

Demystifying Drupal AJAX Callback Commands

<http://nyccamp.org/node/287>

NYC CAMP 2015

#NYCcamp

Goals of this Session

- *Explain what AJAX callback commands are*
- *Demonstrate how to use AJAX callback commands*
- *Outline how to create AJAX callback commands*

Michael Miles

From: Boston, MA USA

Work: [Genuine](#) @WeAreGenuine(.com)

Exp: Working with Drupal since 2008.

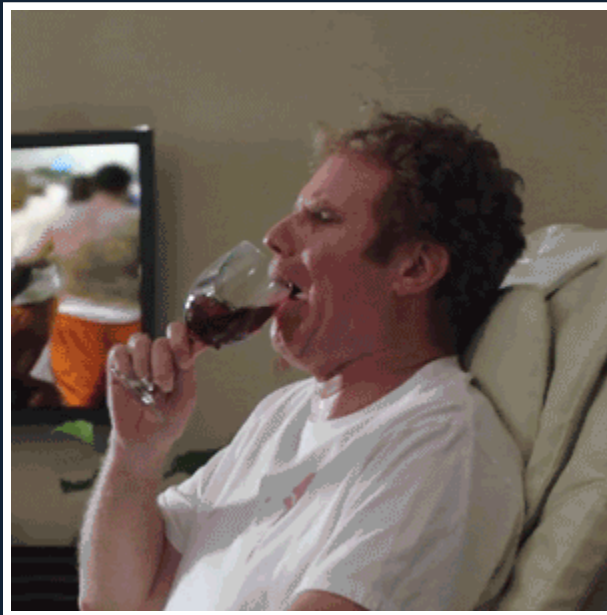
Acquia Certified. 2014 Acquia MVP.

Twitter: @mikemiles86 **Drupal.org:** mikemiles86

All the Places: mikemiles86

Warning About Example Code

- *There is example code*
- *Some coding standards ignored for presentation purposes*
- *Drupal 8 code still subject to change*



What Commands Are



From a High Level

- *The response of AJAX request*
- *PHP code to instruct JavaScript functions*
- *A JavaScript function attached to an object*
- *Defined by Drupal core or modules*
 - *17 AJAX callback commands in Drupal 7*

Example: The "Remove" Command

The screenshot shows a web browser window with the address bar displaying `drupal7.dev/examples/ajax_example/advanced_commands`. The page content is organized into several sections:

- Original contents**
Status: Unknown
- This shows the AJAX 'prepend' command.** Click to put something below the div that says
'Something can be inserted after this'
AJAX 'prepend': Click to prepend something
Something can be prepended to this div...
'After' Command Status: Unknown
- Shows the Ajax 'remove' command.**
☒ AJAX 'remove': Check to remove text. Uncheck to add it back.
text to be removed
Updated remove_command_example (value=FALSE Thu, 16 Jul 2015 21:28:05 -0400)
- Demonstrates the Ajax 'restripe' command.**
1
Restripe the table
first row
'Restripe' Command Status: Unknown
- Submit**

Using the "examples" module (available on Drupal.org), showing using the 'remove' AJAX Callback command. The checkbox is checked, which makes an AJAX request, which returns the command to remove the element with the sample text.

On the Client Side [D7]

- *A function on the 'Drupal.ajax.prototype.commands' object*
 - *Receives 'ajax', 'response' and 'status' arguments*
- *A wrapper for additional javascript*

```
// Provide a series of commands that the server can request the client perform.
Drupal.ajax.prototype.commands = {
  //...
  /**
   * Command to remove a chunk from the page.
   */
  remove: function (ajax, response, status) {
    var settings = response.settings || ajax.settings || Drupal.settings;
    Drupal.detachBehaviors($(response.selector), settings);
    $(response.selector).remove();
  },
  //...
}
```

misc/ajax.js

On the Server Side [D7]

