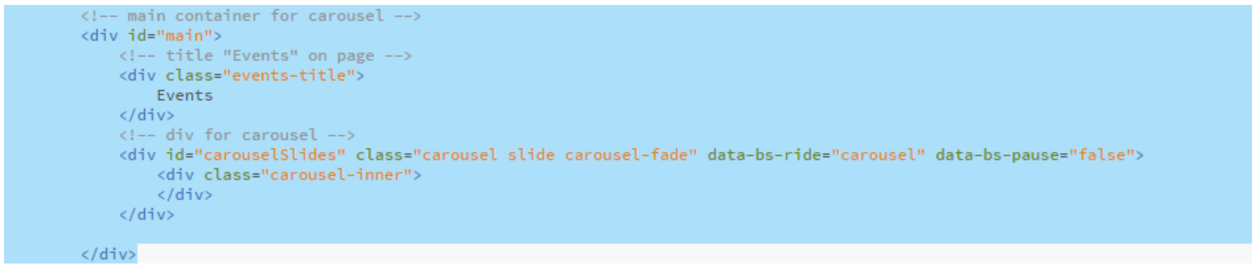
Event Template Documentation

Project Overview

The purpose of this program is to maintain and present a carousel that transitions between events that appear for the museum of natural history using a new API. This program tracks the time, venue and description of events that are sourced from the API and displays them in a format given by the DCA in this template. This template can also be used to display data on exhibits. These are sourced from several sites that encompass several different locales and different types of events/exhibits given by the API.

Carousel

The carousel is designed using bootstrap, though it has been modified to be dynamic in size to fit the number of given events/exhibits using Javascript. The API feed is parsed via Javascript and all relevant information is then extracted via JSON format. This data includes information such as the starting time of events and the ending time of events, the event/exhibit location and the summary description of the event/exhibit. After the information has been extracted it is then inserted into containers (divs) in the carousel and displayed with the given timer interval that is set in Javascript.



```
<!-- main container for carousel -->
<div id="main">
    <!-- title "Events" on page -->
    <div class="events-title">
        Events
    </div>
    <!-- div for carousel -->
    <div id="carouselSlides" class="carousel slide carousel-fade" data-bs-ride="carousel" data-bs-pause="false">
        <div class="carousel-inner">
        </div>
    </div>

</div>
```

Script

The Javascript portion of this project is the most influential. The majority of actions and design decisions occur in the Javscript portion of this program. The primary datapoints are represented by the feedURL, venue_name and event_or_exhibit variables that are determined at runtime but can be easily changed at the top of the script. The feedURL is the given link for the API that is to be parsed. The venue_name given is determinant of which location/venue of events/exhibits to extract from the JSON feed given by the feedURL. The event_or_exhibit variable defines whether to extract events or exhibits from the given feed. These can all be easily changed to suite your needs. Considering adding an interval variable for the time delay between carousel slides at the top of the script. After these are defined the program initiates an async function that awaits a promise from the server for the given API link. If the promised response is null, then a “no events” slide is shown. Otherwise, the response is then parsed via JSON into a variable named data to locally store the information obtained from the API.

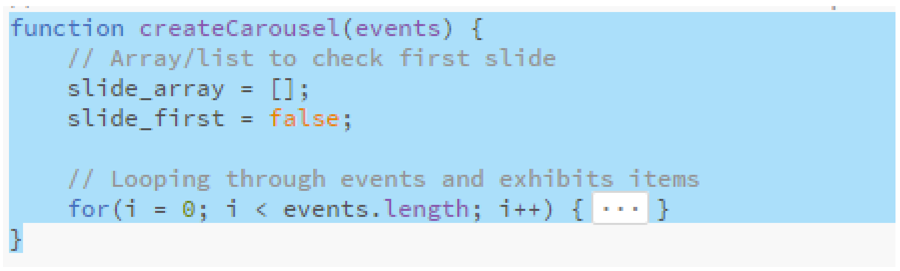
```
// JSON feed URL for events and exhibits
//const feedURL = 'http://nmdcamediadev.wpengine.com/wp-json/tribe/events/v1/events';
const feedURL = 'https://test-dca-mc.nmdca.net/wp-json/tribe/events/v1/events';
// Name of venue for events/exhibits
const venue_name = "New Mexico History Museum";
// Category to search for Event or Exhibits
const event_or_exhibit = "Event";
```

Function

The getData function simply parses the API feed given at the top of the script and then extracts relevant information from the given feed to fit the template.

