Twórz strony dostępne dla każdej przeglądarki i dowolnego połączenia internetowego!

- Najlepsze praktyki pisania kodu HTML, CSS i JavaScript
- Konkretny kod znaczników stylów
- Eliminacja złych nawyków programistycznych
- Tworzenie ogólnodostępnych witryn

tworzenie stron metodą stopniowego ulepszania

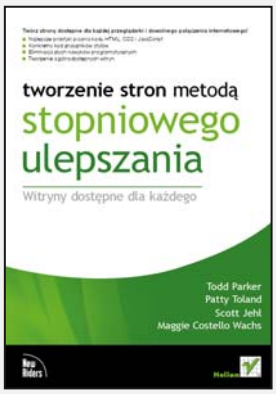
Witryny dostępne dla każdego

Todd Parker
Patty Toland
Scott Jehl
Maggie Costello Wachs

New Riders

Helion

Tworzenie stron metodą stopniowego ulepszania. Witryny dostępne dla każdego



Autorzy: Todd Parker, Scott Jehl, Maggie Costello Wachs, Patty Toland
Tłumaczenie: Łukasz Piwko
ISBN: 978-83-246-2797-4
Tytuł oryginału: Designing with Progressive Enhancement: Building the Web that Works for Everyone
Format: 168×237, stron: 416

Twórz strony dostępne dla każdej przeglądarki i dowolnego połączenia internetowego!

Obecnie tempo powstawania nowych stron i aplikacji jest zawrotne. Korzystanie z sieci jest coraz łatwiejsze, szybsze i bardziej dynamiczne. Co więcej, uwalniamy się od kabli dzięki rozwojowi telefonów komórkowych z dostępem do internetu i kompaktowych netbooków, zapewniających nam połączenie ze światem niezależnie od miejsca, w którym się znajdujemy. Jednak to bogactwo internetowych zasobów ma również swoje wady. Zaawansowane techniki znakomicie działają w najnowszych przeglądarkach, obsługujących technologie oparte na CSS i języku JavaScript. Istnieje jednak mnóstwo urządzeń, które w najlepszym wypadku obsługują je tylko w ograniczonym stopniu, przez co wysiłki programisty mogą spełznąć na niczym. Projektanci i programiści stron internetowych muszą zatem pogodzić trzy wzajemnie wykluczające się cele. Chcą wykorzystać wszystkie dostępne i ekscytujące technologie, aby móc tworzyć przykuwające uwagę interaktywne strony, nie tracąc jednocześnie na ich dostępności. A wszystko to powinni osiągnąć, tworząc czysty i łatwy do utrzymania kod.

Stopniowe ulepszanie to technika tworzenia stron internetowych zapewniająca wszystkim użytkownikom dostęp do podstawowych treści i opcji witryny – przy użyciu dowolnej przeglądarki i połączenia. Jednocześnie posiadaczom szybszych połączeń i bardziej zaawansowanych przeglądarek oferuje ono ulepszoną wersję strony.

„Tworzenie stron metodą stopniowego ulepszania. Witryny dostępne dla każdego” to praktyczny przewodnik po zasadach tworzenia stron za pomocą nowej metodologii. Dzięki niemu dowiesz się, dlaczego standardowe techniki tworzenia stron wykluczają niektórych użytkowników z grona odbiorców, i jak analizować projekty interfejsów, aby były funkcjonalne na każdym poziomie zaawansowania przeglądarki. Poznasz pakiet testów możliwości przeglądarek, a także najlepsze metody pisania kodu HTML, CSS i JavaScript w technice stopniowego ulepszania.

- Planowanie struktury i organizacji witryny
- Pisanie semantycznego kodu
- Stosowanie stylów
- Elementy interaktywne
- Testowanie możliwości przeglądarek
- Karty
- Okna dialogowe i nakładki
- Przyciski, pola wyboru, suwaki i menu
- Konstruktor list

Dostarczaj pełnię wrażeń wszystkim odbiorcom Twoich stron internetowych!

