„Języki internetowe” – Laboratorium 1, 2

Framework

Definicja mówi, że to pewna struktura, wspomagająca rozwój naszych skryptów i programów. Nie jest to jednak ścisły i jednoznaczny opis. Uzupełniając go można powiedzieć, że framework to po prostu zbiór funkcji, dzięki którym możemy osiągnąć dane efekty bez powtarzania zbędnego kodu. Tak więc framework oferuje nam różne funkcjonalności, do których mamy dostęp używając funkcji, jakie przewidział twórca (bądź twórcy) narzędzia.
Popularne Fremeworks to:

- PrototypeJS http://prototypejs.org
- jQuery http://jquery.com
- ExtJS http://extjs.com
- Script.aculo.us http://script.aculo.us
- Mootools http://mootols.net
- YUI http://developer.yahoo.com/yui/
- Dojo http://dojotoolkit.org

jQuery

Czym jest jQuery?

- Opensorsowa biblioteka JavaScript.
- Narzędzie upraszczające pracę przy tworzeniu wysoko interaktywnej strony internetowej.
- Rozwiązanie zgodne ze wszystkimi nowoczesnymi przeglądarkami
- Pozwala na łatwą manipulację zawartością strony internetowej realizowaną zazwyczaj przez DOM (Dokument Object Model).
- Udostępnia wiele wyszukanych efektów.
- Pozwala na pracę ze zbiorami elementów w jednej linii kodu
  $("#main").find("p").not(".ohmy").addClass("new")
- Wykorzystuje standard CSS
- Pozwala na skalowanie poprzez wtyczki (rozszerzenia) pisane przez społeczność developerów.

Tworzenie prostej strony zawierającej jQuery

🕮Ćwiczenie 1
Plik*: zadania/01_overview/FirstjQueryPage_start.html*
Rozwiązanie: *zadania/01_overview/ FirstjQueryPage_finished.html*

Nasza pierwsza strona ma za zadanie

- Dodać referencję do biblioteki jQuery

  ```
  <script type="text/javascript" src="jquery-1.4.2.js"></script>
  ```

- Przechwycić zdarzenie załadowania strony
- Wyświetlić alert informujący o załadowaniu strony wykorzystując mechanizm przechwytywania zdarzeń jQuery

  ```
  $("document").ready(function() {
    alert("Uruchomiłeś właśnie prosty skrypt biblioteki JQuery!");
  });
  ```

Uwaga:
Plik do zadania znajdziesz w katalogu ***zadania\01_przeglad***

Przegląd funkcjonalności jQuery

Funkcje jQuery możemy podzielić na 8 kategorii

- Funkcjonalności jądra - podstawowe narzędzia
- Wyboru i przemieszczania – pozwalające na znajdowanie treści w dokumencie i nawigowanie przez ta treść.
- Manipulacji i CSS – edytowanie treści dokumentu i pracę CSS-em
- Zdarzeń
- Efektów – podstawowa animacja i przemieszczanie obiektów
- Ajax – narzędzia do pracy z Ajax-em; ładowanie treści i obróbka danych JSON
- Interfejs użytkownika – oficjalne wtyczki (UI) np. kalendarze, paski postępu itp.
- Rozszerzenia – pozwala na tworzenie własnych wtyczek opartych na podstawowych bibliotekach.

Pozyskiwanie treści ze strony

Selektory i filtry

Selektory zwracają tablicę obiektów pasujących do podanych kryteriów.
Filtry przetwarzają tablicę wyników zwróconą przez selektor w celu dalszego jej oczyszczenia.
Pamiętaj, że zwrócone obiekty nie są elementami DOM.

