

L'utilisation de JSON

Table des matières

I. Contexte	3
II. La syntaxe	3
III. Exercice : Appliquez la notion	6
IV. JSON avec JavaScript	6
V. Exercice : Appliquez la notion	9
VI. Auto-évaluation	10
A. Exercice final	10
B. Exercice : Défi	11
Solutions des exercices	12

I. Contexte

Durée : 1 h

Environnement de travail : Repl.it

Pré-requis : Programmation AJAX, Promesses et fonctions fléchées

Contexte

Le développement de pages dynamiques dans les applications web implique une modification du contenu en fonction de données variables. Celles-ci peuvent avoir différentes origines, mais, qu'elles soient obtenues depuis un serveur ou qu'elles proviennent des règles de gestion internes de l'application, il est nécessaire qu'elles respectent un formalisme connu et compréhensible par le langage utilisé.

Celui-ci peut prendre différentes formes, comme le XML par exemple. Nous étudierons ici le format **JSON** (pour *JavaScript Object Notation*) en décrivant tout d'abord sa syntaxe, puis en présentant son utilisation en JavaScript lors d'échanges locaux au navigateur, pour finalement décrire son utilisation au sein d'échanges distants.

II. La syntaxe

Objectifs

- Apprendre la syntaxe du format JSON
- Découvrir des outils de validation du format

Mise en situation

Il existe différents moyens de formaliser des données échangées. Nous allons voir que le format JSON est particulièrement adapté aux contraintes des développement web grâce à une syntaxe simple et suffisamment généraliste pour correspondre à n'importe quel cas d'utilisation. Pour cela, nous présenterons en détails les spécificités de la syntaxe du format JSON, avant de présenter les moyens permettant de vérifier la validité de sa structure.

Historique

« Créée par Douglas Crockford entre 2002 et 2005, la première norme du JSON est ECMA-404, publiée en octobre 2003. Elle est actuellement décrite par les deux normes en concurrence : RFC 8259 de l'IETF et ECMA-404 de l'ECMA.

fr.wikipedia.org/wiki/JavaScript_Object_Notation¹



Syntaxe

Le format JSON permet de représenter les différents types JavaScript. Cette opération s'appelle la sérialisation. Voici certaines règles de sérialisation pour obtenir la représentation en JSON des types les plus utilisés en JavaScript :

- tableau, représenté sous la forme d'une structure de données entre [], chacune séparée par une ,
- objet représenté sous la forme d'une structure de données entre { }, chacune séparée par une ,

¹ https://fr.wikipedia.org/wiki/JavaScript_Object_Notation

- booléen (`true` ou `false`)
- nombre entier ou décimal
- chaîne de caractères entre `" "`
- `null`

Pour la représentation des objets, les données en elles-même suivent une notation **clé: valeur** pour laquelle la clé doit être une chaîne de caractères entre `" "`, et la valeur peut utiliser l'une des représentations suscitées.

Exemple

```
1 {
2   "id": 10,
3   "type": "Etudiant",
4   "classes": ["Informatique", "Algorithmie"],
5   "fullTime": true,
6   "person": {
7     "id": 3,
8     "fullName": "John Doe",
9     "phone": null
10  }
11 }
```

La structure de données présentée contient donc :

- un nombre correspondant à la clé `"id"`,
- un tableau contenant les valeurs `"Informatique"` et `"Algorithmie"` associées à la clé `"classes"`,
- un booléen `true` à la clé `"fullTime"`,
- un objet, répertorié à la clé `"person"`, contenant un `id` avec pour valeur 3, une chaîne de caractères `"John Doe"` associée à l'attribut `"fullName"` et une valeur `null` pour l'attribut `"phone"`.

Remarque

Grâce à sa souplesse, le format JSON permet de représenter des structures de données complexes facilement par l'encapsulation des structures de données. Il est donc par exemple possible de représenter un tableau d'objets au sein d'un autre objet.

