Number: B2B Commerce Developer
Passing Score: 800
Time Limit: 120
File Version: 5.0

Exam Code: B2B Commerce Developer

Exam Name: Salesforce Accredited B2B Commerce Developer

Exam A

QUESTION 1
A user wants to have a Contact Us page in the storefront. This page will be a web-tolead form and it should have the header and footer, essentially the same look and feel as all the pages in the application. How can this requirement be fulfilled?

A. Page Include
B. Subscriber Page (CC Page)
C. Subscriber Template
D. Body Include Begin

Correct Answer: B
Section:
Explanation:
To have a Contact Us page in the storefront that is a web-to-lead form and has the same look and feel as all the pages in the application, the requirement can be fulfilled by creating a Subscriber Page (CC Page). This is a custom Visualforce page that can be added to the storefront and use the standard header and footer components. The page can also include a web-to-lead form that submits data to Salesforce as leads. Salesforce Reference:B2B Commerce and D2C Commerce Developer Guide,Subscriber Pages

QUESTION 2
A user wants to have a customized experience for adding items to the cart. The user also wants the mini cart module to reflect changes to the state of the cart afterwords. How should this requirement be fulfilled?

A. Leverage the Addto Cart Global API which add items to the cart and also refreshes the page with the new data.
B. Trigger the global cartChange' event and then trigger changeMiniCart' event after the Add to Cart Action on the custom button.
C. Write a custom Remote Action to refresh the Mini Cart and refresh the Cart Line item count on the Cart Link in the header.
D. Trigger the global cartChange' event after the Add to Cart Action on the custom button.

Correct Answer: D
Section:
Explanation:
To have a customized experience for adding items to the cart and also update the mini cart module, the requirement can be fulfilled by triggering the global ''cartChange'' event after the Add to Cart Action on the custom button. This event will notify all the components that are listening to it that the cart has changed and they should refresh their data accordingly. The mini cart module is one of these components, so it will update its state based on the new cart data. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Events

QUESTION 3
A user wants to leverage a three columnlayout on a page. The user also wants to move the mini-cart widget from the right to the center column. How can this requirement be fulfilled?

A. Gross Layout Override
B. Subscriber Template
C. Page Include
D. HandleBar Template Override

Correct Answer: A
Section:
Explanation:

To leverage a three column layout on a page and move the mini-cart widget from the right to the center column, the requirement can be fulfilled by creating a Gross Layout Override. This is a custom Visualforce page that overrides the default layout of a page and allows changing its structure and content. The user can create a Gross Layout Override that uses a three column layout and places the mini-cart widget in the center column. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Gross Layout Overrides

QUESTION 4
What are the templating, Javascript, and CSS frameworks what the cloudcraze managed package leverages?

A. Angularjs, Backbonejs, and handlebarsjs
B. Bootstrap, Backbonejs, and handlebarsjs
C. Bootstrap, Angularjs, and Backbonejs
D. Angularjs, react.js, and handlebarsjs

Correct Answer: B
Section:
Explanation:
The templating, JavaScript, and CSS frameworks that the cloudcraze managed package leverages are Bootstrap, Backbone.js, and Handlebars.js. Bootstrap is a CSS framework that provides responsive design and layout components. Backbone.js is a JavaScript framework that provides models, views, collections, and events for building single-page applications. Handlebars.js is a templating engine that allows generating HTML from JSON data. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Front-End Development

QUESTION 5
Numerous flags when set, have a direct impact on the result set provided by the Global API's. Which conversion flag allows for sObjects to be returned from the Global API's when provided as a Boolean parameter with a value of true?

A. ccrz.ccAPISizing.SKIPTRZ
B. ccrz.ccAPISizing.SOBJECT
C. ccrz.ccAPI.SZ_SKIPTRZ
D. ccrz.ccAPI.SZ_SOBJECT

Correct Answer: D
Section:
Explanation:
The conversion flag that allows for sObjects to be returned from the Global API's when provided as a Boolean parameter with a value of true is ccrz.ccAPI.SZ_SOBJECT. This flag indicates that the API should return the raw sObjects instead of the transformed objects that are usually returned by the API. For example,ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_SOBJECT,true)will return the Product2 sObjects instead of the ccrz__E_Product__c objects. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Data Sizing Conventions

QUESTION 6
A query containing a subquery is executed. What is appended to the subquery name as part of its transformation by default in Salesforce B2B Commerce?

A. A subscriber-supplied token
B. '__ccrz'
C. The '*' symbol
D. The letter 'S'

Correct Answer: B
Section:

Explanation:
A query containing a subquery is executed. By default, in Salesforce B2B Commerce, ''__ccrz'' is appended to the subquery name as part of its transformation. This is done to avoid conflicts with the standard Salesforce fields and relationships. For example,SELECT Id, Name, (SELECT Id, Name FROM Contacts) FROM Accountwill be transformed toSELECT Id, Name, (SELECT Id, Name FROM Contacts__ccrz) FROM Account__ccrz. Salesforce Reference:B2B Commerce and D2C Commerce Developer Guide,Query Transformation

QUESTION 7
Salesforce B2B Commerce natively provides a robots.txt file, however, a customer can also create its own version.Which three scenarios are valid reasons for customer to create their own robots.txt file? (3 answers)

A. The customer wants to reference multiple storefront sitemap indexes in a single robots.txt file
B. The customer wants to reference a custom sitemap index.
C. The customer wants to have multiple robot.txt files in a single Salesforce Community.
D. The customer's store is not located at the rootof their domain.
E. robot.txt only works if there is one storefront in the org

Correct Answer: A, B, D
Section:
Explanation:
A customer can create its own robots.txt file for three valid reasons:
The customer wants to reference multiple storefront sitemap indexes in a single robots.txt file. This can be useful if the customer has multiple storefronts under the same domain and wants to provide a single entry point for search engines to crawl their sitemaps.
The customer wants to reference a custom sitemap index. This can be useful if the customer has created their own sitemap index that contains custom sitemaps or sitemaps from other sources.
The customer's store is not located at the root of their domain. This can be useful if the customer has their store under a subdirectory or a subdomain and wants to specify a different robots.txt file for their store than for their main domain. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Robots.txt File

QUESTION 8
Salesforce B2B leverages global API's for encapsulating business logic into blocks that can be extended and modified by subscribers. Which three statements are true regarding extending ccServiceProduct and exposing custom fields on the Product Detail
Page? (3 answers)

A. Override the getFieldsMap method and add subscriber specific code.
B. Ensure that any helper methods are defined as private andstatic only.
C. Create a global with sharing class that extends ccrz.ccServiceProduct.
D. Create a public with sharing class that extends ccrz.ccServiceProduct.
E. Override the fetch method and add your subscriber specific code here.

Correct Answer: A, C, E
Section:
Explanation:
To extend ccServiceProduct and expose custom fields on the Product Detail Page, three statements are true:
Override the getFieldsMap method and add subscriber specific code. This method returns a map of field names and field types for the product entity and its related entities. By overriding this method, the subscriber can add their custom fields to the map and make them available for the API.
Create a global with sharing class that extends ccrz.ccServiceProduct. This class will inherit all the methods and properties of the ccServiceProduct class and allow overriding or extending them. The class should be global and with sharing to ensure that it can be accessed by the API framework and respect the sharing rules.
Override the fetch method and add your subscriber specific code here. This method executes the query to fetch the product data and returns a result object. By overriding this method, the subscriber can modify the query or the result object to include their custom fields or logic. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Service Classes,ccServiceProduct Class

QUESTION 9
The sizing keys used in the Salesforce B2B Commerce Global APIs five distinct operations. What are three of these operations? (3 answers)

A. Refetch data (used on some Logic classes)
B. Return formats as Map<String, Object> or SObjects lists
C. Override static DAO classes and methods
D. Related Query to call (sub queries or direct queries)
E. Object type casting

Correct Answer: A, D, E
Section:
Explanation:
The sizing keys used in the Salesforce B2B Commerce Global APIs perform five distinct operations. Three of these operations are:
Refetch data (used on some Logic classes): This operation indicates that the data should be refetched from the database instead of using the cached data. For example,ccrz.ccServiceCart.getCart(ccrz.ccAPI.SZ_REFETCH)will refetch the cart data and refresh the cache.
Related Query to call (sub queries or direct queries): This operation indicates that the related entities should be retrieved by using sub queries or direct queries. For example,ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_SUBQUERY)will use sub queries to fetch the related entities for each product, whileccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_DIRECTQUERY)will use direct queries to fetch the related entities separately.
Object type casting: This operation indicates that the data should be cast to a specific object type. For example,ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_SOBJECT)will cast the data to sObjects instead of transformed objects. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Data Sizing Conventions

QUESTION 10
A user wants the pricing to reflect the price values stored in an external ERP during the checkoutflow. In what way can this requirement be satisfied?

A. Override the computePricingCart method in ccrz.cc_api_PriceAdjustoment and make the callout in this method.
B. None of the above
C. Override the computePricingReview method in ccrz.cc_CartExtension and make the callout in this method.
D. Override the computePricingCart methos in ccrz.cc_api_CartExtension and make the callout in this method.

Correct Answer: D
Section:
Explanation:
To reflect the price values stored in an external ERP during the checkout flow, the requirement can be satisfied by overriding the computePricingCart method in ccrz.cc_api_CartExtension and making the callout in this method. This method is responsible for computing the pricing for the cart and its line items. By overriding this method, the user can make a callout to the external ERP and update the pricing information accordingly. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Logic Classes,ccLogicCart Class

QUESTION 11
What are two guidelines for logging that are used within the core Salesforce
B2B Commerce product? (2 answers)

A. Items or data within computational intensive loops shouldbe logged.
B. The close method of ccrz.ccLog must be called at the end of the remote action.
C. No calls to ccrz.ccLog can be made before cc_CallContext.initRemoteContext is executed.
D. It is okay to log any data on the server that is already logged on the client side.

Correct Answer: B, C

Section:
Explanation:
Two guidelines for logging that are used within the core Salesforce B2B Commerce product are:
The close method of ccrz.ccLog must be called at the end of the remote action. This method will flush the log messages to the browser console or to a custom object, depending on the logging mode. If this method is not called, the log messages may not be displayed or saved properly.
No calls to ccrz.ccLog can be made before cc_CallContext.initRemoteContext is executed. This method will initialize the call context object, which contains information such as the current user, cart, storefront, and configuration settings. These information are required for logging, so calling ccrz.ccLog before initializing the call context will result in an error. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Logging

QUESTION 12
What is a best practice when passing query parameters from user interface to an apex controller?

A. Query parameters should be properly sanitized by using JSINHTMLENCODE within the VisualForce Page or Component.
B. String parameters should be trimmed using String.trim().
C. Query parameters should be passed only to Salesforce B2B Commerce classes that you are extending.
D. Query parameters should be stored on a backbone model prior to passing them to the server

Correct Answer: A
Section:
Explanation:
A best practice when passing query parameters from user interface to an apex controller is to query parameters should be properly sanitized by using JSINHTMLENCODE within the VisualForce Page or Component. This function will encode any special characters in the query parameters to prevent cross-site scripting (XSS) attacks or SOQL injection attacks. For example,ccrz.ccRemoteActions.getProducts('{!JSINHTMLENCODE(searchTerm)}')will encode the searchTerm parameter before passing it to the apex controller. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Security

QUESTION 13
What is a method to resolve if the current storefront customer is a Salesforce B2B Commerce guest user in an apex class?

A. ccrz.cc_CallContext.currUser.isGuest
B. ccrz.cc_CallContext.isGuest
C. UserInfo.getUserType()
D. ... UserType

Correct Answer: B
Section:
Explanation:
A method to resolve if the current storefront customer is a Salesforce B2B Commerce guest user in an apex class is to use ccrz.cc_CallContext.isGuest. This property will return true if the current user is a guest user, or false otherwise. For example,if(ccrz.cc_CallContext.isGuest){ // do something for guest user }will execute some logic only for guest users. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Call Context

QUESTION 14
What is a valid way of referencing the CC Cart Object whose API name is E_Cart__c in a SOQL query?

A. _Cart__c
B. c.E_Cart__c
C. ccrz__E_Cart__c
D. cloudcraze__E_Cart__c

Correct Answer: C
Section:
Explanation:
A valid way of referencing the CC Cart Object whose API name is E_Cart__c in a SOQL query is to use ccrz__E_Cart__c. This is the transformed name of the object that is used by the Salesforce B2B Commerce framework. All custom objects and fields that are part of the cloudcraze managed package have the prefix ccrz__ in their API names. For example,SELECT Id, Name FROM ccrz__E_Cart__cwill query the CC Cart Object records. Salesforce Reference:B2B Commerce and D2C Commerce Developer Guide,Query Transformation

QUESTION 15
What are three advantages of using ccLog over the Salesforce standard System.debug class? (3 answers)

A. There is no need to use string concatenation to easily tag log statements with a subject.
B. ccLog can debug syntax errors found in the JavaScript.
C. There is no need to create a User Trace Flag.
D. Append #ccLog=<Logging Token Name> to the end of the storefront URL in order to get logs in the inspector console.
E. There is no need to manually set a cookie to debug with the Site Guest User.

