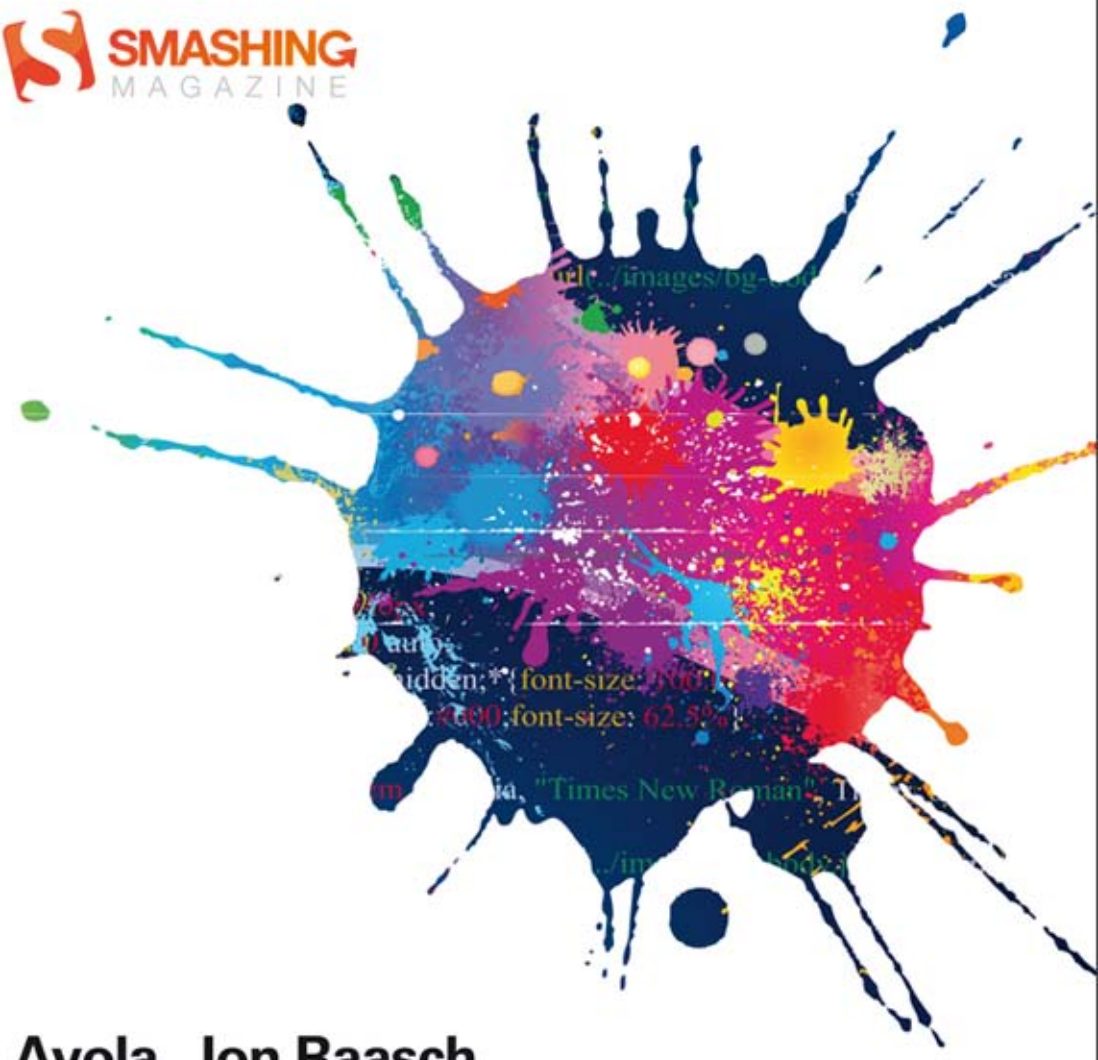


Greg Avola, Jon Raasch

Mobile Web Development

SMASHING MAGAZINE

ODNIEŚ SUKCES NA RYNKU APLIKACJI MOBILNYCH!

Tytuł oryginału: Smashing Mobile Web Development

Tłumaczenie: Marek Pętlicki

ISBN: 978-83-246-7098-7

This edition first published 2013.

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)

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

Printed in Poland.

Spis treści

2 PRZEGLĄD TECHNOLOGII MOBILNYCH

W ŚWIECIE TECHNOLOGII co chwila pojawia się coś nowego. Wykorzystanie smartfonów i przeglądarek mobilnych staje się coraz bardziej popularne, a każdego dnia powstają nowe technologie udoskonalające ich obsługę. Sposób rozumowania twórców aplikacji mobilnych uległ znacznej zmianie — od projektowania stron dla użytkowników wyposażonych w mysz i klawiaturę do obsługi za pomocą palców.

W tym rozdziale dowiesz się nieco na temat technologii wykorzystywanych w tego typu aplikacjach. Wspominam o funkcjach HTML5, które umożliwiają użytkownikom wyszukiwanie lokalizacji, wykrywanie dostępu do sieci, zarządzanie lokalnymi bazami danych i uzyskanie dostępu do sesji oraz lokalnej pamięci podręcznej (ang. *cache*). Te funkcje umożliwiają implementację bardziej zaawansowanych funkcji oraz usprawnienie obsługi aplikacji.

Omawiam również CSS3 i nowe właściwości, dzięki którym aplikacji można nadać wygląd przypominający aplikacje natywne, takie jak: cienie, zaokrąglone narożniki, gradienty i obsługa rozmiarów obrazów tła w celu zachowania wysokiej jakości na wyświetlaczach o różnych rozdzielczościach. Panuje pewne przekonanie dotyczące aplikacji mobilnych, że jeśli chcesz osiągnąć profesjonalny wygląd aplikacji, musisz zatrudnić profesjonalnego grafika. W rzeczywistości często wystarczy umiejętne wykorzystanie nowych możliwości CSS3.

Na końcu rozdziału przejdziemy do omawiania JavaScriptu i jego frameworków, które są szczególnie przydatne w przeglądarkach mobilnych, takich jak jQuery, Sencha Touch, XUI itp. W rozdziale wspomnę też o bibliotekach narzędziowych, jak Modernizr, iScroll czy biblioteki obsługi szablonów dla języka JavaScript, które znacząco ułatwiają tworzenie aplikacji.

HTML

Zanim powstał HTML5, wykorzystywana była specyfikacja HTML-a zatwierdzona 24 grudnia 1999 roku. Była to wersja 4.01, ale większość osób określała ją po prostu terminem HTML. W momencie wprowadzenia standardu HTML5 świat przeglądarek WWW całkowicie się zmienił. Specyfikacja ta wprowadza nowe funkcje, które wcześniej nie były dostępne w przeglądarkach. Dawniej na przykład w celu obsługi takich mediów jak wideo czy dźwięk konieczne było stosowanie zewnętrznych technologii, takich jak Flash. W HTML5 do obsługi wideo na stronie wystarczy odpowiedni znacznik; Flash nie jest już potrzebny (z tej techniki można jednak korzystać w urządzeniach mobilnych ze starszymi wersjami systemu iOS, które obsługują technologię Flash). Inną zaletą korzystania z tych nowych możliwości jest to, że do umieszczenia wideo na stronie nie musisz się uczyć nowych języków programowania — wystarczy użyć w dokumencie znacznika `<video>`.

