VideoLib - sample application dev guide

Windows 8 Store Application

About the document

The tutorial contains the steps required to build a Windows Store application using HTML5 and javascript.

About the application

The application allows the user to see a list of videos organized in groups. The video list is downloaded from a REST/JSON based service exposed the by http://www.vimeo.com

It is not a fully functional application. It only outlines some of the basic APIs and approaches used to build Windows Store applications with HTML5 and javascript.

The application provides the following features:

- Consuming JSON payload from a REST service
- Displaying of multiple grouped items on the screen together with titles, descriptions and images
- Using of roaming settings to guarantee roam-free user experience
- Data sharing using the Share charm
- Using of Promise class to perform chained async operations

Getting started

1. Create a new Blank javascript Windows Store application named VideoLib

2. Run the application and then break and go back to VS2012

Implementing the data access side

1. Create a new javascript file under the /JS folder called data.js
2. Place the following code inside:

```
(function () {
    "use strict";

    WinJS.Namespace.define("Data", {
        list: new WinJS.Binding.List(),
        items: undefined,
        groups: undefined,
        getItemReference: getItemReference,
        getItemsFromGroup: getItemsFromGroup,
        resolveGroupReference: resolveGroupReference,
        resolveItemReference: resolveItemReference,
        downloadVideos: downloadVideos,
        sampleGroups: [],
        currentItem: null

    });

})();
```

The code above creates a namespace called Data and exposes several properties and methods.
You can think of them as static properties and methods.

We will implement the methods later. You can still run the app although some of the methods are not yet implemented unless you do not invoke them.

3. Let's implement some utility methods used by the UI to display the grouped items and the groups. Note that we use the Data namespace to assign properties and methods. Paste the following code after the Data namespace declaration, but before the closing brackets.

```
    Data.items = Data.list.createGrouped(
                    function groupKeySelector(item) { return item.group.key; },
                    function groupDataSelector(item) { return item.group; }
            );
    Data.groups = Data.items.groups;
    // Get a reference for an item, using the group key and item title as a
    // unique reference to the item that can be easily serialized.
    function getItemReference(item) {
        return [item.group.key, item.title];
```

```
    }

    // This function returns a WinJS.Binding.List containing only the items
    // that belong to the provided group.
    function getItemsFromGroup(group) {
        return Data.list.createFiltered(function (item) { return item.group.key ===
group.key; });
    }

    // Get the unique group corresponding to the provided group key.
    function resolveGroupReference(key) {
        for (var i = 0; i < Data.groups.length; i++) {
            if (Data.groups.getAt(i).key === key) {
                return Data.groups.getAt(i);
            }
        }
    }

    // Get a unique item from the provided string array, which should contain a
    // group key and an item title.
    function resolveItemReference(reference) {
        for (var i = 0; i < Data.items.length; i++) {
            var item = Data.items.getAt(i);
            if (item.group.key === reference[0] && item.title === reference[1]) {
                return item;
            }
        }
    }
```

4. Let's implement the REST we service consuming routines.
   Paste the following code before the closing brackets of the data.js file and after the utility methods above:

```
function downloadGroupVideos(groupName) {
        var videos = [];
        var promise = WinJS.xhr({ url: "http://vimeo.com/api/v2/group/" + groupName +
"/videos.json" });
        return promise.then(function (xhr) {
            var parsed = JSON.parse(xhr.responseText);
            for (var i = 0; i < parsed.length; i++) {
                videos.push(parsed[i]);
            }
            return new WinJS.Promise.wrap(videos);
        }
       );
}
```

The function implements code to download a video list from Viemo for a given group/category. It receives JSON which is parsed to produce an array of video items. Each video item contains description, title, image urls , video urls and others.
Note the usage of promises to assure that the function can be invoked and chained in async manner. It actually returns a promise object wrapping the videos array.

5. Let's implement a function to download all the videos from a 3 predefined categories published from the Vimeo video service.

```
    // Returns an array of sample data that can be added to the application's
    // data list.
    function downloadVideos() {
        return new WinJS.Promise(function (complete, error, progress) {
            //clear all videos
            while (Data.list.length > 0) Data.list.pop();
            // These three strings encode placeholder images. You will want to set the
            // backgroundImage property in your real data to be URLs to images.
            var darkGray =
"
nQU1BAACxjwv8YQUAAAAJcEhZcwAADsQAAA7EAZUrDhsAAAANSURBVBhXY3B0cPoPAANMAcOba1BlAAAAAElFTkSuQ
mCC";
            var lightGray =
"
nQU1BAACxjwv8YQUAAAAJcEhZcwAADsQAAA7EAZUrDhsAAAANSURBVBhXY7h4+cp/AAhpA3h+ANDKAAAAAElFTkSuQ
mCC";
            var mediumGray =
"
nQU1BAACxjwv8YQUAAAAJcEhZcwAADsQAAA7EAZUrDhsAAAANSURBVBhXY5g8dcZ/AAY/AsAlWFQ+AAAAAElFTkSuQ
mCC";
            // Each of these sample groups must have a unique key to be displayed
            // separately.
            Data.sampleGroups = [
                { key: "filmschool", title: "Films School", subtitle: "Group Subtitle: 1",
backgroundImage: darkGray, description: "" },
                { key: "shortfilms", title: "Short Films", subtitle: "Group Subtitle: 2",
backgroundImage: lightGray, description: "" },
                { key: "animation", title: "Animation", subtitle: "Group Subtitle: 3",
backgroundImage: mediumGray, description: "" },
            ];
            var showdescr =
(Windows.Storage.ApplicationData.current.roamingSettings.values["showdescr"]);
            Data.sampleGroups.map(
                function (group) {
                    downloadGroupVideos(group.key).then(function (videos) {
                        for (var i = 0; i < videos.length; i++) {

                            Data.list.push(
                                {
                                    group: group,
                                    title: videos[i].title,
                                    description: (showdescr) ? videos[i].description : "",
                                    content: '',
                                    backgroundImage: videos[i].thumbnail_large,
                                    url: videos[i].url
                                }
                                );
                        }
                        if (complete)
```

```
                            complete();
                        }

                    );
                    }
                );

            });
        }
```

