```
function sleep(ms) { //create a promise that resolves after ms time
    return new Promise(resolve => setTimeout(resolve, ms));
}

async function toggleGreyscale() {
    /*
        Handles greyscaling toggling effect.
        The while loop continuously loops ad infinitum and constantly
        toggles the addition and removal of the greyscale effect
    */
    const topImages = document.querySelectorAll(".welcomeright .top-row
.square");
    const bottomImages = document.querySelectorAll(".welcomeright
.bottom-row .square");

    while (true) {
        for (let image of topImages) {
            image.classList.add("grey");
        }
        for (let image of bottomImages) {
            image.classList.remove("grey")
        }
        await sleep(5000);
        for (let image of topImages) {
            image.classList.remove("grey");
        }
        for (let image of bottomImages) {
            image.classList.add("grey")
        }
        await sleep(5000);
    }
}

toggleGreyscale();
```

1) In editingLandingpage.js I wanted this to fulfill the 1 of 2 async/await ideas.
    a) If you look at the html page the first set of images alternate from grayscale to full color.

b) Sleep is kind of a helper function that is called to timeout so the greyscale holds for a reasonable amount of time.
c) toggleGreyscale takes the images so they can be addressed as elements of an array and assigns the images a class called “Grey” which greyscales the image. This is done so long as the page is open.

```
/* will be used to achieve my js idea of alternating grayscale images*/
.grey {
    filter: grayscale(100%);
}
```

2) This is the grey property used in the greyscale function; it takes the filter property and sets it to grayscale.

```
async function imageCarousel() {
    const images = [
        "chatthreads.png",
        "chatthreads.png",
        "chatthreads.png"
    ];

    const communityRight = document.querySelector('.communityright img');

    let currentIndex = 0;

    while (true) {
        communityRight.style.opacity = '0';
        await new Promise(resolve => setTimeout(resolve, 2000)); //wait
for fade out

        communityIndex = (currentIndex + 1) % images.length; //circular
array effect so it can loop back to another image
        communityRight.src = images[currentIndex];

        communityRight.style.opacity = '1';
        await new Promise(resolve => setTimeout(resolve, 5000)); //wait
for 5 seconds before next image
    }
```

```
}
```

3) My implementation of the image carousel is not exactly as shown in class. This is also my second implementation of async/await
    a) In the second section of the editingLandingpage.html it displays a singular image. This image is the image “carousel”
    b) I figured since I already did image carousel in my last project I could try a different style in which it fades out the old image and fades in the new image.
    c) The images are assigned an index and like the toggle grayscale it infinitely loops through the images and uses settimeout to keep the image displayed for a little bit before it gets cycled out.

```
const express = require('express');
const path = require('path');

//create express app
const app = express();

// Serve static files from the 'public' directory
// not sure why but using the approach of serving files from a static
folder called public works?
// Also for some reason have to put other folders in to access other
webpages
app.use(express.static('public'));

//define route to serve html file
app.get('/', (req, res) => {
    //send file
    res.sendFile(path.join(__dirname, 'public',
'editingLandingpage.html'));
});
//start server
const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

4) This is all of my server code.
    a) I set up an HTTP server because when I tried to set up an https

server with keys and certificates I kept getting a “Bad decrypt” error. I tried to generate new pairs which didn't work. I also tried to change the access permissions of the files which also didn't fix it.

b) So instead of no server I just put up an https server just to show that it does work using node.

```
function checkPasswordStrength(password){
    var strengthMeter = document.getElementById('strength-bar');
    var strength = 0;
    var width = 0;

    //If length is at least 8 increase strength
    if(password.length >=8){
        strength += 1;
    }

    //If have lower case letters increase strength
    if(/[a-z]/.test(password)){
        strength += 1;
    }

    //If have uppercase letters increase strength
    if(/[A-Z]/.test(password)){
        strength += 1;
    }

    //If have numbers increase strength
    if(/\d/.test(password)){
        strength += 1;
    }

    //If have special characters increase strength
    if(/[^A-Za-z0-9]/.test(password)){
        strength += 1;
    }

    //Increase size of meter and change color based on strength
    if(strength < 2){
```

```
        strengthMeter.className = 'strength-bar weak';
        width ='30%';
    }
    else if(strength === 2 || strength === 3){
        strengthMeter.className = 'strength-bar medium';
        width = '60%'
    }
    else {
        strengthMeter.className = 'strength-bar strong';
        width = '100%';
    }

    strengthMeter.style.width = width;
}
```

5) As one of my Udemy ideas I did password strength bar in my sign up page
    a) It takes the input and checks for if it has uppercase and lower case letters,numbers, and special characters.
    b) And based on how many criteria are fulfilled the bar changes color depending on how strong the password is.

