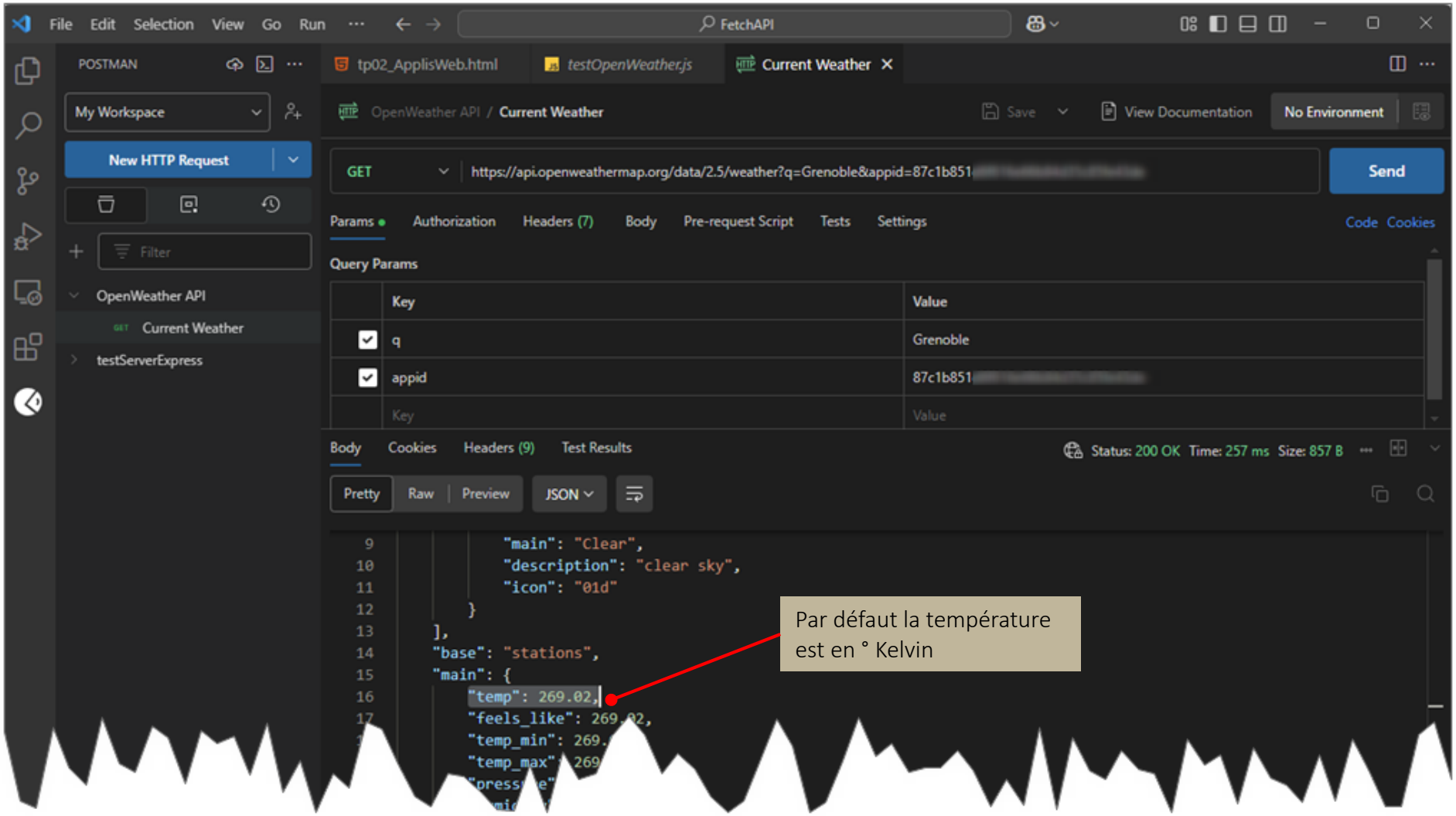
Correction TP 02 AWA
API fetch

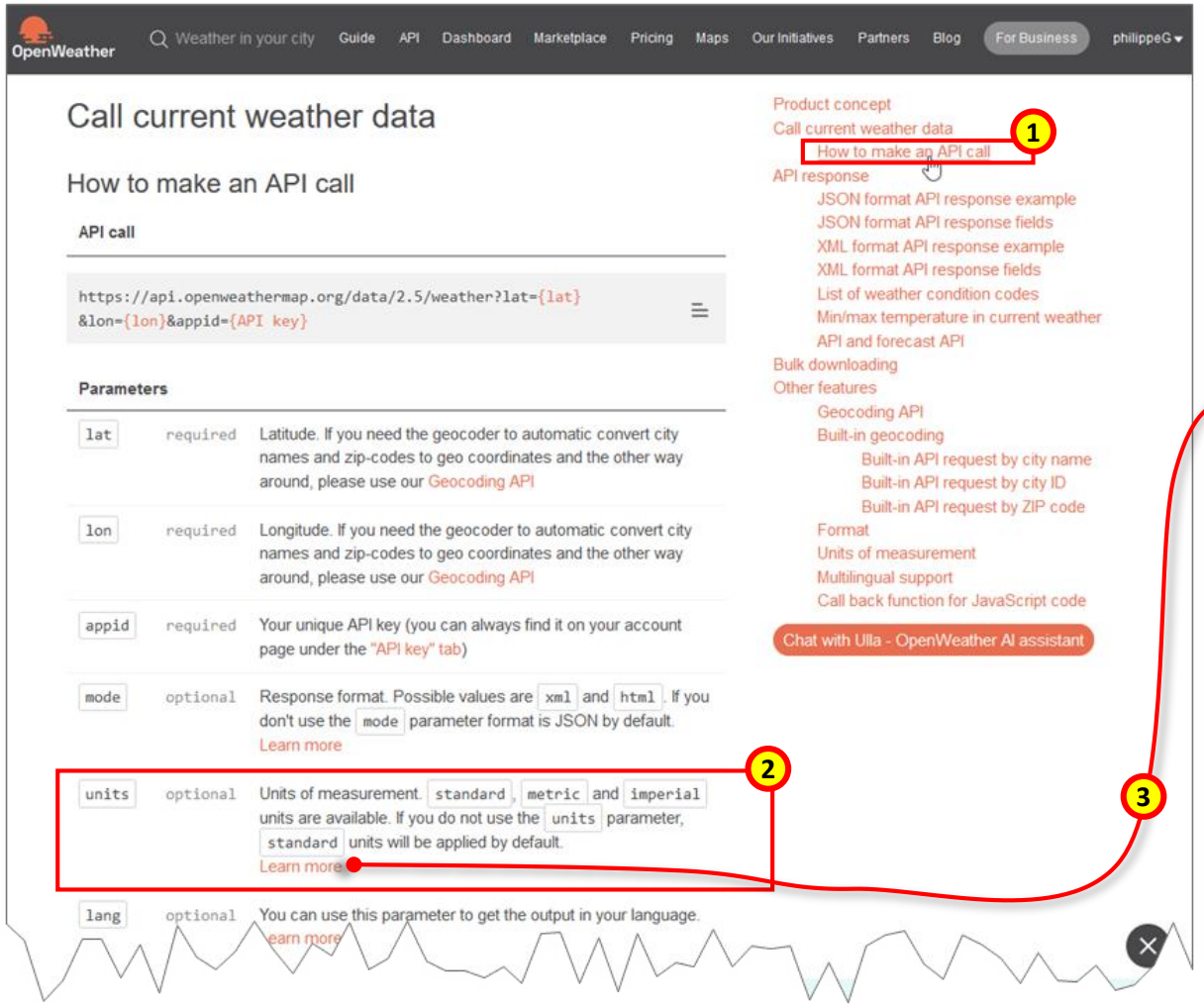
Question 1.7 – récupérer la température en °C