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

Spis treści

CZĘŚĆ II STOPNIOWE ULEPSZANIE — ZAGADNIENIA PRAKTYCZNE

Rozdział pierwszy

Nasze podejście

Najnowsze innowacje w zakresie technologii internetowych — głównie wprowadzane za pomocą zaawansowanych technik CSS, działających po stronie klienta JavaScriptu i Ajaksa, oraz wtyczek typu Flash — zamieniły nowoczesne witryny internetowe w atrakcyjne wizualnie aplikacje o dużych możliwościach interakcyjnych. Jest jednak jeden problem — różne przeglądarki i urządzenia na różne sposoby obsługują wymienione technologie. Mimo iż nowoczesne przeglądarki i najnowsze urządzenia przenośne potrafią wyświetlić nawet najbardziej skomplikowane interfejsy, stanowią one tylko część wszystkich odbiorników stron internetowych. A jak już wspominaliśmy wcześniej, tworząc witrynę lub aplikację dostępną tylko wąskiej grupie najnowszych urządzeń i przeglądarek, znacznie utrudniamy sobie dotarcie do jak najszerszego grona odbiorców.

Chcemy sprawić, aby treść, przesłanie i funkcjonalność naszych klientów były dostępne dla każdego — nie tylko dla tych, którzy mają nowoczesne przeglądarki obsługujące najnowocześniejsze technologie, lecz każdego, kto ma urządzenie z dostępem do sieci. Dlatego kilka lat temu postanowiliśmy zastosować filozofię stopniowego ulepszania w projektach naszych klientów.

Teoretycznie idea stopniowego ulepszania jest prosta — twórz strony zgodne ze standardami i przy użyciu czystego HTML-a, aby mieć pewność, że praktycznie każde urządzenie wyświetli ich treść w jakiś użyteczny sposób. Następnie nałóż na ten szkielet warstwy CSS i JavaScript, z których będą mogły skorzystać te przeglądarki, które obsługują te technologie.

Gdy zaczęliśmy stosować to podejście do tworzenia stron i przetestowaliśmy nasz produkt, dokonaliśmy ważnego odkrycia — zapomnieliśmy, że wiele starych przeglądarek i nowszych urządzeń przenośnych tylko *częściowo* obsługuje CSS i JavaScript oraz że użytkownicy mogą wyłączać ich obsługę, aby zwiększyć szybkość, podnieść poziom bezpieczeństwa lub polepszyć jakieś właściwości użytkowe. Obecnie skomplikowane interfejsy aplikacji i widgety są tworzone przy użyciu ściśle współpracującego kodu CSS i JavaScript — wyobraź sobie korzystanie z kalendarza lub paska przewijania w przeglądarce, która obsługuje JavaScript, ale niepoprawnie obsługuje CSS (rezultat byłby opłakany).

Podczas testowania naszych witryn utworzonych przy zastosowaniu technik stopniowego ulepszania znaleźliśmy kilka przeglądarek, które doskonałe strony zamieniały w bezużyteczny bałagan, ponieważ uruchamiały skrypty i stosowały style, które nie w pełni były przez nie obsługiwane. Jak więc dowiedzieć się, które przeglądarki poprawnie zastosują nasze ulepszenia?

Doszliśmy do wniosku, że aby mieć pewność, iż stopniowe ulepszanie pozwoli opracować produkt dostępny dla każdego, musimy zrobić trzy rzeczy:

- Krytycznie przejrzeć projekt i upewnić się, że każdy jego element — nawet najsprytniejszy komponent Web 2.0 lub Ajax — powstał w oparciu o semantyczny kod HTML o prawidłowej strukturze, który nawet w przeglądarkach nieobsługujących CSS i JavaScriptu zachowuje przynajmniej swoje podstawowe funkcje.
- Przetestować poziom obsługi CSS i JavaScriptu w przeglądarce *przed* zastosowaniem ulepszeń, aby zdecydować, czy w przeglądarce tej poprzestać na podstawowej funkcjonalności, czy dodać dla niej ulepszenia.
- W przypadku przeglądarek obsługujących rozszerzone funkcje należy zapewnić jak najwyższy poziom dostępności do strony za pomocą klawiatury oraz dodać elementy wspomagające pracę czytników ekranu.