Podstawy selektorów jQuery

Selektorem może być:

Selektor	Cel
nazwaTag	Pobranie wszystkich elementów o danej nazwie tag-u
#identyfikator	Pobranie wszystkich elementów o danym identyfikatorze
.nazwaKlasy	Pobranie wszystkich elementów, które posiadają atrybut class z wartością nazwaKlasy
tag.nazwaKlasy	Kombinacje selektorów
tag#id.nazwaKlasy	
*	Pobiera wszystkie elementy ze strony
selektor, selektor	Znajduje jednocześnie wszystkie wskazane selektory
.klasa1.klasa2	Znajduje elementy posiadające dwie klasy: klasę1 i klasę2
rodzic>dziecko	Znajduje wszystkie elementy będące bezpośrednio dziećmi elementu rodzic
przodek potomek	Znajduje wszystkie elementy potomne, które są zawarte w elementach typu przodek
prev + next	Znajduje wszystkie elementy , które są następnikami elementu wskazanego jako prev
prev ~ siblings	Znajduje elementy które są rodzeństwem (siblings – elementami na tym samym poziomie co prev) następującym po elemencie prev

Porównanie selektorów jQuery i DOM

```
<html>
<head>
</head>
<body>
        <ul id="list1">
                <li class="a">item 1</li>
                <li class="a">item 2</li>
                <li class="b">item 3</li>
                <li class="b">item 4</li>
        </ul>
<p class="a">This is paragraph 1</p>
<p>This is paragraph 2</p>
<p class="b">This is paragraph 3</p>
<p>This is paragraph 4</p>
</body>
</html>
```

DOM	**jQuery**
Pobranie znaczników <p>	
document.getElementByTagsName(”p”);	$(”p”);
Pobranie znacznika z id ”list1”	
document.getElementByID(”list1”);	$(”#list”);
Pobranie wszystkich znaczników klasy ”a”	
Tworzenie specjalnych pętli	$(”li.a”);
**Pobieranie znaczników klasy ”b” pod warunkiem, że są wewnątrz znacznika **	
Tworzenie specjalnych pętli	$(”ul .b”);

🕮Ćwiczenie 2

Plik*: zadania/02_overview/BasicSelectors_start.htm*
Rozwiązanie: *zadania/02_overview/BasicSelectors_finished.htm*
Treść zadania*:*

1. Po załadowaniu strony pobierz wszystkie znaczniki <p> i dla wyróżnienia nadaj im obramowanie wykorzystując kod .css("border", "3px solid red");
2. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim elementom posiadającym klasę **a.**
3. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd elementowi posiadającemu id=”**list1**”
4. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom **<p>** posiadającym klasę **b**

🕮Ćwiczenie 3
Plik*: zadania/02_overview/HierCombo_start.html*
Rozwiązanie: *zadania/02_overview /HierCombo_finished.html*
Treść zadania*:*

1. Po załadowaniu strony pobierz wszystkie znaczniki **<p>** oraz **<li class="b">** i dla wyróżnienia nadaj im obramowanie wykorzystując kod .css("border", "3px solid red");
2. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom **<li class="a">** zawartym wewnątrz znacznika **<ul>**
3. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd znacznikowi **<p>** występującemu jako pierwszy po znaczniku **<ul>**
4. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom będącym rodzeństwem **<p>** znacznika o **id="list1"**

Podstawy filtrów jQuery

Tak jak napisano wcześniej filtry przetwarzają tablicę wyników zwróconą przez selektor w celu dalszego jej oczyszczenia.
Filtry podzielone są na sześć kategorii:

- Podstawowe – pozwalają na pobranie pierwszego, ostatniego elementów.
- Filtry zawartości – pozwalają na pobranie elementów, które zawierają określone treści.
- Filtry widoczności
- Filtry atrybutów – poszukują w znacznikach określonych atrybutów
- Filtry pod elementów (dzieci) – znaczniki będące w relacji rodzic – dziecko
- Filtry formularzy – wyszukiwanie w elementach formularzy

