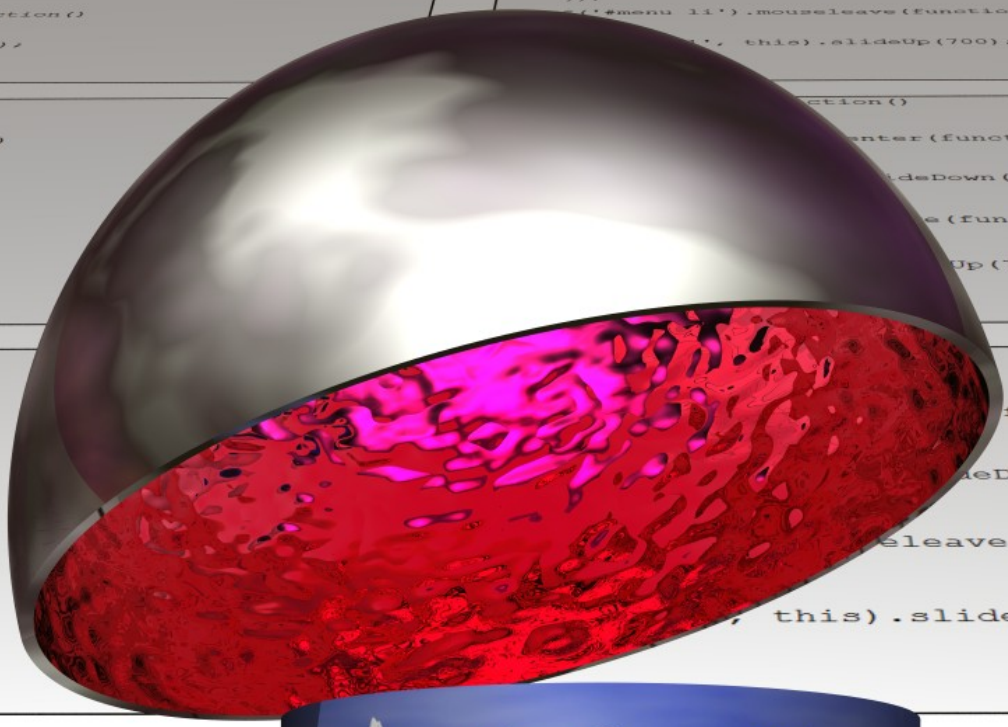
SERWAN



Podstawy frameworka JavaScript jQuery

Leszek Sewastianowicz

© Copyright 2009 Leszek Sewastianowicz
wszelkie prawa zastrzeżone
trener@serwan.pl
www.serwan.pl

Spis treści

Instalacja jQuery....3
Jak dołączyć skrypt napisany w jQuery do strony WWW....3
Od czego zacząć skrypt jQuery....3
Selektory, efekty i zdarzenia....4
    Selektory....5
    Efekty....7
    Zdarzenia....9
Menu rozwijane....10
O autorze....14

Instalacja jQuery.

JQuery jest lekką biblioteką JavaScript często nazywaną frameworkiem. W rzeczywistości jest to jeden plik, który można pobrać z adresu http://code.google.com/p/jqueryjs/downloads/detail?name=jquery-1.3.2.js.

Plik ten można zapisać w katalogu głównym strony lub w podkatalogu np. *js/*. Włączenie pliku z biblioteką odbywa się poprzez umieszczenie w sekcji nagłówkowej strony (pomiędzy znacznikami *<head></head>*) linijki:

```
<script type =”text/javascript” src=”/jquery-
1.3.2.js”></script>
```

lub

```
<script type =”text/javascript” src=”/js/jquery-
1.3.2.js”></script>
```

jeżeli plik *jquery-1.3.2.js* znajduje się w podkatalogu *js/*.

Jak dołączyć skrypt napisany w jQuery do strony WWW.

Tworzymy kolejny plik z rozszerzeniem “js” np. *skrypt.js* i dołączamy go do strony w podobny sposób jak bibliotekę:

```
<script type =”text/javascript” src=”/js/jquery-
1.3.2.js”></script>

<script type =”text/javascript” src=”/js/skrypt.js”></script>
```

Ważne jest, by biblioteka była włączona jako pierwsza, w przeciwnym razie nie będzie widoczna w pliku *skrypt.js* i nie można będzie z niej skorzystać.

