

Project 1 Documentation: 2 Minutes with Emojis

A. Description of the project

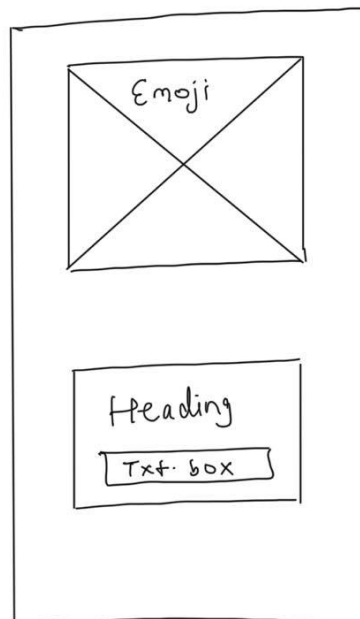
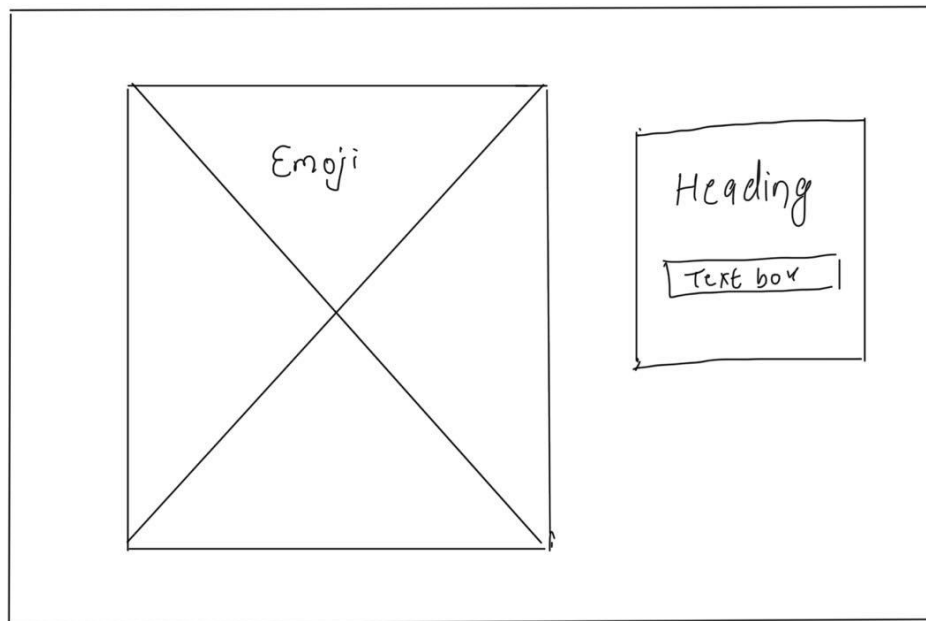
Project 2 Minutes with Emojis is submitted as the first project assignment for the course Connections Labs. It is a single user word guessing game that is written using HTML 5, CSS, and JavaScript. This project is meant to deliver entertainment for children as well as adults through the use of emojis in word guessing games along with a chance to enrich their knowledge of emojis. The game includes timer, hints, and a performance report, all of which are crucial aspects of the user experience to evoke a sense of excitement and connection.

B. Motivation/objectives

It is a universally acknowledged fact that the use of emojis has grown rapidly in our text messaging. The ubiquity of emojis gave me an idea to use emojis to connect to people, so I thought the best way to put emojis into use was to include it in a word guessing game. Although the word guessing game wasn't completely meant to guess the real name of emoji as often the emotions conveyed by emojis are more important than their names, so the words one has to guess are related to those emotions rather than the names itself. Doing this, I realized people can play these games in leisure time when they want to do a bit of critical thinking while also having fun. The last thing that's tangentially related to my project was my implicit goal of emoji literacy. I realized while building this project that despite the fact that we use emojis, our use is limited. Thus, through the game, one can explore what emojis exist and what emotions they convey through the word guessing game.

