helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

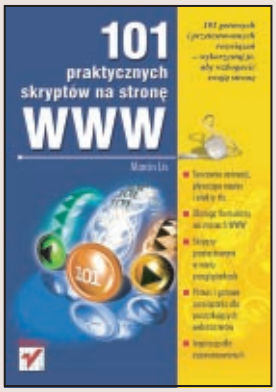
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

101 praktycznych skryptów na stronie WWW



Autor: Marcin Lis
ISBN: 83-7197-879-0
Format: B5, stron: 196

Aby stworzyć atrakcyjną stronę WWW, nie wystarcza już znajomość języka HTML. Witryny wzbogacone przyciągającymi uwagę użytkownika animacjami i elementami interaktywnymi to w Internecie codzienność. Umieszczenie na stronie tych elementów często przerasta umiejętności początkującego webmastera.

„101 praktycznych skryptów na stronę WWW" to książka prezentująca gotowe do wykorzystania, praktyczne skrypty napisane w języku JavaScript, które sprawią, że strona "ożyje" i stanie się bardziej atrakcyjna dla odbiorcy.

Dodaj do swojej strony skrypty obsługujące:

- Formularze
- Odnośniki
- Pływające napisy
- Efekty tła
- Animacje warstw

Książka przeznaczona jest zarówno dla osób początkujących, jak i średnio zaawansowanych (znających podstawy HTML-a, JavaScriptu czy DHTML-a). Każdy skrypt jest przedstawiony w postaci gotowej do uruchomienia. Wystarczy zatem wkleić go do swojej witryny, zupełnie nie przejmując się technicznymi aspektami jego działania, aby osiągnąć zamierzony efekt. Dla osób bardziej zaawansowanych zaprezentowane w książce pomysły mogą być inspiracją do tworzenia własnych projektów.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Wstęp 7

Rozdział 1. Proste zdarzenia i obsługa myszy 9

Skrypt 1. Zdarzenie onLoad. Powitanie użytkownika [E][N][O] 10
Skrypt 2. Zdarzenie onUnload. Pożegnanie użytkownika [E][N][O] 11
Skrypt 3. Powitanie i pożegnanie korzystające z podanego imienia [E][N][O] 11
Skrypt 4. Okno dialogowe wyświetlane po najechaniu myszą na tekst [E][N6][O] 13
Skrypt 5. Zamykanie okna przeglądarki [E][N][O] 14
Skrypt 6. Zamykanie okna przeglądarki potwierdzane przez użytkownika [E][N][O] 15
Skrypt 7. Zmiana napisu na pasku stanu [E][N][O] 17
Skrypt 8. Zmiana kursora myszy na „celownik" [E][N6] 18
Skrypt 9. Wybór kształtu kursora z listy [E] 19
Skrypt 10. Blokowanie funkcji prawego przycisku myszy [E] 22
Skrypt 11. Blokada prawego przycisku myszy rozróżniająca typ przeglądarki [E][N] 24
Skrypt 12. Blokada lewego przycisku myszy [E] 26

Rozdział 2. Skrypty związane z formularzami 29

Skrypt 13. Sprawdzenie ilości formularzy w dokumencie [E][N][O] 30
Skrypt 14. Sprawdzenie czy użytkownik podał wymagane dane [E][N][O] 31
Skrypt 15. Nadanie niewypełnionym polom zadanej wartości [E][N][O] 35
Skrypt 16. Kalkulator wykonujący podstawowe działania arytmetyczne [E][N6][O] 37
Skrypt 17. Przeszukiwanie tekstu [E][N][O] 42
Skrypt 18. Przycisk zmieniający kolor po najechaniu nań kursorem myszy [E][N6] 44
Skrypt 19. Przycisk samoczynnie zmieniający kolor [E][N6] 47
Skrypt 20. Jednoczesna zmiana koloru przycisku i tekstu na przycisku [E][N6] 49
Skrypt 21. Automatyczne przenoszenie kursora między elementami formularza [E][N][O] 50
Skrypt 22. Zablokowanie możliwości wpisywania liter w formularzu [E] 52
Skrypt 23. Pole tekstowe automatycznie zmieniajace swoją wielkość [E] 53

Rozdział 3. Okna i czas 55

Skrypt 24. Otworzenie nowego, pustego okna przeglądarki [E][N] 56
Skrypt 25. Otworzenie nowego okna, o zadanych rozmiarach i zawartości [E][N] 58
Skrypt 26. Wyświetlenie ostrzeżenia o niewłaściwej rozdzielczości ekranu [E][N][O] 60
Skrypt 27. Zmiana koloru paska przewijania [E] 61
Skrypt 28. Odmierzanie czasu [E][N][O] 63
Skrypt 29. Odmierzanie czasu z możliwością zatrzymania zegara [E][N][O] 65
Skrypt 30. Przewijanie treści strony [E][O] 65
Skrypt 31. Pływający ekran [E][N4][O] 68
Skrypt 32. Uciekający ekran [E][O] 69

Skrypt 33. Zegar cyfrowy [E][N][O] 69
Skrypt 34. Wyświetlenie aktualnej daty i czasu [E][N][O] 72
Skrypt 35. Wyświetlenie aktualnej daty i czasu na przycisku [E][N][O] 74
Skrypt 36. Wyświetlenie aktualnej daty i czasu na pasku stanu [E][N][O] 75
Skrypt 37. Wyświetlenie bieżącego dnia tygodnia [E][N][O] 76
Skrypt 38. Wyświetlenie wartości związanych z datą i czasem [E][N][O] 78
Skrypt 39. Zegar podający czas w formacie binarnym [E][N][O] 81

Rozdział 4. Odnośniki 83
Skrypt 40. Automatyczne załadowanie strony [E][N][O] 83
Skrypt 41. Automatyczne załadowanie strony bez użycia JavaScriptu [E][N][O] 84
Skrypt 42. Przyciski będące jednocześnie odnośnikami [E][N][O] 85
Skrypt 43. Odnośniki na liście rozwijanej [E][N][O] 86
Skrypt 44. Odnośniki na liście rozwijanej z bezpośrednią zmianą strony [E][N][O] 88
Skrypt 45. Odsyłacze z dodatkowymi opisami [E][N][O] 88
Skrypt 46. Odnośniki na liście rozwijanej otwierane w nowym oknie [E][N] 90
Skrypt 47. Odnośniki na liście rozwijanej z możliwością wyboru okna [E][N] 92
Skrypt 48. Uniemożliwienie wczytania strony do ramki [E][N][O] 93
Skrypt 49. Dodawanie odsyłacza do zakładki „Ulubione" [E] 94
Skrypt 50. Ustawianie wybranej strony jako startowej [E] 95
Skrypt 51. Akapit tekstowy udający odnośnik [E][N6][O] 96
Skrypt 52. Dodatkowe wyróżnienie odnośnika [E] 97

Rozdział 5. Pływające napisy 101
Skrypt 53. Napis przesuwający się w poziomie w lewo [E][N][O] 101
Skrypt 54. Napis przesuwający się w poziomie w lewo uwzględniający wielkość okna tekstowego [E][N6][O] 102
Skrypt 55. Pływający tekst odbijający się od lewej i prawej strony [E][O] 104
Skrypt 56. Zamiana tekstu przez losowe wstawianie znaków [E][N][O] 105
Skrypt 57. Zamiana tekstów poprzez wymianę znaków od prawej strony [E][N][O] 107
Skrypt 58. Zamiana tekstów poprzez wymianę znaków od lewej strony [E][N][O] 108
Skrypt 59. Symulacja pisania na klawiaturze [E][N][O] 109
Skrypt 60. Tekst rozwijany w prawą stronę i zwijany w lewo [E][N][O] 111
Skrypt 61. Tekst rozwijany w prawą stronę i zwijany w prawo [E][N][O] 112
Skrypt 62. Pływający tekst na pasku stanu [E][N][O] 113
Skrypt 63. Pływająca data na pasku stanu [E][N][O] 114
Skrypt 64. Pływająca data i czas na pasku stanu [E][N][O] 116
Skrypt 65. Pływająca data i czas na pasku stanu II [E][N][O] 117
Skrypt 66. Data i czas na pasku stanu pływająca w obie strony [E][N][O] 119
Skrypt 67. Pływający tekst na pasku tytułu okna przeglądarki [E][N6][O] 120
Skrypt 68. Zegarek w tytule okna [E][N6][O] 122

Rozdział 6. Efekty tła 125
Skrypt 69. Tło zmieniające się po najechaniu na obrazek [E][N6] 125
Skrypt 70. Tło zmieniające się po najechaniu na obrazek II [E][N6] 127
Skrypt 71. Wybór koloru tła z listy rozwijanej [E][N] 128
Skrypt 72. Tło zmieniające cyklicznie kolor [E][N] 129
Skrypt 73. Tło zmieniające kolor losowo [E][N] 130
Skrypt 74. Tło zmieniające kolor losowo II [E][N] 131
Skrypt 75. Zmiana koloru tła w komórce tabeli [E][N6] 132
Skrypt 76. Wybór koloru tła przy użyciu myszy [E][N6] 134
Skrypt 77. Wybór koloru tła przy użyciu myszy II [E][N6] 136
Skrypt 78. Pływające tło [E][N6] 139
Skrypt 79. Tło przesuwające się w poziomie [E][N6] 140
Skrypt 80. Tło z cyklicznie zmieniających się obrazów [E][N6] 141

Skrypt 81. Tło z losowo zmieniających się obrazów [E][N6] 142
Skrypt 82. Rozjaśnianie tła [E][N] 143
Skrypt 83. Ściemnianie tła [E][N] 144
Skrypt 84. Pulsowanie tła [E][N] 144

Rozdział 7. Animacje warstw 147
Skrypt 85. Uciekający obrazek [E][N6][O] 147
Skrypt 86. Pływająca warstwa [E][N4][O] 151
Skrypt 87. Pulsujący tekst [E][N6][O] 152
Skrypt 88. Tekst płynnie zmieniający kolor [E][N6] 153
Skrypt 89. Efekt kurtyny [E][O] 156
Skrypt 90. Wygenerowane dynamicznie tło [E][N4] 158
Skrypt 91. Kurtyna z dynamicznie generowanymi warstwami [E][N4][O] 161
Skrypt 92. Obrazek odbijający się od boków ekranu [E][O] 164
Skrypt 93. Skalowanie obrazka [E][N6] 165
Skrypt 94. Pulsujący obrazek [E][N6] 167
Skrypt 95. Spadające warstwy [E][O] 168
Skrypt 96. Spadające warstwy generowane dynamicznie [E][O] 171
Skrypt 97. Płatki śniegu przemieszczające się w dwóch kierunkach [E][O] 173
Skrypt 98. Realistycznie padający śnieg [E][O] 175
Skrypt 99. Odbijająca się piłka [E][O] 177
Skrypt 100. Piłka poruszająca się po sinusoidzie [E][N6][O] 180
Skrypt 101. Pływające warstwy [E][N][O] 181

