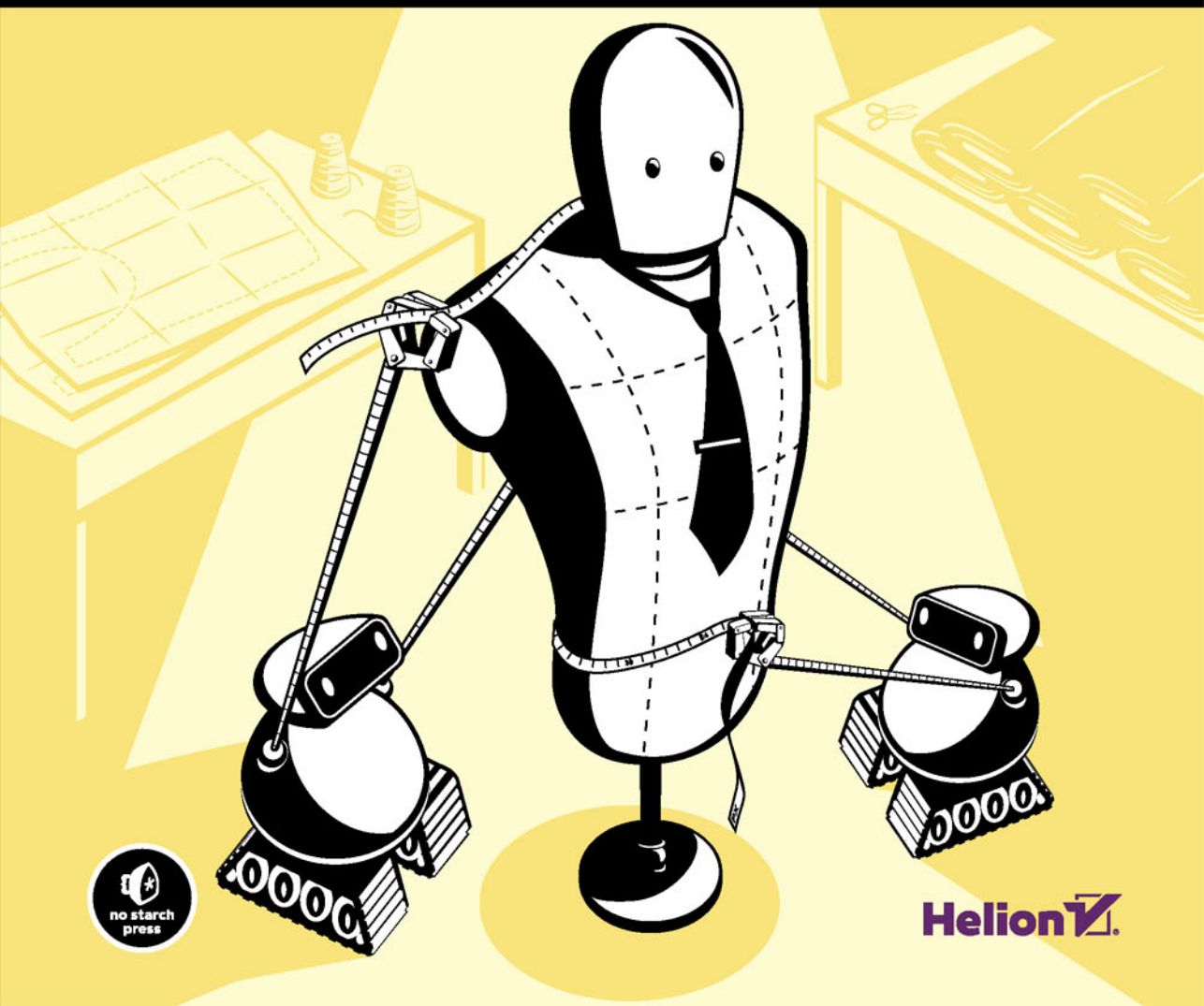
CSS3

PODRĘCZNIK
NOWOCZESNEGO WEBDEVELOPERA

PETER GASSTON



no starch press

Helion

Tytuł oryginału: The Book of CSS3: A Developer's Guide to the Future of Web Design, 2nd Edition

Tłumaczenie: Julia Szajkowska (wstęp, rozdz. 1 – 17), Robert Górczyński (rozdz. 18, 19, dodatki)

ISBN: 978-83-283-0976-0

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/css3pn.zip

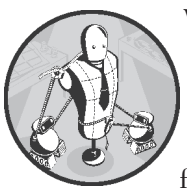
Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/css3pn
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

2

Media Queries — zapytania o media



W CZASACH, GDY STRONY WWW PRZEGLĄDAŁO SIĘ TYLKO W PRZEGLĄDARKACH KOMPUTERÓW STACJONARNYCH I LAPTOPÓW, STOSOWANIE REGUŁ CSS NIE NASTRĘCZAŁO WIĘKSZYCH trudności. Wprawdzie trzeba było zadbać o kompatybilność kodu z różnymi przeglądarkami i upewnić się, że wszystko funkcjonuje jak należy we wszystkich systemach operacyjnych, ale miało się gwarancję, że niemal wszyscy przeglądali stronę na ogół na podobnych urządzeniach. W ciągu kilku ostatnich lat staliśmy się jednak świadkami gwałtownego wzrostu oferty na rynku urządzeń umożliwiających dostęp do internetu. Teraz zasoby sieci ogląda się na konsolach do gier czy urządzeniach przenośnych, takich jak smartfony czy tablety. Wyświetlanie strony zawsze w ten sam sposób nie ma sensu w czasach, gdy odbiorca może równie dobrze korzystać z panoramicznego monitora, jak i wąskiego ekranu urządzenia przenośnego.

Arkusze stylów CSS już od pewnego czasu oferują możliwość przypisywania odmiennych stylów wyświetlania elementów strony różnym rodzajom urządzeń wyjściowych. Wykorzystują do tego atrybut `media` znacznika `link`:

```
<link href="style.css" rel="stylesheet" media="screen">
```

Takie rozwiązanie jednak nie jest pozbawione wad, przede wszystkim zaś jest ono wyjątkowo nieprecyzyjne — ekran, o którym mowa w przedstawionym przykładzie, może mieć przecież przekątną o długości dziewięciu centymetrów albo osiemdziesięciu centymetrów. Ta lista rodzajów sprzętu ma zbyt obszerne kategorie, a wiele urządzeń, które teoretycznie opisuje, w ogóle jej nie rozpoznaje. Na przykład nie słyszałem o ani jednym telewizorze z funkcją przeglądania sieci, który reagowałby poprawnie na rodzaj `tv`. Nic zatem dziwnego, że W3C zaczęło stopniowo odchodzić od korzystania z rodzajów urządzeń wyjściowych.

