Apply Angular Techniques to jQuery Applications – Part 1

There are many frameworks such as Angular and React that help you develop client-side applications in a very structured manner. They help you create Single Page Applications (SPA), perform data binding, and provide many other services. The purpose of this blog post is not to replace these frameworks, but to show you how you might accomplish the same services using JavaScript and jQuery. There are many developers that have a huge investment in JavaScript and jQuery and can't, or maybe don't want to, convert to these new frameworks. They may also want to create a SPA and have a nice structured approach to their jQuery applications. In this blog post I present a method to create a SPA using HTML, Bootstrap JavaScript and jQuery. In addition, I present a way to structure your application files to keep them small and organized.

In this post you will learn the basics of downloading partial HTML pages and insert them into another HTML page at a specified location. This is like the SPA functionality that Angular and React supply. You are going to create small files for style sheets and JavaScript for each page. All of this is a good start on building a SPA using just jQuery, and a model for structuring your different application artifacts.

In the next blog post, I will show you how to build a CRUD page using these same techniques but adding on Web API.

Build a Starting Page

Create a new web project using the tool of your choice.

```
npm init
npm install bootstrap@3 jquery --save
```

Add index.html

Add an index.html in the root folder.

```
<!doctype html>
<html>
  <head>
    <meta http-equiv="Refresh" content="0;
url=http://localhost:3000/src/index.html" />
  </head>
  <body>
    <p>Please follow <a
href="http://localhost:3000/src/index.html">this link</a>.</p>
  </body>
</html>
```

Add src\index.html file

Add a new folder named **src**.

Add an HTML page named **index.html** into the **src** folder.

```
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

In your index.html page, modify the <head> section to look like the following.

```
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport"
        content="width=device-width, initial-scale=1">

  <title>jQuery SPA Sample</title>

  <link href="../node_modules/bootstrap
              /dist/css/bootstrap.min.css"
        rel="stylesheet" />
</head>
```

Just after the <body> tag, add Bootstrap navigation and menu items with their *href* attributes filled in with hashtag values as shown below.

```
<nav class="navbar navbar-fixed navbar-inverse"
     role="navigation">
  <div class="container">
    <div class="navbar-header">
      <a href="#home" class="navbar-brand">
        Home
      </a>
    </div>
    <ul class="nav navbar-nav">
      <li>
        <a href="#about">About</a>
      </li>
      <li>
        <a href="#contact">Contact</a>
      </li>
    </ul>
  </div>
</nav>
```

Just below the navigation area, add a new and use the Bootstrap class of “container”. It is within this area that we want our partial pages to be displayed. Add a new element named called “contentarea”. This is an element name you are just making up.

```
<div class="container">
  <contentarea></contentarea>
</div>
```

Below this code, add <script> tags for jQuery and Bootstrap.

```
<script
src="../node_modules/jquery/dist/jquery.min.js"></script>
<script
src="../node_modules/bootstrap/dist/js/bootstrap.min.js"></scr
ipt>
```

Add a new folder named **scripts**.

Try it Out

If you are using VS Code, install a simple HTTP server, otherwise if you try to run it locally, you get a CORS error.

```
npm install lite-server --save-dev
```

Open package.json and add within the "scripts" property:

```
"scripts": {
  "dev": "lite-server"
},
```

Start the server by opening a terminal window and typing in the following command.

```
npm run dev
```

Create SPA Common JavaScript File

Add a file named **spa-common.js** into this folder.

Add the following code into this file.

```
'use strict';

$(document).ready(function () {
  // Trigger home page
  window.location.replace(window.location.href + "#home");

  // Respond to changes in hash
  window.onhashchange = function () {
    if (window.location.hash) {
      // Remove # to create the page file name
      let pageName = window.location.hash.substr(1);

      // Use jQuery to retrieve partial page
      $("contentarea").load(pageName + ".html",
        function(response, status, xhr) {
        if(status == "error") {
          console.error("Can't retrieve partial page: '"
            + pageName + ".html' - " + JSON.stringify(xhr));
        }
      });
    }
  }
});
```

Add a reference to this file from the index.html page

```
<script src="../scripts/spa-common.js"></script>
```

Home Page

Create a new page called **home.html** within the \src folder. This is going to be one of the partial pages that will be loaded and placed in between the <contentarea> and the </contentarea> tags.

