WYKŁAD 5
JQUERY C.D.

selektory zaawansowane, filtry, kolekcje, łańcuchy wywołań

dr inż. Jacek Paluszak

Selektory zaawansowane – elementy potomne

Selektory elementów potomnych pozwalają odwołać się do znacznika umieszczonego wewnątrz innego.

```
<ul id="navBar">
    <a …>…
    <a …>…
    …
</ul>
```

`$('a')`	pobierze wszystkie odnośniki na stronie
`$('#navBar a')`	standardowa postać selektora CSS, pobierze wyłącznie odnośniki umieszczone wewnątrz listy

Selektory zaawansowane – dzieci

Selektory dzieci pozwalają pobierać elementy będące dziećmi innych elementów. „Dziecko" to **bezpośredni** element potomny innego elementu. Selektor dziecka tworzy się, zapisując najpierw element rodzica, następnie znak mniejszości > i w końcu element dziecka.

```
$('body > p')
```

Selektory zaawansowane – elementy sąsiadujące

Selektory elementów sąsiadujących pozwalają pobierać znaczniki, które w kodzie HTML są umieszczone bezpośrednio za jakimiś innymi znacznikami.

```
$('h2 + div')
```

Selektor umieszczony z prawej strony określa elementy, jakie należy pobrać, przy czym muszą one być poprzedzone elementami pasującymi do selektora umieszczonego z lewej strony znaku +.

Selektory zaawansowane – atrybuty

Selektory atrybutów pozwalają pobierać elementy na podstawie tego, czy posiadają konkretne atrybuty, a nawet, czy atrybuty te posiadają ściśle określone wartości.

np.

```
$('img[alt]')
```

Selektor atrybutu jest umieszczany za nazwą elementu, którego atrybuty chcemy sprawdzać.

JQUERY

Selektory zaawansowane – atrybuty

`[atrybut]` pobierają elementy, w których kodzie HTML został podany konkretny atrybut

np.

`$(a[href])` znajdzie wszystkie znaczniki <a>, w których została podana wartość atrybutu href.

Używając takiego selektora, można **pominąć** wszystkie nazwane odnośniki:

```
<a name="jakiesMiejsceStrony"></a>
```

czyli odnośniki używane do poruszania się w obrębie tej samej strony.

JQUERY

Selektory zaawansowane – atrybuty

`[atrybut="wartość"]` pozwala pobrać elementy, w których konkretny atrybut ma określoną wartość

np.

`$('input[type="text"]')` pozwala pobrać wszystkie pola tekstowe formularza

Selektory zaawansowane – atrybuty

`[atrybut^="wartość"]` odnajduje elementy, w których wartość określonego atrybutu **rozpoczyna** się od podanego ciągu znaków.

np.

`$('a[href^="http://"]')` wszystkie odnośniki wskazujące strony spoza naszej witryny

Nie cała wartość atrybutu musi pasować do łańcucha podanego w selektorze, a jedynie jej **początek**.

Selektory zaawansowane – atrybuty

`[atrybut$="wartość"]` odnajduje elementy, w których wartość określonego atrybutu *kończy się* podanym ciągiem znaków, co jest doskonałym sposobem odnajdywania rozszerzeń plików.

np.

`$('a[href$=".pdf"]')` selektor pozwalający na pobranie wszystkich odnośników do plików PDF

Nie cała wartość atrybutu musi pasować do łańcucha podanego w selektorze, a jedynie jej **koniec**.

JQUERY

Selektory zaawansowane – atrybuty

`[atrybut*="wartość"]` pozwala pobrać wszystkie elementy, których określony atrybut zawiera podany ciąg znaków.

np.

`$('a[href*="github.com"]')` selektor jest na tyle elastyczny, że pozwala także na pobieranie odnośników wskazujących stronę:
https://guides.github.com
https://blog.github.com
https://github.com/new
…

JQUERY

Filtry

Biblioteka jQuery zapewnia także możliwość:

- filtrowania pobieranych elementów na podstawie ich pewnych cech charakterystycznych
- wyszukiwania elementów zawierających:
  - podane inne elementy
  - określony tekst
  - elementy, które nie są aktualnie widoczne,
  - elementy niepasujące do podanego selektora.

Aby użyć takiego filtra, za głównym selektorem należy umieścić **dwukropek** i podać nazwę filtra.

JQUERY

Filtry

`$('tr:even')`	wszystkie parzyste wiersze tabeli
`$('tr:odd')`	wszystkie nieparzyste wiersze tabeli
`$('.stripped tr:even')`	wszystkie parzyste wiersze tabeli należącej do klasy stripped.

JQUERY

Filtry

`:first` oraz `:last` zwracają odpowiednio pierwszy i ostatni element z grupy.

