Grey paper – Elazar Kagan – website project 2

1) HTML

```
<div class='info' id="info">
          <table>
            <tr>
              <th>More Info</th>
              <th></th>
            </tr>
            <tr>
              <td>City:</td>
              <td id="tCity">error</td>
            </tr>
            <tr>
              <td>Country:</td>
              <td id="tCountry">error</td>
            </tr>
            <tr>
              <td>Host Name:</td>
              <td id="tHostname">error</td>
            </tr>
            <tr>
              <td>IP:</td>
              <td id="tIp">error</td>
            </tr>
            <tr>
              <td>Location:</td>
              <td id="tLoc">error</td>
            </tr>
            <tr>
              <td>Postal code:</td>
              <td id="tPostal">error</td>
            </tr>
            <tr>
              <td>Region:</td>
              <td id="tRegion">error</td>
            </tr>
            <tr>
              <td>Time Zone:</td>
              <td id="tTimezone">error</td>
            </tr>
            </table>
        </div>
```

This is the HTML for a table that will be filled out with JavaScript when it gets the location info. I made the preset values to be “error” because the JavaScript should be replacing it with location data so if it doesn’t there must be some sort of error which is what will be displayed. This is similar to src =”” where you give an alternate option in case something doesn’t work

2) HTML:

```
<script async
    src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDbXAn2TrbMzutzSbRYulhU
lhEX84tohTE&callback=initMap">
    </script>
```

This is an API I got from google that will be used in the JavaScript to create the map and get the point of the user location. I had to login to my google account to set this up and they gave me this link for the API.

3) CSS:

```
toggle.addEventListener('click', (e) => {
    const html = document.querySelector('html')
    const navDark = document.querySelector('.navfile')
    const textDark = document.querySelector('.text')
    const navIconDark = document.querySelector('.icon')
    const dateCircleDark = document.querySelector('.date')
    const faqDark = document.querySelector('.faqfile')
    const buttonDark = document.querySelector('.buttonfile')
    const clockSecondDark = document.querySelector('.second')
    const clockMiddleDark = document.querySelector('.center-point')
    const faqButtonDark = document.querySelector('.faq.active')
    const infoButtonDark = document.querySelector('#info')
    console.log(faqButtonDark)


    if(html.classList.contains('dark')) {
        html.classList.remove('dark')
        buttonDark.classList.remove('dark')
        navDark.classList.remove('dark')
        navIconDark.classList.remove('dark')
        textDark.classList.remove('dark')
        faqDark.classList.remove('dark')
        clockSecondDark.classList.remove('dark')
        clockMiddleDark.classList.remove('dark')
        dateCircleDark.classList.remove('dark')
```

```
        faqButtonDark.classList.remove('dark')
        infoButtonDark.classList.remove('dark')
        e.target.innerHTML = 'Dark mode'
    } else {
        html.classList.add('dark')
        navDark.classList.add('dark')
        buttonDark.classList.add('dark')
        navIconDark.classList.add('dark')
        faqDark.classList.remove('dark')
        textDark.classList.add('dark')
        clockSecondDark.classList.add('dark')
        clockMiddleDark.classList.add('dark')
        dateCircleDark.classList.add('dark')
        faqButtonDark.classList.add('dark')
        infoButtonDark.classList.add('dark')
        e.target.innerHTML = 'Light mode'
    }
})
```

One of the cool features of my website is the button to switch the site into dark mode. I took the original file from the clock to start this file and added all the other elements that I also wanted to swich colors. First I get the class/id of the item from the html and assign it to a variable. Then I have an if/else seeing if the site is in dark mode or not and based on that adding or removing classes that contain css for the dark mode. The if/else is triggered by the user click a button.

4) CSS:

```
:root {
    --primary-color: #000;
    --secondary-color: #fff;
    --primary-color2: #ff0080;
    --secondary-color2: #0096FF;
}

html {
    transition: all 0.5s ease-in;
    background-image: linear-gradient(to bottom right, white, grey);
    background-size: cover;
}

html.dark {
    --primary-color: #fff;
    --secondary-color: #333;
}
```

```css
html.dark {
    transition: all 0.5s ease-in;
    background-image: none;
    background-color: #111;
    color: var(--primary-color);
}


.faqfile.dark {
    transition: all 0.5s ease-in;
    background-color: #111;
    color: var(--secondary-color2);
}
```

This is some of the CSS for switching to dark mode described above. Each element has its own attributes in its own files but here I modify the color when it has the class of dark (which was added and removed by the JavaScript in #3). Also, I have some root colors so I can keep all the colors organized and if I want to change anything it only has to be done once.

