11060 – Assessment 2 Interactive Project

Jehru Harris/U3202988

Link to GitHub Repository and GitHub Pages

https://github.com/Jehru/11060-Assessment-2

https://jehru.github.io/11060-Assessment-2/

Project Proposal (Skip to page 3+ for Project Journal)

Project Title

Frogs of Australia – A Memory Game

Design Problem

For this project I wanted to further enhance my previous concept 'Frogs of Australia". The problem statement is that places like museums have gradually used interactive exhibits and interactive displays to increase engagement and enhance the visitor's experience. These exhibits can also be used in archives, art galleries, libraries and zoos.

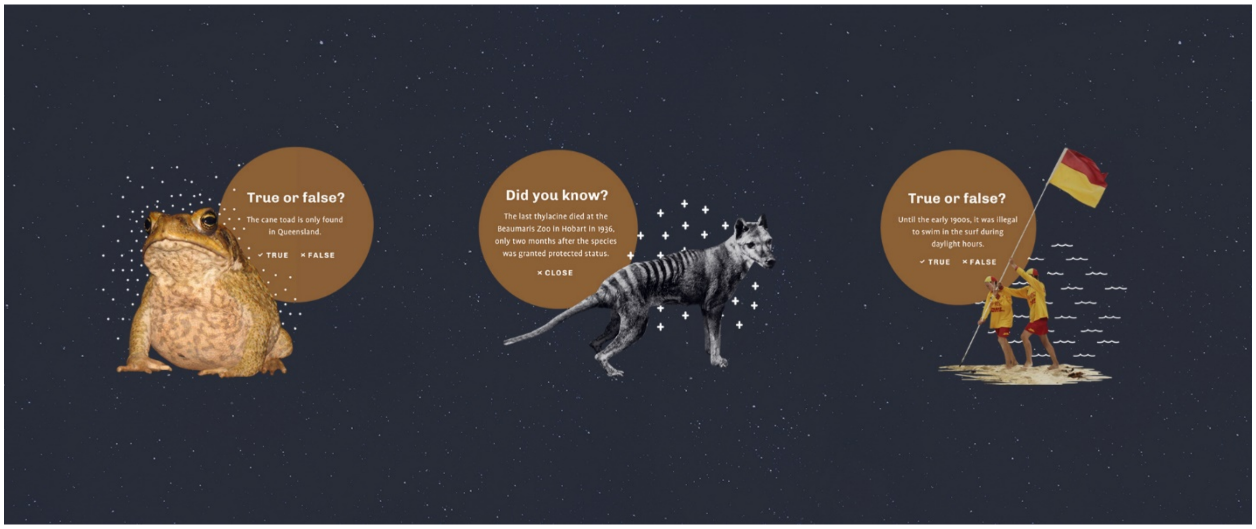
In cultural institutions this high-level of interactivity is something that visitors now have come expect. I wanted to make my web interface informative, innovative, playful and fun. By using a large format touchscreen, it allows for a unique touch-based input compared to a computer's mouse, it also allows for multiple people to engage with at once. (Pekarik, Button, Doering, Sharbaugh & Sutton, 2002)

Initial Ideas (How to resolve problem)

Using Application Programming Interface (API) data and interactive web coding languages such as JavaScript to make an engaging interface.

Some ideas include:

- Have a display of frogs or animals that allows the user to click to show more information.
- Visualise the data on frogs with bubbles.
- Memory game with animals' underneath tiles that link together.
- Guess the animal/frog game or guess specific information related to it.
- A set of true or false questions related to facts about an animal or frog.


True or false?

The cane toad is only found in Queensland.

Did you know?

The last thylacine died at the Beaumaris Zoo in Hobart in 1936, only two months after the species was granted protected status.

True or false?

Until the early 1900s, it was illegal to swim in the surf during daylight hours.

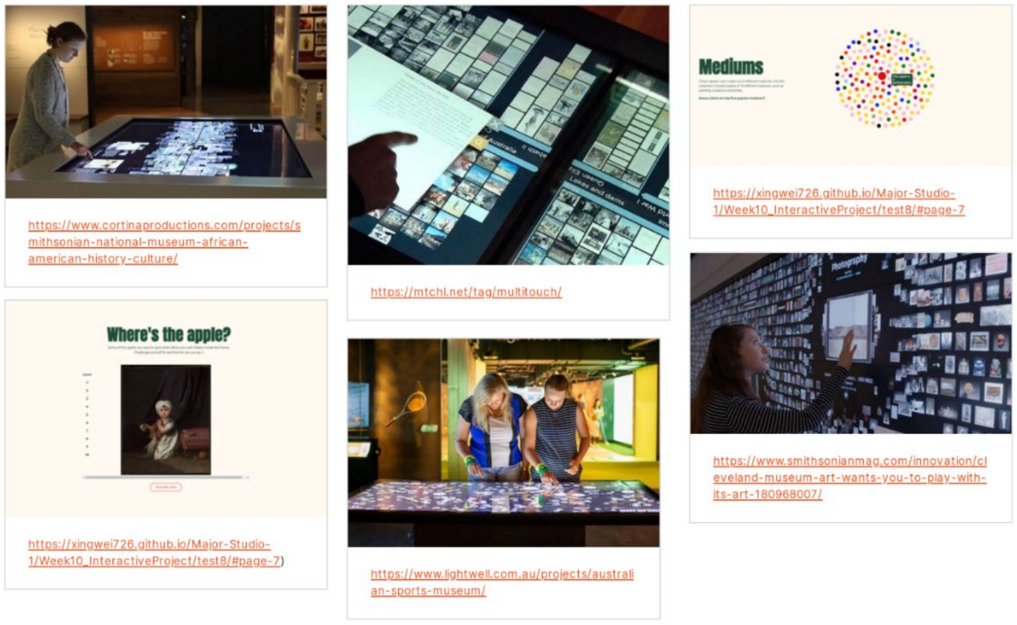
(*Image of Lightwell's Defining Moments display*, n.d.)

