

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Java Script. Księga eksperta

Autor: Danny Goodman

Tłumaczenie: Adam Żytka

ISBN: 83-7197-179-6

Tytuł oryginału: [JavaScript Bible](#)

Format: B5, stron: 1096

oprawa twarda

Zawiera CD-ROM



Interaktywne, pełne dynamiki strony WWW - to cel, który często stawiasz przed sobą. Marzysz o wykorzystaniu apletów Javy bez przeciążania serwera, o projektowaniu stron w języku DHTML. Jeden z najbardziej znanych i poważanych ekspertów w branży, Danny Goodman, odkryje przed Tobą tajniki i potęgę języka JavaScript. W typowym dla siebie przejrzystym i precyzyjnym stylu autor przedstawia kompletny wykład na temat JavaScriptu, prowadząc nas od zagadnień elementarnych aż do zaawansowanych technik programowania. Dodatkowym atutem książki jest obszerny leksykon obiektów, funkcji i innych elementów języka.

Ta książka to wyczerpujący opis:

- Języka JavaScript
- Tworzenia animowanych przycisków
- Obiektów i innych kluczowych elementów języka, w tym struktur sterujących, funkcji, operatorów
- Wykorzystania JavaScriptu do weryfikacji danych wprowadzonych przez użytkownika
- Użycia apletów Javy na stronach WWW
- Różnic pomiędzy implementacjami języka zrealizowanymi przez firmy Netscape i Microsoft
- Niezależnych od przeglądarki aplikacji DHTML
- Debuggera przeglądarki Netscape



Spis treści

O Autorze	19
Podziękowania	20
Przedmowa	21
Wstęp	25
Układ książki.....	25
Co potrzeba, aby poznać JavaScript	28
Formatowanie i konwencje nazewnictwa	31

Część I Pierwsze kroki w JavaScriptcie

Rozdział 1. Rola JavaScriptu w sieci Web	35
Konkurencja w sieci Web	35
Wyprowadzanie standardów	36
Skrypty CGI	37
Aplikacje pomocnicze, plug-iny i aplety	37
Plug-iny.....	38
Aplety Javy	38
JavaScript – język dla wszystkich.....	39
LiveScript zmienia nazwę na JavaScript	40
Nareszcie razem.....	41
Świat Microsoftu	41
JavaScript – odpowiednie narzędzie do odpowiednich zadań	42
Rozdział 2. Uniwersalność stron a wojna przeglądarek.....	45
Kapsle.....	46
Do schronu	47
Aktualne problemy z kompatybilnością	47
Rozdzielenie języka i obiektów	47
Standardowy rdzeń języka	48
Model obiektów dokumentu	49
Kaskadowe Arkusze Stylów (CSS)	50
Dynamiczny HTML.....	50
Strategia pisania skryptów	51

Rozdział 3. Pierwszy skrypt w JavaScriptcie.....	53
Oprogramowanie.....	53
Wybór edytora tekstu.....	54
Wybór przeglądarki.....	54
Przygotowanie środowiska pracy.....	55
Windows.....	55
MacOS.....	56
Do czego posłuży pierwszy skrypt.....	57
Wpisujemy pierwszy skrypt.....	58
Analiza skryptu.....	59
Znacznik <SCRIPT>.....	59
Skrypt dla wszystkich przeglądarek.....	60
Wyświetlanie tekstu.....	61
Eksperymenty.....	62

Część II Kurs JavaScriptu

Rozdział 4. Przeglądarka i obiekty dokumentu	65
Skrypty na scenie	65
JavaScript w działaniu	66
Interaktywny interfejs użytkownika	67
Przeszukiwanie niewielkich zbiorów danych.....	68
Weryfikacja zawartości formularzy.....	69
Interaktywne dane.....	69
Ramki.....	70
Dynamiczny HTML.....	71
Kiedy korzystać z JavaScriptu.....	71
Model obiektów dokumentu	72
Struktura hierarchiczna	73
Po otwarciu dokumentu	75
Prosty dokument	75
Wstawienie formularza.....	76
Wstawienie pola tekstowego	76
Wstawienie przycisku polecenia.....	77
Odwołania do obiektów	77
Nazwy obiektów	78
Prosty dokument	78
Wstawienie formularza.....	79
Wstawienie pola tekstowego	79
Wstawienie przycisku polecenia.....	79
O składni kropkowej	80
Model organizacji grup dyskusyjnych.....	80
Co definiuje obiekt.....	81
Właściwości	82
Metody	83
Zdarzenia	84
Ćwiczenia.....	85
Rozdział 5. Skrypty i dokumenty HTML	87
W którym miejscu dokumentu umieszczać skrypty	87
Znacznik <SCRIPT>	87
Rozmieszczenie znaczników	88
Obsługa starszych przeglądarek	90

Instrukcje JavaScriptu	91
Kiedy uruchamiane są instrukcje skryptu	92
Skrypty uruchamiane podczas otwierania dokumentu – wywołanie natychmiastowe	92
Skrypty wywoływane z opóźnieniem	92
Skrypty a programy	94
Ćwiczenia	95
Rozdział 6. Podstawy programowania część I	97
Praca z informacjami	97
Zmienne	98
Tworzenie zmiennych	99
Nazwy zmiennych	99
Wyrażenia i Obliczenia	100
Wyrażenia w pliku skrypt1.htm	101
Wyrażenia i zmienne	102
Konwersje typów danych	103
Konwersja łańcuchów na liczby	104
Zamiana liczb na łańcuchy	105
Operatory	105
Operatory arytmetyczne	106
Operatory porównania	106
Ćwiczenia	107
Rozdział 7. Podstawy programowania część II	109
Decyzje i pętle	109
Struktury sterujące	110
Konstrukcje z warunkiem if	110
Konstrukcje if...else	111
Powtarzanie instrukcji w pętlach	112
Funkcje	113
Parametry funkcji	114
Zasięg zmiennych	115
O nawiasach klamrowych	118
Tablice	118
Tworzenie tablicy	119
Odczytywanie danych z tablicy	120
Tablice równoległe	120
Tablice z obiektami dokumentów	122
Ćwiczenia	123
Rozdział 8. Okna i obiekty dokumentu	125
Obiekty dokumentu	125
Obiekt window	126
Korzystanie z właściwości i metod obiektu window	126
Utworzenie okna	127
Właściwości i metody obiektu window	129
Właściwość window.status	129
Metoda window.alert()	130
Metoda window.confirm()	130
Metoda window.prompt()	131
Obsługa zdarzenia onLoad=	132
Obiekt location	132
Obiekt history	133

Obiekt document	133
Właściwość document.forms[]	134
Właściwość document.title	134
Metoda document.write()	134
Obiekt link	137
Ćwiczenia	138
Rozdział 9. Formularze i elementy formularzy	139
Obiekt form	139
Formularz jako obiekt i kontener	140
Tworzenie formularza	140
Właściwość form.elements[]	140
Obiekty tekstowe	141
Przycisk polecenia	143
Pole wyboru	144
Przycisk opcji	145
Lista wyboru	146
Przekazywanie do funkcji formularzy i elementów formularzy	148
Przesyłanie formularzy	150
Ćwiczenia	152
Rozdział 10. Łańcuchy, arytmetyka i daty	153
Obiekty rdzenia języka	153
Łańcuchy	154
Łączenia łańcuchów	154
Metody obiektu string	155
Obiekt Math	158
Obiekt Date	159
Obliczanie dat	161
Ćwiczenia	163
Rozdział 11. Obsługa ramek i wielu okien	165
Ramki rodzice i ramki potomne	165
Odwołania wśród członków rodziny	167
Odwołanie rodzic-do-potomka	167
Odwołanie potomek-do-rodzica	168
Odwołania potomek-do-potomka	169
Wskazówki do obsługi ramek za pomocą skryptów	169
Jednoczesne sterowanie kilkoma ramkami – paski nawigacyjne	170
Więcej o odwołaniach do okien	172
Ćwiczenia	174
Rozdział 12. Rysunki i Dynamiczny HTML	175
Obiekt image	175
Wymienne rysunki	176
Buforowanie rysunków	176
Tworzenie animowanego menu	178
Więcej dynamiki w języku HTML	181
Ćwiczenia	182

