Coding with Kids

MyMDb (Dynamic Movie Website)

Coding with Kids > L5

Objectives

Students will be able to:

- Use Bootstrap to design a responsive and dynamic website
- Create a website with multiple tabs
- Make queries to a web API
- Make AJAX requests using jQuery
- Process JSON responses

Prerequisites:

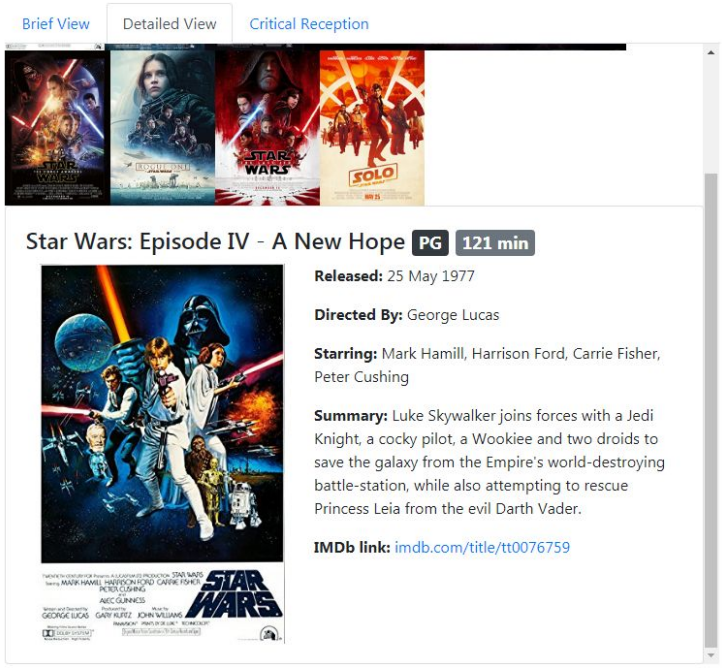
- Knowledge of Bootstrap and jQuery

Pacing:

- 3 - 4 weeks

Resources:

- Bootstrap Starter: codepen.io/CWKStarter/pen/QrrxQy
- OMDb API: omdbapi.com
- Bootstrap Documentation: getbootstrap.com/docs
- jQuery Documentation: api.jquery.com
- Completed Example Code



Project goals:

Basic Version

- Use the OMDb API to access movie data in JSON format.
- Parse the movie JSON data, and create multiple categorical displays.
- Use assorted Bootstrap features to give the page attractive styling.

Added Features

- Add a favorite movies tab, which displays the student's favorite films and a short review of them.

Overview (Links to Phases)

1. Warm Up
   a. Review concepts from previous week.
   b. Go over prerequisite material as needed.
   c. Students set up their project in Cloud 9
2. Motivation/Project Introduction
   a. Show students the version of the project that corresponds to what most of them will be working on.
   b. Discuss the design of the project
3. Phase 1 - Setting Up the Tabs
   a. Show students the documentation for Bootstrap Navs
   b. Discuss how to use Navs to create a dynamic tabbed interface
   c. Create a simple tabbed interface with 2+ tabs
4. Phase 2 - OMDb API
   a. Show students the OMDb API documentation
   b. Show the students how to query the API and what kind of information is there

Coding with Kids

MyMDb (Dynamic Movie Website)

Coding with Kids > L5

   c. Discuss how the results are given in JSON format, which is easy to work with in a JS program
   d. Students pick a movie or television franchise that they want to build a website around
5. **Phase 3 - Using jQuery to make API requests**
   a. Discuss how to use the jQuery "get" function to make API requests for JSON data
   b. Students write and test using the "get" function on a search query
6. **Phase 4 - Brief View Tab**
   a. Discuss how to use Bootstrap table styling
   b. Students create a brief view tab with basic information about the movie results from their "get" call
7. **Phase 5 - Detailed View Tab**
   a. Discuss how to use the Bootstrap "Collapse" component to create links which expand a div when clicked
   b. Discuss how to use Bootstrap and jQuery to only display one at a time
   c. Create a detailed view where the user clicks pictures to see more details about a movie
   d. Students use various Bootstrap styling to create an attractive display
8. **(Optional - Intermediate) Phase 6 - Reception tab**
   a. Discuss other ideas for tabs based on the contents of the JSON data we are working with
   b. Students create content for a third tab
9. **(Optional - Advanced) Additional Challenges**
10. **Completed Code**

