O'REILLY®

Responsywne i wydajne projekty internetowe

Szybkie aplikacje dla każdego

SUPERWYDAJNE APLIKACJE I STRONY WWW!

Helion

Tom Barker

Tytuł oryginału: High Performance Responsive Design

Tłumaczenie: Jakub Hubisz

ISBN: 978-83-283-0838-1

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/rewypr.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/rewypr
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 1.

Stan rynku projektowania responsywnego

Problem projektowania responsywnego

Brałem udział w sesji planowania mapy drogowej z jednym z moich zespołów i naszą właścicielką produktu. Dyskutowaliśmy o zmianie wyglądu naszej sekcji wideo, kiedy lider zespołu zaczął mówić o planach jej responsywności. Opisaliśmy stworzenie pojedynczej strony z wyświetlanym na niej naszym domyślnym odtwarzaczem wideo, który zmieniałby rozmiary i ładował zasoby oraz listy odtwarzania różnych typów filmów, zależnie od urządzenia wykorzystanego do uruchomienia strony. Rozwiązanie to miało być piękne i wszechstronne, a także miało rozszerzać możliwości oglądania filmów dla wielu urządzeń, na których wcześniej nie było to możliwe.

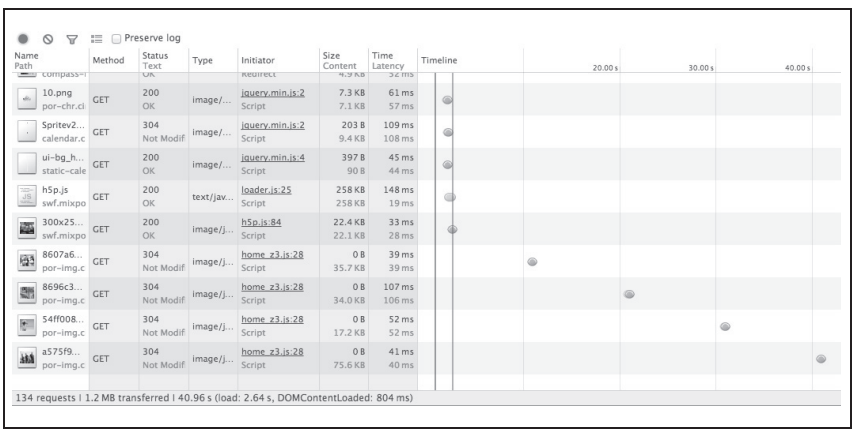
Nasza właścicielka produktu zmarszczyła nos i powiedziała: „Cóż, po tym jak responsywna okazała się strona główna, pozostał nam pewien niesmak".

Zaskoczyło mnie to. Co było nie tak z naszą responsywną stroną główną? Musiałem zbadać ten temat.

Zespół miał wrażenie, że strona główna jest ciężka i wolno się ładuje. Kiedy programiści demonstrowali ją na swoich laptopach, wyglądała świetnie, kiedy jednak zespół próbował, korzystając ze słabszych urządzeń, zaprezentować stronę osobom decyzyjnym, ładowanie jej trwało długo — zbyt długo.

Spojrzałem na *wykres wodospadowy*[1] dla renderowania strony głównej na urządzeniu mobilnym i laptopie. Zauważyłem coś, co — kiedy już wiedziałem, czego szukać — zacząłem dostrzegać na wielu stronach.

Podczas ładowania strony dla smartfona wykorzystywane były te same zasoby co w wersji dla komputerów oraz dodatkowy arkusz stylów CSS i plik ze sprite'ami. Rysunek 1.1 obrazuje to, że rozmiar strony renderowanej dla urządzenia mobilnego był nieznacznie większy niż w wersji dla komputerów (1,2 MB kontra 952 kB) i że wykonywane były dwa dodatkowe żądania.

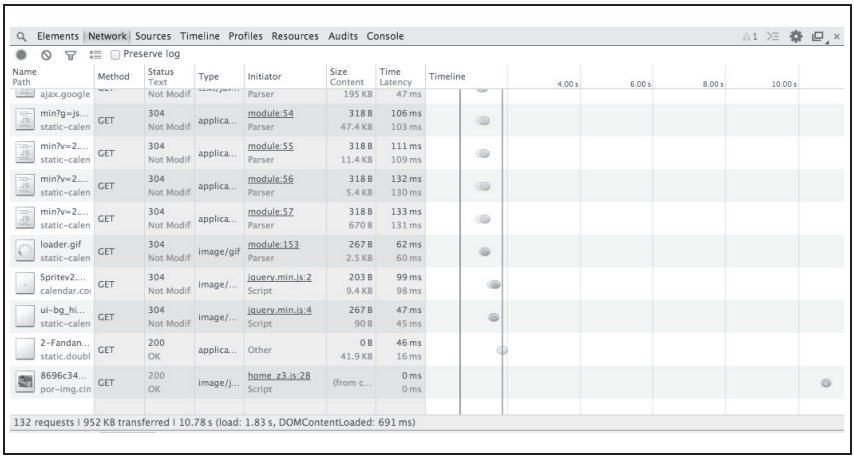


Rysunek 1.1. Wykres wodospadowy dla strony głównej renderowanej dla smartfona

Zauważ, że na rysunku 1.1 rozmiar przesłanych plików wynosi 1,2 MB w 134 żądaniach. Ale jest to przecież wersja dla telefonów i rozmiar powinien być mniejszy. Tak jednak nie jest — co pokazuje rysunek 1.2.

Całkowity rozmiar dla komputerów wynosi 952 kB w 132 żądaniach. To jasne, że w wersji dla telefonów ładuje się ta sama zawartość co w wersji dla komputerów oraz dodatkowe dwa pliki. Nie trzeba dodawać, że nie współgra to z ograniczeniami przepustowości występującymi na urządzeniach mobilnych.

[1] Wykresy wodospadowe wizualizują dane dotyczące żądań HTTP, czas potrzebny do załadowania zasobów oraz wielkość plików związanych z każdym żądaniem, składających się na stronę internetową. Bardziej dogłębna analiza wykresów wodospadowych zostanie przedstawiona w rozdziale 2.



Rysunek 1.2. Wykres wodospadowy dla strony renderowanej dla komputera

Taka sytuacja przeczy całkowicie naszym intencjom stworzenia strony mobilnej.

A nie byliśmy sami. Otworzyłem przeglądarkę na laptopie oraz program HTTPWatch na iPhonie i aby mieć porównanie, przeanalizowałem pierwszych 50 stron z rankingu *Alexa.com*. Zauważyłem, że około 30% tych stron przesyłało więcej danych w wersji dla telefonów niż dla komputerów — wśród nich były strony firm technologicznych, banków i sprzedawców detalicznych.

