

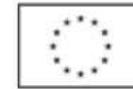


Fundusze
Europejskie
Program Regionalny



URZĄD MARSZAŁKOWSKI
WOJEWÓDZTWA POMORSKIEGO

Unia Europejska
Europejski Fundusz Społeczny



software
development
academy



Tomasz Nastały

nastalytomasz@gmail.com

Czym jest jQuery?



Jest to lekka biblioteka JS, niezależna od przeglądarki,
wydana w 2006 roku

Najnowsza wersja: 3.2.1

Kiedy stosować jQuery?



- Dużo operacji na drzewie DOM
- Obsługa asynchronicznych zapytań HTTP
- Obsługa eventów
- Tworzenie animacji / gier

Zalety jQuery



- **Wielkość** – skompresowana wersja biblioteki zajmuje około **80kB**.
- **Intuicyjność** - bazuje na **CSS**, dzięki czemu odwołania do **elementów DOM** są przyjazne dla dewelopera
- **Niezawodność** – wielokrotnie **wypróbowana i sprawdzona**. Używana na milionach stron
- **Bezpłatna**
- **Ogromna społeczność** – stale rozwijany kod oraz ogromna baza wiedzy i gotowych rozwiązań

Dodanie jQuery do aplikacji / strony



Bibliotekę można pobrać z www.jquery.com
a następnie dodać na koniec body w
dokumencie HTML

```
<body>
...
<script src="ściezka/jquery.js"></script>
</body>
```

`$(document).ready();`

Póki strona się nie załada, nie możemy nią manipulować. Jquery udostępnia nam metodę, która sprawdza czy DOM został załadowany.

```
$(document).ready(function() {  
    // tutaj kod  
});
```

```
$(function() {  
    // tutaj kod  
});
```

<- SKRÓT

1. Selektory

Czym jest selektor?



Selektor – nazwa elementu w arkuszu stylów lub nazwa tagu HTML

Selektor jQuery



```
<p>Hello world</p>

// javascript
document.getElementsByTagName("p");

// jQuery
$("p");
```

Selektor jQuery



```
<div id="container"></div>

// javascript
document.getElementById("container");

// jQuery
$("#container");
```

Selektor jQuery



```
<li class="item"></li>  
  
// javascript  
document.getElementsByClassName('item');  
  
// jQuery  
$(".item");
```

Selektor jQuery - `$(...)`



- po opakowaniu w `$(...)` nasz element zyskuje super moce – nowe metody (addClass, on, append i wiele innych, prawie 140 metod)



Selektor jQuery - \$(...)

- jeśli element wcześniej nie został opakowany w `\$(...)`, to nie ma dostępu do swoich specjalnych funkcji



Selektor jQuery - \$(...)



document.getElementById('iron-man') => \$('#iron-man')



\$(...) ===



Co siedzi pod `$(...)` ?

- funkcja `$(...)` przetwarza przekazany element w kolekcję węzłów DOM w postaci **array-like**,
- w **JS**, kolekcja array-like zachowuje się podobnie jak tablica, ale nie posiada wszystkich jej metod (map, filter etc).

Chwytyanie selektorów

- Selektory łapiemy tak jak w CSS

`$("#demo")` element o id = demo

`$(".active")` wszystkie elementy o klasie active

`$("span")` wszystkie elementy span

Chwytyanie selektorów – inne przykłady

<code>\$("")</code>	wszystkie elementy
<code>\$(this)</code>	aktualny element HTML
<code>\$("p.intro")</code>	wszystkie P z klasą intro
<code>\$("p:first")</code>	pierwszy P
<code>\$("ul li:first")</code>	pierwszy LI w pierwszym UL
<code>\$("[href]")</code>	wszystkie tagi z atrybutem href
<code>\$("tr:even")</code>	wszystkie parzyste rzędy tabeli
<code>\$("tr:odd")</code>	wszystkie nieparzyste rzędy tabeli

Spis metod jQuery

Spis dostępnych metod:

- dokumentacja: **www.jquery.com**
- wpisanie **console.dir(jQuery.prototype)** w narzędziach deweloperskich po dodaniu biblioteki jQuery do strony / aplikacji

Kontekst selektora

```
<div class="container-portfolio">
  <div class="panel">Portfolio</div>
</div>
<div class="container-contact">
  <div class="panel">Contact</div>
</div>
```

```
$(selector, context);
```

```
$(".panel", ".container-contact");
```

Selektory – dobre praktyki

Selektory zapisujemy do zmiennych (caching).
Unikamy powtarzania selektorów



```
// ŹLE
$("#demo").addClass("active");
...
$("#demo").height();
```

```
// DOBRZE
```

```
var $demo = $("#demo")
$demo.addClass("active");
...
$demo.height();
```

Przed obiektem jQuery dodajemy znak dolara – łatwe rozróżnienie obiektów jQuery od innych zmiennych



```
// ŹLE  
var demo = $("#demo")  
  
// DOBRZE  
var $demo = $("#demo")
```

Możliwie jak najprostsze selektory,
im prostszy tym szybszy i czytelniejszy.



Po „id” najlepszy performance.

```
// ŹLE
```

```
$(“div > ul li.item div#demo”);
```

```
// DOBRZE
```

```
$("#demo");
```



Zadania

2. FILTROWANIE KOLEKCJI

Filtrowanie kolekcji

jQuery umożliwia ograniczenie kolekcji elementów za pomocą 4 metod:

- eq()
- filter()
- first()
- last()

Filtrowanie kolekcji



eq(index) – ogranicza kolekcję do elementu na wybranym indeksie. Indeksy są numerowane od zera. W przypadku ujemnego indeksu, element jest liczony od końca kolekcji

```
<ul>
  <li>list item 1</li>
  <li>list item 2</li>
  <li>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

```
$( "li" ).eq( 2 ) // item 3
```

```
$( "li" ).eq( -1 ) // item 5
```

```
$( "li" ).eq( 0 ) // item 1
```

Filtrowanie kolekcji



filter(selector) – ogranicza kolekcję do elementów posiadających zadany selektor. Zamiast selektora można również przekazać funkcję wykonującą test, która spowoduje ograniczenie kolekcji do elementów, które przeszły zadany test.

```
<ul>
  <li>list item 1</li>
  <li class='red'>list item 2</li>
  <li class='red'>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

```
$( "li" ).filter(".red");
```

Filtrowanie kolekcji

first() – ogranicza kolekcję do pierwszego elementu.
Odpowiednik eq(0)

```
<ul>
  <li>list item 1</li>
  <li class='red'>list item 2</li>
  <li class='red'>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

```
$( "li" ).first()
```

Filtrowanie kolekcji

last() – ogranicza kolekcję do ostatniego elementu