Specyfikacja CSS3 udostępnia następujące rozwiązanie tego problemu — zapytania o media znajdujące się wewnątrz modułu *Media Queries* (*http://www.w3.org/TR/css3-mediaqueries/*). Moduł ten oferuje składnię zapytania, dzięki której można uzyskać bardzo dokładne informacje o urządzeniu wykorzystywanym przez użytkownika, i w ten sposób rozszerza wyraźnie listę dostępnych typów mediów. Za jego pomocą można dostosować styl wyświetlania strony do potrzeb urządzenia użytkownika. Opis ten mówi może niewiele, ale, wierz mi, moduł *Media Queries* jest jednym z najbardziej rewolucyjnych w całej specyfikacji CSS3. Dzięki niemu wygląd strony internetowej przestaje być ograniczeniem, a użytkownicy mogą cieszyć się jej zawartością niezależnie od urządzenia, na jakim ją przeglądają.

Moduł *Media Queries* otrzymał status modułu rekomendowanego przez W3C, uważa się go więc za część standardu. Wdrożono go we wszystkich najważniejszych przeglądarkach internetowych, w tym także w Internet Explorerze począwszy od wersji 9.

Zalety modułu Media Queries

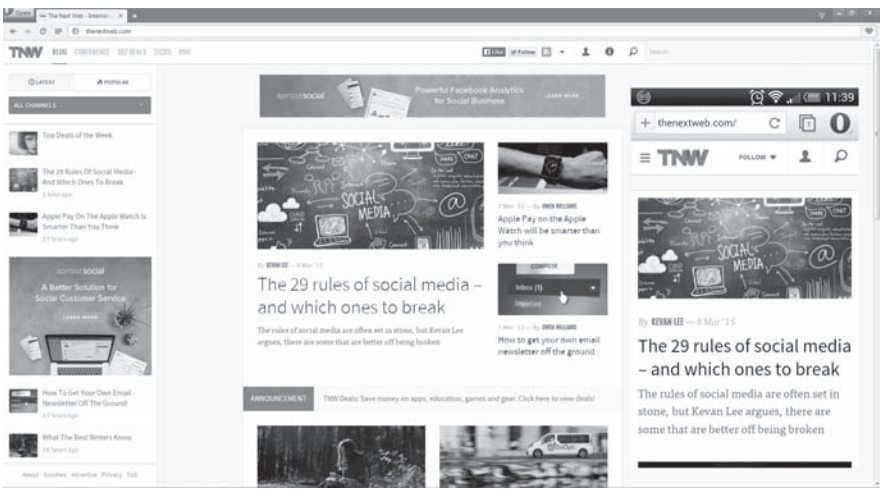
Aby zademonstrować szybko możliwości modułu zapytań o media i pokazać jego elastyczność, zaprezentuję teraz przykład dostosowania wyglądu strony WWW do specyficznego środowiska, jakim jest przeglądarka mobilna. Działanie to, jak się wkrótce przekonasz, nie będzie wymagało wielkiego wysiłku.

Nie można wykluczyć, że odwiedzający Twoją stronę za pomocą urządzeń mobilnych bardzo się męczą. Powody mogą być różne — zbyt mały rozmiar tekstu czy konieczność przewijania dużych fragmentów strony po jej powiększeniu, by dostać się do menu nawigacyjnego. Z kolei samo menu może wykorzystywać funkcję rozwijania po wskazaniu jego elementu kursorem myszy, która to funkcja często nie występuje w urządzeniach przenośnych. Tego rodzaju urządzenia pracują zazwyczaj na słabych łączach internetowych, więc pobieranie grafiki o większych rozmiarach automatycznie staje się problemem — po pierwsze, ze względu na czas pobierania, a po drugie, z powodu zużywania znacznej części pakietu internetowego użytkownika. Twórcy niektórych stron starają się rozwiązać ten problem, przygotowując nowe wersje witryn, projektowane specjalnie z myślą o urządzeniach przenośnych, lecz wiąże się to zazwyczaj z koniecznością wykonania mnóstwa dodatkowej pracy. Należy zdefiniować poddomenę przeznaczoną na taką stronę, umieścić na niej pliki arkuszy stylów i szablony

HTML inne niż te, z których korzysta strona podstawowa. Następnie należy zmienić rozmiary grafik tak, by ilustracje pasowały do mniejszych ekranów, i przygotować skrypt, który rozpozna urządzenie otwierające stronę i w razie potrzeby skieruje użytkownika na przenośną wersję strony. Takie rozwiązanie może powodować wystąpienie dodatkowych problemów — skrypt musi obsługiwać najnowsze wersje wszystkich przeglądarek przenośnych, a aktualizowanie zawartości witryny wiąże się zazwyczaj z duplikowaniem jej zawartości, tylko bowiem w ten sposób można zapewnić sobie zgodność obu wersji strony.

Moduł *Media Queries* pozwala rozwiązać wiele spośród tych problemów. Przede wszystkim zawarte w nim zapytania pozwalają określić rodzaj urządzenia na podstawie jego atrybutów, zwalniają Cię więc z konieczności użycia skryptów badających parametry przeglądarki. Zapytania o media pozwalają wybrać zbiór reguł stylów dostosowanych dokładnie do możliwości danego urządzenia. Jeśli więc zapytanie wykryje urządzenie o małym ekranie, strona zostanie wyświetlona według wskazań odpowiednio dobranego zestawu reguł stylów, który zadba o to, by zniknęły z niej zbędne elementy, wyświetlane grafiki były odpowiednio mniejsze, a tekst — czytelny.

Przykładem strony, na której zastosowano takie rozwiązanie, jest *The Next Web* (*http://thenextweb.com*) przedstawiona na rysunku 2.1.



