

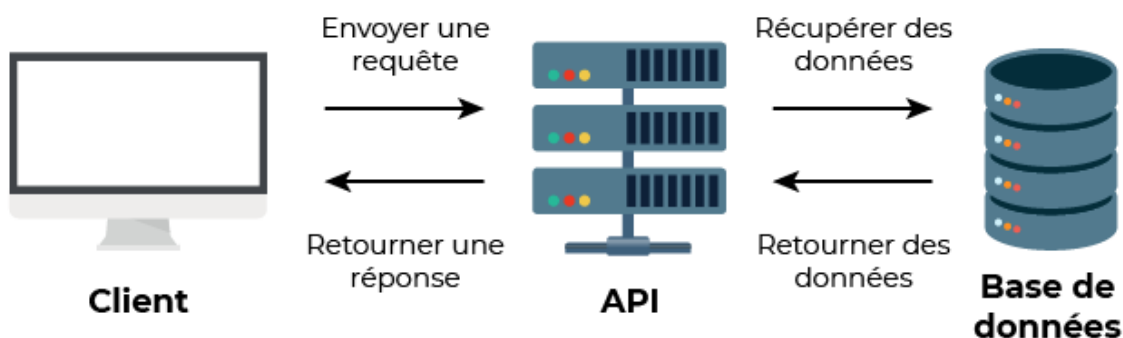
Info 21 - Chaîne météo avec Api

On se propose ici de découvrir les API. On utilisera 3 APIs différentes pour réaliser une page web météo. Après saisie d'une adresse d'un lieu quelconque sur la planète, la page :

- situe le lieu sur une carte google map,
- donne les informations météos actuelles sur ce lieu :
 - températures
 - ciel : soleil, nuageux, pluie, ...
 - heures levé et couché du soleil
 - ...

1. C'EST QUOI UNE API ?

API est un acronyme qui signifie « *Application Programming Interface* ». Une **API**, permet à un navigateur client de demander une information précise à un serveur, généralement par l'intermédiaire de l'URL (*méthode GET*). L'API va chercher cette information précise dans une base de données, puis la renvoie au client, généralement sous la forme d'un fichier au format JSON.



Une API permet en quelque sorte, à un client d'aller consulter une base de données sur un serveur tiers, sans avoir la maîtrise de cette base de données qui est entièrement contrôlée par ce serveur tiers.

2. EXEMPLE D'UNE API METEO :

Pour connaître les conditions météos sur n'importe quel lieu de la planète, on utilisera l'API [OpenWeather](#), [documentation](#). Cette API est généralement payante si le nombre d'envois est important.

Si on demande par exemple les conditions météos actuelles sur le lieu situé sur le point de latitude [45.75297164916992](#) et de longitude [4.791003227233887](#), lieu qui correspond aux coordonnées GPS du lycée Branly, on fait la requête suivante dans l'URL :

<https://api.openweathermap.org/data/2.5/weather?lat=45.75297164916992&lon=4.791003227233887&units=metric&lang=fr&appid=fed2662095a6145165167da7a942885f>

On reconnaît dans cette URL :

- <https://api.openweathermap.org/data/2.5/weather> : l'adresse du serveur tiers accompagnée de paramètres définissant le type de recherche demandée

- `lat=45.75297164916992&lon=4.791003227233887` : les valeurs des variables latitude et longitude du lieu pour lequel on souhaite avoir les infos
- `units=metric&lang=fr` : les valeurs des variables transmises pour définir les unités et la langue de la réponse
- `appid=fed2662095a6145165167da7a942885f` : la valeur de la variable qui identifie le compte créé sur le serveur, pour avoir accès à ce service de météo (compte lié à une solution de paiement)

⇒ Cliquer sur le lien précédent, ou copier-coller le dans la barre d'URL du navigateur.

En réponse, le serveur tiers renvoie le fichier JSON suivant :

```

JSON  Données brutes  En-têtes
Enregistrer Copier Tout réduire Tout développer
▼ coord:
  lon: 4.791
  lat: 45.753
▼ weather:
  ▼ 0:
    id: 800
    main: "Clear"
    description: "ciel dégagé"
    icon: "01d"
    base: "stations"
▼ main:
  temp: 18.78
  feels_like: 18.23
  temp_min: 17.3
  temp_max: 19.78
  pressure: 1019
  humidity: 58
  visibility: 10000
▼ wind:
  speed: 2.06
  deg: 360
▼ clouds:
  all: 0
  dt: 1716922262
▼ sys:
  type: 1
  id: 6505
  country: "FR"
  sunrise: 1716868626
  sunset: 1716923971
timezone: 7200
id: 2973317
name: "Tassin-la-Demi-Lune"
    
```

Conditions météo à l'instant de la demande. Davantage de renseignements sur <https://docs.openweathermap.org/current#list>

Instant précis de la demande. Cet instant est au format « *timestamp* ». Il correspond au nombre de secondes écoulées depuis le 1^{er} janvier 1970. On pourra facilement convertir ce nombre dans un format date, horaire plus compréhensible.

