Bogate Interfejsy Użytkownika
jQuery plugin

Łukasz Rybka · Gdańsk 2015

Co to jest jQuery?

Co to jest jQuery?

szybka i mała biblioteka JavaScript (*)

Co to jest jQuery?

szybka i mała biblioteka JavaScript (*)
ułatwia przeglądanie i manipulację drzewa DOM

Co to jest jQuery?

szybka i mała biblioteka JavaScript (*)
ułatwia przeglądanie i manipulację drzewa DOM
upraszcza obsługę zdarzeń, animację oraz operacje AJAX'owe

Co to jest jQuery?

szybka i mała biblioteka JavaScript (*)
ułatwia przeglądanie i manipulację drzewa DOM
upraszcza obsługę zdarzeń, animację oraz operacje AJAX'owe
wsparcie (i jednakowe API) dla najpopularniejszych przeglądarek

jQuery Foundation Projects

jQuery Foundation Projects

jQuery

jQuery Foundation Projects

jQuery

jQuery UI (User Interface)

jQuery Foundation Projects

jQuery

jQuery UI (User Interface)

jQuery Mobile

jQuery Foundation Projects

jQuery

jQuery UI (User Interface)

jQuery Mobile

QUnit (JS unit testing)

jQuery Foundation Projects

- jQuery
- jQuery UI (User Interface)
- jQuery Mobile
- QUnit (JS unit testing)
- Sizzle (CSS selector engine)

jQuery fundamentals - uruchamianie

```
<script src="//code.jquery.com/jquery-2.1.4.min.js"></script>
<script type="text/javascript">
    window.onload = function() {
        // Some code
    };

    $(window).load(function() {
        // Some code
    });

    $(ready).load(function() {
        // Some code
    });
</script>
```

CDN

CDN

Content Delivery Network (Content Distribution Network)

CDN

Content Delivery Network (Content Distribution Network)

Rozproszony geograficznie system dostarczania treści cyfrowych

CDN

Content Delivery Network (Content Distribution Network)

Rozproszony geograficznie system dostarczania treści cyfrowych

Wysoka wydajność i szybkość

CDN

Content Delivery Network (Content Distribution Network)
Rozproszony geograficznie system dostarczania treści cyfrowych
Wysoka wydajność i szybkość
Serwery CDN często znajdują się w ISP data center

jQuery fundamentals - DOM traversal

```
<script type="text/javascript">
    $(document).ready(function () {
        $('#someID').html('Hello!');

        $('p').css('color', 'red');

        $('.description').css('font-weight', 'bold');

        $('[data-parent-chapter="someChapter"]').remove();

        $('#someID').find('.someClass').text('Some text');
    });
</script>
```

jQuery fundamentals - DOM traversal

Im bardziej specyficzny selektor tym lepiej ("div.data .gonzalez" będzie wolniejsze niż ".data td.gonzalez")

Im bardziej specyficzny selektor tym lepiej ("div.data .gonzalez" będzie wolniejsze niż ".data td.gonzalez")

Pośrednia dokładność jest zbędna - zbyt rozbudowany selector jest wolny

Im bardziej specyficzny selektor tym lepiej ("div.data .gonzalez" będzie wolniejsze niż ".data td.gonzalez")

Pośrednia dokładność jest zbędna - zbyt rozbudowany selector jest wolny

Unikamy uniwersalnych selektorów: np. "*" oraz ":<type>"

jQuery fundamentals - DOM manipulation

```
<script type="text/javascript">
    $(document).ready(function () {
        var $element = $('#someElementID');

        $element.css({
            'border': '1px solid red',
            'text-align': 'center'
        });
        $element.text('Some very important text')
        $element.show();
    });
</script>
```

jQuery fundamentals - events

```
<script type="text/javascript">
    $(document).ready(function () {
        $( "p" ).click(function() {
            console.log( "You clicked a paragraph!" );
        });
    });

    $( "button.alert" ).on( "click", function() {
        console.log( "A button with the alert class was clicked!" );
    });

    $( "<button class='alert'>Alert!</button>" ).
        appendTo( document.body );

    $( "input" ).on("click change",
        function() {
            console.log( "An input was clicked or changed!" );
        }
    );
</script>
```

