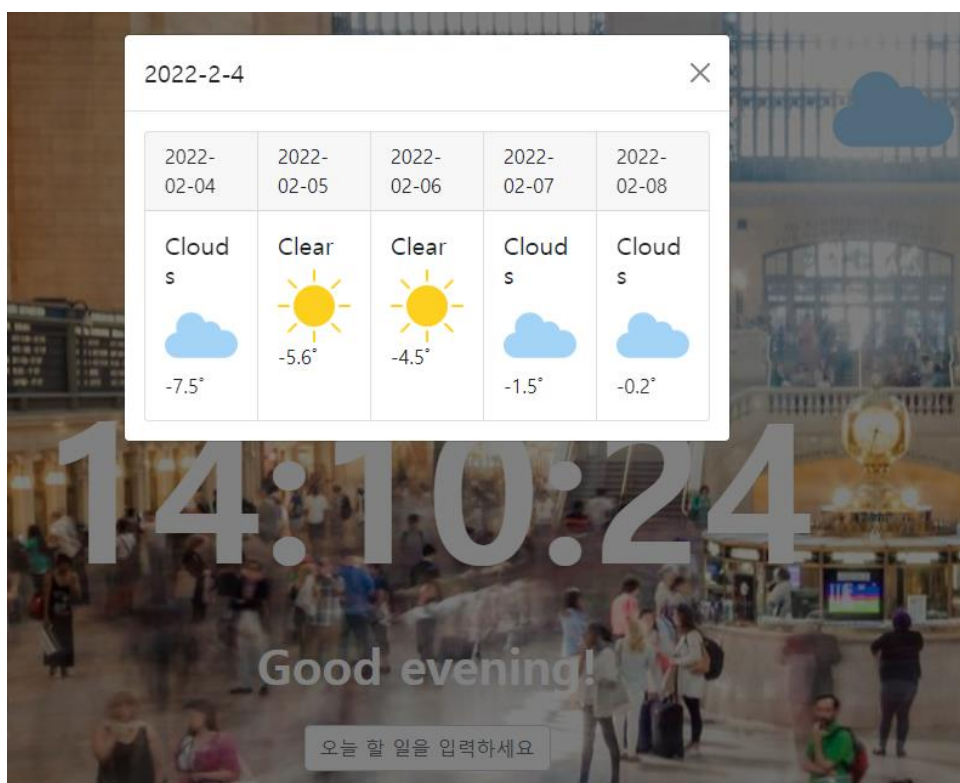


SAFFY 2022 7기 Web PJT 관통 프로젝트

구분	DAY 1
진행일자	2022년 2월 11일
주제	주간날씨 웹서비스 제작
학습내용	<ol style="list-style-type: none"> 1. 프로젝트 기획 및 리소스 준비 2. 날씨 API 받아오기 3. HTML + CSS 구성 4. Axios 설치 5. 이미지 가져오기 6. 날씨 정보 시각화
예상구현기간	8H

Image API를 사용해서 5초 간격으로 배경화면이 자동으로 변하는 화면을 구성한다.

날씨 API를 활용해 현재 위치에 대한 일주일 간 날씨 정보를 나타내는 웹서비스를 제작한다. 날씨 정보를 얻을 수 있는 외부 API를 사용하여, 날씨 정보가 담긴 JSON Data를 axios와 javascript를 사용한 비동기통신으로 받아온다. 그리고 정보를 Parsing하여 화면에 날씨 데이터를 출력한다.



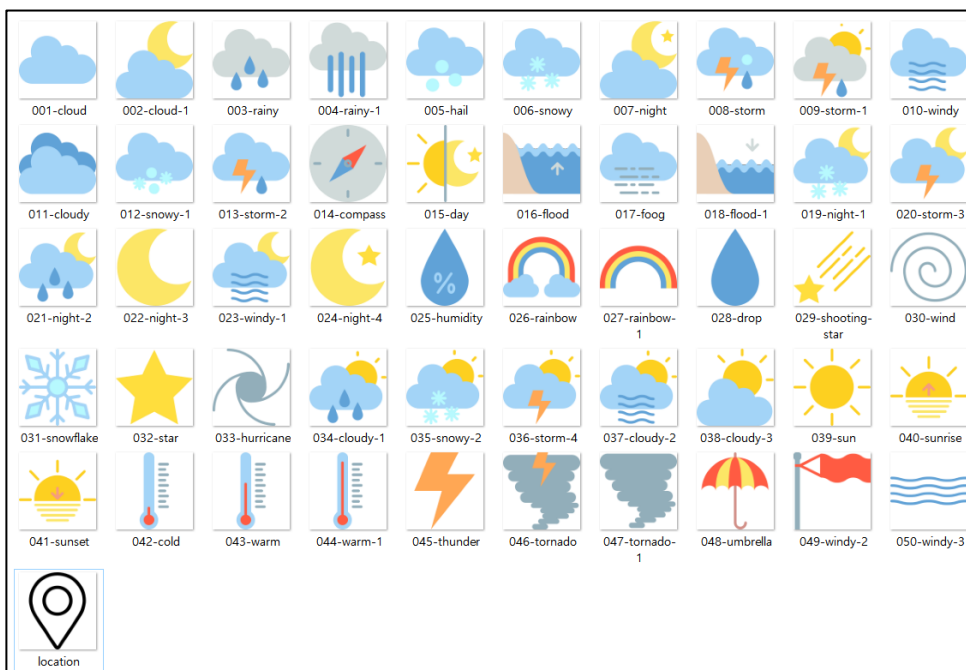
1. 프로젝트 기획 및 리소스 준비

웹서비스를 제작하기 전, 구체적인 계획을 세우고 필요한 리소스 (이미지)를 미리 준비 해 둔다. 그리고 계획에 맞추어 한 단계씩 웹서비스를 제작 해 본다.

A. 절차

- i. 날씨 아이콘 준비
- ii. 이미지 API 받아오기
- iii. 기본 구조 세팅하기
- iv. 배경화면 API 사용하기
- v. 시계 적용하기
- vi. localStorage로 TODO 작성
- vii. 날씨 API 회원 가입 및 API 받아오기
- viii. API 동작 Test
- ix. 받아온 API Parsing
- x. JavaScript 추가 기능 구현

2. 날씨에 쓰일 이미지 다운로드



날씨를 나타내는 아이콘 역할 이미지를 준비한다. 본 프로젝트에서는 위와 같은 이미지가 제공되며, 다른 이미지로 교체해도 괜찮다.

3. 이미지 API 받아오기

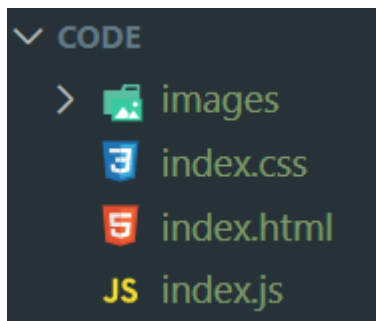
- A. 무료로 이미지 API를 제공해주는 사이트를 사용한다.(<https://picsum.photos/>)
- B. <https://picsum.photos/size>를 통해 이미지를 가져 올 수 있다.

ex) <https://picsum.photos/1280/720>



4. 기본 구조 세팅하기

- A. 기본 파일 구조를 잡는다.