Instant des levés et couchés de soleil ; toujours en format timestamp

Fuseau horaire dans lequel les temps sont donnés. 7200 s = 2 x 3600 s , ce qui correspond au temps GMT + 2h car on est en horaire d'été à 1 fuseau de Greenwich.

Info 21 - Chaîne météo avec Api

3. COMMENT ACCÈDE-T-ON A TOUTES CES VALEURS ? :

- ⇒ Créer un répertoire nommé « *meteo* » par exemple.
- ⇒ Dans ce répertoire, créer un fichier nommé *pageMeteo.html* par exemple.
- ⇒ Y écrire la structure de base (utiliser l'autocomplétion ! + Entrée) :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="pageMeteo.css">
  <title>Quel temps fait-il ?</title>
</head>

<body>

</body>
<script>

</script>
</html>
```

INFO : Copier-coller à partir du pdf n'est pas toujours facile. Ce fichier est aussi proposé au format .docX . Ce sera plus facile d'y rechercher les bouts de codes à copier-coller.

Pour que la ressource au format JSON soit récupérée automatiquement par la page web, on utilise une méthode JavaScript dédiée à cette opération. Cette méthode porte le nom de « *fetch* » qui signifie « *aller chercher* » en français.

- ⇒ Compléter le bloc `<script>` de la page, en copiant-collant le script ci-dessous :

```
<script>
  let urlApiOpenweather = "https://api.openweathermap.org/data/2.5/weather?"
    + "lat=45.75297164916992&lon=4.791003227233887"
    + "&units=metric&lang=fr&"
    + "appid=fed2662095a6145165167da7a942885f"

  let dataMeteo;
  fetch(urlApiOpenweather).then((reponse) => reponse.json())
    .then((meteo) => {
      dataMeteo = meteo
      console.log(dataMeteo);
    })
</script>
```

Requête précédente reconstituée ici par concaténation pour une meilleure lecture

Exécution de la méthode *fetch()* qui prend en argument l'Url

Le traitement de la requête se fait de manière asynchrone par rapport au chargement de la page web. Lorsque le JSON envoyé par le serveur tiers est reçu, la réponse est interprétée en JSON. Quand cette tâche est terminée, un objet nommé ici « *meteo* » est créé. Le navigateur exécute alors ce qu'il y a entre les accolades jaunes, c'est-à-dire que le contenu de la variable locale *meteo* est copié dans la variable *dataMeteo* qui elle a été créée dans le programme principal (variable globale).

Info 21 - Chaîne météo avec Api

⇒ Dans le navigateur, afficher la page « pageMeteo.html ». Avec un clic droit, choisir Inspecter et afficher la console en bas de page.

⇒ Ecrire dans la console `dataMeteo` afin de pouvoir afficher le contenu de cette variable qui est accessible car elle a été définie dans le programme principal :

```
>> dataMeteo
< ▶ Object { coord: {...}, weather: (1) [...], base: "stations", main: {...}, v
```

⇒ Ouvrir le volet afin de pouvoir accéder à la totalité de ce contenu :

