

DevCampRWD — RESS

Responsive Design + Server Side Components

Agenda

1. Nakreślenie tematu
2. Dostępne narzędzia
 - a. WURFL
 - b. detectmobilebrowser.com
 - c. ress.io (private beta)
 - d. deviceatlas.com
 - e. 51degrees.mobi
 - f. detector.dmolsen.com
 - g. mobiledetect.net (Lekki opensource)
 - h. ua-parser (<https://github.com/tobie/ua-parser/>)
 - i. Detector (<http://detector.dmolsen.com>)
3. Integracja z CMS
4. Dobre praktyki (redirecty, link do “pełnej strony”)
5. Potencjalne problemy
 - a. CDN
6. DEMO time! <http://andmag.se/ress/>

1. Nakreślenie tematu

W niektórych przypadkach zdarzają się sytuacje, gdzie ilość zmian pomiędzy poszczególnymi urządzeniami jest zbyt duża, aby bazować na wspólnym kodzie.

Takie zmiany są spowodowane innym kontekstem użycia, który wymaga dość głębokich zmian w architekturze strony. Jako przykład można podać strony linii lotniczych, które zwykle w swoich mobilnych odpowiednikach serwują tylko wybrany podzbiór funkcjonalności z desktopa.

O ile kwestia ograniczania możliwości użytkowników na mobile nie jest tematem tego dokumentu, podoba nam się efekt końcowy - obiektywnie zmniejszona waga strony na urządzeniu mobilnym.

W takich sytuacjach na podstawie wykrycia urządzenia (zwykle po stronie back-end) kierujemy użytkownika do jednego z odseparowanych od siebie interfejsów, co pozwala na dużo prostszą kontrolę nad nimi.

Prawidłowa detekcja

Niestety, user agent string nie zawsze daje nam pełne, lub prawidłowe informacje o urządzeniu. Podstawowe problemy:

1. Nie ma sztywnej granicy pomiędzy mobile / tablet / desktop
2. Niektóre urządzenia przedstawiają się nieprawidłowo
3. "Długi ogon" niestandardowych urządzeń

Media Queries w CSS3 nie dają nam możliwości detekcji wszystkiego. W związku z tym powstały inne narzędzia do tzw. feature detection, takie jak biblioteka Modernizr, co uznaje się za bardzo dobrą praktykę. W przypadku RESS zwykle bazujemy na sprawdzaniu UA string z istniejącej bazy, lub sprawdzamy wsparcie JS-em, zapisujemy w cookie i na tej podstawie serwujemy inną treść w back-endzie.

Porównanie dostępnych bibliotek

Nazwa biblioteki	URL	Cena
WURFL	wurfl.sourceforge.net	Priced by number of sites, detections, and device capabilities accessed
Detect Mobile Browser	detectmobilebrowser.com	Free
Responsive ♥ Server Side	ress.io	?? private beta
Device Atlas	deviceatlas.com	Od 40\$ miesięcznie
51degrees	51degrees.mobi	Od 360\$ rocznie
Detector	detector.dmolsen.com	Free (beta)
mobiledetect	mobiledetect.net	Free

ua-parser	github.com/tobie/ua-parser	Free
-----------	----------------------------	------

Integracja z CMS

Wordpress

- WPTouch
- WP Mobile Detector
- Wordpress Mobile Pack
- Wordpress MobilePress

<https://github.com/serbanghita/Mobile-Detect#3rd-party-modules--submit-new> - lista modułów dla popularnych frameworków / cmsów (gotowe dla większości popularnych - Zend, Cake, Symfony, Drupal, Magento, Joomla, PrestaShop)

Dobre praktyki

Zachowanie tej samej struktury linków

Każdy URL powinien działać na każdym urządzeniu. Nie powinno stosować się np. przekierowań na główną stronę.

Odnosnik do “pełnej wersji”

Szczególnie gdy strona mobilna ma uproszczoną treść, użytkownik mobile powinien mieć możliwość przełączenia się na stałe na wersję klasyczną.

Mit upraszczania treści

Nie powinniśmy ograniczać dostępu do treści użytkownikom korzystającym z urządzeń mobilnych. Mówi się, że korzystając ze smartphonów każdy powinien móc zrobić to samo, co na desktopie - nawet, jeśli jest to trochę bardziej utrudnione.