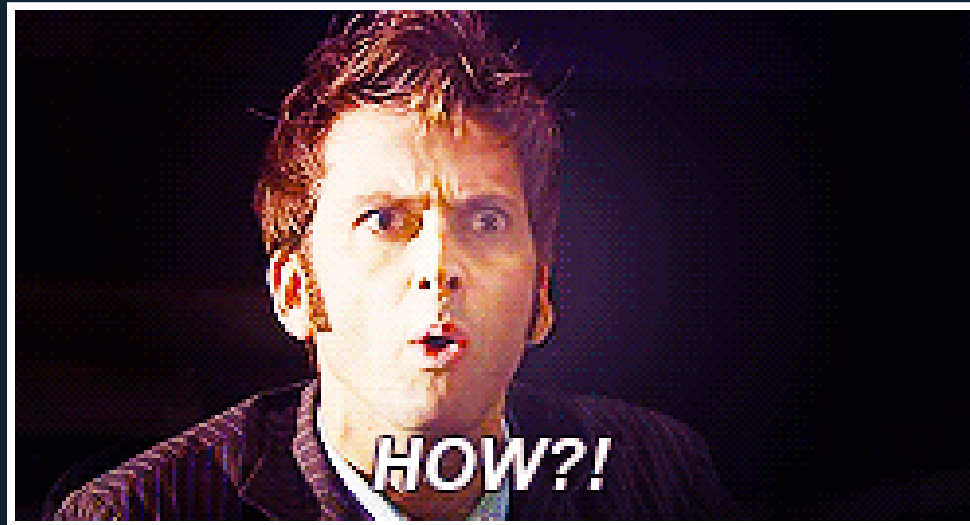
- *A PHP function which returns an associative array*
 - *Array must have an element with key of 'command'*
 - *The value is the name of a JavaScript function*
 - *Additional array elements set as response data*

```
/**
 * Creates a Drupal Ajax 'remove' command.
 */
function ajax_command_remove($selector) {
  return array(
    'command' => 'remove',
    'selector' => $selector,
  );
}
```

includes/ajax.inc



How to Use Commands



Example Scenario

As a user

When I navigate to the 'my-messages' page

Then I see a list of my unread messages

And I see a list of my read messages

When I click on an unread message

Then the full message is retrieved from the server

And I see the content of my message

And the message is removed from the unread messages list

And the message is added to the read messages list

drupal7.dev/my-messages

AJAX Commands demo

Home

Home

Navigation

- My Messages

User login

Username *

Password *

- Create new account
- Request new password

Log in

My Messages

Unread Messages	Drupal 8 is it ready	Read Messages
New BondWithin Contact The system is down 404 page?	Can we start building with Drupal 8 yet. We have some clients who are really interested.	Make money at home He did it again Taco dog An urgent request Great talk Janis In NYC this weekend? Agile Spec doc? Drupal 8 is it ready

drupal7.dev/read-message-callback/nojs/8

Example Scenario. Have a page that lists messages to read. When a message subject is clicked, an AJAX request is made to retrieve the message. The server returns an array of commands to run, which updates the page with the selected message.

To Use Commands...

1. *Include Drupal AJAX JavaScript library*
2. *Attach AJAX to page elements*
3. *Define a callback path and function*
 - *Returns a render array of commands [D7]*

1. Include Drupal AJAX library

```
// Render a page of messages.
function mymodule_messages_page() {
  // Include AJAX library.
  drupal_add_library('system', 'drupal.ajax');
  // Create initial page content.
  $content = theme_item_list(array(
    'title' => t('Unread Messages'),
    'type' => 'ul',
    'attributes' => array('id' => 'unread-msgs'),
    'items' => mymodule_messages_list(mymodule_get_unread_messages()),
  ));
  $content .= '<div id="current-msg"><h2></h2><p></p></div>';
  $content .= theme_item_list(array(
    'title' => t('Read Messages'),
    'type' => 'ul',
    'attributes' => array('id' => 'read-msgs'),
    'items' => mymodule_messages_list(mymodule_get_read_messages(), FALSE),
  ));

  return $content;
}
```

mymodule/mymodule.module

Function to build a page of messages. On line #4, using the `drupal_add_library()` function to include the AJAX library on the page.

2. Attach AJAX to page elements

```
// Return array of message titles for use in a list.
function mymodule_messages_list($messages, $display_link = TRUE) {
  $list_items = array();
  foreach ($messages as $mid => $msg) {
    // Build link to AJAX callback to load message.
    $link = l($msg->subject, 'read-message-callback/nojs/' . $mid, array(
      // Add class to link so Drupal knows to use AJAX.
      'attributes' => array('class' => array('use-ajax')),
    ));
    // Create list item for message.
    $list_items[] = array(
      'id' => 'msg-' . $mid,
      // Set data to read message link or just message subject.
      'data' => $display_link ? $link : $msg->subject,
    );
  }
  return $list_items;
}
```

mymodule/mymodule.module

*Function to build a list of messages. On line #6, creating a link element which links to our callback path.
On line #8, adding the class 'use-ajax' so AJAX framework knows to serve link with an AJAX request.*