```
>> dataMeteo
< ▼ Object { coord: {...}, weather: (1) [...], base: "stations", main: {...}, visibility: 10000,
  base: "stations"
  ▶ clouds: Object { all: 0 }
    cod: 200
  ▶ coord: Object { lon: 4.791, lat: 45.753 }
    dt: 1716927302
    id: 2973317
  ▶ main: Object { temp: 16.02, feels_like: 15.45, temp_min: 14.45, ... }
    name: "Tassin-la-Demi-Lune"
  ▶ sys: Object { type: 1, id: 6505, country: "FR", ... }
    timezone: 7200
    visibility: 10000
  ▶ weather: Array [ {...} ]
  ▶ wind: Object { speed: 3.09, deg: 350 }
  ▶ <prototype>: Object { ... }
```

On constate que cet objet se présente un peu comme les dictionnaires en langage Python. On y retrouve les accolades qui sont les marqueurs des dictionnaires pythons. Pour pouvoir y accéder en lecture ou en écriture, on procède de la même façon qu'en python. Par exemple, en exécutant `>> a = dataMeteo["coord"]["lat"]` dans la console, la variable nommée « `a` » prend comme valeur la latitude écrite dans l'objet JSON.

⇒ Exécuter dans la console : `>> b = dataMeteo["weather"][0]["description"]`

ou `>> b = dataMeteo["weather"][0]["description"]`. Quelle valeur prend la variable nommé ici « `b` » ?



Sympa tout ça, mais comment je fais pour connaître la latitude et la longitude de Tataouine ?

Info 21 - Chaîne météo avec Api

4. API QUI DONNE LES LATITUDE ET LONGITUDE DES LIEUX ? :

Pour connaître les latitudes et longitudes d'un lieu identifié par son adresse postale, dans le monde entier, on utilisera l'Api [positionstack](#), [documentation](#). Si on fait par exemple, la demande des latitude et longitude du lieu identifié par l'adresse postale « 25, rue de tourvielle, Lyon, France », on construit l'url suivante :

https://api.positionstack.com/v1/forward?access_key=8da0cbe78c482c147f68b517b4e5002b&query=25,rue de tourvielle, lyon,france

On reconnaît dans cette URL :

- <https://api.positionstack.com/v1/forward> : l'adresse du serveur tiers accompagnée de paramètres définissant le type de recherche demandée
- [access_key=8da0cbe78c482c147f68b517b4e5002b](#) : la valeur de la variable qui identifie le compte créé sur le serveur tiers pour avoir accès à ce service de localisation (compte lié à une solution de paiement)
- [query=25,rue de tourvielle, lyon,France](#) : la valeur de la variable qui définit l'adresse postale pour laquelle on souhaite en connaître les coordonnées GPS

⇒ Cliquer sur le lien précédent, ou copier-coller le dans la barre d'URL du navigateur. En réponse, le serveur tiers renvoie le fichier JSON suivant :

```
JSON  Données brutes  En-têtes
Enregistrer Copier Tout réduire Tout développer Filtre le JSON
▼ data:
  ▼ 0:
    latitude: 45.752499
    longitude: 4.791187
    type: "address"
    name: "25/25 bis rue de tourvielle"
    number: "25/25 bis"
    postal_code: "69005"
    street: "rue de tourvielle"
    confidence: 1
    region: "Rhône"
    region_code: "RH"
    county: "Lyon"
    locality: "Lyon"
    administrative_area: null
    neighbourhood: null
    country: "France"
    country_code: "FRA"
    continent: "Europe"
    label: "25/25 bis rue de tourvielle, Lyon, France"
  ▼ 1:
    latitude: 45.753087
    longitude: 4.7918
    type: "address"
    name: "25 Rue De Tourvielle"
```

Coordonnées GPS

Localité

Info 21 - Chaîne météo avec Api

5. LECTURE DE CES VALEURS DANS NOTRE PAGE WEB ? :

On utilise à nouveau la méthode « fetch » dans notre code JavaScript. On utilise à peu près les mêmes lignes de codes. Il suffit de modifier dans la requête `fetch()` précédente l'adresse url et le nom des variables. Cela donne :

```
<script>
  let urlPositionStack = "https://api.positionstack.com/v1/forward"
    + "?access_key=8da0cbe78c482c147f68b517b4e5002b"
    + "&query=25,rue de tourvielle, lyon,france"

  let dataPosition;
  fetch(urlPositionStack).then((reponse) => reponse.json())
    .then((position) => {
      dataPosition = position
      console.log(dataPosition);
    })
</script>
```

⇒ Modifier le bloc `<script>` de la page, en copiant-collant l'ensemble précédent.

En rechargeant la page `pageMeteo.html`, on obtient dans la console :

```
>> dataPosition
< ▾ Object { data: (3) [...] }
  ▾ data: Array(3) [ {...}, {...}, {...} ]
    ▶ 0: Object { latitude: 45.752499, longitude: 4.791187, type: "address", ... }
    ▶ 1: Object { latitude: 45.753087, longitude: 4.7918, type: "address", ... }
    ▶ 2: Object { latitude: 45.753037, longitude: 4.791616, type: "address", ... }
    length: 3
```

L'Api a ainsi trouvé 3 lieux sur la planète qui correspondent à l'adresse postale saisie. Les informations relatives à ces 3 lieux sont accessibles dans la liste (*array* en JS) donnée ci-contre et comprenant 3 éléments.

