Optymalizacja serwisów internetowych. Tajniki szybkości, skuteczności i wyszukiwarek

Autor: Andrew B. King
Tłumaczenie: Radosław Meryk
ISBN: 978-83-246-2087-6
Tytuł oryginału: Website Optimization: Speed, Search Engine & Conversion Rate Secrets
Format:168x237, stron: 392

Poznaj sekrety tworzenia popularnych i dochodowych witryn internetowych!

- Jak poprawić widoczność serwisu w wyszukiwarkach?
- Jak zoptymalizować płatne kampanie reklamowe?
- Jak odnieść sukces w Internecie?

Celem optymalizacji witryny WWW jest przede wszystkim maksymalizacja przychodów z serwisu i podniesienie komfortu użytkowników. Aby ten cel osiągnąć, trzeba spełnić kilka podstawowych warunków. Po pierwsze należy zadbać o to, aby serwis można było łatwo odszukać w sieci. Po drugie korzystanie z niego nie może sprawiać trudności, a po trzecie powinien on być szybki, atrakcyjny i interaktywny. Krótko mówiąc, dobry serwis musi być funkcjonalny i wydajny. Aby osiągnąć sukces w sieci, należy zatem wybrać właściwe metody i przyjąć skuteczną strategię działania. Jakie to techniki i jak je stosować, dowiesz się właśnie z tej książki.

Książka „Optymalizacja serwisów internetowych. Tajniki szybkości, skuteczności i wyszukiwarek" wskaże Ci możliwości tworzenia i optymalizacji serwisu internetowego, pozwalające w znacznym stopniu ograniczyć inwestycje w marketing, a jednocześnie zyskać wielu klientów. Z tego podręcznika dowiesz się, jak zoptymalizować serwis pod kątem wyszukiwarek, na czym polega właściwy dobór słów kluczowych, jak zbudować architekturę CSS i tworzyć ciekawe rozwiązania, opierające się na Ajaksie. Ponadto nauczysz się optymalizować kampanie reklamowe dzięki stosowaniu perswazyjnej treści reklam. Po tej lekturze będziesz umiał stworzyć wydajny, funkcjonalny, a przy tym popularny i dobrze zarabiający serwis internetowy.

- Optymalizacja pozycji serwisu w wyszukiwarkach
- Narzędzia do analizowania słów kluczowych
- Nagłówki przyciągające uwagę
- Optymalizacja reklam płatnych (PPC)
- Dobór grupy docelowej i kwalifikowanie odwiedzających
- Wykorzystanie metryk do ustalenia budżetu
- Optymalizacja współczynnika konwersjiv
- Optymalizacja stron WWW
- Budowanie architektury CSS
- Tworzenie własnych rozwiązań, opartych na Ajaksie
- Zaawansowana optymalizacja serwisów WWW
- Metryki optymalizacji wydajności

Optymalizuj serwis i dochody!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Słowo wstępne **9**

Przedmowa **11**

I Optymalizacja marketingu wyszukiwarek **21**

1. Naturalna optymalizacja pozycji serwisu w wyszukiwarkach **25**
Korzyści ze stosowania optymalizacji SEO 26
Zasadnicze techniki SEO 27
Dziesięć kroków do uzyskania wyższych pozycji w rankingach wyszukiwarek 31
Podsumowanie 62

2. Analiza przypadku użycia SEO: PhillyDentistry.com **65**
Serwis w postaci wyjściowej 65
Pierwsza modyfikacja projektu: środek roku 2004 68
Druga modyfikacja projektu: koniec roku 2007 71
Podsumowanie 74

3. Optymalizacja reklam płatnych (PPC) **75**
Podstawowe pojęcia i definicje związane z reklamami PPC 76
Google, Yahoo!, Microsoft i wszyscy inni 78
Określanie celów, pomiary, analiza i zamykanie pętli 83
Identyfikacja, wybór i analiza słów kluczowych 87
Organizowanie i optymalizowanie grup reklam 93
Optymalizacja reklam PPC 95
Optymalizacja stron docelowych 103
Optymalizacja stawek kosztu za kliknięcie 108
Inne problemy związane z kampanią PPC 117
Podsumowanie 123

4. Analiza przypadku kampanii PPC: BodyGlove.com 125
Optymalizacja kampanii PPC serwisu Body Glove 125
Podsumowanie 132

5. Optymalizacja współczynnika konwersji 133
Korzyści ze stosowania optymalizacji CRO 133
Najlepsze praktyki optymalizacji CRO 134
10 najważniejszych czynników wpływających na maksymalizację współczynników konwersji 140
Etapy kampanii CRO 151
Podsumowanie 169

II Optymalizacja wydajności stron WWW 171

6. Optymalizacja stron WWW 179
Znane problemy ze stronami WWW 180
W jaki sposób zoptymalizować szybkość stron WWW? 184
Podsumowanie 209

7. Optymalizacja CSS 211
Budowanie architektury CSS 211
10 wskazówek optymalizacji CSS 214
Podsumowanie 238

8. Optymalizacja Ajaksa 239
Częste problemy z Ajaksem 240
Ajax: nowa, poprawiona komunikacja bazująca na JavaScript 241
Odpowiednie zastosowania technologii Ajax 241
Tworzenie własnych rozwiązań bazujących na Ajaksie 245
Korzystanie z bibliotek Ajaksa 249
Optymalizacja JavaScript 253
Minimalizacja liczby żądań HTTP 266
Rozsądny wybór formatów danych 268
Rozwiązanie problemu z pamięcią podręczną przy korzystaniu z technologii Ajax 271
Rozwiązywanie problemów z rozmiarami sieci 273
Skutki zastosowania architektury Ajax 277
Podsumowanie 278

9. Zaawansowana optymalizacja serwisów WWW 279
Techniki optymalizacji po stronie serwera 279
Techniki poprawy wydajności po stronie klienta 303
Podsumowanie 317

10. Metryki optymalizacji wydajności 319
Metryki sukcesu serwisu internetowego 320
Typy oprogramowania do analizy serwisów internetowych 324
Metryki marketingu w wyszukiwarkach 332
Metryki wydajności serwisów WWW 345
Podsumowanie 369

Skorowidz 371

ROZDZIAŁ 6.

Optymalizacja stron WWW

Celem optymalizacji stron WWW jest uproszczenie ich treści po to, by wyświetlały się szybciej. Szybkie wyświetlanie stron jest kluczem do sukcesu serwisu WWW. Zwiększa zyski, obniża koszty i poprawia satysfakcję klientów (nie mówiąc już o pozycji w rankingach wyszukiwarek, dostępności i łatwości pielęgnacji).

Uproszczenia polegają na takim przekształceniu stron, aby szybciej wyświetlała się treść dostępna natychmiast. Dzięki temu można opóźnić moment ładowania treści z zewnątrz. W tym rozdziale podpowiemy, w jaki sposób zminimalizować liczbę żądań HTTP, przekształcić kod strony na semantyczny zestaw znaczników, ułatwić nadawanie stylistyki dzięki kaskadowym arkuszom stylów (CSS), zoptymalizować grafikę i multimedia oraz opóźnić ładowanie treści z zewnątrz.

W celu maksymalizacji szybkości wyświetlania stron WWW można zastosować 10 technik wymienionych poniżej:

- zminimalizowanie liczby żądań HTTP;
- zmiana rozmiaru i optymalizacja ilustracji;
- optymalizacja multimediów;
- zastąpienie skryptów JavaScript kodem CSS;
- wykrywanie możliwości przeglądarek po stronie serwera;
- optymalizacja kodu JavaScript pod kątem szybkości uruchamiania i rozmiaru plików;
- konwersja na CSS układu bazującego na tabelach;
- zastąpienie stylów wierszowych (ang. *inline*) regułami CSS;
- zminimalizowanie czasu wyświetlania;
- rozsądne ładowanie kodu JavaScript.

Dzięki zastosowaniu dobrych praktyk zaprezentowanych w niniejszym rozdziale przekształcimy kod HTML i multimedia w taki sposób, aby serwis stał się bardziej dynamiczny. Rozpoczniemy od omówienia znanych problemów ze stronami WWW, przed którymi stają inżynierowie wydajności serwisów WWW.

Znane problemy ze stronami WWW

Rozmiar i złożoność kodu wykorzystywanego na stronach WWW w dużej części determinuje ich początkową szybkość wyświetlania. Strony, które są duże i złożone, zwłaszcza te, które zawierają zagnieżdżone tabele oraz niewłaściwie umieszczone wywołania plików CSS i JavaScript, opóźniają moment, w którym wyświetla się użyteczna treść. Uproszczone strony WWW sprawiają *wrażenie* szybszych z powodu szybszego sprzężenia zwrotnego osiąganego dzięki progresywnemu wyświetlaniu. Idea polega na uproszczeniu kodu stron z wykorzystaniem technik bazujących na standardach oraz dążeniu do tego, by kod nie przeszkadzał w ładowaniu treści.

Lokalizacja wywołań plików CSS i JavaScript

Wywoływanie plików CSS na początku strony (wewnątrz elementu `head`), natomiast kodu JavaScript na jej końcu (wewnątrz elementu `body`) umożliwia progresywne renderowanie. Zła lokalizacja kodu CSS lub JavaScript może doprowadzić do opóźnień w renderowaniu treści w przeglądarkach. Więcej informacji znajduje się w punkcie „Umieszczanie kodu CSS na początku, a kodu JavaScript na końcu" w dalszej części tego rozdziału.

Jak pisze Steve Souders w swojej książce *Wydajne witryny internetowe. Przyspieszanie działania serwisów WWW* (Helion), 80% czasu odpowiedzi stron WWW wynika z ich treści. Większą część tego czasu zajmuje obsługa obiektów składających się na stronę WWW. Kiedy liczba obiektów na stronie przekroczy cztery, *koszty czasowe związane z ładowaniem obiektów* stają się dominującym składnikiem opóźnień w ładowaniu stron WWW.

Jak dowiedzieliśmy się z wprowadzenia do części II, większość stron WWW znacznie przekracza próg czterech obiektów. Przeciętnie strona zawiera ponad 50 obiektów, a jej rozmiar przekracza 300 kB. Strony wykorzystujące technologię Ajax, jeśli zawierają błędy kodowania, mogą jeszcze bardziej pogorszyć interaktywność — nawet po załadowaniu strony.

Jak wyraźnie widać, jest miejsce na usprawnienia w wydajności przeciętnej witryny WWW.

Eliminacja niepotrzebnych obiektów

Dzięki upowszechnieniu się technologii Ajax, DHTML i aplikacji agregujących Web 2.0 (ang. *mashups*) niektóre strony WWW zmieniły się z prostych dokumentów HTML w złożone, interakcyjne aplikacje. Zwiększona złożoność jest związana z kosztami: większym rozmiarem stron WWW. W miarę wzrostu złożoności stron WWW proporcjonalnie wzrosła liczba zewnętrznych obiektów. Każdy dodatkowy obiekt to o jedno żądanie HTTP więcej i większe opóźnienia.

Każdy obiekt wprowadza opóźnienia w czasie ładowania strony, zwiększając go średnio o 0,25 sekundy w przypadku połączeń wdzwanianych oraz 40 ms w przypadku połączeń szerokopasmowych[1]. Duża liczba obiektów na stronie jest najbardziej dotkliwa dla użytkowników odległych, ponieważ dalekosiężne połączenia wymagają więcej *przeskoków* i stwarzają więcej okazji do utraty danych.

[1] Chung S. 2007. *The investigation and classifying the web traffic delay & Solution plans presentation*. Referat na konferencji „ICACT2007" 2 (12 – 14 lutego 2007): 1158 – 1161.

Rezygnacja z układu tabel

Tabele są słabym substytutem układu CSS. Pomimo powszechnego przyjęcia CSS w 62,6% serwisów w dalszym ciągu wykorzystuje się układ tabeli[2]. Średnia głębokość tabel od 2006 roku zmniejszyła się o połowę — od prawie 3 do około 1,5[3]. Złożone zagnieżdżone tabele mogą spowodować opóźnienia renderowania w przeglądarkach, ponieważ przed wyświetleniem treści trzeba przetworzyć i wyrenderować tabelę.

W niektórych serwisach sterowanych bazą danych tworzone są moduły treści bazujące na tabelach, które są scalane „w locie" do postaci szablonów. Zagnieżdżone tabele blokują przeglądarki i stają się przyczyną niekorzystnych wartości współczynnika treści do kodu. Wpływa to na obniżenie potencjalnej pozycji stron WWW w rankingach wyszukiwarek.

Głębokość zagnieżdżania tabel można zmniejszyć poprzez stosowanie stylów, etykiet oraz pozycjonowanie obszarów treści przy pomocy CSS. Czasami również do tworzenia układu stron stosuje się prostsze tabele szkieletowe. Następnie można dotrzeć do treści wewnątrz oznaczonych komórek kontenera za pomocą złożonych selektorów następującej postaci:

```
td#main p{}
```

Można również wykorzystać CSS do pozycjonowania, nadania stylu i wskazania treści, w następujący sposób:

```
div#main ul{}
```

Kilka wskazówek dotyczących tworzenia i debugowania układów CSS można znaleźć w punkcie „Krok 7: konwersja układu bazującego na tabelach na CSS" w dalszej części niniejszego rozdziału. Natomiast w rozdziale 7. zamieszczono porady dotyczące tworzenia rozwijanych menu bazujących na CSS. Omówienie wszystkich osobliwości związanych z układem CSS wykracza poza zakres niniejszej książki. Polecamy sięgnięcie do jednej z pozycji dotyczących tego tematu, na przykład *CSS. Kaskadowe arkusze stylów. Przewodnik encyklopedyczny. Wydanie II* autorstwa Erica Meyera (Helion).

Optymalizacja grafiki o dużej objętości

Przeciętnie 54% zawartości strony WWW stanowią elementy graficzne[4]. Na przeciętnej stronie WWW więcej niż 60% pikseli w części strony widocznej na ekranie stanowią elementy graficzne[5].