W tym rozdziale opisany jest test sprawności przeglądarek, za pomocą którego określamy, czy danej przeglądarce można zaserwować rozszerzoną wersję witryny. Następnie przejdziemy do szczegółowego przedstawienia techniki stopniowego ulepszania, którą stosujemy w naszych projektach na co dzień. Tę część zaczniemy od opisu czynności nazywanej przez nas „prześwietlaniem" (ang. *x-ray perspective*). Polega ona na przeanalizowaniu skomplikowanego projektu interfejsu, wyodrębnieniu semantycznego kodu HTML, na bazie którego oparta będzie podstawowa funkcjonalność, oraz zaplanowaniu zaawansowanych elementów CSS i JavaScript przeznaczonych dla nowoczesnych przeglądarek przy jednoczesnym zachowaniu pełnej zgodności z czytnikami ekranu.

Sprawdzanie możliwości przeglądarek

Z naszych wstępnych badań stopniowego ulepszania wynika, że większość programistów rozszerza funkcjonalność swoich stron na jeden z dwóch sposobów — dostarczając rozszerzoną funkcjonalność wszystkim przeglądarkom, które mają włączoną obsługę JavaScriptu lub kierując rozszerzenia do wybranych przeglądarek za pomocą techniki wykrywania typu przeglądarki (ang. *browser sniffing*).

Wykrywanie typu przeglądarki wykluczyliśmy od razu z kilku powodów:

- Efektywne wykrywanie przeglądarek wymaga drobiazgowej wiedzy na temat ich zachowań (oraz wszystkich ich wersji), przez co utrzymanie odpowiadających za to skryptów jest bardzo trudnym i nigdy niekończącym się zadaniem.

- Z definicji metoda ta nie jest przyszłościowa. Można wykrywać tylko te przeglądarki, które są dostępne obecnie, a wszystkie te, które pojawią się w przyszłości, będą blokowane, dopóki nie doda się ich do listy.
- Nawet najpełniejsza lista dopuszczonych przeglądarek może nie spełniać swojego zadania, ponieważ niektóre przeglądarki są skonfigurowane w taki sposób, aby zgłaszały się jako inna aplikacja, aby obejść właśnie te techniki (nazywa się to czasami „robieniem w balona" — ang. *browser spoofing*).

Zatem pierwsze z wymienionych podejść — dostarczanie ulepszeń przeglądarkom z włączoną obsługą JavaScriptu — wydaje się lepszym rozwiązaniem, ponieważ większość przeglądarek obsługujących JavaScript jest w stanie poradzić sobie z rozszerzeniami. Przypomnijmy jednak, że napotkaliśmy zaskakująco dużo przypadków tylko częściowego obsługiwania pewnych ulepszeń przez niektóre przeglądarki, co powodowało błędy JavaScript i układu stron.

Po uważnym przeanalizowaniu tych sytuacji odkryliśmy dwa główne powody ich występowania: niektóre przeglądarki źle interpretowały kod CSS, ponieważ obsługiwały niezgodnie ze standardami model polowy (ang. *box model*), pozycjonowanie, pływanie elementów i wiele innych często używanych własności CSS. Inne natomiast niepoprawnie wykonywały funkcje JavaScript, jak operowanie na modelu DOM, obsługa zdarzeń, zmienianie rozmiaru okna oraz wykonywanie żądań Ajax.

