UDACITY

PROJECT

Memory Game

A part of the Front-End Web Developer Nanodegree Program

PROJECT REVIEW

CODE REVIEW 6

NOTES

js/app.js 4

```
// variables
const img = ['fa-anchor',
               'fa-bicycle',
               'fa-diamond',
               'fa-leaf',
               'fa-bomb',
               'fa-bolt',
               'fa-paper-plane-o',
               'fa-cube'];
let cards = [];
let globalTimer = null;
const timerDiv = document.querySelector('.timer');
const playAgainButton = document.querySelector('.play-again');
const starsDiv = document.querySelector('.score-panel .stars');
const movesDiv = document.querySelector('.moves');
const resetButton = document.querySelector('.restart')
const scorePanel = document.querySelector('.score-panel');
const deck = document.querySelector('.deck');
const deckList = document.querySelectorAll('.deck');
const fragment = document.createDocumentFragment();
let state = {};

```

AWESOME

Good job using es6 variables consistently and using camelCase for your variable names! 👍

```
const startTimer = () => {
  globalTimer = setInterval(function () {
    state.time = state.time + 1;
    state.time === 3600 ? state = {...state, time: 0, hours: state.hours + 1}
    let hours = state.hours < 10 ? `0${state.hours}`: hours;
    let minutes = parseInt(state.time / 60);
    let seconds = parseInt(state.time % 60);
    hours >= 1 ? hours = `${hours}:` : hours = ''
```

AWESOME

Awesome job using ternary operators instead of if conditions! 👏

```
    minutes = minutes < 10 ? `0${minutes}` : minutes;
    seconds = seconds < 10 ? `0${seconds}` : seconds;
    timerDiv.textContent =  `${hours}${minutes}:${seconds}`;
  }, 1000);
}

// Shuffle function from http://stackoverflow.com/a/2450976
function shuffle(array) {
    var currentIndex = array.length, temporaryValue, randomIndex;

    while (currentIndex !== 0) {
        randomIndex = Math.floor(Math.random() * currentIndex);
        currentIndex -= 1;
        temporaryValue = array[currentIndex];
        array[currentIndex] = array[randomIndex];
        array[randomIndex] = temporaryValue;
    }

    return array;
}

//display the winner message
```

AWESOME

You've written apt comments to explain your code. Well done! 👏

```
const handleWinner = (time) => {
  document.querySelector('.final-moves').textContent = state.moves;
  document.querySelector('.final-stars').firstChild.innerHTML = starsDiv.inner
  document.querySelector('.final-time').textContent = time;
  document.querySelector('.game-panel').classList.toggle('hidden');
  document.querySelector('.winner-message').classList.toggle('hidden');
}

//update stars in DOM and state
const updateStars = (num) => {
  let stars = '';
  const starsDiv = document.querySelector('.score-panel .stars');
  starsDiv.innerHTML = '';
  for (let i = 1; i <= num; i++) {
    const star = '<li><i class="fa fa-star"></i></li>';
    stars = stars + star
  }
  starsDiv.innerHTML = stars;
  state.stars = num;
}

//show errors to user
const displayErrors = (err) => {
  closeErrors();
  const errorMessage = `
  <div class="error-message">
    ${err}
    <a class="close" aria-label="Close">
      <span aria-hidden="true">×</span>
    </button>
  </a>`;
  const errorDiv = document.createElement('div');
  errorDiv.className = 'error-div';
  errorDiv.innerHTML = errorMessage;
  scorePanel.parentNode.insertBefore(errorDiv, scorePanel.nextSibling);
  const closeButton = document.querySelector('.close');
  closeButton.addEventListener('click', closeErrors);
}

// removes the error message
const closeErrors = () => {
  const errorDiv = document.querySelector('.error-div');
  errorDiv ? errorDiv.remove() : null;
}

//when a card is click
const handleClick = (e,i) => {
  // exit if user tries to click on a card that is already solved
  if (cards[i].isSolved === true) {
    displayErrors('You found this match already, try clicking a new card');
    return;
  }
  // exit if matching is occuring(2 cards have been selected), user can only c
  if (state.noClicks) {
    displayErrors('Be patient young grasshopper, you can only match two cards
    return;
  }
  // if we are not matching lets just flip the card
  if (!state.isMatching) {
    flipCard(e,i);
  } else {
    // if we are matching make sure we clicked on another card
    if (state.firstIndex === i) {
      state.noClicks = false;
      displayErrors('You just clicked this card, try clicking a new card');
      return;
    } else {
      // check the match
      checkMatch(e,i);
    }
  }
  state.isMatching = !state.isMatching;
}

// shows the card to the user and saves its details in the state object
const flipCard = (e,i) => {
  cards[i].isMatching = true;
```

