Demystifying Ajax Callback Commands

(in Drupal 8)

events.drupal.org/node/8466

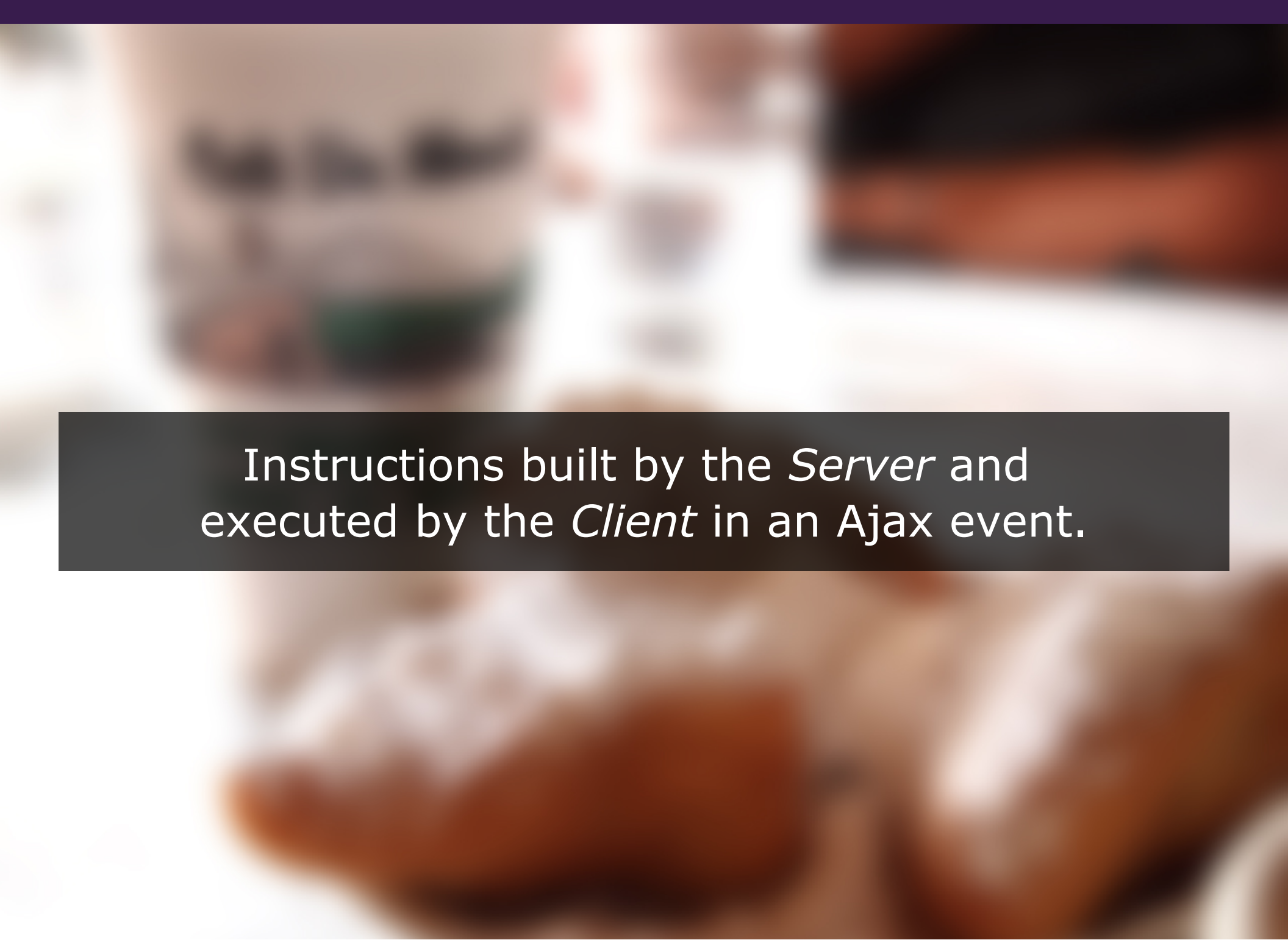
Drupalcon 2016

Mike Miles

Genuine (wearegenuine.com)

All the internet places: mikemiles86

Defining Callback Commands


Instructions built by the *Server* and
executed by the *Client* in an Ajax event.