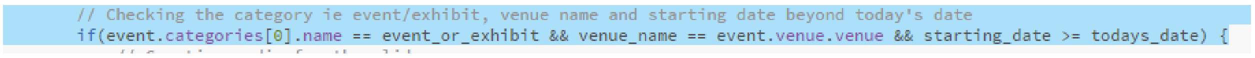
```
// async function to parse json and events/exhibits into slides for a carousel
getData();

// async function to parse json and events/exhibits into slides for a carousel
async function getData() {
    // fetch to grab the promised response from the server
    const response = await fetch(feedURL, { mode: 'cors'});
    // grabbing the JSON data from the feed url promised response
    const data = await response.json();

    // If data is null, placing an empty or no event/exhibit slide
    if (data == null) {
        // Creating a slide div to place into the carousel
        const slide = document.createElement('div');
        // Setting carousel to item/slide to active via class
        slide.className = "carousel-item active";
        // Setting the interval between slides
        slide.setAttribute('data-bs-interval',15000);
        // Using DCA logo in place of other images
        image = "assets/thumbnail_dca-logo.png";
        // Creating the carousel inner HTML with divs and class settings
        slide.innerHTML = `

                <div class="container-fluid p-0 text-white">
                    <div class="row">
                        <div class="col offset-3">
                            <p class="p-0 m-0 event-activities-type">ACTIVITIES</p>
                            <p class="event-datetime-text"> No Event Date <span class="pipe"> | </span> No Event Time </p>
                        </div>
                    </div>
                    <div class="row event-location-container">
                        <p class="col offset-3 text-dark">No Venue</p>
                    </div>
                    <div class="row">
                        <div class="col event-description-text">
                            <div class="pe-2 justify-content-end">
                                <p class="event-title">No Event Description</p>
                            </div>
                        </div>
                    </div>
                    <div class="row mt-3" style="height: 50vh">
                        <div class="col d-flex justify-content-center" style="height: 60vh">
                            <img src=${image} class="img-fluid object-fit-cover h-100">
                        </div>
                    </div>
                </div>
            `;
        // Placing the inner HTML slide into the carousel div
        document.querySelector('.carousel-inner').appendChild(slide);
    } else {
        // What is being responded with if nothing, if events create carousel
        createCarousel(data.events);
    }
}
```

After the feed has been stored in the data variable and the data is a non-null value, a function named createCarousel is called.



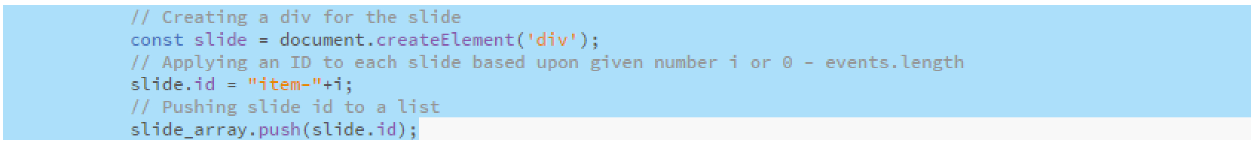
```
function createCarousel(events) {
    // Array/list to check first slide
    slide_array = [];
    slide_first = false;

    // Looping through events and exhibits items
    for(i = 0; i < events.length; i++) {···}
}
```

This function simply establishes the carousel and the contents of the carousel. To start, the events/exhibits are looped through to find all the given events or exhibits determined by the event_or_exhibit variable. If the time of the event/exhibit is beyond the current date and time and if the event/exhibit venue matches the name of the given venue_name variable, given at the top of the script, as well as matching the given event/exhibit category, then a slide is created.



