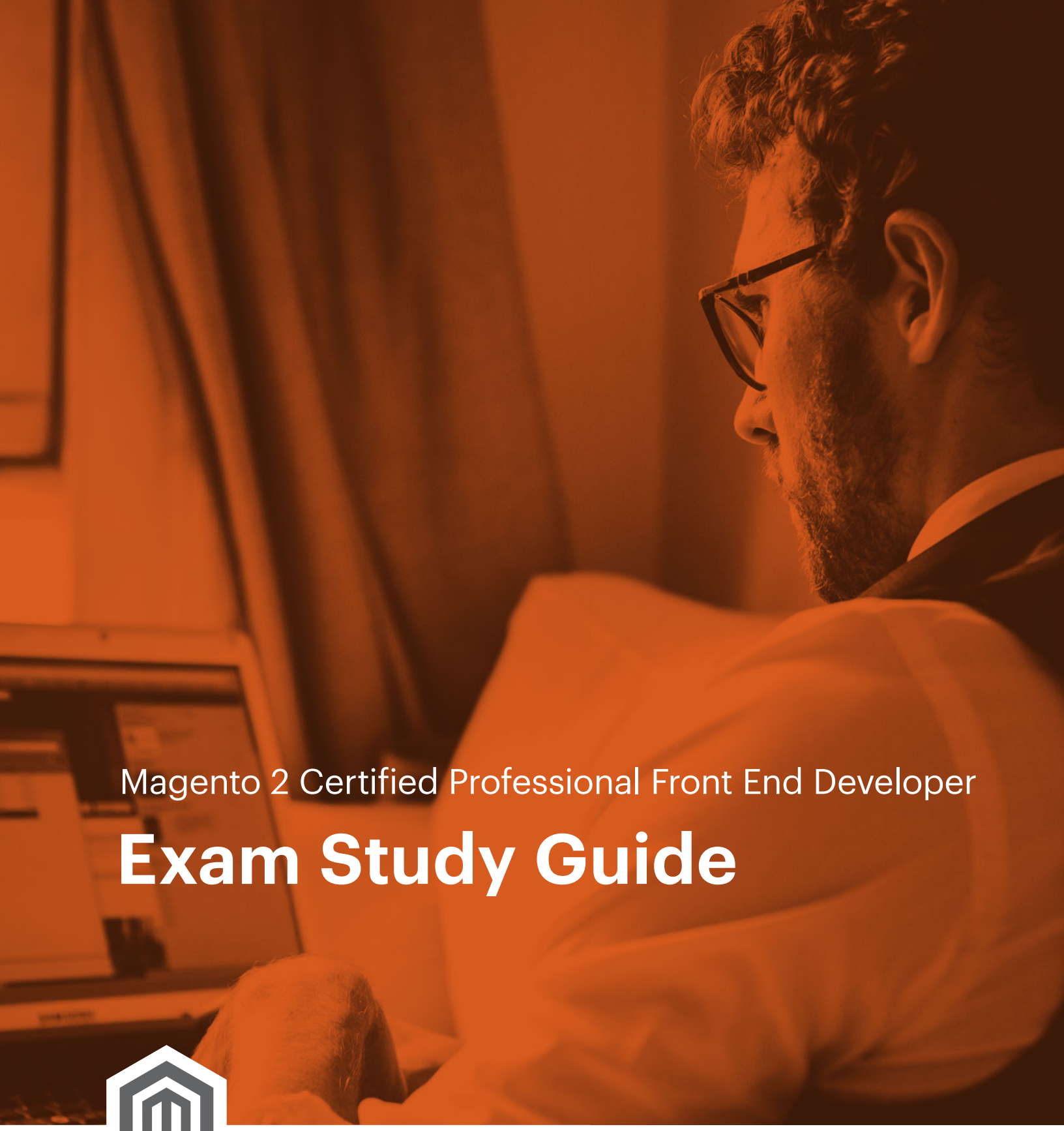

Magento 2 Certified Professional Front End Developer

Exam Study Guide

MAGENTO 2 CERTIFIED Professional Front End Developer

Magento® An Adobe Company | U

training@magento.com

u.magento.com

u.magento.com/certification

Contents

Introduction

The Magento 2 Certified Professional Front End Developer exam, a primarily scenario-based exam, is designed to validate the skills and knowledge needed to understand Magento's theming components and the ability to modify the user interface according to best practices.

A Magento Professional Front End Developer creates and customizes Magento themes, including templates, layouts, CSS, JavaScript, and other components of the front end, including translations, of a Magento site. A Magento Professional Front End Developer uses the Admin UI to implement design-related system configuration and to modify the appearance of specific pages.

This exam is for a Magento 2 Professional Front End Developer with a deep understanding of Magento 2 fundamentals, and with a recommended experience of 1.5 years of experience with the Magento 2 platform.

