Czemu nie potrzebujesz jQuery?

Spis treści

Mimo że zdaje ci się inaczej

Ten tutorial kieruję do wszystkich tych, którzy nie wyobrażają sobie wykonania jakiejkolwiek prostej czynności w JS bez użycia tej wszechstronnej biblioteki. Część zarzutów tyczy się też mnie, gdyż osobiście również korzystam z jQuery (hej, to nie moja wina, że DOM jest tak spaprany!)

Alternatywa?

VanillaJS (http://vanilla-js.com/) + jeden z tych maluczkich (http://microjs.com/)

Podstawowe operacje DOM-owe

Wyszukiwanie po selektorach

Potęgą jQuery jest jego silniczek Sizzle, który pozwala wyszukać dowolny element tylko i wyłącznie na podstawie selektora. To jest często główny/jedyny powód, dla którego część webmasterów sięga po jQ.

```
$( '#bardzo > optymalny .selektor #ktory .na-bank.dziala' );
```

> Oczywiście optymalizacja takich selektorów to zupełnie inna sprawa.

Skupmy się na tym, czy w czystym JS istnieje jakaś alternatywa dla tego pięknego owijacza? Otóż tak!

```
document.querySelectorAll( '#bardzo > optymalny .selektor #ktory .n
a-bank.dziala' );
```

Prawdę mówiąc, jeśli przyjrzymy się kodowi jQ, zauważymy, że wewnętrznie używa on właśnie tej metody. Nie trzeba chyba mówić, co jest szybsze. Oczywiście można pójść o krok dalej i zrobić sobie coś takiego:

```
var $ = document.querySelectorAll.bind( document );
```

Voila! Mamy jQ bez jQ.

Chainowanie metod

Chainowanie metod to to, co decyduje o niezwykłej użyteczności jQ w zastosowaniach DOM-owych.

```
$( '<a>' ).attr( 'href', '#' ).on( 'click', function() {} );
```

Owszem, w VanillaJS nie jestem w stanie uzyskać tak wygodnej obsługi DOM. Ale tu z pomocą przychodzi nam Chainvas (http://leaverou.github.com/chainvas/) – małe cudeńko, które jest w stanie złańcuchować *de facto* wszystko. Dzięki temu można spokojnie napisać:

```
document.createElement( 'a' ).setAttribute( 'href', '#' ).addEventL
istener( 'click', function() {} );
```

I tym sposobem odebraliśmy jQ największą jego przewagę w operacjach DOM-owych, oszczędzając przy tym 31KB kodu (oczywiście mowa tu o wersjach gzipniętych i zminifikowanych).

Event delegation

w jQ jest prosto:

```
$( document ).on( 'click', '.clicky', function( evt ) {
    //awesome stuff
} );
```

w czystym JS też jest prosto

```
document.addEventListener( 'click', function( evt ) {
    if ( !evt.target || !e.target.matches( '.clicky' ) /* Można zmi
enić na !evt.target.classList.contains( 'clicky' )*/ ) {
        return;
    }
    //awesome stuff
}, false );
```

`.clone`

Żeby sklonować obiekt w jQ, wystarczy skorzystać z metody `.clone`. Czy da się zrobić coś takiego w czystym JS? A i owszem.

```
function clone( obj ) {
    return obj.cloneNode ? obj.cloneNode( obj ) : JSON.parse( JSON.
stringify( obj ) );
}
console.log( clone( document.createElement( 'a' ) ) );
```

Prosta i przyjemna sztuczka.

`.extend`

To już nie jest do końca DOM, ale może się tu nadać.

Jak wiadomo, w jQ mamy metodę `.extend` rozszerzającą nam obiekt:

```
console.log( $.extend( {}, { cos: 1 } ) );
```

W czystym JS też się da (https://github.com/shimondoodkin/nodejs-clone-extend/blob/master/index.js#L40). Ba, można być nawet chamskim i po prostu wyciągnąć ten kod z jQ.

