Włodzimierz Gajda

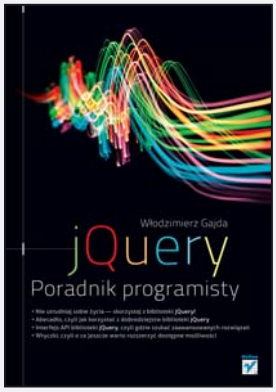
jQuery

Poradnik programisty

- Nie utrudniaj sobie życia — skorzystaj z biblioteki jQuery!
- Abecadło, czyli jak korzystać z dobrodziejstw biblioteki jQuery
- Interfejs API biblioteki jQuery, czyli gdzie szukać zaawansowanych rozwiązań
- Wtyczki, czyli o co jeszcze warto rozszerzyć dostępne możliwości

Helion

JQuery. Poradnik programisty



Autor: Włodzimierz Gajda
ISBN: 978-83-246-2518-5
Format: 158×235, stron: 288

Nie utrudniaj sobie życia – skorzystaj z biblioteki jQuery!

- Abecadło, czyli jak korzystać z dobrodziejstw biblioteki jQuery
- Interfejs API biblioteki jQuery, czyli gdzie szukać zaawansowanych rozwiązań
- Wtyczki, czyli o co jeszcze warto rozszerzyć dostępne możliwości

Biblioteka jQuery, zarówno w wersji pełnej, jak i zminimalizowanej, pozwala programiście znacząco uprościć pracę i stopień skomplikowania kodu tworzonego w języku JavaScript. Korzystając z jej możliwości, programista może zmieniać atrybuty, modyfikować wygląd poszczególnych elementów strony, dodawać lub usuwać elementy drzewa DOM. Może też wykonać zapytania Ajax, stosować efekty specjalne, obsłużyć typowe i nietypowe zdarzenia, a także posłużyć się różnymi wtyczkami, często znacząco rozszerzającymi funkcjonalność kodu.

Książka „jQuery. Poradnik programisty" to doskonałe kompendium wiedzy na temat tej biblioteki. Dowiesz się stąd, jak rozpocząć pracę z jQuery, jak obchodzić się z selektorami i atrybutami, manipulować modelem DOM czy przypisywać wybrany styl do określonych elementów strony. Nauczysz się stosować funkcję jQuery, filtry i operacje na zbiorach elementów. Poznasz także rodzaje i sposób działania wtyczek, sam zaczniesz je tworzyć, minimalizować i kompresować. Jeśli interesuje Cię programowanie z wykorzystaniem możliwości oferowanych przez JavaScript, a nie chcesz spędzać wielu godzin na bezpośrednim wpisywaniu skomplikowanego kodu, biblioteka jQuery jest właśnie dla Ciebie!

- Korzystanie z biblioteki jQuery
- Trzy warstwy dokumentu XHTML: struktura, wygląd i zachowanie
- Selektory CSS i zdarzenia XHTML
- Modyfikacja wyglądu, odczyt i modyfikacja treści elementów
- Odczyt i modyfikacja atrybutów
- Dodawanie i usuwanie węzłów drzewa DOM, wędrówka po drzewie DOM
- Zbiory węzłów, tworzenie i usuwanie węzłów w drzewie DOM
- Efekty specjalne
- Funkcja jQuery() – w skrócie $()
- Odczyt i modyfikacja węzłów drzewa DOM
- Operacje przekształcające zbiór elementów
- Parametry wtyczek, ich tworzenie, minimalizacja i kompresja

Wykorzystaj szanse, jakie daje Ci biblioteka jQuery!

Spis treści

Rozdział 11.

Efekty specjalne

Biblioteka jQuery zawiera kilka metod służących do wykonywania prostych animacji. Należą do nich: `slideDown()` i `slideUp()`, `fadeIn()` i `fadeOut()` oraz `animate()`.