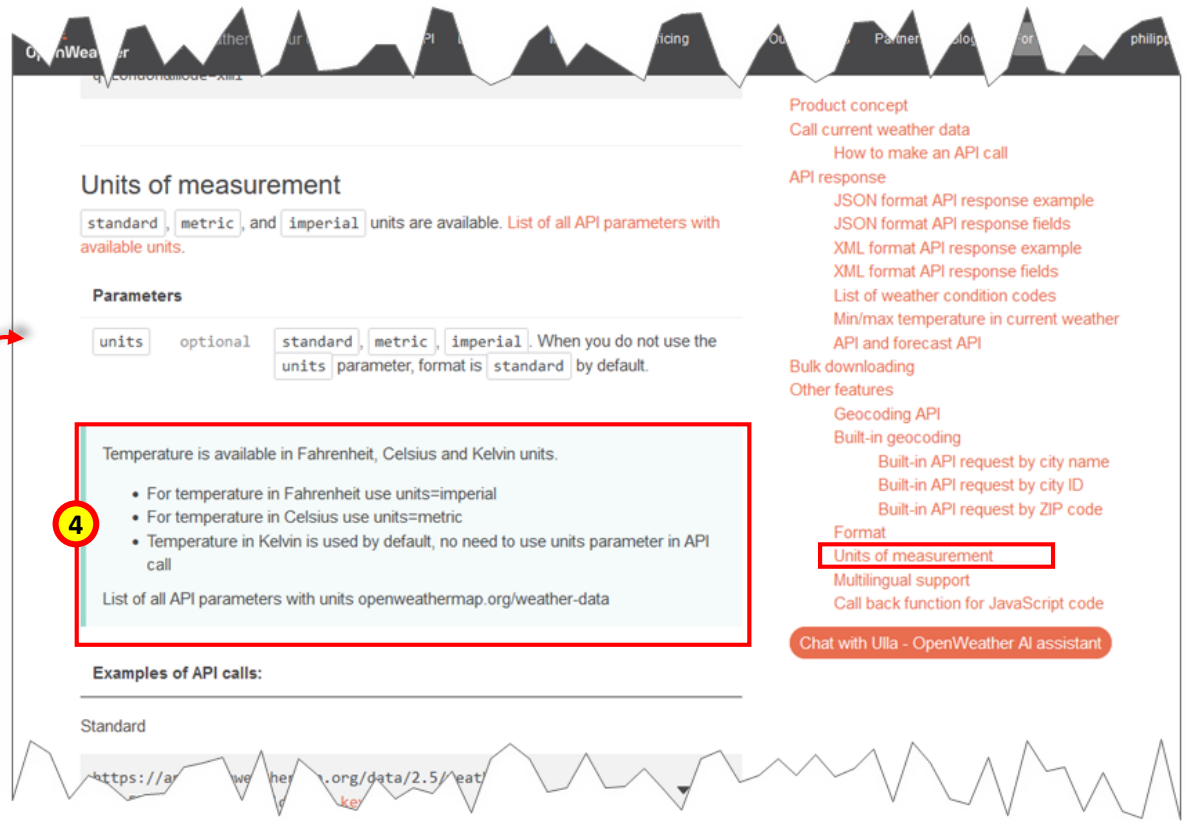


Question 1.7 – récupérer la température en °C

https://openweathermap.org/current

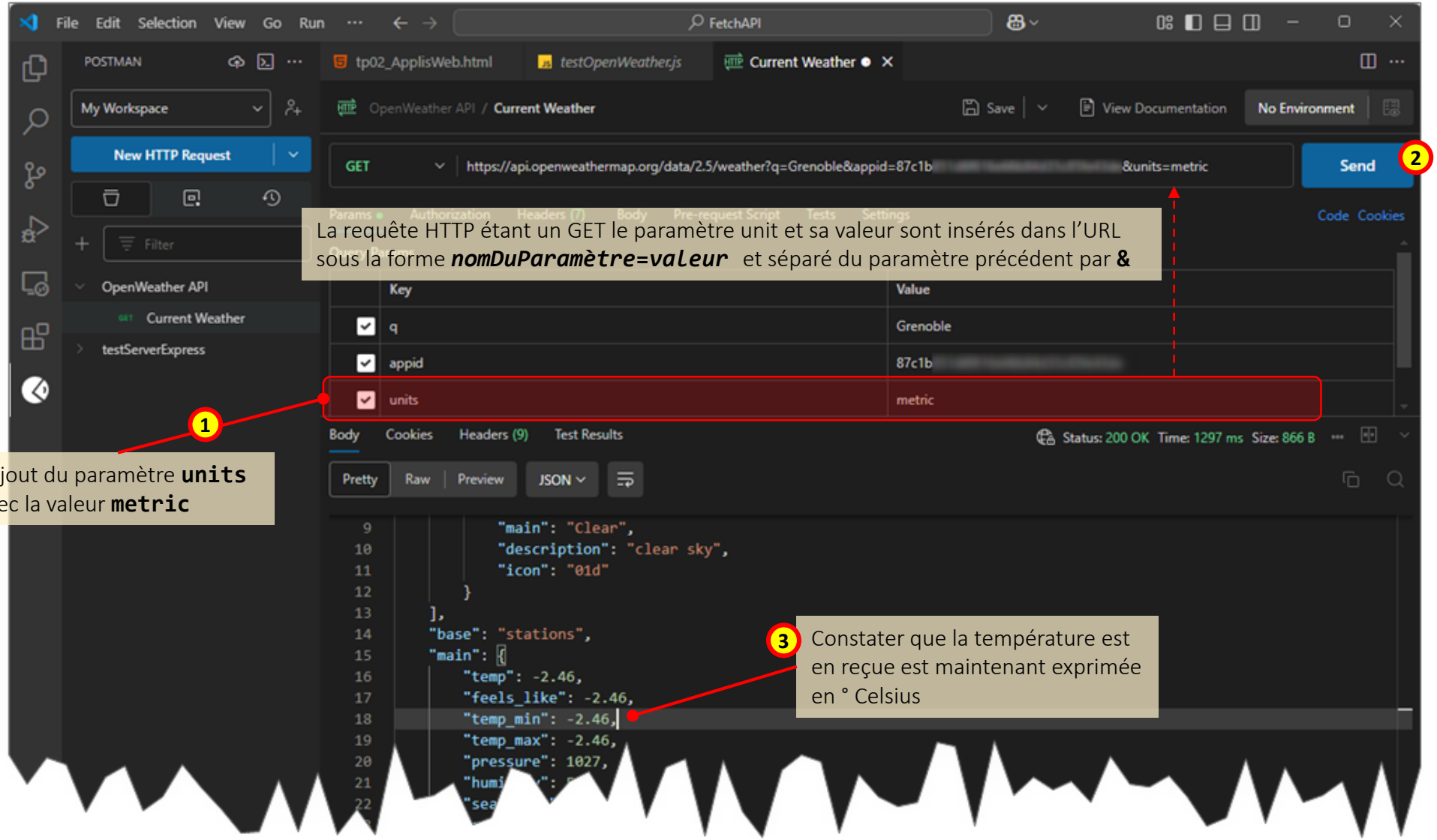
Consulter la documentation de l'API



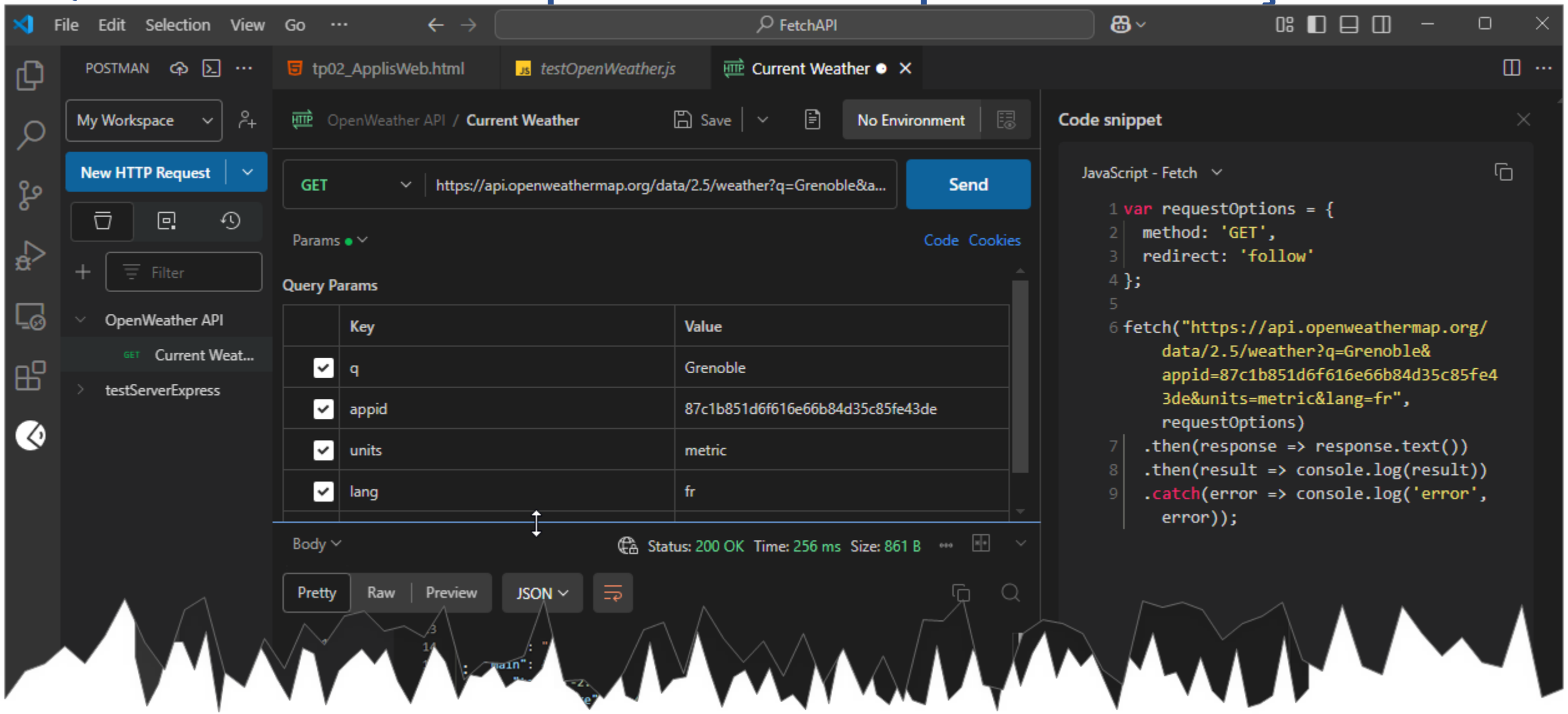


Il faut rajouter à la requête le paramètre **`units`** avec comme valeur **`metric`**

Question 1.7 – récupérer la température en °C



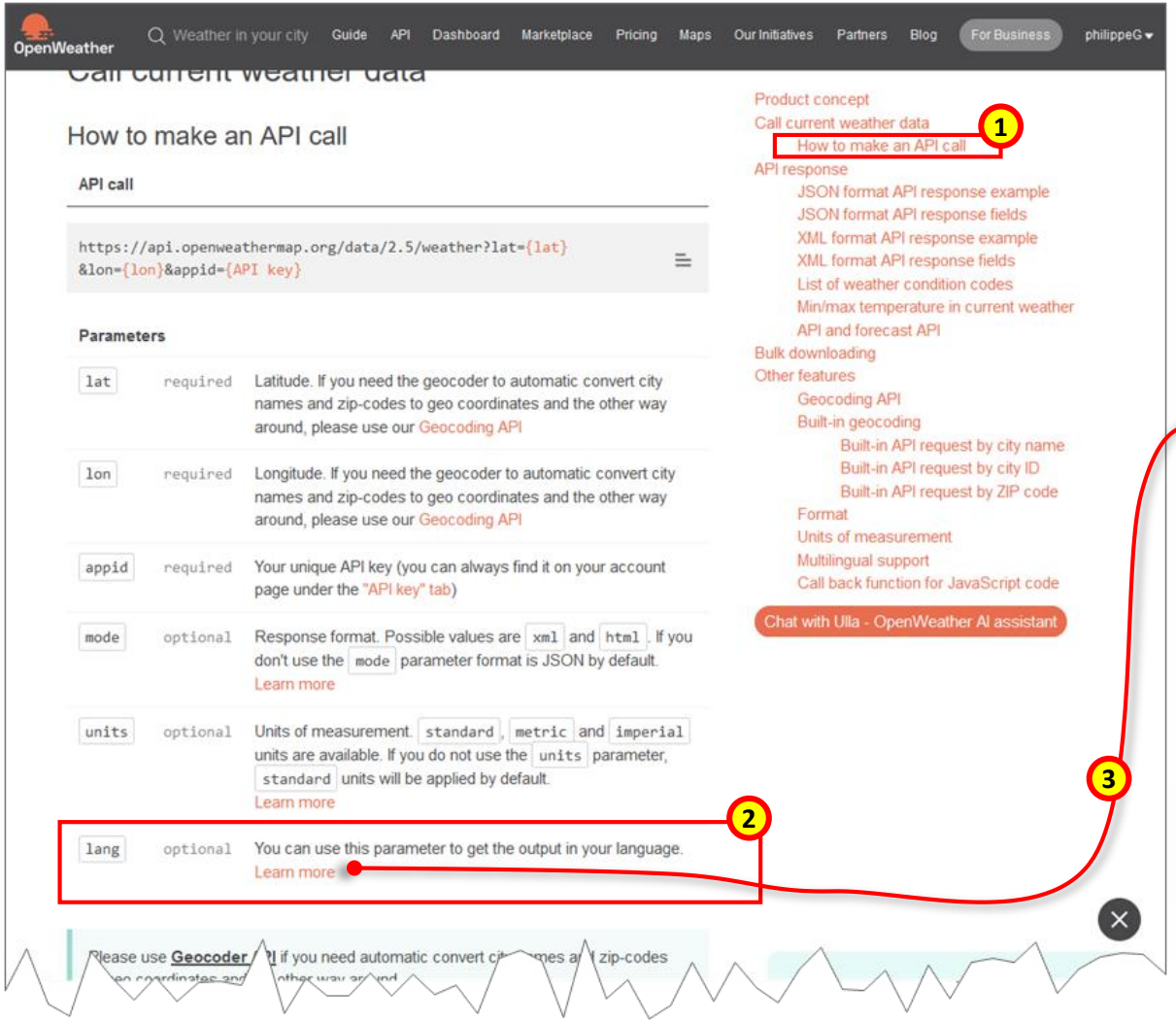
Question 1.8 – récupérer la description en français