Consideration of context (Functions and scope)

I decided that the memory game would be an interesting concept that would work well as a gamified experience. A pop up/modal could appear if the user was successful in finding the pair, rewarding them with more information about their discovery.

The purpose of the website is to serve as a fun way to learn about frogs/animals.

Basic Sketches/Wireframes/Mood board



https://www.cortinaproductions.com/projects/smithsonian-national-museum-african-american-history-culture/

https://xingwei726.github.io/Major-Studio-1/Week10_InteractiveProject/test8/#page-7)

https://mtchl.net/tag/multitouch/

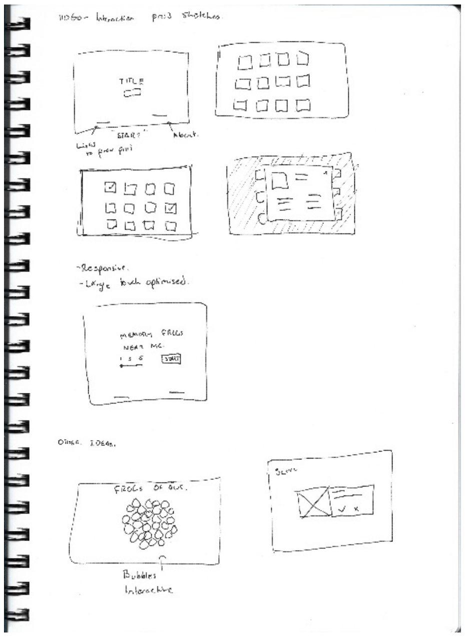
https://www.lightwell.com.au/projects/australian-sports-museum/

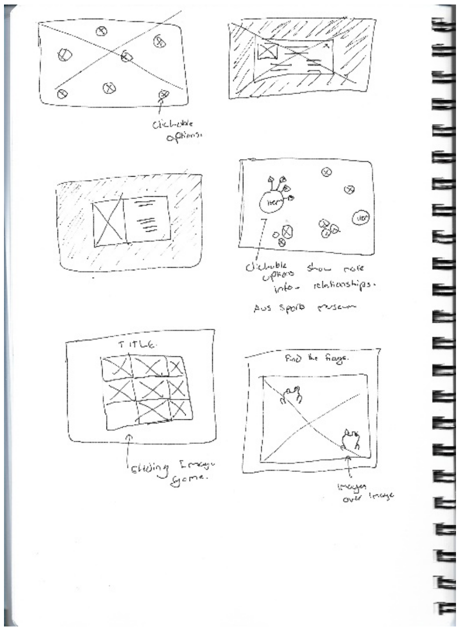
https://xingwei726.github.io/Major-Studio-1/Week10_InteractiveProject/test8/#page-7

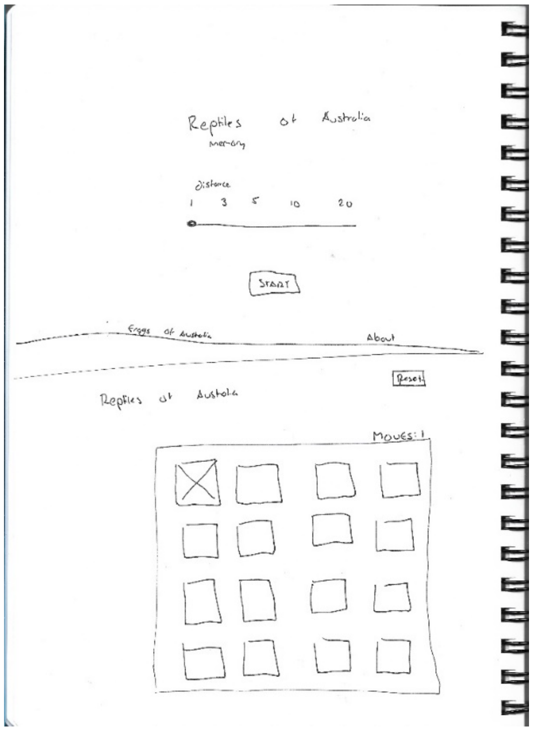
https://www.smithsonianmag.com/innovation/cleveland-museum-art-wants-you-to-play-with-its-art-180968007/