The code again uses promises to allow async and chainable usage of the function. We have a predefined array of video categories (groups) and we invoke the DownloadGroupVideos function for each of them. It adds each of the video items in the Data.list array. We actually will later 'bind' our UI to this array in order to display the videos on the screen.

Declare data.js in default.html above default.js:

```
  <script src="/js/data.js"></script>
```

Implementing UI

1. Add new file calls navigator.js under the /JS folder and paste the following code:

```
(function () {
    "use strict";

    var appView = Windows.UI.ViewManagement.ApplicationView;
    var nav = WinJS.Navigation;

    WinJS.Namespace.define("Application", {
        PageControlNavigator: WinJS.Class.define(
            // Define the constructor function for the PageControlNavigator.
            function PageControlNavigator(element, options) {
                this._element = element || document.createElement("div");
                this._element.appendChild(this._createPageElement());

                this.home = options.home;
                this._lastViewstate = appView.value;

                nav.onnavigated = this._navigated.bind(this);
                window.onresize = this._resized.bind(this);

                document.body.onkeyup = this._keyupHandler.bind(this);
                document.body.onkeypress = this._keypressHandler.bind(this);
                document.body.onmspointerup = this._mspointerupHandler.bind(this);

                Application.navigator = this;
            }, {
                home: "",
                /// <field domElement="true" />
                _element: null,
                _lastNavigationPromise: WinJS.Promise.as(),
                _lastViewstate: 0,

                // This is the currently loaded Page object.
                pageControl: {
                    get: function () { return this.pageElement &&
this.pageElement.winControl; }
                },

                // This is the root element of the current page.
                pageElement: {
                    get: function () { return this._element.firstElementChild; }
                },

                // Creates a container for a new page to be loaded into.
                _createPageElement: function () {
                    var element = document.createElement("div");
                    element.style.width = "100%";
                    element.style.height = "100%";
                    return element;
                },

                // Retrieves a list of animation elements for the current page.
                // If the page does not define a list, animate the entire page.
                _getAnimationElements: function () {
                    if (this.pageControl && this.pageControl.getAnimationElements) {
                        return this.pageControl.getAnimationElements();
                    }
```

```
                    return this.pageElement;
                },

                // Navigates back whenever the backspace key is pressed and
                // not captured by an input field.
                _keypressHandler: function (args) {
                    if (args.key === "Backspace") {
                        nav.back();
                    }
                },

                // Navigates back or forward when alt + left or alt + right
                // key combinations are pressed.
                _keyupHandler: function (args) {
                    if ((args.key === "Left" && args.altKey) || (args.key ===
"BrowserBack")) {
                        nav.back();
                    } else if ((args.key === "Right" && args.altKey) || (args.key ===
"BrowserForward")) {
                        nav.forward();
                    }
                },

                // This function responds to clicks to enable navigation using
                // back and forward mouse buttons.
                _mspointerupHandler: function (args) {
                    if (args.button === 3) {
                        nav.back();
                    } else if (args.button === 4) {
                        nav.forward();
                    }
                },

                // Responds to navigation by adding new pages to the DOM.
                _navigated: function (args) {
                    var newElement = this._createPageElement();
                    var parentedComplete;
                    var parented = new WinJS.Promise(function (c) { parentedComplete = c;
});

                    this._lastNavigationPromise.cancel();

                    this._lastNavigationPromise = WinJS.Promise.timeout().then(function ()
{
                        return WinJS.UI.Pages.render(args.detail.location, newElement,
args.detail.state, parented);
                    }).then(function parentElement(control) {
                        var oldElement = this.pageElement;
                        if (oldElement.winControl && oldElement.winControl.unload) {
                            oldElement.winControl.unload();
                        }
                        this._element.appendChild(newElement);
                        this._element.removeChild(oldElement);
                        oldElement.innerText = "";
                        this._updateBackButton();
                        parentedComplete();
                        WinJS.UI.Animation.enterPage(this._getAnimationElements()).done();
                    }.bind(this));
```

```
                    args.detail.setPromise(this._lastNavigationPromise);
                },

                // Responds to resize events and call the updateLayout function
                // on the currently loaded page.
                _resized: function (args) {
                    if (this.pageControl && this.pageControl.updateLayout) {
                        this.pageControl.updateLayout.call(this.pageControl,
this.pageElement, appView.value, this._lastViewstate);
                    }
                    this._lastViewstate = appView.value;
                },

                // Updates the back button state. Called after navigation has
                // completed.
                _updateBackButton: function () {
                    var backButton = this.pageElement.querySelector("header[role=banner]
.win-backbutton");
                    if (backButton) {
                        backButton.onclick = function () { nav.back(); };

                        if (nav.canGoBack) {
                            backButton.removeAttribute("disabled");
                        } else {
                            backButton.setAttribute("disabled", "disabled");
                        }
                    }
                },
            }
        )
    });
})();
```