Correct Answer: A, D, E
Section:
Explanation:
Three advantages of using ccLog over the Salesforce standard System.debug class are:
There is no need to use string concatenation to easily tag log statements with a subject. ccLog allows passing a subject parameter to the log method, which will prepend the subject to the log message. For example,ccLog.log('This is a message', 'Subject')will log[Subject] This is a message.
There is no need to create a User Trace Flag. ccLog can be enabled by setting the value of CO.logToken to true in CCAdmin, which will activate logging for all users who access the storefront.
There is no need to manually set a cookie to debug with the Site Guest User. ccLog can be enabled for the Site Guest User by appending #ccLog=<Logging Token Name> to the end of the storefront URL in order to get logs in the inspector console. For example,https://my-storefront.com/#ccLog=debugwill enable logging for the Site Guest User with the debug level. Salesforce Reference:B2B Commerce and D2C Commerce Developer Guide,Logging

QUESTION 16
What are three ways to implement custom post Order processing? (3 answers)

A. Use a Salesforce workflow rule that executes when an Order record is created.
B. Extend cc_hk_invoice tohandle custom business logic post Order processing
C. Use cc_hk_Order.placeTarget to define a new Order Confirmation page which executes additional business logic.
D. Modify or add custom Cart formula fields to handle logic.
E. Use Process builder to implement business processes that execute when an Order record is created.

Correct Answer: B, C, E
Section:
Explanation:
Three ways to implement custom post Order processing are:
Extend cc_hk_invoice to handle custom business logic post Order processing. This hook allows modifying the invoice data or performing additional actions after an Order is placed. For example, the hook can send an email notification or update a custom field on the invoice record.
Use cc_hk_Order.placeTarget to define a new Order Confirmation page which executes additional business logic. This hook allows specifying a custom Visualforce page that will be displayed after an Order is placed. The custom page can include additional business logic or user interface elements.
Use Process Builder to implement business processes that execute when an Order record is created. Process Builder is a tool that allows creating workflows and actions based on criteria and conditions. For example, Process Builder can create a task, send an email, or update a record when an Order record is created. Salesforce Reference:B2B Commerce and D2C Commerce Developer Guide,Hooks, Process Builder

QUESTION 17
What are three ways to test the value of Page Label on any Salesforce B2B Commerce Community Page? (3 answers)

A. Access the source HTML for the page viathe browser developer tools.
B. Execute CCRZ.pagevars.pageLabels['PAGE_LABEL_NAME') in the JavaScript console.
C. Execute CCRZ.processPageLabelMap('PAGE_LABEL_NAME') in the JavaScript console.
D. Enable the 'display page label names' in cc admin.
E. Execute (('PAGE_LABEL_NAME')) in the JavaScript console

Correct Answer: B, C, D
Section:
Explanation:
Three ways to test the value of Page Label on any Salesforce B2B Commerce Community Page are:
Execute CCRZ.pagevars.pageLabels['PAGE_LABEL_NAME'] in the JavaScript console. This will return the value of the page label with the given name from the pagevars object, which contains all the page labels that are used on the page.
Execute CCRZ.processPageLabelMap('PAGE_LABEL_NAME') in the JavaScript console. This will return the value of the page label with the given name from the pageLabelMap object, which contains all the page labels that are defined in CCAdmin.
Enable the 'display page label names' in cc admin. This will display the name of each page label next to its value on the storefront pages, which can help identify and verify the page labels. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Page Labels

QUESTION 18
A developer is building a custom component in Lightning web components (LWC) that needs to fetch data from an API.
Which lifecycle hook should the developer use to make the API call?

A. connectedCallback
B. renderedCallback
C. errorCallback
D. disconnectedCallback

Correct Answer: A
Section:
Explanation:
To make an API call in a Lightning web component (LWC), a developer should use the connectedCallback lifecycle hook. The connectedCallback lifecycle hook is invoked when the component is inserted into the DOM. This is the ideal time to make an API call, as the component is ready to receive and display data. The developer can use the fetch API or a third-party library, such as axios, to make the API call and handle the response. The renderedCallback lifecycle hook is not a good choice for making an API call, as it is invoked every time the component is rendered or rerendered. This can cause unnecessary or repeated API calls and affect performance. The errorCallback lifecycle hook is not a good choice either, as it is invoked when an error occurs in the component or in one of its children. This is not related to making an API call, but rather to handling errors. The disconnectedCallback lifecycle hook is not a good choice either, as it is invoked when the component is removed from the DOM. This is not a suitable time to make an API call, as the component is no longer visible or active. Salesforce
Reference:Lightning Web Components Developer Guide: Lifecycle Hooks,Lightning Web Components Developer Guide: Call an Apex Method Imperatively

QUESTION 19
A developer is creating a component to implement a custom Terms and Conditions checkbox at checkout in the Aura Commerce template.
Which method should the developer implement on the Lightning web component to ensure the user accepts the terms and conditions?

A. ComponentValidity
B. Validate
C. SaveCheckout
D. CheckValidity

Correct Answer: B
Section:
Explanation:
To implement a custom Terms and Conditions checkbox at checkout in the Aura Commerce template, a developer should add a Desired Delivery Date input field during the checkout flow. The Desired Delivery Date input field allows the customer to enter a date when they want their order to be delivered. The developer can use the @api decorator to expose this field as a public property of the Lightning web component and bind it to the ccCheckoutOrder object. The developer can also use the @wire decorator to get the current cart object and use its properties, such as shipping address and shipping method, to calculate and display an estimated delivery date based on the desired delivery date. The developer can also add validation logic to ensure that the desired delivery date is valid and acceptable. Adding the Expected Delivery Date field to the order confirmation email is not a good solution, as it does not allow the customer to choose or see their delivery date before placing their order. Displaying the Expected Delivery Date on the order page with a Lightning web component is not a good solution either, as it does not allow the customer to enter or change their delivery date after placing their order. Configuring an email alert to the customer when the Expected Delivery Date changes is not a good solution either, as it does not provide a consistent or reliable way of informing the customer about their delivery date. Salesforce
Reference:B2B Commerce Developer Guide: Checkout Flow,B2B Commerce Developer Guide: Checkout Order Object,Lightning Web Components Developer Guide: Communicate with Properties

QUESTION 20
Northern Trail Outfitters (NTO) has a B2B Commerce store for its resellers. It has received many customer service calls involving questions about the delivery date of customer orders.
How should a developer expose delivery time estimates to NTO's customers in the storefront to reduce call volume?

A. Add the Expected Delivery Date field to the order confirmation email.
B. Add a Desired Delivery Date input field during the checkout flow.
C. Display the Expected Delivery Date on the order page with a Lightning web component.
D. Configure an email alert to the customer when the Expected Delivery Date changes.

Correct Answer: C
Section:
Explanation:
To expose delivery time estimates to NTO's customers in the storefront, a developer should display the Expected Delivery Date on the order page with a Lightning web component. The Expected Delivery Date is a custom field on the Order object that stores the date when the order is expected to be delivered to the customer. The developer can use the @wire decorator to get the current order object and use its properties, such as order number, status, total amount, and expected delivery date, to display them on the order page. The developer can also use Apex methods or third-party APIs to calculate and update the expected delivery date based on various factors, such as inventory availability, shipping method, shipping address, and carrier service level. Displaying the expected delivery date on the order page allows the customer to see their delivery time estimate at any time and reduce their need to call customer service. Adding the Expected Delivery Date field to the order confirmation email is not a good solution, as it does not allow the customer to see their delivery time estimate if they lose or delete their email. Adding a Desired Delivery Date input field during the checkout flow is not a good solution either, as it does not guarantee that the customer's desired delivery date will be met or reflect any changes in delivery time due to unforeseen circumstances. Configuring an email alert to the customer when the Expected Delivery Date changes is not a good solution either, as it can create confusion or frustration for the customer if they receive multiple or conflicting emails about their delivery date. Salesforce
Reference:B2B Commerce Developer Guide: Order Object, [B2B Commerce Developer Guide: Order Page], [Lightning Web Components Developer Guide: Call an Apex Method Imperatively]

QUESTION 21
Which template will correctly display the details message only when areDetailsVisible becomes true given the following code in a Lightning Web Component?

```
import { LightningElement } from 'lwc';
export default class HelloConditionalRendering extends LightningElement {
    areDetailsVisible = false;
    handleChange(event) {
        this.areDetailsVisible = event.target.checked;
    }
}
```

A)

```
<template if:true={areDetailsVisible}>
 <div class="slds-m-vertical_medium">
  These are the details!
 </div>
</template>
```

B)

```
<template if:areDetailsVisible={true}>
 <div class="slds-m-vertical_medium">
  These are the details!
 </div>
</template>
```

C)

```
<template if:true(areDetailsVisible)>
 <div class="slds-m-vertical_medium">
  These are the details!
 </div>
</template>
```

D)

```
<template if:{areDetailsVisible}=true>
 <div class="slds-m-vertical_medium">
  These are the details!
 </div>
</template>
```

A. Option A
B. Option B
C. Option C
D. Option D

Correct Answer: C
Section:
Explanation:
The template that will correctly display the details message only when areDetailsVisible becomes true given the following code in a Lightning Web Component is option C. Option C uses the if:true directive to conditionally render a template block based on the value of areDetailsVisible. If areDetailsVisible is true, the template block inside the <template if:true={areDetailsVisible}> tag will be rendered. Otherwise, it will be skipped. Option A is incorrect because it uses the if:false directive, which does the opposite of if:true. It renders the template block only when areDetailsVisible is false. Option B is incorrect because it uses an invalid syntax for the if directive. The if directive requires a colon (:) after the if keyword, not an equal sign (=). Option D is incorrect because it uses an invalid syntax for the template tag. The template tag requires a closing tag (</template>), not a self-closing tag (<template/>). Salesforce
Reference:Lightning Web Components Developer Guide: Conditional Rendering,Lightning Web Components Developer Guide: Template Syntax

QUESTION 22
What are two advantages of using Lightning Data Service?

A. Communicates with other components
B. Converts between different data formats
C. Combines and de-duplicates server calls
D. Loads record data progressively

Correct Answer: C, D
Section:
Explanation:
Two advantages of using Lightning Data Service are that it combines and de-duplicates server calls and that it loads record data progressively. Lightning Data Service is a service that provides access to Salesforce data and metadata in Lightning web components. It optimizes performance and minimizes server round trips by caching data on the client side and sharing data across components. It also combines and de-duplicates server calls by batching requests for the same record or object and returning a single response. It also loads record data progressively by returning available cached data first and then fetching updated data from the server asynchronously. Communicating with other components and converting between different data formats are not advantages of using Lightning Data Service, as they are not related to its functionality or features. Salesforce Reference:Lightning Web Components Developer Guide: Lightning Data Service,Lightning Web Components Developer Guide: Work with Salesforce Data

QUESTION 23
What is likely to happen if a developer leaves debug mode turned on in an environment?