Od czego zacząć skrypt jQuery.

Początkiem każdego skryptu jQuery jest wywołanie funkcji *$()*. *$()* jest tak naprawdę aliasem funkcji głównej frameworka – *jQuery()*. Ze względu na to, że jest ona bardzo często wywoływana, zastąpienie jej aliasem jest bardzo wygodne. Wywołanie *$(document)* zwraca obiekt, na rzecz którego wywołujemy metodę *ready()*. Funkcja będąca argumentem metody *ready()* zostanie

uruchomiona od razu po wczytaniu przez przeglądarkę drzewa dokumentu (DOM) czyli całego kodu html/xhtml. Parametrem metody *ready()* jest funkcja anonimowa, czyli pozbawiona nazwy. Jej definicja występuje w miejscu wywołania. Funkcję taką zapisujemy wyrażeniem *function()* a po nim w nawiasach „{}” umieszczamy ciało funkcji. W przykładzie poniżej jest to miejsce, gdzie zostaną umieszczone wywołania wszystkich wykorzystanych w skrypcie funkcji biblioteki jQuery. Umieszczanie tego kodu w innym miejscu mijałoby się z celem, gdyż w jQuery manipulujemy elementami strony czyli drzewa dokumentu i wykonanie jakichkolwiek operacji przed wczytaniem tych elementów do przeglądarki nie będzie możliwe.

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

{

    //tu znajdzie się cały kod wykorzystujący jQuery.

});
```

Selektory, efekty i zdarzenia

Wiele ciekawych efektów można osiągnąć prostymi środkami. Np. pokazanie akapitu po kliknięciu nagłówka. Kod HTML użyty w przykładzie wygląda następująco:

```
<h3>Nagłówek</h3>

<p>Akapit, który najpierw zostanie ukryty przy pomocy stylów
CSS, by potem można było go pokazać.</p>
```

Akapit zostaje ukryty przy pomocy stylów.

```
p

{

    display: none;

}
```

W pliku *skrypt.js* dochodzi kilka nowych linii. Przede wszystkim wywołanie funkcji *$()*, której argumentem tym razem jest selektor elementu h3. Funkcja zwraca obiekt reprezentujący wszystkie elementy h3 na stronie. Są to elementy **dopasowane** do selektora. Dodając do tego obiektu zdarzenie *click* definiujemy obsługę kliknięcia nagłówka h3 na stronie. Argumentem zdarzenia jest funkcja wykonywana w reakcji na to zdarzenie. Wewnątrz funkcji na elemencie akapitu (gdyby na stronie umieszczono kilka akapitów, dotyczyłoby to ich wszystkich) zostaje wywołana metoda *show()*, która spowoduje pojawienie się akapitu na stronie.

```
$(document).ready(function()
{
    $('h3').click(function()
    {
        $('p').show();
    });
});
```

Akapit można ponownie ukryć definiując obsługę zdarzenia *click* dla elementów p, a w funkcji obsługującej zdarzenie, wywołując metodę *hide()* na elemencie akapitu.