Notes about project

This project relies heavily on Bootstrap and jQuery for its functionality. Make sure that your students have used these libraries prior to starting this project

IMPORTANT: The OMDb API requires an API key for making requests. API keys can be obtained here for free using an email address. You can either obtain one for you and your class, or have your students obtain their own. If you want students to obtain their own keys, make sure they have advanced notice. It is important to note that free keys are limited to 1000 requests per day, so students will need to be especially careful with how much they use the API. Due to the request limit, a good strategy here is to make a few API requests in the browser for data similar to what you will be using and save the JSON responses into variables in your JS file. This way you can experiment more with layout and design without constantly using up your request quota.

Warm Up

- Review the concept of external web dev libraries and touch on what Bootstrap and jQuery are useful for. Show students where to find the documentation for both libraries so that they can refer to them as you go.
  - Have the students do the initial setup for a web project in Cloud 9 at this point in time.
    - Have them copy the starter from CodePen into an HTML file in Cloud 9 and create CSS and JS files for the project
    - Remind them that their JS file needs to be linked AFTER any JS imports so that it can reference functions from the external library imports.
    - See the Completed Code section for the order of the imports

Motivation/Project Introduction

- Show the students the finished example and demonstrate the features of each tab and the responsiveness of the layout to resizing.
  - **Note: you will need to obtain and plug in your own OMDb API key into the finished code in the places with red text.**
- Explain that the majority of the design and functionality they see was created using the Bootstrap library, and that the movie data was pulled using the OMDb API.
  - Briefly show the students the documentation for the main Bootstrap features used in the example site:
    - Navs (used to create the tabs)
    - Tables (used for the first and third tabs)
    - Collapse (used on the second tab)
    - Grid (used for the layout in the collapsing elements)

- Badges (used in the detailed view to display rating and runtime)
- Show the students that the documentation gives examples of how all of the components work.
  - **Make sure to emphasize that it is more important to be able to navigate and read the documentation for Bootstrap and know the general idea of how certain components work rather than to memorize the exact CSS classes as they can change between versions of Bootstrap.**

Phase 1 - Setting Up the Tabs

Potential Stumbling Blocks

-

Differentiation Opportunities

- **Role and Aria attributes -** If students are interested in developing applications accessible by people with visual disabilities you can explain how the various role and aria tags are used by screen reading applications to describe the content which is currently being displayed. Students can add in these tags.

- **Make sure that students start the project from the starter kit as described in the Warm Up section. This will streamline the initial setup which requires linking bootstrap css and js files, and their dependencies.**

Detailed Walkthrough of Phase:

- Which Bootstrap component will we need to create dynamic tabs?
  - We will need to use “Navs”.
  - Have students open the documentation for Bootstrap Navs to the JavaScript behavior section:
    - getbootstrap.com/docs/4.1/components/navs/#javascript-behavior
  - Try letting the students figure out how to create the tabbed layout based on the first example provided in the documentation.
    - They can put simple text content in each tab to view them working.
    - Let students know that the “role” and “aria” attributes are meant for screen reading applications and do not affect functionality and can be left out if desired.
- Now that we have tabs, how can we position them?
  - We can use divs:
    - We can create an additional div around the tabs and content with the class “container-fluid” which is used for creating centered content.
  - We can use CSS to manipulate the size of the main outer div, as well as style the content div.
- Have the students finish and style their tab layouts.