3.a Define a callback path

```
// Implements hook_menu().
function mymodule_menu() {
  $items = array();
  //.. Other menu items.
  // None JS callback, for graceful degradation.
  $items['read-message-callback/nojs/%'] = array(
    'title' => 'AJAX Callback to Read Message',
    'access arguments' => array('access content'),
    'page callback' => 'mymodule_read_message_callback',
    'page arguments' => array(1,2),
    'type' => MENU_CALLBACK,
  );
  // Create AJAX callback. Add ajax_delivery callback.
  $items['read-message-callback/ajax/%'] = array(
    'delivery callback' => 'ajax_deliver',
  ) + $items['read-message-callback/nojs/%'];

  return $items;
}
```

mymodule/mymodule.module

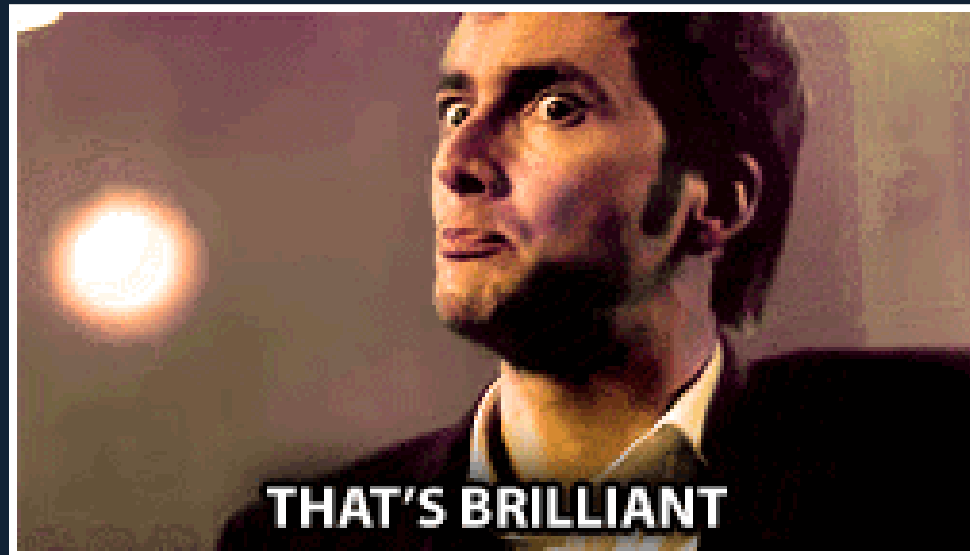
Creating two menu items for callback functions. The first is for handling graceful degradation when reached outside an AJAX Request. The second is the AJAX callback, which add a delivery callback of "ajax_callback" so Drupal know to return an AJAX Response object.

3.b Return array of commands using a render array [D7]

```
// AJAX Callback to read a message.
function mymodule_read_message_callback($method, $mid) {
  $message = mymodule_load_message($mid);
  // @TODO: Graceful degradation if not from ajax method or if no message.
  $commands = array(
    // Replace content of current message subject.
    ajax_command_html('#current-msg h2', $message->subject),
    // Replace content of current message body.
    ajax_command_html('#current-msg p', $message->content),
    // Remove message from unread list.
    ajax_command_remove('#msg-' . $mid),
    // Add message to read list
    ajax_command_append('#read-msgs', '<li>' . $message->subject . '</li>'),
  );
  // Return an AJAX render array.
  return array(
    '#type' => 'ajax',
    '#commands' => $commands,
  );
}
```

mymodule/mymodule.module

Lines #5 thru #14, building an array of Core AJAX Callback commands. Each maps to a javascript function. Line #16 building a render array of type 'ajax'. Drupal will package into JSON and send back to the client.



Creating Custom Commands



To Create a Command...

1. *Add a function to the Drupal ajax command JavaScript object*
2. *Write PHP code that returns an associative array.*
 - *Must have an element with key of 'command'*
 - *Value is name of the JavaScript function*
 - *Additional elements for response data*

That's it.