A. The performance of the org will become slower each day
B. The user will begin getting JavaScript limit exceptions
C. The org will turn off debug mode after 72 hours
D. A banner will be displayed to the user indicating that the org is in debug mode

Correct Answer: B
Section:
Explanation:
If a developer leaves debug mode turned on in an environment, the user will begin getting JavaScript limit exceptions. Debug mode is a setting that enables more detailed logging and error reporting for Lightning web components. However, it also increases the size and complexity of the JavaScript code that is delivered to the browser, which can cause performance issues and JavaScript limit exceptions. The JavaScript limit exceptions are errors that occur when the browser reaches its maximum capacity for executing JavaScript code, such as memory heap size or script execution time. The performance of the org will not become slower each day, as debug mode only affects the client-side performance, not the server-side performance. The org will not turn off debug mode after 72 hours, as debug mode is a persistent setting that can only be changed manually by an administrator. A banner will not be displayed to the user indicating that the org is in debug mode, as debug mode is a transparent setting that does not affect the user interface. Salesforce Reference: Lightning Web Components Developer Guide: Debug Your Code, Lightning Web Components Developer Guide: JavaScript Limit Exceptions

QUESTION 24
A developer has created a custom Lightning web component to display on the Product Detail page in the store. When the developer goes to add the component to the page in Experience Builder, it is missing from the list of custom components.
Which XML fragment should the developer include in the component's configuration XML file to ensure the custom component is available to add to the page?
A)

```
<isExposed>true</isExposedTrue>
<targets>
<target>lightningCommunity_Page</target>
<targets>
```

B)

```
<builder>ExperienceCloud</builder>
<target>RecordPage</target>
```

C)

```
<isExposed target="ExperienceCloud">
<pageType>RecordPage</pageType>
</isExposed>
```

D)

```
<isAvailable>true</isAvailable>
<targets>lightningCommunity_RecordPage<targets>
```

A. Option A
B. Option B
C. Option C
D. Option D

Correct Answer: B
Section:
Explanation:
The XML fragment that the developer should include in the component's configuration XML file to ensure the custom component is available to add to the page is option B. Option B uses the <targets> tag to specify where the component can be used in an app. The <targets> tag contains a list of <target> tags, each of which defines a valid target for the component. The value of the <target> tag for the Product Detail page in the store is lightningCommunity__RecordPage, which means that the component can be used on any record page in a Lightning community. Option A is incorrect because it uses an invalid value for the <target> tag. There is no such target as lightningCommunity__ProductDetailPage. Option C is incorrect because it uses an invalid syntax for the <targets> tag. The <targets> tag should not have any attributes, such as isAvailable. Option D is incorrect because it uses an invalid syntax for the <target> tag. The <target> tag should not be self-closing, but rather have a closing tag (</target>). Salesforce
Reference:Lightning Web Components Developer Guide: Configure Components for Lightning Communities,Lightning Web Components Developer Guide: Configure Components for Different Pages and Apps

QUESTION 25
Which two statements are accurate about the Cart Item with a Type of Charge?

A. It is created with the Cart Delivery Group Method after the shipping integration
B. It is created with the Cart Delivery Group Method after the freight integration
C. It is linked directly to a Cart Id
D. It is linked directly to a Catalog Id

Correct Answer: C, D
Section:
Explanation:
Two statements that are accurate about the Cart Item with a Type of Charge are that it is linked directly to a Cart Id and that it is linked directly to a Catalog Id. A Cart Item with a Type of Charge is a special type of Cart Item that represents an additional charge or fee that is applied to a Cart, such as shipping, handling, or tax. A Cart Item with a Type of Charge is linked directly to a Cart Id, which means that it belongs to a specific Cart and can be retrieved or updated along with other Cart Items. A Cart Item with a Type of Charge is also linked directly to a Catalog Id, which means that it references a specific Catalog that contains the products and prices for the store. A Cart Item with a Type of Charge is not created with the Cart Delivery Group Method after the shipping integration or after the freight integration, as these are not related to the creation of Cart Items. The Cart Delivery Group Method is a method that determines how products are grouped into delivery groups based on their shipping methods and addresses. The shipping integration and the freight integration are integrations that calculate and apply shipping costs and freight charges to a Cart or an Order. Salesforce
Reference:B2B Commerce Developer Guide: Cart Item Object, B2B Commerce Developer Guide: Shipping Integration, B2B Commerce Developer Guide: Freight Integration

QUESTION 26
When a developer configures a tax integration for a store, what happens to the previously calculated tax entries during the checkout flow?

A. Ignored during recalculation
B. Saved prior to recalculation
C. Deleted from the Cart
D. Modified with the new tax calculation

Correct Answer: C
Section:
Explanation:
When a developer configures a tax integration for a store, the previously calculated tax entries during the checkout flow are deleted from the Cart. A tax integration is an integration that calculates and applies tax rates and amounts to a Cart or an Order based on various factors, such as product type, price, quantity, location, and tax rules. A tax integration can use either an external tax service provider or custom Apex code to perform the tax calculation. When a developer configures a tax integration for a store, any existing tax entries in the Cart are deleted before calling the tax integration service or method. This ensures that the tax calculation is accurate and up-to-date based on the current state of the Cart and avoids any conflicts or inconsistencies with previous tax entries. The previously calculated tax entries are not ignored during recalculation, saved prior to recalculation, or modified with the new tax calculation, as these are not valid actions for handling existing tax entries. Salesforce
Reference: B2B Commerce Developer Guide: Tax Integration, B2B Commerce Developer Guide: Tax Calculation Flow

QUESTION 27
Which three actions must a developer take, in a B2B Commerce store, to accept credit card payments using a client's chosen payment provider?

A. Create a named credential for authentication with the payment provider.
B. Create a RegisteredExternalService record for the custom payment provider class.
C. Create an Apex class that implements the sfdc_checkout.PaymentGatewayAdapter
D. Create a PaymentProviderGateway record for the custom payment provider class.
E. Create an Apex class that implements the commercepayments.PaymentGatewayAdapter.

Correct Answer: A, C, D
Section:
Explanation:
Three actions that a developer must take, in a B2B Commerce store, to accept credit card payments using a client's chosen payment provider are: create a named credential for authentication with the payment provider, create an Apex class that implements the sfdc_checkout.PaymentGatewayAdapter interface, and create a PaymentProviderGateway record for the custom payment provider class. Creating a named credential for authentication with the payment provider allows the developer to securely store and manage authentication information, such as username, password, token, or certificate, for connecting to the payment provider's API or service. Creating an Apex class that implements the sfdc_checkout.PaymentGatewayAdapter interface allows the developer to define custom logic for processing credit card payments using the payment provider's API or service. The interface provides methods for validating credit card information, authorizing payments, capturing payments, voiding payments, and refunding payments. Creating a PaymentProviderGateway record for the custom payment provider class allows the developer to register the custom payment provider class as a payment gateway in the store and associate it with a payment method. Creating a RegisteredExternalService record for the custom payment provider class is not a required action, as this is only used for registering external services that are not related to payment processing, such as tax or shipping services. Creating an Apex class that implements the commercepayments.PaymentGatewayAdapter interface is not a required action either, as this is only used for D2C Commerce stores, not B2B Commerce stores. Salesforce
Reference: B2B Commerce Developer Guide: Payment Integration, B2B Commerce Developer Guide: Payment Gateway Adapter Interface, B2B Commerce Developer Guide: Payment Provider Gateway Object

QUESTION 28
Although Salesforce B2B Commerce and Salesforce recommend against using
'without sharing classes' whenever possible, sometimes it is unavoidable. Which three items will open up a major security hole? (3 answers)

A. Executing dynamic SOQL inside a without sharing class with a bind variable fromPageReference.getParameters().
B. Executing dynamic SOQL inside a without sharing class with a bind variable from theUserInfo class.
C. Executing dynamic SOQL inside a without sharing class with a bind variable fromPageReference.getCookies().
D. Executing dynamic SOQL inside a without sharing class with a bind variable fromcc_RemoteActionContentex class.

E. Executing dynamic SOQL inside a without sharing class with a bind variable fromccAPI.CURRENT_VERSION.

Correct Answer: A, C, D
Section:
Explanation:
Executing dynamic SOQL inside a without sharing class with a bind variable from PageReference.getParameters(), PageReference.getCookies(), or cc_RemoteActionContext class will open up a major security hole because these sources of input are not sanitized and can be manipulated by malicious users to inject SOQL queries that bypass the sharing rules and access data that they are not supposed to see. For example, a user can modify the URL parameters or cookies to include a SOQL query that returns sensitive data from the database. To prevent this, it is recommended to use static SOQL or escape the bind variables before executing dynamic SOQL.

QUESTION 29
The ccrz.cc_hk_UserInterface apex class, HTMLHead Include Begin and HTML Head Include End Cloudcraze Page Include sections allow additional content to be added to the HTML <head> tag. What are two reasons that is it preferred to use the ccrz.cc_hk_UserInterface extension over the Cloudcraze Page Include sections? (2 answers)

A. Salesforce apex:include is wrapped in tags.
B. HTML does not support tags inside the <head> C Salesforce apex:include is wrapped in tags.
C. HTML does not support tags inside the <head>

Correct Answer: A
Section:
Explanation:
The ccrz.cc_hk_UserInterface apex class is preferred over the HTMLHead Include Begin and HTML Head Include End Cloudcraze Page Include sections because Salesforce apex:include is wrapped in tags, which are not valid inside the HTML <head> tag. This can cause rendering issues or unexpected behavior in some browsers. The ccrz.cc_hk_UserInterface extension allows adding content to the HTML <head> tag without using apex:include.

QUESTION 30
The ccUtil apex class in Salesforce B2B Commerce provides numerous utility functions that can be leveraged in subscriber classes.
What are two ways to check the input or return data of the Global API's? (2 answers)

A. ccrz.ccUtil.isNotEmpty(Map<String, Object>) andccrz.ccUtil.isNotEmpty(List<Object>)
B. ccrz.ccUtil.isNotValid(Map<String, Object>) andccrz.ccUtil.isNotValid(List<Object>)
C. ccrz.ccUtil.isValid(Map<String, Object>) and ccrz.ccUtil.isValid(List<Object>)
D. ccrz.ccUtil.isEmpty(Map<String, Object>) and ccrz.ccUtil.isEmpty(List<Object>)

Correct Answer: A, D
Section:
Explanation:
The ccUtil apex class provides two methods to check the input or return data of the Global API's: ccrz.ccUtil.isNotEmpty(Map<String, Object>) and ccrz.ccUtil.isEmpty(Map<String, Object>). These methods return true if the map is not null and contains at least one entry, or if the map is null or empty, respectively. Similarly, ccrz.ccUtil.isNotEmpty(List<Object>) and ccrz.ccUtil.isEmpty(List<Object>) return true if the list is not null and contains at least one element, or if the list is null or empty, respectively. These methods are useful for validating the input parameters or the output results of the Global API's.

QUESTION 31
The ccUtil apex class in Salesforce B2B Commerce provides numerous utility functions that can be leveraged in subscriber classes. Which command will return the value in the given Map if found or a default value in the event that the Map is null,
empty, or an object is not found for that key?

A. ccrz.ccUtil.defv (Map<String.Object> mp, String key , Object ob)
B. ccrz.ccUtil.defVal (Map<String.Object> mp, String key, Object ob)
C. ccrz.ccUtil.... (Map<String.Object> mp, String key, Object ob)

D. ccrz.ccUtil.defaultValue(Map<String.Object> mp, String key , Object ob)

Correct Answer: B
Section:
Explanation:
The ccrz.ccUtil.defVal (Map<String.Object> mp, String key, Object ob) method will return the value in the given Map if found or a default value in the event that the Map is null, empty, or an object is not found for that key. This method is useful for providing fallback values for configuration settings or input parameters that may be missing or invalid. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,ccUtil Class

QUESTION 32
A configuration value, CO.NewOrder, is set to TRUE. What is one way of preventing anexisting payment page from being shown on the checkout payment page?

A. Delete the Visualforce page from the code base.
B. Remove the value matching the page name from the pmt.whitelist configurationsetting, then rebuild and activate a new Configuration cache
C. Remove the payment type associated with the payment page from CO.pmts, thenrebuild and activate a new cache.
D. Override the front end template and modify the way the embedded payment page getsloaded from the payment list configuration.

Correct Answer: B
Section:
Explanation:
This approach effectively removes the payment page from the list of allowed pages, ensuring itis not displayed during the checkout process.
Thepmt.whitelistconfiguration setting in Salesforce B2B Commerce is used to manage theVisualforce pages that support all payment types on the storefront1. If you want to prevent anexisting payment page from being shown on the checkout payment page, one way to do it is toremove the value matching the page name from thepmt.whitelistconfiguration setting.Afterdoing this, you would need to rebuild and activate a new Configuration cache for the changesto take effect1.Please note that this information is based on the Salesforce B2B Commercedocumentation and best practices1.

QUESTION 33
A Developer created a custom field that a project wants to expose on a given page.
How does the Developer ensure that the field is available to display on a given page?

A. Override the Service Class that the page uses and update the ServiceManagementin CCAdmin for the given storefront to use this new Service Class.
B. Override the Logic Class that the page uses and update the Service Management inCCAdmin for the given storefront to use this new Service Class
C. Create a new Service Classthat the page uses and update the Service Managementin CCAdmin for the given storefront to use this new Service Class
D. Create a new Logic Class that the page uses and update the Service Managementin CCAdmin for the given storefront to use this new Service Class

Correct Answer: A
Section:
Explanation:
To ensure that a custom field is available to display on a given page, the Developer needs to override the Service Class that the page uses and update the Service Management in CCAdmin for the given storefront to use this new Service Class. The Service Class is responsible for retrieving and setting data for a given page. The Logic Class is responsible for implementing business logic and validation rules for a given page. Creating a new Service Class or Logic Class is not necessary, as overriding an existing one can achieve the same result. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Service Classes,Logic Classes

QUESTION 34
A developer is trying to troubleshoot why a field is not displaying on the Product Detail Page. What should be typed in the Developer Tools Console in the browser to view the fields available for the Product Detail Page?

