

Correction TP 02 AWA

API fetch

Question 1.7 – récupérer la température en °C

The screenshot shows the Postman interface with a GET request to `https://api.openweathermap.org/data/2.5/weather?q=Grenoble&appid=87c1b851-...`. The response status is 200 OK. The response body is shown in JSON format, with the following structure:

```
9      "main": "clear",
10     "description": "clear sky",
11     "icon": "01d"
12   },
13 ],
14 "base": "stations",
15 "main": {
16   "temp": 269.02,
17   "feels_like": 269.02,
18   "temp_min": 269.02,
19   "temp_max": 269.02,
20   "pressure": 1013,
21   "humidity": 65,
22   "wind_speed": 3.6,
23   "wind_deg": 135,
24   "clouds": {
25     "all": 0
26   },
27   "visibility": 10000,
28   "pop": 0,
29   "sys": {
30     "type": 2,
31     "id": 5239,
32     "country": "FR",
33     "sunrise": 1645488000,
34     "sunset": 1645524000
35   },
36   "timezone": 3600,
37   "id": 2998102,
38   "name": "Grenoble",
39   "coord": {
40     "lat": 45.764,
41     "lon": 4.835
42   }
43 }
```

A callout box points to the value `269.02` in the `"temp"` field, stating: "Par défaut la température est en ° Kelvin".

Question 1.7 – récupérer la température en °C

<https://openweathermap.org/current>

Call current weather data

How to make an API call

API call

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

Parameters

Parameter	Required	Description
lat	required	Latitude. If you need the geocoder to automatic convert city names and zip-codes to geo coordinates and the other way around, please use our Geocoding API
lon	required	Longitude. If you need the geocoder to automatic convert city names and zip-codes to geo coordinates and the other way around, please use our Geocoding API
appid	required	Your unique API key (you can always find it on your account page under the "API key" tab)
mode	optional	Response format. Possible values are <code>xml</code> and <code>html</code> . If you don't use the <code>mode</code> parameter format is JSON by default. Learn more
units	optional	Units of measurement. <code>standard</code> , <code>metric</code> and <code>imperial</code> units are available. If you do not use the <code>units</code> parameter, <code>standard</code> units will be applied by default. Learn more
lang	optional	You can use this parameter to get the output in your language. Learn more

Units of measurement

`standard`, `metric`, and `imperial` units are available. [List of all API parameters with available units.](#)

Parameters

`units` optional `standard`, `metric`, `imperial`. When you do not use the `units` parameter, format is `standard` by default.

Temperature is available in Fahrenheit, Celsius and Kelvin units.

- For temperature in Fahrenheit use `units=imperial`
- For temperature in Celsius use `units=metric`
- Temperature in Kelvin is used by default, no need to use `units` parameter in API call

[List of all API parameters with units openweathermap.org/weather-data](#)

Examples of API calls:

Standard

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

Consulter la documentation de l'API

Units of measurement

`standard`, `metric`, and `imperial` units are available. [List of all API parameters with available units.](#)

Parameters

`units` optional `standard`, `metric`, `imperial`. When you do not use the `units` parameter, format is `standard` by default.

Temperature is available in Fahrenheit, Celsius and Kelvin units.

- For temperature in Fahrenheit use `units=imperial`
- For temperature in Celsius use `units=metric`
- Temperature in Kelvin is used by default, no need to use `units` parameter in API call

[List of all API parameters with units openweathermap.org/weather-data](#)

Examples of API calls:

Standard

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

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

Question 1.7 – récupérer la température en °C

The screenshot shows the Postman interface with a GET request to `https://api.openweathermap.org/data/2.5/weather?q=Grenoble&appid=87c1b...&units=metric`. The parameters are listed in a table below:

Key	Value
q	Grenoble
appid	87c1b...
units	metric

The response body is shown in JSON format, with the temperature field set to `-2.46`.

```
9      "main": "Clear",
10     "description": "clear sky",
11     "icon": "01d"
12   },
13 ],
14 "base": "stations",
15 "main": {
16   "temp": -2.46,
17   "feels_like": -2.46,
18   "temp_min": -2.46,
19   "temp_max": -2.46,
20   "pressure": 1027,
21   "humidity": 78,
22   "sea_level": 0,
23   "sea_level": 0
24 }
```

La requête HTTP étant un GET le paramètre unit et sa valeur sont insérés dans l'URL sous la forme **nomDuParamètre=valeur** et séparé du paramètre précédent par **&**

Rajout du paramètre **units** avec la valeur **metric**

Constater que la température est en reçe est maintenant exprimée en ° Celsius

Renvoyer a nouveau la requête

Question 1.8 – récupérer la description en français

The screenshot shows the Postman interface with an HTTP request configured. The URL is `https://api.openweathermap.org/data/2.5/weather?q=Grenoble&ta...`. The method is GET. The query parameters are:

Key	Value
q	Grenoble
appid	87c1b851d6f616e66b84d35c85fe43de
units	metric
lang	fr

The response status is 200 OK, Time: 256 ms, Size: 861 B. The response body is shown in JSON format.

Code snippet (JavaScript - Fetch):

```
1 var requestOptions = {
2   method: 'GET',
3   redirect: 'follow'
4 };
5
6 fetch("https://api.openweathermap.org/
  data/2.5/weather?q=Grenoble&
  appid=87c1b851d6f616e66b84d35c85fe4
  3de&units=metric&lang=fr",
  requestOptions)
7 .then(response => response.text())
8 .then(result => console.log(result))
9 .catch(error => console.log('error',
  error));
```

Question 1.8 – récupérer la description en français

<https://openweathermap.org/current>

Consulter la documentation de l'API

The screenshot shows the 'How to make an API call' section of the OpenWeather API documentation. It includes an API call example, a list of parameters, and a sidebar with navigation links. A red box highlights the 'lang' parameter in the parameters list, and a red circle with the number '2' is next to it. Another red box highlights the 'How to make an API call' link in the sidebar, with a red circle and the number '1' next to it. A red arrow points from the 'lang' parameter to the 'Multilingual support' section in the adjacent screenshot. A red circle with the number '3' is located at the bottom right of the page.

The screenshot shows the 'Multilingual support' section of the OpenWeather API documentation. It explains that the 'lang' parameter can be used to get the output in a specific language. It includes an API call example with the 'lang' parameter set to 'fr'. A red box highlights the 'fr' value in the API call, and a red circle with the number '4' is next to it. A red arrow points from the 'lang' parameter in the previous screenshot to this section. A red circle with the number '3' is located at the bottom right of the page.

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

Question 1.8 – récupérer la description en français

The screenshot shows the Postman interface for a GET request to the OpenWeatherMap API. The URL is `https://api.openweathermap.org/data/2.5/weather?q=Grenoble&appid=87c1...&units=metric&lang=fr`. The 'Query Params' table is as follows:

Key	Value
q	Grenoble
appid	87c1...
units	metric
lang	fr

The response body is a JSON object:

```
1 {
2   "coord": {
3     "lon": 5.8333,
4     "lat": 45.0833
5   },
6   "weather": [
7     {
8       "id": 800,
9       "main": "Clear",
10      "description": "ciel dégagé",
11      "icon": "01d"
12    }
13  ],
14  "base": "stat",
15  "m": ...