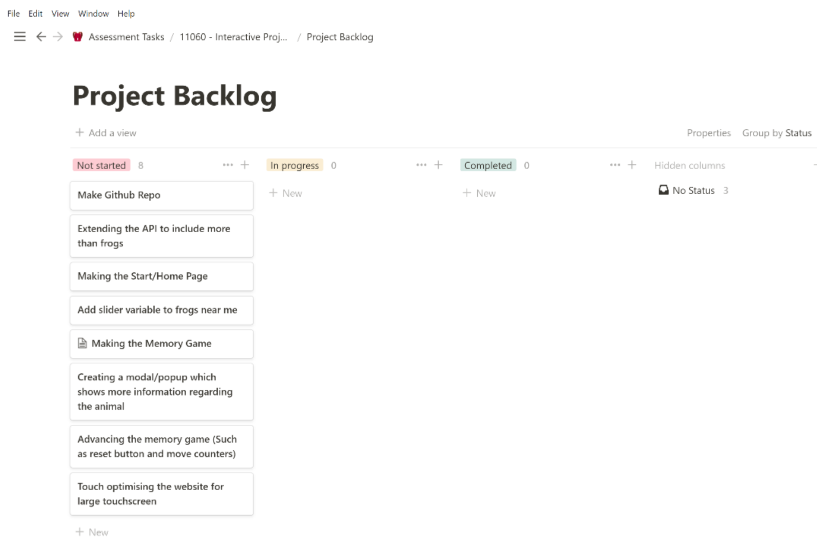
I found many of the projects by lightwell, a design company who makes engaging display and exhibitions for museums and public spaces to be inspirational to my project. ("Multimedia Projects — lightwell", n.d.)

Some Sketches








File Edit View Window Help

Assessment Tasks / 11060 - Interactive Proj... / Project Backlog

Project Backlog

+ Add a view   Properties   Group by Status

Not started 8 | In progress 0 | Completed 0 | Hidden columns

- Make Github Repo
- Extending the API to include more than frogs
- Making the Start/Home Page
- Add slider variable to frogs near me
- Making the Memory Game
- Creating a modal/popup which shows more information regarding the animal
- Advancing the memory game (Such as reset button and move counters)
- Touch optimising the website for large touchscreen

+ New

+ New

+ New

No Status 3

Project Journal

Introductory Research

Following the project proposal, I began my design process by researching touch screen exhibits and kiosks. Some of the research indicates that using interactive displays at museums has increased engagement and interest in non-interactive objects with visitors reporting to be more engaged when an exhibit provides an interactive experience. This higher level of engagement has led to museums adopting more interactive solutions. (Burmistrov, 2015)

Some of the best practices for interactive exhibits include:

- A simple interface.
- Should have an 'attract' screen that makes it clear that it is touchable.
- Clear navigation that provides clear links to guide the user.
- Use as little typing as possible.
- Comply with WCAG 2.1 for levels of colour contrast.

(Bradley & David, 2018)

The main interactive displays which have inspired this project include the Artlens Wall by the Cleveland Museum of Art, The Australian Sports Museum by Lightwell and many of the other projects by Lightwell. ("ArtLens Wall", n.d.) ("Australian Sports Museum — lightwell", n.d.)

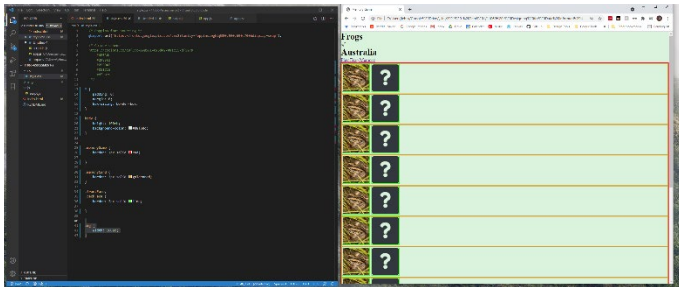
While researching and ideating how I could transfer my previous project to be fun and engaging I came across a giant frog memory game. This provided the inspiration to make my own frog memory based on the Atlas of Living Australia API. Some of the other ideas can be read in the project proposal above. (Horn & Shen, 2009)

Development Process

To begin this project, I set up the GitHub repository and I created three files for the HTML, CSS, and JavaScript. I created a development branch in GitHub and copied over the previous assessment.