B. 기본 HTML 구조를 잡는다. Bootstrap, axios CDN 도 추가한다.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <!-- bootstrap -->
    <!-- CSS only -->
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
      crossorigin="anonymous"
    />
    <!-- mycss -->
    <link rel="stylesheet" href="./index.css" />
    <title>Document</title>
  </head>
  <body>
    <!-- axios -->
    <script src="https://cdn.jsdelivr.net/npm/axios@0.25.0/dist/axios.min.js"></script>
    <!-- bootstrap -->
    <!-- JavaScript Bundle with Popper -->
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYs0g+OMhuP+ILRH9sENB00LRn5q+8nbTov4+1p"
      crossorigin="anonymous"
    ></script>
    <!-- myjs -->
    <script src="./index.js"></script>
  </body>
</html>
```

C. 기본 태그 구성

main 태그를 활용해서 기초 태그 구성을 잡아준다.

```
<main class="main-container">
  <div class="timer-wrapper">
    <div class="timer"></div>
    <div class="timer-content">Good evening!</div>
  </div>
  <div class="memo-input-wrapper">
    <input
      type="text"
      class="memo-input form-control"
      placeholder="오늘 할 일을 입력하세요!"
    />
  </div>
  <div class="memo-wrapper">
    <div class="memo-title">TODAY</div>
    <div class="memo"></div>
  </div>
</main>
```

D. CSS 구성

style tag를 추가해서 활용 하기 위한 CSS를 미리 정의해 둔다.

```
body {  
  background-size: cover;  
  background-repeat: no-repeat;  
}  
  
.main-container {  
  min-height: 100vh;  
  display: flex;  
  align-items: center;  
  justify-content: center;  
  flex-direction: column;  
}  
  
.timer {  
  font-weight: bold;  
  font-size: 10rem;  
}  
  
.timer-content {  
  font-weight: bold;  
  font-size: 2.5rem;  
  text-align: center;  
}  
  
.memo-input-wrapper {  
  margin: 20px;  
}  
  
.memo {  
  font-size: 2rem;  
}
```

여기까지 구성 하면 기초 세팅은 마무리 된다.

Good evening!

오늘 할 일을 입력하세요

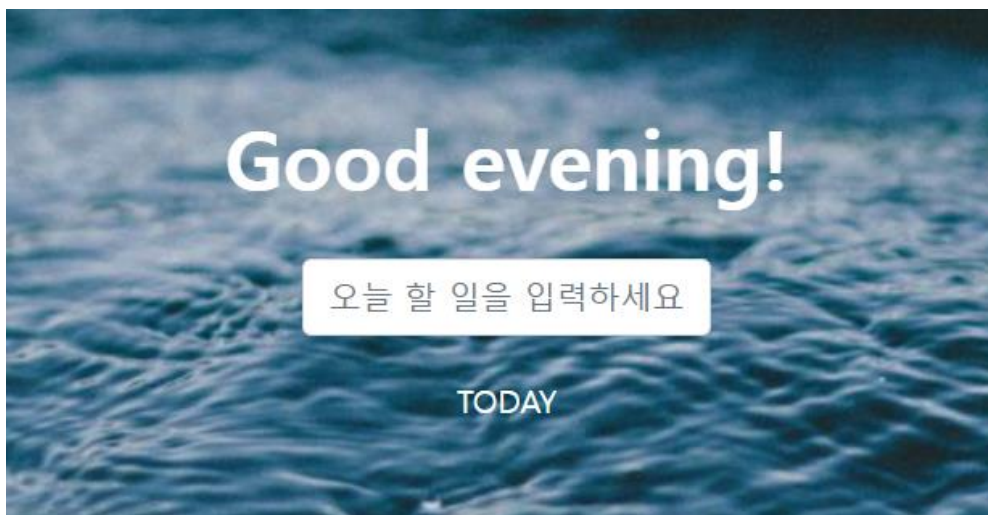
TODAY

5. 이미지 API 사용하기

A. axios를 통해 background를 받아오는 함수를 index.js에 작성한다.

```
const setRenderBackground = async () => {  
  const response = await axios.get("https://picsum.photos/1280/720", {  
    responseType: "blob", // 이미지, 음성 등의 바이너리 데이터  
  });  
  // 해당 이미지가 가지는 url 가져옴  
  const imageURL = URL.createObjectURL(response.data);  
  document.querySelector("body").style.backgroundImage = `url(${imageURL})`;  
};  
  
setRenderBackground();
```

위 함수를 실행하면 이미지가 나타나게 된다.



폰트 컬러만 흰색으로 바꿔주었다.

B. 5초 간격으로 이미지가 변경되게 만들기

setInterval 함수를 활용해서 setRenderBackground를 5초마다 호출한다.

```
setRenderBackground();
setInterval(()=>{
  setRenderBackground();
}, 5000);
```

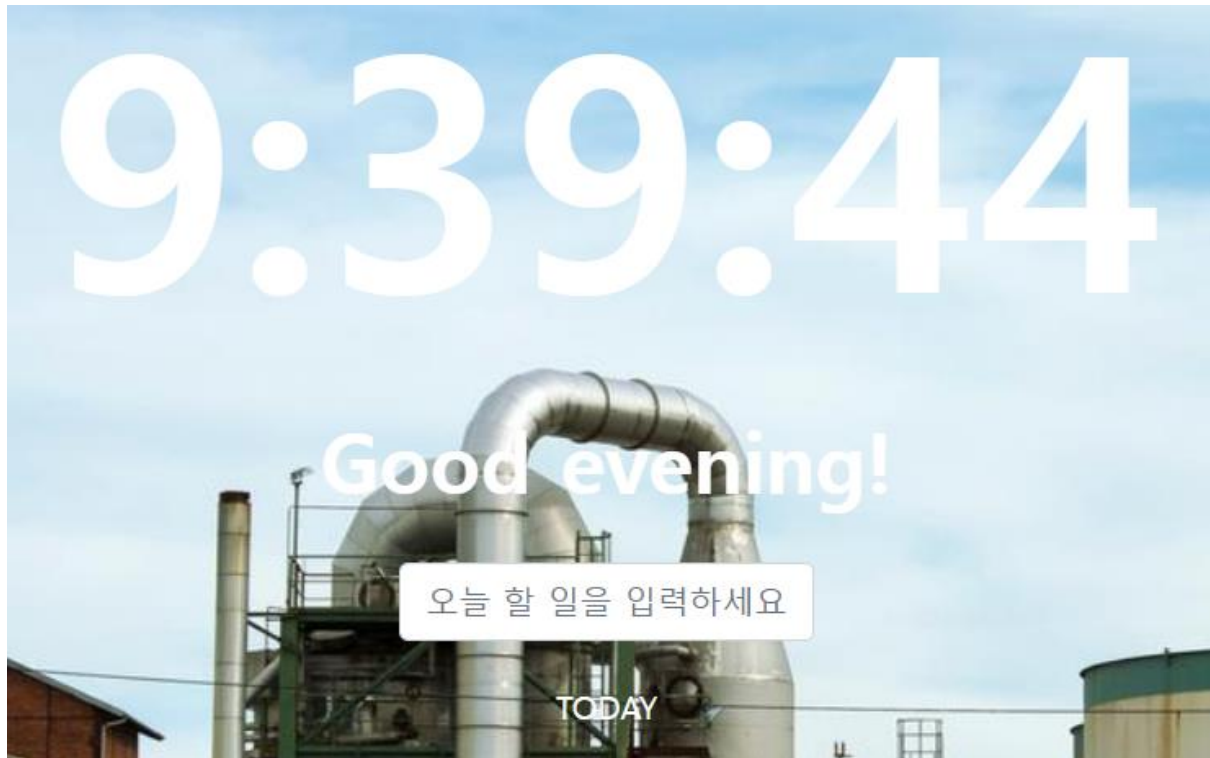
6. 시계 사용하기

1초 간격으로 시계가 갱신되는 함수를 작성한다.