Question 1.8 – récupérer la description en français

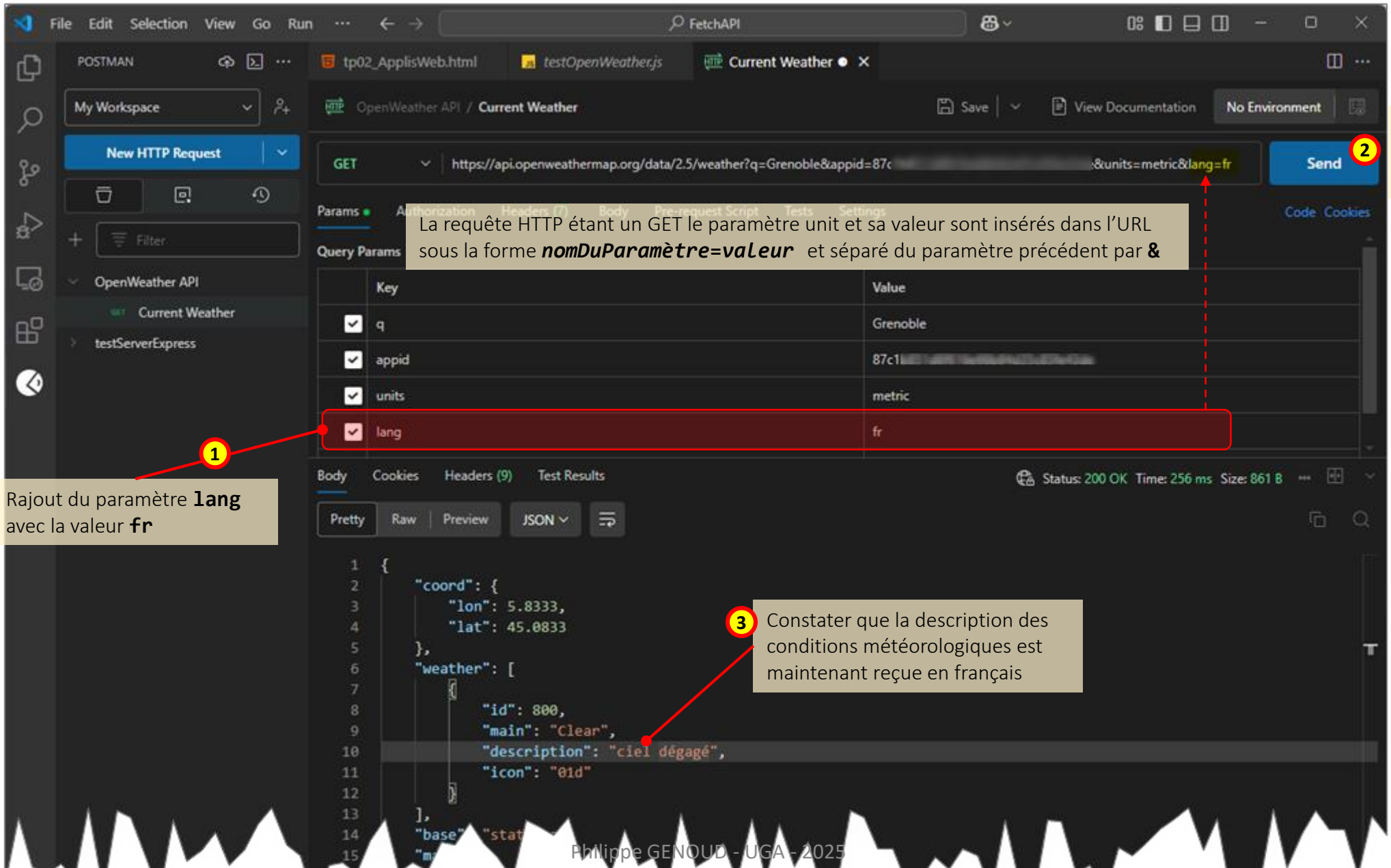
https://openweathermap.org/current

Consulter la documentation de l'API



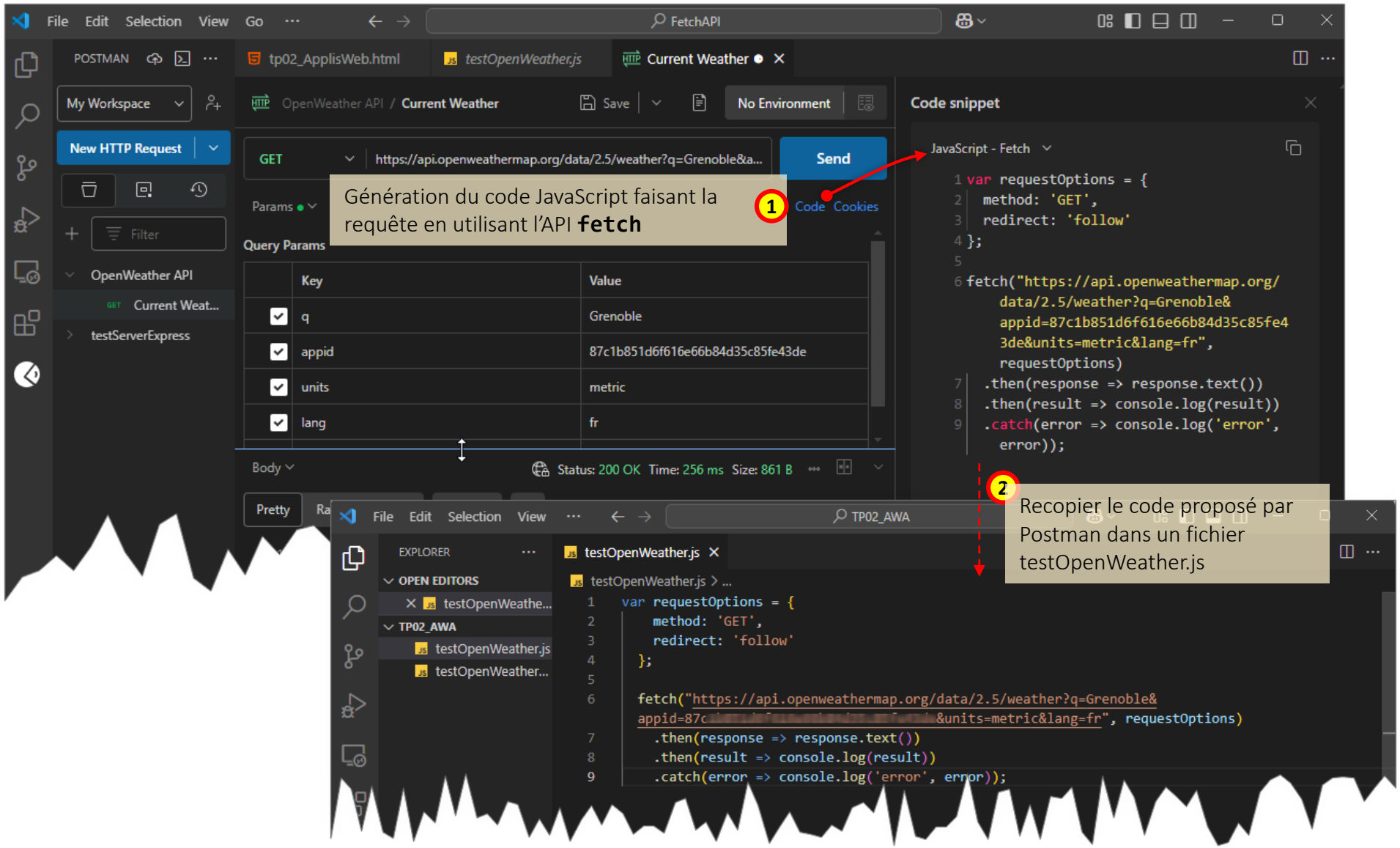
Il faut rajouter à la requête le paramètre **`lang`** avec comme valeur **`fr`**

Question 1.8 – récupérer la description en français



Renvoyer a nouveau la requête

Question 2.3 : utiliser fetch, récupérer la réponse en JSON



Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
var requestOptions = {
    method: 'GET',
    redirect: 'follow'
  };

fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
      &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
   requestOptions)
    .then(response => response.text())
    .then(result => console.log(result))
    .catch(error => console.log('error', error));

console.log('Terminé !');
```

Que fait ce programme et quel affichage produit-il ?

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

Que fait ce programme et quel affichage produit-il ?

effectue la requête HTTP pour connaître les conditions météorologiques courantes à Grenoble et affiche la réponse brute (*raw data*) retournée par le serveur api.openweathermap.org

```
> node . testOpenWeather.js
Terminé !
{"coord":{"lon":5.8333,"lat":45.0833},"weather":[{"id":801,"main":"Clouds",
"description":"peu
nuageux","icon":"02d"}],"base":"stations","main":{"temp":7.9,"feels_like":7
.9,"temp_min":7.9,"temp_max":7.9,"pressure":1025,"humidity":62,"sea_level":
1025,"grnd_level":902},"visibility":10000,"wind":{"speed":0.86,"deg":358,"g
ust":1.78},"clouds":{"all":14},"dt":1738679554,"sys":{"country":"FR","sunri
se":1738651987,"sunset":1738687683},"timezone":3600,"id":3014727,"name":"Gr
enoble","cod":200}
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

Pourquoi `Terminé !` est-il affiché avant la réponse du serveur ?

```
> node . testOpenWeather.js
Terminé !
{"coord":{"lon":5.8333,"lat":45.0833},"weather":[{"id":801,"main":"Clouds",
"description":"peu
nuageux","icon":"02d"}],"base":"stations","main":{"temp":7.9,"feels_like":7
.9,"temp_min":7.9,"temp_max":7.9,"pressure":1025,"humidity":62,"sea_level":
1025,"grnd_level":902},"visibility":10000,"wind":{"speed":0.86,"deg":358,"g
ust":1.78},"clouds":{"all":14},"dt":1738679554,"sys":{"country":"FR","sunri
se":1738651987,"sunset":1738687683},"timezone":3600,"id":3014727,"name":"Gr
enoble","cod":200}
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

Pourquoi `Terminé !` est-il affiché avant la réponse du serveur ?

Parce que `fetch` est une **fonction asynchrone**, elle **lance le traitement** qui consiste à envoyer la requête HTTP et à attendre la réponse du serveur **et rend immédiatement la main**. De même les fonctions `then` et `catch` qui définissent les traitements à effectuer après que `fetch` ait reçu une réponse du serveur sont elles aussi **asynchrones**.

```
> node . testOpenWeather.js
Terminé !
{"coord":{"lon":5.8333,"lat":45.0833},"weather":[{"id":801,"main":"Clouds",
"description":"peu
nuageux","icon":"02d"}],"base":"stations","main":{"temp":7.9,"feels_like":7
.9,"temp_min":7.9,"temp_max":7.9,"pressure":1025,"humidity":62,"sea_level":
1025,"grnd_level":902},"visibility":10000,"wind":{"speed":0.86,"deg":358,"g
ust":1.78},"clouds":{"all":14},"dt":1738679554,"sys":{"country":"FR","sunri
se":1738651987,"sunset":1738687683},"timezone":3600,"id":3014727,"name":"Gr
enoble","cod":200}
>
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

Préciser l'ordre dans lequel les instructions du programme sont exécutées et ce qu'elles font.

```
> node . testOpenWeather.js
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

method spécifie le type de requête HTTP à effectuer. Les méthodes couramment utilisées sont :
GET : Récupérer des données depuis le serveur.
POST : Envoyer des données au serveur.
PUT : Mettre à jour des données existantes sur le serveur.
DELETE : Supprimer des données sur le serveur.

redirect spécifie comment la requête **fetch** doit gérer les redirections HTTP (codes réponses 3xx) .
La valeur **'follow'** indique que la requête **fetch** suivra automatiquement les redirections vers la nouvelle location envoyée dans la réponse HTTP

1 Définit un objet d'options référencé par la variable **requestOptions** qui servira à configurer la requête effectué par l'appel **fetch**

```
> node . testOpenWeather.js
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

promesse initiale créée par `fetch`

status: pending
value : undefined

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
2  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
6        &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

`url` identifiant la ressource (le service web) à interroger

La requête étant un **`GET`** les paramètres peuvent être écrits directement dans l'url. Ils sont introduits par **`?`** et définis par des couples ***`nomparamètre=valeur`*** séparés par **`&`**

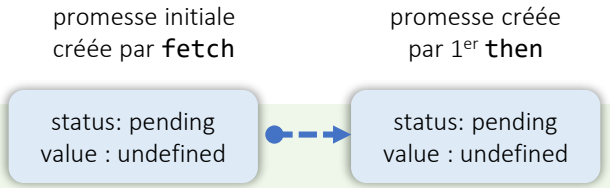
```
> node . testOpenWeather.js
```

2 exécute la fonction `fetch` qui lance de manière asynchrone la requête HTTP pour interroger le serveur `openweathermap`.