```
<ul>
  <li>list item 1</li>
  <li class='red'>list item 2</li>
  <li class='red'>list item 3</li>
  <li>list item 4</li>
  <li>list item 5</li>
</ul>
```

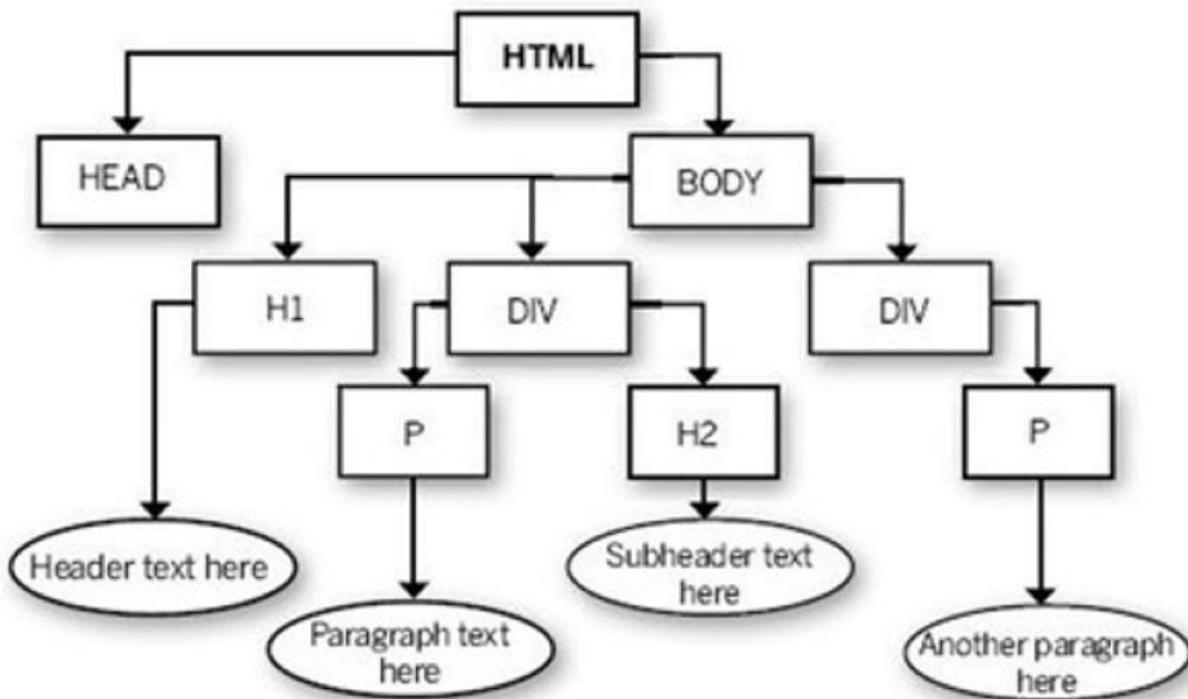
```
$( "li" ).last()
```



Zadania

3. TRAWERSOWNIE DOM

Drzewo DOM



Trawersowanie DOM



Trawersowanie drzewa DOM –
„chodzenie” po drzewie węzłów

Trawersowanie DOM



jQuery udostępnia zbiór metod do przemieszczania się po drzewie DOM

Przemieszczanie się po drzewie



Poruszanie się po drzewie DOM:

- closest()
- find()
- next()
- prev()
- parent()

Trawersowanie DOM



closest(selector) – znajduje najbliższy element o zadanym selektorze względem elementu na którym metoda została wywołana

```
<ul id="list">
  <li>list item 1</li>
  <li class='red'>list item 2</li>
  <li class='red'>list item 3</li>
  <li id="four">list item 4</li>
  <li>list item 5</li>
</ul>
```

```
$("#four").closest("#list")
```

Trawersowanie DOM



find(selector) – wyszukuje elementy o zadanym selektorze w dół drzewa względem elementu na którym metoda została wywołana

```
<ul id="list">
  <li>list item 1</li>
  <li class='red'>list item 2</li>
  <li class='red'>list item 3</li>
  <li id="four">list item 4</li>
  <li>list item 5</li>
</ul>
```

```
$("#list").find("#four")
```

Trawersowanie DOM



prev(selector) – wyszukuje wcześniejszych braci o danym selektorze. W przypadku braku parametru, chwyta najbliższy, wcześniejszy węzeł na tym samym poziomie drzewa

```
<ul id="list">
  <li>list item 1</li>
  <li class='red'>list item 2</li>
  <li class='red'>list item 3</li>
  <li id="four">list item 4</li>
  <li>list item 5</li>
</ul>
```

```
$("#four").prev(".red")
```

```
$("#four").prev()
```

Trawersowanie DOM



next(selector) – wyszukuje kolejnych braci o danym selektorze. W przypadku braku parametru, chwyta najbliższy, kolejny węzeł na tym samym poziomie drzewa

```
<ul id="list">
  <li>list item 1</li>
  <li class='red'>list item 2</li>
  <li class='red'>list item 3</li>
  <li id="four">list item 4</li>
  <li>list item 5</li>
</ul>
```

```
$("#four").next()
```

```
$(".red").next("#four")
```

Trawersowanie DOM



parent() – chwyta bezpośredniego rodzica elementu, na którym metoda została wywołana