Zastosowanie techniki stopniowego ulepszania byłoby znacznie łatwiejsze, gdybyśmy mieli jakiś niezawodny sposób na uzyskanie przekroju *możliwości* przeglądarek. Wówczas moglibyśmy dostarczać rozszerzenia tylko tym aplikacjom, dla których mielibyśmy pewność, że CSS i JavaScript są obsługiwane poprawnie.

Mając ten cel w pamięci, metodą prób i błędów opracowaliśmy szkielet testowy funkcjonalności przeglądarek. Podstawowa lista funkcji do sprawdzenia w zakresie obsługi JavaScriptu była w zasadzie oczywista — zastosowaliśmy metodę o nazwie „wykrywanie obiektów" (ang. *object detection*), która polegała na pytaniu przeglądarki, czy rozpoznaje rodzime obiekty, takie jak funkcja `document.getElementById` i otrzymywaniu jednoznacznej odpowiedzi „prawda" lub „fałsz". Każdy test został napisany w taki sposób, aby przeglądarki, które nie rozpoznawały jakiegoś kodu JavaScript, nie zgłaszały żadnych błędów.

Znacznie trudniej było sprawdzić, czy dana przeglądarka poprawnie obsługiwała zaawansowane techniki CSS. Nie da się po prostu sprawdzić, czy przeglądarka poprawnie obsługuje określone własności CSS, takie jak elementy pływające, pozycjonowanie, pionowe ułożenie elementów czy przezroczystość, za pomocą metody wykrywania obiektów.

Dlatego opracowaliśmy zestaw testów CSS. Każdy z nich wstawia na stronę za pomocą JavaScriptu niewidoczne elementy HTML, stosuje zestaw zaawansowanych reguł CSS, a następnie uruchamia funkcję JavaScript sprawdzającą wynik. Aby na przykład dowiedzieć się,

czy przeglądarka poprawnie obsługuje pozycjonowanie, test umieszcza za pomocą CSS w określonym miejscu element `div`, a następnie uruchamia skrypt porównujący uzyskane współrzędne tego elementu z danymi wzorcowymi.

Jeśli przeglądarka przejdzie wszystkie testy z powodzeniem, mamy pewność, że obsługuje CSS i JavaScript w zgodny ze standardami i spójny sposób, a więc powinna poradzić sobie z rozszerzoną funkcjonalnością witryny. W tym momencie test dynamicznie ładuje zaawansowane arkusze stylów i skrypty przekształcające podstawowy kod HTML we wzbogacone środowisko oraz dodaje odnośnik, za pomocą którego można wrócić do wersji podstawowej. Dodatkowo dodaje plik cookie umożliwiający pominięcie wykonywania testów po raz drugi, dzięki czemu następne otwarcie strony będzie szybsze.

Gdy zaczęliśmy stosować nasz pakiet testów, zauważyliśmy wiele korzyści płynących z jego używania w naszych implementacjach stopniowego ulepszania. Przede wszystkim, pozwolił nam skutecznie odróżnić przeglądarki mogące poprawnie zastosować rozszerzone funkcje od przeglądarek, które tego nie potrafiły. Dzięki temu znacznie zmniejszyliśmy ryzyko zrujnowania podstawowej wersji strony przez źle obsłużone rozszerzenia. Ponadto dzięki temu, że rozszerzenia były ładowane tylko w przeglądarkach, które pomyślnie przeszły testy, pozostałym przeglądarkom mogliśmy dostarczyć znacznie prostszy i lżejszy kod — żadne wielkie ilości kodu znaczników, style ani skrypty nie są ładowane z góry. W ten sposób znacznie skróciliśmy czas pobierania stron i ograniczyliśmy liczbę niepotrzebnych odwołań do serwera.

Nasz pakiet testów ma budowę modularną i jest bardzo elastyczny. Można go dostosować do sprawdzania wybranych funkcji CSS i JavaScript wymaganych w konkretnym projekcie. Jeśli na przykład w tworzonej witrynie nie są wykorzystywane skomplikowane struktury elementów pływających albo skrypty Ajax, możemy odpowiadające im testy wyłączyć.