HTML5 wprowadza nowe atrybuty i elementy semantyczne pomocne przy definiowaniu struktury dokumentów HTML. Wcześniej czytając kod dokumentu HTML, nie wiedziałeś, gdzie kończy się nagłówek, a gdzie zaczyna stopka. W HTML5 można dodać znaczniki `header`, `footer` i `content`, pozwalające w czytelny sposób zdefiniować elementy, które mogą być z łatwością interpretowane przez przeglądarki i odpowiednio prezentowane odbiorcy.

Oprócz znaczników semantycznych HTML5 przynosi inne nowości, jak obsługa baz danych po stronie klienta, geolokalizacja i kontrola nad pamięcią podręczną. Te techniki dają programistom dostęp do większej liczby funkcji urządzenia mobilnego niż w poprzednich wersjach HTML. A ponieważ HTML5 jest standardem, jego możliwości są dostępne w prawie każdej przeglądarce mobilnej.

KLUCZOWE CECHY HTML5

HTML5 posiada wiele cech, które mogą być użyte w aplikacjach, ale kilka z nich jest szczególnie istotnych w kontekście aplikacji mobilnych. Funkcje te pozwalają przesunąć granicę między przeglądarką a dedykowaną aplikacją i dają kontrolę nad takimi mechanizmami jak pamięć podręczna czy wykrywanie lokalizacji geograficznej urządzenia w celu zaprezentowania użytkownikowi lokalnych informacji. Jeśli jest to potrzebne, możesz użyć ich wszystkich!

Mechanizmy geolokalizacji

W ostatnich kilku latach dostęp do informacji o lokalizacji użytkownika stawał się coraz istotniejszym elementem prawie każdej aplikacji lub usługi WWW. Dostęp do lokalizacji umożliwia programistom między innymi dostarczanie użytkownikom lokalnych informacji, instrukcji dla kierowców itp. HTML5 ułatwia odczyt lokalizacji i — w zależności od potrzeb aplikacji — oferuje dodatkowe opcje.

Geolokalizacja w przeglądarce WWW różni się od geolokalizacji w natywnym urządzeniu. Zgodnie ze specyfikacją geolokalizacja w przeglądarce powinna wykorzystać najlepszą z dostępnych w danym momencie metod pozycjonowania. W teorii powinien to zawsze być odbiornik GPS telefonu, jednak w niektórych przypadkach przeglądarka wykrywa lokalizację, korzystając z innych technik, na przykład położenia anten sieci komórkowej czy Wi-Fi. Tabela 2.1 przedstawia metody służące do obsługi funkcji geolokalizacji.

Tabela 2.1. Metody geolokalizacyjne w HTML5

Metoda	Opis
`Navigator.getPosition(success, error, options)`	Odczyt bieżącej lokalizacji użytkownika
`Navigator.watchPosition(success, error, options)`	Odczyt zmian lokalizacji użytkownika w czasie

Opcje metod lokalizacji

Metody `getPosition` i `watchPosition` obsługują dodatkowe opcje służące do aktywacji specjalnych funkcji:

- `enableHighAccuracy` (**Boolean: `true` lub `false`**) — wartość `true` aktywuje najdokładniejszy z dostępnych mechanizmów geolokalizacji. Wadą tej opcji jest większe zużycie energii urządzenia.
- `timeout` (**Integer, milisekundy**) — domyślnie metody geolokalizacyjne nie wykorzystują limitu czasu. Dobre praktyki zalecają jednak określenie czasu, w jakim urządzenie ma podać lokalizację. Po jego upłynięciu wywoływana jest metoda określona w parametrze `error`.
- `maximum` (**Integer, milisekundy**) — określenie maksymalnego okresu ważności lokalizacji. Jeśli czas od odczytu lokalizacji zapisanej w pamięci podręcznej jest dłuższy, urządzenie dokona nowego odczytu lokalizacji. W przeciwnym razie zostanie wykorzystana wartość z pamięci podręcznej.

Funkcje zwrotne metod lokalizacyjnych

Jeśli wywołanie metody geolokalizacyjnej zakończy się sukcesem, automatycznie zostanie wywołana funkcja podana w parametrze `success`, której zostanie przekazany obiekt definiujący pozycję (jego atrybuty są opisane w tabeli 2.2).

Tabela 2.2. Atrybuty argumentu funkcji zwrotnych getPosition i watchPosition

Atrybut	Typ	Opis
`Position.coords.latitude`	`Double`	Szerokość geograficzna w stopniach
`Position.coords.longitude`	`Double`	Długość geograficzna w stopniach
`Position.coords.altitude`	`Double`	Wzniesienie w metrach
`Position.coords.accuracy`	`Double`	Poziom precyzji odczytu długości i szerokości geograficznej; im większa liczba, tym dokładniejszy odczyt
`Position.coords.altitudeAccuracy`	`Double`	Poziom precyzji odczytu wzniesienia
`Position.coords.heading`	`Double`	Kierunek przemieszczania się urządzenia w stopniach; jeśli implementacja nie potrafi zwrócić tej informacji, ten atrybut przyjmuje wartość `null`
`Position.coords.speed`	`Double`	Prędkość przemieszczania się urządzenia; jeśli implementacja nie potrafi zwrócić tej informacji, ten atrybut przyjmuje wartość `null`

Jeśli metoda zakończy się porażką, zostanie wywołana funkcja zwrotna `error`, której zostanie przekazany obiekt o atrybutach `err.code` i `err.message`. Wartości tych atrybutów prezentuje tabela 2.3.

Tabela 2.3. Kody i komunikaty błędów zwracane przez metody getPosition i watchPosition

Atrybut err.code	Atrybut err.message	Opis
`error.code.1`	`PERMISSION_DENIED`	Błąd zwracany, gdy użytkownik odmówi udostępniania lokalizacji stronie WWW lub aplikacji
`error.code.2`	`POSITION_UNAVAILABLE`	Błąd zwracany, gdy lokalizacja nie może być określona. Oznacza to z reguły, że co najmniej jeden z mechanizmów dostarczających informacje lokalizacyjne wygenerował komunikaty o błędach
`error.code.3`	`TIMEOUT`	Błąd zwracany w przypadku upłynięcia czasu oczekiwania. Ten błąd może być zwrócony tylko wtedy, gdy w wywołaniu zostanie ustawiona wartość `true` w atrybucie `timeout` obiektu opcji

Wykorzystanie metody navigator.getPosition() do jednorazowego odczytu lokalizacji

Jeśli aplikacja wymaga informacji o lokalizacji użytkownika w danym punkcie czasu, można wykorzystać metodę `navigator.getPosition()`.

```
<html>
  <body onload="start();">
    <div id="content"></div>
  </body>
</html>
<script>
function start()
{
  if (navigator.geolocation) {
    var options = {
```

```
      enableHighAccuracy: true
    };
    navigator.geolocation.getCurrentPosition(onSuccess, onError, options);
  } else {
    var content = document.getElementById("content");
    content.innerHTML("W tej przeglądarce geolokalizacja nie jest dostępna");
  }
}
// funkcja wywoływana w przypadku sukcesu
function onSuccess(position){
  var content = document.getElementById("content");
  var message = "";
  var lat = position.coords.latitude;
  var lng = position.coords.longitude;
  content.innerHTML = "Twoje współrzędne: szerokość geograficzna <strong>" + lat +
  ➥"</strong> długość geograficzna: <strong>" + lng + "</strong>";
}
function onError(error){
  var content = document.getElementById("content");
  var message = "";
  switch (error.code) {
    case 0:
      message = "Wystąpił błąd: " + error.message;
      break;
    case 1:
      message = "Użytkownik odmówił zgody na odczyt lokalizacji";
      break;
    case 2:
      message = "Przeglądarka nie była w stanie odczytać lokalizacji: " + error.message;
      break;
    case 3:
      message = "Upłynął czas oczekiwania na odczyt lokalizacji";
      break;
  }
  content.innerHTML = message;
}
</script>
```

