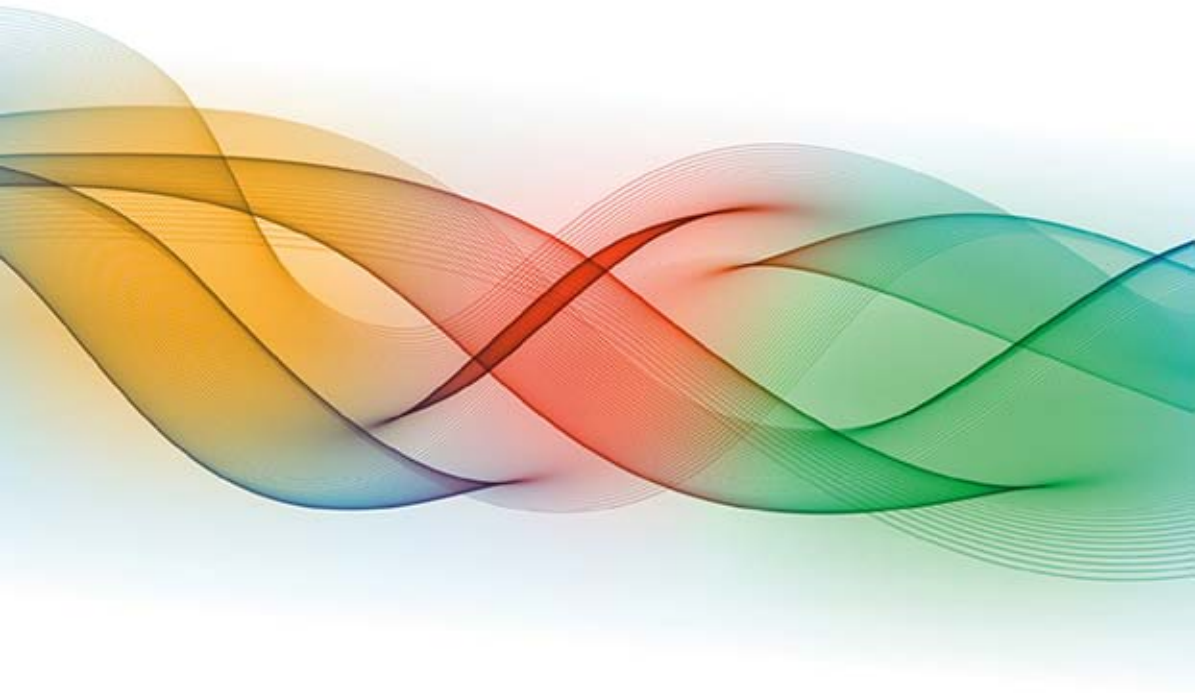
Helion

Dawid Borycki

JavaScript i jQuery

Kompletny przewodnik dla programistów interaktywnych aplikacji internetowych w Visual Studio



JavaScript i jQuery — fantastyczny duet w każdej aplikacji!

Poznaj technologię JavaScript i wykorzystaj jej kolosalne możliwości
Ułatw sobie pracę za pomocą elastycznych narzędzi z biblioteki jQuery i jQuery UI

Redaktor prowadzący: Ewelina Burska
Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki
Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/jsiqkp
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-8283-6

Printed in Poland.

Spis treści

Rozdział 10.

Animacje

Wprowadzenie

JavaScript w połączeniu z kaskadowymi arkuszami stylów pozwala uatrakcyjnić projektowaną witrynę efektami animacji. Dzięki temu poprawi się interaktywność całej aplikacji internetowej, która ułatwi użytkownikowi jej obsługę.

W praktyce implementowanie animacji elementów drzewa DOM sprowadza się do umiejętnego modyfikowania właściwości `style` wybranego obiektu HTML. Ponieważ preferencje estetyczne są kwestią indywidualnego wyboru, więc w tym podrozdziale pokażę, w jaki sposób wykonać proste efekty animacji, takie jak: animacje wejścia i wyjścia (rozjaśnianie i ściemnanie) obiektów, zmiana ich rozmiarów i położenia. Moim celem nie będzie utworzenie pięknych animacji, a jedynie przedstawienie podstaw niezbędnych do samodzielnego implementowania własnych animacji.

Ściemnianie i rozjaśnianie

Dynamiczna kontrola poziomu jasności wybranego elementu HTML sprowadza się do modyfikacji właściwości `opacity`, którą w zależności od przeglądarki internetowej można zmodyfikować za pomocą właściwości `style.filter` w przypadku Internet Explorera lub właściwości `style.opacity` w przypadku pozostałych przeglądarek.

Zanim przejdę do zaimplementowania funkcji, której celem będzie animacja wybranego elementu HTML poprzez ciągłą zmianę poziomu jego jasności, pokażę, w jaki sposób zmienić jego właściwość `opacity`. W tym celu:

1. Utwórz nową witrynę HTML o nazwie *Animacje.htm*.
2. W sekcji `<title>` witryny wpisz jej nazwę, czyli `Animacje`.

3. Korzystając z listingu 10.1, uzupełnij nagłówek strony *Animacje* o definicję klas kaskadowych arkuszy stylów dla bloku `div`.