Część III Kompendium wiedzy o obiektach i składni JavaScriptu

Rozdział 13. Podstawy JavaScriptu	185
Język i obiekty dokumentu	185

Standard rdzenia języka – ECMAScript	186
Standard modelu obiektów dokumentu	186
Hierarchia obiektów	187
Hierarchia jako mapa	188
Mapa obiektów dokumentu	189
Tworzenie obiektów JavaScriptu	190
Właściwości obiektów	190
Metody obiektów	192
Obsługiwanie zdarzeń obiektów	193
Obsługa zdarzenia jako metoda	193
Obsługa zdarzenia jako właściwość	194
Umieszczanie skryptów w dokumentach	195
Znaczniki <SCRIPT>	195
Jednostki JavaScriptu	200
Wykrywanie wersji przeglądarki	201
Czy JavaScript jest włączony?	201
Opracowywanie skryptów dla różnych przeglądarek	203
Projektowanie uniwersalnych stron	208
Przeglądarki w wersjach beta	209
Oznaczenia kompatybilności w rozdziałach części III	210
Definicje obiektów	212
Rozdział 14. Obiekt window	215
Terminologia	215
Ramki	216
Tworzenie ramek	216
Model obiektów dokumentu zawierającego ramki	216
Odwołania do ramek	218
Różnice pomiędzy oknami top i parent	219
Zapobieganie wyświetlaniu ramek	219
Wyłączanie ramek	219
Dziedziczenie a struktura hierarchiczna JavaScriptu	220
Synchronizacja ramek	220
Puste ramki	221
Przeglądanie kodu źródłowego ramki	222
Obiekt window	222
Składnia	223
O obiekcie	223
Właściwości	225
Metody	257
Zdarzenia	309
Obiekt frame	314
Składnia	314
O obiekcie	315

Rozdział 15. Obiekty location i history	317
Obiekt location	317
Składnia	318
O obiekcie	318
Właściwości	320
Metody	334
Obiekt history	337
Składnia	337
O obiekcie	337
Właściwości	338
Metody	340
Rozdział 16. Obiekt document	345
Dokumenty i Dynamiczny HTML	346
Obiekt document	347
Składnia	347
O obiekcie	348
Właściwości	349
Metody	378
Rozdział 17. Obiekt link i obiekt anchor	391
Obiekt link	391
Składnia	392
O obiekcie	392
Właściwości	394
Zdarzenia	395
Obiekt anchor	402
Składnia	402
O obiekcie	402
Właściwości	402
Rozdział 18. Obiekty image i area	405
Obiekt image	405
Składnia	406
O obiekcie	406
Właściwości	409
Zdarzenia	415
Obiekt area	417
Składnia	417
O obiekcie	417
Rozdział 19. Obiekt layer	421
Warstwy Netscape'a	421
Obiekt layer	421
Składnia	422
O obiekcie	423
Odwołania do warstw	423
Kompatybilność przeglądarek	426
Właściwości	427
Metody	453
Zdarzenia	461

Rozdział 20. Obiekt applet.....	463
A jeśli nie znam Javy?	463
Obiekt applet.....	463
Składnia	464
O obiekcie	464
Rozdział 21. Obiekt form.....	467
Formularz w hierarchii obiektów	467
Obiekt form.....	467
Składnia	468
O obiekcie	468
Odwołania do elementów formularzy	469
Przekazywanie do funkcji formularzy i elementów formularzy.....	470
Przesyłanie formularzy pocztą elektroniczną	473
Zmiana atrybutów formularza	474
Przyciski w formularzach	474
Kierowanie użytkownika na stronę z podziękowaniami za przesłanie formularza	475
Tablice elementów formularza	476
Właściwości	477
Metody	482
Zdarzenia	485
Rozdział 22. Obiekty tekstowe	489
Obiekt text.....	490
Składnia	490
O obiekcie	490
Właściwości	493
Metody	499
Zdarzenia	502
Obiekt password.....	507
Składnia	507
O obiekcie	508
Obiekt textarea	508
Składnia	508
O obiekcie	509
Znaki końca wiersza w obszarach tekstowych	510
Obiekt hidden.....	511
Składnia	511
O obiekcie	511
Rozdział 23. Przyciski poleceń, przyciski opcji i pola wyboru.....	513
Przyciski button, submit i reset	513
Składnia	514
O obiektach	514
Właściwości	516
Metody	518
Zdarzenia	519
Obiekt checkbox	520
Składnia	521
O obiekcie	521
Właściwości	522
Metody	526
Zdarzenia	527

Obiekt radio.....	528
Składnia	529
O obiekcie	529
Właściwości	531
Metody	535
Zdarzenia	535
Rozdział 24. Obiekt select i obiekt file.....	539
Obiekt select.....	539
Składnia	540
O obiekcie	540
Modyfikowanie elementów listy	542
Właściwości	547
Metody	555
Zdarzenia	556
Obiekt file	557
Składnia	558
O obiekcie	558
Rozdział 25. Obiekt navigator i inne obiekty środowiskowe	561
Obiekt navigator.....	562
Składnia	562
O obiekcie	562
Właściwości	563
Metody	572
Obiekt mimeType	576
Składnia	576
O obiekcie	576
Właściwości	577
Obiekt plugin.....	580
Składnia	581
O obiekcie	581
Właściwości	582
Metody	583
Wyszukiwanie typów MIME i plug-inów	584
Sprawdzanie dostępności typu MIME.....	585
Sprawdzanie dostępności plug-inu	586
Jednoczesne sprawdzanie plug-inu i typu MIME.....	587
Sterowanie instalacją plug-inów (Navigator 3)	588
Obiekt screen.....	588
Składnia	588
O obiekcie	589
Rozdział 26. Obiekt String.....	593
Typy łańcuchowe i typy liczbowe	593
Proste łańcuchy	593
Tworzenie długich zmiennych łańcuchowych.....	594
Łączenie literałów i zmiennych łańcuchowych.....	595
Znaki specjalne wewnątrz łańcuchów	595
Obiekt String.....	596
Składnia	597
O obiekcie	597
Właściwości	599
Metody rozbioru łańcuchów	601

Użyteczne funkcje do pracy z łańcuchami	618
Metody formatujące łańcuchy	619
Kodowanie i dekodowanie łańcuchów zawierających adresy URL	622
Rozdział 27. Obiekty Math, Number i Boolean.....	623
Liczby w JavaScriptcie	624
Liczby całkowite i liczby zmiennoprzecinkowe.....	624
Liczby w zapisie szesnastkowym i ósemkowym.....	626
Zamiana łańcuchów na liczby	627
Zamiana liczb na łańcuchy	629
Gdy liczba nie jest liczbą.....	630
Obiekt Math	630
Składnia	630
O obiekcie	630
Właściwości	631
Metody	632
Tworzenie liczb losowych	633
Skracanie instrukcji odwołujących się do obiektu Math	633
Obiekt Number.....	634
Składnia	634
O obiekcie	635
Obiekt Boolean	635
Składnia	635
O obiekcie	636
Rozdział 28. Obiekt Date	637
Strefy czasowe i czas Greenwich.....	637
Obiekt Date	639
Tworzenie obiektu Date.....	640
Właściwość prototype obiektu Date	642
Metody obiektu Date	642
Różne strefy czasowe	644
Daty jako łańcuchy	645
Przyjazne formaty zapisu dat.....	645
Przekształcenia raz jeszcze	647
Nowe metody.....	647
Obliczanie dat i czasów	648
Błędy i chochliki w obiekcie Date.....	650
Weryfikowanie dat wpisywanych do formularzy	651
Rozdział 29. Obiekt Array.....	655
Dane o określonej strukturze.....	655
Utworzenie pustej tablicy.....	656
Zapełnianie tablicy	658
Ułatwienia w tworzeniu tablic w JavaScriptcie 1.2	660
Usuwanie tablic i elementów tablic	660
Symulacja tablic dwuwymiarowych	661
Właściwości obiektu Array	664
Metody obiektu Array.....	666
Rozdział 30. Wyrażenie regularne i obiekt RegExp	679
Wyrażenia regularne i wzorce.....	679

Podstawy językowe.....	681
Proste wzorce.....	681
Znaki specjalne	682
Grupowanie i odwołania wsteczne	685
Związki pomiędzy obiektami.....	685
Korzystanie z wyrażeń regularnych.....	690
Czy znaleziono dopasowanie?	690
Informacje na temat odpowiednika	692
Zastępowanie łańcuchów	694
Wyrażenie regularne	696
Składnia	696
O obiekcie	697
Właściwości.....	697
Metody	699
Obiekt RegExp.....	701
Składnia	701
O obiekcie	701
Właściwości	702
Rozdział 31. Struktury sterujące	707
Konstrukcje if i if...else.....	707
Proste decyzje	708
Wyrażenia postaci if (warunek).....	709
Decyzje złożone.....	709
Zagnieżdżanie instrukcji if...else	710
Wyrażenia warunkowe.....	712
Powtarzanie w pętlach for.....	713
Wykorzystanie licznika pętli	715
Wychodzenie z pętli	717
Sterowanie wykonywaniem pętli za pomocą instrukcji continue.....	718
Pętla while.....	719
Pętla do...while.....	720
Pętle odczytujące właściwości	720
Instrukcja with.....	722
Etykiety	723
Instrukcja switch	725
Rozdział 32. Operatory JavaScriptu	729
Kategorie operatorów.....	729
Operatory porównania.....	730
Porównywanie wartości różnych typów	731
Operatory matrymonialne	733
Operatory przypisania	736
Operatory boolowskie	738
Działania na liczbach boolowskich	738
Operatory boolowskie w praktyce	740
Operatory bitowe.....	742
Operator typeof	743
Operator void	744
Operator new.....	745
Operator delete.....	745
Operator this.....	746
Priorytet operatorów	748