```
<ul id="list">
  <li>list item 1</li>
  <li class='red'>list item 2</li>
  <li class='red'>list item 3</li>
  <li id="four">list item 4</li>
  <li>list item 5</li>
</ul>
```

```
$("#four").parent()
```

Trawersowanie – dobre praktyki

Nie używamy wielokrotnie parent()

// ŹLE

```
$("#demo").parent().parent().parent()
```

// DOBRZE

```
$("#demo").closest(".list");
```

Nie używamy wielokrotnie find()

// ŹLE

```
$("#demo").find("ul").find("li").find("#my-item")
```

// DOBRZE

```
$("#demo").find("#my-item")
```

// NAJLEPIEJ – JEŚLI NIE TRZEBA, NIE UŻYWAĆ FIND

```
$("#my-item")
```



Zadania

4. MANIPULACJA DOM

Manipulacja drzewem DOM



Możliwa manipulacja poprzez dodawanie, edytowanie i usuwanie węzłów.

Manipulacja klasami elementu



jQuery udostępnia 4 metody do manipulacji klasami:

- addClass()
- hasClass()
- removeClass()
- toggleClass()

Manipulacja klasami elementu

addClass(nazwaKlasy) – dodaje klase do wybranego elementu

```
var $container = $("container");

$container.addClass("highlight");
$container.addClass("sidebar active");
```

Manipulacja klasami elementu

hasClass(nazwaKlasy) – sprawdza, czy element posiada daną klasę.

Jeśli posiada – zwraca true

Jeśli nie posiada – zwraca false

```
var $container = $("container");
$container.addClass("highlight");

if ($container.hasClass("highlight")) {
    console.log('yes, has highlight class');
} else {
    console.log("no, doesn't have a highlight class");
}
```

Manipulacja klasami elementu

removeClass(nazwaKlasy) – usuwa wybraną klasę
removeClass() – usuwa wszystkie klasy

```
var $container = $("container");
$container.addClass("highlight active");

$container.removeClass("highlight");
$container.removeClass();
```

Manipulacja klasami elementu



toggleClass(nazwaKlasy) – przedstawiają klasę. Jeśli nie istnieje, to ją doda, jeśli istnieje, to usunie.

```
var $container = $('#container');

$container.on('click', function() {
  $(this).toggleClass('highlight');
});
```

Manipulacja CSS



Metoda `css()` umożliwia dodanie reguł CSS do wybranego elementu.

Jeśli chcemy dodać więcej niż jedną regułę, to przekazujemy obiekt z regułami.

Manipulacja CSS



```
var $container = $('#container');

$container.css('background-color', 'red');

$container.css({
  'background-color': 'red',
  'height': '200px',
  'width': '400px',
  'border': '1px solid blue'
});
```

Manipulacja CSS



Metoda `css()` umożliwia dodanie reguł CSS do wybranego elementu.

Tworzenie węzłów w środku elementu

- prepend()
- prependTo()
- append()
- appendTo()
- html()
- text()

Tworzenie węzłów

3. Before

Element na którym wywołujemy metody

2. Prepend

AKTUALNA ZAWARTOŚĆ

1. Append

4. After

Tworzenie węzłów



```
var $container = $('#container');
var $row = $('.row');

$container.prepend($row);
$row.prependTo($container);

$container.append($row);
$row.appendTo($container);

$container.html($row);
$row.text('jakiś tekst');
```

Tworzenie węzłów



```
var $container = $('#container');
var $row = $('.row');

$container.prepend($row); // na początek zawartości $containera dokładamy $row

$row.prependTo($container); // $row zostaje dodany na początek zawartości $containera

$container.append($row); // na koniec zawartości $containera dokładamy $row

$row.appendTo($container); // $row zostaje dodany na koniec zawartości $containera

$container.html($row); // całą zawartość $containera wypełnieniamy $row, wszystko co było, znika

$row.text('jakiś text'); // dodajemy text
```

Tworzenie węzłów poza elementem



- after()
- before()

Tworzenie węzłów poza elementem



```
var $container = $('#container');
var $row = $('.row');

$container.after($row);

