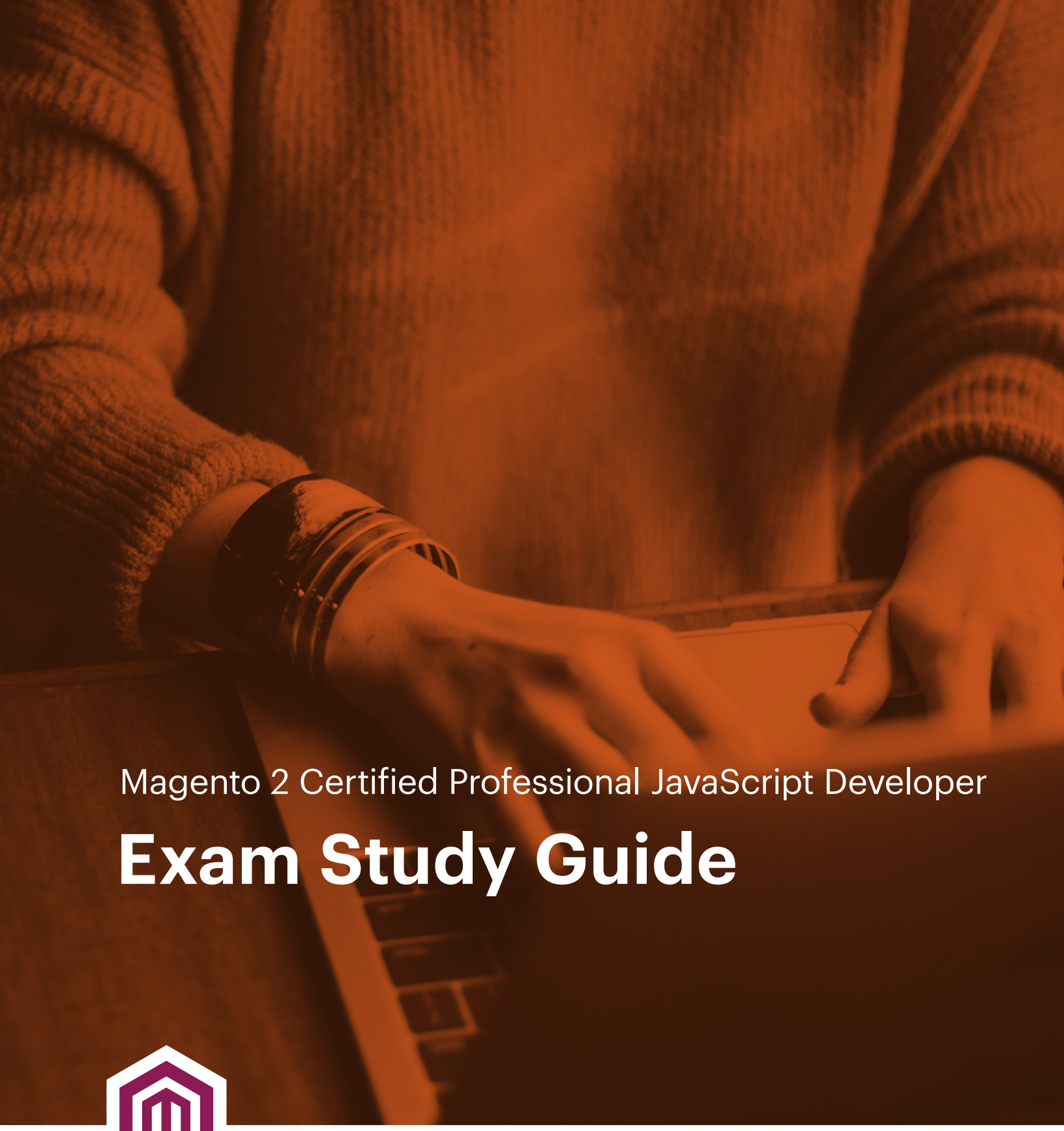

Magento 2 Certified Professional JavaScript Developer

Exam Study Guide

MAGENTO 2 CERTIFIED
Professional JavaScript Developer

Magento® An Adobe Company | U

Magento® U

training@magento.com

u.magento.com

u.magento.com/certification

Contents

Introduction **1**

Topics and Objectives **2**

1 Technology Stack **2**

1.1 Demonstrate understanding of RequireJS 2

1.2 Demonstrate understanding of UnderscoreJS 2

1.3 Demonstrate understanding of jQuery UI widgets 2

1.4 Demonstrate understanding of KnockoutJS 3

2 Magento JavaScript Basics **3**

2.1 Demonstrate understanding of the modular structure of Magento 3

2.2 Describe how to use JavaScript modules in Magento 3

2.3 Demonstrate ability to execute JavaScript modules 3

2.4 Describe jQuery UI widgets in Magento 4

2.5 Demonstrate ability to customize JavaScript modules 4

3 Magento Core JavaScript Library **4**

3.1 Demonstrate understanding of the mage library 4

3.2 Demonstrate understanding of mage widgets 5

3.3 Demonstrate the ability to use the customer-data module 5

4 UI Components **5**

4.1 Demonstrate understanding of Knockout customizations 5

4.2 Demonstrate understanding of Magento UI components 6

4.3 Demonstrate the ability to use UI components 7

4.4 Demonstrate understanding of grids and forms 7

5 Checkout **7**

5.1 Demonstrate understanding of checkout architecture 7

5.2 Demonstrate understanding of payments 8

Example Questions **9**

Question 1 9

Question 2 9

Question 3 10

Answer Key **12**

Introduction

This exam will validate the skills and knowledge needed to develop new JavaScript modules and customize existing ones. It will verify whether the developer understands the core Magento JavaScript framework, is able to use its components in the correct way, and understands best practices for customizing existing components and adding new ones. The exam pays particular attention to UI components, and will validate whether the developer understands their purpose, area of application, architecture, and lifecycle.

