jQuery

biblioteka JavaScript m.in. dla modyfikacji DOM

Filozofia

Zrób coś łatwo i szybko

(Podobno) Ponad 50% najpopularniejszych stron korzysta z jQuery. Szacuje się, że ponad ¼ wszystkich stron w internecie używa bibliotek jquery.

Biblioteka powstała w 2006 r., ale ciągle jest rozwijana

https://trends.builtwith.com/javascript/jQuery

Najczęściej używana przed webdevów do:

Wybór elementów

Edycja DOM

Zdarzenia na elementach (działania wywołane zachowanie użytkownika strony)

Animacje/Efekty

AJAX - asynchroniczny JS (i XML) - kontakt z serwerem bez odświeżania strony.

Czym jest jQuery

U podstaw to JavaScript. Biblioteka jQuery to funkcje i obiekty napisany w JavaScript.

Celem jQuery jest umożliwić napisanie rozwiązania **szybciej i łatwiej** niż w czystym JavaScript.

Coś co można zrobić w jQuery można też napisać samemu w JavaScript (i ktoś to w JavaScript napisał).

Największa wada

Nie uczysz się czystego JavaScript :)

Nie jest już trendy i raczej nie ma świetlanej przyszłości.

Zalety (poza pisz mniej, rób więcej)

Zapewnia kompatybilność z przeglądarkami (nawet starszymi)

Przyjazna, bo relatywnie prosta w nauce i użyciu.

Mało kilobajtów zajmuje.

Ciągle rozwijana i powszechnie używana (więc jej znajomość jest plusem dla web developera, ale bez przesady).

Wersje

1, 2, 3 - oczywiście 3

minifikowana - bez niepotrzebnych znaków (spacje, entery)

Co zrobić by zacząć używać jQuery?

Dodać plik jQuery (bibliotekę jQuery) do dokumentu html

```
<script src="jquery.js"></script>
<script>
        ...piszemy kod
<script>
```

Można pobrać albo użyć CDN-a

Co zrobić by zacząć używać jQuery?

Dodać plik jQuery (bibliotekę jQuery) do dokumentu html

```
<script src="jquery.js”></script>  //biblioteka
<script>
 //...piszemy nasz kod w js/jq
<script>
```

Ważne by biblioteka była zaimportowana zanim użyjemy jQuery (może być w <head> albo tuż przed pisaniem naszego kodu) Inaczej kod jQuery nie zostanie rozpoznany i wystąpi błąd.

Co zrobić by zacząć używać jQuery?

Dodać plik jQuery (bibliotekę jQuery) do dokumentu html

```
<script src="jquery.js”></script>
<script src="code.js"><script>
```

Oczywiście możemy i powinniśmy pisać nasz kod javascript w zewnętrznym pliku.

Czy można mieszać jQuery z czystym JavaScript

???

Czy można mieszać jQuery z czystym JavaScript

OCZYWIŚCIE, NA TYM TO POLEGA. JQUERY "ROZSZERZA" CZYSTY JAVASCRIPT, A NIE GO ELIMINUJE Z NASZEJ PRACY

<script src="jquery.js></script>

<script>

...piszemy kod JAVASCRIPT z wykorzystaniem biblioteki jQuery

<script>

Jak zazwyczaj wykorzystuje się jQuery

1. pobieramy element(y) np. $("p.article span")
2. wykonujemy operację (metodę) np. .addClass("green")

```
$("p.article span").addClass("green")
```

Do używania jQuery wystarczy znajomość podstaw css i JavaScript, dlatego ta biblioteka była bardzo popularna i nadal jest używana.

Pobranie elementu

```
$("p:nth-child(2)")
jQuery("div.red + p")
```

Wynikiem pobrania jest **obiekt jQuery**.

Ten obiekt daje nam wiele możliwości w postaci użycia nowych właściwości i metod, których nie mamy w czystym JavaScript.