```
// 1초 간격으로 시간 바꿈
const setTime = () => {
  const timer = document.querySelector(".timer");
  setInterval(() => {
    const date = new Date();
    timer.textContent = `${date.getHours()}:${date.getMinutes()}:${date.getSeconds()}`;
  }, 1000);
};
```

///

```
// 시간 설정
setTime();
```

시계 폰트컬러만 white 로 지정했다.

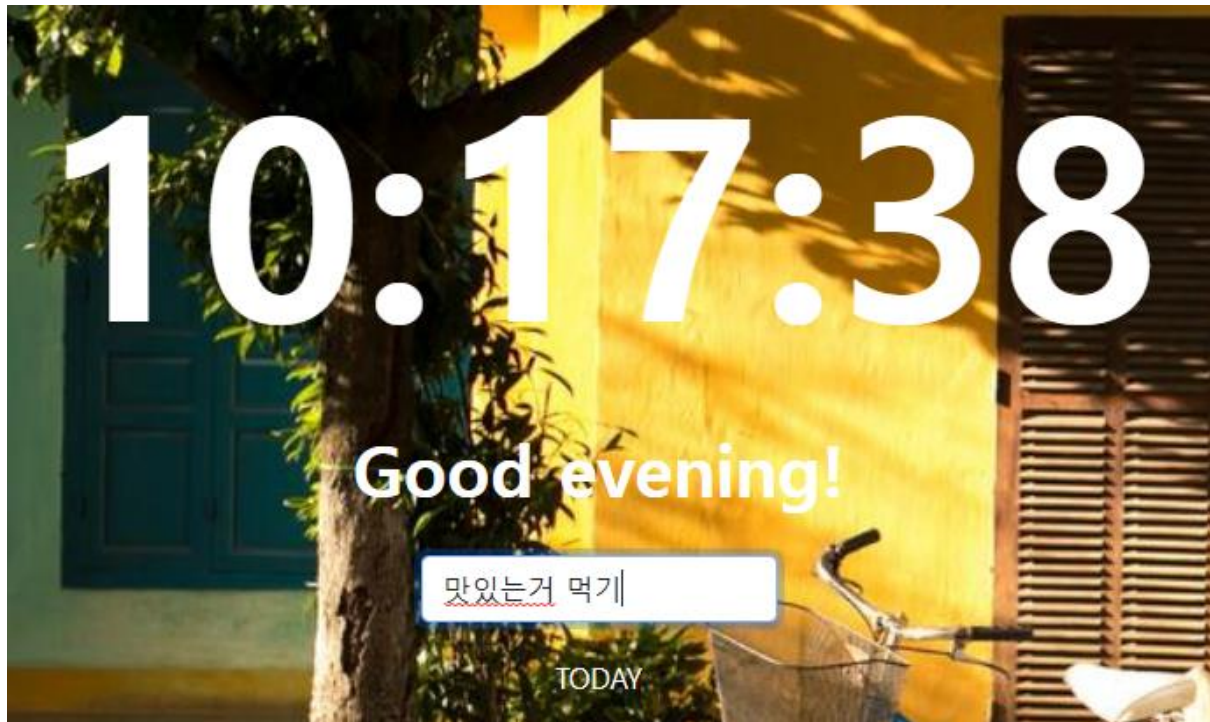
7. localStorage로 memo 작성

localStorage는 브라우저 상에 데이터를 저장 할 수 있는 웹 스토리지의 일종이다.

A. memo를 저장하는 setMemo 작성하기

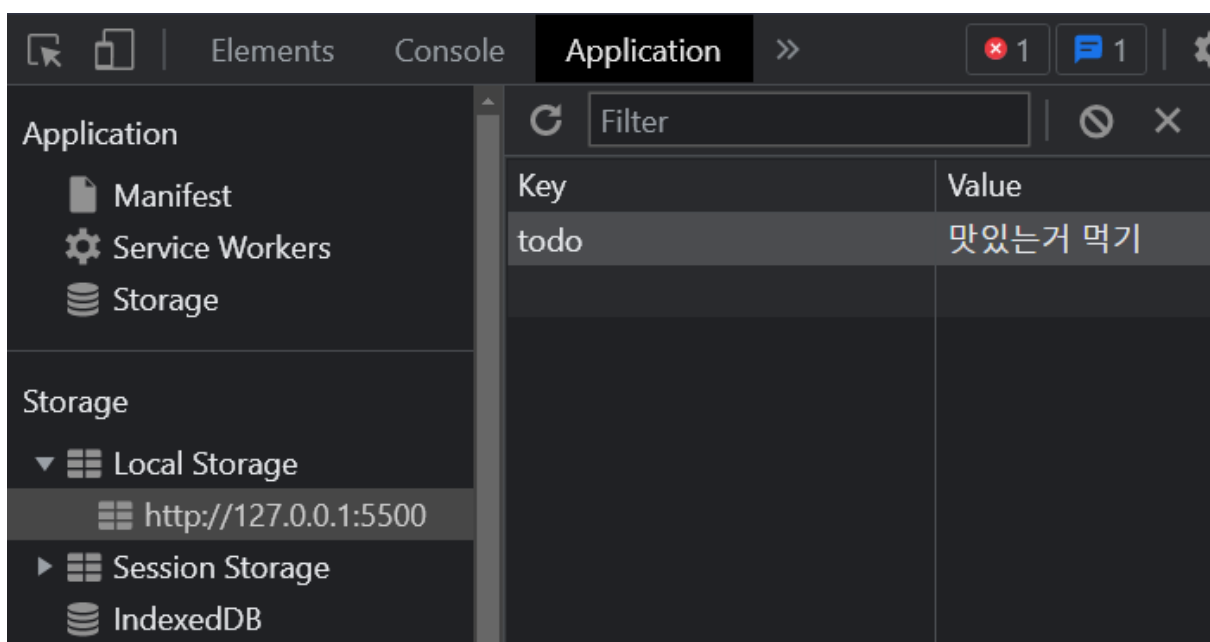
```
const setMemo = () => {  
  memoInput = document.querySelector(".memo-input");  
  memoInput.addEventListener("keyup", (evt) => {  
    // 엔터이면서, 뭐라도 값이 들어있다면  
    if (evt.code === "Enter" && evt.currentTarget.value) {  
      // localStorage에 key:todo, value: e.curentTarget.value 저장  
      localStorage.setItem("todo", evt.currentTarget.value);  
      // 인풋창 깔끔하게 지워줌  
      memoInput.value = "";  
    }  
  });  
};
```

setMemo를 작성한 후 호출해준다.



input란에 test를 입력해 주었다.

Enter를 쳐서 localStorage에 등록 한 후 개발자도구(F12) -> Application -> Local Storage에 접속한다.



방금 입력한 test가 정상적으로 저장되어있으며 새로고침해도 정상적으로 localStorage에 todo 값이 존재한다.

즉, 캐시 비우기 및 강력 새로고침을 해도 남아있다.

B. memo를 가져오는 getMemo 작성하기

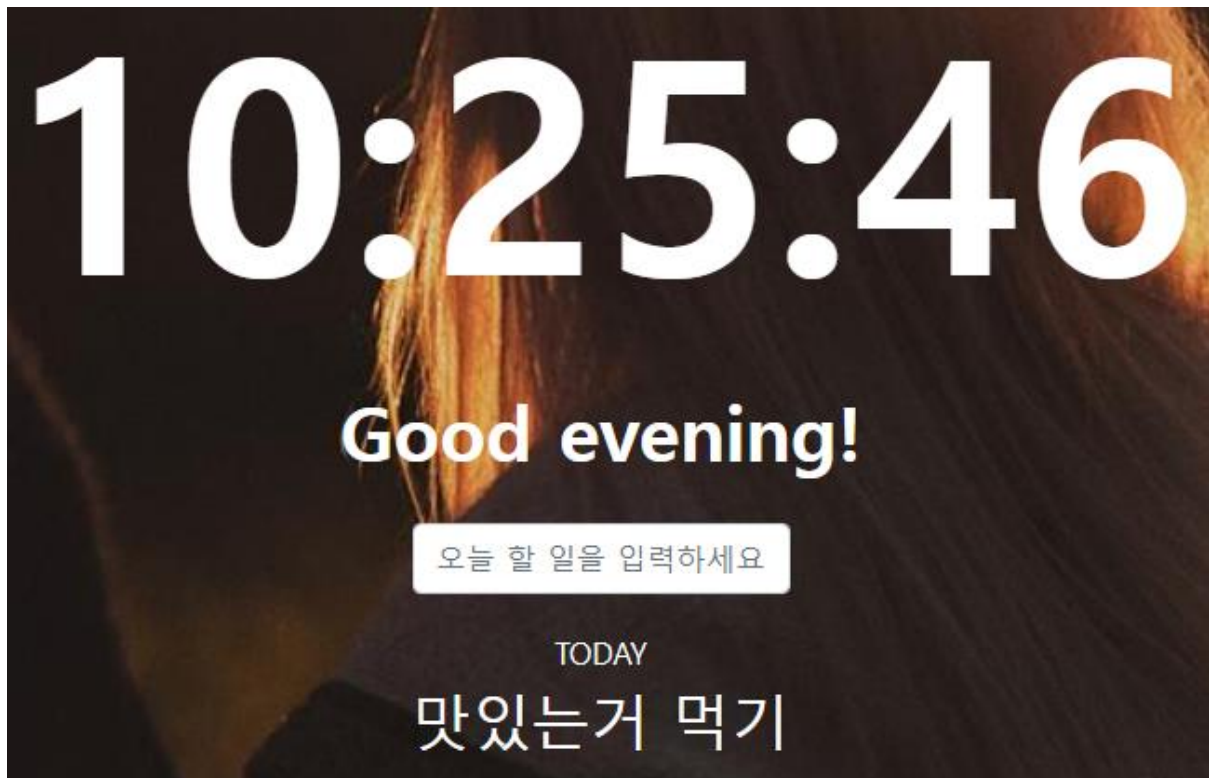
localStorage에 저장되어있는 todo를 가져와서

memo div에 값을 추가해준다.

