on the screens when events occur (Fig. 3.1, 3.2). These validations will quickly highlight issues and shorten the feedback loop to the developers, resulting in faster resolution.

Fig. 3.1: Visual object identification is important as part of responsive website testing (Source: starecat.com)

Some important check points to consider, from a visual standpoint, when testing responsive websites:

- Alignment of text, controls and images
- Text, images, controls and frames do not run into the edges of the screen
- Font size, style and color are consistent for each type of text
- Typed text (data entry) scrolls and displays properly
- Pages should be readable on all resolutions and screen orientations
- Content defined 'important' needs to be visible in all breakpoints

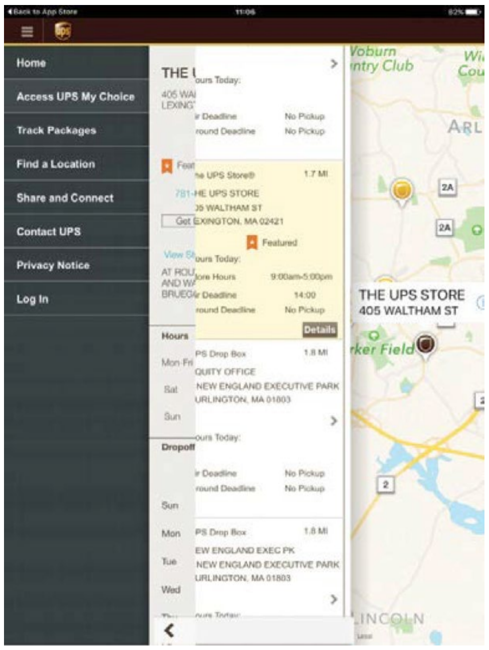


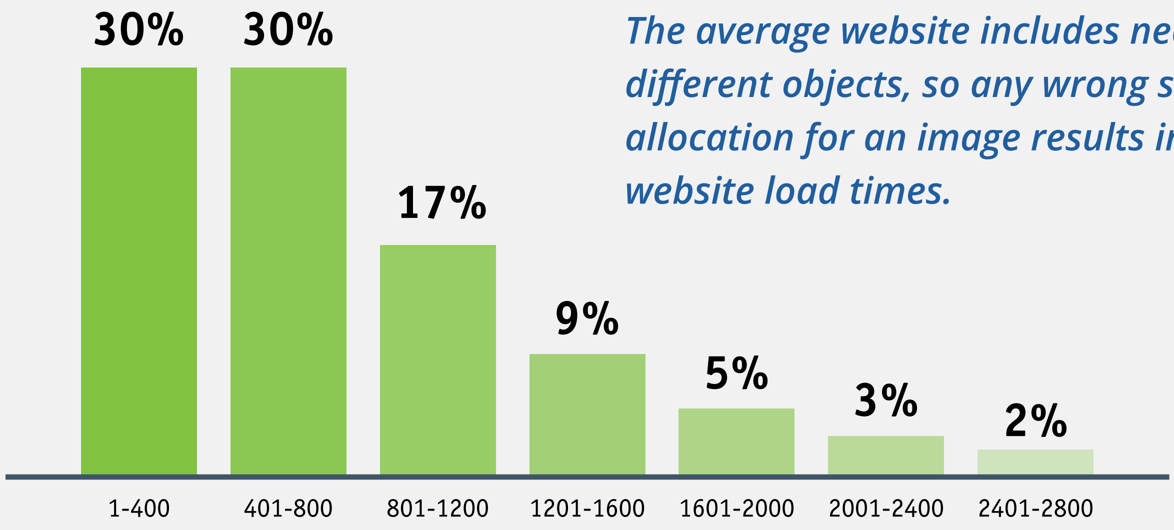
Fig. 3.2: Visual truncation when switching to landscape mode (UPS Mobile App, iPad 3)

2. Do Client Side Performance Testing

One aspect of a RWD test plan that will assure a great user experience is web performance. Because RWD is targeting a large variety of combinations (Safari on specific Mac OS versions, IE browser on Windows XP, etc.), dev and test teams should test the time it takes content and images to load on the various viewports. Teams will also want to look carefully at the overall website performance and how it varies on different platforms and under specific network conditions.

Two of the most common issues with RWD sites are:

- **The lack of large content compression in the website, which causes slow page load times.**
- **Failure to measure the object sizes and make sure they are customized to the screen sizes on which they appear.**



The average website includes nearly 400 different objects, so any wrong screen size allocation for an image results in longer website load times.

Fig 4.: Average number of Web Objects per website (Source: HTTPArchive.org)

The average website includes nearly 400 different objects (Fig. 4), so any wrong screen size allocation for an image results in longer website load times. With that many objects in play, it's easy to see how poor responsive web design can have a negative impact on user experience.

When we looked at a leading responsive site, we found that the original size properties of an image on that site was 611 x 435, while on a device that accessed the responsive site, the actual dimensions were only 274 x 195. This translated into that image loading slowly on that specific mobile device.

