Przetwarzanie dokumentów XML i zaawansowane techniki WWW

"Wprowadzenie do jQuery"

(Zajęcia 09 - 09.05.2016 r.)

1. Biblioteka jQuery (www.jquery.com) - wprowadzenie

Biblioteka napisana w języku JavaScript o "lekkim" charakterze, obsługująca przestrzenie nazw z mechanizmem łatwej rozszerzalności umożliwiająca manipulowanie elementami struktury DOM (Document Object Model). Najnowsze wydanie stabilne:

jQuery 2.2.3 (z 5 kwietnia 2016 r.)
lub
jQuery 1.12.3 (z 5 kwietnia 2016 r.)

Biblioteka występuje w dwóch wersjach:

a. normalnej - http://code.jquery.com/jquery-2.2.3.js
b. skompresowanej - http://code.jquery.com/jquery-2.2.3.min.js

Lub dla wersji 1.X.X

c. normalnej - http://code.jquery.com/jquery-1.12.3.js
d. skompresowanej - http://code.jquery.com/jquery-1.12.3.min.js

Dla zwiększenia wydajności aplikacji lepiej używać wersji (b). Opcjonalnie można używac wersji udostępnianej na zewnętrznym serwerze np.
https://ajax.googleapis.com/ajax/libs/jquery/2.2.3/jquery.min.js

Aby dodać funkcjonalności biblioteki jQuery do naszej aplikacji, musimy ją tą bibliotekę jako skrypt używając znaczników <script>:

```
<!DOCTYPE html>
<html>
<head>
<title>Pierwsza strona z jQuery</title>
<script type="text/javascript" src="jquery-2.1.4.min.js"></script>
</head>
<body>
 <h1>Testujemy bibliotekę jQuery</h1>
</body>
</html>
```

Pierwszą funkcją ją możemy napisac jest wyświetlenie komunikatu:

```
<head>
<title>Pierwsza strona z jQuery</title>
<script type="text/javascript" src="jquery.2.4.1.min.js"></script>
<script type="text/javascript">
      $(document).ready(function(){
        alert('Pierwsza operacja jQuery');
      });
</script>
</head>
```

Zaprezentowana konstrukcja:

```
$(document).ready( );
```

tworzy nowy obiekt na podstawie argumentu “document” i wykonuje na nim metodę “ready()”. Jest to skrótowy zapis składni jQuery gdzie znak “$” wskazuje na przestrzeń nazw “jQuery”. Konstrukcja ta w wersji pełnej wygląda nasępująco:

Dodatkowo metoda “ready()” jest tak naprawde funkcją obsługi zdarzenia “ready”, oznaczająca załadowanie całego dokumentu hipertekstowego. Natomiast funkcja która jest przekazywana jako argument funkcji “ready()” jest tzw. funkcją anonimową, (bez nazwy) i w tym konkretnym wypadku pełni rolę wywołania zwrotnego tzw. callbacka. Jako że większość, metod jQUery chemy wykonywać po załadowaniu dokumenty hiperteksotwego można tą konstrukcję skrócić jeszcze bardziej do:

```
$(function(){
  alert('Pierwsza operacja jQuery');
});
```

Funkcja ta realizuje proste wyświetlanie komunikaty “alert()”.

2. Selektory i zdarzenia w jQuery

Biblioteka jQuery wspiera operacje pozwalające na manipulowanie struktura drzewa DOM. Aby operacje te mogły być wykonywane musimy określić które elementy drzewa DOM mają zostać zmodyfikowane. W jQuery robimy to stosując te same selektory co w przypadku arkuszy styli CSS. Przykłady:

`$('div')` **wybiera** `<div> </div>`
`$('h1')` **wybiera** `<h1> </h1>`

`$('#tresc')` **wybiera** `<div id="tresc"> </div>`
`$('.tresc')` **wybiera** `<div class="tresc"> </div>`

Zatem można poruszać się po drzewie wykorzystując te same własności co w CSS. Dodatkowo obowiązują te same zależności dziedziczenia pomiędzy selektorami co w przypadku CSS.

Poza selektorami dla w jQuery możamy używać tych samych funkcji obsługi zdarzeń co w normlanym języku JavaScript umożliwiając aplikacji reagowanie na działania podejmowane przez użytkownika. Napopularniejszymi zarzeniami są: “click()”, “mouseover()”, “mouseout()”.

Przykład zastosowania selektora i zdarzenia click:

```
<script type="text/javascript">
      $(function(){
        $('#button').mouseover(function(){
              alert("Klikam w link do strony UJ");
        });
      });
</script>
```

To pokazuje jak w łatwy sposób można pisać sterowniki modyfikujące zawartość DOM.