Powyższy kod wykorzystuje mechanizmy opisane w poprzednich punktach i prezentuje właściwy sposób wykorzystania metody `geolocation.getPosition`. W elemencie `div` z identyfikatorem (`id`) o wartości `content` wyświetlane są długość i szerokość geograficzna położenia urządzenia. Jeśli odczyt lokalizacji zakończy się błędem, kod analizuje jego przyczynę za pomocą instrukcji `switch` z użyciem wartości właściwości `err.code`, po czym wyświetla stosowny komunikat.

Wykorzystanie metody navigator.watchPosition() do ciągłego monitorowania lokalizacji

W przeciwieństwie do metody `getPosition()`, która podaje jednorazową pozycję użytkownika, metoda `watchPosition()` służy do ciągłego aktualizowania informacji o jego położeniu. Ta metoda może być wykorzystana, gdy aplikacja potrzebuje informacji o tym, czy użytkownik zmienił swoje fizyczne położenie, na przykład w celu rejestracji trasy. Metoda `watchPosition()` posiada te same opcje co metoda `getCurrent()`, ale sposób jej użycia jest nieco inny:

```
var watchID = navigator.geolocation.watchPosition(success, fail, options);
```

Przy użyciu metody `watchPosition()` warto zdefiniować opcję `timeout`; w przeciwnym razie istnieje ryzyko szybkiego zużycia energii z baterii urządzenia. Metoda zwraca liczbowy identyfikator (przypisany przez nas zmiennej `watchID`). Wartość tę można wykorzystać do zatrzymania śledzenia pozycji, wywołując następujące żądanie:

```
navigator.geolocation.clearWatch(watchId);
```

Jeśli wywołanie metody `watchPosition()` zakończy się powodzeniem, zostaną wykonane te same działania co w przypadku `getPosition()`. Warto zwrócić uwagę na fakt, że `watchPosition()` często zwraca dokładniejsze wyniki, ponieważ pierwszy odczyt lokalizacji jest z reguły przybliżony, a kolejne odczyty zwiększają jego precyzję. Z drugiej strony ta metoda jest oczywiście bardziej energochłonna.

Magazyn danych

Dzięki HTML5 programiści mają dostęp do lepszych narzędzi do przechowywania danych, które zachowują informacje, nawet jeśli przeglądarka zostanie zamknięta, a urządzenie uruchomione ponownie. Magazyn danych HTML5 (ang. *data storage*) ma dwie implementacje dostosowane do różnych potrzeb. Dane zapisane w tych magazynach są dostępne w drzewie obiektów DOM. Implementacji tych można użyć do zapisywania haseł, ustawień aplikacji, tokenów uwierzytelniających, w charakterze pamięci podręcznej HTML i do innych zastosowań.

Obiekty localStorage i sessionStorage

Obiekt `localStorage` przechowuje dane, które zostają zachowane do momentu usunięcia ich przez kod albo wyczyszczenia pamięci podręcznej przez użytkownika. Mechanizm `localStorage` powinien być wykorzystywany, gdy aplikacja ma zapamiętywać ustawienia również po zamknięciu przeglądarki czy wyłączeniu telefonu.

W przypadku `sessionStorage` dane nie są trwałe, to znaczy są usuwane po zamknięciu karty strony lub aplikacji WWW. Obiektu `sessionStorage` nie możesz użyć w przypadku aplikacji uruchomionej lokalnie. Jeśli jednak aplikacja operuje na danych o ograniczonym terminie ważności, jak liczniki czy koszyki zakupów, mechanizm daje ten komfort, że nie musimy się troszczyć o czyszczenie danych przy zamknięciu okna.

Zarówno `localStorage`, jak i `sessionStorage` zapisują wartości w postaci par klucz-wartość i udostępniają te same metody i składnię wywołań. Jedyna różnica polega na wykorzystaniu przestrzeni nazw `localStorage` lub `sessionStorage` w celu wykorzystania wybranego z nich.

Obydwa obiekty obsługują trzy metody: `setItem()`, `getItem()` i `removeItem()`, które służą — odpowiednio — do ustawiania wartości, odczytywania oraz usuwania danych. Na przykład w celu ustawienia wartości klucza `klucz` należy wywołać następujący kod:

```
localStorage.setItem("klucz", "wartość");
```

Jeśli chcesz odczytać wartość tego klucza, wykorzystaj metodę `getItem()`, jak zaprezentowano poniżej:

```
var yourKey = localStorage.getItem("klucz");
```

Do dobrych praktyk zalicza się sprawdzanie, czy klucz jest zainicjalizowany, zanim ustawisz lub usuniesz jego wartość. Oto przykładowy kod realizujący tę funkcję:

```
if (localStorage.getItem("book")) {
  // Doskonale, klucz "book" istnieje w naszym obiekcie localStorage
  var bookTitle = localStorage.getItem("book");
  document.write("Tytuł książki: " + bookTitle):
}
else {
  // Wygląda na to, że klucz "book" nie został odnaleziony, więc trzeba go dodać
  localStorage.setItem("book", "Smashing Magazine");
}
```

Powyższy kod sprawdza, czy klucz `"book"` istnieje w `localStorage`. Jeśli tak, odczytuje jego wartość, wykorzystując `getItem()`, i wypisuje komunikat w oknie przeglądarki. Jeśli klucz nie istnieje, za pomocą metody `setItem()` inicjalizowana jest nowa wartość `"Smashing Magazine"`. Jeśli chcesz usunąć klucz, wykorzystaj następujący kod:

```
if (localStorage.getItem("book")) {
  // Doskonale, klucz "book" istnieje w naszym obiekcie localStorage. Usuńmy go
  localStorage.removeItem("book");:
}
```

Pamięć podręczna aplikacji

Jedną z wad aplikacji WWW jest to, że przy każdym uruchomieniu użytkownik musi od nowa pobierać wszystkie pliki. Mechanizmy HTML5 umożliwiają kontrolę pamięci podręcznej, to znaczy twórca aplikacji może określić, które pliki zostaną zapisane w urządzeniu użytkownika, dzięki czemu nie muszą być pobierane przy kolejnej wizycie. Możliwość tego typu kontroli daje trzy podstawowe korzyści:

- **tryb offline** — aplikacja ładuje zasoby z pamięci podręcznej, dzięki czemu nie ma potrzeby łączenia się z serwerem, czyli aplikacja działa offline;
- **wydajność** — dzięki temu, że pliki nie muszą być pobierane z internetu, wzrasta wydajność uruchamiania oraz działania aplikacji;
- **mniejsze zużycie pasma serwera** — dzięki temu, że z serwera pobieranych jest mniej danych, obciążenie łącza jest mniejsze do czasu, gdy pliki ulegną zmianom.