Cette fonction retourne un objet promesse (`Promise`) qui sera résolue par la réponse du serveur

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman




```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
3 8    .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

Fonction fléchée (*arrow function*)

⇕

```
function(response) {
    return response.text();
}
```

cette fonction sera exécutée quand la promesse créée par **fetch** sera **tenue**

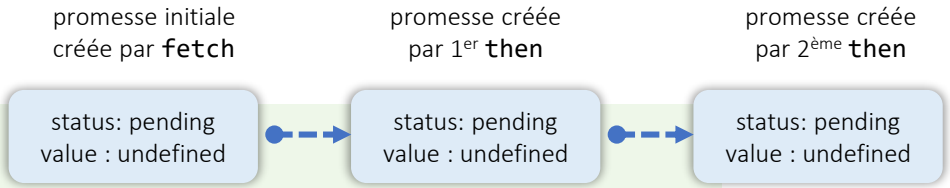
```
> node . testOpenWeather.js
```

3 Envoie un message **then** à l'objet promesse retourné par **fetch**

then est asynchrone, elle renvoie une nouvelle promesse chaînée avec la précédente.

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman




```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
4 9    .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

Fonction fléchée (*arrow function*)

⇕

```
function(result) {
   return console.log(result);
}
```

cette fonction sera exécutée quand la promesse créée par le 1er **then** sera **tenue**

```
> node . testOpenWeather.js
```

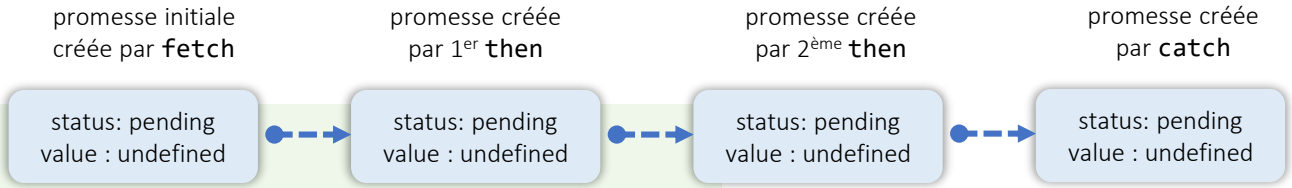
4 Envoie le message **then** à l'objet promesse retourné précédemment (promesse retournée par le 1er **then**)

then est asynchrone, elle renvoie une nouvelle promesse chainée avec la précédente.

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
         &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
5 10   .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```



Fonction fléchée (*arrow function*)

⇕

```
function(result) {
   console.log('error', error);
}
```

cette fonction sera exécutée quand la promesse créée par le 2ème **then** sera **tenue**

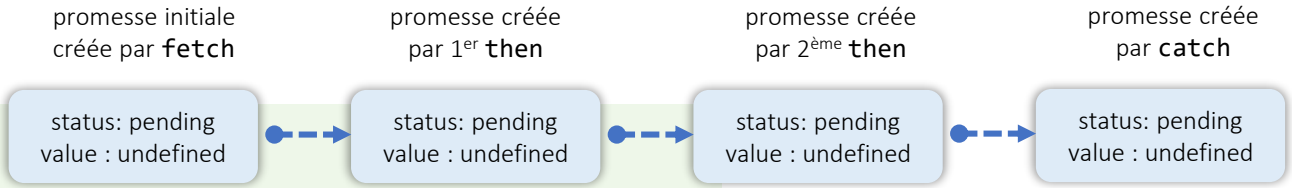
5 Envoie le message **catch** à l'objet promesse retourné précédemment (promesse retournée par le 2ème **then**)

catch est asynchrone, elle renvoie une nouvelle promesse chainée avec la précédente.

```
> node . testOpenWeather.js
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman




```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
6 12 console.log('Terminé !');
```

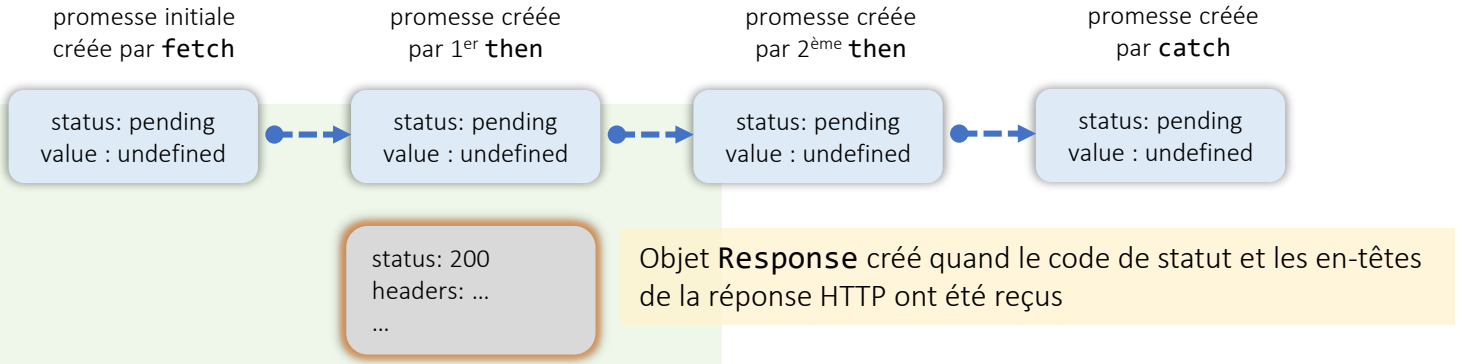
6 Affiche le message `Terminé !` .

```
> node . testOpenWeather.js
Terminé !
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

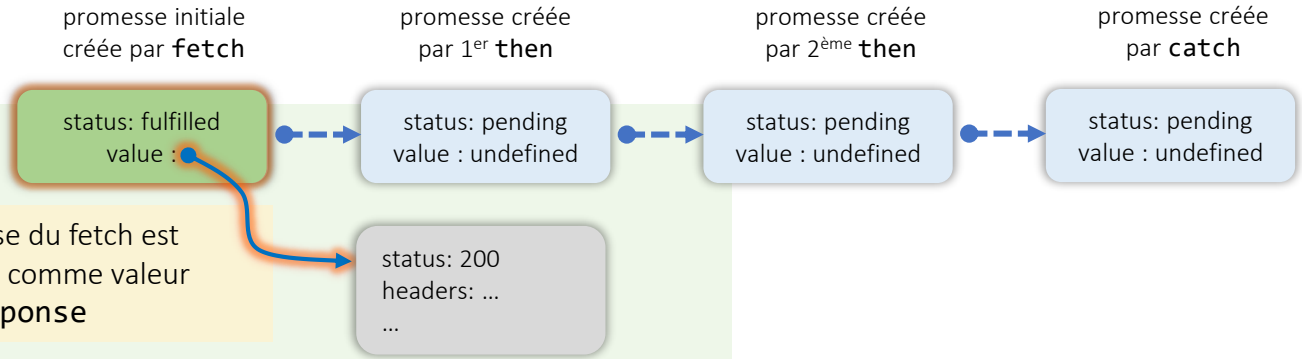



```
> node . testOpenWeather.js
Terminé !
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

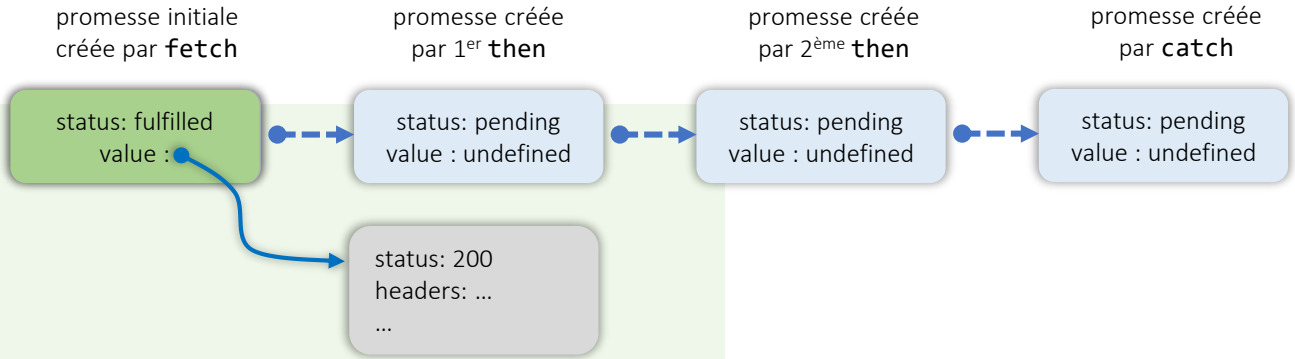



```
> node . testOpenWeather.js
Terminé !
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
                                   (7)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```



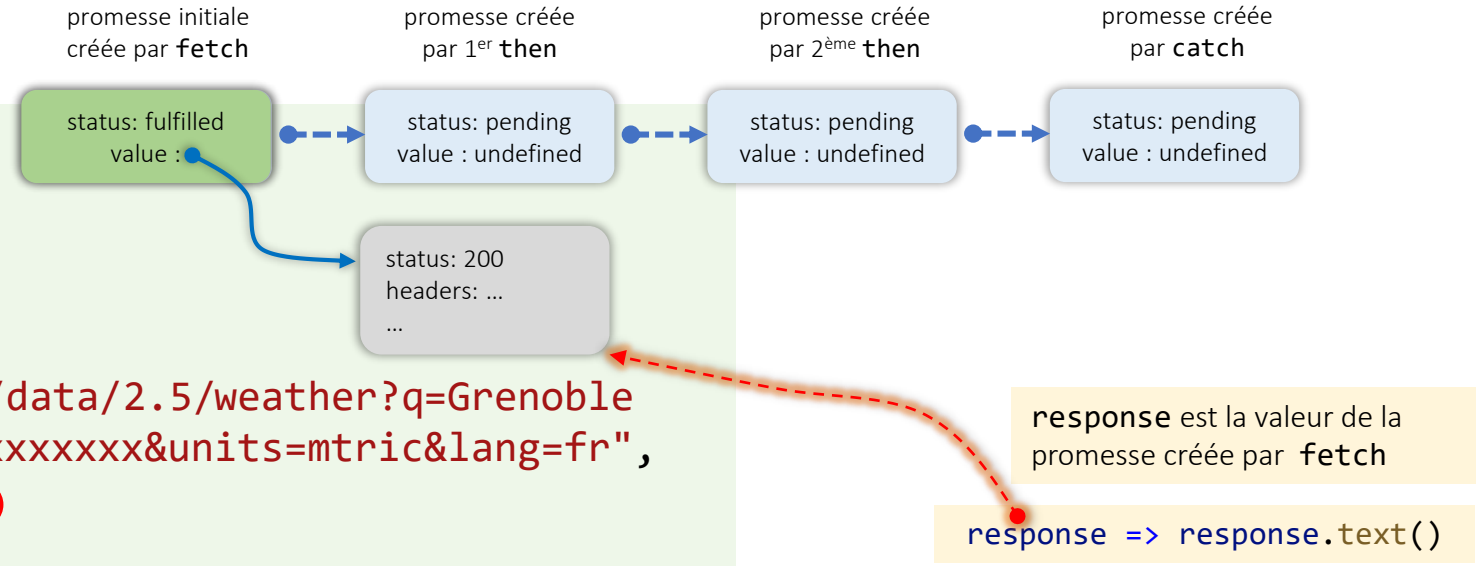
(7) la fonction callback du 1er **then** est exécutée avec comme argument la valeur (l'objet **Response**) attachée à la promesse du **fetch**

```
> node . testOpenWeather.js
Terminé !
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
                                   7
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```




