Cross-Browser Tests und die Tücken des Alltags

Nicole Charlier, akquinet tech@spree GmbH, Bülowstr. 66, 10783 Berlin, nicole.charlier@akquinet.de

Daniel Süß, akquinet tech@spree GmbH, Bülowstr. 66, 10783 Berlin, daniel.suess@akquinet.de

Abstract
Die Diversität der Endgeräte erhöht sich jeden Tag. Ein Usability Professional steht hier vor den Fragen: „Wie teste ich meine Anwendungen und wann bin ich damit fertig?“ Aufwandsprojekte und Festpreisprojekte unterscheiden sich zudem stark im zeitlichen Umfang der Evaluation, sollen aber gleichwertig im Endergebnis sein. Welche Werkzeuge brauche ich als Usability Professional um effizient im Projektablauf die Qualität meiner Anwendungen zu gewährleisten? Praxiserfahrungen aus Festpreisprojekten sollen Usability Professionals helfen, die richtigen Fragen zu stellen und sich das eigene (physische oder virtuelle) „Testlabor“ zusammenzustellen.
Wir erläutern in diesem Beitrag unseren Workflow für das Cross-Browser und Cross-Device-Testen. Dabei benennen wir relevante Entscheidungskriterien für Tools und Methoden und schließen mit der Vorstellung unseres Setup (Testlabor) sowie Erkenntnissen aus Festpreisprojekten.

Keywords
Cross device, Cross browser, Testing, Test Werkzeuge, mobile

Günstig starten

Zu Beginn jedes Projekts liegt der Schwerpunkt auf effizienten, schnell realisierbaren Machbarkeitsstudien, grafischen und technischen MockUps und den ersten Umsetzungen für Webbrowser. Dazu bieten sich zwei Tools an, welche auf den meisten Entwicklungsrechnern vorhanden sind oder schnell eingerichtet werden können: *Webbrowser mit Developer Tools* und *Gerätesimulatoren*.

Web Developer Tools

Ganz ohne Zusatzprogramme können Browserfenster verkleinert und vergrößert werden, um verschiedene Viewport-Größen zu testen. Häufig verwendete Abmessungen für den Schnellzugriff können mit hilfreichen Plugins oder den nachfolgend genannten Tools hinterlegt werden. *Web Developer Tools* sind in allen gängigen Webbrowsern wie Chrome, Firefox, Safari und Internet Explorer integriert oder über Erweiterungen wie Firebug verfügbar. Sie ermöglichen die Interpretation von JavaScript und CSS direkt im Browser zu prüfen, den Netzwerkverkehr einzusehen und erste Performancedaten zu betrachten. Die Tools können wie folgt eingeblendet werden:

- **Chrome** (Version 35.0.x):
  Mac: Anzeigen → Entwickler → Entwickler-Tools

Windows: Menü → Tools → Entwicklertools
https://developer.chrome.com/devtools/index (24.06.2014)

- **Firefox** (Version 30.0):
  Mac und Windows: a) Menü → Entwickler-Werkzeuge → Werkzeuge ein-/ausblenden
  b) Extras → Web-Entwickler → Werkzeuge ein-/ausblenden
  zusätzlich gibt es die Mozilla Responsive Design View im Firefox:
  https://developer.mozilla.org/en-US/docs/Tools/Responsive_Design_View (24.06.2014)
- **Safari** (Version 7.0.x):
  Schritt 1) Safari → Einstellungen → Erweitert → "Menü "Entwickler" in der Menüleiste anzeigen
  Schritt 2) Entwickler → Webinformationen einblenden
  https://developer.apple.com/library/safari/documentation/AppleApplications/Conceptual/Safari_Developer_Guide/Introduction/Introduction.html (24.06.2014)
- **Internet Explorer** (Version 11.0):
  Mac: Tools → "F12 Entwickler Tools"
  Windows: Extras → F12 Entwicklertools
  http://msdn.microsoft.com/en-us/library/ie/bg182326(v=vs.85).aspx (24.06.2014)