W celu wykorzystania pamięci podręcznej w aplikacji musisz stworzyć plik nazywany manifestem pamięci podręcznej (ang. *cache manifest*). W tym pliku definiuje się instrukcje dla przeglądarek dotyczące obsługi pamięci podręcznej: które pliki mają być przechowywane po stronie klienta, a które muszą być zawsze wczytywane z sieci. Aby załadować ten plik, należy zadeklarować go jako atrybut elementu `html`, np.:

```
<html manifest="mycachefile.cache">
...
</html>
```

Plik manifestu może być wskazany ścieżką lokalną lub bezwzględną, a aby uzyskać pełną kontrolę nad pamięcią podręczną, należy umieścić ten atrybut we wszystkich plikach HTML. Nazwa pliku i rozszerzenie mogą być dowolne; powyższy przykład nie jest w żaden sposób zobowiązujący. Jedyny wymóg jest taki, aby zadeklarować go w atrybucie `manifest`.

Dodatkowo istotne jest zadeklarowanie w serwerze WWW odpowiedniego `mime-type` dla tego pliku o wartości `text/cache-manifest`. Dzięki temu serwer przekaże przeglądarce właściwą informację o typie pliku. Dla poprzedniego przykładu w pliku *.htaccess* zdefiniujemy następujący wpis:

```
Addtype text/cache-manifest.cache
```

Manifest pamięci podręcznej może zawierać trzy sekcje; każda z nich jest opcjonalna. W najbardziej podstawowej wersji plik manifestu będzie miał następującą postać:

```
CACHE MANIFEST
index.html
style.css
img/myLogo.png
js/main.js
```


Wszystkie pliki manifestu pamięci podręcznej muszą się rozpoczynać słowami kluczowymi `CACHE MANIFEST`, po których następuje lista plików buforowanych po stronie przeglądarki. W powyższym przykładzie wymienione pliki będą odczytane z serwera tylko raz, a przy każdej kolejnej wizycie będą odczytywane z pamięci podręcznej. Należy jedynie mieć na uwadze, że rozmiar pamięci podręcznej jest ograniczony do 5 MB.

Trzy opcjonalne sekcje pliku manifestu pamięci podręcznej mają następujące znaczenie:

- `CACHE` — domyślna sekcja, która ma zastosowanie, jeśli żadna sekcja nie została zadeklarowana w sposób jawny. Wszystkie pliki w tej sekcji będą wczytywane z pamięci podręcznej niezależnie od tego, czy urządzenie ma dostęp do sieci.
- `NETWORK` — ta sekcja definiuje elementy, które nie będą ładowane z pamięci podręcznej i wymagają dostępu do sieci. W tej sekcji najczęściej deklaruje się adresy URL skryptów logiki aplikacji uruchamianej na serwerze.
- `FALLBACK` — w tej sekcji definiuje się pliki używane przez przeglądarkę, w przypadku gdy podstawowy zasób nie jest dostępny. Jest to sekcja przydatna do definiowania rozwiązań zastępczych na wypadek braku dostępu do sieci, na przykład w celu poinformowania użytkownika o braku dostępu do sieci może być wykorzystany przygotowany wcześniej plik.

Poniższy kod prezentuje sposób wykorzystania wszystkich sekcji manifestu pamięci podręcznej:

```
CACHE MANIFEST
CACHE:
index.html
style.css
img/myLogo.png
```

```
js/main.js

NETWORK:
script.php
login.php
/script/*

FALLBACK:
main.php simple.html
*.html offline.html
```

W przykładowym pliku jako pierwsza zadeklarowana jest sekcja CACHE. Sekcja NETWORK definiuje kilka adresów wymagających dostępu do sieci. Te pliki nie będą buforowane. Ostatnia sekcja, FALLBACK, definiuje elementy zastępcze, które będą wykorzystane, w przypadku gdy podstawowy plik nie jest dostępny. Na przykład pierwsza definicja sekcji FALLBACK wskazuje plik *main.php*, który w przypadku niedostępności lub braku połączenia z siecią zostanie zastąpiony plikiem *simple.html*. W sekcjach można też wykorzystać wzorce dopasowań, na przykład *, które są pomocne w celu dopasowania całej zawartości katalogu.

Aktualizacja pamięci podręcznej

Po wprowadzeniu zmian w plikach zbuforowanych w pamięci podręcznej należy tę pamięć odświeżyć, gdyż w przeciwnym wypadku użytkownikowi będą stale serwowane stare wersje plików. Po zapisaniu w pamięci podręcznej dane są przechowywane do momentu, gdy wystąpi jedna z poniższych okoliczności:

- użytkownik wyczyści pamięć podręczną;
- plik manifestu ulegnie modyfikacji lub zostanie usunięty;
- pamięć podręczna zostanie wyczyszczona za pomocą skryptu.

W celu sprawdzenia, czy któryś z plików zapisanych w pamięci podręcznej został zmodyfikowany, możesz użyć następującego skryptu, który należy wywołać po załadowaniu struktury DOM.

```
window.addEventListener('load', function(e) {
      window.applicationCache.addEventListener("updateready", function(e){
           if (window.applicationCache.status ==
window.applicationCache.UPDATEREADY) {
                        window.applicationCache.swapCache();
                        if (confirm("Została wykryta nowa wersja aplikacji.
                        ↪Czy chcesz ją załadować?")
                        {
                              window.location.reload();
                        }
             }
             else
             {
                 // nic się nie zmieniło!
             }
      });
});
```

Po załadowaniu aplikacji atrybut `UPDATEREADY` zawiera informację o tym, że dostępna jest aktualizacja pliku zapisanego w pamięci podręcznej. W takim przypadku skrypt pobiera nową wersję pliku i podmienia ją w pamięci podręcznej. Następnie wyświetla prośbę o zgodę na przeładowanie interfejsu.

Należy pamiętać, że plik manifestu jest traktowany jako całość, to znaczy że jeśli zmienisz w nim cokolwiek (choćby dopiszesz komentarz), przeglądarka załaduje wszystkie pliki od nowa.

CSS3

Akronim CSS pochodzi od nazwy *Cascading Style Sheets*, czyli „kaskadowe arkusze stylów". Służą one do definiowania wyglądu elementów HTML na stronie. CSS3 jest nową wersją tego standardu, która pozwala projektantom i twórcom stron WWW definiować zaawansowane style wyglądu elementów bez konieczności stosowania skomplikowanych skryptów JavaScript, plików graficznych czy Flasha.

Należy pamiętać, że CSS3 nie jest obsługiwany we wszystkich przeglądarkach w jednakowym stopniu. Informacje o zgodności poszczególnych typów przeglądarek znajdziesz w tabeli 2.4.

