

M2 CPFED

4. Javascript - RequireJS / jQuery

John Hughes

Head of Magento - Fisheye Media Ltd.

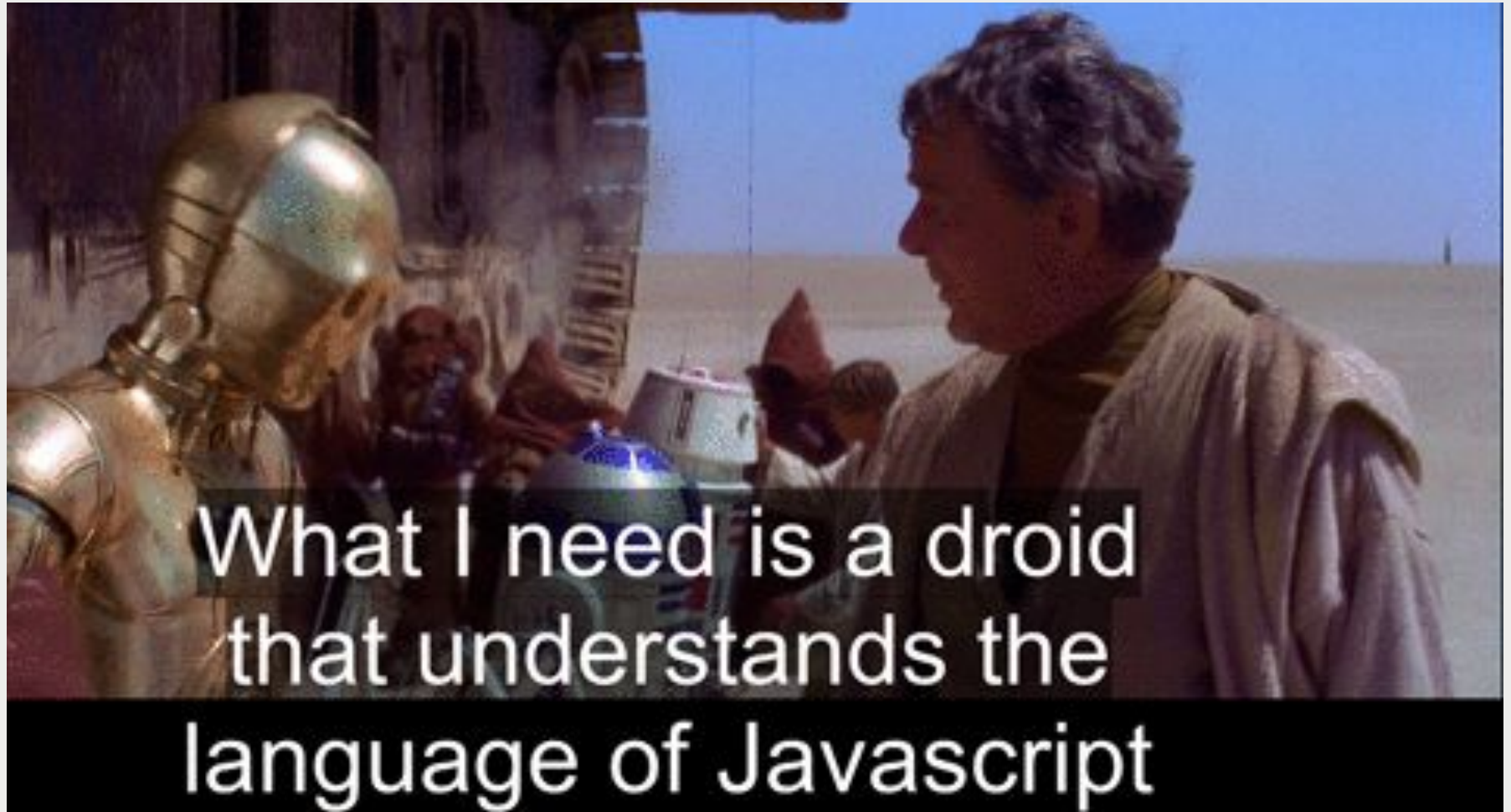


PROFESSIONAL
SOLUTIONS PARTNER

:fisheye

Overview.

- Javascript basics
- RequireJS
- jQuery
- Overrides and mixins
- Minification, merging and bundling



Javascript basics

Javascript basics.