---

[2] Z przeprowadzonego na potrzeby niniejszej książki, w lipcu 2007 roku, badania losowych 500 stron zaindeksowanych przez Ryana Leveringa z Uniwersytetu Binghamton wynika, że w 62,6% stron wykorzystano znacznik table, natomiast w 85,1% wykorzystano znacznik div. Średnia maksymalna głębokość zagnieżdżania tabel wyniosła 1,47, a średnia liczba znaczników table na stronie wyniosła 12,57. Średnia maksymalna głębokość HTML wynosiła 15,35 — widać zatem, że znaczniki div zastąpiły zagnieżdżanie tabel. Dane z tego badania są dostępne pod adresem *http://www.websiteoptimization.com/secrets/web-page/survey.xls*.

[3] Levering R. i M. Cutler. 2006. *The Portrait of a Common HTML Web Page*. Referat na konferencji „DocEng '06" (Amsterdam, Holandia: 10 – 13 października 2006), 200. Średni maksymalny poziom zagnieżdżania tabel wyniósł 2,95.

[4] Zgodnie z badaniami Leveringa z 2007 roku, średni rozmiar elementów graficznych wyniósł 118 683 bajty. Średni całkowity rozmiar strony wynosił 218 937 bajtów oraz 266 070 bajtów w przypadku braku kompresji. Tak więc elementy graficzne stanowią co najmniej 54,2% przeciętnej strony WWW.

Niestety, wiele ilustracji wykorzystywanych w internecie ma zbyt duże rozmiary, znacznie odbiegające od optymalnych. Wraz ze wzrostem rozdzielczości aparatów cyfrowych, rozmiary plików cyfrowych zdjęć znacznie wzrosły, przez co rozmiar niektórych elementów graficznych na stronach internetowych przekracza 1 MB. Próbę przeglądania tak dużych zdjęć w połączeniach wdzwanianych można porównać do prób przeprowadzenia wielbłąda przez ucho igielne.

Koszt reklam w formie banerów

Większość popularnych serwisów multimedialnych i blogów do generowania przychodów wykorzystuje bogato ilustrowane reklamy. Efekt ten jest osiągany kosztem większej liczby obiektów o około jedną szóstą oraz większego opóźnienia o około jedną trzecią[6]. Z badań pierwszych 1 300 serwisów w rankingu firmy Alexa (*http://www.alexa.com*) wynika, że 56% tych stron WWW zawiera reklamy lub „treści dodatkowe" w różnej postaci[7]. Zablokowanie reklam spowodowało zmniejszenie liczby obiektów, a co za tym idzie — bajtów o 25% do 30%, co skutkowało proporcjonalnym zmniejszeniem opóźnień.

Analizie poddaliśmy strony WWW zawierające od 300 kB do 500 kB reklam w formie banerów. Bez zastosowania odpowiednich strategii rozmiaru reklam wpływ reklam na opóźnienia może być jeszcze bardziej znaczący. W przypadku stosowania reklam graficznych należy określić reklamodawcom kryteria rozmiaru plików. Powinny one uwzględniać wymiary banerów.

Wzrost liczby i objętości reklam spowodował znaczące opóźnienia w ładowaniu stron WWW. Wyświetlanie reklam wiąże się również z kosztami zdalnego hostingu (w większości przypadków) oraz dodatkowym kodem potrzebnym do wyświetlenia reklamy na ekranie (zazwyczaj do tego celu wykorzystuje się JavaScript). Zdalny kod JavaScript to najmniej wydajna metoda dostarczania reklam. Jest ona jednak powszechnie stosowana ze względu na wygodę. W punkcie „Krok 1: minimalizacja liczby żądań HTTP" pokazano sposób dostarczania reklam z wykorzystaniem operacji włączania po stronie serwera. Dzięki temu liczba żądań HTTP jest mniejsza. W punkcie „Krok 10: rozsądne ładowanie kodu JavaScript" nauczymy się sposobów asynchronicznego ładowania kodu JavaScript.

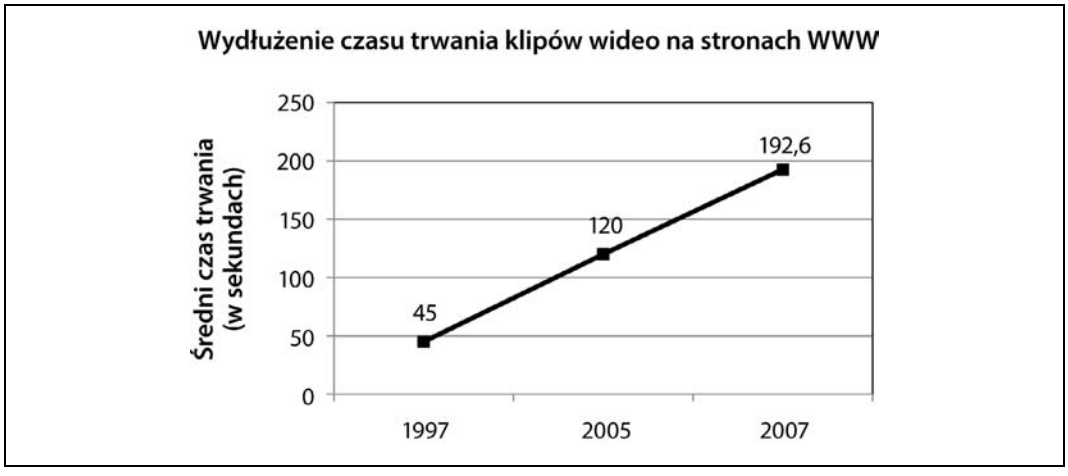
Rozwój multimediów

Wzrost popularności technologii Flash oraz takich serwisów, jak *YouTube*, *Yahoo! Video* i *MySpace*, spowodowały częstsze wykorzystywanie w internecie plików multimedialnych. W miarę rozpowszechnienia się łącz szerokopasmowych zwiększył się rozmiar, współczynnik bitowy (ang. *bit rate*) oraz czas trwania klipów wideo (patrz rysunek 6.1).

[5] Levering i Cutler. *The Portrait of a Common HTML Web Page*, 200. Na przeciętnej stronie WWW ponad 60% obszaru widocznego na ekranie bez przewijania stanowi grafika.

[6] Krishnamurthy B. i C. Wills. 2006. *Cat and Mouse: Content Delivery Tradeoffs in Web Access*. Referat na konferencji „WWW 2006" (Edynburg, Szkocja: 23 – 26 maja 2006), 337 – 346.

[7] Ibid., 346.



Rysunek 6.1. Wydłużenie czasu trwania klipów wideo na stronach WWW

W 1997 roku czas trwania 90% klipów w internecie wynosił mniej niż 45 sekund (patrz rysunek 6.1)[8]. W 2005 roku średni czas trwania klipu wideo wynosił około 120 sekund[9]. W 2007 roku średni czas trwania klipu wideo wyniósł około 192,6 sekundy[10]. Średni współczynnik bitowy klipów wideo umieszczanych na stronach serwisu *YouTube* wzrósł z 200 kb/s w 2005 roku do 328 kb/s w roku 2007. Tak więc średni rozmiar plików wideo w końcu 2007 roku wyniósł ponad 63 MB.

Większość ruchu związanego z multimediami dotyczy plików większych niż 1 MB, ale większość żądań dotyczy plików mniejszych niż 1 MB. Użytkownicy przerywają jednak odtwarzanie ponad 87% wszystkich strumieni multimedialnych w ciągu pierwszych 10 sekund, marnotrawiąc nawet do 20% przepustowości serwera[11]. Chociaż tylko 3% odpowiedzi serwera dotyczy klipów wideo, stanowią one aż 98,6% wszystkich przesyłanych bajtów[12]. Tak więc, chociaż klipy wideo stanowią tylko niewielki procent żądań, to one tworzą największą część ruchu w internecie.

Ogólnie rzecz biorąc, dla klipów wideo dłuższych niż 30 sekund około 13% użytkowników domowych i około 40% użytkowników biznesowych doświadcza obniżenia jakości przekazu strumieni multimedialnych. Przyczyną jest buforowanie, przełączanie strumieni oraz anulowanie wyświetlania klipów wideo. Dla sesji trwających ponad 300 sekund wyniki są jeszcze gorsze. W punkcie „Krok 3: optymalizacja treści multimedialnych" nauczymy się sposobów zwalczania nadmiernego rozrastania się plików multimedialnych z wykorzystaniem specjalizowanych narzędzi i technik.

---

[8] Acharya S i B. Smith. 1998. *An Experiment to Characterize Videos Stored On the Web*. Referat na konferencji „MMCN" 1998 (San Jose, w stanie Kalifornia, styczeń 1998), 166 – 178.

[9] Li M. i inni. 2005. *Characteristics of Streaming Media Stored on the Web*. „ACM Transactions on Internet Technology" 5 (4): 601 – 626.

[10] Gill P. i inni. 2007. *YouTube Traffic Characterization: A View From the Edge*. Referat na konferencji „IMC 2007" (San Diego: 24 – 26 października 2007), 15 – 28. Około 24% klipów wideo jest przerywanych z powodu niskiej wydajności łącz lub słabej jakości.

[11] Guo L. i inni. 2005. *Analysis of Multimedia Workloads with Implications for Internet Streaming*. Referat na konferencji „WWW 2005" (Chiba, Japonia: 10 – 14 maja 2005), 519 – 528.

[12] Gill P. i inni. *YouTube Traffic Characterization*, 20.

W jaki sposób zoptymalizować szybkość stron WWW?

Optymalizację szybkości wyświetlania stron WWW należy rozpocząć od pozbycia się wszystkich stylów definiowanych na poziomie wiersza. Trzeba sprowadzić kod strony do czystej struktury HTML. Następnie przyjrzeć się stronie i zastanowić się, czy istnieją elementy, które można stworzyć w sposób bardziej efektywny. Często możliwe okazuje się zastąpienie elementów bazujących na układzie tabeli strukturalnymi elementami HTML wraz z kodem CSS.

Czym są pliki sprite?

Pliki sprite, używane przez programistów dwuwymiarowych gier do oszczędzania zasobów, zostały zaadaptowane na stronach WWW za pośrednictwem CSS. Sprite zaimplementowany w CSS to siatka ilustracji połączonych w jeden obraz. Tak utworzony sprite ustawia się następnie w kodzie CSS jako obraz tła dla wielu klas. Poszczególne komórki dla każdej z klas wyświetla się za pomocą pozycjonowania tła. Zastosowanie obiektów sprite w CSS pozwala na zmniejszenie liczby żądań HTTP. Podczas korzystania z nich trzeba jednak zachować ostrożność, ponieważ ich użycie wpływa na dostępność serwisu. Analizę wykorzystania obiektów sprite w serwisie *AOL.com* można znaleźć w rozdziale 7.

Po wyeliminowaniu z kodu HTML stylów poziomu wiersza oraz przemodelowaniu go należy dokonać konwersji stylów wbudowanych na arkusze CSS bazujące na regułach. Aby umożliwić wyświetlanie progresywne, należy włączać pliki CSS w nagłówku strony, natomiast skrypty JavaScript powinny się znaleźć na końcu elementu `body`. Warto dążyć do zminimalizowania liczby żądań HTTP poprzez łączenie plików oraz konwersję tekstu w formie graficznej na postać CSS. W celu zminimalizowania liczby żądań HTTP należy korzystać z odstępów CSS, obiektów sprite, map obrazkowych oraz kolorów tła. Wszystkie ilustracje i elementy multimedialne trzeba sprowadzić do najniższej akceptowalnej jakości oraz częstotliwości wyświetlania klatek. Należy włączyć buforowanie trwałych obiektów i rozmieścić je na różnych serwerach w celu zminimalizowania opóźnień. Na koniec — skorzystać z mechanizmów kompresji HTTP w celu zmniejszenia średnio o 75% rozmiaru plików XHTML, CSS i JavaScript. Sposób konfiguracji buforowania i kompresji HTTP na serwerze WWW omówimy w rozdziale 9.

Semantyczne znaczniki

Powstanie omówionych technik wiąże się z przejściem na standardy WWW (XHTML 1.0 oraz CSS2 lub 3). Konwersja niesemantycznego kodu w starym stylu na kod semantyczny ułatwia adresowanie nieprzyległych elementów za pomocą tzw. **selektorów potomka** (ang. *descendant selectors*).

Na stronach WWW tworzonych ręcznie oraz za pomocą niektórych programów typu WYSIWYG mogą się znaleźć sztuczne elementy struktury HTML. W tej „fałszywej strukturze" wykorzystywany jest znacznik `font` oraz kod CSS w celu sztucznej symulacji znaczników strukturalnych, takich jak `<h1>`, `<dl>` oraz `<ul>`[13]. Jednym z problemów dotyczących elementów sztucznej struktury jest brak łatwego sposobu adresowania za pomocą selektorów typu lub potomka — mechanizmów zaprojektowanych do wskazywania elementów strukturalnych.

[13] Z badań przeprowadzonych przez Leveringa w 2007 roku wynika, że w 32,8% stron poddanych analizie wykorzystano znacznik font, a tylko w 58,5% wykorzystano znacznik h1.

