JavaScript dla każdego. Wydanie IV

Autor: Michael Moncur
Tłumaczenie: Adam Jarczyk
ISBN: 83-246-0766-8
Tytuł oryginału: Sams Teach Yourself JavaScript in 24 Hours (4th Edition)
Format: B5, stron: 456

Nadaj nową jakość swoim stronom internetowym

- Poznaj składnię i możliwości języka JavaScript
- Naucz się dynamicznie manipulować stronami internetowymi
- Twórz bardziej efektowne i interaktywne witryny

Znasz już język HTML, umiesz tworzyć proste strony internetowe i chcesz się nauczyć czegoś nowego? JavaScript to doskonały wybór. Język ten pozwala tchnąć życie w statyczne strony, dodać do nich ciekawe efekty oraz ułatwić użytkownikom interakcję z witryną. Duże możliwości połączone z łatwością nauki sprawiają, że jest to jeden z najprostszych sposobów na poprawę jakości Twoich stron.

Książka „JavaScript dla każdego. Wydanie IV" zawiera 24 krótkie lekcje, dzięki którym szybko poznasz składnię tego języka i nauczysz się używać go do tworzenia zaawansowanych stron internetowych. Dowiesz się, czym jest model DOM oraz jak za jego pomocą dynamicznie manipulować zawartością strony. Przeczytasz o technikach tworzenia internetowych aplikacji multimedialnych i możliwościach technologii AJAX. Poszczególne lekcje zawierają szczegółowe instrukcje opisujące typowe operacje języka JavaScript, co pozwoli Ci samodzielnie wykonać wszystkie przykłady. Dodatkowo, na końcu każdego rozdziału znajdują się pytania i ćwiczenia, które pomogą Ci sprawdzić i utrwalić nabytą wiedzę.

- Składnia języka JavaScript
- Pisanie skryptów i umieszczanie ich na stronach
- Używanie modelu DOM do manipulowania elementami strony
- Pobieranie danych od użytkownika
- Stosowanie stylów przy użyciu arkuszy CSS
- Wprowadzenie do technologii AJAX
- Obsługa grafiki, animacji i dźwięków
- Tworzenie efektownych menu rozwijanych
- Pisanie gier internetowych

Doskonała książka dla każdego ambitnego webmastera!

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

Spis treści

Rozdział 4.

Podstawy DOM (Document Object Model)

Rozdział omawia następujące tematy:

- Jak korzystać z różnych obiektów DOM.
- Jak pracować z oknami, używając obiektów `window`.
- Jak pracować z dokumentami, używając obiektów `document`.
- Jak używać obiektów dla łączy i kotwic.
- Jak za pomocą obiektu `location` pracować z adresami URL
- Jak utworzyć oparte na JavaScripcie przyciski *Wstecz* i *Dalej*.

Dotarliśmy do końca części I. Niniejszy rozdział przedstawi Czytelnikowi jedno z najważniejszych narzędzi, których będzie używać z JavaScriptem: obiektowy model dokumentu (DOM — ang. *Document Object Model*), który pozwala na manipulowanie przez skrypty stronami WWW, oknami i dokumentami.

Bez DOM JavaScript byłby po prostu kolejnym językiem skryptowym. Dzięki DOM staje się potężnym narzędziem tworzenia dynamicznych stron WWW. W rozdziale przedstawimy ideę DOM i kilka najczęściej używanych obiektów.

Obiektowy model dokumentu — wprowadzenie

JavaScript ma nad HTML-em tę przewagę, że skrypty mogą manipulować dokumentem WWW i jego zawartością. Skrypt może załadować do przeglądarki nową stronę, zmieniać elementy okna przeglądarki i dokumentu, otwierać nowe okna, a nawet dynamicznie modyfikować tekst na stronie.

Do pracy z przeglądarką i dokumentami JavaScript używa hierarchii obiektów nadrzędnych i potomnych, zwanej Document Object Model (DOM). Obiekty te są zorganizowane w strukturę przypominającą drzewo i reprezentują całą treść i wszystkie składniki dokumentu WWW.