$container.before($row);
```

Tworzenie węzłów poza elementem



```
var $container = $('#container');
var $row = $('.row');
```

```
$container.after($row); // za $containerem zostanie dodany $row
```

```
$container.before($row); // przed $containerem zostanie dodany $row
```

Usuwanie węzłów



- `empty()`
- `remove()`

Tworzenie węzłów poza elementem



```
var $container = $('#container');
var $row = $('.row');

$container.append($row);

$row.remove(); // $row zostaje usunięty
$container.empty(); // cała zawartość $container zostaje usunięta
```

Manipulacja DOM – dobre praktyki

Nie używamy CSS do nadania wielu reguł CSS.
Lepiej użyć addClass() i zbudować odpowiednią klasę w arkuszu stylów.



// ŹLE

```
$container.css({  
  'background-color': 'red',  
  'height': '200px',  
  'width': '400px',  
  'border': '1px solid blue'  
});
```

// DOBRZE

```
$container.addClass('my-container');  
  
// arkusz stylów  
.my-container {  
  'background-color': 'red',  
  'height': '200px',  
  'width': '400px',  
  'border': '1px solid blue'  
};
```

Operacje na drzewie DOM są kosztowne.

Warto się zastanowić, czy budować cały węzeł DOM i dodać go dynamicznie, czy stworzyć go w HTML i przestawić tylko display (z none na block np.) w razie potrzeby ukazania elementu.



Zadania

5. ZDARZENIA (EVENTS)

EVENTS



jQuery pozwala nam w łatwy sposób przypinać eventy do wybranych elementów.

EVENTS



Przykłady eventów:

- click
- mouseenter
- mouseleave
- keypress
- keyup
- scroll

I wiele więcej:

<https://developer.mozilla.org/en-US/docs/Web/Events>

EVENTS

```
var $myDiv = $("#my-div");

// wariant 1 - słaby
$myDiv.click(function() {
    $(this).css('background-color', 'lime');
});

// wariant 2 - średni
$myDiv.on('click', function() {
    $(this).css('background-color', 'lime');
});

// wariant 3 - najlepszy
function highlight() {
    $(this).css('background-color', 'lime');
}

$myDiv.on('click', highlight);
```



element.on(event, handler);

Event: nazwa eventu

Handler: funkcja, która ma zostać zwołana

EVENTS – OBIEKT EVENT



Parametr event – jest to obiekt, który reprezentuje event

```
var $myDiv = $("#my-div");

function highlight(event) {
    event.stopPropagation();
    $(this).css('background-color', 'lime');
}

$myDiv.on('click', highlight);
```

HANDLER Z PARAMETREM



Handler (funkcja) może posiadać wyłącznie parametr event. Jeśli chcemy aby posiadał więcej parametrów, to handler musi zwracać funkcję

HANDLER Z PARAMETREM



Błędne rozwiązanie – nie zadziała

```
var $myDiv = $("#my-div");

function highlight(color) {
    $(this).css('background-color', color);
}

$myDiv.on('click', highlight('red'));
```

HANDLER Z PARAMETREM



Działające rozwiązanie – handler musi zwracać funkcję

```
var $myDiv = $("#my-div");

function highlight(color) {
    return function() {
        $(this).css('background-color', color);
    };
}

$myDiv.on('click', highlight('red'));
```

EVENT DELEGATION

Problem: dodany kolejny element listy nie będzie miał przypiętego eventu

```
function highlight() {  
    $(this).css('background-color', 'red');  
}  
  
$('li').on('click', highlight);
```

```
var myList = $('#my-list');  
myList.append($('<li>'));
```

<- Brak eventu na nowym LI

EVENT DELEGATION

Obejście problemu – rodzic nasłuchuje na event i deleguje event do wybranego dziecka

```
var myList = $('#my-list');

function highlight() {
    $(this).css('background-color', 'red');
}

myList.on('click', 'li', highlight);

myList.append($('<li>').text('Hello SDA'));
```

Element.on(event, selector,
handler)

Element wydeleguje event do
zadanego selektora

<- nowy LI nasłuchuje na click

ODPIĘCIE EVENTU

Za pomocą metody **.off(event)** możemy odpiąć wybrany event. Brak przekazania nazwy eventu, odepnie wszystkie istniejące eventy.

