DUMPSBOSS.

Adobe Commerce Front-End Developer Expert

Adobe AD0-E720

Version Demo

Total Demo Questions: 5

Total Premium Questions: 52

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Magento Architecture	5
Topic 2, Magento Theming	18
Topic 3, Magento Layouts	8
Topic 4, Magento UI Components	2
Topic 5, Magento JavaScript	10
Topic 6, Magento CSS	9
Total	52

QUESTION NO: 1

What is the difference between styles-l.less and styles-m.less ?

A. styles-l.less is used to generate basic and mobile-specific styles and styles-m.less is used to generate desktop-specific styles.

B. styles-l.less is used to generate desktop-specific styles and styles-m.less is used to generate only mobile-specific styles.

C. styles-l.less is used to generate desktop-specific styles and styles-m.less is used to generate basic and mobile-specific styles.

ANSWER: C

Explanation:

`styles-l.less` is used to generate desktop-specific styles, and `styles-m.less` is used to generate the base and mobile-specific styles. Adobe Commerce's Blank theme follows a mobile-first CSS approach: the mobile stylesheet supplies common styling that can apply across viewport sizes, while the large-view stylesheet supplies rules intended for larger, desktop-sized viewports. During static view file deployment, these LESS entry-point files are processed into their corresponding CSS assets, such as `styles-m.css` and `styles-l.css`. This separation allows a storefront to load foundational and small-screen styling first, then apply desktop layout refinements through the large-screen stylesheet and responsive media-query rules. The source entry points in the Blank theme demonstrate this organization: styles-m.less is the mobile/base entry point, while styles-l.less is the desktop-oriented entry point. Therefore, the statement pairing `styles-l.less` with desktop-specific styles and `styles-m.less` with basic and mobile-specific styles accurately describes their roles.

QUESTION NO: 2

An Adobe Commerce developer wants to determine which template is rendering a specific element on the storefront. Which two methods can they use to turn on template hints? (Choose two.)

A. In the Admin, navigate to Stores > Configuration > Advanced > Developer > Debug.Set Enabled Template Path Hints for Storefront to Yes

B. In the Admin, navigate to system > Advanced > Template > Developer > Debug.Set Enabled Template Path Hints for Storefront to Yes

C. On the command line, run the command; php bin/magento setup:enable-template-hints

D. On the command line, run the command; php bin/magento dev:template-hints:enable

ANSWER: A D

Explanation:

The Admin configuration path, *Stores* > *Configuration* > *Advanced* > *Developer* > *Debug*, provides the storefront setting for template path hints. Setting *Enabled Template Path Hints for Storefront* to *Yes* causes Adobe Commerce to display the template file path associated with rendered storefront blocks. This is useful when tracing the source of page markup, locating a theme override, or identifying the template that should be customized.

The `php bin/magento dev:template-hints:enable` command provides the corresponding CLI-based way to enable storefront template hints. It is especially useful for developers working over SSH, in local development environments, or in workflows where configuration changes are managed from the command line. After enabling hints, the storefront must be viewed in the applicable store scope so the rendered template information can be inspected. Because template hints expose implementation details and visibly alter storefront output, they should be used for development and troubleshooting rather than left enabled on a production storefront.

Adobe documents the developer CLI commands at Developer commands and the related Admin developer settings at Developer configuration settings.

QUESTION NO: 3

An Adobe Commerce developer is trying to remove a block using the

Options:

○ A.

US6

○ B.

Use

○ C.

Use Check Answer

A. Use

B. Use

C. To remove a block using layout XML, the developer should use the tag with the name attribute specifying the name of the block and the remove attribute set to true. For example:

ANSWER: A

Explanation:

`<referenceBlock name="test.block" remove="true" />` is the appropriate layout XML instruction because it targets an already declared block by its layout name and marks that element for removal during layout generation. Adobe Commerce layout XML uses `referenceBlock` to modify a block that was created elsewhere in the merged layout. The `name` value must match the block's existing layout element name, such as `test.block`, and `remove="true"` instructs the layout system not to render that block. Element names in layout XML are case-sensitive, so the supported instruction is `referenceBlock` with a capital `B`. The self-closing form is valid XML and is the concise form normally used when no child instructions are needed. This removal is applied after layout handles are merged, enabling a theme or custom module to suppress a block introduced by another layout without changing the original block declaration. Adobe documents the `referenceBlock` instruction and its `remove` attribute in the Layout XML instructions guide. Additional context on how layout XML is merged and processed is available in the Adobe Commerce layouts guide.

QUESTION NO: 4

An Adobe Commerce developer is trying to remove a block using the

Options:

A.

US6

B.

Use

C.

Use

A. Use

B. Use

C. Use

D. On the command line, run the command; php bin/magento dev:template-hints:enable

ANSWER: A

Explanation:

`<referenceBlock name="test.block" remove="true" />` is the valid Adobe Commerce layout XML instruction for removing an existing block from the merged layout. A layout update does not need to recreate the target block to remove it. Instead, `referenceBlock` locates the block by the unique layout name assigned when that block was declared, and the `remove` attribute instructs the layout processor to exclude that element from the resulting page structure. Consequently, the block is not rendered on the storefront. The element name is case-sensitive: Adobe Commerce documents the instruction as `referenceBlock`, using a capital `B`. The `name` value must match the block's layout name, such as `test.block`; it is not a PHP class name or a module-prefixed identifier. This approach can be placed in an appropriate theme or module layout update for the page handles where the block should be absent. Adobe's layout XML reference documents `referenceBlock` and its `remove` attribute, including its use for removing blocks from a page layout. See the Adobe Commerce layout XML instructions reference and the Adobe Commerce layout overview.

QUESTION NO: 5

An Adobe Commerce developer wants to apply a knockout binding to a

to run a function, onClick(), when it's clicked. Which two solutions would the developer use to achieve this? (Choose two.)

A.

B.

C.

D.

ANSWER: A C

Explanation:

`<div data-bind="click: function () { onClick() }"></div>` is a valid solution because the Knockout `click` binding receives a function that is invoked only when the user clicks the element. The wrapper function then calls `onClick()`, which is useful when the handler must be invoked explicitly or when additional statements or arguments are needed in the click handler.

`<div data-bind="click: onClick"></div>` is also the standard concise Knockout syntax. It supplies the `onClick` function itself as the binding value rather than executing it while Knockout evaluates bindings. When the click event occurs, Knockout invokes that function and supplies the current binding context and browser event according to the click-binding contract. In Adobe Commerce UI templates, Knockout bindings are placed in the `data-bind` attribute and evaluated in the component's binding context. This makes the named method available for event handling while preserving Knockout's event lifecycle and context behavior.

See the Knockout click binding documentation and Adobe Commerce's Knockout bindings documentation.