jQuery: how to use the DOMNodeInserted event and why it's important

What happens to the DOM when a new node is inserted? The new DOM specifications answer this question with the introduction of a new event, namely `DOMNodeInserted`. Now imagine this case-scenario: you're using a complex, AJAX-driven notification system with jQuery. The problem with this system is that each message must be removed from the page after a couple of seconds. Knowing the attributes and the position of the messages surely helps you, but you need to automatize the process for every new element inserted via AJAX. Here's where the `DOMNodeInserted` event comes into play.

To properly use this event with jQuery you need the `bind()` method. Passing the `event` object to the event handler allows you to access its `target` property to make a test on the current element. Bear in mind that you get a DOM node reference, so you need to use the `$()` wrapper to access jQuery's methods:

```
$(document).bind('DOMNodeInserted', function(e) {
    var element = e.target;
    setTimeout(function() {
        $(element).fadeOut(1000, function() {
            $(this).remove();
        });
    }, 2000);
});
```

Here's an example:

```
$('#insert').click(function() {
    $('<div id="test"/>').
    text('Test').
    appendTo('body');
});
```

Using the `event.target` reference allows you to make sure that the current element is exactly your desired target element:

```
var element = e.target;
if($(element).is('div.success')) {
        // ...
}
```

Demo

Live demo