```
var myList = $('#my-list');

function highlight() {
    $(this).css('background-color', 'red');
}

myList.on('click', 'li', highlight);

myList.off();
myList.off('click');
```

EVENTS – dobre praktyki

Handlery wynosimy do zewnętrznych funkcji,
dzięki temu nie musimy się wczytywać w kod,
aby dowiedzieć się, jaka akcja zachodzi na
danym event

```
// ŹLE
var $myList = $("#my-list");

$myList.on('click', 'li', function() {
    $(this).css('background-color', 'red');
});
```

```
// DOBRZE
var $myList = $("#my-list");

function highlightListItem() {
    $(this).css('background-color', 'red');
}

$myList.on('click', 'li', highlightListItem);
```



Zadania



6. CHAINING

CHAINING

Chaining (łańcuchowanie) – możliwość wywołania wielu metod na jednym elemencie w pojedynczej deklaracji

Jest to możliwe dzięki temu, że metody jQuery wykonują operację a następnie zwracają aktualny stan elementu (obiektu jQuery)

CHAINING

```
$myDiv
  .addClass('active')
  .attr('id', 'my-div')
  .css('background-color', 'red')
  .on('click', highlight);
```

Kolejne metody dodajemy po kropce.

Częstym błędem jest odruchowe dodawanie średnika na końcu metody

CHAINING

Nie wszystkie metody jQuery mogą być łańcuchowane.

Przykłady metod, które nie mogą wziąć udziału w chainingu:

- `height()`, ponieważ zwraca `number`
- `hasClass()`, ponieważ zwraca `booleana`

CHAINING

Aby metoda mogła wziąć udział w procesie łańcuchowania, musi zwracać obiekt jQuery. Możemy to sprawdzić w dokumentacji:

.addClass()

Categories: [Attributes](#) | [Manipulation](#) > [Class Attribute](#) | [CSS](#)

.addClass(className)

Returns: [jQuery](#)

Description: Adds the specified class(es) to each element in the set of matched elements.

 **.addClass(className)**

version added: 1.0

className

Type: [String](#)

CHAINING – dobre praktyki

Kolejne metody, wywołujemy w nowych liniach.
Taka praktyka poprawia czytelność.

```
// ŹLE
$myDiv.addClass('active').attr('id', 'my-div').css('background-color', 'red').on('click', highlight);
```

```
// DRAMAT...
$myDiv.addClass('active')
$myDiv.attr('id', 'my-div')
$myDiv.css('background-color',
'red')
$myDiv.on('click', highlight);
```

```
// DOBRZE
$myDiv
  .addClass('active')
  .attr('id', 'my-div')
  .css('background-color', 'red')
  .on('click', highlight);
```

7. ANIMACJE

ANIMACJE



jQuery umożliwia nam tworzenie animacji za pomocą metody `animate()`.

Udostępnia nam również parę prostych animacji:

- `fadeOut()`
- `fadeIn()`
- `fadeToggle()`
- `slideUp()`
- `slideDown()`
- `slideToggle()`
- `show()`
- `hide()`
- `delay()` (opóźnienie)

ANIMACJE

W każdej z metod jako parametr, możemy przekazać czas:

```
slideUp(3000)  
slideDown(200)  
slideToggle('slow')  
fadeOut(fast')  
delay
```

ANIMACJE



```
var $container = $('#container');
var $button = $('#my-btn');

$button.on('click', function() {
  $container.delay(2000).slideToggle('slow');
});
```



Zadania

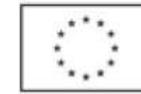


Fundusze
Europejskie
Program Regionalny



URZĄD MARSZAŁKOWSKI
WOJEWÓDZTWA POMORSKIEGO

Unia Europejska
Europejski Fundusz Społeczny



software
development
academy

KONIEC! (DNIA 1;)