

Report

by Daniel Bennett

FILE	14023_DANIEL_BENNETT_REPORT_228704_280318639.DOCX (2.36M)		
TIME SUBMITTED	22-MAY-2021 12:35AM (UTC+0100)	WORD COUNT	780
SUBMISSION ID	154388843	CHARACTER COUNT	3609



ASSIGNMENT 2: MOBILE- ENABLED WEBSITE UTILISING REST SERVICES (JSON)



Daniel Bennett
2021 149266

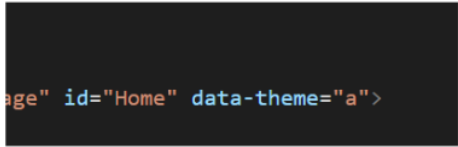
Contents

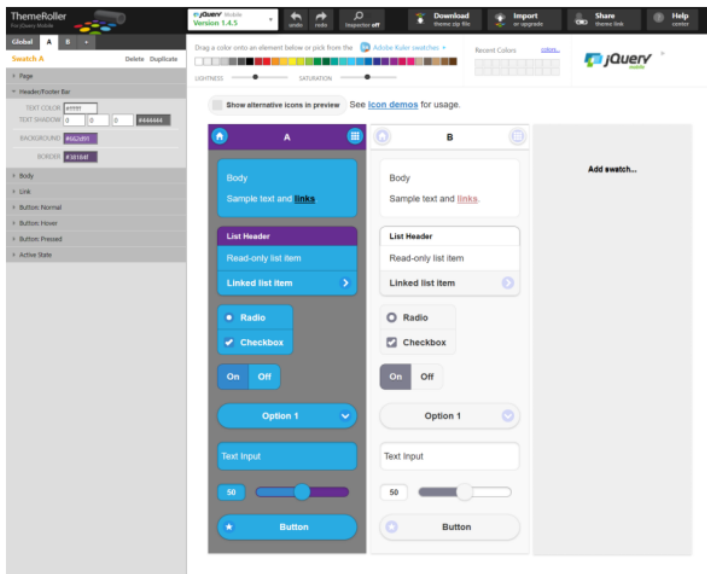
Introduction	2
Style	2
Single Page Application	3
Panels	4
Public AJAX REST API OMBD and Local AJAX JSON Request	6
Public API OMDB	6
Local AJAX JSON Request	7
Conclusion	10

In this document, I will discuss the implementation of jQuery mobile to develop a website, and the use of AJAX requests to get information from external sources.

Style

Below is a screenshot of some of my CSS after using theme roller and my implementation of it inside of my HTML.





Single Page Application

Using jQuery mobile, I was able to implement a three-page website but by only using 1 html page. This was incredibly easy and allowed for shared resources to be used.

Below is an image of the pages being used:

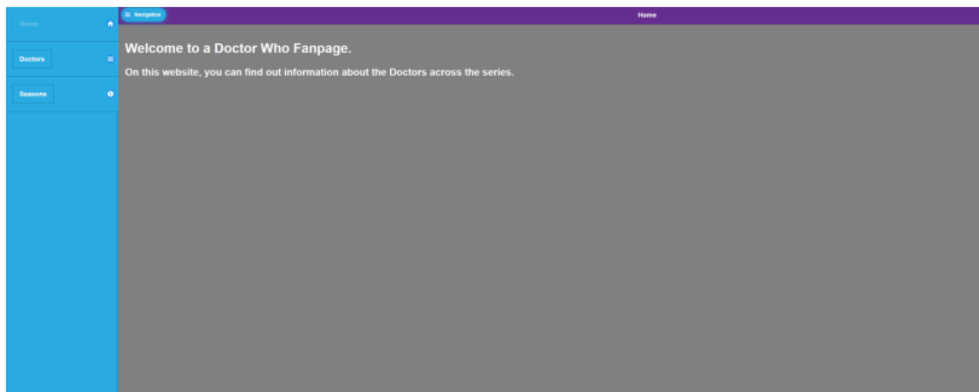
```
<body>
  <div data-role="page" id="Home" data-theme="a">
    <div data-role="header" data-theme="a">
      <a href="#mysidepanel" class="ui-btn ui-shadow ui-corner-all ui-btn-inline ui-btn-icon-left ui-icon-bars" data-ajax="false">Navigation</a>
      <h1>Home</h1>
    </div>
    <!-- /header -->

    <!-- Side Panel -->
    <div data-role="panel" id="mysidepanel" data-position="left" data-display="reveal" data-theme="a">
      <div class="ui-panel-inner">
        <ul data-role="listview">
          <li data-icon="home">
            <a href="#Home"><button class="ui-btn ui-btn-inline" disabled="disabled">Home</button></a>
          </li>
          <li data-icon="bars">
            <a href="#Doctors"><button class="ui-btn ui-btn-inline">Doctors</button></a>
          </li>
          <li data-icon="info">
            <a href="#Seasons"><button class="ui-btn ui-btn-inline">Seasons</button></a>
          </li>
        </ul>
      </div>
    </div>
    <!-- /mysidepanel -->

    <div role="main" class="ui-content" data-theme="a">
      <h1>Welcome to a Doctor Who Fanpage.</h1>
      <h2>On this website, you can find out information about the Doctors across the series.</h2>
    </div>
    <!-- /content -->

    <div data-role="footer" data-position="fixed" data-theme="b">
      <h2>Thanks for using the website!</h2>
    </div>
    <!-- End of footer -->
  </div>
  <!-- /page -->

  <!-- Start of second page -->
  <div data-role="page" id="Doctors">
    <div data-role="header">
      <a href="#mysidepanel" class="ui-btn ui-shadow ui-corner-all ui-btn-inline ui-btn-icon-left ui-icon-bars" data-ajax="false">Navigation</a>
```