jQuery fundamentals - events data binding

```
<script type="text/javascript">
    var inputs = document.getElementsByTagName('input'), i;
    for (i = 0; i < inputs.length; i++) {
        inputs[i].onchange = function () {
            var foo = "bar";
            return function () {
                console.log("Foo = " + bar);
            }
        }();
    }
</script>
```

jQuery fundamentals - events data binding

```
<script type="text/javascript">
    var inputs = document.getElementsByTagName('input'), i;
    for (i = 0; i < inputs.length; i++) {
        inputs[i].onchange = function () {
            var foo = "bar";
            return function () {
                console.log("Foo = " + bar);
            }
        }();
    }

    $( "input" ).on(
        "change",
        { foo: "bar" },
        function( event ) {
            console.log("Foo = " + event.data.foo);
        }
    );
</script>
```

Czym jest plugin jQuery?

Czym jest plugin jQuery?

Biblioteka JavaScript dostosowana do pewnego API

Czym jest plugin jQuery?

Biblioteka JavaScript dostosowana do pewnego API

jQuery jest tutaj tylko przykładem (!) - te same zasady obowiązują w innych bibliotekach (np. underscore.js)

Typy pluginów w jQuery

Typy pluginów w jQuery

mutatory - modyfikacja elementów DOM

Typy pluginów w jQuery

mutatory - modyfikacja elementów DOM

utility - dodatkowe funkcjonalności, nie związane z drzewem DOM

Typy pluginów w jQuery

mutatory - modyfikacja elementów DOM

utility - dodatkowe funkcjonalności, nie związane z drzewem DOM

widgets - bardziej rozbudowane rozszerzenia wspierające stanowość (tylko jQuery UI)

Mutator plugin - podstawy

Prototyp obiektu jQuery znajduje się w property jQuery.fn

Mutator plugin - podstawy

Prototyp obiektu jQuery znajduje się w property jQuery.fn

Funkcje mutujące dodajemy do prototypu jQuery

Mutator plugin - podstawy

Prototyp obiektu jQuery znajduje się w property jQuery.fn

Funkcje mutujące dodajemy do prototypu jQuery

jQuery nie gwarantuje obecności aliasu $ (!)

Mutator plugin - podstawy

Mutator plugin - podstawy

Każda funkcja typu mutator wykonywana jest na obiekcie jQuery

Mutator plugin - podstawy

Każda funkcja typu mutator wykonywana jest na obiekcie jQuery

Obiektem jQuery może być pojedynczy element DOM lub wiele takowych (obiekt "array-like")

Mutator plugin - podstawy

Każda funkcja typu mutator wykonywana jest na obiekcie jQuery

Obiektem jQuery może być pojedynczy element DOM lub wiele takowych (obiekt "array-like")

Referencja do elementu jQuery znajduje się w zmiennej this funkcji mutatora

Mutator plugin - przykład

```
(function ($) {
    $.fn.greenify = function() {
        this.css( "color", "green" );
        this.addClass( "greenified" );
    }
})(jQuery);

$( "a" ).greenify();
```

jQuery chaining

```
<script type="text/javascript">
    $('div.container')
        .css( 'background-color', 'red' )
        .width( 100 )
        .height( 150 )
        .addClass( 'container-fluid' );
</script>
```

jQuery chaining

Chaining pozwala na wykonywanie wielu operacji na jednym elemencie/grupie elementów

jQuery chaining

Chaining pozwala na wykonywanie wielu operacji na jednym elemencie/grupie elementów

Każda operacja powinna być wykonywana na wszystkich elementach grupy

jQuery chaining

Chaining pozwala na wykonywanie wielu operacji na jednym elemencie/grupie elementów