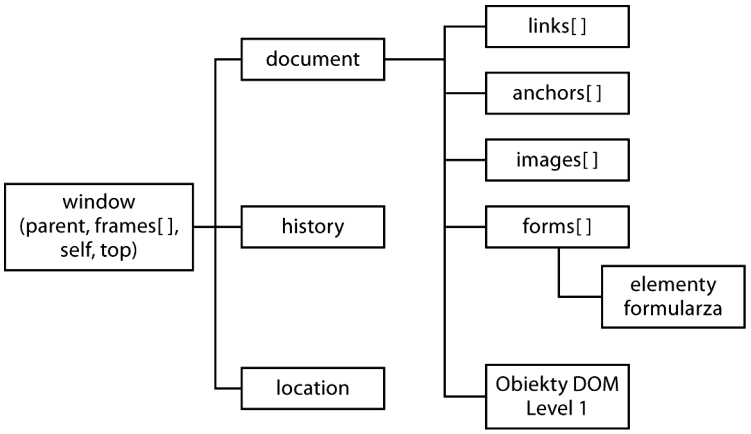
DOM nie należy do języka JavaScript — jest interfejsem programowym aplikacji (API) wbudowanym w przeglądarkę WWW. Wprawdzie DOM najczęściej używany jest z JavaScriptem, lecz może być też wykorzystywany przez inne języki, np. VBScript i Javę.

Obiekty w DOM mają **właściwości** — zmienne, które opisują stronę WWW lub dokument, oraz **metody** — funkcje, które pozwalają manipulować elementami strony WWW.

Aby odwołać się do obiektu, używamy nazwy obiektu nadrzędnego, po której następuje nazwa (lub nazwy obiektu potomnego oddzielone kropkami). Na przykład JavaScript przechowuje obiekty reprezentujące obrazy w dokumencie jako obiekty potomne obiektu `document`. Poniższy zapis wskazuje obiekt `image9` będący obiektem potomnym obiektu `document`, który z kolei jest obiektem potomnym obiektu `window`:

```
window.document.image9
```

`window` jest obiektem nadrzędnym dla wszystkich obiektów, którymi będziemy zajmować się w tym rozdziale. Rysunek 4.1 ilustruje ten wycinek hierarchii DOM i kilka obiektów znajdujących się w nim.



Rysunek 4.1.
Hierarchia obiektów DOM

Powyższy rysunek przedstawia tylko podstawowe obiekty przeglądarki, które zostaną omówione w niniejszym rozdziale. Są one jedynie niewielką częścią DOM. Więcej informacji na ten temat zawiera część III, „Więcej na temat DOM".

Historia DOM

Od chwili wprowadzenia JavaScriptu 1.0 w programie Netscape 2.0 przeglądarki WWW zawierają obiekty, które reprezentują elementy dokumentu WWW i inne funkcje przeglądarki. Nigdy jednak nie istniał prawdziwy standard. Wprawdzie Netscape i Internet Explorer zawierały wiele identycznych obiektów, lecz nie było żadnej gwarancji, że te same obiekty będą zachowywać się tak samo w obu tych programach, a co dopiero w mniej popularnych przeglądarkach WWW.

Nadal, niestety, występują różnice pomiędzy przeglądarkami — lecz jest i dobra wiadomość. Od chwili wydania wersji Netscape 3.0 i Internet Explorer 4.0 wszystkie podstawowe obiekty (omawiane w niniejszym rozdziale) są przez obie przeglądarki obsługiwane zasadniczo tak samo. W nowszych wersjach przeglądarek obsługiwane są znacznie bardziej zaawansowane modele DOM.

Poziomy DOM

Organizacja W3C (ang. *World Wide Web Consortium*) opracowała zalecenia poziomu 1. modelu (ang. *DOM Level 1*). Jest to standard definiujący nie tylko obiekty podstawowe, lecz cały zestaw obiektów, które obejmują wszystkie składniki dokumentu HTML. Standard DOM poziom 2. również został opublikowany, a poziom 3. jest w trakcie tworzenia.

Netscape 4 i Internet Explorer 4 udostępniały własne obiektowe modele dokumentu, które dawały większą kontrolę nad dokumentem, lecz nie były znormalizowane. Na szczęście, zaczynając od wersji Internet Explorer 5 i Netscape 6, oba programy obsługują DOM W3C, więc możemy obsługiwać obie przeglądarki poprzez prosty, zgodny ze standardami kod. Wszystkie dzisiejsze przeglądarki obsługują DOM W3C.

Podstawowa hierarchia obiektów, opisana w niniejszym rozdziale, jest nieformalnie nazywana poziomem 0 DOM, a obiekty te należą do standardu DOM Level 1. W dalszej części książki pokażemy, jak za pomocą DOM W3C pracować z dowolną częścią dokumentu WWW.

DOM W3C pozwala modyfikować stronę w czasie rzeczywistym po tym, jak zostanie załadowana. Jak to zrobić, pokażemy w części III książki.