You can define a page by using the data-role tag which is introduced with jQuery mobile.

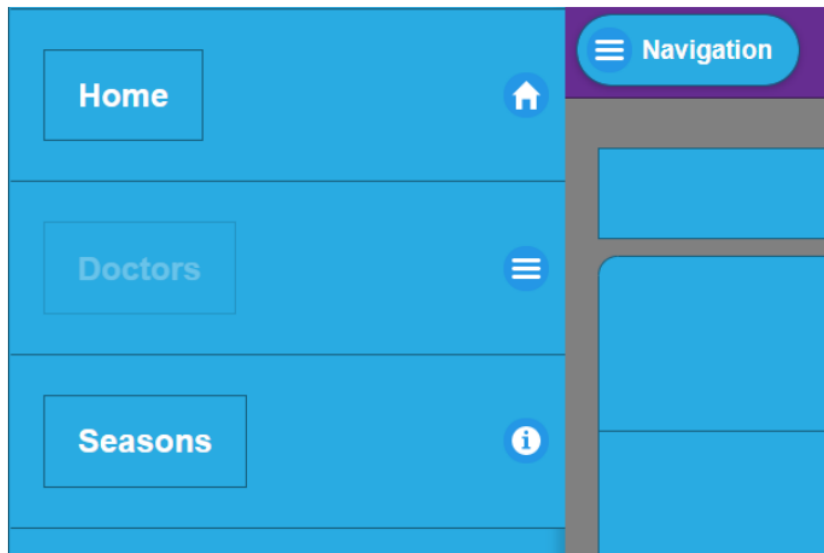
Panels

```
<!--Side Panel-->
<div data-role="panel" id="mysidepanel" data-position="left" data-display="reveal" data-theme="a">
  <div class="ui-panel-inner">
    <ul data-role="listview">
      <li data-icon="home">
        <a href="#Home"><button class="ui-btn ui-btn-inline" disabled="disabled">Home</button></a>
      </li>
      <li data-icon="bars">
        <a href="#Doctors"><button class="ui-btn ui-btn-inline">Doctors</button></a>
      </li>
      <li data-icon="info">
        <a href="#Seasons"><button class="ui-btn ui-btn-inline">Seasons</button></a>
      </li>
    </ul>
  </div>
</div>
</div>
```

Panels were used to create the side panel, which was used for navigation on the website, this allowed for a hidden navbar, which freed up more space for content on the page.

The panel also called a script at the top of the webpage to make sure it is positioned in the correct place.

```
<!--Script to operate my side panel-->
<script>
  jQuery(function() {
    $('#mysidepanel').enhanceWithin().panel();
  });
</script>
```



Collapsibles

By using collapsibles you are able to easily hide content from the user meaning that you can have more relevant content on the page at a time rather than having it all on the page at once.

These items were chosen to be stored into collapsibles on this page to stimulate the interest of the user. Due to the satisfying nature of clicking on something and then something else happening engages the user, and then the phrase “Unlock Now” makes the user want to press the “Load” button. This helps increase engagement with the website and may lead to the user exploring the website more.