Listing 10.1. *Definicja formatowania bloku div*

```
<style type="text/css">
    div.zolty
    {
        color: #996600;
        font-weight: bold;
        background-color: #FFFFCC;
        border-color: #996600;
        border-style: solid;
        border-width: 2px;
        width: 100px;
        text-align: center;
    }

    div.brazowy
    {
        color: #FFFFCC;
        font-weight: bold;
        background-color: #996600;
        border-color: #FFFFCC;
        border-style: solid;
        border-width: 1px;
        width: 100px;
        text-align: center;
    }
</style>
```

4. W nagłówku witryny *Animacje* umieść polecenia z listingu 10.2.

Listing 10.2. *Kontrola poziomu jasności wybranego elementu HTML za pomocą JavaScript*

```
function ustawJasnosc(idElementu, poziomJasnosci) {
    var element = document.getElementById(idElementu);

    if(!element) return;

    element.style.opacity = poziomJasnosci / 10;
    element.style.filter = 'alpha(opacity=' + poziomJasnosci * 10 + ')';
}

function odczytajJasnosc(idElementu) {
    var element = document.getElementById(idElementu);

    if(!element) return;

    var jasnosc;
    if (element.style.filter) {
        var indeksP = element.style.filter.indexOf('=');
        var indeksK = element.style.filter.indexOf(')');

        jasnosc = parseFloat(element.style.filter.substring(indeksP + 1,
        ➥indeksK)) / 10;
    }
```

```
    else
        jasnosc = element.style.opacity * 10;

    return jasnosc;
}
```

5. Sekcję <body> witryny *Animacje* zdefiniuj według wzoru z listingu 10.3.

Listing 10.3. *Zdarzenia onmouseover oraz onmouseout a poziom jasności bloku div*

```
<body onload="ustawJasnosc('blok1', 10);">
    <div id="blok1" class="zolty" onmouseover="ustawJasnosc('blok1', 5);"
    ➥onmouseout="ustawJasnosc('blok1', 10)">Wydawnictwo Helion</div>
</body>
```

Dwa aspekty powyższego rozwiązania zasługują na szczególną uwagę. Są nimi funkcje `ustawJasnosc` oraz `odczytajJasnosc`. Pierwsza z nich pozwala zmodyfikować poziom jasności wybranego fragmentu witryny internetowej poprzez kontrolę właściwości `opacity`. Funkcja ta jest kompatybilna z różnymi przeglądarkami i pozwala ustawić jasność obiektów HTML na wartość z zakresu od `0`, dla której dany obiekt będzie całkowicie przezroczysty (niewidoczny), do `10`, dla której dany obiekt nie będzie przezroczysty.

W zależności od rodzaju przeglądarki internetowej właściwość `opacity` można zmodyfikować za pomocą pola `style.opacity`, do którego należy po prostu przypisać odpowiednią wartość, reprezentującą poziom przezroczystości obiektu HTML. Właściwość `style.opacity` może przyjmować wartości z zakresu od `0` do `1`. Z tego powodu wartość parametru przekazanego do funkcji `ustawJasnosc` jest dzielona przez współczynnik równy `10`. W przypadku przeglądarki Internet Explorer przezroczystość obiektu HTML można zmodyfikować za pomocą właściwości `style.filter`, do której należy przypisać łańcuch tekstowy w postaci `"alpha(opacity=jasność)"`, gdzie parametr `jasność` jest wyrażony w procentach i odpowiada procentowemu poziomowi przezroczystości danego elementu HTML.

Ze względu na fakt, że sposób kontroli jasności (przezroczystości) obiektów HTML zależy od wykorzystywanej przeglądarki internetowej, również i odczytanie poziomu przezroczystości danego obiektu jest uzależnione od rodzaju przeglądarki. Wobec tego funkcja `odczytajJasnosc` z listingu 10.2 najpierw weryfikuje, czy własność `filter` obiektu `style` została wcześniej zdefiniowana. Jeśli tak, to pobieram wartość jasności znajdującą się pomiędzy znakami `=` oraz `)`. W przeciwnym wypadku odczytuję własność `style.opacity`.

Po otwarciu witryny *Animacje.htm* w domyślnej przeglądarce internetowej i przesunięciu kursora myszy w obrębie napisu *Wydawnictwo Helion* nastąpi jego ściemnienie. Po usunięciu kursora myszy z regionu wyznaczonego przez ten napis poziom jego przezroczystości zostanie przywrócony do 100%.