Callback Command: JavaScript

- *Attached to 'Drupal.AjaxCommands.prototype'*
  - *Defined in 'misc/ajax.js'*
- *Accepts 3 arguments:*
  - *ajax*
  - *response*
  - *status*
- *Wrapper for additional JavaScript*

Callback Structure: JavaScript

```
(function ($, window, Drupal, drupalSettings) {
  'use strict';
  /**
   * [commandName description]
   *
   * @param {Drupal.Ajax} [ajax]
   * @param {object} response
   * @param {number} [status]
   */
  Drupal.AjaxCommands.prototype.[commandName] = function(ajax, response, status){

    // Custom javascript goes here ...

  }

})(jQuery, this, Drupal, drupalSettings);
```

[module]/js/[javascript].js

Example of the structure for the JavaScript half of a callback command as defined in a module.

Callback Command: PHP

- *Class that implements CommandInterface*
- *Defines a method called 'render'*
- *Returns an associative array:*
  - *Must have element with key of 'command'*
  - *Value must be name of JavaScript function*
  - *Other elements passed as response data*

Callback Structure: PHP

```php
namespace Drupal\[module]\Ajax
use Drupal\Core\Ajax\CommandInterface;

// An Ajax command for calling  [commandName]() JavaScript method.
class [CommandName]Command implements CommandInterface {

  // Implements Drupal\Core\Ajax\CommandInterface:render().
  public function render() {
    return array(
      'command' => '[commandName]', // Name of JavaScript Method.
      // other response arguments...
    );
  }
}
```

[module]/src/Ajax/[CommandName]Command.php

Example of the structure for the PHP half of a callback command as defined in a module.

Core Example: Remove

```
Drupal.AjaxCommands.prototype = {
  // ...
  /**
   * Command to remove a chunk from the page.
   *
   * @param {Drupal.Ajax} [ajax]
   * @param {object} response
   * @param {string} response.selector
   * @param {object} [response.settings]
   * @param {number} [status]
   */
  remove: function (ajax, response, status) {
    var settings = response.settings || ajax.settings || drupalSettings;
    $(response.selector).each(function () {
      Drupal.detachBehaviors(this, settings);
    })
      .remove();
  },
  //...
```

misc/ajax.js

The JavaScript function for the core 'remove' callback command. It is basically a wrapper for the jQuery 'remove' method.

Core Example: RemoveCommand

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

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

The PHP class for the core 'remove' callback command. Implements CommandInterface, so it must define the method 'render' that returns an associative array.

PHP

```
01 //...
02 public function render() {
03   return array(
04     'command' => 'remove',
05     'selector' => $this->selector,
06   );
07 }
```

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

JavaScript

```
01 //...
02 remove: function (ajax, response, status) {
03   var settings = response.settings || ajax.settings || drupalSettings;
04   $(response.selector).each(function () {
05     Drupal.detachBehaviors(this, settings);
06   })
07     .remove();
08 },
```

misc/ajax.js

Can see how the two halfs are tied together. Value on line #4 of PHP matches JavaScript function name defined on line #2 in JavaScript. Passed CSS selector on line #5 in PHP is used on line #4 in JavaScript.

Callback Commands

- *Used in all Ajax requests*
- *Composed of two parts: JavaScript function, PHP Class*
- *Provided by core and modules*

Creating Callback Commands

Example Scenario

Create a callback command for the jQuery 'SlideDown' animation

Create a Module

```
name: 'Slide Down Command'
type: module
description: Provides an Ajax Callback command for the jQuery SlideDown method.
package: other
core: 8.x
```

slide_down/slide_down.info.yml

Custom Ajax callback commands must be defined in a module.