Wskazówki tworzenia wysokowydajnych serwisów WWW

Niżej wymienione wskazówki pochodzą z książki Steve'a Soudersa *Wydajne witryny internetowe. Przyspieszanie działania serwisów WWW* (Helion, 2008):

- ogranicz liczbę żądań HTTP w celu zmniejszenia kosztów związanych z obiektami;
- korzystaj z sieci dystrybucji zawartości (ang. *Content Delivery Network*);
- skorzystaj z nagłówka `Expires`;
- stosuj kompresję w odniesieniu do komponentów tekstowych;
- ładuj arkusze stylów na początku — w obrębie elementu `head`;
- umieszczaj skrypty w dolnej części elementu `body`;
- unikaj wyrażeń w kodzie CSS; takie wyrażenia intensywnie wykorzystują procesor i są obliczane stosunkowo często;
- kod JavaScript i CSS umieść w osobnych plikach;
- aby zmniejszyć opóźnienia wnoszonych przez system DNS, zminimalizuj liczbę operacji wyszukiwania nazw w systemie DNS; w tym celu podziel operacje wyszukiwania pomiędzy dwa do czterech oddzielnych hostów;
- zmniejsz objętość kodu JavaScript;
- unikaj przekierowań, ponieważ wpływają one na obniżenie wydajności; zamiast nich lepiej skorzystać z rekordu CNAME lub aliasów;
- usuń zdublowane skrypty w celu wyeliminowania nadmiarowych żądań HTTP wysyłanych przez przeglądarkę Internet Explorer;
- skonfiguruj nagłówki `ETags` dla serwisów rozmieszczonych na wielu serwerach; instrukcja `FileETag none` w konfiguracji serwera Apache usuwa nagłówki `Etags` i pozwala uniknąć nieprawidłowej interpretacji treści umieszczonych w buforze;
- zadbaj o możliwości buforowania żądań Ajaksa, aby uniknąć zbędnych żądań HTTP.

Dodatkowo zastosowanie prawidłowych, strukturalnych znaczników w postaci nagłówków, akapitów i elementów list umożliwia przekazywanie przydatnych informacji osobom zarządzającym serwisem. Wykorzystanie semantycznych znaczników pozwala zaoszczędzić wiele godzin podczas modyfikacji projektu witryny. Wyszukiwarki analizują strukturalne znaczniki i na tej podstawie oceniają, które informacje są najistotniejsze. Wykorzystanie strukturalnego zestawu znaczników ułatwia dostęp do serwisu na wielu platformach (na przykład za pomocą urządzeń przenośnych). Elementy struktury są rozpoznawane przez czytniki ekranu. Dzięki semantycznym znacznikom użytkownicy mogą łatwiej poruszać się po serwisie. Z tych powodów należy za wszelką cenę unikać fałszywej struktury.

Na przykład, następujący kod (stworzony na podstawie kodu rzeczywistej strony WWW):

```
<p style="color:red"><strong>Fałszywe opisowe hasło</strong><br>
   <font size="2" color="black">Tu znajduje się opis pierwszego hasła; brak
struktury, brak możliwości adresowania!</font><br>
<strong>Fałszywe opisowe hasło nr 2</strong><br>
   <font size="2" color="black">Tu znajduje się opis drugiego hasła, brak
struktury, brak możliwości adresowania</font></p>...
```

dzięki wyodrębnieniu stylów poziomu wiersza i zastosowaniu elementów struktury przyjmuje następującą postać:

```
<style type="text/css">
<!--
dl dt{font-weight:bold;color:red;}
dl dd{font-size:0.9em;color:#000;}
--></style></head><body>
<dl>
    <dt>Opisowe hasło nr 1</dt>
    <dd>Opis hasła nr 1; nie ma problemów z adresowaniem</dd>
    <dt>Opisowe hasło nr 2</dt>
    <dd>Opis hasła nr 2; styl nadany za pomocą pojedynczej reguły CSS</dd>
</dl>
```

Zwróćmy uwagę, jak czytelny jest strukturalny kod HTML w porównaniu z przykładem kodu bez struktury.

Kod w drugim przykładzie jest łatwiejszy do zaadresowania za pomocą CSS. Wystarczy wykorzystać proste selektory potomków (`dl dt` i `dl dd`). Pierwszy przykład, pozbawiony semantyki, wymusza użycie wbudowanych stylów. Więcej informacji na temat standardów w internecie można znaleźć pod adresem *http://www.webstandards.org* oraz *http://www.w3.org*.

Wykorzystanie komórek kontenera dla selektorów potomka

Jeśli od początku zaplanujemy użycie na stronie WWW elementów `div` kontenerów — głównymi kontenerami mogą być `#masthead`, `#content`, `#navigation` i `#footer` — możemy uniknąć konieczności osadzania klas w obrębie elementów strukturalnych. W takim przypadku możemy adresować zamknięte fragmenty treści za pomocą selektorów potomków. Na przykład, poniższe menu nawigacyjne może zostać zaadresowane za pomocą otaczającego elementu `nawigacja`:

```
<style type="text/css">
<!--
#nawigacja ul, #nawigacja ul li {list-style:none;}
#nawigacja ul li {font-weight:bold;}
--></style>
```

Oto kod HTML:

```
<div id="nawigacja">
    <ul>
        <li>Element 1</li>
        <li>Element 2</li>
    </ul>
</div>
```

Możemy teraz zadeklarować te style dla *wszystkich* elementów nawigacji, treści oraz innych obszarów, bez konieczności osadzania klas wewnątrz elementów HTML. Idea polega na sprowadzeniu kodu HTML do jego struktury, pogrupowaniu zawartości w obrębie elementów `div` oznaczonych etykietami i zaadresowaniu tej struktury za pomocą selektorów CSS — potomka lub innych.

Gdyby wszystkie przeglądarki zachowywały się tak dobrze, jak Opera, Firefox i Safari, można by użyć elementów „grupujących”, takich jak `body` i `html`, aby uniknąć osadzania klas w obrębie elementów `div` kontenerów. Zamiast tego zalecamy używanie elementów `div` oznaczonych etykietami: `#nawigacja`, `#treść` i `#stopka`. Należy wykorzystać identyfikatory CSS dla

głównych elementów `div` kontenerów używanych tylko raz na stronie, a następnie wykorzystać klasy CSS dla większości pozostałych elementów. Trzeba pamiętać, że identyfikatory CSS są bardziej szczegółowe od klas CSS.

Teraz, kiedy dowiedzieliśmy się, w jaki sposób można pokonać znane problemy ze stronami WWW oraz kiedy poznaliśmy podstawy teoretyczne dotyczące tych technik, możemy przejść do omówienia 10 podstawowych kroków, które można wykorzystać w celu optymalizacji stron WWW.

Krok 1: minimalizacja liczby żądań HTTP

Każdy obiekt na stronie WWW wymaga dwukierunkowej komunikacji do serwera i z powrotem — tzn. żądania HTTP i odpowiedzi. Każdy obiekt wprowadza nieoznaczone opóźnienia. Jak dowiedzieliśmy się z wprowadzenia do części II, w przypadku, gdy liczba obiektów na stronie przekroczy cztery, *koszty czasowe związane z ładowaniem obiektów* stają się dominującym składnikiem opóźnienia w ładowaniu stron WWW.

Poprzez zminimalizowanie liczby obiektów na stronach WWW możemy zminimalizować liczbę żądań HTTP potrzebnych do renderowania strony, a tym samym zmniejszyć koszty czasowe związane z ładowaniem obiektów. Dzięki zmniejszeniu liczby żądań HTTP przyspieszamy ładowanie stron i powodujemy, że operacja ładowania staje się bardziej spójna.

Kluczem do zminimalizowania liczby żądań HTTP jest połączenie plików oraz konwersja technik graficznych na postać CSS. Możemy przekształcić tekst w postaci graficznej na tekst CSS, połączyć zewnętrzne ilustracje, skrypty oraz arkusze CSS, a także wyeliminować ramki i dołączany kod JavaScript. Komórki odstępów można przekształcić na marginesy CSS, a operacje implementowane za pomocą JavaScript zastąpić technikami CSS bazującymi na instrukcji `:hover`. Wiele graficznych elementów dekoracyjnych można zastąpić jednym obiektem CSS sprite.

Mechanizmy zastępowania obrazów

Idea działania mechanizmów zastępowania obrazów polega na używaniu tekstu (zwykle nagłówków) zamiast statycznych bądź dynamicznych obrazów. Dostępne są takie techniki zastępowania, jak sIFR3 (*http://novemberborn.net/sifr3*), swfIR (ang. *swf Image Replacement* — *http://www.swfir.com/*), a także schemat DTR (ang. *Dynamic Text Replacement*) Stewarta Rosenbergera (*http://www.stewartspeak.com/projects/dtr/*). Warto zwrócić uwagę, że czytniki ekranu niezbyt dobrze interpretują ukrywanie i wyświetlanie obrazów za pomocą CSS. Do tego zadania bardziej nadaje się JavaScript.

Konwersja tekstu w formie graficznej na tekst z włączonym stylem

Tekst w formie graficznej często wykorzystuje się w nagłówkach i elementach menu po to, by uzyskać określony wygląd. Współczesne wyszukiwarki nie potrafią czytać tekstu osadzonego w grafice. Tekst w postaci graficznej implikuje również niepotrzebne żądania HTTP. Zamiast tekstu w takiej postaci można wykorzystać CSS w celu włączenia stylu do nagłówków lub skorzystać z mechanizmu zastępowania obrazów (patrz ramka „Mechanizmy zastępowania obrazów"

we wcześniejszej części tego rozdziału). Dzięki przekształceniu tekstu w formie graficznej na CSS w pewnym stopniu tracimy kontrolę nad tekstem, ale zyskujemy na szybkości, potencjalnej pozycji w rankingach wyszukiwarek oraz dostępności.

A zatem kod w takiej oto postaci:

```
<div align="center">
<img src="graphictext.gif" width="115" height="24" alt="przykład tekstu w postaci
graficznej">
</div>
```

Tekst w postaci graficznej

należy zastąpić takim:

```
<style type="text/css">
<!--
h1 {font:bold 18px palatino,times,serif;color:#O3c;text-align:center;}
-->
</style></head><body>
```

<h1>Tekst CSS</h1>**Używanie nakładek tekstowych.** Pokrewną operacją jest wyeliminowanie tekstu w postaci graficznej z obrazów tła. Aby uzyskać wysokiej jakości tekst w formacie JPEG, należy poprawić jakość całego obrazu (tak by rozdzielczość była wyższa niż wyświetlana na stronie) lub skorzystać z mechanizmów kompresji. W niektórych przypadkach wydajniejsze jest usunięcie tekstu z pliku JPEG i nałożenie tekstu w formacie CSS lub, w ostateczności, w postaci przezroczystego pliku GIF bądź PNG z tekstem osadzonym wewnątrz ilustracji. W przypadku nakładki na tekst w formie graficznej uzyskujemy mniejszy rozmiar obrazu tła kosztem dodatkowego żądania HTTP. Nałożenie tekstu CSS pozwala uniknąć tego kosztu.

Przekształcenie komórek odstępu na marginesy i wypełnienia

Powszechną praktyką jest wykorzystanie komórek odstępu zawierających jednopikselowy plik GIF, który jest rozciągany w celu wymuszenia odstępu o odpowiedniej szerokości. Oto przykład zastosowania tego mechanizmu, pochodzący z serwisu *Nasa.gov*:

```
<!-- Pusty wiersz odstępu -->
<table><tr>
<td colspan="2" width="223"><img border="O" alt="" height="1O" width="223" src="/
images/common/spacer.gif"></td>
</tr>
```

Nawet naukowcy zajmujący się rakietami czasem potrzebują pomocy przy kodzie HTML. Lepszym sposobem uzyskania odstępu byłoby dodanie odstępów pomiędzy komórkami:

```
<style type="text/css"><!--
.vmargin {margin-top:1Opx;} --></style></head><body>
<table><tr>
<td colspan="2" width="223" class="vmargin">Tutaj będzie treść</td>
</tr>
```

Jeszcze lepiej skorzystać ze względnych odstępów `em`. Skorzystanie z nich umożliwia wprowadzanie zmian w rozmiarze czcionek za pomocą elementów `user` i `div`:

```
<style type="text/css"><!--
.vmargin {margin-top:1em;} --></style></head><body>
<div class="vmargin">Tutaj będzie treść</div>
```

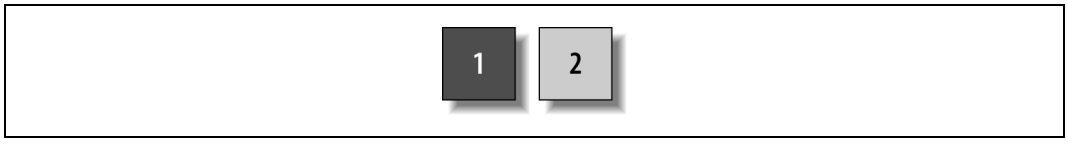
Łączenie kilku obrazów w jeden i stosowanie map obrazkowych lub obiektów CSS sprite

Aby zmniejszyć liczbę żądań HTTP potrzebnych do wyświetlenia stron WWW naszego serwisu, możemy połączyć sąsiadujące ze sobą obrazy w jeden połączony obraz i zmapować łącza za pomocą mechanizmu *mapy obrazkowej*. Zamiast wielu żądań HTTP, w przypadku zastosowania tej techniki jest potrzebne tylko jedno (patrz rysunek 6.2). A zatem kod w następującej postaci:

```
<div align="center">
<h4 align="center">Dwa obrazy = dwa żądania HTTP</h4>
<p><img src="1.gif" alt="pierwszy obraz"> <img src="2.gif" alt="drugi obraz"></p>
</div>
```

dzięki połączeniu dwóch obrazów w jeden i skorzystaniu z instrukcji `usemap` po stronie klienta można przekształcić w postać zamieszczoną poniżej:

```
<div align="center">
<h4 align="center">Jeden połączony obraz = jedno żądanie HTTP</h4>
<map name="mapa1">
<area href="#1" alt="1" title="1" shape="rect" coords="0,0,100,100">
<area href="#2" alt="2" title="2" shape="rect" coords="100,0,210,100"></map>
<img src="combined.gif" width="210" height="100" alt="mapa obrazkowa po stronie
klienta" usemap="#mapa1" border="0">
</div>
```



Rysunek 6.2. Dwa obrazy = dwa żądania

Powyższy kod HTML tworzy mapę obrazkową po stronie klienta z dwoma obszarami docelowymi odpowiadającymi kwadratom „1" i „2" na połączonym obrazie. Dla figury `rect` (prostokąt) współrzędne mierzy się od lewego górnego narożnika obrazu w kierunku prawego dolnego narożnika. Zatem współrzędne `0,0,100,100` definiują obszar rozpoczynający się od lewego górnego narożnika (o współrzędnych `0,0`) do wartości X = 100 pikseli w prawo oraz Y = 100 pikseli w dół (`100,100`).