The development of these objects was very easy as all you have to do is place collapsible items inside a div with the data role of “collapsibleset”. This also has the added benefit of grouping the collapsibles into one item which only allows for one collapsible to be open at a time.

```

</div>
<!-- /mysidepanel -->

<div onload="test()" ole="main" class="ui-content">

  <button id="loadBtn" class="ui-btn">Load</button>

  <div data-role="collapsible" data-theme="a" data-content-theme="a" data-iconpos="right">

    <div id="9" data-role="collapsible" data-iconpos="bottom">
      <h3>The Ninth Doctor</h3>
      <p>Unlock Now</p>
    </div>

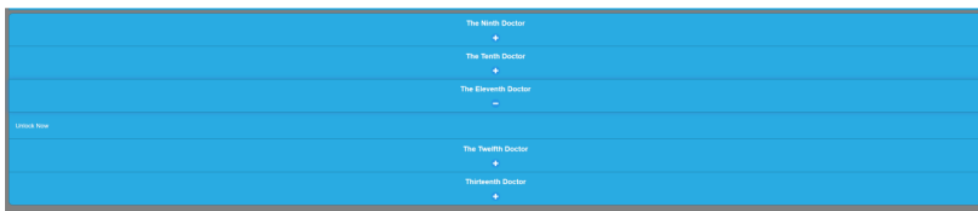
    <div id="10" data-role="collapsible" data-iconpos="bottom">
      <h3>The Tenth Doctor</h3>
      <p>Unlock Now</p>
    </div>

    <div id="11" data-role="collapsible" data-iconpos="bottom">
      <h3>The Eleventh Doctor</h3>
      <p>Unlock Now</p>
    </div>

    <div id="12" data-role="collapsible" data-iconpos="bottom">
      <h3>The Twelfth Doctor</h3>
      <p>Unlock Now </p>
    </div>

    <div id="13" data-role="collapsible" data-iconpos="bottom">
      <h3>Thirteenth Doctor</h3>
      <p>Unlock Now</p>
    </div>
  </div>

```



Public AJAX REST API OMBD and Local AJAX JSON Request

Public API OMBD

This API allowed me to collect information from the website IMDb. For the purposes of this website, I wanted to find out how many seasons Doctor Who had been running for. Thankfully, this API allowed me to collect the general information profile from IMDb.

By calling the below function, the script would replace some innerHTML of a header tag on my website. The data tag held the returned data from the API and .totalSeasons stored how many seasons the season had been running for.

```

// Remote AJAX REST API load data
function loadData() {
    // Create AJAX link to data set
    var ourRequest = new XMLHttpRequest();

    // Creating a request with the OMDb API
    ourRequest.open('GET', 'http://www.omdbapi.com/?apikey=ffcc562f&i=tt0436992');

    // Load data
    ourRequest.onload = function() {
        // Managing errors
        if (ourRequest.status >= 200 && ourRequest.status < 400) {
            ourData = JSON.parse(ourRequest.responseText);
            renderHTML(ourData);
        } else {
            console.log("Server Error")
        }

        return ourData;
    }; // End of function

    // Check for errors
    ourRequest.onerror = function() {
        console.log("Connection Error")
    }

    ourRequest.send();
}; // End of load function

// Render the amount of seasons onto the webpage.
function renderHTML(data) {
    var htmlstring = '';

    // Read correct data from JSON file
    htmlstring += "<h1>" + data.totalSeasons + "</h1>"

    document.getElementById("HowManySeasons").innerHTML = htmlstring
};

```

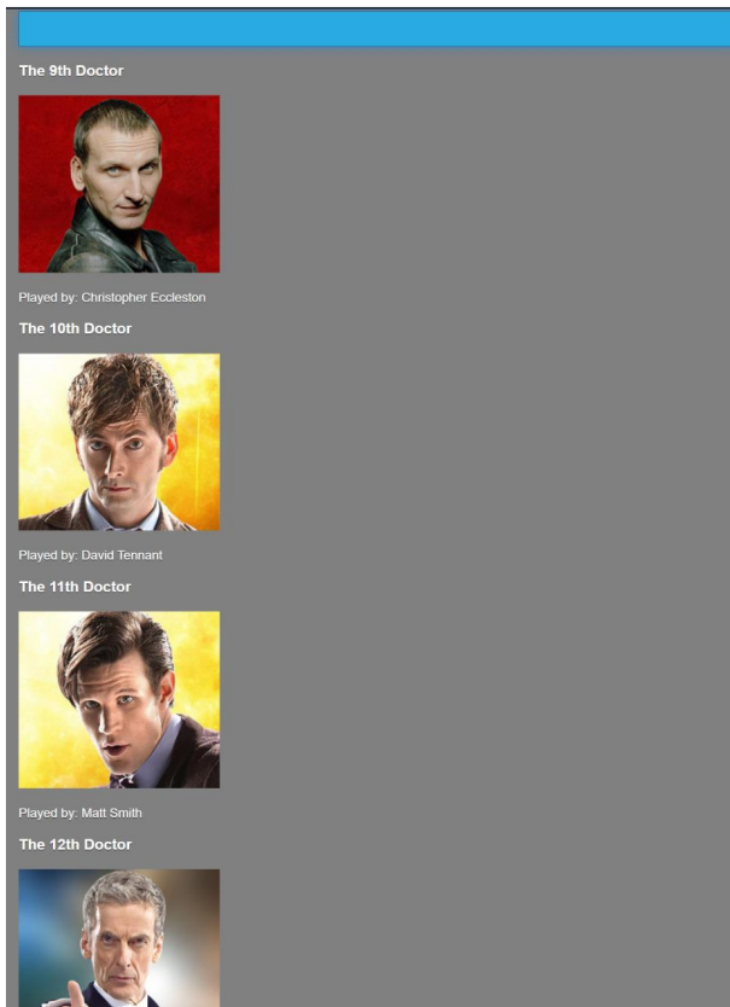
When the button is pressed, the API responds with the data and updates the <p>