The test is built for the 2.3.*x* version of Magento Commerce and Magento Open Source.

Supporting Magento U courses:

- Core Principles for Theming in Magento 2 (Instructor-Led)
- Core Principles for Theming in Magento 2 (On-Demand)
- 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.	Create Themes	7%
2.	Magento Design Configuration System	7%
3.	Layout XML in Themes	18%
4.	Create and Customize Template Files	8%
5.	Static Asset Deployment	5%
6.	Customize and Create JavaScript	17%
7.	Use LESS/CSS to Customize Magento Look and Feel	8%
8.	Customize the Look and Feel of Specific Magento Pages	22%
9.	Implement Internationalization of Frontend Pages	5%
10.	Magento Development Process	3%

Topics and Objectives

1 Create Themes

1.1 Describe folder structure for local and Composer-based themes

In which folders can themes be located? What determines where a theme is installed? What is the difference if a theme is installed in one or the other of the possible directories?

1.2 Describe the different folders of a theme

Which folders can exist within a theme? Which folders are optional and which are required? What is the purpose of each of the folders?

1.3 Describe the different files of a theme

Which files are required to be present in a theme? Which files are optional? What is the purpose of each of the different file types?

1.4 Understand the usage of Magento areas: adminhtml/base/frontend

Which design areas exist? What is the difference between them, and what do design areas have in common? What are design areas used for? How can design areas be utilized for custom themes or customizations?

2 Magento Design Configuration System

2.1 Describe the relationship between themes

What type of relationships can exist between themes? What is the difference between a parent theme and a child theme? How can the relationship between themes be defined and influenced? How is that taken into account when creating a custom theme or customizing an existing theme?

2.2 Configure the design system using the options found in the Admin UI under Content > Design > Configuration

How do the configuration settings affect theme rendering? What happens if a theme is added or removed? What common mistakes can be made in regard to these settings?

2.3 Apply a temporary theme configuration to a store view using the options found in the Admin UI under Content > Design > Schedule

What is the purpose of this feature? How does it influence rendering if a design change is scheduled? What happens at the end of a scheduled design? How is full page caching involved?

2.4 Understand the differences and similarities between Content > Design > Configuration and > Schedule to configure the design fallback

What is the effect if both options are used at the same time?

3 Layout XML in Themes

3.1 Demonstrate knowledge of all layout XML directives and their arguments

What layout XML elements exist and what is their purpose? What is the purpose of the attributes that are available on blocks and other elements?

3.2 Describe page layouts and their inheritance

How can the page layout be specified? What is the purpose of page layouts? How can a custom page layout be created? Where are the existing page layouts used? How can the root page layout be specified for all pages and for specific pages?

3.3 Demonstrate understanding of layout handles and corresponding files

How can the available layout handles for a given page be determined? How do you add a new layout handle? What is the purpose of layout handles? What are the most commonly used layout handles? How can layout handles be used during theme customization?

3.4 Understand the differences between containers and blocks

What is the purpose of blocks? What is the purpose of containers? How can containers be used in theming? How can blocks be used in theming? What is the default block type? How can the order of rendered child blocks be influenced both in containers and in blocks?

3.5 Describe layout XML override technique

How can layout XML be overridden? How can layout overriding be used in theming? What are consequences of layout overrides during upgrades? What is the effect of layout overrides on compatibility?

3.6 Understand layout merging

What is layout merging? How do design areas influence merging? How can merging remove elements added earlier? What are additive changes and what are overriding changes during layout merging?

3.7 Understand processing order of layout handles and other directives

In what order are layout handles processed? In what order is layout XML merged within the same handle? How can the processing order be influenced? What are common problems arising from the merge order of layout declarations?

3.8 Set values on block instances using layout XML arguments

How can arguments be set on blocks? Which data types are available? What are common arguments for blocks?

3.9 Customize a theme's appearance with `etc/view.xml`

What is the `etc/view.xml` file used for? How can it be used to customize a theme? How can values from `etc/view.xml` be used during theming? How does theme inheritance influence values from `etc/view.xml`?

4 Create and Customize Template Files

4.1 Assign a customized template file using layout XML

How can a customized template file be assigned to a block using layout XML? How does overriding a template affect upgradability? What precautions can be taken to ease future upgrades when customizing templates?

4.2 Override a native template file with a customized template file, using the design fallback

How can the design fallback be used to render customized templates? How does that influence upgradability? How can you determine which template a block renders?

4.3 Describe conventions used in template files

What conventions are used in PHP templates? Why aren't the common PHP loop and block constructs used? Which common methods are available on the $block variable?

How can a child block be rendered? How can all child blocks be rendered? How can a group of child blocks be rendered?