```
> node . testOpenWeather.js
Terminé !
```

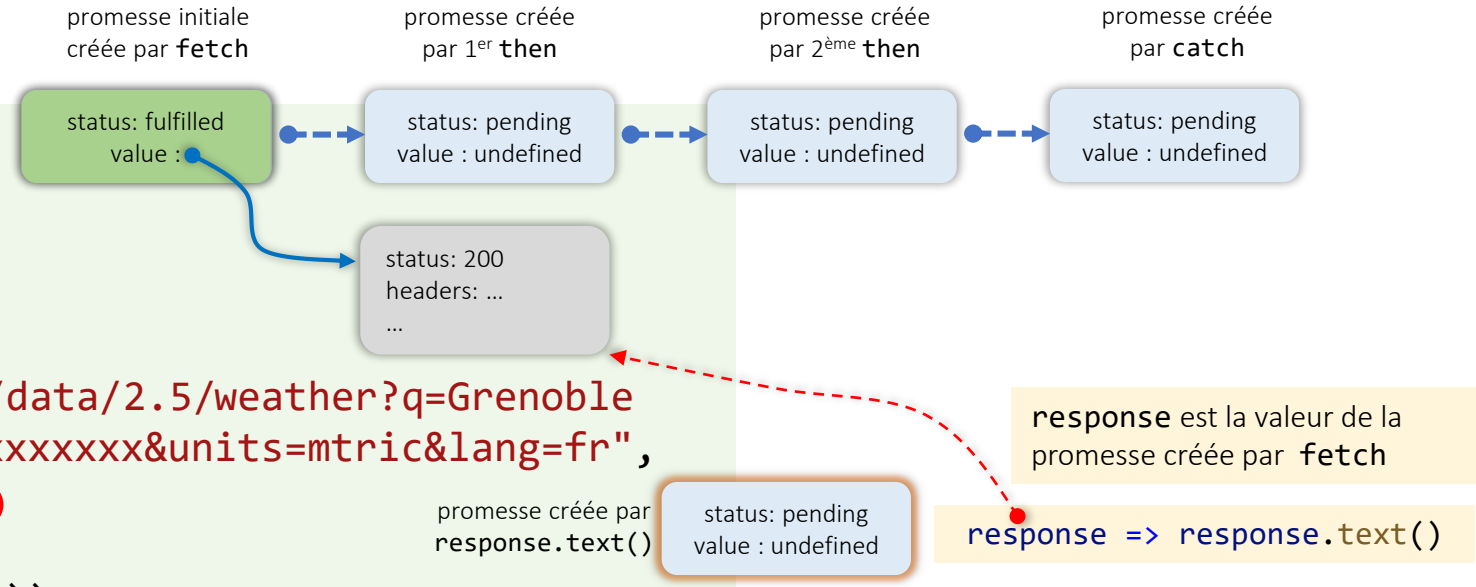
7 la fonction callback du 1er **then** est exécutée avec comme argument la valeur (l'objet **Response**) attachée à la promesse du **fetch**

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

7



La méthode `text()` de l'objet `response` est une méthode asynchrone, elle retourne une promesse qui quand elle sera résolue servira à valuer la promesse du 1er then

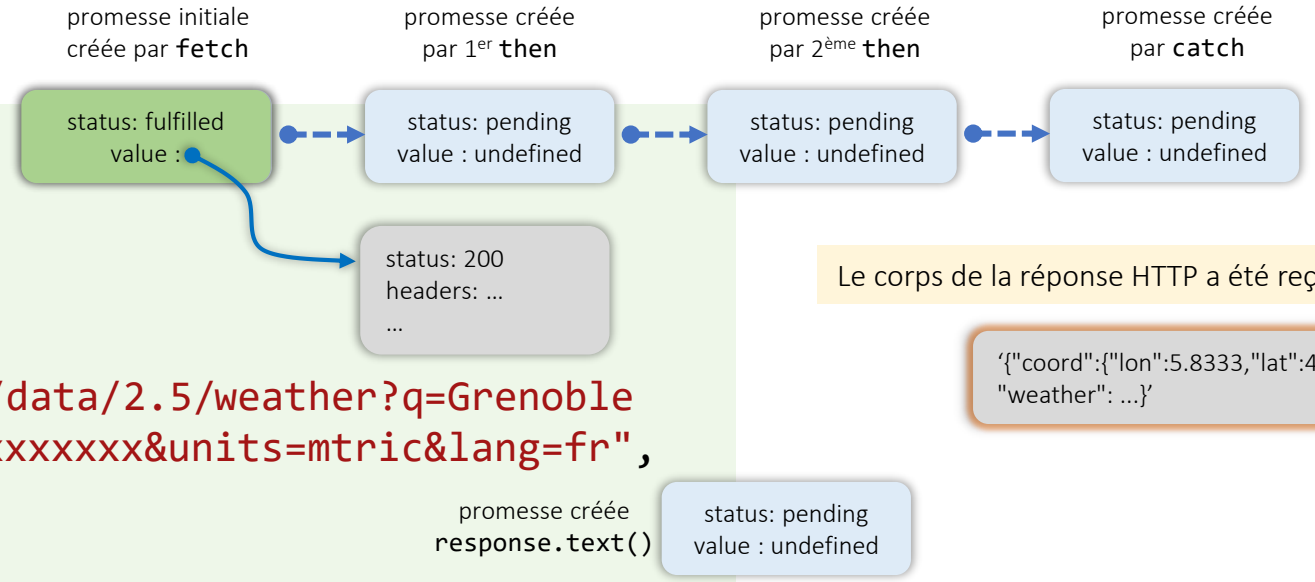
```
> node . testOpenWeather.js
Terminé !
```

7 la fonction callback du 1er `then` est exécutée avec comme argument la valeur (l'objet `Response`) attachée à la promesse du `fetch`

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

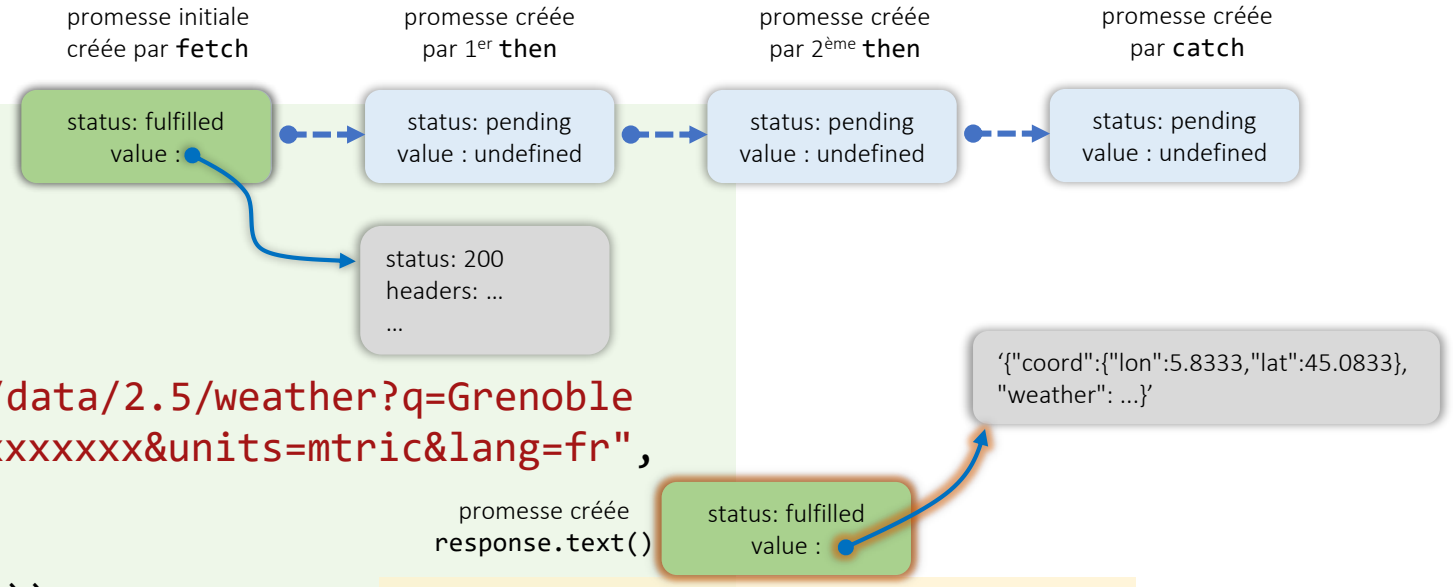



```
> node . testOpenWeather.js
Terminé !
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```