Create JavaScript Function

```
01 (function ($, window, Drupal, drupalSettings) {
02
03   'use strict';
04
05   // Command to Slide Down page elements.
06   Drupal.AjaxCommands.prototype.slideDown = function(ajax, response, status){
07     // Get duration if sent, else use default of slow.
08     var duration = response.duration ? response.duration : "slow";
09     // slide down the selected element(s).
10     $(response.selector).slideDown(duration);
11   }
12 })(jQuery, this, Drupal, drupalSettings);
```

slide_down/js/slidedown-command.js

Attach a JavaScript function to the AjaxCommands object provided by the Ajax Framework. Accepts the three arguments and is a wrapper for the jQuery method.

Create Asset Library

```
slidedown:
  version: VERSION
  js:
    js/slidedown-command.js; {}
  dependencies:
    - core/drupal.ajax
```

slide_down/slide_down.libraries.yml

In Drupal 8 custom JavaScript files must be added to an asset library to be able to be included on a page.

Create PHP Class

```php
namespace Drupal\slide_down\Ajax;
use Drupal\Core\Ajax\CommandInterface;

class SlideDownCommand implements CommandInterface {
  // ...
  // Constructs an SlideDownCommand object.
  public function __construct($selector, $duration = NULL) {
    $this->selector = $selector;
    $this->duration = $duration;
  }

  // Implements Drupal\Core\Ajax\CommandInterface:render().
  public function render() {
    return array(
      'command' => 'slideDown',
      'method' => NULL,
      'selector' => $this->selector,
      'duration' => $this->duration,
    );
  }
}
```

slide_down/src/Ajax/SlideDownCommand.php

Create a PHP class that implements CommandInterface. Must define a 'render' method and return an associative array. In the array, pass the element with key of 'command' and value being the name of the JavaScript function and any repsonse data.

To Create a Callback Command:

- *Create a module*
- *Attach JavaScript function to 'Drupal.AjaxCommands.prototype'*
- *Define an asset library*
- *Create PHP class that implements 'CommandInterface'*

Using Callback Commands

Example Scenario

Load watchdog log message details onto the overview page using Ajax commands.

Add Ajax Library to Page

```php
use \Drupal\dblog\Controller\DbLogController as ControllerBase;

class DbLogController extends ControllerBase {
  // Override overview() method.
  public function overview() {
    $build = parent::overview();
    // ...
    // Add custom library.
    $build['#attached']['library'][] = 'ajax_dblog/ajax-dblog';
    return $build;
  }
  // ...
}
```

ajax_dblog/src/Controller/DbLogController.php

Need to attach custom library onto page so that custom JavaScript and Ajax Framework is included.

```yaml
ajax-dblog:
  version: VERSION
  css:
    component:
      css/ajax_dblog.module.css: {}
  js:
    js/behaviors.js: {}
  dependencies:
    - slide_down/slidedown
```

ajax_dblog/ajax_dblog.libraries.yml

```yaml
slidedown:
  version: VERSION
  js:
    js/slidedown-command.js: {}
  dependencies:
    - core/drupal.ajax
```