Animacje

Co tu dużo mówić – od kiedy w CSS są `transition` i `animation`, większość rzeczy można w ogóle wyjąć poza JS. Natomiast niektóre inne animacje z jQ można równie łatwo napisać w czystym JS. Przykład, lekko poprawiony, ze strony VanillaJS:

```
$( '#thing' ).fadeOut();
//vs
var s = document.getElementById( 'd' ).style;
s.opacity = +( s.opacity || window.getComputedStyle( el )[ 'opacit
y' ] );
( function f() { ( s.opacity -= 1 / 100 ) <= 0 ? s.display = "none"
: requestAnimationFrame( f ) }() );
```

Nie trzeba chyba zaznaczać, że porównanie objętościowe tych trzech linijek i całego jQ wypada korzystnie dla czystego JS.

oczywiście warto zauważyć, że rAF jest tylko w nowszych browserach i trza by zarzucić jakimś fallbackiem:

```
window.requestAnimationFrame = window.requestAnimationFrame || wind
ow.webkitRequestAnimationFrame || window.mozRequestAnimationFrame |
| window.msRequestAnimationFrame || window.oRequestAnimationFrame |
| function( f ){ setTimeout( f, 1000 / 60 ); };
```

XHR

Tu nie będzie przykładów, jedynie krótkie info.

Otóż wrapper w jQ jest cholernie wygodny i w ogóle. Jednak jeśli chcemy mieć dostęp do bardziej zaawansowanych własności obiektu xhr (np. `xhr.upload.progress`), trzeba użyć natywnego obiektu (https://github.com/malsup/form/blob/master/jquery.form.js#L258).

jQUI

Wiele rzeczy da się napisać (http://pulpit.luke.co.pl/) bez tego, wystarcza trochę pomyślunku.

Szybki przykład na resize elementów (akurat wykorzystywany w zupełnie innym kontekście niż browserowy):

```
if ( [ 'l', 'r' ].indexOf( resizing ) !== -1 ) {
    if ( resizing === 'l' ) {
        window.frame.width += x - e.pageX;
        window.frame.left -= x - e.pageX;
    } else {
        window.frame.width -= x - e.pageX;
    }

    if ( window.frame.width < 60 ) {
        window.frame.width = 60;
    }

    x = e.pageX;
    y = e.pageY;
} else if ( [ 't', 'b' ].indexOf( resizing ) !== -1 ) {
    if ( resizing === 't' ) {
        window.frame.height += y - e.pageY;
        window.frame.top -= y - e.pageY;
    } else {
        window.frame.height -= y - e.pageY;
    }

    if ( window.frame.height < 120 ) {
        window.frame.height = 120;
    }

    x = e.pageX;
    y = e.pageY;
} else {
    window.frame.width -= x-e.pageX;
    window.frame.height -= y-e.pageY;
}
```

co prawda kod najwyższych lotów nie jest i raczej przedstawia ogólną ideę jak to może wyglądać, niż jak to powinno wyglądać. Ba, można zdać się na natywne funkcje, takie jak `resize` w CSS (miało działać na wszystkim, ale nie wiem czy ta obietnica jest spełniona) i drag&drop (w IE od wersji 5 bodaj!). Dobitnie to pokazuje, że nawet tak skomplikowane czynności jakie wykonuje za nas jQUI można zrobić zarówno bez niego, jak i jego ojca – jQ.

Ale za to w jQ da się…

Z racji tego, że jQ jest tylko subsetem JS, śmiem twierdzić, że nie ma rzeczy, której nie dałoby się osiągnąć w czystym JS. Jestem w stanie podjąć każde wyzwanie i udowodnić, że się da bez!

Czy porzucić?

Nie trzeba. Sam używam i jestem zadowolony. Cały problem polega na tym, że jQuery jest przedstawiane jako lekarstwo na wszelkie zło JS, a to po prostu wierutna bzdura. jQuery powstało jako helper DOM-owy i to wciąż jest jego główną siłą.

Co więcej, jQuery niszczy świadomość programistów sieciowych. Coraz częściej można spotkać bowiem nie programistów JS, a programistów jQuery, którzy jedynie tą biblioteką są w stanie się posługiwać.

Dlatego nie mówię, żeby przestać jQ używać. Jeśli natomiast do stronki, gdzie trza oskryptować jeden link, wsadzasz jQ, to znak, że coś jest nie tego. Jeśli natomiast umiesz zastąpić jQ własnym rozwiązaniem (http://www.forumweb.pl/javascript/cancelbubble-stoppropagation-problem/397808#397808) i robisz to, gdy potrzebujesz jedną, konkretną rzecz, to oznacza, że jesteś zupełnie normalnym użytkownikiem tej biblioteki i syndrom uzależnienia Ciebie nie dotyczy.