Tabela 2.4. Obsługa CSS3 przez różne silniki przeglądarek

Silnik	Przykładowe przeglądarki	Obsługa CSS3
Gecko	Mozilla Firefox, Mozilla Firefox Mobile (Fennac)	Obsługa CSS3 jest dostępna od wersji 4.0. Właściwości CSS muszą być poprzedzone przedrostkiem `-moz`
WebKit	Apple Safari, Apple Safari Mobile, Google Chrome	CSS3 jest obsługiwany w różnym stopniu w zależności od wersji. Najnowsze wersje przeglądarek powinny w pełni obsługiwać wszystkie funkcje CSS3. Właściwości CSS muszą być poprzedzone przedrostkiem `–webkit`
Mosaic	Internet Explorer, IE Mobile	Internet Explorer obsługuje CSS3 od wersji 10. Wcześniejsze wersje tej przeglądarki obsługują ten standard jedynie częściowo

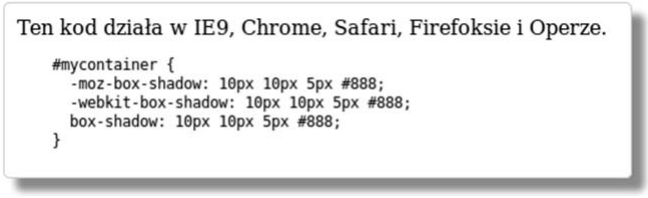
Tworząc aplikacje mobilne, musisz dokładnie zapoznać się z platformą docelową. Jeśli chcesz pisać aplikacje na platformę Windows Mobile (która wykorzystuje IE7 Mobile), musisz wziąć pod uwagę, jakie funkcje CSS3 są przez nią obsługiwane. Wiele narzędzi potrafi wykrywać typy systemów operacyjnych i mechanizmy obsługiwane przez silnik przeglądarki. Te narzędzia, które zostaną omówione w dalszej części rozdziału, są bardzo pomocne w procesie tworzenia aplikacji.

KLUCZOWE CECHY CSS3

CSS3 oferuje spory wybór nowych możliwości, które będą pomocne podczas projektowania działających szybciej i wyglądających atrakcyjniej stron WWW. Celem stworzenia CSS3 było udostępnienie twórcom stron możliwości definiowania wyglądu za pomocą samych arkuszy stylów, bez uciekania się do stosowania JavaScriptu czy plików graficznych. Rzućmy okiem na kilka kluczowych funkcji CSS3, które od razu możesz wykorzystać w swoich aplikacjach mobilnych.

Cień

Atrybut `box-shadow` stosuje się w odniesieniu do elementów formatowanych jako prostokątne bloki na ekranie. W rezultacie uzyskamy efekt cienia wokół prostokąta, dzięki czemu elementy wykorzystujące ten atrybut wyglądają, jakby unosiły się nad powierzchnią (rysunek 2.1 prezentuje przykład tego efektu).



Rysunek 2.1. Efekt cienia uzyskany dzięki atrybutowi box-shadow CSS3

W celu wykorzystania efektu cienia w elemencie należy wpisać następującą deklarację CSS:

```
box-shadow: none | <shadow> [ , <shadow> ]*
```

Oto przykład rzeczywistego kodu:

```
box-shadow: 10px 10px
box-shadow: 10px 10px 5px #C1C1C1
box-shadow: inset 5px 5px 5px 5px #C1C1C1
box-shadow: 0px 0px 5px 5px #C1C1C1, 0px 0px 5px 5px #C1C1C1, 0px 0px 5px 5px #C1C1C1;
```

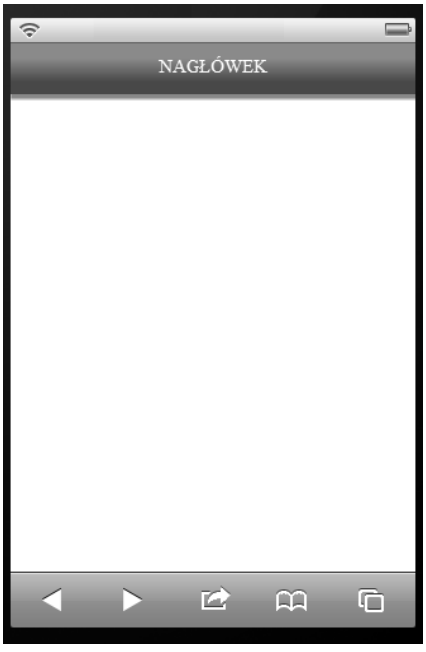

Niektóre z tych przykładów na pierwszy rzut oka wyglądają na dość skomplikowane, ale wystarczy rozbić deklarację na elementy, a wszystko stanie się oczywiste. W pierwszym etapie definiujemy kształt cienia obiektu. W tym celu należy podać od dwóch do czterech wymiarów:

1. Poziomy odstęp cienia od obiektu. Ta liczba może być dodatnia lub ujemna.
2. Pionowy odstęp cienia od obiektu. Jeśli ta wartość jest ujemna, cień będzie przesunięty powyżej obiektu.
3. Opcjonalna wartość promienia rozmycia cienia. W tym przypadku nie należy stosować wartości ujemnych, a im większa wartość, w tym większym stopniu będzie rozmyta krawędź cienia. Nie ma wartości domyślnej, zatem jeśli promień rozmycia nie zostanie określony, cień nie będzie rozmyty.
4. Opcjonalna wartość odległości rozproszenia cienia. Ten parametr przyjmuje wartości dodatnie i ujemne; wartości dodatnie spowodują rozszerzanie się cienia, ujemne mają wpływ na jego kontrast.

Po określeniu wspomnianych parametrów wymiarów cienia można podać opcjonalną deklarację jego koloru. Domyślną wartością jest #000, czyli kolor czarny.

W zależności od potrzeb możesz poeksperymentować z parametrami wymiarów. Jeśli na przykład chcesz stworzyć pasek nagłówka dla aplikacji mobilnej, który rzuca cień w dół, możesz wykorzystać poniższy kod. Efekt końcowy prezentuje rysunek 2.2.

```
#myheader {
        -moz-box-shadow: 2px 2px 2px 2px #888;
        -webkit-box-shadow: 2px 2px 2px 2px #888;
        box-shadow: 2px 2px 2px 2px #888;
}
```



Rysunek 2.2. Wynik działania właściwości CSS box-shadow

Ten kod generuje cień o promieniu 2px z rozmyciem i rozproszeniem 2px. Dzięki niemu wydaje się, że nagłówek unosi się nad treścią.

Doskonałym narzędziem do tworzenia reguł CSS dla różnych przeglądarek jest *css3please.com*. Na tej stronie możesz w łatwy sposób zdefiniować reguły CSS3 działające w różnych przeglądarkach — rozszerzenia, takie jak `webkit`, `moz` itp., zostaną nadane automatycznie.

Zaokrąglone narożniki

W czasach poprzedzających powstanie CSS3 większość zaokrąglonych narożników na stronach WWW tworzyło się za pomocą kombinacji JavaScriptu i plików graficznych ustawianych w tle. Dzięki CSS3 wykorzystanie zaokrąglonych narożników stało się łatwiejsze niż kiedykolwiek wcześniej. Do tworzenia prostokąta z zaokrąglonymi narożnikami stosuje się następującą deklarację:

```
border-*-*-radius: [ <length> | <%> ] [ <length> | <%> ]?
```

Oto kilka przykładów:

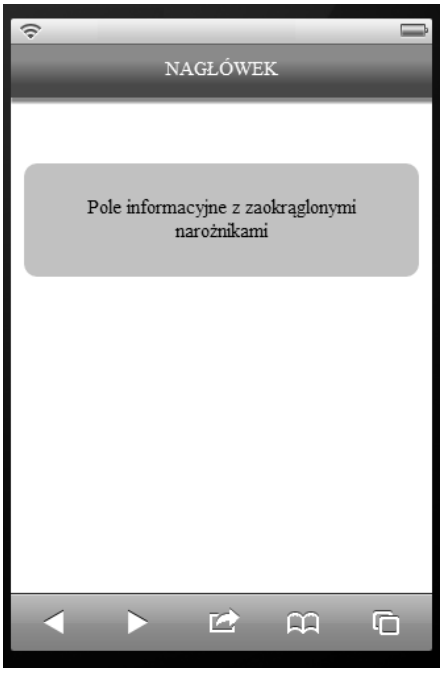
```
border-radius: 10px;
border-top-left-radius: 10px;
border-bottom-left-radius: 75%;
```

Warto zwrócić uwagę, że `border-radius` potrafi wykorzystywać wymiary w procentach i pikselach. Można też podać deklarację narożnika z użyciem `top`, `left`, `bottom` i `right`.