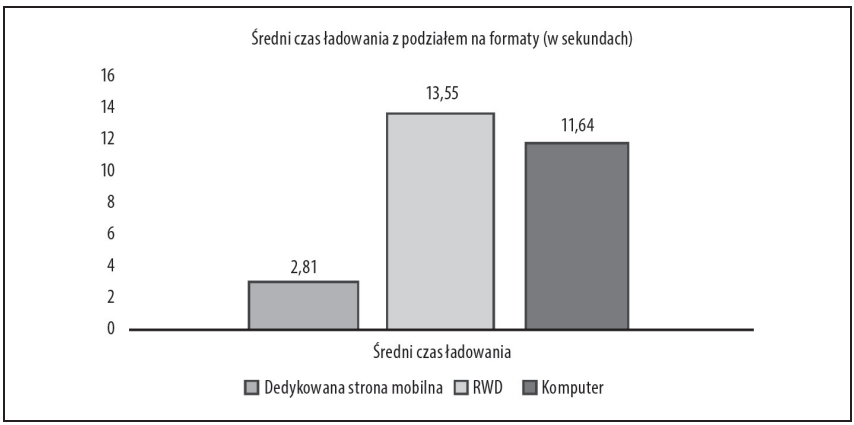
Oprócz moich własnych spostrzeżeń taki stan rzeczy potwierdzało także kilka znaczących raportów. Globalna cyfrowa agencja marketingowa, The Search Agency, przeanalizowała 100 najbardziej znanych stron sklepów internetowych oraz strony firm z listy Fortune 100 i opublikowała następujące raporty:

- „Multichannel Retailers"
  („Sprzedawcy wielokanałowi" — *http://bit.ly/1vqYUPh*),
- „Fortune 100 Companies"
  („Firmy Fortune 100" — *http://bit.ly/1r1SDlA*).

Wskazówka

Aby uzyskać dostęp do tych raportów, będziesz musiał przekazać agencji swój adres e-mail, a raport dostaniesz pocztą.

W jednym z raportów znalazł się wykres przedstawiony na rysunku 1.3, pokazujący, że strony, które wykorzystywały (lub raczej błędnie wykorzystywały) koncepcję projektowania responsywnego, ładowały się średnio 1,91 sekundy dłużej od zwykłych stron. Co więcej, te strony ładowały się o 10,74 sekundy dłużej niż strony dedykowane dla urządzeń mobilnych.



Rysunek 1.3. Porównanie średnich czasów ładowania dla stron responsywnych oraz stron dedykowanych dla urządzeń mobilnych i dla komputerów

Guy Podjarny, dyrektor techniczny w firmie Akamai, na swym blogu także opisał swoje wnioski z wykonywania podobnych testów. Porównywał rozmiary stron dla różnych rozdzielczości i przekonał się, że różnice pomiędzy nimi są bardzo nieznaczne. Artykuł możesz znaleźć pod adresem *http://bit.ly/1tBv6cT*.

Czy wszyscy myliliśmy się w naszej interpretacji koncepcji projektowania responsywnego?

Wnioski z analizy porównawczej

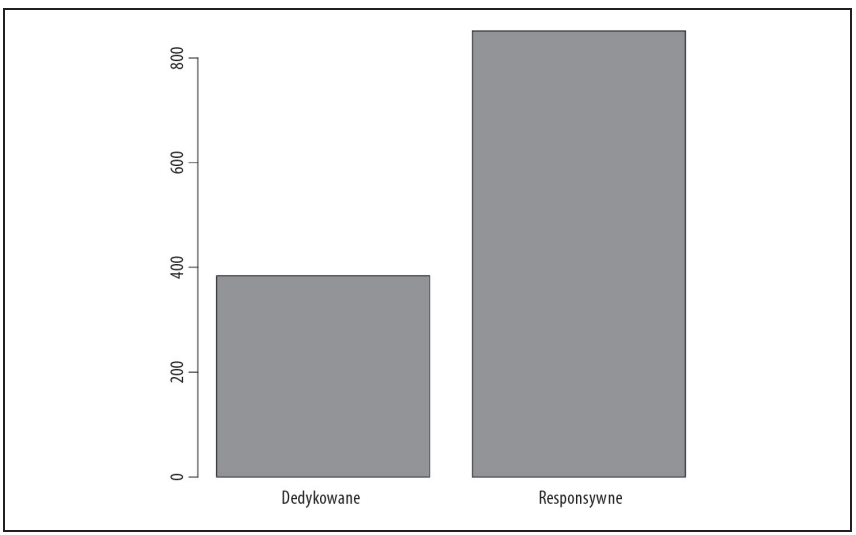
Moje obserwacje stron z listy *Alexa* także pozwoliły uzyskać interesujące dane. Między innymi zauważyłem następujące rzeczy:

- 47% z najbardziej znanych stron w USA nadal korzystało ze stron dedykowanych dla *mdot*[2]. Zastanówmy się przez chwilę nad tą liczbą. Są

[2] Strona *mdot* to strona dedykowana tylko dla urządzeń mobilnych, mająca swój własny adres URL, zazwyczaj w konwencji wykorzystującej *m* jako subdomenę (np. *m.comcast.net* lub *m.homedepot.com*). Istnieją nawet nowsze, dedykowane dla tabletów, wersje stron *mdot*, dla których subdomenę stanowi litera *t* (np. *t.homedepot.com*).

to najczęściej odwiedzane strony, prawdopodobnie należą do liderów w swoich dziedzinach — wśród nich znajdują się tacy giganci jak YouTube, eBay czy Target — a jednak ci liderzy rezygnują z responsywnych stron na rzecz osobnych dedykowanych stron.

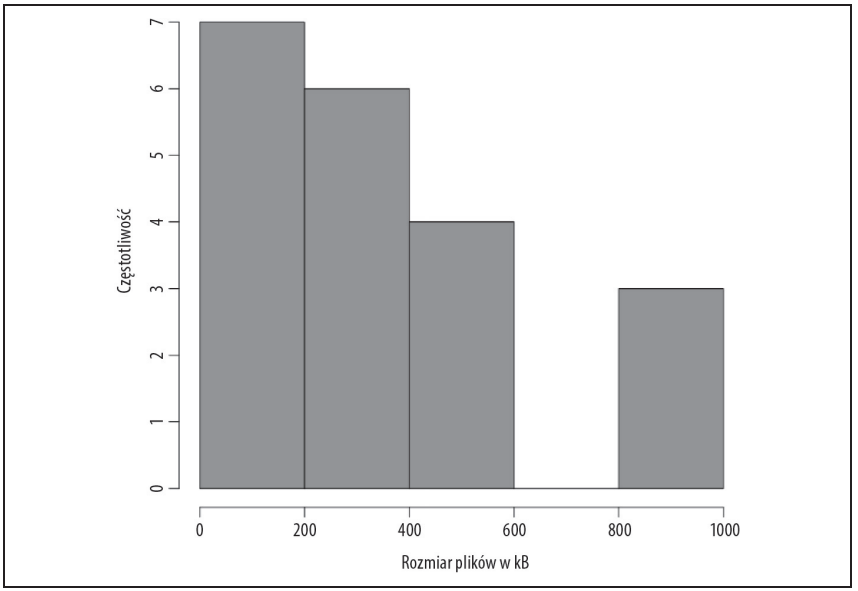
- Te dedykowane strony były średnio o 55% mniejsze od stron responsywnych. Przeciętny rozmiar dla podzbioru wykorzystującego strony *mdot* wynosił 383 kB, podczas gdy dla stron responsywnych aż 851 kB (patrz rysunek 1.4). Pokazuje to ogromną rozbieżność pomiędzy intencjami a implementacją.



