Inhoud

Hoofdstuk 1

Leren werken met Mobile HTML5, CSS3 en JavaScript API's

Al jaren heb ik een hekel aan die vreselijke oudere versies van Internet Explorer. Die browsers zitten vol met fouten. Ze vertonen allemaal steeds weer dezelfde mankementen[1]. We wisten allemaal dat IE6 waardeloos was, maar de rest was net zo waardeloos. Toen we eenmaal wisten hoe we met polyfills konden werken voor IE6, hadden we de oplossing.

1 IE6 was het nieuwste van het nieuwste toen het in 2001 op de markt kwam. Deze browser had bijna het alleenrecht op de browsermarkt en er was weinig concurrentie. Daarom werd deze ook nooit bijgewerkt.

Browsergrillen

In het mobiele landschap bestaan er ook fouten, maar er zitten ook fouten in nieuwere, verschillende manieren die continu veranderen. Diverse browserversies op verschillende apparaten ondersteunen wel veel nieuwe functies, maar doen dit op andere manieren. Het kan ook zijn dat ze een functie ondersteunen die mogelijk niet bruikbaar is. Zo kan een modern apparaat bijvoorbeeld localStorage ondersteunen. Naar sommige apparaten die localStorage ondersteunen, is het mogelijk te schrijven. Zelfs als u met de browser van localStorage kunt lezen, kan dit lang duren en de prestaties belemmeren. En zelfs als u met de browser meestal naar een apparaat kunt schrijven, is het mogelijk dat localStorage zelf de limiet heeft bereikt.

We kunnen niet alle kuren (*quirks*) in alle browsers voor besturingssystemen en apparaten hier bespreken. Zelfs als ik alle nukken kende (en dat doe ik niet), zou ik een boek kunnen vullen, dat al verouderd zou zijn voordat ik klaar was met schrijven. Ook dit boek is eigenlijk al verouderd. Het landschap verandert continu. Het is onmogelijk een boek uit te geven dat volledig up-to-date is. Zodra het boek naar de drukker gaat, of zelfs zodra je een hoofdstuk hebt voltooid, is het landschap veranderd. Hoewel sommige van de in dit boek genoemde browsers, telefoons en sites al verouderd zijn, zijn de procedures die in dit boek worden besproken, voor de komende jaren van toepassing. Het uitgangspunt voor dit boek is als u de gedragscode en handelswijzen volgens de standaarden gebruikt, dat uw code bij uw huidige en alle toekomstige apparaten werkt.

Ik heb browserondersteuning voor functies opgenomen, maar niet het gebrek aan ondersteuning voor browserfuncties, omdat verwacht wordt dat alle browsers straks ondersteuning zullen bieden. Een rarigheid in een browser van vandaag kan morgen zijn opgelost. Daarom moet u de functies voordat u ze gebruikt, detecteren en uitproberen om na te gaan of u de ondersteunde functie echt kunt gebruiken.

Dit boek maakt gebruik van apparaat-, besturingssysteem- en browseronafhankelijke opmaak en geen JavaScript-bibliotheken. Ik gebruik geen bibliotheken en codeer in standaard-JavaScript om ervoor te zorgen dat u de echte code leert. Door te coderen in standaard-JavaScript heb ik hopelijk enige verwarring weggenomen over of een methode systeemeigen is of een raamwerkmethode.

Dit wil niet zeggen dat u geen bibliotheken mag gebruiken. In tegendeel! Opensourcebibliotheken zijn soms de beste locaties voor informatie over browsergrillen. Opensourcebibliotheken hebben honderden, soms wel duizenden,

medewerkers. Deze medewerkers helpen bij het ontwikkelen, testen en verbeteren van apparaten en bibliotheken. Deze medewerkers melden ook bugs aan browserleveranciers om hen te laten weten wat niet goed werkt, zodat deze bugs uit de toekomstige browserversies worden gehaald.

Populaire opensourcebibliotheken en HTML5-JavaScript API-polyfills vormen de beste bronnen voor het snel ontdekken van diverse browsernukken en oplossingen. Ze moeten worden beschouwd als een belangrijk onderdeel van uw serie hulpprogramma's voor ontwikkelaars. Zelfs als u ze niet gebruikt, moet u de broncode lezen om op de hoogte te zijn van de mobiele browserbugs die zijn ontdekt.

De beste manier om HTLM5, CSS3 en de bijbehorende JavaScript API's te leren terwijl u erover leest, is door te coderen. Laten we gaan coderen.

Mobiel HTML5-spelletje

Ik heb HTML5 en CSS3 geleerd door een webapplicatie te ontwikkelen voor een enkele mobiele browser en kijken hoe ver ik kon gaan. Mijn eerste codeerpoging in CSS3 was een mashup van twitter en de basketbalhoofdklasse, genaamd *Pickleview*, geschreven in het weekend in 2007 waarin de eerste iPhone uitkwam. Op dat moment was Safari voor de iPhone de meest geavanceerde browser op de markt (op Opera na misschien). Door deze te programmeren voor een enkele browser, hoefde ik me geen zorgen te maken over IE6, IE7 of Firefox 2 (Chrome bestond nog niet). Dit was de situatie op internet in 2007.

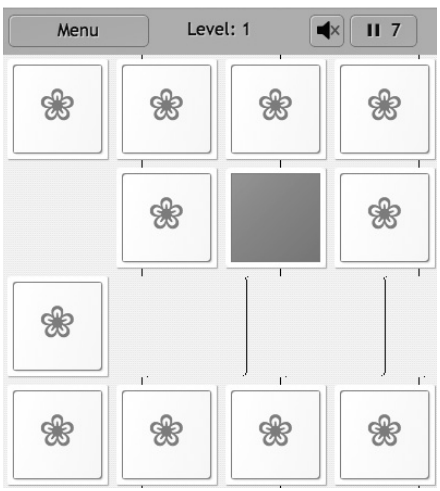
In 2010 deed ik nog een codeerpoging met de meest moderne HTML5 en CSS3 in een enkele browser. Ik maakte samen met vrienden een geheugenspel met animaties, opslagruimte, offline functies en elke nieuwe functie die ik kon vinden in Chrome 12 op een desktopcomputer, maar niet in Safari 3.1 voor mobiel gebruik. Door een enkele browser te gebruiken en alle nieuwe technologieën toe te passen die ik kende, kon ik nieuwere HTML5-, CSS3- en JavaScript-modules leren coderen die nog niet bruikbaar waren bij de productie omdat ze verouderde browsers moesten kunnen ondersteunen. Een aantal browsers hadden zich in 2010 enorm ontwikkeld ten opzichte van 2007. Andere browsers (zoals IE) iets minder.

In 2013 ondersteunden de meeste HTML5 en CSS3. Als ontwikkelaars worden we belemmerd omdat we oudere desktopbrowsers moeten kunnen ondersteunen, zoals Internet Explorer 9 en ouder. Op mobiele apparaten hebben we onze eigen 'IE6'. We worden belemmerd door toekomstige telefoons, en in

zekere mate door smartphones met Android 2.3. Maar zelfs de toekomstige telefoonbrowsers en Android 2.3 ondersteunen beide moderne functies.

