Free Questions for B2B-Commerce-Developer by dumpssheet

Shared by Morton on 24-05-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

A developer has just deployed a new Lightning Web Component to an authorized org. What should the developer do next to use the new component on apage?

Options:

A- Go to 'Deploy LWC' in Setup.

B- Navigate 3 to the Page, Click on the 'Custom Component Editor,' Click 'Publish' on the new component in the list and adjust the positioning.

C- Create a Metadata API (MDAPI) conversion file with the Command Line interface (CLI) then go to the page and adjust the positioning.

D- Go to the page, edit it, and drag the new component onto the page.

Answer:

D

Explanation:

After deploying a new Lightning Web Component (LWC) to an org, the typical next step to use the component on a page involves navigating to that page, entering the edit mode (usually through App Builder or a similar interface), and then dragging the new component from the components palette onto the desired location on the page. This process is consistent with Salesforce's modular design approach, allowing for easy integration and configuration of components within the Salesforce ecosystem.

Question 2

Question Type: **MultipleChoice**

What are two maintainable ways that Lightning Web Components can be made mobile ready? 33m 215

Options:

A- Create a Lightning app page and add the component to the mobile navigation

B- Import a third party JavaScript library

C- Install the mobile extensions plug-in for VS Code

D- Decorate templates with mobile-ready

Answer:

A, D

Explanation:

To make Lightning Web Components mobile-ready, one maintainable approach is to create a Lightning app page and then add the component to the mobile navigation. This ensures that the component is accessible and optimized for mobile users within the Salesforce mobile app. Another approach is to design the component's templates with responsiveness in mind, using CSS and layout techniques that adapt to different screen sizes. Salesforce documentation on mobile-ready development practices provides guidelines on creating responsive designs and optimizing components for mobile use.

Question 3

Question Type: MultipleChoice

Which component can be used in other Salesforce Experience templates outside of B2B Commerce?

Options:

A- Quick Order

B- CMS Collection

C- Product Detail Data

D- Results Layout

Answer:

B

Explanation:

In Salesforce Experience Cloud, components like CMS Collection and Results Layout are designed to be reusable across different Experience templates, not just limited to B2B Commerce. CMS Collection allows for the display of CMS content in a flexible and dynamic layout, while Results Layout can be used to present search or query results in a customizable format. Salesforce documentation on Experience Cloud components emphasizes the reusability and adaptability of these components across various templates and contexts.

Question 4

Question Type: MultipleChoice

Which three file extensions are allowed in a Lightning Web Component folder?

Options:

A- .js-meta.xml

B- .html

C- .Js

D- .gif

E- .jar

Answer:

A, B, C

Explanation:

In a Lightning Web Component folder, the allowed file extensions include .js-meta.xml for the component's metadata, .html for the component's markup, and .js for the component's JavaScript class. These files are essential for defining the structure, behavior, and metadata of a LWC. Salesforce LWC documentation provides detailed information on the structure of a LWC bundle and the purpose of each file type within it.

Question 5

Question Type: MultipleChoice

Which three decorators can be used in Lightning Web Components?

Options:

A- @api

B- @track

C- @wire

D- @class

E- @import

Answer:

A, B, C

Explanation:

In Lightning Web Components, the decorators @api, @track, and @wire play crucial roles. The @api decorator is used to expose public properties and methods, making them accessible to other components. The @track decorator is used to mark private properties as reactive, so the UI updates when their values change. The @wire decorator is used to wire Apex methods or Salesforce data to the component. Salesforce documentation on LWC development extensively covers these decorators, explaining their usage and best

practices.

Question 6

Question Type: MultipleChoice

Which three statements are accurate?

Options:

A- An Aura component can contain another Aura component

B- An Aura component can contain a Lightning Web Component

C- A Lightning Web Component can contain an Aura component

D- A Lightning Web Component cannot contain an Aura component

Answer:

A, B, D

Explanation:

Salesforce documentation clarifies the interoperability between Aura and Lightning Web Components (LWCs). An Aura component can indeed contain another Aura component, as well as a LWC, allowing for a mix of component technologies in a single application. However, due to the architectural and design principles of LWCs, a LWC cannot contain an Aura component. This is because LWCs are designed to be lightweight and leverage web standards, which makes them not fully compatible with the older Aura component framework in terms of containment.

Question 7

Question Type: MultipleChoice

Which two items are required for a developer to bring picklist values into a Lightning Web Component?

Options:

A- import { getPicklistvValues } from 'lightning/uiObjectinfoApi';

B- import { LightningElement, wire } from 'lwc';

C- import { wire } from 'lwc';

D- import { picklistValues } from 'lightning/uiObjectinfoApi';

Answer:

A, B

Explanation:

To bring picklist values into a Lightning Web Component (LWC), a developer needs to import specific modules from the lwc and lightning/uiObjectInfoApi namespaces. The getPicklistValues function from the lightning/uiObjectInfoApi module is used to fetch the picklist values based on record type and field metadata. Additionally, importing { LightningElement, wire } from lwc is essential for defining the LWC class and using the @wire decorator to wire the getPicklistValues to a property or function. Salesforce documentation on LWC and utilizing the uiObjectInfoApi provides clear guidelines on how to implement this functionality.

Question 8

Question Type: MultipleChoice

Which wire adapter can a developer use to retrieve metadata about a specific object?

Options:

A- getObject

B- getObjectMetadata

C- All of the above

D- getObjectinfo

Answer:

D

Explanation:

To retrieve metadata about a specific object in a Lightning Web Component, a developer can use the getObjectInfo wire adapter. This adapter provides access to the metadata of a specified Salesforce object, including fields, record types, and layouts, which is essential for dynamic component rendering based on the object's schema.