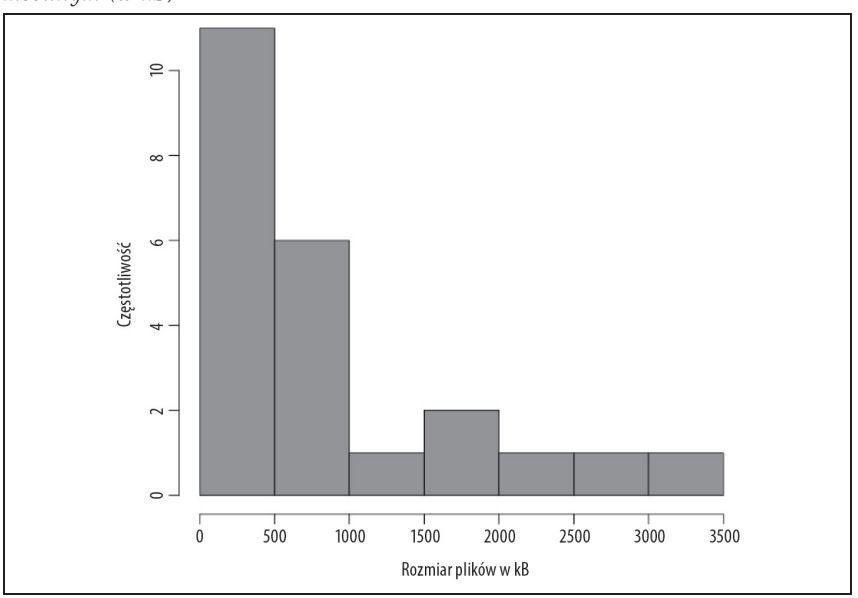
Rysunek 1.4. Średni rozmiar dla stron dedykowanych i responsywnych (w kB)

- Rozmiary plików stron responsywnych były bardzo różne i sięgały nawet 4 MB, podczas gdy rozmiary stron *mdot* nie przekraczały 1 MB. Co więcej, strony *mdot* znajdują się w większości w przedziale od 0 do 200 kB i od 200 do 400 kB. Opracowałem histogramy pozwalające spojrzeć na rozkład rozmiarów plików dla stron *mdot* i responsywnych (znajdziesz je na rysunkach 1.5 i 1.6).

Zwróć uwagę na skalę osi *X* na każdym z wykresów. Dla dedykowanych stron trzy pierwsze słupki są zdecydowanie większe niż ostatni, reprezentujący strony o rozmiarze 1 MB. Dla stron responsywnych strony o rozmiarze 1 MB stanowią drugą najliczniejszą grupę, a rozmiary sięgają nawet 4 MB.

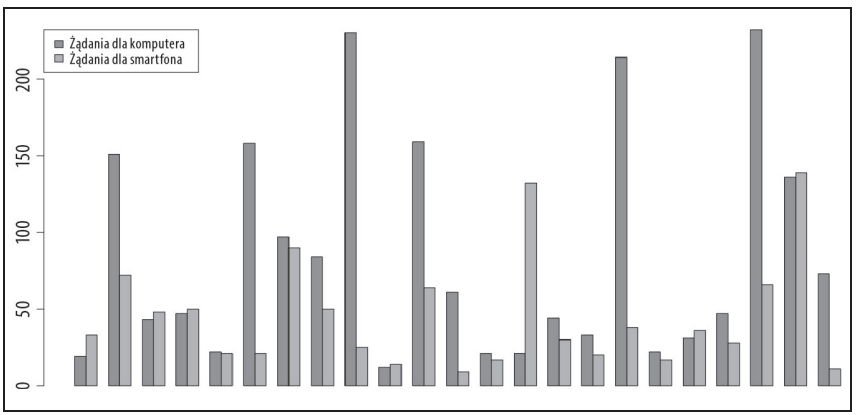


Rysunek 1.5. Rozkład rozmiarów plików dla stron dedykowanych urządzeniom mobilnym (w kB)



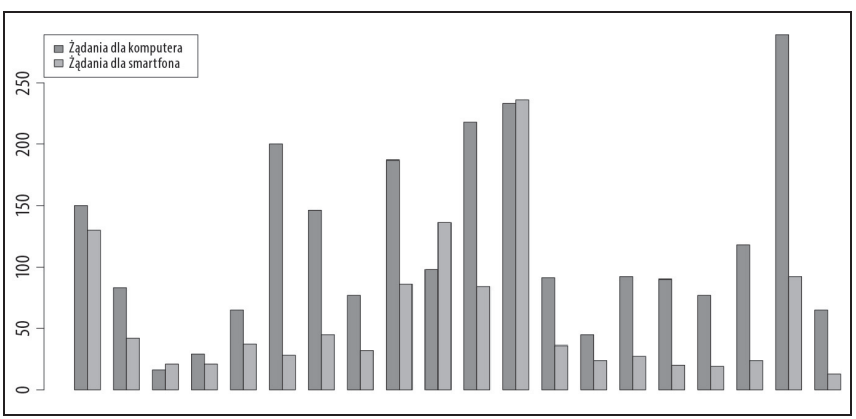
Rysunek 1.6. Rozkład rozmiarów plików dla stron responsywnych (w kB)

- Spośród stron responsywnych 43% miało prawie tyle samo lub nieznacznie więcej żądań HTTP dla urządzeń mobilnych co dla komputerów. W przypadku stron dedykowanych tylko 1,5% z nich miało tyle samo lub więcej żądań HTTP co ich odpowiedniki dla komputerów. Rysunki 1.7 i 1.8 obrazują te dane.



Rysunek 1.7. Wykres żądań HTTP dla urządzeń mobilnych i komputerów w przypadku stron responsywnych

Zauważ, że na rysunku 1.7 w każdej grupie ciemniejszy słupek obrazuje liczbę żądań HTTP dla strony przeznaczonej dla komputera, podczas gdy słupek jaśniejszy to liczba żądań HTTP dla smartfona.