AWESOME

Good job using the es6 syntax for functions. You've divided your code into various functions making it mo

```
  e.target.className = 'card open show';
  setTimeout(function(){
        e.target.firstChild.classList.toggle('hidden')
    }, 250);
  state.firstCard = e;
  state.firstIndex = i;
}

// checks if two cards selected match
const checkMatch = (e,i) => {
  // since we are checking a match lets make sure the user can't click
  state.noClicks = true;
  // get the icon from the card the user selected
  const icon = e.target.lastElementChild.classList[1];
  // filter solution object from cards array
  const solution = cards.filter(c => c.isMatching === true && c.isSolved === f
  // show the card to the user
  e.target.className = 'card open show';
  // check if we have a match
  if ( solution[0].icon === icon) {
    handleMatch(e, i, true);
  } else {
    handleMatch(e, i, false);
  }
}

// display the match information to the user
const handleMatch = (e,i,match) => {
  // if we didn't find a match
  if (!match) {
    // show bad match to user
    e.target.className = 'card bad';
    state.firstCard.target.className = 'card bad';
    setTimeout(function(){
        e.target.firstChild.classList.toggle('hidden');
      }, 250)
    // wait 1 second for animcations and then hide the cards again
    setTimeout(function(){
      e.target.className = 'card close';
      state.firstCard.target.className = 'card close';
      e.target.firstChild.classList.toggle('hidden');
      state.firstCard.target.firstChild.classList.toggle('hidden');
    }, 1000);
  } else {
    // show the match to the user
    e.target.className = 'card match';
    state.firstCard.target.className = 'card match';
    setTimeout(function(){
        e.target.firstChild.classList.toggle('hidden');
      }, 250)
    // set the cards to solved in the cards object
    cards[i].isSolved = true;
    cards[state.firstIndex].isSolved = true;
    // add to the solutions counter
    state.solutions++;
  }
  // wait 1 second for animations
  setTimeout(function(){
    // reset isMatching in cards array for first card
    cards[state.firstIndex].isMatching = false;
    // add to moves counter and update DOM with the new number
    state.moves++;
    movesDiv.textContent = state.moves;
    // update stars when moves reach 11 and 21, default is 3 stars on DOM load
    state.moves === 21 ? updateStars(1) : state.moves === 11 ? updateStars(2)
    //checks if we have matched 8 cards in a game
    if (state.solutions === 8) {
      window.clearInterval(globalTimer);
      handleWinner(timerDiv.textContent);
    }
    // let the user click on cards again
    state.noClicks = false;
  }, 1000)
}

// initializes the game
const startGame = () => {
  // empty cards array then populate it
  cards = [];
  for (let x = 0; x <= 1; x++) {
   img.forEach(c => {
     cards.push({
       icon: c,
       isMatching: false,
       isSolved: false
     })
   })
  }

  // setup initial state
  state = {
    isMatching: false,
    firstCard: {},
    firstIndex: null,
    noClicks: false,
    solutions: 0,
    moves: 0,
    stars: 3,
    time: 0,
    hours: 0
  }
  // update DOM
  movesDiv.textContent = state.moves;
  updateStars(3);
    closeErrors();

  shuffle(cards);

  // build cards elements and append to DOM with event listener
  cards.forEach((c,i) => {
    const card = document.createElement('li');
    card.className = 'card match';
    card.innerHTML = `<i class='fa ${c.icon}'></i>`;
    card.addEventListener('click', (e) => {
      handleClick(e,i);
    });
    fragment.appendChild(card);
  })

  deck.appendChild(fragment);

  // wait 1.5 seconds and then hide the cards and start/clear timer
  setTimeout(function() {
    for (let i = 0; i <= 15; i++) {
      deckList[0].childNodes[i].className = 'card close';
      deckList[0].childNodes[i].firstChild.classList.toggle('hidden');
    }
    window.clearInterval(globalTimer);
    startTimer();
  }, 1500)

}

// on DOM ready start the game
document.addEventListener("DOMContentLoaded", function(event) {
  startGame();
  // setup event listener for reset game button
  resetButton.addEventListener('click', () => {
    timerDiv.textContent = '00:00';
    window.clearInterval(globalTimer);
    deck.innerHTML = '';
    closeErrors();
    startGame();
  })
  // setup event listener for play again winner button
  playAgainButton.addEventListener('click', function() {
    deck.innerHTML = '';
    document.querySelector('.game-panel').classList.toggle('hidden');
    document.querySelector('.winner-message').classList.toggle('hidden');
    startGame();
  })
});
```

css/app.css 1

README.md 1

introduction.txt

instructions.txt

index.html

RETURN TO PATH

Rate this review

Student FAQ