Rysunek 2.1. *Strona internetowa The Next Web wyświetlana w przeglądarce zainstalowanej na komputerze i na telefonie komórkowym*

W wersji przeznaczonej do wyświetlania w oknie przeglądarki komputerowej na górze strony pojawia się długie, poziome menu, po lewej wyświetlane są odnośniki do artykułów powiązanych, a odsyłacze prowadzące do głównej zawartości strony pojawiają się w układzie siatki. Gdy jednak zapytania o media wykryją, że strona została otwarta w węższym oknie, na przykład w przeglądarce zainstalowanej na telefonie komórkowym, sposób jej wyświetlania zmienia się diametralnie — opcje udostępniania zawartości zostają ukryte, odnośniki do

powiązanych artykułów również znikają z ekranu, a zawartość strony głównej zostaje wyświetlona w jednej kolumnie, co współgra doskonale z nawigowaniem wewnątrz telefonu, polegającym przede wszystkim na przewijaniu zawartości strony w dół i w górę.

Oczywiście obecnie strony WWW są wyświetlane nie tylko na komputerach i telefonach komórkowych. Oznacza to, że powinniśmy dołożyć wszelkich starań, by ostatecznie wszystkie strony internetowe zostały przygotowane do wyświetlania na dowolnym urządzeniu. Więcej na ten temat znajdziesz w ramce „Dynamiczny układ strony" poniżej.

Jeśli chcesz się przekonać, jak ludzie wykorzystują możliwości modułu zapytań o media, zapoznaj się z zawartością wspaniałej galerii dostępnej na stronie *http://mediaqueri.es/*. Znajdziesz tam kilka naprawdę niezwykle interesujących przykładów tego, o czym tu piszę.

DYNAMICZNY UKŁAD STRONY

W 2010 roku Ethan Marcotte napisał artykuł zatytułowany *Responsive Web Design* (*http://alistapart.com/article/responsive-web-design/*), w którym nad wyraz sprawnie przedstawił współczesne poglądy na temat tworzenia stron internetowych tak, by ich wygląd przystosowywał się do możliwości urządzenia, na którym są wyświetlane. Oto co możemy przeczytać w tym artykule:

> Dziś bardziej niż kiedykolwiek mamy w pamięci, że wyniki naszej pracy będą oglądane na różnych urządzeniach. Projektowanie dynamiczne, reagujące na zmieniające się warunki otoczenia, pozwala nam wreszcie „tworzyć na dobre i na złe".

Od czasu pojawienia się tego artykułu projektowanie dynamiczne stało się normą. Dziś większość projektantów stron internetowych ma na uwadze mnogość dostępnych urządzeń wyjściowych, przygotowuje więc swoje strony tak, by zachowywały się na nich właściwie. Również stare strony są coraz częściej dostosowywane do nowych standardów. Tego rodzaju podejście wymaga pewnych dodatkowych zabiegów — szczególnie starannie rozważa się wszystkie za i przeciw zwłaszcza w przypadku projektowania całych witryn, bo większość dostępnych narzędzi nadal nie umożliwia łatwego tworzenia dynamicznie zmieniających się stron — ale możemy powiedzieć, że powoli dążymy do osiągnięcia celu, jakim jest internet dostępny dla każdego i wszędzie, niezależnie od tego, jakim urządzeniem do przeglądania jego zasobów posługiwałby się użytkownik.

Składnia

Integralną częścią zapytania o media jest parametr (a czasami cały ich zestaw) umożliwiający określenie, który zestaw reguł stylów CSS należy wczytać dla danego urządzenia. O wyborze decydują właściwości urządzenia, a konkretnie to, czy odpowiadają wartościom podanym w zestawie parametrów. Zapytania o media

stosuje się na jeden z trzech sposobów odpowiadających sposobom wprowadzania stylów CSS do dokumentu. Pierwszy z nich polega na wywołaniu zewnętrznego arkusza stylów za pomocą znacznika `link`:

```
<link href="plik" rel="stylesheet" media="wyrażenie-logiczne wyjście and
(wyrażenie)">
```

Drugim sposobem jest wywołanie zewnętrznego arkusza stylów przy użyciu polecenia `@import`:

```
@import url('plik') wyrażenie-logiczne wyjście and (wyrażenie);
```

Trzeci sposób polega na wprowadzeniu zapytania o media wewnątrz osadzonego znacznika stylu lub z arkusza za pomocą rozszerzonej reguły `@media`:

```
@media wyrażenie-logiczne wyjście and (wyrażenie) { reguły }
```

Z tej właśnie metody będę korzystać do końca rozdziału, ponieważ łatwiej demonstruje się za jej pomocą pewne koncepcje. To, z której metody skorzystasz Ty, będzie zależeć głównie od Twoich preferencji oraz od wymogów narzuconych przez istniejącą strukturę arkuszy stylów.

Skoro przedstawiłem już metody wskazywania medium, zajmę się teraz omówieniem składni. Atrybut `media` powinien być Ci znany — służy deklarowaniu rodzaju medium, do którego należy zastosować odpowiedni zestaw arkuszy stylów, jak na przykład w tym znaczniku HTML `link`:

```
<link href="style.css" rel="stylesheet" media="screen">
```

Najczęściej spotykanymi wartościami przyjmowanymi przez atrybut `media` są `screen` i `print`. Obecnie stosowana składnia pozwala podawać je w postaci listy oddzielanej przecinkami. Wtedy w jednym wyrażeniu można podać kilka zapytań (to jednak staje się coraz mniej stosowanym rozwiązaniem, ponieważ inne rodzaje mediów wychodzą z użycia). Jeśli pominiemy rodzaj medium, atrybut `media` przyjmie domyślną wartość `all`, tak więc gdy układasz właśnie regułę, która ma odnosić się do wszystkich rodzajów mediów, nie musisz wyszczególniać ich w konstruktorze zapytania. Podane niżej przykłady zadziałają tak samo:

```
@media all and (wyrażenie) { reguły }
@media (wyrażenie) { reguły }
```

UWAGA *Aby przykładowy kod w dalszej części książki miał bardziej spójną postać, tam, gdzie będzie to możliwe, będę pomijać rodzaj mediów.*

Pierwszym z nowych atrybutów reguły `@media` jest *`wyrażenie-logiczne`*. Przyjmuje on wartość jednego z dwóch dopuszczalnych słów kluczowych — `only` lub `not` — i jest parametrem opcjonalnym.

```
@media only wyjście and (wyrażenie) { reguły }
@media not wyjście and (wyrażenie) { reguły }
```

Wartość `only` wykorzystuje się przede wszystkim do ukrywania reguły przed starszymi wersjami przeglądarek, w których składnia ta nie została wdrożona. Przeglądarki ją rozpoznające ignorują atrybut o wartości `only`. Wartość `not` służy do zanegowania zapytania. Używa się jej, by zastosować określone regułami style, jeśli atrybuty urządzenia wyjściowego *nie* odpowiadają liście podanej w poleceniu.

