jQuery

Cześć!

Łukasz Krawczyk

Web Developer / Graphic & Web Designer

Czym jest jQuery?

Czym jest jQuery?

Czym jest jQuery?

To jedna z najbardziej znanych **bibliotek programistycznych** dla języka JavaScript. Powstała 2006-08-26, a jej twórcą jest John Resig (USA).

Z początku był to **zbiór prostych funkcji** mających na celu **ułatwiających**:

- odnajdywania elementów strony
- przypisywania zachowań elementom strony
- modyfikowania drzewa DOM
- tworzenia prostych animacji

W ciągu kilku lat, jQuery stało się potężnym i bardzo wydajnym narzędziem, wykorzystywanym coraz częściej przez twórców aplikacji internetowych. Biblioteka jQuery **nie jest standardem W3C**.

Zalety jQuery

Zalety
jQuery

1. udostępnienie pewnego poziomu abstrakcji, dzięki któremu nie musimy bać się, że nasz kod zadziała błędnie lub w ogóle nie zadziała pod którąś z przeglądarek (niezależność od przeglądarki)

2. obsługa selektorów zgodna z CSS3

3. rozszerzalność samej biblioteki (multum naprawdę świetnych, dobrze działających i przydatnych dodatków)

4. uproszczenie wielu zadań — tworzenie animacji, operacji na drzewie DOM, ingerencji w style CSS i zawartość elementów strony, a także wspomaganie dla technologii (a właściwie metodologii) Ajax

Zalety
jQuery

5. uproszczenie podstawowych konstrukcji, takich jak selektory czy pętle
6. wygoda tworzenia wtyczek
7. małe rozmiary wersji produkcyjnej
8. generowany kod jest bardziej zwięzły

Wady jQuery

Wady
jQuery

1. wydajność — ta sama funkcjonalność dobrze napisana w czystym języku JavaScript będzie działała szybciej niż przy wykorzystaniu biblioteki programistycznej

$(document)
.ready(...)

$(document) .ready(...)

Zapis **$(document)** jest równoważny zapisowy **jQuery(document)** i oznacza **wybranie całego dokumentu**, np.

- ```
  $(document).ready(function() {
      alert('Hello, world!'); // kod inicjalizacyjny
  });
  ```

- ```
  jQuery(document).ready(function() {
      alert('Hello, world!'); // kod inicjalizacyjny
  });
  ```

$(document) .ready(...)

Zapis `$(document).ready(...)` jest równoważny zapisowy `$(...)` i oznacza **funkcję**, która zostanie wywołana jeśli **dokument jest gotowy** (struktura drzewa DOM, pliki .css, skrypty .js), np.

- ```
  $(document).ready(function() {
      alert('Hello, world!'); // kod inicjalizacyjny
  });
  ```

- ```
  $(function() {
      alert('Hello, world!'); // kod inicjalizacyjny
  });
  ```

Obsługa zdarzeń

Obsługa zdarzeń

W celu obsługi zdarzeń musimy wiedzieć:

- w jaki sposób **wskazać**, o który **element** chodzi — **selektory**, np. `$('p')`
- w jaki sposób **oprogramować** wybrane **zdarzenie** elementu — **metody jQuery**, np. `.click()`
- jak w treści obsługi zdarzenia **wykonać** pewną **akcję**, np. wymienić treść tego elementu — **funkcja zwrotna** (ang. *callback*), np. `function() { alert('Hello, world!'); }`

Obsługa zdarzeń

Skrypt jQuery ilustrujący oprogramowanie zdarzenia `onclick` akapitu `p`:

```
$(function() {
    $('p').click(function() {
        alert('Hello, world!');
    });
});
```

- `$('p')` — **selektor**
- `.click()` — **metoda jQuery**
- `function() { alert('Hello, world!'); }` — **funkcja zwrotna** (ang. *callback*)

Obsługa zdarzeń

Zdarzenie HTML	Metoda jQuery	Wystąpienie zdarzenia
`onclick`	`click()`	Kliknięcie elementu
`ondblclick`	`dblclick()`	Podwójne kliknięcie elementu
`onmouseup`	`mouseup()`	Zwolnienie przycisku myszy
`onmousedown`	`mousedown()`	Naciśnięcie przycisku myszy

Obsługa zdarzeń

Zdarzenie HTML	Metoda jQuery	Wystąpienie zdarzenia
onmouseover	mouseover()	Najechanie wskaźnikiem myszy
onmouseenter	mouseenter()	
onmouseout	mouseout()	Zjechanie wskaźnikiem myszy poza obszar elementu
onmouseleave	mouseleave()	