Rozdział 33. Obiekt Event	751
Dlaczego korzystamy ze zdarzeń?	751
Obsługa zdarzeń.....	752
Właściwości zdarzeń	753
Nowy Navigator – nowe zdarzenia	754
Udoskonalona obsługa zdarzeń myszki	755
Zdarzenia klawiatury	756
Zdarzenie onDragDrop	756
Zdarzenia zachodzące w chwili zmodyfikowania okna	756
Obiekt Event	757
Składnia	757
O obiekcie	757
Właściwości	759
Rozdział 34. Funkcje i obiekty niestandardowe	767
Obiekt function	767
Składnia	767
O obiekcie	768
Tworzenie funkcji	768
Zagnieżdżanie funkcji.....	770
Parametry funkcji.....	770
Właściwości	771
Uwagi na temat stosowania funkcji	776
Wywoływanie funkcji	776
Zakres zmiennych, zmienne globalne i lokalne	777
Zmienne parametryczne.....	780
Rekurencja w funkcjach	781
Grupowanie funkcji w biblioteki	782
Tworzenie obiektów niestandardowych.....	783
Przykład – obiekty planetarne	783
Dołączanie metod niestandardowych	789
Inne sposoby tworzenia obiektów	790
Metody umożliwiające śledzenie obiektów	791
Korzystanie z obiektów niestandardowych	792
Komponenty JavaScriptu	792
Komponenty JavaScript Beans	793
Scriplety	794
Zastosowanie	795
Rozdział 35. Funkcje i instrukcje globalne	797
Funkcje.....	798
Instrukcje.....	804
Rozdział 36. Obiekty JavaScriptu po stronie serwera.....	807
Dołączanie do dokumentów przekształceń wykonywanych na serwerze	808
Skrypty osadzone w plikach na serwerze	808
Biblioteki po stronie serwera	810
Najważniejsze obiekty po stronie serwera	810
Obiekt server.....	810
Obiekt project	811
Obiekt client.....	812
Obiekt request.....	812

Korzystanie z baz danych za pomocą LiveWire	813
Odwoływanie się do bazy danych	814
Odwoływanie się do rekordów	814
JavaScript klienta czy JavaScript serwera?	815

Część IV Praktyczne wykorzystanie JavaScriptu

Rozdział 37. Weryfikacja poprawności danych	819
Weryfikacja danych w trakcie ich wpisywania i weryfikacja pełnego formularza	819
Uruchamianie weryfikacji natychmiastowej	819
Weryfikacja pełnego formularza	820
Projektowanie filtrów	821
Tworzenie biblioteki funkcji filtrujących	821
Funkcja isEmpty()	822
Funkcja isPosInteger()	823
Funkcja isInteger()	824
Funkcja isNumber()	825
Dostosowywanie funkcji weryfikujących dane	826
Łączenie funkcji weryfikujących dane	827
Sprawdzanie poprawności dat i czasu	829
Weryfikacja poprawności danych zgodna z „wymaganiami przemysłowymi”	829
Struktura	830
Jedna funkcja sprawdzająca	830
Przykładowe sprawdzenia	832
Końcowe sprawdzanie danych	843
Planowanie sprawdzania danych	843
Rozdział 38. Technologia LiveConnect: obsługa apletów Javy i plug-inów	845
Przegląd technologii LiveConnect	845
Dlaczego sterować apletami Javy	846
Odrobina Javy	847
Klasy – podstawowe składowe Javy	847
Metody Javy	848
„Właściwości” apletu Javy	849
Odwołania do pól Javy	849
Praktyczne sposoby sterowania apletami za pomocą skryptów	850
Korzystanie z metod udostępnionych skryptom	850
Ograniczenia nałożone na aplety	854
Aplety kropkowe	854
Przekształcanie typów danych	858
Komunikacja apletu ze skryptem	858
Zmiany niezbędne w aplecie	858
Zmiany niezbędne w kodzie HTML	860
Informacje o pliku JSObject.class	860
Zamiana typów danych	862
Przykład komunikacji apletu ze skryptem	862
Sterowanie plug-inów Navigатора	865
Sterowanie plug-inem LiveAudio	866
Plug-in LiveAudio w działaniu	867
Wykorzystanie skryptów do bezpośredniej obsługi klas Javy	870

Rozdział 39. Zaawansowana obsługa zdarzeń.....	873
„Inny” obiekt Event	873
Przechwytywanie zdarzeń.....	874
Włączanie przechwytywania zdarzeń.....	875
Wyłączanie przechwytywania zdarzeń.....	875
Przekazywanie zdarzeń do obiektów docelowych	878
Centrum sterowania zdarzeniami	881
Modyfikowanie zdarzeń	884
Konkurencyjne modele zdarzeń.....	884
Uniwersalny sposób sprawdzania klawiszy specjalnych.....	885
Przechwytywanie klawiszy w obydwu przeglądarkach	886
Przyszłe zdarzenia	888
Rozdział 40. Bezpieczeństwo i skrypty z certyfikatami	889
Ryglowanie drzwi	889
Zderzenie dwóch światów.....	890
Piaskownica z Javą.....	891
Polityka bezpieczeństwa	892
Identyczne pochodzenie	893
Określanie właściwości document.domain.....	894
Sprawdzanie pochodzenia dokumentu	894
Skrypty z certyfikatami.....	896
Podpisane obiekty i skrypty.....	897
Co uzyskujemy dzięki podpisaniu skryptów?	897
Certyfikaty cyfrowe	898
Jak otrzymać certyfikat?	898
Weryfikacja certyfikatu	899
Włączanie opcji codebase principal.....	900
Podpisywanie skryptów	900
Narzędzia do podpisywania.....	901
Przygotowanie skryptów do podpisania	901
Atrybut ARCHIVE	902
Atrybut ID.....	902
Uruchamianie narzędzia podpisującego skrypty	904
Edycja i przesuwanie podpisanych skryptów	905
Korzystanie z zabezpieczonych właściwości i metod.....	906
Uzyskiwanie przywilejów	906
Określanie adresata.....	907
Skrypty odpowiedzialne za uzyskanie uprawnień	909
Nie uchylaj zbyt szeroko drzwi	909
Myśl o użytkownikach.....	910
Przykłady	910
Wykorzystanie chronionej właściwości okna.....	911
Dostęp do plików lokalnych	911
Obsługa błędów w klasach Javy	914
Inne informacje na temat podpisanych skryptów	915
Eksportowanie i importowanie podpisanych skryptów	915
Zabezpieczanie podpisanych stron	916
Znaki międzynarodowe	917

Rozdział 41. Skrypty i zgodność aplikacji w Dynamicznym HTML-u	919
Co kryje się pod pojęciem DHTML?	920
Kaskadowe Arkusze Stylów (CSS1)	920
Rozmieszczanie Kaskadowych Arkuszy Stylów (CSS-P)	922
Model Obiektów Dokumentu (DOM)	922
Skrypty wykonywane po stronie klienta	923
Trudności w tworzeniu DHTML-owych stron zgodnych z obydwooma przeglądarkami	923
Rozszerzenia firmy Netscape – warstwy	924
Rozszerzenia Microsoftu – style	925
Sporządzanie do wspólnego mianownika	925
Tworzenie obiektów PT	926
Odwołania do obiektów PT	927
Niezgodność nazw właściwości	928
Metody	929
Omijanie niezgodności	930
Rozgałęzienie kodu	930
Ekwiwalencja przeglądarek	931
Własne funkcje API	933
Przeglądarki nieobsługujące DHTML-a	934
Przykład aplikacji w DHTML-u	935
Projekt układanki	936
Szczegóły implementacji	937
Własne funkcje API	938
Program główny	941
Wnioski	952
Rozdział 42. Dynamiczny HTML i arkusze JSS w Navigatorze	953
Style JSS	953
Obiekt tags	954
Obiekt classes	955
Obiekt ids	956
Słowo kluczowe all	956
Style kontekstowe	957
Właściwości stylu	957
Właściwości do formatowania bloków strony	958
Właściwości czcionki i tekstu	960
Właściwości klasyfikacyjne	961
Rozmieszczanie dynamiczne	962
Układanka w Navigatorze	962
Dokument główny	963
Panel z objaśnieniami	966
Wnioski	968
Rozdział 43. Dynamiczny HTML Microsoftu	969
Model obiektów dokumentu w Internet Explorerze	969
Typowy obiekt	970
Obiekt textRange	972
Obiekt style	972
Odwołania do obiektów – kolekcja all	973
Właściwości obiektu style	973
Dynamiczne rozmieszczanie	977
Mapa-układanka w wersji dla Internet Explorera	978
Dokument	978
Wnioski	983

