Testpassport덤프

Testpassport 는 고객님께 일년무료 업데이트 서비스를 제공해드립니다

Testpassport 고품질 인증덤프

http://www.testpassport.kr

Exam : **70-490**

Title : Recertification for MCSD: Windows Store Apps Using HTML5

Version : Demo

1. Topic 1, Scenario 1

Application Information
You are developing two Windows Store apps by using JavaScript: a Personal Trainer app and a Client app. The apps will allow personal fitness trainers to interact with their remote clients.

Business Requirements - Personal Trainer Application
The Personal Trainer app must allow trainers to perform the following tasks:
. Create and store video and audio recordings of workout routines.
. View the profile and workout recordings for only one client at any time.

Business Requirements - Client Application
The Client app must allow clients to perform the following tasks:
. Browse a list of the trainer's workout recordings.
. Record workouts by using the built-in webcam.
. Play, pause, restart, and stop workout recordings.
. If the capability is supported, allow the client's webcam to pan as the client moves around the room.
. Upload workout recordings for trainer review.
. Update their individual profiles to indicate workouts completed, calories burned, and current weight.
The Client app must validate that the client's subscription is valid.

Technical Requirements - General
The Personal Trainer and Client apps must meet the following technical requirements:
. Connect to the Internet.
. Store workout recordings in the cloud.
. Enable retrieval of workout recordings by using a custom URL
. Encapsulate the video player in a custom control.
. Identify the maximum zoom of the user's webcam in millimeters.
. Store client profiles in XML files in the trainers' Documents folders to allow for disconnected editing.
. Synchronize the XML files with cloud storage by using a background task when the Internet is available.
. Send trainer workout videos to cloud storage by using a background task when the trainer's device is idle. Indicate the status of the upload operation each time the trainer starts the app. Suspend the background task when the Internet is not available.
. Separate business and complex logic into WinMD components. The solution debugging settings must include the WinMD components.

Technical Requirements - Hardware Requirements
The Personal Trainer and Client apps must support the following hardware requirements:
. Windows 8
. Webcam, microphone, and speakers
. Internet connection
While testing the apps, you identify the following issues:
. When you start the app for the first time, the system displays this warning message: "This app needs permission to use your camera, which you can change in the app's settings."

. When you run the loadClientProfile() method in the clientData.js file, you receive an 'Access Denied" exception.
. The findCamera() method in the video.js file throws an exception on some devices.
. The recordVideo() method in the video.js file throws an exception when the device does not support tilting.

```
CD01 function loadClientProfile() {
CD02   var fop = new Windows.Storage.Pickers.FileOpenPicker();
CD03   fop.viewMode = Windows.Storage.Pickers.PickerViewMode.thumbnail;
CD04
CD05   fop.fileTypeFilter.replaceAll([".xml"]);
CD06
CD07     (function (file) {
CD08       if (file) {
CD09         display(file);
CD10       }
CD11       else {
CD12         processError(file);
CD13       }
CD14     });
CD15 }
CD16
CD17 function saveClientProfile() {
CD18   var sp = new Windows.Storage.Pickers.FileSavePicker();
CD19   sp.defaultFileExtension = ".xml";
CD20   sp.suggestedFileName = "New Client";
CD21
CD22
CD23   sp.pickSaveFileAsync().then(
CD24     function (file) {
CD25       if (file) {
CD26         displaySaved(file);
CD27       }
CD28       else {
CD29         processError(file);
CD30       }
CD31     });
CD32 }
```

video.js

```
VD01 function recordVideo() {
VD02   var device = new Windows.Media.Capture.MediaCapture();
VD03   var videoDev = device.videoDeviceController;
VD04   var canTilt = videoDev.tilt.capabilities.supported;
VD05
VD06
VD07   ...
VD08 }
VD09
VD10 var cameraID;
VD11
VD12 function findCamera() {
VD13   var deviceInfo = Windows.Devices.Enumeration.DeviceInformation;
VD14   deviceInfo.findAllAsync(Windows.Devices.Enumeration.DeviceClass.videoCapture).then
(function (devices) {
VD15     cameraID = devices[0].id;
VD16   }, errorHandler);
VD17 }
VD18
```

background.js

```
BG01 function registerBackgroundTask(condition) {
BG02   var builder = new Windows.ApplicationModel.Background.BackgroundTaskBuilder();
BG03   builder.name = "videoLoader";
BG04   builder.taskEntryPoint = "background.js";
BG05   builder.setTrigger(
BG06    Windows.ApplicationModel.Background.SystemTrigger(
BG07
BG08    ));
BG09
BG10   ...
BG11 }
BG12
BG13 function unregisterBackgroundTask() {
BG14
BG15   var i = tasks.hasCurrent;
BG16   while (i) {
BG17     var task = tasks.current.value;
BG18     if (task.name === "videoLoader") {
BG19       task.unregister(true);
BG20     }
BG21     i = tasks.moveNext();
BG22   }
BG23 }
```

You need to debug the error that is displayed in the warning message.
What should you do?