Filtry podstawowe

Filtr	Cel
:first	Pierwszy element z tablicy zwróconej przez selektor
:last	Ostatni element z tablicy zwróconej przez selektor
:even	Elementy parzyste (elementy indeksowane są od 0)
:odd	Elementy nieparzyste
:eq(n)	Element na pozycji n
:gt(n)	Elementy na pozycji większej od n
:lt(n)	Elementy na pozycji mniejszej niż n
:header	Elementy nagłówkowe (H1, H2, H3 itp.)
:animated	Elementy, które są w danej chwili animowane
:not(selektor)	Elementy, które nie pasują do danego wzorca

🕮Ćwiczenie 4
Plik*: zadania/02_overview/BasicFilters_start.html*
Rozwiązanie: *zadania/02_overview/BasicFilters_finished.html*
Treść zadania*:*

1. Po załadowaniu strony pobierz pierwszy znacznik **<p>** i dla wyróżnienia nadaj mu obramowanie wykorzystując kod .css("border", "3px solid red");

2. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd ostatniemu znacznikowi **<p>**.
3. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim parzystym znacznikom **<p>**
4. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim nieparzystym znacznikom **<p>**
5. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd pierwszemu elementowi **class="a"**
6. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim parzystym znacznikom **class="b"**
7. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom **<p>** o indeksie większym niż 1.
8. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom **<p>,** które nie są **<p>** o indeksie 2.

Filtry atrybutów

Filtr	Cel
[attribute]	Elementy posiadające dany atrybut.
[attribute=value]	Elementy posiadające określony atrybut o podanej wartości.
[attribute!=value]	Elementy posiadające określony atrybut ale nie jest on równy danej wartości.
[attribute^=value]	Elementy posiadające określony atrybut i jego wartość zaczyna się od podanego ciągu znaków.
[attribute$=value]	Elementy posiadające określony atrybut i jego wartość kończy się na podany ciąg znaków.
[attribute*=value]	Elementy posiadające określony atrybut i jego wartość zawiera określony ciąg znaków.
[attribute1][attribute2]	Elementy posiadające wszystkie podane atrybuty

🕮Ćwiczenie 5
Plik*: zadania/02_overview/AttrFilters_start.html*
Rozwiązanie: *zadania/02_overview/BasicFilters_finished.html*
Treść zadania*:*

1. Po załadowaniu strony pobierz wszystkie znaczniki **<p>** posiadające atrybut **class** i dla wyróżnienia nadaj im obramowanie wykorzystując kod .css("border", "3px solid red");
2. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom **<p>** posiadającym atrybut **id="para1"**.
3. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom **<p>** posiadającym atrybut **id** rozpoczynający się od znaków **para**.
4. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom **<p>** posiadającym atrybut **id** rozpoczynający się od znaków **para** oraz atrybut **lang** zawierający znaki **en-**

Filtry treści, widoczności

Filtr treści	Cel
:contains(text)	Elementy zawierające określony tekst
:empty	Elementy puste

:has(selektor)	Elementy zawierające przynajmniej jeden podany selektor
:parent	Pobiera wszystkie elementy będące rodzicami (zawierają przynajmniej jeden element – również tekst)
Filtr widoczności	Cel
:visible	Elementy, które są widoczne
:hidden	Elementy, które są niewidoczne

Filtrowanie elementów potomnych (dzieci)

Filtr	Cel
:nth-child(index) :nth-child(even) :nth-child(odd) :nth-child(equation)	Elementy potomne o podanym indeksie Elementy potomne parzyste Elementy potomne nieparzyste Elementy potomne o podanym równaniu (equation) Xn+M. Np. 2n –co drugi element (początkowa wartość n to 0) 3n+1 – co trzeci element +1 ***Uwaga***: w tym wypadku elementy indeksowane są od 1
:first-child	Elementy, które są pierwszym potomkiem rodziców
:last-child	Elementy, które są ostatnim potomkiem rodziców
:only-child	Elementy, które są jedynym potomkiem rodziców

🕮Ćwiczenie 6
Plik*: zadania/02_overview/ChildVisCont_start.html*
Rozwiązanie: *zadania/02_overview/ChildVisCont_finished.html*
Treść zadania*:*

