Projekt JS 10 – jQuery

Wprowadzenie

jQuery to poręczna i szybka biblioteka/framework dla JavaScriptu ułatwiająca znacząco implementowanie potrzebnych funkcjonalności w tym języku takich jak obsługa AJAX, pobieranie i modyfikowanie danych w drzewie DOM strony (HTML), obsługa zdarzeń i więcej.

Pozwala ona w szybki, efektywny i dopracowany sposób tworzyć witryny internetowe korzystając z gotowych rozwiązań związanych czy to z technologią AJAX, czy z wieloma innymi aspektami.

Założenie projektu

Projekt będzie podzielony na etapy, z których każdy kończy się oddaniem (pokazaniem kawałka działającego systemu wykorzystującego technologie jQuery.

Termin

Terminy na oddanie poszczególnych etapów projektu będą ustalane indywidualnie z grupą na bieżąco tak by tempo poznawania jQuery i sposób pracy odpowiadały poziomowi grupy i poszczególnych jej członków.

Etapy

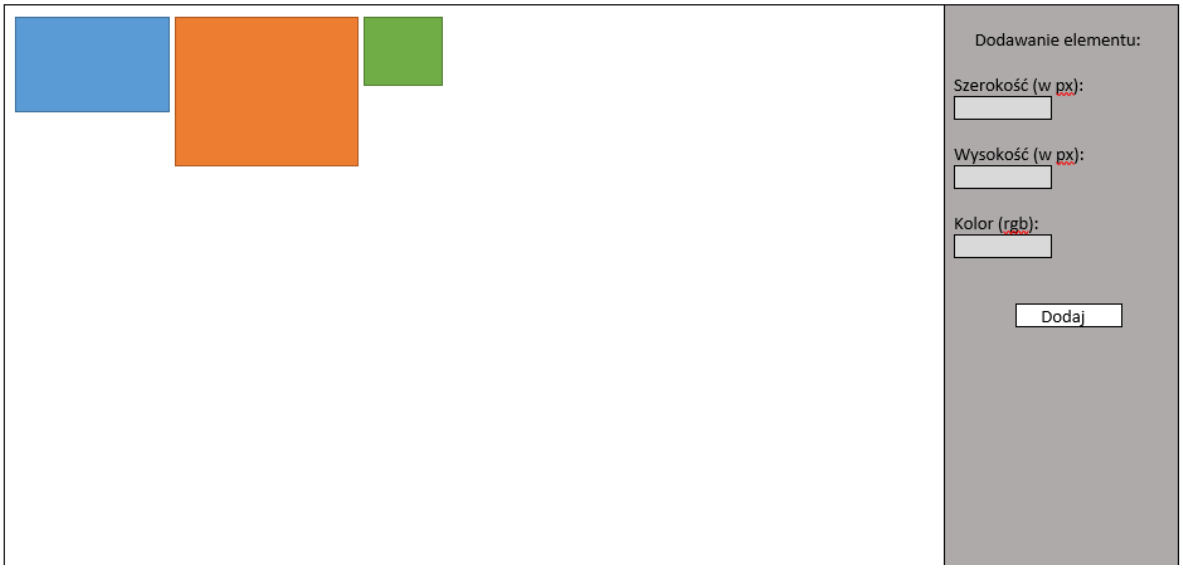
Etap 1

Etap polega na zapoznaniu się ze sposobem dołączania do swojego projektu technologii jQuery oraz podstawowymi informacjami związanymi z jego wykorzystaniem na stronie.

W efekcie ma powstać strona html (wykorzystująca technologie html+css+jQuery) pozwalająca dodawać/usuwać na stronie elementy typu DIV (które układają się sąsiadując – jeden koło drugiego aż wypełnią na szerokość okno, wówczas dalsze układanie ma miejsce poniżej).

Strona ma być podzielona na dwie sekcje (DIVy) jednej będzie służył jako pole (kontener) w którym będą dodawane/usuwane nowe elementy. Druga sekcja ma zawierać interfejs pozwalający dodać nowy element (div) wraz z określeniem jego wymiarów oraz kolory tła.

Powinno to wyglądać mniej więcej tak:



Usuwanie elementów ma się odbywać poprzez dwuklik na wybranym elemencie.