This code implements the navigation behavior of the application. It makes the navigation between the pages possible.
Declare the navigator.js script in default.html by adding the following line above the default.js script declaration:

```
<script src="/js/navigator.js"></script>
```

This will make sure that navigator.js is invoked before default.js
Replace the content inside the body tag of default.html with the following:

```
<div id="contenthost" data-win-control="Application.PageControlNavigator" data-win-
options="{home: '/pages/groupedItems/groupedItems.html'}"></div>
    <div id="appbar" data-win-control="WinJS.UI.AppBar">
        <button data-win-control="WinJS.UI.AppBarCommand" data-win-
options="{id:'btnRefresh', label:'Refresh', icon:'refresh'}" type="button"></button>
    </div>
```

Open default.js and replace the onactivated event handler with the following:

```
if (args.detail.kind === activation.ActivationKind.launch) {
            if (args.detail.previousExecutionState !==
activation.ApplicationExecutionState.terminated) {
```

```
            } else {

            }

            if (app.sessionState.history) {
                nav.history = app.sessionState.history;
            }
            args.setPromise(WinJS.UI.processAll().then(function () {
                if (nav.location) {
                    nav.history.current.initialPlaceholder = true;
                    return nav.navigate(nav.location, nav.state);
                } else {
                    return nav.navigate(Application.navigator.home);
                }
            }));
 }
```

The code 'links' the navigator.js code with the startup of the app.
Add the following line on top of the file after the app variable declaration.

```
var nav = WinJS.Navigation;
```

Adding pages

It is time to create our home page – the first page which will be displayed when the app starts.

1. Create new folder called pages
2. Create new sub-folder under the pages folder called groupedItems
3. Create new html file called groupedItems.html
4. Paste the following html inside:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>groupedItemsPage</title>

    <!-- WinJS references -->
    <link href="//Microsoft.WinJS.1.0/css/ui-dark.css" rel="stylesheet" />
    <script src="//Microsoft.WinJS.1.0/js/base.js"></script>
    <script src="//Microsoft.WinJS.1.0/js/ui.js"></script>

    <link href="/css/default.css" rel="stylesheet" />
    <link href="/pages/groupedItems/groupedItems.css" rel="stylesheet" />
    <script src="/js/data.js"></script>
    <script src="/pages/groupedItems/groupedItems.js"></script>
</head>
<body>
    <!-- These templates are used to display each item in the ListView declared below. -->
    <div class="headertemplate" data-win-control="WinJS.Binding.Template">
        <button class="group-header win-type-x-large win-type-interactive" data-win-
bind="groupKey: key"

onclick="Application.navigator.pageControl.navigateToGroup(event.srcElement.groupKey)"
role="link" tabindex="-1" type="button">
```

```
            <span class="group-title win-type-ellipsis" data-win-bind="textContent:
title"></span>
            <span class="group-chevron"></span>
        </button>
    </div>
    <div class="itemtemplate" data-win-control="WinJS.Binding.Template">
        <div class="item">

            <img class="item-image" src="#" data-win-bind="src: backgroundImage; alt:
title" />

            <div class="item-overlay">
                <h4 class="item-title" data-win-bind="textContent: title"></h4>
                <h6 class="item-subtitle win-type-ellipsis" data-win-bind="textContent:
description"></h6>
            </div>
        </div>
    </div>

    <!-- The content that will be loaded and displayed. -->
    <div class="fragment groupeditemspage">
        <header aria-label="Header content" role="banner">
            <button class="win-backbutton" aria-label="Back" disabled
type="button"></button>
            <h1 class="titlearea win-type-ellipsis">
                <span class="pagetitle">Video Lib</span>
            </h1>
        </header>
        <section aria-label="Main content" role="main">
            <div class="groupeditemslist win-selectionstylefilled" aria-label="List of
groups" data-win-control="WinJS.UI.ListView"
                data-win-options="{ selectionMode: 'none' }"></div>
        </section>
    </div>
</body>
</html>
```

Run the application – it has to start and display the title of the app

5. Create new file called groupedItems.js under the /pages/groupedItems folder
6. Paste the following code:

```
(function () {
    "use strict";

    var appView = Windows.UI.ViewManagement.ApplicationView;
    var appViewState = Windows.UI.ViewManagement.ApplicationViewState;
    var nav = WinJS.Navigation;
    var ui = WinJS.UI;

    ui.Pages.define("/pages/groupedItems/groupedItems.html", {
        // Navigates to the groupHeaderPage. Called from the groupHeaders,
        // keyboard shortcut and iteminvoked.
        navigateToGroup: function (key) {
            nav.navigate("/pages/groupDetail/groupDetail.html", { groupKey: key });
        },

        // This function is called whenever a user navigates to this page. It
        // populates the page elements with the app's data.
        ready: function (element, options) {
            var listView = element.querySelector(".groupeditemslist").winControl;
            listView.groupHeaderTemplate = element.querySelector(".headertemplate");
            listView.itemTemplate = element.querySelector(".itemtemplate");
            listView.oniteminvoked = this._itemInvoked.bind(this);

            // Set up a keyboard shortcut (ctrl + alt + g) to navigate to the
            // current group when not in snapped mode.
            listView.addEventListener("keydown", function (e) {
                if (appView.value !== appViewState.snapped && e.ctrlKey && e.keyCode ===
WinJS.Utilities.Key.g && e.altKey) {
                    var data =
listView.itemDataSource.list.getAt(listView.currentItem.index);
                    this.navigateToGroup(data.group.key);
                    e.preventDefault();
                    e.stopImmediatePropagation();
                }
            }.bind(this), true);

            this._initializeLayout(listView, appView.value);
            listView.element.focus();