1. Po załadowaniu strony pobierz wszystkie znaczniki **<p>** zawierające tekst **3** i dla wyróżnienia nadaj im obramowanie wykorzystując kod .css("border", "3px solid red");
2. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom zawierającym tekst **3**.
3. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom **<p>**, które są rodzicami.
4. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom **<ul>**, które zawierają przynajmniej jeden znacznik **<li class=”a”>**
5. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd elementowi **<li>**, który jest **3** elementem potomnym znacznika **<ul>**
6. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd elementowi **<li>**, który jest ostatnim elementem potomnym znacznika **<ul>**
7. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd co drugiemu znacznikowi **<li>,** będącemu potomkiem znacznika **<ul>.**

Filtrowanie elementów formularza

Selektor	Cel
:input	Pobiera pola input, select, textarea i button
:text	Elementy typu text
:password	Elementy typu password
:radio	Elementy typu radio
:checkbox	Elementy typu chceckbox
:submit	Elementy typu submit

:reset	Elementy typu reset
:image	
:buton	Elementy typu buton
:file	Elementy typu file
:enabled	Elementy, które są aktywne
:disabled	Elementy, które są nieaktywne
:checked	Elementy, które są zaznaczone (radio i checkbox)
:selected	Elementy, które są wybrane

🕮Ćwiczenie 7
Plik*: zadania/02_overview/FormSelectors_start.html*
Rozwiązanie: *zadania/02_overview/FormSelectors_finished.html*
Treść zadania*:*

1. Po załadowaniu strony pobierz wszystkie pola znacznika **form** i dla wyróżnienia nadaj im obramowanie wykorzystując kod .css("border", "3px solid red");
2. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom typu tekstowego w elemencie **form**.
3. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom w elemencie **form,** które są typu tekstowego i są aktywne.
4. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom w elemencie **form,** które są zaznaczone.
5. Zakomentuj poprzedni kod używając // i dodaj nowy nadający ten sam wygląd wszystkim znacznikom typu **checkbox** w elemencie **form,** które są zaznaczone.

Przemieszczanie się po dokumencie

Funkcja/własność	Cel
size(), length	Ilość elementów w zborze rezultatów jQuery
get()	Zwraca rezultat poszukiwań jako tablicę elementów DOM
get(index)	Dostęp do elementu DOM o podanym indeksie
find(expression)	Szuka elementów potomnych pasujących do podanego wyrażenia (expression)
each(fn)	Wykonuje funkcję w kontekście każdego pasującego elementu

🕮Ćwiczenie 8
Plik*: zadania/02_overview/traversing_start.html*
Rozwiązanie: *zadania/02_overview/traversing_finished.html*
Treść zadania*:*

1. Po załadowaniu strony wyświetl komunikat wykorzystując funkcję alert mówiący i ilości znaczników **<p>** w dokumencie.
2. Zakomentuj poprzedni kod używając // i dodaj nowy pobierający wszystkie elementy **<li>** jako DOM. Wynik zapisz w zmiennej **var elems.** Wyświetl alert, który powie jaki jest rozmiar tablicy elems (elems.length).
3. Zakomentuj poprzedni kod używając // i dodaj nowy, który wyświetli w funkcji **alert** informację o typie elementu DOM pobranego jako zerowy element z listy znaczników **<li>.** Aby wyświetlić informacje o typie wystarczy przekazać do funkcji alert obiekt DOM.