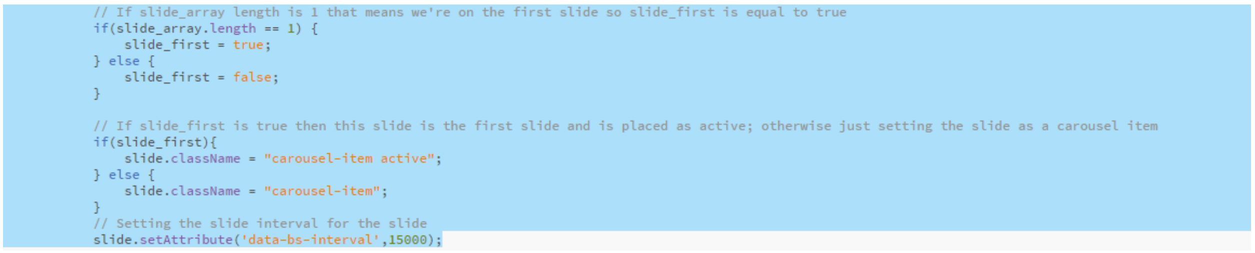
```
        // Checking the category ie event/exhibit, venue name and starting date beyond today's date
        if(event.categories[0].name == event_or_exhibit && venue_name == event.venue.venue && starting_date >= todays_date) {
```

This slide is an HTML div which is later inserted into the carousel. The div is then IDed by the ordered number of the given event in the loop.



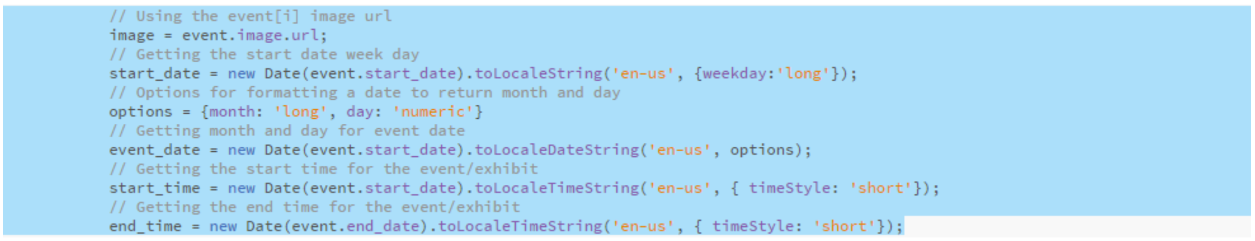
```
            // Creating a div for the slide
            const slide = document.createElement('div');
            // Applying an ID to each slide based upon given number i or 0 - events.length
            slide.id = "item-"+i;
            // Pushing slide id to a list
            slide_array.push(slide.id);
```

The number of the event/exhibit is checked to see whether it is the first slide. If it is the first slide, then its' class is set to active. This determines the first slide shown. Then the timing interval for that slide is set for the current slide in milliseconds.