```
<div class="row">
  <div class="col-xs-12">
    <h1>Home Page</h1>
  </div>
</div>
```

About Page

Create a new page called **about.html** within the \src folder that will also be loaded into the content area.

```
<div class="row">
  <div class="col-xs-12">
    <h1>About Page</h1>
  </div>
</div>
```

Contact Page

Create a new page called **contact.html** within the \src folder that will also be loaded into the content area.

```
<div class="row">
  <div class="col-xs-12">
    <h1>Contact Us Page</h1>
  </div>
</div>
```

Try it Out

Go back to the browser.

Click on the About and Contact menu items to see them displayed. If they are not displayed, check the console window.

Create Functions

Instead of having one large $(document).ready() function, let's separate the functionality into some different functions. This will better prepare us for creating a closure later.

The loadPage Function

The process of loading a partial HTML page and displaying it within a specific area on a page can be made very generic. Let’s break this part of the previous code into its own separate function named loadPage().

```
function loadPage(contentArea, pageName) {
  // Use jQuery to retrieve partial page
  $(contentArea).load(pageName + ".html",
    function (response, status, xhr) {
      if (status == "error") {
        console.error("Can't retrieve partial page: '"
          + pageName + ".html' - " + JSON.stringify(xhr));
      }
    }
  );
}
```

The changePage Function

The process of taking the hash name, stripping out the leading hash symbol, and calling the loadPage() function can be made into another function called changePage().

```
function changePage(contentArea, hashValue) {
  // Remove # to create the page file name
  let pageName = hashValue.substr(1);
  // Load the partial HTML page
  loadPage(contentArea, pageName);
}
```

Rewrite Document Ready Function

From the onhashchange event you now only need to call the changePage() function. Below is the complete document.ready() function you replace with the code that is currently in your page.

```
$(document).ready(function () {
  // Trigger home page
  window.location.replace(window.location.href + "#home");

  // Respond to changes in hash
  window.onhashchange = function () {
    changePage("contentarea", window.location.hash);
  }
});
```

Try it Out

Run the page again and ensure everything works as expected.

Fix Bug

There is a bug in our code where if you have a #home, or any other hash in the address bar and you hit the Enter key on the address bar, you get two hash tags in the address bar.

Modify the $(document).ready() to be as follows:

```
$(document).ready(function () {
  let pageName = window.location.href;

  // Trigger home page if there is not already a hash sign
  if(pageName.indexOf("#") == -1) {
    window.location.replace(pageName + "#home");
  }
  else {
    changePage("contentarea",
pageName.substr(pageName.indexOf("#")));
  }

  // Respond to changes in hash
  window.onhashchange = function () {
    changePage("contentarea", window.location.hash);
  }
});
```

Add Custom CSS and JS to Partial Page

Having custom style sheets and JavaScript files particular to one page is always a best practice. Even though you are using partial pages, there is no reason you can't add styles and JavaScript to those pages. Add a new style sheet named **about.css** within the \src folder and add the following code into this new file.

```
.about-h1 {
  color: blue;
}
```

Now add a JavaScript file named **about.js** within the \src folder and add the following code into this file.

```
'use strict';

function aboutHello() {
  alert("Hello from About Page");
}
```

Open the **about.html** and modify the HTML to match what is shown below.

```
<link href="about.css" rel="stylesheet" />

<div class="row">
  <div class="col-xs-12">
    <h1 class="about-h1">About Page</h1>
  </div>
</div>

<div class="row">
  <div class="col-xs-12">
    <button onclick="aboutHello()">Say Hi</button>
  </div>
</div>

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

Try it Out

Run your page, click on the About menu and you should now see the title is in a different color. Click on the button and you should see an alert pop-up on your browser.