Um die korrekte Darstellung und Funktionsfähigkeit für eine möglichst hohe Anzahl von Endkunden zu erreichen, sollten so viele Plattformvarianten wie möglich auf Kompatibilität überprüft werden, um gegebenenfalls Anpassungen bzw. Polyfills (http://remysharp.com/2010/10/08/what-is-a-polyfill/) umzusetzen. Die Kombinationen wachsen stetig, sind doch Betriebssystem, Browsermodell und Browserversion zusammen wichtige Faktoren. Auf den meisten Betriebssystemen kann standardmäßig von jedem Browsermodell nur eine Version gleichzeitig installiert sein. Updates überschreiben oder aktualisieren die vorherige Installation. Welche Möglichkeiten bieten sich, um lokal mehrere Versionen bereitzustellen? Zum einen können portable Browservarianten neben einer Installation verwendet werden. Diese sind nach dem Entpacken vollständig lauffähig und einer installierten Anwendung gleichwertig. Unter dem Stichwort „portable apps“ oder „pen drive apps“, die Bezeichnung resultiert von USB-Sticks, gibt es heutzutage eine Vielzahl von Programmen, beispielsweise die marktführenden Webbrowser sowie bekannte Grafik- und Bürosoftware. Virtuelle Maschinen (VM) sind eine zweite Möglichkeit, um eine Kombinationen aus Betriebssystemen und Webbrowser bereitzustellen. Entweder installieren wir uns diese von Grund auf selber oder greifen auf VM zurück, die von bekannten Softwareherstellern (bspw. http://www.modern.ie) zur Verfügung gestellt werden.

Gerätesimulatoren

Neben den Web Developer Tools sind *Gerätesimulatoren* in der ersten Projektphase ideal geeignete Werkzeuge. Außer dem Browserverhalten kann darin das gerätespezifische Verhalten annähernd beurteilt werden. Live Prüfen und auch live Verändern sind weiterhin möglich. Chrome bettet Simulatoren für mobile Geräte in die Web Developer Tools ein (https://developers.google.com/chrome-developer-tools/docs/mobile-emulation). Diese gehen über einfache Viewport- und User Agents-Veränderung hinaus und melden der Webseite die Gerätegröße, Pixeldichte, Lagesensoren, Geo-Daten und bietet eine Touch-Simulation. Chrome kann sich also als echtes Gerät ausgeben. Dieses Feature enthält im Moment nur dieser Browser. Zum Aktivieren öffnen wir die Entwickler Tools, blenden mit der „Escape“ Taste das untere Fenster mit Konsole und Emulator ein. Viele Voreinstellungen ermöglichen einen schnellen Einstieg in das Simulationsverhalten. Die einzelnen Werte können jedoch manuell angepasst werden.

Ein Simulator für mehrere iPhone- und iPad-Varianten ist in der Entwicklungsumgebung XCode unter Mac OS integriert. Auch hier sind Touch- und Rotationseingaben möglich, um das Verhalten der Webseite zu überprüfen.
Der Simulator in XCode wird wie folgt gestartet: Menu → Open Developer Tools → iOS Simulator → Gerät → [bspw. iPad Retina]

(https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide/GettingStartedwithiOSStimulator/GettingStartedwithiOSStimulator.html)

Für Android gibt es ebenfalls *Emulatoren*. Im Gegensatz zu den erwähnten Simulatoren, sollen sich diese genau wie die native Plattform verhalten. Dies ist für native Anwendungen ein wichtiger Faktor beim Testen.
Der Emulator von Google bietet kein Multi-Touch und nur begrenzte Unterstützung von Hardwarekomponenten wie der Kamera. Es steht initial auch nur der Standard Browser zur Verfügung, da Google auf den PlayStore verzichtet. Die Installation von Chrome, Firefox und Co gestaltet sich als Herausforderung und die Paketdateien enthalten nicht immer den vollen Funktionsumfang (http://paul.kinlan.me/installing-chrome-for-android-on-an-emulator/). Es gibt jedoch auch alternative Emulatoren für Android.