This includes not only the in-browser JavaScript framework, but also the related server-side configuration for layout, customer data sections, and adminhtml UI components.

This exam is for a JavaScript developer with broad experience in customizing Magento JavaScript for at least 1 year.

The test is built for the 2.2.*x* version of Magento Commerce and Magento Open Source.

Supporting **Magento U** course:

- **JavaScript Development in Magento 2 (Instructor-Led)**

This exam consists primarily of scenario-based questions in a multiple-choice format. Sample questions are included at the end of this guide.

Test time: 90 minutes. Passing score: 63% or above, 60 questions. Available to take remotely or at a test center.

Exam topics and the percentage covered in the test.

1. **Technology Stack** 20%
2. **Magento JavaScript Basics** 25%
3. **Magento Core JavaScript Library** 19%
4. **UI Components** 24%
5. **Checkout** 12%

Topics and Objectives

1 Technology Stack

1.1 Demonstrate understanding of RequireJS

Describe the RequireJS framework and its approach to JavaScript module organization

- What is the main purpose of the RequireJS framework?
- What are the pros and cons of the AMD approach to JavaScript file organization?
- Which capabilities does RequireJS provide to create and customize JavaScript modules?

Demonstrate the ability to use requirejs-config.js files in the development process

- What is a requirejs-config.js file?
- In which cases it is necessary to add configurations to it?
- What tools does it provide?
- What are global callbacks?
- How can mappings be used?

  https://requirejs.org/docs/api.html#config-callback

Demonstrate the ability to use RequireJS plugins

- What are RequireJS plugins?
- What are the text and domReady plugins used for?

1.2 Demonstrate understanding of UnderscoreJS

Demonstrate understanding of underscore utility functions

- What are the benefits of using the underscore library versus native JavaScript?