Korzystanie z obiektów window

Na szczycie hierarchii obiektów przeglądarki znajduje się obiekt `window`, który reprezentuje okno przeglądarki. Użyliśmy już przynajmniej jednej metody obiektu `window`: metody `window.alert()` lub po prostu `alert()`, która wyświetla komunikat w okienku komunikatu.

W jednej chwili może istnieć kilka obiektów window, z których każdy będzie reprezentować otwarte okno przeglądarki. Ramki również są reprezentowane przez obiekty window. O oknach i ramkach powiemy więcej w rozdziale 10., „Okna i ramki".

Warstwy, które pozwalają wstawiać, modyfikować i pozycjonować dynamiczną treść w dokumencie WWW, również są podobne do obiektów window. Zostały opisane w rozdziale 13., „Korzystanie z DOM W3C".

Praca z dokumentami WWW

Obiekt document reprezentuje dokument (stronę) WWW. Dokumenty WWW są wyświetlane w oknach przeglądarek, więc nikogo nie powinno zaskoczyć, że document jest obiektem potomnym obiektu window.

Obiekt window zawsze reprezentuje bieżące okno (czyli to, które zawiera skrypt), więc za pomocą window.document można odwołać się do bieżącego dokumentu. Możemy też po prostu odwołać się do obiektu document; wówczas automatycznie zostanie przyjęte bieżące okno.

Używaliśmy już metody document.write do wyświetlenia tekstu w dokumencie WWW. Przykłady w poprzednich rozdziałach obejmowały tylko jedno okno i jeden dokument, więc nie było trzeba używać window.document.write — lecz ta dłuższa składnia zadziałałaby równie dobrze.

Gdy używamy więcej niż jednego okna lub ramki, możemy mieć do czynienia z kilkoma obiektami window, z których każdy będzie miał własny obiekt document. Aby wykorzystać jeden z tych obiektów document, należy użyć nazwy okna i nazwy dokumentu.

W następnych punktach omówimy kilka właściwości i metod obiektu document, które przydadzą się przy pisaniu skryptów.

Zdobywanie informacji o dokumencie

Kilka właściwości obiektu document zawiera ogólne wiadomości o bieżącym dokumencie:

- document.URL podaje URL dokumentu jako proste pole tekstowe Tej właściwości nie można zmienić. Aby wysłać użytkownika pod inny adres, należy użyć obiektu window.location omówionego w dalszej części rozdziału.
- document.title podaje tytuł bieżącej strony zdefiniowany przez znacznik HTML <title>.
- document.referrer jest adresem URL strony, którą użytkownik wyświetlał przed bieżącą — zwykle była to strona z łączem do strony bieżącej.

- `document.lastModified` jest datą ostatniej modyfikacji dokumentu. Data ta jest wysyłana przez serwer wraz ze stroną WWW.
- `document.bgColor` i `document.fgColor` są kolorami tła i pierwszego planu (tekstu) dokumentu. Odpowiadają atrybutom `BGCOLOR` i `TEXT` znacznika `<body>`.
- `document.linkColor`, `document.alinkColor` i `document.vlinkColor` są kolorami łączy w dokumencie. Odpowiadają atrybutom `LINK`, `ALINK` i `VLINK` znacznika `<body>`.
- `document.cookie` pozwala odczytać lub ustawić cookie dla dokumentu. Informacje o cookies można znaleźć pod adresem *http://www.jsworkshop.com/cookies.html*.

Jak przykład właściwości dokumentu listing 4.1 przedstawia krótki dokument HTML, który za pomocą JavaScriptu wyświetla datę ostatniej modyfikacji.

Listing 4.1. *Wyświetlanie daty ostatniej modyfikacji*

```
<html>
<head>
<title>Dokument testowy</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-2">
</head>
<body>
<p>Ta strona została ostatnio zmodyfikowana:
<script language="JavaScript" type="text/javascript">
document.write(document.lastModified);
</script>
</p>
</body>
</html>
```

Może to informować użytkownika, kiedy strona została ostatnio zmieniona. Używając JavaScriptu, nie musimy pamiętać, by zaktualizować datę po każdej modyfikacji strony (za pomocą skryptu moglibyśmy też zawsze wyświetlać bieżącą datę zamiast daty ostatniej aktualizacji, ale to byłoby oszustwem).

Może się okazać, że właściwość `document.lastModified` nie zadziała poprawnie w stronie WWW lub zwróci błędną wartość. Data jest odbierana z serwera WWW, a niektóre serwery nie utrzymują poprawnie dat modyfikacji.