```
>> dataPosition["data"]
< ▶ Array(3) [ {...}, {...}, {...} ]
```

Le 1^{er} élément de cette liste est donné ci-contre :

```
>> dataPosition["data"][0]
< ▶ Object { latitude: 45.752499, longitude: 4.791187,
  type: "address", name: "25/25 bis rue de tourvielle",
  number: "25/25 bis", postal_code: "69005", street: "rue
  de tourvielle", confidence: 1, region: "Rhône",
  region code: "RH", ... }
```

Pour lire les coordonnées Gps de ce lieu, on crée les variables ci-contre :

```
>> lat = dataPosition["data"][0]["latitude"]
< 45.752499
>> lon = dataPosition["data"][0]["longitude"]
< 4.791187
```

Info 21 - Chaîne météo avec Api

6. PAGE HTML :

On a jusqu'à présent découvert 2 Api qui nous permettent d'obtenir les coordonnées GPS d'un lieu identifié par son adresse postale pour la première Api et les informations météo d'un lieu identifié par ses coordonnées Gps, pour la seconde.

Dans ce paragraphe, on voit ce que l'on va mettre dans la page *pageMeteo.html* que l'on est en train de construire.

On propose dans un premier temps de réaliser la page suivante :

Au chargement de la page, on a :

Adresse postale :

Météo à

- Heure en France :
- Pays :
- Décalage horaire par rapport à l'Angleterre: h
- Température en °C :
- Description :
- Vitesse du vent en m/s :
- Levée de soleil (heure Française) :
- Couché de soleil (heure Française) :

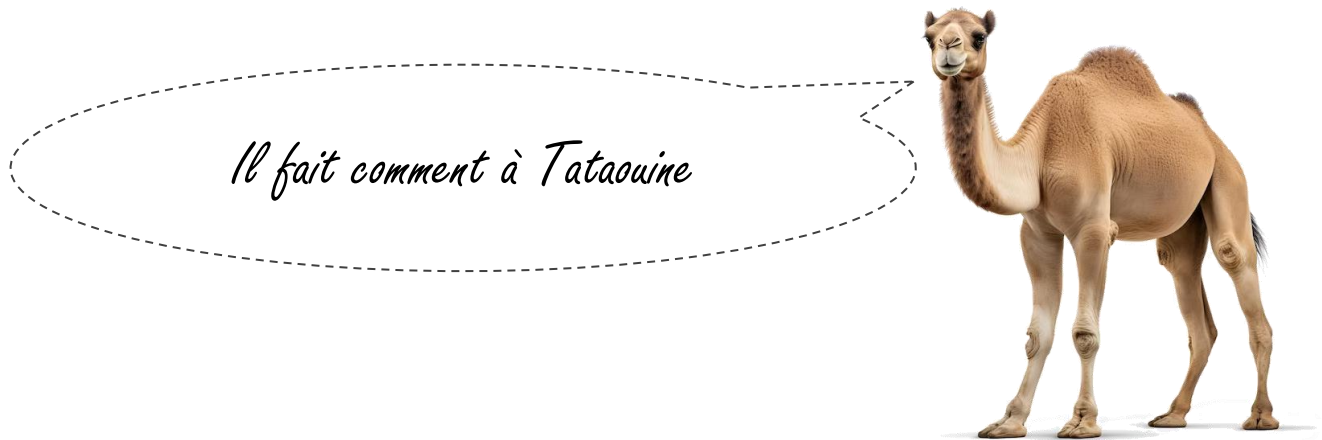
Après saisie de l'adresse : « 55 rue du Faubourg-Saint-Honoré 75008 Paris, France » qui correspond à celle du palais présidentiel de l'Élysée et click sur Valider, on obtiendra :

Adresse postale :

Météo à 55 rue du Faubourg-Saint-Honoré 75008 Paris, France

- Heure en France : 17:42:26
- Pays : FR
- Décalage horaire par rapport à l'Angleterre: 2 h
- Température en °C : 16.16
- Description : légère pluie
- Vitesse du vent en m/s : 5.14
- Levée de soleil (heure Française) : 05:53:50
- Couché de soleil (heure Française) : 21:42:43

Info 21 - Chaîne météo avec Api



Après saisie de l'adresse : « tataouine, tunisie » qui correspond à celle du domicile de ce *Camelus dromedarius* et click sur Valider, on obtiendra :

Adresse postale : Valider Reset

Météo à tataouine, tunisie

- Heure en France : 17:52:11
- Pays : TN
- Décalage horaire par rapport à l'Angleterre: 1 h
- Température en °C : 27.43
- Description : ciel dégagé
- Vitesse du vent en m/s : 8.12
- Levée de soleil (heure Française) : 06:11:43
- Couché de soleil (heure Française) : 20:19:55

Ainsi la page *pageMeteo.html* que l'on doit construire, devra :

- Comporter un champ de saisie pour relever l'adresse postale
- 2 éléments html `<button>` pour valider et réaliser un reset
- Un bloc *html* qui affiche les résultats qui seront obtenus grâce aux 2 Api étudiées précédemment

Pour ne pas perdre de temps sur cette partie html, on propose le code html du `<body>` qui est donné sur la page suivante. Vous pouvez le copier-coller dans votre fichier *pageMeteo.html* .