C. Wireframing the project

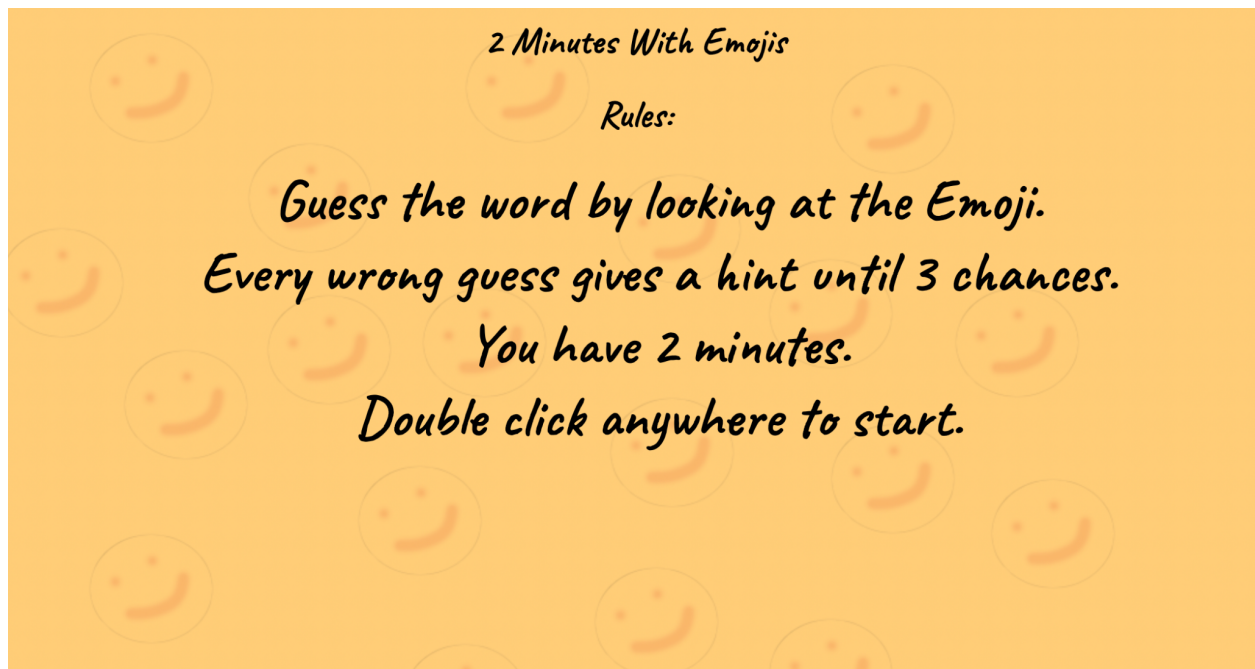
My first thoughts for the web page were menus, instruction tabs, buttons, and all, so my first wireframe idea was too convoluted. As I discovered more wireframing ideas, I came up with a minimal look yet elegant enough to hook the players into the game. Since the game was a simple word guessing game, I thought it would be useless to include unnecessary graphics and elements on the webpage that can interfere with the user experience and focus. Therefore, I have in my wireframe the least number of block elements needed for a robust and elegant structure of the web page. Below are my wireframes for mobile and desktop viewports.



These were my initial wireframes but as we will see the final project looks slightly different although the main structure still remains the same.

D. Writing the html page

The wireframe of the web page was utilized to write the index.html that includes all the div elements and input elements required for the web page. Writing html was not difficult because my blocks were already in place and my web page required a limited number of elements. I have tried my best to stick to the BEM convention for naming all my classes. The tags that I have used are h1, div, input, p, button, and section. Although the structure of the page looks effortless, I wanted users to just come to the site and without much hassle of navigation just go into the game immediately. However, one important modification I made as a part of the professor's focus on on-boarding of the game was I created a landing page that has clear and only needed information on what the game or page does. I have used four bullet points and asked users to double click to start. This means the users can start the game easily while also having the knowledge of what is expected of the web page. Below is my landing page.



E. Writing the style.css

Writing the CSS was an important part of the user experience and mainly included designing the landing page background, making the page responsive, resizing the elements and formatting the headings. My landing page has a background of emojis with little opacity that gives the users idea that the game has something to do with emojis and provides a feeling of comfort as they start the game. This background was created in Adobe Photoshop. I have used the google font Caveat throughout the web page to create a more personal tone. I felt this font suited the overall sense meant to be conveyed by the web page i.e. slightly informal, more connecting (it looks more like someone's handwriting than the perfect computer-produced text).

In the main game page, a light gray background is used so that text and input box as well as emojis are conspicuous. The div elements in the desktop version are arranged using flexbox with the flex direction as row that changes to column when viewed in mobile devices. Here, I have attached the images of the web page before and after applying CSS.

Before CSS:


2 Minutes With Emojis

Rules:

- Guess the word.
- Every wrong guess gives a hint until 3 chances.
- You have 2 minutes.
- Check your performance at the end.
- Scroll down to start.

2 Minutes With EMOJIs

After CSS:




*2 Minutes With
EMOJIs*

Guess word 0:58

✖

*It's a 3 letter word.
The word starts with c.*



*2 Minutes
With
EMOJIs*

Guess word 1:26

F. Working on the functionality of the web page/ writing app.js

I always wondered how my friends built dictionaries and other websites: where did they get so much data from? It was enlightening to know that the use of APIs makes that possible. I have used a json file (<https://raw.githubusercontent.com/amurani/unicode-emoji-list/master/full-emoji-list.json>) to display the emojis on the web page. This json file contains emojis along with the unicodes and keywords associated with them. The game basically works this way:

- a. An emoji is fetched from the json file.
- b. The user has to guess the word associated with the emoji. The timer countdown starts and continues for 2 minutes.
- c. After every 1 wrong guess, a hint is displayed to assist the user. With 3 wrong guesses, the emoji disappears and is appended to the learning list. Next emoji appears. This process repeats until 2 minutes are over.
- d. Once 2 minutes have passed, the results are displayed: score is shown, emojis that user got wrong are displayed with their unicodes, description, and keyword associated with it. This helps the user to learn these emojis.
- e. A restart button appears once the game is over to play the game from the beginning.
- f. An audio is played in a loop, and every time the user gets an answer correct, he/she hears a beep sound.