Każda z właściwości przyjmuje do czterech wartości, podobnie jak `box-shadow`. W przypadku wybranego narożnika, np. `border-top-left-radius`, wartości definiują kolejno: rozmiar poziomy, pionowy, ćwiartkę elipsy oraz zakrzywienie narożnika. Jeśli definiujemy wszystkie narożniki z użyciem selektora `border-radius`, wartości te określą promienie każdego z narożników. Jeśli zostanie podana jedna wartość, będzie ona zastosowana do wszystkich narożników.

Jeśli chcesz zdefiniować zaokrąglone narożniki dla szarego prostokąta, jak na rysunku 2.3, wystarczy przypisać wybranemu elementowi następujący arkusz stylów:

```
#rounded-box {
        width: 250px;
        margin: 50px auto;
        text-align: center;
        padding: 25px;
        background-color: #c1c1c1;
        border-radius: 10px;
}
```

Rysunek 2.3. Zaokrąglone narożniki nadają aplikacji nowoczesny wygląd

Zaokrąglone narożniki można również zdefiniować, wykorzystując skróconą wersję zapisu z wszystkimi atrybutami w jednym wierszu. Kolejność narożników jest następująca: prawy górny, prawy dolny, lewy dolny i prawy górny. Oto przykład:

```
border-radius: 10px 10px 10px 10px;
```

Gradienty

Zanim zagłębisz się w składnię gradientów, zastanów się chwilę, jak wygląda gradient w aplikacji mobilnej. Jak widzisz na rysunku 2.3, gradient jest płynnym przejściem tonalnym między kilkoma barwami, nadającym obiektowi efekt połysku. Przed pojawieniem się CSS3 gradienty tworzyło się z użyciem plików graficznych tła, których ładowanie mogło zajmować dłuższą chwilę. W CSS3 nie ma pliku obrazu do załadowania, dzięki czemu gradienty stały się dobrym sposobem uatrakcyjnienia wizualnego stron i aplikacji, gdyż nadają im nowoczesny wygląd.

Składnia deklaracji gradientu może się wydawać nieco nieczytelna. W naszych przykładach skupimy się na liniowej wersji gradientu, dostępnej w większości mobilnych przeglądarek. Składnia ta różni się nieco między Firefoksem a Safari i Chrome.

- **Firefox**: `-moz-linear-gradient(<point> || <angle>, color-stops)`,
- **Safari/Chrome**: `-webkit-linear-gradient(<point>, color-stops)`.

Poniższy przykład ustawia dokładnie ten sam gradient w przeglądarkach Firefox, Safari i Chrome:

- `background: -moz-linear-gradient(top, #00abeb, #fff);`,
- `background: -webkit-linear-gradient(top, #00abeb, #fff);`.

Gradient liniowy wymaga określenia kolorów, między którymi będą generowane płynne przejścia. Ręczna praca nad paletami kolorów może być nieco uciążliwa, szczególnie w przypadku gradientów. Poniżej zostało wymienionych kilka łatwych w obsłudze generatorów gradientów online:

- **CSS3 Gradient Generator**: *gradients.glrzad.com*,
- **Colorzilla Gradient Editor**: *http://www.colorzilla.com/gradient-editor/*,
- **Microsoft CSS Gradient Background Maker**: *ie.microsoft.com/testdrive/graphics/cssgradientbackgroundmaker/*.

Warto pamiętać, że podczas definiowania gradientów musisz mieć na uwadze przeglądarki, które nie obsługują ich na urządzeniach mobilnych. W tym celu warto zdefiniować jednolity kolor tła, który będzie wyświetlany zamiast gradientu:

- `background: -moz-linear-gradient(top, #00abeb, #fff);`,
- `background: -webkit-linear-gradient(top, #00abeb, #fff);`,
- `background-color: #c1c1c1;`.

Jeśli zdefiniujesz style w taki sposób, a aplikacja zostanie załadowana w przeglądarce nieobsługującej gradientów, tło elementu zostanie wyświetlone jako jasnoszare. Taki mechanizm może być przydatny na przykład do zwizualizowania przycisków w starszych przeglądarkach.

Selektory CSS3

Arkusze stylów CSS wykorzystują notację selektorów służących do wskazania elementów dokumentu, których dotyczy dany styl. W CSS3 zostały dodane nowe selektory umożliwiające uzyskanie znacznie wyższego poziomu dokładności tych definicji. Większość nowych selektorów pozwala na wskazanie określonych elementów w strukturze. Jeśli na przykład chcesz wyróżnić nieparzyste wiersze tabeli, możesz zastosować następujący kod:

```
tr:nth-child(odd) td {
  background-color: #c1c1c1;
}
```

Ten kod wyszukuje nieparzyste wiersze i nadaje im kolor tła. Ta składnia jest znacznie łatwiejsza w zastosowaniu od dostępnej w poprzedniej wersji CSS.

W CSS3 do dyspozycji mamy więcej użytecznych selektorów ułatwiających pisanie czytelnego i spójnego kodu. Wszystkie znajdziesz w specyfikacji CSS3 na stronie W3C (*www.w3.org/TR/selectors/*).

Przekształcenia

Dawniej przekształcenia obiektów w przeglądarce były możliwe wyłącznie z użyciem kodu JavaScript. CSS3 przyniósł łatwe w użyciu mechanizmy przekształcania kształtu, generowania cienia oraz przesuwania obiektów. Jednym z istotniejszych elementów są klatki kluczowe (ang. *keyframes*). Służą one do definiowania animacji przez wskazanie pośrednich stanów właściwości obiektu. W aplikacji mobilnej efekt ten można wykorzystać do realizacji przejścia między stronami. Klatki kluczowe przed użyciem muszą być zadeklarowane:

```
@-webkit-keyframes mymove /* Safari i Chrome*/
{
  from {top:0px;}
  to {top:200px;}
}
@-moz-keyframes mymove /* Mozilla*/ {
  from {top:0px;}
  to {top:200px;}
}
```

W powyższym kodzie definiujemy kierunek przesunięcia obiektu: od góry ekranu (`top:0px`) w dół (`top:200px`). Kod ten definiuje jedynie parametry przesunięcia, należy je jeszcze zastosować na obiekcie.

```
#myBox {
  -moz-animation:mymove 5s infinite; /* Firefox */
  -webkit-animation:mymove 5s infinite; /* Safari i Chrome */
}
```

Powyższy kod wywołuje przekształcenie, przypisując elementowi właściwość `mymove`. Kolejne dwa atrybuty definiują czas trwania przekształcenia (5 sekund) oraz liczbę wywołań animacji (w tym przypadku nieograniczoną dzięki słowu kluczowemu `infinite`).