Wanneer u HTML5, CSS3 en de bijbehorende JavaScript API's wilt leren, moet u tijdelijk de oudere browsers vergeten. Samen leren we wat er mogelijk is met deze nieuwere technologieën. Ik heb het merendeel van de functies die door veel moderne browsers worden ondersteund, in de codevoorbeelden in dit boek geplaatst.

CubeeDoo, zoals weergegeven in afbeelding 1.1, is een volledig front-end gecodeerd geheugenspel. Ik zal door het hele boek codevoorbeelden uit dit spel gebruiken, samen met een replica van een systeemeigen iPhone-instellingen-applicatiereplica (zoals in afbeelding 9.3). Het spel wordt opgemaakt met HTML5-elementen. Sommige thema's beschikken over bijbehorende pictogrammen gemaakt met gegenereerde inhoud. CSS-transformatie, -transitie en -animatie kunnen samen met verlooptinten, afgeronde hoeken en andere CSS-functies worden gebruikt om de *look en feel* van het spel te creëren. Het spel bevat ook SVG, JSON, de verouderde, maar mobiel ondersteunde webSQL, localStorage, sessionStorage, gegevenskenmerken, HTML5-formulieren, audio, mediaquery's en gegevens-URI's.



Afbeelding 1.1 *Schermopname van het geheugenspel CubeeDoo.*

De code in dit boek gebruikt geen enkel type raamwerk. Zoals eerder vermeld, is alles handmatig gecodeerd in standaard-JavaScript, HTML5 en CSS. Het doel is om u de echte API's te leren en niet de polyfills. Tijdens de productie zult u waarschijnlijk de polyfills willen gebruiken, maar om deze goed te kunnen gebruiken, moet u weten hoe polyfills werken. Met dit boek leert u dat.

Dit boek bespreekt CSS3, HTML5 en de bijbehorende API's. De focus is het leren van technologieën in een mobiel landschap. We wonen in een mobiele wereld, maar er is geen 'mobiel internet', alleen het gewone internet. Maar als u zich alleen op de desktopbrowser richt, is het mogelijk dat de webappversie die u maakt, niet werkt voor het merendeel van de personen dat internet alleen mobiel gebruikt via mobiele apparaten. Bovendien bent u dan alleen bezig voor de laagste gemene deler van oudere Internet Explorer-versies.

Breng nooit een applicatie in productie die slechts in één browser werkt. Wilt u echter technologieën leren die nog in de kinderschoenen staan, dan kunt u door de oudere browsers te negeren, leren, uzelf uitdagen, buiten het geijkte stramien denken en de top bereiken. Gebruik alles wat u in dit boek leert om in een enkele browser zoveel mogelijk uit te proberen op codeergebied. Dan weet u weer waarom u zo van webontwikkeling houdt.

Alles wat u nodig hebt, is een browser, een IDE en een beetje tijd.

Hulpprogramma's voor ontwikkelaars

Voordat u begint met het ontwerpen van uw eerste mobiele webapplicatie, wilt u uw ontwikkelomgeving instellen met de beste hulpprogramma's die beschikbaar zijn. Goed nieuws! U hebt deze hulpprogramma's al.

Het enige wat u, behalve dit boek, nodig hebt, zijn een computer met een teksteditor en een browser. U hebt zelfs geen telefoon nodig, hoewel een mobiel apparaat wel zeer handig zou zijn.

Teksteditor

U moet in een platteteksteditor ontwikkelen of een *geïntegreerde ontwikkelomgeving* (IDE). Een IDE is software die over het algemeen bestaat uit een teksteditor, een hulpprogramma voor foutopsporing en andere functies of plug-ins, zoals een File Transfer Protocol (FTP), die u mogelijk nodig hebt om de applicatie te maken. Veel mensen hebben een favoriete IDE. Kies uw eigen favoriete IDE. Mijn voorkeur gaat uit naar Sublime Text, maar u kunt ook TextMate, Dreamweaver, Eclipse of WebStorm gebruiken. U hebt alleen een platteteksteditor nodig, waarbij u zult ontdekken dat u met IDE het ontwikkelproces eenvoudig kunt organiseren en stroomlijnen. Het is raadzaam een IDE te gebruiken en deze goed onder de knie te krijgen. IDE's kunnen heel krachtige hulpmiddelen zijn die ervoor zorgen dat u veel plezier krijgt in ontwikkelen.

Browser

U hebt ook een browser nodig. Ik ontwikkel het liefst in Chrome Canary, de bètaversie van Google Chrome. Deze browser heeft mijn voorkeur vanwege de hulpprogramma's voor foutopsporing. Alle moderne browsers hebben hulpprogramma's voor foutopsporing, maar het hulpprogramma voor foutopsporing van Chrome is een van de beste en het hulpprogramma voor foutopsporing van Canary geeft me inzicht en toegang tot alle nieuwe functies en opties voordat ze in een nieuwe browser terechtkomen.

Als u geen Apple-computer hebt, kunt u niet zomaar systeemeigen aanraakapplicaties voor de iPhone, iPad of iPod maken. Hebt u geen Windows 8, dan is het moeilijk om applicaties te ontwikkelen die formeel bekend staan als de Metro-stijl applicaties. Er is niets aan de hand! Voor datgene wat u hier leert, hebt u alleen een moderne browser nodig. Het besturingssysteem of apparaat is niet van belang. U kunt alle voorbeelden in dit boek testen op telefoons en tablets met Windows, Unix en Android en op Macs.

Uw IDE en desktopbrowser zijn uw belangrijkste hulpmiddelen voor de ontwikkeling van mobiele webapplicaties. Uw mobiele applicatie wordt vooraf bekeken en tijdens het ontwikkelproces wordt deze continu op fouten gecontroleerd. Er zijn eigenschappen die uw desktopbrowser niet kan emuleren, zoals de nauwkeurigheid van de weergave op mobiele apparaten, de JavaScript-prestaties, de geheugen- en bandbreedtelimieten en de API-beschikbaarheid. Deze verschillen zijn echter te overbruggen met andere hulpmiddelen en door direct te testen op echte of virtuele apparaten.

Hoewel u fijn uw favoriete browser kunt gebruiken om te ontwikkelen, moet uw toolkit meerdere browsers bevatten om testen te kunnen uitvoeren. U hebt toegang nodig tot Internet Explorer om de omgeving van uw Windowstelefoon makkelijker te kunnen testen. Met Safari of Google Chrome kunt u testen uitvoeren voor Android, Bada, Blackberry en iOS. U hebt ook Firefox voor Gecko-apparaten nodig. Opera is momenteel nodig voor het testen van alle apparaten waarop de rendering engine Presto draait, maar omdat Opera Mobile 14 op Chromium is gebaseerd en de meest recente Opera en Chrome op Blink[2]. De browsers die u nodig hebt om te ontwikkelen, moeten worden bijgewerkt om geschikt te zijn voor het landschap waarin u ontwikkelt.

Zorg dat u Safari hebt gedownload als u op een Mac werkt, of de nieuwste Internet Explorer als u in Windows werkt. Download ook Chrome, Firefox en

2 Blink is een aftakking van het WebCore-component van WebKit gemaakt bij revisie 147503. Het is de browserengine in Chrome vanaf versie 28, Opera vanaf versie 15 en andere op Chromium gebaseerde browsers die steeds geavanceerder worden.