All diese Tools können auf einem Computer installiert werden und sind für jede folgende Projektrunde oder Kundenbesuche verfügbar. Kurzfristige und schnelle Code-Änderungen oder demonstrative Umsetzungsvarianten lassen sich effektiv und effizient realisieren. Für das Cross-Browser-Szenario sind die Developer-Tools-Simulatoren aber performanter und von der Funktionalität ausgeprägter. Performance-Betrachtungen innerhalb Simulatoren sind jedoch nur bedingt aussagekräftig, da der Host-Computer meist über höhere Rechengeschwindigkeit als Smartphones oder Tablets verfügt. Der nächste Schritt in die reale Welt der Anwender erfordert einen Testaufbau mit echten Geräten.

Einzelne, echte Geräte verwenden

Echte Hardware zeigt echtes Verhalten, reales Multi-Touch und keine unbekannten Abweichungen bei der Implementierung. Die Performance basiert hier auf der wirklich vorhandenen CPU, Grafikkarte und dem Arbeitsspeicher des Gerätes. Wir können weiterhin mit den uns bekannten Browsertools die Anwendungen auf den Geräten untersuchen und verändern. Leistungsanalysen führen wir auf dem mobilen Gerät durch und erhalten realistische Daten zu Engpässen bei Skripten, Verzögerungen durch Darstellungsberechnungen oder unerwünschte Datenübertragung wegen fehlerhaftem Conditional Loading (Conditional Loading: Bedingtes Laden von Inhalten z. B. abhängig nach Viewport-Größe).
Dafür verbinden wir die Geräte per USB-Kabel mit dem Computer. Bei Android-Geräten kann der Desktop-Chrome mit dem mobilen Chrome verbunden werden. Aus Sicherheitsgründen muss der Modus „Remote Debugging“ erst am Gerät aktiviert werden. Details dazu listet Google unter den Suchbegriffen „Chrome Dev Tools Remote Debugging“ auf (https://developers.google.com/chrome-developer-tools/docs/remote-debugging). Im Desktop-Chrome kann anschließend unter der URL „chrome://inspect/#devices“ das Remote-Gerät ausgewählt und der mobile Chrome gesteuert und untersucht werden. Unter Mac OS kann Safari angeschlossene Apple Geräte remote debuggen. Safari → Develop → [Menüpunkt mit Gerätename]. Beide Anbieter zeigen in einem Fenster auf dem Hostsystem die Browseransicht des mobilen Gerätes. Interaktionen werden sowohl vom Touchscreen zum Desktop und in die andere Richtung synchronisiert.
Dieser Entwicklungsmodus eignet sich beispielsweise auch, um Performance-Betrachtungen und mögliche Optimierungen unterwegs oder beim Kunden auf einer geringen Anzahl von Endgeräten durchzuführen, ohne aufwändiges Setup aufbauen zu müssen. Nachteilig bei der Kabelverbindung ist die begrenzte Anzahl von Anschlüssen am Computer. Natürlich können über einen USB-Hub mehrere Geräte angeschlossen werden. Dennoch ist dies kein praktikables Testlabor-Setup, mit dem synchrones Testen möglich ist.

Mehrere Geräte synchronisiert Testen

Bisher hatten wir uns auf Interaktionen mit genau einem Gerät fokussiert. Für ein effizienteres Testen werden wir die Interaktionen von einem Hauptfenster gleichzeitig auf andere Geräte spiegeln. Durch synchrones Cross-Browser Testen steigt die Effizienz.
Um einen großen „Gerätepark“ vorzuhalten, zu synchronisieren und zu testen existieren zwei Möglichkeiten: *Web Services*, als entfernte Dienste in der Cloud, oder eine *lokale Umgebung* mit den Geräten vor Ort. Beleuchten wir beide Varianten etwas detaillierter.