Należy wziąć pod uwagę skutki, jakie efekty przekształceń CSS3 mogą wywołać w przeglądarkach mobilnych. Na desktopie przeglądarka ma dostęp do wydajnego CPU i dużej ilości RAM, jednak wykorzystanie elementów 3D na stronie może znacząco spowolnić działanie przeglądarki mobilnej. Wykorzystuj tylko te metody, które są niezbędne aplikacji do działania, ponieważ nadużycie tych właściwości może obniżyć wydajność aplikacji albo nawet spowodować jej wadliwe działanie.

JAVASCRIPT

Język JavaScript był dostępny na długo przed pojawieniem się technologii HTML5, jednak zakres możliwości ingerencji w elementy dokumentu z poziomu JavaScriptu znacznie się zwiększył dzięki nowym właściwościom CSS3. Możemy korzystać z wielu bibliotek ułatwiających i przyspieszających tworzenie aplikacji. Większość bibliotek JavaScriptu nie jest przeznaczona do zastosowań mobilnych — duża część ich kodu jest poświęcona zapewnianiu zgodności między różnymi wersjami przeglądarek, jak IE6. Ten kod jest zupełnie zbędny na platformach mobilnych. Im większy rozmiar pliku, tym wolniej będzie się ładować aplikacja. Najlepiej jest znaleźć taką bibliotekę, która będzie realizować potrzeby bez zbędnych dodatków.

JQUERY

Biblioteka jQuery jest jednym z najpopularniejszych frameworków JavaScriptu. Zawiera ona proste w użyciu metody, które ułatwiają implementację skomplikowanych funkcji CSS3, takich jak `CSS3-Rotate`, `CSS3-Animations` itp. Wadą tej biblioteki jest to, że zajmuje 31 KB. Choć może się wydawać, że to niewiele, sporo z tego kodu odwołuje się do specyfiki przeglądarki IE6, która nie ma zastosowania w urządzeniach mobilnych. To powoduje, że aplikacja za pierwszym razem ładuje się dość wolno.

XUI

XUI jest nowym frameworkiem JavaScriptu stworzonym przez Briana Leroux (członka zespołu programistów projektu PhoneGap). Celem projektu jest stworzenie prostego w użyciu frameworku o składni zbliżonej do jQuery, ale o mniejszych rozmiarach i funkcjach dostosowanych do urządzeń mobilnych. Podczas gdy jQuery zajmuje 31 KB, XUI ma zaledwie 4,2 KB, a ich możliwości są zbliżone. Jeśli szukasz biblioteki, która wykorzystuje tę samą składnię, ale charakteryzuje się mniejszymi rozmiarami, XUI może się sprawdzić.

ZEPTO

Zepto to kolejna biblioteka oferująca składnię zbliżoną do jQuery, ale jest przeznaczona wyłącznie dla przeglądarek opartych na silniku WebKit. Plik biblioteki ma zaledwie między 5 a 10 KB rozmiaru (w zależności od wybranych opcji), a oferuje większość metod jQuery istotnych na platformach mobilnych. Wadą tego rozwiązania jest to, że działa wyłącznie na WebKicie. Aplikacja nie będzie zatem działać na Windows Phone 7, Mozilla Firefox Mobile czy w innych przeglądarkach wykorzystujących niezgodny silnik.

JQTOUCH

jQTouch jest wtyczką do Zepto lub jQuery, uzupełniającą możliwości frameworku o elementy interfejsu użytkownika oraz dodatkowe mechanizmy JavaScriptu. Podobnie jak Zepto, biblioteka ta jest przeznaczona na przeglądarki oparte na WebKicie. Ta biblioteka do działania wymaga jQuery lub Zepto, w niektórych przypadkach może mieć zatem za duże rozmiary. Jej zaletą jest to, że oferuje mnóstwo doskonałych animacji z wykorzystaniem CSS3, dzięki którym aplikacja będzie wyglądać w sposób zbliżony do natywnej, z przejściami między stronami, trybem pełnoekranowym itp.

FRAMEWORKI I NARZĘDZIA DO TWORZENIA INTERFEJSU UŻYTKOWNIKA

Gdy tworzysz mobilne aplikacje WWW, masz do dyspozycji mnóstwo bibliotek, dzięki którym nie będziesz musiał wynajmować projektanta interfejsu. Te frameworki są zaprojektowane w taki sposób, aby aplikacja wyglądała w sposób natywny, i wykorzystują technologie zgodności między przeglądarkami oraz właściwości CSS3. W przypadku starszych przeglądarek nieobsługujących tych funkcji istnieją biblioteki pomocnicze, które pomogą Ci się zorientować, jakie mechanizmy JavaScriptu i CSS3 są dostępne dla użytkownika.

SENCHA TOUCH

Twórcy Sencha Touch określają ten framework jako najlepiej przystosowany do tworzenia aplikacji mobilnych w HTML5. Obsługuje on wiele różnych przeglądarek, systemy iOS, Android i BlackBerry od wersji 6 wzwyż, czyli również silniki inne niż WebKit. Większość twórców aplikacji mobilnych uznaje Sencha Touch za podstawowe narzędzie z kategorii frameworków javascriptowych. Znajdziesz tu zarówno mechanizmy interfejsu użytkownika, jak i inne narzędzia JavaScriptu, ale w przeciwieństwie do innych opisanych tu frameworków nie oferuje składni zgodnej z jQuery, przez co framework ten może być nieco trudny w zastosowaniu dla początkujących programistów.

JQUERY MOBILE

Udostępniony pod koniec 2011 roku framework jQuery Mobile jest wtyczką do jQuery, która umożliwia tworzenie zaawansowanych aplikacji WWW bez pisania kodu CSS. Ten framework przenosi możliwości jQuery na platformy mobilne, udostępniając dedykowane im metody i funkcje. jQuery Mobile udostępnia na przykład proste metody wykrywania orientacji ekranu, obsługuje gesty dotykowe oraz mechanizmy nawigacji z użyciem wtyczki jQuery History. Wadą tego frameworku jest to, że wymaga jQuery (31 KB), a sam dodatkowo „waży" niebagatelne 24 KB. Choć to niewątpliwie framework o potężnych możliwościach, weź pod uwagę większe zapotrzebowanie na ilość przesyłanych danych oraz wolniejsze czasy ładowania aplikacji.