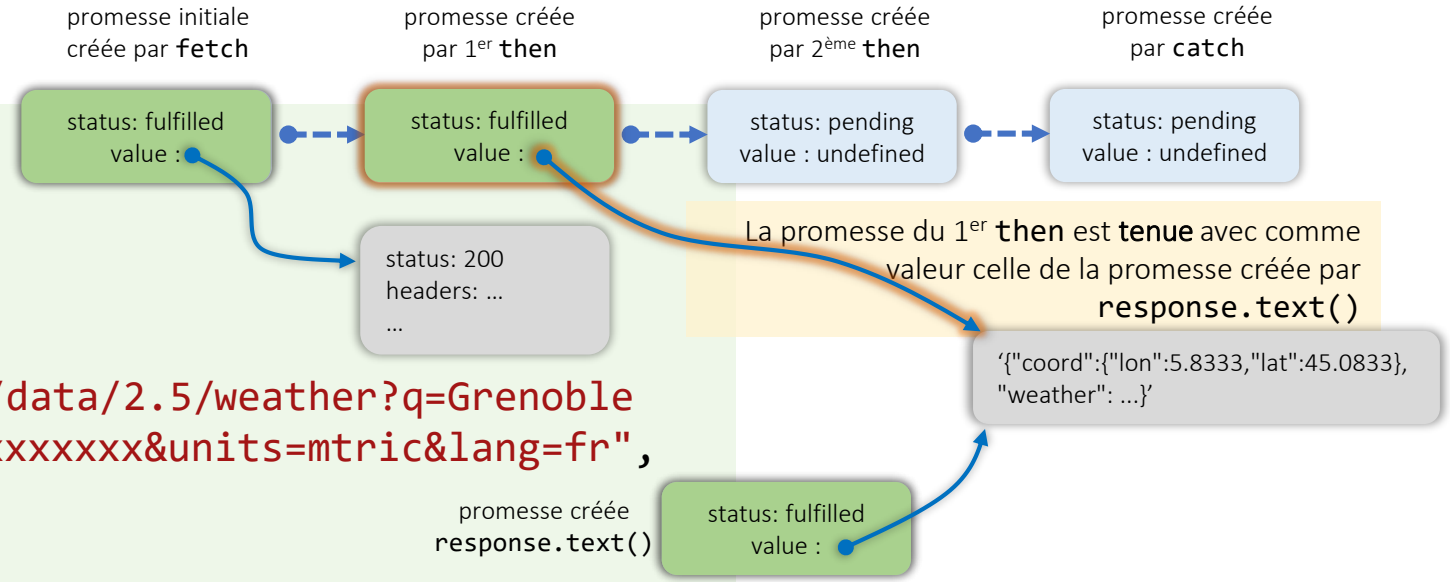
La promesse créée par `response.text()` est tenue sa valeur est le corps de la réponse HTTP au format texte (*raw data*)

```
> node . testOpenWeather.js
Terminé !
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

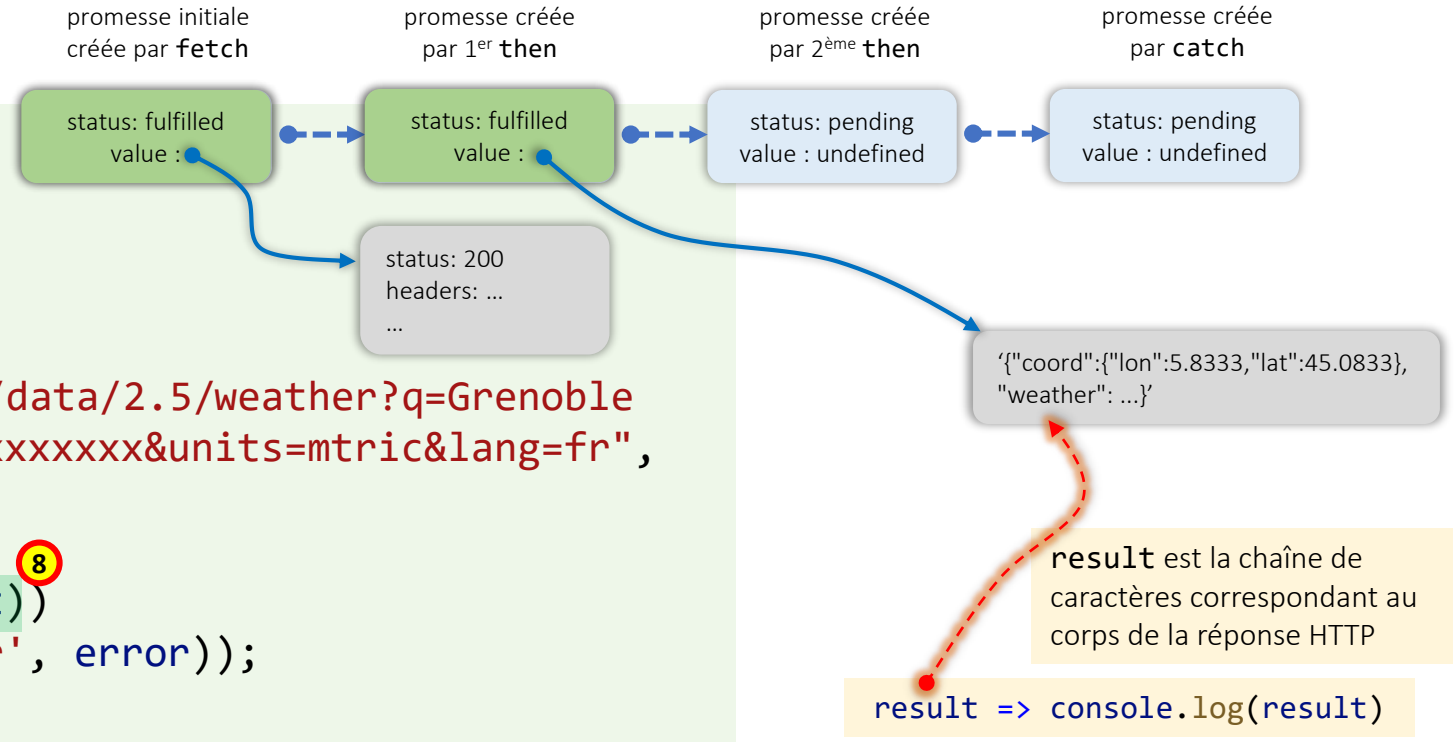



```
> node . testOpenWeather.js
Terminé !
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text()) 8
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```



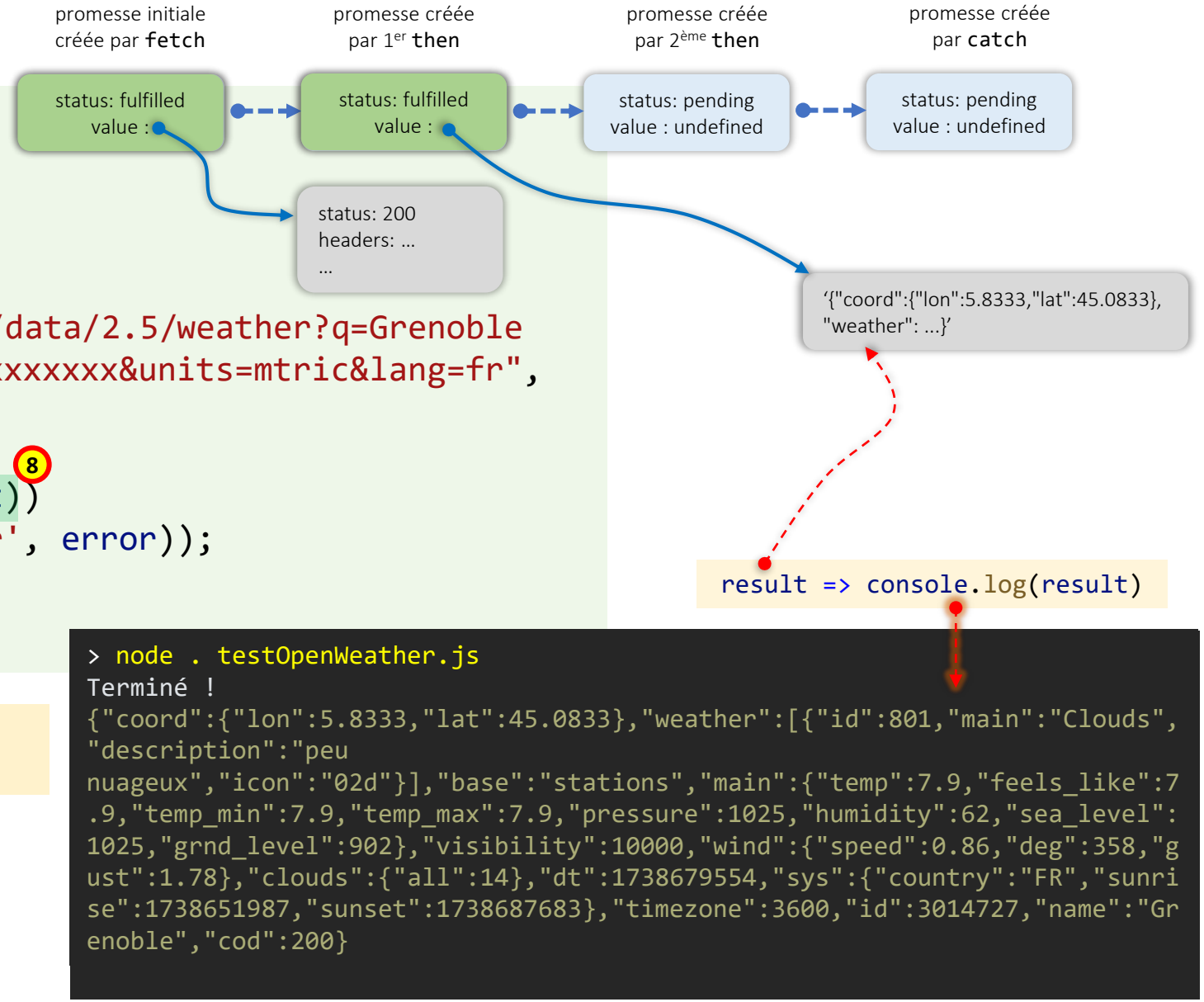

```
> node . testOpenWeather.js
Terminé !
```

8 la fonction callback du 2ème **then** est exécutée avec comme argument la valeur de la promesse précédente

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

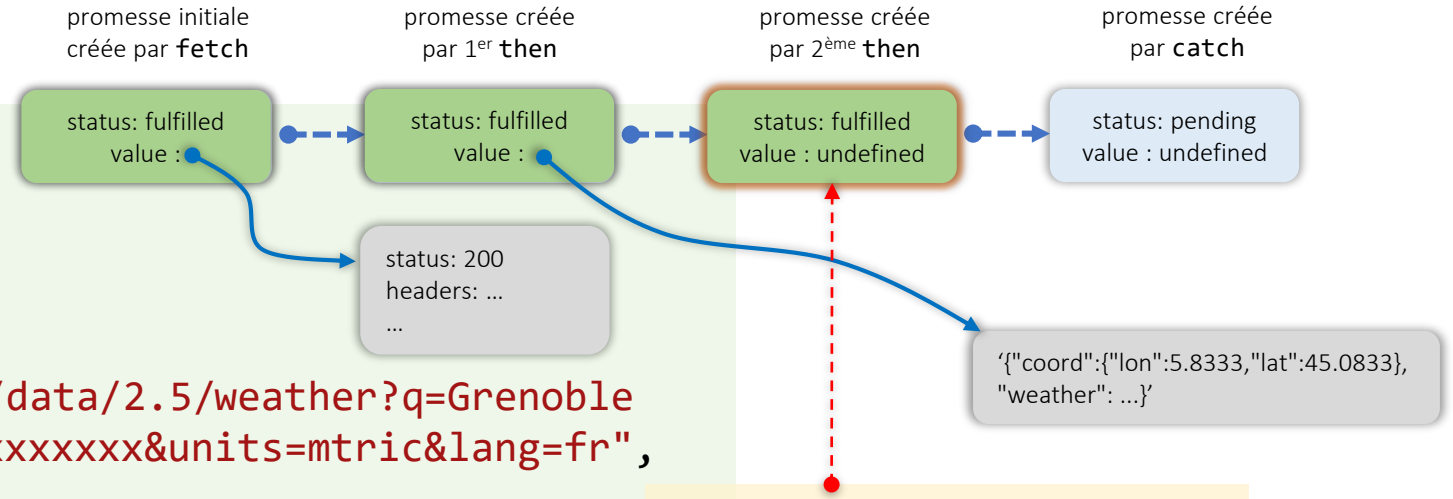


8 la fonction callback du 2ème **then** est exécutée avec comme argument la valeur de la promesse précédente