Metody `slideDown()` i `slideUp()` pozwalają na płynne rozwijanie i zwijanie elementu. Efekt graficzny polega na animacji elementu poprzez zwiększenie lub zmniejszenie jego wysokości. Jeśli na ukrytym akapicie:

```
$('p#info').hide();
```

wywołamy metodę `slideDown()`:

```
$('p#info').slideDown();
```

to akapit ten pojawi się na stronie w sposób animowany. Jego wysokość będzie płynnie zwiększana od 0 do odpowiedniej wartości. Wywołanie metody `slideUp()` spowoduje ponowne ukrycie akapitu:

```
$('p#info').slideUp();
```

Tym razem jego wysokość będzie zmniejszana do 0. Domyślnie animacja trwa 400 milisekund, lecz możemy to zmienić, przekazując do metod `slideDown()` oraz `slideUp()` parametr określający czas trwania animacji. Parametrem tym może być liczba określająca, ile milisekund ma trwać animacja, np.:

```
$('p#info').slideUp(1200);
$('p#info').slideDown(2500);
```

bądź jeden z napisów: `slow` lub `fast`. Napis `slow` ustala czas trwania animacji na 200 milisekund, a `fast` — na 600. Drugim, również opcjonalnym, parametrem funkcji `slideUp()` i `slideDown()` jest funkcja anonimowa, wywoływana po zakończeniu animacji. Jeśli chcesz, by po zakończeniu rozwijania akapitu kolor jego tła stał się czerwony, użyj kodu:

```
$('p#info').slideDown(2500, function(){
    $('p#info').css('background', 'red');
});
```

Zwróć uwagę, że rozwiązanie niewykorzystujące funkcji anonimowej:

```
$('p#info').slideDown(2500);
$('p#info').css('background', 'red');
```

jest błędne. Wywołanie funkcji `slideDown()` nie powoduje wstrzymania wykonywania skryptu. Metoda `css()` w powyższym kodzie będzie wywołana natychmiast po uruchomieniu animacji, a nie po jej zakończeniu.

Kolejne dwie funkcje dotyczące efektów, czyli `fadeIn()` oraz `fadeOut()`, powodują pokazanie i ukrycie elementu przez zwiększanie i zmniejszanie jego przezroczystości. Mają one identyczne parametry jak `slideDown()` i `slideUp()`. Po wywołaniu:

```
$('p#info').fadeIn();
```

akapit `p#info` pojawi się na stronie, zaś instrukcja:

```
$('p#info').fadeOut();
```

spowoduje ukrycie go. Tempo pojawiania się i znikania ustalamy, przekazując liczbę lub napisy `slow` albo `fast`:

```
$('p#info').fadeIn(1500);
$('p#info').fadeIn('fast');
$('p#info').fadeIn('slow');
```

Funkcja automatyczna przekazana jako drugi parametr będzie wywołana po zakończeniu efektu:

```
$('p#info').fadeIn(1000, function(){
    //funkcja wywoływana po zakończeniu animacji
});
```

Ostatnia z wymienionych na wstępie funkcji, `animate()`, pozwala na płynne modyfikowanie dowolnej właściwości CSS. Instrukcja:

```
$('p#info').animate({
    font-size: '200%',
    left: '200px',
    borderWidth: '10px'
});
```

spowoduje płynną zmianę rozmiaru czcionki do 200%. Jeśli czcionka była większa, to będzie zmniejszona, a jeśli była mniejsza — to będzie zwiększona. Właściwość `left` będzie płynnie dążyła do wartości `200px`, zaś grubość obramowania — do `10px`. Pierwszym parametrem metody `animate()` jest tablica asocjacyjna właściwości CSS, do których animacja ma płynnie dążyć. Zwróć uwagę, że atrybuty pisane w kodzie CSS z dywizem, np. `border-width`, są w jQuery zapisywane w notacji `borderWidth`. Dwa opcjonalne parametry metody `animate()` ustalają czas trwania animacji oraz pozwalają na wykonanie dowolnych akcji po jej zakończeniu:

```
$('p#info').animate(
    {
        font-size: '200%',
        left: '200px',
        borderWidth: '10px'
    },
    5000,
```

```
    function() {
        //funkcja wywoływana po zakończeniu animacji
    }
);
```

Ćwiczenie 11.1.