W rozdziale 7. omówimy sposób wykorzystania obiektów CSS sprite do konsolidacji obrazów.

Łączenie i optymalizacja plików arkuszy CSS i skryptów JavaScript

Wielu programistów tworzy osobne arkusze stylów i importuje je do swoich stron w miarę potrzeb. Takie podejście stwarza dwa problemy: (1) wymaga dodatkowych żądań HTTP oraz (2) może doprowadzić do wyczerpania limitu połączeń w tej samej domenie. Problemów tych można uniknąć dzięki połączeniu plików w nagłówkach dokumentów HTML. Przeglądarki muszą załadować i zinterpretować zewnętrzne pliki CSS wywołane w obrębie elementu `head` dokumentu HTML, zanim zinterpretują treść elementu `body`. Dzięki zmniejszeniu obciążenia spowodowanego liczbą żądań HTTP możemy przyspieszyć ładowanie treści. A zatem kod w następującej postaci:

```
<link rel="stylesheet" type="text/css" href="/css/fonts.css" />
<link rel="stylesheet" type="text/css" href="/css/nav.css" />
<script src="/js/functions.js" type="text/javascript"></script>
<script src="/js/validation.js" type="text/javascript"></script>
```

dzięki połączeniu plików CSS i JavaScript w jeden przyjmuje następującą postać:

```
<link rel="stylesheet" type="text/css" href="/css/combined.css" />
<script src="/js/combined.js" type="text/javascript"></script>
```

Łączenie plików CSS lub JavaScript na serwerze. Podobny sposób minimalizowania liczby żądań HTTP polega na automatycznym łączeniu zewnętrznych plików CSS lub JavaScript. Proces ten jest wykonywany na serwerze i bywa niekiedy określany terminem *zszywanie* (ang. *suturing*). Na żądanie można połączyć wiele arkuszy stylów lub skryptów JavaScript w jeden główny plik. Jeśli operacja ta zostanie odpowiednio przeprowadzona, połączone pliki można również buforować.

Tę cyfrową operację chirurgiczną dla plików CSS przeprowadza się w sposób opisany poniżej. Należy polecić serwerowi wykonywanie dwóch operacji: po pierwsze, interpretowanie kodu PHP w plikach CSS, oraz po drugie, przesyłanie prawidłowego typu MIME. W pliku konfiguracyjnym serwera Apache — *httpd.conf* — należy wprowadzić następujące instrukcje:

```
AddHandler application/x-httpd-php .css
header('Content-type: text/css');
```

Następnie można połączyć pliki CSS z kodem PHP wewnątrz pliku CSS, w następujący sposób:

```
<?php
  include("layout.css");
  include("navigation.css");
  include("advanced.css");
?>
```

Aby pliki były dostarczane na podstawie zmiennych środowiskowych przeglądarki (na przykład aby zasymulować polecenie `@import` w celu odfiltrowania starszych przeglądarek), można skorzystać z programu phpsniff, dostępnego pod adresem *http://sourceforge.net/projects/phpsniff/*.

Buforowanie dynamicznych plików. Jak powiedzieliśmy wcześniej, występują problemy z buforowaniem dynamicznych plików CSS. Jeśli dodamy poniższe nagłówki na początku skryptu PHP, za typem zawartości, pliki te będą buforowane przez trzy godziny (wartość 10 800 sekund można zmodyfikować w miarę potrzeb):

```
header('Cache-control: must-revalidate');
header('Expires: ' . gmdate('D, d M Y H:i:s', time() + 10800) . ' GMT');
```

Umieszczanie kodu CSS na początku, a kodu JavaScript na końcu. Steve Souders odkrył, że przeniesienie arkuszy stylów na początek, do elementu `head`, powoduje szybsze ładowanie stron, ponieważ strony ładują się w sposób progresywny. W przypadku skryptów należy postąpić odwrotnie. Jeśli to możliwe, zewnętrzne pliki JavaScript należy przenieść na koniec stron. Alternatywnie można opóźnić ładowanie skryptów JavaScript w nagłówku. Progresywne renderowanie jest blokowane dla wszystkich elementów, które zostaną umieszczone w kodzie HTML za skryptami.

Optymalizację kodu CSS omówimy bardziej szczegółowo w rozdziale 7. W rozdziale 8. zajmiemy się niektórymi zagadnieniami związanymi z optymalizacją kodu JavaScript. W rozdziale 9. pokażemy, w jaki sposób opóźnić ładowanie skryptów nawet wtedy, gdy zostaną one wywołane wewnątrz elementu `head` dokumentów HTML.

Eliminacja ramek oraz plików dołączanych do skryptów JavaScript

W ponad 52% stron WWW wykorzystuje się ramki, z czego zdecydowana większość to ramki pływające (ang. *iframes*) wykorzystywane do wyświetlania reklam[14]. Ramki (zwykłe i pływające) oraz operacje włączania plików w kodzie JavaScript mogą znacznie obniżyć wydajność stron WWW, ponieważ stwarzają konieczność wysyłania dodatkowych żądań HTTP, a czasami powodują włączanie całych stron WWW w obrębie innych stron WWW.

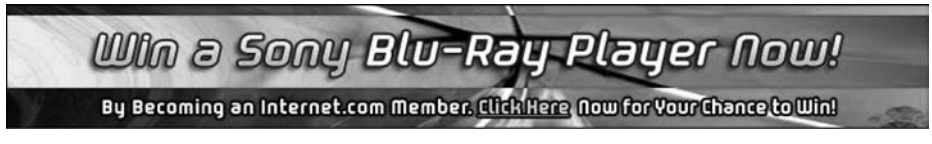
W przypadku reklam dodatkowe żądania HTTP, niezbędne w metodach zaprezentowanych poprzednio, można wyeliminować poprzez zastosowanie systemu dostarczania reklam bazującego na plikach włączanych po stronie serwera, na przykład Open AdStream firmy 24/7 Real Media (*http://www.247realmedia.com*). Oto przykładowy kod z serwisu *Internet.com*. Umieszczenie go na stronie spowoduje dodanie reklamy w formie baneru:

```
<!--#include virtual="/banners/adstream_sx.ads/_PAGE_@750x100-1"-->
```

W tej technice wykorzystano technikę włączania plików po stronie serwera (SSI) w celu włączenia banera reklamowego bezpośrednio na stronie. W ten sposób wyeliminowano konieczność użycia żądania HTTP. Włączony kod ma następującą postać:

```
<div id="topvisibility"><table ALIGN="CENTER">
<tr>
<td align="center">
<A HREF="http://itmanagement.earthweb.com/RealMedia/ads/click_lx.cgi/intm/it/www.
datamation.com/datbus/article/3739896i/1286136569/468x60-1/OasDefault/SSO_BluRay_
GEMS_1d/bluray2_750x100.jpg/343765653435643634376666633265 3O" target="_top"><IMG
SRC="http://itmanagement.earthweb.com/RealMedia/ads/Creatives/OasDefault/SSO_BluRay_
GEMS_1d/bluray2_750x100.jpg"  ALT=""  BORDER="0"></A><img src="http://itmanagement.
earthweb.com/RealMedia/ads/adstream_lx.cgi/intm/it/www.datamation.com/datbus/article/
3739896i/1286136569/468x60-1/OasDefault/SSO_BluRay_GEMS_1d/bluray2_750x100.jpg/
343765653435643634376666633265 3O?_RM_EMPTY_" Width="1" Height="1" Border="0"></td>
</tr>
</table>
</div>
```

Baner reklamowy, który wyświetla się bez potrzeby wysyłania żądania HTTP, pokazano na rysunku 6.3.



Rysunek 6.3. Baner reklamowy dołączony na stronie za pomocą techniki SSI (750×100 pikseli)

Redaktorzy serwisu *Internet.com* po zastąpieniu mechanizmu serwowania reklam bazującego na JavaScript mechanizmem bazującym na SSI zaobserwowali poprawę szybkości działania serwisu. Idea tej techniki polega na zleceniu serwerowi części operacji wykonywanych tradycyjnie przez przeglądarkę.

[14] Z badań przeprowadzonych przez Leveringa w 2007 roku wynika, że większość ramek to ramki pływające (znaleziono je w 51,2% stron). Zwykłe ramki wykorzystywano tylko w 0,8% stron. Należy zwrócić uwagę, że podane liczby nie uwzględniają ramek tworzonych dynamicznie, zatem rzeczywiste wartości są wyższe.

Krok 2: zmiana rozmiaru i optymalizacja ilustracji

Więcej megapikseli! To jest życzenie często powtarzane przez producentów aparatów cyfrowych. W rezultacie tego rozrastania się liczby pikseli zdjęcia publikowane na stronach WWW stały się większe pod względem rozmiaru i mają wyższą rozdzielczość. Często spotykamy serwisy zawierające niezoptymalizowane pliki JPEG w pełnym rozmiarze lub częściowo zoptymalizowane pliki zmniejszone do rozmiaru miniaturek za pomocą atrybutów `height` i `width`. Choć pliki te czasami zajmują obszar zaledwie 100×100 pikseli, zdarza się, że ich rozmiar wynosi ponad 1 MB. Jeden megabajt to około sto razy za dużo w porównaniu z tym, ile powinna wynosić prawidłowa objętość tych plików.

Lepszym sposobem jest przycięcie i zmiana rozmiaru obrazów do ich końcowych wymiarów — takich, w jakich mają się wyświetlić na stronie. Następnie należy je zoptymalizować za pomocą dobrego programu graficznego, na przykład Photoshop lub Fireworks. Za pomocą specjalizowanych narzędzi graficznych takich firm, jak BoxTop Software (*http://www.boxtopsoft.com*), VIMAS Technologies (*http://www.vimas.com*), Xat (*http://www.xat.com*) czy Pegasus Imaging (*http://www.pegasusimaging.com*), można osiągnąć wyższe współczynniki konwersji. Idea polega na zredukowaniu grafiki do najniższej akceptowalnej jakości i rozdzielczości w internecie (72 dpi).

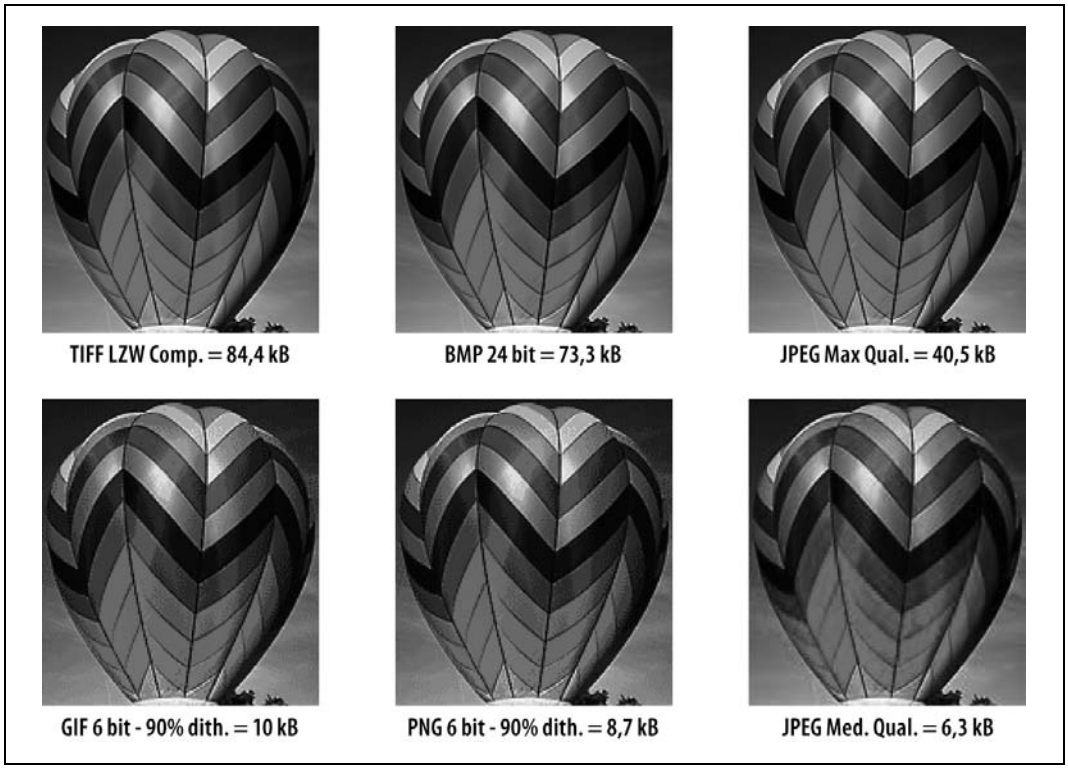
Program JPEG Wizard firmy Pegasus Imaging jest jednym z niewielu narzędzi optymalizacji grafiki, za którego pomocą można zdekompresować i ponownie poddać kompresji obrazy JPEG bez strat właściwych dla cyklicznej dekompresji i kompresji. Efekt ten osiągnięto dzięki przekształcaniu obrazu za pomocą dyskretnej transformaty kosinusowej (ang. *Discrete Cosine Transform* — DCT) w celu wyeliminowania fazy dekompresji.

Istnieje możliwość przełączania formatów, co pozwala zaoszczędzić jeszcze więcej bajtów. Na przykład, często można zastąpić format PNG-8 z opcją rozsiewania (ang. *dithering*) lub bez niej obrazami JPEG lub GIF o mniejszym rozmiarze pliku. Wpływ formatu i jakości pliku na jego rozmiar pokazano na rysunku 6.4.