A Custom Command in Drupal 7

1. *Add a function to 'Drupal.ajax.prototype.commands'*
2. *Define a PHP function to return an array*

1. Add a function to 'Drupal.ajax.prototype.commands'

```
(function($, Drupal) {  
  // Add new command for reading a message.  
  Drupal.ajax.prototype.commands.readMessage = function ajax, response, status){  
    // Place content in current-msg div.  
    $('#current-msg h2').html(response.subject);  
    $('#current-msg p').html(response.content);  
    // Remove from unread list.  
    $('#msg-' + response.mid).remove();  
    // Add message to read list.  
    $('#read-msgs').append('<li>' + response.subject + '</li>');  
  }  
})(jQuery, Drupal);
```

mymodule/js/commands.js

Attach a new 'readMessage' command to the Drupal AJAX Commands JavaScript object. Function takes in the 3 arguments "ajax", "response" & "status". Wraps jQuery functions to display a message on the page.

2. Define a PHP function to return an associative array

```
/**
 * AJAX command to read a message.
 */
function mymodule_command_read_message($message) {
    return array(
        'command' => 'readMessage',
        'mid' => $message->mid,
        'subject' => $message->subject,
        'content' => $message->content,
    );
}
```

mymodule/mymodule.module

PHP function that maps to the new JavaScript 'readMessage' array. Returns an associative array, with an element that has a key of 'command' and value of the JavaScript function name 'readMessage'. Then additional elements to be sent as request data.

Using a custom command...

1. *Tell Drupal to include custom JavaScript*
2. *Return custom command in callback function*

Again...that's it.



Using a Custom Command in Drupal 7

1. *Use `drupal_add_js()` to include custom JavaScript on page*
2. *Add command to render array returned by callback*

1. Use `drupal_add_js()` to include custom JavaScript on page

```
// Render a page of my messages
function mymodule_messages_page() {
  // Include AJAX library.
  drupal_add_library('system', 'drupal.ajax');
  // Include custom JavaScript.
  drupal_add_js(drupal_get_path('module', 'mymodule') . '/js/commands.js');

  // .. Build $content.

  return $content;
}
```

mymodule/mymodule.module

The example function for creating the messages page. In addition to adding the Drupal AJAX library on line #4 also using `drupal_add_js()` to include custom javascript file on line #6.

2. Add command to render array returned by callback

```
// AJAX Callback to read a message.
function mymodule_read_message_callback($method, $mid) {
    $message = mymodule_load_message($mid);
    // @TODO: Graceful degradation if not from ajax method or if no message.
    $commands = array(
        // Call the readMessage javascript function.
        mymodule_command_read_message($message),
    );
    // Return an AJAX render array.
    return array(
        '#type' => 'ajax',
        '#commands' => $commands,
    );
}
```

mymodule/mymodule.module

The example callback function that returns an AJAX render array. Commands array on lines #5 thru #8, only returns a single command. The new custom command that was created.



Pretty cunning, don't you think?

AJAX Commands in Drupal 8



Changes to AJAX Commands

- *22 AJAX callback commands in Drupal 8*
- *JavaScript attaches to 'Drupal.AjaxCommands.prototype' object*
- *A PHP object that implements CommandInterface*
 - *Must implement 'render' method*

Example: The "Remove" Command

The screenshot shows a web browser window with the address bar displaying 'drupal7.dev/examples/ajax_example/advanced_commands'. The page content is organized into several sections:

- Original contents**
Status: Unknown
- This shows the AJAX 'prepend' command.** Click to put something below the div that says
'Something can be inserted after this'
AJAX 'prepend': Click to prepend something
Something can be prepended to this div...
'After' Command Status: Unknown
- Shows the Ajax 'remove' command.**
☒ AJAX 'remove': Check to remove text. Uncheck to add it back.
text to be removed
Updated remove_command_example (value=FALSE Thu, 16 Jul 2015 21:28:05 -0400)
- Demonstrates the Ajax 'restripe' command.**
1
Restripe the table
first row
'Restripe' Command Status: Unknown
- Submit**

Using the "examples" module (available on Drupal.org), showing using the 'remove' AJAX Callback command. The checkbox is checked, which makes an AJAX request, which returns the command to remove the element with the sample text.