```
$(document).ready(function()
{
    $('h3').click(function()
    {
        $('p').show();
    });
    $('p').click(function()
    {
        $('p').hide();
    });
});
```

Selektory

Oprócz selektorów elementów w jQuery możemy używać też selektora globalnego *$('*')*, selektorów identyfikatorów – np. *$('#galeria')*, selektorów klasy – np. *$('.obrazek')*), selektorów klas – np. *$('.obrazek.wybrany').* Elementami dopasowanymi będą w tym przypadku elementy należące do wszystkich wpisanych klas. Dostępne są też selektory wskazujące na hierarchie elementów: selektory elementów zawartych w innych elementach – np. *$('#galeria img'),* selektor elementów leżących bezpośrednio wewnątrz innych elementów – np. *$('body>div')*, selektor elementu występującego bezpośrednio po innym elemencie – np. *$('.opis+a')*, selektor elementów

leżących na tym samym poziomie zagnieżdżenia, a następujących po wskazanym elemencie – np. *$('#wstep~div')*. W jednym wywołaniu funkcji możemy wskazać na kilka selektorów. Poszczególne selektory umieszcza się rozdzielając je przecinkiem – np. *$('img, p, #opis')*.

Do selektorów można dodatkowo zastosować filtry, które modyfikując ich działanie dają twórcom stron ogromne możliwości. Przykładowo można odrębnie sformatować parzyste i nieparzyste wiersze tabeli, albo pierwszy akapit, albo niezaznaczone okienka checkbox w formularzu. Filtry dodajemy do selektora po dwukropku np. *$('tr:even')*.

Najpopularniejsze filtry prezentuje poniższa tabela.

Nazwa filtru	**Opis**
:first	pierwszy element
:last	ostatni element
:not(selektor)	z wyjątkiem elementu wskazanego selektorem np.: $("input:**not**(:checked)”)
:even	elementy parzyste
:odd	elementy nieparzyste
:eq(index)	element wskazany indeksem
:gt(index)	elementy o indeksach niższych lub równym wskazanemu
:lt(index)	elementy o indeksach wyższych lub równym wskazanemu
:header	elementy nagłówków czyli h1, h2, h3...
:animated	element aktualnie animowany
:conteins(tekst)	elementy zawierające tekst
:empty	elementy niezawierające innych elementów w tym tekstu
:has(selektor)	elementy zawierające przynajmniej jeden element zawierający element wskazany selektorem
:parent	elementy będące rodzicami (zawierające inne elementy w tym tekst)

Nazwa filtru	Opis
:hidden	elementy ukryte
:visible	elementy widoczne

Efekty

Oprócz metod *show()* i *hide()* można wywołać też inne efekty. Ich wykaz wraz z opisami prezentuje poniższa tabela.

Nazwa efektu	Opis
show()	Wyświetla dopasowane elementy jeżeli zostały wcześniej ukryte.
show(*prędkość*, [*funkcja_zwrotna*])	Wyświetla dopasowane elementy jeżeli zostały wcześniej ukryte. Efekt trwa określoną ilość milisekund wskazaną w argumencie *prędkość*. Po zakończeniu animacji wywołana zostanie funkcja zwrotna.
hide()	Ukrywa dopasowane elementy.
hide(*prędkość*, [*funkcja_zwrotna*])	Ukrywa dopasowane elementy. Efekt trwa określoną ilość milisekund wskazaną w argumencie *prędkość*. Po zakończeniu animacji wywołana zostanie funkcja zwrotna.
toggle()	Zmienia widoczność dla dopasowanych elementów. Jeżeli są widoczne, ukrywa je metodą *hide()*. Jeżeli są niewidoczne, wyświetla je metodą *show()*.
toggle(*przełącznik*)	Zmienia widoczność dla dopasowanych elementów w zależności od wartości argumentu *przełącznik*. Dla wartości true pokazuje wszystkie elementy, dla wartości false ukrywa wszystkie elementy.
toggle(*prędkość*, [*funkcja_zwrotna*])	Zmienia widoczność dla dopasowanych elementów. Jeżeli są widoczne, ukrywa je metodą *hide()*. Jeżeli są niewidoczne, wyświetla je metodą *show()*. Efekt trwa określoną ilość milisekund wskazaną w argumencie *prędkość*. Po zakończeniu animacji wywołana zostanie funkcja zwrotna.
slideDown(*prędkość*, [*funkcja_zwrotna*])	Rozwija dopasowane ukryte elementy do dołu aż do uzyskania pełnej wysokości. Marginesy pionowe – zewnętrzne i wewnętrzne – są również animowane.

	Efekt trwa określoną ilość milisekund wskazaną w argumencie *prędkość*. Po zakończeniu animacji wywołana zostanie funkcja zwrotna.
slideUp(*prędkość*, [*funkcja_zwrotna*])	Ukrywa dopasowane elementy płynnie zmniejszając ich wysokość aż do całkowitego ukrycia. Marginesy pionowe – zewnętrzne i wewnętrzne – są również animowane. Efekt trwa określoną ilość milisekund wskazaną w argumencie *prędkość*. Po zakończeniu animacji wywołana zostanie funkcja zwrotna.
slideToggle(*prędkość*, [*funkcja_zwrotna*])	Zmienia widoczność dla dopasowanych elementów. Jeżeli są widoczne, ukrywa je metodą *slideUp()*. Jeżeli są niewidoczne, wyświetla je metodą *slideDown()*. Efekt trwa określoną ilość milisekund wskazaną w argumencie *prędkość*. Po zakończeniu animacji wywołana zostanie funkcja zwrotna.
fadeIn(*prędkość*, [*funkcja_zwrotna*])	Pokazuje dopasowane ukryte elementy płynnie zmniejszając ich przezroczystość. Efekt trwa określoną ilość milisekund wskazaną w argumencie *prędkość*. Po zakończeniu animacji wywołana zostanie funkcja zwrotna.
