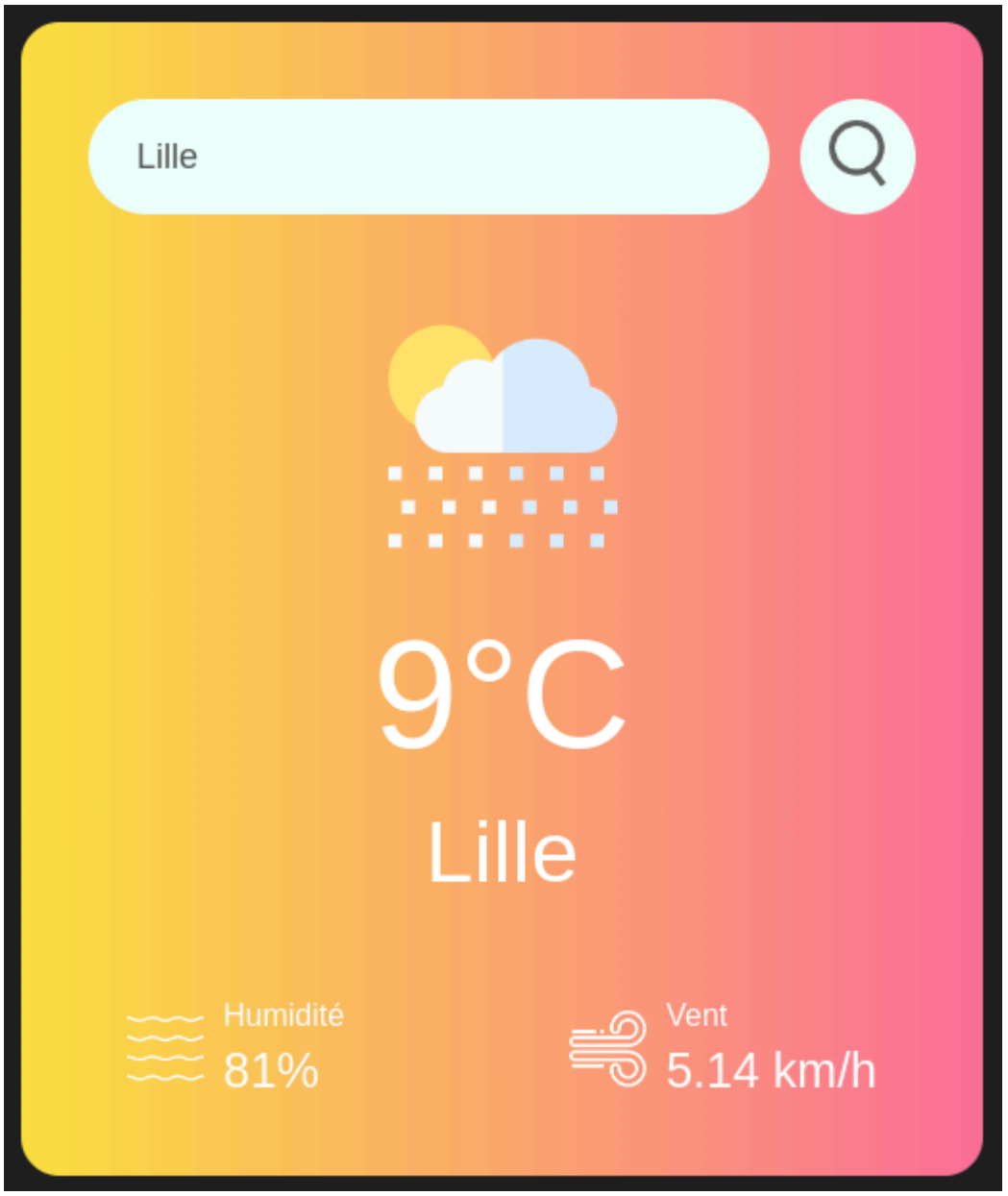
Sujet Web Météo - Stage de Seconde

Bonjour à tous et bienvenue dans cet atelier de développement Web ! Aujourd'hui nous allons réaliser une page web dans laquelle nous pourrons afficher la météo d'une ville ou d'un pays à l'aide d'une clé API. Pour cela nous utiliserons le langage de balisage HTML, le CSS ainsi que le JavaScript. Voici un petit avant-goût du résultat final que vous pourriez obtenir à la fin de cet atelier :



1ère étape – Préparation

Dans le fichier Atelier météo, qu'il faut dézipper, vous trouverez un fichier appelé *index.html* ainsi qu'un fichier appelé *style.css.* Ces fichiers sont ceux où nous écrivons le code du site météo. Pour commencer, vous allez devoir ouvrir ces deux fichiers avec un éditeur de code (comme Visual Studio Code par exemple).