Zmodyfikuj kod kontrolki +/- wykonanej w ćwiczeniu 8.1 w taki sposób, by pokazywanie i ukrywanie treści odbywało się w sposób płynny — z wykorzystaniem metod `slide ↪Down()` oraz `slideUp()`. Kod XHTML kontrolki zawiera hiperłącze `a`, w którym umieszczamy znak + bądź -, oraz akapit `p` z treścią. Akapit należy ukryć lub pokazać. Zarys kodu XHTML z ćwiczenia jest przedstawiony na listingu 11.1. Rozwiązaniem ćwiczenia jest skrypt jQuery przedstawiony na listingu 11.2.

Listing 11.1. *Zarys kodu XHTML kontrolki +/–*

```
<div class="element">
    <h3><a href="#">+</a>Lorem ipsum</h3>
    <p>
        Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
    </p>
    <div class="clearing"></div>
</div>
```

Listing 11.2. *Rozwiązanie ćwiczenia 11.1*

```
$(function(){
    $('.element p').hide();
    $('.element a').click(function(){
        if ($(this).html() == '+') {
            $(this).html('-');
            $(this).parent().next().slideDown();
        } else {
            $(this).html('+');
            $(this).parent().next().slideUp();
        }
    });
});
```

W skrypcie jQuery postępujemy identycznie jak w ćwiczeniu 8.1. W odpowiedzi na kliknięcie hiperłącza `a`, w zależności od tego, czy zawiera ono znak, + czy -, rozwijamy bądź zwijamy akapit. Do akapitu docieramy od klikniętego hiperłącza (tj. od `$(this)`), wywołując metody `parent()` oraz `next()`. W celu pokazania akapitu wywołujemy — zamiast metody `show()` — metodę `slideDown()`:

```
$(this).parent().next().slideDown();
```

Natomiast ukrycie akapitu wymaga wywołania — zamiast metody `hide()` — metody `slideUp()`:

```
$(this).parent().next().slideUp();
```

Ćwiczenie 11.2.

Dany jest plik *menu.html* oraz kilka plików, których nazwa rozpoczyna się od słowa *fragment*, np. *fragment-ulica-bramowa.html*. W pliku *menu.html* znajduje się menu główne oraz element `div#tresc`. Zarys pliku *menu.html* jest przedstawiony na listingu 11.3. Napisz skrypt, który zmodyfikuje działanie menu w taki sposób, by kliknięcie opcji menu powodowało przeładowanie elementu `div#tresc`. Do elementu tego należy załadować zawartość pliku, którego nazwa jest zawarta w klikniętym hiperłączu. Zadanie rozwiąż w taki sposób, by przeładowanie było płynne — z wykorzystaniem efektów `fadeIn()` oraz `fadeOut()`. Skrypt powinien działać w taki sposób, by podczas animacji opcje menu były nieczynne. Rozwiązaniem zadania jest skrypt z listingu 11.4.

Listing 11.3. *Plik menu.html z ćwiczenia 11.2*

```
<div id="pojemnik">
  <ul id="menu">
    <li><a href="fragment-ulica-bramowa.html">ulica bramowa</a></li>
    <li><a href="fragment-plac-po-farze.html">plac po farze</a></li>
    ...
  </ul>
  <div id="tresc"></div>
</div>
```