```
// 로컬스토리지에서 메모 가져와서 붙임
const getMemo = () => {
  const memo = document.querySelector(".memo");
  const memoValue = localStorage.getItem("todo");
  memo.textContent = memoValue;
}
```

getMemo 는 setMemo 안에서 실행한다.

```
// 로컬스토리지에 메모 저장
const setMemo = () => {
  memoInput = document.querySelector(".memo-input");
  memoInput.addEventListener("keyup", (evt) => {
    // 엔터이면서, 뭐라도 값이 들어있다면
    if (evt.code === "Enter" && evt.currentTarget.value) {
      // localStorage에 key:todo, value: e.curentTarget.value 저장
      localStorage.setItem("todo", evt.currentTarget.value);
      // 메모 가져와서 붙임.
      getMemo();
      // 인풋창 깔끔하게 지워줌
      memoInput.value = "";
    }
  });
};
```



글자 white로 지정하고, memo-title 을 가운데정렬했다.

정상적으로 localStorage의 값을 가져온다.

C. memo를 삭제하는 deleteMemo 작성하기

localStorage의 데이터를 지움과 동시에 html 태그의 값 또한 비워주는 deleteMemo를 작성한다.

```
const deleteMemo = () => {  
  document.addEventListener("click", (evt) => {  
    if (evt.target.classList.contains("memo")) {  
      // 로컬에 아이템 지우기  
      localStorage.removeItem("todo");  
      // 글 내용 지우기  
      evt.target.textContent = "";  
    }  
  });  
};
```

왜 조건문을 넣었을까? 이벤트 버블링때문이다.

이벤트는 부모로 올라가기때문에, memo 클래스를 만날때만 실행한다.

해당 memo를 클릭하는 경우 메모가 지워지게 된다.

////