A function on the 'Drupal.AjaxCommands.prototype' object

```
/**
 * Provide a series of commands that the client will perform.
 */
Drupal.AjaxCommands.prototype = {
  //...
  /**
   * Command to remove a chunk from the page.
   */
  remove: function (ajax, response, status) {
    var settings = response.settings || ajax.settings || drupalSettings;
    $(response.selector).each(function () {
      Drupal.detachBehaviors(this, settings);
    }).remove();
  },
  //...
}
```

misc/ajax.js

An object that implements CommandInterface

Must implement 'render' method

```
namespace Drupal\Core\Ajax;
use Drupal\Core\Ajax\CommandInterface;
// AJAX command for calling the jQuery remove() method.
class RemoveCommand implements CommandInterface {
    protected $selector;
    // Constructs a RemoveCommand object.
    public function __construct($selector) {
        $this->selector = $selector;
    }
    // Implements Drupal\Core\Ajax\CommandInterface:render().
    public function render() {
        return array(
            'command' => 'remove',
            'selector' => $this->selector,
        );
    }
}
```

core/lib/Drupal/Core/Ajax/RemoveCommand.php

To Use Commands in Drupal 8

1. *Include Drupal AJAX JavaScript library*
 - *Must attach to a render array*
2. *Attach AJAX to page elements*
3. *Define a callback path and function*
 - *Returns an AjaxResponse object*

drupal7.dev/my-messages

AJAX Commands demo

Home

Home

Navigation

- My Messages

User login

Username *

Password *

- Create new account
- Request new password

Log in

My Messages

Unread Messages	Drupal 8 is it ready	Read Messages
New BondWithin Contact The system is down 404 page?	Can we start building with Drupal 8 yet. We have some clients who are really interested.	Make money at home He did it again Taco dog An urgent request Great talk Janis In NYC this weekend? Agile Spec doc? Drupal 8 is it ready

drupal7.dev/read-message-callback/nojs/8

Example Scenario. Have a page that lists messages to read. When a message subject is clicked, an AJAX request is made to retrieve the message. The server returns an array of commands to run, which updates the page with the selected message.

Attach library to a render array

```
// Render a page of my messages
function mymodule_messages_page() {

    // .. build $content

    // Create render array.
    $page[] = array(
        '#type' => 'markup',
        '#markup' => $content,
    );
    // Attach JavaScript library.
    $page['#attached']['library'][] = 'core/drupal.ajax';

    return \Drupal::service('renderer')->render($page);
}
```

mymodule/mymodule.module

Function for building messages page is now using a render array. to the '#attached', 'library' array of that render array, declaring the drupal.ajax library to be included.

Callback returns commands using an AjaxResponse() [D8]

```
// AJAX Callback to read a message.
function mymodule_read_message_callback($method, $mid) {
  $message = mymodule_load_message($mid);
  // @TODO: Graceful degradation if not from ajax method or if no message.
  // Create AJAX Response object.
  $response = new AjaxResponse();
  // Replace content of current message subject.
  $response->addCommand(new HtmlCommand('#current-msg h2', $message->subject));
  // Replace content of current message body.
  $response->addCommand(new HtmlCommand('#current-msg p', $message->content));
  // Remove message from unread list.
  $response->addCommand(new RemoveCommand('#msg-' . $mid));
  // Add message to read list.
  $item = '<li>' . $message->subject . '</li>';
  $response->addCommand(new AppendCommand('#read-msgs', $item));
};
// Return ajax response.
return $response;
}
```

mymodule/mymodule.module

Returning an AjaxResponse object instead of a render array like in Drupal 7. AjaxResponse has an 'addCommand' method, which takes the argument of a command object.

Custom Commands in Drupal 8

1. *Add function to 'Drupal.AjaxCommands.prototype'*
2. *Define a command object that implements CommandInterface*

1. Add function to 'Drupal.AjaxCommands.prototype'

```
(function($, Drupal) {  
  /**  
   * Add new command for reading a message.  
   */  
  Drupal.AjaxCommands.prototype.readMessage = function(ajax, response, status){  
    // Place content in current-msg div.  
    $('#current-msg h2').html(response.subject);  
    $('#current-msg p').html(response.content);  
    // Remove from unread list.  
    $('#msg-' + response.mid).remove();  
    // Add message to read list.  
    $('#read-msgs').append('<li>' + response.subject + '</li>');  
  }  
})(jQuery, Drupal);
```

mymodule/js/commands.js

Adding the custom 'readMessage' function to the Drupal 8 JavaScript AJAX Commands object. Still takes the same three arguments as in Drupal 7.