Rozdział 44. JScript i model obiektów w Internet Explorerze 4	985
Rdzeń języka	985
Model obiektów dokumentu	986
Obiekty-elementy	987
Kolekcje	988
Zdarzenia	991
Sprawdzanie interpretera obsługującego skrypty	992
Rozdział 45. Uruchamianie skryptów	993
Błędy składniowe i błędy wykonania	993
Komunikaty o błędach	994
Kilka okien z komunikatami o błędach	995
Komunikaty o błędach	996
Odnajdywanie problemów	1001
Sprawdzanie HTML-owych znaczników	1001
Analiza dokumentu źródłowego	1001
Chimeryczne skrypty	1002
Skrypty nie działające w tabelach	1002
Ponowne otwieranie pliku	1003
Sprawdź, co działa	1003
Wyłączanie instrukcji za pomocą komentarzy	1004
Sprawdzanie wartości obliczanych w wyrażeniach	1004
Sprawdzanie właściwości odwołań do obiektów	1005
Korzystanie z Debuggera JavaScriptu	1005
Instalacja Debuggera	1006
Uruchamianie Debuggera	1006
Układ Debuggera	1006
Praca krokowa Debuggera	1007
Poruszanie się po kodzie	1008
Samodzielne śledzenie wartości	1009
Automatyczne śledzenie wartości – czujki	1009
Opracowanie własnej aplikacji do śledzenia kodu	1010
Biblioteka .js	1010
Otwieranie biblioteki .js	1011
Przygotowanie dokumentów do współpracy z biblioteką trace.js	1012
Wywoływanie funkcji trace()	1012
Uwagi o obliczaniu czasu	1013
Awaryjne Nawigatora	1014
Zapobieganie problemom	1014
Odpowiednia struktura	1014
Stopniowa rozbudowa kodu	1015
Testowanie wartości obliczanych wyrażeń	1016
Testowanie funkcji	1016
Testowanie arcydzieła	1016
Rozdział 46. Narzędzia do tworzenia stron	1019
Infuse 2.0 firmy Acadia	1020
Visual JavaScript firmy Netscape	1021
Inne narzędzia działające po stronie serwera	1023

Aplikacje w JavaScriptcie.....	1025
--------------------------------	------

Dodatki

Dodatek A Mapa obiektów Navigatora i informacje o zgodności przeglądarek	1029
--	------

Dodatek B Zastrzeżone słowa JavaScriptu	1035
---	------

Dodatek C Odpowiedzi na pytania z części II	1037
---	------

Odpowiedzi na pytania z rozdziału 4.....	1037
Odpowiedzi na pytania z rozdziału 5.....	1038
Odpowiedzi na pytania z rozdziału 6.....	1040
Odpowiedzi na pytania z rozdziału 7.....	1040
Odpowiedzi na pytania z rozdziału 8.....	1044
Odpowiedzi na pytania z rozdziału 9.....	1045
Odpowiedzi na pytania z rozdziału 10.....	1049
Odpowiedzi na pytania z rozdziału 11.....	1050
Odpowiedzi na pytania z rozdziału 12.....	1051

Dodatek D Materiały o JavaScriptcie w Internecie.....	1053
---	------

Grupy dyskusyjne	1053
Listy dyskusyjne	1053
Dokumentacja w Internecie	1053
Sieć Web	1054

Dodatek E Korzystanie z płyty CD.....	1055
---------------------------------------	------

Wymagany system	1055
Zawartość płyty CD	1055
Listingi przykładów	1055
Mapa obiektów Navigatora z dodatku A (format .pdf)	1056
Program Adobe Acrobat Reader.....	1056
Siedem dodatkowych rozdziałów z aplikacjami JavaScriptu	1057

Skorowidz.....	1059
----------------	------

Rozdział 8.

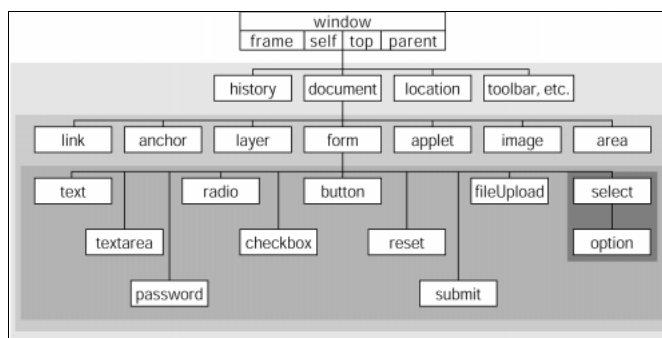
Okna i obiekty dokumentu

Teraz, kiedy masz już podstawowe wiadomości z zakresu programowania, łatwiej będzie zademonstrować, jak za pomocą skryptów obsługiwać obiekty dokumentów. Począwszy od tego rozdziału kurs JavaScriptu powraca do modelu obiektów dokumentu, bliżej zajmując się obiektami, z których często będziesz korzystać we własnych dokumentach.

Obiekty dokumentu

Dla przypomnienia przeanalizuj model obiektów dokumentu Netscape Navigatora zamieszczony ponownie na rysunku 8.1.

Rys. 8.1.
Model obiektów dokumentu Netscape Navigatora



Ten rozdział poświęcony jest obiektom umieszczonym na szczycie lub blisko szczytu hierarchii – obiektom: `window`, `location`, `history` i `document`. Oprócz podawania podstawowych wiadomości umożliwiających pisanie prostych skryptów, rozdział stanowić ma również wprowadzenie do dogłębnej analizy każdego z tych obiektów, jego właściwości, metod i obsługi zdarzeń, którą przedstawimy w części III. Na tym etapie kursu omawiam jedynie podstawowe właściwości, metody i zdarzenia – o wiele więcej dowiesz się z części III. Do analizy przykładów tam umieszczonych niezbędna jest znajomość podstaw programowania omówiona we wcześniejszych rozdziałach.

Obiekt `window`

Na samym szczycie hierarchii obiektów dokumentu znajduje się obiekt `window`. W piramidzie obiektów zajmuje on eksponowane miejsce – jest bowiem głównym kontenerem zawartości stron oglądanych w przeglądarce. O ile okno przeglądarki jest otwarte – nawet jeśli nie ma w nim żadnych dokumentów – w aktualnym modelu pamięci definiowany jest obiekt `window`.

Oprócz obszaru okna, w którym umieszczane są dokumenty, do sfery wpływów obiektu `window` zalicza się także wymiary okna oraz wszystkie elementy okna otaczające zawartość strony. Netscape obszar ten (w którym umieszczane są paski przewijania, paski narzędzi, paski stanu i – na platformach innych niż Macintosh – także pasek menu) nazywa krojem lub *wykończeniem* okna. Nie wszystkie przeglądarki, ani nawet nie wszystkie wersje Navigатора, mogą w pełni kontrolować wykończenie głównego okna, łatwo można jednak opracować skrypty tworzące dodatkowe okna o zadanych rozmiarach, w których umieszczane są tylko te elementy wykończenia, na które się zdecydujesz.

Chociaż ramkami zajmujemy się dopiero w rozdziale 11., mogę już teraz wspomnieć, że każda ramka jest także traktowana jako obiekt typu `window`. Po chwili zastanowienia, należy przyznać, że jest to rozwiązanie logiczne – w każdej ramce może być umieszczony inny dokument. W strukturze obiektów ramkę, w której wyświetlany jest dokument zawierający skrypt, tenże skrypt traktuje jako obiekt `window`.

Jak się za chwilę przekonasz, metody wyświetlające modalne okna dialogowe oraz modyfikujące tekst na pasku stanu na dole okna przeglądarki wygodnie jest dołączyć do obiektu `window`. Obiekt `window` udostępnia także metodę umożliwiającą tworzenie na ekranie samodzielnych okien dialogowych. Analizując właściwości, metody i zdarzenia zdefiniowane dla obiektu `window` w modelu obiektów Netscape'a (zobacz dodatek A), jasne powinno się stać, dlaczego umieszczone są one właśnie tam – porównaj ich zasięg z zasięgiem okna przeglądarki.