A. Insert the following code segment at line VD18:

```
var result = new Windows.Media.Capture.CameraCaptureUI();
result.videoSettings.setCameraEnabled();
```

B. In the appsettings.appxmanifest file, add **Camera Settings** to the available declarations.

C. Insert the following code segment at line VD18:

```
var result = new Windows.Media.Capture.CameraCaptureUI();
result.photoSettings.setCameraEnabled();
```

D. In the package.appxmanifest file, set the **Webcam** property in the **Capabilities** list.

A. Option A
B. Option B
C. Option C
D. Option D
Answer: D

2.You need to set the default storage location for the client profiles.
Which code segment should you insert at line CD04?
A. fop.suggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.documents Library;
B. fop.defaultFolder = Windows.Storage.Piekers.PickerLocationId.documents Library;
C. fop.suggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.videosLibrary;
D. fop.suggestedStartLocation = "%AppData%";
Answer: B

3.You need to retrieve the background task collection for the iteration loop.
Which code segment should you insert at line BG14?
A. var tasks = Windows.ApplicationModel.Background.BackgroundTaskRegistration.first();
B. var tasks = Windows.ApplicationModel.Background.BackgroundTaskRegistration.allTasks.first();
C. var tasks = Windows.ApplicationModel.Background.BackgroundTaskBuilder.allTasks.first();
D. var tasks = Windows.ApplicationModel.Background.BackgroundTaskRegistration.allTasks();
Answer: B

4.You need to complete the code to start the background task.
Which code segment should you insert at line BG07?
A. Windows.ApplicationModel.Background.SystemTriggerType.servicingComplete.true
B. Windows.ApplicationModel.Background.SystemTriggerType.userPresent.true
C. Windows.ApplicationModel.Background.SystemTriggerType.internetAvailable.false
D. Windows.ApplicationModel.Background.SystemTriggerType.userAway.false
Answer: C

5. Topic 2, Scenario 2

Background
You are developing an app for an automotive manufacturer. The app will display information about the vehicle, the vehicle owner's manual, and the maintenance schedule. The app will be available to install from the Windows Store.

Business Requirements
The app must meet the following business requirements: . Display the company logo on the Main screen, Owner's Manual screen, and Service Record screen of the app. . Allow users to store their vehicle information to identify the correct information to display within the app. . Prominently display a stock image of the user's vehicle on the Main screen at the full height of the app. . Send notifications by using tile updates when a scheduled maintenance is approaching or past due. . Insert service data in the appropriate locations, and update the Service Record screen with data received from a cloud service. . Display related media within the app.

Technical Requirements
The app must meet the following technical requirements.

User Experience.
The app user interface must follow Microsoft design guidelines.
. The user must be able to insert or update service records.
. The user must be able to filter service records by date or service type.
. The user must be able to navigate between various Darts of the app including but not limited to the Mainscreen, Service Record screen, and Owner's Manual screen.
. The user cannot switch between categories by using the Back button.
. The data from the cloud service must automatically populate the Service Record screen.
. The Main screen must have a dark background. All other screens must have a light background with contrasting colored text.
. All multimedia must provide a full-screen mode that can be activated by the end user.
. All media items must start when the user interacts with them and stop immediately when a video ends.
. The app must accept and display tile messages and notification messages from the cloud service.
. The navigational icons must not be displayed if the content of the screen does not require such display. . Any page of the owner's manual must be able to be pinned to the Windows Start screen.
. When the app is pinned to the Windows Start screen all live tile sizes must be available to the users.
. When a specific app page is pinned to the Windows Start screen, the page tile cannot be wider or taller than the dimensions of a wide tile.
. Short names and display names must not be displayed on square tiles.