Opera naar uw apparaat, zelfs als u met Unix werkt. U kunt ook *builds* van Chrome Canary, Aurora, Opera Next en WebKit Nightly downloaden om testen uit te voeren in de volgende versies van de grootste browsers. Dit zijn de huidige desktopbrowsers op het moment dat ik dit boek schrijf, maar het landschap blijft veranderen.

Hulpprogramma's voor foutopsporing

Browsers bevatten hulpprogramma's voor ontwikkelaars. Hulpprogramma's voor ontwikkelaars zijn ingebouwde browserfuncties waarmee u uw broncode kunt controleren en de fouten erin kunt opsporen. Met de hulpprogramma's kunt u de DOM (Document Object Model) aanpassen, de JavaScript-code bewerken en fouten erin opsporen, CSS bewerken en fouten erin opsporen, bronaanvragen analyseren en de prestaties van live webinhoud en webapplicaties testen.

Hulpprogramma's voor ontwikkelaars zijn over het algemeen verborgen omdat de meeste gebruikers, in tegenstelling tot ontwikkelaars, deze browserfuncties niet gebruiken. Mobiele browsers hebben vaak foutopsporingsmogelijkheden in de apparaatbrowser. Deze beperkte hulpprogramma's voor foutopsporing zijn gewoonlijk beschikbaar via de interface voor apparaatinstellingen. Hoewel foutopsporing op apparaatniveau wel mogelijk is, is het veel makkelijker om fouten in applicaties op te sporen met de veel robuustere hulpprogramma's die u op uw desktop vindt.

Hulpprogramma's voor foutopsporing op desktops

Als u al enige tijd websites ontwikkelt, bent u waarschijnlijk bekend met Firebug[3], F12, Web Inspector en/of DragonFly. Firebug is een Mozilla-uitbreiding. F12, Web Inspector en DragonFly worden respectievelijk geleverd bij Internet Explorer, Chrome/Safari en Opera. Met al deze hulpprogramma's voor ontwikkelaars kan CSS, HTML, DOM en JavaScript van een website worden bewerkt en bewaakt en kunt u de fouten erin opsporen. Ook is het mogelijk functies te analyseren, zoals HTTP-aanvragen, lokale opslag en het geheugengebruik.

Firebug is beschikbaar via http://getFirebug.com. De hulpprogramma's voor ontwikkelaars van Safari zijn te vinden in het menu **Ontwikkel**, maar moeten beschikbaar worden gemaakt via **Voorkeuren**, **Geavanceerd** door de optie **Ontwikkel-menu in menubalk tonen** te kiezen. In Chrome kunt u de

3 Firefox wordt geleverd met hulpprogramma's voor ontwikkelaars, maar de meeste ontwikkelaars gebruiken Firebug, een Firefox-invoegtoepassing.

hulpprogramma's voor ontwikkelaars openen via **Google Chrome aanpassen en beheren**, **Extra**, **Hulpprogramma's voor ontwikkelaars**.

U kunt ook hulpprogramma's voor foutopsporing van Chrome, Safari, Firebug en Opera openen met Command+Option+I of Control+I. F12 en Firebug kunnen ook worden geopend door op F12 te klikken. Deze hulpprogramma's zijn het beste hulpmiddel van de browser om fouten in CSS, JavaScript en HTML op te sporen.

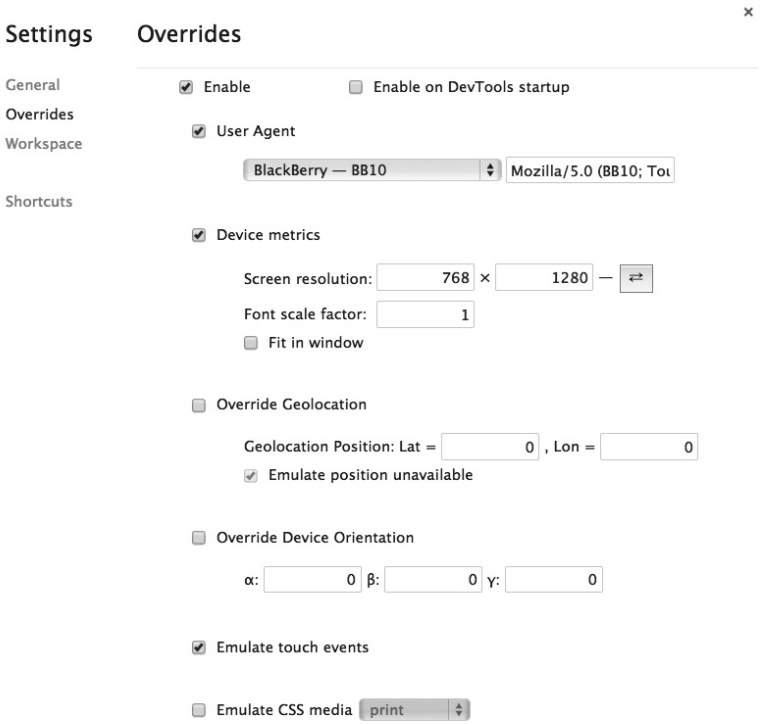
U raakt vertrouwd met de Web Inspector, de foutconsole, en de gebruikers-agentswitcher. Met deze hulpprogramma's voor foutopsporing kunt u de CSS, HTML, JavaScript, DOM en kopteksten van een webpagina controleren. Of u nu Web Inspector, Firebug, DragonFly, F12, hulpprogramma's voor ontwikkelaars of een combinatie hiervan gebruikt, zorg dat u weet over welke hulpprogramma's voor foutopsporing u beschikt. U zult hiervan zeker veel gebruik maken.

Waarschijnlijk hebt u al een aantal jaar ervaring met hulpprogramma's voor foutopsporing voor desktopapplicaties. Daarom zullen we hierop niet te diep ingaan. Als u ze een jaar of vijf gebruikt, is de kans groot dat u slechts een klein deel kent van alle geweldige functies die een dergelijk programma te bieden heeft. Ik raad u aan om u eens goed in het programma te verdiepen door overal op te klikken en te rechtsklikken. In hoofdstuk 14 wordt het tabblad Timeline van de hulpprogramma's voor ontwikkelaars besproken.

Viewport van mobiele apparaten

Wanneer u de viewport van mobiele apparaten wilt nabootsen, hoeft u het browservenster van de desktop alleen groter of kleiner te maken overeenkomstig de viewport van het mobiele apparaat die u wilt testen. De viewport van de desktopbrowser is het browservenster. Op mobiele apparaten is de viewport wat u ziet, maar dat is niet altijd alles wat op het scherm wordt weergegeven. Door het venster aan te passen komt u echter vaak dicht genoeg in de buurt voor de meeste testen die u moet uitvoeren.

Wanneer u uw browser handmatig aanpast, kunt u verschillende groottes krijgen. Op het tabblad **Overrides** van het venster **Settings**, dat wordt afgebeeld in afbeelding 1.2, bieden de hulpprogramma's voor ontwikkelaars van Chrome meerdere vooraf ingestelde groottes voor het apparaat. Open het instellingenvenster van Web Inspector door te klikken op de knop rechtsonder van de hulpprogramma's voor ontwikkelaars.