Rysunek 1.8. Wykres żądań HTTP dla urządzeń mobilnych i komputerów w przypadku dedykowanych stron mdot

I znów: zauważ, że dla każdej grupy słupki ciemniejsze reprezentują żądania HTTP dla komputerów, a słupki jaśniejsze dla smartfonów.

Jak widać, sposób implementacji projektów responsywnych sprawia problemy. Istnieją też niezaprzeczalne korzyści wynikające z oferowania stron dedykowanych dla poszczególnych urządzeń, przynajmniej w kontekście liczby żądań HTTP i objętości plików wykorzystywanych do renderowania strony (chociaż należy zaznaczyć, że strony *mdot* obarczone są swoimi własnymi problemami, które omówimy wkrótce). Moja teoria i myśl przewodnia, którą powinieneś wielokrotnie zauważyć podczas czytania tej książki, mówi, że projektowanie responsywne i strony dedykowane nie są wykluczającymi się rozwiązaniami, ale są aspektami tej samej filozofii.

Oprócz przedstawionych wyżej wskaźników zaobserwowałem również kilka *antywzorców*[3] i wzorców, które zastosowane zostały przy tworzeniu sprawdzanych przeze mnie stron.

Antywzorce

Kiedy przyglądałem się każdej ze stron z rankingu Alexa, zacząłem dostrzegać często powtarzające się problemy — antywzorce, które zostały w nich zastosowane. Poniżej przyjrzymy się kilku takim antywzorcom.

Ładowanie tych samych zasobów

Niektóre ze stron ładowały dokładnie te same zasoby zarówno dla wersji mobilnej, jak i dla komputerów. W obu przypadkach ładowały ten sam plik CSS zawierający zapytania o media, które obsługiwały zmiany w rozdzielczości. Ładowały te same obrazy, które były przeskalowywane przez przeglądarkę, kiedy wykrywała, że wymaga tego rozdzielczość.

Za ten błąd płaci się zwiększonym transferem. Strony, które miały dokładnie tyle samo żądań dla wszystkich urządzeń, najprawdopodobniej były zbudowane w ten sposób. To rozwiązanie nie sprawdza się jednak w przypadku urządzeń o większej rozdzielczości, takich jak ekran Retina firmy Apple czy telewizory Ultra HD.

Ładowanie dodatkowych zasobów

Chociaż ładowanie tych samych zasobów dla wszystkich urządzeń ignoruje różnice pomiędzy nimi, ładowanie dodatkowych, oprócz standardowego

[3] Antywzorce to często wykorzystywane niewydajne, nieefektywne lub nieproduktywne rozwiązania problemów. Są przeciwieństwem wzorców projektowych, które są przetestowanymi i niezawodnymi rozwiązaniami powszechnych problemów.

zestawu, zasobów dla smartfonów stoi całkowicie w sprzeczności ze wszystkim, co wiemy o tworzeniu rozwiązań dla urządzeń mobilnych. Te dodatkowe zasoby to zazwyczaj dodatkowy plik CSS i dodatkowy plik ze sprite'ami.

Takie rozwiązania obserwowałem w przypadku stron, które miały więcej żądań HTTP i większe zapotrzebowanie na przepustowość dla urządzeń mobilnych. Jak już wcześniej wspomniałem, ten antywzorzec znalazł zastosowanie także na mojej stronie.

Ładowanie obrazów o dwukrotnie za dużym rozmiarze

Najgorszym błędem było to, że niektóre strony dla wersji mobilnej ładowały dodatkowy zestaw obrazów o rozmiarze dwukrotnie większym niż te z wersji dla komputerów. Oczywiście, oprócz standardowego zestawu obrazów.

Powodem ładowania większych obrazów i przeskalowywaniem ich jest to, że wtedy obrazy wydają się być ostrzejsze. Niestety, efektem ubocznym jest to, że takie strony mają często w przypadku urządzeń mobilnych o 30% większe zapotrzebowanie na przepustowość niż w przypadku komputerów.

Wszystkie te problemy mają kilka wspólnych cech:

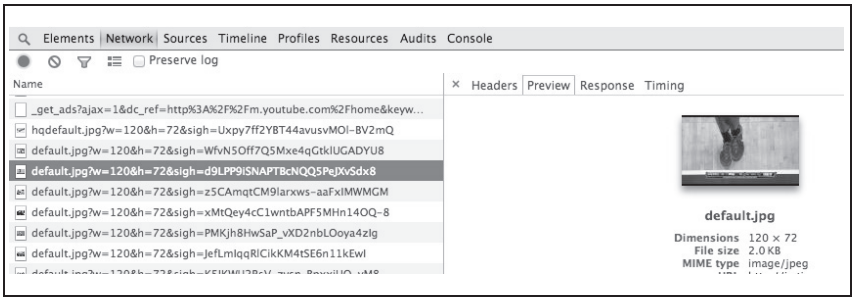
- Wersja dla komputerów uznawana była za wersję bazową, do której dodawane są elementy. Poprawne podejście to rozpoczynanie od najmniejszej wersji i rozbudowywanie jej.
- Nie brano pod uwagę ograniczeń każdej z platform.
- Próbowano rozwiązać problem wyłączenie po stronie klienta.

Wzorce

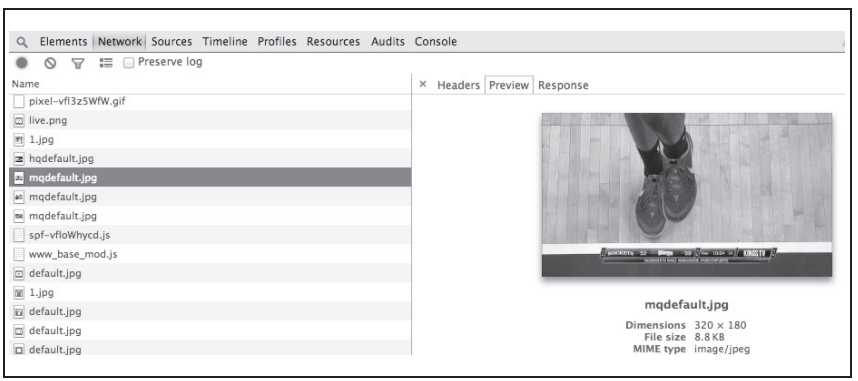
Nie wszystkie strony z listy Alexa robiły to źle — niektóre oferowały doskonałe wrażenia zoptymalizowane dla urządzenia i rozdzielczości, na które były przeznaczone. Przyjrzyjmy się kilku zastosowanym w nich wzorcom projektowym.

Ładowanie zasobów dostosowanych do urządzenia

Zamiast dla urządzeń mobilnych ładować obrazy o dwa razy większym rozmiarze niż dla komputera, niektóre strony ładowały obrazy o połowę mniejsze niż dla komputerów. Rysunki 1.9 i 1.10 przedstawiają taki przykład.