4.4 Render values of arguments set via layout XML

How can values set on a block in layout XML be accessed and rendered in a template?

4.5 Demonstrate ability to escape content rendered and template files

How can dynamic values be rendered securely in HTML, HTML attributes, JavaScript, and in URLs?

5 Static Asset Deployment

5.1 Describe the static asset deployment process for different file types

What commands must be executed to deploy static file types? What are common mistakes during the process?

5.2 Describe the effect of deploy modes on frontend development

What are the differences between development and production mode in regard to frontend development?

5.3 Demonstrate your understanding of LESS > CSS deployment and its restrictions in development

Which LESS compilation options are available in Magento? How are they different? How do they influence the developer workflow during theming?

6 Customize and Create JavaScript

6.1 Include custom JavaScript on pages

What options exist to include custom JavaScript on a page? What are the advantages and disadvantages of inline JavaScript? How can JavaScript be loaded asynchronously on a page? How can JavaScript on a page be configured using block arguments in layout XML? How can it be done directly in a .phtml template?

6.2 Demonstrate understanding of using jQuery

Demonstrate understanding of jQuery and jQuery UI widgets. Demonstrate understanding of how to use Magento core jQuery widgets. How can jQuery UI Widget methods be instantiated? How can you call jQuery UI Widget methods? How can you add new methods to a jQuery UI Widget? How can a jQuery UI Widget method be wrapped with custom logic?

6.3 Demonstrate understanding of requireJS

How do you load a file with require.js? How do you define a require.js module? How are require.js module dependencies specified? How are module aliases configured in requirejs-config.js? How do you regenerate the compiled requirejs-config.js file after changes? How do you configure module aliases in requirejs-config.js? How do you debug which file a requireJS alias refers to? Demonstrate that you understand how to create and configure Magento JavaScript mixins.

6.4 Configure JavaScript merging and minify in the Admin UI

What options are available to configure JavaScript minification and bundling? How does Magento minify JavaScript? What is the purpose of JavaScript bundling and minification?

6.5 UI component configuration

How can you specify configuration options on a UiComponent widget in JSON and in Layout XML? What configuration options are available on UiComponents? How do you specify the ko template for a UiComponent? Demonstrate an understanding of default.tracks

6.6 Understanding knockout framework

How do you use knockout.js bindings? How do you bind a ko view model to a section of the DOM with the scope binding? How do you render a ko template of a UiComponent? Demonstrate an understanding of the different types of knockout observables. What common ko bindings are used? Demonstrate an understanding of ko virtual elements.

6.7 Understanding dependency between components

Demonstrate an understanding of the links, imports, exports, and listens UiComponent configuration directives.

6.8 Understanding string templates

Demonstrate an understanding of ES5 string literal templates like ${$.provider}. What does $. Inside of ${ } resolve to?

7 Use LESS/CSS to Customize the Magento Look and Feel

7.1 Explain core concepts of LESS

Describe features like file import via @import directive, reusable code sections via mixins together with parameters and the usage of variables. Demonstrate your understanding of the special variable @arguments.

Demonstrate how to use the nesting code formatting, and the understanding of media queries together with nesting. Describe how the & (Ampersand) works and its function. Describe how calculations are possible as well.

7.2 Explain Magento's implementation of LESS (@magento_directive)

Demonstrate the process from magento-less files via php preprocessing into real LESS files with extracted @import directives. Where can the intermediate files be found?

What do you have to remember, when you change a less file? Which files will be re-processed on file changes? Are the original files copied or symlinked in developer environments?

7.3 Describe the purpose of _module.less, _extend.less, _extends.less

Demonstrate LESS has no fallback capabilities and therefor magento created @magento_import directives to enable FE devs to inject or replace parts of existing less structures of modules and themes.

7.4 Show configuration and usage of CSS merging and minification

Demonstrate the primary use case for merging and minifaction. Determine how these options can be found in the backend. Understand the implications merging has in respect to folder traversal.

7.5 Magento UI library usage

Demonstrate your understanding of magento's UI library, a LESS-based library of mixins and variables for many different standard design elements on websites. How can you take advantage of the UI library? What do you have to do to enable it in your theme?

Which file is primarily used for basic setup of variables? Where can UI library files be found? How can it be extended? How can you change specific parts of the UI library?

8 Customize the Look and Feel of Specific Magento Pages

8.1 Utilize generic page elements

Demonstrate an understanding of customizing generic page elements that can be found on most pages: page header and footer, quick search, store view (language) switcher, mini cart, breadcrumbs, and sidebar menu.

8.2 Customizing product detail pages

How can design changes (page layout) be configured on product detail pages? How can design changes be configured for specific product types?

How can you use custom layout updates for specific product pages? Demonstrate an understanding of how to use the container blocks provided by Magento to display additional information on category pages.