slide_down/slide_down.libraries.yml

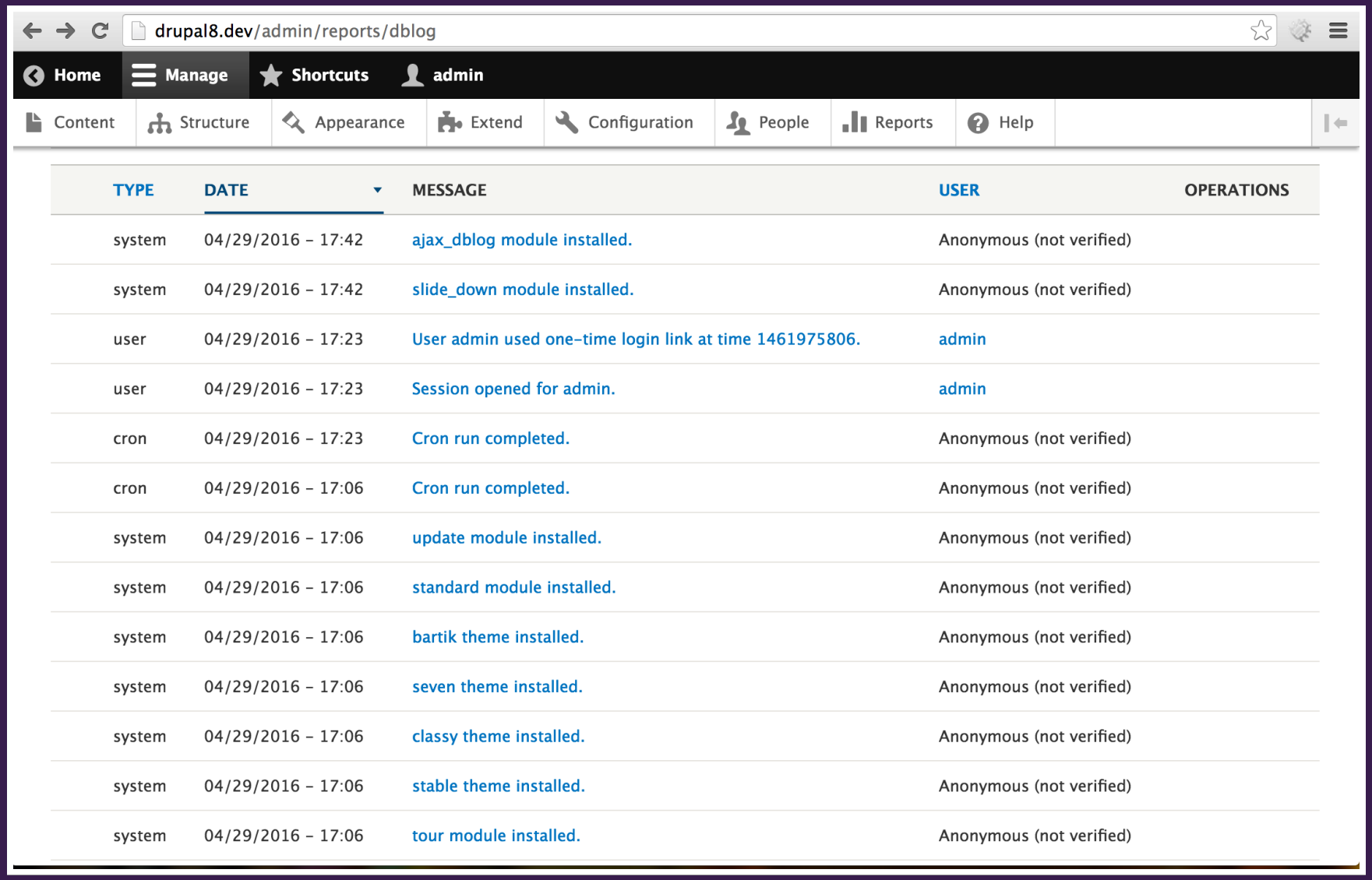
Defining a dependency in the library on another library. The other library depends on the Ajax Framework. Drupal will follow chain to include all depended JavaScript files.