Rysunek 1.9. Ładowanie przystosowanych do urządzenia mobilnego obrazów o rozmiarze 120×72 pikseli i o wielkości 2 kB (sprawdzane przy wykorzystaniu Narzędzi dla programistów w przeglądarce Chrome)



Rysunek 1.10. Ładowanie przystosowanych dla komputerów obrazów o rozmiarze 120×180 pikseli i wielkości 8,8 kB (sprawdzane przy wykorzystaniu Narzędzi dla programistów w przeglądarce Chrome)

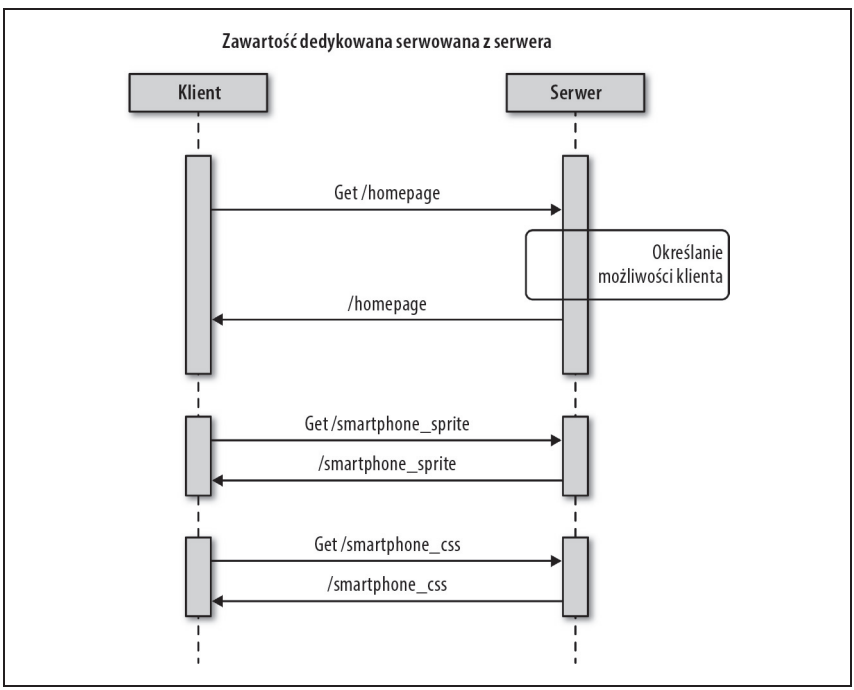
Zauważ, że obrazy na rysunkach 1.9 i 1.10 są takie same; są po prostu przeskalowane z uwzględnieniem zasobów urządzenia klienta.

Niektóre strony w ten sam sposób ładowały dostosowane do urządzenia sprite'y i style CSS — zamiast, a nie oprócz standardowego zestawu dla komputerów. Takie rozwiązanie bierze pod uwagę ograniczenia przepustowości i koszty sieci komórkowych. Niestety, większość takich stron z listy Alexa stanowiły dedykowane strony *mdot*. Możemy jednak wykorzystać ten wzorzec także w przypadku stron responsywnych, co dokładniej omówimy w rozdziale 4.

Serwowanie dedykowanych stron z serwera

Najlepiej dostosowane były strony, które serwowały całkowicie dedykowaną zawartość. Niektóre były odrębnymi stronami *mdot*, inne miały dedykowany układ i zasoby przystosowane do urządzenia i przesyłane do strony z serwera. Takie rozwiązane nazywane jest czasami RESS (*Responsive Design + Server-Side Components*), ale tak naprawdę oznacza po prostu wykorzystanie tej samej logiki, za pomocą której ruch kierowany jest na strony *mdot*, do załadowania odpowiednich zasobów dla predefiniowanego przedziału rozdzielczości. To rozwiązanie zostanie dokładniej omówione w rozdziale 4.

Aby lepiej zrozumieć tę architekturę, rzuć okiem na diagram sekwencji na rysunku 1.11.



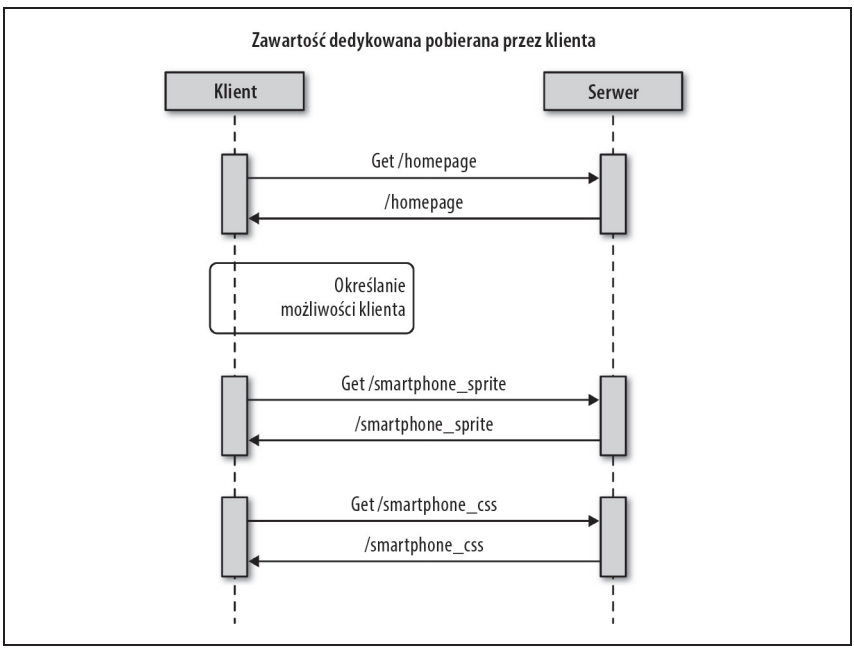
Rysunek 1.11. Diagram sekwencji dla przesyłania z serwera zawartości dostosowanej do urządzenia

Zauważ, że strony, które dostarczały dedykowaną zawartość, miały najmniejszy rozmiar i były najbardziej wydajne. Prawdopodobnie dlatego 47% sprawdzanych stron nadal oferuje dedykowaną zawartość.

Leniwe ładowanie dedykowanej zawartości po stronie klienta

Niektóre strony stosowały *leniwe ładowanie*[4] nie tylko dla obrazów, lecz także dla całych modułów zawartości, zarówno poniżej, jak i powyżej aktualnej pozycji. Dzięki temu uniknęły ładowania zawartości dla każdego punktu przerwania i mogły inteligentnie ładować tylko zawartość niezbędną do aktualnego działania i dostosowaną do możliwości klienta. Zamiast jednak sprawdzać to po stronie serwera, można to zrobić po stronie klienta. Tę taktykę omówimy w rozdziale 5.