Create a Closure

Wrap up the functions you created into a closure. Open the **spa-common.js** file in your \scripts folder and modify the code to look like the following.

```
'use strict';

$(document).ready(function () {
  let pageName = window.location.href;

  // Trigger home page if there is not already a hash sign
  if(pageName.indexOf("#") == -1) {
    window.location.replace(pageName + "#home");
  }
  else {
    spaController.changePage("contentarea",
pageName.substr(pageName.indexOf("#")));
  }

  // Respond to changes in hash
  window.onhashchange = function () {
    spaController.changePage("contentarea",
window.location.hash);
  }
});

/**
 * SPA Component
 */
let spaController = (function () {
  /**
   * Private functions
   */
  function _loadPage(contentArea, pageName) {
    // Use jQuery to retrieve partial page
    $(contentArea).load(pageName + ".html",
      function (response, status, xhr) {
        if (status == "error") {
          console.error("Can't retrieve partial page: '"
            + pageName + ".html' - " + JSON.stringify(xhr));
        }
      }
    );
  }

  function _changePage(contentArea, hashValue) {
    // Remove # to create the page file name
    let pageName = hashValue.substr(1);
    // Load the partial HTML page
    _loadPage(contentArea, pageName);
  }

  /**
   * Public functions
   */
  return {
    "loadPage": _loadPage,
    "changePage": _changePage
  };
})();
```

The above code creates a closure and assigns that to a variable named **spaController**. You may now access the loadPage() and changePage() functions through this variable.

Try it Out

Run the page and ensure everything is still working like it was before.

Add Title for Each Page

As you click on each menu, it would be good if the title of each partial page updated the tab in your browser. Let's use the 'title' attribute to specify the name of each page. Modify each anchor tag on your page to add a 'title' attribute as shown below.

```
<a href="#home" title="jQuery SPA Sample" class="navbar-
brand">
  Home
</a>

<a href="#about" title="About Us">About</a>

<a href="#contact" title="Contact Us">Contact</a>
```

Add code to the changePage() function to set the document.title property with the value from the 'title' attribute just clicked upon.

```
function _changePage(contentArea, hashValue) {
  // Remove # to create the page file name
  let pageName = hashValue.substr(1);

  // Set document title
  window.document.title = $("a[href='" + hashValue +
"']").attr("title");

  // Load the partial HTML page
  _loadPage(contentArea, pageName);
}
```

Try it Out

Run the application and click on each menu item and watch the title change in the tab of your browser.

Move Pages to Different Folder

Most likely, you will want to separate many of your pages into separate folders.

Create a **\common** folder under the \src folder and move the home.html, about.html and the contact.html pages into the new common folder.

After moving them, you need to somehow specify the path to find these pages. You don't want to modify the *href* attribute, so add a new **'data-page-path'** attribute to specify the path as shown in the following code.

```
<a href="#home"
   title="Home"
   data-page-path="./common/"
   class="navbar-brand">
  Home
</a>
<a href="#about"
   title="About"
   data-page-path="./common/">
  About
</a>
<a href="#contact"
   title="Contact Us"
   data-page-path="./common/">
  Contact
</a>
```

Retrieve this path from the 'data-page-path' attribute using jQuery. Not every anchor tag has a data-page-path attribute so set the path to an empty string if the attribute is not found. Concatenate the path and the pageName together when passing the file to the loadPage() function.

```
function _changePage(contentArea, hashValue) {
  // Get path for partial page
  let path = $("a[href='" + hashValue + "']").data("page-
path") || "";

  // Remove # to create the page file name
  let pageName = hashValue.substr(1);

  // Set document title
  window.document.title = $("a[href='" + hashValue +
"']").attr("title");

  // Load the partial HTML page
  _loadPage(contentArea, path + pageName);
}
```

One note here; if you reference .CSS or .JS files from your partial pages within the \common folder, you will need to fully qualify those references. For instance, in the **about.html** page, add the **"./common/"** in front of the each file name as shown below.

```
<link href="./common/about.css" rel="stylesheet" />

<script src="./common/about.js"></script>
```

Try it Out

Run the pages to ensure everything still works as expected.

Reset Menu Focus

Run your index.html page and again click on the About menu, then click on the Contact menu. Now click the back button. Pay attention to where the focus on the menu bar is located. It is still on the Contact menu. Click the back button again so you are back at the home page. The menu focus is still on the Contact menu. This is obviously something we need to fix.