Każda operacja powinna być wykonywana na wszystkich elementach grupy

Element/grupa powinna być zwracana przez funkcję do dalszego chainingu

jQuery chaining - przykład

```
(function ($) {
    $.fn.greenify = function() {
        this.each(function (_, element) {
            var $element = $(element);

            $element.css( "color", "green" );
            $element.addClass( "greenified" );
        });

        return this;
    }
})(jQuery);

$( "a" ).greenify().show();
```

jQuery plugin options

Opcje przekazywane do funkcji powinny być opcjonalne kiedy tylko to możliwe

jQuery plugin options

Opcje przekazywane do funkcji powinny być opcjonalne kiedy tylko to możliwe

Domyślny zestaw opcji powinien być dostępny globalnie i modyfikowalny

jQuery plugin options

Opcje przekazywane do funkcji powinny być opcjonalne kiedy tylko to możliwe

Domyślny zestaw opcji powinien być dostępny globalnie i modyfikowalny

Funkcje wykorzystywane przez plugin także powinny być dostępne globalnie i modyfikowalne

jQuery plugin options

```
(function ($) {
    // Plugin definition.
    $.fn.hilight = function( options ) {
        var opts = $.extend( {}, $.fn.hilight.defaults, options );
    };

    // Plugin defaults – added as a property on our plugin function.
    $.fn.hilight.defaults = {
        foreground: "red",
        background: "yellow"
    };
})(jQuery);

fn.hilight.defaults.background = "red";

$( "a" ).hilight();
$( "p" ).hilight( { foregrounbd: "yellow" } );
$( "span" ).hilight( { foregrounbd: "blue", background: "white" } );
```

jQuery plugin callbacks

jQuery plugin callbacks

Istnieją dwa sposoby dostarczania funkcjonalności reakcji na pewne zdarzenia biblioteki

Istnieją dwa sposoby dostarczania funkcjonalności reakcji na pewne zdarzenia biblioteki

Callbacks - czysty JavaScript, jako opcje,

Callbacks Driven Development

jQuery plugin callbacks

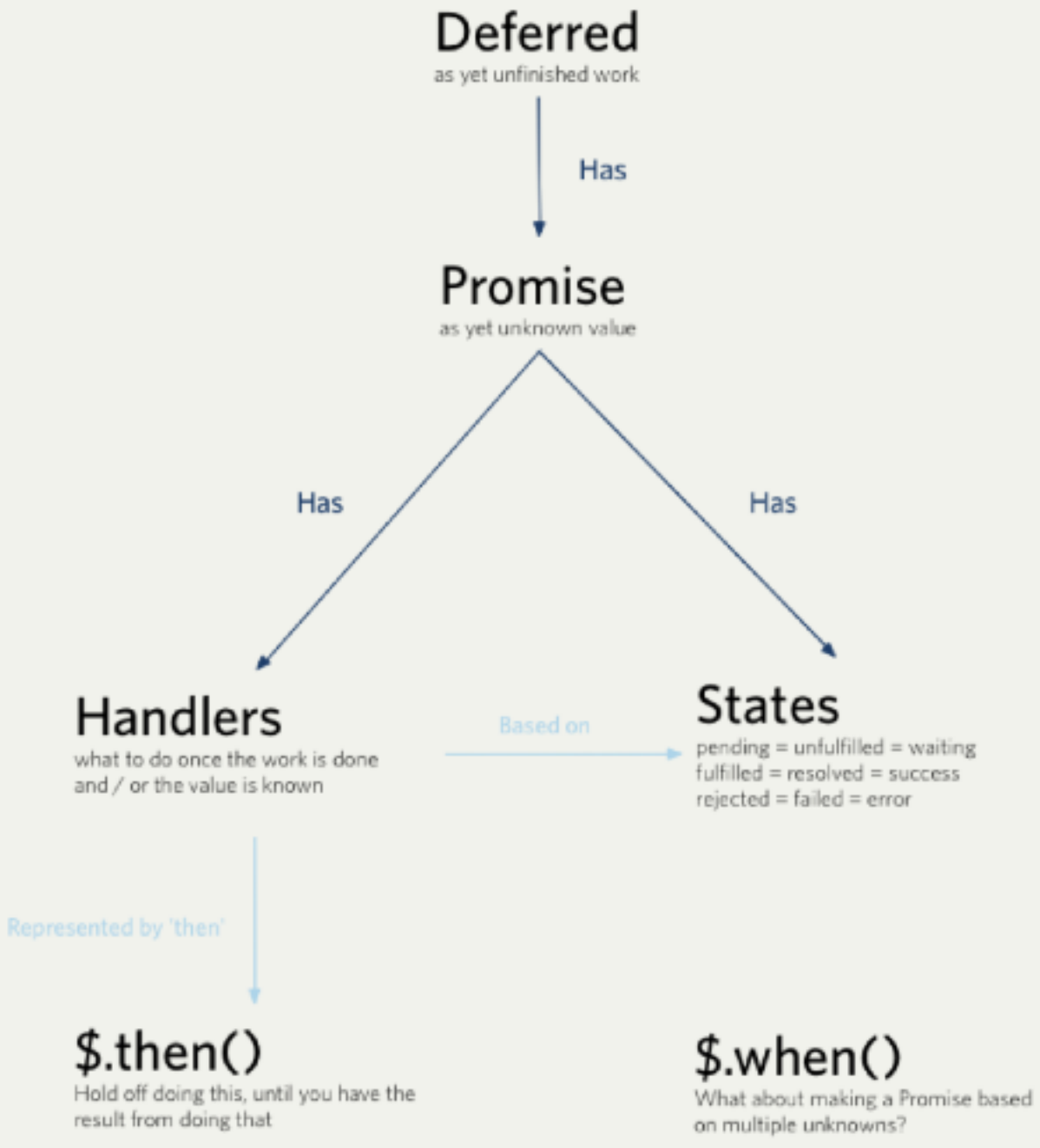
Istnieją dwa sposoby dostarczania funkcjonalności reakcji na pewne zdarzenia biblioteki