Jeśli użyjesz w zapytaniu atrybutów *`wyrażenie-logiczne`* lub *`wyjście`*, będziesz musiał wprowadzić do niego także operator `and`, co pokazują zresztą poprzednie przykłady. Operator ten pozwala łączyć atrybuty z odpowiednim atrybutem *`wyrażenie`*. Atrybut ten służy do określania parametrów, których działanie wychodzi poza funkcje oferowane przez atrybut *`wyjście`*. Wspomniane parametry określane są wspólną nazwą **cech mediów**, i to właśnie one decydują o tym, że zapytania o media są tak potężnym narzędziem. Pora zatem przyjrzeć się im z bliska.

Cechy mediów

Cechy mediów przechowują informacje dotyczące urządzeń, na których wyświetlana jest strona internetowa: ich wymiary, rozdzielczość i tym podobne. Informacje te są wykorzystywane do określania wartości wyrażenia *`wyrażenie`*, a uzyskany w ten sposób wynik staje się podstawą do wybrania odpowiedniej reguły stylu. Wyrażenie *`wyrażenie`* może przekazywać na przykład następujący komunikat: „zastosuj te style tylko, jeśli ekran urządzenia jest szerszy niż 480 pikseli albo tylko jeśli urządzenie jest ustawione poziomo”.

Większość wyrażeń pojawiających się w zapytaniach o media wymaga podania pewnej wartości:

```
@media (cecha: wartość) { reguły }
```

Wartość ta jest niezbędna do ułożenia przykładowych wyrażeń, jakie podałem nieco wcześniej. Istnieją jednak przypadki, w których można ją pominąć i sprawdzić jedynie, czy medium charakteryzuje się konkretną cechą. Wtedy otrzymujemy wyrażenie:

```
@media (cecha) { reguły }
```

Kwestia wyrażeń wyjaśni się na pewno, gdy zacznę omawiać konkretne cechy mediów oraz gdy powiem, które wartości są konieczne do działania strony, a z których można zrezygnować.

Skoro poznaliśmy już tajniki składni, pora zająć się najważniejszymi cechami mediów. Te, o których będę teraz opowiadać, odnoszą się przede wszystkim do kolorowych wyświetlaczy, na których najczęściej przeglądamy zawartość sieci. Istnieją też inne cechy mediów, ale będziesz je wykorzystywać głównie do obsługi innych mediów, na przykład telewizorów czy terminali o ustalonych sztywno parametrach (o ile urządzenia te w ogóle pozwolą na korzystanie ze wspomnianych cech).

Szerokość i wysokość

Cecha `width` opisuje szerokość pola rysowania zawartości dla określonego rodzaju medium, co w praktyce przekłada się na szerokość okna przeglądarki (razem z polem paska przewijania) zainstalowanej w systemie komputera. Podstawowa składnia wymaga podania długości:

```
@media (width: 600px) { reguły }
```

W tym przypadku reguły znajdą zastosowanie tylko w tych przeglądarkach, których okno będzie miało dokładnie 600 pikseli szerokości, co oczywiście jest zbyt szczegółowym określeniem warunku. Cecha `width` przyjmuje też jeden z dwóch przedrostków: `max-` lub `min-`, które pozwalają określać maksymalną i minimalną wartość cechy:

```
@media (max-width: 480px) { reguły }
@media (min-width: 640px) { reguły }
```

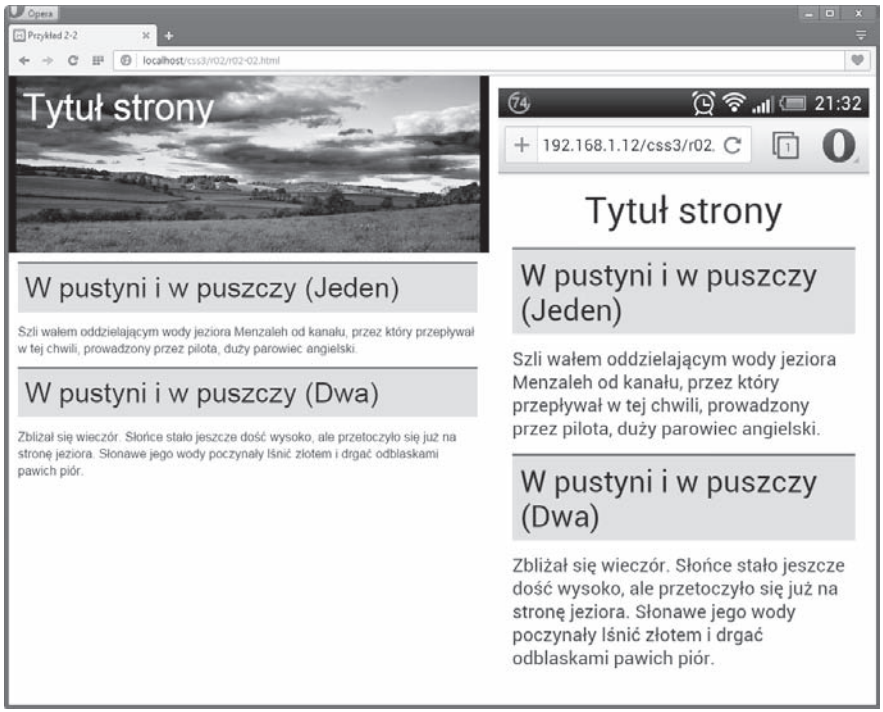
Pierwsze zapytanie spowoduje użycie zestawu reguł we wszystkich przeglądarkach, których okno nie jest szersze niż 480 pikseli, drugie zaś w tych, w których szerokość okna nie przekracza 640 pikseli.

Zajmijmy się teraz przykładem praktycznym. Wykorzystam tu możliwość sprawdzania szerokości okna przeglądarki do wyświetlania ozdobnego nagłówka strony. Pojawi się on tylko w szerszych oknach (żeby nie zaciemniać przykładu, niektóre reguły zostały usunięte):

```
@media (min-width: 400px) {
  h1 { background: url('landscape.jpg'); }
}
```

To zapytanie o media sprawdza, czy okno przeglądarki ma co najmniej 400 pikseli szerokości, i jeśli tak jest, to umieszcza zdjęcie w elemencie `h1` kodu HTML.

Jeśli okno przeglądarki ma przynajmniej 400 pikseli, nagłówek strony ozdobiony jest zdjęciem. Gdy okno okazuje się węższe, w nagłówku wyświetla się wyłącznie tekst. Przykład ten można zobaczyć na rysunku 2.2.