```

Renvoyer a nouveau la requête

Rajout du paramètre **lang** avec la valeur **fr**

La requête HTTP étant un GET le paramètre unit et sa valeur sont insérés dans l'URL sous la forme **nomDuParamètre=vaLeur** et séparé du paramètre précédent par **&**

Constater que la description des conditions météorologiques est maintenant reçue en français

Question 2.3 : utiliser fetch, récupérer la réponse en JSON

The image shows a workflow in Postman and VS Code. In Postman, a GET request is configured for the OpenWeather API with query parameters: q=Grenoble, appid=87c1b851d6f616e66b84d35c85fe43de, units=metric, and lang=fr. A callout box with a red circle containing the number 1 points to the 'Send' button, with the text: "Génération du code JavaScript faisant la requête en utilisant l'API **fetch**".

Clicking 'Send' generates a JavaScript code snippet in the 'Code snippet' panel. A callout box with a red circle containing the number 2 points to this panel, with the text: "Recopier le code proposé par Postman dans un fichier testOpenWeather.js".

The code snippet is as follows:

```
1 var requestOptions = {
2   method: 'GET',
3   redirect: 'follow'
4 };
5
6 fetch("https://api.openweathermap.org/data/2.5/weather?q=Grenoble&appid=87c1b851d6f616e66b84d35c85fe43de&units=metric&lang=fr", requestOptions)
7   .then(response => response.text())
8   .then(result => console.log(result))
9   .catch(error => console.log('error', error));
```

In VS Code, the file 'testOpenWeather.js' is open, showing the same code snippet pasted into it. The Explorer sidebar shows the file structure with 'testOpenWeather.js' selected under the 'TP02_AWA' folder.

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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```

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

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

```
> 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=metric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```

promesse initiale
créée par fetch

status: pending
value : undefined

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 **&**

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

```
> 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```

promesse initiale
créée par fetch

status: pending
value : undefined

promesse créée
par 1^{er} then

status: pending
value : undefined

Fonction fléchée (*arrow function*)

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

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

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

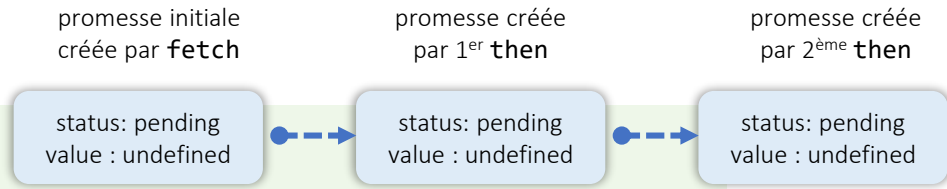
then est asynchrone, elle renvoie une nouvelle promesse
chaînée avec la précédente.

```
> 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```



Fonction fléchée (*arrow function*)

↕

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

cette fonction sera exécutée quand la promesse créée par le 1^{er} then sera tenue

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

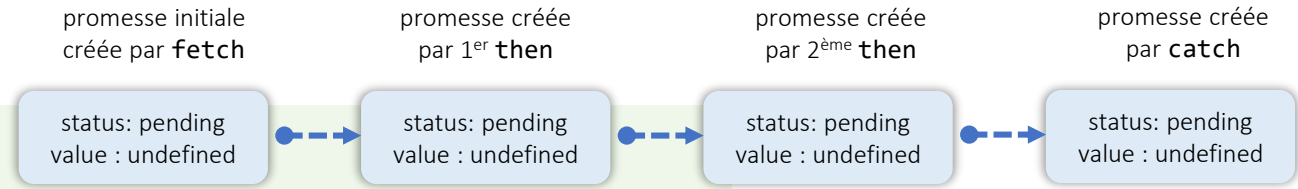
then est asynchrone, elle renvoie une nouvelle promesse chaînée avec la précédente.

```
> 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```



Fonction fléchée (arrow function)



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

cette fonction sera exécutée quand la promesse créée par le 2^{ème} then sera tenue

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

catch est asynchrone, elle renvoie une nouvelle promesse chaînée 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```

promesse initiale
créée par fetch

status: pending
value : undefined

promesse créée
par 1^{er} then

status: pending
value : undefined

promesse créée
par 2^{ème} then

status: pending
value : undefined

promesse créée
par catch

status: pending
value : undefined

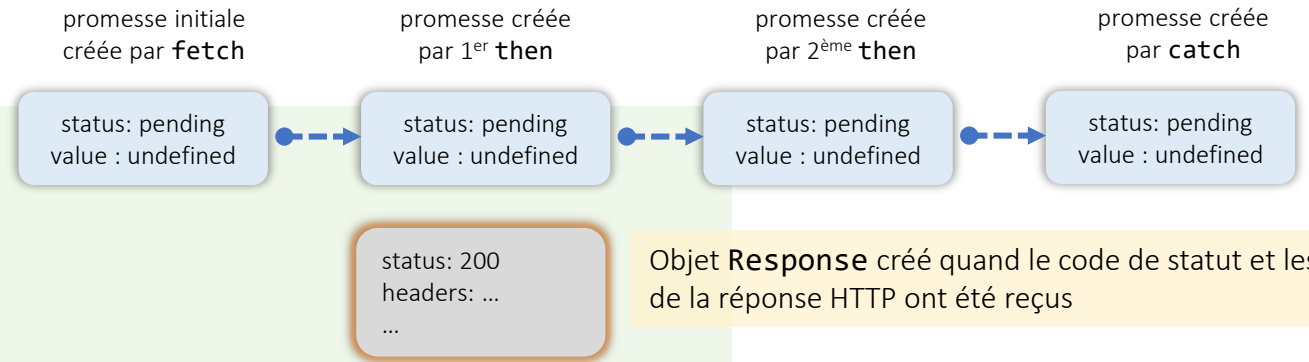
```
> node . testOpenWeather.js
Terminé !
```

6 Affiche le message 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```

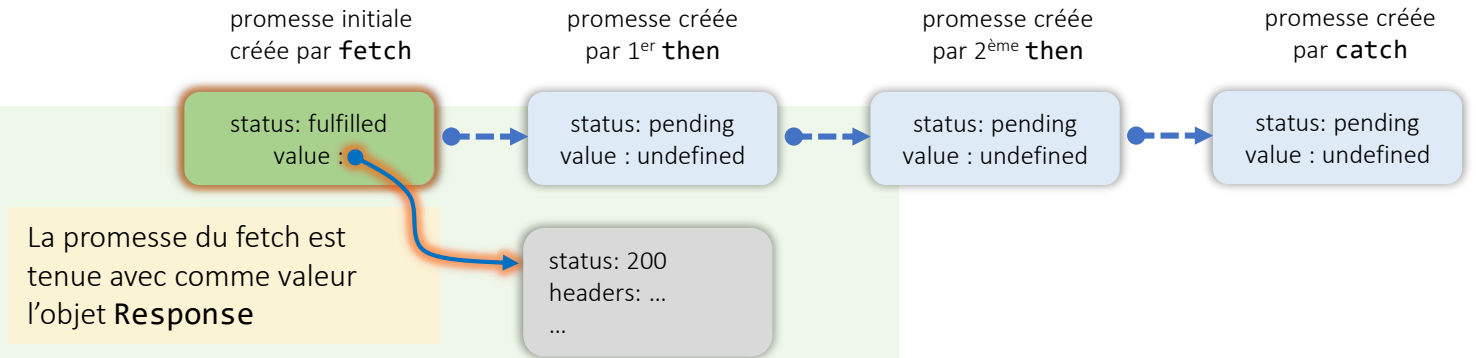


```
> node . testOpenWeather.js
Terminé !
```