fadeOut(*prędkość*, [*funkcja_zwrotna*])	Ukrywa dopasowane elementy płynnie zwiększając ich przezroczystość aż do całkowitego ukrycia. Efekt trwa określoną ilość milisekund wskazaną w argumencie *prędkość*. Po zakończeniu animacji wywołana zostanie funkcja zwrotna.
fadeTo(*prędkość*, *przezroczystość*, [*funkcja_zwrotna*])	Zmienia płynnie przezroczystość dopasowanych elementów aż do wartości argumentu *przezroczystość*. Argument *przezroczystość* przyjmuje wartości z zakresu od 0 do 1. Wartość 0 to element całkowicie niewidoczny, a 1 to brak przezroczystości. Aby element był przezroczysty częściowo, można temu argumentowi nadać wartość np. 0.45. Efekt trwa określoną ilość milisekund wskazaną w argumencie *prędkość*. Po zakończeniu animacji wywołana zostanie funkcja zwrotna.
animate(*parametry*, [*prędkość*], [*złagodzenie*], [*funkcja_zwrotna*])	Płynnie transformuje dopasowane elementy ze stanu obecnego do stanu zdefiniowanego w argumencie *parametry* przy pomocy nowych wartości stylów. Efekt trwa określoną ilość milisekund wskazaną w argumencie *prędkość*. Argument *złagodzenie* definiuje sposób zakończenia animacji (dostępne są dwa efekty "linear" i "swing"). Po zakończeniu animacji

	wywołana zostanie funkcja zwrotna.
stop([*czyść_kolejkę*], [*na_koniec*])	Zatrzymuje aktualnie uruchomione animacje dopasowanych elementów. Argument *czyść_kolejkę* może mieć wartość *true* lub *false*. Przy wartości *true* zatrzymana zostaje cała kolejka animacji. Argument *na_koniec* może mieć wartość *true* lub *false*. Przy wartości *true* animacja przechodzi natychmiast do stanu końcowego animacji.
jQuery.fx.off	Ustawienie wartości *true* dla tej właściwości powoduje wyłączenie wszelkich animacji.

Argumenty w nawiasach kwadratowych są opcjonalne. Argument *prędkość* poza wartościami wyrażonymi w milisekundach może też przyjąć predefiniowane wartości: *slow*, *normal* lub *fast*.

Zdarzenia

Animacje mogą być wywołane dla różnych zdarzeń.

Nazwa zdarzenia	**Opis**
blur(*fn*)	Wywołuje funkcję *fn* w reakcji na opuszczenie dopasowanego elementu.
change(*fn*)	Wywołuje funkcję *fn* w reakcji na zmianę zawartości dopasowanego elementu.
click(*fn*)	Wywołuje funkcję *fn* w reakcji na kliknięcie dopasowanego elementu.
dblclick(*fn*)	Wywołuje funkcję *fn* w reakcji na dwukrotne kliknięcie dopasowanego elementu.
error*(fn)*	Wywołuje funkcję *fn* w reakcji na wystąpienie błędu w dopasowanych elementach.
focus(*fn*)	Wywołuje funkcję *fn* w reakcji na uaktywnienie dopasowanego elementu.
keydown(*fn*)	Wywołuje funkcję *fn* w reakcji na wciśnięcie klawisza w dopasowanym elemencie.
keypress(*fn*)	Wywołuje funkcję *fn* w reakcji na wciśnięcie i puszczenie klawisza w dopasowanym elemencie.
keyup(*fn*)	Wywołuje funkcję *fn* w reakcji na puszczenie klawisza w dopasowanym elemencie.
load(*fn*)	Wywołuje funkcję *fn* w reakcji na załadowanie dopasowanego elementu.
mousedown(*fn*)	Wywołuje funkcję *fn* w reakcji na wciśnięcie klawisza myszy nad

	dopasowanym elementem.
mouseenter(*fn*)	Wywołuje funkcję *fn* w reakcji na najechanie kursorem myszy na dopasowany element.
mouseleave(*fn*)	Wywołuje funkcję *fn* w reakcji na opuszczenie przez kursor myszy dopasowanego elementu.
mousemove(*fn*)	Wywołuje funkcję *fn* w reakcji na zmianę pozycji kursora myszy nad dopasowanym elementem. Możliwe jest odczytanie bieżącej pozycji
mouseout(*fn*)	Wywołuje funkcję *fn* w reakcji na opuszczenie przez kursor myszy dopasowanego elementu lub przejście pomiędzy elementami zawartymi w dopasowanym elemencie.
mouseover(*fn*)	Wywołuje funkcję *fn* w reakcji na najechanie kursorem myszy na dopasowany element lub przejście pomiędzy elementami zawartymi w dopasowanym elemencie.
mouseup(*fn*)	Wywołuje funkcję *fn* w reakcji na puszczenie klawisza myszy nad dopasowanym elementem.
resize(*fn*)	Wywołuje funkcję *fn* w reakcji na zmianę rozmiaru dopasowanego elementu.
scroll(*fn*)	Wywołuje funkcję *fn* w reakcji na przewijanie zawartości dopasowanego elementu.
select(*fn*)	Wywołuje funkcję *fn* w reakcji na zaznaczenie dopasowanego elementu.
submit(*fn*)	Wywołuje funkcję *fn* w reakcji na próbę wysłania dopasowanego elementu.
unload(*fn*)	Wywołuje funkcję *fn* w reakcji na opuszczenie dopasowanego elementu przez przeglądarkę.

Menu rozwijane

Przykładem bardziej rozbudowanego kodu jQuery, który może zostać wykorzystany na wielu stronach, będzie menu rozwijane.

Najprostsze menu rozwijane w jQuery można zrealizować dzięki odpowiednio sformatowanym dwóm listom, z których jedna jest zagnieżdżona w drugiej. Poniższy przykład prezentuj użycie listy nieuporządkowanej, ale identyczny efekt można osiągnąć dzięki liście uporządkowanej. Ze względu na poprawne wyświetlanie menu w starszych wersjach Internet Explorera elementy zagnieżdżonej listy powinny być zapisane w jednej linii bez żadnych przerw. Niezastosowanie się do tej uwagi może spowodować wyświetlanie się menu podrzędnego z przerwą pomiędzy kolejnymi pozycjami. Głównej liście nadajemy identyfikator *menu*.