4. Zakomentuj poprzedni kod używając // i dodaj nowy, który wykorzystując funkcję **find** znajdzie w rezultatach selektora pobierającego znaczniki **<ul>** te elementy, które są znacznikami **<li class=”b”>** i nada im obramowanie wykorzystując kod .css("border", "3px solid red");
5. Zakomentuj poprzedni kod używając // i dodaj nowy który:
    a. Utworzy zmienną **var leftmargin=0;**
    b. Utworzy zmienną **var border=3;**
    c. Wykorzystując funkcję **each** doda do każdego znacznika **<p>** funkcję następującej postaci:

```
function(){
        //ustawia dla bieżącego elementu obramowanie o grubości równej
        wartości zmiennej border
        $(this).css(”border”, border+ ”px solid red”);

        //ustawia dla bieżącego elementu margines w rozmiarze równym
        wartości zmiennej leftmargin
        $(this).css(”margin-left”, leftmargin);

        //zwiększa wartość zmiennej border o 2
        border +=2;

        //zwiększa wartość zmiennej leftmargin o 10
        leftmargin +=10;
}
```

Deklaracje łańcuchowe

Jedna z najbardziej ułatwiających pracę funkcji jQuery jest możliwość łączenia kilku operacji w jednej linii kodu.

$(slelector).fn1().fn2().fn3();

Ćwiczenie podsumowujące

Plik*: zadania/02_overview/AutoPDFIcons_Start.htm*
Rozwiązanie: *zadania/02_overview/AutoPDFIcons_Finished.htm*
Po załadowaniu strony napisz kod, który po każdym linku (**<a>**) posiadającym atrybut **href** o wartości kończącej się na **.pdf** doda obrazek ****.
Podpowiedź:
Dla elementów zwróconych przez selektor wykorzystaj funkcję **after(kod_html);**

Manipulacja treścią strony

Jeżeli pozyskałeś za pomocą selektorów i filtrów elementy strony prawdopodobnie będziesz chciał coś z nimi zrobić. Czasami będziesz chciał tworzyć treść dynamicznie i dodawać ją do strony. jQuery daje nam możliwość tworzenia, kopiowania, kasowania i przemieszczania elementów strony. Pozwala też zmieniać i dodawać parametry CSS formatujące elementy.

Tworzenie, ustawianie i pobieranie treści

Do tworzenia nowej treści HTML można użyć łańcucha znaków prezentującego kod HTML i przekazać go do funkcji $(), np. var newHeader=$("<h1>Nowy nagłówek</h1>"); Dodatkowo można użyć metod **html()** i **text()** do pobierania i ustawiania treści.

Funkcja	Znaczenie
html()	Zwraca kod HTML z elementu wybranego przez selektor
html(New_content)	Umieszcza kod HTML w elemencie wybranym przez selektor
text()	Zwraca tekst z elementu wybranego przez selektor
text(newtext)	Umieszcza tekst w elemencie wybranym przez selektor

🕮Ćwiczenie 9
Plik*: zadania/03_manipulating/creating_start.htm*
Rozwiązanie: *zadania/03_manipulating/creating_finished.htm*
Treść zadania*:*

1. Po załadowaniu strony wyświetl komunikat wykorzystując funkcję alert prezentujący kod HTML zawarty w elemencie o **id=list1**.
2. Zakomentuj poprzedni kod używając // i dodaj nowy, który zamieni zawartość znacznika o **id=list1** na nową "
To jest nowy element listy
".
3. Zakomentuj poprzedni kod używając // i dodaj nowy, który:
    a. Utworzy zmienną **var newItem** przechowującą obiekt jQuery **$(" To jest nowy akapit ");**
    b. Zamieni kod HTML w znaczniku o **id=para2** na kod HTML nowo powstałego obiektu.
4. Zakomentuj poprzedni kod używając // i dodaj nowy, który pobierze ostatni znacznik ** ** i zamieni tekst w nim zawarty na **„to jest ostatni akapit"**

Manipulacja atrybutami

Do sprawdzania i zmiany wartości atrybutów w znacznikach jQuery udostępnia kilka funkcji:

Funkcja	Znaczenie
attr(name)	Zwraca wartość podanego atrybutu w pierwszym elemencie kolekcji na rzecz której została wywołana. Jeżeli nie ma tego atrybutu zwraca undefined
attr(properties)	Dodaje serie atrybutów do wszystkich elementów kolekcji na rzecz której została wywołana. $("img").attr({ src: "/image/hat.gif", title: "jQuery", alt: "jQuery logo"});
attr(key, value)	Dodaje pojedynczy atrybut z wartością do wszystkich elementów kolekcji na rzecz której została wywołana.
attr(key, fn)	Dodaje pojedynczy atrybut z wartością, która jest obliczana w

	przekazanej funkcji
removeAttr(name)	Usuwa podany atrybut z wszystkich elementów kolekcji na rzecz której została wywołana.