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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```

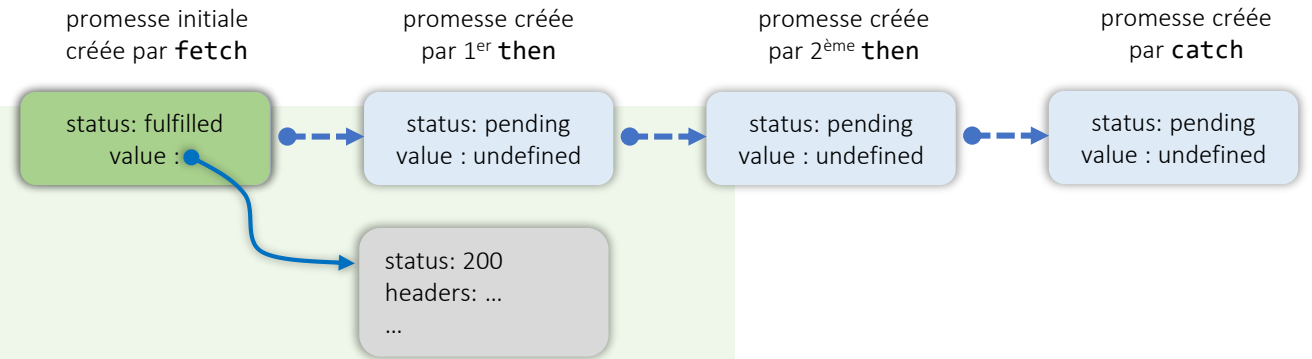


```
> node . testOpenWeather.js
Terminé !
```

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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```



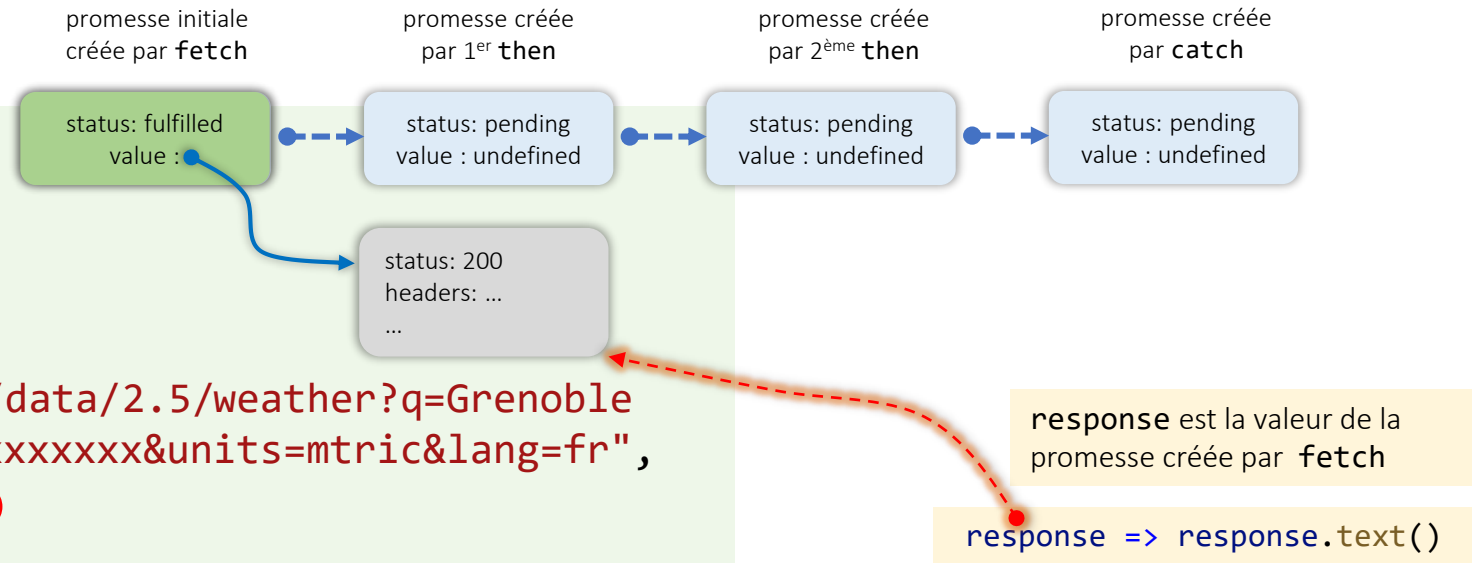
7 la fonction callback du 1^{er} then est exécutée avec comme argument la valeur (l'objet **Response**) attachée à la promesse du **fetch**

```
> 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```



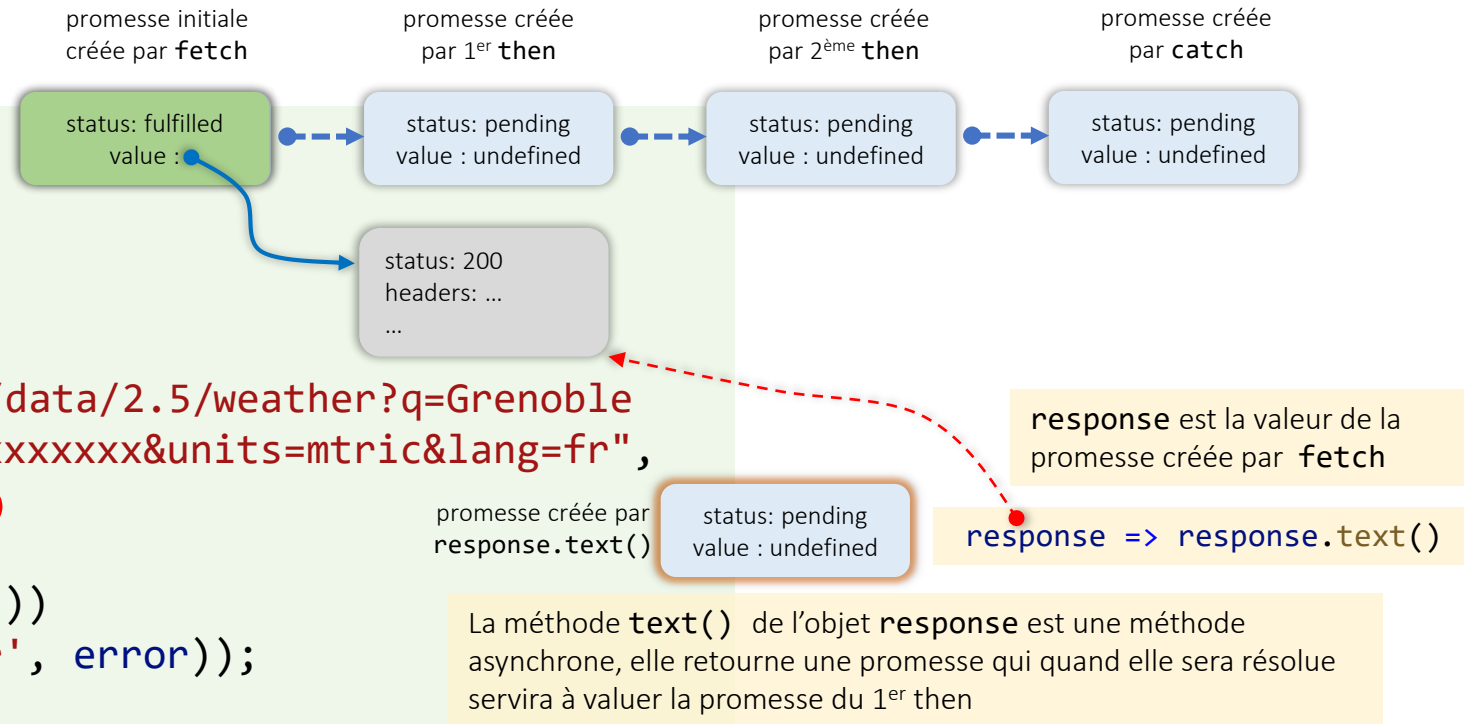
7 la fonction callback du 1^{er} then est exécutée avec comme argument la valeur (l'objet **Response**) attachée à la promesse du **fetch**