Obsługa zdarzeń

Zdarzenie HTML	Metoda jQuery	Wystąpienie zdarzenia
`onmousemove`	`mousemove()`	Wskaźnik myszy przemieszcza się w ramach obszaru wskazanego obiektu
`onselect`	`select()`	Zaznaczenie tekstu **(nawet jakiegoś jednego znaku)** w `textarea` albo w tekstowym elemencie `input`

Obsługa zdarzeń

Zdarzenie HTML	Metoda jQuery	Wystąpienie zdarzenia
`onkeypress`	`keypress()`	Naciśnięcie **logiczne (case sensitive)** przycisku na klawiaturze — nie jest uruchamiane przez klawisze funkcyjne, np. Ctrl, Shift
`onkeydown`	`keydown()`	Naciśnięcie **fizyczne** przycisku na klawiaturze
`onkeyup`	`keyup()`	Zwolnienie przycisku na klawiaturze

Obsługa zdarzeń

Przykład. Obsługa zdarzenia HTML `ondblclick` elementów klasy `wyraz`:

- **Kod HTML**

```
<p>Lorem <strong class="wyraz">ipsum</strong> dolor sit
<strong class="wyraz">amet</strong>, suspendisse
<strong>nunc</strong>.</p>
```

- **Skrypt jQuery**

```
$(function() {
    $('.wyraz').dblclick(function() {
        alert('Hello, world!');
    });
});
```

Obsługa zdarzeń

- Parametr **`this`** w przypadku obsługi zdarzeń jest **obiektem DOM, dla którego dane zdarzenie zostało wywołane**, np.

```
$(function() {
    $('div').click(function() {
        $(this).css('background', 'lime');
    });
});
```

- Selektor `$('p')` wybiera z dokumentu wszystkie akapity, ale — przez zastosowanie **`$(this)`** — **modyfikuje tylko ten kliknięty**.

Odczyt i modyfikacja wyglądu elementów HTML

Odczyt i modyfikacja wyglądu elementów HTML

- Dostęp do wyglądu elementów HTML zapewnia metoda `.css()`.
- By odczytać wartość własności CSS elementu, metodę tę wywołujemy z **1 parametrem** odpowiadającym nazwie własności CSS, np.
```
var tlo = $('div').css('background-color');
```
- By zmodyfikować wartość własności CSS, metodę tę wywołujemy z **2 parametrami** ustalającymi nową **wartość** dla **własności CSS**, np.
```
$('div').css('background', 'orange');
```
- Wykorzystując **2 wywołania** metody `.css()`, możemy **właściwość z jednego elementu przypisać do drugiego**, np.
```
$('div').css('background', $('a').css('color'));
```

Zadanie 1.

Odczyt i modyfikacja wyglądu elementów HTML

Zadanie 1.
Odczyt i modyfikacja wyglądu elementów HTML

a) Dany jest dokument zawierający kilka kolorowych pojemników. Korzystając z metody `.css()`, wymień kolor tła dokumentu, w zależności od tego, na który kolorowy pojemnik najechano.

   Domyślnie kolor tła dokumentu jest biały.

   Skorzystaj z metody `.css()`, aby odczytać kolory z poszczególnych pojemników.

Odczyt i modyfikacja atrybutów i właściwości elementów HTML

Odczyt i modyfikacja
atrybutów i właściwości elementów HTML

- Dostęp do **atrybutów** (ang. *attribute*) elementów zapewnia metoda `.attr()`, zaś do **właściwości** (ang. *property*) — metoda `.prop()`.

- W celu odczytania wartości atrybutu lub właściwości elementu HTML, metody te należy wywołać z **1 parametrem** odpowiadającym nazwie atrybutu lub właściwości, np.

```
var link = $('a').attr('href');
```

Odczyt i modyfikacja
atrybutów i właściwości elementów HTML

- W celu zmodyfikowania wartości atrybutu lub właściwości elementu HTML, metody te należy wywołać z **2 parametrami** ustalającymi nową **wartość** dla danego **atrybutu** lub danej **właściwości**, np.

```
$('a').attr('disabled', 'disabled');
$('a').prop('disabled', true);
```

- Metoda `.removeAttr()` usuwa co najmniej jeden atrybut z wybranych elementów, np.

```
$('a').removeAttr('title');
$('a').removeAttr('id class');
```

Zadanie 2.