Rysunek 1.12 przedstawia diagram sekwencji dokładniej obrazujący to podejście.



Rysunek 1.12. Ładowanie zawartości dostosowanej dla urządzenia po stronie klienta

[4] *Leniwe ładowanie* jest wzorcem projektowym polegającym na tym, że inicjalizacja obiektu lub pobranie zasobu jest odroczone do momentu, w którym ten obiekt lub zasób ma zostać wykorzystany.

Jak mogliśmy tego nie zauważyć?

Wcześniej opisywałem jak demonstrowaliśmy naszą responsywną stronę główną naszym właścicielom produktu. Podczas analizy w sprincie otworzyliśmy stronę na jednym z naszych laptopów, przesłaliśmy obraz na ekran i zmienialiśmy rozmiar naszej przeglądarki, aby odwzorować różne punkty przełamania. Chociaż dobrze było zobaczyć jak strona płynnie się zmienia, takie testy zupełnie nie brały pod uwagę podstawowego zadania — oferowania zawartości dla różnych urządzeń.

Testowaliśmy ją w ten sam sposób, w jaki ją tworzyliśmy: na komputerze Macbook Pro, korzystając z firmowego łącza. Oczywiście wydajność nie budziła naszych zastrzeżeń. Nie mieliśmy żadnych predefiniowanych założeń odnośnie wydajności (np. SLA — *Service Level Agreement*[5]). Nie korzystaliśmy z urządzenia mobilnego i sieci komórkowej. Wtedy nie mieliśmy nawet żadnych urządzeń do testów — poza naszymi osobistymi urządzeniami.

Najważniejsze było to, że nie mieliśmy ustalonego SLA dotyczącego wydajności. Spójność z naszą istniejącą stroną stanowiła jedyne kryterium akceptacji i nie umieściliśmy żadnych czerwonych flag w istniejących systemach monitorowania wydajności. Więcej na ten temat dowiesz się z rozdziału 3.

Jak znaleźliśmy się w tym punkcie?

Dawno temu, w 2008 roku, zanim narodziło się pojęcie projektowania responsywnego, utrzymywalibyśmy dwa adresy URL: *mojastrona.pl* i *m.mojastrona.pl* (nasza strona *mdot*). Obie strony byłyby różnymi stronami tej samej aplikacji internetowej, albo też mogłyby być osobnymi aplikacjami, być może nawet utrzymywanymi przez różne zespoły. Jednak mogłoby się tak zdarzyć tylko wtedy, jeśli wykazalibyśmy się dalekowzrocznością lub mielibyśmy inne strony mobilne, co w tamtym czasie nie było zbyt częste.

Potem, w 2011 roku, została uruchomiona nowa wersja strony „The Boston Globe", a terminy *projektowanie responsywne* i *progresywne ulepszanie* stały się tematami wszystkich publikacji na blogach i paneli dyskusyjnych.

[5] SLA definiuje reguły kontraktu usługi. Definicja ta może być, zgodnie z potrzebami, formalna lub nie. Może dotyczyć dostawcy interfejsu programistycznego aplikacji (API — *Application Programming Interface*), który zobowiązuje się zapewnić pewien minimalny czas działania i reagować na problemy w określonym czasie. Może również dotyczyć zespołu programistów i zobowiązania do naprawienia wszystkich błędów ujawnionych w określonym czasie.

Wszyscy czytaliśmy artykuły opisujące sposoby tworzenia stron dostosowanych do możliwości urządzenia użytkownika, wszyscy też eksperymentowaliśmy z tą koncepcją i daliśmy się uwieść. Byli ludzie, którzy pamiętali tworzenie w pierwszych latach XXI wieku płynnych układów graficznych z relatywnie ustalonymi wysokościami i szerokościami; na początku nie dostrzegali różnicy, ale kiedy zobaczyli, jak skalowane są rozmiary czcionek i obrazy, nawet oni zostali porwani przez koncepcję projektowania responsywnego.

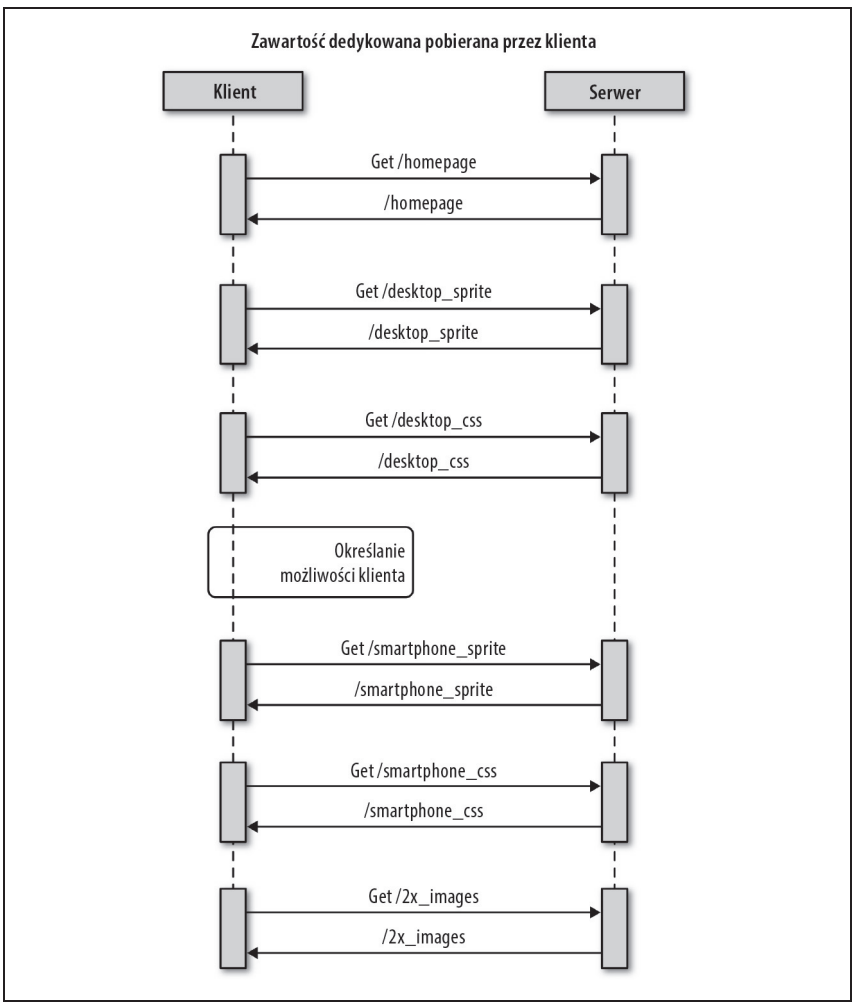
Napisano książki, poczyniono słowne ustalenia i wszyscy zaczęli tworzyć responsywne strony. Wszyscy zaczęli korzystać i mówić o zapytaniach o media pozwalających enkapsulować style dla różnych rozdzielczości. Eksperymentowaliśmy też z różnymi sposobami skalowania obrazów.