Web Services, entfernte Dienste in der Cloud

Für Web Services spricht, dass ein schneller Zugriff auf viele Kombinationen aus Gerät, Betriebssystem sowie Browser existiert. In diesem Bereich existieren viele Anbieter, die das Cross-Browser-Testing als Tool anbieten. Wichtige Kriterien für die Auswahl eines Angebots sind die Kombinationsvielfalt, die Möglichkeit interaktiv mit entfernten Geräten zu interagieren und mehrere ausgewählte Geräte synchron zu testen. Anbieter werben dann für Cross-Browser-Tests mit Kombinationen von über 100 Browsern sowie unterschiedlichen Betriebssystem und Geräten.

Mit der Produktvielfalt steigt jedoch oft auch der Preis für den Cloud Service, während kostenfreie Anbieter meist auf wenige Browser spezialisiert sind. Auch gibt es Unterschiede im Zugriff auf URLs. Tools einiger Anbieter können nur auf öffentliche, im World Wide Web zugängliche Seiten navigieren. Wenn Auftraggeber jedoch eine Geheimhaltung bis zum offiziellen Release einfordern, reduziert sich die Anzahl der Cloud Services auf jene, welche lokal gehostete Projekte übertragen und auswerten können. Denn das öffentliche Listen von Screenshots aus diesen Tests ist bei einigen Anbietern nicht abwählbar. Der Sicherheitsaspekt kann also ein wichtiges Auswahlkriterium sein.

Die Anforderung nach Interaktion und einer Fernsteuerung der entfernten Geräte schränkt die Dienste weiter ein. So gibt es z. B. für die Testkombinationen Screenshots zum manuellen Vergleich, Videoaufzeichnungen von allen Testgeräten und/oder Layoutvergleiche mit automatischer Analyse von Differenzen. Erwähnt sei, dass nicht jeder Cloud Service echte Hardware für die Tests bereitstellt. Vor allem die günstigen Anbieter bieten oft nur servergehostete Simulatoren und Emulatoren, mit den bereits beschriebenen Vor- und Nachteilen. Schwierig wird die Beurteilung der Performance von entfernten Geräten. Besonders auffällig wird es, wenn per Bildschirmübertragung zugegriffen wird und die Ansicht latenzbedingt verzögert oder übertragungsbedingt fragmentiert aktualisiert. Eine realistische Beurteilung des Anwendererlebnisses ist hierbei kaum möglich.

Lokale Umgebung

In einer lokalen Umgebung greifen wir auf echte Geräte zu. Ohne Zeitverzögerung ist Performance wahrnehmbar, anders als bei den Web Services. Da wir die Hardware direkt vor uns haben, können wir Geräte in die Hand nehmen, reale Touch-Interaktionen, vom Single Touch bis hin zum Multi-Touch mit vier Fingern, durchführen. Das Erlebnis ist wesentlich näher am Anwender als bei entfernten Diensten. Ein weiterer Vorteil: Wenn wir eine offline-fähige Webanwendung entwickeln, können wir bei lokalen Geräten die Seite laden, die Internetkonnektivität deaktivieren und anschließend die korrekte Funktionsweise überprüfen. Die Synchronisierung mehrerer Browser übernehmen Produkte wie GhostLab, BrowserSync oder Adobe Edge, um nur einige zu nennen. Diese starten einen Webserver auf dem Hostsystem, registrieren alle per WLAN oder LAN verbundenen Klienten und übertragen die Interaktionen zwischen den Klienten. Die Steuerung erfolgt meist innerhalb der einzelnen Webseite im Browser mittels JavaScript und Endgeräte benötigen daher keine zusätzliche Software. Diese Architektur vereinfacht das Setup für weitere Testgeräte und erleichtert deren schnelle Austauschbarkeit. Die Programme bieten teilweise die Möglichkeit, lokale Daten selber via Webserver bereitzustellen oder den Webserver nur als Interaktionsvermittler (Proxy) zwischen den Geräten und einer vorhandenen Webseite zu verwenden. So kann beispielsweise der Zugriff auch auf das Webfrontend in einem