2. Define command object that implements CommandInterface

```
namespace Drupal\mymodule\Ajax;
use Drupal\Core\Ajax\CommandInterface;

class ReadMessageCommand implements CommandInterface {
    protected $message;
    // Constructs a ReadMessageCommand object.
    public function __construct($message) {
        $this->message = $message;
    }
    // Implements Drupal\Core\Ajax\CommandInterface::render().
    public function render() {
        return array(
            'command' => 'readMessage',
            'mid' => $this->message->mid,
            'subject' => $this->message->subject,
            'content' => $this->message->content,
        );
    }
}
```

mymodule/src/Ajax/ReadMessageCommand.php

Creating a Command object which implements the CommandInterface. Must declare a 'render' method, which will return an associative array. This associative array must have an element with the key of 'command' and value of the JavaScript function to run.

Using a Custom Command in Drupal 8

1. *Tell Drupal about custom JavaScript library*
2. *Attach library to a render array*
3. *Add command to AjaxResponse object in callback*

1. Tell Drupal about custom JavaScript library

```
mymodule.commands:  
  version: VERSION  
  js:  
    js/commands.js: {}  
  dependencies:  
    - core/drupal.ajax
```

mymodule/mymodule.libraries.yml

Must tell Drupal about custom library with a libraries.yml file. Provide an Asset library name, lists all the js files needed, and any dependencies on other libraries.

2. Attach library to a render array

```
// Render a page of my messages
function mymodule_messages_page() {

    // .. build $content

    // Create render array.
    $page[] = array(
        '#type' => 'markup',
        '#markup' => $content,
    );
    // Attach JavaScript library.
    $page['#attached']['library'][] = 'mymodule/mymodule.commands';

    return \Drupal::service('renderer')->render($page);
}
```

mymodule/mymodule.module

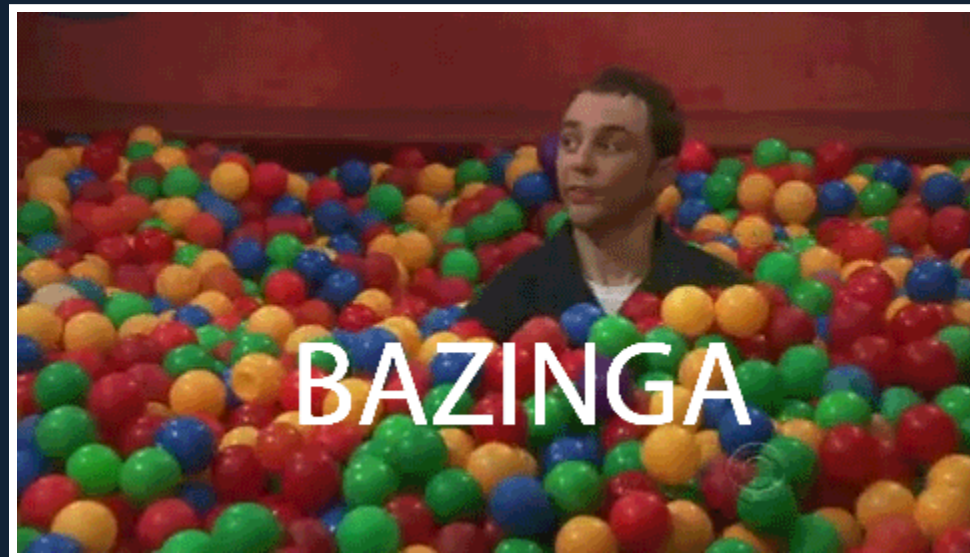
Function for building messages page is now using a render array. to the '#attached', 'library' array of that render array, including the custom library. No longer including the AJAX library, as Drupal will know to include it from the libraries.yml file.