Kiedy przyszedł czas, aby nowe koncepcje wykorzystać „naprawdę", wszyscy wiedzieliśmy, że powinniśmy zacząć od najmniejszego rozmiaru ekranu i progresywnie go rozbudowywać. W praktyce jednak osoby decyzyjne chciały widzieć „kompletną" wersję (np. wersję dla komputerów) tego, co będą pokazywać swoim zwierzchnikom. Zespoły projektowe traktowały więc te prace priorytetowo i takie właśnie wersje były budowane jako pierwsze. Mogliśmy jednak korzystać z zapytań o media utrzymujących style CSS dla punktów przełamania i minimalizujących niepożądane wrażenia wizualne — wszystko wydawało się być w porządku, prawda?

Nasze podstawowe pliki CSS i JavaScript były wersjami dla komputerów (najprawdopodobniej miały po kilkaset kilobajtów), a pliki dla smartfonów i tabletów były nakładane jako następna warstwa po zdeterminowaniu możliwości urządzenia po stronie klienta. Następnie demonstrowaliśmy wyniki osobom decyzyjnym, one demonstrowały je swoim zwierzchnikom, aż w końcu produkt trafiał na środowisko produkcyjne. W końcu jakiś programista wpadał na pomysł, że powinniśmy pomyśleć o refaktoryzacji, ponieważ nasz podstawowy CSS przeznaczony jest dla komputerów i — no właśnie — wszystkie nasze łącza są i tak podpięte do wersji dla komputerów. Niestety, nigdy nie było zbytniego zapału, aby to zmienić — produkt działał i nie było czasu na zmiany, bo wkrótce przecież startował następny projekt i potrzebni byli wszyscy pracownicy.

Produkt działał, ale problem polegał na tym, że wciąż patrzyliśmy tylko na front-end. Zapytania o media i skalowanie obrazów wyglądały dobrze, ale skupianie się tylko na nich mijało się z podstawowym celem tworzenia holistycznego doświadczenia dla urządzenia, z którego w danej chwili korzysta użytkownik. Były to tylko pozory responsywności bez prawdziwej responsywności.

Nie skupialiśmy się tylko na tym, jak wygląda aplikacja; po stronie klienta umieszczaliśmy także logikę. Polegając wyłącznie na zapytaniach o media przy określeniu rozdzielczości i testując możliwości urządzenia w JavaScript po stronie klienta, pobieraliśmy niepotrzebne zasoby. To doprowadziło do powstania antywzorców, które zidentyfikowaliśmy wcześniej. Rysunek 1.13 przedstawia diagram sekwencji dla wszystkich tych antywzorców.



Rysunek 1.13. Diagram sekwencji dla antywzorców

Jeżeli skupiamy się na wyglądzie strony tylko po stronie klienta, różnice pomiędzy urządzeniami, włączając w to infrastrukturę internetową, moc obliczeniową, czas pracy baterii i dostępną pamięć, są ignorowane. W rzeczywistości są to jednak czynniki, które powinniśmy brać pod uwagę. Są głównymi powodami, dla których główni gracze w internecie nadal wspierają odrębne strony *mdot*.

Dlaczego nie skorzystać z mdot?

Czytając o tych wszystkich zaletach stron *mdot*, możesz zastanawiać się, czemu nie piszę o tym, dlaczego wszyscy powinniśmy zacząć z nich korzystać. Nie mam zamiaru promować stron *mdot*. Chociaż pod względem wydajności górują nad większością aktualnie istniejących stron responsywnych, mają kilka minusów.

Narzut zasobów

Moja pierwsza strona *mdot* powstała na początku XXI wieku i wymagała powołania nowego zespołu dedykowanego do jej stworzenia i utrzymywania. Pierwszym powodem było to, że aktualny zespół nie chciał poświęcać czasu na odpowiednik mobilny kosztem właściwej strony. Innym powodem było natomiast to, że strony mobilne były, i nadal mogą być, bardzo pracochłonne, ponieważ muszą wspierać nie tylko najbardziej znane urządzenia z systemami iOS i Android, ale również ogromny wachlarz urządzeń z różnymi rozmiarami ekranów i możliwości, włączając w to brak obsługi JavaScript lub wsparcie tylko dla części jego funkcjonalności.

Nawet jeżeli nie utrzymujesz osobnego zespołu, prace nad stronami *mdot* i tak musisz śledzić jako odrębne od prac nad stroną podstawową; co więcej, w przypadku niektórych telefonów pewne funkcjonalności mogą być niemożliwe do osiągnięcia.

Podzielony kod źródłowy

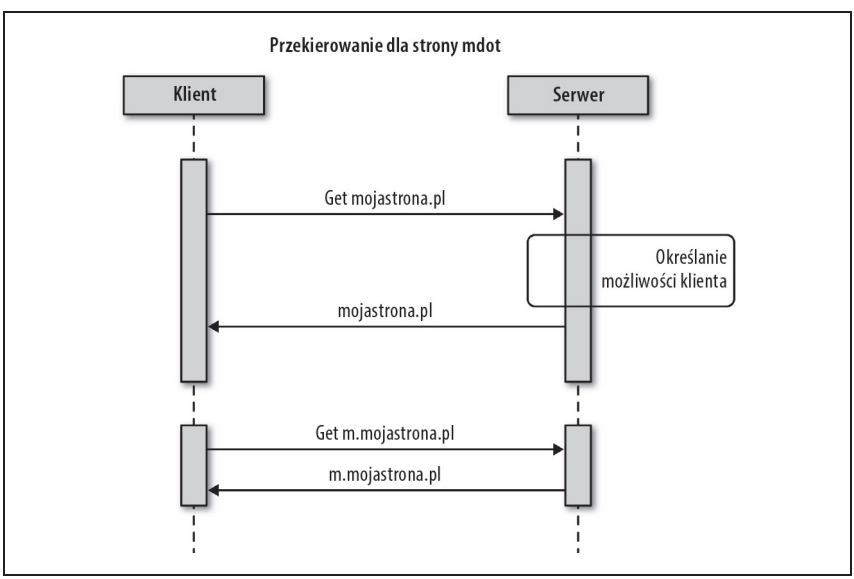
Utrzymywanie odrębnej strony oznacza najprawdopodobniej konieczność utrzymywania odrębnej aplikacji i odrębnego kodu źródłowego. Utrzymanie zgodności pomiędzy kodem źródłowym obu stron to odwieczny problem wymagający czujności i uwagi, co oznacza, że prędzej czy później kody się rozsynchronizują. Kiedy się rozsynchronizują, zawartość poszczególnych stron również będzie różna, a przyszłe aktualizacje będą wymagały większych nakładów pracy.