Rysunek 2.2. *Różne reguły stylów zastosowane za pomocą cechy medium width. Ta sama strona przedstawiona w przeglądarce komputerowej i w aplikacji mobilnej*

Cecha `height` działa w ten sam sposób. Jedyna różnica polega na tym, że za pomocą tej wartości narzucamy warunek dotyczący wysokości okna przeglądarki. Składnia pozostaje taka sama. Tu również możemy korzystać z przedrostków `max-` oraz `min-`.

```
@media (height: wartość) { reguły }
@media (max-height: wartość) { reguły }
@media (min-height: wartość) { reguły }
```

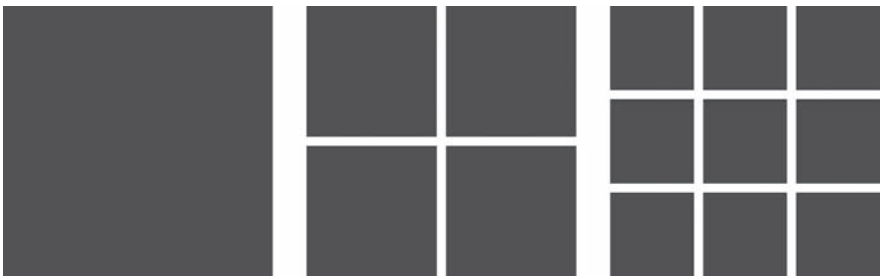
Ponieważ jednak przewijanie w pionie jest znacznie częściej wykorzystywaną metodą nawigowania po stronach, z cechy `height` korzysta się znacznie rzadziej niż z cechy `width`.

Rozdzielczość sprzętowa

W arkuszach stylów CSS pod pojęciem piksela (px) rozumie się obiekt odpowiadający jednemu pikselowi na ekranie komputera. Jeśli obszar wyświetlania w przeglądarce ma 1024 piksele szerokości, a Ty umieścisz na stronie WWW element o szerokości 1024 pikseli, to będziesz słusznie zakładać, że wypełni on całą szerokość okna przeglądarki. Jednakże w wielu nowych urządzeniach, przede wszystkim zaś w smartfonach i tabletach, montuje się ekrany o bardzo dużej rozdzielczości. Wyświetlany na nich element o szerokości 1024 pikseli będzie niewielki, a jego szczegóły w zasadzie nieczytelne.

Aby ominąć wspomniany problem, w urządzeniach tych bardzo często definiuje się nominalny piksel arkusza stylów CSS, który nie ma nic wspólnego z fizycznymi rozmiarami piksela na ekranie urządzenia. W ten sposób zyskujemy możliwość powiększania i zmniejszania zawartości strony nawet na bardzo małych ekranach, nie tracąc przy tym jej czytelności. Stosunek rozmiarów fizycznych pikseli do rozmiarów pikseli arkuszy stylów jest znany jako **rozdzielczość sprzętowa** (DPR — ang. *device pixel ratio*). Przykładowo iPhone 5S ma DPR równy 2, co oznacza, że jeden piksel CSS jest równy czterem pikselom fizycznym: dwóm w pionie i dwóm w poziomie.

Opisywany przykład został zilustrowany na rysunku 2.3. Po lewej stronie widać, w jaki sposób piksel CSS jest wyświetlany na „zwykłym" ekranie o rozdzielczości sprzętowej 1:1. Środkowa ilustracja przedstawia taki sam piksel CSS na ekranie o DPR wynoszącym 2, jak we wspomnianym przypadku iPhone'a — tu tę samą przestrzeń zajmują cztery piksele. Ostatnia ilustracja, ta po prawej stronie, pokazuje, jak piksel CSS wyglądałby na ekranie o DPR wynoszącym 3, na przykład takim jak w Nexusie 5. Tu jednemu pikselowi w arkuszu stylów odpowiada dziewięć pikseli na ekranie.



Rysunek 2.3. *Piksel CSS na ekranie o rozdzielczości sprzętowej: 1:1 (po lewej), DPR równym 2 (środek) i DPR równym 3 (po prawej)*

W praktyce oznacza to, że choć fizyczna rozdzielczość ekranu iPhone'a 5S (przykładowo) wynosi 640×1136, to jego rozdzielczość CSS wynosi zaledwie 320×568, czyli dokładnie połowę, bo każdy piksel CSS odpowiada dwóm fizycznym pikselom. (Należy jednak pamiętać, że zależność ta jest prawdziwa wyłącznie wtedy, gdy urządzenie pracuje w „trybie mobilnym"; więcej na ten temat znajdziesz w punkcie „Szerokość i wysokość ekranu urządzenia").

Niestety duże wartości DPR, choć powodują, że skalowalna zawartość stron internetowych (na przykład tekst czy grafiki wektorowe) jest zawsze ostra i wyraźna, to jednocześnie sprawiają, że bitmapy wyglądają na ekranach o takich rozdzielczościach nie najlepiej. Ten problem rozwiązuje się za pomocą nowej cechy medium, `resolution`, która pozwala wybierać urządzenia w zależności od ich DPR:

```
@media wyjście and (resolution: wartość) { reguły }
```

Wartość cechy `resolution` to liczba określająca rozdzielczość w jej typowych jednostkach: **DPI** (ang. *dots per inch* — **punkty na cal**), **DPCM** (ang. *dots per centimeter* — **punkty na centymetr**) czy z naszego punktu widzenia najwygodniejszej — **DPPX** (ang. *dots per pixel* — **punkty na piksel**). Jednostka DPPX oddaje rozdzielczość sprzętową urządzenia DPR, więc aby zapisać regułę, która zostanie zastosowana wyłącznie w urządzeniach o wartości DPR wynoszącej `1.5`, należałoby posłużyć się następującym poleceniem:

```
@media (resolution: 1.5dppx) { reguły }
```

Jak w przypadku pozostałych cech medium i tu możemy wykrywać maksymalne i minimalne rozdzielczości:

```
@media (max-resolution: liczba) { reguły }
@media (min-resolution: liczba) { reguły }
```

Taka elastyczność znacznie ułatwia dostarczanie obrazów o wyższych rozdzielczościach do przeglądarek charakteryzujących się większą gęstością pikseli, co widać na przykładzie podanego tu fragmentu kodu:

```
❶ E { background-image: url('image-lores.png'); }
❷ @media (min-resolution: 1.5dppx) {
    background-image: url('image-hires.png');
❸    background-size: 100% 100%;
  }
```

Pierwsza z reguł (❶) stwierdza, że w przeglądarkach działających w standardowej (niskiej rozdzielczości) użyjemy zwykłej grafiki (*image-lores.png*). Natomiast jeśli strona internetowa zostanie wyświetlona na urządzeniu z DPR wynoszącym co najmniej 1,5, użyjemy grafiki o wyższej rozdzielczości (*image-hires.png*; ❷). Zauważ też, że w poleceniach pojawiła się nieznana nam dotąd właściwość `background-size` (❸). Należy używać jej w czasie wyświetlania obrazów o dużych rozdzielczościach, by zagwarantować, że nie wyświetlą się one poza obszarem znacznika, do którego są przypisywane. (O właściwości `background-size` opowiem więcej w rozdziale 8.).