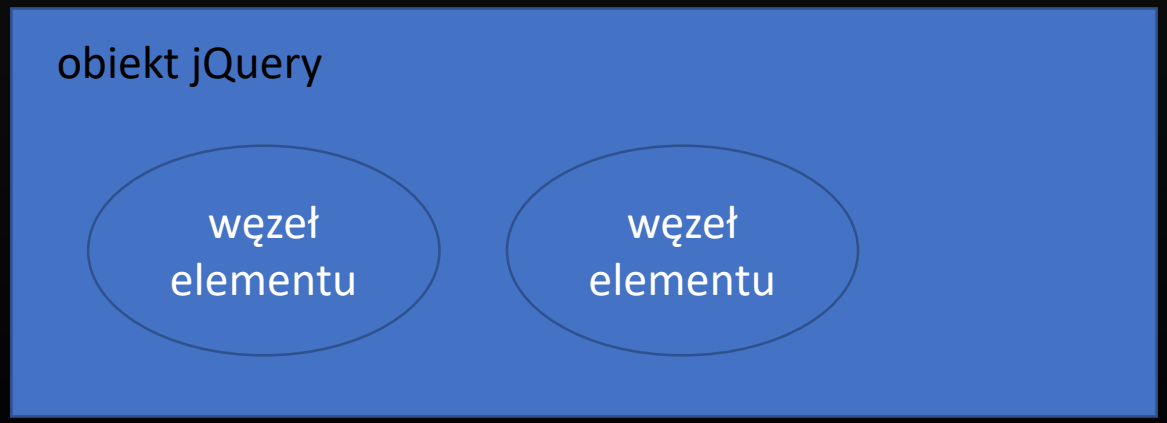
Obiekt jQuery

```
$('p')
▶ init(24) [p.info, p.info, p.info, p, p, p, p, p, p, p,
  selector: "p"]
```

$("p:nth-child(2)")

```
$('p.red')
▶ init [prevObject: init(1), context: document, selector: "p.red"]
```

Taki obiekt zawiera listę węzłów elementów (elementów html) które pasują do zapytania (selektora). Obiekt może więc zawierać listę, która ma jeden element, wiele elementów lub żadnego.



Przypisanie obiektu jQuery do zmiennej

```
$("p:nth-child(2)");
```

Oczywiście często warto przypisać taki obiekt do zmiennej. Szczególnie warto wtedy jeśli będziemy na nim pracować wielokrotnie.

```
var secondP = $("p:nth-child(2)");
```

$zmienna

Konwencja jest taka, że zmienne, które przechowują obiekty jQuery (pamiętaj, że element(y) pobrane z DOM można traktować jako owiniętr przez obiekt jQuery) powinny zaczynać się od znaku dolara. Dlaczego? Dla odróżnienia, bo mają inne metody.

```
var $secondP = $("p:nth-child(2)")
```

Ale to tylko konwencja (której warto przestrzegać)

$

Co można zrobić z obiektem jQuery

$("selektor").metoda()

można wywołać metodę, która zwróci nam jakąś informację lub wykonana jakieś zadanie na obiekcie.

Oczywiście metodę można wywołać bezpośrednio na zapytaniu (jak wyżej) albo na zmiennej przechowującej obiekt (jak niżej)

$paragraph.metoda()

Obiekt jQuery

Obiekt jQuery ma mnóstwo metod i właściwości, które przyśpieszą i usprawnią naszą prace ze stroną.

Reasumując: obiekt zawiera kolekcję węzłów które zostały pobrane. **Każdy obiekt jQuery jest "wyposażony" w metody i właściwości.**

Obiekt jQuery

Te metody i właściwości JQ można wykonać **tylko na obiekcie jQuery** tzn. nie można ich wykonać na elementach węzłów i listach węzłów pobranych w za pomocą metod czystego JavaScript.

`$("p").addClass("black")` - TAK (dla każdego elementu)

`document.querySelector("p").addClass("black")` - NIE

`document.querySelectorAll("p").addClass("black")` - NIE

Obiekt jQuery

W drugą stronę też to nie działa

$("p").classList.add("black") - NIE bo to nie działa na obiektach JQ

document.querySelector("p").classList.add("black") - TAK

document.querySelectorAll("p").classList.add("black") – NIE, bo to metoda obiektu DOM a nie listy węzłów (nodeList).

$("p"). addClass("black") - TAK, wszystkie "p„ otrzymają klasę „black”

Obiekt jQuery

Zbiór wyników (czyli ten nasz obiekt jQuery) zawiera elementy (węzły) w tej samej kolejności w której węzły te znajdują sie w DOM.

```
<p class="red"><strong>aaa</strong><span>bbb</span><p>
```

$("*") -> [p.red, strong, span]

Metody jQuery

$(“p span.green”).removeClass(“green”)

Jak w czystym JS metody (oczywiście) mogą mieć swoje argumenty (które podajemy przy wywołaniu). Często metoda do prawidłowego działania wymaga podania argumentów. Wiele metod inaczej zachowuje się z podanymi argumentami i bez podanych argumentów.