```
> 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```



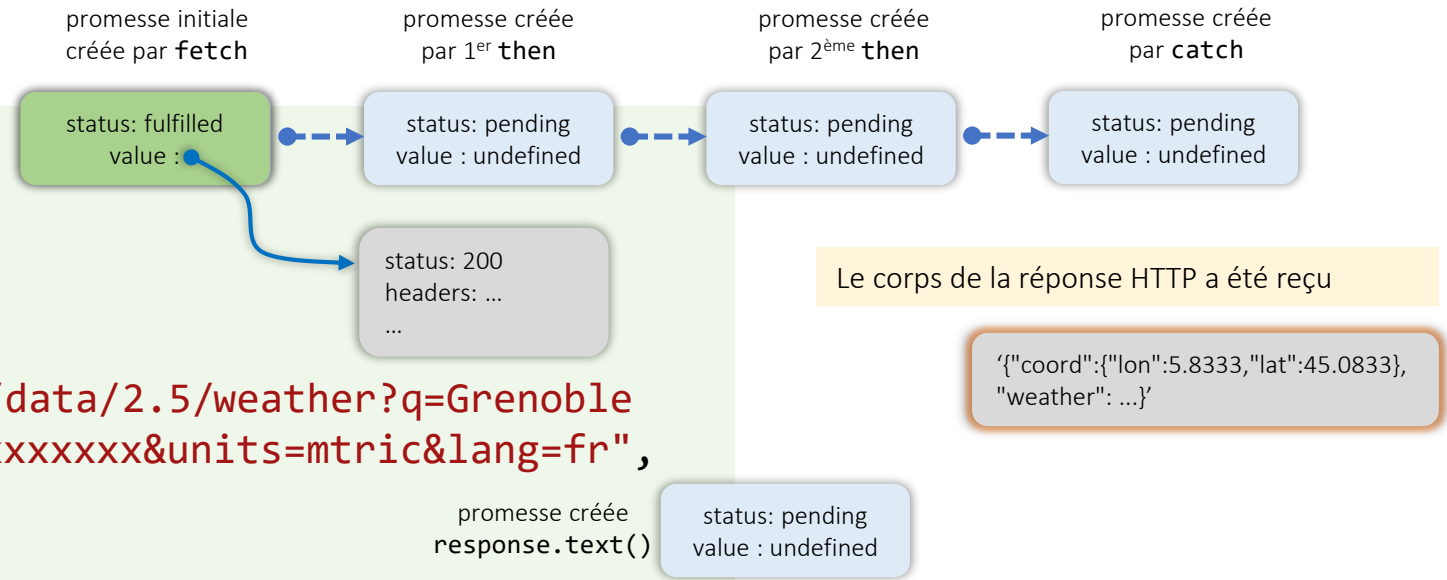
7 la fonction callback du 1^{er} then est exécutée avec comme argument la valeur (l'objet `Response`) attachée à la promesse du `fetch`

```
> 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```

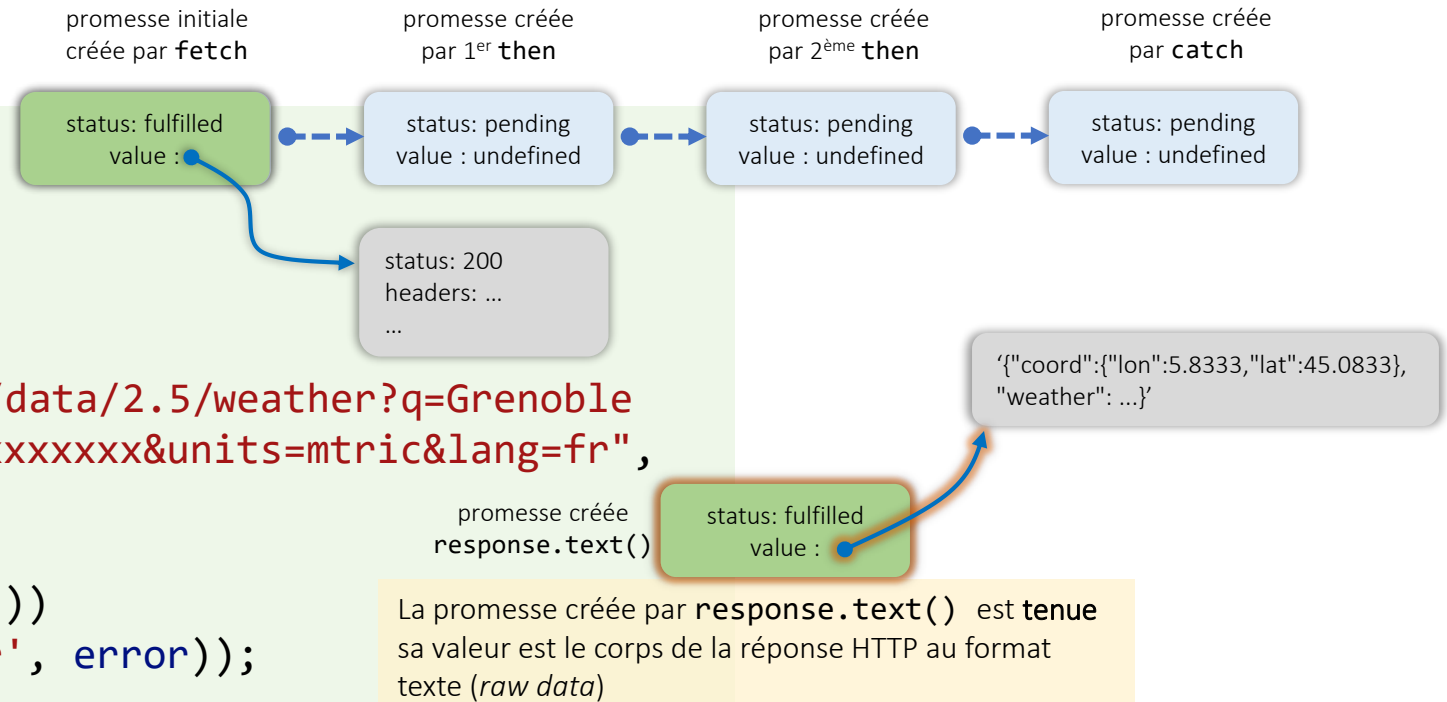