Clé API

Pour avoir la clé API (qui vous servira à avoir les informations météorologiques des pays/villes) vous allez devoir vous rendre sur le site https://openweathermap.org. Une fois sur le site, vous allez devoir créer un compte. Une fois le compte créé et votre adresse mail validée, vous pouvez vous rendre sur l'onglet *API keys*. Vous trouverez ici votre clé API, copier la quelque part, elle sera utile plus tard.

Live server

Sur Visual Studio, il est fortement conseillé d'installer l'extension *Live server*, celle-ci vous servira à voir votre site s'actualiser à chaque changement que vous faites. Pour cela, cliquez sur ce petit logo à gauche de votre écran.

Une fois sur cet onglet, il vous suffira de taper Live Server et de l'installer. Ensuite, pour lancer le live server, il vous suffit de faire clique droit sur le fichier HTML, et de sélectionner ouvrir avec Live server.

2ème étape – Onglet de recherche

1- Couleur de fond de notre page web

Tout d'abord, commençons par changer la couleur du fond de notre page web. Pour changer le style d'un élément, il faut se rendre dans le fichier css, puis mettre le nom de l'élément à styliser et de ensuite rajouter des accolades ({ }). Les éléments de style sont ensuite à mettre entre ces accolades. Pour changer la couleur de fond du site, c'est l'élément *body* que nous devons styliser. On obtiendra ce code par exemple (choisissez la couleur que vous souhaitez, ce site https://htmlcolorcodes.com/fr/ vous sera utile !) :

```
*{
    margin: 0;
    padding: 0;
    font-family: 'Poppins', sans-serif;
    box-sizing: border-box;
}
body{
    background: #222;
}
```

2- Carte météo

Nous allons maintenant créer la *carte* dans laquelle nous mettrons tous nos éléments. Pour cela, nous allons devoir créer une * * (élément de division du contenu, il n'a pas d'effet sur la mise en page tant qu'il n'est pas modifié dans le fichier CSS) à l'intérieur de notre *<body>* dans le fichier html.

Attention, lorsque vous créez une * * il faut bien faire attention à la fermer comme ceci : * *. La plupart des éléments du html ont besoin d'être fermés en ajoutant un "/" devant leurs noms.

Il faudra ajouter une *classe* (attribut qui permet de manipuler et styliser des éléments) "card" à cette division ce qui nous permettra de la modifier dans le fichier css. Cette * * de classe "card" contiendra tous les sous éléments visuels que nous afficherons sur notre page web.

```
<div class="search">
</div>
```

Il est maintenant temps de créer la barre de recherche. A l'intérieur de la div que vous venez de créer, il va falloir créer une nouvelle division de classe "search". Dans cette div, il va falloir créer un *<input>* de type *text* (entrée de données).

```
<input type="text" placeholder="Entrez une localisation" spellcheck="false">
```

Il va aussi falloir créer (toujours à l'intérieur de la * * "search" un bouton (*<button>)* et une image (*<img>)*. Pour l'image, l'élément *<img>* doit se trouver dans l'élément *<button>* , la source *src* (qui permet de sélectionner une image du dossier *images*) que vous devez attribuer à votre image est *search.png* dans le dossier images.

```
<button><img src="images/search.png"></button>
```

3- Le style

Il est maintenant temps de styliser tout ça dans le fichier css. Ce site : https://www.w3schools.com/cssref/index.php répertorie tout les éléments de style en css, je vous conseille de le feuilleter un peu afin de trouver des éléments avec des propriétés intéressantes.

