Adobe AD0-E720 Practice Questions

Shared by Monroe on 19-08-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

The merchant needs to create a new website, and is need modify a template the third party vendor's, because the customer is different. The file is found in a module here: app/code/Vendor/Module

Keep it simple in your mind!

Options:

A- Create another layout for the new website and configure new file.phtml. app/code/Vendor/Module/view/frontend/templates/file.phtml
B- Create a new module for extends layout.xml and include new file.phtml. app/code/Vendor/Module_Two/view/frontend/templates/file.phtml
C- Create a new theme, define a new website and customize in app/design. app/design/frontend/Custom/Theme/Vendor_Module/templates/file.phtml

Answer:

C

Explanation:

The best way to customize a template file from a third-party module is to create a new theme that inherits from the parent theme and override the template file in the app/design/frontend/Custom/Theme/Vendor_Module/templates directory. This way, the customization is isolated from the original module and can be applied to a specific website or store view. Creating another layout file or a new module would not be as simple or flexible as creating a new theme. Reference:Frontend development guide, [Create a theme], [Theme inheritance]

Question 2

Question Type: MultipleChoice

An Adobe Commerce developer wants to create a new theme Vendor_Orange which extends from MagentoMum

a. Which file is responsible for specifying the parent theme?

Options:

A- view.xml
B- registration.php
C- theme.xml

Answer:

C

Explanation:

The theme.xml file is responsible for specifying the parent theme of a custom theme. The file should contain the element with the value of the parent theme's directory, such as MagentoMuma. The view.xml file is used to configure the theme's images, fonts, and layout. The registration.php file is used to register the theme in the system. Reference: [Create a theme], [theme.xml]

Question 3

Question Type: MultipleChoice

The merchant needs to create a new website, and is need modify a template the third party vendor's, because the customer is different. The file is found in a module here: app/code/Vendor/Module

Keep it simple in your mind!

Options:

A- Create another layout for the new website and configure new file.phtml. app/code/Vendor/Module/view/frontend/templates/file.phtml
B- Create a new module for extends layout.xml and include new file.phtml. app/code/Vendor/Module_Two/view/frontend/templates/file.phtml
C- Create a new theme, define a new website and customize in app/design. app/design/frontend/Custom/Theme/Vendor_Module/templates/file.phtml

Answer:

C

Explanation:

The best way to customize a template file from a third-party module is to create a new theme that inherits from the parent theme and override the template file in the app/design/frontend/Custom/Theme/Vendor_Module/templates directory. This way, the customization is isolated from the original module and can be applied to a specific website or store view. Creating another layout file or a new module would not be as simple or flexible as creating a new theme. Reference:Frontend development guide, [Create a theme], [Theme inheritance]

Question 4

Question Type: MultipleChoice

An Adobe Commerce developer wants to add a custom widget that extends the default Calendar Widget. What would the contents of this file look like?

A)

```
define([
  'mage/utils/wrapper',
  'mage/calendar'
], function(wrapper, calendarWidget){
     var updateCalendarWidget = wrapper.wrap(targetWidget.prototype._function,
     function (original) {
          calendarWidget({...});
          return original();
     });
     targetWidget.prototype._function = updateCalendarWidget;
     return targetWidget;
});
```

B)

```
define([
  'jquery',
  'jquery-ui-modules/widget',
  'mage/calendar'
], function($){
     $.widget('custom.calendar', $.mage.calendar, { ... });
     return $.custom.calendar;
});
```

C)

```
define([
  'jquery',
  'mage/calendar'
], function(calendarWidget){
     return calendarWidget.extend({
          _init: function() {
               this._super();
          }
     })
});
```

Options:

A- Option A
B- Option B
C- Option C

Answer:

B

Explanation:

To add a custom widget that extends the default Calendar Widget, the contents of the file would look like option B. This is because option B follows the correct syntax and structure for defining a jQuery widget in Magento. The code does the following steps:

Defines a module with the name ''Vendor_Module/js/calendar-widget'' that depends on the ''jquery/ui'' and ''Magento_Ui/js/lib/knockout/bindings/datepicker'' modules.

Returns a function that creates a new widget with the name ''vendor.calendarWidget'' that extends the base calendar widget class.

Overrides the init function of the base calendar widget class to add custom logic or functionality to the widget.

Option A is not correct because it does not use the correct syntax for defining a jQuery widget. It uses a script tag instead of a define function, which is not valid for creating a module. It also uses an incorrect name for the widget, which should use a dot instead of a slash. Option C is not correct because it does not use the correct syntax for extending a widget. It uses an extend function instead of a widget function, which is not valid for creating a new widget. It also does not return anything from the module, which will cause an error. Reference: [jQuery widgets], [Calendar Widget]

Question 5

Question Type: MultipleChoice

An Adobe Commerce developer wants to override the following Layout XML file in the theme ExampleCorp/orange.

app/design/frontend/ExampleCorp/blank/Vendor_Module/layout/catalog_product_view.xml

What path would the developer use inside the layout directory of the theme to override the file?