W kolejnym przykładzie zmodyfikuję kod witryny *Animacje.htm* w taki sposób, aby zmiana jasności napisu *Wydawnictwo Helion* realizowana była w sposób ciągły. Wymaga to dokonania następujących czynności:

1. Skrypt witryny `Animacje.htm` uzupełnij o polecenia z listingu 10.4.

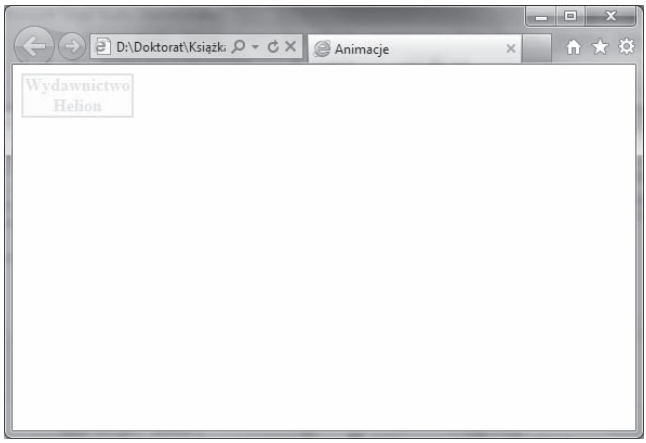
Listing 10.4. *Animowanie zmian jasności obiektu HTML*

```
var animacjaZmianyJasnosci;

function animujJasnosc(idElementu, sciemnianie, jasnoscDocelowa,
➥krokZmianyJasnosci) {
    var element = document.getElementById(idElementu);

    if (!element) return;

    var aktualnaJasnoscElementu = odczytajJasnosc(idElementu);
    var nowaJasnosc;

    if (sciemnianie) {
        nowaJasnosc = aktualnaJasnoscElementu - krokZmianyJasnosci;
        if (nowaJasnosc < jasnoscDocelowa) nowaJasnosc = jasnoscDocelowa;
    }
    else {
        nowaJasnosc = aktualnaJasnoscElementu + krokZmianyJasnosci;
        if (nowaJasnosc > jasnoscDocelowa) nowaJasnosc = jasnoscDocelowa;
    }

    ustawJasnosc(idElementu, nowaJasnosc);

    // Czy docelowy poziom jasności został już osiągnięty?
    if (nowaJasnosc != jasnoscDocelowa) {
    // Jeśli nie, to animuj dalej...

        animacjaZmianyJasnosci = setTimeout("animujJasnosc('" + idElementu
                    + "', " + sciemnianie
                    + ", " + jasnoscDocelowa
                    + ", " + krokZmianyJasnosci
                    + ");", 50);
    }
    else {
        // W przeciwnym wypadku przerwij animację
        zatrzymajAnimacje(animacjaZmianyJasnosci);
    }
}

function zatrzymajAnimacje(animacja) {
    clearTimeout(animacja);
}
```

2. Definicję sekcji `<body>` witryny zmodyfikuj według następującego wzoru:

```
<body onload="ustawJasnosc('blok1', 10);">
    <div id="blok1" class="zolty" onmouseover="ustawJasnosc('blok1', 5);"
    ➥onmouseout="ustawJasnosc('blok1', 10)">Wydawnictwo Helion</div>
    <div id="blok1" class="zolty" onmouseover="zatrzymajAnimacje(
    ➥animacjaZmianyJasnosci);animujJasnosc('blok1', true, 2, 1);"
    ➥onmouseout="zatrzymajAnimacje(animacjaZmianyJasnosci);animujJasnosc(
    ➥'blok1', false, 10, 1);">Wydawnictwo Helion</div>
</body>
```

Po zaimplementowaniu powyższych zmian w witrynie `Animacje.htm` należy ją otworzyć w domyślnej przeglądarce internetowej. Umieszczenie kursora myszy ponad napisem *Wydawnictwo Helion* spowoduje wyzwolenie animacji zmieniającej jego jasność. Efekt końcowy powinien być analogiczny do przedstawionego na rysunku 10.1. Przesunięcie kursora myszy w obszarze zajmowanym przez napis *Wydawnictwo Helion* powoduje wyzwolenie kodu JavaScript związanego ze zdarzeniem `mouseover`, czyli wywołanie metod `zatrzymajAnimacje` oraz `animujJasnosc`. Pierwsza z nich powoduje zatrzymanie wcześniej wykonywanej animacji i rozpoczęcie nowej. Ciągła zmiana poziomu jasności wybranego elementu HTML została zaimplementowana w funkcji `animujJasnosc`, której działanie sprowadza się do odczytania bieżącego poziomu jasności danego obiektu HTML. W następnym kroku, w zależności od wartości parametru `sciemnianie`, zmniejszam lub odpowiednio zwiększam nowy poziom jasności. Jeśli ten nowy poziom odpowiada wartości docelowej, to animacja zostaje zakończona. W przeciwnym wypadku następuje ponowne wywołanie funkcji `animujJasnosc` przy użyciu znanego już mechanizmu, wykorzystującego funkcję `setTimeout`.



Rysunek 10.1. *Efekt końcowy animacji typu ściemnianie*

Dzięki temu, że funkcja `animujJasnosc` jest napisana stosunkowo ogólnie, można ją również wykorzystać zarówno do ściemniania, jak i rozjaśniania wybranego obiektu HTML. Z tego powodu funkcję `animujJasnosc` skojarzyłem ze zdarzeniem `mouseout`, z tą tylko różnicą, że w przypadku rozjaśniania obiektu poziomem docelowym jasności jest jasność maksymalna, czyli `10`. Wobec tego usunięcie kursora myszy z obszaru wyznaczonego przez napis *Wydawnictwo Helion* spowoduje jego rozjaśnienie.

