

Cours : Manipulation de données JSON avec HTML & JavaScript

JSON — Rappel

Le format **JSON (JavaScript Object Notation)** est un format de données textuel, très utilisé pour échanger des informations entre applications. Il ressemble à des objets JavaScript, mais **ce n'est que du texte**.

Exemple :

```
[
  { "nom": "Nom0", "prenom": "Prénom0", "tel": "0601020304" },
  { "nom": "Nom1", "prenom": "Prénom1", "tel": "0605060708" }
]
```

Ce JSON contient un **tableau** (`[...]`) de **deux objets** (`{...}`) représentant des personnes.

Créer votre espace de travail

1. Créez un dossier **TP_JSON**
2. Vous travaillerez avec les fichiers :

- `donnees.json`
 - `index.html`
-

Partie 1 — Créer un fichier JSON et l'ouvrir dans un navigateur

Étape 1 : Créer `donnees.json`

Dans votre dossier **TP_JSON**, créez un fichier : `donnees.json`

Copiez-y ces données :

```
[
  { "nom": "Nom0", "prenom": "Prénom0", "tel": "0601020304" },
  { "nom": "Nom1", "prenom": "Prénom1", "tel": "0605060708" }
]
```

Tester l'ouverture du JSON

Objectif : montrer que JSON = texte.

- Clic droit sur `donnees.json` → **Ouvrir avec** → Chrome
- Clic droit sur `donnees.json` → **Ouvrir avec** → Firefox
- Chrome l'affiche sous forme de texte brut.
- Firefox affiche le JSON de manière claire (et repliable).

Partie 2 — Intégrer les données JSON directement dans le code JavaScript

Étape 1 : Créer `index.html`

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>JSON intégré en JS</title>
</head>
<body>

<h2>Données JSON intégrées dans le script</h2>
<div id="resultat"></div>

<script>
// ---- JSON intégré directement ----
const personnes = [
  { "nom": "Nom0", "prenom": "Prénom0", "tel": "0601020304" },
  { "nom": "Nom1", "prenom": "Prénom1", "tel": "0605060708" }
];

// ---- Affichage ----
const div = document.getElementById("resultat");

personnes.forEach(p => {
  div.innerHTML += `<p>${p.prenom} ${p.nom} – ${p.tel}</p>`;
});

// Conseil : regarder dans la console
console.log("Données JSON :", personnes);
</script>

</body>
</html>
```

Utilisation de l'inspecteur d'éléments

1. Ouvrir les outils de développement

- F12 (ou clic droit : inspecter ...)

- Onglet **Console**
- Voir le tableau JSON (`console.log(personnes)`)

2. Mettre un breakpoint

- Dans l'onglet **Sources**
- Mettre un point d'arrêt sur `forEach`
- Observer les variables

3. Afficher avec un template HTML

Modifier le code d'affichage des données dans le fichier HTML et actualiser la page :

```
div.innerHTML += `  
  <p>  
    <strong>${p.prenom}</strong> ${p.nom}<br>  
    Tel : ${p.tel}  
  </p>  
`;  
`;
```

TP 1 — Afficher les données dans un tableau HTML

À partir du JSON, afficher les données dans un tableau `<table>` :

Objectif :

- créer dynamiquement un tableau
- créer une ligne `<tr>` par personne
- afficher : **Prénom, Nom, Téléphone**

Exemple attendu :

Prénom	Nom	Téléphone
Prénom0	Nom0	0601020304
Prénom1	Nom1	0605060708

Corrigé du TP

```
<!DOCTYPE html>  
<html lang="fr">  
<head>  
  <meta charset="UTF-8">  
  <title>TP JSON - Tableau</title>  
</head>
```

```
<body>

<h2>Affichage des données dans un tableau</h2>

<table border="1" id="tableau">
  <tr>
    <th>Prénom</th>
    <th>Nom</th>
    <th>Téléphone</th>
  </tr>
</table>