Zakończenie 185
Skrypt 102. Pływająca sinusoida [E][N][O] 185

Skorowidz 189

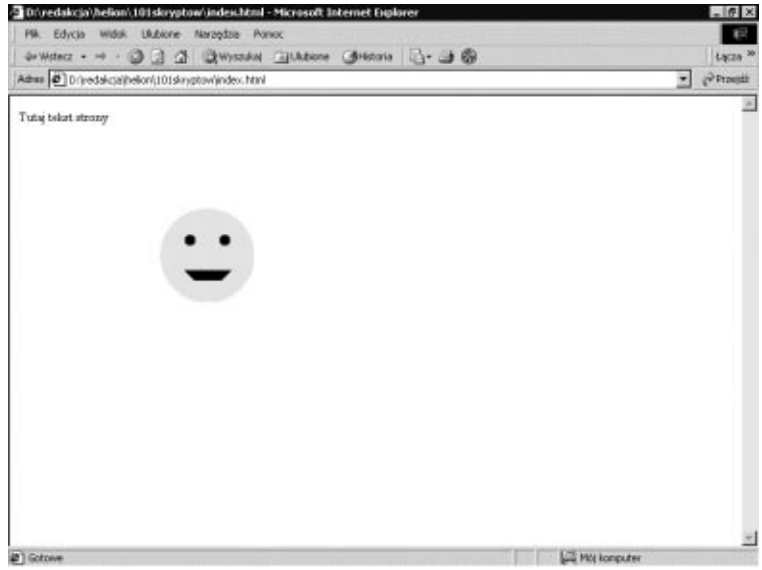
Rozdział 7.

Animacje warstw

Skrypt 85. [E][N6][O]

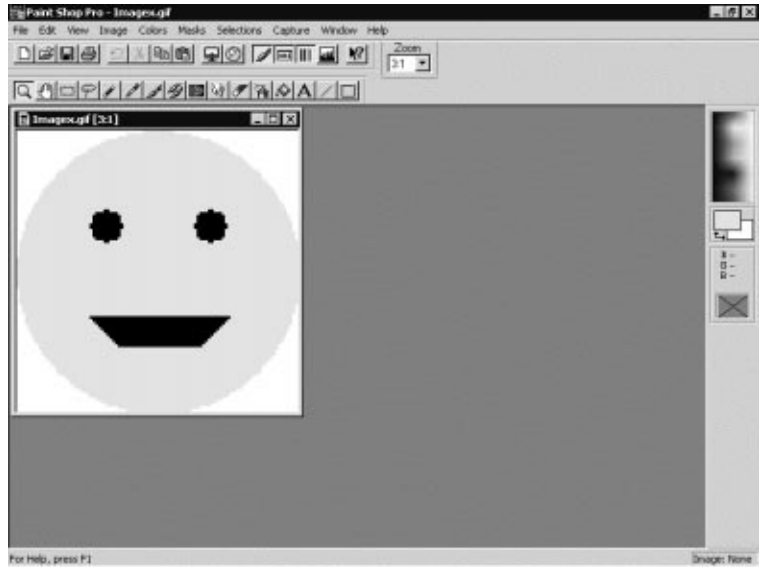
Uciekający obrazek.

Na ekranie wyświetlany jest obrazek zapisany w pliku *image1.gif* (rysunek 7.1). Będzie zmieniał miejsce położenia po najechaniu na niego myszą. W funkcji `init()` ustawiamy położenie warstwy w zależności od rozdzielczości ekranu. Służy nam tutaj do tego obiekt `screen` i jego właściwości `height` i `width` odzwierciedlające odpowiednio wysokość i szerokość ekranu.

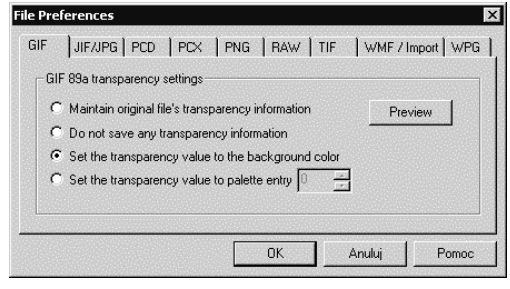


Rysunek 7.1. *Widoczny obrazek będzie „uciekał" przed wskaźnikiem myszy*

Obrazek znajdujący się na warstwie możemy przygotować za pomocą dowolnego edytora graficznego, np. Paint Shop Pro (rysunek 7.2). Zapisujemy go w formacie `gif`, pamiętając jednak o ustawieniu koloru tła jako przezroczystego (rysunek 7.3). Kiedy użytkownik najedzie myszą na warstwę, wykonana zostanie funkcja `over()`, która dokona przesunięcia warstwy. Kierunek przesunięcia jest losowany. Korzystamy tutaj z obiektu `Math`, który udostępnia nam różne stałe (tabela 7.1) i funkcje matematyczne (tabela 7.2).



Rysunek 7.2. *Program Paint Shop Pro pozwoli na przygotowanie grafiki używanej w skrypcie*



Rysunek 7.3. *Kolor tła ustawiamy jako przezroczysty*

Nas interesują następujące funkcje: `Random()`, która zwraca losową liczbę z przedziału 0 – 1 oraz `Round()`, która zwraca argument zaokrąglony do najbliższej liczby całkowitej. Ponieważ potrzebujemy losową wartość całkowitą z przedziału 0 – 7, obu funkcji używamy w sposób następujący:

```
direction = Math.round ((Math.random() * 100) / (12.5));
```

Zmiennej tej używamy następnie jako parametru w instrukcji warunkowej `switch..case`.

Tabela 7.1. *Stałe matematyczne dostępne w JavaScripcie*

Nazwa stałej	Znaczenie	Przybliżona wartość
E	stała Eulera (e)	2,718
LN2	logarytm naturalny z 2	0,693
LN10	logarytm naturalny z 10	2,302
LN2E	logarytm o podstawie 2 z e	1,442
LN10E	logarytm o podstawie 10 z e	0,434
PI	liczba Pi	3,14159
SQRT1_2	pierwiastek kwadratowy z 1/2	0,707
SQRT2	pierwiastek kwadratowy z 2	1,414

Tabela 7.2. *Funkcje matematyczne dostępne w JavaScripcie*

Nazwa metody	Znaczenie
abs()	Zwraca wartość bezwzględną argumentu.
acos()	Zwraca arcus cosinus argumentu.
asin()	Zwraca arcus sinus argumentu.
atan()	Zwraca tangens sinus argumentu.
ceil()	Zwraca najmniejszą liczbę całkowitą większą bądź równą argumentowi.
cos()	Zwraca cosinus argumentu.
exp()	Zwraca e do potęgi równej argumentowi.
Floor()	Zwraca największą liczbę całkowitą mniejszą bądź równą argumentowi.
log()	Zwraca logarytm dziesiętny argumentu.
max()	Zwraca większy z podanych dwóch argumentów.
min()	Zwraca mniejszy z podanych dwóch argumentów.
pow()	Zwraca wartość będącą argumentem pierwszym podniesionym do potęgi równej argumentowi drugiemu.
random()	Zwraca wartość pseudolosową z zakresu 0 – 1.
Round()	Zwraca wartość argumentu zaokrągloną do najbliższej liczby całkowitej.
sin()	Zwraca sinus argumentu.
sqrt()	Zwraca pierwiastek kwadratowy argumentu.
tan()	Zwraca tangens argumentu.

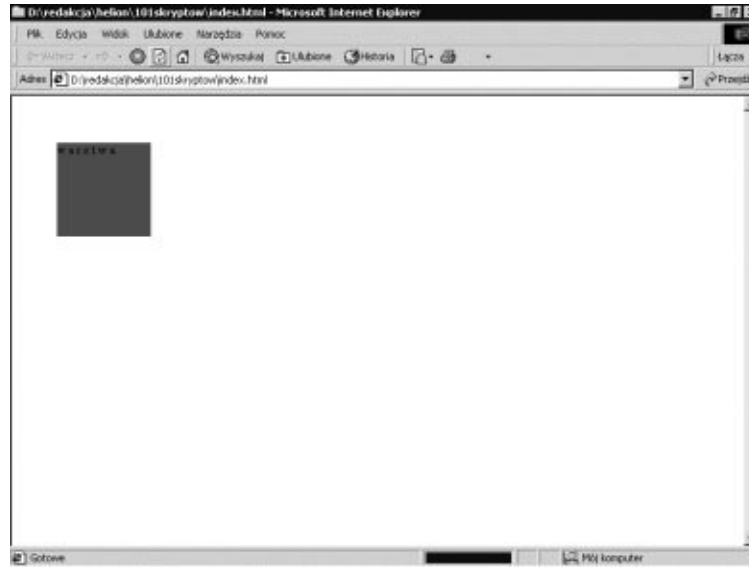
Skrypt 85

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
</HEAD>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
function init(){
```

```
 selector = document.getElementById('warstwa1').style;
 selector.top = screen.height / 2 - 160;
 selector.left = screen.width / 2 - 80;
}
function over(){
 topVal = parseInt(selector.top);
 leftVal = parseInt(selector.left);
 if (isNaN(topVal))
        topVal = screen.height / 2 - 160;
 if (isNaN(leftVal))
        leftVal = screen.width / 2 - 80;
 direction = Math.round ((Math.random() * 100) / (12.5));
 switch(direction){
        case 0 : {leftValAdd = 0; topValAdd = -160; break;}
        case 1 : {leftValAdd = 160; topValAdd = -160; break;}
        case 2 : {leftValAdd = 160; topValAdd = 0; break;}
        case 3 : {leftValAdd = 160; topValAdd = 160; break;}
        case 4 : {leftValAdd = 0; topValAdd = 160; break;}
        case 5 : {leftValAdd = -160; topValAdd = 160; break;}
        case 6 : {leftValAdd = -160; topValAdd = 0; break;}
        case 7 : {leftValAdd = -160; topValAdd = 160; break;}
 }
 newTopVal = topVal + topValAdd;
 newLeftVal = leftVal + leftValAdd;
 if (newTopVal < 0)
        newTopVal = screen.height - 160;
 if (newTopVal > screen.height - 160)
        newTopVal = 0;
 if (newLeftVal < 0)
        newLeftVal = screen.width - 160;
 if (newLeftVal > screen.width - 160)
        newLeftVal = 0;
 selector.top = newTopVal;
 selector.left = newLeftVal;
}
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY onLoad="init()">
<DIV     ID = "warstwa1"
 STYLE = "visibility: visible;
        color: red; position: absolute;
        top: 10; left: 10;"
        onMouseOver = "over()">
<IMG SRC="imagex.gif">
</DIV>
<P>
Tutaj tekst strony
</P>
</BODY>
</HTML>
```

Skrypt 86. [E][N4][O]

Pływająca warstwa (rysunek 7.4).



Rysunek 7.4. *Po wczytaniu skryptu warstwa zaczyna płynąć po ekranie*

W poprzednim skrypcie warstwa skakała po ekranie, jeśli najechaliśmy na nią kursorem myszy. Tym razem sprawmy, aby w miarę płynnie sama przesuwała się po ekranie. Przesunięcie w poziomie osiągamy, modyfikując parametr `left` warstwy.

Skrypt 86

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
</HEAD>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speed = 1;
var vv = 1;
var width = 0;
var layerWidth = 100;
function init(){
 if (document.layers){
         selector = document.layers['warstwa1'];
         width = screen.width;
 }
 else if (document.all){
         selector = document.all['warstwa1'].style;
         width = parseInt(document.body.clientWidth);
 }
```

```
 count = 1;
}
function przesun(){
 if (count < width - layerWidth){
        count++;
 }
 else{
        count = 1;
        vv = -vv;
 }
 selector.left = parseInt(selector.left) + vv;
 setTimeout("przesun()", speed);
}
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY onLoad="init();przesun();">
<DIV     ID="warstwa1"
 style = "visibility:visible; background-color:red;
        position:absolute; top:50;
        left:0; width:100; height:100;
        layer-background-color:red;">
w a r s t w a
</DIV>
</BODY>
</HTML>
```