```
> node . testOpenWeather.js
Terminé !
```

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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```

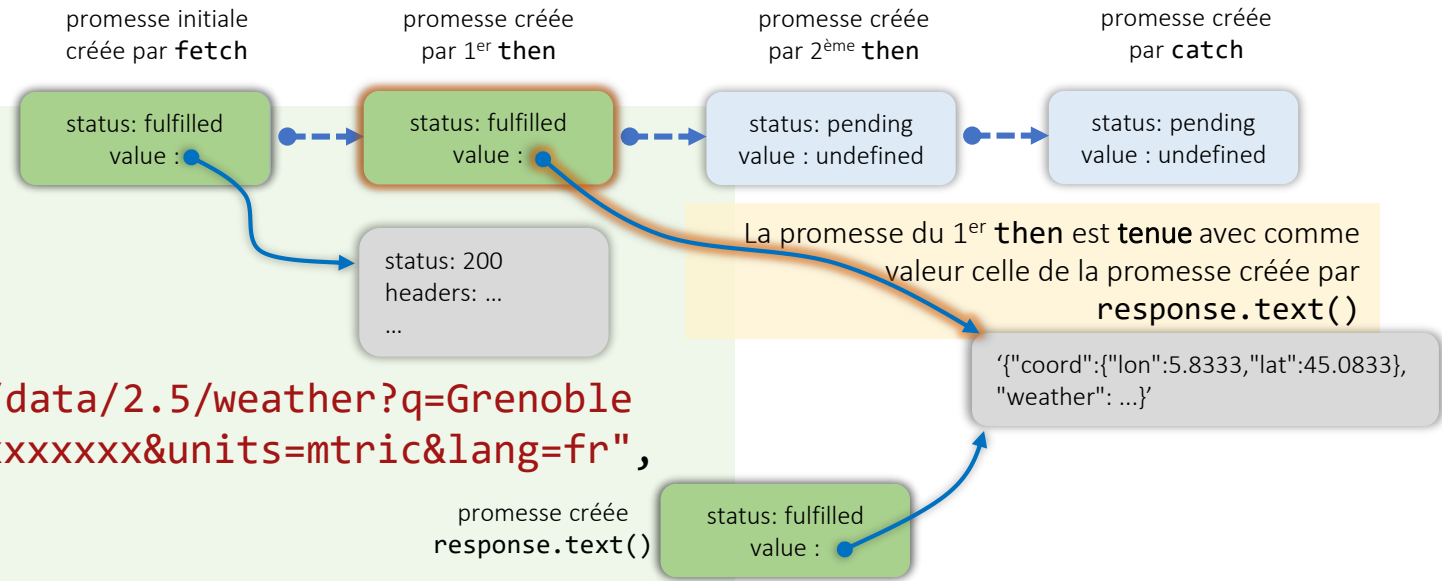


```
> node . testOpenWeather.js
Terminé !
```

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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```

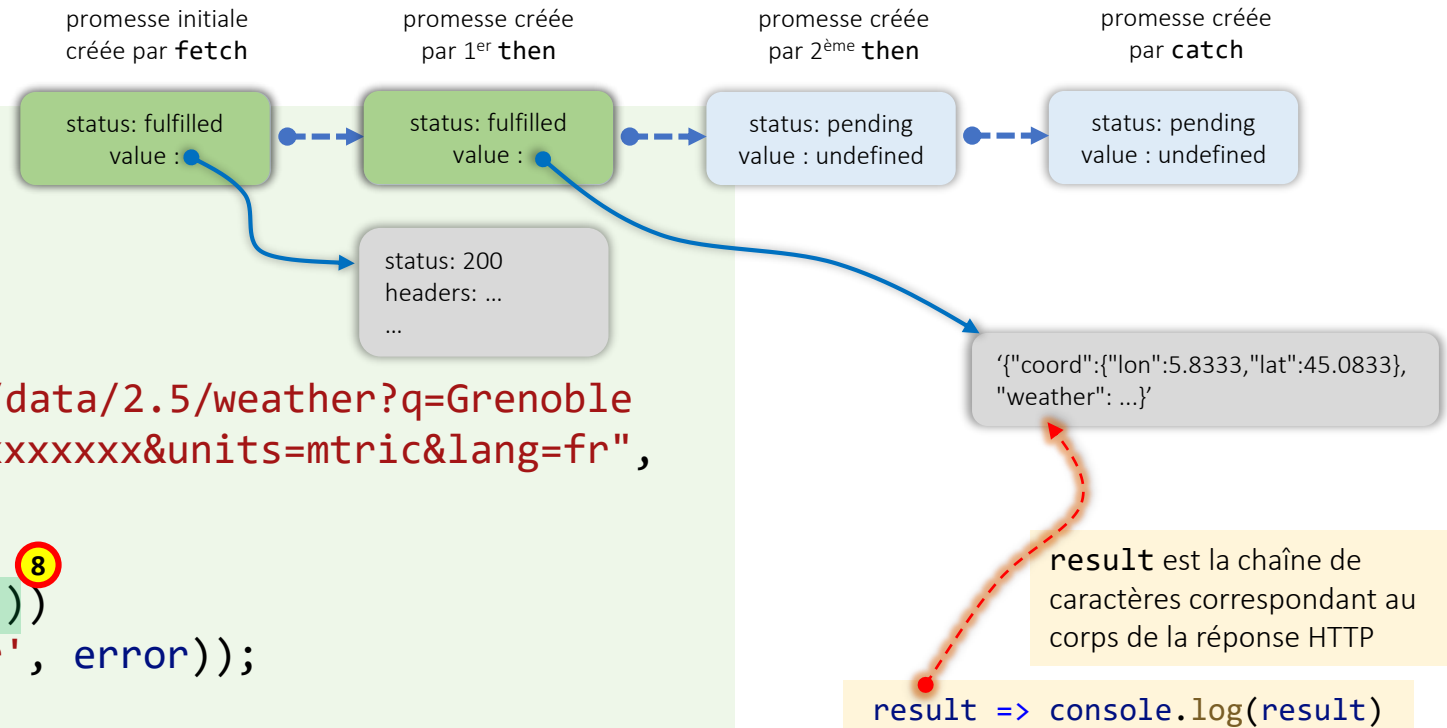


