Zobacz ten artykuł na stronie autora: tworzenie własnych wtyczek jQuery .

jQuery to obecnie najpopulariejszy i najszybciej rozwijający się framework javascript. O jego wielkim sukcesie decyduje prostota i efektywność (programisty, niekoniecznie szybkość skryptów).

Bardzo ciekawą opcją w jQuery jest możliwość tworzenia własnych wtyczek, rozszerzających standardowe możliwości biblioteki.

Co powinieneś umieć / mieć:

- jak używać jQuery
- 5 minut wolnego czasu

Tworzenie wtyczki

Funkcjonalność wtyczki będzie prosta. Rozwinę artykuł " Opisy w chmurce ".

Kod tworzący chmurkę nad węzłem zadawanym jako parametr konstruktora wygląda tak:

{codecitation class='brush: js'}$(document).ready(function() {

$('#a').mouseover(function() {

```
        $('').html('opis').appendTo($(this));

    }).mouseout(function() {

        $(this).find('span.bubble').remove();

    });

});{/codecitation}
```

Chcemy osiągnąć coś takiego:

```
{codecitation class='brush: js'}$(document).ready(function()  {

    $('#a').bubbleSpeech({'description' : 'opis'});

});{/codecitation}
```

Prawda, że ładniej :). No to do dzieła!

Struktura pluginu jQuery

Oto struktura wtyczki do jQuery wg dokumentacji . Każda wtyczka posiada taki szkielet.

{codecitation class='brush: js'}// uniknijmy konfliktow nazw przez wykorzystanie anonimowych funkcji

```
(function($){

    $.fn.extend({

        // w miejsce 'pluginname' wpisujemy nazwe wtyczki

        pluginname: function(options) {

            var defaults = {

                k1 : v1, // domyślne wartości

                k2 : v2 // które można nadpisać przekazując opdowiednie dane w konstruktorze
```

```
            }

            options =  $.extend(defaults, options); // nadpiszą się tu



            // dla każdego węzła spełaniającego warunki

            return this.each(function() {

                // wykonaj ten kod

            });

        }

    });

})(jQuery); // przekaz do funkcji referencje na framework{/codecitation}
```

Jeśli zastanawiasz się, co oznacza anonimowa funkcja to warto przeczytać artykuł o zasięgu zmiennych w JS

.

W oparciu o powyższy kod można stworzyć dowolnie wiele wtyczek.

Krok 1 - "hello world"

Na początek w ogóle uruchommy ten kod, z prostym alertem

{codecitation class='brush: html'}

(function($){

```
$.fn.extend({

    bubbleSpeech: function() {

        return this.each(function() {

            // dla każdego węzła dodajemy obsługę zdarzenia 'click'

            $(this).click(function() {

                alert('HelloWorld');

            });

        });

    }

});

})(jQuery);
```

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

    $('#a').bubbleSpeech();

    $('.b').bubbleSpeech();

});
```

```
    #a { position: relative;   }

    #a span.bubble  {  position: absolute;  top: 20px;  left: 0;

        width: 180px; padding: 10px; line-height: 30px;

        border: 1px solid #000; background: #fff;

    }
```

To jest akapit z id a

To jest akapit z klasą b

To jest pojemnik z klasą b

{/codecitation}

Powyższy kod pozwala już na używanie naszej własnej wtyczki! Jak widzisz, może kodu jest trochę więcej niż w pierwszy mwypadku, ale za to o ile łatwiej i przyjemniej się tego teraz

używa! Choć - trzeba przyznać - efekt póki co nie jest powalający. Same alerty :/

Krok 2 - przekazanie parametrów

Teraz przekażmy parametry do naszej wtyczki

{codecitation class='brush: js'}(function($) {

```
$.fn.extend({

    bubbleSpeech: function(options) {

        var defaults = {

            description : 'Hello World' // tu mamy domyślne wartości [3]

        };



        options = $.extend(defaults, options); // nadpiszmy domyślne wartości [4]
```

```
        return this.each(function() {

            // dla każdego węzła dodajemy obsługę zdarzenia 'click'

            $(this).click(function() {

                alert(options.description); // wyświetlmy wynik [5]

            });

        });

    }

});

})(jQuery);

$(document).ready(function() {

    $('#a').bubbleSpeech({description : 'To jest mój opis 1'}); // to dajemy obiekt z naszymi wartościami [1]
```

```
    $('.b').bubbleSpeech(); // tu bez żadnych danych przekazanych [2]

});

// reszta kodu HTML pozostaje bez zmian{/codecitation}
```

Jak widać, teraz gdy klikamy opisy są inne. Dzieje się tak dzięki sparametryzowaniu: [1], [2]. Możemy oczywiście zmienić nazwę klucza w przekazywamyn JSON-ie . Należałoby wtedy także dokonać odpowiednich zmian w [3], [5]. Czas, aby nasza wtyczka zaczęła robić, to czego oczekujemy - wyświetlać "chmurkę", a nie alertować.

Krok 3 - "chmurka" z opisem

Tym razem zmiany będą dotyczyć tylko samej wtyczki:

```
{codecitation class='brush: js'}(function($) {

    $.fn.extend({

        bubbleSpeech: function(options) {
```

```
 var defaults = {

    description : 'Hello World'

 };

options = $.extend(defaults, options);

return this.each(function() {

    var self = $(this); // [1]

    self.mouseover(function() {

        var zIndex = self.css('z-index'); // [2]

        $('').

            attr({'zindex' : zIndex}). // [3]

            html(options.description).

            appendTo(self);
```

```
                self.css({'z-index' : 10000000000}); // [4]

            }).mouseout(function() {

                var bubble = self.find('span.bubble'); // [5]

                var zIndex = bubble.attr('zindex'); // [6]

                self.css({'z-index' : zIndex}). // [7]

                    find('span.bubble').remove(); // [8]

            });

        });

    }

});

})(jQuery);{/codecitation}
```

W powyższym kodzie po kolei robimy:

1. Tworzymy obiekt dla aktualnego węzła DOM spełniającego warunki podane w konstrukltorze [np. id = "a" => $('#a')].
2. Zapamiętujemy aktualną wartość z-index...
3. ... i zapisujemy ją na przyszłośc w specjalnym - niezgodnym ze standardami atrybucie

4. Ustawiamy bardzo duży z-index (aby nasza chmurka przykryła wszystko inne).
5. Chwytamy naszą chmurkę.
6. Odczytujemy wcześniej [2] zapisany z-index.
7. Ponownie przypisujemy naszemu węzło wi jego pierwotny z-index [aby niczego nie popsuć]
8. Usuwamy chmurkę.

Tym sposobem zyskujemy już gotową wtyczkę.

Co można poprawić?

Zawsze jest coś do poprawienia. Tu chociażby warto dodać nowy parametr - nazwę klasy dla chmurki. Póki co jest zahardkodowane "bubble". Istnieje jednak możliwość, że ktoś zechce tak nazwać klasę na swojeje stronie. Warto więc może dać tym sterować.