Cat Meme Emotion Picker:

Use of the for of loops and how to nest for loops inside of each other.

//old way using nested regular for loops…………… // function getEmotionsArray(cats){

//     const emotionsArray = []

//     for (let i = 0; i < cats.length; i++){

//         for (let j=0; j < cats[i].emotionTags.length; j++){

//                 emotionsArray.push(cats[i].emotionTags[j])

//         }

//     }

Challenge:

1. Set up a "for of" in getEmotionsArray to iterate

   over the data.

2. For now, just log out each cat object individually.

\*/

Const catsData = [

{

emotionTags: ["hungry"],

isGif: true,

image: "hungry.gif",

alt: "A cat looking hungry",

},

{

emotionTags: ["hungry"],

isGif: true,

image: "hungry2.gif",

alt: "A cat looking hungry",

},

] //end of catsData array of objects

The general syntax of the for loop:

for (let cat of cats){

console.log(cat)

}

Ex. Let’s say the cats is an actual array of objects from above and we want to access the emotionTags property (which happens to be an array)…we would need to do cat.emotionTags to access the object properties.

  for (let cat of cats) {

    for (let emotion of cat.emotionTags) {

      console.log(emotion);

    }

  }

}

The first for loop will iterate in the first cat object from the cats array

2. then it will access inside the object the property called emotionTags by doing cat.emotionTags..this is saying lets iterate the emotionTags array from each cat

So part 1 we have:

const catEmotionsArray = [];

function getEmotionsArray(cats) {

  for (let cat of cats) {

    //console.log(cat.alt);

    for (let emotion of cat.emotionTags) {

      //push each cat emotion to the catEmotions array

      catEmotionsArray.push(emotion);

    }

  }

  return catEmotionsArray;

} //end getEmotionsArray function

This is accessing the catsData array….which is a iterating out the catEmotios property which is an array

and it returns an array called catEmotionsArray whenever it’s called.

Part2: we have another function called renderEmotionsRadios

This is going to take in a single parameter called cats….

And inside the function we are setting a variable to hold the contends of the getEmotionsArray which will take in the cat parameter…..the cat parameter is actually the catsData object being passed in to the function renderEmotionsRadios which will then be passed inside the getEmotionsArray

function renderEmotionsRadios(cats) {

  const emotions = getEmotionsArray(cats);

  console.log(emotions);

}

renderEmotionsRadios(catsData);

----------------------------------------------------------------------------------------------------------------------------------------

part 3..now we are going to render out the cat emotions in seperate <p></p>

..first we are going to gain access to the emotions-radios div tag so we can plainly render it via the html.

Then we are going to set a let to hold an empty string so we can store the cat emotions in one entire string.

Next…we will iterate over the emotions variable which holds the emotions properties of the catsData array and put each emotion in a <p></p> and inside template literals so we can use easier access to the variable so `<p>${i}</p>`

And we we will add it to the catEmotions empty string

Finally outside of the for of loop we will do an innerHTML and store it back in html of emotionsRadios

const emotionsRadios = document.getElementById("emotion-radios");

function renderEmotionsRadios(cats) {

  const emotions = getEmotionsArray(cats);

  let catEmotions = "";

  for (let i of emotions) {

    catEmotions += `<p>${i}</p>`;

  }

  emotionsRadios.innerHTML = catEmotions;

}

//pass the catsData array to the renderEmotionsRadios

renderEmotionsRadios(catsData);

part 4 we are going to translate out the catsData array into it’s own separate file so we will be using imports and exports…more separations of concerns

so 1. In the file where we have the data source stored we need to export the data as this is the data that will be exported to other files.

export const dinnerPartyGuests = [

'Elvis Presley',

'The Queen of England',

'Alan Turing',

'Nelson Mandela',

'Mahatma Gandhi',

'Aristotle',

'Albert Einstein'

]

2. Next we will need to import the data into the js file with the name of the ie: array name dinnerPartyGuests….

Import dinnerPartyGuests ………and inside curly braces……….import { dinnerPartyGuests }

We also need to tell js where it’s coming from

import { dinnerPartyGuests } from ‘./data.js’

3. Now inside the html file we also need to identify that it’s a module by placing type=”module” inside the script file

<script src="index.js" type="module"></script>

Part 5 below-----------------------------------------------------------------------------------------------------------------

Part 5 lets render the cat emotions inside a label form with radio buttons

First part we removed the <p></p> tags because we age going to dynamically our for loop ${i} that renders out the cat emotions

We need to add template literal strings `` …….inside these we will start with a label….<label for=”${i}”>${i}</label>……what is going on here…well after the label tag we have a for=…the for is for focus..if the user clicks the name and not the radio button the actual button wll be selected as for…focus….after the closing label tag <label> we dynamically add the ${i}…this will print out the name of the emotion inside a label …..and then we close off the label tag </label>…