```
<body>
  <div id="formulaire">
    <label for="adresse">Adresse postale : </label>
    <input type="text" id="adresse" autofocus>
    <button id="bouton">Valider</button>
    <button id="reset">Reset</button>
  </div>
  <div>
    <h2>Météo à <span id="lieu"></span></h2>
    <ul>
      <li>Heure en France : <span id="h"></span></li>
      <li>Pays : <span id="p"></span></li>
      <li>Décalage horaire par rapport à l'Angleterre: <span id="utc"></span> h</li>
      <li>Température en °C : <span id="t"></span></li>
      <li>Description : <span id="d"></span></li>
      <li>Vitesse du vent en m/s : <span id="v"></span></li>
      <li>Levée de soleil (heure Française) : <span id="ls"></span></li>
      <li>Couché de soleil (heure Française) : <span id="cs"></span></li>
    </ul>
  </div>
</body>
```

7. CODE JAVASCRIPT :

Il reste à créer le code JS dans la partie `<script>`. Après saisie de l'adresse et click sur le bouton *Valider*, ce code devra exécuter les actions suivantes :

- Lire l'adresse (string) qui a été saisie
- Interroger l'Api « *positionstack* » avec la méthode `fetch()` pour convertir cette adresse en coordonnées GPS (latitude et longitude)
- Utiliser ces coordonnées pour interroger l'Api « *openWheater* » avec la méthode `fetch()`
- Extraire les informations météos du json reçu
- Afficher les résultats dans la page *pageMeteo.html*

Convertir Une Date En Timestamp	
26 / 07 / 2024 📅	20:00:00
Résultat :	
1722016800	
<input type="button" value="CONVERTIR"/>	

INFO : Les temps donnés par les Api sont en format « *timestamp* ». Par exemple la cérémonie d'ouverture des jeux olympiques de Paris est fixée au 26 juillet 2024 à 20h00. Ce moment en informatique est repéré par le timestamp 1722016800. Cette valeur correspond au nombre de secondes écoulées depuis le 1 janvier 1970 à 01h00. Jetez un coup d'œil sur ce [convertisseur en ligne](#) qui vous permettra de mieux comprendre ce format de temps.

Info 21 - Chaîne météo avec Api

Là aussi pour gagner du temps, on donne une partie du code JS que vous pouvez copier-coller.

Ce code fonctionne et se termine temporairement sur l'appel de la fonction *affichage()* dans laquelle on retrouve les objets Json envoyés par les 2 Api.

```
<script>
// définition des variables
let inputAdresse = document.querySelector("#adresse");
let bouton = document.querySelector("#bouton");
let reset = document.querySelector("#reset");
inputAdresse.value = ""

// évènement lié au click sur le bouton valider
bouton.addEventListener("click",rechercher);

//FONCTIONS
function rechercher(){
  let urlPositionStack = "https://api.positionstack.com/v1/forward"
    + "?access_key=8da0cbe78c482c147f68b517b4e5002b"
    + "&query="
    + inputAdresse.value;
  fetch(urlPositionStack).then((reponse) => reponse.json())
    .then((position) => {
    let lat = position["data"][0]["latitude"];
    let lon = position["data"][0]["longitude"];
    let continent = position["data"][0]["continent"];
    let urlApiOpenweather = "https://api.openweathermap.org/data/2.5/weather?"
      + "lat="+lat+"&lon="+lon
      + "&units=metric&lang=fr&"
      + "appid=fed2662095a6145165167da7a942885f";
    fetch(urlApiOpenweather).then((reponse) => reponse.json())
      .then((meteo) => {
        affichage(position,meteo)
      })
    })
  })
}