Listing 11.4. *Rozwiązanie ćwiczenia 11.2*

```
<script type="text/javascript">
trwa = 0;
function wylacz_hiperlacza()
{
    trwa = 1;
    $('a').
        css('text-decoration', 'none').
        hover(
            function(){
                $(this).css('text-decoration', 'none');
            },
            function(){
                $(this).css('text-decoration', 'none');
            }
        );
}

function wlacz_hiperlacza()
{
    trwa = 0;
    $('a').
        css('text-decoration', 'none').
        hover(
            function(){
                $(this).css('text-decoration', 'underline');
            },
            function(){
                $(this).css('text-decoration', 'none');
            }
```

```
        );
}

$(function(){
    wylacz_hiperlacza();
    $('#tresc').hide();
    $('#tresc').load('fragment-ulica-dominikanska.html');
    $('#tresc').fadeIn(2000, function(){
        wlacz_hiperlacza();
    });

    $('a').click(function(){
        if (trwa == 0) {
            wylacz_hiperlacza();
            var url = $(this).attr('href');
            $('#tresc').fadeOut(2000, function(){
                $('#tresc').load(url);
                $('#tresc').fadeIn(2000, function(){
                    wlacz_hiperlacza();
                });
            });
        }
        return false;
    });
});
</script>
```

W skrypcie z listingu 11.4 najpierw definiujemy zmienną `trwa`. Zmienna ta będzie flagą, informującą o tym, że animacja jest w trakcie wykonywania. Jeśli wartość zmiennej `trwa` wynosi `1`, to animacja właśnie jest wykonywana i hiperłącza nie powinny działać. W przeciwnym razie, czyli gdy wartość zmiennej `trwa` wynosi `0`, animacja nie jest wykonywana i hiperłącza powinny działać.

Za włączanie i wyłączanie działania hiperłączy odpowiadają funkcje `wlacz_hiperlacza()` oraz `wylacz_hiperlacza()`. Ustalają one wartość zmiennej `trwa` oraz modyfikują style CSS hiperłączy. Wewnątrz tych funkcji wybieramy wszystkie hiperłącza:

```
$('a').
```

ustalamy ich style CSS:

```
css('text-decoration', 'none').
```

i modyfikujemy efekt rollover:

```
hover(
    function(){
        $(this).css('text-decoration', 'none');
    },
    function(){
        $(this).css('text-decoration', 'none');
    }
);
```

Metoda `hover()` przypisuje obsługę dwóch zdarzeń: `onmouseenter` oraz `onmouseleave`[1]. Powyższa instrukcja jest równoważna dwóm instrukcjom:

```
mouseenter(function(){
    $(this).css('text-decoration', 'none');
}).mouseleave(function(){
    $(this).css('text-decoration', 'none');
});
```

Działanie skryptu z listingu 11.4 rozpoczynamy od wyłączenia hiperłączy, ukrycia elementu `#tresc`, załadowania do elementu `#tresc` pliku *fragment-ulica-dominikanska.html* oraz animowanego wyświetlenia elementu `#tresc`:

```
wylacz_hiperlacza();
$('#tresc').hide();
$('#tresc').load('fragment-ulica-dominikanska.html');
$('#tresc').fadeIn(2000, function(){
    wlacz_hiperlacza();
});
```

Animacja będzie trwała 2 sekundy (tj. 2000 milisekund). Po zakończeniu animacji hiperłącza zostaną włączone, za co odpowiada wywołanie funkcji `wlacz_hiperlacza()`.

Zasadniczą częścią skryptu jest przypisanie obsługi zdarzenia `onclick` do hiperłączy. Hiperłącza będą działały wyłącznie wtedy, gdy wartość zmiennej `trwa` wynosi 0:

```
$('a').click(function(){
    if (trwa == 0) {
        ...
    }
    return false;
});
```

W ten sposób wyłączamy działanie hiperłączy w trakcie trwania animacji. Zdarzenie `onclick` będzie generowane, jednak dzięki powyższej instrukcji `if` nie spowoduje ono wykonania żadnych akcji.