3. Test Your Navigation Across Platforms

As more consumers and brands go digital, responsive web users will most likely use your site across different devices and OS platforms. They will either start from a smartphone, move to a tablet and then to a desktop browser, or sometimes even reach your RWD site from a non-responsive site. From a testing perspective, these kinds of user paths need to be covered as part of [navigation testing.]

Navigation testing is done to assure that a user can comfortably complete a full end-to-end run through your site. As part of the process, you need to make sure that the screen orientations in mobile and desktops work well so that nothing breaks when moving from portrait to landscape and vice versa. Testing screen orientations and other navigation elements such as shortcuts, menus and other web elements can improve the user experience when users access the site from a smaller screen.

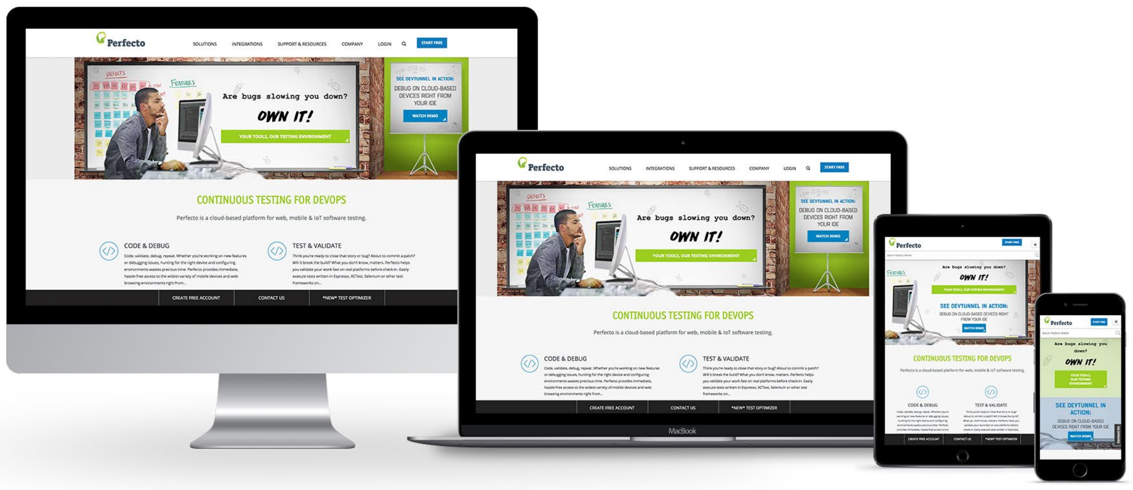


Fig. 5: Responsive Perfecto Homepage (Source: Perfecto)

When moving between screen sizes and devices and resizing windows, Google recommends the following best practices for breakpoints (the points at which your site is not displaying information correctly on certain screen sizes):

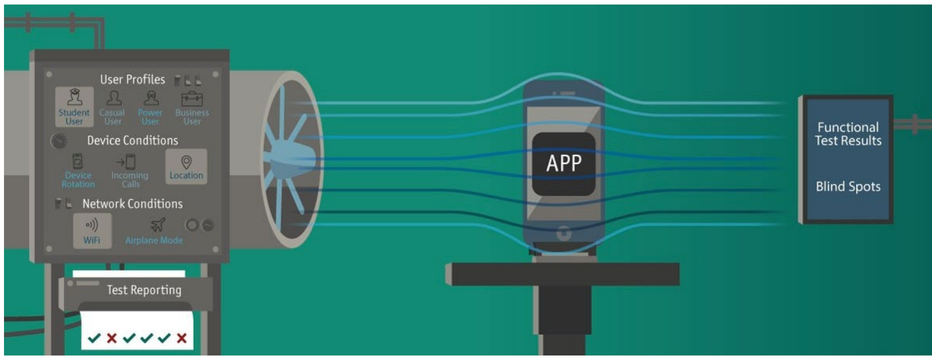
- **Create breakpoints based on content, never on specific devices, products or brands**
- **Design for the smallest mobile device first, then progressively enhance the experience as more screen real estate becomes available**
- **Keep lines of text to a maximum of 70–80 characters**

Another tip is to test what happens when users transition between responsive and non-responsive web environments.

4. Integrate Real User Conditions into Your Testing

Now that you're testing for mobile and desktop combinations, you need to guarantee that these environments also mimic your users' daily, real-world conditions. We recommend you start by collaborating with marketing and business groups on target user data, including insight into who your target user is, where they live, and what are their network conditions.

Testing ideal "happy path" scenarios (i.e. strong Wi-Fi, full battery, no competing apps) will provide very limited insights into how your app will actually behave under real user conditions, so consider creating personas (profiles of your target customers) that account for complex, real-world environments.



Instead of running only functional tests such as log-in, search, payment and checkout under generic conditions, we recommend you also test responsive websites against user conditions including location, preferred device(s), network coverage and other apps running in the background. This will help ensure that your content is laid out correctly on any viewport.

By definition, RWD requires constant network connection, so it's key to test real-world conditions with both Internet connectivity and without, and test for the poor connectivity that occurs when moving through tunnels and switching to airplane mode.

In today's competitive landscape, websites are also exposed to interruptions caused by web add-ons and adware popups. Pop-up interruptions are one of the most common web "user conditions" to test for, as they can block user flows within the web page and cover up important content.