Aby na przykład pobrać pierwszy akapit strony, należy użyć następującego wywołania:

```
$('p:first');
```

Z kolei poniżej przedstawiono wywołanie pozwalające na pobranie ostatniego akapitu:

```
$('p:last');
```

JQUERY

Filtry

`:not()` można użyć, by odszukać elementy, które nie pasują do podanego selektora.

np.

`$('a:not(.navButton)');` wszystkie znaczniki `<a>` z wyjątkiem tych, które należą do klasy `navButton`.

`$('a:not([href^="http://"])')` wszystkie odnośniki, których adresy nie zaczynają się od `http://`

W wywołaniu funkcji `:not()` przekazywany jest selektor, który chcemy **ignorować**.

JQUERY

Filtry

`:has()` zwraca wszystkie elementy zawierające inny, podany selektor.

np.

`$('li:has(a)');` wszystkie znaczniki `<li>`, jednak wyłącznie wtedy, gdy wewnątrz nich umieszczony jest znacznik `<a>`.

Takie rozwiązanie różni się znacząco od selektora elementów potomnych, gdyż pozwala pobrać nie elementy `<a>`, lecz elementy `<li>` zawierające wewnątrz jakieś odnośniki.

JQUERY

Filtry

`:contains()` zwraca wszystkie elementy zawierające podany tekst.

np.

`$('a:contains(Kliknij mnie!)');` wszystkie odnośniki z napisem Kliknij mnie!

JQUERY

Filtry

`:hidden()` odnajduje elementy ukryte, czyli takie, których:

- właściwość `display` CSS ma wartość `none` (co oznacza, że nie są one wyświetlane na stronie),
- elementy ukryte przy użyciu funkcji `hide()`
- elementy o wysokości lub szerokości wynoszącej zero
- ukryte pola formularzy.

np.

```
$('div:hidden').show();
```

pobranie i ponowne wyświetlenie ukrytych elementów `<div>`.

`:visible` jest przeciwieństwem filtra `:hidden`. Zwraca on wszystkie widoczne elementy strony.

JQUERY

Kolekcje

Wybierając elementy strony przy użyciu obiektu jQuery:

```
$('navBar a')
```

nie otrzymujemy tradycyjnej listy węzłów DOM,
takich jak zwracane przez metody:

`getElementById()` lub `getElementsByTagName()`.

Zamiast tego zwracana jest specjalna, charakterystyczna dla biblioteki jQuery kolekcja elementów.

JQUERY

Kolekcje

Podczas korzystania ze standardowych metod DOM dysponujemy zazwyczaj grupą elementów strony, a następnie musimy utworzyć pętlę.

Ponieważ przetwarzanie elementów kolekcji w pętli jest tak często realizowaną czynnością, została ona **wbudowana** w funkcje biblioteki jQuery.

Wykonując jakąś funkcję jQuery na grupie elementów, nie musimy jawnie tworzyć pętli, gdyż funkcja wykona ją **automatycznie**.

JQUERY

Kolekcje

`$('#slideshow img').hide();` zostaną ukryte **wszystkie** obrazki umieszczone wewnątrz znacznika `<div>` o identyfikatorze `slideshow`.

Kolekcja znaczników pobranych przez wywołanie `$('#slideshow img')` może liczyć na przykład 50 elementów.

Funkcja `hide()` automatycznie pobierze każdy z nich i go ukryje.

JQUERY

Łańcuchy wywołań funkcji

```
$('#popUp').width(300).height(300);
```

Biblioteka jQuery korzysta z rozwiązania nazywanego **łańcuchami wywołań**, które pozwala zapisywać kilka wywołań funkcji jQuery jedno bezpośrednio za drugim.

Wywołanie każdej funkcji jest połączone z następnym przy użyciu **kropki** , a każda funkcja operuje na tej samej kolekcji elementów, co poprzednia.

A zatem powyższe wywołanie najpierw zmienia szerokość elementu o identyfikatorze `popUp`, a następnie jego wysokość.

JQUERY

Łańcuchy wywołań funkcji

```
$('#popUp').width(300).height(300).text('Siema!').fadeIn(1000);
```

Kod ten wywołuje cztery funkcje jQuery:

- `width()`
- `height()`
- `text()`
- `fadeIn()`

przy czym każda z nich modyfikuje element o identyfikatorze popUp.

JQUERY

Łańcuchy wywołań funkcji

Długie sekwencje wywołań funkcji jQuery mogą być mało czytelne, dlatego też wielu programistów zapisuje je w osobnych wierszach:

```
$('#popUp').width(300)
.height(300)
.text('Siema!')
.fadeIn(1000);
```

Jeśli tylko średnik zostanie umieszczony wyłącznie za **ostatnim** wywołaniem, interpreter JavaScriptu potraktuje taki kod jak jedną instrukcję.

W takiej sekwencji można używać wyłącznie **wbudowanych funkcji biblioteki jQuery** (czyli nie można skorzystać ani z funkcji pisanych samodzielnie, ani z wbudowanych funkcji języka JavaScript).

DZIĘKUJĘ ZA UWAGĘ