Web Dev Notes

▼ CSS

▼ POSITIONS :-

1. **Relative**:- position of an element relative to its parent element.
2. **Absolute**:- It allows you to place any page element exactly where you want it. You use the positioning attributes top, left, bottom, and right to set the location.
3. **Static** : Default locked position of an element, it cannot be moved by right, left, top bottom, but can be moved using only margins and padding.
4. **Fixed** : The element won't move even one scrolling, it will be fixed.

- Margins will affect all the other elements in the page, but if the position is set to be relative (TBLR) will not affect other elements.
- an element can be absolute and relative to an item at the same time if its parent is set to be relative

▼ FLEXBOX :-

1. When display: flex is ON, all the child (individually) will be arranged in one line, ie if child are made up of divs then also same case.

- Agar insta post ke upar dp spherical dp ke side me 2 seperate lines daalni hai to name and username ko ek div me rkhna padega.

▼ JS

▼ Projects

▼ Weather Web App

▼ Explaination

Project Overview

My project is a weather web application named "SkyCast" that allows users to get weather information for their current location or search for weather information by city name. The application uses the OpenWeatherMap API to fetch weather data.

HTML Structure

The HTML structure consists of several key components:

1. **Wrapper**: The main container for the entire application.
2. **Tab Container**: Contains tabs for switching between "Your Weather" and "Search Weather".
3. **Weather Container**: Contains sub-containers for granting location access, searching weather, loading screen, and displaying weather information.

Key HTML Elements

```html
<div class="wrapper">
   <h1>SkyCast</h1>

   <div class="tab-container">
      <p class="tab" data-userWeather>Your Weather</p>
      <p class="tab" data-searchWeather>Search Weather</p>
   </div>

   <div class="weather-container">
      <!--Grant Location Container-->
      <div class="sub-container grant-location-container">
         <img src="assets/location.png" width="80" height="80" alt="location" loading="lazy">
         <p>Grant Location Access</p>
         <p>Allow access to get weather Information</p>
         <button class="btn" data-grantAccess>Grant Access</button>
      </div>

      <!--Search weather page-->
      <form class="form-container" data-searchForm>
```

```
            <input placeholder="Search for City.." data-searchInput>
            <button class="btn">
                <img src="assets/magnifying-glass-location-solid.svg" height="20" width="20" alt="search" loading="lazy">
            </button>
        </form>

        <!--Loading Screen-->
        <div class="sub-container loading-container">
            <img id="load" src="assets/Gear Loader.gif" height="90" width="90" alt="loading" loading="lazy">
            <p>Loading</p>
        </div>

        <!--Show weather info-->
        <div class="sub-container user-info-container">
            <div class="name">
                <p data-cityName></p>
                <img data-countryIcon>
            </div>
            <p data-weatherDesc></p>
            <img data-weatherIcon>
            <p data-temp></p>
            <div class="parameter-container">
                <div class="parameter">
                    <img src="<https://cdn-icons-png.flaticon.com/128/959/959737.png>" height="60" width="60" alt="wind" loading="lazy">
                    <p>windspeed</p>
                    <p data-windspeed></p>
                </div>
                <div class="parameter">
                    <img src="<https://cdn-icons-png.flaticon.com/512/5664/5664993.png>" width="60" height="60" alt="humidity" loading="lazy">
                    <p>humidity</p>
                    <p data-humidity></p>
                </div>
            <div class="parameter">
               <img src="<https://cdn-icons-png.flaticon.com/128/414/414927.png>" width="60" height="60" alt="cloudiness" loading="lazy">
               <p>Clouds</p>
               <p data-cloudiness></p>
            </div>
         </div>
      </div>
   </div>
</div>
```

JavaScript Functionality

The JavaScript code handles the logic for switching tabs, fetching weather data, and updating the UI.

Key JavaScript Functions and Event Listeners

1. **Initialization and Tab Switching**

```
const userTab = document.querySelector("[data-userWeather]");
const searchTab = document.querySelector("[data-searchWeather]");
const userContainer = document.querySelector(".weather-container");
const grantAccessContainer = document.querySelector(".grant-location-container");
const searchForm = document.querySelector("[data-searchForm]");
const loadingScreen = document.querySelector(".loading-container");
const userInfoContainer = document.querySelector(".user-info-container");

let currentTab = userTab;
const API_key = "e02bf734e960b661d46b5fca0e771631";
currentTab.classList.add("current-tab");
```

```
// Clear session storage to reset grant access permission
sessionStorage.removeItem("user-coordinates");

// Check session storage for coordinates
getfromSessionStorage();

function switchTab(clickedTab){
    if(clickedTab != currentTab){
        currentTab.classList.remove("current-tab");
        currentTab = clickedTab;
        currentTab.classList.add("current-tab");

        if(!searchForm.classList.contains("active")){
            userInfoContainer.classList.remove("active");
            grantAccessContainer.classList.remove("active");
            searchForm.classList.add("active");
        } else {
            searchForm.classList.remove("active");
            userInfoContainer.classList.remove("active");
            getfromSessionStorage();
        }
    }
}

userTab.addEventListener("click", () ⇒ switchTab(userTab));
searchTab.addEventListener("click", () ⇒ switchTab(searchTa
b));
```