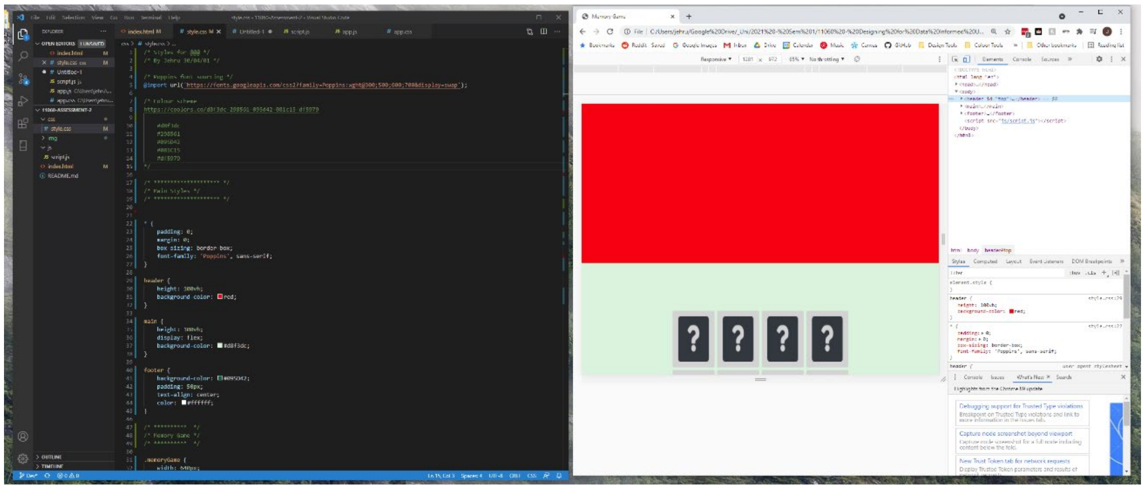
To create a card matching game, I researched and found a variety of JavaScript tutorials. I choose to follow a YouTube video tutorial as I found that the instructor explained why the code was implemented. (*Memory Card Game - JavaScript Tutorial*, 2018)

This tutorial was also chosen as it used images for the front and back cards, this would be useful later on when I would need to append an image from the API.



Reference of the question mark image used as the back face of the cards

(*Question Mark Icon*, n.d.)



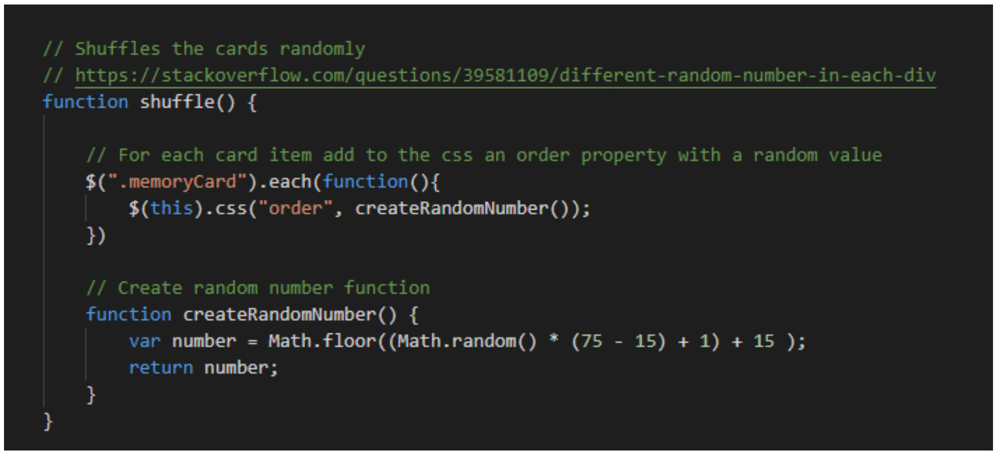
```
/* Styles for @@@ */
/* By Jehru 30/04/01 */

/* Poppins font sourcing */
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600;700&display=swap');

/* Colour scheme
https://coolors.co/d8f3dc-298561-095d42-081c15-df5979

    #d8f3dc
    #298561
    #095D42
    #081C15
    #df5979
*/

/* ******************* */
/* Main Styles */
/* ******************* */


* {
    padding: 0;
    margin: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}

header {
    height: 100vh;
    background-color: red;
}

main {
    height: 100vh;
    display: flex;
    background-color: #d8f3dc;
}

footer {
    background-color: #095D42;
    padding: 50px;
    text-align: center;
    color: #ffffff;
}

/* *********  */
/* Memory Game */
/* *********  */

.memoryGame {
    width: 640px;
```

I first changed the “Frogs of Australia” page so that it was took up the full screen as I wanted to give my website an introduction and title page as I was planning to have the game board take up the full screen.

Following the tutorial, I placed the game cards inside a flex grid. I have heard previously that a flex grid may not be the best implementation with responsive design however it was found that the flex grid worked well on 1080p – 2160p screens in a 16:9 aspect ratio, further testing on smaller screens may be required.

The tutorial I was following used pure JavaScript however I decided to use jQuery so, I could further develop my skills and write out the code more concisely. One of the challenges of using jQuery is that it required further problem solving to get it to work.