        },

        // This function updates the page layout in response to viewState changes.
        updateLayout: function (element, viewState, lastViewState) {
            /// <param name="element" domElement="true" />

            var listView = element.querySelector(".groupeditemslist").winControl;
            if (lastViewState !== viewState) {
                if (lastViewState === appViewState.snapped || viewState ===
appViewState.snapped) {
                    var handler = function (e) {
                        listView.removeEventListener("contentanimating", handler, false);
                        e.preventDefault();
                    }
                    listView.addEventListener("contentanimating", handler, false);
                    this._initializeLayout(listView, viewState);
                }
```

```
            }
        },

        // This function updates the ListView with new layouts
        _initializeLayout: function (listView, viewState) {
            /// <param name="listView" value="WinJS.UI.ListView.prototype" />

            if (viewState === appViewState.snapped) {
                listView.itemDataSource = Data.groups.dataSource;
                listView.groupDataSource = null;
                listView.layout = new ui.ListLayout();
            } else {
                listView.itemDataSource = Data.items.dataSource;
                listView.groupDataSource = Data.groups.dataSource;
                listView.layout = new ui.GridLayout({ groupHeaderPosition: "top" });
            }
        },

        _itemInvoked: function (args) {
            if (appView.value === appViewState.snapped) {
                // If the page is snapped, the user invoked a group.
                var group = Data.groups.getAt(args.detail.itemIndex);
                this.navigateToGroup(group.key);
            } else {
                // If the page is not snapped, the user invoked an item.
                var item = Data.items.getAt(args.detail.itemIndex);
                nav.navigate("/pages/itemDetail/itemDetail.html", { item:
Data.getItemReference(item) });
            }
        }
    });
})();
```

The code implements a single page displaying the videos(items) grouped by video categories. It 'overrides' each of the page's lifecycle functions(methods).

7. Styling the page:
   -Create new file groupedItems.css under pages/groupedItems
   -Paste the following code:

```
.groupeditemspage section[role=main] {
    -ms-grid-row: 1;
    -ms-grid-row-span: 2;
}
.groupeditemspage .groupeditemslist {
    height: 100%;
    position: relative;
    width: 100%;
    z-index: 0;
}
    /* This selector is used to prevent ui-dark/light.css from overwriting changes
       to .win-surface. */
    .groupeditemspage .groupeditemslist .win-horizontal.win-viewport .win-surface {
        margin-bottom: 60px;
        margin-left: 45px;
        margin-right: 115px;
```

```
        margin-top: 128px;
    }
    .groupeditemspage .groupeditemslist .win-groupheader {
        padding: 0;
    }
    /* Use grid and top level layout for truncation */
    .groupeditemspage .groupeditemslist .group-header {
        -ms-grid-columns: minmax(0, max-content) 7px max-content;
        -ms-grid-rows: max-content;
        display: -ms-inline-grid;
        line-height: 1.5;
    }
    /* Override default button styles */
    .groupeditemspage .groupeditemslist .group-header, .group-header:hover, .group-
header:hover:active {
        background: transparent;
        border: 0;
        margin-bottom: 1px;
        margin-left: 5px;
        margin-right: 5px;
        margin-top: 1px;
        min-height: 0;
        padding: 0;
    }
    .groupeditemspage .groupeditemslist .group-header .group-title {
            display: inline-block;
    }
    .groupeditemspage .groupeditemslist .group-header .group-chevron {
            -ms-grid-column: 3;
            display: inline-block;
    }
    .groupeditemspage .groupeditemslist .group-header .group-chevron::before {
                content: "\E26B";
                font-family: 'Segoe UI Symbol';
     }
  .groupeditemspage .groupeditemslist .item {
        -ms-grid-columns: 1fr;
        -ms-grid-rows: 1fr 90px;
        display: -ms-grid;
        height: 250px;
        width: 250px;
    }

        .groupeditemspage .groupeditemslist .item .item-image {
            -ms-grid-row-span: 2;
        }

        .groupeditemspage .groupeditemslist .item .item-overlay {
            -ms-grid-row: 2;
            -ms-grid-rows: 1fr 21px;
            display: -ms-grid;
            padding: 6px 15px 2px 15px;
        }

            .groupeditemspage .groupeditemslist .item .item-overlay .item-title {
                -ms-grid-row: 1;
                overflow: hidden;
                width: 220px;
```

```
            }

            .groupeditemspage .groupeditemslist .item .item-overlay .item-subtitle {
                -ms-grid-row: 2;
                width: 220px;
            }

@media screen and (-ms-view-state: fullscreen-landscape), screen and (-ms-view-state:
fullscreen-portrait), screen and (-ms-view-state: filled) {
    .groupeditemspage .groupeditemslist .item .item-overlay {
        background: rgba(0,0,0,0.65);
    }

        .groupeditemspage .groupeditemslist .item .item-overlay .item-title {
            color: rgba(255,255,255,0.87);
        }

        .groupeditemspage .groupeditemslist .item .item-overlay .item-subtitle {
            color: rgba(255,255,255,0.6);
        }
}

@media screen and (-ms-view-state: fullscreen-landscape) and (-ms-high-contrast), screen
and (-ms-view-state: fullscreen-portrait) and (-ms-high-contrast), screen and (-ms-view-
state: filled) and (-ms-high-contrast) {
    .groupeditemspage .groupeditemslist .item .item-overlay {
        color: WindowText;
    }
}

@media screen and (-ms-view-state: snapped) {
    .groupeditemspage section[role=main] {
        -ms-grid-row: 2;
        -ms-grid-row-span: 1;
    }

