

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

```
01 (function ($, window, Drupal, drupalSettings) {
02   'use strict';
03   /**
04    * [commandName description]
05    *
06    * @param {Drupal.Ajax} [ajax]
07    * @param {object} response
08    * @param {number} [status]
09    */
10   Drupal.AjaxCommands.prototype.[commandName] = function(ajax, response, status){
11
12     // Custom javascript goes here ...
13
14   }
15
16 })(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

```
01 namespace Drupal\[module]\Ajax
02 use Drupal\Core\Ajax\CommandInterface;
03
04 // An Ajax command for calling [commandName]() JavaScript method.
05 class [CommandName]Command implements CommandInterface {
06
07     // Implements Drupal\Core\Ajax\CommandInterface:render().
08     public function render() {
09         return array(
10             'command' => '[commandName]', // Name of JavaScript Method.
11             // other response arguments...
12         );
13     }
14 }
```

`[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

```
01 Drupal.AjaxCommands.prototype = {
02   // ...
03   /**
04    * Command to remove a chunk from the page.
05    *
06    * @param {Drupal.Ajax} [ajax]
07    * @param {object} response
08    * @param {string} response.selector
09    * @param {object} [response.settings]
10    * @param {number} [status]
11    */
12   remove: function (ajax, response, status) {
13     var settings = response.settings || ajax.settings || drupalSettings;
14     $(response.selector).each(function () {
15       Drupal.detachBehaviors(this, settings);
16     })
17     .remove();
18   },
19   //...
```

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

```
01 namespace Drupal\Core\Ajax;
02 use Drupal\Core\Ajax\CommandInterface;
03 /**
04  * Ajax command for calling the jQuery remove() method.
05  * ...
06  */
07 class RemoveCommand Implements CommandInterface {
08     // ...
09     /**
10      * Implements Drupal\Core\Ajax\CommandInterface:render().
11      */
12     public function render() {
13         return array(
14             'command' => 'remove',
15             'selector' => $this->selector,
16         );
17     }
18 }
```

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 halves 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

```
01 name: 'Slide Down Command'  
02 type: module  
03 description: Provides an Ajax Callback command for the jQuery SlideDown method.  
04 package: other  
05 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

```
01 slidedown:  
02   version: VERSION  
03   js:  
04     js/slidedown-command.js; {}  
05   dependencies:  
06     - 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

```
01 namespace Drupal\slide_down\Ajax;
02 use Drupal\Core\Ajax\CommandInterface;
03
04 class SlideDownCommand implements CommandInterface {
05     // ...
06     // Constructs an SlideDownCommand object.
07     public function __construct($selector, $duration = NULL) {
08         $this->selector = $selector;
09         $this->duration = $duration;
10     }
11
12     // Implements Drupal\Core\Ajax\CommandInterface:render().
13     public function render() {
14         return array(
15             'command' => 'slideDown',
16             'method' => NULL,
17             'selector' => $this->selector,
18             'duration' => $this->duration,
19         );
20     }
21 }
```

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 response 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

```
01 use \Drupal\dblog\Controller\DbLogController as ControllerBase;
02
03 class DbLogController extends ControllerBase {
04     // Override overview() method.
05     public function overview() {
06         $build = parent::overview();
07         // ...
08         // Add custom library.
09         $build['#attached']['library'][] = 'ajax_dblog/ajax-dblog';
10         return $build;
11     }
12     // ...
13 }
```

ajax_dblog/src/Controller/DbLogController.php

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

```
01 ajax-dblog:
02   version: VERSION
03   css:
04     component:
05       css/ajax_dblog.module.css: {}
06   js:
07     js/behaviors.js: {}
08   dependencies:
09     - slide_down/slidedown
```

ajax_dblog/ajax_dblog.libraries.yml

```
01 slidedown:
02   version: VERSION
03   js:
04     js/slidedown-command.js: {}
05   dependencies:
06     - 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.

drupal8.dev/admin/reports/dblog

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

TYPE	DATE	MESSAGE	USER	OPERATIONS
system	04/29/2016 - 17:42	ajax_dblog module installed.	Anonymous (not verified)	
system	04/29/2016 - 17:42	slide_down module installed.	Anonymous (not verified)	
user	04/29/2016 - 17:23	User admin used one-time login link at time 1461975806.	admin	
user	04/29/2016 - 17:23	Session opened for admin.	admin	
cron	04/29/2016 - 17:23	Cron run completed.	Anonymous (not verified)	
cron	04/29/2016 - 17:06	Cron run completed.	Anonymous (not verified)	
system	04/29/2016 - 17:06	update module installed.	Anonymous (not verified)	
system	04/29/2016 - 17:06	standard module installed.	Anonymous (not verified)	
system	04/29/2016 - 17:06	bartik theme installed.	Anonymous (not verified)	
system	04/29/2016 - 17:06	seven theme installed.	Anonymous (not verified)	
system	04/29/2016 - 17:06	classy theme installed.	Anonymous (not verified)	
system	04/29/2016 - 17:06	stable theme installed.	Anonymous (not verified)	
system	04/29/2016 - 17:06	tour module installed.	Anonymous (not verified)	

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 AjaxResponse 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).

The screenshot shows a Drupal 8 admin interface at `drupal8.dev/admin/reports/dblog`. The top navigation bar includes links for Home, Manage, Shortcuts, and the user 'admin'. Below this is a secondary navigation bar with categories like Content, Structure, Appearance, Extend, Configuration, People, Reports, and Help.

The main content area displays a table of database log entries. The table has columns for TYPE, DATE, MESSAGE, USER, and OPERATIONS. One entry is visible: a 'system' message from '04/29/2016 - 17:42' with the message 'ajax_dblog module installed.' and user 'Anonymous (not verified)'. A 'Close' button is located to the right of the message.

Below the table, a modal window shows the 'Type' as 'system'. At the bottom of the page, the browser's developer tools are open to the 'Network' tab. A single request is shown with the path `32?_wrapper_format=drupal_ajax/admin/reports/dblog/ajax/event`. The 'Response' tab is selected, showing a JSON string: `[{"command":"remove","selector":".dblog-event-row"}, {"command":"insert","method":"after","selector":"#dblog-event-32","data":"\u0026"}]`. The JSON string is highlighted with a red border.

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

```
01 [
02   {
03     "command": "remove",
04     "selector": ".dblog-event-row"
05   },
06   {
07     "command": "insert",
08     "method": "after",
09     "selector": "#dblog-event-32",
10     "data": "...",
11     "settings": null
12   },
13   {
14     "command": "slideDown",
15     "method": null,
16     "selector": "#dblog-event-details-32",
17     "duration": null
18   }
19 ]
```

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!



#DrupalCon



@WeAreGenuine D8 Ajax Commands / Michael Miles