Zapisywanie tekstu w dokumencie

Najprostsze metody obiektu `document` są zarazem używane najczęściej. W istocie jedną z nich już wykorzystaliśmy. Metoda `document.write` wyświetla tekst jako element strony WWW w oknie dokumentu. Instrukcja ta jest używana, by wyświetlić wyjście skryptu na stronie WWW.

Alternatywna instrukcja `document.writeln` również wyświetla tekst, lecz dołącza na koniec znak końca wiersza (`\n`). Przydaje się to, gdy wyświetlany tekst ma być ostatnim elementem wiersza.

Znak końca wiersza jest przez przeglądarki WWW wyświetlany jako spacja, poza jednym wyjątkiem — wewnątrz kontenera `<pre>`. Aby wstawić faktyczny koniec wiersza, należy użyć znacznika `<br>`.

Wymienionych metod można używać jedynie w treści strony WWW, więc są wykonywane podczas ładowania strony. Nie można ich użyć, aby dodać tekst do załadowanej już strony, bez konieczności przeładowania dokumentu.

W nowszych przeglądarkach można bezpośrednio modyfikować tekst na stronie WWW za pomocą funkcji nowego DOM. Techniki te zostały omówione w rozdziale 14.

Metody `document.write` można używać pomiędzy znacznikami `<script>` w treści dokumentu HTML. Może też być używana w funkcji, pod warunkiem że w treści dokumentu zawrzemy wywołanie tej funkcji.

Stosowanie łączy i kotwic

Kolejnym obiektem potomnym obiektu `document` jest obiekt `link`. Dokument może zawierać wiele obiektów `link`, z których każdy zawiera informację o łączu wskazującym inny adres lub kotwicę.

Kotwice oznaczają nazwane miejsca w dokumencie HTML, do których można bezpośrednio przeskoczyć. Definiowane są za pomocą znacznika następująco: `<a name=➥"part2">`. Po zdefiniowaniu kotwicy można ją wskazać poprzez znacznik `<a href=➥"#part2">`.

Obiekty `link` są dostępne w tablicy `links`. Każdy element tablicy jest jednym z obiektów `link` na bieżącej stronie. Właściwość tablicy `document.links.length` wskazuje liczbę łączy na stronie.

Każdy obiekt `link` (inaczej element tablicy `links`) ma listę właściwości definiujących URL. Właściwość `href` zawiera kompletny URL, a inne właściwości definiują elementy tego adresu. Są to właściwości takie same jak obiektu `location`, który zostanie omówiony w dalszej części rozdziału.

Do właściwości można się odwołać, podając numer łącza i nazwę właściwości. Na przykład poniższa instrukcja przypisuje cały URL pierwszego łącza do zmiennej link1:

```
link1 = links[0].href;
```

Obiekty anchor są również potomne względem obiektu document. Każdy obiekt anchor reprezentuje kotwicę w bieżącym dokumencie — zdefiniowaną lokalizację, do której można przejść bezpośrednio.

Podobnie jak łącza kotwice są dostępne w tablicy (o nazwie anchors). Każdy element tej tablicy jest obiektem anchor. Właściwość document.anchors.length podaje liczbę elementów w tablicy anchors.

Dostęp do historii przeglądarki

Obiekt history jest kolejnym obiektem potomnym (właściwością) obiektu window. Mieści informacje o adresach URL odwiedzonych przed i po bieżącym i zawiera metody, które pozwalają przejść do wcześniejszej lub następnej lokalizacji.

Obiekt history ma dostępną jedną właściwość:

- history.length zawiera informację o długości listy historii — inaczej mówiąc liczbę różnych lokalizacji odwiedzonych przez użytkownika.

Obiekt history ma właściwości current, previous i next, w których przechowywane są URL dokumentów z listy historii. Jednakże z uwagi na bezpieczeństwo i prywatność obiekty te nie są standardowo dostępne w dzisiejszych przeglądarkach.

Obiekt history ma trzy metody, którymi możemy się posłużyć do przemieszczania się po liście adresów w historii:

- history.go() otwiera URL z listy historii. Aby użyć tej metody, należy podać w nawiasach liczbę dodatnią lub ujemną. Na przykład history.go(-2) jest odpowiednikiem dwukrotnego kliknięcia przycisku *Wstecz*.
- history.back() ładuje poprzedni URL z listy historii — jest odpowiednikiem naciśnięcia przycisku *Wstecz*.
- history.forward() ładuje następny URL z listy historii, jeśli taki adres jest dostępny. Stanowi odpowiednik naciśnięcia przycisku *Dalej*.