Odczyt i modyfikacja atrybutów elementów

Zadanie 2.
Odczyt i modyfikacja atrybutów elementów

a) Stwórz taką galerię obrazków, w której kliknięcie miniaturowego obrazka wewnątrz komórki tabeli HTML powoduje **wyświetlenie odpowiadającego mu dużego obrazka** wewnątrz elementu `div` położonego poniżej.

   Zadbaj również o to, by dynamicznie wygenerowany duży obrazek posiadał odpowiadający klikniętej miniaturce **tekst alternatywny**, tj. atrybut `alt`.

   Duże obrazy znajdują się w folderze `400x300`, a małe w folderze `100x75`. Nazwy plików w obu folderach są identyczne.

Odczyt i modyfikacja treści elementów HTML

Odczyt i modyfikacja
treści elementów HTML

- Dostęp do treści elementów HTML zapewniają metody `.text()` oraz `.html()`. Pierwsza z nich operuje **tekstem**, a druga **kodem HTML**.

- Metoda `.html()` działa w dokumentach HTML, natomiast **nie działa w przypadku dokumentów XML** (ang. *Extensible Markup Language*).

- W celu odczytania zawartości elementu HTML, metody te należy wywołać **bezparametrowo**, np.
```
var kod = $('p').html();
```

- W celu zmodyfikowania zawartości elementu HTML, metody te należy wywołać z **parametrem** ustalającym nową **zawartość elementu**, np.
```
$('p').text('Hello, world!'); // zwraca Hello, world!
```

Odczyt i modyfikacja
treści elementów HTML

- Działanie metod `.text()` oraz `.html()` jest **identyczne** przy **braku kodu HTML** w przekazywanym do nich parametrze, np.

```
$('p').text('Hello, world!'); // zwraca Hello, world!
$('p').html('Hello, world!'); // zwraca Hello, world!
```

- Jeśli parametr przekazany do obu metod zawiera znaczniki HTML, to **zachowanie metod będzie różne**, np.

```
$('p').html('<strong>Hello world!</strong>');
// zwraca Hello world!
$('p').text('<strong>Hello world!</strong>');
// zwraca &lt;strong&gt;Hello world!&lt;/strong&gt;
```

Dodawanie i usuwanie węzłów drzewa DOM

Dodawanie i usuwanie węzłów drzewa DOM

Drzewo DOM (ang. *Document Object Model*) reprezentuje **zawartość dokumentu**. Jego strukturę możemy analizować m.in. przy użyciu narzędzi deweloperskich przeglądarki internetowej Google Chrome (ang. *Google Chrome Developer Tools*).

Oprócz poznanej już metody `.html()`, do **tworzenia nowych węzłów** drzewa DOM służą metody:

- `.append()` oraz `.prepend()`
- `.apendTo()` oraz `.prependTo()`
- `.after()` oraz `.before()`
- `.insertAfter()` oraz `.insertBefore()`
- `.wrap()`
- `.wrapInner()`

Dodawanie i usuwanie węzłów drzewa DOM

- Metody `.append()` i `.prepend()` dodają określone parametrem **nowe dziecko** odpowiednio jako **ostatnie** oraz jako **pierwsze** do istniejącego węzła drzewa DOM, na rzecz którego metoda jest wywoływana, np.

```
$('div').append('<p>A</p>');
$('div').prepend('<p>B</p>');
```

- Metody `.appendTo()` oraz `.prependTo()` różnią się **kolejnością parametrów**, tzn. jako pierwszy podajemy nowe dziecko, a jako drugi istniejący węzeł drzewa DOM, np.

```
$('<p>C</p>').appendTo('div');
$('<p>D</p>').prependTo('div');
```

Dodawanie i usuwanie węzłów drzewa DOM

- Metody `.after()` oraz `.before()` dodają określone parametrem **nowe rodzeństwo** odpowiednio **za** oraz **przed** istniejącym węzłem drzewa DOM, na rzecz którego metoda jest wywoływana, np.

```
$('div').after('<p>A</p>');
$('div').before('<p>B</p>');
```

- Metody `.insertAfter()` oraz `.insertBefore()` różnią się **kolejnością parametrów**, tzn. jako pierwszy podajemy nowe rodzeństwo, a jako drugi istniejący węzeł drzewa DOM, np.

```
$('<p>C</p>').insertAfter('div');
$('<p>D</p>').insertBefore('div');
```

Dodawanie i usuwanie węzłów drzewa DOM

- Metoda `.wrap()` **otacza** istniejący **węzeł drzewa DOM**, na rzecz którego metoda jest wywoływana, określonym w parametrze **kodem** (otoczenie zewnętrzne), np.
  `$('h1').wrap('<div></div>');`

- Metoda `.innerWrap()` **otacza zawartość** istniejącego **węzła drzewa DOM**, na rzecz którego metoda jest wywoływana, określonym w parametrze **kodem** (otoczenie wewnętrzne), np.
  `$('p').wrapInner('<strong></strong>');`

- Metoda `.innerWrap()` działa podobnie jak metoda `.html()`, ale musi tu być dodany **konkretny węzeł**, a ponadto **nie ma możliwości odczytu**.