Warstwa „pływa” w poziomie w prawo i w lewo. Ilość pikseli, o którą ma nastąpić przesunięcie w pojedynczym kroku, definiujemy, ustalając wartość zmiennej `vv`. Dokonywane jest to w funkcji `init()`. Możemy za jej pomocą dodatkowo regulować prędkość przesuwu. Trzeba jednak pamiętać, że wpływa to w znacznym stopniu na płynność ruchu. Tzn. im większa wartość zmiennej `vv`, tym warstwa będzie bardziej „skakać”. Lepiej jednak korzystać z parametru funkcji `setTimeout()`, który określa, co jaki czas nastąpi kolejne wywołanie funkcji `przesun()`.

Skrypt 87. [E][N6][O]

Pulsujący tekst.

Pulsowanie tekstu, a dokładniej jego ściemnianie i rozjaśnianie uzyskujemy w bardzo prosty sposób. Definiujemy warstwę, na której będzie znajdował się napis, a następnie cyklicznie zmieniamy atrybut jej stylu o nazwie `color`. Stosujemy zatem konstrukcję:

```
document.all.nazwa_warstwy.style.color = kolor;
```

Samą definicję koloru konstruujemy z poszczególnych składowych *R*, *G*, *B*, podobnie jak w skryptach 82 – 84, gdzie zmienialiśmy barwę tła.

Skrypt 87

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speed = 1;
var C = 0;
var i = 1;
function textChange(){
 var color = (C < 16)? '0' + C.toString(16):C.toString(16);
 color += color + color;
 color = '#' + color;
 document.getElementById('text').style.color = color;
 C += i;
 if (C > 254 || C < 1) i = -i;
 setTimeout("textChange()", speed);
}
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY onLoad='textChange()'>
<DIV     ID="text"
 style = "visibility: visible;
 width=40%;
 heigth=20%;
 text-Align=center;
 position: relative;
 top: 10%;
 left: 30%;">
<H2>Witamy na naszej stronie!</H2>
</DIV>
</BODY>
</HTML>
```

Skrypt 88 [E][N6]

Tekst płynnie zmieniający kolor.

Skrypt ten jest rozwinięciem pomysłu ze skryptu 87. Skoro możemy ściemniać i rozjaśniać napis, moglibyśmy spowodować, żeby płynnie zmieniał on swoją barwę. Nie będziemy tu jednak generować każdego koloru w funkcji `textChange()`, jak poprzednio, przygotujemy za to tablicę barw. Będą odpowiedzialne za to dwie funkcje: `makeTable()` i `prepareColors()`.

Skrypt 88

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
```

```
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speed = 10;
var step = 1;
var index = 0;
var colorTable = new Array();
var cR = new Array();
var cG = new Array();
var cB = new Array();
function prepareColors(RF, GF, BF, RT, GT, BT, pos){
 if (RF == RT){
        for (i = 0; i < 256; i++){
               cR[i] = RT;
        }
 }
 else{
        if (RF < RT){
               for (i = RF; i < RT; i++){
                      cR[i] = i;
               }
        }
        else{
               x = 0;
               for (i = RF; i > RT; i--){
                      cR[x++] = i;
               }
        }
 }
 if (GF == GT){
        for (i = 0; i < 256; i++){
               cG[i] = GT;
        }
 }
 else{
        if (GF < GT){
               for (i = GF; i < GT; i++){
                      cG[i] = i;
               }
        }
        else{
               x = 0;
               for (i = GF; i >= GT; i--){
                      cG[x++] = i;
               }
        }
 }
 if (BF == BT){
        for (i = 0; i < 256; i++){
               cB[i] = BT;
        }
 }
 else{
        if (BF < BT){
               for (i = BF; i < BT; i++){
                      cB[i] = i;
```

```
                }
        }
        else{
                x = 0;
                for (i = BF; i >= BT; i--){
                        cB[x++] = i;
                }
        }
 }
 for (i = 0; i < 255; i++){
        color = (cR[i] < 16)? '0' + cR[i].toString(16):cR[i].toString(16);
        color += (cG[i] < 16)? '0' + cG[i].toString(16):cG[i].toString(16);
        color += (cB[i] < 16)? '0' + cB[i].toString(16):cB[i].toString(16);
        color = '#' + color;
        colorTable[pos * 256 + i] = color;
 }
}
function makeTable(){
 prepareColors(255, 255, 255, 255, 255, 0, 0);
 prepareColors(255, 255, 0, 0, 255, 0, 1);
 prepareColors(0, 255, 0, 0, 255, 255, 2);
 prepareColors(0, 255, 255, 0, 0, 255, 3);
 prepareColors(0, 0, 255, 255, 0, 255, 4);
 prepareColors(255, 0, 255, 255, 0, 0, 5);
 prepareColors(255, 0, 0, 255, 255, 255, 6);
}
function textChange(){
 document.getElementById('text').style.color = colorTable[index];
 index += step;
 if (index > colorTable.length || index < 0) step = -step;
 setTimeout("textChange()", speed);

}
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY onLoad='makeTable();textChange()'>
<DIV     ID="text"
 style = "visibility: visible;
 width=40%;
 heigth=20%;
 text-Align=center;
 position: relative;
 top: 10%;
 left: 30%;">
<H2>Witamy na naszej stronie!</H2>
</DIV>
</BODY>
</HTML>
```

Ustalenie przejść pomiędzy kolorami jest sprawą indywidualną. W powyższym przypadku zachodzą następujące zmiany barw: *biały->żółty->zielony->seledynowy->niebieski->fioletowy->czerowny->biały*. Odpowiednie składowe *R*, *G*, *B* i kody kolorów przedstawione są w tabeli 7.3.

Tabela 7.3. *Przejścia kolorów dla skryptu 87*

Lp.	R	G	B	#RRGGBB
0	255–>255	255–>255	255–>0	#FFFFFF->#FFFF00
1	255–>0	255–>255	0–>0	#FFFF00->#00FF00
2	0–>0	255–>255	0–>255	#00FF00->#00FFFF
3	0–>0	255–>0	255–>255	#00FFFF->#0000FF
4	0–>255	0–>0	255–>255	#0000FF->#FF00FF
5	255–>255	0–>0	255–>0	#FF00FF->#FF0000
6	255–>255	0–>255	0–>255	#FF0000->#FFFFFF

Tabelę tę należy czytać następująco. W iteracji (kolumna lp.) 0 składowa *R* zmienia się od 255 do 255, czyli ma wartość stałą. Podobnie składowa *G* cały czas jest równa 255. Składowa *B* zmienia się od 255 do 0. Ostatecznie prowadzi to do płynnego przejścia od koloru o kodzie `#FFFFFF` (*biały*) do koloru o kodzie `#FFFF00` (*żółty*). Kolejne iteracje (1 – 6) definiują kolejne zmiany składowych *R*, *G* i *B*.

Generacją przejść tonalnych zajmuje się funkcja `prepareColors(RF, GF, BF, RT, GT, BT, pos)`. Jej parametry określają kierunek zmian poszczególnych składowych koloru. Parametry z literą *F* (od ang. *from*) określają, od jakiej wartości ma się rozpocząć iteracja, parametry z literą *T* (od ang. *to*), na jakiej wartości ma się zakończyć. Parametr `pos` określa numer kolejnej iteracji, a tym samym miejsce w wynikowej tabeli kolorów.

Parametry kolejnych wywołań funkcji `prepareColors()` odczytujemy bezpośrednio z tabeli 7.3. Zatem zerową iterację określa wywołanie:

```
prepareColors(255, 255, 255, 255, 255, 0, 0);
```

iterację pierwszą wywołanie:

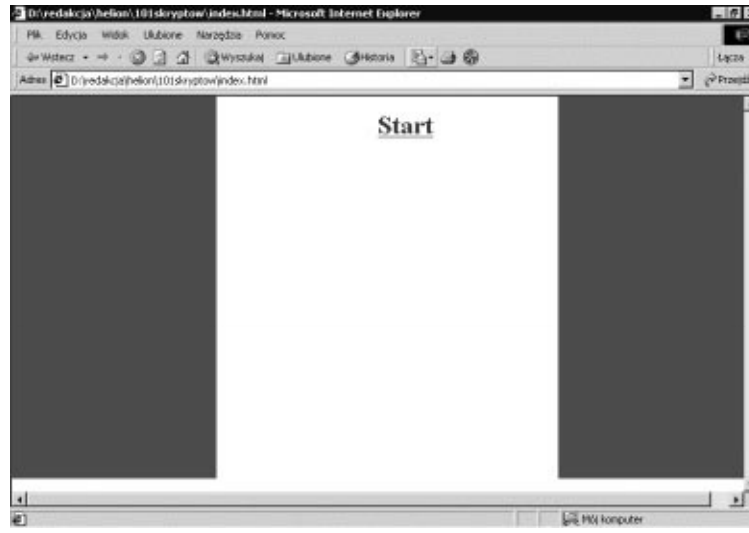
```
prepareColors(255, 255, 0, 0, 255, 0, 1);
```

itd.

Skrypt 89. [E][O]

Efekt kurtyny.

Przykład ten umożliwi nam uzyskanie efektu rozsuwającej się kurtyny odkrywającej zawartość strony WWW (rysunek 7.5). Wykorzystamy do tego dwie warstwy, które będą przesuwały się od środka na boki. W celu ustalenia wielkości warstw użyjemy znanych już nam właściwości `height` i `width` obiektu `screen`. Odpowiednich przypisań dokonamy w wywoływanej przy załadowaniu dokumentu funkcji `init()`. Sama animacja będzie wykonywana w funkcji `przesun()`.