<script>
// Données JSON intégrées
const personnes = [
  { "nom": "Nom0", "prenom": "Prénom0", "tel": "0601020304" },
  { "nom": "Nom1", "prenom": "Prénom1", "tel": "0605060708" }
];

// Récupération du tableau HTML
const table = document.getElementById("tableau");

// Pour chaque personne, créer une ligne <tr>
personnes.forEach(p => {

  // Création d'une ligne
  const tr = document.createElement("tr");

  // Création et remplissage des cellules
  const tdPrenom = document.createElement("td");
  tdPrenom.textContent = p.prenom;

  const tdNom = document.createElement("td");
  tdNom.textContent = p.nom;

  const tdTel = document.createElement("td");
  tdTel.textContent = p.tel;

  // Ajout des cellules dans la ligne
  tr.appendChild(tdPrenom);
  tr.appendChild(tdNom);
  tr.appendChild(tdTel);

  // Ajouter la ligne dans le tableau
  table.appendChild(tr);
});
</script>

</body>
</html>
```

Utiliser des données JSON en direct depuis Internet (API Météo)

Le format JSON — Rappel

JSON est un format textuel utilisé pour échanger de l'information entre applications.

Il permet de transmettre :

- des **valeurs simples** : nombres, texte, vrai/faux
- des **objets** : { ... }
- des **tableaux** : [...]

Exemple d'objet JSON représentant une température :

```
{
  "temperature": 8.5,
  "unite": "°C",
  "heure": "2025-11-24T12:00"
}
```

Exemple de tableau d'objets :

```
[
  {"ville": "Paris", "temp": 8},
  {"ville": "Lyon", "temp": 11}
]
```

L'API météo : qu'est-ce que c'est ?

Une **API** est un service accessible sur internet. Elle fournit des données en JSON chaque fois qu'on appelle son adresse.

Adresse de l'exemple :

```
https://api.open-meteo.com/v1/forecast?
latitude=44.3511408&longitude=2.572849&current_weather=true
```

Cette URL signifie :

- **latitude=44.3511408** → position Nord/Sud
- **longitude=2.572849** → position Est/Ouest
- **current_weather=true** → on veut seulement la météo actuelle

Copier et coller le lien dans la barre de navigation du navigateur Firefox et visualiser les données envoyées par l'API.

Que renvoie l'API ? (structure JSON)

Voici une version **simplifiée** des données reçues :

```
{
  "latitude": 44.36,
  "longitude": 2.58,
  "generationtime_ms": 0.08,
  "utc_offset_seconds": 0,
  "timezone": "GMT",
  "elevation": 623.0,

  "current_weather": {
    "temperature": 7.8,
    "windspeed": 17.1,
    "winddirection": 210,
    "weathercode": 61,
    "time": "2025-11-24T11:00"
  }
}
```

Comment lire ce JSON ?

C'est un **objet** contenant :

Propriété	Type	Signification
latitude	nombre	latitude du lieu
longitude	nombre	longitude du lieu
elevation	nombre	altitude
current_weather	objet	la partie météo intéressante

La partie météo est un **objet dans un objet** :

```
"current_weather": {
  "temperature": 7.8,
  "windspeed": 17.1,
  "weathercode": 61,
  "time": "2025-11-24T11:00"
}
```

Comment accéder à ces données en JavaScript ?

JSON → JavaScript → Objet JS → Accès aux données

Lorsqu'on fait :

```
fetch(url)
  .then(r => r.json())
  .then(data => {
```

La variable `data` contient le **JSON transformé en objet JavaScript**.

On peut donc écrire :

Donnée souhaitée	Code JS
Température	<code>data.current_weather.temperature</code>
Vitesse du vent	<code>data.current_weather.windspeed</code>
Heure des relevés	<code>data.current_weather.time</code>
Altitude du lieu	<code>data.elevation</code>

- Tout se lit **avec des points**, de la même manière qu'un dossier contenant un sous-dossier.

Exemple :