Afbeelding 1.2 *Het tabblad Overrides onder Settings van de hulpprogramma's voor ontwikkelaars van Chrome.*

Wanneer u een apparaat selecteert in het keuzemenu **User Agent**, wisselt Chrome de gebruikersagent om voor de geselecteerde gebruikersagent van het apparaat en maakt een viewport binnen het browservenster dat de grootte heeft van het geselecteerde apparaat. Zo krijgt u een browserviewport die dezelfde grootte heeft als de viewport van het geselecteerde apparaat.

Als uw apparaat niet in de lijst staat, voert u eenvoudigweg de breedte en de hoogte in van het apparaat in de twee invoervakken onder Device metrics. U kunt schakelen tussen de afmetingen van een staande en liggende modus door rechts van de Device metrics op de wisselknop te klikken. Ga eens naar http://beta.screenqueri.es om de exacte schermformaten van apparaten te bekijken. U kunt ook emulatie van aanraakgebeurtenissen inschakelen of *thumbs.js* als een TouchEvent-polyfill gebruiken.

Met de hulpprogramma's voor ontwikkelaars van Chrome kunt u ook de geolocatie negeren om een specifieke lengte- en breedtegraad na te bootsen, en ook al heeft uw laptop een gyroscoop, dan kunt u nog de stand van uw apparaat simuleren.

Nadat u de eerste fase van uw applicatie met uw desktopbrowser hebt ontwikkeld, kunt u deze testen op een mobiel apparaat. Het grootste nadeel bij het testen op een mobiel apparaat, is dat u geen gebruik kunt maken van de krachtige inspectors waaraan u op uw desktop zo gewend bent geraakt. Daarom zijn die externe webinspectors zo geweldig.

Foutopsporing op afstand

Er bestaan hulpprogramma's waarmee u op afstand via uw desktopbrowser fouten in uw mobiele browser kunt opsporen. Met hulpprogramma's voor foutopsporing op afstand kan uw desktopbrowser communiceren met externe apparaten om op afstand codes uit te voeren en vast te leggen. Net als de gewone hulpprogramma's voor foutopsporing kunt u deze hulpprogramma's op afstand gebruiken om uw HTML en CSS te controleren, uw DOM te wijzigen en live bewerkingen te maken en fouten in uw scripts op te sporen.

De Opera-browserengine wordt vervangen. Hoewel ik niet weet wat de toekomst brengt, kan Opera al sinds 2008 foutopsporing op afstand op de mobiele browser van Opera uitvoeren met het hulpprogramma voor foutopsporing Dragonfly voor Opera-desktops. Zo is het mogelijk HTML en CSS op afstand te controleren, de DOM bij te werken, onderbrekingspunten toe te voegen en alles wat u met Dragonfly ook op de desktop kunt doen.

WebKit begonmet de ondersteuning van foutopsporing op afstand via de USB-poort met Android 4 en iOS6. Wanneer uChrome wilt gebruiken om op afstand fouten op te sporen, moet u Chrome starten vanaf de opdrachtregel met een markering in plaats van het pictogram:

```
chrome.exe –remote-debugging-port=9222 –user-data-dir=remote-profile
```

of

```
/Applications/Chromium.app/Contents/MacOS/Chromium –remote-debug-
ging-port=9222
```

Wanneer u fouten wilt opsporen met de mobiele Firefox-browser moet u de foutopsporings-API, vroeger de Crossfire-uitbreiding genoemd, toevoegen aan Firebug.

De huidige staat is natuurlijk continu aan verandering en verbetering onderhevig. Blijf op de hoogte met het protocol voor foutopsporing op afstand (http://www.w3.org/2011/08/browser-testing-charter.html) als u hierin geïnteresseerd bent.

Hulpprogramma's voor foutopsporing van Android

De Android SDK omvat de API-bibliotheken en hulpprogramma's voor ontwikkelaars die nodig zijn om apps voor Android te maken, te testen en fouten erin op te sporen. U kunt webapplicaties direct op fouten controleren vanaf uw apparaten of vanaf emulatoren die u met de SDK kunt maken, zoals gezien in afbeelding 1.3

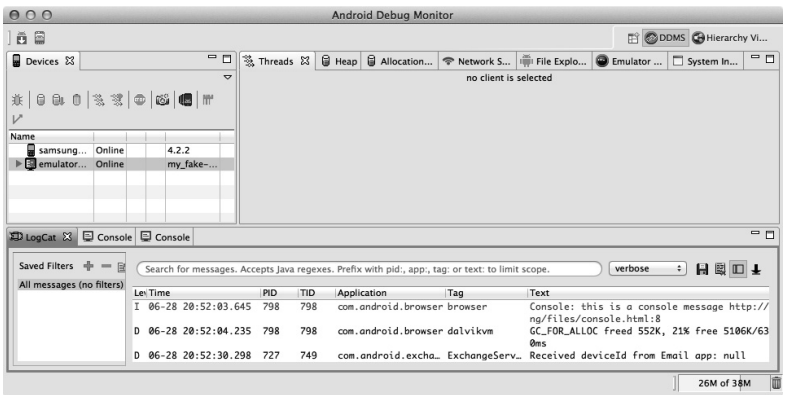
Afbeelding 1.3 *Android 4.2.2-emulator die op OS X werkt.*

Wanneer u de SDK downloadt via http://developer.android.com/sdk/ krijgt u de Android Debug Bridge (ADB), met opties voor foutopsporing, consolecontroles en het maken en starten van emulatoren.

Zoek in de gedownloade items naar de map **tool** en open **android** om toegang te krijgen tot ADB. De ADB beschikt over diverse apparaatbeheeropties, zoals het verplaatsen en synchroniseren van bestanden naar de emulator, het uitvoeren van een UNIX-shell op het apparaat of de emulator en het bieden van een communicatiemiddel voor verbonden emulatoren en apparaten.

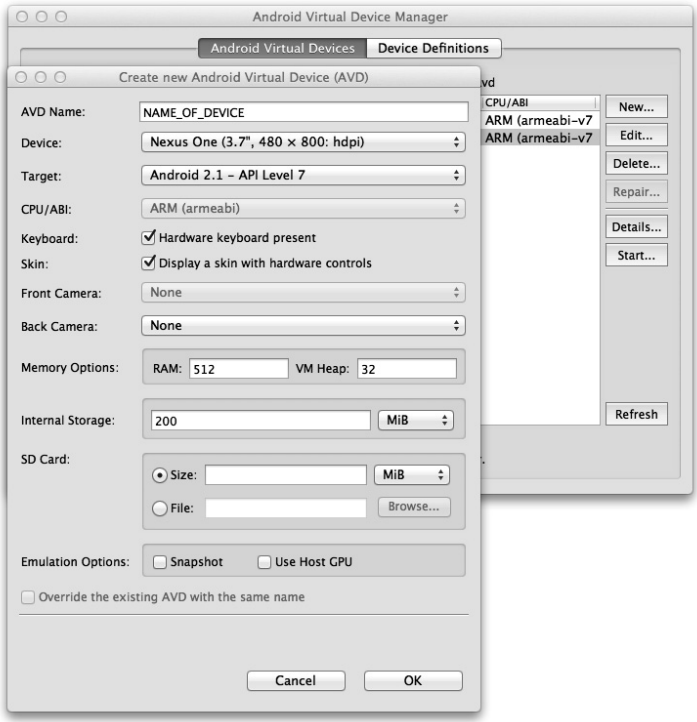
Er bestaat ook een plug-in (https://github.com/repenaxa/ADBPlugin) die een Chrome-uitbreiding is en werkt op een ADB-daemon. Hiermee is foutopsporing op afstand mogelijk voor mobiele apparaten zonder dat u de SDK hoeft te downloaden.