- **How can JS be included on a page via layout?**
- How can JS be deferred / loaded async?
- What are the pros / cons of inline JS?

default_head_blocks.xml

```
<?xml version="1.0"?>
<page xmlns:xsi="..." xsi:noNamespaceSchemaLocation="...">
  <head>
    <script src="js/my-awesome-scriptz.js"/>
    <link src="Vendor_Module::js/my-epic-jquery.js"/>
  </head>
</page>
```

Javascript basics.

- How can JS be included on a page via layout?
- **How can JS be deferred / loaded async?**
- What are the pros / cons of inline JS?

default_head_blocks.xml

```
<?xml version="1.0"?>
<page xmlns:xsi="..." xsi:noNamespaceSchemaLocation="...">
  <head>
    <script src="js/my-script.js" async="true"/>
  </head>
</page>
```

default_head_blocks.xml

```
<?xml version="1.0"?>
<page xmlns:xsi="..." xsi:noNamespaceSchemaLocation="...">
  <head>
    <script src="js/my-script.js" defer="defer"/>
  </head>
</page>
```

Javascript basics.

How can JS be deferred / loaded async?

- Use RequireJS

Javascript basics.

- How can JS be included on a page via layout?
- How can JS be deferred / loaded async?
- **What are the pros / cons of inline JS?**

Javascript basics.

Pros	Cons
Easy to include	Not easily reusable
No additional requests	Can block page rendering (not deferred)
	Harder to manage
	Can't use Content-Security-Policy
	Higher risk of XSS exploit



PROFESSIONAL
SOLUTIONS PARTNER

fisheye

RequireJS

RequireJS intro.

- **What is RequireJS?**
- How do you include RequireJS files?
- How do you define a RequireJS module?
- How do you define RequireJS dependencies?

RequireJS intro.

What is RequireJS?

- A JavaScript file and module loader
- Based on **AMD** (**A**synchronous **M**odule **D**efinition)
- Encapsulated JS modules (components)
 - i.e. not globally accessible
- Allows for dependencies on other modules

RequireJS intro.

- What is RequireJS?
- **How do you include RequireJS files?**
- How do you define a RequireJS module?
- How do you define RequireJS dependencies?

RequireJS intro.

How do you include RequireJS files?

- Generally within **.phtml** template files
 - Using the **data-mage-init** attribute
 - Or **x-magento-init** script type

script-example.phtml

```
<div data-mage-init='{ "Magento_Theme/js/path/to/script":{}} '>
```

```
...
```

```
</div>
```

script-example.phtml

```
<div data-mage-init='{ "Magento_Theme/js/path/to/script":{}}'>  
  ...  
</div>
```



Resolves to: **'web/js/path/to/script.js'** within **Magento_Theme**
(whether directly in the module or a theme)

script-example.phtml

```
<div id="some-element">
```

```
    <p><?= __('Script example') ?></p>
```

```
</div>
```

```
<script type="text/x-magento-init">
```

```
{
```

```
    "#some-element": {
```

```
        "Magento_Theme/js/path/to/script": {}
```

```
    }
```

```
}
```

```
</script>
```

RequireJS intro.

- What is RequireJS?
- How do you include RequireJS files?
- **How do you define a RequireJS module?**
- How do you define RequireJS dependencies?

web/js/path/to/script.js

```
define([], function() {  
  
    return function(config, element) {  
  
        console.log({  
            element: element,  
            config: config  
        });  
    }  
});
```

RequireJS intro.

- What is RequireJS?
- How do you include RequireJS files?
- How do you define a RequireJS module?
- **How do you define RequireJS dependencies?**

web/js/path/to/script2.js

```
define([  
    'jquery',  
    'Magento_Theme/js/path/to/script'  
], function($) {  
  
    return function(config, element) {  
        ...  
    }  
});
```

requirejs-config.js

- **What is the purpose of requirejs-config.js?**
- Where are requirejs-config.js files found?
- What are aliases and map / paths?
- How can you debug / resolve an alias?
- What are deps and shims for?
- How do you regenerate requirejs-config.js?

requirejs-config.js

What is the purpose of requirejs-config.js?