In such cases, dev and testing teams need to assess the user impact of pop-ups. We all know they can be quite disruptive. Unfortunately, blocking the ads does not solve the problem because the ad vendor will sense that ad blockers are on and will pop up on a different page.

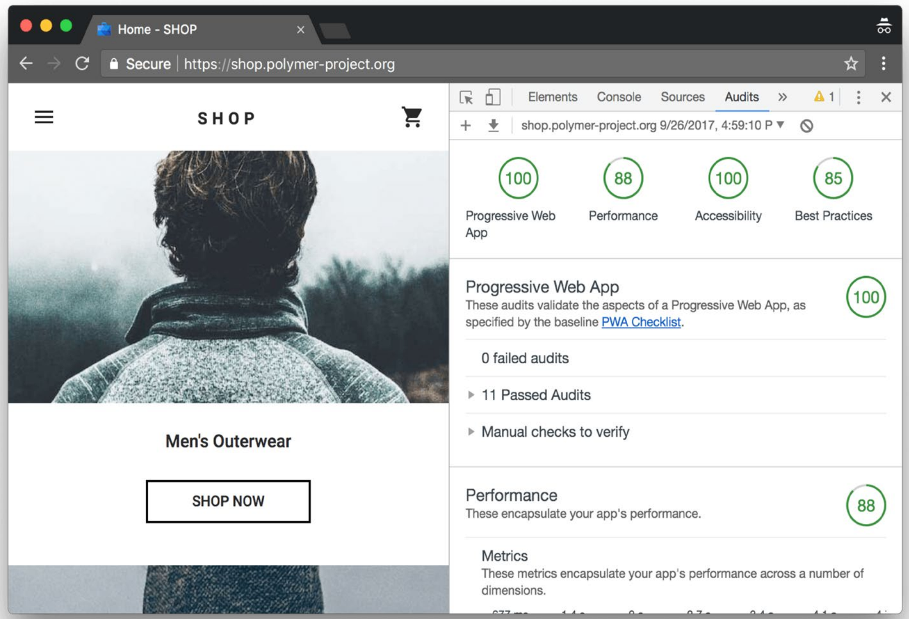
As part of user condition testing, and especially for mobile, we recommend mimicking popups and add-ons as well as incoming events such as calls and security alerts. Assess their impact on the user experience and make sure they do not cover any critical web content.

5. Implement Accessibility Testing Continuously

Web accessibility means opening accessibility of the web to everyone, specifically those who have disabilities, allowing them to perceive, understand, navigate and interact with the Web. These disabilities cover all levels, including auditory, physical, speech, cognitive and neurological. Most Websites have some sort of accessibility barrier that makes it difficult for a person with a disability to use their site. web accessibility assists making sure that people with all disabilities do not face these roadblocks when accessing the web.

Organizations are struggling today to automate and perform per each release cycle the required accessibility tests according to WCAG and other requirements. While for web, there are tools such as WAVE and others, for mobile, that involves more advanced interfaces like voice, and sensors, it is much harder to comply and automate all of the requirements. Even if automation is a challenge, it is critical to at least manually perform most of accessibility tests on your RWD at critical product milestone to assure no regressions found when a new browser OS is introduced or a new mobile OS is released.

With that, it is important to follow the innovations in the marketplace and the open-source community and identify new tools that can help reduce some of the manual tests. An example of such tool, is the built-in Chrome developer tool called Lighthouse. Using the tool to perform audit on your web page, can produce useful accessibility and quality reports related to your RWD site (see example to right).



Progressive Web App 100
These audits validate the aspects of a Progressive Web App, as specified by the baseline PWA Checklist.
0 failed audits
11 Passed Audits
Manual checks to verify

Performance 88
These encapsulate your app's performance.
Metrics
These metrics encapsulate your app's performance across a number of dimensions.

Fig. 6: Google Lighthouse DevTool (Source: Google Developer portal)

6. Use Analytics to Establish Your Digital Test Coverage Plan

Responsive web relies on optimizing user experience across a huge digital landscape that consists of thousands of desktops (Fig. 7) and mobile combinations.

The key to defining the right digital test coverage plan lies in leveraging various analytical sources — some are available within the [organization developing the RWD](), while in other cases you'll need to find a vendor.

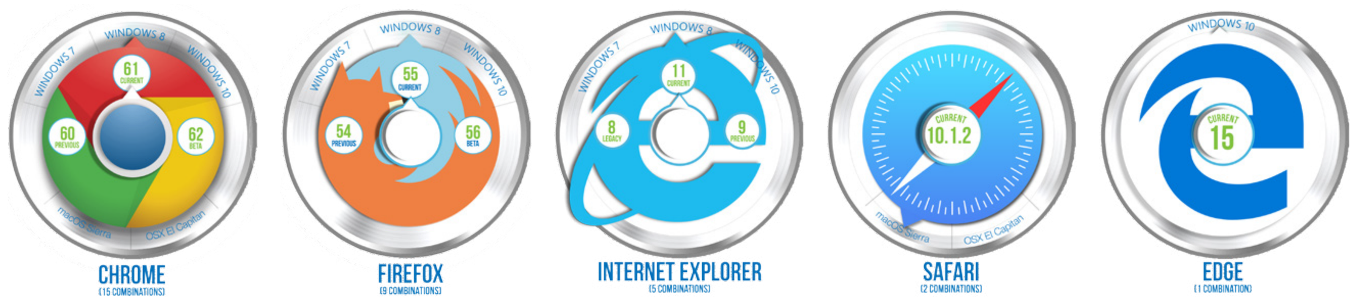


Fig. 7: The Web Coverage Matrix (Source: Perfecto)