Pliki w formacie TIFF, BMP oraz JPEG w maksymalnej jakości nie nadają się do publikowania na stronach WWW (patrz pierwszy wiersz na rysunku 6.4; wszystkie pliki zapisane w programie Photoshop miały rozmiar powyżej 40 kB). Przełączenie się na inny format pliku może spowodować znaczącą różnicę w rozmiarze pliku. Grafika z rysunku 6.4 w formacie PNG ma o 13% mniejszy rozmiar od obrazu w formacie GIF przy analogicznych ustawieniach. Choć zaprezentowany przykład wielokolorowego balonu jest ekstremalny, w przypadku większości obrazów o mniejszej liczbie kolorów format PNG, z powodu lepszego algorytmu kompresji, ma o 10% do 30% mniejszy rozmiar od formatu GIF.

Oto zestawienie technik, które można zastosować w celu pełnego zoptymalizowania obrazów. Lista ta pochodzi ze strony *http://www.websiteoptimization.com/speed/tweak/graphic-optimization/*:

- *Planuj zawczasu* w celu maksymalizacji oszczędności w rozmiarach plików (na przykład upraszczaj obrazy tła).
- *Przycinaj obrazy według kontekstu*, tak by wyświetlały się tylko najważniejsze części obrazu.
- *Modyfikuj rozmiar obrazów* do dokładnych wymiarów pikselowych, takich, jakie mają się wyświetlić na stronie.



Rysunek 6.4. Rozmiar pliku obrazu a jego format

- *Łącz obrazy* w celu zmniejszenia liczby żądań HTTP. Opcjonalnie można tworzyć mapy `usemap` lub obiekty CSS sprite.
- *Wykorzystuj efekty rozmywania tła dla obrazów JPEG* — eksperymentuj z ustawieniami „rozmywania powierzchni", aby się przekonać, które z nich gwarantują uzyskanie czytelnego, a jednocześnie uproszczonego wyglądu.
- *Implementuj ramki lub tła za pomocą CSS*, zamiast osadzać ramki w obrazach. Nie pozostawiaj pustych ramek tła w jednym kolorze w celu uzyskania efektów związanych z układem. Zamiast tego wykorzystuj dokładnie przycięte obrazy w połączeniu z kodowanym kolorem tła.
- *Tam, gdzie to możliwe, zastępuj obrazy w formatach GIF i JPEG obrazami w formacie PNG*; jeśli trzeba, stosuj rozpraszania.
- *Określ rozmiar obrazu* w kodzie HTML za pomocą atrybutów `width` i `height`.
- *Stosuj filtr Smart Sharpen* w programie Photoshop CS2 lub późniejszym w celu wyostrzenia obrazów.
- W celu uzyskania wyższego współczynnika kompresji, zamiast osadzać tekst w obrazach JPEG, *nakładaj tekst za pomocą CSS* lub przezroczystych obrazów GIF bądź PNG.
- Przed optymalizacją *zminimalizuj szumy* we wszystkich obrazach. Zwykle pozwala to na zredukowanie rozmiaru plików o 20% do 30%. Do eliminowania szumów polecamy programy Noise Ninja (*http://www.picturecode.com/*) i Neat Image (*http://www.neatimage.com/*).

- *Zminimalizuj efekt rozsiewania* dla plików GIF i PNG.
- *Zmniejsz głębokość bitową* dla obrazów GIF i PNG.
- *Wykorzystuj ważoną optymalizację* (kompresję obszarów) z wykorzystaniem masek alfa w celu optymalizacji tła w większym stopniu niż pierwszego planu.
- Dla mniejszych obrazów GIF i PNG *stosuj kompresję ze stratami* (jeśli jest dostępna).
- *Zminimalizuj lub wyeliminuj efekt rzucanego cienia* w obrazach wielowarstwowych. Warstwy można dostosować za pomocą programu Photoshop w celu zmniejszenia szerokości i głębokości rzucanych cieni. Dzięki temu można uzyskać wyższy współczynnik kompresji obrazów.

Krok 3: optymalizacja treści multimedialnych

Zgodnie z tym, czego dowiedzieliśmy się z punktu „Rozwój multimediów" we wcześniejszej części tego rozdziału, treści multimedialne stanowią tylko niewielki odsetek liczby żądań do serwera, ale generują większość ruchu w internecie. W związku z tym optymalizacja strumieni multimedialnych — a w szczególności filmów — nabrała większego znaczenia dla maksymalizacji szybkości stron WWW i zmniejszenia rachunków za wykorzystywane pasmo.

Autorzy treści multimedialnych „za kulisami" tworzą tzw. *film referencyjny* wskazujący na filmy o różnych rozmiarach. Zadaniem filmu referencyjnego jest odczytanie szybkości połączenia używanego przez użytkownika z programu QuickTime Control Panel, w celu wybrania właściwej wersji filmu. Użytkownicy zazwyczaj jednak nie wiedzą o tym, że powinni ustawić ten parametr, w związku z czym pasmo użytkownika zwykle nie jest testowane. W efekcie użytkownicy łącz szerokopasmowych oglądają filmy o niskiej jakości. Na szczęście, w nowszej wersji programu QuickTime Control Panel pojawiło się domyślne ustawienie *automatic*. Choć opcja ta jest przydatna dla osób, które nie wiedzą, jak należy ustawić szybkość połączenia, lepiej ustawić szybkość odpowiadającą odpowiedniemu typowi połączenia — tzn. modemowe, DSL, kablowe itp.

Optymalizacja klipów wideo na potrzeby publikacji w internecie

Filmy zoptymalizowane pod kątem publikacji w internecie powinny być krótkie, mieć niewielkie rozmiary oraz być zoptymalizowane dzięki zastosowaniu odpowiedniego *kodeka*. Spotykaliśmy klipy wideo o czasie trwania od 10 do 30 minut, które były automatycznie ładowane i odtwarzane na stronach głównych serwisów — ich rozmiary wynosiły od 50 MB do 175 MB. Chociaż takie filmy mogą przyciągać uwagę użytkowników łącz szerokopasmowych, lepiej okazać szacunek dla ich pasma i umieścić w miejscu filmu na stronie statyczną ilustrację oraz przycisk odtwarzania.

Warto odwiedzić witrynę Apple.com, gdzie można znaleźć przykład dobrej praktyki prezentacji klipu wideo potencjalnie szerokiej grupie odbiorców (*http://www.apple.com/trailers/*). Firma Apple przyjęła rozwiązanie polegające na umożliwieniu użytkownikom wyboru filmów o różnych rozmiarach w zależności od możliwości wykorzystywanego połączenia. Możliwe jest przeglądanie filmów w różnych rozmiarach — od „małego" (320×240 pikseli) do jakości HD (1 920×1 080 pikseli). Zapewnienie użytkownikom takich możliwości wymagało od firmy Apple wiele pracy — trzeba było wielokrotnie skompresować jeden film do wielu różnych rozmiarów. Ten dodatkowy wysiłek został jednak nagrodzony przez zadowolenie gości serwisu, którzy mogą znaleźć treści multimedialne zgodne z ich potrzebami oraz dostępnym pasmem.

Współczynniki szybkości klatek i wymiary obrazów wideo. Im większa szybkość klatek (wyrażana w liczbie klatek na sekundę — ang. *frame per second* — fps), tym większa postrzegana płynność obrazu. Trzeba jednak pamiętać, że przy tym samym rozmiarze pliku wyższa wartość współczynnika szybkości klatek oznacza 50% więcej danych na klatkę w porównaniu z bezpośrednio niższą wartością współczynnika. Aby poświęcić nieco płynności w celu zwiększenia użyteczności klipu i udostępnienia go większej liczbie użytkowników, można obniżyć współczynnik szybkości klatek do poziomu 8 fps. Trzeba jednak pamiętać, że przy współczynniku szybkości klatek niższym niż 12 fps – 15 fps użytkownicy dostrzegali obniżoną jakość wideo[15].

Minimalny rozmiar obrazu powinien wynosić 320×240 pikseli. Obrazy o mniejszym rozmiarze słabo oddziałują na użytkowników i są trudne do oglądania. Użytkownikom szybszych łącz można zaoferować klipy wideo w rozdzielczości 640×480 pikseli. W celu utrzymania jakości, można zwiększyć współczynnik szybkości przesyłania danych proporcjonalnie do rozmiaru obrazu dzięki wykorzystaniu poniższego wzoru (zwłaszcza dla kodeka H.264):

> Szybkość przesyłanych danych = (liczba klatek na sekundę×szerokość obrazu×wysokość obrazu)/30 000

Równanie to można zapisać następująco:

> DR = (FPS×W×H)/30 000

Należy pamiętać, że podwojenie rozmiaru obrazu (z 320×240 do 640×480) wymaga czterokrotnego (a nie dwukrotnego) wzrostu szybkości przesyłanych danych. Na przykład, film w rozdzielczości 320×240 odtwarzany z szybkością 15 fps musi być skompresowany do około 38,4 kB danych na sekundę, podczas gdy film w rozdzielczości 640×480 przy tej samej szybkości przesyłania dla utrzymania jakości wymaga kompresji do około 153,6 kB danych na sekundę. Więcej informacji na temat kompresji podamy w dalszej części niniejszego rozdziału.

Wskazówki tworzenia klipów wideo: zminimalizuj szumy i ruch. Utworzenie zoptymalizowanych klipów wideo wymaga wyjścia od oryginalnych plików wideo o wysokiej jakości.

Proces tworzenia klipu wideo można porównać do wojny przeciwko niepotrzebnemu cyfrowemu szumowi w treści. Im więcej szumów w klipach wideo, tym mniejsze możliwości kompresji i tym większy ostateczny rozmiar pliku. Im mniej ruchu w filmie, mniejsze szumy i mniej szczegółów w tle, tym mniejszy rozmiar klipu wideo. Oto kilka wskazówek dotyczących tworzenia wysokiej jakości klipów wideo, które łatwo poddają się optymalizacji:

- zminimalizuj ruchy kamery, jeśli to możliwe skorzystaj ze statywu;
- zminimalizuj ruchy filmowanych obiektów;
- stosuj odpowiednie oświetlenie;
- wykorzystuj proste tło lub zastosuj rozmyte tło (unikaj ruchów w tle);
- unikaj zbliżeń i kadrowania;
- korzystaj z profesjonalnego sprzętu;
- używaj cyfrowego formatu;
- jeśli nie można skorzystać ze statywu, użyj żyroskopowego stabilizatora obrazu (*http://www.ken-lab.com*) lub soczewek stabilizujących obraz.

[15] Gulliver S. i G. Ghinea. 2006. *Defining User Perception of Distributed Multimedia Quality*. „ACM Transactions on Multimedia Computing, Communications and Applications" 2 (4): 241 – 257.

Edycja klipów wideo. Po nagraniu klipów wideo przy minimalnych szumach należy zadbać o wyeliminowanie niepotrzebnych ramek i przetestowanie odtwarzania. Dłuższe klipy wideo trzeba podzielić na mniejsze segmenty trwające co najwyżej kilka minut. Warto usunąć te części filmu, które nie mają istotnego znaczenia dla przekazywanego komunikatu. Oto kilka dodatkowych wskazówek:

- zminimalizuj wymiary do standardów obowiązujących w internecie;
- wykorzystuj minimalną szybkość klatek zapewniającą płynne odtwarzanie obrazu;
- wytnij nieostre kształty;
- zredukuj szumy w klipie wideo (za pomocą filtrów);
- dostosuj kontrast;
- dostosuj poziom gamma (w celu umożliwienia przeglądania na wielu platformach);
- przywróć czerń i biel;
- wyeliminuj przeplot;
- wybierz kodek, który najbardziej nadaje się do wykonywanego zadania.

Na przykład, w przypadku kompresji klipu wideo do opublikowania w internecie należy wybrać kodek stosowany w internecie, na przykład H.264 lub WMV. Natomiast do archiwizacji klipu wideo do późniejszego wykorzystania lepiej nadaje się kodek Photo-JPEG. Istnieje około 30 różnych typów kodeków. Każdy z nich ma inne zastosowanie. Platforma, na której będzie odtwarzany klip wideo, powinna determinować wykorzystywany kodek. H.264 jest jednym z lepszych kodeków nadających się do publikowania w internecie oraz odtwarzania w urządzeniach przenośnych (na przykład telefonach komórkowych). Z tego względu w tym rozdziale skoncentrujemy się na nim.

Kompresja klipów wideo przeznaczonych do opublikowania w internecie. Po przygotowaniu i dostrojeniu klipu wideo można przystąpić do jego kompresji. Więcej zwolenników ma *kompresja czasowa* od *kompresji przestrzennej* (ramka po ramce). Należy dokonać kompresji klipu wideo w celu zmniejszenia jego rozmiarów. Dzięki temu można pomyślnie przesłać strumień wideo do docelowej grupy użytkowników. Mniejszy klip wideo jest również łatwiejszy do ściągnięcia. Proces kompresji określa się w branży za pomocą terminu *kodowanie*. Składa się na niego szereg trudnych, wzajemnie zależnych od siebie decyzji:

Format strumienia multimedialnego
: QuickTime, RealMedia czy Windows Media.

Obsługiwane platformy odtwarzania
: Windows, Mac lub obydwie.