Jeśli animacja nie jest wykonywana, wówczas obsługa kliknięcia przebiega następująco: najpierw wyłączamy działanie hiperłączy:

```
wylacz_hiperlacza();
```

Następnie z atrybutu `href` kliknięty hiperłącza do zmiennej `url` pobieramy nazwę pliku:

```
var url = $(this).attr('href');
```

[1] Zdarzenia `mouseeneter` oraz `mouseleave` nie występują w drzewie DOM. Są one emulowane przez bibliotekę jQuery. Pierwotnie zdarzenia te pojawiły się w przeglądarce Internet Explorer. Zdarzenia `mouseeneter` i `mouseleave` różnią się od zdarzeń `mouseover` i `mouseout` tym, że są generowane dokładnie jeden raz, dopóki kursor myszki nie opuści wybranego elementu. Podczas jednokrotnego wskazania elementu wskaźnikiem myszki zdarzenia te zostaną wygenerowane dokładnie jeden raz. Zdarzenia `mouseover` oraz `mouseout` mogą być wygenerowane kilkukrotnie.

Teraz wyłączamy widoczność elementu #tresc. Element ten stopniowo zniknie z ekranu:

```
$('#tresc').fadeOut(2000, function(){
    //kod wykonywany, gdy element #tresc jest już niewidoczny
});
```

Gdy element #tresc jest niewidoczny, ładujemy do niego zawartość pliku, którego nazwa znajduje się w zmiennej url:

```
$('#tresc').load(url);
```

po czym włączamy widoczność elementu #tresc:

```
$('#tresc').fadeIn(2000, function(){
    //kod wykonywany, gdy element #tresc jest już widoczny
});
```

Element ten pojawi się stopniowo na ekranie. Gdy element #tresc jest już widoczny, ponownie włączamy działanie hiperłączy:

```
wlacz_hiperlacza();
```

Ćwiczenie 11.3.

Dokument *index.html* zawiera znacznik img, który umieszcza na stronie WWW zdjęcie zawarte w pliku *fotka.jpg*. Poniżej zdjęcia znajdują się trzy hiperłącza. Jedno jako etykietę ma strzałkę w lewo, drugie — napis *reset*, a trzecie — strzałkę w prawo. Zarys kodu XHTML jest przedstawiony na listingu 11.5. Napisz skrypt jQuery, który zmodyfikuje działanie hiperłączy. Strzałka w lewo ma powodować, że obraz płynnym ruchem wyjedzie poza lewą krawędź nadrzędnego elementu div, strzałka w prawo ma powodować płynne wyjechanie zdjęcia poza prawą krawędź elementu div, zaś napis *reset* ma powodować powrót zdjęcia do środka elementu div. Rozwiązaniem zadania jest skrypt przedstawiony na listingu 11.6.

Listing 11.5. *Plik index.html z ćwiczenia 11.3*

```
<body>
  <div id="element">
    <img src="fotka.jpg" alt="" />
  </div>
  <div>
    <a id="lewo" href="#">&lt;&lt;</a>
    <a id="reset" href="#">reset</a>
    <a id="prawo" href="#">&gt;&gt;</a>
  </div>
</body>
```

Listing 11.6. *Skrypt z ćwiczenia 11.3*

```
$(document).ready(function(){

    $("#lewo").click(function(){
        $('#element img').animate({
            'left' : "-100px"
        });
        return false;
    });


    $("#prawo").click(function(){
        $('#element img').animate({
            'left' : "200px"
        });
        return false;
    });

    $("#reset").click(function(){
        $('#element img').animate({
            'left' : "60px"
        });
        return false;
    });

});
```

Skrypt z listingu 11.6 zawiera instrukcje, które przypiszą procedury obsługi zdarzenia `onclick` trzem hiperłączom zawartym w dokumencie. Sednem rozwiązania jest płynne przesuwanie zdjęcia w trzy różne miejsca. W celu przesunięcia fotografii poza lewą krawędź elementu `div` modyfikujemy właściwość CSS `left`, nadając jej wartość `-100` pikseli:

```
$('#element img').animate({
    'left' : "-100px"
});
```

Przesunięcie elementu `img` poza prawą krawędź elementu `div` wykonuje instrukcja:

```
$('#element img').animate({
    'left' : "200px"
});
```

Natomiast powrót do pozycji startowej polega na przypisaniu właściwości `left` wartości 60 pikseli:

```
$('#element img').animate({
    'left' : "60px"
});
```

Funkcja `animate()` spowoduje płynną zmianę właściwości `left` od wartości bieżącej do jednej z podanych liczb, czyli `-100px`, `60px` lub `200px`.