Potencjalne problemy

W przypadku korzystania z load-balancera lub usług CDN, nie możemy pozwolić sobie na klasyczne serwowanie innej treści dla użytkownika, bo dzięki agresywnemu cachowaniu serwujemy statyczne pliki do użytkownika.

Wtedy zdani jesteśmy na usługodawcę, na szczęście wiele z nich oferuje wiele możliwości usprawnień takich samych, jak daje nam RESS (optymalizacja wydajności). Szczególnie Akamai daje dużo możliwości np. detekcji łącza użytkownika, co może pomóc nam w ustaleniu kontekstu użycia i w próbie lepszego dopasowania funkcjonalności.

Przykładowy flow wywołania strony

1. Zapytanie HTTP REQUEST
2. Sprawdzenie Cookie z szerokością strony.
 - 2a. Jeżeli takiego Cookie nie ma (pierwsze wywołanie) - rozpoznanie po User-Agent wersji strony (mobile, tablet, desktop)
3. Serwowanie HTTP RESPONSE z odpowiednim kodem html, listą stylów css, plików javascript na podstawie określonej wersji
4. Jeżeli brak Cookie lub przy akcji zmiany wymiaru okna - po stronie przeglądarki zapisanie szerokości okna w Cookie

Zalety

- Potencjalnie mniejsza ilość załadowanego kodu HTML, CSS, JS
- Potencjalnie mniejsza ilość requestów (obrazki)
- Możliwość na podstawie Cookie serwowania odpowiednio lżejszego obrazka
- Przy współpracy Server-Side z Client-Side jesteśmy w stanie wykryć problemy z łączem użytkownika i serwować mu lżejsze obrazki


Original website			Add image resizing		Add adaptive JS and CSS	
	Page size	Loading time	Page size	Loading time	Page size	Loading time
Desktop (high-speed)	1027 KB	3s	921 KB	3s	921 KB	3s
Desktop (56k modem)	1027 KB	3m 27s	921 KB	3m 14s	921 KB	2m 05s
iPhone 3G	1027 KB	14s	253 KB	7s	253 KB	6s
iPhone GPRS	1027 KB	2m 30s	253 KB	40s	253 KB	40s
Feature phone (2G)	1027 KB	∞	203 KB	35s	87 KB	25s

Original website			Add network adaptivity	
	Page size	Loading time	Page size	Loading time
Desktop (high-speed)	1027 KB	3s	921 KB	3s
Desktop (56k modem)	1027 KB	3m 27s	227 KB	40s
iPhone 3G	1027 KB	14s	153 KB	5s
iPhone GPRS	1027 KB	2m 30s	153 KB	25s
Feature phone (2G)	1027 KB	∞	25 KB	12s

Przykładowe wywołanie biblioteki WURFL w Firefox (1920x1080)

```
array(12) {
  ["mobile_browser"]=>
  string(0) ""
  ["device_os"]=>
  string(0) ""
  ["pointing_method"]=>
  string(5) "mouse"
  ["mobile_browser_version"]=>
  string(0) ""
  ["is_tablet"]=>
  string(5) "false"
  ["device_os_version"]=>
  string(0) ""
  ["is_wireless_device"]=>
  string(5) "false"
  ["preferred_markup"]=>
  string(12) "html_web_4_0"
  ["xhtml_support_level"]=>
  string(1) "4"
  ["resolution_height"]=>
  string(3) "600"
  ["resolution_width"]=>
  string(3) "800"
  ["ux_full_desktop"]=>
  string(4) "true"
}
```

Przykładowe wywołanie biblioteki WURFL w telefonie komórkowym



```
array(12) {
  ["mobile_browser"]=>
  string(14) "Android Webkit"
  ["device_os"]=>
  string(7) "Android"
  ["pointing_method"]=>
  string(11) "touchscreen"
  ["mobile_browser_version"]=>
  string(0) ""
  ["is_tablet"]=>
  string(5) "false"
  ["device_os_version"]=>
  string(3) "2.3"
  ["is_wireless_device"]=>
  string(4) "true"
  ["preferred_markup"]=>
  string(12) "html_web_4_0"
  ["xhtml_support_level"]=>
  string(1) "4"
  ["resolution_height"]=>
  string(3) "240"
  ["resolution_width"]=>
  string(3) "320"
  ["ux_full_desktop"]=>
  string(5) "false"
}
```