Add Ajax to Elements

```
01 namespace Drupal\ajax_dblog\Controller;
02 use \Drupal\dblog\Controller\DbLogController as ControllerBase;
03
04 class DbLogController extends ControllerBase {
05   // Override overview() method.
06   public function overview() {
07     $build = parent::overview();
08     // Alter the links for each log message.
09     foreach ($build['dblog_table']['#rows'] as &$row) {
10       // ...
11       // Build route parameters.
12       $params = array(
13         'method' => 'nojs',
14         //...
15       );
16       // Build link options.
17       $ops = array( 'attributes' => array(
18         'class' => array('use-ajax', 'dblog-event-link'),
19       ));
20       // Replace with a new link.
21       $row['data'][3] = Link::createFromRoute($txt,'ajax_dblog.event',$params,$ops
22     }
```

ajax_dblogs/src/Controller/DbLogController.php

Need to have elements that will trigger an Ajax request. Rebuilding links on page to point to new route (line #21). Links will have the class 'use-ajax' (line #18), which the Ajax Framework will look for.



Page still renders the same. However now includes Ajax Framework including the custom SlideDown command. The message title links will now trigger an ajax request.

Create Ajax Request Endpoint

```
01 ajax_dblog.event:
02   path: '/admin/reports/dblog/{method}/event/{event_id}'
03   defaults:
04     _controller: '\Drupal\ajax_dblog\Controller\DbLogController::ajaxEventDetails'
05   requirements:
06     _permission: 'access site reports'
07     method: 'nojs|ajax'
```

ajax_dblog/ajax_dblog.routing.yml

/admin/reports/dblog/nojs/event/123

/admin/reports/dblog/ajax/event/123

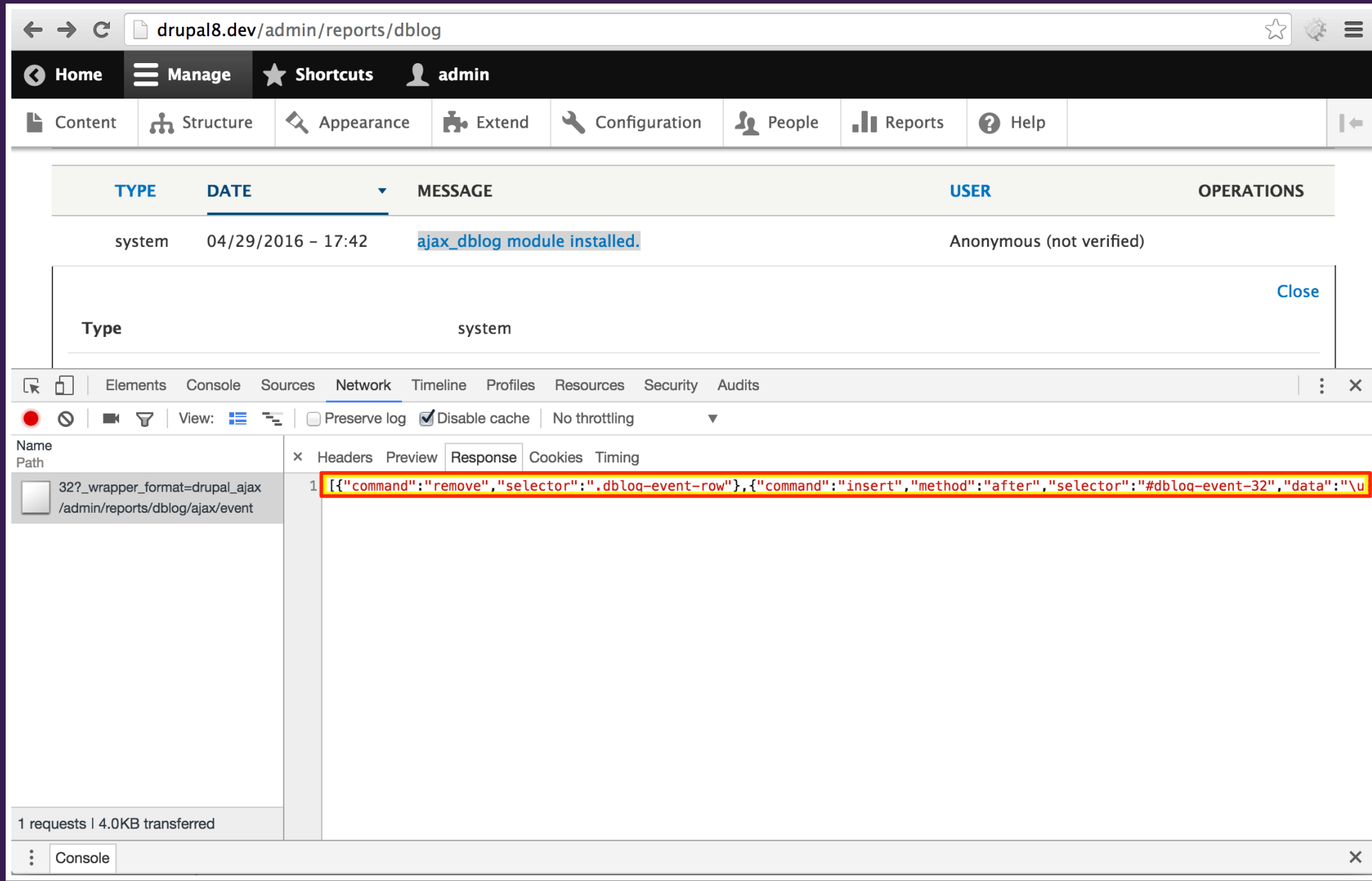
Create an endpoint that will handle Ajax Requests. The ajax framework will replace 'nojs' with 'ajax' on all request. Can use as a check to handle graceful degradation.

