I/ Définition

JSON (*JavaScript Object Notation*) est un format léger d'échange de données, facile à lire et écrire pour les humains et à traiter par les machines. Il est largement utilisé pour l'échange de données entre un serveur et un client web.

Intérêts du format JSON :

- Plus léger et plus rapide que XML.
- Facile à lire et comprendre.
- Utilisé par la plupart des API modernes.
- Compatible avec de nombreux langages de programmation, notamment JavaScript.

II/ Structure de JSON

Un document JSON est une collection de paires clé-valeur, où :

- Les clés sont toujours des chaînes de caractères.
- Les valeurs peuvent être de différents types.

1) Types de données en JSON

Type	Exemple
Chaîne	"nom": "Alice"
Nombre	"age": 20
Booléen	"estEtudiant": false
Tableau	"cours": ["Maths", "Informatique"]
Objet	"adresse": { "ville": "Taverny", "codePostal": 95150 }
Valeur null	"téléphone": null

2) Exemple d'un objet JSON

```
{
	"nom": "Alice",
	"age": 20,
	"estEtudiant": false,
	"cours": ["Maths", "Informatique"],
	"adresse": {
		"ville": "Taverny",
		"codePostal": 95150
	}
}
```

III/ Manipulation de JSON en JavaScript

JavaScript intègre nativement le support de JSON avec les méthodes JSON.parse() et JSON.stringify().

1) Conversion d'une chaîne JSON en objet JavaScript

On utilise JSON.parse() pour convertir une chaîne JSON en objet JavaScript :

```
const jsonString = '{"nom": "Alice", "age": 20}';
const personne = JSON.parse(jsonString);

console.log(personne.nom); // Affiche "Alice"
console.log(personne.age); // Affiche 20
```

2) Conversion d'un objet JavaScript en JSON

On utilise JSON.stringify() pour convertir un objet JavaScript en une chaîne JSON :

```
const objet = {
	nom: "Bob",
	age: 30,
	ville: "Herblay"
};

const jsonString = JSON.stringify(objet);
console.log(jsonString); // Affiche '{"nom":"Bob","age":30,"ville":"Herblay"}'
```

IV/ JSON et les API Web

Dans les applications web modernes, JSON est couramment utilisé pour échanger des données avec des serveurs via des requêtes HTTP.

1) Récupérer des données JSON depuis une API avec fetch()

La fonction fetch() permet d'effectuer des requêtes HTTP et de récupérer des réponses au format JSON :

```
fetch('https://jsonplaceholder.typicode.com/users/1')
  .then(response => response.json()) // Convertit la réponse en JSON
  .then(data => console.log(data))   // Affiche l'objet JSON reçu
  .catch(error => console.error('Erreur:', error));
```

2) Envoyer des données JSON avec fetch()

On peut envoyer des données JSON à une API avec fetch() en utilisant la méthode POST :

```
const user = {
	nom: "Charlie",
	email: "charlie@example.com"
};

fetch('https://jsonplaceholder.typicode.com/users', {
	method: 'POST',
	headers: {
		'Content-Type': 'application/json'
	},
	body: JSON.stringify(user)
})
.then(response => response.json())
.then(data => console.log('Utilisateur ajouté:', data))
.catch(error => console.error('Erreur:', error));
```