```
<ul id="menu">

    <li>

         <a href="/">Oferta</a>

         <ul>

              <li><a href="xhtml">XHTML i CSS w praktyce</a>
</li><li><a href="jquery">jQuery w praktyce</a></li>

         </ul>

    </li>

    <li>

         <a href="kontakt">Kontakt</a>

    </li>

</ul>
```

Podstawowe style dla takiego menu powinny wyglądać tak jak poniżej. Pomijając właściwości odpowiadające jedynie za wygląd menu takie jak: *border*, *background-color*, *width*, *padding*, *color*, *text-decoration*, czasem nawet *margin*; pozostałe powinny wyglądać identycznie w zasadzie przy każdym wykorzystaniu takiego menu. Właściwość *list-style-type* głównej listy usuwa punktory elementów listy i jest dziedziczona przez listę podrzędną. W elementach *li* listy właściwość *display* zmienia je na elementy blokowe. Dzięki temu zajmują one tylko zadeklarowaną szerokość. Natomiast właściwość *float* powoduje ustawienie tych elementów kolejno po sobie w jednym wierszu. Ustawienie właściwości *display: block* odnośników powoduje, że klikalny jest cały obszar pozycji menu, a nie tylko sam napis.

W liście podrzędnej właściwość *display: none* ukrywa listę, a *position: absolute* sprawia, że pozostałe elementy nie odsuwają się w dół w momencie pojawienia się tej listy. Z kolei ustawienie właściwości *float: none* elementów listy podrzędnej likwiduje ustawienie dziedziczone z listy nadrzędnej.