Finished Code:

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyMDb</title>
        <!-- Required meta tags for Bootstrap and JQuery -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

        <!-- CSS imports -->
        <link rel="stylesheet" href="css/style.css" type="text/css" />
        <link rel="stylesheet"
href="https://code.jquery.com/ui/1.12.0/themes/ui-lightness/jquery-ui.css">
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.css"
>

        <!-- JS Imports -->
        <script src="https://code.jquery.com/jquery-3.3.1.js"></script>
        <script
src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"></script>
```

```
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.js"
></script>
        <!-- Rename the JQueryUI 'tooltip' and 'button' classes to
'uitooltip' and 'uibutton' so that they won't conflict with Bootstrap's
'tooltip' and 'button' classes. -->
        <script>
            $.widget.bridge('uitooltip', $.ui.tooltip);
            $.widget.bridge('uibutton', $.ui.button);
        </script>
        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.js"></
script>
        <script type="text/javascript" src="js/index.js"></script>
    </head>
    <body>
        <div id="movie-tabs" class="container-fluid">
            <ul class="nav nav-tabs" id="myTabs">
                <li class="nav-item">
                    <a class="nav-link active" id="brief-tab"
data-toggle="tab" href="#brief-view">Brief View</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" id="detail-tab" data-toggle="tab"
href="#detail-view">Detailed View</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" id="reception-tab"
data-toggle="tab" href="#reception">Critical Reception</a>
                </li>
            </ul>
            <div class="tab-content" id="myContent">
                <div class="tab-pane fade show active" id="brief-view">
                    Brief View Tab
                </div>
                <div class="tab-pane fade" id="detail-view">
                    Detailed View Tab
                </div>
                <div class="tab-pane fade" id="reception">
                    Reception Tab
                </div>
            </div>
        </div>
    </body>
</html>
```

CSS

```
#movie-tabs{
    width: 80%;
}

#myContent{
    height: 40rem;
    overflow-y: scroll;
}
```

Phase 2 - OMDb API

Potential Stumbling Blocks	Detailed Walkthrough of Phase:
•	• Navigate the students to the OMDb API: omdbapi.com

Differentiation Opportunities

-

- Explain:
  - The OMDb API contains a wealth of information on movies and tv series which we can make requests to and use the results to add content to our web pages.
  - In order to request information from the API we need an API key, which can be obtained for free using an email address and activated in the resulting email.
    - **Note: as mentioned previously you should either make one key for the class of notify students ahead of time to create one.**
  - Free API keys can only make 1,000 request to the API daily so we need to be careful not to overuse the key and only make requests a necessary amount of times.
  - The anatomy of an OMDb API request is as follows:
    - https://www.omdbapi.com/?[parameters]&apikey=**YOUR_API_KEY**
    - Parameters are listed as "name=value" and separated by the "&" character.
      - When searching for movies in this project it is good to include "type=movie" as a parameter.
    - **Note: using "http" rather than "https" will work in browser, but will fail when making the request in JS later on.**
  - There are two main ways to query the API
    - By title or IMDb ID
    - By search term
    - The parameters for both methods are listed on the front page of the API website.
  - Demonstrate searching by a term first by entering the following URL in a new tab
    - https://www.omdbapi.com/?s=star+wars&apikey=**YOUR_API_KEY**
    - Students should note that the result is in JSON (JavaScript Object Notation) format and the query only returns the top 10 results for the subject.
    - What is nice about the result being in JSON notation?
      - It is easy to parse in a JavaScript file.
  - Point out that the query gave us the IMDb IDs of all the results.
  - Open another new tab and run a query using one of the returned IDs:
    - https://www.omdbapi.com/?i=tt0086190&apikey=**YOUR_API_KEY**
    - What is different about the results of this query?
      - The information is much more detailed compared to the search results.
  - How could we get detailed results on all movies in a search?
    - We could iterate through the results and request the information using the IMDb ID.
- Have the students pick a movie series to work with, query for it, and keep the result open in a browser tab.
  - Good choices are series with 10 or more official installments with the name of the series in the title.
  - Encourage students to use either Star Wars or Star Trek as both series meet the 10 or more installments recommendation.
    - **Note: there is no content filtering in the API so search carefully!**
- Have the students query the ID for one or two of the movies listed and leave the results open in new tabs.

Example Queries:

Searching Star Wars:

`https://www.omdbapi.com/?s=star+wars&type=movie&apikey=YOUR_API_KEY`

Looking up two Star Wars films by ID:

```
https://www.omdbapi.com/?i=tt0076759&apikey=YOUR_API_KEY
https://www.omdbapi.com/?i=tt0086190&apikey=YOUR_API_KEY
```

Searching Star Trek:

```
https://www.omdbapi.com/?s=star+trek&type=movie&apikey=YOUR_API_KEY
```

Looking up two Star Trek films by ID:

```
https://www.omdbapi.com/?i=tt1408101&apikey=YOUR_API_KEY
https://www.omdbapi.com/?i=tt2660888&apikey=YOUR_API_KEY
```

Phase 3 - Using jQuery to make API requests

Potential Stumbling Blocks

-

Differentiation Opportunities

-

Detailed Walkthrough of Phase:

- We have demonstrated making API requests in browser and have seen the results. How can we actually use the result in a program?
  - Students may suggest copying the results from the open tabs and pasting them as variable contents in their program.
    - **This is a good strategy for debugging the program with stock data (as to not exhaust our API quota during the design phase, but limits the customization of our site).**
  - Explain that we can use jQuery to query the API and process the results using the function "jQuery.get" and have students navigate to the documentation:
    - api.jquery.com/jQuery.get/
  - This function is a shorthand Ajax function which is used to send a GET request to a server and set up an event handler to process the result.
  - We can provide the function with an object literal with a "url" property (the OMDb search), and a "success" property which is a callback function taking in the three standard parameters (shown below).
- Where should we put the call to jQuery.get?
  - To follow best practices we should put it inside "$(function(){});" (the shorthand document ready event handler).
- To make the site more easily customized we can create a variable for the search term and concatenate it into the result (as shown below).
- For now have the success function simply log the resulting data so that students can get an idea of what they will be working with.

Finished Code:

```
$(function(){
    var searchTerm = "Star Wars";

    $.get({
        url:
"https://www.omdbapi.com/?s="+searchTerm+"&type=movie&apikey=YOUR_API_KEY",
        success: function(data, textStatus, jqXHR){
            console.log(data);
        }

    });
});
```

Phase 4 - Brief View Tab

Coding with Kids

MyMDb (Dynamic Movie Website)

Coding with Kids > L5

Potential Stumbling Blocks

- **Exhausting the API key quota** - Debugging a site like this tends to require running it many times, which has the potential to exhaust the API key quota. You can counteract this by commenting out the "get" call and storing the results of some requests in variables and using those for debugging purposes.

Differentiation Opportunities

- **Sorting results** - The results returned are in order they would come up in an IMDb search. If students want to sort them chronologically instead, show them how to create a comparison function and use the built in "sort" method on the array of search results (see finished code).

Detailed Walkthrough of Phase:

- What does the "Brief View" tab consist of?
  - The movie poster image, title and year in a table.
- Have the students navigate to the Bootstrap table documentation:
  - getbootstrap.com/docs/4.1/content/tables/
  - Have the students look through the various table settings which are available.
  - Guide them to the "Striped rows" section.
- What table content should we have in our HTML? What should we populate in JS?
  - We should create the content for the "thead" tag in HTML since it will always be the same, but we should leave the "tbody" tag blank so that it can be populated with the data from our API request.
- How can we fill the table with content from our request results?
  - We can use a for loop to iterate through the array of results stored under the "Search" property in the JSON data.
  - On each iteration we can use the jQuery function "append" to add a new row with entries for each column to the table in the brief view tab.
    - Appending this much text becomes very long quickly. To get around this you can use string concatenation to distribute the text across multiple lines.
    - See finished code for how to use append in this case.
- Have the students create and test the functionality to create a brief view.

Finished Code:

HTML

```
            <div class="tab-content" id="myContent">
                <div class="tab-pane fade show active" id="brief-view" >
                    <table class="table table-striped">
                        <thead>
                            <th scope="col">Poster</th>
                            <th scope="col">Title</th>
                            <th scope="col">Year</th>
                        </thead>
                        <tbody></tbody>
                    </table>
                </div>
                <div class="tab-pane fade" id="detail-view">
                </div>
                <div class="tab-pane fade" id="reception">
                </div>
            </div>
```

CSS

```
td > img{
    height: 6rem;
}
```

JS

```
$(function(){
    var results;
    var searchTerm = "Star Wars";

    // optional for sorting results by year
    function compareMovies(m1, m2){
        if(m1.Year < m2.Year){
            return -1;
        }else if (m1.Year > m2.Year){
            return 1
```

Coding with Kids

MyMDb (Dynamic Movie Website)

Coding with Kids > L5

