Words: 1002

MEMORY GAME REPORT

1. Overview

1.1 Description of achievements

This project has realized a single matching memory card game. The game allows the player to click two cards at a time, and if the pattern is the same, the cards pair up and disappear, thus giving the player one point. Otherwise, the card is upside down and the player gets no points. When all players are paired, the game is played and the next round begins. During the game, if you want to quit the game, you can click the "Give Up" button to go back to the home page and restart the game.

The card pattern in the game is obtained by calling the DOG API. Players can play different games by choosing different dog breeds and different difficulty levels. The game has three levels of difficulty: Easy, Normal and Hard. In "Easy" mode, players need to match 8 cards, in "Normal" mode, players need to match 12 cards, and in "Hard" mode, players need to match 18 cards.

1.2 List of which requirements have been met

1. Random images can be fetched from the Dog API.

2. Randomly retrieves a different number of images from the API depending on the level selected by the user.

3. Allows the player to randomly click on "two cards" and display the image associated with each box.

4. Enables the cards to match when the second of the two boxes is opened. If the cards are the same, the selected image disappears. Otherwise, the cards are reversed back.

5. Allows the player to 'open' pairs of boxes repeatedly until all matches are found.

6. Allows the player to keep track of how many pairs of boxes they have opened during the game.

7. Enables a new round to be played by clicking on the 'start' button.

8. The "quit" button is clicked to abandon the game and return to the interface.

9. The home page of this game lists all the available dog breeds and the player can select the breed they want to start the game.

10. Paging of the acquired breeds. Players can only see 10 breeds at a time. Allows the player to load the previous and following breed pages.

11. Set the game timer

12. Implement the timer function

13. Implement the function of storing user data

2. Technologies & Resources

In order to beautify the game interface, increase the user experience. The background of the dynamic of the game using the “Animated Background-13” dynamic background CSS style in this website(https://csspoint101.com/30-css-animated-background/).

The layout of the game, including the "record panel" and the main page, is based on the GitHub blogger Kaylee4m. Referenced blog posts: https://github.com/kaylee4m

The logic of card game design is based on the blogger Kubosania. Referenced blog posts: https://github.com/kubowania/memory-game .

I referred to this blog post when writing the timer feature：

https://www.freecodecamp.org/news/javascript-timers-everything-you-need-to-know-5f31eaa37162/

I used the Google browser for testing.

3. Design

3.1 Selection of Breeds

Because the game needs to be retrieved through the fetch API, I use Ajax to send a request to get the image. Get information on all varieties.

To traverse all varieties, I defined a hash map.

In order to realize the function of turning pages and displaying 10 varieties per page, I defined two functions to accomplish this function by implementing the quantitative relationship algorithm between the number of pages and the total number of varieties.

3.2 Main Game

Call the hidden and show methods. When clicking the "Start" button, hide the home page, and when clicking the "Give up" button, return to the home page.

3.2.1 Image to load

Use "Ajax" to call the Dog API and get the dog graph randomly. Define the function "AjaxGetFun" to achieve the function of getting pictures and picture information. Since the easy, medium and hard modes in the game require 4, 6, and 8, respectively, but the API calls only get one dog graph at a time, the function is designed with the parameter "times". The numbers 4, 6,

and 8 are passed in as parameters to the function. Call the function three times to realize the loading of the game pictures under the mode.

3.2.2. Game Section

The first step is to design a function that implements the pairing. First, the function "CreateBoard" creates the box. Assign the corresponding image properties to the box, and use "addEventListener" to listen for the function of "flip". Then, the design function "checkForMath" implements the pairing. By comparing the ID of the picture that is clicked by the user, judge whether it matches or not. If it matches, fill the white picture. If it doesn't match, fill in the box image.

The second step is to create the “Cardarray” array to store the game pictures, and call the CreateBoard function to load the box for the game.

The third step is to design the function. "Quit" to realize the function that users can return to the main interface at any time.

4. Evaluation

In this project, the implementation of the basic game features went well. At the same time, there are some difficulties. First, I initially used Ajax for asynchronous transfer, which caused the image to not load correctly. After many attempts, I chose to change "true" to "false". This change enables Ajax to synchronize transfers and load images correctly. Second, I was having a hard time using JavaScript native code to track users' game logs and timers, so I searched online for code to implement similar features. Third, when the selection of variety function and game function merge, there is a phenomenon that the game picture can not be eliminated. After debugging, I found that the 'img' attribute clashed in the two functions, so I created the class name 'Cardimg' to avoid the phenomenon of conflict by calling the class name. If I had more time, I would have implemented all the functionality in JavaScript native code. And realize the function of clicking on the picture to view the information of the dog,