```
#menu

{

    margin: 0;

    list-style-type: none;

    border: 1px solid #ffffff;

}
```

```
#menu li
{
    margin: 0;
    width: 200px;
    display: block;
    float: left;
    background-color: #ccffdd;
}

#menu li a
{
    padding: 8px 20px;
    display: block;
    color: #003366;
}

#menu ul
{
    margin: 0;
    padding: 0;
    display: none;
    position: absolute;
}

#menu ul li
```

```
{
    margin: 0;
    background-color: #ccddff;
    width: 270px;
    float: none;
}
```

Teraz w pliku *skrypt.js* można dla zdarzenia *mouseenter* elementu listy o *id="menu"* wywołać funkcję, która rozwinie listę podrzędną zawartą w tym elemencie. Zapis *$('ul', this)* oznacza, że dopasowane zostaną elementy *ul* zawarte w elemencie, dla którego zaszło zdarzenie, czyli rozwinięta zostanie tylko ta lista, która znajduje się wewnątrz najechanego kursorem myszy elementu *li*. Słowo *this* zawsze oznacza element, dla którego zaszło określone zdarzenie. Zdarzenie *mouseleave* jest podobne do poprzedniego. Różni się tylko tym, że funkcja rozwijająca menu podrzędne *slideDown()* została zastąpiona funkcją zwijającą *slideUp()*.

```
$(document).ready(function()
{
    $('#menu li').mouseenter(function()
    {
        $('ul', this).slideDown(400);
    });
    $('#menu li').mouseleave(function()
    {
        $('ul', this).slideUp(700);
    });
});
```

O autorze

Leszek Sewastianowicz – trener programowania używający autorskiej metody nauki programowania skoncentrowanej na rozwiązania, egzaminator ECDL (European Computer Driving Licence), programista z 11 letnim stażem, redaktor biuletynu „Sekrety Webmasterów”, autor publikacji w Software Developer’s Journal i Software Developer’s Journal Kariera. Pracował dla m.in.: Towarzystwo Naukowe Organizacji i Kierownictwa, Zakład Kształcenia i Doradztwa Ekonomicznego, T-Matic Grupa Computer Plus, Synerway S.A. Projektował i programował strony internetowe, bazy danych oraz kluczowe aplikacje wspierające wewnętrzne procesy biznesowe firmy. Autor szkoleń:

“Zawodowy Webmaster czyli jak tworzyć profesjonalne strony WWW” – szkolenie wideo na DVD - szkolenie programowania profesjonalnych stron www z zastosowaniem technologii: XHTML, CSS, PHP, MySQL, JavaScript, jQuery, Smarty i ADODB. W trakcie szkolenia, pod okiem doświadczonego trenera IT i programisty w jednej osobie, wykonasz samodzielnie 6 profesjonalnych stron www…

“Zawodowy Webmaster czyli jak tworzyć profesjonalne strony WWW” – szkolenie wideo z dostępem przez Internet - szkolenie programowania profesjonalnych stron www z zastosowaniem technologii: XHTML, CSS, PHP, MySQL, JavaScript, jQuery, Smarty i ADODB. W trakcie szkolenia, pod okiem doświadczonego trenera IT i programisty w jednej osobie, wykonasz samodzielnie 6 profesjonalnych stron www…

“Jak profesjonalnie ciąć layout na stronę WWW” – szkolenie wideo z dostępem przez Internet Dzięki temu szkoleniu nauczysz się jak ciąć grafikę w programach graficznych i osadzać ją na stronie WWW, jak przygotować graficzne menu i ramki o atrakcyjnym wyglądzie…

“Programowanie obiektowe w PHP” – szkolenie wideo z dostępem przez Internet Wyjątkowe szkolenie wideo, dzięki któremu nauczysz się krok po kroku programowania w PHP obiektowym na najwyższym poziomie…

“Bazy danych w aplikacjach internetowych” – szkolenie wideo z dostępem przez Internet Cykl szkoleń „Bazy danych w aplikacjach internetowych” sprawi, że rozszerzysz swoje możliwości zawodowe, a strony i aplikacje internetowe, które wykonasz, będą bardziej funkcjonalne i atrakcyjne. Szkolenia „Bazy danych w aplikacjach internetowych” skupiają się na bazie MySQL, ale wiele poruszanych zagadnień dotyczy każdej relacyjnej bazy danych…