W swoich projektach nasz pakiet testów dzieli przeglądarki tylko na „podstawowe" i „rozszerzone", co pozwala nam zapanować nad kodem i ułatwia jego utrzymanie. To binarne rozdzielenie jest dla nas sposobem na dotarcie do możliwie jak największej liczby urządzeń przy jak najmniejszym wysiłku. Na tej podstawie można oczywiście opracować bardziej zaawansowane skrypty, dzielące przeglądarki na więcej bardziej szczegółowych grup albo dostarczające funkcje zoptymalizowane pod kątem konkretnych urządzeń, np. iPhone lub Kindle.

Strukturę i działanie tego pakietu testów szczegółowo omówimy w rozdziale 6., w którym również przedstawimy kilka sytuacji, w których podejście modularne może być korzystne.

Planowanie stopniowego ulepszania — prześwietlanie

Dostrajając nasz pakiet testów możliwości przeglądarek, zaczęliśmy rozwijać technikę rozkładania skomplikowanych projektów interfejsów sieciowych w celu przygotowania ich do zaprojektowania zgodnie z techniką stopniowego ulepszania.

Czasami udawało się bez problemu znaleźć odpowiedni element HTML do obsługi zaawansowanych składników projektu — indywidualnie wystylizowane menu rozwijane wygląda i zachowuje się tak podobnie do elementu `select`, że oczywiste było, iż należy go użyć. Analogicznie stylizowanie pól wyboru może stanowić nie lada wyzwanie, ale od samego początku wiadomo, że to jest tylko pole wyboru.

Niestety, nie zawsze było tak łatwo. Który z podstawowych elementów HTML powinien zostać użyty do reprezentowania ajaksowego komponentu do oceniania za pomocą gwiazdek w stylu Netflix? To samo dotyczy widocznych na wielu stronach internetowych z wiadomościami komponentów z kartami „most popular/emailed/commented" czy suwaków zakresu daty i ceny służących do filtrowania wyników w witrynie Kayak i innych witrynach e-commerce. Nie wspomnimy już o jeszcze bardziej skomplikowanych, ale bardzo popularnych aplikacjach wykorzystujących Ajax, w których stosowane są techniki przeciągania i upuszczania oraz wiele innych zaawansowanych rodzajów interakcji, takich jak np. Gmail. Żaden z elementów podstawowego języka HTML nie pasuje w naturalny sposób do tych obiektów.

Jednakże, mimo iż wszystkie przedstawione przykłady opisują wysoce zindywidualizowane i interaktywne elementy, oferowane przez nie funkcje — wybieranie opcji na skali, przechodzenie z jednego bloku treści do innego, sortowanie, wyszukiwanie i pobieranie danych — można bez wątpienia uzyskać również za pomocą standardowego HTML-a. Trzeba tylko trochę pomyśleć i rozłożyć je na czynniki w celu określenia elementów HTML, które również mogą zostać użyte do wykonania tego samego zadania.

Najtrudniejsze w tym wszystkim jest zajrzeć pod style, animacje i inne zachowania, aby zobaczyć podstawowe komponenty leżące u samego podłoża. Ten proces analizowania komponentów nazwaliśmy prześwietlaniem. Np. kod CSS i skrypty decydujące o interaktywności suwaka były skórą i ubraniem. Natomiast w podstawowej wersji szkieletem było tekstowe pole wejściowe służące do ustawiania wysokich i niskich wartości liczbowych albo przyciski radiowe reprezentujące niewielki zestaw opcji do wyboru albo nawet menu wyboru zawierające długą listę opcji.

Prześwietlanie staje się bardziej skomplikowane i interesujące, gdy prześwietla się skomplikowane projekty, np. witryny zawierające różne rodzaje treści i interaktywne komponenty, aplikacje sieciowe zawierające treść dynamiczną oraz skomplikowane układy i schematy pracy, które dostarczają treść w zależności od działań użytkownika. I to tylko kilka z wielu możliwych przykładów.