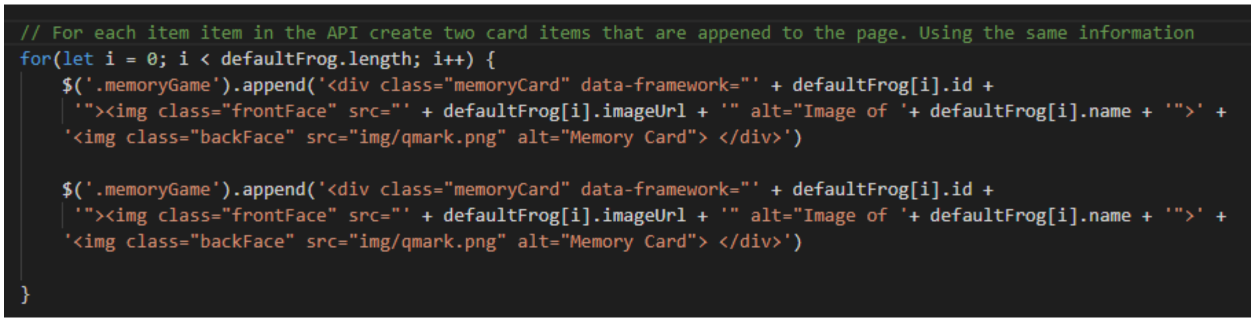
The first problem that I encountered was that the deck wouldn’t shuffle each time the page was loaded. The issue lay with the random number generator using the same random number for all the items appended to the page. This was because all the appended items had the same class name. I researched the ‘Math.random’ method which gives a random number and found a stack overflow article which provided a solution to random numbering for items with the same class name. The solution used jQuery's '.each' method and adjusted the CSS 'order:' property to give it a random number each time. I realised that using a separate function to create a random number would then provide a different random number each time it ran through the '.each' block of code. See the image below. ("Math.random() - JavaScript | MDN", 2021) ("different random number in each div", 2019)



```
// Shuffles the cards randomly
// https://stackoverflow.com/questions/39581109/different-random-number-in-each-div
function shuffle() {

    // For each card item add to the css an order property with a random value
    $(".memoryCard").each(function(){
        $(this).css("order", createRandomNumber());
    })

    // Create random number function
    function createRandomNumber() {
        var number = Math.floor((Math.random() * (75 - 15) + 1) + 15 );
        return number;
    }
}
```

I then added in the API, as previously I had been testing the JavaScript memory game with the content already on the HTML page and no API integration. The first problem that was encountered was that the for loop would only append one version of the card to the page, providing no pairs. This was solved by copying and pasting the append method in the for loop using the exact same information as the first card, however this is not likely the most effective method as its writing the same code out twice.

I tried the game with the location API but would show images with 'no images' and this would break picture matching experience, the code also was also much more complex. I decided that this could be implemented at the end as the memory game still was not fully developed and nothing would happen if a pair was matched.



```
// For each item item in the API create two card items that are appened to the page. Using the same information
for(let i = 0; i < defaultFrog.length; i++) {
    $('.memoryGame').append('<div class="memoryCard" data-framework="' + defaultFrog[i].id +
     '"><img class="frontFace" src="' + defaultFrog[i].imageUrl + '" alt="Image of '+ defaultFrog[i].name + '">' +
    '<img class="backFace" src="img/qmark.png" alt="Memory Card"> </div>')

    $('.memoryGame').append('<div class="memoryCard" data-framework="' + defaultFrog[i].id +
     '"><img class="frontFace" src="' + defaultFrog[i].imageUrl + '" alt="Image of '+ defaultFrog[i].name + '">' +
    '<img class="backFace" src="img/qmark.png" alt="Memory Card"> </div>')

}
```

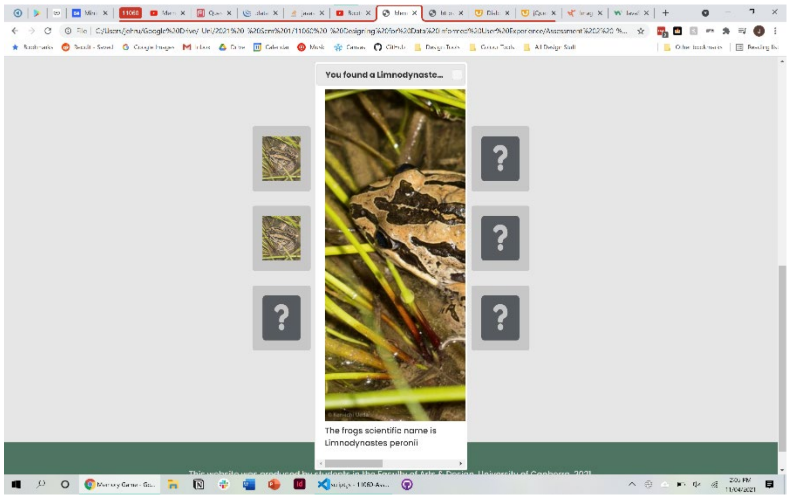
The next challenge that I encountered was that when two cards were selected they were no longer clickable and effectively removed from the game, this occurred when any two cards were selected not just a pair. The tutorial solved this by removing an event listener on the matched cards however, this wouldn't work for me as I didn't have any event listeners.

To solve this issue I added a class item called 'flip' when a card was clicked, if the cards match based on a given dataset.framework id code then they would turn off the ability to click from the game. This used jQuery's '.off' method. If the cards didn't match, then they would both remove the 'flip' class using jQuery's '.removeClass'. The flip class turns the cards 180 degrees showing the image side.