Zmiana rozmiaru

W tym podrozdziale zaimplementuję animację zmiany rozmiaru wybranego elementu HTML. Realizację tego zadania można wykonać poprzez analogię do przykładu przedstawionego w poprzednim podrozdziale. Wobec tego wykonaj następujące kroki:

1. Sekcję `<body>` witryny *Animacje.htm* uzupełnij o następujące polecenia:

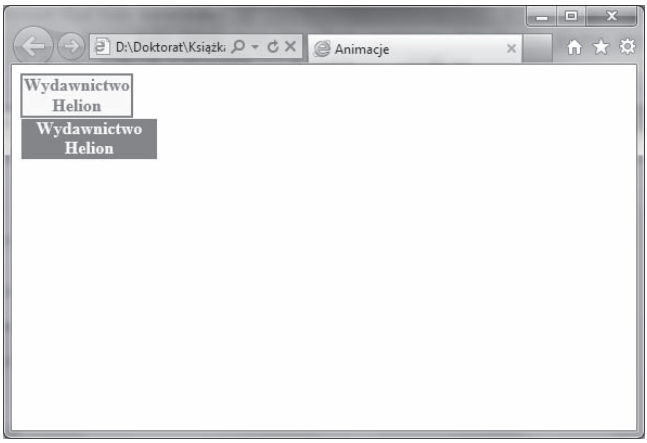
```
<div id="blok2" class="brazowy" onmouseover="animujSzerokosc('blok2', true,
➥100, 125, 1);" onmouseout="animujSzerokosc('blok2', false,
➥parseInt(this.style.width), 100, 1);">Wydawnictwo Helion</div>
```

2. W nagłówku witryny przejdź do sekcji `<script>` i umieść w niej definicję funkcji z listingu 10.5.

Listing 10.5. *Animacja zmiany szerokości wybranego elementu HTML*

```
function animujSzerokosc(idElementu, rozszerzanie, szerokoscPoczatkowa,
➥szerokoscDocelowa, krokZmianySzerokosci) {
    var element = document.getElementById(idElementu);

    if (!element) return;

    var aktualnaSzerokoscElementu =
        element.style.width ? parseInt(element.style.width) : szerokoscPoczatkowa;
    var nowaSzerokosc;

    if (rozszerzanie) {
        nowaSzerokosc = aktualnaSzerokoscElementu + krokZmianySzerokosci;
        if (nowaSzerokosc > szerokoscDocelowa) nowaSzerokosc = szerokoscDocelowa;

    }
    else {
        nowaSzerokosc = aktualnaSzerokoscElementu - krokZmianySzerokosci;
        if (nowaSzerokosc < szerokoscDocelowa) nowaSzerokosc = szerokoscDocelowa;
    }

    element.style.width = nowaSzerokosc + "px";

    // Czy docelowa szerokość została już osiągnięta?
    if (nowaSzerokosc != szerokoscDocelowa) {
        // Jeśli nie, to animuj dalej...
        element.animacjaZmianySzerokosci = setTimeout("animujSzerokosc('" +
        ➥idElementu
                    + "', " + rozszerzanie
                    + ", " + szerokoscPoczatkowa
                    + ", " + szerokoscDocelowa
                    + ", " + krokZmianySzerokosci
                    + ");", 10);
    }
    else {
        // W przeciwnym wypadku przerwij animację
        zatrzymajAnimacje(element.animacjaZmianySzerokosci);
    }
}
```

Konstrukcja powyższego przykładu jest bardzo podobna do animacji przezroczystości obiektów HTML. Jednakże zamiast właściwości `style.opacity` modyfikuję wartość zapisaną w polu `style.width`. Dzięki temu po umieszczeniu kursora myszy w obrębie wyznaczonym przez napis *Wydawnictwo Helion* uzyskuje się efekt analogiczny do przedstawionego na rysunku 10.2.