Callbacks - czysty JavaScript, jako opcje,

Callbacks Driven Development

Deferred/Promise pattern

Deferred/Promise pattern



Deferred/Promise pattern

```
(function ($) {
    $.fivesecs = function() {
        var defer = new $.Deferred();

        function logic(num, dfd) {
            setTimeout(function(){
                if (num == 5) {
                    dfd.resolve();
                } else {
                    dfd.notify(num * 20);
                }
            }, num * 1000);
        }

        for (var i = 1; i < 6; i++){
            logic(i, defer);
        }

        return defer.promise();
    };
})(jQuery);
```

Deferred/Promise pattern

```
jQuery.fivesecs().
    done(function () {
        console.log("Countdown complete!");
    }).
    progress(function (percentage) {
        console.log("Countdown done in " + percentage + "%");
    });
```

Deferred/Promise - $.when()

```
$.when( $.ajax( "test.aspx" ) ).then(function( data, textStatus, jqXHR
) {
    alert( jqXHR.status ); // Alerts 200
});

$.when().then(function() {
    alert( "I fired immediately" );
});
```

Deferred/Promise - $.when()

```
var d1 = $.Deferred(), d2 = $.Deferred(),
    defers = [d1, d2];

$.when( d1, d2 ).done(function ( v1, v2 ) {
    console.log( v1 ); // "Fish"
    console.log( v2 ); // "Pizza"
});

$.when.apply( $, defers ).done(function ( v1, v2 ) {
    console.log( v1 ); // "Fish"
    console.log( v2 ); // "Pizza"
});

d1.resolve( "Fish" );
d2.resolve( "Pizza" );
```

jQuery UI Widget Factory

jQuery UI Widget Factory

Dostępne w jQuery UI od wersji 1.8

jQuery UI Widget Factory

Dostępne w jQuery UI od wersji 1.8

Pozwala budować pluginy jQuery posiadające stan - przynależny do każdej instancji

jQuery UI Widget Factory

Dostępne w jQuery UI od wersji 1.8
Pozwala budować pluginy jQuery posiadające stan - przynależny do każdej instancji
Widget to nazwa oficjalnie wspieranych funkcjonalności jQuery UI

jQuery UI Widget Factory

Dostępne w jQuery UI od wersji 1.8

Pozwala budować pluginy jQuery posiadające stan - przynależny do każdej instancji