```
// 시간 설정  
setTime();  
// 메모 로컬스토리지에 등록 및 화면에 붙이기  
setMemo();  
// 메모 로컬스토리지에서 삭제 및 화면에서 지우기  
deleteMemo();
```

마찬가지로 함수실행해준다.

8. Modal 태그 추가하기

Modal을 클릭하면 날씨 정보를 가져올 수 있게하는 Modal Tag를 구성한다.

<https://getbootstrap.com/docs/5.1/components/modal/>

Static backdrop

When backdrop is set to static, the modal will not close when clicking outside it. Click the button below to try it.

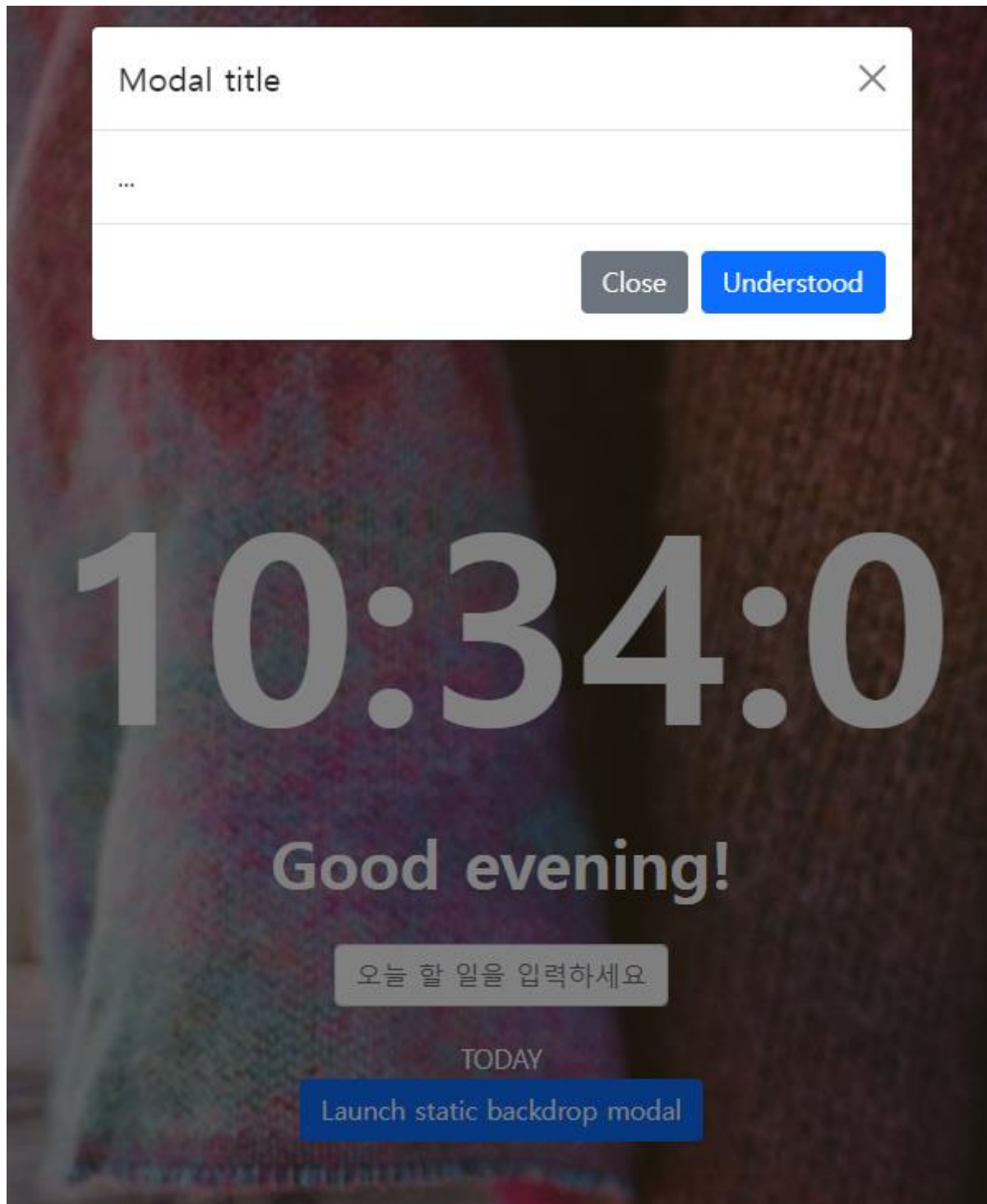
Launch static backdrop modal

Copy

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#staticBackdrop">
  Launch static backdrop modal
</button>

<!-- Modal -->
<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-keyboard="false" tabinde:
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="staticBackdropLabel">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Understood</button>
      </div>
    </div>
  </div>
</div>
```

위의 modal로부터 해당 코드를 기반으로 가져와서 작성한다.



정상적으로 동작한다.

이제 Modal의 내용을 수정 해줄 것이다. 가져온 예제 소스를 분석해서, 나에게 맞게 바꿔보자.

일단, 버튼 태그는 `<div>` 로 바꾸고, 구름 모양 아이콘으로 설정할 것이다.

```

<!-- 모달 버튼 아이콘 -->
<div
  class="modal-button"
  data-bs-toggle="modal"
  data-bs-target="#staticBackdrop"
>
</div>

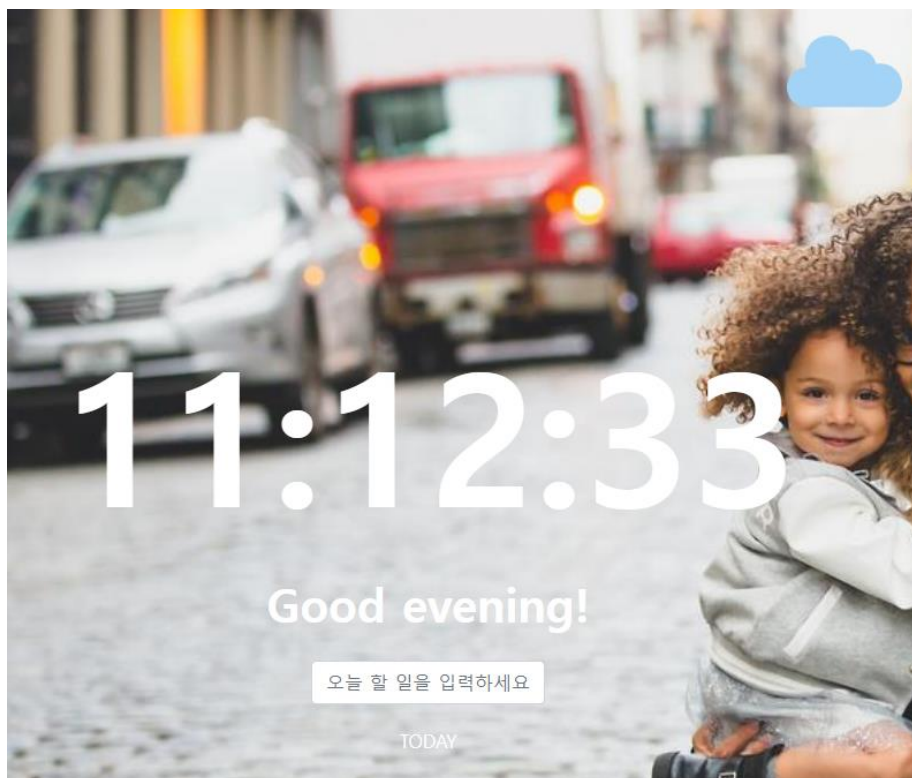
```