```
        }
        return 0;
    }

    // Commenting this out for now to not exhaust my API quota
    $.get({
        url:
"https://www.omdbapi.com/?s="+searchTerm+"&type=movie&apikey=YOUR_API_KEY",
        success: function(data, textStatus, jqXHR){
            results = data["Search"];
            results.sort(compareMovies);
            for(var i=0; i<results.length; i++){
                $("#brief-view > table > tbody").append("<tr>"+
                        "<td><img src='"+results[i]["Poster"]+"'
title='"+results[i]["Title"]+"'</img></td>"+
                        "<td
class='align-middle'>"+results[i]["Title"]+"</td>"+
                        "<td
class='align-middle'>"+results[i]["Year"]+"</td>"+
                        "</tr>");
            }

        }

    });
});
```

Phase 5 - Detailed View Tab

Potential Stumbling Blocks

-

Differentiation Opportunities

-

Detailed Walkthrough of Phase:

- Bring the students attention to the “detailed view” tab.
- What Bootstrap component can we use to create the detailed view?
  - We can use the Collapse component:
    - getbootstrap.com/docs/4.1/components/collapse/
- How can we get the data for the detailed view tab?
  - Students should notice that the data is from a request made using an IMDb ID and not a search.
  - To get the data we could call the “jQuery.get” function using the IMDb ID of each result we iterate through in the for loop we created in the last phase.
  - To save the data we could create an array in the outer function and push the results into the array in the “success” handler.
- We won’t want to process the results until they have all been returned. How can we tell if all the results are back? Where should we process them?
  - We could set up a counter which starts at zero and is incremented each time the success function runs. When the count reaches the amount of results we have we know that all results have been returned and we can process them.
  - We can process the results in the success handler after all the results have been returned back.
- Bring the students attention to the “Multiple Targets” section on the Collapse documentation page.
  - Based on the example how can we create a detailed view where images are clicked to expand a more detailed element?
    - We can use anchor elements with image elements inside of them as the trigger to expand the collapsed elements which will contain more detailed information about the movies.

- We need to have the attribute data-toggle="collapse" and set the href to the id of the collapse. What could we use as the id?
    - We could just use the IMDb ID since it is guaranteed to be unique.
- Due to the organization of the elements in the example we will need to iterate through the items twice. Once to append the image/anchor elements and a second time to append all the collapse elements to the detailed view div.

- How can we create the collapse elements?
    - We can put all the content for the collapse inside of nested divs.
    - The outer div should have the class “collapse” and the IMDb ID as the element id.
    - The inner div should have the class “card card-body” to give it an aesthetically pleasing look.
    - Beyond this students can add whatever content they want. The example code uses the bootstrap grid system for content placement and badge components for additional style. (see below for ideas of what to include in this view).

- As the students work, they will likely notice that the collapse elements do not go away when another is clicked, but when the image is clicked twice.
    - How can we make it so that any open collapse will close when another image is clicked?
        - We can add a click event to the anchor elements in the detail view, which selects all collapse elements and calls the function “collapse” on them passing in the parameter “hide” (more information on this is at the bottom of the Collapse documentation).
- Have the students experiment with the design of their detailed view and help with tips and debugging.

Finished Code:

CSS

```
#detail-view > * > img{
    height: 10rem;
}
.row > * > img{
    width: 100%;
}
```

JS

```
$(function(){
    var results;
    var detail = [];
    var searchTerm = "Star Wars";

    $.get({
        url:
"https://www.omdbapi.com/?s="+searchTerm+"&type=movie&apikey=YOUR_API_KEY",
        success: function(data, textStatus, jqXHR){
            results = data["Search"];
            results.sort(compareMovies);
            var count = 0;
            detail = [];
            for(var i=0; i<results.length; i++){
                $("#brief-view > table > tbody").append("<tr>"+
                        "<td><img src='"+results[i]["Poster"]+"'
title='"+results[i]["Title"]+"'</img></td>"+
```

Coding with Kids

MyMDb (Dynamic Movie Website)

Coding with Kids > L5