Rysunek 7.5. *Złożona z warstw kurtyna odsłania elementy strony WWW*

Skrypt 89

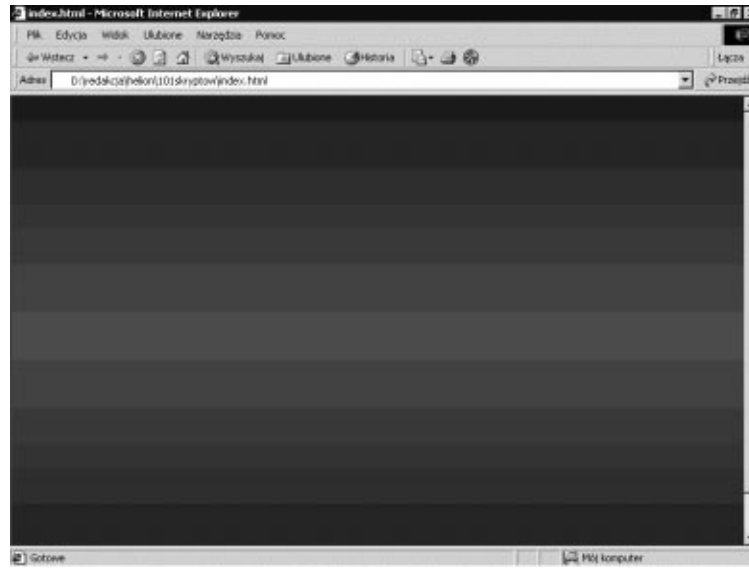
```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
</HEAD>
<STYLE>
#warstwa1{
 visibility:visible;
 background-color:red;
 position:absolute;
 top:0;
 left:0;
 width:0;
 height:0;
 layer-background-color:red;
}
#warstwa2{
 visibility:visible;
 background-color:red;
 position:absolute;
 top:0;
 left:0;
 width:0;
 height:0;
 layer-background-color:red;
}
</STYLE>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
function init(){
 if (document.all){
         document.all['warstwa1'].style.height = screen.height - 150;
         document.all['warstwa1'].style.width = screen.width / 2;
```

```
        document.all['warstwa1'].style.left = 0;
        document.all['warstwa2'].style.height = screen.height - 150;
        document.all['warstwa2'].style.width = screen.width / 2;
        document.all['warstwa2'].style.left = screen.width / 2;
 }
 count = (screen.width / 2) / 2;
}

function przesun(){
 if (document.all){
        x2 = parseInt(document.all['warstwa2'].style.left);
        h1 = parseInt(document.all['warstwa1'].style.width);
        h2 = parseInt(document.all['warstwa2'].style.width);
        document.all['warstwa2'].style.left = x2 + 2;
        document.all['warstwa1'].style.width = h1 - 2;
        document.all['warstwa2'].style.width = h2 - 2;
 }
}
function start(){
 if (count > 0){
        przesun();
        setTimeout("start()", 1);
        count--;
 }
}
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY onLoad="init()">
<H1 align = "center">
<DIV id="warstwa1" style="">
</DIV>
<DIV id="warstwa2" style="">
</DIV>
<DIV style="position:absolute">
<A HREF="javascript:start()">Start</A>
</DIV>
</H1>
<P>
Tutaj tekst strony
</P>
</BODY>
</HTML>
```

Skrypt 90. [E][N4]

Wygenerowane dynamicznie tło.

Skrypt ten będzie generował wielokolorowe tło. Utworzymy w tym celu dużą ilość warstw, z których każda kolejna będzie ma nieco zmieniony kolor. Następnie wyświetliliśmy je wszystkie na ekranie (rysunek 7.6). Kolor (w postaci liczbowej) dla warstwy możemy zdefiniować na cztery sposoby:



Rysunek 7.6. *Wygenerowane za pomocą warstw wielokolorowe tło*

```
style="color:#ff0"
style="color:#ffff00"
style="color:rgb(255,255,0)"
style="color:rgb(100%,100%,0%)"
```

Najwygodniejsze w tym przypadku było podanie wartości składowych *RGB* w postaci trzech liczb dziesiętnych z zakresu 0 – 255. Zbudujemy więc na początek tablicę kolorów, manipulując kolorem czerwonym i niebieskim w następujący sposób:

```
colorTable = new Array();
for (i = 0; i < 256 + 256; i++){
 colorTable[i] = "rgb(" + i + ",0," + (255 - i) + ");";
 colorTable[i + 256] = "rgb(" + (255 - i) + ",0," + i + ");";
}
```

Tak stworzoną tablicę wykorzystamy do wygenerowania odpowiedniej ilości warstw, za co odpowiedzialna będzie funkcja `generuj()`. Jako parametry przyjmie ona szerokość i wysokość okna, przy czym dla uproszczenia wykorzystamy wyłącznie parametr dotyczący szerokości:

```
function generuj(w, h){
 str = "<STYLE type='text/css'>";
 for (i = 0; i < 256 + 256; i++){
         str += "#warstwa" + i + "{position:absolute;";
         str += "width:" + w + "; height:1; left:0;";
         str += "top:" + i + ";background-color:" + colorTable[i];
         str += "layer-background-color:" + colorTable[i];
         str += "clip:rect(0," + w + ",1,0);}";
 }
 str = str + "</STYLE>";
 document.write(str);
}
```

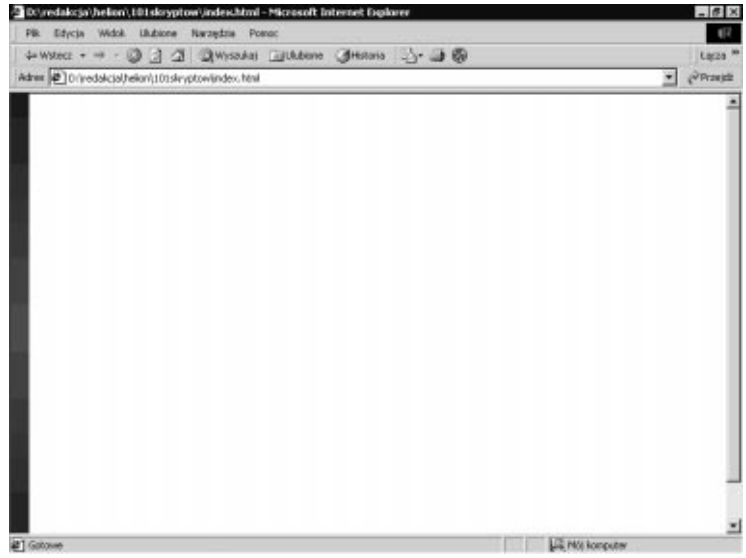
W ten sposób stworzymy zestaw stylów osadzonych, których z kolei użyjemy przy generacji warstw ze znacznikiem DIV. Odpowiedzialny będzie za to następujący fragment kodu:

```
for (i = 0; i < 256 + 256; i++){
 str = "<DIV id='warstwa" + i + "'></DIV>";
 document.write(str);
}
```

Skrypt 90

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
</HEAD>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
colorTable = new Array();
for (i = 0; i < 256; i++){
 colorTable[i] = "rgb(" + i + ",0," + (255 - i) + ");";
 colorTable[i + 256] = "rgb(" + (255 - i) + ",0," + i + ");";
}
function generuj(w, h){
 str = "<STYLE type='text/css'>";
 for (i = 0; i < 256 + 256; i++){
         str += "#warstwa" + i + "{position:absolute;";
         str += "width:" + w + "; height:1; left:0;";
         str += "top:" + i + ";background-color:" + colorTable[i];
         str += "layer-background-color:" + colorTable[i];
         str += "clip:rect(0," + w + ",1,0);}";
 }
 str = str + "</STYLE>";
 document.write(str);
 str = "";
}
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY color="white">
xxx
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
if (document.all){
 generuj(document.body.scrollWidth, document.body.scrollHeight);
}
else if (document.layers){
 generuj(window.innerWidth, window.innerHeight);
}
for (i = 0; i < 256 + 256; i++){
 str = "<DIV id='warstwa" + i + "'></DIV>";
 document.write(str);
}
// Koniec kodu JavaScript -->
</SCRIPT>
</BODY>
</HTML>
```

Warto zauważyć, że trzy literki *xxx*, znalazły się za znacznikiem <BODY> nieprzypadkowo. Nie będą one widoczne, jako że zostaną przykryte przez warstwy. Są one jednak niezbędne dla prawidłowej interpretacji skryptu przez Internet Explorera w wersjach poniżej 6. Jeżeli bowiem plik *HTML* nie zawiera żadnej wyświetlanej treści w sekcji *BODY* (np. użytych przez nas znaków „x"), przeglądarki te generują warstwy o nieprawidłowej szerokości. Łatwo możemy się o tym przekonać, usuwając wspomniany fragment kodu. Efekt będzie mniej więcej taki jak na rysunku 7.7.



Rysunek 7.7. *Usunięcie fragmentu kodu powoduje błędne działanie skryptu*

Przykład ten należy też traktować jako pokaz możliwości dynamicznego generowania warstw. Stosowanie go w praktyce napotka na spore problemy ze względu na wolne działanie i duże zużycie zasobów systemowych. Trzeba pamiętać, że wygenerowaliśmy 512 warstw.

Skrypt 91. [E][N4][O]

Kurtyna z dynamicznie generowanymi warstwami.

Poprzedni skrypt pokazał nam, jak generować warstwy z poziomu JavaScript, możemy wiedzę tą wykorzystać do zmodyfikowania naszego przykładu z rozsuwającą się kurtyną (skrypt 89). Za wygenerowanie odpowiedniego stylu będzie teraz odpowiedzialny następujący kod:

```
if (document.layers){
 str = "<STYLE type='text/css'>";
 str += "#warstwa1 {visibility:visible; background-color:red; ";
 str += "position:absolute; top:0; left:0; width:50; height:50;";
 str += "layer-background-color:red;clip:rect(0,";
 str += screen.width / 2 + ",";
 str += screen.height - 138 + ", 0);}";
 str += "#warstwa2 {visibility:visible; background-color:red; ";
```

```
 str += "position:absolute; top:0; left:" + screen.width / 2 + ";"
 str += "width:50; height:50;";
 str += "layer-background-color:red;clip:rect(0,";
 str += screen.width / 2 + ",";
 str += screen.height - 138 + ", 0);}";
 str += "</STYLE>"
 document.write(str);
}
```

Wystarczy teraz wprawić tak wygenerowane warstwy w ruch. Dokonujemy tego w sposób analogiczny jak w skrypcie 89, manipulując parametrami `left` oraz `width` warstw. W przypadku przeglądarki Netscape używamy metody `offset()` w postaci:

```
document.layers[nazwa_warstwy].offset(offsetH, offsetV);
```

gdzie *offsetH* to przesunięcie warstwy w poziomie (w pikselach), a *offsetV* przesunięcie w pionie.

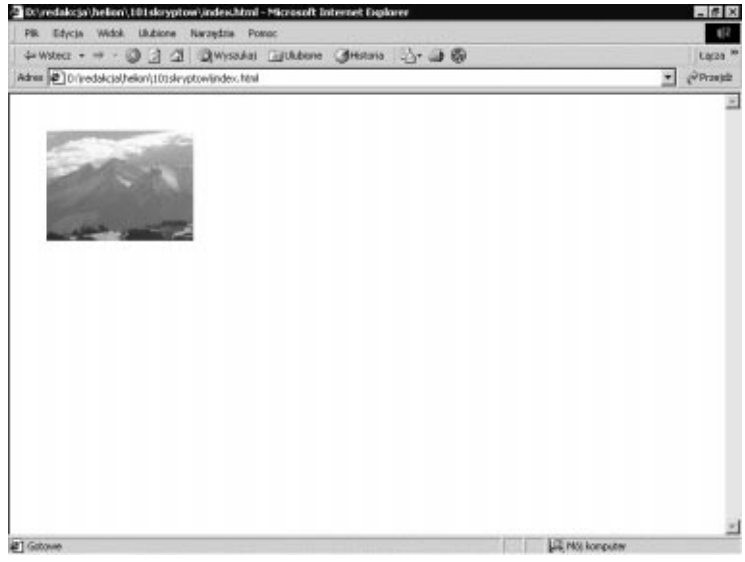
Skrypt 91

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
</HEAD>
<STYLE>
#warstwa1{
 visibility:visible;
 background-color:red;
 position:absolute;
 top:0;
 left:0;
 width:0;
 height:0;
 layer-background-color:red;
}
#warstwa2{
 visibility:visible;
 background-color:red;
 position:absolute;
 top:0;
 left:0;
 width:0;
 height:0;
 layer-background-color:red;
}
</STYLE>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
if (document.layers){
 str = "<STYLE type='text/css'>";
 str += "#warstwa1 {visibility:visible; background-color:red; ";
 str += "position:absolute; top:0; left:0; width:50; height:50;";
 str += "layer-background-color:red;clip:rect(0,";
 str += screen.width / 2 + ",";
 str += screen.height - 138 + ", 0);}";
```

```
 str += "#warstwa2 {visibility:visible; background-color:red; ";
 str += "position:absolute; top:0; left:" + screen.width / 2 + ";"
 str += "width:50; height:50;";
 str += "layer-background-color:red;clip:rect(0,";
 str += screen.width / 2 + ",";
 str += screen.height - 138 + ", 0);}";
 str += "</STYLE>"
 document.write(str);
}
function init(){
 if (document.all){
         document.all['warstwa1'].style.height = screen.height - 150;
         document.all['warstwa1'].style.width = screen.width / 2;
         document.all['warstwa1'].style.left = 0;
         document.all['warstwa2'].style.height = screen.height - 150;
         document.all['warstwa2'].style.width = screen.width / 2;
         document.all['warstwa2'].style.left = screen.width / 2;
 }
 count = (screen.width / 2) / 2;
}
function przesun(){
 if (document.all){
         x2 = parseInt(document.all['warstwa2'].style.left);
         h1 = parseInt(document.all['warstwa1'].style.width);
         h2 = parseInt(document.all['warstwa2'].style.width);
         document.all['warstwa2'].style.left = x2 + 2;
         document.all['warstwa1'].style.width = h1 - 2;
         document.all['warstwa2'].style.width = h2 - 2;
 }
 else if (document.layers){
         document.layers['warstwa1'].offset(-2,0);
         document.layers['warstwa2'].offset(+2,0);
 }
}
function start(){
 if (count > 0){
         przesun();
         setTimeout("start()", 1);
         count--;
 }
}
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY onLoad="init()">
<H1 align = "center">
<DIV id="warstwa1" style="">
</DIV>
<DIV id="warstwa2" style="">
</DIV>
<DIV style="position:absolute">
<A HREF="javascript:start()">Start</A>
</DIV>
</H1>
<P>
Tutaj tekst strony
</P>
</BODY>
</HTML>
```

Skrypt 92. [E][O]

Obrazek odbijający się od boków ekranu (rysunek 7.8).



Rysunek 7.8. *Widoczny obrazek będzie „pływał" po ekranie, odbijając się od jego boków*

Efekt obrazka pływającego po ekranie i odbijającego się od jego boków osiągniemy, manipulując parametrami `left` oraz `top` obiektu `style` odpowiedniej warstwy. Niezbędne jest również oczywiście wykrywanie kolizji z brzegami okna przeglądarki. Funkcje te realizuje kod w skrypcie 92.

Skrypt 92

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
</HEAD>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speed = 40;
var stepX = 1;
var stepY = 1;
var yPos = 1, xPos = 1;
function scroll(){
 var screenWidth = document.body.clientWidth;
 var screenHeight = document.body.clientHeight;
 var layerWidth = warstwa1.offsetWidth;
 var layerHeight = warstwa1.offsetHeight;
 xPos = xPos + stepX;
 yPos = yPos + stepY;
 warstwa1.style.left = xPos;
 warstwa1.style.top = yPos;
```

```
 if((xPos >= screenWidth - layerWidth) || (xPos < 1)){
        stepX = - stepX;
 }
 if((yPos >= screenHeight - layerHeight) || (yPos < 1)){
        stepY = - stepY
 }
 setTimeout("scroll()", speed);
}
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY onLoad="scroll();">
<DIV ID="warstwa1" style="position:absolute;">
<IMG SRC="image1.jpg">
</DIV>
</BODY>
</HTML>
```

Skrypt ten działa w sposób następujący. Tworzymy warstwę o nazwie `warstwa1` i umieszczamy na niej obrazek *image1.jpg*. Oczywiście plik o takiej nazwie musi znajdować się w podanej lokalizacji. Warstwę tę przesuwamy w funkcji `scroll()` w taki sposób, by po osiągnięciu końców ekranu odbijała się od nich jak piłka. Osiągamy to, manipulując zmiennymi `xStep` i `yStep`. Jeżeli bieżąca pozycja *x* animowanej warstwy jest mniejsza od 1, zmienna `xStep` musi być dodatnia, jeżeli natomiast pozycja ta jest większa lub równa szerokości ekranu, `xStep` musi przyjąć wartość ujemną. Ze zmienną `yStep` postępujemy analogicznie. Oczywiście, aby efekt był prawidłowy, w obliczeniach należy uwzględnić szerokość i wysokość warstwy. Szerokość warstwy uzyskujemy dzięki linii:

```
var layerWidth = warstwa1.offsetWidth;
```

a wysokość:

```
var layerHeight = warstwa1.offsetHeight;
```

Skrypt 93. [E][N6]

Skalowanie obrazka.

Manipulując właściwościami `width` i `height` umieszczonego na warstwie obiektu obrazka definiowanego znacznikiem `<IMG>`, możemy osiągnąć ciekawy efekt jego przeskalowania. Może on np. płynnie powiększać się od zera, aż do osiągnięcia swoich pełnych wymiarów (rysunek 7.9).

Skrypt 93

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
```

Rysunek 7.9. *Obrazek jest skalowany do zadanych wymiarów*

```
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speed = 100;
var sizeX = 320;
var sizeY = 200;
var stepX = 0;
var stepY = 0;
var steps = 20;
var imgObj;
function ustaw(){
 imgObj = document.getElementById('obrazekId');
 stepX = Math.ceil(sizeX / steps);
 stepY = Math.ceil(sizeY / steps);
}
function resize(){
 currentW = imgObj.width;
 currentH = imgObj.height;
 if ((currentW < sizeX) && (currentH < sizeY)){
         imgObj.width = currentW + stepX;
         imgObj.height = currentH + stepY;
         setTimeout("resize()", speed);
 }
}
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY onLoad='ustaw();resize();'>
<H1 ALIGN="center">Treść strony<H1>
<DIV ID="imgLayer" ALIGN="center">
<IMG     SRC="image1.gif"
 NAME="obrazek"
 ID="obrazekId"
 HEIGHT="1"
 WIDTH="1">
</DIV>
</BODY>
</HTML>
```

Ilość kroków, w których obrazek zostanie powiększony do swojej oryginalnej wielkości, definiowany jest poprzez zmienną `steps`. Określa ona tym samym jakość i, częściowo, szybkość całej animacji. Drugim parametrem wpływającym na prędkość dokonywanych zmian jest znana nam dobrze zmienna `speed` odpowiadająca za czas, który ma upłynąć pomiędzy kolejnymi wywołaniami funkcji `resize()`.

W każdym wywołaniu funkcji `resize()` zwiększamy wysokość obrazka o wielkość `stepX`, natomiast szerokość o wielkość `stepY`. Parametry `stepX` i `stepY` wyliczamy w funkcji `ustaw()` na podstawie zdefiniowanej wcześniej w zmiennej `steps` ilości kroków oraz wielkości obrazka. Służą do tego wzory:

```
stepX = Math.ceil(sizeX / steps);
stepY = Math.ceil(sizeY / steps);
```

Skrypt 94. [E][N6]

Pulsujący obrazek.

Skrypt ten jest modyfikacją skryptu 93. Powoduje on, że zdefiniowany na warstwie obrazek cyklicznie zwiększa i zmniejsza swoje rozmiary. Efekt ten osiągamy poprzez zmianę znaku wartości `stepX` i `stepY` w przypadku, gdy wielkość obrazu jest mniejsza od zera lub większa od wielkości oryginalnej.

Skrypt 94

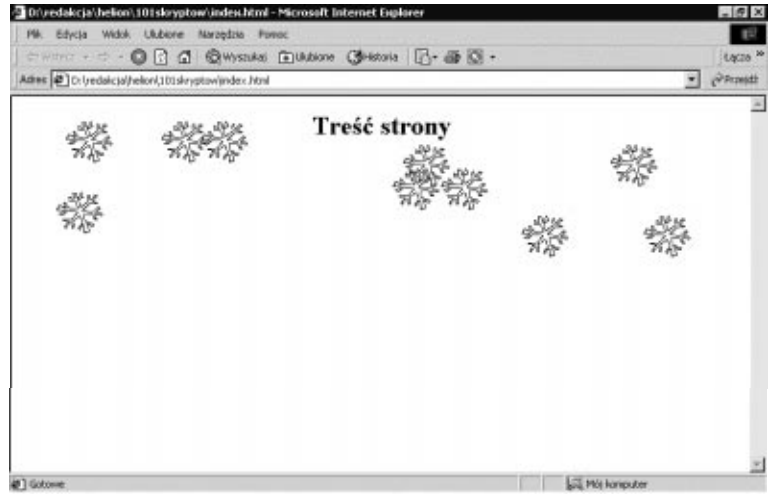
```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speed = 100;
var sizeX = 320;
var sizeY = 200;
var stepX = 0;
var stepY = 0;
var steps = 20;
var imgObj;
function ustaw(){
 imgObj = document.getElementById('obrazekId');
 stepX = Math.ceil(sizeX / steps);
 stepY = Math.ceil(sizeY / steps);
}
function resize(){
 currentW = imgObj.width;
 currentH = imgObj.height;
 if (
 ((currentW >= sizeX) || (currentH >= sizeY)) ||
 ((currentW <= 0) || (currentH <= 0))
```

```
 ){
        stepX = -stepX;
        stepY = -stepY;
 }
 imgObj.width = currentW + stepX;
 imgObj.height = currentH + stepY;
 setTimeout("resize()", speed);
}
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY onLoad='ustaw();resize();'>
<H1 ALIGN="center">Treść strony<H1>
<DIV ID="imgLayer" ALIGN="center">
<IMG     SRC="image1.gif"
 NAME="obrazek"
 ID="obrazekId"
 HEIGHT="1"
 WIDTH="1">
</DIV>
</BODY>
</HTML>
```

Skrypt 95. [E][O]

Spadające warstwy.