One of the main functions that I wanted to include in the webpage was when a user found a matched pair, it would provide a popup with more information regarding that frog species. I initially looked into the default JavaScript alert (window.alert) popup however it looks visually dated and is reminds me of a malicious websites telling you that you have installed a virus. I wanted to use a full screen popup, which darkens or blurs the background and presents the user with an image of the frog and some further information about the frog.

While I could have created my own modal, it would have been a lot of work to look visually engaging and to function responsively, so I searched for modal/popup libraries. I tried out a several solutions such as Modaal, jQuery UI's dialog and TingleJS. Modaal and jQuery UI either didn't work for me or didn't provide the look and functionality that I wanted. I found that TingleJS provided a consistent look with the rest of the website and worked well. ("Tingle.js, 2kB vanilla modal plugin", n.d.) (Rocheleau, 2020)

After figuring out how TingleJS works, I found that it worked flawlessly except on smaller screens. This will then query the Encyclopedia of Life (EOL) life API using the frog's name. I decided to use the EOL API as well as I felt that the single word response from the previous project wasn't as rewarding. For the memory game I wanted to provide a random fact or short description of the frog. Two API's that I found were the Encyclopedia of Life API and the Fish watch API. I decided to use the EOL API as I would have to change the content of the project to be about marine species if I was to use the FishWatch API.


You found a Limnodynaste...

The frogs scientific name is Limnodynastes peronii

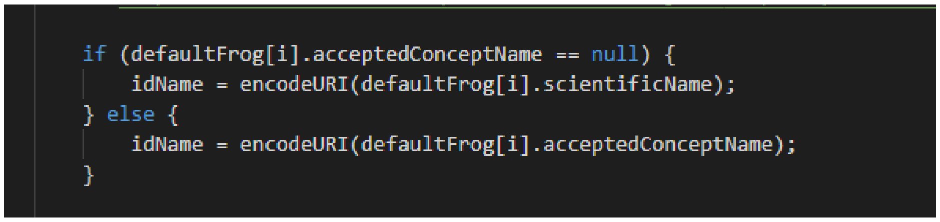
This website was produced by students in the Faculty of Arts & Design, University of Canberra 2021

("Alaska Snow Crab | FishWatch", n.d.) ("Encyclopedia of Life - Classic API's", n.d.)

The modal would have to read which card pair was matched and then present the information from the new EOL API in the modal.

To get the modal to popup with the matched cards I set up a for loop that would loop through each item in the API and match the card 'dataset.framework' id of the card flipped and all the id codes of the frogs in the API. This would match the frog that is flipped with the frog in the array of 6 frogs. I was then able to create a modal using the frog that was flipped.

I added code so that it reads either the ‘acceptedConceptName’ or the ‘scientificName’ as the ALA API wouldn’t always provide a frog name using the accepted concept name. This would result in nothing showing in the modal, so I checked if the value exists and used an alternative property. This then encodes the name of the frog to query the EOL API.



```
if (defaultFrog[i].acceptedConceptName == null) {
    idName = encodeURI(defaultFrog[i].scientificName);
} else {
    idName = encodeURI(defaultFrog[i].acceptedConceptName);
}
```

It was found later on that modals should be avoided in most cases and only used for errors or other important alerts (Fessenden, 2017). This made me realise why there are not many modal libraries.

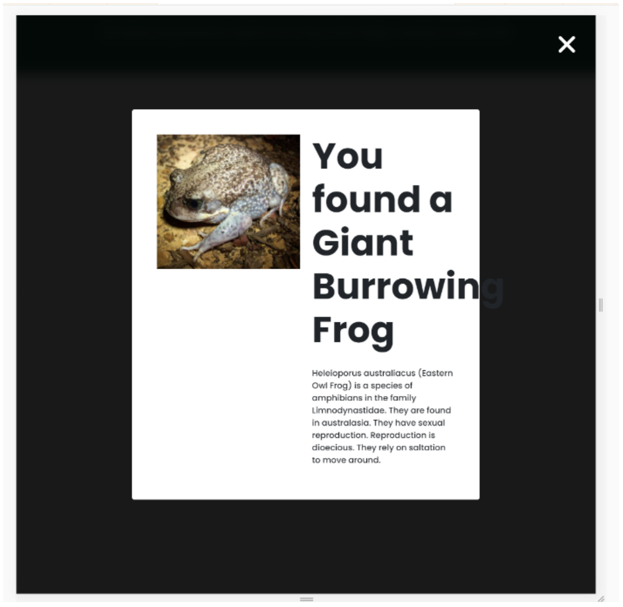
I asked the tutor and class for feedback in week 10. Some of the areas that needed improvement wasn’t able to function on narrower screens. The images became too small, and the cards would squeeze together, the titles were also too big.

I worked on this feedback adding in media queries and adjusting the sizing of elements. I also made the flex grid switch to three columns instead of four as four columns would distort the cards. I removed the padding on the smaller screens and made the titles smaller.