Open in dezelfde map **tools** de optie **Monitor** om toegang te krijgen tot de Android Debug Monitor. Dit venster bevat een console waarmee u fouten in uw applicaties kunt opsporen, alsook eventueleconsole.log()s kunt bekijken die u in uw site hebt opgenomen. De apparaten waarin fouten worden opgespoord, zijn te vinden in het tabblad **Devices** links in afbeelding 1.4 en het consolelogboek vindt u onderaan.



Afbeelding 1.4 *Het venster Android Debug Monitor.*

Wanneer het venster is geopend, vindt u in het menu **Windows** het venster **Android Virtual Device Manager**, zoals afgebeeld in afbeelding 1.5. Via dit venster kunt u nieuwe emulatortestapparaten maken en starten, zoals afgebeeld in afbeelding 1.3.



Afbeelding 1.5 *Met de Android Virtual Device Manager kunt u emulatoren maken van een beperkt aantal te selecteren apparaten of een onbeperkt aantal onafhankelijk gedefinieerde configuraties.*

Weinre

Weinre, wat staat voor *webinspector in remote*, is een krachtig hulpprogramma voor foutopsporing op afstand waarmee u JavaScript, HTML en CSS kunt controleren en op fouten kunt checken. Weinre maakt deel uit van het PhoneGap-project; u kunt het lokaal gebruiken of via http://debug.phonegap.com. Weinre is ook de basis van Adobe Edge Inspect, beschreven in de paragraaf *Adobe Edge Inspect en Ghostlab*.

Weinre is een hulpprogramma voor foutopsporing op afstand waarmee u het actuele browserscherm van uw mobiele apparaat kunt verbinden met een uitgeklede versie van de externe WebKit-inspector. Weinre maakt nu gebruik van Node.js en WebSockets[4].

4 Oorspronkelijk was het programma Java-gebaseerd. Voor WebSockets gebruikte het CORS, JSON en XHR.

Op het moment dat ik dit boek schrijf, is het een uitgekleed hulpprogramma voor foutopsporing. Met Weinre hebben we een liveweergave van de DOM en toegang tot de JavaScript-console, maar zijn er geen onderbrekingspunten of stapeltraceringen beschikbaar. Zoals verwacht bevat de JavaScript-console fouten, wat foutopsporing moeilijker maakt, maar niet ondoenlijk.

Weinre gebruiken

Weinre kan worden geïnstalleerd via Java of JavaScript. Wanneer u het wilt installeren met JavaScript, moet u Node.js downloaden en installeren, dat beschikt over npm, de Node Package Manager. Voer op de opdrachtregel het volgende in:

```
npm -g install weinre
```

om Weinre te installeren. U kunt Weinre nu starten door op de opdrachtregel het volgende te typen:

```
weinre
```

De Weinreserver werkt standaard op localhost:8080 tenzij deze wordt stilgezet met Control+C, dan wordt de computer opnieuw gestart of wordt de server uitgeschakeld.

Als u fouten wilt opsporen, moet u een Weinrescript aan de applicatie toevoegen met:

```
<src="http://localhost:8080/target/target-script-min.js#anonymous">
</script>
```

U kunt in elke WebKit-browser op de desktop http://localhost:8080/client/#anonymous openen om toegang te krijgen tot het hulpprogramma voor foutopsporing. De inspector wordt weergegeven in het volledige browservenster dat veel lijkt op de hulpprogramma's voor ontwikkelaars van Chrome, maar met beperkte functionaliteit en minder tabbladen.

In het tabblad **Remote** vindt u een lijst van de actuele browservensters van mobiele apparaten waarmee u fouten kunt opsporen en die op hetzelfde netwerk werken als uw Weinrescript. De tabbladen **Elements**, **Resources**, **Network**, **Timeline** en **Console**, zoals afgebeeld in afbeelding 1.6, zijn vergelijkbaar met die van de desktopwebinspector. U ziet dat de tabbladen **Sources**, **Profiles** en **Audit** ontbreken in dit vereenvoudigde opsporingsprogramma (hoewel deze in de toekomst mogelijk weer teruggebracht worden).

Afbeelding 1.6 *Hulpprogramma voor foutopsporing Weinre.*

Adobe Edge Inspect en Ghostlab

Om het foutopsporingsproces te vereenvoudigen waarbij de voorafgaande stappen bijna geautomatiseerd worden, kunt u met Adobe Edge Inspect op een soortgelijke manier fouten opsporen als met Weinre, waarop deze is gebaseerd. Dit gebeurt door taken, zoals het starten van de server, het invoeren van de URL in de browser en het toevoegen van scripts aan uw opmaak te bedekken.

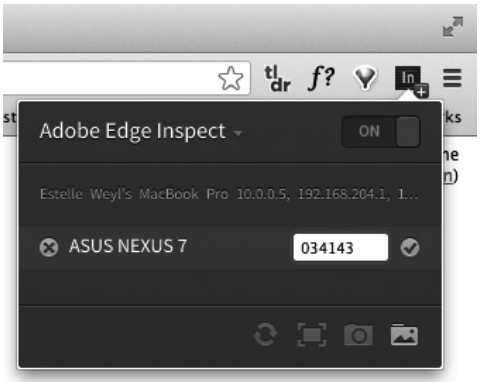
U moet Adobe Edge Inspect op al uw externe apparaten installeren en als Chrome-browseruitbreiding op uw desktop. Als het testapparaat en de desktop zich op hetzelfde netwerk bevinden, kunt u een verbinding maken met het apparaat.

Wanneer u Edge opent op uw mobiele apparaat, ontvangt u een wachtwoordcode voor het apparaat die u moet invoeren in de Edge-browseruitbreiding op de desktop. Schakel Edge in uw desktopbrowser in door eerst de applicatie te openen en u aan te melden bij Adobe.

Wanneer u zich hebt aangemeld, klikt u op het pictogram van de Edge-browseruitbreiding, zoals afgebeeld in afbeelding 1.7, zodat de browser de opdracht geeft apparaten te zoeken op het netwerk. Wanneer uw apparaat is gevonden, voert u de wachtwoordcode van het apparaat in het Edge-venster in.

De wachtwoordcode geeft aan dat u toestemming geeft dat uw computer en het mobiele apparaat met elkaar communiceren en voorkomt dat ongewenste computers uw apparaat en computer gebruiken om telefoons van anderen te besturen.