Metoda przetwarzająca zbiór elementów

Metoda przetwarzająca
zbiór elementów

Jeśli chcemy iteracyjnie wykonać **operacje** (zestaw instrukcji) na określonym **zbiorze elementów**, a **nie przypisujemy mu obsługi żadnego zdarzenia**, możemy skorzystać z metody `.each()`, np.

- Kod HTML

```
<ul>
    <li>Europa</li>
    <li>Afryka</li>
    <li>Azja</li>
</ul>
```

Metoda przetwarzająca zbiór elementów

- Skrypt jQuery

```
$(function() {
    $('li').each(function() {
        console.log($(this).text());
    });
});
```

Zmienna `this` odnosi się do **aktualnie przetwarzanego elementu zbioru**.

W wyniku otrzymamy **3 logowania do konsoli**, które odpowiadają zawartościom tekstowym **poszczególnych elementów listy wypunktowanej**.

Metoda przetwarzająca
zbiór elementów

- Skrypt jQuery

```
$(function() {
    console.log($('li').text());
});
```

Jeżeli nie skorzystamy z metody `.each()`, w wyniku otrzymamy tylko **1 logowanie do konsoli** w postaci **połączonych** (sklejonych ze sobą) zawartości tekstowych **poszczególnych elementów listy wypunktowanej**.

```
// zwraca EuropaAfrykaAzja
```

Efekty specjalne (animacje)

Efekty specjalne (animacje)

- `.show()` – pojawienie się
- `.hide()` – ukrywanie
- `.toggle()` – naprzemiennie `.show()` oraz `.hide()`

- `.slideDown()` – wysuwanie (pojawienie się)
- `.slideUp()` – wsuwanie (ukrywanie)
- `.slideToggle()` – naprzemiennie `.slideDown()` oraz `.slideUp()`

- `.fadeIn()` – stopniowe pojawienie się
- `.fadeOut()` – zanikanie
- `.fadeToggle()` – naprzemiennie `.fadeIn()` oraz `.fadeOut()`

Efekty specjalne (animacje)

- `.animate()` — płynna modyfikacja dowolnej **wartości numerycznej właściwości CSS**

Wszystkie podane metody, **poza `.animate()`**, możemy wywołać:

- bezparametrowo — domyślny **czas trwania efektu** wynosi 400 ms, ale dla metod `.show()` oraz `.hide()` domyślny czas trwania efektu to 0

- z parametrami, gdzie każdy z nich jest opcjonalny — **czas trwania efektu** (ang. *speed*), **sposób wykonania efektu** (ang. *easing*), **instrukcje** (w postaci funkcji nazwanej lub funkcji anonimowej), jakie mają zostać wykonane **po zakończeniu trwania efektu** (ang. *callback*)

Efekty specjalne (animacje)

- **Czas trwania efektu** (ang. *speed*) określa, **jak długo** dany efekt ma być wykonywany, np.
```
('p').hide('fast'); // 0,2 s
('p').fadeOut('medium') albo ('p').fadeOut(); // 0,4 s
('p').show('slow'); // 0,6 s
('p').slideUp(2000); // 2 s
```

- **Sposób wykonania efektu** (ang. *easing*) to **łańcuch tekstowy** przyjmujący wartość `'swing'` (domyślnie) albo `'linear'` powodującą wykonanie efektu w stałym tempie (jednakowy przyrost efektu), np.
```
('p').slideDown(800, 'linear'); // 0,8 s
```

Efekty specjalne (animacje)

- **Instrukcje po zakończeniu efektu** (ang. *callback*) to **nazwa funkcji** lub **funkcja anonimowa** zawierająca instrukcje, które zostaną wykonane **po zakończeniu trwania efektu**, np.

```
('p').fadeIn(2000, function() {
    // instrukcje do wykonania dopiero po 2 s
});
```

- Metodę `.animate()` musimy wywołać z co najmniej jednym parametrem — **tablicą asocjacyjną numerycznych właściwości CSS**, do których animacja ma płynnie dążyć. Ponadto, opcjonalnie do listy parametrów możemy dodać czas trwania animacji, sposób wykonania oraz funkcję zawierającą instrukcje, które zostaną wykonane po zakończeniu trwania animacji.