As noted earlier, web development and testing teams should obtain the most recent mobile and web traffic analysis. This information will give DevTest teams data about popular locations, browsers, and mobile OS/devices that were used to access their website. But while this is a great start, it's not enough to build a test coverage plan because it does not factor in the larger market, your competitors or the newest platforms and configurations. However, solid traffic analysis will at least give you an understanding of all the browsers and mobile platforms you need to consider for your test coverage plan.

To stay on top of market changes, we recommend the following best practices:

- **Continuously test using a sanity automation test suite that includes RWD supported platforms and a feedback loop between Dev and Test.**
  - Make sure to include functional, performance and usability aspects in your sanity tests
  - Consider testing on beta versions for both mobile and desktop OSes to reduce risks
- **Review the market trends and tune your test coverage accordingly. It's critical to include desktop and mobile beta OS versions in your coverage plan. Retire irrelevant setups and make room for new platforms and beta OS versions as needed.**

Advanced Topics in Testing RWD Sites

Covering the site navigation, visual and performance are a critical piece of the testing activities for a RWD site. The constrains projects have today around time, budget and resources drives both developers and testers to optimize the quality-related activities throughout the entire DevOps pipeline.

In a recent ebook that covers the "10 Test Automation Frameworks for Cross Browser Testing", we've found some key insights around the desktop browser marketplace and what they can offer. Some of the findings can specifically impact the overall RWD testing strategy.

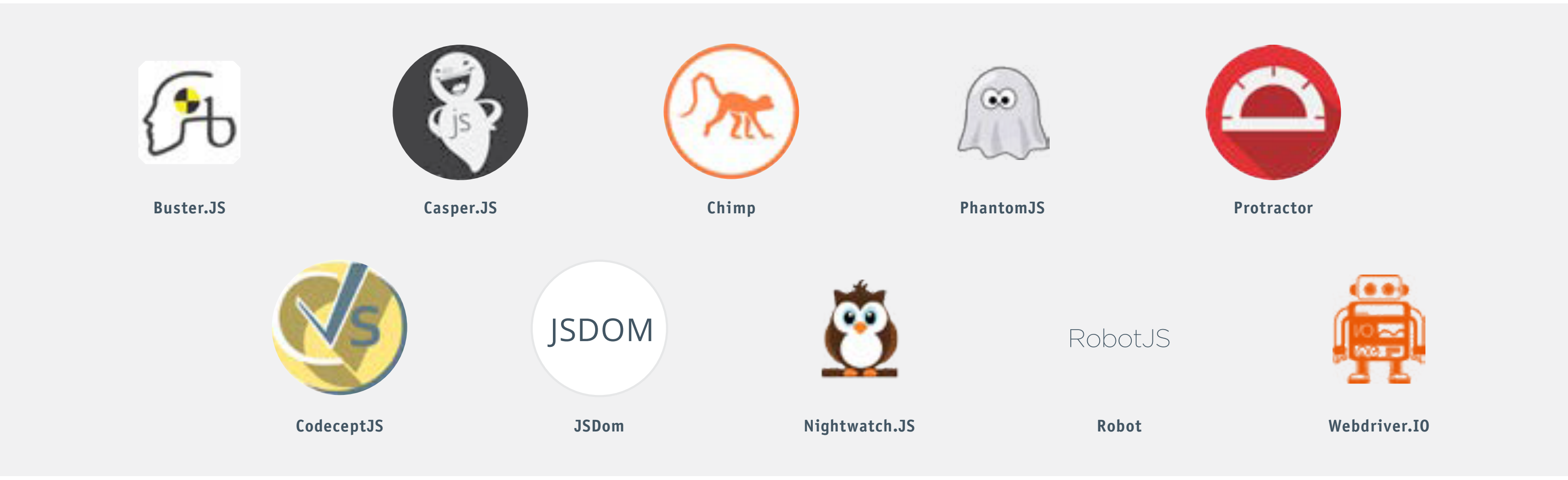


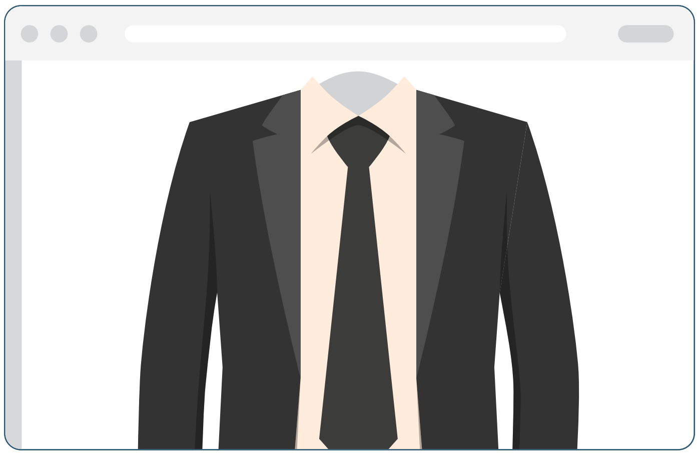
Fig. 8: Top 10 Test Frameworks (Source: Perfecto)