A. CCRZ.productSearchView
B. CCRZ.cartView
C. CCRZ.productDetailModel

D. CCRZ.productDetailView

Correct Answer: C
Section:
Explanation:
To view the fields available for the Product Detail Page, the developer should type CCRZ.productDetailModel in the Developer Tools Console in the browser. This will display the product detail model object, which contains the product data and attributes that are rendered on the page. The other options are either not valid or not relevant for the Product Detail Page.

QUESTION 35
For which two reasons is it preferable to extend the Salesforce B2B Commerce remote invocation object instead of using the standard Salesforce remote action invocation manager (2 answers)

A. A standard remote action will not have access to Salesforce B2B Commerce objects.
B. The APEX method called by the remote action will be passed as a Salesforce B2B Commerce context object.
C. Salesforce B2B Commerce includes do not support standard SalesForce remote actions.
D. The Salesforce B2B Commerce logger cannot be utilized in standard remote actions

Correct Answer: B, D
Section:
Explanation:
It is preferable to extend the Salesforce B2B Commerce remote invocation object instead of using the standard Salesforce remote action invocation manager for two reasons:
The APEX method called by the remote action will be passed as a Salesforce B2B Commerce context object, which contains useful information such as the current user, cart, storefront, and configuration settings. This can simplify the development and testing of the remote action.
The Salesforce B2B Commerce logger can be utilized in the remote action, which allows logging messages and errors to the debug log or to a custom object. This can facilitate debugging and troubleshooting of the remote action.

QUESTION 36
How are variables bound when services use the ccSercviceDao classto execute queries?

A. Global variables
B. Apex local variables
C. String substitution
D. Apex class variables

Correct Answer: C
Section:
Explanation:
When services use the ccServiceDao class to execute queries, variables are bound by string substitution. This means that the query string contains placeholders for variables that are replaced by their values at runtime. For example,ccrz.ccServiceDao.getQuery('SELECT Id FROM Account WHERE Name = :name')will replace:namewith the value of thenamevariable.

QUESTION 37
How are version related upgrades passed on to subscriber API extensions/overrides?

A. APIs callback with specific versions specified; the user must know which version number to use.
B. Copy and paste of specific code is 'built-in'
C. Extensions and overridden APIs don't support-related upgrades.
D. The 'delegate' allows inherited method calls access to the most recentlyspecified service version

Correct Answer: D

Section:

QUESTION 38
How can the display of CC Menu Items be customized for different users?

A. cc_hk_Category extension to pre-process which category items are cached as menu items
B. cc_hk_Menu extension to post-process any cached menu items
C. cc_hk_Menu extension to pre-process which menu items are cached
D. cc_hk_Category extension to post-process any cached menu items

Correct Answer: B
Section:
Explanation:
The display of CC Menu Items can be customized for different users by using the cc_hk_Menu extension to post-process any cached menu items. This extension allows modifying the menu items based on the user context, such as the user role, account, or cart. For example, the extension can hide or show certain menu items based on the user's permissions or preferences.

QUESTION 39
How does a project implement the process to persist payment information datain the >Checkout flow for Salesforce B2B Commerce version 4.2 and beyond?

A. Trigger a remote action when the process payment button is selected to capture the payment.
B. Trigger a remote action to store the payment information in the URL query parameters.
C. Trigger the processPayment event and pass in the payment information object as an argument.
D. Trigger the externalprocessedPayment and pass in the payment information object as an argument.

Correct Answer: C
Section:
Explanation:
To persist payment information data in the Checkout flow for Salesforce B2B Commerce version 4.2 and beyond, the project needs to trigger the processPayment event and pass in the payment information object as an argument. This event will invoke the processPayment method of the ccServicePayment class, which will validate and process the payment information and return a payment result object. The payment result object will contain the status and details of the payment transaction.

QUESTION 40
How do the REST APIs in Salesforce B2B Commerce support pass-through parameter handling

A. An exception is generated for unknown API keys
B. Parameters are passed through the service handlers
C. Parameters are filtered out before the request is processed
D. Parameters are separated, but unused

Correct Answer: B
Section:
Explanation:
The REST APIs in Salesforce B2B Commerce support pass-through parameter handling by passing parameters through the service handlers. This means that any parameters that are not recognized by the REST API framework will be passed to the service handler class that implements the API logic. The service handler class can then use these parameters for custom logic or validation.

QUESTION 41
How is a price group dynamically set?

A. By overriding the ccLogicProductPrice class
B. By using contract pricing
C. By extending the ccApiPriceList API
D. By extending the cc_hk_priceing hook

Correct Answer: D
Section:
Explanation:
A price group can be dynamically set by extending the cc_hk_pricing hook. This hook allows modifying the price group based on various factors, such as the user, cart, product, or storefront. For example, the hook can apply a different price group for a specific product category or for a specific user segment.

QUESTION 42
Inwhich three different ways can a theme be enabled in
Salesforce B2B Commerce? (3 answers)

A. A Storefront setting
B. An Account Group field value
C. A per user setting
D. Account
E. Dynamically through a hook

Correct Answer: A, B, E
Section:
Explanation:
A theme can be enabled in Salesforce B2B Commerce in three different ways:
A Storefront setting: The theme can be specified in the Storefront Configuration settings in CCAdmin. This will apply the theme to all users who access the storefront.
An Account Group field value: The theme can be specified in the Theme field of an Account Group record in Salesforce. This will apply the theme to all users who belong to that account group.
Dynamically through a hook: The theme can be specified dynamically by extending the cc_hk_theme hook. This hook allows changing the theme based on various factors, such as the user, cart, product, or storefront. For example, the hook can apply a different theme for a specific product category or for a specific user segment.

QUESTION 43
In which threeways can Salesforce B2B Commerce API sizing blocks support multiple API sizing requests? (3 answers)

A. When different entities are specified in the method invocation.
B. The sizing block is not removed.
C. SZ_ASSC is used.
D. The sizing block is removedafter the first handler.
E. SZ_ASSC is not used.

Correct Answer: A, C, E
Section:
Explanation:
Salesforce B2B Commerce API sizing blocks can support multiple API sizing requests in three ways:
When different entities are specified in the method invocation. For example,ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_M, ccrz.ccAPI.SZ_L)will use the SZ_M sizing block for the product entity and the SZ_L sizing block for the related entities.
SZ_ASSC is used. This flag indicates that the associated entities should use the same sizing block as the primary entity. For example,ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_M, ccrz.ccAPI.SZ_ASSC)will use the SZ_M sizing block for both the product entity and the related entities.
SZ_ASSC is not used. This flag indicates that the associated entities should use the default sizing block, which is SZ_M, unless otherwise specified. For example,ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_M)will use the

SZ_M sizing block for the product entity and the SZ_M sizing block for the related entities. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Data Sizing Conventions

QUESTION 44
In which three ways should useful debugging information in Salesforce B2B Commerce implementation be garnered? (3 answers) A) Enabling the logging token via

A. Admin andsubsequently inspecting the logs via the browser console.
B. Logging a case with Salesforce support to enable advanced debugging options.
C. Enabling debugging options for the current user and visually inspecting the Salesforce debug logs.
D. Placing a System.debug() statement anywhere in the class being debugged.
E. Logging into the community as a system administrator to identify any potential permissions or Visualforce exceptions.

Correct Answer: A, C, E
Section:
Explanation:
Useful debugging information in Salesforce B2B Commerce implementation can be garnered in three ways:
Enabling the logging token via Admin and subsequently inspecting the logs via the browser console. This will enable logging messages and errors to the browser console, which can be viewed by opening the Developer Tools in the browser. The logging token can be enabled by setting the value of CO.logToken to true in CCAdmin.
Enabling debugging options for the current user and visually inspecting the Salesforce debug logs. This will enable logging messages and errors to the Salesforce debug logs, which can be viewed by opening the Debug Logs page in Salesforce Setup. The debugging options can be enabled by creating a Debug Level and a Trace Flag for the current user in Salesforce Setup.
Logging into the community as a system administrator to identify any potential permissions or Visualforce exceptions. This will allow viewing any errors or warnings that may occur on the community pages due to insufficient permissions or Visualforce issues. The system administrator can also access CCAdmin and other tools from within the community. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Logging, Debug Your Code

QUESTION 45
A new payment type for the Checkout flow has been implemented. Which three descriptors follow best practice for possible configuration metadata are needed to enable a flow? (3 answers)

A. *.pay
B. Cart
C. Checkout
D. *.Edit
E. *.New

Correct Answer: A, D, E
Section:
Explanation:
To enable a new payment type for the Checkout flow, three possible configuration metadata descriptors are needed:
*.pay: This descriptor defines the payment type name, label, description, icon, and handler class. For example,CO.pmts.CreditCard.paydefines the payment type for credit card payments.
*.Edit: This descriptor defines the Visualforce page that is used to edit or update an existing payment of this type. For example,CO.pmts.CreditCard.Editdefines the page that allows editing a credit card payment.
*.New: This descriptor defines the Visualforce page that is used to create a new payment of this type. For example,CO.pmts.CreditCard.Newdefines the page that allows creating a new credit card payment. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Payment Configuration Settings

QUESTION 46
Numerous flags ... have a directimpact on the result set provided by the Global
API's. What Global API Data-Sizing convention flag prevents an API request from propagating to further requests when provided as a Boolean parameter with a value of true?

A. ccrz.ccAPI.SZ_REL
B. ccrz.ccAPI.SZ_ASSC

C. ccrz.ccAPISizing.ASSC
D. ccrz.ccAPISizing.REL

Correct Answer: B
Section:
Explanation:
The Global API Data-Sizing convention flag that prevents an API request from propagating to further requests when provided as a Boolean parameter with a value of true is ccrz.ccAPI.SZ_ASSC. This flag indicates that only one API request should be executed with the specified sizing block, and any subsequent requests should use their own default sizing blocks. For example,ccrz.ccServiceCart.getCart(ccrz.ccAPI.SZ_L,true)will use the SZ_L sizing block for retrieving the cart data, but any other requests that are triggered by this method will use their own default sizing blocks. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Data Sizing Conventions

QUESTION 47
Numerous flags, when set, have a direct impact on the result set provided by the Global API's. What is the default Global API DataSizing convention flag that is used by theAPI's unless otherwise specified?

A. CCRZ.ccPAI.SZ_XL
B. CCRZ.ccPAI.SZ_M
C. CCRZ.ccPAI.SZ_L
D. CCRZ.ccPAI.SZ_S

Correct Answer: B
Section:
Explanation:
The default Global API Data-Sizing convention flag that is used by the API's unless otherwise specified is CCRZ.ccAPI.SZ_M. This flag indicates that the medium sizing block should be used for retrieving data, which includes the most commonly used fields and related entities. For example,ccrz.ccServiceProduct.getProducts()will use the SZ_M sizing block by default, unless another sizing block is specified as a parameter. Salesforce
Reference:B2B Commerce and D2C Commerce Developer Guide,Data Sizing Conventions

QUESTION 48
A developer is trying to integrate a new shipping provider to use during checkout in a storefront Which two steps must the developer take to make an integration available for selection?

A. Create a RegisteredExternalService record using Workbench.
B. Create an Apex class that uses the integration framework.
C. Modify the StoreIntegratedService to map to an Apex class ID using Workbench.
D. Enter the integration class name and version in the store administration.

Correct Answer: A, B
Section:
Explanation:
To make an integration available for selection, a developer must create a RegisteredExternalService record using Workbench and create an Apex class that uses the integration framework. Creating a RegisteredExternalService record using Workbench allows the developer to register their custom integration class as an external service in Salesforce B2B Commerce. The RegisteredExternalService record contains information such as the class name, version, display name, description, and category of the integration class. The category determines where and how the integration class can be used in B2B Commerce, such as ShippingService or TaxService. Creating an Apex class that uses the integration framework allows the developer to define custom logic for integrating with an external service provider's API or service. The integration framework provides interfaces and classes for various types of integrations, such as shipping, tax, payment, inventory, and freight. The developer can implement these interfaces and classes in their custom Apex class and override their methods with their own logic. Modifying the StoreIntegratedService to map to an Apex class ID using Workbench is not a required step for making an integration available for selection, as it is only used for registering internal services that are provided by Salesforce B2B Commerce out-of-the-box. Entering the integration class name and version in store administration is not a required step either, as it is only used for selecting an existing integration class that has already been registered as an external service. Salesforce
Reference: [B2B Commerce Developer Guide: Integration Framework], [B2B Commerce Developer Guide: RegisteredExternalService Object]

QUESTION 49

While in the process of gathering requirements from a customer about how they would like to set up their net new storefront checkout experience, a consultant learns that the customer needs the ability to add new shipping and billing addresses during checkout.
Which approach should a developer take to meet this requirement?