Podzielony adres URL

Posiadanie odrębnej strony *mdot* oznacza konieczność stworzenia i utrzymania odrębnego adresu. Stoi to w sprzeczności z całą koncepcją adresów URL, które mają być pojedynczą lokalizacją danego zasobu. Strona *mdot* jest kolejną lokalizacją Twojej strony. Co więcej, gdzie znajduje się granica określająca, co ma trafić na stronę *mdot*? Czy telefony? Smartfony? Czy tablety także? A co z phabletami? Czy wszystkie te urządzenia mają mieć wspólną stronę *mdot*, czy będziesz utrzymywał odrębne strony wykorzystywane w zależności od rozmiaru ekranu i możliwości urządzenia? Czy zauważyłeś, jak szybko taka segregacja może stać się uciążliwa?

Bezcelowe przekierowania

Utrzymywanie odrębnych adresów URL oznacza także konieczność dodania przekierowania. Dodanie takiego przekierowania powoduje powstanie dodatkowego ruchu, ponieważ serwer musi odpowiedzieć klientowi statusem 302 lub 304, a klient musi następnie wykonać dodatkowe żądanie dla nowej lokalizacji — proces ten został przedstawiony na rysunku 1.14.



Rysunek 1.14. Odrębne adresy URL dla stron mobilnych wprowadzają konieczność wykonywania przekierowań

To ma znaczenie ze względu na skalę

Do tej pory dyskutowaliśmy głównie o smartfonach i komputerach, ponieważ to one, wraz z tabletami, są najczęściej stosowanymi urządzeniami. Branża ciągle się jednak zmienia i rośnie, a w ciągu ostatnich kilku lat powstało wiele nowych urządzeń, mających nietypowe rozdzielczości, infrastrukturę sieciową i funkcjonalności.

Na przykład, kiedy na rynek został wprowadzony nowy ekran Retina firmy Apple, musieliśmy wspólnie z zespołem projektowym opracować unikatowe obrazy, które wyglądałyby dobrze na urządzeniach wyposażonych w taki ekran. Ten trend będzie kontynuowany — aplikacje sieciowe będą stosowane w telewizorach, a w ich przypadku rozdzielczości ciągle rosną: mamy już telewizory 4K i 8K Ultra HD.

Kiedy okulary Google Glass zaczną być szeroko stosowane, będziemy musieli myśleć o tym, jak nasza aplikacja wygląda także na tym urządzeniu. Na razie Google udostępnia API o nazwie Mirror API oraz biblioteki działające po stronie klienta i współpracujące z Mirror API (*http://bit.ly/1rXkSpb*).

To tylko niektóre z nowości, które niebawem wejdą do powszechnego użytku. Jest ich jednak znacznie więcej.

Jeżeli nadal będziemy wykorzystywać projektowanie responsywne jako narzędzie działające po stronie klienta, problem zbyt dużych stron będzie tylko narastał. Albo więcej firm powróci do korzystania ze stron *mdot*.

Podsumowanie

Rynek powoli obraca się przeciwko projektowaniu responsywnemu. Prawie połowa audytowanych przeze mnie stron to strony dedykowane — z tego rozwiązania korzystaliśmy już na początku XXI wieku — zamiast stron responsywnych.

Projektowanie responsywne nie jest wadliwą metodologią — staje się takie tylko wtedy, jeżeli jest błędnie wykorzystane i traktowane jako dodatek, a nie całościowa filozofia, wtedy też może owocować zbyt dużymi stronami i nieintuicyjnym zachowaniem strony. Podobnie, kiedy skupiamy się na tylko jednym aspekcie responsywności — w szczególności jeżeli jest to front-end — tracimy na wydajności stron responsywnych. Wydajność jest jednak jednym z aspektów responsywności i musimy o nią dbać już na etapie planowania i projektowania. Musi być częścią sposobu budowania przez nas rozwiązań.

W tym rozdziale zidentyfikowaliśmy kilka wzorców projektowych pozwalających wbudować wydajność w responsywność. W następnych rozdziałach omówimy je dokładniej.

Jeżeli tego nie zrobimy i w naszych responsywnych rozwiązaniach nie zadbamy o wydajność, problem będzie się pogarszał wraz z pojawianiem się nowych urządzeń z coraz to większą rozdzielczością i nowymi rozwiązaniami wymagającymi unikalnych interakcji z klientem.

Skorowidz

A

B

C

D

E

F

G

H

I

J

K

L

Ł

M

O

P

Q

R

S

T

U

V

W

Y

Z

Ż

O'REILLY®

Poznaj sposoby tworzenia wydajnych i responsywnych aplikacji internetowych!

1024×768, 800×600 to najpopularniejsze rozdzielczości ekranu w czasach przed mobilną rewolucją. Obecnie użytkownicy korzystają z przeróżnych ekranów, wyświetlających obrazy w wielu innych rozdzielczościach. Jak tworzyć strony WWW, które będą działać poprawnie na każdym dostępnym urządzeniu? Jak rozwiązać problemy z wydajnością? Na te i dziesiątki innych pytań odpowiada ta niezwykła książka.

Poznaj najnowsze trendy w tworzeniu responsywnych i wydajnych projektów webowych. Z tą książką nauczysz się mierzyć wydajność aplikacji i dowiesz się, jak zwiększyć jej poziom i testować ją po stronie klienta oraz serwera. Przekonaj się, jak tworzyć projekty dopasowane do każdego urządzenia i stosować pamięć podręczną. Poznaj dostępne responsywne szkielety oraz ich fachową ocenę dokonaną przez autora. Jest to doskonała lektura dla wszystkich osób podążających za trendami w tworzeniu stron WWW i aplikacji internetowych.

Tom Barker – jest związany z wytwarzaniem oprogramowania od lat 90. XX wieku. Aktualnie pełni funkcję dyrektora ds. wytwarzania oprogramowania w firmie Comcast. Interesuje się analizą danych i ich wizualizacją. Szczególną wagę przykłada do tworzenia kodu wysokiej jakości.

Dzięki tej książce:

- poznasz kluczowe elementy mające wpływ na wydajność aplikacji
- nauczysz się testować wydajność stron i aplikacji internetowych
- zaznajomisz się z narzędziami wspierającymi proces analizy wydajności
- zdobędziesz wiedzę na temat dostępnych szkieletów

Helion

33985 numer katalogowy

księgarnia internetowa

http://helion.pl

zamówienia telefoniczne

0 801 339900

0 601 339900

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-283-0838-1

Informatyka w najlepszym wydaniu

cena: 34,90 zł