Nous allons commencer par styliser l'élément de classe "card ". Comme la plupart d'entre vous n'ont jamais fais de css, le style vous sera donné, mais gardez en mémoire que vous êtes libre et que vous pouvez le modifier à votre guise (vous pouvez par exemple chercher et trouver un autre linear-gradient).

```css
.card{
    width: 90%;
    max-width: 500px;
    background: linear-gradient(90deg, #FEE140 0%, #FA709A 100%);
    color: #fff;
    margin: 100px auto 0;
    border-radius: 20px;
    padding: 40px 35px;
    text-align: center;
}
```

Voici le code css pour le *search input, search button* et *search button img :*

```css
.search input{
    border: 0;
    outline: 0;
    background: #ebfffc;
    color: #555;
    padding: 10px 25px;
    height: 60px;
    border-radius: 30px;
    flex: 1;
    margin-right: 16px;
    font-size: 18px;
}
.search button{
    border: 0;
    outline: 0;
    background: #ebfffc;
    border-radius: 50%;
    width: 60px;
    height: 60px;
    cursor: pointer;
}
.search button img{
    width: 30px;
}
```

Regardez votre page web. Comme vous pouvez le voir, le bouton et l'input ne sont pas alignés, c'est à vous de modifier la classe *search* (qui englobe ces éléments) afin que les éléments soient mieux placés.

3ème étape – Message d'erreur

Nous allons désormais nous occuper de la gestion d'erreur dans le fichier HTML, nous allons devoir afficher un message d'erreur lorsque la localisation recherchée n'existe pas. Pour ça rien de plus simple, il suffit de créer une * * de classe "error" à l'intérieur de notre *card* et d'y ajouter un message d'erreur (selon vos goûts) avec l'élément paragraphe *<p>.* Il est

important que dans le fichier css ce message d'erreur ne soit pas affiché de base sur la page internet (renseignez-vous sur l'élément de style display !)

Avec cette étape finie, vous devriez pour l'instant avoir quelque chose ressemblant à ça :

4ème étape – Section météo

Il est maintenant temps d'ajouter une * * avec une classe "weather" dans le fichier HTML toujours à l'intérieur de notre classe principale *card* pour afficher les détails météorologiques.

À l'intérieur, incluez une image *<img>* de classe "weather_icons" (choisissez n'importe laquelle dans le dossier images) pour les icônes météo. Ajoutez ensuite la température et le nom de la ville avec les éléments titre *<h1>* ayant pour classe "temp" et *<h2>* ayant pour classe "city".

(Les éléments *<h1>* et *<h2>* sont simplements des titres qui s'affichent sur la page web).

Vous pouvez pour le moment mettre la température que vous voulez et le nom de ville que vous voulez à l'intérieur des éléments que vous venez de créer afin d'avoir une aperçue du rendu.

5ème étape – Détails (vent et humidité)

Il va falloir maintenant ajouter une sous division * * de classe "details" (à l'intérieur de notre * * de classe "weather") destinée à afficher les détails supplémentaires (attention, il est facile de se perdre lors de cette étape, prenez bien le temps de lire le sujet et de comprendre quel élément se trouver à l'intérieur duquel).

Dans celle-ci il faut à nouveau créer une sous division * * (donc à l'intérieur de notre * * de classe "details") de classe "col" qui représente l'humidité. Dans celle-ci, il faut afficher l'image représentant l'humidité. Encore une fois il faut faire une sous division * * qui contiendra deux paragraphes *<p>* , l'un affichant le mot humidité, l'autre le pourcentage d'humidité (choisissez un pourcentage aléatoire pour le moment et donnez à cet élément la classe "humidity".)