```
let meteo = data.current_weather;
console.log(meteo.temperature);
```

Lien direct JSON → HTML

Pour afficher la température :

```
document.getElementById("meteo").innerHTML =
  `<p>Température : ${data.current_weather.temperature}°C</p>`;
```

Et on peut faire pareil pour les autres champs.

Schéma visuel des données

```
data
├── current_weather
│   ├── temperature
│   ├── windspeed
│   ├── weathercode
│   └── time
```

→ En JavaScript : `data.current_weather.temperature` `data.current_weather.windspeed`

Exercice applicatif

"Comment récupérer l'altitude ?"

Réponse :

```
data.elevation
```

"Comment récupérer la direction du vent ?"

```
data.current_weather.winddirection
```

Résumé

- JSON est un **format texte**
 - `fetch()` récupère ce texte
 - `.json()` → transforme en objet JavaScript
 - On accède aux données JSON transformées → **comme des objets JS classiques**
-

EXERCICES — Comprendre et Lire du JSON

Objectif :

- savoir reconnaître la structure d'un fichier JSON
 - savoir repérer les objets / tableaux
 - savoir accéder aux valeurs dans un objet JSON
 - comprendre ce que signifie `data.xxx.yyy` en JavaScript
 - être capable de lire un JSON provenant d'une API (exemple météo)
-

Exercice 1 — Identifier objets et tableaux

Voici un JSON :

```
{
  "ville": "Paris",
  "population": 2161000,
  "coordonnees": { "lat": 48.85, "lon": 2.35 },
  "monuments": ["Tour Eiffel", "Arc de Triomphe", "Louvre"]
}
```


Questions :

1. Quel est le type de la valeur `ville` ?
2. Que contient `coordonnees` ?
3. Combien y a-t-il d'éléments dans `monuments` ?
4. Comment accéder à la longitude (`lon`) ?
5. Comment accéder au deuxième monument ("Arc de Triomphe") ?

Corrigé

1. Une chaîne de caractères (string)
2. Un **objet** contenant deux valeurs : `lat` et `lon`.
3. 3 éléments.
4. `coordonnees.lon`
5. `monuments[1]`

Exercice 2 — Lire des données météo

Voici un extrait simplifié de l'API Open-Meteo :

```
{
  "latitude": 48.75,
  "longitude": 2.13,
  "elevation": 163.0,
  "current_weather": {
    "temperature": 17.3,
    "windspeed": 3.9,
    "weathercode": 3,
    "time": "2024-05-02T15:00"
  }
}
```

Questions :

1. Où se trouvent les données météo (dans quel champ) ?
2. Quel est le type de `current_weather` ?
3. Quelle est la température ?
4. Comment accéder à l'heure (`time`) ?
5. Comment accéder à l'altitude ?
6. Écrire le chemin complet (avec des points) pour accéder à `windspeed`.

Corrigé

1. Dans `current_weather`.
2. Un **objet**.
3. `17.3`
4. `current_weather.time`
5. `elevation`

6. `current_weather.windspeed`

Exercice 3 — Comprendre un tableau d'objets

Voici un JSON retourné par une API de géolocalisation :

```
{
  "results": [
    { "name": "Paris", "latitude": 48.85, "longitude": 2.35 },
    { "name": "Lyon", "latitude": 45.75, "longitude": 4.85 }
  ]
}
```

Questions :

1. Quelle est la nature de `results` (objet ? tableau ?)
2. Combien contient-il d'éléments ?
3. Comment accéder :
 - au nom du premier résultat ?
 - à la latitude de Lyon ?
 - à la longitude de Paris ?
4. Écrire le chemin complet vers le nom du second résultat.

Corrigé

1. Un **tableau**.
 2. 2 éléments.
 3. Comment accéder :
 - `results[0].name`
 - `results[1].latitude`
 - `results[0].longitude`
 4. `results[1].name`
-

Exercice 4 — Lire un JSON et prédire le JS

Soit le JSON :