Metody jQuery

```
$(“.green”).removeClass(“green");
```

Co stało się w tej instrukcji?

1. ?
2. ?

Metody jQuery

$(“.green”).removeClass(“green”)

Co tu się stało?

1. pobranie pasujących elementów i umieszczenie ich w obiekcie jQuery
2. Usunięcie z KAŻDEGO elementu klasy (czyli ze wszystkich elementów znajdujących się w obiekcie JQ, pobranych za pomocą funkcji jQuery w pierwszej instrukcji).

W czystym JS piszemy tutaj najczęściej pętle for lub korzystamy z metody forEach - czyli czegoś co wymaga już poznania całkiem nieźle JavaScript. W każdym razie nie jest to tak proste jak w jQuery.

Metody jQuery

$(“p span.green”).removeClass(“green”)

Większość metod JQ uaktualnia wszystkie elementy kolekcji (czyli elementy dokumentu HTML).

W JS by wszystkie uaktualnić trzeba użyć pętli czy odpowiedniej metody, więc znacznie łatwiej zrobić to w JQ.

Łańcuch metody jQuery (łańcuch instrukcji)

$(“p span.green”).removeClass(“green"). addClass("blue")

W JQ bardzo popularne jest tworzenie łańcucha metod (instrukcji). Najpierw mamy obiekt jQuery na którym kolejno (od lewej) wykonujemy metody.

W przykładzie

- najpierw usunęliśmy klasę "green"
- potem dodaliśmy klasę "blue"

Łancuch Metody jQuery

```
$(“p span.green”)
.removeClass(“green”)
.addClass("blue")
```

Można to też zapisać tak. Dla niektórych bardziej czytelne.

Selektor w zapytaniu funkcji jQuery

Jak CSS najnowszy a nawet jeszcze bardziej rozbudowane. Wybór elementu był jedną z najważniejszych zalet jQuery względem czystego JS, choć dziś już to się trochę zmieniło za sprawą metody querySelector()

$("div * span ~ a")

document ready? – często spotkasz, ale czy używać?

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

});
```

```
//Mechanika jak nasłuchiwanie poniżej
//$('button').click(function(){})
```

document ready? – używać?

$(document).ready(function(){})

Kiedyś powszechnie używana metoda dziś już niekoniecznie. Konieczne tylko jeśli nie umieszczamy kodu tuż przed znacznikiem zamykającym body.

Tak naprawdę jest to event (zdarzenie). "ready" jest bowiem zdarzeniem, które mówi „DOM został stworzony”. "ready" oczekuje na informacją od przeglądarki, ze cały DOM jest już przygotowany. A większość naszej pracy odbywa się na DOM, więc JQ powinniśmy wykonywać gdy jest DOM wczytany.

document ready? DOM

```
<body>
... kod ...
        //tuż przed znacznikiem zamykającym body cały DOM już
        jest stworzony, więc nie ma co robić „ready”, bo jest ready ☺
        <script></script>
</body>
```

document ready - zdarzenie jak click

```
$(document).ready(function() {
...cały kod
})
```

Metoda ready jest nasłuchiwanie informacji od przeglądarki, ze cały DOM jest już przygotowany (przeglądarka taką informację daje a ready na nią czeka).

Kiedy zdarzenie wystąpi (a wystąpi raz) wywyołuje funkcję w środku.

skrót document ready

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

Częściej używa się skrótu, który oznacza dokładnie to samo.

```
$(function() {
..cały kod
});
```

Dlaczego warto $(function() {...});

1. (podstawowa) Bo mamy pewność, że DOM już jest i możemy użyć jQuery nawet w <head> (choć akurat to zła praktyka)

2 . (zaawansowana) Bo tworzymy nową przestrzeń nazw (poza zakresem globalnym). W funkcji anonimowej w tym przypadku - i to akurat najczęściej pożądane zachowanie. Przydatne przy bardziej zaawansowanych projektach.

Pobranie elementów – najważniejsza funkcja jQuery

```
$("p")
jQuery('p')
```

Pobiera wszystkie elementy i opakowuje je w obiekt jQuery (jeśli nie ma pasujących elementów, to obiekt jest pusty)

$("selektor").length

Właściwość length pozwala nam stwierdzić ile elementów zostało znalezionych i umieszczonych w obiekcie jQuery.

$("p span").length

var spn = $("p span").length

Zwróć uwagę, że jest to właściwość (nie ma wywołania jak metoda, czyli nie jest length())

$("selektor").length