🕮Ćwiczenie 10
Plik*: zadania/03_manipulating/attributes_start.htm*
Rozwiązanie: *zadania/03_manipulating/attributes_finished.htm*
Treść zadania*:*

1. Po załadowaniu strony pobierz wszystkie znaczniki **<a>** i dodaj im atrybut o nazwie **target** z wartością **_blank.** Pozwoli to na otwieranie stron do których prowadzą linki w nowych oknach przeglądarki.
2. Zakomentuj poprzedni kod używając // i dodaj nowy, który dla wszystkich znaczników **<a>** usunie atrybut **href**.
3. Zakomentuj poprzedni kod używając // i dodaj nowy, który doda do wszystkich znaczników **<img>** dwa atrybuty jednocześnie
    a. src: ”images/Spring.jpg”
    b. alt: ”Wiosna”;

Opakowywanie, zastępowanie, usuwanie treści.

Funkcja	Znaczenie
wrap(html)	Opakowuje każdy element kolekcji na rzecz której została wywołana podanym kodem html
wrap(element)	Opakowuje każdy element kolekcji na rzecz której została wywołana podanym elementem
wrapAll(html)	Opakowuje wszystkie element kolekcji na rzecz której została wywołana podanym kodem html
wrapAll(element)	Opakowuje wszystkie element kolekcji na rzecz której została wywołana podanym elementem
wrapInner(html)	
wrapInner(element)	
replaceWith(content)	Zamienia wszystkie pasujące elementy na elementy HTML lub DOM
replaceAll(selektor)	
empty()	Usuwa wszystkie elementy potomne dla każdego elementu kolekcji na rzecz której została wywołana
remove()	Usuwa z DOM wszystkie elementy kolekcji na rzecz której została wywołana
clone()	Klonuje pasujące elementy DOM i wskazuje na sklonowany element
clone(bool)	Klonuje pasujące elementy DOM oraz ich zdarzenia i wskazuje na sklonowany element

🕮Ćwiczenie 11
Plik*: zadania/03_manipulating/wrapping_start.htm*
Rozwiązanie: *zadania/03_manipulating/wrapping_finished.htm*
Treść zadania*:*

1. Po załadowaniu strony pobierz wszystkie znaczniki **<p>** i opakuj każdy z osobna następującym kodem HTML ”** **”.

2. Zakomentuj poprzedni kod używając // i dodaj nowy, który pobierze wszystkie znaczniki **<p>** i opakuje je kodem HTML ** .**
3. Zakomentuj poprzedni kod używając // i dodaj nowy, który usunie wszystkie elementy potomne znacznika **<ul>**

Praca z CSS.

JQuery dostarcza proste i wydajne mechanizmy do ustawiania właściwości CSS, pracy z pozycjonowaniem i wymiarowaniem elementów oraz zarządzania klasami

Funkcja css()

Filtr	Znaczenie
css(name)	Zwraca wartość podanej własności CSS w pierwszym elemencie kolekcji na rzecz którego został wywołany.
css(properties)	Ustawia właściwości CSS dla każdego elementu kolekcji na rzecz którego został wywołany używając notacji obiektowej var cssObj= { 'background-color' : '#dddddd', 'font-weight' : '', 'color' : 'rgb(0,40,244)' } $(this).css(cssObj);
css(property, value)	Ustawia pojedynczą wartość stylu dla wszystkich elementów. Jeżeli zostanie wykryta wartość liczbowa to automatycznie przekształcana jest na piksele za wyjątkiem: z-index, font-weight, opacity, zoom i line-height