By ułatwić realizację zadanie zachęcam do zapoznania się z podstawami dot. jQuery:

Teoria (materiał ze strony: http://shebang.pl/artykuly/jquery-poczatek/)

1. Wstępna konfiguracja

Na początek należy pobrać kopię biblioteki jQuery z głównej strony pobierania. W pakiecie jQuery Starterkitznajdują się pliki HTML i CSS, z którymi będziemy pracować. Pobierz go i rozpakuj w dowolnym folderze, do którego skopiuj również plik jquery.js. Następnie otwórz pliki starterkit.html i custom.js w dowolnym edytorze oraz dodatkowo plik starterkit.html otwórz w przeglądarce internetowej.
Mamy wszystko, czego nam trzeba do utworzenia tradycyjnego programu „Witaj, świecie”.

Interesujące odnośniki:

- jQuery Starterkit

- Pobieranie biblioteki jQuery

2. Witaj, jQuery

Zaczniemy od pustej strony HTML:

```
<html>
<head>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
  // Tutaj wpiszmy nasz kod JavaScript
</script>
</head>
<body>
  <!-- Tutaj wpiszmy nasz kod HTML -->
</body>
</html>
```

Ta strona wczytuje tylko bibliotekę jquery.js — pamiętaj, aby podać prawidłową ścieżkę do swojej kopii biblioteki! W tym przykładzie przyjęto założenie, że biblioteka ten znajduje się w tym samym folderze, co plik HTML. Za pomocą komentarzy zostały oznaczone miejsca, w których będziemy dodawać kod.
Gdy używamy biblioteki jQuery do odczytywania obiektowego modelu dokumentu (DOM) lub jego modyfikowania, zdarzenia musimy zacząć dodawać dopiero, gdy model ten jest już gotowy do użycia.
Dlatego pracę zaczynamy od zarejestrowania zdarzenia ready dla dokumentu.

```
$(document).ready(function() {
   // Kod wykonywany, gdy DOM jest gotowy
 });
```

Umieszczanie alertu w tej funkcji mija się z celem, ponieważ do wyświetlenia alertu DOM nie musi być wczytany. Spróbujemy zatem czegoś bardziej wyszukanego — wyświetlimy alert w odpowiedzi na kliknięcie odnośnika.
Wpisz poniższy kod w elemencie `<body>`:

```
<a class="external-link" href="">Łącze</a>
```

Teraz wpisz poniższy kod w procedurze obsługi zdarzenia `$(document).ready`:

```
$(document).ready(function() {
  $("a").click(function() {
    alert("Witaj, świecie!");
  });
```

```
4    });
5
```

Ten kod powinien spowodować wyświetlenie alertu, gdy użytkownik kliknie odnośnik. Skrypt jest gotowy, więc możesz go przenieść do pliku custom.js. Gdy to zrobisz, otwórz stronę starterkit.html w przeglądarce i kliknij dowolny ze znajdujących się na niej odnośników. Kliknięcie któregokolwiek łącza powinno spowodować pojawić się okna dialogowego z tekstem „Witaj, świecie!”.
Przeanalizujmy dokładnie nasz kod. Ciąg `$("a")` to selektor jQuery, który wybiera wszystkie elementy a. Znak `$` jest w jQuery aliasem słowa `class`, a więc zapis `$()` oznacza utworzenie nowego obiektu jQuery. Funkcja `click()`, którą wywołujemy dalej jest metodą tego obiektu. Wiąże ona zdarzenie kliknięcia z wszystkimi wybranymi elementami (w tym przypadku z jedną kotwicą) i wykonuje podaną funkcję, gdy zdarzenie to wystąpi.
Jest to coś podobnego poniższego kodu:

```
1    <a class="external-link" href="" onclick="alert('Witaj, świecie')">Łącze</a>
```

Różnica między tym skryptem, a poprzednim jest oczywista: nie musimy pisać zdarzenia onclick dla każdego elementu. Dzięki temu całkowicie oddzieliliśmy warstwę struktury (HTML) od warstwy mechanicznej (JS), podobnie jak oddzielamy strukturę od prezentacji za pomocą CSS.
Pamiętając o tym przyjrzymy się selektorom i zdarzeniom nieco dokładniej.

Interesujące odnośniki:

- jQuery Base
- jQuery Expressions
- jQuery Basic Events

3. Znajdź mnie — selektory i zdarzenia

W jQuery elementy można wybierać na dwa sposoby. Pierwszy polega na użyciu kombinacji selektorów CSS i XPath, które przekazuje się jako łańcuchy do konstruktora jQuery (np. `$("div > ul a")`). Drugi natomiast polega na wykorzystaniu kilku metod obiektu jQuery. Obie metody można stosować na raz.
Aby wypróbować niektóre selektory, wybierzemy i zmodyfikujemy pierwszą nieuporządkowaną listę na naszej stronie internetowej.