```
// Set the target date and time for the countdown
var targetDate = new Date('2024-12-31T23:59:59').getTime();

// Update the countdown every second
var countdown = setInterval(function() {
  // Get the current date and time
  var now = new Date().getTime();

  // Calculate the remaining time
  var distance = targetDate - now;

  // Calculate days, hours, minutes, and seconds
  var days = Math.floor(distance / (1000 * 60 * 60 * 24));
  var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 *
60));
  var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
  var seconds = Math.floor((distance % (1000 * 60)) / 1000);
```

```
  // Display the countdown in the element with id "timer"
  document.getElementById('timer').innerHTML = days + 'd ' + hours + 'h '
+ minutes + 'm ' + seconds + 's ';

  // If the countdown is finished, display a message
  if (distance < 0) {
    clearInterval(countdown);
    document.getElementById('timer').innerHTML = 'EXPIRED';
  }
}, 1000); // Update every second (1000 milliseconds)
```

6) My Second udemy project was not exactly something from what was listed but seeing the “Timer” project gave me the idea to do a timer for a discount on a membership.
    a) The basic idea is that a target date is set and just for this example our current date is also taken. The time is then calculated by each category and displayed in the html element.
    b) Eventually the time between target and current day will be 0 and if that point is reached the timer will say “Expired” to signify the discount period is over.

```
function toggleAnswer(element) {
    var answer = element.nextElementSibling;
    answer.classList.toggle('show');
}
```

7) My third udemy project was a very simple collapsing FAQ tab.
    a) Each question tab can be clicked to show the answer or hide it by adding or removing ‘show’

```
// Define a function to process user input and generate responses
function processInput(input) {
    // Convert input to lowercase for case-insensitive matching
    input = input.toLowerCase();

    // Predefined responses for specific questions
    switch (input) {
        case 'how are you':
```

```
            return 'I am bad.'
        case 'Can you help me':
            return 'No.'
        case 'Who is the president':
            return 'Joe Biden.'
        default:
            return 'Sorry I dont understand the question.';
    }
}

// Define a function to display messages in the chat
function displayMessage(sender, message) {
    const chat = document.getElementById('chat');
    const p = document.createElement('p');
    p.textContent = `${sender}: ${message}`;
    chat.appendChild(p);
}

// Define a function to handle sending messages
function sendMessage() {
    const userInput = document.getElementById('userInput');
    const message = userInput.value.trim();

    if (message !== '') {
        // Display user message
        displayMessage('You', message);

        // Process user input and generate response
        const response = processInput(message);

        // Display ChatBot's response
        displayMessage('ChatBot', response);

        // Clear input field
        userInput.value = '';
    }
}
```

8) My implementation of a chatbot takes user input and spits out answers

based on silly responses and questions. If it can't come up with an answer the default is a “I don't know response”

a) Each function serves a different purpose: processInput( ) processes and chooses a response.
b) displayMessage( ) takes that chosen response and outputs it in the html by appending it
c) And sendMessage calls the other two methods and handles how the messages are displayed i.e it shows the question you asked and the response the chatbot gives.

```
.navbar {
    position: sticky;
    top: 0;
    background-color: var(--darkpurple);
    color: white;
    padding: 10px 0px;
    padding-left: 40px;
    text-align: left;
    z-index: 1000;
    /* so the navbar wil stay above everything else*/
}
```

9) My fourth udemy project doesn’t involve js it assigns the navbar div I created in html with a position called sticky.
    a) And I also set the z-index to 1000 so that no matter what the nav bar will always be showing front and foremost.

```
.blur-load {
    filter: blue(10px);
    /* starting blur effect*/
    animation: blurLoad 1s ease forwards;
}
@keyframes blurLoad {
    from {
        filter: blur(10px);
        /* starting blur animation */
    }

    to {
```

```
        filter: blur(0);
        /* transition to no blur*/
    }
}
```

10) My fifth Udemy project dealt with a blurry load in for the web page.
    a) What I did was take this class called blur load and assigned it to every large div container so that when the page loads it starts the initial blurring effect.
    b) Then I created a keyframe that gradually changes the filter from really blurry to not blurry(normal) to simulate a blurring loading in effect.