- To set RequireJS configuration, such as:
 - Mapping aliases to actual files
 - Setting global dependencies
 - Handling non AMD file inclusions / dependencies
 - Declaring mixins

requirejs-config.js

- What is the purpose of requirejs-config.js?
- **Where are requirejs-config.js files found?**
- What are aliases and map / paths?
- How can you debug / resolve an alias?
- What are deps and shims for?
- How do you regenerate requirejs-config.js?

requirejs-config.js

Where are requirejs-config.js files found?

- Modules
 - **view/<area>/requirejs-config.js**
- Themes
 - **<Vendor_Module>/requirejs-config.js**
 - **web/requirejs-config.js**

requirejs-config.js

- What is the purpose of requirejs-config.js?
- Where are requirejs-config.js files found?
- **What are aliases and map / paths?**
- How can you debug / resolve an alias?
- What are deps and shims for?
- How do you regenerate requirejs-config.js?

requirejs-config.js

What are aliases and map / paths?

- Aliases allow creating shorthand names for AMD modules
- These can be set using **map** or **paths**
 - **map** allows for 'prefix mapping'
 - **paths** only allows for exact matches

requirejs-config.js

```
var config = {  
    'map': {  
        '*': {  
            'theme/script': 'Magento_Theme/js/path/to/script'  
        }  
    },  
    'paths': {  
        'theme/script2': 'Magento_Theme/js/path/to/script2'  
    }  
};
```

script-example.phtml

```
<script type="text/x-magento-init">
    {
        "#some-element": {
            "theme/script": {}
        }
    }
</script>
```

requirejs-config.js

- What is the purpose of requirejs-config.js?
- Where are requirejs-config.js files found?
- What are aliases and map / paths?
- **How can you debug / resolve an alias?**
- What are deps and shims for?
- How do you regenerate requirejs-config.js?

requirejs-config.js

How can you debug / resolve an alias?

- Use **grep** on CLI
- Use your IDE to find the alias and filter search to files named '**requirejs-config.js**'

requirejs-config.js

- What is the purpose of requirejs-config.js?
- Where are requirejs-config.js files found?
- What are aliases and map / paths?
- How can you debug / resolve an alias?
- **What are deps and shims for?**
- How do you regenerate requirejs-config.js?

requirejs-config.js

What are deps and shims for?

- **deps** - globally include a JS file
- **shims** - add support for non AMD JS files
 - Dependencies can be declared for these files

requirejs-config.js

```
var config = {  
    'deps': [  
        'Magento_Theme/js/global-script',  
        'Magento_Theme/js/path/to/nonAmdScript'  
    ],  
    'shim': {  
        'Magento_Theme/js/path/to/nonAmdScript': {  
            'deps': ['jquery']  
        }  
    }  
};
```

requirejs-config.js

- What is the purpose of requirejs-config.js?
- Where are requirejs-config.js files found?
- What are aliases and map / paths?
- How can you debug / resolve an alias?
- What are deps and shims for?
- **How do you regenerate requirejs-config.js?**

requirejs-config.js

How do you regenerate requirejs-config.js?

- **developer** mode - auto regenerated on each page load
 - Except when FPC enabled and page in cache
- **default** / **production** mode - regenerated if doesn't exist

Passing data to JS files.

- **How can config be passed to JS files?**
- What is the standard way to pass config from layout XML?

Passing data to JS files.

How can config be passed to JS files?

- Via JSON in the **data-mage-init** / **x-magento-init** declaration

script-example.phtml

```
<div data-mage-init='{ "Magento_Theme/js/path/to/script":  
    {"someKey": "value"}  
'>  
    ...  
</div>
```

script-example.phtml

```
<script type="text/x-magento-init">
    {
        "#some-element": {
            "Magento_Theme/js/path/to/script": {
                "someKey": "value"
            }
        }
    }
</script>
```

Passing data to JS files.

- How can config be passed to JS files?
- **What is the standard way to pass config from layout XML?**

Passing data to JS files.

What is the standard way to pass config from layout XML?

- Using the **jsLayout** argument

default.xml

```
<block name="js.layout.example"
      template="Magento_Theme::js-layout-example.phtml">
  <arguments>
    <argument name="jsLayout" xsi:type="array">
      <item name="some_string"
            translate="true"
            xsi:type="string">Some String</item>
      <item name="some_number"
            xsi:type="number">42</item>
    </argument>
  </arguments>
</block>
```

js-layout-example.phtml

```
<script type="text/x-magento-init">
{
    "#js-layout-element": {
        "theme/jsLayoutExample":
            <?= $block->getJsLayout() ?>
    }
}
</script>
```

Magento\Framework\View\Element\AbstractBlock

```
/**  
 * Retrieve serialized JS layout configuration  
 * ready to use in template  
 *  
 * @return string  
 */  
public function getJsLayout()  
{  
    return json_encode($this->jsLayout);  
}
```

jQuery

jQuery.

