scottoffen / **jquery.toaster**

jQuery plugin for displaying customizable toast notifications via Bootstrap alerts http://scottoffen.github.io/jquery.toaster/

scottoffen Update README.md — Latest commit 576deb1 on Sep 10, 2015

.gitignore	Update .gitignore	2 years ago
LICENSE	Create LICENSE	a year ago
README.md	Update README.md	6 months ago
demo.html	Update demo.html	7 months ago
jquery.toaster.js	Fixed issue #8	7 months ago
toaster.jquery.json	Create toaster.jquery.json	a year ago

README.md

jQuery Toaster

Requires Bootstrap 3.0+

Toaster is a jQuery plugin for displaying toast notifications. It comes with a sublime set of defaults that you can use out of the box, while at the same time remains completely flexible; bending to meet your specific design needs.

> Toaster includes a polyfill for `Array.indexOf` to support older browsers (lte IE8).

What Is This Toast You Speak Of?

While *toast* is most often used to describe sliced bread that has been browned by dry heat, it can really refer to any kind of material that has been browned in such fashion.

Or - as is our case - toast can refer to a **non-modal**, unobtrusive UI element used to display brief, **auto-expiring** (aka auto-dismissing) windows of **information** to a user. It does not accept focus or user input, nor does it interrupt the current activity.

> While the definitive origin of the term "toast" to describe this type of notification system is unknown to me, it could reasonably be a reference to either (a) the salutation or words of congratulations, good wishes, appreciation, etc., spoken immediately before drinking to a person or event *or* (b) the fact that it pops up like bread from a toaster. Cheers!

Usage

Include the `jquery.toaster.js` JavaScript file on your HTML page after referencing jQuery, and then display toast messages anytime like this:

```
/*
* Pass an parameters as an object:
* The only required paramter is message
* Order does not matter
*/
$.toaster({ message : 'Your message here' });
$.toaster({ message : 'Your message here', title : 'Your Title' });
$.toaster({ message : 'Your message here', title : 'Your Title', priority : 'danger' });

/*
* Pass parameters as strings:
* The only required parameter is message
* Order matters!
*/
$.toaster('Your message here');
$.toaster('Your message here', 'Your Title');
$.toaster('Your message here', 'Your Title', 'danger');
```

Priorities

The `priority` property is based on the contextual colors available on the Bootstrap alert component. Available options built into Bootstrap are:

- success
- info
- warning
- danger

In this fashion, Toaster messages will automatically match the Bootstrap theme you are using. If you have defined (named) additional contextual colors that apply to the Bootstrap alert component, feel free to use them! Everything should work just fine.

How It Works

Simply put, auto-dismissing alerts are added to a `div` element designed to hold them. This 'toast holder' element - referred to as the 'toaster' - will be created if it doesn't already exist, so you really don't need to do *anything* to use Toaster out of the box.

> If the defaults work for you, great! You don't need to read any further than this. Live as long as is expedient and may your prosperity be equivalent to the effort you invest and the value of your output.

Settings

As noted earlier, Toaster has a great set of default settings that mean you can use it without modification. But, if you'd like to customize the settings, it can easily be done - and undone!

Customizing

To customize the settings, pass a `settings` object with your desired changes.

```
$.toaster({ settings : {...} });
```

Only what is defined in the object you send will override the default settings, all other settings will remain untouched. The settings are applied before any message (that may have been passed at the same time) is created and displayed.

> This only needs to be done once. Thereafter, toaster will use the settings you have provided until you change them

> again or reset them.

To clear all changes made and revert to the default settings:

```
$.toaster.reset();
```

Settings Object

The settings object allows you to change everything about the plugin. Let's take a look at the default settings, and then we'll dig into them from top to bottom!

```
{
    'toaster'         :
    {
        'id'        : 'toaster',
        'container' : 'body',
        'template'  : '<div></div>',
        'class'     : 'toaster',
        'css'       :
        {
            'position' : 'fixed',
            'top'      : '10px',
            'right'    : '10px',
            'width'    : '300px',
            'zIndex'   : 50000
        }
    },

    'toast'       :
    {
        'template' :
        '<div class="alert alert-%priority% alert-dismissible" role="alert">' +
            '<button type="button" class="close" data-dismiss="alert">' +
                '<span aria-hidden="true">&times;</span>' +
                '<span class="sr-only">Close</span>' +
            '</button>' +
            '<span class="title"></span>: <span class="message"></span>' +
        '</div>',

        'css'      : {},
        'cssm'     : {},
        'csst'     : { 'fontWeight' : 'bold' },

        'fade'     : 'slow',

        'display'    : function ($toast)
        {
            return $toast.fadeIn(settings.toast.fade);
        },

        'remove'     : function ($toast, callback)
        {
            return $toast.animate(
                {
                    opacity : '0',
                    padding : '0px',
                    margin  : '0px',
                    height  : '0px'
                },
                {
                    duration : settings.toast.fade,
                    complete : callback
                }
            );
```

```
        }
    },

    'debug'        : false,
    'timeout'      : 1500,
    'stylesheet'   : null,
    'donotdismiss' : []
}
```