3. Add command to AjaxResponse object in callback

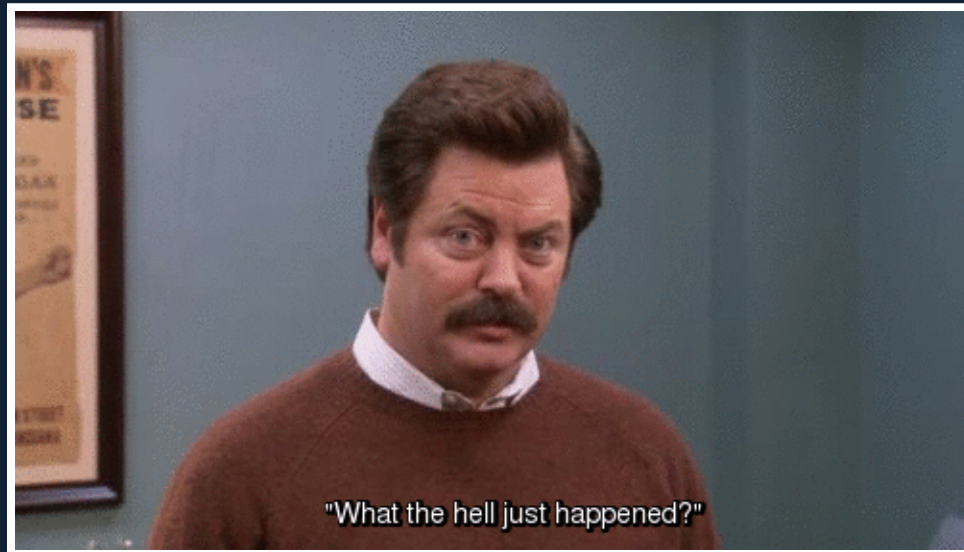
```
// AJAX Callback to read a message.
function mymodule_read_message_callback($method, $mid) {
    $message = mymodule_load_message($mid);
    // @TODO: Graceful degradation if not from ajax method or if no message.
    // Create AJAX Response object.
    $response = new AjaxResponse();
    // Call the readMessage javascript function.
    $response->addCommand( new ReadMessageCommand($message) );
};
// Return ajax response.
return $response;
}
```

mymodule/mymodule.module

Returning an AjaxResponse object instead of adding the core commands, just adding the new custom command using the 'addCommand' method.



Let's Review



AJAX Callback Commands Are...

- *Returned by all AJAX Framework requests*
- *PHP abstraction for JavaScript functions*
- *Defined by core and modules*

To Use AJAX Callback Commands...

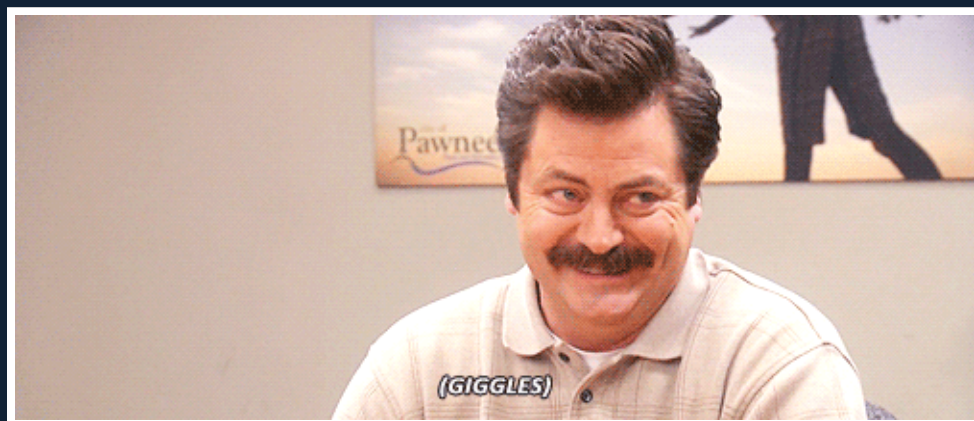
- *Include Drupal AJAX JavaScript library*
- *Attach AJAX to page elements*
- *Have server side callback return array of commands*

To Create AJAX Callback Commands...

- *Add function to Drupal ajax command JavaScript object.*
- *Write PHP code to return associative array with a 'command' key*

To Use Custom AJAX Callback Commands...

- *Include custom JavaScript*
- *Include custom command in callback function*



(GIGGLES)

Resources

Drupal AJAX Framework: bit.ly/DrupalAJAX

Examples Module: bit.ly/DrupalExMod

Drupal 7 Commands: bit.ly/D7ajaxCMD

Drupal 8 Commands: bit.ly/D8ajaxCMD

This Presentation: bit.ly/NYCcampAJAX

Presentation Slides: bit.ly/NYCcampAJAXSlides

Example Code: bit.ly/NYCcampAJAXEx

Send me feedback!

@mikemiles86

#NYCcamp

Thank You!

Questions?



#NYCcamp



@WeAreGenuine ~~AI~~ Callback Commands / @mikemiles86