function affichage(position,meteo){
  console.log(position,meteo)
}
</script>
```

Ce bloc entre accolades est exécuté lorsque la ressource *position* est réceptionnée par le navigateur (réception asynchrone)

Ce bloc entre accolades et surligné en jaune est exécuté lorsque la ressource *meteo* est réceptionnée par le navigateur (réception asynchrone)

⇒ Compléter le code JS en suivant les étapes suivantes :

- Compléter la fonction *affichage()* qui permet d'extraire et ensuite d'afficher les données qui nous intéressent.
- Créer un évènement lié au bouton Reset et créer la fonction callback associée.

Info 21 - Chaîne météo avec Api

Autre code donnée ci-contre :

Pour convertir un temps au format *timestamp*, ce n'est pas si facile que ça ... Il faut tenir compte des années bissextiles, etc ...

La gestion des dates est un classique de l'informatique. Des outils JS tout près, permettent de ne pas se poser trop de questions ... on les utilise. Vous pouvez ci-contre, copier-coller le code JS de la fonction *timestamp()* qui effectue cette conversion.

```
function timestamp(ts, code=0){
  // convert unix timestamp to milliseconds
  let ts_ms = ts * 1000;
  let date_ob = new Date(ts_ms);
  // year as 4 digits (YYYY)
  let year = date_ob.getFullYear();
  // month as 2 digits (MM)
  let month = ("0" + (date_ob.getMonth() + 1)).slice(-2);
  // date as 2 digits (DD)
  let date = ("0" + date_ob.getDate()).slice(-2);
  // hours as 2 digits (hh)
  let hours = ("0" + date_ob.getHours()).slice(-2);
  // minutes as 2 digits (mm)
  let minutes = ("0" + date_ob.getMinutes()).slice(-2);
  // seconds as 2 digits (ss)
  let seconds = ("0" + date_ob.getSeconds()).slice(-2);
  // date as YYYY-MM-DD format
  let YYYY_MM_DD = date + "-" + month + "-" + year
  let hh_mm_ss = hours + ":" + minutes + ":" + seconds
  if (code == 0) {return hh_mm_ss }
  return YYYY_MM_DD
}
```

```
>> timestamp(1722016800,1)
< "26-07-2024"
>> timestamp(1722016800)
< "20:00:00"
```

Ci-contre un exemple d'utilisation sur le timestamp de la date d'ouverture des JO de Paris :

8. AJOUT D'UNE CARTE GOOGLE MAP :

On améliore cette page en rajoutant une carte « *google map* » :

Adresse postale :

Météo à

- Heure en France :
- Pays :
- Décalage horaire par rapport à l'Angleterre: h
- Température en °C :
- Description :
- Vitesse du vent en m/s :
- Levée de soleil (heure Française) :
- Couché de soleil (heure Française) :



Info 21 - Chaîne météo avec Api

Au chargement de la page, la carte place un marqueur et se centre sur les coordonnées GPS du lycée Branly.

Pour intégrer cette carte dans votre script, il suffit de rajouter dans le bloc `<head>` la balise `<script>` ci-dessous, à copier-coller :

```
<script async src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBOhLSONkSQS-7r7gnBkswpCj7qBMHJ-3I&callback=console.debug&libraries=maps,marker&v=beta">
</script>
```

et après le dernier `<div>` de `<body>`, de rajouter le bloc ci-dessous :

```
<gmp-map center="45.75297164916992,4.791003227233887" zoom="6" map-id="DEMO_MAP_ID">
  <gmp-advanced-marker position="45.75297164916992,4.791003227233887" title="Branly"></gmp-advanced-marker>
</gmp-map>
```

Les valeurs des attributs des balises html ci-dessus, permettent :

- De centrer la carte sur les coordonnées GPS de l'attribut « *center* »
- De définir le zoom autour de ce point central (valeur entre 1 et 12) pour l'attribut « *zoom* »
- De définir les coordonnées GPS du marqueur rouge sur la carte, pour l'attribut « *position* »
- De définir le commentaire qui apparait au survol du marqueur pour l'attribut « *title* »

Dans le fichier `pageMeteo.css`, il s'agit de rajouter les commandes css suivantes (à copier-coller). L'attribut class « *resultat* » doit être ajouté à la balise ouvrante du dernier `<div>` :

Cela donne finalement l'html suivant :