Rysunek 10.2. *Efekt końcowy animacji zmiany szerokości elementu HTML*

Animacja zmiany rozmiarów elementów HTML znajduje zastosowanie w wyróżnianiu aktywnych (aktualnie zaznaczonych) elementów menu aplikacji internetowej. W celu zasymulowania takiego menu trzykrotnie powielę brązowy blok `div` zawierający napis *Wydawnictwo Helion*. Po tej operacji sekcja `<body>` witryny *Animacje.htm* powinna mieć następującą postać:

```
<body onload="ustawJasnosc('blok1', 10);">
    <!--<div id="blok1" class="zolty" onmouseover="ustawJasnosc('blok1', 5);"
    ➥onmouseout="ustawJasnosc('blok1', 10)">Wydawnictwo Helion</div>-->
    <div id="blok1" class="zolty" onmouseover="zatrzymajAnimacje(
    ➥animacjaZmianyJasnosci);animujJasnosc('blok1', true, 2, 1);"
    ➥onmouseout="zatrzymajAnimacje(animacjaZmianyJasnosci);animujJasnosc('blok1',
    ➥false, 10, 1);">Wydawnictwo Helion</div>
    <div id="blok2" class="brazowy" onmouseover="zatrzymajAnimacje(
    ➥animacjaZmianySzerokosci);animujSzerokosc('blok2', true, 100, 125, 1);"
    ➥onmouseout="zatrzymajAnimacje(animacjaZmianySzerokosci);animujSzerokosc(
    ➥'blok2', false, parseInt(this.style.width), 100, 1);">Wydawnictwo
    ➥Helion</div>
    <div id="blok3" class="brazowy" onmouseover="zatrzymajAnimacje(
    ➥animacjaZmianySzerokosci);animujSzerokosc('blok3', true, 100, 125, 1);"
    ➥onmouseout="zatrzymajAnimacje(animacjaZmianySzerokosci);animujSzerokosc(
    ➥'blok3', false, parseInt(this.style.width), 100, 1);">Wydawnictwo
    ➥Helion</div>
    <div id="blok4" class="brazowy" onmouseover="zatrzymajAnimacje(
    ➥animacjaZmianySzerokosci);animujSzerokosc('blok4', true, 100, 125, 1);"
    ➥onmouseout="zatrzymajAnimacje(animacjaZmianySzerokosci);animujSzerokosc(
    ➥'blok4', false, parseInt(this.style.width), 100, 1);">Wydawnictwo
    ➥Helion</div>
    <div id="blok5" class="brazowy" onmouseover="zatrzymajAnimacje(
    ➥animacjaZmianySzerokosci);animujSzerokosc('blok5', true, 100, 125, 1);"
    ➥onmouseout="zatrzymajAnimacje(animacjaZmianySzerokosci);animujSzerokosc(
    ➥'blok5', false, parseInt(this.style.width), 100, 1);">Wydawnictwo
    ➥Helion</div>
</body>
```

Po otwarciu tak zmodyfikowanej witryny w domyślnej przeglądarce internetowej i przesuwaniu kursora myszy po brązowych elementach HTML (rysunek 10.2) nietrudno stwierdzić, że efekt działania aplikacji jest daleki od zamierzonego. Poszczególne efekty

animacji są zatrzymywane i uruchamiane w sposób chaotyczny. Dzieje się tak dlatego, że każdorazowe przesunięcie kursora myszy z jednego obiektu HTML na inny zatrzymuje animację danego elementu i uruchamia animację kolejnego obiektu HTML. Powinniśmy zapewnić, aby efekt animacji wszystkich elementów wykonywany był niezależnie. Wobec tego funkcję `animujSzerokosc` należy zmodyfikować według wzoru z listingu 10.6. Natomiast w sekcji `<body>` witryny *Animacje.htm* należy usunąć wywołania funkcji `zatrzymajAnimacje`:

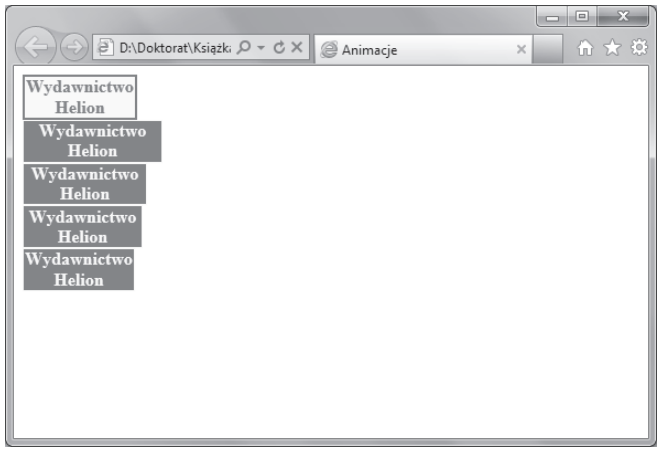
```
<body onload="ustawJasnosc('blok1', 10);">
    <!--<div id="blok1" class="zolty" onmouseover="ustawJasnosc('blok1', 5);"
    ➥onmouseout="ustawJasnosc('blok1', 10)">Wydawnictwo Helion</div>-->
    <div id="blok1" class="zolty" onmouseover="zatrzymajAnimacje(
    ➥animacjaZmianyJasnosci);animujJasnosc('blok1', true, 2, 1);"
    ➥onmouseout="zatrzymajAnimacje(animacjaZmianyJasnosci);animujJasnosc('blok1',
    ➥false, 10, 1);">Wydawnictwo Helion</div>
    <div id="blok2" class="brazowy" onmouseover="animujSzerokosc('blok2', true,
    ➥100, 125, 1);" onmouseout="animujSzerokosc('blok2', false,
    ➥parseInt(this.style.width), 100, 1);">Wydawnictwo Helion</div>
    <div id="blok3" class="brazowy" onmouseover="animujSzerokosc('blok3', true,
    ➥100, 125, 1);" onmouseout="animujSzerokosc('blok3', false,
    ➥parseInt(this.style.width), 100, 1);">Wydawnictwo Helion</div>
    <div id="blok4" class="brazowy" onmouseover="animujSzerokosc('blok4', true,
    ➥100, 125, 1);" onmouseout="animujSzerokosc('blok4', false,
    ➥parseInt(this.style.width), 100, 1);">Wydawnictwo Helion</div>
    <div id="blok5" class="brazowy" onmouseover="animujSzerokosc('blok5', true,
    ➥100, 125, 1);" onmouseout="animujSzerokosc('blok5', false,
    ➥parseInt(this.style.width), 100, 1);">Wydawnictwo Helion</div>
</body>
```