The functions that have been created in the javascript file work for following purposes:

- a. `getRandomInt(min,max)` : returns a random integer between the min and max values (including min but excluding max)
- b. `updateCounter()` : controls the timer and displays the countdown
- c. `async function changeEmoji(counter)` : updates the emoji on the page
- d. `sleep (ms)` : accepts ms as a parameter for the `setTimeout` to wait ms milliseconds before the promise is resolved
- e. `reset ()` : restarts the game from beginning

G. Snippets of code for some important functions

- a. **Sleep** function to create a promise and using setTimeout to resolve it after ms seconds

```
function sleep(ms)
{
    return new Promise((accept)=>{
        setTimeout(()=>{
            accept();
        },ms);
    });
}
```

- b. **Async/await changeEmoji** function to update emoji

```
async function changeEmoji(randomNumber)
{
    if (mistakes==0){
        await sleep(300); //immediately display the first emoji
    }
    else{
        await sleep(1500); //wait for 1500ms before another emoji is loaded
    }
    emojiContainer.innerHTML=emojiColl[randomNumber].emoji; //loads
emoji on the page
    mistakes=0; //mistakes set to 0 for a new emoji
    hint.innerHTML="";
    rightWrongSymbol.innerHTML="";
}
```

H. Debugging and revisions of code

I am not a creative person, so I struggled with the idea for the project. First, I decided to build a dictionary by using a dictionary api. I had almost completed the structure and css for the page (I have attached a screenshot of that work too), but I was not satisfied with the idea as I felt the project would be too boring. Since the course is about connections, I thought a dictionary, although interactive, isn't connecting many people. Meanwhile, I thought of a word guessing game that was fun and more appealing to the users. I still don't know if this aligns well with the definition of connections, but I have made it more animated and interactive from before. For this particular code, I had to do a bit of work coding the javascript file, so it was a lot of debugging. Working in the console was helpful to see what went wrong at which step. I have addressed almost all of the feedback both from my professor and peers, which I have described below.

The code has been revised many times, each time including the feedback received from peers from the class and from the users outside the class. The initial version of the game didn't have any hints for users, which was pointed out to be unnecessarily challenging that repels users. Following feedback was considered in the final version of the game.

Feedback from peers:

1. I have included a timer in the game at the recommendation of my peers, who thought that time limit could make the game more interesting.
2. I have also included hints that can assist users to guess the right word for the emoji.
3. I have also addressed what will happen if the user gets the word correct or what will happen after the game ends. The game ends with a chart that shows the performance of the user and helps the user to learn the emojis that he/she got incorrect.
4. I have also used the audio for correct and wrong answers along with the interface and game over audio considering the feedback that sounds can make the game more interesting.

H. Self-evaluation of the final product

I have self-evaluated my final product, the webpage by talking about some of the strengths and weaknesses from my perspective.

1. Strengths

- a. It is simple, easy to understand and use.
- b. It can be addictive if one is fond of emojis, so it's appealing to users.
- c. It is accessible in terms of color combinations and usability.

2. Weaknesses

- a. The webpage is plain without any animations other than typing the right word, so it might become dull after frequent use. However, the randomized data set of emojis means the users will guess new emojis.
- b. I am not sure if this is a weakness, but the web page has a limited user interface (only few elements and no navigation tools). I will continue working on that and increasing the user interactions.

I. Challenges/difficulties

1. Since I am new to web development and javascript, I had some struggle with controlling the sequence of events in the game. For that, I had to learn await/async and understand the role of promises.
2. The dataset I had included was a little cluttered with unnecessary details as this was a json file I got from a github account, so I did a bit of data cleaning to make sure that the right words are chosen for each emojis that match the expression of emojis.

J. Next Steps

1. Currently, the game is a single-user game. Once we learn how to work the back end, I am planning to incorporate multi-users functionality so that everyone can join a room through a link and the one who gets the word correct first will receive points.
2. I am also planning on keeping the leaderboard functionality so that top high scores are displayed.

3. In response to the feedback I received during my presentation, I am planning to change the background music so that it matches the timer of the game.
4. Meanwhile, the game currently has some emojis for which correct words are far from the meaning they convey, so I am looking forward to more data cleaning.

K. Major takeaways from the process

1. I have learned how to design a responsive webpage using HTML 5 and CSS.
2. I have learned DOM manipulation and working with the console for debugging.
3. I have learned how to handle event listeners and how to incorporate event-driven interactions in the web page.
4. I have learned what a json file is and how to fetch data from it into the web page and manipulate that data in javascript.
5. I have come to know the importance of APIs in web development.

L. Credits

1. Professor Mathura Govindarajan
2. Github account (Kevin Murani) for json file
3. Stack Overflow
4. Youtube channel Toby Ho for learning await/async
5. My friend Bishnu for user testing the game
6. My peers from the class

M. Link

https://dt1930.github.io/connections_labs2022/Project1/index.html