Metody te wypróbujemy w sekcji „Zrób to sam” na końcu niniejszego rozdziału.

Korzystanie z obiektu location

Trzecim obiektem potomnym obiektu window jest location. Obiekt ten przechowuje informacje o bieżącym adresie URL otwartym w oknie. Na przykład poniższa instrukcja wczytuje URL do bieżącego okna:

```
window.location.href="http://www.starlingtech.com"
```

Właściwość `href` użyta w tej instrukcji zawiera kompletny URL bieżącego adresu okna. Za pomocą innych właściwości obiektu `location` możemy też uzyskać dostęp do różnych fragmentów URL. Weźmy na przykład poniższy adres URL:

```
http://www.jsworkshop.com:80/test.cgi?lines=1#anchor
```

Poszczególne elementy tego adresu reprezentują następujące właściwości:

- `location.protocol` wskazuje protokół (w tym przykładzie `http:`).
- `location.hostname` oznacza nazwę hosta w URL (w tym przykładzie `www.jsworkshop.com`).
- `location.port` oznacza numer portu (w tym przykładzie `80`).
- `location.pathname` wskazuje nazwę pliku ze ścieżką (w tym przykładzie `test.cgi`).
- `location.search` wskazuje zapytanie (w tym przykładzie `lines=1`), jeśli adres URL je zawiera. Zapytania takie najczęściej wykorzystywane są przez skrypty CGI.
- `location.hash` jest nazwą kotwicy (w tym przykładzie `#anchor`), jeśli adres URL ją zawiera.

Obiekt `link`, przedstawiony wcześniej, również zawiera listę właściwości dających dostęp do elementów adresu URL.

Wprawdzie właściwość `location.href` zwykle zawiera ten sam URL co właściwość `document.URL`, omówiona wcześniej, lecz właściwości `document.URL` nie można modyfikować. Aby załadować nową stronę, należy zawsze posługiwać się `location.href`.

Obiekt `location` ma dwie metody:

- `location.reload()` przeładowuje bieżący dokument. Jest odpowiednikiem przycisku *Odśwież* na pasku narzędzi. Jeśli (opcjonalnie) dodamy parametr `true`, pamięć podręczna przeglądarki będzie ignorowana i odświeżenie dokumentu zostanie wymuszone niezależnie od tego, czy uległ zmianie czy nie.
- `location.replace()` zastępuje bieżącą lokalizację nową. Jest to podobne do ustawiania bezpośrednio właściwości obiektu `location`. Różnica polega na tym, że metoda `replace` nie ma wpływu na historię przeglądarki — inaczej mówiąc, do poprzedniej lokalizacji nie można wrócić za pomocą przycisku *Wstecz*. Przydaje się to do okien tytułowych lub tymczasowych stron, do których powrót byłby bezużyteczny.

▼ Zrób to sam

Tworzenie przycisków Wstecz i Dalej

Za pomocą metod `back` i `forward` obiektu `history` możemy do dokumentu WWW dodać własne przyciski *Wstecz* i *Dalej*. Przeglądarka oczywiście ma już te przyciski, lecz czasem przydaje się udostępnić własne łącza, które będą pełnić to samo zadanie.

Utworzymy teraz skrypt, który wyświetla przyciski *Wstecz* i *Dalej* i za pomocą metod `back` i `forward` pozwala na nawigację w przeglądarce. Oto kod, który utworzy przycisk *Wstecz*:

```
<input type="button"
   onClick="history.back();" value="<-- Wstecz">
```

Znacznik `<input>` definiuje przycisk oznaczony *Wstecz*. Funkcja obsługi zdarzenia `onClick` używa metody `history.back()`, aby powrócić do poprzedniej strony w historii. Kod przycisku *Dalej* jest podobny:

```
<input type="button"
   onClick="history.forward();" value="Dalej -->">
```

Teraz pozostało nam tylko zbudować resztę dokumentu HTML. Listing 4.2 przedstawia kompletny dokument. Po załadowaniu go do przeglądarki możemy odwiedzić inne adresy URL i sprawdzić, czy przyciski działają poprawnie (rysunek 4.2).