komplexen Application Server erfolgen. Hilfreich ist ein Beobachtungsmodus (watch mode), der in definierten Verzeichnissen auf Veränderungen des Quellcodes reagiert und die Klienten-Ansichten aktualisiert. Eine wichtige Entscheidung bezüglich des lokalen Geräteparks ist, ob alle benötigten selber Geräte gekauft werden oder ob ein temporäres Mieten wirtschaftlicher wäre. Die Kriterien differenzieren sicherlich zwischen den Firmen oder auch den Projekten. Einige Kriterien sind: die Häufigkeit der Verwendung, die finanziellen Aufwendungen, Anforderungen des Kunden, technische Administration der Geräte, ausreichend Räumlichkeiten für ein Testlabor. Vor dem Kauf diverser Smartphones, Tablets und Co. sollte auch die Frage geklärt werden, wie weitreichend getestet werden soll und wo Cross-Browser Testen endet. Tendiert die Entscheidung in Richtung Mietkonzept, lohnt sich ein Blick auf „Device Labs“. In Deutschland und auch Berlin existieren immer mehr Firmen, die bei sich vor Ort komplett ausgestattete Arbeitsplätze für Cross-Browser und Cross-Device-Testen vermieten. Der Wartungsaufwand wird hier gespart.

Varianten und Klassifizierung

Wir listen hier als Übersicht alle Varianten und Kriterien auf, die unsere Auswahl der Tools beeinflusst haben:

- Simulation oder Hardware
- Einzel- oder Multi-Browsertest
- Lokal Ausführung oder Cloud Service
- Öffentliche oder private Auswertung der Daten
- Software kostenfrei oder kostenpflichtig
- Geräte kaufen, vor Ort mieten oder virtuell im Cloud Service nutzen

Warum haben wir uns für diese Werkzeuge entschieden?

Viele unserer Kunden fordern eine Geheimhaltung der Projekte, bis sie von ihnen selbst offiziell veröffentlicht werden. Daher setzen wir verstärkt auf unser lokales Testlabor. Die Kosten-Nutzen-Abwägung führt jedoch zu einer Kombination aus lokalem Setup und Cloud-Diensten. Wir haben uns hierbei entschieden, nicht jedes Gerätemodell vorzuhalten, sondern beschaffen primär Testsysteme mit einem hohen Marktanteil bezüglich Endgerät, Betriebssystem und Browser. Exotische Modelle werden gegebenenfalls gemietet oder über Web Services getestet.

Öffentliche Projekte erleichtern das breite Testen durch die Verwendung von Cloud-Diensten. Bei den lokalen Synchronisierungsprogrammen haben wir noch nicht das „komplett zufriedenstellende Produkt“ gefunden. Bei interaktiven Tests werden hierbei nicht alle Klicks oder Touch-Eingaben abgeglichen. Häufig tritt dies bei Buttons mit JavaScript-Aktionen auf. Aktuell setzen wir daher sowohl ein kommerzielles als auch ein freies Produkt ein.

Erfahrungen aus Festpreisprojekten

Natürlich beeinflussen die Anforderungen des Kunden das Testsetup bei jedem Projekt. Erfahrungen aus unseren Festpreisprojekten zeigen schnell, dass es auf die richtige Kommunikation mit den Kunden und die vollständige Anforderungsanalyse ankommt, um die zu unterstützenden Geräte zu identifizieren und zu definieren. Wir haben vor dem synchronen Testen auf unterschiedlichen Browsern und Geräten nur die Möglichkeit gehabt, einzelne Geräte nacheinander zu testen. Mit dem synchronen Testen haben wir nun einen zeitlichen Vorteil, der uns in geringerer Zeit die gleichen qualitativen Ergebnisse sicher stellt. Es steigt zwar immer noch der Aufwand für die Tests, je nachdem wie umfangreich die Browser und Geräte definiert wurden, aber bei der stetig steigenden Geräteanzahl auf dem