Listing 10.6. *Dynamiczna modyfikacja definicji obiektu*

```
~~var animacjaZmianySzerokosci;~~

function animujSzerokosc(idElementu, rozszerzanie, szerokoscPoczatkowa,
➥szerokoscDocelowa, krokZmianySzerokosci) {
    var element = document.getElementById(idElementu);

    if (!element) return;

    if (element.animacjaZmianySzerokosci)
        zatrzymajAnimacje(element.animacjaZmianySzerokosci);

    var aktualnaSzerokoscElementu =
        element.style.width ? parseInt(element.style.width) : szerokoscPoczatkowa;
    var nowaSzerokosc;

    if (rozszerzanie) {
        nowaSzerokosc = aktualnaSzerokoscElementu + krokZmianySzerokosci;
        if (nowaSzerokosc > szerokoscDocelowa) nowaSzerokosc = szerokoscDocelowa;

    }
    else {
        nowaSzerokosc = aktualnaSzerokoscElementu - krokZmianySzerokosci;
        if (nowaSzerokosc < szerokoscDocelowa) nowaSzerokosc = szerokoscDocelowa;
    }

    element.style.width = nowaSzerokosc + "px";
```

```
        // Czy docelowa szerokość została już osiągnięta?
        if (nowaSzerokosc != szerokoscDocelowa) {
            // Jeśli nie, to animuj dalej...
            element.animacjaZmianySzerokosci = setTimeout("animujSzerokosc('" + idElementu
                        + "', " + rozszerzanie
                        + ", " + szerokoscPoczatkowa
                        + ", " + szerokoscDocelowa
                        + ", " + krokZmianySzerokosci
                        + ");", 10);
        }
        else {
            // W przeciwnym wypadku przerwij animację
            zatrzymajAnimacje(element.animacjaZmianySzerokosci);
        }
    }
```

JavaScript jest językiem dynamicznym, co umożliwia dynamiczną zmianę definicji obiektów w trakcie interpretacji poleceń. Wykorzystałem tę właściwość do uzupełnienia definicji animowanych obiektów o pole `animacjaZmianySzerokosci`, które służy do przechowania informacji o tym, czy dany element jest aktualnie animowany. Dzięki temu witryna działa poprawnie (rysunek 10.3).



Rysunek 10.3. *Animacja zmiany rozmiaru działa teraz niezależnie dla każdego elementu HTML. Dzięki dynamicznemu charakterowi języka JavaScript informacja o aktualnym stanie animacji została zapisana w definicji odpowiedniego obiektu HTML*

Ruch

Projekt animacji elementów HTML rozpocznę od zaimplementowania funkcji, która „przesuwa” wybrane obiekty drzewa DOM. Innymi słowy, projektowana procedura będzie modyfikowała współrzędne określające położenie tych elementów. Do tego celu wykorzystam właściwości `style.position`, `style.left` oraz `style.top`. Pierwsza z nich posłuży mi do skonfigurowania sposobu pozycjonowania wybranego obiektu HTML na względny (ang. *relative*). Dzięki temu dany element drzewa DOM będzie mógł zostać przesunięty w dowolne miejsce witryny względem swojej pozycji początkowej.

Właściwości `style.left` oraz `style.top` pozwalają zdefiniować współrzędne (w pikselach) określające odległość od lewego górnego narożnika witryny do lewego górnego narożnika wybranego obiektu HTML. Należy pamiętać, że wartości `style.left` oraz `style.top` i wszystkie pozostałe służące do konfiguracji współrzędnych wykorzystują komputerowy układ współrzędnych, którego początek znajduje się w lewym górnym rogu ekranu. Kierunek wzrostu odciętych pozostaje niezmieniony, natomiast wartości rzędnych rosną wraz z rosnącą odległością od górnej krawędzi ekranu.

Biorąc pod uwagę te ogólne zasady, definicja funkcji, której zadaniem jest zmiana położenia wybranego elementu drzewa DOM, może mieć postać przedstawioną na listingu 10.7. W celu zaprezentowania działania tej procedury powiążę ją ze zdarzeniem `onmouseover` bloku o identyfikatorze `blok6` (listing 10.8). Po umieszczeniu kursora myszy w obrębie tego bloku zostanie on przesunięty o wektor [200 px, 200 px], czyli do punktu o współrzędnych:

$$x_n = x + 200\ [px]$$

$$y_n = y + 200\ [px]$$

gdzie punkt o współrzędnych (x, y) jest pozycją początkową bloku o identyfikatorze `blok6`, którą w JavaScript można odczytać za pomocą własności `clientWidth` oraz `clientHeight`.