I had previous experience with bootstrap and it was utilised however it didn’t completely solve the problem.

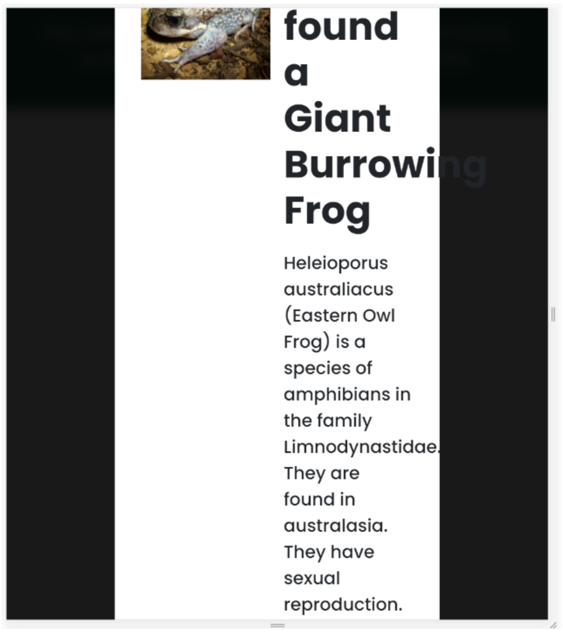
I initially used a ‘.col-8’ and ‘.col-4’ grid to make the image larger. I found that the main title text was always clipping over the edge of the modal. I changed the grid to be ‘.col-6’ for both components, this makes the image slightly smaller but fixed the problem.

At about 600px the text would clip again; I changed the width of both elements to change to span 100% of the modal width rather than the default 50%. This changed the modal so the text would appear under the image on smaller screens.



You found a Giant Burrowing Frog

Heleioporus australiacus (Eastern Owl Frog) is a species of amphibians in the family Limnodynastidae. They are found in australasia. They have sexual reproduction. Reproduction is dioecious. They rely on saltation to move around.



found a Giant Burrowing Frog

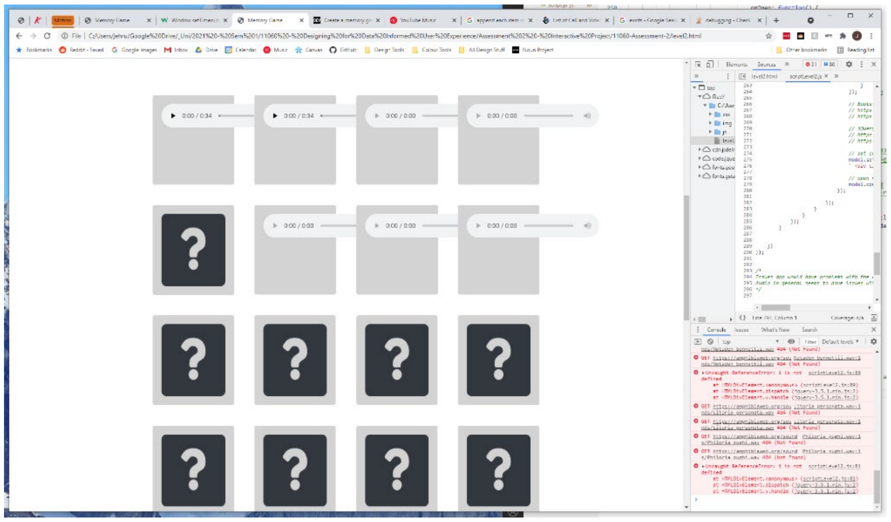
Heleioporus australiacus (Eastern Owl Frog) is a species of amphibians in the family Limnodynastidae. They are found in australasia. They have sexual reproduction.

To finish the project, I added some more gamification elements such as a timer and move counter. These elements create a sense of pressure for the player and give them a sense of accomplishment if they beat themselves. I then added a function so when the user completes all the cards pairs, they would be transferred to a new page, which would be level 2.

However, I never got around to developing level 2 due to the audio on click not working like the images.

I found this tutorial (Damilola, 2020) to be the best solution, but I found that it would always play lidonstes peronii's frog noise and not the frog that was clicked. This was later identified as the only frog that had an audio file in the group of 6.

I removed the level 2 audio game as I found that playing audio based on click was unreliable, some of the Amphibia Webs audio files have the name of the frog at the beginning and if the site didn't use HTML's audio controls, then I found it difficult for the user to control what was playing.



I decided to use an audio controls similar to the first assessment where if there is an audio file the user has the option to play it.

I tested the application on the large format screen at University, I found that some of the textual elements appeared too small on a 4k display. I then included a media query for very large displays about 2150px to 3900px. One issue was identified that if the user scrolls when clicking an item this 'breaks' the experience with the user stating, "what's gone on". I created a new class called hidden that would toggle "overflow:hidden" property to the body field when the player clicks "play frog memory" and a "Go back" link which will return them to the main page. Overflow hidden will lock the scrolling of the page when its applied to a html or body element (Po, 2018) ("Toggle CSS with jQuery", 2013)

I added a winner screen into the site based on the tutor's feedback, I attempted to get a new popup to appear after the user closes the last matched modal popup in the game. This however would not work, the other solutions I found while testing this was the JavaScript '.alert' which pops up the alert dialog and the 'window.location' which changes the page for the user. I used the window.location and created a 'winner' page that congratulates the user. I wanted the congratulations page to also show the users total moves and time they took; this wouldn't work when the page changes to 'winner' page.