Chrome, Firefox i Internet Explorer 10 (oraz wyższe wersje tej przeglądarki) rozpoznają polecenie `resolution`, choć w przypadku IE nie można mówić o pełnej funkcjonalności, bo przeglądarka ta nie rozpoznaje poprawnie wartości zapisanych jako DPPX. Aby strona wyświetlała się poprawnie w IE, musisz podawać rozdzielczość w DPI, czyli mnożyć DPR przez 96 (tyle wynosi DPI typowego ekranu). Oto przykład:

```
@media (resolution: 1.5dppx), (resolution: 144dpi) { reguły }
```

Z kolei Safari nie rozpoznaje polecenia `resolution` i wykorzystuje zamiast niego cechę zwaną `-webkit-device-pixel-ratio` (dostępną także z przedrostkami `max-` i `min-`). Jej wartości podaje się bez jednostek, przyjmując jednostkę domyślną DPR. Zatem jeśli reguła ma być poprawnie zinterpretowana przez wszystkie liczące się obecnie przeglądarki, musi wyglądać następująco:

```
@media (resolution: 1.5dppx), (resolution: 144dpi),
(-webkit-device-pixel-ratio: 2) { reguły }
```

Parametr `resolution` został wprowadzony do silnika WebKit pod koniec 2012 roku, dziwi mnie więc, że w ciągu prawie dwóch lat, bo tyle czasu minęło od tamtej chwili do rozpoczęcia prac nad tą książką, nie został on zaimplementowany w przeglądarce Safari. Można jedynie liczyć, że to niedopatrzenie zostanie wkrótce naprawione.

Szerokość i wysokość ekranu urządzenia

Parametry medium `width` i `height` opisują wymiary okna przeglądarki, ale przecież nie zawsze jest ono tak duże jak ekran urządzenia. Jeśli interesują Cię parametry ekranu, a nie okna programu, jakie się na nim pojawia, możesz skorzystać z właściwości `device-width` i `device-height` oraz ich wersji poprzedzonych przedrostkami `max-` i `min-`. Parametrów tych nie używa się zbyt często, lecz by odpowiedzieć na pytanie dlaczego, muszę uciec się do niewielkiej dygresji.

W poprzednim punkcie wyjaśniałem, czym różni się piksel w rozumieniu arkuszy stylów CSS od piksela fizycznego. Właściwość medium `width` jest mierzona w pikselach CSS, natomiast wartość właściwości `device-width` mierzymy w pikselach fizycznych. Aby zawartość strony była czytelna i miała „naturalnie wyglądający" rozmiar, obie te wielkości muszą sobie odpowiadać. W tym celu w nagłówku dokumentu umieszcza się **znacznik meta `viewport`**:

```
<meta name="viewport" content="width=device-width">
```

Gdy znacznik meta `viewport` zawierający wspomniane wartości pojawia się w sekcji `head` strony, przeglądarki zainstalowane na urządzeniach mobilnych przechodzą w „tryb mobilny", w którym okno przeglądarki zostaje dopasowane

rozmiarem dokładnie do rozmiaru ekranu. W efekcie zawartość jest wyświetlana w sposób lepiej dopasowany do możliwości urządzenia.

UWAGA *Głębszą analizę problemu związanego z zagadnieniami obszaru wyświetlania i rozmiarem pikseli znajdziesz na stronie holenderskiego projektanta stron internetowych publikującego pod pseudonimem PPK: „A pixel is not a pixel" (http://www.quirksmode.org/blog/archives/2010/04/a_pixel_is_not.html).*

Obszar wyświetlania w przeglądarce zainstalowanej na urządzeniu przenośnym ma zazwyczaj wymiary jego wyświetlacza, więc obie wspomniane tu wartości są w tym przypadku identyczne. Z kolei w przeglądarkach komputerowych wymiary treści dostosowuje się zazwyczaj do szerokości i wysokości okna, a nie do rozmiarów ekranu. Dlatego też właściwość `device-width` ma znacznie węższe zastosowanie niż właściwość `width` i raczej nie będziesz jej nadmiernie stosować w praktyce.

Znacznik meta `viewport` wszedł do standardu CSS jako reguła `@viewport`. Więcej na jego temat znajdziesz w podrozdziale „Obsługa w urządzeniach" w rozdziale 19.

Orientacja

Jeśli bardziej niż rzeczywiste wymiary urządzenia interesuje Cię jego ułożenie w przestrzeni, i to właśnie do niego chcesz dostosowywać wygląd zawartości strony, musisz odwołać się do cechy `orientation`. Cecha ta pozwala określić, czy strona ma być wyświetlana w poziomie (jak w każdej przeglądarce zainstalowanej na komputerze stacjonarnym czy laptopie), czy raczej w pionie (jak w telefonach komórkowych czy czytnikach książek). Oto składnia potrzebna do skorzystania z tej cechy:

```
@media (orientation: wartość) { reguły }
```

Parametr `orientation` może przyjmować jedną z dwóch wartości: `landscape` lub `portrait`. Wartość `landscape` odpowiada sytuacji, w której szerokość przeglądarki jest większa od jej wysokości, natomiast wartość `portrait` opisuje sytuację przeciwną. Parametr ten jest przydatny podczas tworzenia stron z myślą o przeglądarkach komputerowych, ale swój rzeczywisty potencjał ujawnia dopiero, gdy przychodzi do uwzględniania potrzeb urządzeń przenośnych, których jedną z głównych cech jest to, że sposób wyświetlania dostosowują do ułożenia w przestrzeni (użytkownik może je obracać). Przykładami takich urządzeń są smartfony i tablety.