A. Create a new shipping address checkout subflow that utilizes the Buyer Managed Contact Point Addresses component.
B. Enable Buyer Managed Contact Point Addresses within the Shipping Address standard component in the Checkout subflow.
C. Enable Buyer Managed Contact Point Addresses within Commerce Administration.
D. Create a Lightning web component that enables this functionality and replaces the current shipping address screen within the Checkout subflow.

Correct Answer: B
Section:
Explanation:
To enable the ability to add new shipping and billing addresses during checkout, a developer should enable Buyer Managed Contact Point Addresses within the Shipping Address standard component in the Checkout subflow. The Buyer Managed Contact Point Addresses is a feature that allows customers to add, edit, or delete their shipping and billing addresses during checkout. The developer can enable this feature by setting the buyerManagedContactPointAddressesEnabled attribute to true in the Shipping Address standard component in the Checkout subflow. The Shipping Address standard component is a component that displays and collects the shipping address information for the cart or order. The Checkout subflow is a subflow that defines the steps and components for the checkout process in the storefront. Creating a new shipping address checkout subflow that utilizes the Buyer Managed Contact Point Addresses component is not a valid way to enable this feature, as there is no such component as Buyer Managed Contact Point Addresses. Enabling Buyer Managed Contact Point Addresses within Commerce Administration is not a valid way either, as this feature is not configurable in Commerce Administration. Creating a Lightning web component that enables this functionality and replaces the current shipping address screen within the Checkout subflow is not a valid way either, as this is unnecessary and complicated when there is already a standard component that supports this feature. Salesforce
Reference: [B2B Commerce Developer Guide: Buyer Managed Contact Point Addresses], [B2B Commerce Developer Guide: Shipping Address Component], [B2B Commerce Developer Guide: Checkout Subflow]

QUESTION 50
What is the fastest route to setting up a B2B Commerce Store as a developer?

A. Set up B2B Commerce on Lightning Experience manually
B. Create a new store in the Commerce app
C. Import a previously exported store archive
D. Use sfdx setup scripts

Correct Answer: D
Section:
Explanation:
The fastest route to setting up a B2B Commerce store as a developer is to use sfdx setup scripts. Sfdx setup scripts are scripts that use Salesforce CLI commands to automate the creation and configuration of a B2B Commerce store. The scripts can perform tasks such as creating scratch orgs, installing packages, importing data, assigning permissions, and deploying code. The scripts can save time and effort for developers who need to set up a B2B Commerce store quickly and easily. Setting up B2B Commerce on Lightning Experience manually is not the fastest route to setting up a B2B Commerce store, as it involves many steps and actions that can be tedious and error-prone. Creating a new store in the Commerce app is not the fastest route either, as it also requires manual configuration and customization of various settings and features. Importing a previously exported store archive is not the fastest route either, as it depends on the availability and quality of the store archive and may not reflect the latest changes or updates. Salesforce
Reference: [B2B Commerce Developer Guide: Set Up Your Development Environment], [B2B Commerce Developer Guide: Create Your Store]

QUESTION 51
Which two behaviors does a target value of lightning__FlowScreen in metadata allow for a Lightning web component?

A. It allows the Lightning web component to replace standard functionality in flows and subflows
B. It allows the Lightning Web component to be dragged onto a page in Lightning AppBuilder
C. It allows the Lightning web component to be used in guided user experiences to gather input
D. It automatically generates configuration properties for the Lightning web component

Correct Answer: C, D

Section:
Explanation:
A target value of lightning__FlowScreen in metadata allows for a Lightning web component to be used in guided user experiences to gather input and to automatically generate configuration properties for the Lightning web component. The lightning__FlowScreen target value specifies that the component can be used on a flow screen, which is a type of flow element that displays information and collects user input in a flow. A flow is a guided user experience that can automate business processes and guide users through screens, actions, and decisions. By using the lightning__FlowScreen target value, the developer can expose their Lightning web component as a custom screen component that can be added to any flow screen. The developer can also use the @api decorator to expose public properties of their Lightning web component as configuration properties that can be set in Flow Builder. Flow Builder is a tool that allows the developer to create and modify flows using a drag-and-drop interface. A target value of lightning__FlowScreen does not allow the Lightning web component to replace standard functionality in flows and subflows or to be dragged onto a page in Lightning App Builder, as these are not related to the flow screen target value. Salesforce
Reference:Lightning Web Components Developer Guide: Configure Components for Flows,Lightning Web Components Developer Guide: Communicate with Flow,Lightning Flow Developer Guide: Screen Components Basics

QUESTION 52
How can a developer introduce new screen behavior in a checkout flow step?

A. Modify the property mappings in checkoutSteps.xml
B. Edit the default subflow directly
C. Clone the appropriate subflow and replace the Lightning web components in it
D. Adjust next-state in previous subflow configuration

Correct Answer: C
Section:
Explanation:
To introduce new screen behavior in a checkout flow step, a developer should clone the appropriate subflow and replace the Lightning web components in it. A subflow is a reusable flow that can be invoked from another flow as an element. A checkout flow is a flow that defines the steps and components for the checkout process in the storefront. A checkout flow consists of several subflows, each of which corresponds to a checkout step, such as Cart, Shipping Address, Payment, or Order Summary. To introduce new screen behavior in a checkout flow step, a developer should clone the subflow that matches the checkout step they want to modify and replace the Lightning web components in the cloned subflow with their custom components. The developer can then update the checkoutSteps.xml file to point to the cloned subflow instead of the original subflow. Modifying the property mappings in checkoutSteps.xml is not a valid way to introduce new screen behavior in a checkout flow step, as it only affects how data is passed between subflows, not how screens are displayed or rendered. Editing the default subflow directly is not a valid way either, as it can cause conflicts or errors with other stores or environments that use the same default subflow. Adjusting next-state in previous subflow configuration is not a valid way either, as it only affects how subflows are transitioned or navigated, not how screens are displayed or rendered. Salesforce
Reference: B2B Commerce Developer Guide: Checkout Flow, B2B Commerce Developer Guide: Checkout Subflows, B2B Commerce Developer Guide: Customize Checkout Steps

QUESTION 53
While working on a commerce rollout, a developer needs to update the checkout process so that buyers can purchase with one of the below payment types.
* Credit Card
* Purchase Order
* Contract Now & Pay Later
Additionally, the developer needs to show only Purchase Order and Contract Now & Pay Later if a custom checkbox field on the account is checked.
How should the developer meet these requirements?

A. Create a custom Lightning web component that can be used with the standard payment component. Use a publish-<g, subscribe (pub-sub) model to listen to events from the standard component to determine which additional payment options should be shown.
B. Create a custom Lightning web component for the checkout flow that has all the options available. Within that component, pull data from the account to determine which options to show.
C. Modify the standard payment component settings in the checkout screen flow and add the new payment method. Use the component visibility feature in screen flows to fulfill the account-based payment option criteria.
D. Add a new payment gateway through the reference implementation steps so the payment shows up on the checkout payment screen. Configure the different payment options required.

Correct Answer: B
Section:
Explanation:
To update the checkout process so that buyers can purchase with one of the below payment types:
Credit Card

Purchase Order
Contract Now & Pay Later Additionally, show only Purchase Order and Contract Now & Pay Later if a custom checkbox field on the account is checked, a developer should create a custom Lightning web component for the checkout flow that has all the options available. Within that component, pull data from the account to determine which options to show. Creating a custom Lightning web component for the checkout flow allows the developer to define custom logic and user interface for processing payments using different payment types. The developer can use Apex methods or third-party APIs to integrate with payment service providers or payment gateways and handle payment authorization, capture, void, and refund. The developer can also use @wire or @api decorators to get data from the account object and use its properties, such as the custom checkbox field, to determine which payment options to show or hide based on business logic. Creating a custom Lightning web component that can be used with the standard payment component is not a valid way to meet this requirement, as it does not allow the developer to replace or modify the standard payment component's logic or user interface. Modifying the standard payment component settings in the checkout screen flow and adding the new payment method is not a valid way either, as it does not allow the developer to add custom payment types or conditional logic based on account data. Adding a new payment gateway through the reference implementation steps so the payment shows up on the checkout payment screen is not a valid way either, as it does not allow the developer to add multiple payment options or conditional logic based on account data. Salesforce
Reference: B2B Commerce Developer Guide: Payment Integration, B2B Commerce Developer Guide: Payment Component, B2B Commerce Developer Guide: Checkout Subflow

QUESTION 54
How should data for Lightning web components be provided?

A. A few properties that contain sets (objects) of data
B. One property that contains all data in one set (object)
C. A single property object that contains sets (objects) of data
D. Independent properties that take simpler, primitive values (e.g. String, Number, Boolean, Array)

Correct Answer: D
Section:
Explanation:
Data for Lightning web components should be provided as independent properties that take simpler, primitive values (e.g. String, Number, Boolean, Array). Providing data as independent properties allows the developer to expose data as public or private properties of the Lightning web component and communicate data between components or services. Providing data as simpler, primitive values allows the developer to use data types that are supported by JavaScript and Lightning web components and avoid unnecessary or complex conversions or transformations. Providing data as a few properties that contain sets (objects) of data is not a good way to provide data for Lightning web components, as it can create confusion or inconsistency in data structure and access. Providing data as one property that contains all data in one set (object) is not a good way either, as it can create complexity or inefficiency in data management and manipulation. Providing data as a single property object that contains sets (objects) of data is not a good way either, as it can create redundancy or duplication in data storage and retrieval. Salesforce
Reference: Lightning Web Components Developer Guide: Communicate with Properties, Lightning Web Components Developer Guide: Data Types

QUESTION 55
A developer needs to loop through a series of child components which are tiles. What is the correct syntax for this if the child component is called appTile?
A)

```
<template for:each={apps.data.records} for:item="app">
  <lightning-layout-item  key={app.Id} size="5"
flexibility="auto" class="slds-p-around_xxx-small">
    <c-app-tile  key={app.Id} app={app}
draggable={tilesAreDraggable} onselected={navigateMaster}>
    </c-app-tile>
  </lightning-layout-item>
</template>
```

B)

```
<template {for app : apps.data.records}>
  <lightning-layout-item  key={app.Id} size="5"
flexibility="auto" class="slds-p-around_xxx-small">
    <c-app-tile  key={app.Id} app={app}
draggable={tilesAreDraggable} onselected={navigateMaster}>
    </c-app-tile>
  </lightning-layout-item>
</template>
```

C)

```
<template for:{apps.data.records}.each() item="app">
  <lightning-layout-item  key={app.Id} size="5"
flexibility="auto" class="slds-p-around_xxx-small">
    <c-app-tile  key={app.Id} app={app}
draggable={tilesAreDraggable} onselected={navigateMaster}>
    </c-app-tile>
  </lightning-layout-item>
</template>
```

D)

```
<template for:each={apps.data.records} item="app">
  <lightning-layout-item  key={app.Id} size="5"
flexibility="auto" class="slds-p-around_xxx-small">
    <c-app-tile  key={app.Id} app={app}
```


A. Option A
B. Option B
C. Option C
D. Option D

Correct Answer: A
Section:
Explanation:
The correct syntax for looping through a series of child components which are tiles is option A. Option A uses the for:each directive to iterate over a collection of items and render a template block for each item. The for:each directive requires an expression that evaluates to an array or an iterable object and an item alias that represents the current item in the iteration. The item alias can be used to access the item's properties or pass them to child components. In option A, the expression is appTiles, which is an array of objects that represent app tiles, and the item alias is appTile, which represents the current app tile object in the iteration. The appTile object's properties, such as name, description, and icon, are passed to the app-tile child component using attributes. Option B is incorrect because it uses an invalid syntax for the for:each directive. The for:each directive requires a colon (:) after the for keyword, not an equal sign (=). Option C is incorrect because it uses an invalid syntax for the for:each directive. The for:each directive requires an item alias that represents the current item in the iteration, not a key alias that represents the current index in the iteration. Option D is incorrect because it uses an invalid syntax for the template tag. The template tag requires a closing tag (</template>), not a self-closing tag (<template/>). Salesforce
Reference: Lightning Web Components Developer Guide: Iterate Over a Collection, Lightning Web Components Developer Guide: Template Syntax

QUESTION 56
A developer needs to implement specific styling for a standard component on a single page of the B2B Commerce store using an Aura template. The component should use the default style on all other pages
How should the developer implement the required changes over multiple instances?

A. Use a Custom CSS file in a static resource and add the import using the Edit Head Markup Editor in the Experience Builder.
B. Create a Custom Content Layout Lightning web component that imports the custom CSS file. Set up the page to use this Content Layout.
C. Create a Custom Theme Layout Aura component that imports the custom CSS file. Set up the page to use this Theme Layout.
D. Use the Override CSR Editor in the Experience Builder and add the desired CSS to change the styles.