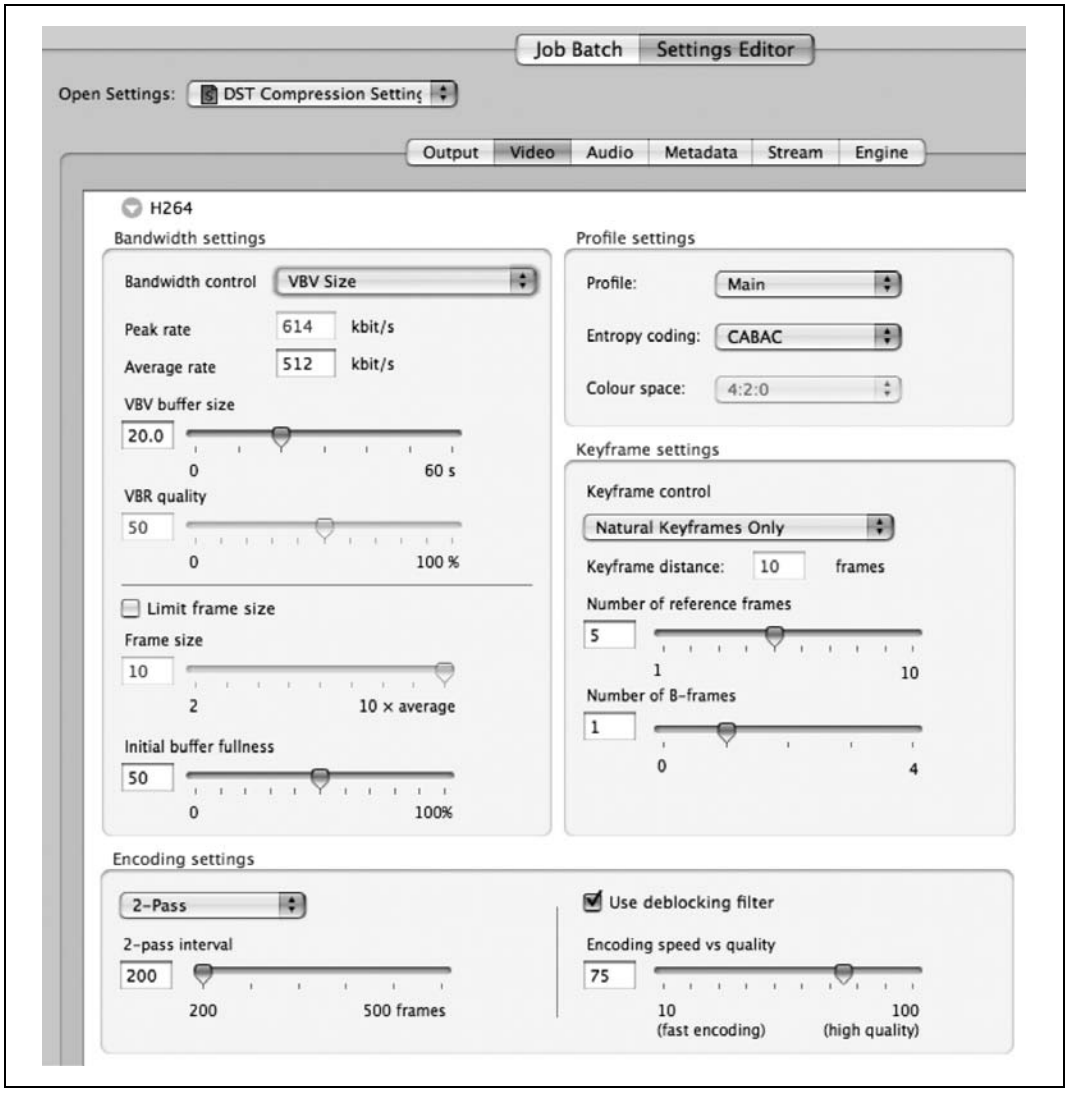
Metoda dostarczania
: Rzeczywisty strumień czy strumień HTTP.

Ogólna szybkość danych
: Kompresja, wymagana jakość a wymagana szerokość pasma.

Jakość dźwięku
: Dźwięk monofoniczny czy stereofoniczny; jakość CD, taśmowa lub telefoniczna.

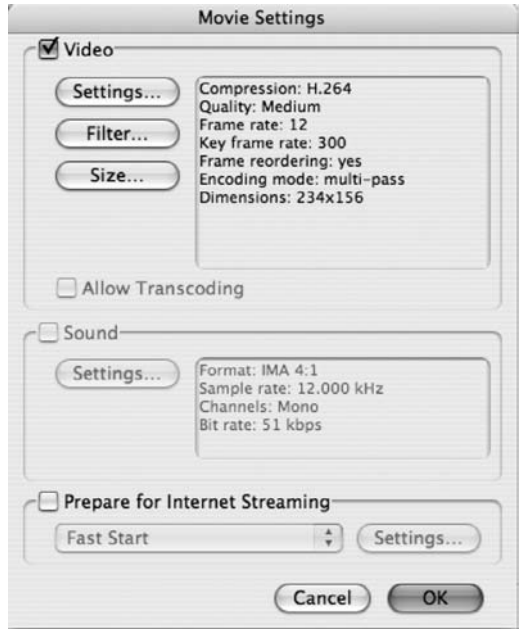
Kodek
: H.264, Sorenson czy WMV (obecnie najczęściej stosowane).

Należy podjąć decyzje pozwalające na uzyskanie najlepszego kompromisu pomiędzy jakością a rozmiarem. Szybki i wygodny sposób tworzenia zoptymalizowanych klipów wideo zapewnia system QuickTime Pro. Większy poziom kontroli można uzyskać dzięki wykorzystaniu programu Cleaner firmy Autodesk (*http://www.autodesk.com*). Użycie systemu Sorenson Video 3 Pro (*http://www.sorensonmedia.com*) w niektórych przypadkach umożliwia tworzenie klipów wideo o mniejszych rozmiarach niż w przypadku zastosowania kodeka H.264, przy podobnej jakości. Wreszcie program Episode Pro firmy Telestream (*http://www.telestream.net/*) oferuje największą kontrolę nad kompresją wideo. Pozwala między innymi na kompresję do formatu H.264, Flash, iPod oraz innych (patrz rysunek 6.5). Jest to doskonała aplikacja pozwalająca na kompresję klipów wideo w trybie wsadowym do wszystkich popularnych formatów i strumieni roboczych.



Rysunek 6.5. Optymalizacja klipu wideo w programie Episode Pro

Na rysunku 6.6 pokazano ustawienia, jakie wykorzystaliśmy do zoptymalizowania testowego klipu wideo w programie QuickTime Pro.



Rysunek 6.6. Optymalizacja klipu wideo w programie QuickTime Pro

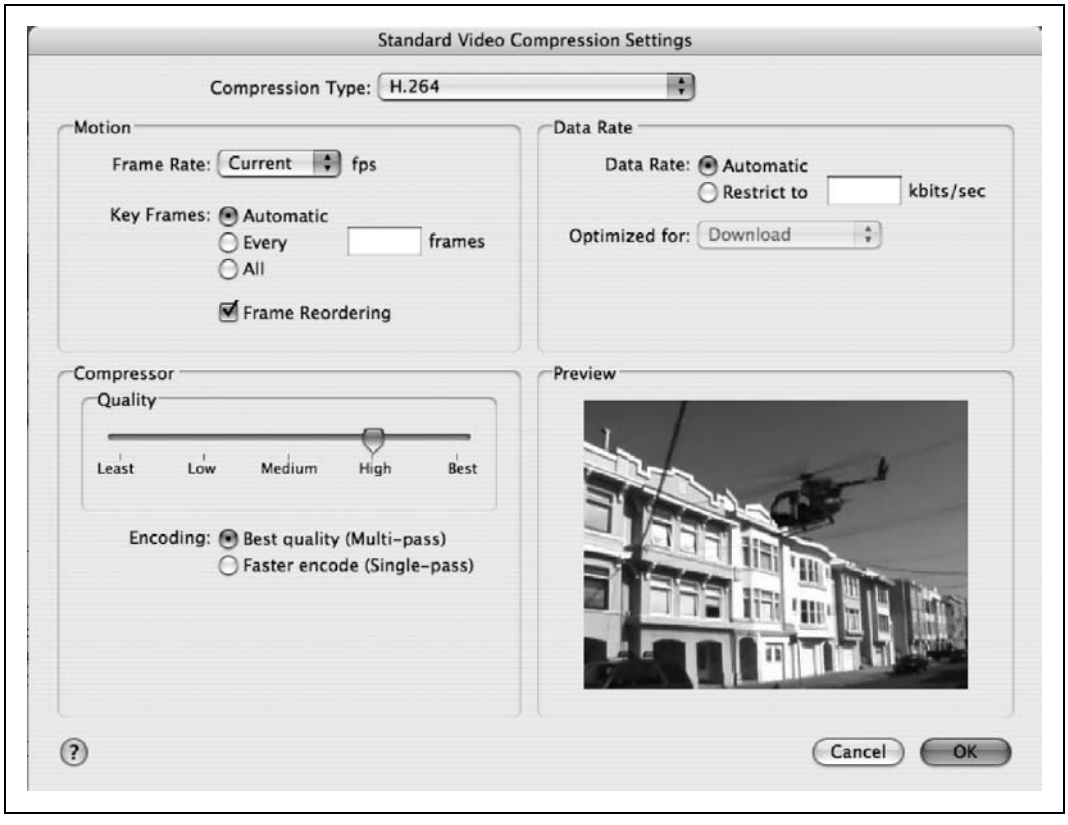
Niezoptymalizowany, 30-sekundowy klip wideo miał rozmiar 6,8 MB, natomiast wersja po optymalizacji miała rozmiar 816 kB przy wymiarach 360×240 oraz 544 kB dla wymiarów 234×156 pikseli. Ponieważ naszym zdaniem najlepszym kodekiem jest H.264, w dalszej części tego rozdziału zajmiemy się nim bardziej szczegółowo.

Standardowe okno dialogowe do ustawiania kompresji wideo w systemie QuickTime Pro pokazano na rysunku 6.7. Jak można zauważyć, ustawiliśmy w nim H.264 jako wybrany typ kompresji. Okno to składa się z trzech głównych części: *Motion*, *Data Rate* i *Compressor*.

W części Motion można określić szybkość klatek (w fps) oraz ramki kluczowe. Jeśli ktoś planuje kompresję klipu wideo, powinien wybrać w polu *Frame Rate* inną opcję niż *Current*, ponieważ w przypadku wybrania wartości *Current* nie zostaną usunięte żadne ramki. Dobrym punktem wyjścia jest szybkość 15 fps. Wartość ta pozwala na 50-procentową redukcję rozmiaru w porównaniu z klipem o szybkości 30 fps (lub dokładniej 29,97 fps).

W sekcji *Motion* znajduje się również obszar *Key Frames* (klatki kluczowe). Klatki kluczowe zawierają nieskompresowane dane i występują co określoną liczbę klatek w klipie. Na ich podstawie tworzone są pozostałe klatki. Jeśli zatem ustawimy częstotliwość klatek kluczowych na 15 (a szybkość klatek wynosi 15 fps), to co druga ramka w klipie pozostanie nieskompresowana.

Liczbę klatek kluczowych zawsze należy ustawiać na wielokrotność szybkości klatek! Jeśli zatem szybkość klatek wynosi 15 fps, to częstotliwość klatek kluczowych należy określić jako 15, 30, 45, 60, 90 itp.



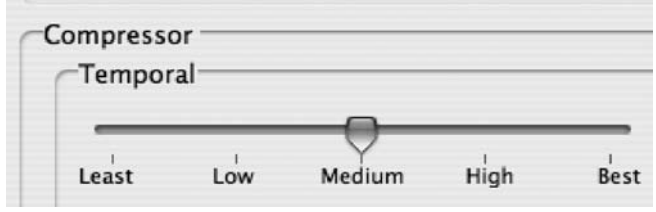
Rysunek 6.7. Standardowe ustawienia kompresji wideo w programie QuickTime Pro

Kodek H.264 zapewnia doskonałą opcję automatycznego określania częstotliwości klatek kluczowych. Warto tę możliwość wypróbować. Należy również pamiętać o zaznaczeniu opcji *Frame Reordering* (chyba że korzystamy z kodowania w czasie rzeczywistym dla przekazu na żywo).

Niżej znajduje się sekcja *Compressor*, w której są trzy opcje: *Quality*, *Encoding* i *Temporal*. Zwróćmy jednak uwagę, że na rysunku 6.7 opcja *Temporal* pozostaje niewidoczna. Jest to opcja ukryta. Wkrótce powiemy, jak można ją znaleźć.

Obszar *Quality* umożliwia kontrolę wyglądu indywidualnych ramek. Najlepiej ustawić tę wartość na *Medium*, dokonać kompresji i sprawdzić, jaki jest rozmiar filmu. Ustawienie *Medium* gwarantuje uzyskanie zaskakująco dobrej jakości. Dla opcji *Encoding* zawsze należy wybrać wartość *Best quality (Multi-pass)*. Co prawda kompresja w tym trybie zajmuje dwukrotnie więcej czasu, ale pozwala na uzyskanie pliku o połowę mniejszego.

Poniżej opisano sposób wyświetlenia ukrytego suwaka *Temporal*. Należy umieścić kursor nad suwakiem *Quality* i wcisnąć klawisz *Option* (w komputerach Mac) lub *Alt* (w komputerach PC). Zwróćmy uwagę, że po wciśnięciu klawisza *Alt* etykieta suwaka zmieni się na *Temporal* (patrz rysunek 6.8). Oznacza to, że można oddzielić ustawienia kompresji przestrzennej (klatka po klatce) od czasowej (płynność odtwarzania lub jakość klatek delta).



Rysunek 6.8. Suwak kompresji czasowej (Temporal)

Ostatnią sekcją okna dialogowego jest obszar *Data Rate*. Przy pierwszej próbie zalecamy kliknięcie przycisku *Automatic*. Jeśli jednak dążymy do zmniejszenia rozmiarów pliku, możemy obniżyć wartość szybkości przesyłania danych. Odpowiednie wartości można odczytać z tabeli 6.1.