The properties of the settings object are:

Property	Default	Description
toaster	object	see toaster
toast	object	see toast
debug	false	A boolean (or truthy/falsey) value to indicate that debuging mode is on/off. If it is on (true/truthy), then the notification element is written out to `console.log` prior to being added to the DOM so it can be inspected.
timeout	1500	An integer, the number of milliseconds to wait before calling the `settings.toast.remove` method.
stylesheet	null	A path to a stylesheet that should be included whenever this plugin is used. You can hardcode a value here if you'd prefer. If the stylesheet referenced is not found on the page, it will be added.
donotdismiss	array	This is expected to be an array of priorities that should not be auto-dismissed, empty by default. Any notification with a priority in this array will not have the `settings.toast.remove` method called on it after the `settings.timeout` amount of time, and will need to be manually dismissed.

toaster

The toaster is where all toast notifications will appear. This section of the settings allows us to identify and/or configure the toaster.

Property	Default	Description
id	toaster	The id attribute of the toaster element. You can create the toaster, give it this id and style it yourself, or this is the id that will be assigned to the toaster DOM element that gets created. If an element with this id already exists, the rest of the toaster properties are ignored.
container	`body`	The container element that the toaster element will be attached to.
template	`<div></div>`	The template used to create the toaster element.
class	toaster	The class (or classes) to be applied to the template when creating the toaster DOM element.
css	object	Style attributes to be applied to the toaster DOM element.

toast

The toast is the notification template that will be used to create all toaster notifications. This section is used to define that template and its relevant attributes.

Property	Default	Description
		The html template (as a string) that will be used to create each notification. It can (and should)

template	alert	contain placeholders for the priority (using `%priority%`), and there should be elements with the classes `title` and `message`, as these will be used to put the text for the title and message property values.
css	object	Style attributes to be applied to each toast generated.
cssm	object	Style attributes to be applied to each element with the class `message`.
csst	object	Style attributes to be applied to each element with the class `title`.
fade	slow	Defines the duration for the fade in/out effect on the toast notification.
display	function	Callback to handle the initial display of the toast notification element.
remove	function	Callback to handle the remove of the toast notification element.

Note that `settings.toast.fade` is only used by the default `settings.toast.display` and `settings.toast.remove` callback functions. If you wanted to have toast elements fade in at different speeds than they fade out, you can use this modification:

```
$.toaster({ settings :
{
    toast :
    {
        fade : { in : 'fast', out : 'slow' },

        display : function ($toast)
        {
            return $toast.fadeIn(settings.toast.fade.in);
        },

        remove : function ($toast, callback)
        {
            return $toast.animate(
            {
                opacity: '0',
                height: '0px'
            },
            {
                duration: settings.toast.fade.out,
                complete: callback
            });
        }
    }
}});
```

Also note that when you customize settings, you can **add** properties, too! So if, for example, you don't want the notifications to fade in/out, but want them them to slide instead; you can use this (completely untested) modification:

```
$.toaster({ settings :
{
    toast :
    {
        fade : { in : 'fast', out : 'slow' },
        easing : { in : 'swing', out : 'linear' },

        display : function ($toast)
        {
            return $toast.slideToggle(settings.toast.fade.in, settings.toast.easing.in);
        },

        remove : function ($toast, callback)
        {
            return $toast.slideToggle(settings.toast.fade.out, settings.toast.easing.out, callback);
        }
```

```
    }
}});
```

Remember that you have complete access to the `settings` object inside these callback methods, so if you want to key off of another value - either a documented one like `settings.debug` or a custom one of your own creation - you can do that!

> In all cases, note that the `settings.toast.remove` callback should take a callback method as the second argument, and don't forget to call it!

Variable Timeouts

If you want notifications of different priorities to expire at different intervals, you'll be happy to know that the `settings.timeout` property can also be an object! In that case, the key should be the priority and the value the number of milliseconds to wait before auto-dismissing the notification.

Any priority without a key-value pair will default to a 1500 millisecond timeout.