Skrypt ten pozwala na uzyskanie efektu padającego śniegu (rysunek 7.10), deszczu czy też spadania innego typu obiektów. W czasie walentynek, na niektórych stronach spotyka się na przykład spadające serca. W najprostszym przypadku zadanie to jest dosyć łatwe w realizacji. Definiujemy na stronie odpowiednią ilość warstw zawierających obrazki o zadanym motywie, a następnie wprawiamy je w ruch.



Rysunek 7.10. *Padający na stronie „śnieg”*

Należy tylko pamiętać, aby każda „spadała” z właściwą sobie, różną od innych prędkością. Uzyskujemy to dzięki losowaniu dla każdej warstwy ilości pikseli, o jaką ma się przemieścić w pojedynczym ruchu. Podobnie, losowana powinna być współrzędna *x*położenia początkowego.

Skrypt 95

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speedTable = new Array();
var speed = 50;
var speedLevels = 5;
function ustaw(){
 for (i = 0; i < 10; i++){
         if (document.all){
               var posX = Math.ceil(Math.random() * (document.body.clientWidth - 50));
                document.all['el' + i].style.left = posX;
         }
         speedTable[i] = Math.ceil(Math.random() * speedLevels);
 }
}
function start(){
 for (i = 0; i < 10; i++){
         if (document.all){
                x = parseInt(document.all['el' + i].style.top);
                x += speedTable[i];
                if (x >= document.body.clientHeight - 45){
                        x = 1;
                        var posX = Math.ceil(Math.random() * (document.body.
                        ➥clientWidth - 50));
                        document.all['el' + i].style.left = posX;
                        speedTable[i] = Math.ceil(Math.random() * speedLevels);
                }
                document.all['el' + i].style.top = x;
         }
 }
 setTimeout("start()", speed);
}
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY onLoad="ustaw();start();">
<DIV     ID="el0"
 style = "visibility: visible;
 width=10px;
 heigth=10px;
 position: absolute;
 top: 1;
 left: 1;">
<IMG SRC='snow2.gif'>
</DIV>
```

```
<DIV     ID="el1"
 style = "visibility: visible;
 width=10px;
 heigth=10px;
 position: absolute;
 top: 1;
 left: 1;">
<IMG SRC='snow2.gif'>
</DIV>
<DIV     ID="el2"
 style = "visibility: visible;
 width=10px;
 heigth=10px;
 position: absolute;
 top: 1;
 left: 1;">
<IMG SRC='snow2.gif'>
</DIV>
<DIV     ID="el3"
 style = "visibility: visible;
 width=10px;
 heigth=10px;
 position: absolute;
 top: 1;
 left: 1;">
<IMG SRC='snow2.gif'>
</DIV>
<DIV     ID="el4"
 style = "visibility: visible;
 width=10px;
 heigth=10px;
 position: absolute;
 top: 1;
 left: 1;">
<IMG SRC='snow2.gif'>
</DIV>
<DIV     ID="el5"
 style = "visibility: visible;
 width=10px;
 heigth=10px;
 position: absolute;
 top: 1;
 left: 1;">
<IMG SRC='snow2.gif'>
</DIV>
<DIV     ID="el6"
 style = "visibility: visible;
 width=10px;
 heigth=10px;
 position: absolute;
 top: 1;
 left: 1;">
<IMG SRC='snow2.gif'>
</DIV>
<DIV     ID="el7"
 style = "visibility: visible;
 width=10px;
```

```
 heigth=10px;
 position: absolute;
 top: 1;
 left: 1;">
<IMG SRC='snow2.gif'>
</DIV>
<DIV     ID="el8"
 style = "visibility: visible;
 width=10px;
 heigth=10px;
 position: absolute;
 top: 1;
 left: 1;">
<IMG SRC='snow2.gif'>
</DIV>
<DIV     ID="el9"
 style = "visibility: visible;
 width=10px;
 heigth=10px;
 position: absolute;
 top: 1;
 left: 1;">
<IMG SRC='snow2.gif'>
</DIV>
<H1 align="center">Treść strony<H1>
</BODY>
</HTML>
```

Funkcja `ustaw()` odpowiada za zainicjowanie niezbędnych tablic i zmiennych. Dokonujemy tu losowania prędkości oraz pozycji *x*dla każdej warstwy. Ilość poziomów prędkości określana jest zmienną `speedLevels`. Za losowanie odpowiada więc prosta konstrukcja:

```
Math.ceil(Math.random() * speedLevels);
```

W podobny sposób losujemy pozycję warstwy na ekranie. Uwzględniamy tu jednak wielkość ekranu i szerokość warstw. Odpowiada za to konstrukcja:

```
var posX = Math.ceil(Math.random() * (document.body.clientWidth - 4*lWidth));
```

Przemieszczanie warstw jest realizowane w wywoływanej cyklicznie funkcji `start()`. Modyfikujemy właściwość `top` każdej warstwy, stosując przypisanie:

```
document.all['el' + i].style.top = x;
```

Skrypt 96. [E][O]

Spadające warstwy generowane dynamicznie.

Skrypt ten jest modyfikacją skryptu 95. Realizuje on taki sam efekt, jednak tym razem warstwy generowane są dynamicznie. Dzięki temu kod skryptu jest krótszy, pozbawia nas to jednak możliwości wpływania na wygląd każdej warstwy z osobna. W tym

przykładzie wszystkie one będą identyczne. Którą z wersji wybrać? Zależy to oczywiście od konkretnego zastosowania. Jeśli chcemy, aby każda z warstw miała swoje własne właściwości, wybierzemy kod ze skryptu 95, jeśli wszystkie mają być takie same, kod ze skryptu 96.

Skrypt 96

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speedTable = new Array();
var speed = 50;
var speedLevels = 5;
var layersCount = 10;
function ustaw(){
 for (i = 0; i < layersCount; i++){
        if (document.all){
               document.all['el' + i].style.left = Math.ceil(Math.random() *
               ➥(document.body.clientWidth - 50));
        }
        speedTable[i] = Math.ceil(Math.random() * speedLevels);
 }
}
function start(){
 for (i = 0; i < layersCount; i++){
        if (document.all){
               x = parseInt(document.all['el' + i].style.top);
               x += speedTable[i];
               if (x >= document.body.clientHeight - 45){
                      x = 1;
                      document.all['el' + i].style.left = Math.ceil(Math.random()
                      ➥* (document.body.clientWidth - 50));
                      speedTable[i] = Math.ceil(Math.random() * speedLevels);
               }
               document.all['el' + i].style.top = x;
        }
 }
 setTimeout('start();', speed);
}
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY onLoad='ustaw();start();'>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
for (i = 0; i < layersCount; i++){
 str = '<DIV    ID="el' + i + '" ';
 str += 'style = "visibility: visible; ';
 str += 'width=50px; ';
 str += 'heigth=45px; ';
 str += 'position: absolute; ';
 str += 'top: 1; ';
```

```
 str += 'left: 1;">';
 str += '<IMG SRC="snow2.gif">';
 str += '</DIV>';
 document.write(str);
}
//document.write('<H1 align="center">Treść strony<H1>');
// Koniec kodu JavaScript -->
</SCRIPT>
<H1 align="center">Treść strony<H1>
</BODY>
</HTML>
```

Skrypt 97. [E][O]

Płatki śniegu przemieszczające się w dwóch kierunkach.

Skrypt 95 pokazał nam, w jaki sposób wygenerować „spadające" z ekranu warstwy. Jako obrazów użyliśmy wtedy płatków śniegu. Jednak śnieg z reguły przemieszcza się również w kierunku poziomym. Aby więc efekt był bardziej realistyczny, powinniśmy dodać przemieszczanie warstw w poziomie.

Nie jest to oczywiście skomplikowana modyfikacja. Użyjemy dwóch tablic do określania prędkości warstw. Tablicy `speedTableY` dla przemieszczeń w pionie oraz `speedTableX` dla przemieszczeń w poziomie. W przypadku przemieszczania po osi x musimy dodatkowo wylosować kierunek ruchu każdej warstwy. Inaczej wszystkie płatki będą się poruszały tylko w jednym kierunku, co da bardzo nienaturalny efekt.

Ostatecznie do losowania użyjemy następującego kodu:

```
speedTableY[i] = Math.ceil(Math.random() * speedLevelsY);
var dir = (Math.random() < 0.5)?1:-1;
speedTableX[i] = dir * Math.ceil(Math.random() * speedLevelsX);
```

Oczywiście kod ten należy umieścić w pętli `for`, gdzie zmienną sterująca jest `i`.