Development
. The app must use Microsoft Visual Studio preconfigured templates with built-in data structures.
. External notifications must be delivered by using Windows Push Notification Services (WNS).
. The app must be able to receive push notifications from a Microsoft Azure Mobile Services endpoint.
. Each HTML file must be supported by similarly named JavaScript and CSS files (for example, myFile.html, myFile.js myFile.css).
File: main.html

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

```
MH01  <!DOCTYPE html>
MH02  <html>
MH03  <head>
MH04  ...
MH05  <div class="section3" data-win-control="WinJS.UI.HubSection" data-win-options=
"{ isHeaderStatic: true }"
MH06   data-win-res="{ winControl: {'header': 'Section3'} }">
MH07    <div class="top-image-row">
MH08      <video id="playMedia" style="position: relative;"
poster="/images/blank1.jpg"></video>
MH09    </div>
MH10    <div class="sub-image-row">
MH11      <img src="/images/media1.jpg" id="playMedia1" />
MH12      <img src="/images/media2.jpg" id="playMedia2" />
MH13      <img src="/images/media3.jpg" id="playMedia3" />
MH14    </div>
MH15    <div class="win-type-medium" data-win-res="{ textContent: 'DescriptionText'
}"></div>
MH16    <div class="win-type-small">
MH17      <span data-win-res="{ textContent: 'Section3Description' }"></span>
MH18    </div>
MH19  </div>
MH20  ...
MH21  </html>
```

File: main.js

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

```
MJ01  (function () {
MJ02    "use strict";
MJ03    ...
MJ04    });
MJ05  })();
MJ06  function playMedia(e) {
MJ07    return function () {
MJ08      var vid = WinJS.Utilities.query("#playMedia")[0];
MJ09      switch(e)
MJ10      {
MJ11        case 1:
MJ12          src = "media/movie1.mp4";
MJ13          break;
MJ14        case 2:
MJ15          src = "media/movie2.mp4";
MJ16          break;
MJ17        default:
MJ18          src = "media/movie3.mp4";
MJ19      }
MJ20      vid.src = src;
MJ21      vid.play();
MJ22    };
MJ23  }
```

File: manual.html

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

```
OH01  <!DOCTYPE html>
OH02  <html>
OH03  <head>
OH04    <meta charset="utf-8" />
OH05    <title>manual</title>
OH06    <link href="//Microsoft.WinJS.2.0/css/ui-dark.css" rel="stylesheet" />
OH07    <script src="//Microsoft.WinJS.2.0/js/base.js"></script>
OH08    <script src="//Microsoft.WinJS.2.0/js/ui.js"></script>
OH09    <link href="manual.css" rel="stylesheet" />
OH10    <script src="manual.js"></script>
OH11  </head>
OH12  <body>
OH13    <div class="manual fragment">
OH14      <header aria-label="Header content" role="banner">
OH15        <button data-win-control="WinJS.UI.BackButton"></button>
OH16        <h1 class="titlearea win-type-ellipsis">
OH17          <span class="pagetitle">Owner's Manual</span>
OH18        </h1>
OH19      </header>
OH20      <section aria-label="Main content" role="main">
OH21        <p>Owner's Manual Content</p>
OH22      </section>
OH23      <div id="appBar" data-win-control="WinJS.UI.AppBar" data-win-options="">
OH24        <button data-win-control="WinJS.UI.AppBarCommand"
            data-win-options="{id:'cmdPin',label:'Pin To Start',icon:'pin',
            section:'global',tooltip:''}"></button>
OH25      </div>
OH26    </div>
OH27  </body>
OH28  </html>
```