```
                        "<td
class='align-middle'>"+results[i]["Title"]+"</td>"+
                        "<td
class='align-middle'>"+results[i]["Year"]+"</td>"+
                        "</tr>");
                $.get({
                    url:
"https://www.omdbapi.com/?i="+results[i]["imdbID"]+"&apikey=YOUR_API_KEY",
                    success: function(data, textStatus, jqXHR){
                        detail.push(data);
                        count++;
                        if (count == results.length){
                            detail.sort(compareMovies);
                            for(var i=0; i<detail.length; i++){
                                $("#detail-view").append("<a
data-toggle='collapse' href='#"+detail[i]["imdbID"]+"'>"+
                                    "<img src='"+detail[i]["Poster"]+"'
title='"+detail[i]["Title"]+"'></img></a>");
                            }
                            for(var i=0; i<detail.length; i++){
                                $("#detail-view").append("<div
class='collapse' id='"+detail[i]["imdbID"]+"'>"+
                                    "<div class='card card-body'>"+
                                    "<h4>"+detail[i]["Title"]+
                                    " <span class='badge
badge-dark'>"+detail[i]["Rated"]+"</span>"+
                                    " <span class='badge
badge-secondary'>"+detail[i]["Runtime"]+"</span></h4>"+
                                    "<div class='container'><div
class='row'>"+
                                    "<div class='col-lg-3 col-md-5
col-sm-6'>"+
                                    "<img src='"+detail[i]["Poster"]+"'
title='"+detail[i]["Title"]+"'></img></div>"+
                                    "<div class='col-lg-9 col-md-7
col-sm-6'>"+
                                    "<p><b>Released:
</b>"+detail[i]["Released"]+"</p>"+
                                    "<p><b>Directed By:
</b>"+detail[i]["Director"]+"</p>"+
                                    "<p><b>Starring:
</b>"+detail[i]["Actors"]+"</p>"+
                                    "<p><b>Summary:
</b>"+detail[i]["Plot"]+"</p>"+
                                    "<p><b>IMDb link: </b><a
href='https://www.imdb.com/title/"+detail[i]["imdbID"]+
"'>imdb.com/title/"+detail[i]["imdbID"]+"</a></p>"+
                                    "</div></div></div></div></div>");
                            }
                            $("#detail-view > a").click(function(){
                                $(".collapse").collapse("hide")
                            });
                        }
                    }
                });
            }

        }

    });
```

```
});
```

(Optional - Intermediate) Phase 6 - Reception tab

Potential Stumbling Blocks

-

Differentiation Opportunities

- **Error Handling -** Not every movie in the OMDb database has ratings from all three ratings sources. If students run into this problem with the search they chose they can construct the HTML string to append in a for loop using a "try-except" structure to errors which prevent the content from loading.

Detailed Walkthrough of Phase:

- How can we add a reception tab?
  - We could add a table to our HTML in a third tab with a table header and an empty table body, and populate the table in JS.
- What columns could the table have?
  - We could have a poster picture along with ratings from IMDb, Rotten Tomatoes, Metacritic and any awards the film won.
- Where can we find the data to populate the columns?
  - By looking through the JSON data from the individual movie queries we can see that there is an "Awards" property which has info about the awards and a "Ratings" property which contains an array with IMDb, RT, and MC ratings.
  - We can populate the columns using this data (see below for details).
- Have the students add in a reception tab.

Finished Code:

HTML

```
<div class="tab-pane fade" id="reception">
    <table class="table table-striped">
        <thead>
            <th scope="col">Poster</th>
            <th scope="col">IMDb Score</th>
            <th scope="col">Rotten Tomatoes</th>
            <th scope="col">Metacritic</th>
            <th scope="col">Awards</th>
        </thead>
        <tbody></tbody>
    </table>
</div>
```

JS

```
$(function(){

    $.get({
        url:
"https://www.omdbapi.com/?s="+searchTerm+"&type=movie&apikey=YOUR_API_KEY",
        success: function(data, textStatus, jqXHR){
            results = data["Search"];
            results.sort(compareMovies);
            var count = 0;
            detail = [];
            for(var i=0; i<results.length; i++){
                // see previous phases
                $.get({
                    url:
"https://www.omdbapi.com/?i="+results[i]["imdbID"]+"&apikey=YOUR_API_KEY",
                    success: function(data, textStatus, jqXHR){
                        detail.push(data);
                        count++;
                        if (count == results.length){
                            detail.sort(compareMovies);
                            for(var i=0; i<detail.length; i++){
                                $("#reception > table >
tbody").append("<tr>"+
```