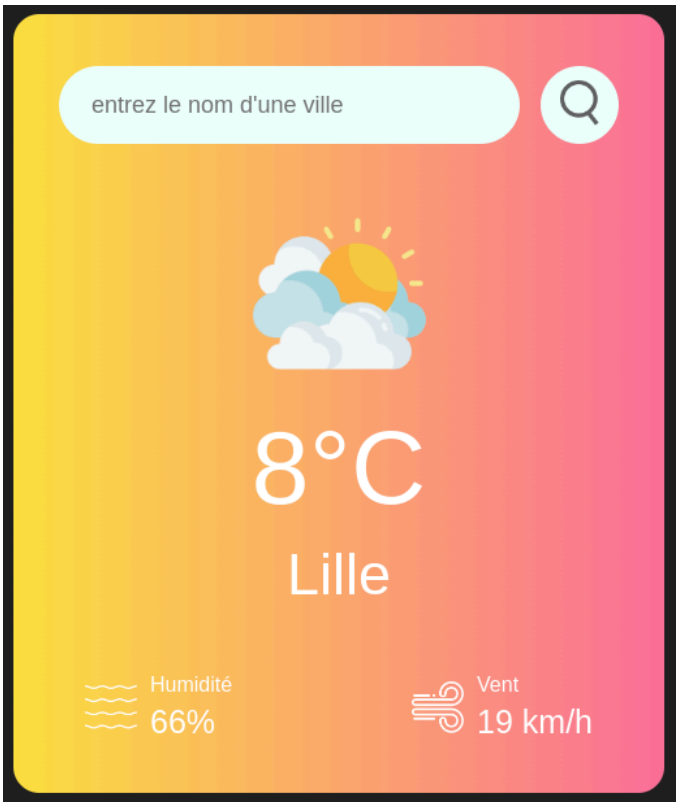
Vous devez ensuite reproduire ces étapes (à partir de la * * de classe "col" pour faire la même chose mais avec le vent (la première sous division doit aussi avoir comme classe "col" et la classe du deuxième paragraphe *<p>* aura comme classe "wind").

Une fois tout cela fait, vous pouvez styliser les éléments avec ce code :

```css
.weather_icons{
   width: 170px;
   margin-top: 30px;
}
.weather h1{
   font-size: 80px;
   font-weight: 500;
}
.weather h2{
   font-size: 45px;
   font-weight: 400;
   margin-top: 10px;
}
.details{
   display: flex;
   align-items: center;
   justify-content: space-between;
   padding: 0 20px;
   margin-top: 50px;
}
.col{
   display: flex;
   align-items: center;
   text-align: left;
}
.col img{
   width: 40px;
   margin-right: 10px;
}
```

```css
.humidity, .wind{
    font-size: 25px;
    margin-top: 5px;
}
```

La partie affichage est désormais terminée ! Votre site affiche désormais toutes les informations nécessaires et devrait ressembler à ça :



6ème étape – Script JavaScript (dans le fichier HTML)

Bravo ! Il est maintenant temps de s'attaquer au script pour pouvoir afficher les bonnes informations en fonction de la location. Pour commencer il va falloir créer un élément *<script>* dans lequel nous écrirons notre code.

Ensuite, il est temps d'initialiser notre clé API, pour cela il vous suffit de créer une variable *const* ayant pour nom apiKey et de lui assigner votre clé.

```
const apiKey = "67b44daef2220e6ea2141f608b1feac8";
```

De plus, il nous faut une variable contenant l'url de l'API (attention ce n'est pas un lien il est inutile de cliquer dessus):

```
const apiURL =
"https://api.openweathermap.org/data/2.5/weather?units=metric&q=";
```

Il va maintenant falloir créer 3 variables pour les éléments HTML avec lesquels nous allons interagir :
-l'input
-le bouton de recherche
-les icônes météos
Pour cela nous allons utiliser ce format :

const {nom_de_variable} = document.querySelector("{nom_élément}") ;

Voici par exemple à quoi ressemble la variable correspondant à l'input :

```
const searchBox = document.querySelector(".search input");
```

C'est à vous de trouver les bon noms pour les 2 const restants (indice : c'est en rapport avec le nom des classes que vous avez donné à ces éléments!).
Donnez comme nom à la variable pour le bouton const searchButton et pour la variable pour les icônes météos const weatherIcons.

7ème étape – Gestion d'erreur

C’est maintenant l’heure d’écrire la fonction qui nous servira a directement changer la valeur des informations que nous voulons afficher. Nous allons pour cela créer une fonction *check_weather* qui aura cette forme (faites attention à bien écrire le reste du script à l’intérieur des brackets :

```
async function check_weather(city){}
```

Pour commencer, nous avons tout d’abord besoin de regarder si nous ne recevons pas un code d’erreur par l’API:

```
const response = await fetch(apiURL + city + `&appid=${apiKey}`);
```

Une fois cette information stockée, nous devons regarder que le statut de la réponse est différent de 404 (erreur). Pour cela, il vous faudra utiliser un *if* qui comparera *response.status* (renseignez-vous sur google ou demandez de l’aide à un cobra!).