In addition to the ebook referenced above, here are some additional related activities that can help in the quality assurance process:

1. **Headless browsers in the context of cross-browser testing**
2. **Optimize performance and security through HAR file analysis**
3. **Use in-browser capabilities to increase test productivity**

1. Headless Browsers Role in the Context of Cross-Browser Testing

Headless browsers are serving a unique goal as part of the overall quality validation, however the usage of such tools should be tuned to the right maturity level of the RWD site, as well as to the type of tests you wish to author and execute.



Headless browser tools like PhantomJS, JSDOM, FireFox headless, and the newly introduced Puppeteer project from Google, are helping developers and testers to perform unit tests, basic functional tests as well as security and network traffic analysis validation hat does not require the full blown browser UI.

While PhantomJS is becoming obsolete, and the desktop browser vendors like Google and Mozilla, are taking ownership for these tools, the value remains the same.

1. Headless Browsers Role in the Context of Cross-Browser Testing

Headless Browsers are beneficial because of the following:

- **Zero environment setup and quick test code development – reduces overall test cycle**
- **Fast feedback and robust executions – less flakiness, more test automation productivity, faster CI cycles**
- **Leverage Java/JavaScript/Selenium WebDriver to code against headless browser – familiar tools and scripting languages for test automation engineers and developers**
    - Get specific insights from your RWD site around:
    - Page load performance
    - Page rendering through screenshots
    - Network traffic validations through HAR file
    - Basic navigation and functional steps

Headless Browsers are not suitable for the following:

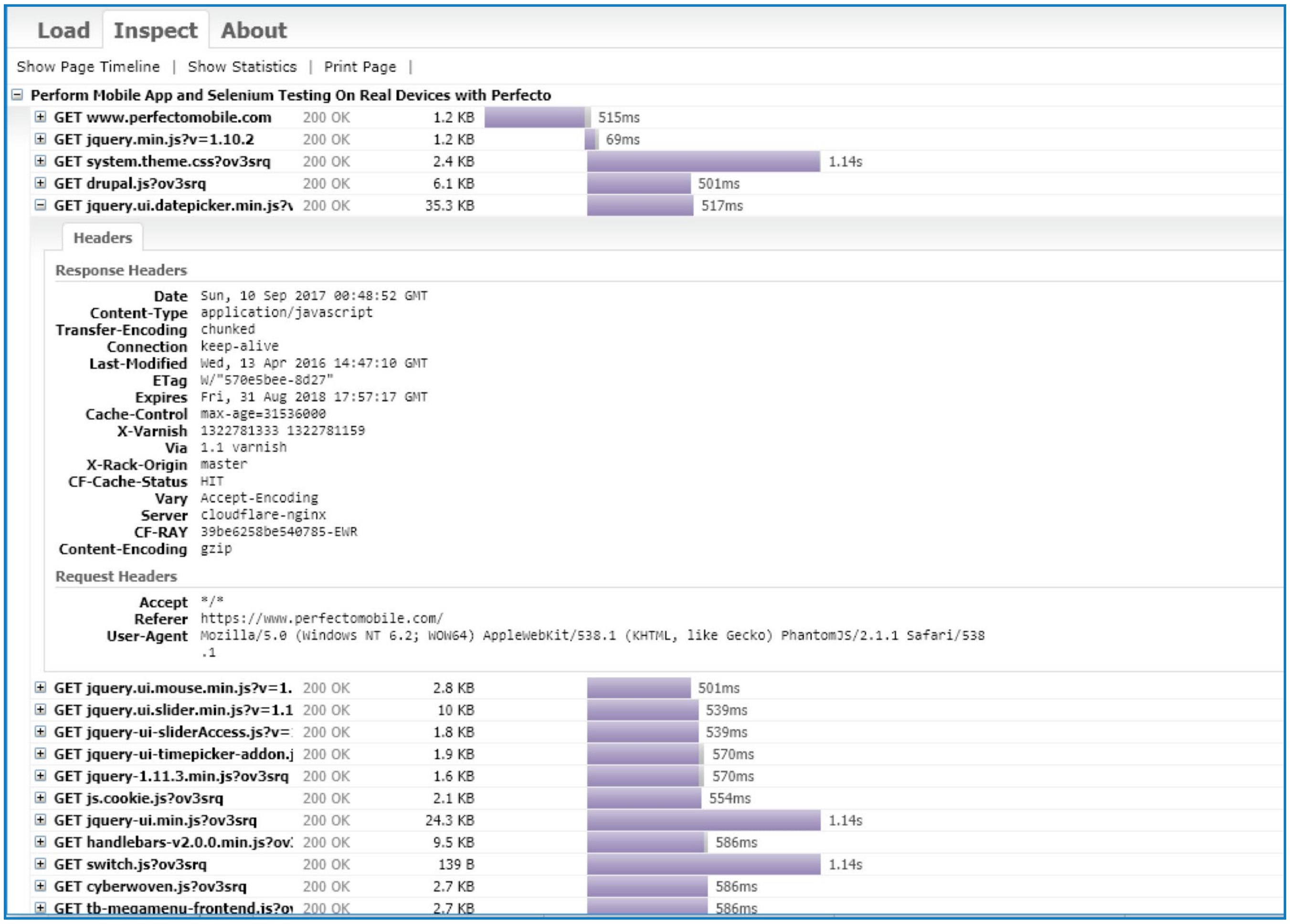
- **Limited UI testing, text truncations, viewport look and feel**
- **No E2E testing tool – can't cover efficiently complex test scenarios as well as objects operations (clicks, swipes, etc.)**
- **Multi-platform testing including mobile are not the goal for headless browsers**
- **Parallel testing and automation at scale aren't supported**

```
var page = require ('webpage').create();
        page.viweportSize = { width: 1024, height: 768 };
page.clipRect = { top: 0, left: 0, width: 1024, height: 768 };
page.open ('http://www.perfectomobile.com', function (status)
{
console.log("Status: " + status);
if(status === "success");
        page.render('Perfecto.png')'
}
phantom.exit();
});
```