Use underscore templates in customizations

- Describe how underscore templates are used.
- What are the pros and cons of using underscore templates?
- Describe how underscore templates are used in Magento together with the text RequireJS plugin.

1.3 Demonstrate understanding of jQuery UI widgets

Demonstrate understanding of the jQuery framework and its role in the Magento JavaScript framework

- What is a jQuery library?
- What different components does it have (jQuery UI, jQuery events and so on)?
- How does Magento use it?

1.4 Demonstrate understanding of KnockoutJS

Describe key KnockoutJS concepts

- Describe the architecture of the Knockout library: MVVC concept, observables, bindings.

Demonstrate understanding of knockout templates

- What is the main concept of knockout templates?
- What are the pros and cons of knockout templates?
- Compare knockout templates with underscore JavaScript templates.

Demonstrate understanding of the knockout-es5 library

2 Magento JavaScript Basics

2.1 Demonstrate understanding of the modular structure of Magento

Demonstrate understanding of the JavaScript file organization in Magento

- What file structure is used to organize JavaScript modules?
- Where does Magento locate a module's JavaScript file?
- Where does Magento store the JavaScript library?

Describe how static content is organized in Magento

- How does Magento expose a module's static content to the web requests?
- What are the different ways to deploy static content?

2.2 Describe how to use JavaScript modules in Magento

Use requirejs-config.js files to create JavaScript customizations

- How do you ensure that a module will be executed before other modules?
- How can an alias for a module be declared?
- What is the purpose of requirejs-config.js callbacks?

Describe different types of Magento JavaScript modules

- Plain modules
- jQuery UI widgets
- UiComponents

2.3 Demonstrate ability to execute JavaScript modules

Demonstrate the ability to use the data-mage-init attribute to run JavaScript modules

- What is the purpose and syntax of the data-mage-init attribute?
- How is it used to execute JavaScript modules?

Demonstrate the ability to use text/x-magento-init scripts to execute JavaScript modules

- What is the purpose and syntax of the text/x-magento-init script tag?
- What is the difference between the text/x-magento-init and the data-mage-init methods of JavaScript module execution?

Describe advanced methods of executing JavaScript modules

- How do you execute a JavaScript module in an AJAX response and in dynamic code stored in a string?

2.4 Describe jQuery UI widgets in Magento

Describe how Magento uses jQuery widgets, and demonstrate an understanding of the $.mage object

- What is the role of jQuery widgets in Magento?
- What are typical widgets?
- How are Magento jQuery widget modules structured?

Describe how Magento executes jQuery widgets

- How are Magento jQuery widgets executed with data-mage-init and text/x-magento-init?

2.5 Demonstrate ability to customize JavaScript modules

Use Magento JavaScript mixins for customizations

- Describe advantages and limitations of using mixins.
- What are cases where mixins cannot be used?

Describe how to customize jQuery widgets in Magento

- How can a new method be added to a jQuery widget?
- How can an existing method of a jQuery widget be overridden?
- What is the difference in approach to customizating jQuery widgets compared to other Magento JavaScript module types?

3 Magento Core JavaScript Library

3.1 Demonstrate understanding of the mage library

Describe different types of Magento JavaScript templates

- Magento pseudo ES6 literals and underscore templates in Magento

Describe how to use storage and cookies for JavaScript modules

- How do you use cookies in the module?
- How is localStorage used in Magento?

Demonstrate the ability to use the JavaScript translation framework

- How are CSV translations exported to be available in JavaScript modules?
- How is a new translation phrase added using JavaScript methods?

Describe the capabilities of the JavaScript framework utils components

- What are the different components available through the mage/utils module?

Demonstrate the ability to use and customize the validation module

- What is the architecture of the validation modules in Magento?
- How can a custom validation rule be added?
- How can an existing rule be customized?

3.2 Demonstrate understanding of mage widgets

Describe the collapsible jQuery widgets in the Magento JavaScript library and how to use them

- Which collapsible widgets are available in Magento?
- How do you use them?