I remember the tutor talking about session storage previously and upon further research I found that you could transfer a value across the html pages using the Session Storage API. I found a stackoverflow tutorial which helped to fix this problem this. ("Share variables between html pages", 2014)

The final solution captured the values via 'sessionStorage.setItem' and retrieve them using sessionStorage.getItem. Using this I was able to return the users previous results onto the winning screen. I changed the colour to light blue to make it clear that the user didn't refresh the page.

Final Reflections

Due to time limitations the audio matching level was not fully functional, and the modal took a while to load when the user had found a match. While developing the site I never noticed the loading time as I knew that the modal would popup however in my user testing, I found that the delay was too long, and it would interrupt people as they would continue to play after finding a pair. To further develop the website a loading screen could be implemented to let the user know that they are supposed to wait for the modal. The API could return frog results based on the user's location to provide more relevant information to them and frogs they may be able to find in their area.

References

Alaska Snow Crab | FishWatch. Retrieved 2 May 2021, from https://www.fishwatch.gov/profiles/alaska-snow-crab

ArtLens Wall. Retrieved 2 May 2021, from https://www.clevelandart.org/artlens-gallery/artlens-wall

Australian Sports Museum — lightwell. Retrieved 2 May 2021, from https://www.lightwell.com.au/projects/australian-sports-museum/

Bradley, C., & David, C. (2018). Best Practices for Interactive Exhibits - Gecko Group. Retrieved 8 May 2021, from https://geckogroup.com/2018/11/05/digital-and-mechanical-interactive-exhibits/

Burmistrov, I. (2015). *Touchscreen Kiosks in Museums* [Ebook]. InterUX. Retrieved from https://interux.com/publications/Burmistrov-Touchscreen_Kiosks_in_Museums-2015.pdf

Damilola, F. (2020). Create a memory game with JavaScript. Retrieved 2 May 2021, from https://dev.to/fakorededamilola/create-a-memory-game-with-js-1l9j#content-4

different random number in each div. (2019). Retrieved 2 May 2021, from https://stackoverflow.com/questions/39581109/different-random-number-in-each-div

Encyclopedia of Life - Classic API's. Retrieved 8 May 2021, from https://eol.org/docs/what-is-eol/classic-apis

Fessenden, T. (2017). Modal & Nonmodal Dialogs: When (& When Not) to Use Them. Retrieved 2 May 2021, from https://www.nngroup.com/articles/modal-nonmodal-dialog/

Horn, M., & Shen, C. (2009). *Frogs and Toads Memory: A Voronoi Twist on the Classic Children's Game* [Ebook]. Retrieved from https://www.researchgate.net/publication/220939567_Frogs_and_Toads_Memory_A_Voronoi_Twist_on_the_Classic_Children%27s_Game

Image of Lightwell's Defining Moments display. [Image]. Retrieved from https://www.lightwell.com.au/projects/defining-moments/

Math.random() - JavaScript | MDN. (2021). Retrieved 2 May 2021, from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

Memory Card Game - JavaScript Tutorial. (2018). [Video]. Retrieved from Encyclopedia of Life - Classic API's. Retrieved 8 May 2021, from https://eol.org/docs/what-is-eol/classic-apis

Multimedia Projects — lightwell. Retrieved 2 May 2021, from https://www.lightwell.com.au/projects

Pekarik, A., Button, K., Doering, Z., Sharbaugh, A., & Sutton, J. (2002). *Developing Interactive Exhibitions at the Smithsonian* [Ebook]. Smithsonian Institution. Retrieved from https://www.si.edu/Content/opanda/docs/Rpts2002/02.05.InteractiveExhibitions.Final.pdf

Po, W. (2018). Body scroll lock — making it work with everything. Retrieved 2 May 2021, from https://medium.com/jsdownunder/locking-body-scroll-for-all-devices-22def9615177

Question Mark Icon. [Image]. Retrieved from https://friconix.com/icon/fi-snsuxl-question-mark/

Rocheleau, J. (2020). 10 Free JavaScript Modal Window Plugins - Hongkiat. Retrieved 2 May 2021, from https://www.hongkiat.com/blog/best-free-javascript-modal-window-plugins/

Share variables between html pages. (2014). Retrieved 8 May 2021, from https://stackoverflow.com/questions/21128692/share-variables-between-html-pages

Tingle.js, 2kB vanilla modal plugin. Retrieved 2 May 2021, from https://tingle.robinparisi.com/

Toggle CSS with jQuery. (2013). Retrieved 2 May 2021, from https://stackoverflow.com/questions/18467143/toggle-css-with-jquery/18467206