```
$('div').length
247
```

Zadanie dla Ciebie

Wejdź na stronę onet.pl (tylko na chwilę, nie czytaj teraz informacji ;)), otwórz konsolę w przeglądarce i wpisz instrukcje:

```
$('div');
$('div').length;
$('*').length;
```

Zobaczysz jak wygląda obiekt jQuery i ile elementów zawiera. Ps. Te funkcje (za)działają, gdy strony korzystają z biblioteki JQ!

Metoda .html()

$("p").html()

Pobiera informacje o zawartości elementów html.

Pobieranie informacji odnosi się do pierwszego elementu.

Sprawdź: $('p').html()

Metoda .html(argument)

$("div").html(" pusto ")

Zamienia zawartość wszystkich elementów w obiekcie jQuery.

Zadanie

- wpisz na stronie $("div").length //ile elementów masz?
- wykonaj metodę $("div").html(" brak ");
- Zobacz jak wygląda strona ☺

.html() - pobiera
.html(" aktualizacja ") - zmiana

Metoda .text()

Pobiera zawartość tekstową, ale nie pierwszego elementu (jak w metodzie .html), ale wszystkich!

aaaa

bbbddd ccc

$("p").text() -> "aaaabbbddd ccc"

Łączy bez spacji zawartość tekstową elementów, które zawiera bez pokazywania html-a (a więc znaczników).

Metoda .text(argument)

Zastąpienie tekstem WSZYSTKICH elementów.

$("p").text("nowy tekst")

Zadanie

- wpisz na stronie $("body").text() - zobacz efekt
- wpisz na stronie $("body").text("cała strona") - zobacz efekt również w zakładce Elements w narzędziu programistycznym.

Metoda .remove()

Usunięcie wszystkich pasujących elementów.

$("div").remove()

Praca z elementami - tworzenie elementu

```
var $div = $("<div>");
// w funkcj jQuery zamiast "div" umieszczamy "<div>", a więc znacznik.

var $divBlue = $("<div class='blue'>Coś</div>");
//można też stworzyć bardziej rozbudowany obiekt, np. z tekstem i atrybutami.
```

Praca z elementami - dodawanie elementu

```
var $div = $("<div>");
var $divBlue = $("<div class='blue'>Coś</div>");


$("body").before($div)  //1. Przed czym (elementem body) 2. Co mam wstawić (element ze zmiennej $div)
$("body").after($divBlue) //za elementem body wstaw element znajdujący się w zmiennej $divBlue.

//uwaga element body jest jeden, ale gdyby wybranych elementów byłoby więcej to przed/po każdym byłby wstawiony nasz element.
```

Praca z elementami - dodawanie elementu

```
var $div = $("<div>");
var $divBlue = $("<div class='blue'>Coś</div>");


$("section").before($div)  //1. Przed czym (elementem section) 2. Co mam wstawić (element ze zmiennej $div)
$("h1.title").after($divBlue) //za każdym elementem h1 z klasą title
```

Praca z elementami - dodawanie elementu

```
$("p").after("<div>Nowość</div>")
```

Po **każdym** p dodaj to co jest w stringu. Można i tak.

Prepend - dodaj na początku

```
var $div = $("<div>");


$("body").prepend($div) // dodaj wewnątrz na początku wybranych elementów (u nas jeden element body) element określony w prepend.
```

PrependTo - dodawanie do - na początku

```
var $div = $("<div>");


$div.prependTo("body") // dodaj wybrany element na początku
do środka elementu (jego początku) wskazanego jako argument
w metodzie prependTo()
```

Append i AppendTo - dodaj na końcu

```
$("div").append('<p>dodawany element</p>');
```

dodaj...

odpowiednik element.apendChild()

```
$('<p>dodawany element</p>').appendTo("div");
```

dodaj do...

Atrybuty - metoda attr()

$("li, p").attr("id") - pobranie atrybutu id PIERWSZEGO pasującego elementu.

$("div").attr("id", "super") - przypisanie atrybutu do wszystkich pasujących elementów. //swoją drogą to nie przypujmy tak id ;)

$("div").attr("class", "active");

Uwaga metoda ta zastępuje dany atrybut a nie dopisuje.

$("div").attr("class", "active red");
W istocie metoda ta podmienia po prostu stringa, który jest, na nowy.

Atrybuty - metoda attr()

$("*").removeAttr("class") - usunięcie wszystkich atrybutów danego typu ze wszystkich pasujących elementów.

$("*").removeAttr("data-name") - gwiazdka oznacza każdy element.

Zobaczmy w kodzie...