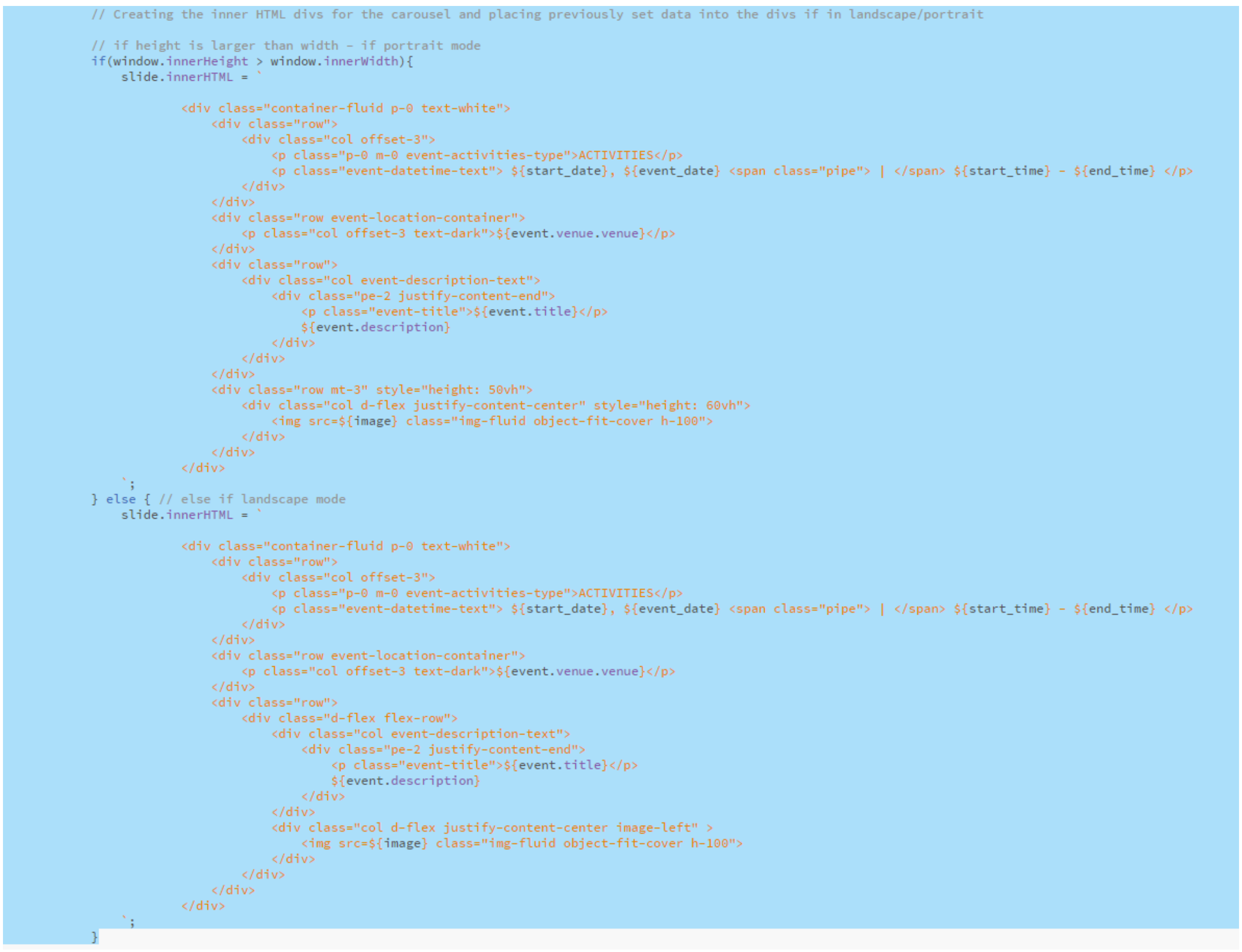
```
            // If slide_array length is 1 that means we're on the first slide so slide_first is equal to true
            if(slide_array.length == 1) {
                slide_first = true;
            } else {
                slide_first = false;
            }

            // If slide_first is true then this slide is the first slide and is placed as active; otherwise just setting the slide as a carousel item
            if(slide_first){
                slide.className = "carousel-item active";
            } else {
                slide.className = "carousel-item";
            }
            // Setting the slide interval for the slide
            slide.setAttribute('data-bs-interval',15000);
```

All relevant information for the current event/exhibit is then extracted to fill the current slide.



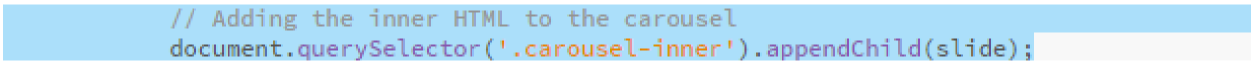
```
            // Using the event[i] image url
            image = event.image.url;
            // Getting the start date week day
            start_date = new Date(event.start_date).toLocaleString('en-us', {weekday:'long'});
            // Options for formatting a date to return month and day
            options = {month: 'long', day: 'numeric'}
            // Getting month and day for event date
            event_date = new Date(event.start_date).toLocaleDateString('en-us', options);
            // Getting the start time for the event/exhibit
            start_time = new Date(event.start_date).toLocaleTimeString('en-us', { timeStyle: 'short'});
            // Getting the end time for the event/exhibit
            end_time = new Date(event.end_date).toLocaleTimeString('en-us', { timeStyle: 'short'});
```