Markt haben wir einen Testablauf, der uns darin unterstützt. Es darf hierbei aber nicht der Anpassungsaufwand an der Anwendung nach der Testauswertung vergessen werden bezüglich Konzeption und Entwicklung. Werden die Anforderungen zu "weich" spezifiziert, können beim Kunden Erwartungen entstehen, die während eines Festpreisprojektes nicht gehalten werden können. Daher sind die Absprachen bezüglich Verbindlichkeit bei der Geräteunterstützung wichtig. Hierbei gibt es mehrere Möglichkeiten, die der Kunde nennen kann. Der Kunde möchte seine Mitarbeiter erst beim Release der Software mit Smartphones ausstatten. In diesem Fall wünschte der Kunde eine Geräteempfehlung, die im angemessenen Kosten-Nutzen-Verhältnis steht, und wir konnten gezielt Optimierungen für dieses Gerät umsetzen. Bei Bring-Your-Own-Device (BYOD) gibt es diesen Luxus nicht. Oft besteht aber die Möglichkeit, zum Beispiel das Alter der Geräte oder bestimmter Browser einzugrenzen. Möchte der Kunde jedoch eine Anwendung, die weitreichend optimal verfügbar ist, bedeutet es im schlimmsten Fall eine hohe Abwärtskompatibilität anzubieten. Mit dieser Anforderung steigen sehr schnell die Aufwände in Konzeption, Implementierung und Qualitätssicherung.

Der Vorteil bei einem Testlabor ist, das die Geräte vor Ort kontinuierlich getestet werden können. Die mit dem Kunden abgesprochenen zusätzlichen Geräte können dann entweder virtuell getestet oder nach Kauf einfach zu den Testgeräten hinzugesteckt werden. Geräteparks des Kunden sind so unter Kontrolle und die Testaufwände können besser vor dem Angebot kalkuliert werden.

Diese Möglichkeiten sollten bei der Anforderungsanalyse betrachtet und benannt werden, damit daraus später im Projekt keine Selbstverständlichkeitsanforderung wird.

Fazit

Unser Workflow vom Cross-Browser-Testing unterteilt sich in mehrere Phasen und die Methoden verändern sich mit dem Voranschreiten des Projekts. Lokale Entwicklungswerkzeuge und Simulatoren erlauben uns eine schnelle, effektive Umsetzung von ersten Ergebnissen. Sobald die Entwicklung voranschreitet und die Anforderungen spezifischer sind, testen wir regelmäßig mit echten Endgeräten. Nur mit diesen können wir die Nutzererfahrung hinsichtlich Interaktion und Performanz realistisch beurteilen. Für effektivere Tests verwenden wir unser Testlabor mit einem Gerätepark. Wenn möglich verwenden auch die Entwickler das Testlabor während der Implementierung, um schnelle Rückmeldung bezüglich der Darstellung, Funktion und Bedienung zu erhalten. Noch eine subjektive Beobachtung zum Schluss: Der echte Gerätepark schafft, im Gegensatz zu zwei Bildschirmen mit vielen Simulatorfenstern, bei den Kunden ein höheres Vertrauen in unsere Arbeit. Die Kunden sehen die Tests auf realen Smartphones und Tablets und können sogar vor Ort selber testen.

Viten der Autoren

Nicole Charlier

Nicole Charlier ist User Experience Engineer bei der akquinet AG in Berlin. Sie leitet dort das Competence Center für UX. Dabei ist Nicole verantwortlich für Inhouse Usability Projekte im Bereich Analyse, Konzeption, Bewertung und Optimierung von Softwareprodukten. Nicole ist Mitglied der German UPA und Mitglied bei den Arbeitskreisen Barrierefreiheit und Inhouse Usability. Als Moderatorin der XING Gruppe User Experience Berlin fördert sie den Austausch von Erfahrungen, Ideen, Fragen etc. zum Thema UX: www.uxberlin.org. Sie ist regelmäßig auf Fachkonferenzen als Referentin zum Thema User Experience vertreten. Details finden Sie auch auf XING.

Daniel Süß

Daniel Süß ist Softwareentwickler mit dem Schwerpunkt Frontend-Entwicklung bei der akquinet AG in Berlin. Er initiierte und betreut das interne Testlabor für Cross-Browser und Cross-Device-Testing.