Pozostało nam jeszcze podjąć decyzję, co zrobić w przypadku, kiedy dana warstwa osiągnie prawy bądź lewy brzeg ekranu. Najciekawiej będzie, jeśli w takiej sytuacji odbije się ona od tego brzegu i zacznie przesuwać się w przeciwnym kierunku. Efekt ten osiągniemy, zmieniając znak odpowiedniej wartości w tablicy `speedTableX`, czyli pisząc:

```
speedTableX[i] = -speedTableX[i];
```

Skrypt 97

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
```

```
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speedTableY = new Array();
var speedTableX = new Array();
var speed = 50;
var speedLevelsY = 5;
var speedLevelsX = 2;
var layersCount = 10;
function ustaw(){
 for (i = 0; i < layersCount; i++){
        if (document.all){
               document.all['el' + i].style.left = Math.ceil(Math.random() *
               ➥(document.body.clientWidth - 50));
        }
        speedTableY[i] = Math.ceil(Math.random() * speedLevelsY);
        var dir = (Math.random() < 0.5)?1:-1;
        speedTableX[i] = dir * Math.ceil(Math.random() * speedLevelsX);
 }
}
function start(){
 for (i = 0; i < layersCount; i++){
        if (document.all){
               x = parseInt(document.all['el' + i].style.left);
               x += speedTableX[i];
               if ((x > document.body.clientWidth - 60) || (x < 0)){
                      speedTableX[i] = -speedTableX[i];
               }
               document.all['el' + i].style.left = x;

               y = parseInt(document.all['el' + i].style.top);
               y += speedTableY[i];
               if (y >= document.body.clientHeight - 45){
               y = 1;
                      document.all['el' + i].style.left = Math.ceil(Math.random()
                      ➥* (document.body.clientWidth - 50));
                      speedTableY[i] = Math.ceil(Math.random() * speedLevelsY);
                      var dir = (Math.random() < 0.5)?1:-1;
                      speedTableX[i] = dir * Math.ceil(Math.random() *
                      ➥speedLevelsX);
               }
               document.all['el' + i].style.top = y;
        }
 }
 setTimeout('start();', speed);
}
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY onLoad='ustaw();start();'>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
for (i = 0; i < layersCount; i++){
 str = '<DIV    ID="el' + i + '" ';
 str += 'style = "visibility: visible; ';
 str += 'width=50px; ';
 str += 'heigth=45px; ';
 str += 'position: absolute; ';
 str += 'top: 1; ';
```

```
 str += 'left: 1;">';
 str += '<IMG SRC="snow2.gif">';
 str += '</DIV>';
 document.write(str);
}
// Koniec kodu JavaScript -->
</SCRIPT>
<H1 align="center">Treść strony<H1>
</BODY>
</HTML>
```

Skrypt 98. [E][O]

Realistycznie padający śnieg.

Skrypt ten generuje najbardziej realistycznie padający śnieg z dotychczas zaprezentowanych. Otrzymamy płatki bardzo płynnie poruszające się po ekranie i „samodzielnie" zmieniające kierunek ruchu. Osiągniemy to dzięki skorzystaniu z funkcji sinus. Zatem procedury realizujące ruch w pionie, tak jak we wszystkich poprzednich przykładach, pozostają bez zmian. Kolejne fazy ruchu w poziomie będą wyliczane ze wzoru:

```
newX = sin(b + (  / smooth)) * amplitude) + startX;
```

gdzie `b` jest parametrem oznaczającym przesunięcie danego płatka na sinusoidzie, natomiast `smooth` odpowiada za płynność ruchu. Dokładniej określa, ile faz ruchu zmieścić ma się w przedziale od 0 do Π. Wartość `apmlitude` wyznacza rozciągnięcie sinusoidy w poziomie, czyli to, jak bardzo dany płatek ma się odchylać w prawo i w lewo. `StartX` to pozycja początkowa danej warstwy.

Wzór ten po przełożeniu na kod powinien wyglądać następująco:

```
tabB[i] += Math.PI / smooth[i];
x = parseInt(Math.sin(tabB[i]) * amplitude[i]) + posX[i];
```

Zamiast parametrów występują tutaj tablice, jako że w każdym kroku generujemy pozycje oddzielnie dla każdej warstwy. W funkcji `ustaw()` będziemy wypełniać wszystkie tablice wartościami początkowymi w sposób następujący (oczywiście instrukcje te należy umieścić w pętli `for`):

```
posX[i] = Math.ceil(Math.random() * (document.body.clientWidth - 50));
speedTableY[i] = Math.ceil(Math.random() * speedLevelsY);
amplitude[i] = Math.ceil(Math.random() * 40 + 20);
smooth[i] = Math.ceil(Math.random() * 48) + 10;
tabB[i] = Math.random() * 4;
```

Kod ten nie musi być aż tak bardzo sparametryzowany w konkretnej realizacji, jednak dzięki temu każdy może uzyskać najlepsze według niego ustawienia opisujące ruch płatków. Wystarczy manipulować podanymi parametrami. Najlepiej po prostu poeksperymentować i dobrać je według własnego uznania.

Skrypt 98

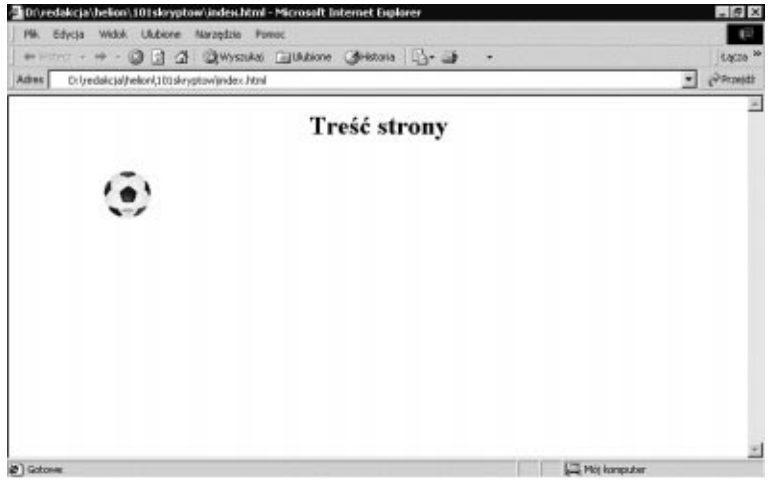
```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speedTableY = new Array();
var posX = new Array();
var smooth = new Array();
var amplitude = new Array();
var tabB = new Array();
var speed = 50;
var speedLevelsY = 5;
var layersCount = 10;
function ustaw(){
 for (i = 0; i < layersCount; i++){
         if (document.all){
               posX[i] = Math.ceil(Math.random() * (document.body.clientWidth - 50));
         }
         speedTableY[i] = Math.ceil(Math.random() * speedLevelsY);
         amplitude[i] = Math.ceil(Math.random() * 40 + 20);
         smooth[i] = Math.ceil(Math.random() * 48) + 10;
         tabB[i] = Math.random() * 4;
 }
}
var b = 0;
function start(){
 for (i = 0; i < layersCount; i++){
         if (document.all){
                y = parseInt(document.all['el' + i].style.top);
                y += speedTableY[i];

                tabB[i] += Math.PI / smooth[i];
                x = parseInt(Math.sin(tabB[i]) * amplitude[i]) + posX[i];
                if (x >= document.body.clientWidth - 60){
                        x = document.body.clientWidth - 60;
                }
                document.all['el' + i].style.left = x;

                if (y >= document.body.clientHeight - 45){
                        y = -50;
                        posX[i] = Math.ceil(Math.random() *
                        ➥(document.body.clientWidth - 50));
                        speedTableY[i] = Math.ceil(Math.random() * speedLevelsY);
                        amplitude[i] = Math.ceil(Math.random() * 100 + 20);
                        smooth[i] = Math.ceil(Math.random() * 48) + 48;
                }
                document.all['el' + i].style.top = y;
         }
 }
 setTimeout('start()', speed);
}
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
```

```
<BODY onLoad='ustaw();start();'>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
for (i = 0; i < layersCount; i++){
 str = '<DIV    ID="el' + i + '" ';
 str += 'style = "visibility: visible; ';
 str += 'width=50px; ';
 str += 'heigth=45px; ';
 str += 'position: absolute; ';
 str += 'top: 1; ';
 str += 'left: 1;">';
 str += '<IMG SRC="snow2.gif">';
 str += '</DIV>';
 document.write(str);
}
//document.write('<H1 align="center">Treść strony<H1>');
// Koniec kodu JavaScript -->
</SCRIPT>
<H1 align="center">Treść strony<H1>
</BODY>
</HTML>
```

Skrypt 99. [E][O]

Odbijająca się piłka.

Skrypt ten pozwala uzyskać efekt odbijającej się od dolnej krawędzi okna przeglądarki piłki. Piłka ta będzie oczywiście obrazkiem umieszczonym na oddzielnej warstwie (rysunek 7.11), którą będziemy odpowiednio przesuwać po ekranie.



Rysunek 7.11. *Piłka będzie odbijać się od dolnej krawędzi okna przeglądarki*

Aby efekt wyglądał realistycznie, musimy zasymulować ruch jednostajnie przyspieszony, kiedy obiekt spada, oraz jednostajnie opóźniony, kiedy wznosi się po odbiciu. Nie

będziemy stosować jednak żadnych skomplikowanych wzorów fizycznych. Zamiast tego będziemy zwiększać (lub zmniejszać) ilość pikseli, o jaką ma się przemieścić warstwa w każdej fazie ruchu. Za ten efekt odpowiadać będzie zmienna `speedStep`.

Musimy także pamiętać, że po każdym kolejnym odbiciu piłka nie może osiągać tej wysokości, od której zaczynał się ruch. Inaczej odbijałaby się w nieskończoność od obu brzegów ekranu. Kolejnym zadaniem jest więc regulacja wysokości, na jaką ma wracać animowana warstwa. Dokonujemy tego, przypisując odpowiednią wartość zmiennej `iter`. Konkretny wzór wygląda tu następująco:

```
nowa_wysokość = wysokość + wysokość / iter
```

Kierunek ruchu kontrolowany jest przez zmienną `direction`, która może przyjmować dwie wartości — `up` dla ruchu w górę i `down` dla ruchu w dół.

Nasza piłka porusza się również w poziomie. Ruch ten generowany jest przez cykliczne zwiększanie parametru `left` warstwy zawierającej obraz.

Skrypt 99

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speed = 20;
var speedStep = 1;
var ltop;
var bottom;
var direction = 'down';
var id;
var iter = 4;
function ustaw(){
 if (document.all){
         document.all['ball'].style.left = 0;
         ltop = 0;
         bottom = document.body.clientHeight - 50;
 }
}
y = 0;
x = 0;
function start(){
 x = parseInt(document.all['ball'].style.left);
 document.all['ball'].style.left = ++x;
 y = parseInt(document.all['ball'].style.top);
 if (y < bottom && direction == 'down'){
         y += speedStep++;
         if (y > bottom){
                document.all['ball'].style.top = bottom;
         }
         else{
                document.all['ball'].style.top = y;
         }
```

```
        id = setTimeout('start()', speed);
        return;
 }
 else if (y >= bottom && direction == 'down'){
        direction = 'up';
        ltop = ltop + Math.ceil((bottom - ltop) / iter);
        if (ltop >= bottom){
               clearTimeout(id);
               return;
        }
        id = setTimeout('start()', speed);
        return;
 }
 else if (y > ltop && direction == 'up'){
        y -= speedStep--;
        if (y < ltop){
               document.all['ball'].style.top = ltop;
        }
        else{
               document.all['ball'].style.top = y;
        }
        id = setTimeout('start()', speed);
        return;
 }
 else if (y <= ltop && direction == 'up'){
        direction = 'down';
        if (speedStep%3 == 0) speedStep--;
        id = setTimeout('start()', speed);
 }
}
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY onLoad='ustaw();start();'>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
for (i = 0; i < 1;i++){
 str = '<DIV    ID="ball" ';
 str += 'style = "visibility: visible; ';
 str += 'width=50px; ';
 str += 'heigth=45px; ';
 str += 'position: absolute; ';
 str += 'top: 1; ';
 str += 'left: 1;">';
 str += '<IMG SRC="ball.gif">';
 str += '</DIV>';
 document.write(str);
}
// Koniec kodu JavaScript -->
</SCRIPT>
<H1 align="center">Treść strony<H1>
</BODY>
</HTML>
```

Skrypt 100. [E][N6][O]

Piłka poruszająca się po sinusoidzie.