This information includes an image URL for the given event image, the start date and end date, venue location and description. This data is then formatted and placed into divs and inserted into the slide's inner HTML.



```
// Creating the inner HTML divs for the carousel and placing previously set data into the divs if in landscape/portrait

// if height is larger than width - if portrait mode
if(window.innerHeight > window.innerWidth){
    slide.innerHTML = `

            <div class="container-fluid p-0 text-white">
                <div class="row">
                    <div class="col offset-3">
                        <p class="p-0 m-0 event-activities-type">ACTIVITIES</p>
                        <p class="event-datetime-text"> ${start_date}, ${event_date} <span class="pipe"> | </span> ${start_time} - ${end_time} </p>
                    </div>
                </div>
                <div class="row event-location-container">
                    <p class="col offset-3 text-dark">${event.venue.venue}</p>
                </div>
                <div class="row">
                    <div class="col event-description-text">
                        <div class="pe-2 justify-content-end">
                            <p class="event-title">${event.title}</p>
                            ${event.description}
                        </div>
                    </div>
                </div>
                <div class="row mt-3" style="height: 50vh">
                    <div class="col d-flex justify-content-center" style="height: 60vh">
                        <img src=${image} class="img-fluid object-fit-cover h-100">
                    </div>
                </div>
            </div>
        `;
} else { // else if landscape mode
    slide.innerHTML = `

            <div class="container-fluid p-0 text-white">
                <div class="row">
                    <div class="col offset-3">
                        <p class="p-0 m-0 event-activities-type">ACTIVITIES</p>
                        <p class="event-datetime-text"> ${start_date}, ${event_date} <span class="pipe"> | </span> ${start_time} - ${end_time} </p>
                    </div>
                </div>
                <div class="row event-location-container">
                    <p class="col offset-3 text-dark">${event.venue.venue}</p>
                </div>
                <div class="row">
                    <div class="d-flex flex-row">
                        <div class="col event-description-text">
                            <div class="pe-2 justify-content-end">
                                <p class="event-title">${event.title}</p>
                                ${event.description}
                            </div>
                        </div>
                        <div class="col d-flex justify-content-center image-left" >
                            <img src=${image} class="img-fluid object-fit-cover h-100">
                        </div>
                    </div>
                </div>
            </div>
        `;
}
```

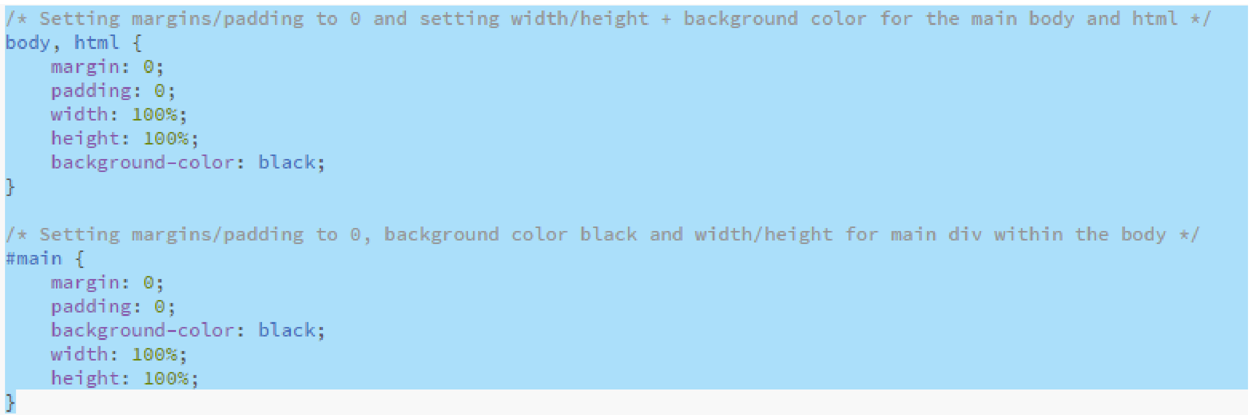
After this the slide is simply appended/added to the page's carousel div and displayed appropriately.



```
// Adding the inner HTML to the carousel
document.querySelector('.carousel-inner').appendChild(slide);
```

CSS

The CSS or cascading stylesheet aids in helping to design and format the given HTML. The margin and padding of the page are initially set to 0 pixels using the body and html tags in CSS. The width and height are set to 100% but may need to be changed to fit different screens in different orientations.