Fig. 9: Page Rendering Code Snippet Using PhantomJS (Source: Perfecto)

2. Optimize Performance and Security through HAR File Analysis

Leveraging a HAR file to analyze the network traffic of your RWD site serves an important aspect regarding security and performance of your site. Teams can generate quite fast such a network traffic file, and analyze the various requests and everything that goes through your backend services and more.



Load | Inspect | About

Show Page Timeline | Show Statistics | Print Page |

⊟ Perform Mobile App and Selenium Testing On Real Devices with Perfecto

⊞ GET www.perfectomobile.com	200 OK	1.2 KB	515ms
⊞ GET jquery.min.js?v=1.10.2	200 OK	1.2 KB	69ms
⊞ GET system.theme.css?ov3srq	200 OK	2.4 KB	1.14s
⊞ GET drupal.js?ov3srq	200 OK	6.1 KB	501ms
⊟ GET jquery.ui.datepicker.min.js?v	200 OK	35.3 KB	517ms

Headers

Response Headers

Date	Sun, 10 Sep 2017 00:48:52 GMT
Content-Type	application/javascript
Transfer-Encoding	chunked
Connection	keep-alive
Last-Modified	Wed, 13 Apr 2016 14:47:10 GMT
ETag	W/"570e5bee-8d27"
Expires	Fri, 31 Aug 2018 17:57:17 GMT
Cache-Control	max-age=31536000
X-Varnish	1322781333 1322781159
Via	1.1 varnish
X-Rack-Origin	master
CF-Cache-Status	HIT
Vary	Accept-Encoding
Server	cloudflare-nginx
CF-RAY	39be6258be540785-EWR
Content-Encoding	gzip

Request Headers

Accept	*/*
Referer	https://www.perfectomobile.com/
User-Agent	Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/538.1 (KHTML, like Gecko) PhantomJS/2.1.1 Safari/538 .1