5) JavaScript:

```javascript
async function fetchText() {
    let url = 'https://ipinfo.io/json?token=1cc230fe868729';
    let response = await fetch(url);
    let data = await response.json();
    console.log(data);
    document.getElementById('city').innerHTML = data.city;
    document.getElementById('region').innerHTML = data.region;
    document.getElementById('timezone').innerHTML = data.timezone;
    document.getElementById('tCity').innerHTML = data.city;
    document.getElementById('tRegion').innerHTML = data.region;
    document.getElementById('tTimezone').innerHTML = data.timezone;
    document.getElementById('tCountry').innerHTML = data.country;
    document.getElementById('tPostal').innerHTML = data.postal;
    document.getElementById('tLoc').innerHTML = data.loc;
    document.getElementById('tIp').innerHTML = data.ip;
    document.getElementById('tHostname').innerHTML = data.hostname;
}
fetchText();
```

Here is the JavaScript for the HTML in #1 that fills in the user location into a table based on the users IP address. First, I had to get a second API from ipinfo.com that was able to get all the info for me. This info was stored in data, I was then able to overwrite the html for the elements in my table with the info from data. And I did this for all the info I wanted to show the user.

6) JavaScript:

```
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(successFunction, errorFunction);
}
function successFunction(position) {
  let lat = position.coords.latitude;
  let long = position.coords.longitude;
  console.log(lat, long)
 initMap(lat, long);
}
function errorFunction(){
  alert("Geolocation failed");
};

function initMap(latitude, longitude) {
  console.log(latitude, longitude);
  const locationGM = { lat: latitude, lng: longitude };
  const map = new google.maps.Map(document.getElementById("map"), {
    zoom: 16,
    center: locationGM,
  });
  const marker = new google.maps.Marker({
    position: locationGM,
    map: map,
  });
  }

  window.initMap = initMap();
```

This code probably took me the most time to do. It was very complicated trying to get this to work and there was a lot of errors in the beginning, but I got it to work. First it uses the JavaScript function that gets the users location as a latitude and longitude. Then I had to pass that into some code I got from google that makes a marker on a map. There's also code that actually creates the map. With all the information given.

7) HTML:

```
window.addEventListener("load", function(){
    setTimeout(
        function open(event){
            document.querySelector(".popup").style.display = "block";
        },
        1000
    )
});
```

```
document.querySelector(".pclose1").addEventListener("click", function(){
    document.querySelector(".popup").style.display = "none";
});
document.querySelector(".pclose2").addEventListener("click", function(){
    document.querySelector(".popup").style.display = "none";
});it
```

This is the JavaScript for popup the user is greeted with when accessing the site. It tells them that the website will access their location and asks the user if he wants to continue to the site. This code makes the popup appear and disappear from the screen (revealing the website) by changing the CSS of the popup elements to display = “block” or display = “none”. When the user enters the website, the display is set to “block” and when the user clicks the “x” or the “lets go” button it sets the display to “none” making the popup disappear and showing the website beneath.

8) JavaScript:

```
const buttonInfo = document.getElementById('buttonInfo')
const showInfo = document.getElementById('info')

buttonInfo.addEventListener('click', () => showInfo.classList.toggle('showInfo'))
```

This is the JavaScript that makes the button imported from Udemy also clickable that when clicked it will display table with all the additional information about the user’s location. This is done by assigning html elements to variables and adding an event listener that when the button is clicked toggles the showInfo class thus revealing the table and clicking again removes the class and thereby removes the table

9) HTML:

```
<h3 class="faq-title">
              Here is a section that has a list of 5 images.
            </h3>
            <p class="faq-text">
              <li><img src="img1.webp" alt="lines of code"></li>
              <li><img src="img2.jpg" alt="person with neon lights and shadows"></li>
              <li><img src="img3.jpg" alt="an astronaut chilling on the moon with earth in the background"></li>
              <li><img src="img4.webp" alt="a panda wearing sunglasses"></li>
              <li><img src="img5.jpg" alt="a futuristic tornado made out of lightning"></li>
            </p>
```

One of the FAQ sections where I put a list of different images I found. I also added the alts to each pictures in case for some reason it wont be able to display still a user can get a quick description of what they were supposed to see.

10) HTML:

```
<div class="popup">
```

```
      <div class="popupText">
        <button class="pclose1">&times;</button>
        <h2>Welcome to WhereAmI.mcon</h2>
        <p>
            This website finds your location and displays the city, state,
timezone, time, and date<br> Please accept the prompt from your browser to allow
location services.
        </p>
        <a class="pclose2">Let's Go</a>
      </div>
      <div class="popupBG"></div>
    </div>
```

CSS:

```
.popupBG {
    background-image: linear-gradient(to bottom right, teal, cyan);
    width: 100vw;
    height: 100vh;
    position: fixed;
    left: -189%;
    top: -540px;
    border-radius: 0;
    z-index: 10;
}

.popup {
    background-color: #ffffff;
    width: 450px;
    padding: 30px 40px;
    position: fixed;
    transform: translate(-50%,-50%);
    left: 50%;
    top: 50%;
    border-radius: 8px;
    font-family: "Poppins",sans-serif;
    display: block;
    z-index: 2;
}
```

I had to make a separate section for the popup in the beginning of the site that will block the user from entering the site until he agrees to the site getting his location. It's the first HTML section so it's the first thing to show up. The display is set to fixed so you cant scroll down to see more content and the height is set to 100vh so it takes up the whole screen. Also I added a z-index to put it on top of everything else as the position was now fixed.