Wanneer er eenmaal een verbinding is tussen uw computer en een of meer apparaten, kunt u regelen welke pagina in al uw mobiele browsers wordt geladen. Het momenteel geopende tabblad in Chrome wordt opgehaald en weergegeven op de mobiele apparaten die zijn verbonden via Edge Inspect.



Afbeelding 1.7 *Adobe Edge Inspect verbindt een Nexus 7 en Google Chrome voor foutopsporing.*

Wanneer u vanaf een apparaat fouten in een webpagina wilt opsporen, bladert u naar de desbetreffende pagina in Chrome of op het apparaat. Wanneer u op het menu **Adobe Edge Inspect** in de Chrome-uitbreiding klikt, kiest u het apparaat waarvoor u fouten wilt opsporen. Weinre wordt gestart op uw lokale pc en het apparaat en de webpaginatitel worden weergegeven als actieve link op het tabblad **Remote** in Weinre, het meest linkse tabblad afgebeeld in afbeelding 1.6.

Met de gratis versie van Adobe Edge Inspect kunt u slechts met één apparaat tegelijk communiceren. Met een maandabonnement kunt u alle apparaten tegelijk beheren. Bovendien is het hiermee mogelijk screenshots te maken.

Werkt u op een Mac en wilt u ook meerdere apparaten testen, dan kunt u hiervoor Ghostlab (http://vanamco.com/ghostlab/) gebruiken. Wilt u een van beide hulpprogramma's aanschaffen, dan bent u met Ghostlab goedkoper uit dankzij de eenmalige kostprijs vergeleken met het maandelijks terugkerende abonnementsgeld van Adobe Edge.

JavaScript-foutopsporing met Aardwolf

Als het opsporen van fouten in JavaScript van groot belang is, kunt u het beste Aardwolf gebruiken. Aardwolf is een extern JavaScript-hulpprogramma voor foutopsporing waarmee u JavaScript kunt uitvoeren en vastleggen. Aardwolf werkt door uw code te herschrijven op de server en *hooks* voor foutopsporing

toe te voegen. Net als Weinre met een Node.js backend, gebruikt Aardwolf synchrone XHR-aanroepen om onderbrekingen op onderbrekingspunten in te schakelen. Met Aardwolf is het mogelijk uw code op afstand te doorlopen met ondersteuning voor het bekijken van objecten, onderbrekingspunten en aanroepstapels.

Hulpprogramma voor foutopsporing voor de BlackBerry 10

Ondanks dat Weinre een fantastisch hulpprogramma is, is het hulpprogramma dat bij de Blackberry 10 wordt geleverd, veel krachtiger.

Net als Weinre gebruikt de BlackBerry-browser een client-serverarchitectuur die de functies van Web Inspector beschikbaar maken. Maar in tegenstelling tot Weinre werkt de BlackBerry-browser als een webserver en biedt de webpagina aan via HTTP via een USB- of WiFi-verbinding. U controleert de inhoud dan op afstand vanaf een desktopbrowser. U kunt elke op een WebKit gebaseerde desktopbrowser op hetzelfde WiFi-netwerk gebruiken om naar het IP-adres en poortnummer te bladeren dat door de BlackBerry-browser wordt gebruikt, en beginnen met het controleren van de code.

Om de inspector te gebruiken moet u foutopsporing inschakelen via de BlackBerry-opties. Wanneer de webinspector is ingeschakeld, geeft de browser of applicatie het IP-adres en poortnummer weer die gebruikt worden om de inhoud aan te bieden.

U schakelt webinspector op de BlackBerry 10 vanaf de browserapplicatie in, door van boven naar beneden te swipen om de menubalk van de browser weer te geven. Klik op het instellingenpictogram en vervolgens op **Developer Tools** om de webinspector in te schakelen. Gebruikt u de tablet, dan vindt u deze functie onder **Options**, **Privacy & Security**. De browser geeft het IP-adres en het poortnummer weer die vereist zijn om vanaf uw desktopbrowser verbinding te maken. Als hierom wordt gevraagd, voert u uw apparaatwachtwoord in om het inschakelproces af te ronden. Klik op **Back** om de gegevens op te slaan en keer terug naar het browservenster. U kunt nu een verbinding maken met de BlackBerry-browser om de getoonde pagina's op afstand te controleren op fouten.

Testprogramma's

Het is raadzaam om uw sites op echte apparaten te bekijken. Maar het is onmogelijk om ze op alle apparaten te testen, aangezien er duizenden apparaten zijn en er telkens nieuwe op de markt komen. Daarom kunt u ze het best testen op een aantal representatieve apparaten met verschillende configuraties van besturingssystemen, browsers, apparaatformaten en

functionaliteiten, zoals verschillende schermresoluties, geheugengroottes en bandbreedtetoegang.

Het kan een dure en tijdrovende klus zijn om voor het testen echte apparaten te gebruiken. Behalve de hulpprogramma's voor foutopsporing uit de vorige paragraaf, zijn er nog andere diverse hulpprogramma's die ons helpen optimaal te testen.

Emulatoren en simulatoren

Een emulator is software die de functies van een of meer mobiele apparaten op een computer dupliceert of emuleert, zodat het geëmuleerde gedrag lijkt op het gedrag van het echte apparaat. Deze focus op de exacte reproductie van gedrag is het verschil tussen emuleren en simuleren. Bij simuleren wordt een abstract model van het mobiele besturingssysteem nagebootst.

Met emulatoren kunt u mobiele software op uw desktop gebruiken, waardoor u uw code kunt uitvoeren en op fouten kunt controleren zonder over alle apparaten te beschikken. Zelfs als u in emulatoren en simulatoren test, kunt u nog steeds niet alle apparaten testen. Emulatoren en simulatoren zorgen er slechts voor dat u aan de slag kunt en ze versnellen het ontwikkel- en foutopsporingsproces. U moet de applicatie nog steeds testen op een verscheidenheid aan mobiele apparaten.

Wanneer u uw website in een simulator uitvoert, voert u deze eigenlijk uit in een simulatieapplicatie op uw desktop. Sommige simulatoren zijn geschikt voor afzonderlijke apparaten en bij andere kunt u kiezen welk apparaat u wilt emuleren. Met de iOS-simulator kunt u bijvoorbeeld de iPhone of iPad kiezen. Via menu's kunt u de stand wijzigen tussen staand en liggend. Er zijn virtuele knoppen die de knoppen van het apparaat voorstellen. En op apparaten zonder aanraakfunctionaliteit kunt u met uw muis deze aanraakgebeurtenissen nabootsen.

De simulator geeft de apparaathardware niet exact weer en u hebt geen garantie dat de applicatie precies hetzelfde zal werken op het echte apparaat. Er zijn bepaalde bibliotheken die prima verzamelen en verbinden wanneer u werkt met de simulator (omdat deze in werkelijkheid op de desktop werkt), maar niet wanneer u met het apparaat werkt.