```
> node . testOpenWeather.js
Terminé !
```

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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```



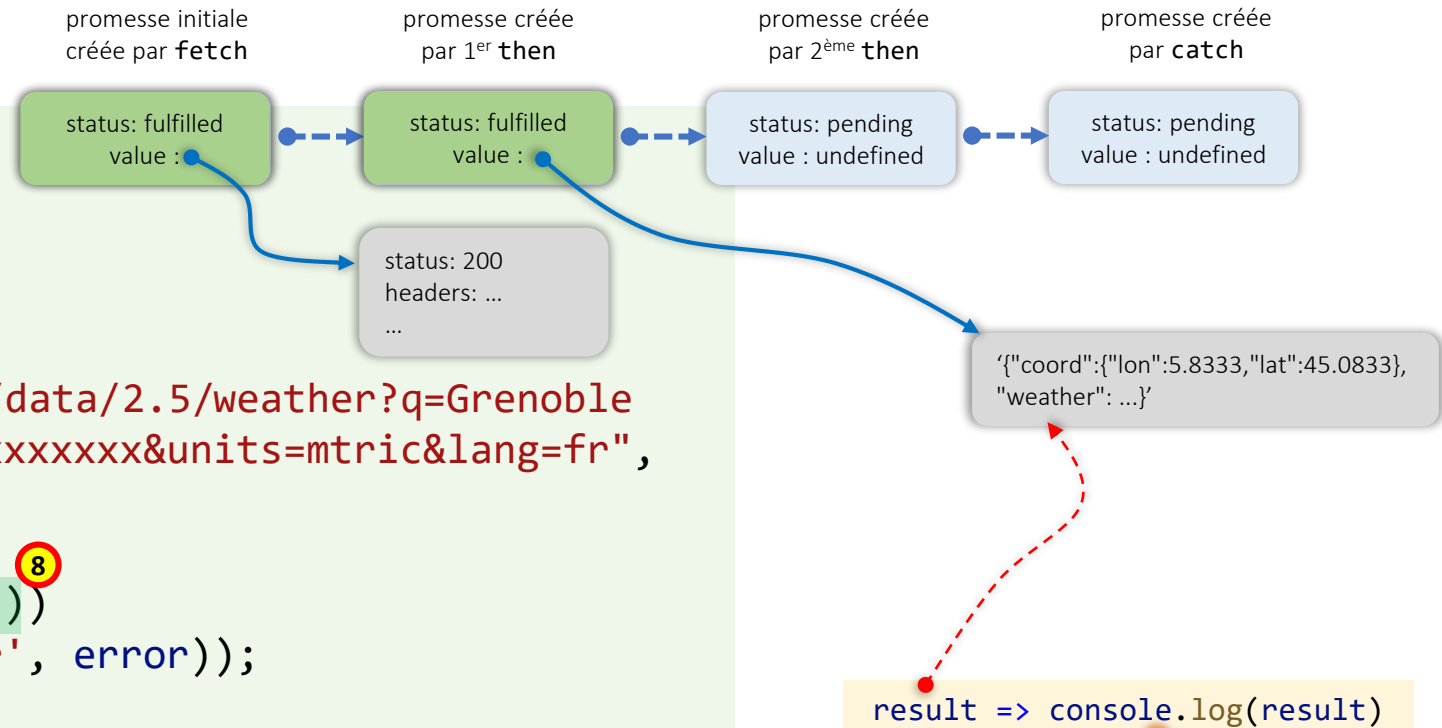
8 la fonction callback du 2^{ème} then est exécutée avec comme argument la valeur de la promesse précédente

```
> 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```



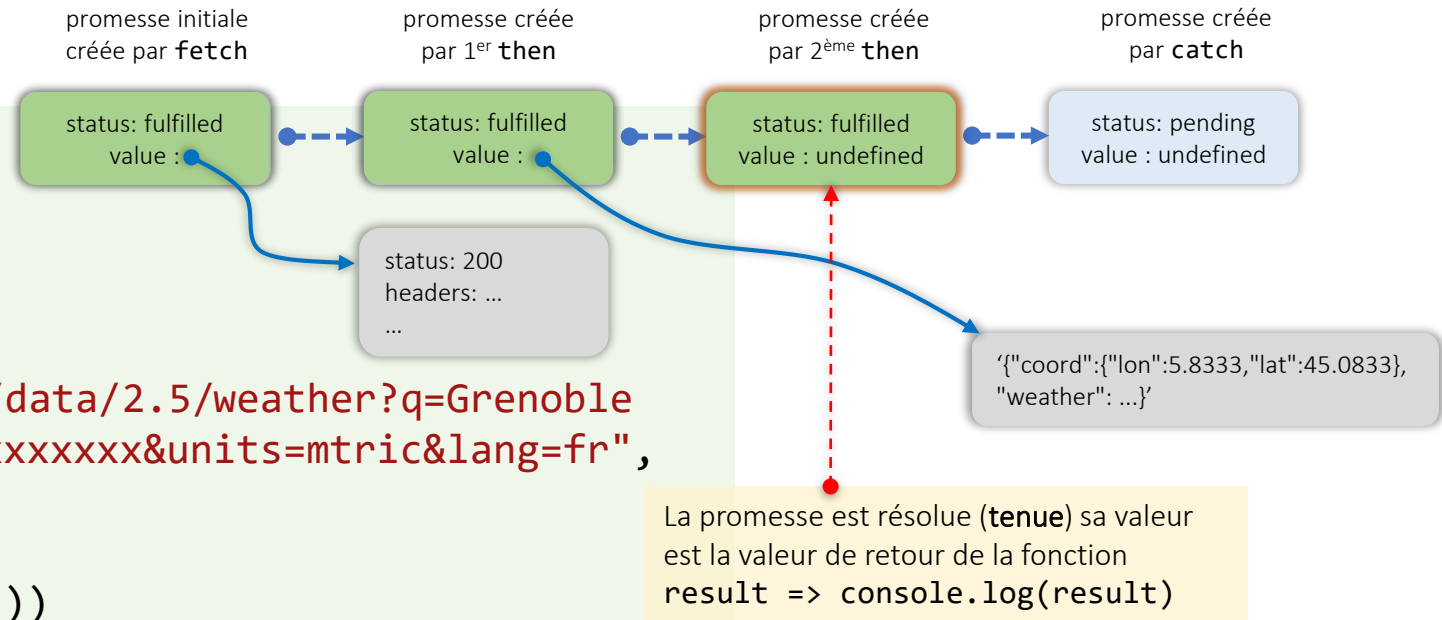
8 la fonction callback du 2^{ème} then est exécutée avec comme argument la valeur de la promesse précédente

```
> 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
12 console.log('Terminé !');
```