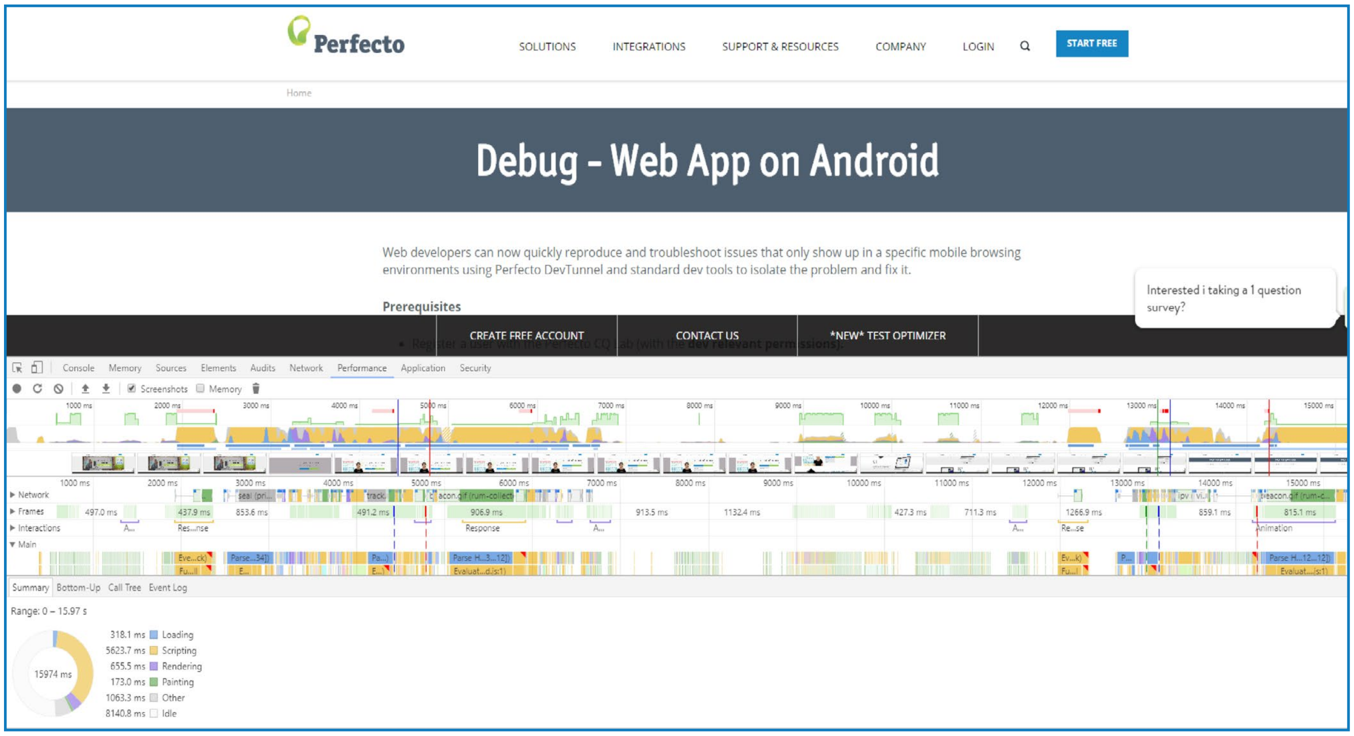
⊞ GET jquery.ui.mouse.min.js?v=1.	200 OK	2.8 KB	501ms
⊞ GET jquery.ui.slider.min.js?v=1.1	200 OK	10 KB	539ms
⊞ GET jquery-ui-sliderAccess.js?v=	200 OK	1.8 KB	539ms
⊞ GET jquery-ui-timepicker-addon.j	200 OK	1.9 KB	570ms
⊞ GET jquery-1.11.3.min.js?ov3srq	200 OK	1.6 KB	570ms
⊞ GET js.cookie.js?ov3srq	200 OK	2.1 KB	554ms
⊞ GET jquery-ui.min.js?ov3srq	200 OK	24.3 KB	1.14s
⊞ GET handlebars-v2.0.0.min.js?ov:	200 OK	9.5 KB	586ms
⊞ GET switch.js?ov3srq	200 OK	139 B	1.14s
⊞ GET cyberwoven.js?ov3srq	200 OK	2.7 KB	586ms
⊞ GET tb-megamenu-frontend.js?o	200 OK	2.7 KB	586ms

Fig. 10: Webpage HAR File Network Traffic Example (Source: Perfecto)

Tools like PhantomJS would be able to generate such file, as well as other open-source tools. The use of these tools by developers, performance engineers and test engineers can increase the quality productivity and insights to reduce performance bottlenecks when loading the wrong resources, as well as identify security vulnerabilities through network traffic requests.

3. Use In-Browser Capabilities to Increase Test Productivity

Desktop browsers can offer today advanced capabilities around performance analysis, test coverage, logs and other useful insights that can enhance the overall coverage—take advantage of these, they're free. As demonstrated in the following screenshot taken from Chrome 62 browser, Google is offering a nice set of tools that can dovetail into your existing test automation suite.



Web developers can now quickly reproduce and troubleshoot issues that only show up in a specific mobile browsing environments using Perfecto DevTunnel and standard dev tools to isolate the problem and fix it.

Prerequisites

Interested i taking a 1 question survey?

Range: 0 – 15.97 s

15974 ms

318.1 ms	Loading
5623.7 ms	Scripting
655.5 ms	Rendering
173.0 ms	Painting
1063.3 ms	Other
8140.8 ms	Idle

Fig. 11: Google Chrome Developer Tools - Performance Tool Example (Source: Perfecto)

Among the key features, there are performance and network recording capabilities that can allow users to record a session that can then be easily analyzed from event log perspective, load timing, call tree and visual graphics with additional insights. There are other tools including coverage, memory analysis, security and certificates validity, audit of the web site to provide accessibility insights, and many more.

Create Actionable Test Reports for RWD

One of the biggest struggles for RWD dev and test teams is identifying specific bugs. Even when they do identify the bugs, reproducing them is a huge challenge. For DevTest teams to efficiently develop, test and run through the continuous SDLC, they must have a deep, actionable, side-by-side synchronized report for all digital platforms that shows which browser/OS combination was affected by the bug.

We've touched upon the importance of UI and visual validation as part of RWD test automation, but including validations of failure in reports makes a big difference in the time it takes to resolve bugs.

What should a RWD test report include?