Tabela 6.1. Rozdzielczość wideo a zalecane szybkości klatek

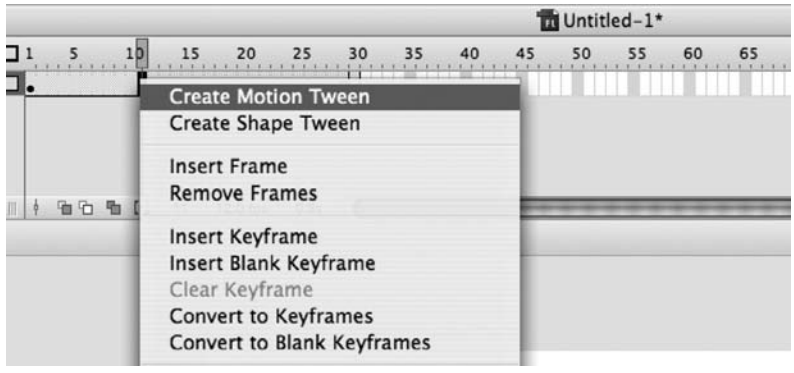
Zastosowanie	Rozdzielczość i szybkość klatek	Przykładowe szybkości danych
Urządzenia przenośne	176×144, 10 – 15 fps	50 – 60 kb/s
Internet/standardowa jakość	640×480, 24 fps	1 – 2 Mb/s
Jakość High Definition	1 280×720, 24 fps	5 – 6 Mb/s
Jakość Full High Definition	1 920×1 080, 24 fps	7 – 8 Mb/s

Podsumowanie. Stworzenie zoptymalizowanych klipów wideo przeznaczonych do opublikowania w internecie wymaga wykonania szeregu skoordynowanych działań.

Po pierwsze, należy stworzyć czysty, pozbawiony szumów klip wideo z jak najmniejszą możliwą liczbą zbliżeń, kadrowania oraz szczegółów i ruchu w tle. Następnie trzeba przygotować klip wideo do kompresji poprzez przycięcie rozmytych krawędzi, dostrojenie kontrastu i parametru gamma oraz usunięcie niepotrzebnych klatek. Na koniec należy poddać klip wideo kompresji za pomocą programu kompresującego wysokiej jakości, na przykład Episode Pro. Punktem wyjścia do obliczenia szybkości przesyłania danych powinien być wzór (FPS×W×H)/30 000. Zawsze należy stosować dwuprzebiegową, zmienną szybkość bitową (VBR), natomiast częstość występowania ramek kluczowych powinna wynosić dziesięciokrotność wartości współczynnika szybkości klatek.

Wskazówki dotyczące optymalizacji animacji Flash

Do typowych problemów z animacjami Flash należą niezoptymalizowane obrazy oraz zbyt wiele klatek zamiast transformacji. *Transformacja* (ang. *tween*) to obliczenie wszystkich zmian pomiędzy klatkami. Wykorzystanie transformacji jest znacznie efektywniejsze od animacji z dużą liczbą klatek (patrz rysunek 6.9). Rozmiar plików z animacjami Flash można znacznie zmniejszyć dzięki optymalizacji obrazów w Photoshopie zamiast we Flashu. Należy zredukować liczbę ramek, zminimalizować liczbę czcionek, a następnie skorzystać z transformacji pomiędzy symbolami.



Rysunek 6.9. Tworzenie transformacji ruchu w animacji Flash

Krok 4: zastąpienie skryptów JavaScript kodem CSS

Wbudowane skrypty JavaScript są powszechnie stosowane w internecie. Fragmenty kodu JavaScript wykorzystuje się w 84,8% ogółu stron.

Kod JavaScript jest wykorzystywany do walidacji danych w formularzach, implementacji menu, efektów rollover, wykrywania własności przeglądarek, statystyk oraz implementacji złożonych aplikacji Ajaksa. Niektóre z tych technik można jednak zaimplementować wydajniejszymi metodami.

Rozwijane menu oraz efekty rollover można zaimplementować za pomocą pseudoklasy CSS `:hover` (więcej informacji na ten temat można znaleźć w książce Erica Meyera *CSS według Erica Meyera. Kolejna odsłona* [Helion, 2005].) Przykład konwersji rozwijanego menu pokazano w rozdziale 7. Dzięki zastąpieniu kodu JavaScript pseudoklasą CSS `:hover` uzyskano zmniejszenie rozmiaru dokumentu HTML o 46,4%. Zazwyczaj, dzięki zastosowaniu technik bazujących na pseudoklasie CSS `:hover`, uzyskuje się obniżenie objętości plików HTML i JavaScript o 40% do 60%, przy tylko nieznacznym wzroście rozmiaru plików CSS (dla których dobrze sprawdza się buforowanie). Teraz, kiedy przeglądarka Internet Explorer w wersji 7 i późniejszych obsługuje pseudoklasę `:hover` dla najważniejszych elementów, powszechnie stosowane obejście dla pseudoklasy `:hover` przestanie wreszcie być potrzebne[16]. Zamiast wykorzystywania statystyk po stronie klienta można używać plików dzienników po stronie serwera. Wykrywanie własności przeglądarek można zrealizować wydajniej za pomocą takich narzędzi, jak BrowserHawk (sposób ten omówiono w następnym punkcie).

Krok 5: wykrywanie możliwości przeglądarek po stronie serwera

Wykrywanie możliwości przeglądarek to jeden z obszarów, w którym powszechnie wykorzystuje się skrypty JavaScript. W celu zminimalizowania ilości kodu JavaScript, jaki muszą ściągać użytkownicy, można zastąpić tę funkcję wykonywanym po stronie serwera kodem

[16] Sposób ten polega na wykorzystaniu kodu JScript w celu dodania pseudoklasy `:hover` do elementów innych niż kotwice w przeglądarkach Internet Explorer 5 do 7. Przeglądarki te nie obsługują prawidłowo pseudoklasy `:hover` dla wszystkich elementów.

PHP lub JSP. W programie BrowserHawk firmy cyScape (*http://www.cyscape.com*) wykorzystano techniki wykrywania możliwości przeglądarek po stronie serwera lub wykrywanie hybrydowe do detekcji różnego rodzaju parametrów, na przykład obsługi Flash, rozmiaru ekranu, szybkości połączenia, plików cookie, a także wersji przeglądarek i oprogramowania (patrz rysunek 6.10).

Rysunek 6.10. Strona główna programu BrowserHawk służącego do odczytywania zmiennych środowiskowych przeglądarki

Wykrywanie możliwości przeglądarek za pomocą programu BrowserHawk

Oto przykład kodu, w którym pokazano wykorzystanie na stronie programu BrowserHawk do odczytania niektórych parametrów:

```
<%
// Najpierw importujemy przestrzeń nazw com.cyscape.browserhawk.
%>
<%@ page import = "com.cyscape.browserhawk.*" %>

<%
// Otrzymujemy egzemplarz obiektu przeglądarki w trybie tylko do odczytu.
//  Obiekt ten reprezentuje podstawowe własności obsługiwane przez przeglądarkę.
%>
```

```
<% BrowserInfo browser = BrowserHawk.getBrowserInfo(request); %>

<%
// W tym momencie obiekt zawiera informacje o wszystkich podstawowych możliwościach przeglądarki
//  dostępnych dla bieżącego użytkownika. Informacje te można wyświetlić na ekranie.
%>

Używana przeglądarka: <%= browser.getBrowser() %> <%= browser.getFullversion() %><P>

Platforma: <%= browser.getPlatform() %><P>
Główny numer wersji przeglądarki: <%= browser.getMajorver() %><P>
Pomocniczy numer wersji przeglądarki: <%= browser.getMinorver() %><P>
Przeglądarka kontenera: <%= browser.getContainerBrowser() %><P>
Wersja kontenera: <%= browser.getContainerVersion() %><P>
Pełna wersja kontenera: <%= browser.getContainerFullversion() %><P>
Obsługa AJAX? <%= browser.getXMLHttpRequest() %><P>
Obsługa kontrolek ActiveX? <%= browser.getActiveXControls() %><P>
Wersja pliku danych przeglądarki: <%= browser.getBDDVersion() %>, dated: <%= browser.
getBDDDate() %><P>
Używana wersja BrowserHawk: <%= BrowserHawk.getVersion() %><P>
```

W przypadku rozszerzonych właściwości — takich, które mogą zmieniać się z każdą sesją — można wykorzystać następujący kod:

```
<%
  // Najpierw tworzymy obiekt ExtendedOptions. Ten obiekt służy do
  // ustawiania różnych właściwości i opcji, na przykład wybierania
  //  rozszerzonych właściwości do przetestowania oraz powiązanych z nimi parametrów testowania.
  ExtendedOptions options = new ExtendedOptions();

  //  Następnie informujemy program BrowserHawk o tym, jakie testy mają być przeprowadzone
  // dla przeglądarki. Jeśli istnieją inne ustawienia, które chcielibyśmy dodatkowo sprawdzić, możemy
  // dodać je do tej listy. Więcej informacji można znaleźć w opisie klasy ExtendedOptions
  // w dokumentacji programu BrowserHawk.
 %>
options.addProperties("PersistentCookies, SessionCookies, JavaScriptEnabled, Width,
Height, WidthAvail, HeightAvail, Plugin_Flash, Broadband");

Czy włączono pliki cookie sesji? <%= extBrowser.getSessionCookies() %><p>
Czy włączono trwałe pliki cookie? <%= extBrowser.getPersistentCookies() %><p>
Czy włączono JavaScript? <%= extBrowser.getJavaScriptEnabled() %><p>
Rozdzielczość ekranu: <%= extBrowser.getWidth() %> x <%= extBrowser.getHeight() %><p>
Dostępny rozmiar okna przeglądarki: <%= extBrowser.getWidthAvail() %> x <%=
extBrowser.
getHeightAvail() %><p>
Zainstalowana wersja wtyczki Flash: <%= extBrowser.getPluginFlash() %><p>
Połączenie szerokopasmowe? <%= extBrowser.getBroadband() %><p>
```

Można również zbuforować wyniki i uzyskać bardziej szczegółowe dane dotyczące szybkości połączenia związane z użytkownikiem, numerami wersji i innymi możliwościami. Po wykryciu możliwości przeglądarki można skorzystać z tych zmiennych w celu wygenerowania warunkowej treści.

Wykrywanie możliwości przeglądarki za pomocą XSSI

Za pomocą technologii warunkowego włączania kodu po stronie serwera (XSSI) można stworzyć zmienne środowiskowe, które bardzo przypominają wykrywanie możliwości przeglądarek za pomocą JavaScript. Na przykład, poniższy typowy filtr w JavaScript:

```
IS_IE = (document.all) ? true : false;
IS_MAC = (navigator.appVersion.indexOf(" Mac") != -1);
IS_OPERA = (navigator.userAgent.indexOf(" Opera") != -1);
IS_OPERAMAC = IS_OPERA && IS_MAC;
```

można zastąpić odpowiednikiem XSSI w następującej postaci:

```
<!--#if expr="$(HTTP_USER_AGENT) = /MSIE [4-9]//" -->
    <!--#set var="isIE" value="true" -->
<!--#endif -->
<!--#if expr="$(HTTP_USER_AGENT) = /Mac/" -->
    <!--#set var="isMAC " value="true" -->
<!--#endif -->
<!--#if expr="$(HTTP_USER_AGENT) = /Opera/" -->
    <!--#set var="isOPERA" value="true" -->
<!--#endif -->
<!--#if expr="(${isOPERA} && ${isMAC})/" -->
    <!--#set var="isOPERAMAC" value="true" -->
<!--#endif -->
```

Następnie można skorzystać z tak utworzonych zmiennych XSSI w celu warunkowego załadowania kodu, bez konieczności korzystania z JavaScript.

```
<!--#if expr="${isIE}" -->
    ie.js
<!--#elif expr="${isOPERAMAC}" -->
    operamac.js
<!--#elif expr="${isOPERA}" -->
    opera.js
...
<!--#endif -->
```

Szybszy sposób ustawienia zmiennych środowiskowych na serwerze to skonfigurowanie pliku *httpd.conf* za pomocą instrukcji `BrowserMatchNoCase`. Na przykład:

```
BrowserMatchNoCase "MSIE [4-9]" isIE
BrowserMatchNoCase Mac isMAC
BrowserMatchNoCase Opera isOPERA
```

Krok 6: optymalizacja kodu JavaScript pod kątem poprawy szybkości działania i rozmiaru plików

Po zastąpieniu jak największej ilości kodu JavaScript arkuszami CSS oraz technologiami działającymi po stronie serwera należy zoptymalizować pozostały kod JavaScript w celu zminimalizowania rozmiaru pliku. Do zmniejszenia liczby bajtów można wykorzystywać okrojone nazwy obiektów, zmiennych i funkcji. Aby zautomatyzować ten proces, warto wykorzystać odpowiednie narzędzia, na przykład w3compiler — program optymalizujący kod JavaScript pod kątem długości nazw i spacji.

Oprócz minimalizowania rozmiarów kodu JavaScript, często możliwe okazuje się także zapisanie procedur wykonujących te same czynności za pomocą mniejszej ilości kodu (patrz *http://www.refactoring.com*). Należy pamiętać, aby najpierw mierzyć, a potem optymalizować. Do lokalizacji wydajnościowych „wąskich gardeł" w kodzie JavaScript można zastosować narzędzia do profilowania kodu. Profile kodu JavaScript można tworzyć za pomocą programu Venkman JavaScript Debugger dostępnego dla przeglądarki Mozilla (*http://www.mozilla.org/projects/venkman/*).

Optymalizacja pętli polegająca na ich upraszczaniu pozwala uzyskać kilka cykli i dzięki temu przyspieszyć działanie kodu JavaScript. Często się zdarza, że funkcje wbudowane działają szybciej od kodu pisanego ręcznie. Porady dotyczące sposobów zwiększenia szybkości działania kodu JavaScript oraz zmniejszania jego objętości można znaleźć w rozdziale 8. Na koniec warto zadbać o scalenie zewnętrznych plików JavaScript i ich kompresję. Dzięki temu można dodatkowo zaoszczędzić na liczbie żądań HTTP oraz wymaganym paśmie.

Krok 7: konwersja układu bazującego na tabelach na CSS

Dzięki wykorzystaniu CSS do zdefiniowania układu strony można znacząco „odchudzić" kod strony. Zazwyczaj objętość jest niższa o 25% do 50%[17]. Najpierw należy przyjrzeć się układowi stron, by stwierdzić, czy uda się skorzystać z list CSS oraz pozycjonowanych elementów `div` w celu zasymulowania efektów zazwyczaj uzyskiwanych za pomocą tabel. Następnie — sprowadzić treść stron do kodu strukturalnego i stworzyć go z wykorzystaniem od początku stylów CSS i pozycjonowania. Po stworzeniu kodu w tej postaci należy przetestować nowy układ w różnych przeglądarkach. W celu szybkiego przetestowania nowego układu bazującego na CSS w różnych przeglądarkach zalecamy skorzystanie z programu BrowserCam (*http://www.browsercam.com*) — patrz rysunek 6.11.