Add a function named _resetMenu() in the spa-controller.js file. You will pass the current hash value like #home or #about to this function. You use this value to locate the anchor tag contained in the Bootstrap menu system. Once located, you set focus to that menu. Unfortunately, this also draws a dotted outline around that menu item. You can remove that outline by setting the outline style to 0. Create the following function on your index.html page.

```
function _resetMenu(hashValue) {
  // Set focus to menu that matches current page
  $("a[href='" + hashValue + "']").focus();
  // Remove outline around anchor when setting focus
  $("a[href='" + hashValue + "']").css("outline", "0");
}
```

Call this new _resetMenu() function from the bottom of the _changePage() function.

```
function _changePage(contentArea, hashValue) {
  // REST OF THE CODE HERE
  // ...

  // Reset menu focus
  _resetMenu(hashValue);
}
```

Try it Out

Run the index.html page and ensure that everything still works, and that the back button now resets the menu focus to the current partial page in your spa.

NOTE:	Be aware that some browsers may be caching old versions of pages. Be sure to use your browsers "clear cache" functionality periodically. For example, on Chrome, open the F12 tools and press Ctrl-F5 to clear the cache.

Set Specific Start Page

In the document.ready() function you see the "#home" and the "contentarea" are hard-coded references to the partial page to start with and the element name on index page where all partial pages should be loaded. We can get rid of the "#home" page and place that into a "data-" attribute on the index.html page. Open the index.html page and modify the <contentarea> tag to include a ‘data-page-start’ attribute. Set that attribute to the first partial page you want to display when this page loads.

```
<contentarea data-page-start="#home"></contentarea>
```

Open the spa-common.js file and make the document.ready() function look like the following:

```
$(document).ready(function () {
  let pageName = window.location.href;
  let contentElement = "contentarea";
  let start = $(contentElement).data('page-start');

  // Trigger home page if there is not already a hash sign
  if (pageName.indexOf("#") == -1) {
    window.location.replace(pageName + start);
  }
  else {
    spaController.changePage(contentElement,
pageName.substr(pageName.indexOf("#")));
  }

  // Respond to changes in hash
  window.onhashchange = function () {
    spaController.changePage(contentElement,
window.location.hash);
  }
});
```

Try it Out

Run the web site and ensure everything still runs as expected.

Find Content Area Element

While you still need to have "contentarea" hard-coded as the element name, you can override this name if you want. All you need to do is check to see if the 'data-page-start' attribute exists on an element in your page. If it does, then you know the element that contains this attribute is the element into which all your partial pages should be rendered. Open the spa-common.js file and add a new method named _init(). You can then copy the code within the document.ready() function into this new method and add some additional code as shown below.

```
function _init() {
  let pageName = window.location.href;
  let contentElement = "contentarea";
  let start = $(contentElement).data('page-start');

  // Get content area by reading nodeName
  contentElement = $("[data-page-start]") ? $("[data-page-
start]")[0].nodeName : contentElement;

  // Trigger home page if there is not already a hash sign
  if (pageName.indexOf("#") == -1) {
    window.location.replace(pageName + start);
  }
  else {
    spaController.changePage(contentElement,
pageName.substr(pageName.indexOf("#")));
  }

  // Respond to changes in hash
  window.onhashchange = function () {
    spaController.changePage(contentElement,
window.location.hash);
  }
}
```

Expose this method from the closure.

```
return {
  "loadPage": _loadPage,
  "changePage": _changePage,
  "init": _init
};
```

Modify the document.ready() function to call the init() method in the closure.

```
$(document).ready(function () {
  spaController.init();
});
```

The first line locates the element with the 'data-spa-start' attribute and retrieves the nodeName property. The nodeName property contains the name of the element, which in our case is “contentarea”.

The second line then locates that content area and retrieves the value in the data-spa-start attribute. That value, in our case, is “#home”. This value is then passed to the replace() method. Finally, in the call to the _changePage() method, you pass the value in the **content** variable.

Try it Out

Run the page and ensure your application still works as expected.

Try removing the data-page-start attribute and make sure the page still loads. You won't see the home page displayed when the index.html page loads since you did not set the #home partial page to load. If you click on the various menus, all the partial pages should load as expected.

Summary

In this blog post you learned how to create your own Single Page Application (SPA) structure using just jQuery, Bootstrap and HTML. With very little code you can create a nice separation of your HTML, CSS and JavaScript. Sure, they are frameworks that do all this for you, but sometimes those frameworks come with a price. Sometimes keeping it simple is a very good way to code. In my next blog post I show you how to create a CRUD page using the techniques presented here.

You can get the samples at www.pdsa.com/downloads. Choose “PDSA Blogs” from the Category, then select “Apply Angular Techniques to jQuery Applications – Part 1”.