File: manual.js

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

```
OJ01 (function () {
OJ02   "use strict";
OJ03   WinJS.UI.Pages.define("/pages/manual/manual.html", {
OJ04     ready: function (element, options) {
OJ05       document.getElementById("cmdPin").addEventListener("click",
           pinSecondaryTile, false);
OJ06       var navState = {
OJ07       backStack: WinJS.Navigation.history.backStack.slice(0),
OJ08       forwardStack: WinJS.Navigation.history.forwardStack.slice(0),
OJ09       current: WinJS.Navigation.history.current
OJ10       };
OJ11       WinJS.Navigation.history = navState;
OJ12
OJ13     },
OJ14     unload: function () {
OJ15     },
OJ16     updateLayout: function (element) {
OJ17     }
OJ18   });

OJ19   function pinSecondaryTile() {
OJ20     var SecondaryTileId = "SecondaryTile.Logo";
OJ21     var square70x70Logo =
          new Windows.Foundation.Uri("ms-appx:///Images/square70x70Tile-sdk.png");
OJ22     var square150x150Logo =
          new Windows.Foundation.Uri("ms-appx:///Images/square150x150Tile-sdk.png");
OJ23     var wide310x150Logo =
          new Windows.Foundation.Uri("ms-appx:///Images/wide310x150Tile-sdk.png");
OJ24     var square30x30Logo =
          new Windows.Foundation.Uri("ms-appx:///Images/square30x30Tile-sdk.png");
OJ25     var var1 = new Date();
OJ26     var newTileActivationArguments = SecondaryTileId + " WasPinnedAt=" + var1;
OJ27     var tile = new Windows.UI.StartScreen.SecondaryTile(SecondaryTileId,
          "Owner's Manual", newTileActivationArguments,
          square150x150Logo,
          Windows.UI.StartScreen.TileSize.Square150x150);
OJ28     tile.visualElements.wide310x150Logo = wide310x150Logo;
OJ29     tile.visualElements.square30x30Logo = square30x30Logo;
OJ30
OJ31     tile.visualElements.foregroundText = Windows.UI.StartScreen.ForegroundText
         .dark;
OJ32     tile.requestCreateForSelectionAsync({ x:0, y:0, width: 640, height: 400 },
          Windows.UI.Popups.Placement.below);
OJ33   }
OJ34 })();
```

File: service.html

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

```
SH01  <!DOCTYPE html>
SH02  <html>
SH03  <head>
SH04    ...
SH05  </head>
SH06  <body>
SH07
SH08  </body>
SH09  </html>
```

File: service.js

Relevant portions of the app files are shown below. Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.

```
SJ01  (function () {
SJ02    "use strict";
SJ03    WinJS.UI.Pages.define("/pages/service/service.html", {
SJ04      ready: function (element, options) {
SJ05        var buttonSave = document.getElementById("btnAddServiceRecord");
SJ06        var client = new WindowsAzure.MobileServiceClient(
SJ07          "https://cardataservice17.azure-mobile.net/","myPrivateKey"
SJ08        );
SJ09        ...
SJ10    });

SJ11    function updateTile() {
SJ12      var Notification = Notification;
SJ13      var square310x310Xml = Notification.TileUpdateManager.getTemplateContent
           (Notification.TileTemplateType.tileSquare310x310Text05);
SJ14      square310x310Xml.getElementsByTagName("text")[0].setAttribute("id", "1");
SJ15      square310x310Xml.getElementsByTagName("text")[0].appendChild
           (square310x310Xml.createTextNode("My Alert"));
SJ16      var wide310x150Xml = Notification.TileUpdateManager.getTemplateContent
           (Notification.TileTemplateType.tileWide310x150Text03);
SJ17      var tileTextAttributes = wide310x150Xml.getElementsByTagName("text");
SJ18      tileTextAttributes[0].appendChild(wide310x150Xml.createTextNode
          ("My Alert"));
SJ19      var square150x150Xml = Notification.TileUpdateManager.getTemplateContent
           (Notification.TileTemplateType.tileSquare150x150Text04);
SJ20      var squareTileTextAttributes = square150x150Xml.getElementsByTagName
          ("text");
SJ21      squareTileTextAttributes[0].appendChild(square150x150Xml.createTextNode
          ("My Alert"));
SJ22      var node =
           square310x310Xml.importNode(square150x150Xml.getElementsByTagName
           ("binding").item(0), true);
SJ23      square310x310Xml.getElementsByTagName("visual").item(0).appendChild(node);
SJ24      node =
           square310x310Xml.importNode(wide310x150Xml.getElementsByTagName("binding")
           .item(0), true);
SJ25      square310x310Xml.getElementsByTagName("visual").item(0).appendChild(node);
SJ26      var appNotification = new Notification.TileNotification(square310x310Xml);
SJ27
SJ28    }
SJ29  })();
```

You need to implement the navigation between screen categories.
What should you do?
A. Place one button for each category on every screen and use the WinJS.navigate command to go to the category screens.
B. Implement category navigation controls on the nav bar on every screen.
C. Implement category navigation controls on the app bar on every screen.
D. Place one link for each category on every screen and use an <href> tag to go to the category screens.
Answer: B