Za pomocą cechy `orientation` można na przykład zmieniać sposób wyświetlania menu z poziomego na pionowy w zależności od ułożenia okna przeglądarki. Odpowiedzialny za to kod ma następującą postać:

```
ul { overflow: hidden; }
li { float: left; }
@media (orientation: portrait) {
  li { float: none; }
}
```

Domyślnie elementy zawarte w znacznikach `li` są wyrównywane do lewej przy użyciu cechy `float` o wartości `left`. Reguła ta zmusza je do ułożenia się w poziomie wzdłuż szerokości okna przeglądarki. Gdy jednak strona jest przeglądana w urządzeniu, dla którego parametr `orientation` przyjmuje wartość `portrait` — efekt ten uzyskuje się albo przez przeskalowanie okna przeglądarki tak, by jego wysokość była większa od szerokości, albo przez wyświetlenie strony w urządzeniu, które ma taką właśnie orientację — cecha `float` otrzymuje wartość `none`, co sprawia, że elementy objęte znacznikami `li` zostają wyświetlone w pionie. Efekt widać na rysunku 2.4.

Rysunek 2.4. *Cecha orientation wykorzystana do wyświetlania zawartości strony w przeglądarce urządzenia przenośnego: wartość portrait (z lewej) i landscape (z prawej)*

Ponieważ istnieją tylko dwie dopuszczalne wartości cechy `orientation`, to podanie przy jednej z nich określonego zestawu reguł sprawi, że druga przyjmie reguły przeciwne. W przedstawionym przykładzie użyłem wyłącznie wartości `portrait`, więc wszystkie reguły znajdujące się poza tą cechą zostaną automatycznie przypisane wartości `landscape`.

Proporcje

Moduł zapytań o media pozwala tworzyć warunki, które zostaną zastosowane, gdy medium wyjściowe będzie charakteryzować się określonym stosunkiem szerokości do wysokości. Cecha `aspect-ratio` pozwala sprawdzić proporcje okna przeglądarki, a `device-aspect-ratio` określa proporcje ekranu. Oto składnia umożliwiająca skorzystanie z tych cech:

```
@media (aspect-ratio: w-poziomie/w-pionie) { reguły }
@media (device-aspect-ratio: w-poziomie/w-pionie) { reguły }
```

Wartości kryjące się za określeniami *w-poziomie* i *w-pionie* to dodatnie liczby całkowite obrazujące stosunek szerokości do długości ekranu urządzenia lub okna przeglądarki. Kwadratowy wyświetlacz byłby opisany wartościami 1/1, a szeroki ekran kinowy — 16/9.

UWAGA *Niektóre urządzenia — między innymi iPhone'y — zgłaszają, że ekran urządzenia ma zawsze proporcje portretowe, nawet jeśli akurat działa w orientacji pejzażowej.*

Jednak wybieranie zestawu reguł na podstawie proporcji ekranu urządzenia to potencjalnie źródło wielu problemów. Przykładowo ekran kinowy ma według niektórych producentów proporcje 16/9, według innych — 16/10, a dla jeszcze innych — 15/10. Co więcej, rzeczywiste proporcje ekranu urządzenia mogą być odmienne niż deklarowane przez producenta. Na przykład iPhone 5S ma rzekomo proporcje ekranu 16/9, ale zgłasza faktycznie, że wynoszą one nieco więcej niż 40/71 (w układzie portretowym). Dlatego cech `aspect-ratio` i `device-aspect-ratio` warto używać z przedrostkami `max-` i `min-`. Przyjrzyj się poniższemu poleceniu, które sprawia, że reguły podane w zapytaniu zostaną zastosowane na każdym urządzeniu o proporcjach ekranu większych niż 16/9:

```
@media (min-device-aspect-ratio: 16/9) {...}
```

Łączone cechy mediów

Dla jednego rodzaju medium zapytania można łączyć w większe grupy. W tym celu należy wprowadzić między nie operator `and`:

```
@media wyrażenie-logiczne wyjście and (wyrażenie) and (wyrażenie) { reguły }
```

Zgodnie z tą składnią styl zostanie zastosowany dopiero wtedy, gdy spełnione będą oba podane warunki. Przykładowo, aby sprawdzić, czy ekran urządzenia, którego proporcje nie przekraczają 15/10, jest wąski, należy zapisać następujący warunek:

```
@media (max-device-aspect-ratio: 15/10) and (max-width: 800px) {...}
```

W wyrażeniu może też pojawić się operator warunkowy, który w tym przypadku przyjmuje postać przecinka:

```
@media wyrażenie-logiczne wyjście and (wyrażenie), wyrażenie-logiczne
wyjście and (wyrażenie) { reguły }
```

W tej sytuacji reguły zostaną zastosowane zawsze, gdy spełniony zostanie choć jeden z zapisanych warunków. W poniższym przykładzie będzie to dotyczyć stron wyświetlanych w urządzeniu ułożonym poziomo lub stron drukowanych w układzie pionowym:

```
@media screen and (orientation: landscape), print and (orientation:
portrait) {...}
```

Naturalnie możesz też łączyć te zapisy w dowolny sposób.

Najpierw na urządzenia przenośne

Obecnie dobrym nawykiem programistycznym jest przygotowanie strony internetowej **najpierw na urządzenia przenośne**, czyli w taki sposób, by jej zawartość na pewno wyświetlała się poprawnie na mniejszych ekranach. Dopiero po osiągnięciu tego celu dodaje się większe składniki strony i tworzy bardziej skomplikowane rozwiązania przeznaczone dla użytkowników większych ekranów.

Praktyka ta powstała w reakcji na sposób wczytywania przez przeglądarki niektórych elementów strony określonych w regułach stylów. Kłopoty pojawiały się, gdyż pionierzy stosowania zapytań o media mieli w zwyczaju, na przykład, umieszczać duże grafiki w tle strony i dopiero potem chować je na mniejszych urządzeniach za pomocą odpowiedniej reguły:

```
E { background-image: url('huge-image.jpg'); }
@media (max-width: 600px) {
  E { display: none; }
}
```

Obrazy te, choć niewidoczne dla użytkownika, były tak czy inaczej pobierane przez przeglądarkę i przechowywane w jej pamięci podręcznej. Takie rozwiązanie zwiększało czas ładowania strony i zużywało limit transferu, a przez to było bardzo niekorzystne z punktu widzenia użytkowników niemających dostępu do bezprzewodowego połączenia internetowego.