    .groupeditemspage .groupeditemslist .win-vertical.win-viewport .win-surface {
        margin-bottom: 30px;
        margin-top: 0;
    }

    .groupeditemspage .groupeditemslist .win-container {
        margin-bottom: 15px;
        margin-left: 13px;
        margin-right: 35px;
        padding: 7px;
    }

    .groupeditemspage .groupeditemslist .item {
        -ms-grid-columns: 60px 1fr;
        -ms-grid-rows: 1fr;
        display: -ms-grid;
        height: 60px;
        width: 272px;
    }

        .groupeditemspage .groupeditemslist .item .item-image {
            -ms-grid-column: 1;
```

```
            -ms-grid-row-span: 1;
            height: 60px;
            width: 60px;
        }

        .groupeditemspage .groupeditemslist .item .item-overlay {
            -ms-grid-column: 2;
            -ms-grid-row: 1;
            -ms-grid-row-align: stretch;
            background: transparent;
            display: inline-block;
            margin-left: 10px;
            padding: 0;
        }

            .groupeditemspage .groupeditemslist .item .item-overlay .item-title {
                margin-top: 4px;
                max-height: 40px;
                width: 202px;
            }

            .groupeditemspage .groupeditemslist .item .item-overlay .item-subtitle {
                opacity: 0.6;
                width: 202px;
            }
}

@media screen and (-ms-view-state: fullscreen-portrait) {
    .groupeditemspage .groupeditemslist .win-horizontal.win-viewport .win-surface {
        margin-left: 25px;
    }
}
```

8. Create new file defaut.css under the /CS folder

```
#contenthost {
    height: 100%;
```

```
    width: 100%;
}

.fragment {
    /* Define a grid with rows for a banner and a body */
    -ms-grid-columns: 1fr;
    -ms-grid-rows: 128px 1fr;
    display: -ms-grid;
    height: 100%;
    width: 100%;
}

    .fragment header[role=banner] {
        /* Define a grid with columns for the back button and page title. */
        -ms-grid-columns: 120px 1fr;
        -ms-grid-rows: 1fr;
        display: -ms-grid;
    }

        .fragment header[role=banner] .win-backbutton {
            margin-left: 39px;
            margin-top: 59px;
        }

        .fragment header[role=banner] .titlearea {
            -ms-grid-column: 2;
            margin-top: 37px;
        }

            .fragment header[role=banner] .titlearea .pagetitle {
                width: calc(100% - 20px);
            }

    .fragment section[role=main] {
        -ms-grid-row: 2;
        height: 100%;
        width: 100%;
    }

@media screen and (-ms-view-state: snapped) {
    .fragment header[role=banner] {
        -ms-grid-columns: auto 1fr;
        margin-left: 20px;
    }

        .fragment header[role=banner] .win-backbutton {
            margin: 0;
            margin-right: 10px;
            margin-top: 76px;
        }

            .fragment header[role=banner] .win-backbutton:disabled {
                display: none;
            }

        .fragment header[role=banner] .titlearea {
            -ms-grid-column: 2;
            margin-left: 0;
```

```
            margin-top: 68px;
        }
}

@media screen and (-ms-view-state: fullscreen-portrait) {
    .fragment header[role=banner] {
        -ms-grid-columns: 100px 1fr;
    }

        .fragment header[role=banner] .win-backbutton {
            margin-left: 29px;
        }
}
```

Let's call the video downloading routine and check if the videos will appear on the screen:

9. Open default.js and replace the args.setPromise(…) code with the following code:

```
  args.setPromise(WinJS.UI.processAll().then(function () {
                if (nav.location) {
                    nav.history.current.initialPlaceholder = true;
                    return nav.navigate(nav.location, nav.state);
                } else {
                    return nav.navigate(Application.navigator.home);
                }
            }).then(function () {
                btnRefresh.onclick = function (e) {
                    refreshTheData();
                };
                refreshTheData();
            }));
```

The code invokes the refreshData() function after the UI enhancement passes.
Let's implement the refreshData() function in default.js:

```
function refreshTheData() {
        var promise = Data.downloadVideos();
        return promise.then(
                function () {
                    tiles.displayTileNotification(Data.items.length,
Data.sampleGroups.length);
                }
            );
    }
```

The refreshTheData() function invokes tiles.displayTileNotification() function to update the primary tile of the application. Let's implement it.

1. Create new file tiles.js under the /JS folder
2. Paste the following code:

```
(function () {
    "use strict";

    WinJS.Namespace.define("tiles", {
        displayTileNotification: function (title, subtitle) {

            var notifications = Windows.UI.Notifications;
            var squareTemplate = notifications.TileTemplateType.tileSquareBlock;
            var squareTileXml =
notifications.TileUpdateManager.getTemplateContent(squareTemplate);
            var squareTileTextAttributes =
squareTileXml.selectSingleNode('//text[@id="1"]');
            squareTileTextAttributes.appendChild(squareTileXml.createTextNode(title));
            squareTileTextAttributes =
squareTileXml.selectSingleNode('//text[@id="2"]');

squareTileTextAttributes.appendChild(squareTileXml.createTextNode(subtitle));
            var tileNotification = new notifications.TileNotification(squareTileXml);

notifications.TileUpdateManager.createTileUpdaterForApplication().update(tileNotificat
ion);

        }
    }
   )