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```

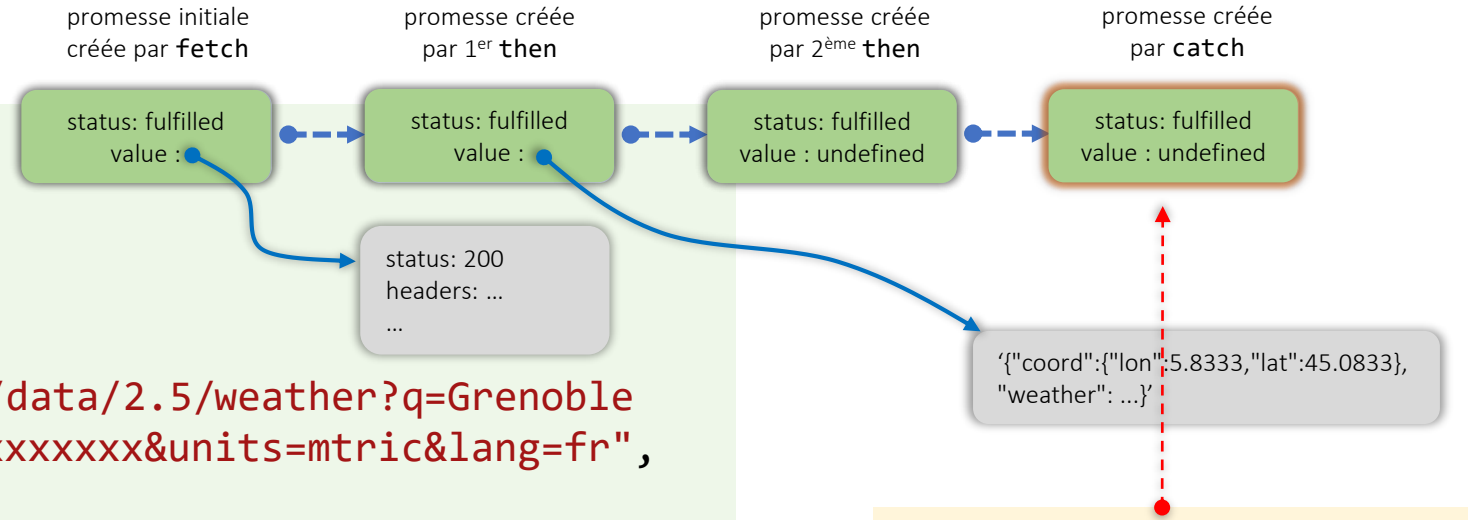



```
> node . testOpenWeather.js
Terminé !
{"coord":{"lon":5.8333,"lat":45.0833},"weather":[{"id":801,"main":"Clouds",
"description":"peu
nuageux","icon":"02d"}],"base":"stations","main":{"temp":7.9,"feels_like":7
.9,"temp_min":7.9,"temp_max":7.9,"pressure":1025,"humidity":62,"sea_level":
1025,"grnd_level":902},"visibility":10000,"wind":{"speed":0.86,"deg":358,"g
ust":1.78},"clouds":{"all":14},"dt":1738679554,"sys":{"country":"FR","sunri
se":1738651987,"sunset":1738687683},"timezone":3600,"id":3014727,"name":"Gr
enoble","cod":200}
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
11
12 console.log('Terminé !');
```



La promesse est résolue. Comme il n'y a pas eu d'erreur dans les traitement mis en place par les promesses précédentes, elle est **tenue** et sa valeur est la valeur de la promesse qui la précède (**undefined**)

```
> node . testOpenWeather.js
Terminé !
{"coord":{"lon":5.8333,"lat":45.0833},"weather":[{"id":801,"main":"Clouds",
"description":"peu
nuageux","icon":"02d"}],"base":"stations","main":{"temp":7.9,"feels_like":7
.9,"temp_min":7.9,"temp_max":7.9,"pressure":1025,"humidity":62,"sea_level":
1025,"grnd_level":902},"visibility":10000,"wind":{"speed":0.86,"deg":358,"g
ust":1.78},"clouds":{"all":14},"dt":1738679554,"sys":{"country":"FR","sunri
se":1738651987,"sunset":1738687683},"timezone":3600,"id":3014727,"name":"Gr
enoble","cod":200}
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.text())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
```

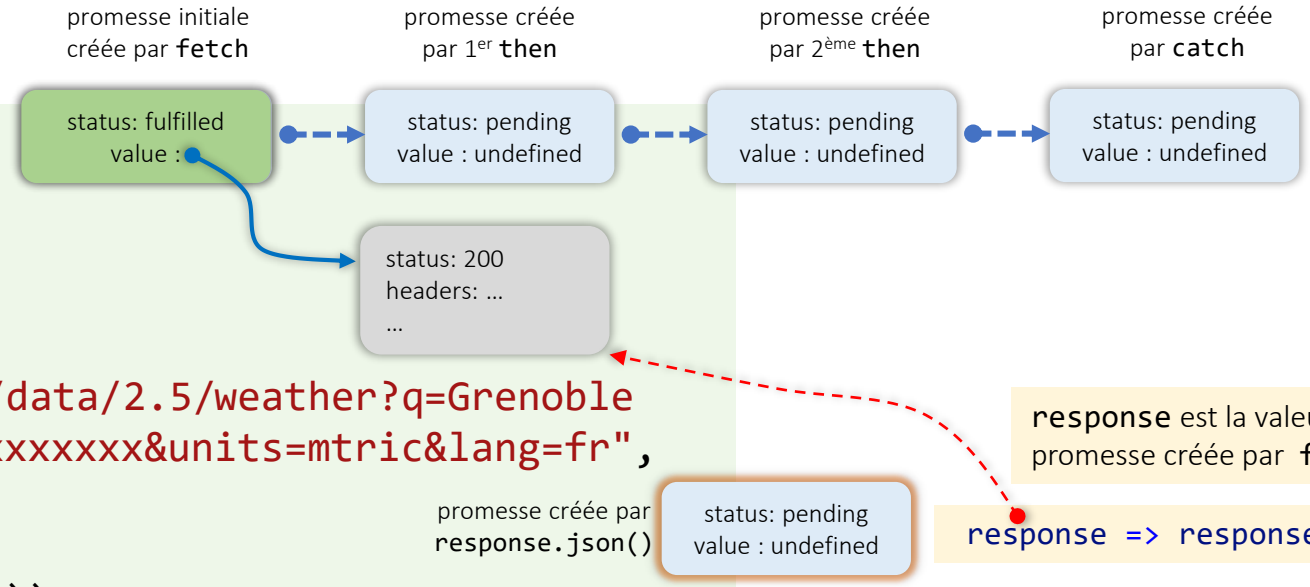
Comment faire pour afficher la description des conditions météo, la température et la date ?

```
> node .\testOpenWeather1.js
Météo à Grenoble
Conditions météo : ciel dégagé
Température : -4.69 + °C
date : 5 févr. 2025 à 01:05:19
>
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.json())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
```



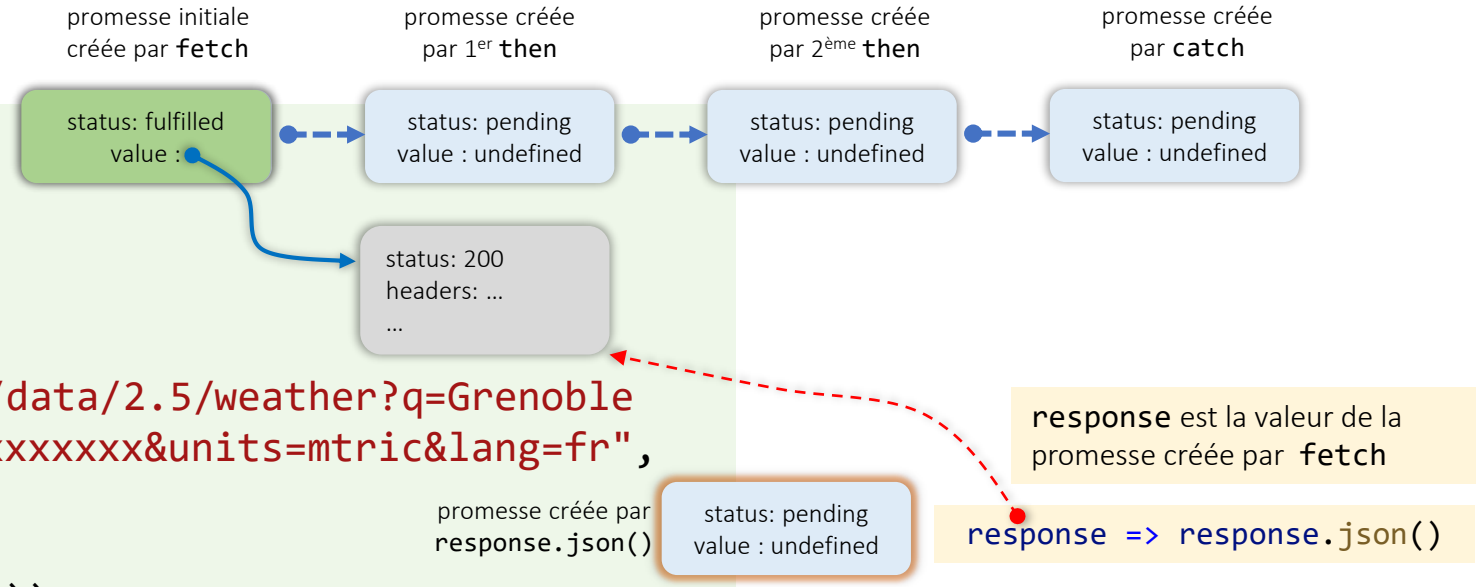
La méthode `json()` de l'objet `response` est une méthode asynchrone, elle retourne une promesse qui quand elle sera résolue servira à valuer la promesse du 1er then