```
                                        "<td><img
src='"+results[i]["Poster"]+"' title='"+results[i]["Title"]+"'</img></td>"+
                                        "<td
class='align-middle'>"+detail[i]["Ratings"][0]["Value"]+"</td>"+
                                        "<td
class='align-middle'>"+detail[i]["Ratings"][1]["Value"]+"</td>"+
                                        "<td
class='align-middle'>"+detail[i]["Ratings"][2]["Value"]+"</td>"+
                                        "<td
class='align-middle'>"+detail[i]["Awards"]+"</td>"+
                                        "</tr>");
                                    // see previous phase
                                }
                                for(var i=0; i<detail.length; i++){
                                    // see previous phase
                                }
                                $("#detail-view > a").click(function(){
                                    $(".collapse").collapse("hide")
                                });
                            }
                        }
                    });
                }

            }

        });
});
```

(Optional - Advanced) Phase 7 - Additional Challenges

- **Favorites tab -** Students can create a favorites tab which lists out all of their favorite movies and a paragraph with their thoughts on the movies and why they like them.

Completed Example Code

HTML (Red Text needs to be changed)

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyMDb</title>
        <!-- Required meta tags for Bootstrap and JQuery -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

        <!-- CSS imports -->
        <link rel="stylesheet" href="css/style.css" type="text/css" />
        <link rel="stylesheet" href="https://code.jquery.com/ui/1.12.0/themes/ui-lightness/jquery-ui.css">
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.css">

        <!-- JS Imports -->
        <script src="https://code.jquery.com/jquery-3.3.1.js"></script>
        <script src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.js"></script>
        <!-- Rename the JQueryUI 'tooltip' and 'button' classes to 'uitooltip' and 'uibutton' so that they won't
conflict with Bootstrap's 'tooltip' and 'button' classes. -->
        <script>
            $.widget.bridge('uitooltip', $.ui.tooltip);
            $.widget.bridge('uibutton', $.ui.button);
```

Coding with Kids

MyMDb (Dynamic Movie Website)

Coding with Kids > L5

```
        </script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.js"></script>
        <script type="text/javascript" src="js/index.js"></script>
    </head>
    <body>

        <div id="head">
            <h1>Welcome to MyMDb</h1>
            <h4 id='status'></h4>
        </div>
        <div id="movie-tabs" class="container-fluid">
            <ul class="nav nav-tabs" id="myTabs">
                <li class="nav-item">
                    <a class="nav-link active" id="brief-tab" data-toggle="tab" href="#brief-view">Brief
View</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" id="detail-tab" data-toggle="tab" href="#detail-view">Detailed View</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" id="reception-tab" data-toggle="tab" href="#reception">Critical
Reception</a>
                </li>
            </ul>
            <div class="tab-content" id="myContent">
                <div class="tab-pane fade show active" id="brief-view" >
                    <table class="table table-striped">
                        <thead>
                            <th scope="col">Poster</th>
                            <th scope="col">Title</th>
                            <th scope="col">Year</th>
                        </thead>
                        <tbody>

                        </tbody>
                    </table>
                </div>
                <div class="tab-pane fade" id="detail-view">
                </div>
                <div class="tab-pane fade" id="reception">
                    <table class="table table-striped">
                        <thead>
                            <th scope="col">Poster</th>
                            <th scope="col">IMDb Score</th>
                            <th scope="col">Rotten Tomatoes</th>
                            <th scope="col">Metacritic</th>
                            <th scope="col">Awards</th>
                        </thead>
                        <tbody>

                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </body>
</html>
```

CSS

```
td > img{
    height: 6rem;
}

#head{
```

Coding with Kids

MyMDb (Dynamic Movie Website)