Listing 10.7. *Zmiana położenia wybranego elementu drzewa DOM*

```
function przesunElement(idElementu, x, y) {
    var element = document.getElementById(idElementu);

    if (!element) return;

    var szerokosc = parseInt(element.clientWidth);
    var wysokosc = parseInt(element.clientHeight);

    // Czy nowe współrzędne nie spowodują wysunięcia obiektu poza dostępny obszar?
    // Jeśli tak, to skoryguj nowe współrzędne
    if (x + szerokosc > window.innerWidth)
        x = window.innerWidth - szerokosc;

    if (y + wysokosc > window.innerHeight)
        y = window.innerHeight - wysokosc;

    element.style.position = "absolute";
    element.style.left = x + "px";
    element.style.top = y + "px";
}
```

Listing 10.8. *Umieszczenie kursora myszy w obrębie obszaru zajmowanego przez obiekt blok6 spowoduje jego przesunięcie o wektor [200 px, 200 px]*

```
<div id="blok6" class="zolty" onmouseover="przesunElement('blok6', 200,
➥200)">Wydawnictwo Helion</div>
```

Na podstawie przedstawionej powyżej metody przemieszczania obiektów drzewa DOM można implementować procedury, których celem jest animacja ruchu obiektów HTML. W moim przykładzie wybrany element drzewa DOM będzie poruszał się równolegle do osi odciętych. Ruch będzie wykonywany cyklicznie do momentu jego zatrzymania. Do tego celu wykorzystam metodę `przesunElement` oraz zestaw funkcji `setTimeout` i `clearTimeout`:

1. Uzupełnij skrypt witryny *Animacje.htm* o definicję funkcji `ruchPoziomy` z listingu 10.9.

Listing 10.9. *Implementacja animacji ruchu wybranego elementu drzewa DOM wzdłuż osi odciętych*

```
function ruchPoziomy(idElementu, poczatek, koniec, krok) {
    var element = document.getElementById(idElementu);

    if (!element) return;

    // Korekcja parametrów początkowych
    var szerokoscElementu = element.clientWidth;

    if (poczatek < 0) poczatek = 0;
    if (poczatek > koniec) poczatek = koniec;

    if (koniec + szerokoscElementu > window.innerWidth)
        koniec = window.innerWidth - szerokoscElementu;

    if (koniec < poczatek) koniec = poczatek;

    if (!element.aktualnaPozycja)
        element.aktualnaPozycja = element.clientLeft;

    // Wartość zmiennej nowaPozycja zależy od tego,
    // czy element powraca do pozycji początkowej,
    // czy porusza się w kierunku pozycji końcowej
    var nowaPozycja = element.powrot ? element.aktualnaPozycja - krok :
        element.aktualnaPozycja + krok;

    element.aktualnaPozycja = nowaPozycja;

    // Przesunięcie odbywa się równolegle do osi odciętych.
    // Z tego powodu współrzędna y punktu docelowego nie ulega zmianie
    przesunElement(idElementu, nowaPozycja, element.style.top);
    element.animacja = setTimeout("ruchPoziomy('" + idElementu + "', " + poczatek
        + ", " + koniec + ", " + krok + ");", 10);

    if (nowaPozycja >= koniec)
        element.powrot = true;
    else if(nowaPozycja <= poczatek)
        element.powrot = false;
}

function zatrzymajRuchPoziomy(idElementu) {
    var element = document.getElementById(idElementu);

    if (!element) return;

    zatrzymajAnimacje(element.animacja)
}
```

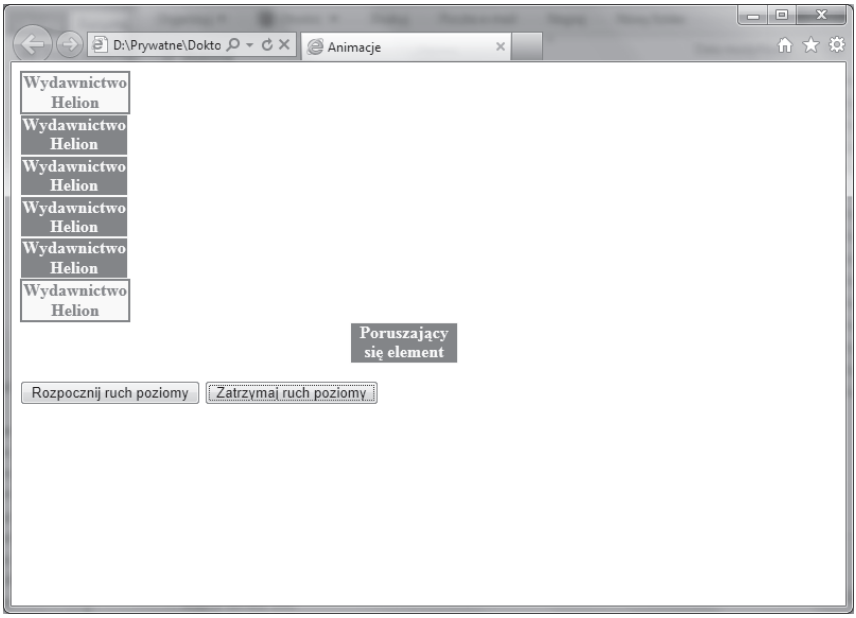
2. W sekcji `<body>` witryny *Animacje.htm* umieść definicje dwóch przycisków oraz bloku o identyfikatorze `blok7` (listing 10.10).