- **What is jQuery / jQuery UI?**
- What are widgets and how are they used in M2?
- How can widgets be instantiated & methods called?
- How do you create widgets / use common methods?
- How can you extend / customise widgets?

jQuery.

What is jQuery / jQuery UI?

- **jQuery** is library that provides shortcut functions to simplify common javascript needs and aids with cross browser compatibility
- **jQuery UI** provide reusable UI components known as **widgets** for common functionality such as menus, tabs etc.

jQuery.

- What is jQuery / jQuery UI?
- **What are widgets and how are they used in M2?**
- How can widgets be instantiated & methods called?
- How do you create widgets / use common methods?
- How can you extend / customise widgets?

jQuery.

What are widgets and how are they used in M2?

- In M2 terminology they are custom built **jQuery UI widgets** wrapped in **RequireJS** by Magento for common use cases
- These include: **modal**, **accordion**, **quicksearch** and more

jQuery.

What are widgets and how are they used in M2?

- **jQuery UI widgets** are included / configured just like any other **RequireJS** file using **data-mage-init** / **x-magento-init**
- <https://devdocs.magento.com/guides/v2.2/javascript-dev-guide/widgets/jquery-widgets-about.html>

jQuery.

- What is jQuery / jQuery UI?
- What are widgets and how are they used in M2?
- **How can widgets be instantiated & methods called?**
- How do you create widgets / use common methods?
- How can you extend / customise widgets?

jQuery.

How can widgets be instantiated & methods called?

- Using **require()** or passing as dependency in AMD module
- Using **data-mage-init** / **x-magento-init**
 - Not supported by all widgets (e.g. modal)

accordion-example.phtml

```
<div id="accordion-example">
    <div data-role="collapsible">
        <div data-role="title">
            <?= $block->escapeHtml('Accordion Title') ?>
        </div>
        <div data-role="content">
            <p><?= $block->escapeHtml('Accordion content...') ?></p>
        </div>
    </div>
    ...
</div>
```

accordion-example.phtml

data-role attributes
determine the role of
elements for the accordion

```
<div id="accordion-example">
  <div data-role="collapsible">
    <div data-role="title">
      <?= $block->escapeHtml('Accordion Title') ?>
    </div>
    <div data-role="content">
      <p><?= $block->escapeHtml('Accordion content...') ?></p>
    </div>
  </div>
  ...
</div>
```

accordion-example.phtml

```
<script type="text/x-magento-init">
```

```
{
```

```
    "#accordion-element":
```

```
        {"accordion": {
```

```
            /* Config settings here */
```

```
        }
```

```
    }
```

```
}
```

```
</script>
```

accordion is an alias for
mage/accordion (found in lib/web)

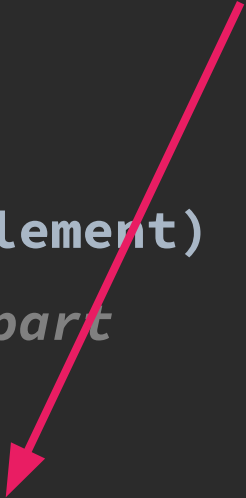
modal-example.js

```
require([  
    'jquery',  
    'Magento_Ui/js/modal/modal'  
], function(modal) {  
    /* Implementation, without  
    defining a new AMD module */  
    $('some-element').modal();  
});
```

modal-example.js

```
define([  
    'jquery',  
    'Magento_Ui/js/modal/modal'  
], function(modal) {  
    return function (config, element) {  
        /* Implementation, as part  
        of a new AMD module */  
        element.modal(config);  
    }  
});
```

Modal can be assigned to
element and config passed
from template / layout



jQuery.

How can widgets be instantiated & methods called?

- Methods can be called by passing method name as an argument to the widget

modal-example.js

```
define([
    'jquery',
    'Magento_Ui/js/modal/modal'
], function(modal) {
    return function (config, element) {
        /* Implementation, as part
        of a new AMD module */
        element.modal(config);
        element.modal('openModal');
    }
});
```

jQuery.