W skali makro przeprowadzamy proces prześwietlania, którego celem jest określenie, jak większe części strony (lub wzorce na wielu stronach witryny) współpracują ze sobą, znalezienie wzorców zachowań, które mogłyby stanowić wskazówkę na temat kluczowych elementów treści i funkcjonalności, oraz ocenienie, jak można zoptymalizować te zasoby, aby działały jak najlepiej zarówno w podstawowej, jak i rozszerzonej wersji.

Podczas tej wysokopoziomowej analizy podstawowe zasady na poziomie komponentów i elementów pozostają niezmienione — w każdym przypadku chodzi o zidentyfikowanie wszystkich najważniejszych informacji i funkcji potrzebnych użytkownikowi, znalezienie elementów HTML, które można wykorzystać w określonej sytuacji (na podstawie formatów treści, wymagań w zakresie danych lub zasad biznesowych, ogólnego przepływu), oraz zdecydowanie się na standardowe elementy HTML, które najlepiej spełnią stawiane zadanie.

Bardziej szczegółowo o prześwietlaniu napiszemy w rozdziale 2., w którym dodatkowo przedstawimy kilka procesów dekonstrukcji na kilku skomplikowanych projektach. Natomiast prześwietlaniem pojedynczych komponentów interfejsu zajmiemy się w rozdziałach 18. i 19. Zobaczymy, jak techniki stopniowego ulepszania pozwalają uzyskać cel utworzenia jak najpełniejszych, najlepiej dostępnych i najbardziej użytecznych wersji rozszerzonej i podstawowej witryny.

Zanim jednak przejdziemy do szczegółowego omówienia praktycznego zastosowania techniki prześwietlania, przedstawimy proces, który opracowaliśmy w celu zastosowania techniki stopniowego ulepszania w naszych projektach.

Od prześwietlania do działania — technika stopniowego ulepszania

W miarę udoskonalania naszego pakietu testowego i stosowania techniki prześwietlania po jakimś czasie wypracowaliśmy swoistą terminologię, za pomocą której opisywaliśmy nasz proces stopniowego ulepszania. Dostarczamy dwie wersje produktów — podstawową, która działa we wszystkich urządzeniach z dostępem do internetu (na ile to możliwe), oraz rozszerzoną, która jest dostarczana tylko odpowiednio zaawansowanym przeglądarkom. Kod pierwszej z nich stanowi podstawę, na której opiera się wszystko pozostałe, dlatego nazwaliśmy go „kodem bazowym". Natomiast kod HTML, CSS i skrypty odpowiadające za zaawansowane funkcje prezentacyjne i zachowania zyskały miano zasobów zaawansowanych lub rozszerzonych.

Aby z powodzeniem dostarczyć te dwie wersje użytkownikom, musimy przestrzegać trzech podstawowych zasad stopniowego ulepszania:

- Zacząć od czystej treści i kodu HTML o poprawnej strukturze.
- Precyzyjnie oddzielić warstwę prezentacyjną od układu.
- Nieinwazyjnie dołączyć zaawansowane zachowania i style po skrupulatnym przemyśleniu zagadnień związanych z dostępnością.

Ponieważ podstawowa wersja jest tworzona w celu umożliwienia dostępu do niej każdemu, pracę musimy *zacząć od klarownej treści i poprawnie skonstruowanego pod względem semantycznym kodu HTML*. Taki kod HTML służy jako funkcjonalna podstawa, na bazie

której można zbudować dodatki rozszerzające. Jest prawdopodobne, że będzie ona działać w większej liczbie różnych urządzeń, a ponadto stanowi klarowną strukturę do poruszania się dla technik wspomagających.