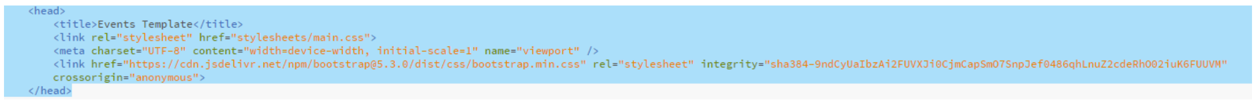


```
/* Setting margins/padding to 0 and setting width/height + background color for the main body and html */
body, html {
    margin: 0;
    padding: 0;
    width: 100%;
    height: 100%;
    background-color: black;
}

/* Setting margins/padding to 0, background color black and width/height for main div within the body */
#main {
    margin: 0;
    padding: 0;
    background-color: black;
    width: 100%;
    height: 100%;
}
```

A vertical or portrait orientation may require defining the pixel width and height of the display in use. The background color is also set to black. Each variable class simply uses the viewport width (vw) to maintain the size of the fonts for the page and determines the font family, as well as color of the font. Most of the formatting is done via Javascript so there isn't much done in the stylesheet. All of the fonts, font sizes and font families can easily be changed here.

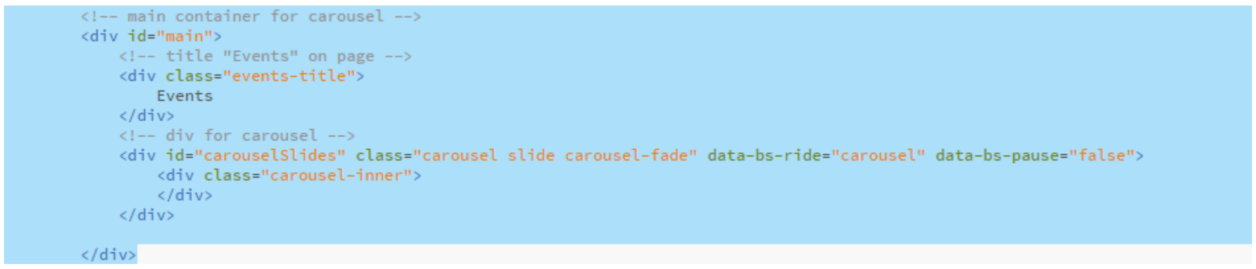
HTML

The HTML is very minimal, as most of the design and operation of this page is done via Javascript. The head tag contains the title and links to the stylesheet, as well as external resources for using bootstrap.

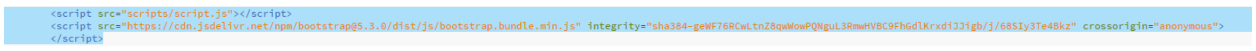


```
<head>
    <title>Events Template</title>
    <link rel="stylesheet" href="stylesheets/main.css">
    <meta charset="UTF-8" content="width=device-width, initial-scale=1" name="viewport" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM"
    crossorigin="anonymous">
</head>
```

The body simply contains a "main" container div. This container contains a div for the event title "Events", which can be changed. The main container also contains the primary div container for the carousel. This carousel is IDed as "carouselSlides". This container is used to display the carousel and is used in the Javascript to insert other divs, information and formatting. After the carousel div there is a script tag for the local Javascript and for the external bootstrap resource.



```
        <!-- main container for carousel -->
        <div id="main">
            <!-- title "Events" on page -->
            <div class="events-title">
                Events
            </div>
            <!-- div for carousel -->
            <div id="carouselSlides" class="carousel slide carousel-fade" data-bs-ride="carousel" data-bs-pause="false">
                <div class="carousel-inner">
                </div>
            </div>

        </div>
```



```
    <script src="scripts/script.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz" crossorigin="anonymous">
    </script>
```