Projektowanie stron najpierw na urządzenia przenośne to filozofia zakładająca utworzenie podstawowego arkusza stylów, który sprawdzi się we wszystkich przeglądarkach, również tych instalowanych na urządzeniach przenośnych. Dopiero później wprowadza się kolejne rozszerzenia przygotowane dla użytkowników urządzeń o większych ekranach. Odpowiedzialne za ich dodawanie reguły są umieszczane w zapytaniach o media, w których pojawia się cecha `min-width`:

```
@media (min-width: 600px) {
   E { background-image: url('huge-image.jpg'); }
}
```

Taka zmiana podejścia gwarantuje, że wspomniana wcześniej grafika tła nigdy nie zostanie pobrana do pamięci urządzenia o zbyt małym ekranie. Rozwiązanie takie można wykorzystać do wprowadzania całych arkuszy stylów:

```
<link href="basic.css" rel="stylesheet">
<link href="desktop.css" rel="stylesheet" media="(min-width: 600px)">
```

Rozdzielone w ten sposób arkusze stylów są wczytywane w niektórych przeglądarkach bardziej sprawnie. Na przykład w przeglądarce Chrome wczytanie arkusza *desktop.css* — niepotrzebnego w urządzeniach o ekranie węższym od 600 pikseli — zostanie przesunięte do momentu, w którym przeglądarka pobierze inne, ważniejsze elementy strony. To bardzo wygodne rozwiązanie.

Metoda projektowania najpierw z myślą o urządzeniach przenośnych sprawdza się doskonale w większości dostępnych na rynku przeglądarek od kilku lat. Tylko naprawdę stare wersje przeglądarek nie rozpoznają odpowiednio poleceń i wczytują zawsze arkusz *basic.css*. Zresztą może lepiej dla nich, bo tak wiekowe aplikacje pewnie nie poradziłyby sobie z zaawansowanymi możliwościami stron, o których opowiem w dalszych rozdziałach.

Podsumowanie

Składnia zapytań o media nie jest skomplikowana, lecz nie zmniejsza to w niczym możliwości, jakie oferują te narzędzia. Coraz większe zainteresowanie korzystaniem z zasobów sieci na urządzeniach przenośnych sprawiło, że programiści i projektanci zaczęli rozumieć, jak ważne jest przygotowywanie zawartości stron w sposób inny niż dotychczas — bez wykorzystywania skryptów badających możliwości przeglądarki czy tworzenia oddzielnej wersji strony na urządzenia przenośne (i zupełnie różnej od wersji „komputerowej").

Wzrost popularności ruchu związanego z projektowaniem stron zmieniających się dynamicznie, jaki obserwujemy od kilku lat, jest bezpośrednio spowodowany rozwinięciem możliwości oferowanych przez zapytania o media. Zapytania te nie potrzebowały wiele czasu, by stać się jednymi z najpotężniejszych narzędzi, jakimi dysponują programiści. Jeśli dobrze je poznasz i zastanowisz się nad tym, jak z nich korzystać, zaczniesz tworzyć strony WWW dopasowujące się bez problemów do potrzeb użytkownika, niezależnie od tego, z jakiego urządzenia będzie on otwierać Twoją stronę.

Zapytania o media — wdrożenie w przeglądarkach

	Chrome	Firefox	Safari	IE
Zapytania o media	Tak	Tak	Tak	Tak

Skorowidz

G

H

I

J

K

L

Ł

M

N

O

P

R

S

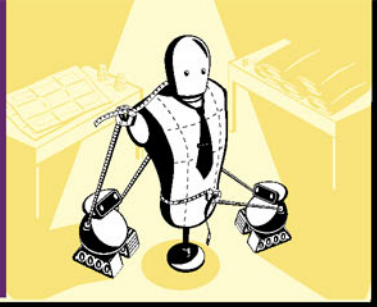
Ś

T

U

W

Z



POZNAJ POTENCJAŁ NOWEJ SPECYFIKACJI CSS3!

CSS to język, dzięki któremu możesz opisać wygląd tworzonej strony WWW. Jego pierwsza wersja ujrzała światło dzienne w 1996 roku i od tej pory język ten jest nieustannie rozwijany. Dziś natomiast obowiązuje wersja CSS3 — podzielona na moduły, do których zostały przypisane dotychczasowe atrybuty oraz dodane nowe możliwości. Próba przebrnięcia przez oficjalną dokumentację będzie Cię kosztowała wiele wysiłku, trudu oraz nieprzespanych nocy. Zapewne chcesz tego uniknąć? Ta książka to rozwiązanie idealne dla Ciebie!

Znajdziesz w niej najważniejsze informacje na temat CSS3, starannie opracowane i podane w przystępnej formie. Kolejne rozdziały pozwolą Ci zdobyć wiadomości o tak zwanych media queries, pozwalających zmieniać wygląd elementów strony w zależności od urządzenia, oraz szczegółową wiedzę dotyczącą selektorów. Ponadto przekonasz się, jak pseudoklasy i pseudoznaczniki mogą ułatwić Ci życie oraz jak pozbyć się nudnych czcionek dzięki możliwościom @font-face. Następnym wyzwaniem stojącym przed Tobą jest zapoznanie się z transformacjami 2D i 3D, efektami specjalnymi oraz elastycznymi układami elementów (Flexbox). Ta książka jest doskonałą lekturą dla projektantów stron, którzy chcą błyskawicznie poznać i wykorzystać nowości CSS3 w swoich projektach.

Dzięki tej książce:

- poznasz potęgę *media queries*
- zastosujesz niestandardowe czcionki
- przygotujesz transformacje trójwymiarowe
- wykorzystasz potencjał nowej specyfikacji CSS3
- bezboleśnie zwiększysz atrakcyjność tworzonej strony

Peter Gasston — projektant stron WWW z czternastoletnim doświadczeniem. Twórca licznych stron zarówno dla małych i średnich firm, jak i dla korporacji. Współtwórca portalu CSS3.info. Jego artykuły były publikowane w takich serwisach jak Smashingmagazine.com oraz alistapart.com. Autor książek poświęconych tworzeniu stron WWW.

Helion

37334 numer katalogowy

księgarnia internetowa

http://helion.pl

zamówienia telefoniczne

0 801 339900

0 601 339900

Informatyka w najlepszym wydaniu

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

sięgnij po WIĘCEJ

KOD KORZYŚCI

ISBN 978-83-283-0976-0

cena: 59,00 zł

no starch press