Korzystanie z właściwości i metod obiektu `window`

Odwołania skryptu do właściwości i metod obiektu `window` można zapisywać na kilka sposobów, kierując się raczej kaprysem i swoimi upodobaniami, a nie wymaganiami składniowymi. Najbardziej logiczny i najczęściej stosowany sposób tworzenia takich odwołań korzysta z obiektu `window`:

```
window.nazwaWłaściwości  
window.nazwaMetody([parametry])
```

Gdy skrypt odwołuje się do okna zawierającego dokument, obiekt `window` posiada synonim. Synonimem tym jest słowo `self`. Składnia odwołania jest wtedy następująca:

```
self.nazwaWłaściwości  
self.nazwaMetody([parametry])
```

Początkowych nazw obiektów możesz w odwołaniach używać zamiennie, ja skłaniam się jednak ku wykorzystywaniu słowa `self` w bardziej złożonych skryptach, w których umieszczono wiele ramek i okien. Słowo `self` bardziej dobitnie określa aktualne okno, w którym umieszczony jest dokument zawierający skrypt. Według mnie skrypt taki łatwiej się czyta – zarówno autorowi jak i innym osobom.

W rozdziale 4. wspomniałem, że podczas działania dowolnego skryptu, zawsze dostępny jest obiekt `window`. Obiekt ten można więc opuścić z odwołań do innych obiektów umieszczonych w oknie. W zapisie składniowym podanym poniżej przeglądarka przyjmuje, że właściwości i metody odwołują się do aktualnego okna:

```
nazwaWłaściwości  
nazwaMetody([parametry])
```

Jak się za chwilę przekonamy, zapis niektórych metod może okazać się bardziej czytelny, jeśli opuszczysz odwołanie do obiektu `window`. Bez względu na to, na które rozwiązanie się zdecydujesz, oba działać będą poprawnie.

Utworzenie okna

Skrypt nie tworzy głównego okna przeglądarki. Robi to użytkownik, uruchamiając przeglądarkę lub otwierając adres URL, czy jakiś plik za pomocą menu (jeśli w przeglądarce nie jest jeszcze otwarty żaden dokument). Po otwarciu głównego okna, skrypt umieszczony w dokumencie tego okna może natomiast utworzyć dowolną liczbę podokien.

Metodą, która tworzy nowe okno jest metoda `window.open()`. Metoda ta może zawierać do trzech parametrów, w których można określić charakterystyczne cechy okna, takie jak adres URL dokumentu, który ma być otwarty, nazwę okna (stosowaną w HTML-owych odwołaniach wykorzystujących słowo `TARGET`) oraz wygląd okna (czyli jego rozmiar i elementy wykończenia). Nie będę na razie zajmował się szczegółami dotyczącymi poszczególnych parametrów (dokładniej są one omówione w rozdziale 14.), natomiast chciałbym zwrócić uwagę na pewien istotny element metody `window.open()`.

Przyjrzyj się poniższej instrukcji powodującej otwarcie nowego okna o podanym rozmiarze i zawierającego dokument HTML pobrany z serwera:

```
var subWindow = window.open("definition.html", "def",  
    "HEIGHT=200,WIDTH=300")
```

Należy zwrócić uwagę, że jest to instrukcja przypisania. Coś przypisywane jest do zmiennej `subWindow`. Cóż to jest? Okazuje się, że uruchomienie metody `window.open()` powoduje nie tylko otwarcie nowego okna, stosownie do wyspecyfikowanych parametrów – metoda oprócz tego przyjmuje wartość odwołania do nowego okna. W żargonie programistów mówi się, że metoda *zwraca* wartość, w tym przypadku odwołanie do prawdziwego obiektu. Właśnie ta zwrócona wartość przypisywana jest do zmiennej.

Skrypt może teraz wykorzystać tę zmienną jako pełnoprawne odwołanie do podokna. Jeśli chcesz uzyskać dostęp do jego właściwości lub metod, odwołując się do nich powinienś skorzystać z tejże zmiennej. Na przykład, aby zamknąć podokno za pomocą

skryptu w oknie głównym, odwołanie do metody `close()` tego podokna powinno wyglądać w następujący sposób:

```
subWindow.close
```

Jeśli skrypt w głównym oknie wywołałby metodę `window.close()`, `self.close()` lub po prostu `close()`, spowodowałaby ona zamknięcie głównego okna, a nie podokna. Odwołując się do metod lub właściwości innego okna, nie wolno opuszczać odwołania do niego. Wywiera to wpływ na tworzony kod, zechcesz bowiem prawdopodobnie, aby zmienna zawierająca odwołanie do podokna była dostępna tak długo, jak długo dokument główny dostępny jest w przeglądarce. Żeby tak się stało, zmienną należy zainicjalizować jako zmienną globalną, a nie jako zmienną lokalną wewnątrz funkcji (choć wewnątrz funkcji można następnie przypisać do zmiennej wartość). W ten sposób jedna funkcja może otwierać okno, a druga je zamykać.

Na listingu 8.1 przedstawiona jest strona, na której umieszczono przyciski do otwierania nowego, pustego okna i zamykania go z poziomu okna głównego. Aby sprawdzić działanie skryptów, zmniejsz główne okno przeglądarki, tak aby zajmowało mniej niż pół ekranu. Następnie, utworzywszy nowe, małe okno, przesun je, tak by było ono widoczne także po wyświetleniu na pierwszym planie okna głównego. W kodzie na listingu 8.1 kluczowe znaczenie odgrywa to, że zmienną `newWindow` zdefiniowano jako zmienną globalną, dzięki czemu dostęp do niej mają obydwie funkcje. Po zadeklarowaniu zmiennej, jeśli nie zostanie dokonane do niej przypisanie, jej wartość to `null`. Okazuje się, że wartość `null` w warunkach spełnia taką samą rolę jak wartość `false`, natomiast dowolna wartość różna od wartości `null` odpowiada wartości `true`. Warunek w funkcji `closeNewWindow()` sprawdza się więc do sprawdzenia, czy przed wywołaniem metody `close()` otwarto podokno.

Listing 8.1. *Odwołania do obiektów window*

```
<HTML>
<HEAD>
<TITLE>Otwieranie i zamykanie okien</TITLE>
<SCRIPT LANGUAGE="JavaScript">
var newWindow
function makeNewWindow() {
    newWindow = window.open("", "", "HEIGHT=300,WIDTH=300")
}
function closeNewWindow() {
    if (newWindow) {
        newWindow.close()
    }
}
</SCRIPT>
</HEAD>
<BODY>
<FORM>
    <INPUT TYPE="button" VALUE="Utwórz nowe okno"
    ⚡onClick="makeNewWindow()" ">
    <INPUT TYPE="button" VALUE="Zamknij nowe okno"
    ⚡onClick="closeNewWindow()" ">
</FORM>
</BODY>
</HTML>
```

Właściwości i metody obiektu window

Jedna właściwość i trzy metody obiektu `window` opisane w tym rozdziale, wywierają bezpośredni wpływ na interakcje użytkownika ze stroną. Działają one we wszystkich obsługujących skrypty przeglądarkach. Złożone przykłady demonstrujące wykorzystanie poszczególnych właściwości i metod obiektu `window` można znaleźć w części III.

Właściwość `window.status`

Na pasku stanu na dole przeglądarki w chwili wskazania myszką łącza, zazwyczaj wyświetlany jest adres URL. Podczas otwierania dokumentu, inicjalizacji apletów Javy itd. pojawiają się na nim inne komunikaty. Za pomocą JavaScriptu na pasku stanu można jednak wyświetlać własne komunikaty, jeśli tylko miałyby one okazać się pomocne dla osób odwiedzających strony. Przykładowo, zamiast adresu URL, do którego prowadzi łącze, można na pasku stanu wyświetlić jakiś inny, bardziej czytelny opis strony (lub obydwa te elementy, co zadowoli zarówno żółtodziobów, jak i sieciowych weteranów).

Tekst do właściwości `window.status` można przypisać w dowolnej chwili. Aby na pasku stanu w odpowiednim momencie pojawiały się opisy łącz, umieszczony na nim tekst musi być zmieniany przez obsługę zdarzenia `onMouseOver` obiektu `link` (łącza). Specyfika zmieniającej tekst na pasku stanu obsługi zdarzenia `onMouseOver` polega na konieczności zastosowania instrukcji `return true`. W JavaScriptcie wymagane jest to niezwykle rzadko, w tym konkretnym przypadku okazuje się jednak niezbędne do pomyślnego nadpisania przez skrypt tekstu na pasku stanu.