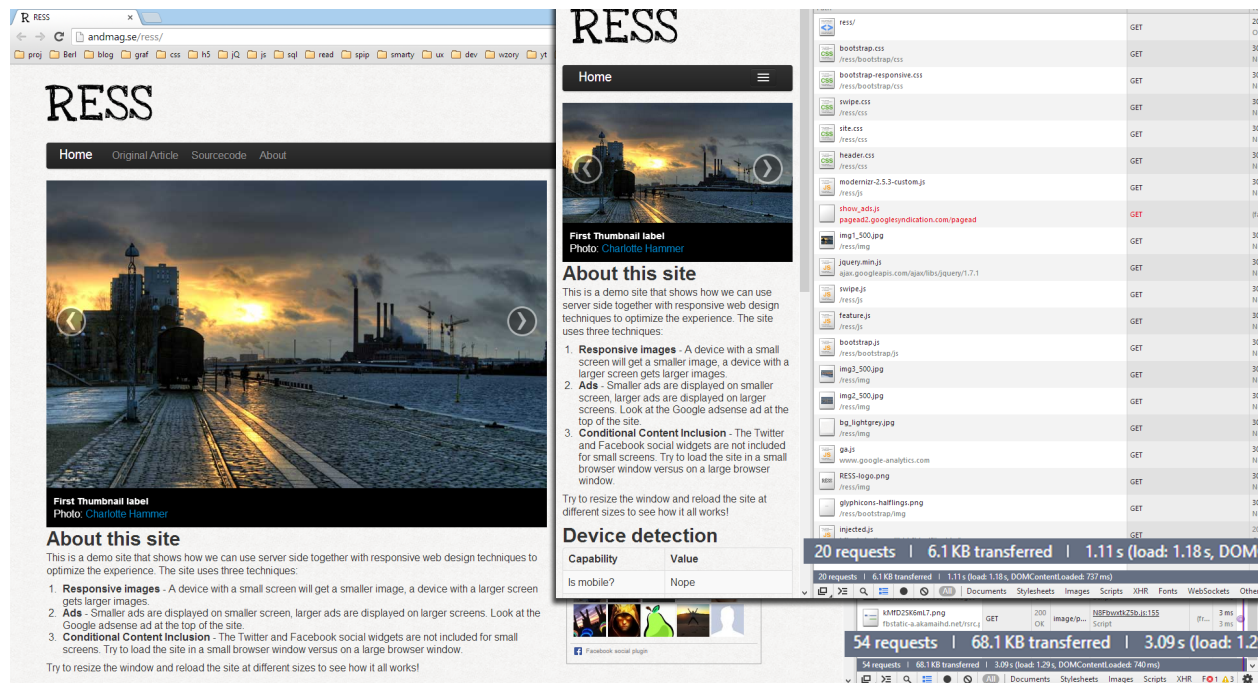
18:47



192.168.0.170/ress



```
array(12) {  
  ["mobile_browser"]=>  
    string(14) "Android Webkit"  
  ["device_os"]=>  
    string(7) "Android"  
  ["pointing_method"]=>  
    string(11) "touchscreen"  
  ["mobile_browser_version"]=>  
    string(0) ""  
  ["is_tablet"]=>  
    string(5) "false"  
  ["device_os_version"]=>  
    string(3) "4.2"  
  ["is_wireless_device"]=>  
    string(4) "true"  
  ["preferred_markup"]=>  
    string(12) "html_web_4_0"  
  ["xhtml_support_level"]=>  
    string(1) "4"  
  ["resolution_height"]=>  
    string(4) "1280"  
  ["resolution_width"]=>  
    string(3) "720"  
  ["ux_full_desktop"]=>  
    string(5) "false"  
}
```

Materialy

<http://www.lukew.com/ff/entry.asp?1392>

<http://wurfl.sourceforge.net/>

<https://github.com/dmolsen/Detector>

<http://mobile.smashingmagazine.com/2013/10/08/responsive-website-design-with-ress/>