})();
```

3. Declare the js file in default.html above default.js

```
<script src="/js/tiles.js"></script>
```

Adding the ItemDetail page

1. Create new folder /pages/itemDetail
2. Create new file /pages/itemDetail/itemDetail.html
3. Paste the following code inside:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>itemDetailPage</title>

    <!-- WinJS references -->
    <link href="//Microsoft.WinJS.1.0/css/ui-dark.css" rel="stylesheet" />
    <script src="//Microsoft.WinJS.1.0/js/base.js"></script>
    <script src="//Microsoft.WinJS.1.0/js/ui.js"></script>

    <link href="/css/default.css" rel="stylesheet" />
    <link href="/pages/itemDetail/itemDetail.css" rel="stylesheet" />
    <script src="/js/data.js"></script>
    <script src="/pages/itemDetail/itemDetail.js"></script>
</head>
<body>
    <!-- The content that will be loaded and displayed. -->
    <div class="itemdetailpage fragment">
        <header aria-label="Header content" role="banner">
            <button class="win-backbutton" aria-label="Back" disabled
type="button"></button>
            <h1 class="titlearea win-type-ellipsis">
                <span class="pagetitle"></span>
            </h1>
        </header>
        <div class="content" aria-label="Main content" role="main">
            <article>
                <div>
                    <header>
                        <h2 class="item-title"></h2>
                        <h4 class="item-subtitle"></h4>
                    </header>
                    <img class="item-image" src="#" />
                    <div class="item-content"></div>
                    <a class="item-watch" >Watch</a>
                </div>
            </article>
        </div>
    </div>
</body>
</html>
```

4. Create new file /pages/itemDetail/itemDetail.js
5. Paste the following:

```
(function () {
    "use strict";

    WinJS.UI.Pages.define("/pages/itemDetail/itemDetail.html", {
        // This function is called whenever a user navigates to this page. It
        // populates the page elements with the app's data.
        ready: function (element, options) {
            var dataTransferManager =
Windows.ApplicationModel.DataTransfer.DataTransferManager.getForCurrentView();
```

```
            dataTransferManager.addEventListener("datarequested", this.datarequest);

            var item = options && options.item ?
Data.resolveItemReference(options.item) : Data.items.getAt(0);
            element.querySelector(".titlearea .pagetitle").textContent =
item.group.title;
            element.querySelector("article .item-title").textContent = item.title;
            element.querySelector("article .item-subtitle").textContent =
item.description;
            element.querySelector("article .item-image").src = item.backgroundImage;
            element.querySelector("article .item-image").alt = item.subtitle;
            element.querySelector("article .item-content").innerHTML = item.content;
            var aa = element.querySelector("article .item-watch");
            aa.href = item.url;
            element.querySelector(".content").focus();
            Data.currentItem = item;


        },
        unload: function () {

        }

    })
}());
```

6. Create new file /pages/itemDetail/itemDetail.css
7. Paste the following code inside:

```
.itemdetailpage .content {
    -ms-grid-row: 1;
    -ms-grid-row-span: 2;
    display: block;
    height: 100%;
    overflow-x: auto;
    position: relative;
    width: 100%;
    z-index: 0;
}

    .itemdetailpage .content article {
        /* Define a multi-column, horizontally scrolling article by default. */
        column-fill: auto;
        column-gap: 80px;
        column-width: 480px;
        height: calc(100% - 183px);
        margin-left: 120px;
        margin-top: 133px;
        width: 480px;
    }

        .itemdetailpage .content article header .item-title {
            margin-bottom: 19px;
        }

        .itemdetailpage .content article header .item-subtitle,item-watch {
            margin-bottom: 16px;
            margin-top: 0;
        }
```

```
        .itemdetailpage .content article .item-image {
            height: 240px;
            width: 460px;
        }

        .itemdetailpage .content article .item-content p {
            margin-top: 10px;
            margin-bottom: 20px;
            margin-right: 20px;
        }

@media screen and (-ms-view-state: snapped) {
    .itemdetailpage .content {
        -ms-grid-row: 2;
        -ms-grid-row-span: 1;
        overflow-x: hidden;
        overflow-y: auto;
    }

        .itemdetailpage .content article {
            /* Define a single column, vertically scrolling article in snapped mode.
*/
            -ms-grid-columns: 300px 1fr;
            -ms-grid-row: 2;
            -ms-grid-rows: auto 60px;
            display: -ms-grid;
            height: 100%;
            margin-left: 20px;
            margin-top: 6px;
            width: 300px;
        }

            .itemdetailpage .content article header .item-title {
                margin-bottom: 10px;
            }

            .itemdetailpage .content article .item-image {
                height: 140px;
                width: 280px;
            }

            .itemdetailpage .content article .item-content {
                padding-bottom: 60px;
            }
}

@media screen and (-ms-view-state: fullscreen-portrait) {
    .itemdetailpage .content article {
        margin-left: 100px;
    }
}
```

Adding the groupDetail page

1. Create new folder /pages/groupDetail
2. Create new file /pages/groupDetail/groupedItems.html
3. Put the following HTML inside:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>groupDetailPage</title>

    <!-- WinJS references -->
    <link href="//Microsoft.WinJS.1.0/css/ui-dark.css" rel="stylesheet" />
    <script src="//Microsoft.WinJS.1.0/js/base.js"></script>
    <script src="//Microsoft.WinJS.1.0/js/ui.js"></script>