css 는 다음과 같이 작성했다.

```

.modal-button {
  position: fixed;
  right: 40px;
  top: 40px;
  background: url("../images/001-cloud.png");
  width: 100px;
  height: 100px;
  background-size: cover;
}

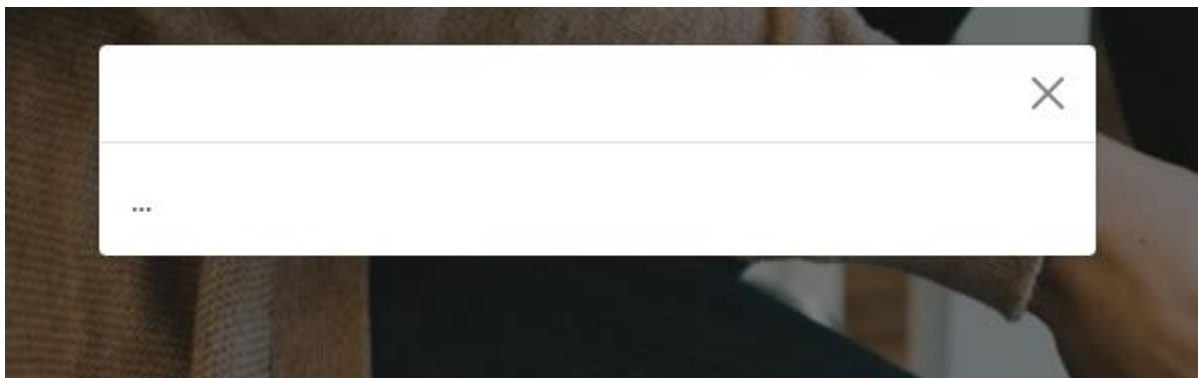
```




```

<!-- Modal -->
<div
  class="modal fade"
  id="staticBackdrop"
  data-bs-backdrop="static"
  data-bs-keyboard="false"
  tabindex="-1"
  aria-labelledby="staticBackdropLabel"
  aria-hidden="true"
>
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="staticBackdropLabel"></h5>
        <button
          type="button"
          class="btn-close"
          data-bs-dismiss="modal"
          aria-label="Close"
        ></button>
      </div>
      <div class="modal-body">...</div>
    </div>
  </div>
</div>

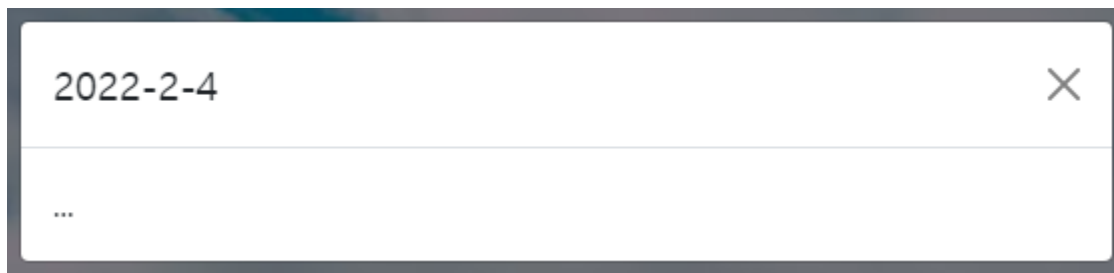
```



헤더엔 시간을 넣을 것이다. 함수를 만들어주자.

```
const setModalDate = () => {
  const modalDate = document.querySelector(".modal-date");
  const date = new Date();
  modalDate.textContent = `${date.getFullYear()}-${
    date.getMonth() + 1
  }-${date.getDate()}`;
};
```

함수 실행을 해주고, 결과를 확인해보자.

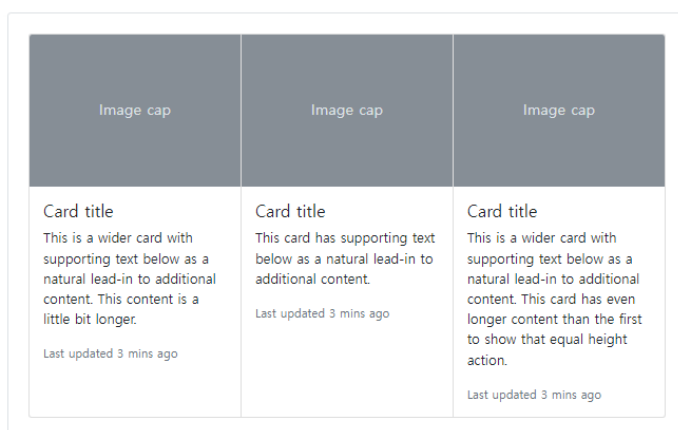


이제 이 안에 들어갈 내용을 카드로 꾸밀 것이다.

<https://getbootstrap.com/docs/5.0/components/card/>

Card groups

Use card groups to render cards as a single, attached element with equal width and height columns. Card groups start off stacked and use `display: flex;` to become attached with uniform dimensions starting at the `sm` breakpoint.



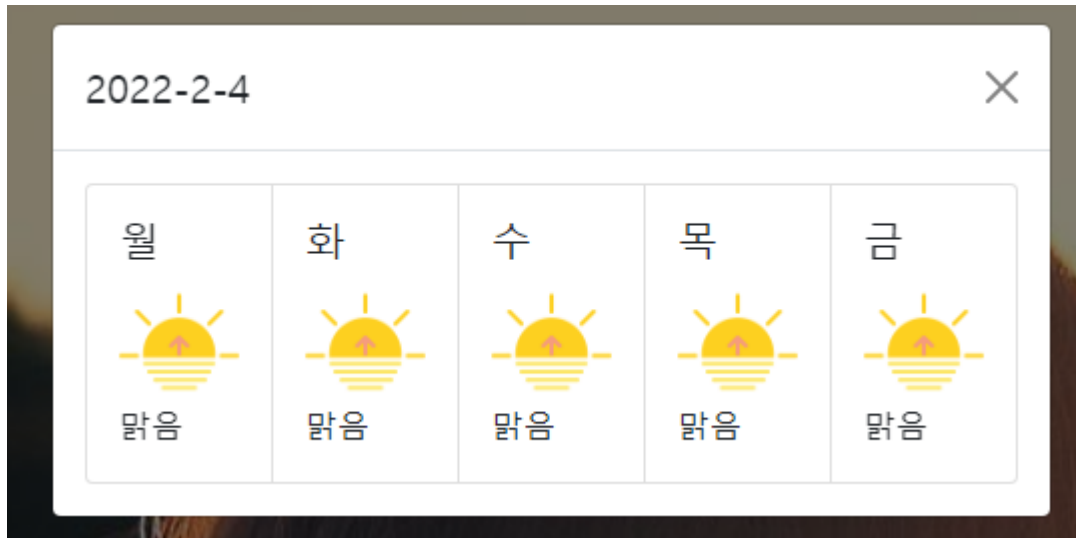
이게 적절해보이므로, 복사 붙여넣기 한 후 목적에 맞게 수정해주자.

```

<div class="modal-body">
  <div class="card-group">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">월</h5>
        
        <p class="card-text">맑음</p>
      </div>
    </div>
    <div class="card"> ...
  </div>
  <div class="card"> ...
</div>
  <div class="card"> ...
</div>
  <div class="card"> ...
</div>
  <div class="card"> ...
</div>
</div>
</div>

```

총 5일의 결과를 보여줄 것이므로, 똑같은 card 내용은 접어놨다.




9. 날씨 API 받아오기

A. 다양한 업체에서 날씨 API 제공한다. 제공해주는 사이트는 다음과 같다.

서비스 업체	URL	비고
OpenWeatherMap	openweathermap.org/api	해외 서비스
기상청	data.go.kr	공공데이터 사이트, 대기 필요
야후	developer.yahoo.com/weather	



B. OpenWeatherMap 대해서


날씨 데이터를 제공하는 영국 웹서비스 업체이며, 무료 서비스 및 유료 서비스를 제공한다. 우리는 무료 서비스를 이용하여, 즉시 Key를 발급받아 API를 쓸 수 있도록 준비를 할 예정이다.