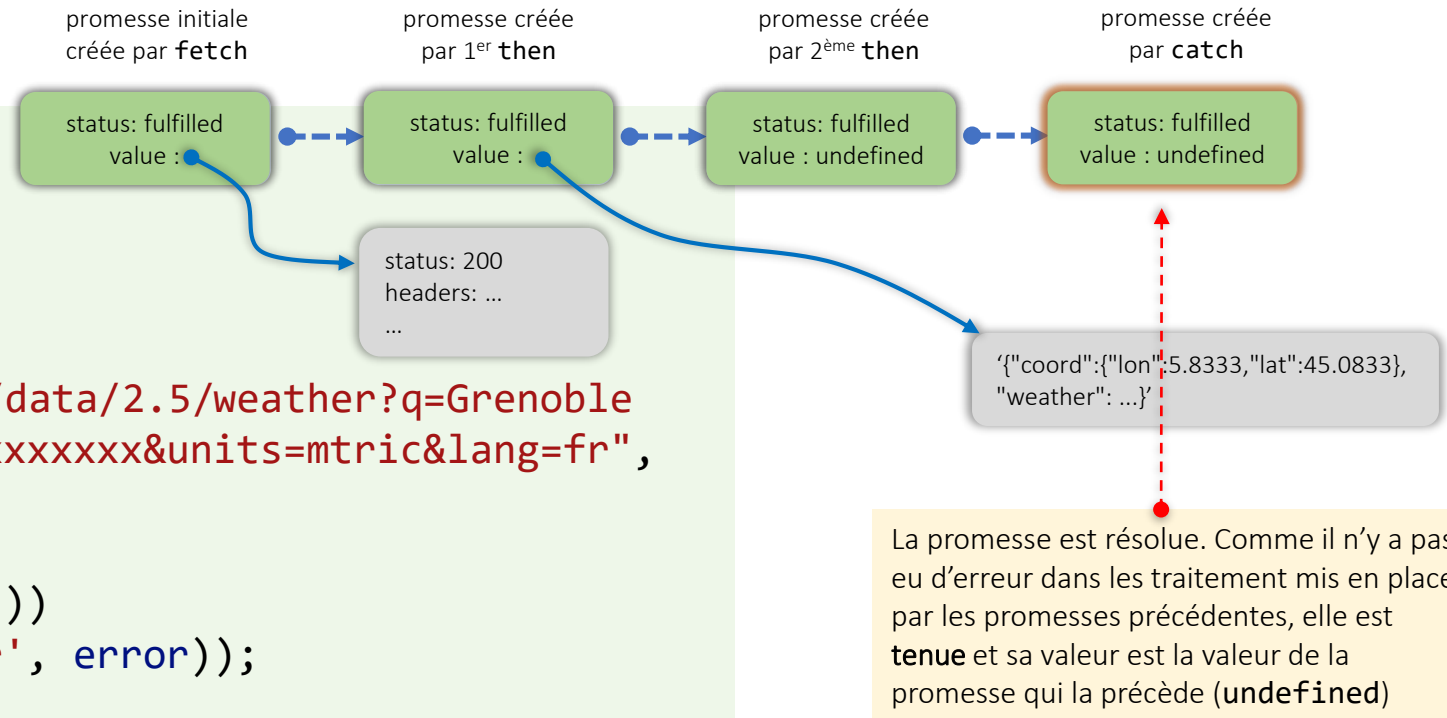


```
> node . testOpenWeather.js
Terminé !
{"coord":{"lon":5.8333,"lat":45.0833},"weather":[{"id":801,"main":"Clouds",
"description":"peu nuageux","icon":"02d"}],"base":"stations","main":{"temp":7.9,"feels_like":7.9,"temp_min":7.9,"temp_max":7.9,"pressure":1025,"humidity":62,"sea_level":1025,"grnd_level":902},"visibility":10000,"wind":{"speed":0.86,"deg":358,"gust":1.78},"clouds":{"all":14},"dt":1738679554,"sys":{"country":"FR","sunrise":1738651987,"sunset":1738687683},"timezone":3600,"id":3014727,"name":"Grenoble","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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
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,"gust":1.78},"clouds":{"all":14},"dt":1738679554,"sys":{"country":"FR","sunrise":1738651987,"sunset":1738687683},"timezone":3600,"id":3014727,"name":"Grenoble","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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.text())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
```

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

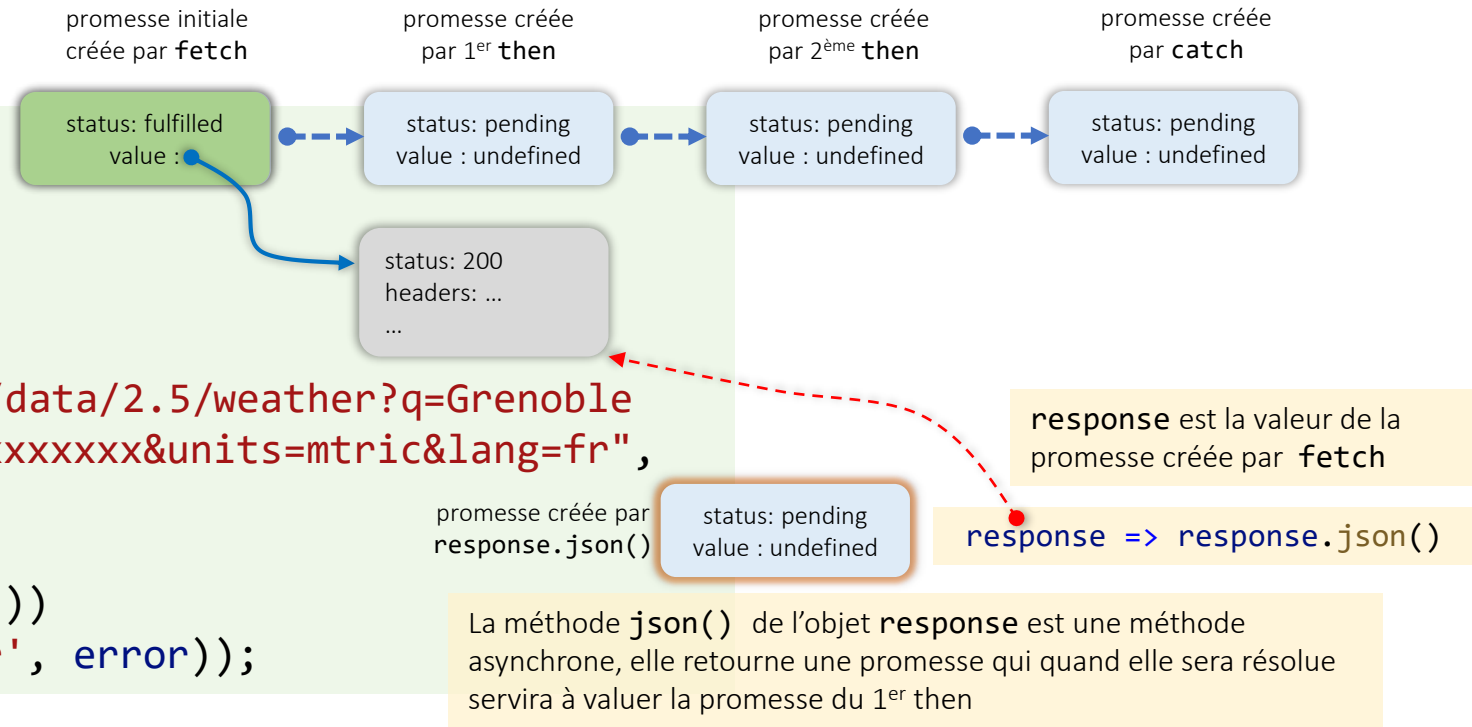


```
> node .\testOpenWeather1.js
Météo à Grenoble
Conditions météo : ciel dégagé
Température : -4.69 + °C
date : 5 févr. 2025 à 01:05:19
>
```