Ze względu na prostotę przypisywania wartości do właściwości `window.status`, bardzo często instrukcje skryptów umieszcza się bezpośrednio w znacznikach, w definicjach obsługi zdarzeń. Jest to rozwiązanie korzystne w przypadku krótkich skryptów, nie jest bowiem wówczas konieczne definiowanie samodzielnej funkcji oraz umieszczanie na stronie znaczników `<SCRIPT>`. Instrukcje umieszczane są po prostu wewnątrz znacznika połączenia `<A>`:

```
<A HREF="http://home.netscape.com" onMouseOver="window.status=
  'Odwiedź stronę firmy Netscape (home.netscape.com)';
  return true">Netscape</A>
```

Przyjrzyj się dobrze dwóm instrukcjom skryptu, umieszczonym w obsłudze zdarzenia `onMouseOver`.

Te dwie instrukcje to:

```
window.status='Odwiedź stronę firmy Netscape (home.netscape.com)';
return true
```

W przypadku umieszczania instrukcji wewnątrz znaczników należy oddzielać je od siebie za pomocą średników. Ważne jest też to, aby wokół nich umieszczony był podwójny cudzysłów ("..."). Aby łańcuch przypisywany do właściwości `window.status`

umieścić wewnątrz skryptu (atrybutu znacznika) zamkniętego w podwójnym cudzysłowie, łańcuch ten otaczamy apostrofami (' . . . ').

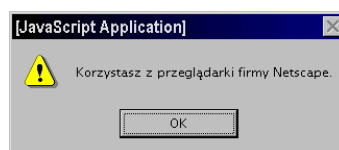
Zmiana napisów na pasku stanu wymaga niewielkiego nakładu pracy, a jest niezwykle użyteczna. Efektem ubocznym udostępnienia możliwości obsługi za pomocą skryptów pasków stanu są pojawiające się na nich – na niektórych stronach – okropne, przewijające się napisy. Ohyda!

Metoda `window.alert()`

Z metody `window.alert()` w niniejszym kursie korzystałem już wielokrotnie. Metoda ta wyświetla okno dialogowe z tekstem, który przekazywany jest do niej jako parametr (zobacz rysunek 8.2). Zamknięcie okna umożliwia umieszczony w nim przycisk OK (jego wyglądu nie można zmieniać).

Rys. 8.2.

*Okno dialogowe
z komunikatem
utworzone za
pomocą JavaScriptu*



Wygląd tego i dwóch innych, opisanych niżej okien dialogowych nieznacznie się zmienił w różnych wersjach przeglądarek obsługujących skrypty. Niektóre wersje Navigатора na pasku tytułowym zamiast tekstu JavaScript Application, który jasno wskazuje, że mamy do czynienia z aplikacją JavaScriptu, wyświetlają napis Netscape. Napisu tego nie można zmieniać za pomocą skryptów. Zmieniana może być tylko treść komunikatu.

Wszystkie trzy metody wyświetlające okna dialogowe dobrze ilustrują możliwość użycia metod obiektu `window` bez konieczności podawania w odwoływaniu nazwy tego obiektu. Pomimo tego, że metoda `alert()` pod względem formalnym jest metodą obiektu `window`, pomiędzy oknem generującym komunikat a oknem z komunikatem nie zachodzą żadne bliższe związki. W stronach, które tworzę zazwyczaj nie korzystam więc z pełnego odwołania:

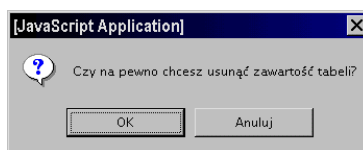
```
alert("Okno komunikatu utworzone za pomocą JavaScriptu." )
```

Metoda `window.confirm()`

W kolejnym typie okna dialogowego, którym się teraz zajmiemy, umieszczone są dwa przyciski. W większości przeglądarek, bez względu na wersję i platformę sprzętową, są to przyciski Cancel (Anuluj) i OK. Okno to, pokazane na rysunku 8.3, nazywane jest oknem potwierdzenia. Co ważniejsze, metoda, która je wyświetla zwraca wartość – jest to wartość `true`, jeśli użytkownik naciśnie przycisk OK i wartość `false` w przypadku naciśnięcia przycisku Cancel (Anuluj). Okno i zwracana przez nie wartość może być wykorzystana do tego, aby zdecydować, które instrukcje skryptu należy wykonać.

Rys. 8.3.

Okno dialogowe z prośbą o potwierdzenie, utworzone za pomocą JavaScriptu



Ponieważ omawiana metoda zawsze zwraca wartość boolowską, możemy wykorzystać ją w warunkach tworzonych za pomocą słowa `if` lub w konstrukcji `if...else`. W poniższym fragmencie kodu pytamy na przykład użytkownika, czy wykonywanie aplikacji ma się rozpocząć od początku. Jeśli tak, w przeglądarce otwierana jest domyślna strona witryny.

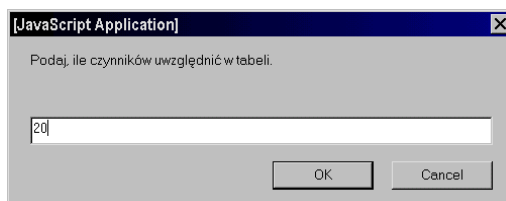
```
if (confirm("Czy na pewno chcesz rozpocząć od początku?")) {  
    location = "index.html"  
}
```

Metoda `window.prompt()`

Ostatnie okno dialogowe tworzone za pomocą metody obiektu `window` zawiera określony w skrypcie tekst komunikatu i pole tekstowe umożliwiające użytkownikowi wpisanie danych (zobacz rysunek 8.4). Okna można się pozbyć za pomocą jednego z dwóch przycisków: `Cancel` (Anuluj) lub `OK`. Pierwszy z nich powoduje anulowanie odpowiedzi, a drugi zaakceptowanie tekstu wpisanego do okna.

Rys. 8.4.

Okno dialogowe z prośbą o wpisanie tekstu, utworzone za pomocą JavaScriptu



Metoda `window.prompt()` korzysta z dwóch parametrów. Pierwszym z nich jest komunikat, stanowiący objaśnienie dla użytkownika. W polu tekstowym można zasugerować domyślną odpowiedź, podając jako drugi parametr zmienną typu `string`. Jeśli nie chcesz, aby w oknie pojawiała się odpowiedź domyślna, skorzystaj po prostu z łańcucha pustego (zapisywanego za pomocą znaków cudzysłowu bez odstępów w środku).

Po naciśnięciu przez użytkownika dowolnego przycisku, metoda zwraca wartość. Po naciśnięciu przycisku `Cancel` (Anuluj) zwracana jest wartość `null`, bez względu na to, co wpisano do pola. Naciśnięcie przycisku `OK` powoduje natomiast, że zwracany jest łańcuch wpisany przez użytkownika. Z informacji tych skrypty mogą korzystać w konstrukcjach sterujących `if` bądź `if...else`. Wartość `null` w warunkach traktowana jest jak wartość `false`. W ten sam sposób traktowane są również łańcuchy puste. Dzięki temu w warunku można łatwo sprawdzić, czy do pola wpisano jakieś znaki, co ułatwia znacznie jego zapisanie. Widać to na poniższym przykładzie:


```
var answer = prompt("Jak masz na imię?", "");
if (answer) {
    alert("Cześć, " + answer + "!")
}
```

Metoda `alert()` wywoływana jest jedynie wówczas, gdy użytkownik wpisał w oknie jakiś tekst i nacisnął przycisk OK.

Obsługa zdarzenia `onLoad=`

Obiekt `window` obsługuje kilka zdarzeń systemowych i zdarzenia wywoływane przez użytkownika. Zdarzeniem, z którego będziesz zapewne korzystał najczęściej jest zdarzenie zachodzące w chwili, gdy wszystkie elementy strony zostaną otwarte przez przeglądarkę. Zdarzenie to nie zachodzi dopóki przeglądarka pobiera rysunki, aplety Javy i pliki danych dla plug-inów. Próba odczytania obiektów dokumentu podczas otwierania strony może okazać się niebezpieczna. Jeśli bowiem obiekt nie został jeszcze odczytany (na przykład ze względu na wolne połączenie sieciowe lub przeciążenie serwera) skrypt spowoduje błąd. Zaletą używania obsługi zdarzenia `onLoad=` do wywoływania funkcji jest to, że mamy pewność, iż wszystkie obiekty dokumentu są już pobrane i umieszczone w modelu obiektów dokumentu przeglądarki.