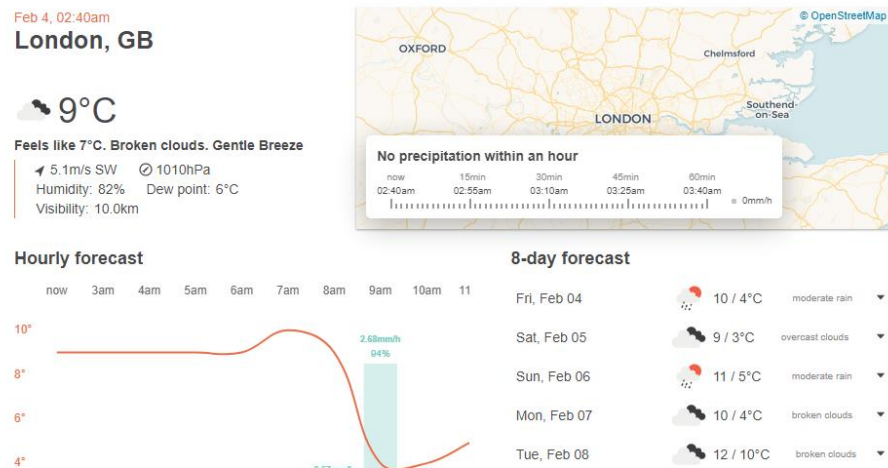

[Weather in your city](#)
[Guide](#)
[API](#)
[Dashboard](#)
[Pricing](#)
[Maps](#)
[Our Initiatives](#)
[Partners](#)
[Blog](#)
[Marketplace](#)
[Sign in](#)
[Support](#)

OpenWeather mobile app

A scientific yet simple approach to weather forecast.
Free. No ads.





C. OpenWeatherMap 사이트 접속하기

접속 후 API 메뉴로 들어가면 다음과 같은 화면을 볼 수 있다.

Weather API

[Home](#) / [Weather API](#)

Please, [sign up](#) to use our fast and easy-to-work weather APIs for free. In case your requirements go beyond our freemium account conditions, you may check the entire list of our [subscription plans](#). You can read the [How to Start](#) guide and enjoy using our powerful weather APIs right now.

Current & Forecast weather data collection

Current Weather Data

[API doc](#)

[Subscribe](#)

- Access current weather data for any location including over 200,000 cities
- We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations
- JSON, XML, and HTML formats
- Included in both free and paid subscriptions

Hourly Forecast 4 days

[API doc](#)

[Subscribe](#)

- Hourly forecast is available for 4 days
- Forecast weather data for 96 timestamps
- JSON and XML formats
- Included in the Developer, Professional and Enterprise subscription plans

One Call API

[API doc](#)

[Subscribe](#)

- Make one API call and get current, forecast and historical weather data
- **Minute forecast** for 1 hour
- **Hourly forecast** for 48 hours
- **Daily forecast** for 7 days
- **Historical data** for 5 previous days
- **National weather alerts**
- JSON format
- Included in both free and paid subscriptions

Daily Forecast 16 days

[API doc](#)

[Subscribe](#)

- 16 days forecast is available for any location on the globe
- 1-day step for 16 days
- JSON and XML formats
- Included in all paid subscription plans

Climatic Forecast 30 days

[API doc](#)

[Subscribe](#)

- Forecast weather data for 30 days
- JSON format
- Included in the Developer, Professional and Enterprise subscription plans

Bulk Downloading

[API doc](#)

[Subscribe](#)

- You may request current weather and forecasts in bulk with a variable data granulation
- Current weather bulk is available for 209,000+ cities
- Variety of hourly and daily forecast bulks depends on the frequency of data collection

여기서 무료로 One Call API 서비스 / Current Weather Data 서비스와 5 Day / 3 hour Forecast 서비스를 사용할 수 있다. 우리는 이 중에서 Current Weather Data 서비스와 5 Day 3 hour Forecast 서비스만을 이용할 예정이다. 각 서비스는 다음과 같은 특징을 갖는다.

i. Current Weather Data

1. 대부분의 전세계의 도시의 현재 날씨 정보를 받아온다.
2. JSON / XML / HTML 형태로 정보를 얻을 수 있다.

ii. 5 Day 3 Hour Forecast

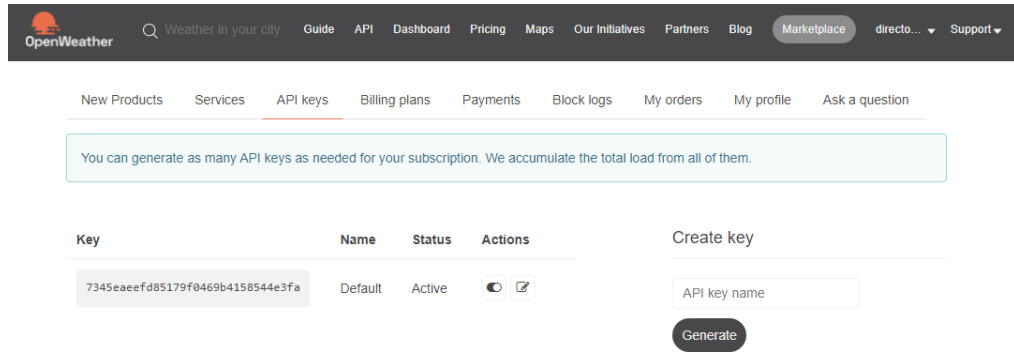
1. 대부분의 지역에서 5일간의 날씨 정보를 받아온다.
2. 각 Day별 3 시간 단위로 날씨 예측 데이터를 가져올 수 있다.

iii. One Call API

1. 한번 호출에 하나의 정보를 얻을 수 있다.
2. 분별 / 시간별 / Day별 / 지난 5일간 단위로 날씨를 가져올 수 있다.

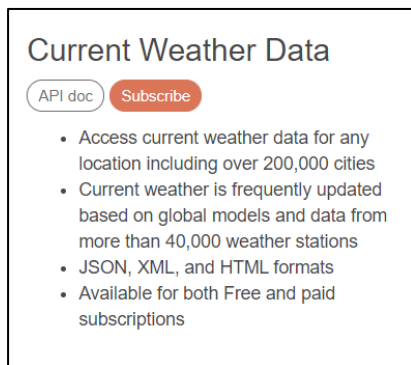
D. OpenWeather 로그인 후 개인 API Key 발급

이 키를 포함시켜 API를 호출해야 한다. 따라서 이 API Key는 필수이다.



E. OpenWeather API 사용방법

API 메뉴에서, "API doc"를 눌러보자.



API를 호출하는 방법이 안내 되어있다.

Current weather data

Access current weather data for any location on Earth including over 200,000 cities! We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations. Data is available in JSON, XML, or HTML format.

Call current weather data

How to make an API call

API call

```
api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}
```

이중 lat lon은 위도 경도를 뜻한다. 이곳에 자신이 있는 위도와 경도를 입력하면, 그 곳의 날씨를 얻을 수 있다.

서울은 위도(Latitude) : 북위 37.57도, 경도(Longitude)는 동경 126.98 이다.