Local AJAX JSON Request

The local file I had available was a JSON file containing the name of the different Doctors and a small string which was the directory position of an image of the doctor the section was referencing.

In the following image, you can see that I called the function 5 times. This was for each doctor of the modern series of Doctor Who. I stored the formatted HTML into a string and then by using the `i` which took the input value from when the function was first called, updated the correct HTML element on the webpage.



After pressing the button, the collapsibles are removed and the information gotten from the JSON file are added.

```

// Local Ajax Request Load data
function LoadDoctorData(dataIndex) {
    // Create AJAX Link to data set
    var doctorsRequest = new XMLHttpRequest();

    // Connect to the Local JSON file
    doctorsRequest.open('GET', 'doctors.json')

    // Load data
    doctorsRequest.onload = function() {
        // Managing errors
        if (doctorsRequest.status >= 200 && doctorsRequest.status < 400) {
            ourData = JSON.parse(doctorsRequest.responseText);

            lastElement = ourData.length - 1

            // Checking data is in range
            if (dataIndex < ourData.length) {
                buildDoctors(ourData, dataIndex);
            } else {
                buildDoctors(ourData, lastElement);
            }
        } else {
            console.log("Server Error")
        }

        return ourData;
    }; // End of function

    // Check for errors
    doctorsRequest.onerror = function() {
        console.log("Connection Error")
    }

    doctorsRequest.send();
}; // End of Load function

function buildDoctors(data, i) {

    var htmlstring = '';

    // Format strings for output
    htmlstring += "<div id=" + (i + 9) + " data-role=\"collapsible\" data-iconpos=\"bottom\"></div>"

    // Read data from JSON file for correct information
    htmlstring += '<h3>The ' + data[i].No + 'th Doctor</h3>'
    htmlstring += '</img>'
    htmlstring += '<p>Played by: ' + data[i].Data["Name"] + '</p>'

    htmlstring += "</div>"

    document.getElementById(i + 9).innerHTML = htmlstring

}

// Add button event listener
LoadBtn.addEventListener("click", function() {
    // Load data for each doctor
    LoadDoctorData(0)
    LoadDoctorData(1)
    LoadDoctorData(2)
    LoadDoctorData(3)
    LoadDoctorData(4)
});

```

Conclusion

In conclusion, it is very easy to create websites which are viable for the mobile platform with jQuery Mobile. I think using this technology it is very easy to create mobile-compatible websites in a short period of time whilst keeping standards high.

In addition to this, I think the use of REST API's and AJAX to access information outside of the standard CSS file is an amazing technology. I think it will allow for incredible features to be included in a simple way as you no longer have to implement the scripting inside of JavaScript and can easily link to a database.

The ability to add functionality to a website easily, without having to worry about its ability to resize and be functional on all platforms makes the process exceedingly easy. The extensive documentation of the library also allows for quick development.

Report

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

45/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

GRADING FORM: ASSIGNMENT2

DANIEL BENNETT

45

COGNITIVE/INTELLECTUAL

Analysis of the technical development of the application (eg how does your Javascript interact with the JSON file?) Why did you take the approach you did?

-  The report offers a very limited attempt at critical evaluation and analysis of concepts and principles, tending towards description. The student has provided very little evidence explaining how the JavaScript interacts with the JSON files or why they choose the approach they took.

6

KNOWLEDGE AND UNDERSTANDING

Annotated code listing (a code listing of relevant sections clearly commented is sufficient)

-  Your code is suitable but shows adequate knowledge and critical understanding of the well-established concepts and principles of web and mobile technologies, with a few gaps in the selection of material.

7

PRACTICAL SKILLS

-  The website contains the following requirements of the assignment: Created a 3 page site with navigation using a collapsible panel and demonstrates mobile website features. Implemented a customised style. Made a AJAX call to a local JSON file (Own data). The website doesn't provide AJAX call to a public JSON API The site is readable, with navigation and layout appropriate to the site. The application of the practical work is

32

mainly consistent, accurate and logical, however the website is a little simplistic. Overall, the student has used good and appropriate applications of established methods and techniques for developing a mobile website.