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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.json())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
```

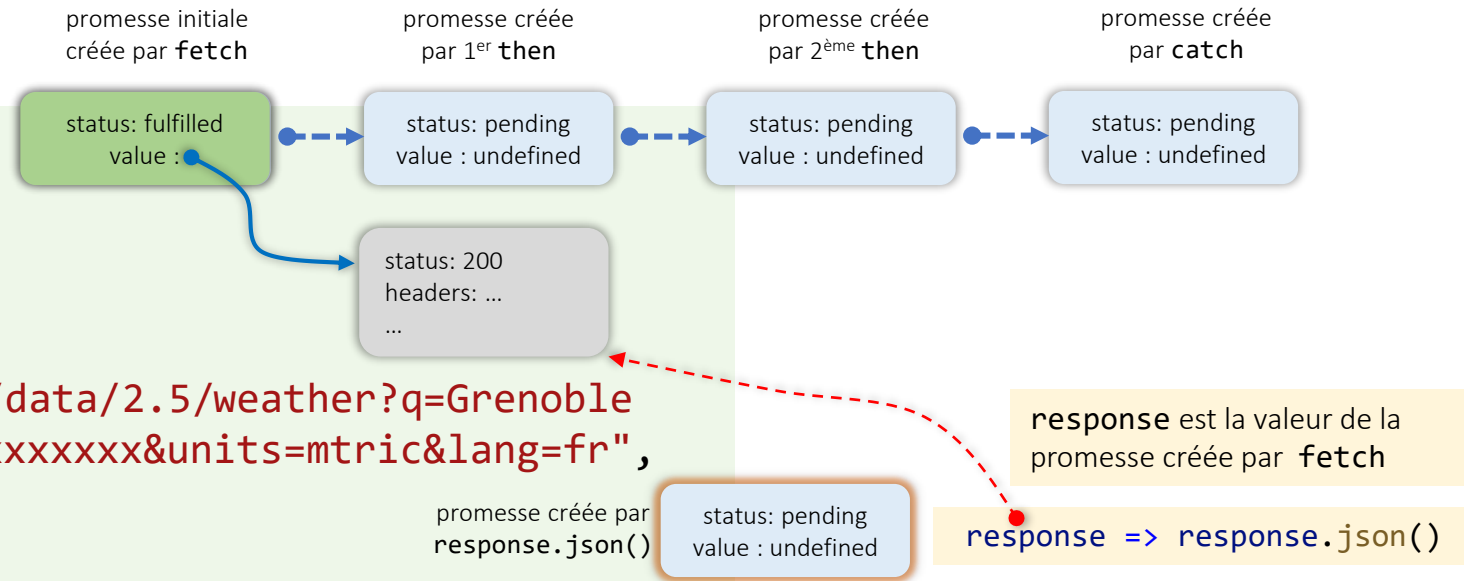


1 Remplacer `response.text()` par `response.json()`

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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.json())
10  .then(result => console.log(result))
11  .catch(error => console.log('error', error));
```



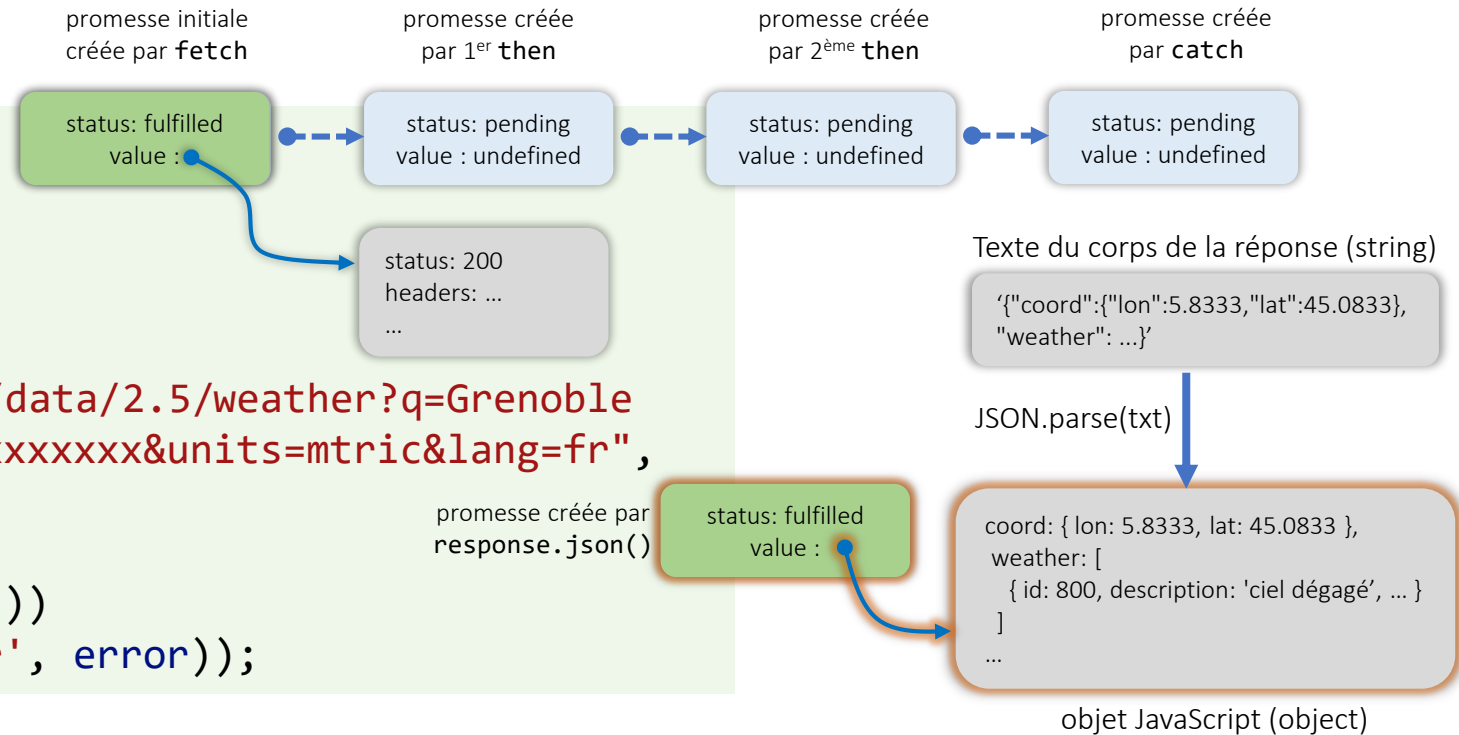
De manière identique à `text()`, la méthode `json()` de l'objet `response` est une méthode asynchrone qui quand elle s'exécute retourne une promesse

```
> 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
7   &appid=87c1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&units=mtric&lang=fr",
8   requestOptions)
9   .then(response => response.json())
10  .then(result => console.log(result))
   .catch(error => console.log('error', error));
```



Quand le corps de la réponse HTTP 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
```