MODERNIZR

Modernizr uwalnia programistę od problemów z tworzeniem kodu zgodnego z różnymi platformami. Wykrywanie zgodności przeglądarek w zakresie HTML5 i CSS3 może sprawiać kłopoty, ale dzięki bibliotece Modernizr staje się prostym zadaniem. To narzędzie pozwala programiście pisać proste warunki `if...then` sprawdzające poszczególne mechanizmy. Ta biblioteka, napisana w JavaScripcie, po uruchomieniu sprawdza wszystkie możliwości przeglądarki w zakresie HTML5 i CSS3 i dodaje do elementu `body` odpowiednie klasy, co prezentuje rysunek 2.4. Dzięki temu sprawdzenie określonej funkcji CSS3 czy HTML5 sprowadza się jedynie do wywołania prostego kodu.


```
if (Modernizr.localstorage) {
  // przeglądarka obsługuje mechanizm localStorage!
  return true;
} else {
  // localStorage nie jest obsługiwany, konieczne jest zastosowanie mechanizmów zastępczych
  return false;
}
```

Rysunek 2.4. Wykorzystanie biblioteki Modernizr do wykrywania HTML5

ISCROLL

Przed wersją iOS 5 nie było możliwości zaimplementowania w mobilnej przeglądarce Safari dolnego i górnego paska statusu, ponieważ właściwość `overflow: scroll` była nieaktywna. Od iOS 5 ta funkcja jest już dostępna, ale w jej implementacji nadal pozostaje kilka błędów, które firma Apple powinna naprawić. Aby upewnić się, że użytkownicy iOS 4 będą w stanie wykorzystać Twoją aplikację, możesz użyć wygodnej biblioteki JavaScriptu pod nazwą iScroll. Gdy zostanie włączona, u góry i na dole okna zostanie wyświetlony pasek narzędziowy. Biblioteka iScroll nie jest napisana z użyciem jQuery, nie ma również wymagań instalacji

jakiegokolwiek innego frameworku. Rozwiązanie to nie jest idealne, ale doskonale sprawdza się w przypadku stron zawierających statyczne treści i oferuje prostą w użyciu składnię poprawiającą integrację aplikacji ze środowiskiem natywnym.

Jeśli chcesz użyć biblioteki iScroll, ale Twój element treści ma zmienną wysokość (co oznacza, że treść będzie dodawana w sposób dynamiczny), możesz w prosty sposób wykorzystać możliwości iScroll do odświeżania przewijanego obszaru. Załóżmy, że konstrukcja strony jest następująca:

```
<div id="scrollarea">
  <div id="page1"></div>
  <div id="page2"></div>
  <div id="page3"></div>
</div>
```

Gdy w sekcji `page1` zostaną umieszczone wszystkie elementy, należy odświeżyć wysokość obiektu `iScroll`, co spowoduje, że stanie się przewijalny. Realizujemy to w następujący sposób:

```
function onCompletion () {
  // w tym miejscu modyfikujemy elementy sekcji page1
  setTimeout(function () {
    myScroll.refresh();
  },
  0);
};
```


Po wykonaniu tego kodu obszar treści będzie przewijany. Tę samą technikę można wykorzystać do pozostałych sekcji aplikacji.

MUSTACHE I INNE MECHANIZMY SZABLONÓW DLA JAVASCRIPTU

Mustache jest biblioteką szablonów dla JavaScriptu pozwalającą na definiowanie treści w postaci obiektów JSON zamiast tworzenia kompletnej struktury dokumentu w HTML-u. Gdy treść ma być pobierana na urządzenie mobilne z sieci, najlepiej jest maksymalnie zmniejszyć rozmiar przesyłanych danych. JSON (ang. *JavaScript Object Notation*) jest tekstowym formatem zapisu danych, który ułatwia przesyłanie struktur danych (np. tablic). Większość popularnych sieci społecznościowych, w tym Facebook, Twitter czy foursquare, oraz wiele API innych serwisów sieciowych wykorzystuje ten format do wymiany danych.

Z użyciem Mustache możesz budować proste szablony w HTML-u, a następnie kompilować je z danymi pobranymi w formacie JSON. Załóżmy, że mamy dostęp do danych w formacie JSON o następującej strukturze:

```
data = {
  first_name: "Greg",
  last_name: "Avola"
}
```

Szablon Mustache tworzymy w następujący sposób:

```
<div id="mustache_tmp">
  <p class="first">{{first_name}}</p>
  <p class="last">{{last_name}}</p>
</div>
```

W przypadku Mustache miejsca do wypełnienia definiuje się za pomocą podwójnych nawiasów klamrowych, w których umieszcza się identyfikator danych odpowiadający kluczowi w strukturze JSON:

```
<script>
  var content = $("#mustache_tmp").html();
  // zmienna data zawiera dane w strukturze z poprzedniego przykładu
  var newTemplate = Mustache.to_html(content, data);
  $("#mustache_tmp").html(newTemplate);
</script>
```

W tym przykładzie wykorzystaliśmy składnię jQuery do wczytania zawartości szablonu `mustache_tmp`, a następnie za pomocą Mustache wypełniliśmy ten szablon strukturą danych w formacie JSON. Po załadowaniu danych do szablonu, ponownie z użyciem jQuery, wstawiamy wynik do dokumentu.

W tym prostym przykładzie wykorzystanie szablonów może się wydać dość mało praktyczne, ale dzięki tej technice można zaoszczędzić mnóstwo pracy z dynamicznym tworzeniem kodu HTML za pomocą JavaScriptu, szczególnie gdy mamy dostęp do danych w formacie JSON. Również generowanie struktury JSON po stronie serwera będzie szybsze niż generowanie gotowego dokumentu HTML. Mustache oferuje mnóstwo innych możliwości, między innymi obsługę warunków i pętli, dzięki którym można tworzyć listy i inne powtarzalne struktury w kilku wierszach kodu szablonu.

PODSUMOWANIE

W tym rozdziale omówiliśmy zagadnienia związane z kodem HTML5, JavaScriptem i CSS3, które mogą być pomocne podczas tworzenia aplikacji mobilnych. Mimo że głównym zadaniem rozdziału było dostarczenie sugestii dotyczących dodatkowych materiałów informacyjnych, udało nam się również omówić kilka zagadnień, takich jak:

- geolokalizacja;
- manifest pamięci podręcznej;
- selektory CSS3;
- zaokrąglone narożniki;
- mobilne frameworki dla JavaScriptu (np. jQuery czy Zepto).

FENOMENALNE ŹRÓDŁO INFORMACJI O APLIKACJACH MOBILNYCH!

Smashing Magazine to jeden z najpopularniejszych serwisów poświęconych zagadnieniom z dziedziny programowania, między innymi tworzenia profesjonalnych stron WWW. Jest to obowiązkowa pozycja wśród ulubionych serwisów każdego projektanta i dewelopera. Na łamach tego portalu najwybitniejsi eksperci dzielą się wiedzą z zakresu projektowania interfejsów użytkownika, aplikacji mobilnych i stron WWW.

Kolejna książka z tej serii w całości jest poświęcona aplikacjom mobilnym. W trakcie lektury poznasz podstawy HTML5, CSS3 i JavaScriptu oraz zaznajomisz się z możliwościami, jakie dają one deweloperom. Dowiesz się, jak skonfigurować platformę rozwojową i produkcyjną, jak stworzyć prototyp oraz jaką bazę danych warto wykorzystać. Ponadto nauczysz się w praktyce stosować mechanizmy geolokalizacji i integracji z serwisami społecznościowymi oraz kontrolować wydajność Twojej aplikacji. Na koniec przekonasz się, że przygotowanie aplikacji do premiery wcale nie jest takie trudne. Z tą książką odniesiesz sukces!

Sprawdź:

- jakie nowości kryją HTML5 i CSS3
- jak przygotować platformę rozwojową i produkcyjną
- jak przechowywać dane w aplikacji mobilnej
- dlaczego warto tworzyć aplikacje na urządzenia przenośne
- jak odnieść sukces

helion.pl
księgarnia internetowa

WILEY

ISBN 978-83-246-7098-7

Cena 49,00 zł

9 788324 670987

Nr katalogowy: 14859

Księgarnia internetowa:
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

Helion

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

Informatyka w najlepszym wydaniu