W rozwiązaniu tym ważną rolę odgrywają style CSS. Element `img` jest pozycjonowany bezwzględnie wewnątrz zawierającego go elementu `div`:

```
#element {
    position: relative;
    overflow: hidden;
}
#element img {
    display: block;
    width: 80px;
    height: 60px;
    position: absolute;
    top: 10px;
    left: 60px;
}
```

Nie utrudniaj sobie życia — skorzystaj z biblioteki jQuery!

- Abecadło, czyli jak korzystać z dobrodziejstw biblioteki jQuery
- Interfejs API biblioteki jQuery, czyli gdzie szukać zaawansowanych rozwiązań
- Wtyczki, czyli o co jeszcze warto rozszerzyć dostępne możliwości

Biblioteka jQuery, zarówno w wersji pełnej, jak i zminimalizowanej, pozwala programiście znacząco uprościć pracę i stopień skomplikowania kodu tworzonego w języku JavaScript. Korzystając z jej możliwości, programista może zmieniać atrybuty, modyfikować wygląd poszczególnych elementów strony, dodawać lub usuwać elementy drzewa DOM. Może też wykonać zapytania Ajax, stosować efekty specjalne, obsłużyć typowe i nietypowe zdarzenia, a także posłużyć się różnymi wtyczkami, często znacząco rozszerzającymi funkcjonalność kodu.

Książka „jQuery. Poradnik programisty" to doskonałe kompendium wiedzy na temat tej biblioteki. Dowiesz się z niej, jak rozpocząć pracę z jQuery, jak obchodzić się z selektorami i atrybutami, manipulować modelem DOM czy przypisywać wybrany styl do określonych elementów strony. Nauczysz się stosować funkcję jQuery, filtry i operacje na zbiorach elementów. Poznasz także rodzaje i sposób działania wtyczek, sam zaczniesz je tworzyć, minimalizować i kompresować. Jeśli interesuje Cię programowanie z wykorzystaniem możliwości oferowanych przez JavaScript, a nie chcesz spędzać wielu godzin na bezpośrednim wpisywaniu skomplikowanego kodu, biblioteka jQuery jest właśnie dla Ciebie!

- Trzy warstwy dokumentu XHTML: struktura, wygląd i zachowanie
- Korzystanie z biblioteki jQuery
- Selektory CSS i zdarzenia XHTML
- Modyfikacja wyglądu, odczyt i modyfikacja treści elementów
- Odczyt i modyfikacja atrybutów
- Dodawanie i usuwanie węzłów drzewa DOM, wędrówka po drzewie DOM
- Zbiory węzłów, tworzenie i usuwanie węzłów w drzewie DOM
- Efekty specjalne
- Funkcja jQuery() — w skrócie $()
- Odczyt i modyfikacja węzłów drzewa DOM
- Operacje przekształcające zbiór elementów
- Parametry wtyczek, ich tworzenie, minimalizacja i kompresja

Wykorzystaj szanse, jakie daje Ci biblioteka jQuery!

Cena: 39,00 zł

Nr katalogowy: 5455

Księgarnia internetowa: http://helion.pl

Zamówienia telefoniczne: 0 801 339900, 0 601 339900

Wydawnictwo Helion

ul. Kościuszki 1c, 44-100 Gliwice
44-100 Gliwice, skr. poczt. 462
32 230 98 63
http://helion.pl
e-mail: helion@helion.pl

helion.pl księgarnia internetowa

ISBN 978-83-246-2518-5

Informatyka w najlepszym wydaniu