Widget to nazwa oficjalnie wspieranych funkcjonalności jQuery UI

Pluginy tworzone za pomocą Widget Factory nie muszą być związane z UI (!)

jQuery UI Widget Factory - podstawy

$.widget(name, functionsObject) - funkcja tworząca nasz widget

jQuery UI Widget Factory - podstawy

$.widget(name, functionsObject) - funkcja tworząca nasz widget

Kontekstem widgetu jest obiekt, nie element DOM (!)

jQuery UI Widget Factory - podstawy

$.widget(name, functionsObject) - funkcja tworząca nasz widget

Kontekstem widgetu jest obiekt, nie element DOM (!)

Nazwa pluginu musi zawierać przestrzeń nazw ("<namespace>.<name>")

jQuery UI Widget Factory - podstawy

$.widget(name, functionsObject) - funkcja tworząca nasz widget

Kontekstem widgetu jest obiekt, nie element DOM (!)

Nazwa pluginu musi zawierać przestrzeń nazw ("<namespace>.<name>")

Przestrzeń nazw "ui" jest zarezerwowana dla biblioteki jQuery UI

jQuery UI Widget Factory - _create

```
$.widget( "nmk.progressbar", {

    _create: function() {
        var progress = this.options.value + "%";
        this.element.addClass( "progressbar" ).text( progress );
    }

});

$( "<div />" ).appendTo( "body" ).progressbar({ value: 20 });
```

jQuery UI Widget Factory - opcje domyślne

```
$.widget( "nmk.progressbar", {

    // Default options.
    options: {
        value: 0
    },

    _create: function() {
        var progress = this.options.value + "%";
        this.element.addClass( "progressbar" ).text( progress );
    }

});

$( "<div />" ).appendTo( "body" ).progressbar();
```

jQuery UI Widget Factory - metody publiczne

```
$.widget( "nmk.progressbar", {

    // ...

    value: function( value ) {

        if ( value === undefined ) {
            return this.options.value;
        } else {
            this.options.value = this._constrain( value );
            var progress = this.options.value + "%";
            this.element.text( progress );
        }
    }
});

var $progressbar = $( "<div />" ).appendTo( "body" ).progressbar();

$progressbar.value(40);
console.log($progressbar.value()); // 40
```

jQuery UI Widget Factory - metody prywatne

```
$.widget( "nmk.progressbar", {

    // ...

    _constrain: function( value ) {

        if ( value > 100 ) {
            value = 100;
        }

        if ( value < 0 ) {
            value = 0;
        }

        return value;
    }
});
```

jQuery UI Widget Factory - zmiana opcji

```
$.widget( "nmk.progressbar", {
    // ...

    _create: function() {
        this.element.addClass( "progressbar" );
        this._update();
    },

    _setOption: function( key, value ) {
        this.options[ key ] = value;
        this._update();
    },

    _update: function() {
        var progress = this.options.value + "%";
        this.element.text( progress );
    }
});

var $progressbar = $( "<div />" ).appendTo( "body" ).progressbar();

$progressbar.option("value", 35);
console.log($progressbar.option("value")); // 35
```

jQuery UI Widget Factory - callbacks

```
$.widget( "nmk.progressbar", {
    // ...

    _update: function() {
        var progress = this.options.value + "%";
        this.element.text( progress );

        if ( this.options.value == 100 ) {
            this._trigger( "complete", null, { value: 100 } );
        }
    }
});

var $progressbar = $( "<div />" ).appendTo( "body" ).progressbar({
    complete: function( event, data ) {
        console.log( "Callbacks are great!" );
    }
});

$progressbar.option("value", 100);
```