Inny przykład: załóżmy że na stronie html mamy następujący element:

```
<div id="tresc">
 <h2> Tytuł treści </h2>
</div>
```

Chcemy następnie obsłużyć zdarzenie "click" na tym elemencie w celu wyświetlenia komunikatu za pomocą jQuery:

```
<script type="text/javascript">
      $(function(){
        $('tresc h2').click(function(){
           alert("Klikam w h2");
        });
      });
</script>
```

3. Modyfikacja wyglądu strony za pomocą CSS

W jQuery możemy wpływać na wygląd styli CSS używając metody "css()" na dowolnym elemencie. Metoda ta przyjmuje jako dwa argumenty: cechę i wartość cechy którą chcemy nadać elementowi DOM (cecha i warość muszą być podane jako string). Przykład:

```
<script type="text/javascript">
      $(function(){
        $('span').css('background', '#CCC000');
      });
</script>
```

Zatem nadaliśmy cechy pojedyńczemu elementowi. Możlwie jest również odczytywanie warości cech CSS elementów i nadawanie ich innym elementom:

```
<script type="text/javascript">
      $(function(){
          $('div#tresc').mouseover(function(){
             $('p').css('background', $(this).css('background-color'));
          }).mouseout(function(){
             $('p').css('background', 'white');
          });
      });
</script>
```

Ważne aby odczytywana cechy była cechą JEDNOWARTOŚCIOWĄ. Pojawijące się w tym kontekście słowo kluczowe "this" wskazuje na element drzewa który wywołał funkcję obsługi zdarzenia "mousover()".

Możemy poszczególnym elementom nadawać cechy CSS dodając ich do odpowiednich klas lub usuwając je z nich:

```
<style type="text/css">
   .wazny { color: red; }
```

```
</style>

<script type="text/javascript">
      $(function(){
        $('li').mouseover(function(){
           $(this).addClass('wazny');
        }).mouseout(function(){
           $(this).removeClass('wazny');
       });
      });
</script>
```

Zatem możemy przygotować w CSS odpowiednie definicje reguł i w zależności od systuacji nadawać je elementom.
W powyższym przykładzie widać również ważna cechę jQuery, tzw. “łancuch wywołań”.

Inny przykład pokazujący jak można pobrać wartość nadanej cechy i przypisać ją innemu elementowi. Załóżmy, że na stronie mamy element:

```
<div id="tresc" style="background-color: red;">
    <p> Testowa tresc </p>
</div>
```

Chemy aby po najechaniu na element “p” znajdujący się wewnątrz elementu “div” nadawać mu taki sam kolor jak posiada element “div”. W jQuery mozemy taką operacje zrealizwać następująco:

```
<script type="text/javascript">
      $(function(){
          $('div#tresc').mouseover(function(){
             $('p').css('background', $(this).css('background-color'));
          }).mouseout(function(){
             $('p').css('background', 'white');
          });
      });
</script>
```

4. Ukrywanie elementów DOM

W jQuery możemy wykorzystać trzy metody do ukrywania i pokazywania elementów: “hide()”, “show()”, “toggle()”. Przykłady użycia:

```
      $(function(){
        $('button#hide').click(function(){
           $('p').hide();
        });
        $('button#show).click(function(){
           $('p').show();
       });
      });
```

To samo na jednym elemencie można uzyskać za pomocą metody toggle:

```
$(function(){
  $('button').click(function(){
     $('span').toggle();
  });
});
```

Dodatkowo można modyfikować działanie tych funkcji za pomocą przekazywanych argumentów określających czas wykonywania danej operacji:

```
$(function(){
  $('button#hide').click(function(){
     $('p').hide("slow"); // może być również "fast"
  });
  $('button#show).click(function(){
     $('p').show(2000); // wartość w milisekundach
 });
});
```

4. Modyfikacja elementów DOM

jQuery został stworzony do modyfikacji elementów DOM. Dwom najprostszymi operacjiami jakie chcemy wykonywac jest dodawanie teksu i nowych znaczników.

```
$(function(){
  $('div#tresc1').text('Nowa treść wpisana dynamicznie w
jQuery');
  $('div#tresc2').html('Nowa treść wpisana dynamicznie w
jQuery');
});
```

Różnica polega na tym że metoda "text()" sparsuje wszystkie znaki specjalne i zamieni je na enecje HTML.

```
$(function(){
  $('div#tresc1').text('<span> To jest treść</span>');
  $('div#tresc2').html('<span> To jest treść</span>');
});
```

Proszę przetestować działałanie obu metod.