Listing 4.2. *Strona WWW z dodanymi za pomocą JavaScriptu przyciskami Wstecz i Dalej*

```
<html>
<head>
<title>Przyciski Wstecz i Dalej</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-2">
</head>
<body>
<h1>Przyciski Wstecz i Dalej</h1>
<p>Ta strona pozwala przechodzić do strony wcześniejszej i późniejszej w historii
przeglądarki.
Poniższe przyciski powinny być odpowiednikiem przycisków <i>Wstecz</i> i
<i>Dalej</i> na pasku narzędzi przeglądarki WWW.</p>
<p>
<input type="button"
   onClick="history.back();" value="<-- Wstecz">
<input type="button"
   onClick="history.forward();" value="Dalej -->">
</p>
</body>
</html>
```

▲

Podsumowanie

W niniejszym rozdziale poznaliśmy Document Object Model — wykorzystywaną przez JavaScript hierarchię obiektów strony WWW. Pokazaliśmy, jak używać obiektu `document` do pracy z dokumentami i jak za pomocą obiektów `history` i `location` kontrolować URL wyświetlany w przeglądarce.

Czytelnik powinien teraz dysponować podstawową wiedzą o DOM i kilku obiektach z tego modelu — w dalszej części książki obiekty będą opisywane bardziej szczegółowo.

Rysunek 4.2. *Przyciski Wstecz i Dalej w przeglądarce Internet Explorer*

Gratulacje! Dotarliśmy do końca części pierwszej niniejszej książki. W części II powrócimy do nauki języka JavaScript, zaczynając od rozdziału 5., „Zmienne, łańcuchy i tablice".

Pytania i odpowiedzi

P: Mogę używać zapisu `history` i `document` zamiast `window.history` i `window.document`. Czy w innych przypadkach też mogę pominąć obiekt `window`?

O: Tak. Na przykład do wyświetlenia komunikatu może posłużyć `alert` zamiast `window.alert`. Obiekt `window` zawiera bieżący skrypt, więc jest traktowany jako obiekt domyślny. Nie można jednak pomijać nazwy obiektu `window` przy pracy z ramkami, warstwami, więcej niż jednym oknem oraz w funkcji obsługi zdarzenia.

P: Spróbowałem za pomocą metody `document.lastModified` wyświetlić datę modyfikacji mojej strony, lecz została wyświetlona data z roku 1970 (lub inna, o której wiem, że jest niepoprawna). Co się stało?

O: Działanie tej funkcji jest zależne od serwera WWW wysyłającego do przeglądarki datę ostatniej modyfikacji dokumentu. Niektóre serwery nie wykonują tej czynności prawidłowo lub wymagają określonych atrybutów plików, by funkcja działała poprawnie.

P: Czy mogę zmieniać wpisy w historii przeglądarki albo uniemożliwić użytkownikowi korzystanie z przycisków *Wstecz* i *Dalej*?

O: Wpisów w historii nie można zmieniać. Nie można też zapobiec użyciu przycisków *Wstecz* i *Dalej*, lecz można za pomocą `location.replace()` załadować serię stron, które nie pojawią się w historii. Istnieje kilka sztuczek, które uniemożliwiają poprawne działanie przycisku *Wstecz*, lecz nie radzę z nich korzystać — takie właśnie rozwiązania psują reputację JavaScriptu.

Pytania kontrolne

Sprawdź swoją wiedzę na temat JavaScriptu, odpowiadając na poniższe pytania:

1. Który z poniższych obiektów może posłużyć do załadowania nowego URL do okna przeglądarki?

   a) `document.url`

   b) `window.location`

   c) `window.url`

2. Który obiekt zawiera metodę `alert()`?

   a) `window`

   b) `document`

   c) `location`

3. Który z poniższych poziomów DOM opisuje obiekty omówione w niniejszym rozdziale?

   a) DOM Level 0

   b) DOM Level 1

   c) DOM Level 2

Odpowiedzi

1. (b) Do wysłania przeglądarki pod nowy adres URL może posłużyć obiekt `window.location`.
2. (a) Metodę `alert()` zawiera obiekt `window`.
3. (a) Obiekty opisane w niniejszym rozdziale mieszczą się w nieformalnej specyfikacji DOM Level 0.

Ćwiczenia

Aby lepiej zapoznać się z możliwościami JavaScriptu przedstawionymi w niniejszym rozdziale, wykonaj następujące ćwiczenia:

- Zmodyfikuj przykład z listingu 4.2, dodając do przycisków *Wstecz* i *Dalej* przycisk *Odśwież* (przycisk ten powinien wyzwolić metodę `location.reload()`).
- Zmodyfikuj ten sam przykład tak, by wyświetlić aktualną liczbę wpisów w historii.