```
{
  "film": "Inception",
  "details": {
    "realisateur": "Christopher Nolan",
    "acteurs": ["DiCaprio", "Page", "Gordon-Levitt"]
  }
}
```

```
}  
}
```

On veut compléter le code JavaScript ci-dessous :

```
console.log( _____ ); // doit afficher "Inception"  
console.log( _____ ); // doit afficher "Christopher Nolan"  
console.log( _____ ); // doit afficher "Page"
```

Compléter les parties manquantes.

Corrigé

1. `data.film`
2. `data.details.realisateur`
3. `data.details.acteurs[1]`

Code JS à compléter :

```
console.log( data.film );  
console.log( data.details.realisateur );  
console.log( data.details.acteurs[1] );
```

Arbre des données :

```
data  
├── film  
└── details  
    ├── realisateur  
    └── acteurs  
        ├── [0] : DiCaprio  
        ├── [1] : Page  
        └── [2] : Gordon-Levitt
```

Exercice 5 — Lire la documentation d'une API

Voici un JSON retourné par une API fictive :

```
{  
  "status": "ok",  
  "temperature": { "value": 24.5, "unit": "°C" },  
}
```

```
"forecast": [
  { "hour": "10:00", "temp": 23 },
  { "hour": "11:00", "temp": 24 },
  { "hour": "12:00", "temp": 26 }
]
```

Questions :

1. Quelle est la température actuelle ?
2. Quelle unité est utilisée ?
3. Quelle est la température prévue à 12h ?
4. Comment accéder au champ `temp` du 2e élément ?
5. Comment accéder à `"ok"` ?

Corrigé

1. `24.5` (dans `temperature.value`)
2. `"°C"` (dans `temperature.unit`)
3. `26` → `forecast[2].temp`
4. `forecast[1].temp`
5. `status`

Exercice 6 — Mini JavaScript (lecture seule)

Voici le JSON :

```
{
  "produit": "Clavier",
  "prix": 29.9,
  "caracteristiques": { "sans_fil": true, "couleur": "noir" }
}
```

Compléter les lignes JS :

```
let data = { ...JSON ci-dessus... };

console.log( data.produit );           // ?
console.log( data.caracteristiques.couleur ); // ?
console.log( data.caracteristiques.sans_fil ); // ?
```

Corrigé

```
console.log( data.produit );           // "Clavier"
console.log( data.caracteristiques.couleur ); // "noir"
```

```
console.log( data.caracteristiques.sans_fil ); // true
```

Application météo

Code HTML (`meteo.html`)

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Météo Rodez - Affichage sous forme de paragraphe</title>
</head>
<body>
  <h1>Météo à Rodez</h1>
  <div id="meteo"></div>

  <script>
    // 1. URL de l'API avec latitude / longitude
    const url = "https://api.open-meteo.com/v1/forecast?
latitude=44.3511408&longitude=2.572849&current_weather=true";

    // 2. Appel à l'API
    fetch(url)

      // 3. Conversion en JSON
      .then(r => r.json())

      // 4. Lecture et affichage
      .then(data => {
        const meteo = data.current_weather;

        document.getElementById("meteo").innerHTML = `
          <p><strong>Température :</strong> ${meteo.temperature} °C</p>
          <p><strong>Vent :</strong> ${meteo.windspeed} km/h</p>
          <p><strong>Heure :</strong> ${meteo.time}</p>
          <p><strong>Altitude :</strong> ${data.elevation} m</p>
          <p><strong>Code météo :</strong> ${meteo.weathercode}</p>
        `;
      })

      // 5. Gestion des erreurs
      .catch(err => {
        document.getElementById("meteo").innerHTML =
          "<p style='color:red;'>Erreur : Impossible de charger les données
météo</p>";
        console.error("Erreur API météo :", err);
      });
  </script>
</body>
</html>
```

TP 1 — Afficher la météo d'une autre ville

Objectif : Modifier l'URL pour afficher la météo de Lyon, Paris, Marseille, Toulouse.