Return an AjaxResponse of Callback Commands

```
01 use Drupal\Core\Ajax\AjaxResponse;
02 use Drupal\Core\Ajax\AfterCommand;
03 use Drupal\Core\Ajax\RemoveCommand;
04 use Drupal\slide_down\Ajax\SlideDownCommand;
05
06 class DbLogController extends ControllerBase {
07   // ...
08   public function ajaxEventDetails($method, $event_id) {
09     //...
10     if ($method == 'ajax') {
11       $event = parent::eventDetails($event_id);
12       $event_details = [ ... ];
13       // Create an AjaxResponse.
14       $response = new AjaxResponse();
15       // Remove old event details.
16       $response->addCommand(new RemoveCommand('.dblog-event-row'));
17       // Insert event details after event.
18       $response->addCommand(new AfterCommand('#dblog-event-' . $event_id, $event_d
19       // SlideDown event details.
20       $response->addCommand(new SlideDownCommand('#dblog-event-details-' . $event_
21     }
22     // ...
```

ajax_dblog/src/Controller/DbLogController.php

Have a method that is the endpoint for the Ajax request (line #8). Need to build an AjaxReponse object (line #14). Will add commands to this response using the 'addCommand' method and creating a new instance of the relevant Callback Command class (lines #16, #18, #20).



When a message title is clicked, the Ajax request is made. The endpoint builds an AjaxResponse of commands and Drupal returns a JSON string.

Ajax Response

```
[
  {
    "command":"remove",
    "selector":".dblog-event-row"
  },
  {
    "command":"insert",
    "method":"after",
    "selector":"#dblog-event-32",
    "data":"...",
    "settings":null
  },
  {
    "command":"slideDown",
    "method":null,
    "selector":"#dblog-event-details-32",
    "duration":null
  }
]
```

The returned JSON array is parsed by Ajax Framework. Finds JavaScript function to execute and the passes the object as the data for the response argument of the function.

To Use Callback Commands

- *Include the Ajax library and commands on the page.*
- *Have endpoint that returns an AjaxResponse*
- *Add commands to response using 'addCommand'*

Resources

Drupal 8 Ajax Framework: bit.ly/Drupal8Ajax

This Presentation: bit.ly/Con16Ajax

Presentation Slides: bit.ly/Con16AjaxSlides

Example Code: bit.ly/Con16AjaxCode

Creating Commands in D8: bit.ly/D8AjaxCmds

My Blog: mike-miles.com

Feedback

@mikemiles86

Thank You!