Definicje obsługi wszystkich zdarzeń obiektu `window` umieszczane są wewnątrz znaczników `<BODY>`.

Mimo tego, że z czasem zaczniesz kojarzyć atrybuty znacznika `<BODY>` z właściwościami obiektu `document`, definicje obsługi zdarzeń umieszczane w jego wnętrzu dotyczą obiektu `window` (z obiektem `document` nie są skojarzone żadne zdarzenia).

Obiekt `location`

Czasem obiekt w hierarchii przedstawia coś, co nie ma fizycznego odpowiednika, takiego jak okno czy przycisk. Sytuacja taka zachodzi w przypadku obiektu `location`. Obiekt ten reprezentuje adres URL dokumentu odczytanego w oknie. Jest on zatem czymś innym niż obiekt `document` omówiony w dalszej części rozdziału. Obiekt `document` to właściwa zawartość strony, podczas gdy obiekt `location` to po prostu adres URL.

O ile nie znasz stosunkowo dobrze mechanizmów działania sieci Web, zapewne nie zdajesz sobie sprawy z tego, że adres URL składa się z wielu komponentów określających lokalizację i sposób transferu danych pliku. Na adres URL składa się nazwa protokołu (na przykład `http:`) i nazwa hosta (na przykład `www.giantco.com`). Wszystkie te elementy można odczytać jako właściwości obiektu `location`. Najczęściej jednak skrypty, które będziesz tworzył, korzystać będą tylko z jednej właściwości – właściwości `href` opisującej pełny adres URL.

Przypisanie wartości do właściwości `location.href` to najprostszy sposób pozwalający na sterowane za pomocą skryptów przechodzenia przeglądarki na inne strony, bądź

to w aktualnym oknie bądź w innej ramce. Do strony we własnej witrynie można przejść podając względny adres URL (tzn. adres względny wobec aktualnie otwartej strony) zamiast adresu pełnego, zawierającego informacje o protokole i nazwie hosta.

Skrót dobrze sprawdzający się w przypadku Navigатора polega na opuszczeniu w odwołaniu właściwości `href`. Można po prostu przypisać do obiektu `location` adres URL (względny lub pełny), a Navigator sam przejdzie do żądanej strony. Dlatego wykonanie obydwu poniższych instrukcji daje ten sam efekt:

```
location="http://www.dannyg.com"  
location.href="http://www.dannyg.com"
```

Jeśli strona ma być otwarta w innym oknie lub ramce, w instrukcji należy umieścić odwołanie do odpowiedniego obiektu. Przykładowo, jeśli skrypt otwiera nowe okno i do zmiennej `newWindow` przypisuje odwołanie do niego, otwarcie w tym oknie nowego dokumentu umożliwi poniższa instrukcja:

```
newWindow.location = "http://www.dannyg.com"
```

Obiekt history

Kolejnym obiektem, który nie ma fizycznego odpowiednika na stronie jest `history`. W każdym oknie przechowywana jest lista ostatnio odwiedzanych przez przeglądarkę stron. Choć na liście umieszczone są adresy URL tych stron, odczytywanie adresów za pomocą skryptu, nie jest w zasadzie możliwe. Metody obiektu `history` pozwalają natomiast poruszać się do przodu i do tyłu po liście adresów (czyli historii) obowiązującej dla aktualnie otwartej strony.

Obiekt document

W obiekcie `document` umieszczana jest właściwa zawartość strony. Właściwości oraz metody obiektu `document` mają ogólnie rzecz biorąc wpływ na wygląd oraz zawartość dokumentu wyświetlanego w oknie. Pominąwszy niektóre zaawansowane opcje Dynamicznego HTML-a wprowadzone w Internet Explorerze 4, po odczytaniu dokumentu, tekst na stronie nie może być za pomocą skryptu odczytywany ani zmieniany.

Jednak jak miałeś okazję przekonać się już w pierwszym skrypcie w rozdziale 3., metoda `document.write()` umożliwia dynamiczne sterowanie zawartością strony podczas jej otwierania. Wiele właściwości obiektu `document` ustalanych jest za pomocą atrybutów w znaczniku `<BODY>`. Pozostałe właściwości to często tablice innych obiektów w dokumencie.

Uzyskanie dostępu do właściwości i metod obiektu `document` nie jest skomplikowane, co widać w poniższych konstrukcjach składniowych:

```
[window.]document.nazwaWłaściwości  
[window.]document.nazwaMetody
```

Odwołanie do obiektu `window` nie jest obowiązkowe wówczas, gdy skrypt odwołuje się do obiektu `document`, w którym sam się zawiera.

Właściwość `document.forms[ ]`

Jednym z obiektów często umieszczanych w dokumencie jest obiekt typu `form`. Ponieważ nie ma powodów, dla których w dokumencie nie można byłoby umieścić więcej niż jednego formularza, kolejne formularze przechowywane są jako elementy tablicy we właściwości `document.forms[ ]`. Jak sobie zapewne przypominasz z opisu tablic umieszczonego w rozdziale 7., numer indeksu wewnątrz nawiasów kwadratowych wskazuje na jeden z elementów tablicy. Aby więc uzyskać dostęp do pierwszego formularza w dokumencie, należy skorzystać z następującego odwołania:

```
document.forms[0]
```

Zazwyczaj lepszym rozwiązaniem jest jednak korzystanie z nazw formularzy przypisywanych do nich za pomocą atrybutu `NAME`. Odwołanie takie wygląda następująco:

```
document.nazwaFormularza
```

Za pomocą obydwu sposobów uzyskujemy dostęp do tego samego obiektu. Jeśli natomiast skrypt ma obsługiwać obiekty wewnątrz formularza, pełne odwołanie do nich powinno zawierać słowa `document` i `form`.

Właściwość `document.title`

Nie wszystkie właściwości obiektu `document` opisuje się wewnątrz znacznika `<BODY>`. Tytuł dokumentu, czyli właściwość `document.title` jest określana wewnątrz znaczników `<TITLE>` w sekcji `<HEAD>`. Ustawienie to pełni w zasadzie rolę kosmetyczną, pozwalając na nadanie stronie zwykłej nazwy tekstowej. Nazwa ta widoczna jest na pasku tytułowym, jak również w historii i zakładkach.

Metoda `document.write()`

Metoda `document.write()` może być wykorzystywana zarówno w skryptach tworzących treść strony podczas jej otwierania, jak i w skryptach wywoływanych z opóźnieniem, używanych do tworzenia nowych stron w tym samym bądź innym oknie.

W wywołaniu metody podaje się jeden łańcuch, stanowiący fragment HTML-owego kodu wpisywanego do okna lub ramki. Parametrami łańcuchowymi mogą być zmienne lub dowolne inne wyrażenia przyjmujące wartości łańcuchowe. Bardzo często we wpisywanej do okna zawartość umieszczane są HTML-owe znaczniki.

Pamiętaj, że po otwarciu strony, strumień wyjściowy przeglądarki jest automatycznie zamykany. Po tym fakcie, metoda `document.write()` wykonana w odniesieniu do aktualnej strony, powoduje natychmiastowe wymazanie jej bieżącej zawartości (w tym zmiennych i innych wartości w dokumencie). Jeśli więc zawartość aktualnej strony ma być zastąpiona nowym HTML-owym kodem wpisywanym do dokumentu za pomocą skryptu, kod ten należy umieścić w zmiennej i wpisać go za pomocą tylko jednej metody `document.write()`. Nie jest konieczne usuwanie starej zawartości strony ani otwieranie nowego strumienia danych. Wszystkim zajmie się metoda `document.write()`.

Ostatnia dobra rada dotycząca metody `document.write()` wiąże się z pokrewną jej metodą `document.close()`. Skrypt musi zamknąć strumień wyjściowy po zakończeniu zapisywania zawartości do okna (aktualnego bądź dowolnego innego). Po wykonaniu ostatniej metody `document.write()`, umieszczonej w skrypcie wywoływanym z opóźnieniem należy pamiętać, aby wywołać metodę `document.close()`. Jeśli tego zaniechamy, to może się okazać, że rysunki i formularze nie będą widoczne, także dowolna inna metoda `document.write()` wywołana później dopisze jedynie tekst do strony, zamiast wyczyścić aktualną jej zawartość i rozpocząć pisanie od nowa. Przykład użycia metody `document.write()` znaleźć można w dwóch poniższych wersjach tej samej aplikacji. W jednej z nich tekst wpisywany jest do tego samego dokumentu, który zawiera skrypt, a w drugiej do samodzielnego okna. Wpisz każdy z dwóch dokumentów, zapisz go i otwórz w przeglądarce.