- What is jQuery / jQuery UI?
- What are widgets and how are they used in M2?
- How can widgets be instantiated & methods called?
- **How do you create widgets / use common methods?**
- How can you extend / customise widgets?

widget-example.js

```
define([
    'jquery'
], function($) {
    'use strict';
    $.widget('fisheyeAcademy.example', {
        options: {...},
        _init: function () {
            /* Init widget here - do something */
        },
    });
    return $.fisheyeAcademy.example;
});
```

requirejs-config.js

```
var config = {  
    'map': {  
        '*': {  
            'fisheyeAcademy/widget-example':  
                'Magento_Theme/js/widget-example'  
        }  
    }  
};
```

widget-example.phtml

```
<div data-mage-init='{ "fisheyeAcademy/widget-example": {} }'>
    <div data-role="content">
        <p><?= __('Some content') ?></p>
    </div>
    <button data-role="showContent">
        <?= __('Show') ?>
    </button>
    <button data-role="hideContent">
        <?= __('Hide') ?>
    </button>
</div>
```

widget-example.phtml

```
<div data-mage-init='{ "fisheyeAcademy/widget-example": {} }' >
    <div data-role="content">
        <p><?= __('Some content') ?></p>
    </div>
    <button data-role="showContent">
        <?= __('Show') ?>
    </button>
    <button data-role="hideContent">
        <?= __('Hide') ?>
    </button>
</div>
```

jQuery.

Method	Usage
<code>_init()</code>	Called on widget initialisation and any further time widget is called Should be used for default functionality that needs to run each time
<code>_create()</code>	Called when widget binded to element, should be used to bind events. Only ever called once per instance
<code>_destroy()</code>	Runs when call <code>.destroy()</code> on widget To replace / remove widget to avoid conflicts / clean up

jQuery.

- What is jQuery / jQuery UI?
- What are widgets and how are they used in M2?
- How can widgets be instantiated & methods called?
- How do you create widgets / use common methods?
- **How can you extend / customise widgets?**

jQuery.

How can you extend / customise widgets?

- Widgets can be extended (inherited) using jQuery UI
- Widgets can be overridden using RequireJS path / map
 - Can also be used in conjunction with an extend
- Use a mixin to override specific methods / settings only

Mixins

jQuery.

- **What are mixins and what are they for?**
- How do you declare a mixin?
- How do you extend / override target file methods?

jQuery.

What are mixins and what are they for?

- Mixins allow you to modify the functionality of JS component without extending or replacing/overriding it
- A component can have multiple mixins applied to it unlike if you were to extend or replace/override it

jQuery.

- What are mixins and what are they for?
- **How do you declare a mixin?**
- How do you extend / override target file methods?

requirejs-config.js

```
var config = {  
  config: {  
    mixins: {  
      'image/accordion': {  
        'Vendor_Module/js/accordion-mixin': true  
      }  
    }  
  }  
};
```

jQuery.

- What are mixins and what are they for?
- How do you declare a mixin?
- **How do you extend / override target file methods?**

accordion-mixin.js

```
define([
    'jquery',
], function ($) {
    'use strict';
    return function (widget) {
        $.widget('fisheyeAcademy.accordion', widget, {
            options: {
                active: [1] // Override option defaults
            },
            _create: function () { // Override method
                this._super(); // Call parent method
                // Do something here...
            }
        });
        return $.fisheyeAcademy.accordion;
    };
});
```

Wrap up / resources

Code challenge 1.

- Add JS files to the page via **<head/>** in layout XML
- Create a RequireJS config file and add global files using dep (AMD module) and shim (non AMD module, with dependency)

Code challenge 2.

- Add a template via layout and use **data-mage-init** / **x-magento-init** to include a simple JS file that logs or alerts data
- Add an alias in a RequireJS config file and update the file call in the template
- Pass config data to your JS file via the template, then by layout using jsLayout

Code challenge 3.

- Create a jQuery widget to perform a simple interaction on the page
- Set default options and use data-role attributes to set default properties / roles to elements
- Create a mixin for your widget to modify it's functionality

Credit / Resources / Thanks.

- **Official Magento Study Guide**
- **Swift Otter Study Guide**
- **Magento DevDocs**