2. Still inside the template literal string we next start to setup our input tags for the radio buttons…..<input….we add the usual type=”radio” which tell the browser that we want radio input buttons….after that we need to add and id tag which will identify with the id=”${i}” as this will be dynamic…next we add a value tag…value=”${i}”…then after we give it a name for the entire radio input name=”emotions”

3. We need to setup some styling so inside the beginning template literal string ` we add a div tag that has class radio from our css and right before the closing template literal ` we add the closing </div> tag.

function renderEmotionsRadios(cats) {

  const emotions = getEmotionsArray(cats);

  let catEmotions = "";

  for (let i of emotions) {

    // catEmotions += `<p>${i}</p>`;

    catEmotions += `

                    <div class="radio">

                    <label for="${i}">${i}</label>

                    <input

                    type="radio"

                    id="${i}"

                    value="${i}"

                    name="emotions"

                    >

                    </div>

                    `;

  }

///so the takeaway is that we dynamically render out the label name, the focus with for=, the id tag, the value tag….they are all dynamically rendered out with the for loop.

Example using the shopping list…using the .includes() method to check if an array has a given value

What is going on here…

1. The addItemBtn is attached to the Add button…which we created an eventListener….
2. We take in the input value with itemInput.value and we check it against the shoppingList array…we want to make sure that there are not any duplicates in the array so we will don’t have any repeats…
3. If the input.value is a duplicate we just console log out “no duplicates”
4. If it’s not a duplicate then the else statement continues….the itemInput.value gets pushed to the shoppingList array…
5. Next the render() function will be called.
6. Also note that the itemInput.value input field will be cleared with itemInput.value= “” to an empty string…….note that this also clears the input field it a duplicate is detected as the console.log entry will just print out “no duplicated” and nothing will be pushed to the shoppingList array….
7. Now when the render() function is called it iterates through the shoppingList array and also note that a variable called html=”” is created above.
8. After each iteration we dynamically create an <li> that is attached to the <ul></ul> with and id of list-item…the list items are stored inside the html variable:
9. html += `<li class="list-item">${item}</li>`;
10. Then after the html variable with the list item is rendered to the ul with list.innerHTML = html;

const shoppingList = [];

addItemBtn.addEventListener("click", function () {

 if (shoppingList.includes(itemInput.value)) {

    //itemInput.value = "";

    console.log("no duplicates");

  } else {

    shoppingList.push(itemInput.value);

    render();

  }

  //this clears the input field for the user see an empty input field in the item input box.

  itemInput.value = "";

}); //end of addItemBtn eventListener

function render() {

  let html = "";

  for (let item of shoppingList) {

    html += `<li class="list-item">${item}</li>`;

  }

  list.innerHTML = html;

}

render();

USE OF classList, getElementsByClassName…

Note we use classList when we want to target more that one class names at once (when they all share the same class name).

We can use **getElementsByClassName …**note the property is pleural….

Color panel example

    <h1>Moody<span class="accent">Blues</span></h1>

    <div class="container" id="container">

        <div class="palette-section blue-1"></div>

        <div class="palette-section blue-2"></div>

        <div class="palette-section blue-3"></div>

        <div class="palette-section blue-4"></div>

        <div class="palette-section blue-5"></div>

    </div>

    <div class="btn-controls">

        <button id="sort-btn">Sort by Shade</button>

    </div>

const sortBtn = document.getElementById("sort-btn");

const container = document.getElementById("container");

sortBtn.addEventListener("click", function () {

  container.classList.toggle("reverse");

});

Cat meme example:

  const radioArray = document.getElementsByClassName("radio");

  for (let radio of radioArray) {

    radio.classList.remove("highlight");

  }

Document.querySelector

So lets say you have a list of radio buttons and you don’t know all of the class or id’s. You can easily find which radio button has been checked using querySelector

Inside the addEventListener when the submitBtn has been clicked…(1) we are going to query the input field (2). The type will be for radio. (3) It has to be the checked option is what we are searching for

Next we can console out the checkedRadio and the value

submitBtn.addEventListener("click", function () {

  const checkedRadios = document.querySelector('input[type="radio"]:checked');

  console.log(checkedRadios);

  console.log(checkedRadios.value);

});

Note that the line:

document.querySelector("input[type='radio']:checked"

will tell us if a radio button has been checked

………………………we can further this to make sure that we don’t get any console errors in case we do not select a radio button **BUT**  we click the submit button by adding an if statement…the if statement will tell the code to run if a radio button has been selected

if (document.querySelector("input[type='radio']:checked")) {

    const checkedRadio = document.querySelector(

      "input[type='radio']:checked"

    ).value;

    console.log(checkedRadio);

  }

So the next challenge was to filter out from our data of cats the selected cats when we clicked on a radio button that had the selected emotion.