Najpierw musimy dostać się do samej listy. Lista ta ma identyfikator `orderedlist`. W klasycznym JavaScripcie użylibyśmy do tego celu następującej instrukcji: `document.getElementById("orderedlist")`. Przy użyciu jQuery natomiast zrobimy to tak:

```
1    $(document).ready(function() {
2      $("#orderedlist").addClass("red");
3    });
```

W pakiecie początkowym znajduje się arkusz stylów zawierający klasę o nazwie `red`, która zmienia kolor tła elementu na czerwony. Dlatego po odświeżeniu strony pierwsza lista nieuporządkowana powinna mieć czerwone tło. Druga lista pozostaje bez zmian.
Teraz dodamy jeszcze klasy do elementów potomnych tej listy.

```
1  $(document).ready(function() {
2    $("#orderedlist > li").addClass("blue");
3  });
```

Powyższy kod wybiera elementy `li` znajdujące w elemencie o identyfikatorze orderedlist i dodaje do nich klasę `blue`.
Teraz zrobimy coś bardziej wyszukanego: chcemy dodawać i usuwać klasę w odpowiedzi na najechanie lub opuszczenie kursorem elementu `li`, przy czym interesuje nas tylko ostatni element naszej listy.

```
1  $(document).ready(function() {
2    $("#orderedlist li:last").hover(function() {
3      $(this).addClass("green");
4    },function(){
5      $(this).removeClass("green");
6    });
7  });
```

Dostępnych jest jeszcze wiele innych selektorów o składni podobnej do CSS i XPath. Więcej przykładów i listę wszystkich dostępnych wyrażeń można znaleźć tutaj.
Dla każdego zdarzenia typu onxxx, np. onclick, onchange, onsubmit, istnieje odpowiednik w jQuery. Istnieją też inne zdarzenia, jak `ready` i `hover`, które ułatwiają wykonywanie pewnych zadań.
Listę wszystkich zdarzeń można znaleźć w dokumentacji zdarzeń jQuery.
Już za pomocą samych tych selektorów można wiele zdziałać, ale to nie wszystko.

```
1  $(document).ready(function() {
2     $("#orderedlist").find("li").each(function(i) {
3       $(this).append( " BAM! " + i );
4     });
5   });
```

Za pomocą funkcji `find()` można wyszukiwać potomków już wybranych elementów, a zatem np. instrukcja `$("#orderedlist").find("li")` jest w zasadzie równoważna instrukcji `$("#orderedlist li")`.
Funkcja `each()` pozwala przejrzeć po kolei wszystkie elementy i przetworzyć je w dowolny sposób. Jest ona używana m. in. przez takie metody, jak `addClass()`.

W tym przykładzie funkcja `append()` została użyta do dołączenia tekstu na końcu każdego elementu.
Inną często wykonywaną czynnością jest wywoływanie metod na elementach DOM, które nie są obsługiwane przez jQuery. Może to być na przykład formularz, który trzeba zresetować po zatwierdzeniu go przy użyciu Ajaksa.

```
$(document).ready(function() {
  // Skrypt ten służy do zresetowania jednego formularza.
  $("#reset").click(function() {
    $("form")[0].reset();
  });
});
```

Powyższy kod wybiera pierwszy element formularza i wywołuje na jego rzecz metodę `reset()`. Jeśli na stronie jest więcej formularzy, można posłużyć się następującym skryptem:

```
$(document).ready(function() {
  // Ten skrypt służy do resetowania wielu formularzy na raz.
  $("#reset").click(function() {
    $("form").each(function() {
      this.reset();
    });
  });
});
```

Powyższy skrypt wybierze wszystkie znajdujące się na stronie formularze, a następnie przejrzy je wszystkie za pomocą iteracji i na każdym z nich wywoła metodę `reset()`. Zwróć uwagę, że w funkcji `.each()` słowo kluczowe `this` dotyczy bieżącego elementu. Ponadto zauważ, że ponieważ funkcja `reset()` należy do elementu formularza, a nie do obiektu jQuery, nie można zastosować zwykłego wywołania `$("form").reset()`, aby zresetować wszystkie formularze na stronie.
Dodatkowym wyzwaniem jest wybór tylko niektórych z kilku podobnych lub identycznych elementów. W jQuery do tego celu służą funkcje `filter()` i `not()`. Funkcja `filter()` wybiera tylko te elementy, które spełniają warunki jej wyrażenia, a funkcja `not()` działa odwrotnie — usuwa elementy, które spełniają warunki jej wyrażenia. Za pomocą tych funkcji można na przykład wybrać wszystkie elementy `<li>` listy nieuporządkowanej, które nie zawierają żadnych zagnieżdżonych elementów `<ul>`.

```
$(document).ready(function() {
  $("li").not(":has(ul)").css("border", "1px solid black");
});
```