Listing 10.10. *Definicja animowanego bloku oraz przycisków, których domyślne metody zdarzeniowe uruchamiają i zatrzymują animację ruchu wzdłuż osi OX*

```
<div id="blok7" class="brazowy">Poruszający się element</div>

<p>
    <input id="button1" type="button" value="Rozpocznij ruch poziomy"
    ➥onclick="ruchPoziomy('blok7', 0, window.innerWidth, 5)" />
    <input id="button2" type="button" value="Zatrzymaj ruch poziomy"
    ➥onclick="zatrzymajRuchPoziomy('blok7')" />
</p>
```

3. Otwórz witrynę *Animacje.htm* w domyślnej przeglądarce internetowej.

4. Kliknij przycisk z etykietą *Rozpocznij ruch poziomy*. Wynik działania aplikacji powinien być analogiczny do tego z rysunku 10.4.



Rysunek 10.4. *Animacja ruchu wzdłuż osi OX*

Naturalnym rozszerzeniem funkcjonalności zaimplementowanej w tym podrozdziale będzie zaprojektowanie funkcji, której zadaniem będzie animacja ruchu wzdłuż osi rzędnych oraz w dwóch kierunkach jednocześnie. Pozostawiam to jednak jako zadanie dla czytelnika.

Podsumowanie

Animacje są często spotykanym elementem interaktywnych aplikacji internetowych. W tym rozdziale pokazałem, w jaki sposób można je zaimplementować za pomocą technologii JavaScript. Nie ulega wątpliwości fakt, że dzięki popularnym bibliotekom JavaScript, jak chociażby jQuery, która jest zintegrowana z domyślnym szablonem aplikacji ASP.NET MVC, tworzenie i aplikowanie animacji jest znacznie prostsze niż samodzielne ich implementowanie. Jednakże w sytuacji, gdy dostępne sposoby animacji są niewystarczające na potrzeby konkretnego projektu, samodzielna umiejętność ich implementacji może okazać się nieoceniona.

Część II
Biblioteka jQuery

Skorowidz

G

H

I

J

K

L

Ł

M

N

O

P

R

S

T

U

W

Z

Ż

JavaScript i jQuery

Kompletny przewodnik dla programistów interaktywnych aplikacji internetowych w Visual Studio

Technologia JavaScript nie od dziś jest uważana za potężne narzędzie, doskonale wpasowujące się w sposób działania aplikacji internetowych po stronie serwera. To zaś jest zaleta nie do przecenienia w czasach, gdy przez globalną sieć przesyłane są niewiarygodnie wielkie ilości danych, a internet w zasadzie służy do załatwiania wszelkich codziennych spraw. Każda współczesna aplikacja internetowa musi zapewniać szybki transfer i bezpieczeństwo danych, a także oferować użytkownikom końcowym wygodny interaktywny widok wraz z możliwością otwierania i wprowadzania danych na różnych urządzeniach. Połączenie JavaScriptu i jQuery pozwala osiągnąć taki stan rzeczy bez nadmiernego obciążania sieci i serwerów.

W tej książce znajdziesz informacje o technologii JavaScript oraz bibliotekach jQuery oraz jQuery UI, które umożliwiają tworzenie interaktywnych widoków aplikacji internetowych. Dowiesz się więcej o składni języka JavaScript i zapewnianiu interakcji z różnymi elementami strony oraz o tworzeniu i wykorzystywaniu bibliotek skryptów. Nauczysz się zmieniać właściwości różnych elementów witryny, odkryjesz, do czego służą formularze i obiekty wbudowane, a ponadto zrozumiesz, jak ważną rolę w procesie projektowania i implementowania aplikacji internetowych odgrywają biblioteki jQuery i jQuery UI. Warto, przekonaj się sam!

- Konfiguracja obsługi JavaScriptu i biblioteki skryptów
- Debugowanie kodu w JavaScripcie
- Dostęp do wybranych elementów stron internetowych
- Obsługa zdarzeń i drzewo DOM
- Tabele danych, źródła XML oraz pętla for in
- Formularze, obiekty wbudowane i animacje
- Podstawy jQuery: selektory, zdarzenia, programowanie asynchroniczne
- Zaawansowane aspekty jQuery: animacje, formularze, AJAX, wtyczki i rozszerzenia
- jQuery UI: wtyczki, interakcja z komponentami oraz zaawansowane efekty animacji

Interaktywna strona WWW?

Tylko z JavaScriptem i jQuery!

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

ISBN 978-83-246-8283-6

9 788324 682836

Cena: 49,00 zł