1. We need to use .filter and .includes methods
2. We need to setup a const variable to store the matching cats when they are filtered out and then we first add the catsData array then filter because we are filtering the catsData array then inside the filter function we setup a function with a parameter of emotion..this will then return the emotion that is searching through each cat then filter through the emotionTags and see if it includes the selected emotion

 const catsMatchingData = catsData.filter(function (emotion) {

      return emotion.emotionTags.includes(checkedRadio);

    });

Notice here we have a checkedRadio parameter…where did this come from?

From Here:

if (document.querySelector("input[type='radio']:checked")) {

    const checkedRadio = document.querySelector(

      "input[type='radio']:checked"

    ).value;

Note that the checkedRadio is being querySelector from the input with the type of radio and if it is checked (radio buttons are also marked as checked) and we need to put .value so the value of the emotion is stored not the entire object

Note that the if statement helps us prevent any errors if the submit/get image button is selected w/o any radiobuttons selected no error will be displayed it will just be ignored until a radiobutton has been selected.

Now we need to filter both the selected cat emotion and if the check box with Gifs only..So basically if the user selects a cat emotion and if they want selections with animated Gifs only…if the user only selects a cat emotion and leaves the Animated GIFS only empty then it will return the radio button selection…Note we’ve already done the major work by selecting the cat emotion and the isGif checked…

Also note that when using the filter method we have to return the data back…here we are returning the catsMatchingData variable;

const catsMatchingData = catsData.filter(function (emotion) {

      if (isGif) {

        return emotion.emotionTags.includes(checkedRadio) && emotion.isGif;

      } else {

        return emotion.emotionTags.includes(checkedRadio);

      }

    });

return catsMatchingData;

  } // end if statement

} //end getMatchingCatsArray

Here we have the array catsMatchingData returned from the filter method. 1. We are taking the catsData array then we are using the filter method…we are passing in the function with emotion as a parameter. 2. We need to use an if statement saying if (isGif ) meaning if the isGif has been checked then inside the curly braces return the cat with the selected radio button emotion **and**  if that cat has a GIF animated picture

1. If the user does not select the Gif then return just cats with selected emotions.
2. And very importantly we must return the catsMatchingData variable back

Now we are going to return a single cat object whether there is just one cat or if there is more than one then we are going to generate a random number with the length of the catsArray then we will use that number to select it from the catsArray

function getSingleCatObject() {

  const catsArray = getMatchingCatsArray();

  if (catsArray.length === 1) {

    console.log(catsArray[0]);

  } else {

    //  console.log(catsArray[Math.floor(Math.random()) \* catsArray.length]);

    const randomNumber = Math.floor(Math.random() \* catsArray.length);

    console.log(catsArray[randomNumber]);

  }

Here we have in the first part of the if statement the catsArray.length === 1{

Console.log(catsArray[0]

}

This is just actually selecting the cat object itself with catsArray[0] and not the array

THE SECOND PART OF THE IF STATEMENT THE ELSE CLAUSE

We have first 1. We generated a random number based on the index of the catsArray length

Then 2. We took that number and selected out a random cat in the catsArray…..

-------------------------------------------------------------------------------------------------------------------------------------

Next tasks: inside the renderCat() function

  /\*

Challenge:

1. Take the object that is returned by

   getSingleCatObject and save it to a const

   called "catObject".

2. Set memeModalInner’s innerHTML to the HTML

   string below, remembering to insert the relevant

   data from catObject to replace the UPPERCASE text.

3. Set memeModal’s display property to "flex".

       `<img

        class="cat-img"

        src="./images/CAT IMAGE"

        alt="CAT ALT TEXT"

        >`

\*/

function renderCat() {

  const catObject = getSingleCatObject();

  memeModalInner.innerHTML = `<img

        class="cat-img"

        src="./images/${catObject.image}"

        alt=${catObject.image}

        >`;

  memeModal.style.display = "flex";

Here we first saved the getSingleCatObject() to a const variable catObject

2. We set memeModalInner to the innerHTML to the string with the catObject inside template literals

NOTE that we are accessing the data.js array which has the object and we want to access the image so we need to do catObject.image….look below at where image is a key/value so catObject.image

  {

    emotionTags: ["moody"],

    isGif: false,

    image: "angry.jpeg",

    alt: "A cat looking moody",

  },

//Wire up the modal close button and set modal display to none

//Note that we didn't use the other addEventListener with function inside because that sets an anonyomous function w/o a name...this way we set our event listener with a name called closeModal but w/o the () parentheses and set a function below with the logic and when the close button is clicked the eventListener fires and the function runs

memeModalCloseBtn.addEventListener("click", closeModal);

function closeModal() {

  memeModal.style.display = "none";

}