Funkcję sinus zna ze szkoły chyba każdy. Odpowiednie jej użycie pozwala na uzyskiwanie ładnych wizualnie efektów graficznych. W prosty sposób możemy sprawić, aby znana nam ze skryptu 99 piłka poruszała się w poziomie po torze sinusoidalnym. Oczywiście nie musi być to piłka. Zajmujemy się przecież animacją warstwy, a znajdujący się na niej obrazek może być dowolny.

Kolejne fazy ruchu generować będziemy na bieżąco, nie tablicując wyników. Za wyliczenia odpowiadać będzie następujący fragment kodu:

```
z += Math.PI / smooth;
y = Math.ceil((Math.sin(z) * amplitude)) + posTop;
```

Parametr `smooth` odpowiada za rozciągnięcie sinusoidy w poziomie, tzn. im mniejsza będzie ta wartość, tym tor ruchu będzie bardziej „zagęszczony", `amplitude` odpowiada za rozciągnięcie toru w pionie, a `posTop` to umiejscowienie sinusoidy na ekranie. Wszystkie te parametry można dobrać według własnego uznania.

Skrypt 100

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
var speed = 10;
var x = 0;
var amplitude = 100;
var posTop = 200;
var smooth = 100;
var sWidth;
var selector;
function ustaw(){
 selector = document.getElementById('ball').style;
 if (document.all){
         sWidth = document.body.clientWidth;
 }
 else{
         sWidth = document.width;
 }
 selector.left = 0;
}
var z = 0;
function start(){
 z += Math.PI / smooth;
 y = Math.ceil((Math.sin(z) * amplitude)) + posTop;
 x = parseInt(selector.left);
 if (x > sWidth - 50) x = 0;
```

```
 selector.left = ++x;
 selector.top = y;
 setTimeout("start()", speed);
}
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY onLoad='ustaw();start();'>
<SCRIPT LANGUAGE= "JavaScript" type= "text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
for (i = 0; i < 1;i++){
 str = '<DIV    ID="ball" ';
 str += 'style = "visibility: visible; ';
 str += 'width=50px; ';
 str += 'heigth=45px; ';
 str += 'position: absolute; ';
 str += 'top: 1; ';
 str += 'left: 1;">';
 str += '<IMG SRC="ball.gif">';
 str += '</DIV>';
 document.write(str);
}
// Koniec kodu JavaScript -->
</SCRIPT>
<H1 align="center">Treść strony<H1>
</BODY>
</HTML>
```

Skrypt 101 [E][N][O]

Pływające warstwy.

Umiemy już poruszać jedną warstwę po sinusoidzie, możemy pokusić się zatem o stworzenie napisu, którego litery będą przemieszczały się w górę i w dół, tworząc sinusoidę? Da to bardzo ładny efekt wizualny, który osiągniemy stosunkowo prostym sposobem. Dla każdej litery zdefiniujemy oddzielna warstwę, tak by można było nimi niezależnie poruszać. Napis będzie sześcioliterowy, zdefiniujemy wiec sześć warstw. Przy czym, ponieważ większość definicji będzie taka sama, nie będziemy ich powtarzać, lecz zrobimy to w sposób następujący:

```
<STYLE>
#warstwa1, #warstwa2, #warstwa3, #warstwa4, #warstwa5, #warstwa6{
 visibility:visible;
 background-color:red;
 position:absolute;
 width:50;
 height:50;
 layer-background-color:red;
}
```

Zaoszczędzamy w ten sposób sporo miejsca. Litery będą się poruszały w pionie, musimy więc wygenerować kolejne położenia. Moglibyśmy to robić przy każdym przebiegu animacji, ale wygodniej będzie zrobić to raz, a wyniki zapisać w tablicy. Nasze 6 liter rozmieszczamy równomiernie na sinusoidzie w przedziale od 0 do 2Π. Wygenerujemy dla każdej z nich sześć pozycji, zatem kolejne pozycje będą od siebie odległe o *2*Π */ 6*, czyli Π */ 3*. Obliczenie wykona za nas funkcja `init()` w postaci:

```
function init(){
 tab = new Array();
 x = 0;
 j = 0;
 for(i = 0; i < 6; i++){
        tab[j++] = Math.sin(x) * 25 + 100;
        x += Math.PI / 3;
 }
}
```

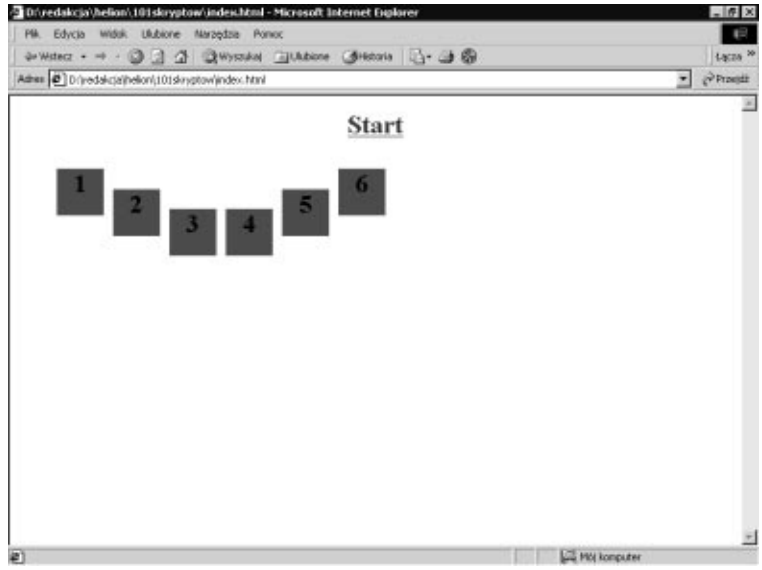
Teraz musimy zająć się animacją. W pierwszym ruchu warstwa 1. powinna przyjąć pozycję zapisaną w tablicy w miejscu o indeksie 0, w drugim ruchu o indeksie 1 itd. Po dojściu do indeksu 6 wszystko powinno zacząć się od nowa. Analogicznie warstwa druga zaczyna od indeksu 1, warstwa 3 od indeksu 2 itd. Jednak rozpisanie tego w ten sposób byłoby bardzo niewygodne. Zdecydowanie lepszym pomysłem jest przypisywanie każdej warstwie cały czas jednej komórki tablicy, tzn. warstwie 1. komórki 0, warstwie 2. komórki 1 itd. Natomiast w każdym ruchu przesuwać należy zawartość komórek samej tablicy. Dzięki temu funkcja `przesun()` będzie miała bardzo prosty kod:

```
function przesun(){
 x = tab[0];
 for (i = 0; i < 5; i++){
        tab[i] = tab[i + 1];
 }
 tab[5] = x;
}
```

Funkcja ustawiania warstw wyglądać będzie natomiast następująco:

```
function ustawWarstwy(){
 if (document.layers){
        document.layers['warstwa1'].top = tab[0];
        document.layers['warstwa2'].top = tab[1];
        document.layers['warstwa3'].top = tab[2];
        document.layers['warstwa4'].top = tab[3];
        document.layers['warstwa5'].top = tab[4];
        document.layers['warstwa6'].top = tab[5];
 }
 else{
        document.getElementById('warstwa1').style.top = tab[0];
        document.getElementById('warstwa2').style.top = tab[1];
        document.getElementById('warstwa3').style.top = tab[2];
        document.getElementById('warstwa4').style.top = tab[3];
        document.getElementById('warstwa5').style.top = tab[4];
        document.getElementById('warstwa6').style.top = tab[5];
 }
}
```

Ja widać, nie jest to nic bardzo skomplikowanego. Kod w warunku `if(document.layers)` zapewnia nam obsługę przeglądarki Netscape Navigator w wersji 4.5 – 4.7. Kod po `else` będzie rozpoznawany przez Explorera, Operę i Navigatora w wersji 6. Teraz już tylko komponujemy wszystko w kod strony WWW i nasz napis tworzy animowaną sinusoidę (rysunek 7.12).



Rysunek 7.12. *Warstwy poruszające się po sinusoidzie*

Skrypt 101

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis    e-mail: 101scripts@marcinlis.com  -->
</HEAD>
<STYLE>
#warstwa1, #warstwa2, #warstwa3, #warstwa4, #warstwa5, #warstwa6{
 visibility:visible;
 background-color:red;
 position:absolute;
 width:50;
 height:50;
 layer-background-color:red;
}
</STYLE>
<SCRIPT LANGUAGE="javascript" type="text/javascript">
<!-- Ukrycie przed przeglądarkami nie obsługującymi JavaScriptu
function init(){
 tab = new Array();
 x = 0;
 j = 0;
 for(i = 0; i < 6; i++){
         tab[j++] = Math.sin(x) * 25 + 100;
         x += Math.PI / 3;
 }
```

```
}
function ustawWarstwy(){
 if (document.layers){
         document.layers['warstwa1'].top = tab[0];
         document.layers['warstwa2'].top = tab[1];
         document.layers['warstwa3'].top = tab[2];
         document.layers['warstwa4'].top = tab[3];
         document.layers['warstwa5'].top = tab[4];
         document.layers['warstwa6'].top = tab[5];
 }
 else{
         document.getElementById('warstwa1').style.top = tab[0];
         document.getElementById('warstwa2').style.top = tab[1];
         document.getElementById('warstwa3').style.top = tab[2];
         document.getElementById('warstwa4').style.top = tab[3];
         document.getElementById('warstwa5').style.top = tab[4];
         document.getElementById('warstwa6').style.top = tab[5];
 }
}
function przesun(){
 x = tab[0];
 for (i = 0; i < 5; i++){
         tab[i] = tab[i + 1];
 }
 tab[5] = x;
}
function start(){
 setTimeout("start()", 200);
 ustawWarstwy();
 przesun();
}
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY onLoad="init()">
<H1 align = "center">
<DIV id="warstwa1" style="top:50; left:50;">
1
</DIV>
<DIV id="warstwa2" style="top:50; left:110; ">
2
</DIV>
<DIV id="warstwa3" style="top:50; left:170; ">
3
</DIV>
<DIV id="warstwa4" style="top:50; left:230; ">
4
</DIV>
<DIV id="warstwa5" style="top:50; left:290; ">
5
</DIV>
<DIV id="warstwa6" style="top:50; left:350; ">
6
</DIV>
<A HREF="javascript:start()">Start</A>
</H1>
</BODY>
</HTML>
```