Efekty specjalne (animacje)

Przykłady:

- ```
  ('p').animate({opacity: '0.5', height: '50%'});
  // zmiana przezroczystości i wysokości do 50%
  ```

- ```
  ('p').animate({opacity: '0.5', height: '50%'}, 5000);
  // zmiana przezroczystości i wysokości do 50% w ciągu 5 s
  ```

- ```
  ('p').animate({opacity: '0.5', height: '50%'}, 5000,
  'linear', function() {
      // instrukcje do wykonania dopiero po 5 s
  });
  // zmiana przezroczystości i wysokości do 50% w ciągu 5 s,
  liniowo oraz dodatkowe instrukcje po zakończeniu animacji
  ```
  ```

Efekty specjalne (animacje)

- Kolejne wywołania metod animacji (`.slideDown()`, `.fadeIn()`, `.animate()` itd.) **ustawiają się w kolejce**, czyli dopiero po zakończeniu jednej animacji rozpoczyna się następna, np.

```
$('#elipsa').animate({left: '+=400'}, 3000).slideUp(2000);
// 3-sekundowa animacja przesuwająca elipsę, a następnie
jej 2-sekundowe wsuwanie (ukrywanie)
```

- Metoda **`.delay()`** (ang. *delay* — opóźnienie, zwłoka) **odkłada**, na określony pierwszym parametrem czas, **wykonanie funkcji z kolejki**, np.

```
$('prostokat').slideDown(3000).delay(1000).slideUp(2000);
// wysuwanie (pojawienie się) elementu przez 3 s,
opóźnienie 1 s i wsuwanie (ukrywanie) przez 2 s
```

Efekty specjalne (animacje)

- Wywołanie metody z 2 parametrami `.stop(false, false)` — albo po prostu **bez żadnych parametrów** — na elemencie, dla którego wykonuje się **animacja**, spowoduje **zatrzymanie jej wykonywania** (nie przyjmując docelowej postaci). Ponadto, jeżeli w kolejce czekają inne animacje, to pierwsza z nich zostanie uruchomiona w momencie wywołania metody `.stop()`.

- Jeśli wywołamy metodę z parametrem `.stop(true)`, to **kolejka zostanie skasowana (wyczyszczona)**.

- Jeżeli wywołamy metodę z 2 parametrami `.stop(false, true)`, to bieżąca animacja się natychmiast kończy, **przyjmując docelową postać**.

Zadanie 3.

Efekty specjalne (animacje)

Zadanie 3.
Efekty specjalne (animacje)

Dana jest strona z menu 2-poziomowym (podmenu pojawia się pod pozycją drugą).

Przetestuj działanie metod animacji:

a) z predefiniowanym parametrem `'fast'`
b) z predefiniowanym parametrem `'medium'`
c) z predefiniowanym parametrem `'slow'`
d) bez żadnych parametrów

Zadanie 4.

Efekty specjalne (animacje)

Zadanie 4.
Efekty specjalne (animacje)

a) Wykorzystaj metodę `.animate()` oraz przyciski „W lewo" i „W prawo", żeby otrzymać prostokąt przesuwający się w poziomie o `400px` po naciśnięciu odpowiedniego przycisku.

Zadanie 5.

Efekty specjalne (animacje)

Zadanie 5.
Efekty specjalne (animacje)

a) Dana jest strona z pustym węzłem `body` i regułą CSS dla klasy `hello`. Napisz skrypt jQuery, który utworzy nowy węzeł `div` z tekstem „Hello, world!", a następnie doda go do węzła `body` (tj. dołączy go jako pierwsze lub ostatnie dziecko), wyposaży w klasę CSS `hello` oraz uzupełni o prostą animację, np. stopniowe pojawienie się.

b) Wykonaj w jQuery przewijający się w nieskończoność pasek z logotypami. Szerokość pliku .jpg z logotypami to `1744px`. Pasek powinien zatrzymywać się po najechaniu na niego myszką, zaś po zjechaniu wskaźnikiem myszy poza jego obszar, pasek powinien ruszyć dalej od momentu, w którym się zatrzymał.

Dzięki!

Masz jakieś pytania?

- lukasz.krawczyk.lublin@gmail.com
- goldenline.pl/lukasz-krawczyk81/
- pl.linkedin.com/in/lukasz-krawczyk