    <link href="/css/default.css" rel="stylesheet" />
    <link href="/pages/groupDetail/groupDetail.css" rel="stylesheet" />
    <script src="/js/data.js"></script>
    <script src="/pages/groupDetail/groupDetail.js"></script>
</head>
<body>
    <!-- These templates are used to display each item in the ListView declared below. -->
    <div class="headertemplate" data-win-control="WinJS.Binding.Template">
        <h2 class="group-subtitle" data-win-bind="textContent: description"></h2>
        <img class="group-image" src="#" data-win-bind="src: backgroundImage; alt: title"
/>
    </div>
    <div class="itemtemplate" data-win-control="WinJS.Binding.Template">
        <div class="item">
            <img class="item-image" src="#" data-win-bind="src: backgroundImage; alt:
title" />
            <div class="item-info">
                <h4 class="item-title" data-win-bind="textContent: title"></h4>
                <h6 class="item-subtitle win-type-ellipsis" data-win-bind="textContent:
description"></h6>
                <h4 class="item-watch"><a data-win-bind="href: url">Watch</a></h4>
            </div>
        </div>
    </div>

    <!-- The content that will be loaded and displayed. -->
    <div class="groupdetailpage fragment">
        <header aria-label="Header content" role="banner">
            <button class="win-backbutton" aria-label="Back" disabled
type="button"></button>
            <h1 class="titlearea win-type-ellipsis">
                <span class="pagetitle"></span>
            </h1>
        </header>
        <section aria-label="Main content" role="main">
            <div class="itemslist win-selectionstylefilled" aria-label="List of this
group's items"
```

```
                data-win-control="WinJS.UI.ListView" data-win-options="{ selectionMode:
'none' }"></div>
        </section>
    </div>
</body>
</html>
```

4. Create new file /pages/groupDetail/groupDetail.css
5. Paste the following code inside:

```
.groupdetailpage section[role=main] {
    -ms-grid-row: 1;
    -ms-grid-row-span: 2;
}

/* This selector is used to prevent ui-dark/light.css from overwriting changes
   to .win-surface. */
.groupdetailpage .itemslist .win-horizontal.win-viewport .win-surface {
    margin-bottom: 35px;
    margin-left: 120px;
    margin-right: 120px;
    margin-top: 128px;
}

.groupdetailpage .itemslist {
    height: 100%;
    position: relative;
    width: 100%;
    z-index: 0;
}

    .groupdetailpage .itemslist .win-groupheader {
        -ms-grid-columns: 1fr;
        -ms-grid-rows: auto auto 1fr;
        display: -ms-grid;
        height: 100%;
        margin-left: 0px;
        margin-right: 13px;
        padding: 0;
        width: 480px;
    }

        .groupdetailpage .itemslist .win-groupheader .group-subtitle {
            -ms-grid-row: 1;
            margin-bottom: 14px;
            margin-top: 6px;
            max-height: 48pt;
            overflow: hidden;
        }

        .groupdetailpage .itemslist .win-groupheader .group-image {
            -ms-grid-row: 2;
            background-color: rgba(147, 149, 152, 1);
            height: 238px;
            margin: 0;
            margin-bottom: 20px;
            width: 480px;
```

```
        }

        .groupdetailpage .itemslist .win-groupheader .group-description {
            -ms-grid-row: 3;
            margin-bottom: 55px;
            overflow: hidden;
        }

    .groupdetailpage .itemslist .win-container {
        margin-bottom: 15px;
        margin-right: 0;
        margin-left: 60px;
        padding: 7px;
    }

    .groupdetailpage .itemslist .item {
        -ms-grid-columns: 110px 1fr;
        -ms-grid-rows: 1fr;
        display: -ms-grid;
        height: 110px;
        width: 470px;
    }

        .groupdetailpage .itemslist .item .item-info {
            -ms-grid-column: 2;
            margin-left: 10px;
        }

            .groupdetailpage .itemslist .item .item-info .item-title {
                margin-top: 4px;
                max-height: 20px;
                overflow: hidden;
            }

            .groupdetailpage .itemslist .item .item-info .item-subtitle {
                opacity: 0.6;
            }

            .groupdetailpage .itemslist .item .item-info .item-description {
                max-height: 60px;
                overflow: hidden;
            }

@media screen and (-ms-view-state: snapped) {
    .groupdetailpage section[role=main] {
        -ms-grid-row: 2;
        -ms-grid-row-span: 1;
    }

    .groupdetailpage .itemslist .win-vertical.win-viewport .win-surface {
        margin-bottom: 30px;
        margin-top: 0;
    }

    .groupdetailpage .itemslist .win-groupheader {
        visibility: hidden;
    }
```

```
    .groupdetailpage .itemslist .win-container {
        margin-left: 13px;
        margin-right: 35px;
    }

    .groupdetailpage .itemslist .item {
        -ms-grid-columns: 60px 1fr;
        height: 60px;
        width: 272px;
    }

        .groupdetailpage .itemslist .item .item-info .item-title {
            max-height: 30pt;
        }

        .groupdetailpage .itemslist .item .item-info .item-description {
            visibility: hidden;
        }
}

@media screen and (-ms-view-state: fullscreen-portrait) {
    .groupdetailpage .itemslist .win-horizontal.win-viewport .win-surface {
        margin-left: 100px;
    }
}
```

6. Create new file /pages/groupDetail.js
7. Paste the following code inside

```
(function () {
    "use strict";

    var appViewState = Windows.UI.ViewManagement.ApplicationViewState;
    var ui = WinJS.UI;

    ui.Pages.define("/pages/groupDetail/groupDetail.html", {
        /// <field type="WinJS.Binding.List" />
        _items: null,