Correct Answer: C
Section:
Explanation:
To implement specific styling for a standard component on a single page of the B2B Commerce store using an Aura template, a developer should create a custom theme layout Aura component that imports the custom CSS file and set up the page to use this theme layout. A theme layout is a type of Aura component that defines the header and footer of a page in the storefront. A theme layout can also import custom CSS files from static resources and apply them to the page. A developer can create a custom theme layout Aura component that imports the custom CSS file that contains the specific styling for the standard component and assign it to the page that needs the custom styling. This way, the custom styling will only affect the standard component on that page and not on other pages that use a different theme layout. Using a custom CSS file in a static resource and adding the import using the Edit Head Markup Editor in the Experience Builder is not a valid way to implement specific styling for a standard component on a single page, as it will affect all pages that use the same template. Creating a custom content layout Lightning web component that imports the custom CSS file and setting up the page to use this content layout is not a valid way either, as it will not affect the standard component that is outside of the content layout. Using the Override CSR Editor in the Experience Builder and adding the desired CSS to change the styles is not a valid way either, as it will affect all pages that use the same template. Salesforce Reference:B2B Commerce Developer Guide: Theme Layout Component,B2B Commerce Developer Guide: Content Layout Component,B2B Commerce Developer Guide: Override CSR Editor

QUESTION 57
Which two methods from the platformResourceLoader module are relevant for including third party JavaScript and CSS in a Lightning web component?

A. loadClientScript
B. loadScript
C. loadCss
D. loadStyle

Correct Answer: B, D
Section:
Explanation:
Two methods from the platformResourceLoader module that are relevant for including third party JavaScript and CSS in a Lightning web component are loadScript and loadStyle. The platformResourceLoader module is a module that provides methods for loading JavaScript or CSS files from static resources or external URLs into a Lightning web component. The loadScript method is used to load a JavaScript file and execute it in the component. The loadStyle method is used to load a CSS file and apply it to the component. The loadClientScript method does not exist or is not relevant for including third party JavaScript in a Lightning web component. The loadCss method does not exist or is not relevant for including third party CSS in a Lightning web component. Salesforce
Reference:Lightning Web Components Developer Guide: Load Scripts and Style Sheets, [Lightning Web Components Developer Guide: platformResourceLoader Module]

QUESTION 58
In which two ways can events fired from Lightning web components be handled?

A. Programmatically adding event listeners
B. Adding callbacks to components
C. Listening for all possible events at the document root
D. Attaching handlers to DOM elements

Correct Answer: A, D
Section:
Explanation:
Two ways that events fired from Lightning web components can be handled are programmatically adding event listeners and attaching handlers to DOM elements. Programmatically adding event listeners is a way of handling events by using JavaScript code to register functions that are invoked when an event occurs. The developer can use methods such as addEventListener or @wire to add event listeners to components or services that fire events. Attaching handlers to DOM elements is a way of handling events by using HTML attributes to bind functions that are invoked when an event occurs. The developer can use attributes such as onclick or onchange to

attach handlers to DOM elements that fire events. Adding callbacks to components is not a valid way of handling events fired from Lightning web components, as it is not related to event handling, but rather to asynchronous programming. Listening for all possible events at the document root is not a valid way either, as it is not efficient or recommended for event handling, as it can cause performance issues or conflicts with other event listeners. Salesforce
Reference: [Lightning Web Components Developer Guide: Handle Events], [Lightning Web Components Developer Guide: Communicate with Events]

QUESTION 59
Which two components in a B2B store template should a developer use to customize a storefront page?

A. My Lists
B. Product List
C. Order List
D. Address List

Correct Answer: A, B
Section:
Explanation:
Two components in a B2B store template that a developer can use to customize a storefront page are My Lists and Product List. My Lists is a standard component that displays and manages lists of products that customers can create, save, or share in the storefront. A developer can use this component to enable customers to organize their favorite or frequently purchased products into lists and access them easily. Product List is a standard component that displays and filters products from one or more product lists in the storefront. A developer can use this component to showcase products from different categories or collections and allow customers to browse or search for products based on various criteria. Order List is not a component in a B2B store template, but rather an object that stores information about orders placed by customers in the storefront. Address List is not a component either, but rather an object that stores information about addresses associated with customers in the storefront. Salesforce
Reference: [B2B Commerce Developer Guide: My Lists Component], [B2B Commerce Developer Guide: Product List Component], [B2B Commerce Developer Guide: Order List Object], [B2B Commerce Developer Guide: Address List Object]

QUESTION 60
During checkout flow customizations, a developer receives an error on shipping cost calculation integrations with the error code: INSUFFICIENT_ACCESS_OR_READONLY.
What is causing this error?

A. The storefront user does not have access to the Cart Delivery Method object.
B. An error has occurred during the cart shipping charge integration.
C. The storefront user does not have access to custom fields on the Order Delivery Method object.
D. The cart is no longer in a valid Checkout State.

Correct Answer: D
Section:
Explanation:
The error code INSUFFICIENT_ACCESS_OR_READONLY is caused by the cart being no longer in a valid Checkout State during checkout flow customizations. A cart is an object that represents a collection of products and charges that a customer intends to purchase in the storefront. A cart has a Checkout State field that indicates the current state of the checkout process for the cart. The Checkout State can have values such as Draft, InProgress, Completed, or Cancelled. A cart can only be modified or updated when it is in Draft or InProgress state. A cart cannot be modified or updated when it is in Completed or Cancelled state. If an attempt is made to modify or update a cart that is in Completed or Cancelled state, an error with the code INSUFFICIENT_ACCESS_OR_READONLY will be thrown. This error means that the user does not have permission to edit or delete a record because it is read-only or locked. The storefront user does not have access to the Cart Delivery Method object is not a cause of this error code, as it is not related to the cart checkout state or data modification. The Cart Delivery Method object is an object that stores information about the delivery method selected for a cart in the storefront. An error has occurred during the cart shipping charge integration is not a cause of this error code either, as it is not related to the cart checkout state or data modification. The cart shipping charge integration is an integration that calculates and applies shipping charges to a cart based on various factors such as delivery method, location, weight, volume, etc. The storefront user does not have access to custom fields on the Order Delivery Method object is not a cause of this error code either, as it is not related to the cart checkout state or data modification. The Order Delivery Method object is an object that stores information about the delivery method selected for an order summary in the storefront. Salesforce
Reference:B2B Commerce Developer Guide: Cart Object, [B2B Commerce Developer Guide: Cart Delivery Method Object], [B2B Commerce Developer Guide: Order Delivery Method Object], [Salesforce Help: Common Error Messages]

QUESTION 61

Based on error emails flowing in, a developer suspects that recent edits made to a checkout flow have created a defect. The developer has data points available to use as inputs in reproducing the scenario.
What should the developer do next?

A. Open the flow, select Debug, provide the session ID for replay, and select Run.
B. Open the flow, select Attach to Live Session, provide the session ID, and select Attach.
C. Open the flow, select Debug, provide the Input values, and select Run.
D. Open the flow, select Debug with Inputs, provide the Input values, and select Run.

Correct Answer: C
Section:
Explanation:
The next step that the developer should do after suspecting that recent edits made to a checkout flow have created a defect and having data points available to use as inputs in reproducing the scenario is to open the flow, select Debug, provide the Input values, and select Run. A flow is a type of application that automates a business process by collecting data and performing actions in Salesforce or an external system. A flow can be used to customize the checkout process in the storefront by defining the steps and logic that are executed when a customer places an order. A flow can be edited or modified using Flow Builder, a point-and-click tool that allows developers to create and manage flows. Flow Builder also provides debugging and testing tools that allow developers to run and troubleshoot flows before deploying them. To debug or test a flow, the developer can open the flow in Flow Builder, select Debug from the toolbar, provide the Input values for the flow variables, and select Run. This will execute the flow in debug mode, which simulates how the flow runs in the org with real data. The developer can use debug mode to verify if the flow works as expected or if there are any errors or issues with the flow logic or actions. Open the flow, select Attach to Live Session, provide the session ID, and select Attach is not a valid next step, as it is not a feature or option available in Flow Builder or Salesforce CLI. Attach to Live Session is a feature that allows developers to attach a debugger to a running Apex session and inspect the state of the code execution. Open the flow, select Debug with Inputs, provide the Input values, and select Run is not a valid next step either, as it is not a feature or option available in Flow Builder or Salesforce CLI. Debug with Inputs is a feature that allows developers to debug an Apex class or trigger with predefined input values and breakpoints. Open the flow, select Debug, provide the session ID for replay, and select Run is not a valid next step either, as it is not a feature or option available in Flow Builder or Salesforce CLI. Replay is a feature that allows developers to replay an Apex log file and inspect the state of the code execution at each line. Salesforce
Reference: [B2B Commerce Developer Guide: Customize Checkout Flows], [Salesforce Help: Flow Builder], [Salesforce Help: Debug Your Flows], [Salesforce Developer Blog: Apex Replay Debugger]

QUESTION 62
Which two technologies can subscribe to the CommerceDiagnosticEvents event?

A. Aura Components
B. Processes
C. Streaming API
D. Lightning web components

Correct Answer: A, D
Section:
Explanation:
Two technologies that can subscribe to the CommerceDiagnosticEvents event are Aura Components and Lightning web components. CommerceDiagnosticEvents is an event that is fired by Salesforce B2B Commerce when an error occurs in the storefront. CommerceDiagnosticEvents contains information about the error, such as error code, error message, error type, and error details. CommerceDiagnosticEvents can be subscribed by other components or services that want to handle or display the error information in different ways. Aura Components are a type of component that can be used to create custom user interfaces for Salesforce apps. Aura Components can subscribe to CommerceDiagnosticEvents using anaura:handlertag in their markup file. Theaura:handlertag specifies an event name, an action attribute that defines a controller function to handle the event, and other optional attributes. Lightning web components are another type of component that can be used to create custom user interfaces for Salesforce apps. Lightning web components can subscribe to CommerceDiagnosticEvents using an @wire decorator in their JavaScript file. The @wire decorator specifies an event name, a function name that defines a handler for the event, and other optional parameters. Processes are not a technology that can subscribe to CommerceDiagnosticEvents, as they are not related to user interface development or event handling. Processes are automated workflows that execute actions based on certain criteria or conditions in Salesforce. Streaming API is not a technology that can subscribe to CommerceDiagnosticEvents either, as it is not related to user interface development or event handling. Streaming API is an API that allows applications to receive notifications of data changes in Salesforce in near real-time. Salesforce
Reference: [B2B Commerce Developer Guide: Handle Errors], [Aura Components Developer Guide: Handle Component Events], [Lightning Web Components Developer Guide: Communicate with Events], [Salesforce Help: Process Automation], [Salesforce Developer Guide: Streaming API]

QUESTION 63
In checkout, what event should the developer's code listen for in order to help troubleshoot and respond to actions?

A. CommerceBubbleEvents
B. CommerceErrorEvents
C. CommerceActionEvents
D. CommerceDiagnosticEvents

Correct Answer: D
Section:
Explanation:
To help troubleshoot and respond to actions in checkout, the developer's code should listen for CommerceDiagnosticEvents. CommerceDiagnosticEvents is an event that is fired by Salesforce B2B Commerce when an error occurs in the storefront. CommerceDiagnosticEvents contains information about the error, such as error code, error message, error type, and error details. CommerceDiagnosticEvents can be subscribed by other components or services that want to handle or display the error information in different ways. The developer's code can listen for CommerceDiagnosticEvents using anaura:handlertag in Aura Components or an @wire decorator in Lightning web components. The developer's code can also use the event information to perform custom logic or actions based on the error, such as logging, reporting, or notifying. CommerceBubbleEvents is not an event that the developer's code should listen for in checkout, as it is not related to troubleshooting or responding to actions. CommerceBubbleEvents is an event that is fired by Salesforce B2B Commerce when a user interacts with a bubble component in the storefront. A bubble component is a user interface element that displays information or options in a pop-up window when clicked or hovered over. CommerceBubbleEvents contains information about the user interaction, such as bubble name, bubble type, and bubble value. CommerceErrorEvents is not an event that the developer's code should listen for in checkout either, as it is not related to troubleshooting or responding to actions. CommerceErrorEvents is an event that is fired by Salesforce B2B Commerce when a validation error occurs in the storefront. CommerceErrorEvents contains information about the validation error, such as field name, field label, and error message. CommerceActionEvents is not an event that the developer's code should listen for in checkout either, as it is not related to troubleshooting or responding to actions. CommerceActionEvents is an event that is fired by Salesforce B2B Commerce when a user performs an action in the storefront. CommerceActionEvents contains information about the user action, such as action name, action type, and action parameters. Salesforce
Reference:B2B Commerce Developer Guide: Handle Errors,B2B Commerce Developer Guide: Handle User Interactions with Bubble Components,B2B Commerce Developer Guide: Handle Validation Errors,B2B Commerce Developer Guide: Handle User Actions

QUESTION 64
Which two are considered discrete units of work (code units) within a transaction in the debug logs?