1. Remplacer `response.text()` par `response.json()`

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.json())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
```



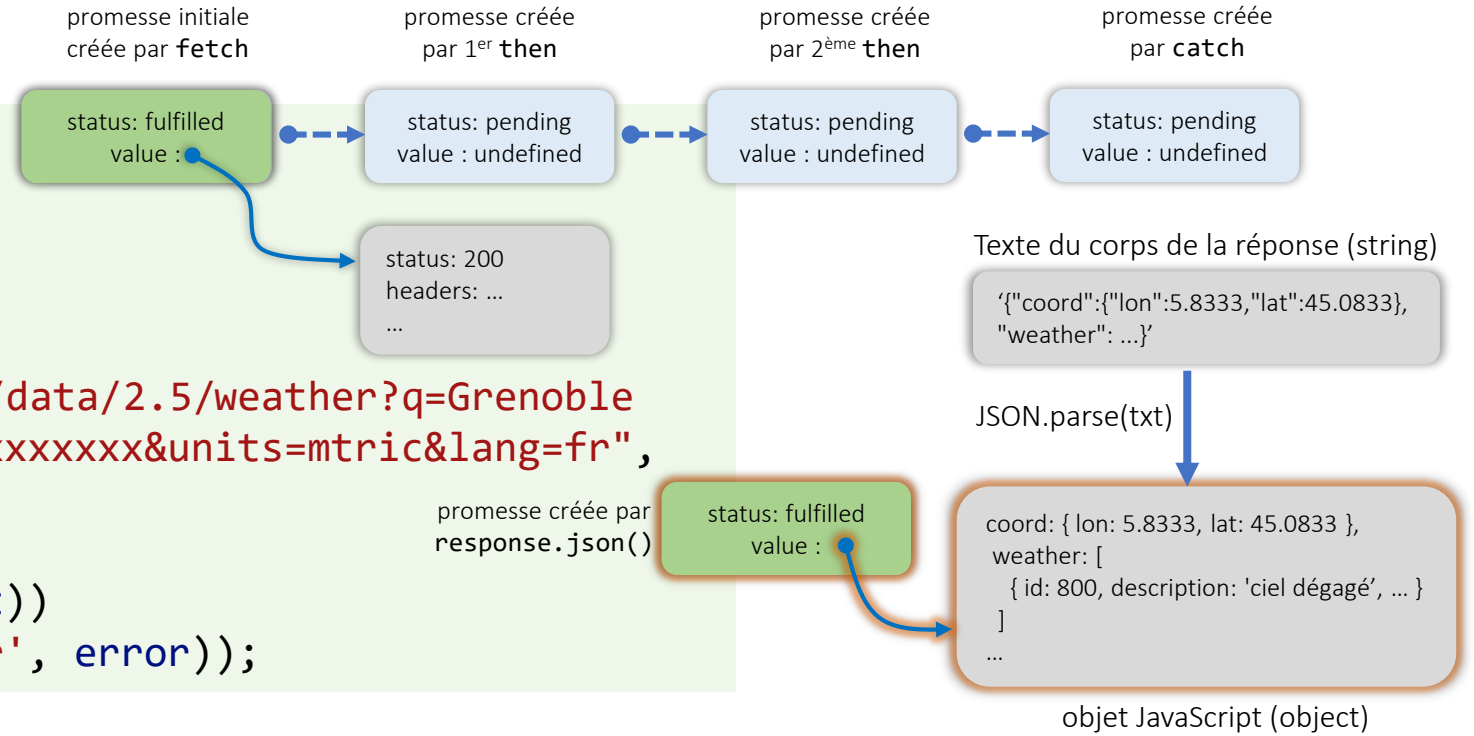
De manière identique à **text()**, la méthode **json()** de l'objet **response** est une méthode asynchrone qui quand elle s'exécute retourne une promesse

```
➢ node .\testOpenWeather1.js
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
         &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.json())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
```



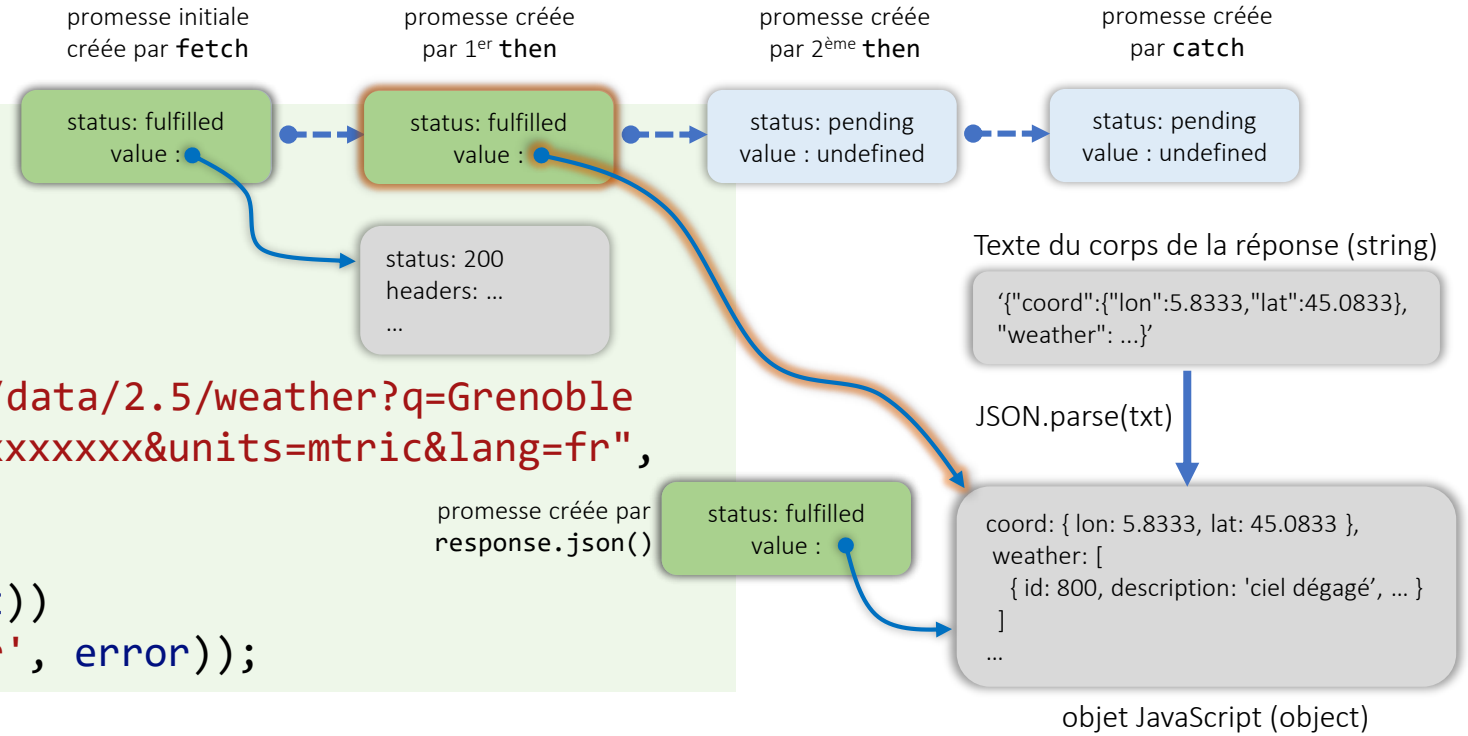
Quand le corps de la réponse HTPP est obtenu, son texte (du JSON) est transformé en un objet JavaScript et la promesse créée par `response.json()` est tenue avec cet objet comme valeur

```
➢ node .\testOpenWeather1.js
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.json())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
```



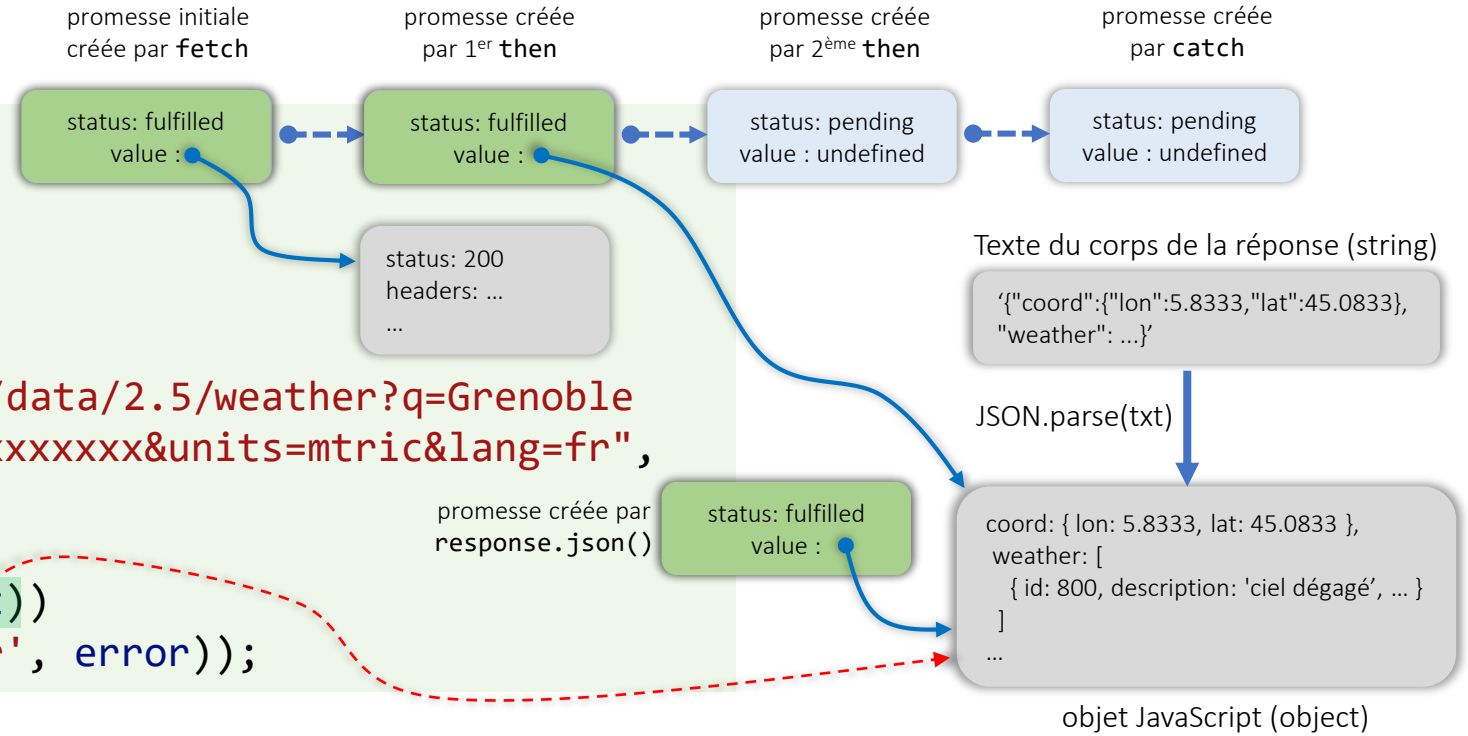
La valeur de cette promesse est transmise à la promesse créée par la méthode then qui devient elle aussi tenue avec comme résultat cet objet

```
➢ node .\testOpenWeather1.js
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble
       &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
     requestOptions)
8      .then(response => response.json())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', error));
```



la fonction associée au 2ème **then** est exécutée avec l'objet JavaScript comme paramètre **result**.

Affichage de l'objet **result**

```
> node .\testOpenWeather1.js
{
  coord: { lon: 5.8333, lat: 45.0833 },
  weather: [
    { id: 800, main: 'Clear', description: 'ciel dégagé', icon: '01n' }
  ],
  ...
}
```

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

Le code généré par Postman

```
1  var requestOptions = {
2      method: 'GET',
3      redirect: 'follow'
4    };
5
6  fetch("https://api.openweathermap.org/data/
         &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxx
     requestOptions)
8      .then(response => response.json())
9      .then(result => console.log(result))
10     .catch(error => console.log('error', err
```

promesse initiale créée par **fetch**

promesse créée par 1er **then**

promesse créée par 2ème **then**

promesse créée par **catch**

status
va

```
{
    console.log("Météo à Grenoble");
    console.log(`Conditions météo : ${result.weather[0].description}`);
    console.log(`Température : ${result.main.temp} + °C`);
    const timestamp = result.dt * 1000;
    const date = new Date(timestamp);
    const dateString = date.toLocaleDateString("fr-FR", {
          day: "numeric",
          month: "short",
          year: "numeric",
    });
    const timeString = date.toLocaleTimeString(undefined, {
          hour: "2-digit",
          minute: "2-digit",
          second: "2-digit",
    });
    console.log(`date : ${dateString} à ${timeString}`);
}
```

Pour afficher la description des conditions météo, la température et la date, il suffit de remplacer l'instruction **`console.log(result)`** par un bloc de code accédant directement aux propriétés de l'objet **`result`**

```
> node .\testOpenWeather1.js

Météo à Grenoble
Conditions météo : ciel dégagé
Température : -4.69 + °C
date : 5 févr. 2025 à 01:05:19
>
```