Designer Touch Demo Script

Steps
[Commentary]
(Bugs/TODO)
Can be skipped over in demo for time constraints
{Save Point} https://github.com/lojjic/Sencha-Expenses-Touch

- Create new touch project
  - [Brief overview of screen, toolbox, canvas controls, etc.]

- Create Login screen
  - Add FormPanel at top level
  - [Explain Project Inspector, Film, Component Config panel]
  - Change userClassName to LoginForm
  - Add a FieldSet - [explain advantage of using FieldSet]
  - Remove the FieldSet's title config in the Config panel
  - First Field
    - Transform first field into Email Field
    - [explain placeHolder]
    - Change first field's label to "Username"
    - Change name config to "username"
  - Second Field
    - Transform the field into a Password Field
    - Change label to "Password"
    - Change name config to "password"
  - Adjust labels of both fields to 35%
  - Add Button to LoginForm
    - Change button text to "Log In"
    - [Explain "UI"]
    - Open button's flyout and change UI Style to "confirm"
  - Align form contents:
    - [Brief explanation of layouts]
    - Open LoginForm's flyout, select vbox layout and pack:center
  - [Verify display in different orientations and device sizes]
  - [Switch to Code view, walk through what got generated]
  - [Save project]

{step1}
- Create Expense List
  - [Explain what region of layout this will encompass: the list and the Add Expenses button at the bottom]
  - Add a Panel at top level
  - Change userClassName to ListPanel
  - Add a List to the Panel
  - [Oops, list is invisible! Show it is zero height by selecting it in the inspector tree, explain that best way to give it dimensions is to set fit layout on parent]

    - Change layout of ListPanel to 'fit'
    - [Explain the default dummy store]
    - Open flyout, click Edit Item Template button, change value to `<div class="merchant">{merchant}</div><div class="category">{category}</div><div class="amount">{amount}</div>`, click Done
    - Set the itemId of the List to 'expenseList'
    - Add toolbar to ListPanel
    - [Explain docking]
    - Set toolbar to docked:bottom
    - Add button to toolbar
        - Change button text to "Add New Expense"
        - Change button UI Style to "action-round"
        - Change button flex to 1 [Explain flex]
    - [Verify display in different orientations and device sizes]

{step2}

- Create the Add New Expense form
    - Add a FormPanel to the top level
    - Change userClassName to AddExpenseForm
    - Add FieldSet
    - Change FieldSet title to "Add New Expense"
    - First Field
        - Change label to "Merchant"
        - Change name config to 'merchant'
        - Change required to true
    - Second field
        - Transform to a Number Field
        - Change label to "Amount"
        - Change name config to 'amount'
        - Change required to true
    - Third field
        - Drag a Select Field into the fieldset
        - Change label to "Category"
        - Change name config to 'category'
        - [Explain 'options' config, also mention store config]
        - Change options config to `[{text:"Uncategorized",value:null},{text:"Lodging",value:"lodging"},{text:"Transportation",value:"transportation"},{text:"Food/Drink",value:"food"}]`
    - Change width of labels of all 3 fields to 35%
    - Add two buttons to the form
    - First button
        - Change text to "Submit"
        - Change UI Style to "confirm"
    - Second button
        - Change text to "Cancel"
        - Change UI Style to "Decline"
        - [Point out space between buttons; explain "style" config]
        - Change style config to "margin-top: 1em;"
    - Change AddExpenseForm layout to vbox, pack to center

- [Verify display in different orientations and device sizes]

{step3}

- Main Panel
  - [Explain card layout]
  - [Explain that this panel combines the ListPanel and AddExpenseForm into a card view, and buttons will eventually have events attached to switch between them]
  - Add a Panel to the top level
  - Change userClassName to MainPanel
  - Add a Toolbar
    - Add a Component to the toolbar
      - set itemId to loggedInUserName
      - Set html config to "[email@example.com]"
      - Set style config to "color:#FFF; padding:0 .5em; font-size: 0.8em;"
    - Add a Button to the toolbar
      - Change text to "Log Out"
    - Change loggedInUserName component's flex to 1
  - [Explain instance linking]
  - Change MainPanel's layout to card
  - Drag ListPanel onto MainPanel, choose "Link"
    - set its itemId to “listPanel”
  - Drag AddExpenseForm onto MainPanel, choose "Link"
    - set its itemId to “addExpenseForm”
  - [Point out line connecting linked instances to their classes]
  - [Point out selecting each card switches the view to that card]
  - Change MainPanel's card switch animation to slide
  - [Point out card switching now has animation]