Konstrukcja kodu znaczników w treści strony ma ogromny wpływ na to, jak można będzie zastosować rozszerzenia, np. CSS i JavaScript, oraz to, w jakim stopniu strona będzie dostępna dla niewidomych użytkowników. Najlepiej do stylizowania i dodawania elementów interaktywnych nadają się strony o jasnej i precyzyjnie zorganizowanej strukturze. Ponadto taki kod łatwiej wykorzystać też w innych projektach.

Warunkiem, od którego spełnienia zależy to, czy uda się osiągnąć solidną podstawę służącą do opracowania zarówno podstawowej, jak i rozszerzonej wersji strony, jest zrozumienie właściwości, możliwości i ograniczeń semantyki języka HTML oraz znajomość elementów i atrybutów dostępnych w aktualnie obowiązującej wersji tego języka (nie wspominając o nowych właściwościach, będących jeszcze w fazie powstawania specyfikacji). Dostępne możliwości i najlepsze praktyki, które zalecamy, są opisane w rozdziale 3.

Druga kluczowa zasada, której ściśle się trzymamy w naszym podejściu stopniowego ulepszania, to *oddzielenie układu od treści*. Najpierw budujemy szkielet strony, a dopiero potem wypełniamy go treścią. To pozwala nam znacznie uprościć proces tworzenia spójnego systemu szablonów dla całego projektu. Ponadto oddzielamy od treści cały kod CSS dotyczący prezentacji i stylu.

Gdy struktura i styl strony powstają niezależnie od jej treści, łatwiej jest tworzyć różne wersje tego samego układu i dostarczać wersje działające optymalnie w różnych przeglądarkach i urządzeniach, ponieważ żaden strukturalny kod CSS nie wpływa na styl treści. To z kolei pozwala nam wziąć pod uwagę rozszerzone typy mediów — standardowe ekrany komputerowe, wyświetlacze telefonów komórkowych i urządzenia wspomagające dla niepełnosprawnych — i selektywnie dostarczać podstawowe i zaawansowane style do wybranych z nich. W rozdziale 4. pokazujemy, które rodzaje stylów można bezpiecznie stosować w większości środowisk, jak zachowują się i wzajemnie oddziałują między sobą skomplikowane reguły CSS oraz jak można scentralizować i zoptymalizować style, aby otrzymać najczystsze i najbardziej niezawodne wersje podstawową i rozszerzoną.

Zaawansowane zachowania i elementy prezentacyjne uzyskane za pomocą JavaScriptu mogą znacznie wzbogacić doznania użytkownika. Jeśli jednak zastosuje się je niewłaściwie lub bezmyślnie, mogą one również sprawić, że witryna będzie całkowicie bezużyteczna dla wielu użytkowników. Istnieją bardzo dobre wytyczne i jasno określone najlepsze praktyki tworzenia i dodawania skryptów w taki sposób, aby *nieinwazyjnie wzbogacały strony* i bezpiecznie je ulepszały w przeglądarkach, które potrafią z nich skorzystać, oraz zapewniające, że podstawowa funkcjonalność nie zostanie zatracona. Te zasady i techniki oraz związane z nimi zagadnienia dotyczące dostępności są opisane w rozdziale 5.

Wyposażeni w dodatkową wiedzę na temat współpracy między kodem HTML, CSS i nieinwazyjnym JavaScriptem, szczegółowo przyjrzymy się naszemu pakietowi testów w rozdziale 6. Dowiemy się, jak wykorzystywane są w nim opisane powyżej zasady i techniki w celu umożliwienia zastosowania bardziej niezawodnego podejścia stopniowego ulepszania.

Praktyka

W kilku następnych rozdziałach zwięźle opisujemy najlepsze praktyki, które należy znać, aby móc zaimplementować stopniowe ulepszanie w realnych projektach dla klientów. Natomiast w dalszych rozdziałach nauczymy się pisać poprawny kod HTML, efektywnie stosować CSS oraz dodawać rozszerzenia za pomocą JavaScriptu. Na zakończenie szczegółowo opiszemy nasz pakiet testów.