Funkcje do zarządzania klasami CSS

Funkcja	Znaczenie
addClass(class)	Dodaje określoną kasę/y to każdego elementu kolekcji na rzecz której została wywołana
hasClass(class)	Zwraca prawdę jeżeli podana klasa znajduje się przynajmniej w jednym elemencie kolekcji na rzecz której została wywołana funkcja
removeClass(class)	Usuwa podaną klasę/y ze kolekcji elementów
toggleClass(class)	Dodaje podaną klasę jeżeli nie istnieje lub usuwa jeżeli istnieje
tottleClass(class, switch)	Dodaje podaną klasę jeżeli switch=true, usuwa jeżeli switch=false

Funkcje CSS do zarządzania pozycją elementów

Funkcja	Znaczenie
offset()	Zwraca w pikselach aktualny offset pierwszego elementu w stosunku do dokumentu
offsetParent()	
position()	Zwraca pozycję top i left w stosunku do rodzica
scrollTop()	
scrollTop(val)	
scrollLeft()	

scrollLeft(val)	

Funkcje CSS określające rozmiar elementu

Funkcja	Znaczenie
height()	Zwraca w pikselach aktualną wysokość pierwszego elementu
height(val)	Ustawia wysokość każdego elementu
width()	Zwraca w pikselach aktualną szerokość pierwszego elementu
width(val)	Ustawia szerokość każdego elementu
innerHeight()	Zwraca wewnętrzną wysokość (bez brzegu ale z wartością wcięcia – padding) pierwszego elementu
innerWidth()	Zwraca wewnętrzną szerokość (bez brzegu ale z wartością wcięcia – padding) pierwszego elementu
outerHeight(margin)	Zwraca zewnętrzną wysokość (z obramowaniem i wcięciem) pierwszego elementu. Jeżeli wartość margin=true to wtedy margines też jest wliczany.
outerWidth(margin)	Zwraca zewnętrzną szerokość (z obramowaniem i wcięciem) pierwszego elementu. Jeżeli wartość margin=true to wtedy margines też jest wliczany.

W pliku ***zadania/03_manipulating/css_sizing.htm*** znajdziesz przykład prezentujący w jaki sposób pobrać wartości rozmiaru. Sprawdź jego działanie w różnych przeglądarkach.

Ćwiczenie podsumowujące

Plik*: zadania/ 03_manipulating/ AutoTOC_start.html*
Rozwiązanie: *zadania/03_manipulating/AutoTOC_finished.html*
W pliku została wywołana funkcja **buildBookmarks**, która jest aktualnie pusta.
Jej zadaniem jest zbudowanie spisu treści wykorzystując do tego wszystkie znaczniki **<h3>** i umieszczenie go na końcu znacznika o **id="header".** Do funkcji dodaj następujące elementy:

1. Zmienne
   var cAnchorCount =0;
   var oList= $("<ul id='bookmarksList' >");
2. Pobierz wszystkie znaczniki o nazwie przekazanej do funkcji **buildBookmarks** jako parametr **strWhichTag** które znajdują się w znacznikach ** ** za wyjątkiem ** **
3. Wykorzystując funkcję **forach** dla każdego znalezionego znacznika (z punktu 2) wykonaj funkcję, która umieści na jego początku kotwicę do której będzie można się odwołać (**""**). Pamiętaj aby określając nową zawartość znacznika nie usunąć starej.
4. Będąc dalej w funkcji z punktu 3 do obiektu **oList** dodaj używając funkcji **append** nowy element listy **$(" " + $(this).text() + "
").** W kolejnej linii zwiększ wartość zmiennej **cAnchorCount** o jeden.
5. Na końcu funkcji **buildBookmarks** dodaj kod umieszczający obiekt listy **oList** na końcu elementu o id przekazanym do funkcji jako **sBookMarkNode** (w tym wypadku header).