Demonstrate the ability to use the popup and modal widgets

- How do you create a new popup, dialog, or modal with the Magento components?

Describe the different jQuery widgets in the Magento JavaScript library

- Which other widgets are available in the Magento JavaScript library?
- How do you use them?

3.3 Demonstrate the ability to use the customer-data module

Demonstrate understanding of the customer-data module concept

- What is private data?
- Why do we need to store information in the browser?
- What are performance considerations?

Demonstrate understanding of how to use the customer-data module in customizations

- How is the customer-data module structured?
- How is data accessed, invalidated, or set?

Describe how to use sections.xml to modify the information available through the customer-data module

- How can a sections.xml file be used to add a new section and to modify the invalidation rules of an existing section?
- How can a customization change the way existing sections work?

4 UI Components

4.1 Demonstrate understanding of Knockout customizations

Describe Magento modifications to the Knockout template engine

- Describe the remote template engine, template syntax, custom tags and attributes, and the rendering mechanism.

Demonstrate the ability to use the custom Knockout bindings provided by Magento

- Where can a full list of custom bindings be found?
- What are they used for?
- What alternatives are there?

Describe the Knockout scope binding

- What is the purpose of the scope binding?
- What Knockout problem does it solve?
- How exactly does this binding work?

Demonstrate the ability to use the scope binding in customizations

- How is the scope binding used?
- How do nested scopes work?
- How can data of a parent scope be accessed from a child?
- How can the scope binding be applied to HTML in Ajax responses?

4.2 Demonstrate understanding of Magento UI components

Describe how uiComponents are executed in Magento

- What is the difference in uiCompnent execution compared to other JavaScript module types?
- What does it mean to "execute a uiComponent"?
- Why do we need the app component to execute uiComponents?
- What is the role of the layout component?

Describe the structure of a UiComponent

- What is uiClass?
- How does it instantiate uiComponents?
- How can existing component instances be accessed?
- How can a uiComponent be modified?
- How do you extend an existing uiComponent?
- What is the role of the uiElement and uiCollection modules?

Demonstrate the ability to create a uiComponent with its own data, or operate with data of existing uiComponents

- How does a uiComponent access the data it needs?
- What are the requirements for a subcomponent to provide data?
- How can data be loaded by Ajax?
- How can a component receive the data when it is loaded?

Describe the process of sharing data between components

- How can one uiComponent instance access data of another instance?
- How can components communicate while taking into account their asynchronous nature?

4.3 Demonstrate the ability to use UI components

Describe the uiComponents lifecycle

- What are the stages of uiComponent execution?
- What is the role of the layout module, and how does it load components, children, and data?
- What are the types of components it supports?

Demonstrate the ability to use uiComponents configuration to modify existing instances and create new instances

- Describe the definitons.xml file and uiComponent instance XML files.
- How can you modify an existing instance of a uiComponent using a configuration file?
- What is the role of the Magento layout in the uiComponent workflow?

4.4 Demonstrate understanding of grids and forms

Customize existing grids and create new grids

- How do you create a grid?
- How do you add an image column, standard validation, custom validation rules, and custom column types to a grid?
- How do you modify existing grids?
- How do you customize the data loading process for a grid, including filters and sorting?

Customize existing forms and create new forms

- How do you create a form, a form with tabs, a form with groups of fields, a form with dynamic fields (when one field change will cause a change in another place)?
- How do you customize existing forms?
- How do you add validation to fields, including custom validation rules?
- How can you add a file upload field, an image field, and a custom field to a form?

5 Checkout

5.1 Demonstrate understanding of checkout architecture

Describe key classes in checkout JavaScript: Actions, models, and views

- What are actions, models, and views used for in checkout?
- How does Magento store checkout data?
- What type of classes are used for loading/posting data to the server?
- How does a view file update current information?