        // This function is called whenever a user navigates to this page. It
        // populates the page elements with the app's data.
        ready: function (element, options) {
            var listView = element.querySelector(".itemslist").winControl;
            var group = (options && options.groupKey) ?
Data.resolveGroupReference(options.groupKey) : Data.groups.getAt(0);
            this._items = Data.getItemsFromGroup(group);
            var pageList = this._items.createGrouped(
                function groupKeySelector(item) { return group.key; },
                function groupDataSelector(item) { return group; }
            );

            element.querySelector("header[role=banner] .pagetitle").textContent =
group.title;

            listView.itemDataSource = pageList.dataSource;
```

```
            listView.itemTemplate = element.querySelector(".itemtemplate");
            listView.groupDataSource = pageList.groups.dataSource;
            listView.groupHeaderTemplate = element.querySelector(".headertemplate");
            listView.oniteminvoked = this._itemInvoked.bind(this);

            this._initializeLayout(listView,
Windows.UI.ViewManagement.ApplicationView.value);
            listView.element.focus();
        },

        unload: function () {
            this._items.dispose();
        },

        // This function updates the page layout in response to viewState changes.
        updateLayout: function (element, viewState, lastViewState) {
            /// <param name="element" domElement="true" />

            var listView = element.querySelector(".itemslist").winControl;
            if (lastViewState !== viewState) {
                if (lastViewState === appViewState.snapped || viewState ===
appViewState.snapped) {
                    var handler = function (e) {
                        listView.removeEventListener("contentanimating", handler, false);
                        e.preventDefault();
                    }
                    listView.addEventListener("contentanimating", handler, false);
                    var firstVisible = listView.indexOfFirstVisible;
                    this._initializeLayout(listView, viewState);
                    if (firstVisible >= 0 && listView.itemDataSource.list.length > 0) {
                        listView.indexOfFirstVisible = firstVisible;
                    }
                }
            }
        },

        // This function updates the ListView with new layouts
        _initializeLayout: function (listView, viewState) {
            /// <param name="listView" value="WinJS.UI.ListView.prototype" />

            if (viewState === appViewState.snapped) {
                listView.layout = new ui.ListLayout();
            } else {
                listView.layout = new ui.GridLayout({ groupHeaderPosition: "left" });
            }
        },

        _itemInvoked: function (args) {
            var item = this._items.getAt(args.detail.itemIndex);
            WinJS.Navigation.navigate("/pages/itemDetail/itemDetail.html", { item:
Data.getItemReference(item) });
        }
    });
})();
```

Implementing the Share feature

1. Open /pages/itemDetail/itemDetail.js
2. Implement a new method/function inside the page:

```
datarequest: function (e) {
            var request = e.request;
            request.data.properties.title = Data.currentItem.title;
            request.data.properties.description = Data.currentItem.subtitle;
            request.data.setUri(new Windows.Foundation.Uri(Data.currentItem.url));
        }
```

3. Place the following code inside the **unload** method:

```
var dataTransferManager =
Windows.ApplicationModel.DataTransfer.DataTransferManager.getForCurrentView();
            dataTransferManager.removeEventListener("datarequested",
this.datarequest);
```

Run the application and test the Share function

Implementing the Settings pane

1. Open js/default.js
2. Place the following code above the app. onactivated handler

```
app.onsettings = function (e) {
        e.detail.applicationcommands = {
            "about": {
                href: "/pages/about/about.html",
                title: "About"
            },
            "settings": {
                href: "/pages/Settings/Settings.html",
                title: "Settings"
            }
        }
        WinJS.UI.SettingsFlyout.populateSettings(e);
    }
```

3. Create new folder /pages/about/
4. Create new file /pages/about/about.html
5. Place the following code inside:

```
<div id="settingsContainer" data-win-control="WinJS.UI.SettingsFlyout" aria-
label="About the application" data-win-
options="{settingsCommandId:'about',width:'narrow'}">

    <div class="win-ui-dark win-header">
        <button type="button" onclick="WinJS.UI.SettingsFlyout.show()" class="win-
backbutton"></button>
        <div class="win-label">About</div>
    </div>
    <div class="win-content">
```

```
        <div class="win-settings-section">
            <form>
                <h1>VideoLib  - demo app</h1>
            </form>
        </div>
    </div>
</div>
```

6. Create new folder /pages/settings
7. Create new file /pages/settings/settings.html

```
<script src="settings.js"></script>

<div id="settingsContainer" data-win-control="WinJS.UI.SettingsFlyout" aria-
label="About the application" data-win-
options="{settingsCommandId:'settings',width:'narrow'}">

    <div class="win-ui-dark win-header">
        <button type="button" onclick="WinJS.UI.SettingsFlyout.show()" class="win-
backbutton"></button>
        <div class="win-label">Settings</div>
    </div>
    <div class="win-content">
        <div class="win-settings-section">
            <form>
                Show descriptions:<input type="checkbox" id="chbShowDescriptions" />
            </form>
        </div>
    </div>
</div>
```

8. Create new file /pages/settings/settings.js with the following code inside:

```
(function () {
    "use strict";

    var self = null;



    WinJS.UI.Pages.define("/pages/Settings/Settings.html", {


        ready: function (element, options) {
            self = this;
            self.getSettings();

document.getElementById("settingsContainer").winControl.addEventListener("afterhide",
this.unload);

        },

        getSettings: function () {

           document.getElementById("chbShowDescriptions").checked =
Windows.Storage.ApplicationData.current.roamingSettings.values["showdescr"] == true;
```

```
        },

        unload: function () { // Doesn't appear to be called
            self.updateSettings();
        },

        updateSettings: function () {

Windows.Storage.ApplicationData.current.roamingSettings.values["showdescr"] =
document.getElementById("chbShowDescriptions").checked
         },

    });
}())
```