Powyższa instrukcja znajduje wszystkie elementy `<li>`, które zawierają element `<ul>` i usuwa je z grupy wybranych. Dzięki temu obramowanie będą miały wszystkie elementy `<li>` z wyjątkiem tego, który zawiera zagnieżdżony element `<ul>`. Przy użyciu składni `[wyrażenie]` z XPath można wybierać elementy wg atrybutów. Poniżej znajduje się przykładowa instrukcja wybierająca wszystkie kotwice mające zdefiniowany atrybut `name`:

```
1  $(document).ready(function() {
2    $("a[name]").css("background", "#eee");
3  });
```

Kolor tła wszystkich kotwic z atrybutem name zostanie zmieniony.

Częściej jednak kotwice wybiera się nie wg atrybutu `name`, a wg atrybutu `href`. Może to powodować problemy, ponieważ przeglądarki różnie interpretują to, czym jest zawartość atrybutu `href` (uwaga: problem ten w wersjach biblioteki jQuery nowszych od 1.1.1 jest już rozwiązany). Aby dopasować tylko część wartości, można użyć selektora w postaci `*=` zamiast `=`:

```
1  $(document).ready(function() {
2    $("a[href*='/content/gallery']").click(function() {
3      // Kod dotyczący odnośników prowadzących do katalogu /content/gallery.
4    });
5  });
```

Do tej pory selektorów używaliśmy do wybierania elementów potomnych lub filtrowania wybranych grup elementów. Czasami jednak trzeba też wybrać element poprzedni lub następny, czyli tzw. element siostrzany. Wyobraź sobie na przykład stronę z odpowiedziami na najczęściej zadawane pytania, na której początkowo wszystkie odpowiedzi są ukryte i pojawiają się, gdy użytkownik kliknie pytanie. Poniżej znajduje się kod jQuery realizujący taką funkcję:

```
1  $(document).ready(function() {
2     $('#faq').find('dd').hide().end().find('dt').click(function() {
3       $(this).next().slideToggle();
4     });
5   });
```

W kodzie tym połączyliśmy kilka instrukcji w jeden łańcuch, aby zmniejszyć rozmiar kodu i zwiększyć jego szybkość działania, ponieważ element `#faq` jest wybierany tylko raz. Dzięki wywołaniu metody `end()`, pierwsze wywołanie metody `find()` nie zostaje dokończone, co pozwala nam na rozpoczęcie nowego wyszukiwania przy użyciu drugiego wywołania metody `find()` w elemencie `#faq`, zamiast w elementach dd.
W procedurze obsługi kliknięcia, funkcji przekazywanej metodzie `click()`, za pomocą instrukcji `$(this).next()` poszukujemy następnego elementu siostrzanego,

zaczynając od bieżącego elementu dt. W ten sposób szybko wybieramy odpowiedź znajdującą się za kliknięty pytaniem.
Oprócz elementów siostrzanych można także wybierać elementy rodzice (zwane także elementami nadrzędnymi lub przodkami w terminologii XPath). Możemy na przykład wyróżnić akapit będący rodzicem odnośnika, nad którym znajduje się kursor. Posłuży nam do tego poniższy kod:

```
$(document).ready(function() {
   $("a").hover(function(){
     $(this).parents("p").addClass("highlight");
   },function(){
     $(this).parents("p").removeClass("highlight");
   });
 });
```

Gdy użytkownik umieści kursor nad którymkolwiek elementem kotwicy, nastąpi odszukanie akapitu będący rodzicem tego elementu, a następnie akapitowi temu zostanie dodana i odjęta klasa `highlight`.
Zanim przejdziemy dalej, zrobimy jeden krok wstecz: biblioteka jQuery pozwala znacznie skrócić kod, co sprawia że jest on prostszy i łatwiejszy w utrzymaniu. Poniżej znajduje się skrót zapisu `$(document).ready(callback)` :

```
$(function() {
  // Kod, który ma zostać wykonany, gdy model DOM będzie gotowy do użytku.
});
```

Przenosząc to do przykładu Witaj, świecie!, otrzymamy:

```
$(function() {
  $("a").click(function() {
    alert("Witaj, świecie!");
  });
});
```

Znając podstawy możemy przejść do innych zagadnień. Zaczniemy od Ajaksa.