10.Axios로 날씨 테스트하기

A. Axios 로 서울 날씨 정보 Console창 출력하기

```
// 서울의 위도와 경도
const lat = 37.57;
const lon = 126.98;
// openweathermap API Key;
const apiKey = 'XXXXXXXXXXXX';

axios
  .get(
    `https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${apiKey}`
  )
  .then((res) => console.log(res));
```

```
// 서울의 위도와 경도
const lat = 37.57;
const lon = 126.98;
// openweathermap API Key;
const apiKey = "aaaaaaaaaaaa";

axios
  .get(`https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${apiKey}`)
  .then((res) => console.log(res));
```

출력결과는 다음과 같다.

```
▼ {data: {...}, status: 200, statusText: 'OK', headers: {...}, config: {...}, ...} ⓘ
  ► config: {transitional: {...}, transformRequest: Array(1), transformResponse: Array(1), timeout: 0, withCredentials: false, ...}
  ▼ data:
    base: "stations"
    ► clouds: {all: 3}
    cod: 200
    ► coord: {lon: 126.98, lat: 37.57}
    dt: 1643947592
    id: 1835848
    ► main: {temp: 272.42, feels_like: 266.75, temp_min: 270.44, temp_max: 272.92, pressure: 1013}
    name: "Seoul"
    ► sys: {type: 1, id: 5509, country: 'KR', sunrise: 1643927617, sunset: 1643965096}
    timezone: 32400
    visibility: 10000
    ► weather: [{...}]
    ► wind: {speed: 6.04, deg: 297, gust: 10.18}
    ► [[Prototype]]: Object
  ► headers: {content-length: '512', content-type: 'application/json; charset=utf-8'}
  ► request: XMLHttpRequest {onreadystatechange: null, readyState: 4, timeout: 0, withCredentials: false, ...}
    status: 200
    statusText: "OK"
  ► [[Prototype]]: Object
```


11. 날씨 정보 시각화

먼저 Modal 내의 Card body 내부의 부분들을 주석처리해준다.

```
<div class="modal-body">
  <div class="card-group">
    <!-- <div class="card">
      <div class="card-body">
        <h5 class="card-title">월</h5>
        
        <p class="card-text">맑음</p>
      </div>
    </div>
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">화</h5>
        
      </div>
    </div>
  </div>
</div>
```

A. 날씨 정보에 따라 아이콘을 매칭해주는 matchIcon 함수를 제작한다

```
const matchIcon = (weatherDate) => {
  if (weatherDate === "Clear") {
    return "./images/039-sun.png";
  }
  if (weatherDate === "Clouds") {
    return "./images/001-cloud.png";
  }
  if (weatherDate === "Rain") {
    return "./images/003-rainy.png";
  }
  if (weatherDate === "Snow") {
    return "./images/006-snowy.png";
  }
  if (weatherDate === "Thunderstorm") {
    return "./images/008-storm.png";
  }
  if (weatherDate === "Drizzle") {
    return "./images/033-hurricane.png";
  }
};
```

만들어만두고 실행하진 않는다.

B. 날씨를 가져오는 getWeather 함수를 제작한다.

```
const getWeather = async (lat, lon, apiKey) => {  
  const data = await axios.get(  
    `https://api.openweathermap.org/data/2.5/forecast?lat=${lat}&lon=${lon}&appid=${apiKey}`  
  );  
  return data;  
};
```

아까 테스트한 axios 를 함수를 따로 만들어서 넣은 것인데, 이젠 5 Day / 3 Hour Forecast API 를 사용할 것이다.

C. 데이터에 따라 동적으로 변화하는 컴포넌트를 생성해 준다.

```
const weatherWrapperComponent = (li) => {  
  // 현재 절대온도인데, 섭씨로 바꿔준다.  
  // 소수 1자리까지  
  const changeToCelsius = (temp) => (temp - 273.15).toFixed(1);  
  return `  
    <div class="card">  
      <div class="card-header">${li.dt_txt.split(" ")[0]}</div>  
      <div class="card-body">  
        <h5 class="card-title">${li.weather[0].main}</h5>  
          
        <p class="card-text">${changeToCelsius(li.main.temp)}°</p>  
      </div>  
    </div>`;  
};
```

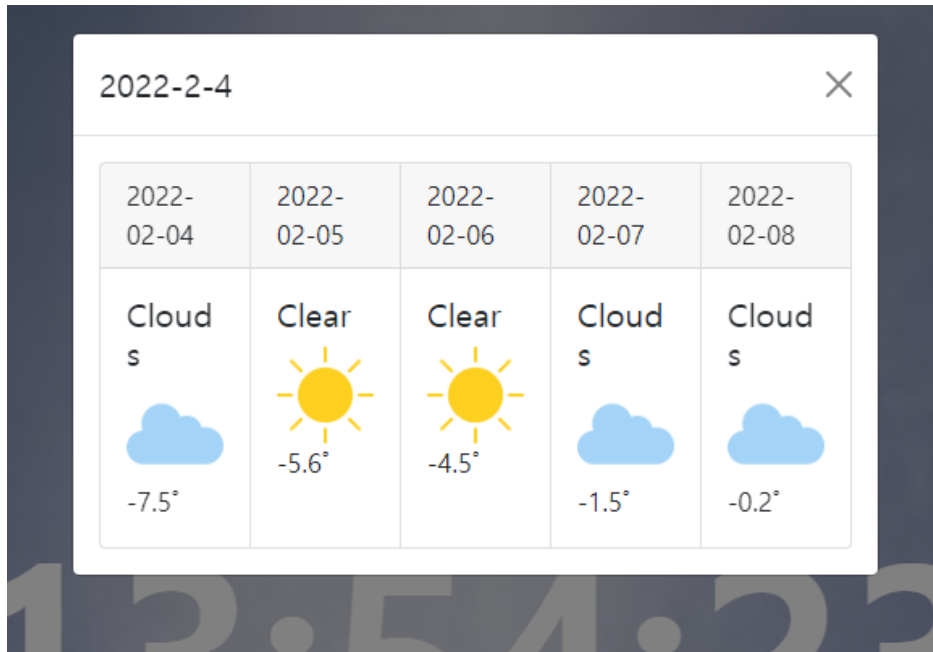
D. 컴포넌트를 기반으로 날씨를 그려내는 renderWeather 함수를 제작한다

```

const renderWeather = async (lat, lon, apiKey) => {
  const weatherResponse = await getWeather(lat, lon, apiKey);
  const weatherData = weatherResponse.data;
  console.log(weatherData);
  const weatherList = weatherData.list.reduce((total, cur) => {
    if (cur.dt_txt.indexOf("18:00:00") > 0) {
      total.push(cur);
    }
    return total;
  }, []);
  // 10개의 배열이 출력된다
  console.log(weatherList);
  const cardGroup = document.querySelector(".card-group");
  cardGroup.innerHTML = weatherList
    .map((li) => {
      return weatherWrapperComponent(li);
    })
    .join("");
};

```

최종적으로 만들어진 결과물은 다음과 같다.



CSS 를 조정하면 훨씬 깔끔해질 것이다.

12. 심화과제

- A. 시간에 따라 evening, morning 나눠 보여주기
 - i. 현재 시간을 기반으로 오전이면 Good morning
 - ii. 오후면 Good evening을 출력하기
- B. 오른쪽 상단의 Modal Button 날씨 이미지를 현재 날씨로 변경하자.
 - i. javascript를 활용해서 현재 위치를 기반으로 Modal Button의 이미지를 현재 날씨로 변경하게 만들기
- C. 기존에 만들었던 포트폴리오 웹사이트에 메뉴를 추가하여, 포트폴리오 웹사이트에서 지금까지 제작한 날씨 웹서비스 링크를 걸어둔다. **(차후 포트폴리오 활용을 위함)**