Kod z listingu 8.2 tworzy przycisk, którego naciśnięcie powoduje wpisanie do dokumentu nowego HTML-owego kodu, w tym HTML-owych znaczników określających tytuł dokumentu i atrybut `BGCOLOR` dla znacznika `<BODY>`. Następnie za pomocą jednej instrukcji `document.write()`, cały gotowy już kod wpisywany jest do dokumentu, wymazując wszelkie pozostałości elementów strony zapisanych na listingu 8.2. Instrukcja `document.close()` jest natomiast niezbędna do odpowiedniego zamknięcia strumienia wyjściowego. Po odczytaniu dokumentu oraz naciśnięciu przycisku polecenia zmieniającego zawartość strony, warto zwrócić uwagę, że zmieniony został również tytuł dokumentu w oknie przeglądarki. Wróć do dokumentu oryginalnego za pomocą przycisku Back (Wstecz) i ponownie naciśnij przycisk polecenia w dokumencie, zwracając tym razem uwagę na to, że dynamiczne tworzenie strony przebiega bardzo szybko, szybciej nawet niż ponowne odczytanie dokumentu oryginalnego.

Listing 8.2. Wykorzystanie metody `document.write()` w odniesieniu do aktualnego okna

```
<HTML>
<HEAD>
<TITLE>Wpisywanie tekstu to bieżącego dokumentu</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function reWrite() {
    // przygotowanie zawartości nowego okna
    var newContent = "<HTML><HEAD><TITLE>Nowy dokument
↵</TITLE></HEAD>"
    newContent += "<BODY BGCOLOR='aqua'><H1>Ten dokument jest
↵całkiem nowy.</H1>"
    newContent += "Naciśnij przycisk Back (Wstecz), aby
↵zobaczyć dokument oryginalny."
    newContent += "</BODY></HTML>"
```

```

    // wpisanie dokumentu HTML do nowego okna
    document.write(newContent)
    document.close() // koniec wpisywania tekstu
}
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="button" VALUE="Zmień treść dokumentu"
onClick="reWrite()" ">
</FORM>
</BODY>
</HTML>

```

Na listingu 8.3 mamy do czynienia z nieco bardziej złożoną sytuacją, bowiem skrypt tworzy podokno, do którego wpisuje następnie generowany automatycznie dokument. Aby w obu wykorzystywanych w skrypcie funkcjach możliwe było odwoływanie się do nowego okna, zmienna `newWindow` deklarowana jest jako zmienna globalna. Natychmiast po otwarciu strony, obsługa zdarzenia `onLoad` wywołuje funkcję `makeNewWindow()`. Funkcja ta tworzy puste podokno. W trzecim parametrze metody `window.open()` podałem właściwość powodującą wyświetlenie w podoknie paska stanu.

Przycisk na stronie wywołuje funkcję `subWrite()`. Pierwsze zadanie funkcji polega na sprawdzeniu właściwości `closed` podokna. Właściwość ta (dostępna jedynie w nowszych wersjach przeglądarek) zwraca wartość `true`, w przypadku gdy okno, do którego się odwołujemy jest zamknięte. Jeśli rzeczywiście zachodzi taka sytuacja (użytkownik mógł samodzielnie zamknąć okno) ponownie wywoływana jest funkcja `makeNewWindow()`, która po raz kolejny otwiera okno.

Teraz, gdy nowe okno jest już otwarte, w zmiennej łańcuchowej przygotowujemy jego zawartość. Tak jak w przypadku kodu na listingu 8.2, wszystko wpisywane jest do okna za jednym razem (choć nie jest to konieczne, gdy tekst wpisywany jest do samodzielnych okien). Następnie wywoływana jest metoda `close()`. Warto zwrócić uwagę na istotną różnicę w wywołaniach metod zarówno `close()` jak i `open()` – w przeciwieństwie do kodu na listingu 8.2 w jednym i drugim przypadku podawane jest odwołanie do podokna.

Listing 8.3. Wykorzystanie metody `document.write()` w samodzielnym oknie

```

<HTML>
<HEAD>
<TITLE>Wpisywanie tekstu to okna potomnego</TITLE>
<SCRIPT LANGUAGE="JavaScript">
var newWindow
function makeNewWindow() {
    newWindow = window.open("", "", "status,height=200,width=300")
}
function subWrite() {
    // utworzenie okna, jeśli ktoś je wcześniej zamknął
    if (newWindow.closed) {
        makeNewWindow()
    }
    // przygotowanie zawartości nowego okna

```

```
var newContent = "<HTML><HEAD><TITLE>Nowy dokument  
⚡</TITLE></HEAD>"  
newContent += "<BODY BGColor='coral'><H1>Ten dokument jest  
⚡całkiem nowy.</H1>"  
newContent += "Naciśnij przycisk Back (Wstecz), aby zobaczyć  
⚡dokument oryginalny."  
newContent += "</BODY></HTML>"  
// wpisanie dokumentu HTML do nowego okna  
newWindow.document.write(newContent)  
newWindow.document.close() // koniec wpisywania tekstu  
}  
</SCRIPT>  
</HEAD>  
<BODY onLoad="makeNewWindow()">  
<FORM>  
<INPUT TYPE="button" VALUE="Zmień treść dokumentu"  
⚡onClick="subWrite()">  
</FORM>  
</BODY>  
</HTML>
```

Obiekt link

W hierarchii obiektów, do obiektu `document` przyporządkowany jest również obiekt `link`, czyli łącze. Dokument może zawierać dowolną liczbę łączy, tak więc w razie potrzeby w odwołaniach do tych obiektów najczęściej używa się składni korzystającej z tablic i indeksów:

```
document.links[n].nazwaWłaściwości
```

Najczęściej łącze nie obsługuje się za pomocą skryptów. Wiąże się jednak z nimi pewien ciekawy aspekt języka. Jeśli kliknięcie łącza ma powodować wykonanie skryptu, a nie natychmiastowe przejście pod wskazany adres URL, w atrybucie `HREF` znacznika można dokonać przekierowania, tak aby wskazywana była funkcja skryptu.

Technika ta wiąże się z zastosowaniem pseudoadresu URL, rozpoczynającego się od łańcucha `javascript:`. Jeśli za tym łańcuchem umieścimy nazwę funkcji, przeglądarki obsługujące skrypty uruchomią ją. Funkcja ta, aby nie wprowadzać zamieszania, powinna ostatecznie dokonywać jakiejś nawigacji, zastosowanie skryptu pozwala jednak na wykonanie również innych czynności, na przykład na jednoczesną zmianę zawartości dwóch ramek w zdefiniowanym układzie.

Konstrukcja, którą należy umieścić wewnątrz łącza wygląda następująco:

```
<A HREF="javascript:void nazwaFunkcji  
⚡([parametr1]...[parametrN])">...</A>
```

Dzięki zastosowaniu słowa kluczowego `void` łącze nie będzie próbowało wyświetlać wartości, które mogą być zwracane przez funkcję. O stosowaniu adresu `javascript:` należy pamiętać w przypadku wszystkich znaczników, zawierających atrybuty `HREF` i `SRC`. Jeśli w danym atrybucie możliwe jest podanie adres URL, poprawny będzie rów-

niez adres rozpoczynający się od łańcucha `javascript:`. Może okazać się to pomocne przy pisaniu skryptów obsługujących mapy po stronie klienta, które niekoniecznie prowadzą w jakieś miejsce, ale powodują, że na stronie zaczyna się coś dziać.

Kolejny logiczny krok w poznawaniu hierarchii obiektów dokumentu wiąże się z formularzem. Zajmiemy się nim w kolejnym rozdziale.

Ćwiczenia

1. Rozstrzygnij, które z poniższych odwołań są poprawne, a które nie. Omów błędy występujące w odwołaniach nieprawidłowych.
 - a. `window.document.form[0]`
 - b. `self.entryForm.entryField.value`
 - c. `document.forms[2].name`
 - d. `entryForm.entryField.value`
 - e. `newWindow.document.write("Hej!")`
2. Zapisz instrukcję JavaScriptu, wyświetlającą wiadomość na pasku stanu, która wita użytkowników na stronie sieci Web.
3. Napisz instrukcję JavaScriptu, wyświetlającą tę samą wiadomość w dokumencie jako nagłówek `<H1>` strony.
4. Utwórz stronę, która w czasie otwierania pyta się użytkownika o jego imię (za pomocą okna dialogowego), a następnie wita użytkownika po imieniu za pomocą tekstu wyświetlanego w części `<BODY>`.
5. Utwórz stronę, po otwarciu której automatycznie wyświetlane byłoby okno dialogowe informujące użytkownika o aktualnym adresie URL.