Dans le cas où nous recevons 404, nous allons avoir besoin d’afficher le message d’erreur que nous avons précédemment écrit et ne plus afficher le reste des éléments. Pour cela il faut utiliser ce code :

```
document.querySelector("{élément_à_afficher}").style.display = "block" ;
document.querySelector("{élément_à_ne_plus_afficher}").style.display = "{option_pour_ne_plus_afficher}" ;
```

8ème étape - Sélection de la bonne image

Dans le cas où on ne reçoit pas 404, nous devons changer les éléments en fonction de la localisation indiquée. Nous pouvons donc faire un *else* après la bracket ("}") qui ferme notre *if.*

Nous devons tout d’abord stocké les informations renvoyées par l’API dans une variable qu’on appellera *data* :

```
const data = await response.json();
```

Maintenant, nous avons besoin de voir la structure de ces informations afin de pouvoir sélectionner les bonnes informations, pour cela nous devons afficher le contenu de notre variable data dans la console à l'aide de la fonction console.log({élément à afficher}).

Pour pouvoir voir les éléments dans la console, il est nécessaire d'appeler notre fonction. Pour cela il va falloir écrire le nom de notre fonction à l'intérieur du *<script>* (mais en dehors de la fonction). Il suffit juste d'écrire le nom de notre fonctions ("check_weather()") en mettant entre les parenthèses le nom de la ville dont vous souhaitez afficher les informations, par exemple :

```
check_weather("Lille");
```

Une fois cela fait, vous allez devoir inspecter votre page web et aller dans l'onglet console afin de voir ou sont les informations dont nous allons avoir besoin.
Nous avons 4 éléments à modifier :
-la localisation
-la température
-l'humidité
-le vent
Pour les modifier, il faut ce code :

document.querySelector("{élément_à_modifier}").innerHTML = data.{information_à_trouver_dans_la_console} ;

Exemple pour la localisation :

```
document.querySelector(".city").innerHTML = data.name;
```

(Attention, pour la température, l'humidité et le vent, vous devez rajouter l'unité, pour cela il faut rajouter *+ "{unité}"* à la fin de la ligne de code, de plus il faut arrondir la température à l'aide de la fonction *Math.round*).

Une fois que vous avez fini cette étape, vous pouvez effacer l'appel de votre fonction, il ne nous sera plus utile.

Il est maintenant temps de changer l'icône en fonction de la condition météo. Nous allons devoir utiliser une condition *if* pour chacune des météos possible :
-Clouds
-Clear
-Drizzle
-Mist
-Rain
-Snow

Vos conditions *if* devront comparer l'élément *data.weather[0].main* avec les différentes météos possibles, si la condition est vraie, vous devrez attribuer à votre variable qui gère les icônes l'image correspondant à la météo.
(Attention, le nom de votre variable doit être succédé par *.src*).

Voici à quoi devrait ressembler vos *if* :

```
if (data.weather[0].main === "Clouds") {
   weatherIcons.src = "images/clouds.png";
}
```

De plus, après vos conditions, il sera nécessaire de changer le display de la classe *weather* (il va être nécessaire que la classe weather ne soit pas afficher de base) et *error* afin d'afficher l'un et de ne pas voir l'autre (souvenez-vous de comment vous aviez fait dans le cas ou c'était une erreur !). Bravo ! Notre fonction est désormais terminée et nous pouvons passer à la dernière étape !

Il ne nous manque désormais plus qu'à faire en sorte que notre fonction soit appelée dés que nous cliquons sur le bouton de recherche, pour cela il faut ajouter un écouteur d'évènement après votre fonction mais toujours dans le script :

```
searchButton.addEventListener("click", ()=>{
    check_weather(searchBox.value);
})
```

Félicitations !!! Votre site météo devrait désormais être totalement fonctionnel ! Si vous rencontrez le moindre souci, n'hésitez pas à demander de l'aide à un Cobra.

Pour aller plus loin vous pouvez essayer de vous même afficher d'autres éléments météorologiques fournis par l'API !