Interesujące odnośniki:

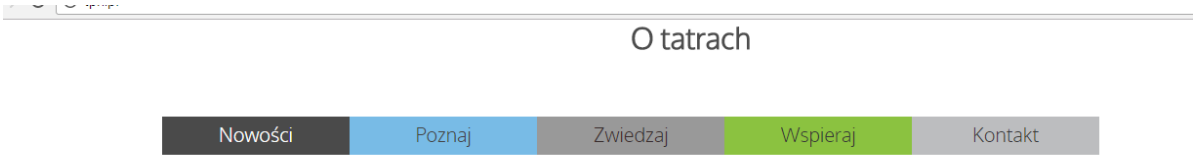
- Dokumentacja API jQuery
- Visual jQuery — dokumentacja API z podziałem na kategorie i możliwością przeszukiwania.
- Selektory jQuery

- Zdarzenia jQuery
- Przeglądanie drzewa DOM za pomocą jQuery

Etap 2

Na tym etapie należy stworzyć stronę www wykorzystującą technologie html+css+jQuery działającą zgodnie z wytycznymi:

-strona wygląda mniej więcej tak jak ta z przykładu (treści wyświetlają się pod menu w jednym div'ie):



-Treści mają być zapisane w postaci tekstu w plikach (np. z rozszerzeniem html) – same treści są do pobrania ze strony z materiałami (paczka projekt_js_10_materialy.zip)

-Treści mają się ładować odpowiednio po kliknięciu na wybrana pozycję meny – jednak tu pojawia się zasadnicza różnica – mają się one zaczytywać asynchronicznie – tj. wczytywana jest treść i podmieniana dynamicznie na stronie.

Efekt: po kliknięciu na menu, strona nie przeładowuje się, zmianie ulega jedynie treść wyświetlana poniżej menu!!!!

Teoria – pomocna do zrozumienia tematu (zaczerpnięto ze strony: http://skrypty.klocus.pl/2012/08/ladowanie-podstron-bez-odswiezania.html)

Strony, które ładują swą zawartość bez odświeżania wydają się być bardziej nowoczesne i płynniejsze w działaniu. W tym wpisie pokażę jak uzyskać taki efekt stosując bibliotekę jQuery, która zawiera elementy AJAX.

Zacznijmy od poukładania struktury folderów i plików:

```
+-[GŁÓWNY KATALOG STRONY]
|
|---->index.html
|
|--+-[js]
   |--->scripts.js
|--+-[pages]
   |--->home.html
   |--->about.html
   |--->contact.html
```

Okej, wypełnijmy teraz index.html:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <title>Moja Strona</title>
```

```
    <script type="text/javascript" src="http://code.jquery.com/jquery-1.7.2.min.js"></script>
    <script type="text/javascript" src="js/scripts.js"></script>
</head>
<body>
    <ul id="menu">
        <li><a href="home.html">Strona Główna</a></li>
        <li><a href="about.html">O Mnie</a></li>
        <li><a href="contact.html">Kontakt</a></li>
    </ul>
    <div id="content"></div>
</body>
</html>
```

Zwróćmy uwagę na to, iż między znacznik *head* zostały dołączone dwa pliki javascriptu: biblioteka jQuery oraz nasze przyszłe funkcje korzystające z tej biblioteki. Szkielet tej strony zawiera również menu w liście punktów *#menu* oraz diva *#content*, do którego będą wczytywane podstrony.

Kolejną czynnością jest stworzenie podstron. W tym celu tworzymy w katalogu *pages* następujące pliki: *home.html, about.html oraz contact.html*. Ich zawartość może być dowolna. Ja zastosowałem jedynie nagłówek oraz paragraf z treścią:

```
<h1>O Mnie</h1>
<p>Mam na imię Paweł i lubię muzykę.</p>
```

Ostatnią czynnością jest napisanie kodu JS. Tworzymy zatem nowy plik – *scripts.js* w katalogu *js* o następującej zawartości:

```
$(document).ready(function() {
    //Strona ladowana jako pierwsza:
    $('#content').load('pages/home.html');

    //Ladowanie pozostalych podstron:
    $('ul#menu li a').click(function() {
        var podstrona = $(this).attr('href');
        $('#content').html('Ładuję…');
        $('#content').load('pages/'+podstrona);
        return false;
    });
});
```

C.D.N – więcej zostawiamy na później

A na razie zachęcam do zapoznania się z polskojęzyczną stroną dot. jQuery:

http://shebang.pl/kursy/podstawy-jquery/

Szczególnie zwróć uwagę na tematykę dot. Efektów oraz Ajax– przyda się niebawem