2. **Fetching Weather Data**

```
async function fetchUserWeatherInfo(coordinates){
    const {lat, lon } = coordinates;
    grantAccessContainer.classList.remove("active");
    loadingScreen.classList.add("active");

    try{
        const response = await fetch(
            `https://api.openweathermap.org/data/2.5/weather?lat
=${lat}&lon=${lon}&appid=${API_key}&units=metric`
        );
        const data = await response.json();
        loadingScreen.classList.remove("active");
        userInfoContainer.classList.add("active");
        renderWeatherInfo(data);
    } catch(err){
        loadingScreen.classList.remove("active");
        console.log("error occurred due to ", err);
    }
}

async function fetchSearchWeatherInfo(city){
    loadingScreen.classList.add("active");
    userInfoContainer.classList.remove("active");
    grantAccessContainer.classList.remove("active");

    try {
        const response = await fetch(
            `https://api.openweathermap.org/data/2.5/weather?q=
${city}&appid=${API_key}&units=metric`
        );
        const data = await response.json();
        loadingScreen.classList.remove("active");
        userInfoContainer.classList.add("active");
        renderWeatherInfo(data);
    } catch (error) {
        loadingScreen.classList.remove("active");
        console.log("error occurred due to ", error);
    }
}
```

3. **Rendering Weather Information**

```
function renderWeatherInfo(weatherInfo){
    const cityName = document.querySelector("[data-cityNam
e]");
```

```
    const countryIcon = document.querySelector("[data-countryIcon]");
    const desc = document.querySelector("[data-weatherDesc]");
    const weatherIcon = document.querySelector("[data-weatherIcon]");
    const temp = document.querySelector("[data-temp]");
    const wind = document.querySelector("[data-windspeed]");
    const humidity = document.querySelector("[data-humidity]");
    const cloudiness = document.querySelector("[data-cloudiness]");

    if (cityName) cityName.innerText = weatherInfo?.name;
    if (countryIcon) countryIcon.src = `https://flagcdn.com/144x108/${weatherInfo?.sys?.country.toLowerCase()}.png`;
    if (desc) desc.innerText = weatherInfo?.weather?.[0]?.description;
    if (weatherIcon) weatherIcon.src = `http://openweathermap.org/img/w/${weatherInfo?.weather?.[0]?.icon}.png`;
    if (temp) temp.innerText = `${weatherInfo?.main?.temp}°C`;
    if (wind) wind.innerText = `${weatherInfo?.wind?.speed} m/s`;
    if (humidity) humidity.innerText = `${weatherInfo?.main?.humidity}%`;
    if (cloudiness) cloudiness.innerText = `${weatherInfo?.clouds?.all}%`;
}
```

4. **Geolocation and Error Handling**

```
function getLocation(){
    if(navigator.geolocation){
        navigator.geolocation.getCurrentPosition(showPosition, showError);
    } else {
        alert("Geolocation is not supported by this browser");
    }
}

function showPosition(position){
    const userCoordinates = {
        lat: position.coords.latitude,
        lon: position.coords.longitude
    }
    sessionStorage.setItem("user-coordinates", JSON.stringify
(userCoordinates));
    fetchUserWeatherInfo(userCoordinates);
}

function showError(error) {
    switch(error.code) {
        case error.PERMISSION_DENIED:
            alert("User denied the request for Geolocation.");
            break;
        case error.POSITION_UNAVAILABLE:
            alert("Location information is unavailable.");
            break;
        case error.TIMEOUT:
            alert("The request to get user location timed out.");
            break;
        case error.UNKNOWN_ERROR:
            alert("An unknown error occurred.");
            break;
    }
}

const grantAccessButton = document.querySelector("[data-g
rantAccess]");
grantAccessButton.addEventListener("click", () ⇒ {
    sessionStorage.removeItem("user-coordinates");
    getLocation();
});

let searchInput = document.querySelector("[data-searchInpu
t]");
searchForm.addEventListener("submit", (e) => {
    e.preventDefault();
    let cityName = searchInput.value;

    if(cityName === ""){
        alert("Please enter a city name");
        return;
    } else {
        fetchSearchWeatherInfo(cityName);
    }
});
```

Flow of the Application

1. **Initialization**: When the page loads, the session storage is cleared, and the application checks if there are any saved coordinates in the session storage.
2. **Tab Switching**: Users can switch between "Your Weather" and "Search Weather" tabs. The `switchTab` function handles the tab switching logic.
3. **Grant Access**: When the user clicks the "Grant Access" button, the `getLocation` function is called to get the user's current location using the browser's geolocation API.
4. **Fetch Weather Data**: Depending on whether the user is using their current location or searching by city name, the appropriate fetch function (`fetchUserWeatherInfo` or `fetchSearchWeatherInfo`) is called to get the weather data from the OpenWeatherMap API.
5. **Render Weather Information**: The `renderWeatherInfo` function updates the UI with the fetched weather data.

CSS Styling

The CSS file contains styles for various components of the application, including the wrapper, tabs, buttons, containers, and weather information display.