{step4}

- Viewport
  - Add a Panel to the top level
  - Change userClassName to Viewport [Explain autoCreateViewport naming convention, and that this will be simplified in the future]
  - Set fullscreen to true
  - Set Application’s autoCreateViewport to true
  - Set userAlias to “topview” [will be used later on for ComponentQuery] (itemId would be better but SDK doesn’t allow it as a class config default)
  - Add a toolbar
    - Change toolbar title to "Sencha Expenses"
  - Change layout to card
  - Change card switch animation to "fade"
  - Drag LoginForm onto Viewport, choose "Link"
    - set its itemId to loginForm
  - Drag MainPanel onto Viewport, choose "Link"
    - set its itemId to mainPanel
  - [Point out card switching does fade transition]
  - [Point out that on the MainPanel card its toolbar looks odd next to the top toolbar]
  - Double-click on the MainPanel linked instance to go to its definition, select the toolbar and change its UI Style to 'light'

{step5}

- Events
  - Select Log In button
    - [Point out Basic Event Binding object in toolbox, and Events tab]
    - Add a 'tap' event binding to the button
    - Change its fn to ‘onLoginTap’
    - Select tap binding and go to code view
    - [Describe code editor]
    - [Explain ComponentQuery and itemId]
    - Add code to button tap handler: [normally this would have validation logic, call out to an auth service, handle errors, etc.]

```
var top = button.up('topview');
MyApp.username =
button.up('formpanel').getValues().username;
top.down('#loggedInUserName').setHtml(MyApp.username);
top.setActiveItem(1);
```

  - Select Log Out button
    - Add a ‘tap’ event binding to the button
    - Change its fn to ‘onLogoutTap’
    - Select tap binding and go to code view
    - Add code:

```
var top = button.up('topview');
top.down('#loginForm').reset();
top.setActiveItem(0);
```

  - Select Add New Expense button in the list panel toolbar
    - Add a ‘tap’ event binding to the button
    - Change its fn to ‘onSwitchToAddTap’
    - Select tap binding and go to code view
    - Add code:

```
button.up('#mainPanel').setActiveItem(1);
```

  - Select Cancel button in the add expense form
    - Add a ‘tap’ event binding to the button
    - Change its fn to ‘onCancelTap’
    - Add code:

```
button.up('formpanel').reset();
button.up('#mainPanel').setActiveItem(0);
```

  - Select the Submit button in the add expense form
    - Add a ‘tap’ event binding to the button
    - Change its fn to ‘onSubmitTap’
    - Add code:

```
var form = button.up('formpanel'),
   mainPanel = form.up('#mainPanel'),
   store = mainPanel.down('#expenseList').getStore(),
   ts = new Date(),
   expenseRecord = form.getValues();

// Add additional data
expenseRecord.username = MyApp.username;
expenseRecord.timestamp = ts;
```

```
// Send message to the queue - no relation to the store
Ext.io.send('queue/expense', {
    'type': 'newexpense',
   'storeid': 'expenseDB',
   'username': expenseRecord.username,
   'merchant': expenseRecord.merchant,
   'category': expenseRecord.category,
   'amount': expenseRecord.amount,
   'timestamp': Ext.Date.format(ts, 'Y-m-d g.i')
});

//Store to local storage and sync to remote syncstorage
store.add(expenseRecord);
store.sync();

//Confirmation
Ext.Msg.alert(null, "Expense Added Successfully",
function() {
   //Back to list
   form.reset();
   mainPanel.setActiveItem(0);
});
```

- Select the List component in the ListPanel
  - Add a ‘painted’ event binding to the List
  - Change its fn to ‘onListPainted’
  - Add code:

```
// Create Model
Ext.define('Expense', {
  extend: "Ext.data.Model",
  fields: [
  {name: 'username', type: 'string'},
  {name: 'merchant', type: 'string'},
  {name: 'category', type: 'string'},
  {name: 'amount', type: 'int'},
  {name: 'timestamp',type: 'date'}
  ],
  proxy: {
      id: 'expenseDB', //Uses new user/timestamp
      type: 'syncstorage',
      key: 'expense'
  }
});

// Create store and assign to List
var store = Ext.create('Ext.data.Store', {
  model: 'Expense',
  sorters: [ {
      property: 'timestamp',
      direction: 'DESC'
  } ],
```

```
    autoLoad: true
});
component.setStore(store);

// Sync IO Setup
Ext.io.setup({key: "expense"});
Ext.io.init();

// Sync data from localstorage
store.sync();
```

{step6}

- Finalizing
  - Save project
  - Copy ext-overrides.js, sencha-io-debug, and custom.css into the project files dir
  - In project files dir, copy designer.html to index.html
  - Add the following to index.html (after ST library imports, before designer.js):
    - `<script type="text/javascript" src="ext-overrides.js"></script>`
    - `<script type="text/javascript" src="sencha-io-debug.js"></script>`
    - `<link rel="stylesheet" type="text/css" href="custom.css"/>`

{step7/final}