```
<body>
<div id="formulaire">
  <label for="adresse">Adresse postale : </label>
  <input type="text" id="adresse" autofocus>
  <button id="bouton">Valider</button>
  <button id="reset">Reset</button>
</div>
<div class="resultat">
  <h2>Météo à <span id="lieu"></span></h2>
  <ul>
    <li>Heure en France : <span id="h"></span></li>
    <li>Pays : <span id="p"></span></li>
    <li>Décalage horaire par rapport à l'Angleterre: <span id="utc"></span> h</li>
    <li>Température en °C : <span id="t"></span></li>
    <li>Description : <span id="d"></span></li>
    <li>Vitesse du vent en m/s : <span id="v"></span></li>
    <li>Levée de soleil (heure Française) : <span id="ls"></span></li>
    <li>Couché de soleil (heure Française) : <span id="cs"></span></li>
  </ul>
</div>
<gmp-map center="45.75297164916992,4.791003227233887" zoom="6" map-id="DEMO_MAP_ID">
  <gmp-advanced-marker position="45.75297164916992,4.791003227233887" title="Branly"></gmp-advanced-marker>
</gmp-map>
</body>
```

```
gmp-map {
  width:48%;
  height: 400px;
  display: inline-block;
  margin:10px;
}
html, body {
  height: 100%;
}
.resultat{
  display: inline-block;
  vertical-align:top;
  margin:10px;
}
```

Info 21 - Chaîne météo avec Api

9. MODIFICATION DU CENTRAGE SUR LA CARTE :

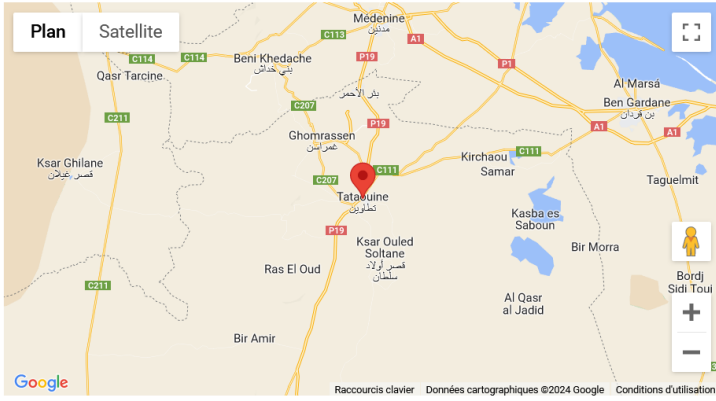
⇒ Compléter le code JS afin modifier les attributs « *center* », « *position* » et « *title* » pour centrer la carte google-map sur le lieu saisi par l'utilisateur. Pour éviter une réinitialisation de l'attribut « *zoom* », il est nécessaire de le redéfinir à sa valeur 9. On donne ci-dessous, l'exemple d'une saisie pour « Tataouine , Tunisie » :

On donne ci-dessous le début du code JS avec notamment l'appel de la fonction `centrageCarte()` qui est à créer.

Adresse postale : Valider Reset

Météo à tataouine tunisie

- Heure en France : 19:58:45
- Pays : TN
- Décalage horaire par rapport à l'Angleterre: 1 h
- Température en °C : 23,71
- Description : ciel dégagé
- Vitesse du vent en m/s : 7,11
- Levée de soleil (heure Française) : 06:11:43
- Couché de soleil (heure Française) : 20:19:55