Demonstrate the ability to use the checkout steps for debugging and customization

- How do you add a new checkout step?
- How do you modify the order of steps?
- Debug the data flow of each step.
- How do you customize a step's logic?

Customize the shipping step rendering and saving

- How does Magento save information about the shipping address for different types of checkout (logged in with default address, without default address, not logged in)?
- How does Magento obtain the list of available shipping methods?
- Which events can trigger this process?
- How does Magento save a selected address and shipping method?

5.2 Demonstrate understanding of payments

Describe the architecture of JavaScript payment modules

- Add new payment method and payment methods renderers.
- Modify an existing payment method.

Demonstrate the ability to use the payment data flow for debugging and customizations

- How does a payment method send its data to the server?
- What is the correct approach to deal with sensitive data?

Demonstrate the ability to customize and debug the order placement process in JavaScript

- Describe the data flow during order placement
- Which modules are involved?
- How can the payment step be separated from the order placement?

Example Questions

See the Answer Key following the questions for answers.

Question 1

A merchant wants you to add a close button to collapse an opened tab on a product page. The Magento Tabs widget is used for the tabs. You add this code to your template: `<button data-action="close" translate="'Close'"/>`

Keeping upgradability in mind, how do you implement this functionality?

A. Extend the Magento Tabs widget and add a callback for the button there

B. Add an attribute `onclick="$.mage.tabs.close()"` to the button

C. Add a callback for the button in the Magento Tabs widget in `lib/web/mage/tabs.js`

D. Add a `data-bind="click: close"` attribute to the button

Reference

https://devdocs.magento.com/guides/v2.2/javascript-dev-guide/javascript/custom_js.html

Question 2

A merchant has asked that the price of their free shipping method, which natively is displayed as $0.00, instead is displayed as FREE in the checkout in their store's theme.

How do you make the change?

A. Create a `requirejs-config.js` file and create a Magento mixin to replace the native `shipping-method-item` Knockout HTML template.

B. Copy the appropriate `view/shipping.js` file from Magento_Checkout to your theme and change the form uiComponent's `rate` property to:

```
rates: shippingService.getShippingRates().replace('$0.00', 'FREE');
```

C. Add the translation `'$0.00','FREE'` to your theme's translation CSV file.

D. Copy the `shipping-method-item` Knockout HTML template from `Magento_Checkout` to your theme and add the following to the template:

```
<!-- ko if: (element.getRegion('price') === '$0.00') -->FREE<!-- /ko -->
<!-- ko if: (element.getRegion('price') !== '$0.00') -->element.getRegion('price')<!-- /ko
-->
```

References
http://knockoutjs.com/documentation/if-binding.html

https://devdocs.magento.com/guides/v2.2/javascript-dev-guide/javascript/js_mixins.html

https://devdocs.magento.com/guides/v2.2/frontend-dev-guide/translations/translate_theory.html

Question 3

During a code review you notice the following code:

```
$('#element').collapsible({
    ajaxContent: 'true',
    ajaxUrlElement: '.my-ajax-url',
    content: '.my-content'
});
```

What is your peer's desired result in using `ajaxUrlElement`?

A. Magento will do a `GET AJAX` request on the `data-url` attribute in `my-ajax-url` and load the content in the
B. `.my-ajax-url` element.

C. Magento will do a `POST AJAX` request on the href in `.my-content` and load the content in the `.my-ajax-url` element.

D. Magento will replace the element `.my-ajax-url` with content loaded via `AJAX` from the `data-url` attribute set in the element `.my-ajax-url`.

E. Magento will find the element `.my-ajax-url` and load the content from that element's `href` attribute into `.my-content`.

References
https://devdocs.magento.com/guides/v2.2/javascript-dev-guide/widgets/widget_collapsible.html

https://github.com/magento/magento2/blob/2.2/lib/web/mage/collapsible.js#L518

Answer Key

Answer Key

Question 1

Answer: **A**

Question 2

Answer: **D**

Question 3

Answer: **D**