Simulatoren en emulatoren beschikken over het algemeen over een volledige SDK voor het testen van systeemeigen applicaties in een niet-systeemeigen omgeving. We kunnen onze code het beste testen met emulatoren en simulatoren die over een browser beschikken, wat alle emulatoren en simulatoren

hebben. Waarschijnlijk wilt u uw website downloaden naar en testen in de browsers van de volgende emulatoren en simulatoren:

Android-emulator

De gratis Android-emulator voor Windows, Mac OS en Linux is beschikbaar in combinatie met de SDK van http://developer.android.com. Zoals beschreven in de paragraaf *Android debugging tools*, moet u eerst de basis-SDK downloaden en daarna elk Android-OS afzonderlijk. De download biedt een Android-terminalopdracht in Mac/Linux en een SDK **Setup.exe** -applicatie voor Windows.

Met de Android-emulator kunt u het geheugen van het virtuele apparaat beperken zodat de telefoon beter wordt gesimuleerd. Selecteer het apparaat in de Android Virtual Device Manager en klik op **Edit** (afgebeeld in afbeelding 1.5). Klik bij de hardware op **New** en selecteer de hoeveelheid RAM van het apparaat in de vervolgkeuzelijst **Property**.

iOS-simulator

De iOS-simulator (http://developer.apple.com/iphone) is alleen beschikbaar voor Mac OS X, en biedt een gratis simulatieomgeving, zoals Mobile Safari. De iPhone SDK is ongeveer 2 GB. Het duurt dus lang voordat u deze hebt gedownload.

Dit is een simulator, geen emulator en bevat geen hardware-emulatie of prestatie-indicatoren. Het controleert of uw code werkt en hoe uw website wordt weergegeven, maar kan de websiteprestaties normaal gesproken niet beoordelen.

Als u even wilt kijken hoe uw ontwerp eruit gaat zien, zonder emulatie of simulatie, dan zijn er andere hulpprogramma's om te gebruiken, zoals iPhoney (http://www.marketcircle.com/iphoney) en iPad Peek (http://ipad-peek.com) die uw website gewoon openen in een browser die eruitziet als een ouder apparaatmodel.

BlackBerry-simulator

De BlackBerry-simulatoren (http://blackberry.com/developers) voor het Windows-besturingssysteem omvatten de proxyserver, plug-ins voor Eclipse en Visual Studio voor webontwikkelaars en de simulatoren.

Windows Phone-emulator

De Windows Phone-emulator is alleen beschikbaar op Windows-computers. Windows Phone-emulator is een desktopapplicatie die een Windows Phone-apparaat emuleert. U kunt de Windows Phone-SDK downloaden via http://dev.windowsphone.com/en-us/downloadsdk. De huidige versie en informatie over installatie is te vinden op http://bit.ly/16t5utu.

Momenteel is Emulator WVGA 512 MB de standaardemulator in Visual Studio, die een Phone 8-telefoon met een beperkt geheugen emuleert.

Firefox OS-simulator

De add-on Firefox OS Simulator (https://addons.mozilla.org/en-US/firefox/addon/firefox-os-simulator/) voor de Firefox-browser is de emulator voor het Firefox OS, die een Firefox OS-achtige omgeving biedt die eruitziet en aanvoelt als een mobiele telefoon. Ga na de installatie naar **Web Developer**, **Firefox OS Simulator** in uw Firefox-desktopbrowser.

Opera Mobile-emulator

De Opera Mobile-emulator voor Windows, Mac en Linux kan worden gedownload via http://www.opera.com/developer/tools.

Opera Mini-simulator

Een volledige Opera Mini-applicatie van de huidige versie van Opera Mini als een Java-applet is beschikbaar via http://www.opera.com/mini/demo.

Dit zijn de meest gangbare mobiele besturingssystemen. De meeste mobiele besturingssystemen, zoals Symbian en WebOS, hebben SDK's die u naar uw desktop kunt laden en waarmee u hun omgeving kunt simuleren. Afhankelijk van uw doelgroep, moet u alle besturingssytemen testen die uw doelpubliek waarschijnlijk zal gebruiken. Ga voor meer emulatoren naar http://www.mobilexweb.com/emulators.

Online hulpprogramma's

Wanneer u snel de belangrijke statistieken van uw apparaat wilt nagaan die invloed hebben op de basismediaquery's, moet u http://www.quirksmode.org/m/tests/widthtest.html openen in de browser van uw apparaat.

De W3C mobileOK Checker (http://validator.w3.org/mobile/) controleert uw website op de beste methoden en biedt informatie en links waarmee u de site mobielvriendelijker kunt maken. mobiReady (http://mobiready.com) is een online hulpmiddel dat gebruikmaakt van de W3C mobileOK Checker, en geeft de resultaten zodanig weer dat het u aanzet uw site mobielvriendelijker te maken.

De add-on Modify Headers for Firefox (http://mzl.la/17bqALt<C255U) is handig voor het ontwikkelen van mobiele applicaties, uitvoeren van HTTP-testen en privacy, en stelt u in staat HTTP-aanvraagheaders die naar webservers zijn verzonden, te wijzigen, toe te voegen, te vervangen en te filteren. Links naar deze bronnen (en alle andere bronnen die zijn vermeld in dit boek) vindt u op http://www.standardista.com/mobile.

Telefoons

Het testen van echte apparaten is een belangrijke stap van het ontwikkelproces. Maar het aanschaffen van een verzameling mobiele apparaten kan een aardige kostenpost zijn. Het wijzigen van de schermgroottes en het gebruik van emulatoren kan niet de werkelijke websiteprestaties, apparaatmogelijkheden, pixeldichtheid en impact van het mobiele netwerk exact weergeven.

Als u systeemeigen applicaties maakt, moet u echt apparaten zien te bemachtigen met de besturingssystemen waarvoor u ontwikkelt. In dit boek ontwikkelen we voor HTML5, CSS3 en JavaScript die niet systeemeigen zijn. Onze code werkt dus niet in browsers op alle apparaten. Hoewel we applicaties ontwikkelen voor de browser, moeten we in veel verschillende apparaten testen, inclusief een aantal op netwerken van telefoonproviders. Test uw code altijd op echte apparaten met echte verbindingen, zoals WiFi hotspots, 3G, 4G of zelfs EDGE. Neem de bus of de trein en probeer uw applicatie te openen vanuit verschillende locaties terwijl u door de stad, door woonwijken en buiten de bebouwde kom rijdt.

Browserlabs

Het testen op echte mobiele apparaten maakt deel uit van het ontwikkelproces dat onontbeerlijk is. Er zijn veel browserlabs. Zoek er dus een bij u in de buurt. Als er geen apparaatlabs zijn, moet u deze samen met anderen zelf maken.

Als u liever uw eigen apparaatlab hebt, moet u apparaten met verschillende formaten, besturingssystemen, mogelijkheden en browsers hebben. U kunt vrij goedkoop uw eigen apparaatlab maken dat een goed representatief beeld geeft van het mobiele landschap. Het is ondoenlijk om elk apparaat aan te schaffen. Wat u wel kunt doen, is exemplaren bemachtigen met verschillende formaten, browsers en besturingssystemen.