A. Validation rule
B. Lightning component load
C. Web service invocation
D. Apex class

Correct Answer: C, D
Section:
Explanation:
Two data types that are considered discrete units of work (code units) within a transaction in the debug logs are web service invocation and Apex class. A discrete unit of work (code unit) is a segment of executable code that runs as part of a transaction in Salesforce. A transaction is a sequence of operations that are treated as a single unit of work and are executed under certain isolation and consistency rules. A transaction can consist of one or more discrete units of work (code units) that are executed sequentially or concurrently depending on various factors such as triggers, asynchronous calls, or limits. A debug log is a record of database operations, system processes, and errors that occur when executing a transaction or running unit tests in Salesforce. A debug log can capture information about each discrete unit of work (code unit) within a transaction, such as its start time, end time, duration, events, variables, and limits. A web service invocation is a type of discrete unit of work (code unit) that involves calling an external web service from Apex code using SOAP or REST protocols. A web service invocation can be synchronous or asynchronous depending on the method used to make the callout. A web service invocation can be captured in a debug log with its details and results. An Apex class is another type of discrete unit of work (code unit) that involves executing Apex code that defines a class with properties and methods. An Apex class can be invoked from various sources such as triggers, Visualforce pages, Lightning components, or API calls. An Apex class can be captured in a debug log with its details and results. A validation rule is not a type of discrete unit of work (code unit) within a transaction in the debug logs, as it is not a segment of executable code but rather a formula expression that defines a business rule for a field or object. A validation rule can be evaluated during a transaction and cause an error if the rule condition is not met, but it cannot be captured as a separate code unit in a debug log. A Lightning component load is not a type of discrete unit of work (code unit) within a transaction in the debug logs either, as it is not a segment of executable code but rather an event that occurs when a Lightning component is rendered on a web page or application. A Lightning component load can be measured by various performance tools such as Lightning Inspector or Lighthouse, but it cannot be captured as a separate code unit in a debug log. Salesforce
Reference: [Salesforce Developer Blog: Transactions and Request Processing], [Salesforce Help: Debug Logs], [Salesforce Developer Guide: Invoking Callouts Using Apex], [Salesforce Developer Guide: Apex Classes], [Salesforce Help: Validation Rules], [Salesforce Developer Blog: Measuring Lightning Component Performance]

QUESTION 65
A developer is on a tight timeline and needs to implement a Lightning web component which can read, create and modify single records. What is the recommended path forward?

A. Use base components
B. Write custom functions against a wire adapter
C. Create an Apex Controller
D. Use Lightning Data Service

Correct Answer: D
Section:
Explanation:
To implement a Lightning web component which can read, create and modify single records, the recommended path forward is to use Lightning Data Service. Lightning Data Service is a service that provides access to Salesforce data and metadata, cache management, and data synchronization across components. Lightning Data Service allows developers to use standard components or base components to perform CRUD (create, read, update, delete) operations on single records without writing Apex code or SOQL queries. Lightning Data Service also handles data caching, performance optimization, and conflict resolution automatically. Using base components is not a sufficient way to implement a Lightning web component which can read, create and modify single records, as base components are only user interface elements that do not provide data access or manipulation functionality by themselves. Base components need to be used with Lightning Data Service or other services to perform CRUD operations on single records. Writing custom functions against a wire adapter is not a recommended way to implement a Lightning web component which can read, create and modify single records, as it involves writing complex and error-prone JavaScript code that may not be efficient or scalable. Writing custom functions against a wire adapter also requires handling data caching, performance optimization, and conflict resolution manually. Creating an Apex controller is not a recommended way either, as it involves writing Apex code that may not be necessary or optimal for performing CRUD operations on single records. Creating an Apex controller also requires exposing Apex methods using @AuraEnabled or @RemoteAction annotations and invoking them from JavaScript code using imperative calls or promises. Salesforce
Reference:Lightning Web Components Developer Guide: Access Salesforce Data,Lightning Web Components Developer Guide: Base Components, [Lightning Web Components Developer Guide: Call Apex Methods]

QUESTION 66
Which two blocks of code are needed to implement a custom getter in a Lightning web component?
A)

```
set rows(value) {
this.state.rows = value;
}
```

B)

```
get rows() {
return this.state.rows;
}
```

C)

```
@api
set rows(value) {
this.state.rows = value;
}
```

D)

```
@api
get rows() {
return this.state.rows;
}
```

A. Option A
B. Option B
C. Option C
D. Option D

Correct Answer: A, D
Section:

QUESTION 67
A developer has created a custom Lightning web component for the Cart page that needs to react to changes to cart items from the standard cart component.
How should the developer implement the custom component so changes to cart items and quantities are reflected?

A. Subscribe to events on the lightning_commerce_cartChanged channel using the Lightning Message Service.
B. Add a listener for the cartItemUpdate Lightning event.
C. Listen for events on the lightning_commerce_cartChanged channel with the Lightning Event ''Listener component.
D. Add an event listener for the cartchanged DOM (Document Object Model) event.

Correct Answer: A
Section:

QUESTION 68
Which two methods should a developer implement in a Lightning web component to successfully handle the subscription lifecycle of a Message Channel?

A. Subscribe()
B. stopListener()
C. startListener()
D. unsubscribe()
Answer: A, D

Correct Answer: A, D
Section:
Explanation:
Subscribe and Unsubscribe from a Message Channel
Lifecycle Hooks
Use message channel in both direction

QUESTION 69
Which practice is allowed when it comes to naming a Lightning web component's folder and associated files?

A. Including whitespace
B. Using a single underscore
C. Using consecutive underscores
D. Using a single hyphen (dash)

Correct Answer: D
Section:

QUESTION 70
Which element can be used to pass HTML from a parent component to a child component? 03m 19s

A. <html></html>
B. <template></template>
C.
D. <slot></slot>

Correct Answer: D
Section:

QUESTION 71
A developer has made a component with a lightning combobox in the follow! markup. To handle changes on the combobox, what should replace <CHANGE FVENT>?

```
<template>
  <lightning-combobox
    name="billingAddressSelector"
    label={title}
    options={selectableAddresses}
    value={initialSelectedAddressId}
    class="slds-p-bottom_medium"
    onchange=<CHANGE_EVENT>
    required={billingAddressRequired}
  >
  </lightning-combobox>
</template>
```

A. {event:handleChange}
B. javascript:void(0);handleChange();
C. {handleChange()}
D. {handleChange}

Correct Answer: D
Section:
Explanation:
To handle changes on the combobox, the developer should replace <CHANGE EVENT> with {handleChange}. The handleChange is a function that is defined in the JavaScript file of the Lightning web component and is invoked

when the value of the combobox changes. The developer can use this function to perform custom logic or actions based on the selected value of the combobox, such as updating other components or properties, calling Apex methods or services, or firing events. The developer can use the onchange attribute to bind the handleChange function to the combobox element in the HTML file of the Lightning web component. The onchange attribute is an HTML attribute that specifies a function to be executed when the value of an element changes. {event:handleChange} is not a valid way to handle changes on the combobox, as it is not a valid syntax for binding a function to an element. javascript:void(0);handleChange(); is not a valid way either, as it is not a valid syntax for binding a function to an element. {handleChange()} is not a valid way either, as it is not a valid syntax for binding a function to an element. Salesforce
Reference: Lightning Web Components Developer Guide: Handle Events, Lightning Web Components Developer Guide: Communicate with Properties, Lightning Web Components Developer Guide: lightning-combobox

QUESTION 72
A developer is working in Visual Studio Code on a previously deployed project which is rather large and deployments are time consuming. The developer wants to deploy some small CSS changes without waiting for the entire project deployment. What are two ways this can be accomplished?

A. Right-click the folder for the component and choose Deploy Source to Org
B. Right-click the CSS file that was edited and select Deploy Single File
C. Right-click the CSS file and choose Deploy Source to Org
D. Click the Tools menu and select Deploy styles
E. Deploy the entire project. Only the change will be copied

Correct Answer: A, B
Section:
Explanation:
Two ways that a developer can deploy some small CSS changes without waiting for the entire project deployment are right-clicking the folder for the component and choosing Deploy Source to Org and right-clicking the CSS file that was edited and selecting Deploy Single File. Deploying source to org is a way of deploying metadata from a local project to an org using Salesforce CLI commands. The developer can use Visual Studio Code to execute these commands by right-clicking on files or folders in the project and choosing from various deployment options. Right-clicking the folder for the component and choosing Deploy Source to Org allows the developer to deploy only the files that belong to that component, such as HTML, JavaScript, CSS, and XML files. Right-clicking the CSS file that was edited and selecting Deploy Single File allows the developer to deploy only that CSS file and not any other files in the project. These options can save time and bandwidth for deploying small changes without affecting other components or files in the project. Modifying the StoreIntegratedService to map to an Apex class ID using Workbench is not a way of deploying CSS changes, as it is only used for registering internal services that are provided by Salesforce B2B Commerce out-of-the-box. Entering the integration class name and version in store administration is not a way of deploying CSS changes either, as it is only used for selecting an existing integration class that has already been registered as an external service. Salesforce
Reference: Salesforce CLI Command Reference: force:source:deploy, Salesforce Developer Tools for Visual Studio Code, B2B Commerce Developer Guide: Integration Framework, B2B Commerce Developer Guide: RegisteredExternalService Object

QUESTION 73
Northern Tail Outfitters (NTO) is converting an existing aura component into a Lightning Web Component. The aura component has the following source code:

```
<!-- owner.cmp -->
<aura:component>
   <aura:attribute name="fName" type="String" />
   <c:child firstName="{!v.fName}"/>
</aura:component>
```

What is the equivalent of this code in a Lightning Web Component?
A)

```
<!-- owner.html -->
<template>
   <c-child first-name={!fName}></child>
</template>
```

B)

```
<!-- owner.html -->
<template>
   <c-child first-name={fName}></child>
</template>
```

C)

```
<!-- owner.html -->
<template>
   <c-child first-name={<fName>}></child>
</template>
```

D)

```
<!-- owner.html -->
<template>
   <c-child first-name={item.fName}></child>
</template>
```

A. Option A
B. Option B
C. Option C
D. Option D

Correct Answer: B
Section:
Explanation:
The equivalent of this code in a Lightning web component is option B. Option B uses the @api decorator to expose firstName as a public property of the Lightning web component and communicate it with other components or services. The @api decorator is a decorator that marks a property or method as public, which means that it can be accessed by other components or services that use or consume this component. The @api decorator also makes the property reactive, which means that it can track changes and update the component accordingly. In option B, firstName is exposed as a public property using the @api decorator and passed to the child element using an attribute. Option A is incorrect because it uses an invalid syntax for exposing firstName as a public property. The @api decorator should be used before the property declaration, not after it. Option C is incorrect because it uses an invalid syntax for exposing firstName as a public property. The @api decorator should be used with parentheses, not without them. Option D is incorrect because it uses an invalid syntax for exposing firstName as a public property. The @api decorator should be used with camel case, not with hyphens. Salesforce
Reference: Lightning Web Components Developer Guide: Communicate with Properties, Lightning Web Components Developer Guide: Decorators

QUESTION 74
An administrator has just provided a developer with a completely new org and a username. Assuming the username is me@my-new-org.org, what is the correct set of steps to authorize the org for Command Line Interface (CLI) access so the developer can deploy Lightning web components?

A. Run the command: 'sfdx force:login -r 'https://login.salesforce.com'' and supply the credentials in the browser when it opens.
B. Run the command 'sfdx force:auth:web:login -a 'https://login.salesforce.com'' and then supply the credentials in the browser when it opens.
C. Run the command: 'sfdx force:auth:web:login -r 'https://login.salesforce.com' and then supply the credentials in the ^ browser when it opens ^
D. Run the command 'sfdx force:auth:web:login -r 'https://login.salesforce.com' -username^'mefaJmy-new-org.org''

Correct Answer: C
Section:
Explanation:

To authorize the org for Command Line Interface (CLI) access so the developer can deploy Lightning web components, the developer should run the command: 'sfdx force:auth:web:login -r ''https://login.salesforce.com'' and then supply the credentials in the browser when it opens. The sfdx force:auth:web:login command is a Salesforce CLI command that authorizes an org using the web server flow. The web server flow is an OAuth 2.0 authentication flow that opens a browser window and prompts the user to log in to Salesforce and allow access to the CLI. The -r flag specifies the login URL of the org, which is https://login.salesforce.com for production or developer orgs. Running this command will open a browser window and ask the developer to enter their username and password for the org. After successfully logging in, the developer will be able to use the CLI to perform various tasks with the org, such as deploying or retrieving metadata, running tests, or executing commands. Running the command: 'sfdx force:login -r ''https://login.salesforce.com'' is not a valid way to authorize the org for CLI access, as there is no such command as sfdx force:login. Running the command 'sfdx force:auth:web:login -a ''https://login.salesforce.com''' is not a valid way either, as the -a flag specifies an alias for the org, not a login URL. Running the command 'sfdx force:auth:web:login -r ''https://login.salesforce.com'' -username'me@my-new-org.org'' is not a valid way either, as there is no such flag as -username. Salesforce
Reference:Salesforce CLI Command Reference: force:auth:web:login,Salesforce Developer Tools for Visual Studio Code

QUESTION 75
A developer needs to import some new product data contained in a JSON file one time. What are two viable ways to do this? .