Coding with Kids > L5

```
    margin:auto;
    text-align: center !important;
}

#movie-tabs{
    width: 80%;
}

#myContent{
    height: 40rem;
    overflow-y: scroll;
}

#detail-view > * > img{
    height: 10rem;
}
.row > * > img{
    width: 100%;
}
```

JavaScript

```
$(function(){
    var results;
    var detail = [];
    var searchTerm = "Star Wars";

    function compareMovies(m1, m2){
        if(m1.Year < m2.Year){
            return -1;
        }else if (m1.Year > m2.Year){
            return 1
        }
        return 0;
    }

    $.get({
        url: "https://www.omdbapi.com/?s="+searchTerm+"&type=movie&apikey=API_KEY_HERE",
        success: function(data, textStatus, jqXHR){
            if(textStatus === "success"){
                results = data["Search"];
                results.sort(compareMovies);
                var count = 0;
                detail = [];
                $("#status").html(data["totalResults"]+" results for '"+searchTerm+"', "+
                    "displaying the Top 10");
                for(var i=0; i<results.length; i++){
                    $("#brief-view > table > tbody").append("<tr>"+
                            "<td><img src='"+results[i]["Poster"]+"'
title='"+results[i]["Title"]+"'</img></td>"+
                            "<td class='align-middle'>"+results[i]["Title"]+"</td>"+
                            "<td class='align-middle'>"+results[i]["Year"]+"</td>"+
                            "</tr>");
                    $.get({
                        url: "https://www.omdbapi.com/?i="+results[i]["imdbID"]+"&apikey=API_KEY_HERE",
                        success: function(data, textStatus, jqXHR){
                            detail.push(data);
                            count++;
                            if (count == results.length){
                                detail.sort(compareMovies);
                                for(var i=0; i<detail.length; i++){
                                    $("#reception > table > tbody").append("<tr>"+
                                        "<td><img src='"+results[i]["Poster"]+"'
title='"+results[i]["Title"]+"'</img></td>"+
                                        "<td class='align-middle'>"+detail[i]["Ratings"][0]["Value"]+"</td>"+
```

```
                                        "<td class='align-middle'>"+detail[i]["Ratings"][1]["Value"]+"</td>"+
                                        "<td class='align-middle'>"+detail[i]["Ratings"][2]["Value"]+"</td>"+
                                        "<td class='align-middle'>"+detail[i]["Awards"]+"</td>"+
                                        "</tr>");
                                    $("#detail-view").append("<a data-toggle='collapse'
href='#"+detail[i]["imdbID"]+"'>"+
                                        "<img src='"+detail[i]["Poster"]+"'
title='"+detail[i]["Title"]+"'></img></a>");
                                }
                                for(var i=0; i<detail.length; i++){
                                    $("#detail-view").append("<div class='collapse'
id='"+detail[i]["imdbID"]+"'>"+
                                        "<div class='card card-body'>"+
                                        "<h4>"+detail[i]["Title"]+
                                        " <span class='badge badge-dark'>"+detail[i]["Rated"]+"</span>"+
                                        " <span class='badge
badge-secondary'>"+detail[i]["Runtime"]+"</span></h4>"+
                                        "<div class='container'><div class='row'>"+
                                        "<div class='col-lg-3 col-md-5 col-sm-6'>"+
                                        "<img src='"+detail[i]["Poster"]+"'
title='"+detail[i]["Title"]+"'></img></div>"+
                                        "<div class='col-lg-9 col-md-7 col-sm-6'>"+
                                        "<p><b>Released: </b>"+detail[i]["Released"]+"</p>"+
                                        "<p><b>Directed By: </b>"+detail[i]["Director"]+"</p>"+
                                        "<p><b>Starring: </b>"+detail[i]["Actors"]+"</p>"+
                                        "<p><b>Summary: </b>"+detail[i]["Plot"]+"</p>"+
                                        "<p><b>IMDb link: </b><a
href='https://www.imdb.com/title/"+detail[i]["imdbID"]+
                                        "'>imdb.com/title/"+detail[i]["imdbID"]+"</a></p>"+
                                        "</div></div></div></div></div>");
                                }
                                $("#detail-view > a").click(function(){
                                    $(".collapse").collapse("hide")
                                });
                            }
                        }
                    });
                }

            }
        }

    });
});
```