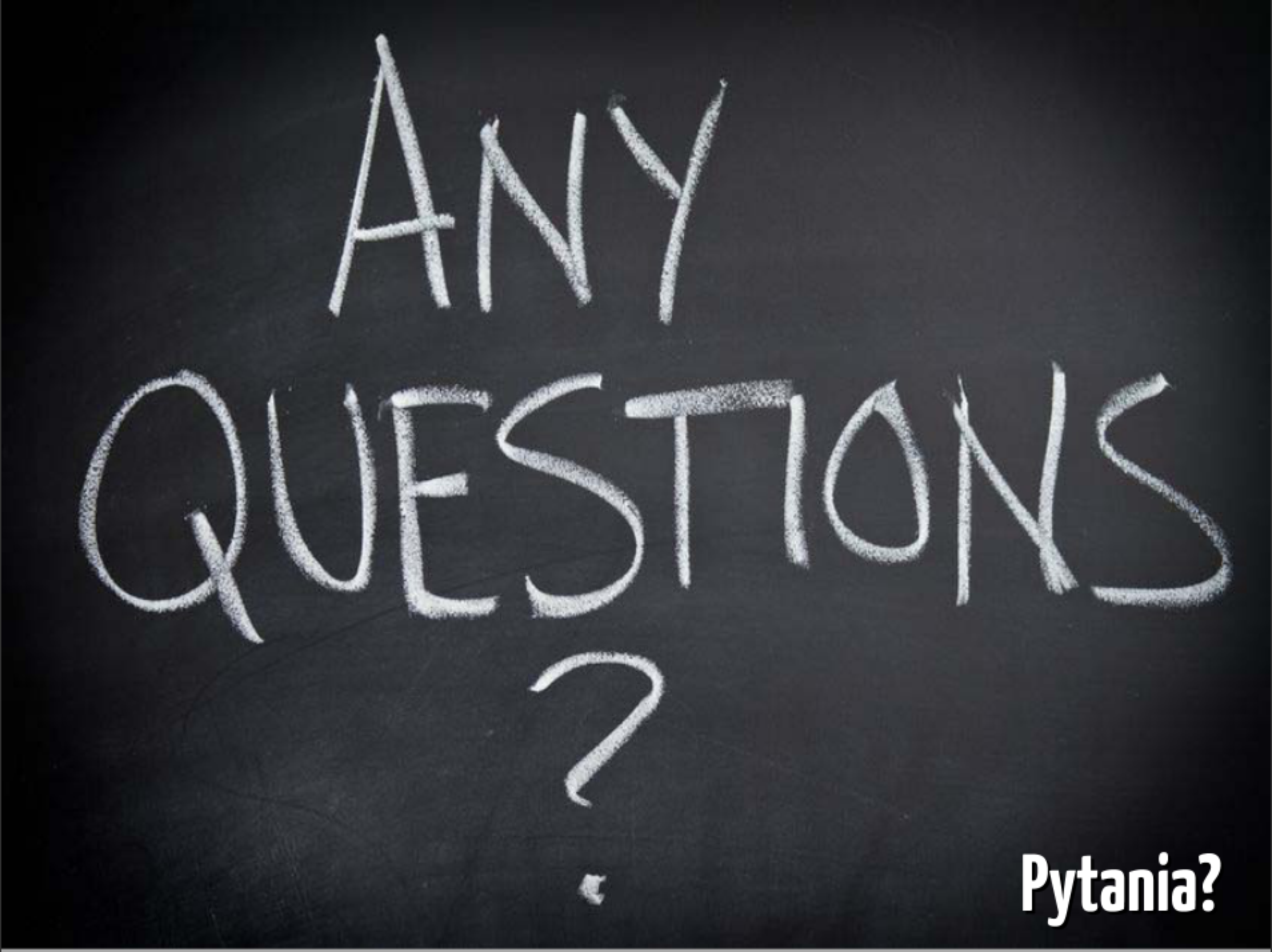
jQuery UI Widget Factory - callbacks

```
$.widget( "nmk.progressbar", {
    // ...

    _destroy: function() {
        this.element
            .removeClass( "progressbar" )
            .text( "" );
    }
});
```

```
var $progressbar = $( "<div />" ).appendTo( "body" ).progressbar();

$progressbar.destroy();
```



Pytania?