- **A side-by-side synchronized view of test results across all the platforms you are testing your website against**
- **The ability to filter only the failed combinations (Fig. 12)**
- **Actionable information from the failed test with:**
    - Screenshots
    - Platform debug logs
    - Network logs
    - Video
    - Network traffic log (HAR file)
    - Test history and suite health
    - CI Health through dashboard
    - Link test failures to defects (e.g. Jira integration)

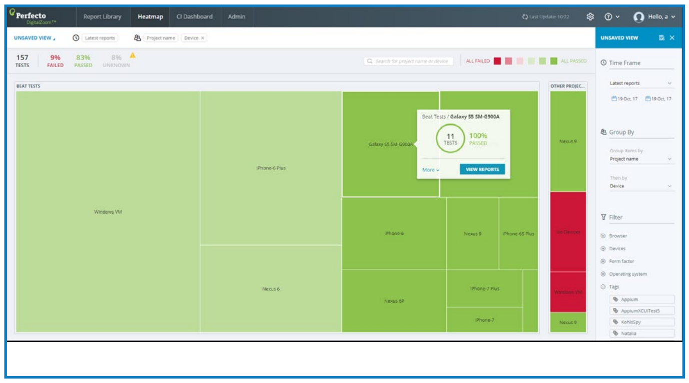


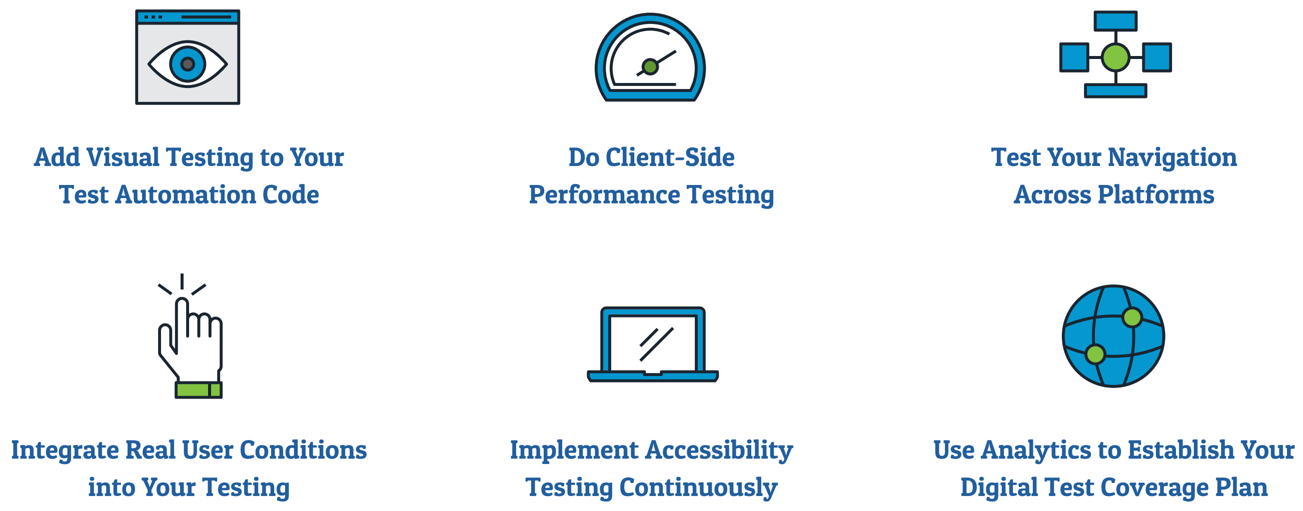
Fig. 12: Perfecto's Multi-platform Heatmap (Source: Perfecto)

A robust, detailed report gives app development, testing and agile teams the information they need to quickly troubleshoot any issue. When adding new platforms and beta versions to testing cycles, teams should be able to easily filter and examine the results of new platforms against previous tests.

Summary: Answering the RWD Challenge

Expanding your digital footprint across every browser, device and user scenario will depend on following best practices, and using the rights tools and processes.

We've discussed some of the best practices (Fig. 13), but delivering great responsive websites goes beyond that. We don't want to discount the important of the right image formats, the best CSS and other coding practices, and proper page design. These, by themselves, require a great deal of planning and execution.



Add Visual Testing to Your Test Automation Code
Do Client-Side Performance Testing
Test Your Navigation Across Platforms
Integrate Real User Conditions into Your Testing
Implement Accessibility Testing Continuously
Use Analytics to Establish Your Digital Test Coverage Plan

Fig. 13: RWD Test Plan Building Blocks (Source: Perfecto)

But when it comes to testing, we believe that a successful RWD release begins with an optimized test plan that covers the right platforms. Once a digital quality test plan is in place and teams are acitvely managing and maintaining it, you should adopt a continous quality stragety that relies on robust test automation including both visual objects analysis and DOM native object identification.

Teams that have access to a test report for all digital platforms will be able to address RWD issues, and maintain the RWD code to support new features, new platforms and other market changes as they occur.

We guarantee that if you have the right test environment, your responsive web journey will be a rewarding one for both you and your users.

About Perfecto

Perfecto's Continuous Quality Lab enables DevOps teams to accelerate development, achieve continuous testing and monitoring and drive fast feedback through actionable analytics for web, mobile & IoT apps. More than 3,000 customers rely on Perfecto’s cloud based Continuous Quality Lab as their digital apps test environment and for authoring test automation executed on real browsers, smart phones and devices under real end-user conditions. For more information about Perfecto, Perfecto, visit perfectomobile.com, join our community, or follow us on Twitter at @PerfectoMobile.