Er zijn ook virtuele apparaatlabs, zoals DeviceAnywhere en Nokia Remote Access. Dit zijn echte apparaten die u op afstand kunt bedienen. Aangezien dit echte apparaten zijn, komt u in de wachtrij te staan als een apparaat door iemand in gebruik is.

iOS

In Noord-Amerika zijn iOS-apparaten goed voor slechts 5% van het totale internetverkeer, maar voor meer dan 50% van het mobiele verkeer[5]. Als u of een van uw gezinsleden nog niet over een iOS-apparaat beschikt, en uw applicatie is niet uitsluitend gericht op de arme bevolking in onderontwikkelde landen, moet u zich er één aanschaffen.

5 http://bit.ly/HaW2PV> en http://bit.ly/1diKHLb> .

Pak één apparaat met het nieuwste iOS-besturingssysteem en één met een oudere versie van het besturingssysteem. Oudere apparaten zijn voor weinig geld verkrijgbaar op Craigslist of eBay. Momenteel beschikt slechts 1,8% van de iOS-gebruikers of 0,13% van de internetgebruikers over iOS 4.3 of ouder, en beschikt 12,5% van de iOS-gebruikers of 0,93% van de internetgebruikers over iOS 5.

Wanneer u een apparaat aanschaft, moet de browser werken. Dat is alles. Als u een klein budget heeft, kunt u apparaten met een kapot scherm krijgen voor bijna niks. Eén van de apparaten moet een telefoon zijn. De andere kunnen een telefoon, iPad of iPod touch zijn.

Wanneer u uw iOS-apparaten in huis hebt, downloadt u Opera Mini, dat gratis beschikbaar is via iTunes.

Als al uw apparaten hogeresolutieschermen hebben, moet u zorgen dat enkele van de andere apparaten dat niet hebben. Bovendien moeten niet al uw apparaten telefoons zijn; gebruik ook eens een of twee tablets.

Android

Android is het meest populaire en gevarieerde mobiele besturingssysteem ter wereld. Android werkt op talloze apparaten, zoals telefoons en tablets. Zorg dat u ten minste twee (liever meer) Android-apparaten heeft: een smartphone met veel mogelijkheden en een recent besturingssysteem en een goedkope telefoon die op een oudere versie werkt. Op het moment dat dit boek wordt geschreven, wordt de Android 2.3, hoewel verouderd, nog steeds verkocht en is deze momenteel de meest populaire versie van Android. Het systeem beslaat 34% van de Android-markt, en wordt door 2,3% van internetgebruikers wereldwijd gebruikt[6]

Behalve meerdere Android OS-versies, moet u apparaten hebben met verschillende formaten, verwerkingsvermogens, resoluties en producenten. Op uw Android-apparaat kunt u andere browsers toevoegen, zoals Chrome, Opera Mini en Mobile, Firefox Mobile en Dolphin Mini en HD.

Windows

Als u wilt investeren in een Windows-apparaat, moet u kiezen voor het meest recente besturingssysteem. De Windows Phone 7 is nooit enorm populair geweest, maar de Windows Phone 8 kan dit wel worden. Ze beschikken beide over de Metro UI-interface. Behalve uw applicatie testen om na te gaan of de opmaak werkt, moet u wat met uw Windows-telefoon spelen. De gebruikers-

6 http://developer.android.com/about/dashboards/index.html> .

interactie van het apparaat verschilt aanzienlijk van die van Android en iOS. Mogelijk wilt u een aantal UI-interacties wijzigen om beter aan de standaard gedragingen aan te passen die zijn ontwikkeld door het gebruik van het Windows-apparaat.

BlackBerry

Het BlackBerry 10-apparaat heeft de beste hulpprogramma's voor foutopsporing van alle mobiele apparaten, maar niet de grootste gebruikersbasis.

Er zijn meer van de oudere BlackBerry-apparaten op de markt dan de BlackBerry 10. BlackBerry-gebruikers van zowel nieuwe als oudere apparaten surfen op internet. U kunt het beste een BB6 of BB7 nemen. Oudere telefoons zijn gelukkig goedkoper, en het is verstandig om een apparaat te hebben zonder aanraakfunctionaliteit om uw websites op te testen.

Bij apparaten ouder dan de BB6 was de browser niet op WebKit gebaseerd. Er zijn steeds minder mensen die deze heel oude apparaten gebruiken. Als uw doelgroep over een BB5 of ouder beschikt, is het raadzaam een derde BlackBerry-apparaat aan te schaffen.

Nokia

Met Nokia bedoel ik het Symbian OS, niet de Lumia Windows Phone.

Symbian, Series 40, Samsung en in mindere mate Sony Mobile en Motorola, zijn in sommige landen gangbaarder dan Android, iOS, BlackBerry en Windows. Als ik een bepaald apparaat aanbeveel, bestaat de kans dat het op het moment dat u dit leest verouderd is. Onthoud alleen dat Nokia internationaal een grote rol speelt op de mobiele markt en een enorm bereik heeft. Ik raad u aan een featuretelefoon te kopen met een D-pad en een klein scherm, zodat u een idee krijgt wat een groot deel van de wereld ziet wanneer ze uw site bekijken.

Kindle

Vergeet ook de Kindle Fire met de WebKit-gebaseerde Silk-browser niet.

WebOS

WebOS wordt niet meer gemaakt, maar wel nog steeds gebruikt. De Palm Pre of Pixi is te koop voor minder dan $30.

Geautomatiseerd testen

De testprogramma's die zojuist zijn vermeld, helpen u visueel en handmatig te testen. Om echt goed te testen moet u draaien, in- of uitzoomen, pannen, klikken en schelden uit frustratie. Om te zien hoe de pagina wordt weergegeven, moet u deze echt bekijken op verschillende apparaatformaten en in verschil-

lende browsers en besturingssystemen. Voor de statische inhoud is dit voldoende en kunnen hulpmiddelen als Adobe Edge hierbij helpen.

Voor webapplicaties moet u het testen waarschijnlijk automatiseren. U dient de applicatie steeds opnieuw te testen om zeker te zijn dat de code echt werkt door al uw gebeurtenissen en resultaten te testen. Er zijn diverse testbibliotheken voor JavaScript.

Jasmine (http://pivotal.github.io/jasmine/) is een gedragsgestuurd ontwikkelframework. PhantomJS (http://phantomjs.org) is een headless WebKit, geen testbibliotheek, met een systeemeigen ondersteuning voor diverse webstandaarden, inclusief DOM-verwerking, CSS-selectoren en JSON. U kunt een vooraf gecompileerd binair bestand downloaden voor een willekeurig OS op de PhantomJS-website.

Wanneer u PhantomJS wilt gebruiken voor geautomatiseerde, front-endtesten, moet u CasperJS (http://casperjs.org/) downloaden. Als u AJAX-aanroepen wilt nabootsen, kunt u Sinon.JS (http://sinonjs.org) gebruiken. Elke site biedt goed geschreven documentatie om u aan de slag te helpen met deze bibliotheken voor het testen in WebKit. Het is echter geen oplossing voor testen op mobiele apparaten.

Er bestaan ook online testprogramma's. Met sommige, zoals SauceLabs (http://saucelabs.com), kunt u testen met honderden mobiele en desktopbrowser-/OS-platforms.

Kies de optie die voor u en uw applicatie het beste is, maar voer altijd testen uit.

Laten we nu gaan coderen, zodat we iets hebben om te testen.