```
// définition des variables
let inputAdresse = document.querySelector("#adresse");
let bouton = document.querySelector("#bouton");
let reset = document.querySelector("#reset");
inputAdresse.value = ""
let map = document.querySelector("gmp-map")
let marker = document.querySelector("gmp-advanced-marker")
// évènement lié au click sur le bouton valider
bouton.addEventListener("click",rechercher);
reset.addEventListener("click",init);
//FONCTIONS
function rechercher(){
  let urlPositionStack = "https://api.positionstack.com/v1/forward"
  +"?access_key=8da0cbe78c482c147f68b517b4e5002b"
  +"&query="
  + inputAdresse.value;
  fetch(urlPositionStack).then((reponse) => reponse.json())
    .then((position) => {
      let lat = position["data"][0]["latitude"];
      let lon = position["data"][0]["longitude"];
      let localite = position["data"][0]["locality"];
      centrageCarte(lat,lon,localite)
      let urlApiOpenweather = "https://api.openweathermap.org/data/2.5/weather?"
      +"lat="+lat+"&lon="+lon
      +"&units=metric&lang=fr&"
      +"appid=fed2662095a6145165167da7a942885f";
      fetch(urlApiOpenweather).then((reponse) => reponse.json())
        .then((meteo) => {
          affichage(position,meteo)
        })
    })
}
```

INFO : Pour modifier en JS l'attribut « center » de la balise html `<gmp-map`, on utilise la méthode `setAttribute()` vue dans le poly :
`let n = String(x)+","+String(y)`
`map.setAttribute("center",n)`

La fonction `centrageCarte()` est exécutée seulement lorsque la ressource position est réceptionnée

Info 21 - Chaîne météo avec Api

10. CENTRAGE DYNAMIQUE SUR LA CARTE :

On se propose ici d'obtenir un centrage sur la carte qui soit progressif. La vidéo qui accompagne ce Tp sur nsibrantly.fr montre le résultat attendu. Par exemple, si la carte est centrée initialement sur Lyon et que l'on demande la météo sur le lieu « *Dehli en Inde* », l'affichage des données écrites est presque instantané. Par contre, le changement de lieu sur la carte se fait en plusieurs secondes :

- Le zoom passe de la valeur 9 à la valeur 4 en 2 secondes environ,
- Le centrage sur la carte est modifié en 6 secondes environ,
- Le zoom repasse à la valeur 9 en 2 secondes environ.

Pour modifier le centrage progressivement, on utilise la fonction JavaScript `setInterval()`. Pour s'approprier cette fonction qui est un peu l'équivalent de la méthode

« `after()` » de Tkinter en Python, testez le script donné ci-contre :

```
<script>
  let intervalID = setInterval(myCallback, 500 , 2024 , 'coucou')
  let compteur = 0

  function myCallback(a,b) {
    console.log(compteur,a,b)
    compteur = compteur + 1
    if(compteur==5){clearInterval(intervalID)}
  }
</script>
```

Cette fonction permet d'arrêter le processus de répétition

Le code JavaScript de la fonction `centrageCarte()` devient alors celui donné ci-contre :

```
let intervalID
function centrageCarte(xB,yB,localite){
  let xA = parseFloat(map.getAttribute("center").split(',')[0])
  let yA = parseFloat(map.getAttribute("center").split(',')[1])
  compteurCentrage = 0
  compteurZoom = 0
  intervalID = setInterval(mouvementCarte, 300 , xA,yA,xB,yB);
  marker.setAttribute("title",localite)
}
```

Les variables ci-dessous peuvent être créées dans le programme principal, pour leur donner un statut de variable global :

```
let compteurCentrage
let compteurZoom
let intervalID
const nInterval = 20
const nIntervalZoom = 6
```

Variable dont la valeur permettra d'arrêter le processus de répétition avec `clearInterval()`

Nombres de répétitions : 20 pour modifier le centrage et 6 pour dézoomer et ensuite zoomer.

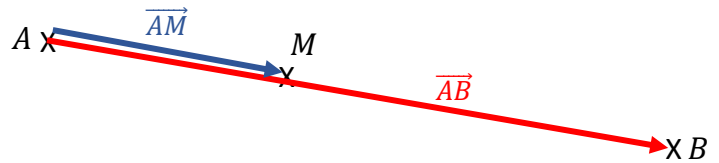
⇒ Il reste à trouver le script de la fonction `mouvementCarte()` qui est répétée.

Info 21 - Chaîne météo avec Api

AIDE : Pour trouver les coordonnées d'un point $M(x;y)$ qui se déplace sur un segment de droite entre un point $A(x_A;y_A)$ et un point $B(x_B;y_B)$, on utilise la relation vectorielle suivante :

$\vec{AM} = k \vec{AB}$ avec k qui est un coefficient compris entre 0 et 1.

Cela donne :



$$\begin{pmatrix} x - x_A \\ y - y_A \end{pmatrix} = k \begin{pmatrix} x_B - x_A \\ y_B - y_A \end{pmatrix}, \text{ soit } \begin{cases} x = x_A + k(x_B - x_A) \\ y = y_A + k(y_B - y_A) \end{cases}.$$

11. AMELIORATION AVEC DU CSS, DES IMAGES ...:

⇒ Améliorer l'ensemble en rajoutant à la page :

- Une image pour accompagner la description du temps qu'il fait (nécessité de télécharger une petite galerie d'images météo)
- Un fond d'écran sombre s'il fait nuit et claire dans le cas contraire.
- Une modification progressive des attributs « *center* », « *position* » et « *zoom* » de la carte google-map.