8.3 Customizing category pages

How can design changes (page layout) be configured on category pages? How can the layered navigation be configured? Demonstrate an understanding of configuring design inheritance for category pages. How can a CMS block be configured as a category landing page?

8.4 Customizing CMS pages

How can design changes (page layout) be configured on CMS pages? Demonstrate an understanding of static variables in CMS blocks and pages. Demonstrate an understanding of the use of CMS template directives (var, store, block, …).

8.5 Customizing widgets

How is a widget instance created? Where can widgets be used? How can a custom widget target be created? Demonstrate an understanding of configuring a widget instance.

8.6 Customizing CMS blocks

How do you create and insert CMS blocks? Demonstrate an understanding of the use of CMS template directives (var, store, block, …).

Demonstrate an understanding of Page Builder folder structure and theme inheritance.

8.7 Customizing customer account pages

How do you remove or add an item from the customer account navigation using layout XML? Demonstrate an understanding of formatting customer addresses.

8.8 Customizing one-page checkout

Demonstrate an understanding of the container blocks provided in the Magento checkout to display additional information.

8.9 Understand customization of transactional email templates

How do you create and assign custom transactional email templates? How do you use template variables available in all emails? How do you access properties of variable objects (for example, var order.getCustomer.getName)?

How can you create a link to custom images from transactional email templates? How do you create links to store pages in transactional email templates?

9 Implement Internationalization of Frontend Pages

9.1 Create and change translations

Demonstrate an understanding of internationalization (i18n) in Magento. What is the role of the theme translation dictionary, language packs, and database translations?

Understand the pros and cons of applying translations via the translate.csv file versus the core_translate table. In what priority are translations applied?

9.2 Translate theme strings for .phtml, emails, UI components, .js files

Demonstrate an understanding of string translation in JavaScript.

10 Magento Development Process

10.1 Determine ability to manage cache

Demonstrate an understanding of configuring the Magento cache types for development and production.

10.2 Understand Magento console commands

How do you switch between deploy modes? What bin/magento commands are commonly run during frontend development?

Magento 2 Certified Professional Front End Developer Exam *Example Questions*

See the Answer Key following the questions for answers.

Question 1

You are developing a German language theme for the Magento Marketplace named mytheme. In your Magento installation you have a third-party German language package installed.

The theme "mytheme" already contains a `mytheme/i18n/de_DE.csv` file. The graphic designer wants you to rename the Add to Cart button to make the text shorter.

Keeping upgradability in mind, where do you add the new string?

A. Override the `Magento_Catalog/view/frontend/templates/product/view/addtocart.phtml` template in mytheme and replace the string.
B. Add the string to `mytheme/i18n/de_DE.csv`
C. Add the string to the third-party German language package.
D. Add the translation to the `core_translate` database table.

Reference: Magento Dev Docs: Use translation dictionary to customize strings

https://devdocs.magento.com/guides/v2.3/frontend-dev-guide/translations/theme_dictionary.html

Question 2

Which three properties are set in `theme.xml`?

A. Title of a theme
B. Composer package version
C. Theme area: frontend or adminhtml
D. Theme preview image
E. Parent theme

Reference: Magento Dev Docs: Create a new storefront theme

http://devdocs.magento.com/guides/v2.3/frontend-dev-guide/themes/theme-create.html

Question 3

You need to add a custom block of HTML to the header of every page using a layout file in `MyCompany/mytheme`.

Keeping upgradability in mind, how do you add the block?

A. Put your customization in `MyCompany/mytheme/layout/extend.xml`
B. Include `<extend file="Magento_Theme::default.xml">` in your `MyCompany/mytheme/Magento_Theme/layout/default.xml`
C. Put your customization in `MyCompany/mytheme/Magento_Theme/layout/default.xml`
D. Keep track of your changes and apply them to `Magento/Theme/layout/default.xml` after upgrade

Reference: Magento Dev Docs: Extend a layout

http://devdocs.magento.com/guides/v2.3/frontend-dev-guide/layouts/layout-extend.html

Question 4

A merchant asks you to add frontend validation to their newsletter form. You notice the form has the following attributes:

```
<form class="form newsletter" novalidate id="newsletter-validate-detail">
```

Which of the following choices will add frontend validation?

A. Add `form_key` input inside of `form`.
B. Remove the `novalidate` attribute from `form`.
C. Add a validation object using `data-mage-init attribute`.
D. Add the class `mage-validate` to `form`.

Reference: Magento Dev Docs: Calling and initializing JavaScript

https://devdocs.magento.com/guides/v2.3/javascript-dev-guide/javascript/js_init.html

Answer Key

Answer Key

Question 1

Answer: B

Question 2

Answers: A, D, E

Question 3

Answer: C

Question 4

Answer: C