4. Modyfikacja atrybutów elementów DOM

jQuery możemy również manipulować atrybutami konkretnych elemntów DOM, służy do tego metoda "attr()". Przykłąd zastosowania:

```
$(function(){
  $('img').attr('src', 'obrazek.png');
});
<img src="" alt="obrazek"/>
```

Manipulację atrybutami elementów można wykorzystać np. Do stworzenia prostej galerii obrazków w jQuery. Załóżmy że na stronie mamy następujący kod HTML pokazujący 4 miniatury 4 obrazów. Po kliknięciu wybranej miniatury obrazek zostaje pokazany w w większym rozmiarze:
Miniatury ułożone w tabeli:

```
<tr>
   <td><img src="maly/obraz-1.png"/></td>
   <td><img src="maly/obraz-2.png"/></td>
   <td><img src="maly/obraz-3.png"/></td>
   <td><img src="maly/obraz-4.png"/></td>
</tr>
```

Element "DIV" w którym będziemy pokazywać dużą wersje obrazka:

```
<div>
   <img src="duzy/obraz-1.png" alt="obrazek" />
</div>
```

Kod jQuery obsługujący taką funkcjonalnośc znajduje się poniżej:

```
$(function(){
  $('td img').click(function(){
     var adres = $(this).atrr('src').replace('maly', 'duzy');
     $('div img').atrr('src', adres);
   });
});
```

Co robi funkcja "replace()" ?

Zadanie 1:

Proszę przygotować dokument HTML w którym będzie lista nienumerowana <ul> z kilkoma pozycjami. Następnie prosze przygotować skrypt który będzie reagował na zdarzenie click() na elemencie listy i w wyniku wywołania zwrotnego usuwał daną pozycje listy.

5. Modyfikacja drzewa DOM

Modyfikacja drzewa DOM pozwala na dodawania/usuwanie elementów ze struktury hierarchicznej. Metody służące do tego celu to:

.append( ) - Dodaje do danego węzła DOM, nowy tworzony element (węzeł) - wewnątrz (**potomek**)
.appendTo( ) - Nowy element będzie dzieckiem elementu do którego nowy element jest doklejany.
.prepend( ) - umiesza dziecko na końcu
.prependTo( ) - umieszcza dziecko na początku
.after( ) - Dodaje do danego węzła DOM, nowy element ale na zewnątrz **(brat)**
.insertAfter( ) - Nowy element są umieszcza za danym selektrem
.before( ) - umiesza brata za węzłem
.insertBefore( ) - umieszcza brata przed węzłem
.wrap( ) - Metoda ta "owija" istniejący element w nowy element podany jako argument omawianej metody.
.wrapInner( ) - Metoda ta owija zawartość wybranego elementu elementem podanym jako argument omawianej metody.

Przykład zastosowania metody "**append**" dla istniejącej struktury html w której istnieje tylko element "BODY"

```
$(function(){
   $('body').append('<div><img
      src="obraz.png"/></div>');
   });
```

1. Wybranie elementu <body> z drzewa DOM.
2. Wywołanie metody append().
3. Utworzenie węzła .
4. Dowiązanie elementu jako dziecka.

To samo zrobiony za pomocą metody **"appendTo":**

```
$(function(){
   $('<div><img src="obraz.png"/>
      </div>').appendTo('body');
});
```

1. Utworzenie węzła .
2. Wywołanie metody appendTo().
3. Wybranie elementu <body> z drzewa DOM.
4. Dowiązanie elementu jako dziecka.

Wynikiem działania obu metod jest ten sam kod HTML:

```
<body>
        <div><img src="obraz.png"/></div>
</body>
```

Podobnie działają metody **"before" i "after":**

```
$(function(){
  $('p').before('<div id="przed"><img src="obraz1.png"/></div>');
});

$(function(){
  $('p').after('<div id="po"><img src="obraz2.png"/></div>');
});

<body>
   <p>Tu jest akapit</p>
</body>
```

W wyniku otrzymujemy:

```
<body>
        <div id="przed"><img src="obraz1.png"/></div>
        <p>Tu jest akapit</p>
        <div id="po"><img src="obraz2.png"/></div>
</body>
```

Motedę "appendTo" można wykorzystać również do tworzenia zupełnie nowych elementów i nadawania im cech:

```
$(function(){
  $('<div/>',{
       'class': 'nowy',
        text: 'Nowy element blokowy',
        click: function(){
           $(this).css('color': 'red');
        }
  }).appendTo('body');
});
```

W ten sposób tworzymy element:

```
<body>
        <div class=”nowy”>Nowy element blokowy</div>
</body>
```

Który odrazu posiada atrybut oraz zawartość tekstową.

Innym często wykorzystywanym mechanizmem jest metpda “wrap” która pozwala dodawać nowe węzły owijając nowymi elementami już istniejące. Przykład dla istnijącego pojedyńczego węzła:

```
      <p>Tekst akapitu</p>
```

Możemy za pomocą “wrap”:

```
$(function(){
    $('p').wrap('<div></div>');
      });
```

Owinąć istniejący węzeł w nowy:

```
      <div>
         <p>Tekst akapitu</p>
      </div>
```

Natomiast metoda “wrapInner”: pozwoli uzyskać następujący efekt:

```
$(function(){
    $('p').wrap('<strong></strong>');
      });
```

pozwoli uzyskać następujący efekt:

```
<p><strong>Tekst akapitu</strong></p>
```