A. Convert the JSON to an xlsx file and use Workbench to import it
B. Run a command like: sfdx force:data:tree:import -f NewProducts.json -u <your username>
C. Convert the JSON to a CSV file and use Data Loader to import it
D. Run a command like: sfdx force:data;import:bulk -f NewProducts.json -u <your username>

Correct Answer: B, C
Section:
Explanation:
Two viable ways that a developer can import some new product data contained in a JSON file one time are running a command like: sfdx force:data:tree:import -f NewProducts.json -u <your username> and converting the JSON to a CSV file and using Data Loader to import it. Running a command like: sfdx force:data:tree:import -f NewProducts.json -u <your username> allows the developer to import data from a JSON file into an org using Salesforce CLI commands. The sfdx force:data:tree:import command is a Salesforce CLI command that imports data into an org using JSON files that conform to the SObject Tree API specification. The SObject Tree API specification is a format that defines how records are represented in JSON files for data import or export. The -f flag specifies the path of the JSON file that contains the data to be imported. The -u flag specifies the username or alias of the org where the data will be imported. Running this command will create records in the org based on the data in the JSON file. Converting the JSON to a CSV file and using Data Loader to import it allows the developer to import data from a CSV file into an org using Data Loader. Data Loader is a tool that allows users to import or export data between Salesforce and CSV files. The developer can use an online converter or a spreadsheet application to convert their JSON file into a CSV file that matches the structure and format of their Salesforce object. The developer can then use Data Loader to import the CSV file into their org and create records based on the data in the CSV file. Converting the JSON to an xlsx file and using Workbench to import it is not a viable way to import some new product data contained in a JSON file one time, as Workbench does not support xlsx files for data import or export. Workbench is a web-based tool that provides access to various Salesforce features and functionalities, such as data manipulation, REST Explorer, and Apex Execute. Running a command like: sfdx force:data;import:bulk -f NewProducts.json -u <your username> is not a viable way either, as there is no such command as sfdx force:data;import:bulk. The correct command for importing data using bulk API is sfdx force:data:bulk:upsert. Salesforce
Reference: [Salesforce CLI Command Reference: force:data:tree:import], [Salesforce Developer Tools for Visual Studio Code], [Data Loader Guide: Import Data into Salesforce], [Workbench], [Salesforce CLI Command Reference: force:data:bulk:upsert]

QUESTION 76
A developer has the task to bring some historical data into an org. The data must reside in the org, but cannot be populated in standard or custom objects. The customer is fine with developers building UI components to surface this data on a case-by-case basis.
Which option allows a developer to meet all of these requirements?

A. Big objects
B. Lightning Canvas
C. External objects
D. Lightning Out

Correct Answer: A
Section:
Explanation:
To bring some historical data into an org, the data must reside in the org, but cannot be populated in standard or custom objects, and the customer is fine with developers building UI components to surface this data on a case-by-case basis, the option that allows a developer to meet all of these requirements is big objects. Big objects are a type of object that can store and manage massive amounts of data on the Salesforce platform. Big

objects can store up to billions of records and are accessible through a subset of SOQL or custom user interfaces. Big objects are not subject to the same storage limits or performance issues as standard or custom objects and are suitable for storing historical or archived data that does not need to be updated frequently. Big objects can be defined using Metadata API or declaratively in Setup. Lightning Canvas is not an option that allows a developer to meet all of these requirements, as it is a framework that allows developers to integrate third-party applications into Salesforce. Lightning Canvas does not store data in the org, but rather displays data from external sources using an iframe. External objects are not an option either, as they are a type of object that map to data stored outside Salesforce. External objects do not store data in the org, but rather access data from external systems using OData services. Lightning Out is not an option either, as it is a feature that allows developers to use Lightning components outside Salesforce. Lightning Out does not store data in the org, but rather renders components on external web pages or applications. Salesforce
Reference:Salesforce Help: Define Big Objects,Salesforce Help: Lightning Canvas Developer's Guide,Salesforce Help: External Objects,Salesforce Developer Blog: Lightning Out

QUESTION 77
Which three data types are supported for custom fields while using CSV file format for importing data for a store?

A. Text Area(Long)
B. Picklist (Multi-Select)
C. Lookup Relationship
D. Address
E. Currency

Correct Answer: A, C, E
Section:
Explanation:
Three data types that are supported for custom fields while using CSV file format for importing data for a store are Text Area(Long), Lookup Relationship, and Currency. A custom field is a field that is added by a developer or an administrator to an object to store additional information or data. A data type is a property that defines the type, format, and validation rules of a field. A CSV file is a file format that stores tabular data in plain text using commas to separate values. A store is a record that represents a B2B or B2C storefront in Salesforce. Text Area(Long) is a data type that allows users to enter up to 131,072 characters on separate lines. Text Area(Long) is supported for custom fields while using CSV file format for importing data for a store. Lookup Relationship is a data type that allows users to create a relationship between two objects and select a value from another record. Lookup Relationship is supported for custom fields while using CSV file format for importing data for a store. Currency is a data type that allows users to enter currency values and automatically convert them based on the user's locale and currency settings. Currency is supported for custom fields while using CSV file format for importing data for a store. Picklist (Multi-Select) is a data type that allows users to select one or more values from a predefined list of values. Picklist (Multi-Select) is not supported for custom fields while using CSV file format for importing data for a store. Address is a data type that allows users to enter address values and automatically format them based on the user's locale settings. Address is not supported for custom fields while using CSV file format for importing data for a store. Salesforce
Reference: [Salesforce Help: Custom Field Attributes], [Salesforce Help: Data Types], [Data Loader Guide: Import Data into Salesforce], [B2B Commerce Developer Guide: Store Object]

QUESTION 78
Northern Trail Outfitters (NTO) has acquired a company and is looking to manage product data across the org seamlessly. The company has a governance policy to not install any tool or use third-party API applications to export or import the data into Salesforce. However, users have access to Salesforce CLI.
Which set of tasks must a developer perform whento export data from Salesforce or import data into Salesforce?

A. sfdx force:data:bulk:export -Product2 -all 0 and sfdx force:data:bulk:lmport -f Product2.json -all
B. sfdx force:data;tree:export -Product2 -all q and sfdx force:data:tree:lmport -f Product2.json -all
C. sfdx force:tree:data:export -q 'SELECT Id, Name FROM Product2' -u <your username> and sfdx force:tree:data:import -f Product2Json -all
D. sfdx force:data:tree:export -q 'SELECT Id, Name FROM Product2' -u '<your username>' and sfdx force:data:tree:import -f Product2.json -u w<your username>'

Correct Answer: D
Section:
Explanation:
The correct answer for how to export data from Salesforce or import data into Salesforce using Salesforce CLI commands is running a command like: sfdx force:data:tree:export -q ''SELECT Id, Name FROM Product2'' -u ''<your username>'' and sfdx force:data:tree:import -f Product2.json -u ''<your username>''. The sfdx force:data:tree:export command is a Salesforce CLI command that exports data from an org into JSON files that conform to the SObject Tree API specification. The SObject Tree API specification is a format that defines how records are represented in JSON files for data import or export. The -q flag specifies the SOQL query that selects the records and fields to be exported. The -u flag specifies the username or alias of the org where the data will be exported from. Running this command will generate JSON files that contain the data from the org based on the SOQL query. The sfdx force:data:tree:import command is a Salesforce CLI command that imports data into an org using JSON files that conform to the SObject Tree API specification. The -f flag specifies the path of the JSON file that

contains the data to be imported. The -u flag specifies the username or alias of the org where the data will be imported to. Running this command will create records in the org based on the data in the JSON file. Running a command like: sfdx force:data:bulk:export -Product2 -all 0 and sfdx force:data:bulk:import -f Product2.json -all is not a correct answer, as it uses invalid syntax and flags for the sfdx force:data:bulk:export and sfdx force:data:bulk:import commands. The correct syntax and flags for these commands are sfdx force:data:bulk:upsert -s Product2 -f Product2.csv -w 10 -u <your username> and sfdx force:data:bulk:status -i <job ID> -u <your username>. Running a command like: sfdx force:data;tree:export -Product2 -all q and sfdx force:data:tree:import -f Product2.json -all is not a correct answer either, as it uses invalid syntax and flags for the sfdx force:data:tree:export and sfdx force:data:tree:import commands. The correct syntax and flags for these commands are sfdx force:data:tree:export -q ''SELECT Id, Name FROM Product2'' -u <your username> and sfdx force:data:tree:import -f Product2.json -u <your username>. Running a command like: sfdx force:tree:data:export -q ''SELECT Id, Name FROM Product2'' -u <your username> and sfdx force:tree:data:import -f Product2Json -all is not a correct answer either, as there is no such command as sfdx force:tree:data:export or sfdx force:tree:data:import. The correct commands are sfdx force:data:tree:export and sfdx force:data:tree:import. Salesforce Reference: [Salesforce CLI Command Reference: force:data:tree:export], [Salesforce CLI Command Reference: force:data:tree:import], [Salesforce CLI Command Reference: force:data:bulk], [Salesforce Developer Tools for Visual Studio Code]

QUESTION 79
A developer is working on a storefront and is seeing unexpected Ul behavior in one of the custom Lightning web components (LWCs) their team has built.
How should the developer investigate the issue?

A. Enable Debug Mode for a storefront user, log in to the storefront, and use Browser Inspection tools and debugger points.
B. Enable Debug Mode for a storefront user, load the LWC in Visual Studio (VS) Code, attach to session, and view debug logs in VS Code.
C. Enable debug logs for a storefront user, log in to storefront and perform action, and view debug logs in Setup.
D. Identify the user, inputs, and failure, then ask Salesforce support to investigate the issue with the custom LWC.

Correct Answer: A
Section:
Explanation:
To investigate the issue of seeing unexpected UI behavior in one of the custom Lightning web components (LWCs) their team has built, the developer should enable Debug Mode for a storefront user, log in to the storefront, and use Browser Inspection tools and debugger points. Debug Mode is a feature that allows developers to debug and troubleshoot custom LWCs in the storefront by disabling performance optimizations and enabling source maps. Source maps are files that map the minified or obfuscated code to the original source code, making it easier to read and debug. To enable Debug Mode for a storefront user, the developer can go to Setup, enter Users in the Quick Find box, select Users, click Edit next to the user name, and select Debug Mode. After enabling Debug Mode, the developer can log in to the storefront as the user and use Browser Inspection tools and debugger points to inspect and debug the custom LWC. Browser Inspection tools are tools that are built into web browsers that allow developers to examine and modify the HTML, CSS, JavaScript, and other aspects of a web page. Debugger points are statements that are added to the JavaScript code of a LWC that pause the execution of the code at a certain point and allow the developer to inspect the values of variables, expressions, and other elements. Enable Debug Mode for a storefront user, load the LWC in Visual Studio (VS) Code, attach to session, and view debug logs in VS Code is not a valid way to investigate the issue of seeing unexpected UI behavior in one of the custom LWCs their team has built, as it is not possible to attach to a session or view debug logs for LWCs in VS Code. Enable debug logs for a storefront user, log in to storefront and perform action, and view debug logs in Setup is not a valid way either, as debug logs do not capture information about LWCs or UI behavior. Debug logs are records of database operations, system processes, and errors that occur when executing a transaction or running unit tests. Identify the user, inputs, and failure, then ask Salesforce support to investigate the issue with the custom LWC is not a valid way either, as it is not a recommended or efficient way of debugging or troubleshooting custom LWCs. Salesforce support may not be able to provide assistance or guidance for custom LWCs that are developed by third-party developers. Salesforce Reference:B2B Commerce Developer Guide: Debug Lightning Web Components,Lightning Web Components Developer Guide: Debug Your Code,Salesforce Help: Debug Logs