Options:

A- /override/ExampleCorp/blank/catalog_product_view.xml
B- /override/theme/ExampleCorp/blank/catalog_product_view.xml
C- /catalog_product_view.xml

Answer:

C

Explanation:

To override a layout XML file from a parent theme, the developer just needs to place the modified file in the same path relative to the layout directory of the child theme. In this case, the file would be app/design/frontend/ExampleCorp/orange/Vendor_Module/layout/catalog_product_view.xml. The override directory is not used for overriding layout files, but for overriding templates and web assets. Reference: [Layout instructions], [Override templates and layout files]

Question 6

Question Type: MultipleChoice

an Adobe commerce developer wants to override the core Magento Ul library dropdowns in your theme. Which is the correct way to achieve this?

Options:

A- /web/css/source/_dropdowns.less

B- lib/web/css/source/.dropdowns.less
C- /web/css/source/lib/.dropdowns.less

Answer:

A

Explanation:

To override the core Magento UI library dropdowns in a custom theme, the developer needs to create a file named _dropdowns.less in the /web/css/source directory of the theme. This file will override the default _dropdowns.less file from the lib/web/css/source/lib directory and apply the custom styles to the dropdown elements. The lib/web/css/source/_dropdowns.less and /web/css/source/lib/_dropdowns.less files are not valid and will not work, as they do not follow the theme structure or the naming convention. Reference: [Dropdowns], [Theme structure]

Question 7

Question Type: MultipleChoice

An Adobe Commerce developer is using a view model within an existing block:

```
<referenceBlock name="blog.posts.list">
    <arguments>
        <argument name="view_model" xsi:type="object">ExampleCorp\Blog\ViewModel\MyNewViewModel</argument>
    </arguments>
</referenceBlock>
```

What are two ways to access the view model class in the template? (Select two.)

Options:

A- $block->getData('view_model')
B- $block->viewModel()
C- $block->getViewHodel()
D- $block->getData('viewModel)

Answer:

A, D

Explanation:

To access a view model within an existing block, the developer can use either of the following ways:

$block->getData('view_model'): This method will return the view model object that is assigned to the argument name ''view_model'' in the layout XML file. For example:

<referenceBlock name=''blog_posts_list''> ExampleObjectModel/ExampleObjectModel </referenceBlock>

In the template file, the developer can access the view model object by using:

$block->getData('view_model')

$block->getData('viewModel'): This method will return the view model object that is assigned to the argument name ''viewModel'' in the layout XML file. For example:

<referenceBlock name=''blog_posts_list''> ExampleObjectModel/ExampleObjectModel </referenceBlock>

In the template file, the developer can access the view model object by using:

$block->getData('viewModel')

The following methods are not valid and will not work:

$block->viewModel(): This method does not exist and will cause an error.

$block->getViewHodel(): This method is misspelled and will cause an error.

Question 8

Question Type: MultipleChoice

By creating a Custom_Module, an Adobe Commerce Developer has implemented a new Page Builder viewport for tablet devices but the viewport's tablet selector button is missing.

The button .svg has been properly added to the path: CustomJ^odule/web/css/images/switcher/switcher-tablet .svg. How the developer would implement the viewport button icon?

A)

By setting the node icon in the theme's etc/view.xml file for the respective viewport configuration data.

```
<view>
    <vars module="Magento_PageBuilder">
        <var name="breakpoints">
            <var name="tablet">
                <!-- ... -->
                <var name="icon">Custom_Module::css/images/switcher/switcher-tablet.svg</var>
                <!-- ... -->
            </var>
        </var>
    </vars>
</view>
```

B)

By setting the node button-image in the theme's etc/view.xml file for the respective viewport configuration data.

```
<view>
    <vars module="Magento_PageBuilder">
        <var name="breakpoints">
            <var name="tablet">
                <!-- ... -->
                <var name="button-image">Custom_Module::css/images/switcher/switcher-tablet.svg</var>
                <!-- ... -->
            </var>
        </var>
    </vars>
</view>
```

C) By adding the node image in the theme's etc/viewport.xml file for the respective viewport configuration data.

```
<viewport>
<vars module="Magento_PageBuilder">
    <var name="breakpoints">
        <var name="tablet">
            <!-- ... -->
            <var name="image">PageBuilder_Breakpoints::css/images/switcher/switcher-tablet.svg</var>
            <!-- ... -->
        </var>
    </var>
</vars>
</viewport>
```

D)

By adding the node image in the theme's etc/viewport.xml file for the respective viewport configuration data.

```
<viewport>
<vars module="Magento_PageBuilder">
    <var name="breakpoints">
        <var name="tablet">
            <!-- ... -->
            <var name="image">PageBuilder_Breakpoints::css/images/switcher/switcher-tablet.svg</var>
            <!-- ... -->
        </var>
    </var>
</vars>
</viewport>
```

Options:

A- Option A
B- Option B
C- Option C
D- Option D

Answer:

C

Explanation:

Option C is the correct way to implement the viewport button icon. The image node specifies the path to the .svg file relative to the web/css directory of the module. Option A is incorrect because there is no icon node in the viewport configuration data. Option B is incorrect because there is no button-image node in the viewport configuration data. Option D is incorrect because the image node value should not include the web/css part of the path.
https://api.flutter.dev/flutter/widgets/PageView-class.html

P2P exams