const express = require('express');

const path = require('path');

const app = express();

const port = 3000;

//allowing node to also use css and js files in the same directory

app.use(express.static(path.join(\_\_dirname, '.')))

//default localhost:3000 request on browser to sent "Landing.html"

app.get('/', (request, response) =>{

    response.sendFile(path.join(\_\_dirname, 'Landing.html'));

});

app.get('/home', (request, response) =>{

  response.sendFile(path.join(\_\_dirname, 'Home.html'));

});

app.listen(port, () => {

    console.log('express server listening on port ' + port);

});

* Utilized Node.js and the express API to have a local http host for the website.
* app.get functions reads the provided url in a browser and determines what file to send to the user depending on what domain was typed in.
  + Default localhost:3000 would result in Landing.html
  + Localhost:3000/home would result in Home.html
  + Any other variants would result in the user getting a url does not exist message
* app.use function then allows the html and express have access to all other needed files in the same directory such as the css and js files

<form action="/email" method="POST">

                <p>

                    <input class="text-container" type="text" name="email" id="email" required>

                    <label for="email" class="text-label">Email</label>

                </p>

                <input type="submit" class="btn">

            </form>

/\* JS node.js code for reading from a form post to output results to a text document \*/

const fs = require('fs');

//turns the req json from the form into usuable body format for req.body.email

app.use(express.json());

app.use(express.urlencoded({extended: true}));

app.post('/email', function(req, res){

  //getting email from the request

  const content = req.body.email;

  //appending the gotten email from request to text file

  fs.appendFile('text.txt', content + '\n', err => {

    if (err) {

      console.error(err);

    }

  });

});

* Using node fs and node express to get information from a html form to write to a text file.
* Html form has an action of “/email” and a method of post. Using post as the method allows express to read it, specifying an “/email”. The information from the html form is then the request read from express. With this setup content is then equal to the email section of the json file, extracting the needed information from the request. Lastly, appendFile adds the content to the specifies text file.
* Initially content could not extract the email section from the request due to it being a full .json file. This request was not usable as a json file with a lot of extra information so in order to extract the needed information from it app.use was passed before the post function to turn inputted json files to usable and discernable information.

<div class="VerticalSlider">

            <div class="left-side">

                <div style="background-color:#113768;">

                    <h1>APU</h1>

                    <p>Innovation in Processing Units</p>

                </div>

                <div style="background-color:#555555;">

                    <h2>Join Us</h2>

                    <a href="Home.html"><u>Click Here</u></a>

                </div>

                <div style="background-color:#0598ce;">

                    <h2>Why Us</h2>

                    <p>Committed to Integrity and Quality for all</p>

                </div>

                <div style="background-color:#0598ce;">

                    <h2>What we do</h2>

                    <p>Innovation and production on state of the art <br>central and graphical processing units</p>

                </div>

            </div>

            <div class="right-side">

                <div style="background-image: url(Images/tech\_bg1.jpg);"></div>

                <div style="background-image: url(Images/tech\_bg3.jpg);"></div>

                <div style="background-image: url(Images/tech\_bg2.jpeg);"></div>

                <div style="background-image:url(Images/factory.jpg);"></div>

            </div>

            <div class="action-button">

                <button class="down-button"> <p>v</p> </button>

                <button class="up-button"> <p>^</p> </button>

            </div>

        </div>

/\*CSS code for the Vertical Slider\*/

.VerticalSlider{

    position: relative;

    overflow: hidden;

    width: 100vw;

    height: 100vh;

}

.left-side{

    height: 100%;

    width: 30%;

    position: absolute;

    top: 0;

    left: 0;

}

.left-side > div{

    height: 100%;

    width: 100%;

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    color: white;

}

.right-side{

    height: 100%;

    position: absolute;

    top:0;

    left: 30%;

    width: 70%;

}

.right-side > div{

    background-repeat: no-repeat;

    background-size: cover;

    background-position: center center;

    height: 100%;

    width: 100%;

}

.VerticalSlider button{

    background-color: white;

    border: none;

    cursor: pointer;

    font-size: 20px;

    padding: 10px;

    color: lightgray;

}

.VerticalSlider button:hover{

    color:black;

}

.VerticalSlider .action-button button{

    position: absolute;

    left: 30%;

    top: 50%;

    z-index: 100;

}

.VerticalSlider .action-button .down-button{

    transform: translateX(-100%);

    border-top-left-radius: 5px;

    border-bottom-left-radius: 5px;

}

.VerticalSlider .action-button .up-button{

    transform: translateY(-100%);

    border-top-right-radius: 5px;

    border-bottom-right-radius: 5px;