Corrigé : Paris

```
const url = "https://api.open-meteo.com/v1/forecast?latitude=48.85&longitude=2.35&current_weather=true";
```

- Le reste du code ne change pas.

TP 2 — Afficher une icône météo

Open-Meteo utilise des *weather codes*.

Code	Signification	Icône
0	Soleil	
1-3	Nuageux	
45-48	Brouillard	
51-67	Bruine / pluie légère	
71-79	Neige	
95-99	Orages	

Corrigé

Ajouter une fonction :

```
function icone(code) {  
  if (code === 0) return "☀️";  
  if (code >= 1 && code <= 3) return "☁️";  
  if (code >= 45 && code <= 48) return "🌫️";  
  if (code >= 51 && code <= 67) return "🌧️";  
  if (code >= 71 && code <= 79) return "❄️";  
  if (code >= 95 && code <= 99) return "⚡️";  
  return "❓";  
}
```

Puis dans l'affichage :

```
<p>Icône météo : ${icone(meteo.weathercode)}</p>
```

Fiche récapitulative

Qu'est-ce que le JSON ?

JSON est un **format texte** permettant d'échanger des données entre applications (sites web, API, programmes...). Il ressemble à des objets JavaScript, mais ce n'est **que du texte**.

- lisible par les humains
- compris nativement par JavaScript
- facile à envoyer sur Internet

Les 2 structures de base

1. Objet JSON → entouré par { }

Représente un élément avec des propriétés.

```
{  
  "prenom": "Alice",  
  "age": 25  
}
```

- "prenom" → clé
- "Alice" → valeur
- les clés sont entre guillemets
- chaque paire clé/valeur est séparée par une virgule

Accès en JavaScript : `data.prenom` → "Alice"

2. Tableau JSON → entouré par []

Contient une liste de valeurs ou d'objets.

```
[  
  {"ville": "Paris"},  
  {"ville": "Lyon"}  
]
```

Accès en JavaScript : `data[0].ville` → "Paris"

Types de données JSON

Type	Exemple
string (texte)	"Bonjour"
number	42, 19.8
boolean	true, false
null	null
object	{ "a": 1, "b": 2 }
array	[1, 2, 3]

Lire un JSON : méthode simple

Exemple :

```
{
  "ville": "Paris",
  "meteo": {
    "temperature": 11.5,
    "vent": 5.3
  }
}
```

Accès aux données :

- `data.ville` → "Paris"
- `data.meteo.temperature` → 11.5
- `data.meteo.vent` → 5.3

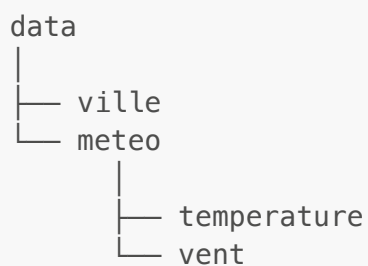
Lecture d'un tableau d'objets

```
{
  "resultats": [
    { "nom": "Alice", "age": 22 },
    { "nom": "Bob", "age": 25 }
  ]
}
```

Accès :

- `data.resultats[0].nom` → "Alice"
- `data.resultats[1].age` → 25

Schéma visuel



- On "descend" dans la structure avec des **points** et des **indices**.

JSON et JavaScript

Dans JS, après un `fetch()` :

```
fetch(url)
  .then(r => r.json())
  .then(data => {
    console.log(data.meteo.temperature);
  });
```

- JSON → converti en **objet JavaScript**
- On peut utiliser `data.xxx.yyyy`

Erreurs courantes à éviter

- Virgule en trop
- Clé sans guillemets
- Utilisation de commentaires (interdits en JSON)
- Mélanger simple `'` et double `"` (en JSON → seulement `"`)

Un JSON valide = guillemets doubles uniquement