Rysunek 6.11. Narzędzie BrowserCam renderuje strony WWW w różnych przeglądarkach

[17] Zgodnie z książką Jeffreya Zeldmana *Designing with Web Standards* (New Riders), konwersja stron na układ CSS zazwyczaj przyczynia się do obniżenia objętości plików XHTML oraz ogólnej objętości kodu serwisu o 25% do 50%. W konwersjach, które przeprowadziliśmy, uzyskaliśmy podobne wyniki.

Układ stron bazujący na CSS

Arkusze CSS można wykorzystać w celu określenia pozycji wszystkich elementów strony lub do sformatowania mniejszych fragmentów stron WWW. W wielu witrynach wykorzystuje się tabele do formatowania stron w miejscach, w których bardziej efektywne byłoby zastosowanie CSS. Za pomocą ramek pływających i marginesów stosowanych w odniesieniu do elementów `div`, można uzyskać efekt wielokolumnowych układów bazujących na CSS (*http://alistapart.com/topics/code/css/*). Przy użyciu list zarządzanych za pomocą CSS (w odróżnieniu od często stosowanych list zarządzanych za pomocą JavaScript) można stworzyć złożone hierarchiczne menu. Przykłady ich tworzenia można znaleźć w książce Erica Meyera *CSS według Erica Meyera. Kolejna odsłona* (Helion, 2005). Dzięki CSS można również stworzyć proste efekty rollover zarówno z wykorzystaniem grafiki, jak i bez niej. Przykłady efektów rollover implementowanych za pomocą CSS oraz konwersji menu można znaleźć w rozdziale 7.

Krok 8: zastąpienie stylów wierszowych regułami CSS

Zastąpienie układu bazującego na tabelach układem CSS przyczynia się do oszczędności pasma oraz minimalizuje problemy związane z utrzymaniem serwisu. Zredukowanie kodu serwisu do samej struktury oraz zastąpienie stylów wierszowych regułami CSS pozwala na pełną optymalizację kodu HTML.

Na style wierszowe składają się niezalecany znacznik `font`, bloki wierszowe `style` oraz spacje nierozdzielające. Stosowanie stylów wierszowych następującej postaci:

```
<p style="font-size:12px;color:black;">Tekst zakodowany "na sztywno".</p>
<p style="font-size:12px;color:black;">Przykład stylu wierszowego</p>
```

powoduje wzrost objętości kodu oraz utrudnia wprowadzanie zmian w stylistyce serwisu. Bardziej efektywnym sposobem nadawania stylów jest utworzenie reguł CSS następujących postaci:

```
<style type="text/css">
p{font-size:12px;color:#000;}
</style></head></body>
<p>Tekst niezależny od stylu</p>
<p>Łatwe i wygodne.</p>
```

Zastąpienie stylów wierszowych, znaczników `font` i spacji nierozdzielających regułami CSS może przyczynić się do znacznej redukcji objętości kodu HTML (nawet o 15% do 20%). Uzyskane oszczędności zależą od częstości stosowania stylów wbudowanych. Najważniejszym celem takiego porządkowania kodu jest zaplanowanie adresowania elementów treści za pomocą CSS z wykorzystaniem elementów architektury CSS, o których opowiemy w rozdziale 7.

W architekturze CSS wykorzystywane są strukturalne znaczniki HTML (`p`, `ul`, `dt` itp.) oraz kontenery oznaczone etykietami (`#treść`, `#nawigacja`, `#stopka`). Dzięki nim można łatwo adresować nieprzylegające do siebie elementy treści za pomocą selektorów potomka i typu. Po stworzeniu architektury CSS adresowanie podobnej treści sprowadza się do stworzenia reguł CSS, wykorzystania selektorów w celu nadania stylów elementom tego samego typu oraz do wprowadzenia deklaracji mających na celu zastosowanie stylów. Więcej informacji na temat optymalizacji kodu HTML za pomocą CSS, a także zmniejszania objętości arkuszy stylów można znaleźć w rozdziale 7.

Krok 9: zminimalizowanie czasu wyświetlania strony

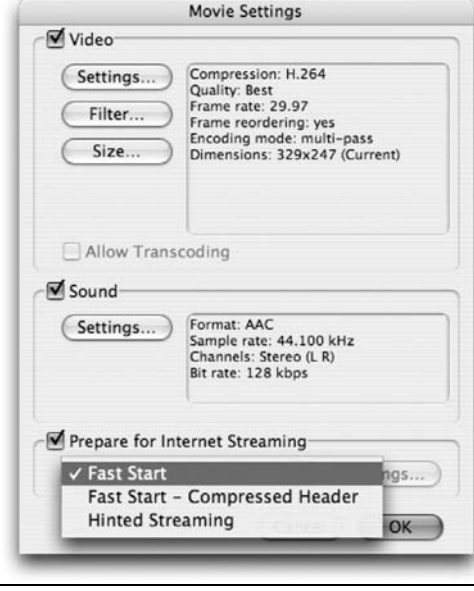
Dzięki szybkiemu załadowaniu użytecznych treści można poprawić postrzeganą szybkość ładowania stron WWW.

Na przykład, na stronie głównej serwisu *Weather Underground*, w lewym górnym rogu strony bardzo szybko wyświetla się formularz wyszukiwania prognozy pogody (patrz rysunek 6.12). W odróżnieniu od innych serwisów pogodowych, w których najpierw ładują się różne elementy, w serwisie *Weather Underground* nadano priorytet najważniejszej części strony — formularzowi pozwalającemu na szybkie wyszukanie prognozy pogody dla wskazanego regionu.

Rysunek 6.12. W serwisie Weather Underground najpierw ładowana jest użyteczna treść

Aby zapewnić szybkie ładowanie użytecznej treści (tzn. takiej, po której użytkownicy mogą się poruszać), można podzielić na warstwy użyte tabele lub elementy `div`.

Techniki szybkiego startu można również wykorzystywać w odniesieniu do treści multimedialnych. W prezentacjach we Flashu najpierw może załadować się jeden plik, podczas gdy kolejne ładują się w tle. W przypadku filmów można szybko załadować statyczny obraz zastępczy lub podgląd, tak by użytkownicy mieli się czym zająć w czasie ładowania zasadniczej części filmu. Program QuickTime Pro umożliwia skonfigurowanie filmu w taki sposób, by jego odtwarzanie rozpoczęło się przed całkowitym ściągnięciem. Mechanizm ten włącza się za pomocą opcji *Fast Start* (patrz rysunek 6.13).



Rysunek 6.13. Opcja szybkiego startu w programie QuickTime Pro

Optymalizacja stron WWW nie polega tylko na poprawie szybkości wyświetlania, ale także na zarządzaniu wrażeniami odnoszonymi przez użytkowników.

Krok 10: rozsądne ładowanie kodu JavaScript

Zewnętrzne skrypty wywoływane wewnątrz elementu `head` stron WWW są szczególnie szkodliwe, ponieważ powodują opóźnienia w wyświetlaniu zasadniczej treści strony. Jeśli opóźnienia występują przed wyświetleniem zasadniczej treści strony, istnieje większe prawdopodobieństwo, że użytkownicy porzucą serwis, zanim całkowicie wyświetli się jego pierwsza strona. Z badań interakcji ludzi z komputerami (ang. *Human-Computer Interaction* — HCI) wynika, że opóźnienia przed przeglądaniem stron są mniej frustrujące od tych, które występują po załadowaniu strony[18].

Opóźnienia występujące po załadowaniu stron stanowią powszechny problem serwisów korzystających z technologii Ajax. Jeśli kod Ajaksa jest nieodpowiednio napisany, może znacznie utrudnić obsługę serwisu — zwłaszcza w przypadku użytkowników łącz wąskopasmowych. Nawet w przypadku zastosowania kompresji HTTP opóźnienia spowodowane koniecznością pobierania wielu osobnych plików mogą spowodować uciążliwe przestoje. W technologii Ajax stosuje się również odpytywanie za pośrednictwem obiektu `XMLHttpRequest`. Komunikacja bazująca na tym obiekcie może być źródłem problemów z wydajnością. Zajmiemy się nimi w rozdziale 8.

Niebezpieczeństwa związane z zewnętrznymi widżetami

Webmasterzy często korzystają z usług sieciowych udostępnianych za pośrednictwem różnego rodzaju widżetów. Są to mechanizmy osadzania w serwisach różnych usług — od *Google AdWords*, poprzez fotografie Flickr i mikroblogi Twitter, po listy odtwarzania iTunes. Problem z zewnętrznymi widżetami polega na tym, że mogą one powodować wielosekundowe opóźnienia w wyświetlaniu stron WWW oraz powodować przestoje o zmiennym czasie trwania. Widżety są zwykle używane razem z fragmentami zewnętrznego kodu JavaScript, a ich wydajność zależy od czasu odpowiedzi zewnętrznego serwera udostępniającego usługę. Większość dostawców usług sieciowych nie dysponuje farmami serwerów do obsługi danych, a tym samym nie jest w stanie odpowiadać na żądania równie wydajnie, jak Google. Z naszych doświadczeń wynika, że mechanizmy zewnętrznych sond, kod śledzenia blogów Technorati, a nawet usługa Google Analytics w początkowym okresie jej działania, powodowały zawieszanie się przeglądarek i opóźnienia w ładowaniu stron WWW. Pozbycie się tego rodzaju widżetów lub przeniesienie ich na koniec kodu pozwala zminimalizować powodowane przez nie wrażenie opóźniania.

Zastosowanie technologii WEDJE. Istnieje jednak lepszy sposób. Dzięki wykorzystaniu technologii WEDJE (ang. *Widget Enabled DOM JavaScript Embedding*) można przepisać kod osadzania widżetów w taki sposób, aby kod JavaScript działał w sposób asynchroniczny. Technologia WEDJE umożliwia odroczone ładowanie elementów dzięki wykorzystaniu obiektowego modelu dokumentów (ang. *Document Object Model* — DOM). Najpierw dołączany jest element `div`, potem tworzony jest element `script`, a następnie element `script` zostaje dołączony do elementu `div` — wszystko to realizowane za pomocą JavaScript. Oto przykład zastosowania tej techniki:

[18] Dellaert B. i B. Kahn. 1999. *How Tolerable is Delay? Consumers' Evaluations of Internet Web Sites after Waiting*. „Journal of Interactive Marketing" 13 (1): 41 – 54.

```
<script type="text/javascript">      // poniżej utworzono element div
(function(){document.write('<div id="wedje_div_example">Ładowanie widżetu...<\/div>');
s=document.createElement('script'); // utworzenie elementu script
s.type="text/javascript";           // przypisanie skryptu do elementu script
s.src="http://www.example.com/scripts/widget.js";
                                    // przypisanie skryptu s do elementu div
setTimeout("document.getElementById('wedje_div_example').appendChild(s)",1);})()
</script>
```

Połączenie wymienionych elementów w taki sposób powoduje, że proces ładowania i uruchamiania dołączonego kodu JavaScript zostaje rozdzielony. Dzięki temu widżet uruchamia się w sposób asynchroniczny! Oto zawartość zewnętrznego pliku JavaScript *widget.js*, którego zadaniem jest pobranie elementu `div` utworzonego wcześniej i załadowanie obrazu:

```
document.getElementById('wedje_div_example').innerHTML+='<img src="http://www.
➥example.com/images/example.gif" width="60" height="60" />';
```

Można również skorzystać z ramek pływających do ładowania reklam. Zastosowanie ramek pływających może jednak uniemożliwić korzystanie z mechanizmu wykrywania kontekstu w przypadku reklam kontekstowych, dlatego należy zachować ostrożność podczas posługiwania się nimi. Więcej informacji na temat techniki WEDJE można znaleźć we wpisie na blogu Mike'a Davidsona, pod adresem *http://www.mikeindustries.com/blog/archive/2007/06/widget-deployment-with-wedje* (należy zwrócić uwagę, że wykorzystanie tej techniki z przeglądarką Internet Explorer w wersji 6 sprawia problemy; można jednak zastosować komentarze warunkowe w celu wykorzystania jej tylko z przeglądarkami Internet Explorer w wersji 7 i późniejszych).

Więcej informacji na temat opóźniania ładowania zewnętrznych skryptów można znaleźć w rozdziale 9.

Podsumowanie

Optymalizacja stron WWW ma na celu ich uproszczenie, tak by ściągały się i ładowały szybciej. Poprawa wydajności serwisów WWW obniża współczynnik porzucania serwisu przez użytkowników oraz koszty za wykorzystane pasmo, a jednocześnie podwyższa współczynniki konwersji i zyski. W niniejszym rozdziale podpowiedzieliśmy, w jaki sposób zminimalizować liczbę żądań HTTP, zoptymalizować treści graficzne i multimedialne, zastąpić mechanizmy wykrywania możliwości przeglądarki działające po stronie klienta analogicznymi mechanizmami po stronie serwera oraz jak rozsądnie ładować kod JavaScript.

W celu zmniejszenia kosztów związanych z dużą liczbą obiektów na stronie, które stanowią przyczynę większości opóźnień w ładowaniu stron WWW, należy zmniejszyć liczbę obiektów wywoływanych na stronach WWW. Trzeba również zadbać o to, aby obrazy, zarówno statyczne, jak i dynamiczne, były zapisane w plikach o jak najmniejszej objętości. Należy zminimalizować rozmiar elementu `head` dokumentów HTML i podzielić kod serwisu na warstwy w taki sposób, by użyteczna treść wyświetlała się szybciej. Dzięki temu podniesiemy pozycję serwisu w rankingach wyszukiwarek. Wreszcie, aby umożliwić progresywne wyświetlanie, należy przenieść kod CSS na początek stron, natomiast skrypty — na ich koniec.