}

/\* JS code for Vertical Slider \*/

const sliderContainer = document.querySelector('.VerticalSlider')

const slideRight = document.querySelector('.right-side')

const slideLeft = document.querySelector('.left-side')

const upButton = document.querySelector('.up-button')

const downButton = document.querySelector('.down-button')

const slidesLength = slideRight.querySelectorAll('div').length

let activeSlideIndex = 0

upButton.addEventListener('click',() => changeSlide('up'))

downButton.addEventListener('click',() => changeSlide('down'))

const changeSlide = (direction) => {

    console.log(activeSlideIndex)

    const slideHeight = sliderContainer.clientHeight

    //Increment Index+1 when pressing up button

    if(direction === 'up'){

        activeSlideIndex++

        if(activeSlideIndex > slidesLength - 1){

            activeSlideIndex = 0

        }

    }

    //Increment Index-1 when pressing down button

    else if(direction === 'down'){

        activeSlideIndex--

        if(activeSlideIndex < 0){

            activeSlideIndex = slidesLength - 1

        }

    }

    slideRight.style.transform = `translateY(-${activeSlideIndex\*slideHeight}px)`

    slideLeft.style.transform = `translateY(-${activeSlideIndex\*slideHeight}px)`

}

* Html, css, and js code for a double vertical slider from the Udemy course.
* Initially the default code from the course did not work, the inclusion of the line “slideLeft.style.top=`-${(slidesLength-1)\*100}vh`” in the js code actually broke the indenting of the code. This line allowed for the activeSlideIndex variable in the function to increment beyond the set limit of slidesLength-1. Then removing this line fixed the code and website together.

<div class="gallery-product" style="background-image: url(Images/Products/GPU/ASRock\_Radeon\_RX\_6600-2.png);">

                <img src="Images/Products/GPU/ASRock\_Radeon\_RX\_6600.jpg" alt="GPU">

            </div>

.gallery-product{

    float:left;

    height: 25vw;

    width: 25vw;

    margin-left: 1.5vw;

    margin-right: 1.5vw;

    margin-top: 2vh;

    margin-bottom: 3vh;

    overflow: hidden;

    text-align: center;

    vertical-align: middle;

    background-size:contain;

    background-position: center;

    background-repeat: no-repeat;

    background-color:white;

}

.gallery-product img{

    height: 25vw;

    width: 25vw;

    display: inline-block;

    vertical-align: middle;

}

.gallery-product.hover img{

    visibility: hidden;

}

/\* JS code for image hover change \*/

const images = document.querySelectorAll(".gallery-product")

images.forEach((image) => {

    image.addEventListener('mouseenter', ()=>{

        image.classList.add('hover')

    })

    image.addEventListener('mouseleave', ()=> {

        image.classList.remove('hover')

    })

})

* A gallery product image that changes the image in the box whenever the user mouse hover’s the image.
* My implementation causes the displayed image to change to the set background image whenever the event listener detected the user’s mouse hovered over the image. This can be also changed to have text appear instead of a background image.
* Whenever a mouse enters the box, a hover class is added to the div containing that image only. This hover class then changes the css to hide the img, revealing the background image that was behind it.

<nav class="" id="nav">

            <ul>

                <li><a href="Home.html">Home</a></li>

                <li><a href="Products\_GPU.html">GPU</a></li>

                <li><a href="Products\_CPU.html" class="current">CPU</a></li>

            </ul>

            <button class="icon" id="toggle">

                <div class="line line1"></div>

                <div class="line line2"></div>

            </button>

        </nav>

nav{

    background-color: #0598ce;

    padding: 20px;

    width: 40px;

    display: flex;

    align-items: center;

    justify-content: center;

    border-radius: 3px;

    box-shadow: 0 2px 5px black;

    transition: width 0.6s linear;

    margin-top: 2vh;

    margin-left: 2vw;

    border-radius: 5px;

    position: fixed;

}

nav.active{

    width: 225px;

}

nav ul{

    display: flex;

    list-style-type: none;

    margin: 0;

    padding: 0;

    width: 0;

    transition: width 0.6s linear;

}

nav.active ul{

    width: 100%;

}

nav ul li{

    transform: rotateY(0deg);

    opacity: 0;

    transition: transform 0.6s linear, opacity 0.6s linear;

}

nav.active ul li{

    opacity: 1;

    transform: rotateY(360deg)

}

nav ul a{

    position: relative;

    color: white;

    text-decoration: none;

    margin: 0 10px;

}

nav ul a.current{

    color:#113768;

}

.icon{

    background-color: #0598ce;

    border: 0;

    cursor: pointer;

    padding: 0;

    position: relative;

    height: 30px;

    width: 30px;

}

.icon:focus{

    outline: 0;

}

.icon .line{

    background-color: white;

    height: 2px;

    width: 20px;

    position: absolute;

    top:10px;

    left: 5px;

    transition: transform 0.6s linear;

}

.icon .line2{

    top: auto;

    bottom: 10px;

}

nav.active .icon .line1{

    transform: rotate(-765deg) translateY(5.5px);

}

nav.active .icon .line2{

    transform: rotate(765deg) translateY(-5.5px);

}

/\* JS code for animated navigation \*/

const toggle = document.getElementById('toggle')

const nav = document.getElementById('nav')

toggle.addEventListener('click', ()=>{

    nav.classList.toggle('active')

})

* Html, css, and js code for the animated navigation bar from the udemy course
* Acts as a normal navigation bar except it has a closing and opening animation.
* The CSS code controls what is displayed during the animation, opened, and closed.
* JS code toggles which of the css code is displayed by adding an event listener for the button which toggles the ‘active’ class for the html <div>.
* The button itself consists of two individual bars which rotate in order for the animation to play and the button to change from bars to a x.