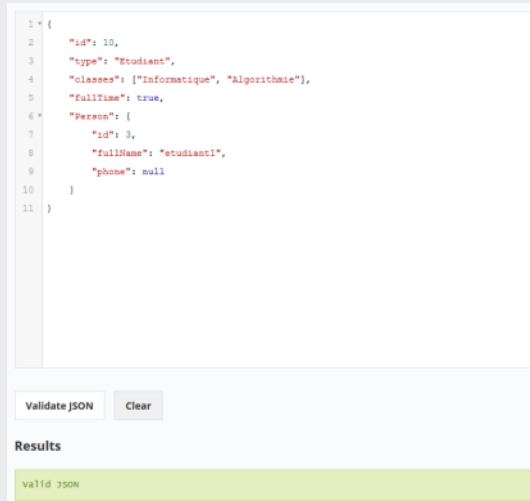
Exemple

```
1 {
2   "person": {
3     "adresses": {
4       "id": 45,
5       "data": [
6         {
7           "id": 89,
8           "city": "Paris"
9         },
10        {
11          "id": 14,
12          "city": "Lyon"
13        }
14      ]
15    }
16  }
17 }
```

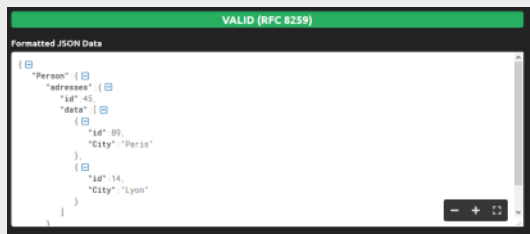
Attention

Le formalisme JSON ne prend pas en charge l'ajout de commentaires dans la description des structures de données.

Afin de s'assurer que les structures JSON transmises soient valides, il existe des outils en ligne permettant de valider le formalisme utilisé, comme <https://jsonlint.com/> ou <https://jsonformatter.curiousconcept.com/>. Ces outils permettent de saisir une structure en JSON et de vérifier qu'elle respecte bien les normes de syntaxe de ce format.

Exemple

Exemple de résultat de validation de la structure JSON vue dans le premier exemple sur *jsonlint.com*.

Exemple

Exemple de résultat de validation de la structure JSON vu dans le second exemple sur *jsonformatter.curiousconcept.com*.

Syntaxe **À retenir**

- Il est possible de représenter des structures de données en suivant le format JSON de manière simple et rapide, grâce à sa syntaxe souple et non typée.

Complément

RFC 8259¹

Présentation de JSON²

1 <https://www.bortzmeyer.org/8259.html>

2 <https://www.json.org/json-fr.html>

III. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



Question 1

[solution n°1 p.13]

Décrivez chaque composante de la structure de données présentée ci-dessous en précisant le type de chaque valeur.

```

1 {
2   "quote": {
3     "id": 18,
4     "price": 245,
5     "customer": {
6       "id": 20,
7       "phone": "8888888888",
8       "products": [
9         {
10          "id": 8,
11          "label": "Produit 1"
12        },
13        {
14          "id": 52,
15          "label": "Produit 10"
16        }
17      ],
18      "address": null
19    }
20  }
21 }
```

Question 2

[solution n°2 p.13]

Structurez les informations suivantes au format JSON et assurez-vous de la validité du résultat avec l'outil de validation de votre choix.

Un objet contenant :

- un stade (*stadium*) qui contient un identifiant (*id*) 18, un libellé (*label*) "Stade de France" et deux terrains (*fields*)
- un premier terrain qui contient un identifiant (*id*) 2, un libellé (*label*) "Terrain de foot" et une surface (*area*) 200
- un second terrain qui contient un identifiant (*id*) 5, un libellé (*label*) "Terrain de rugby" et une surface (*area*) null

IV. JSON avec JavaScript

Objectif

- Apprendre à sérialiser et désérialiser des données au format JSON grâce à JavaScript

1 <https://repl.it/>

Mise en situation

Comme la plupart des typages de données existants, le format JSON est indépendant du langage utilisé. Son origine liée à la structure intrinsèque des objets de JavaScript assure une grande facilité d'utilisation avec ce langage. Nous allons voir comment JavaScript transpose des données au format JSON, puis nous verrons comment il nous permet d'utiliser des données structurées selon ce format.

L'objet JSON de JavaScript

JavaScript embarque nativement un objet permettant la gestion du format JSON. Cet objet propose deux méthodes :

- `JSON.stringify()` pour sérialiser des données en JSON
- `JSON.parse()` pour désérialiser des données JSON en un type JavaScript

Méthode `JSON.stringify()`

La méthode `stringify()` de l'objet JSON de JavaScript prend en paramètre un type JavaScript, et le sérialise en une chaîne de caractères au format JSON.

Exemple

```
1 const fields = [  
2   {  
3     id: 2,  
4     label: "Terrain de foot",  
5     area: 200  
6   },  
7   {  
8     id: 5,  
9     label: "Terrain de rugby",  
10    area: null  
11  }  
12 ]  
13  
14 const data = {  
15   stadium: {  
16     id: 18,  
17     label: "Stade de France",  
18     fields: fields  
19   }  
20 }  
21  
22 const json = JSON.stringify(data);  
23  
24 console.log(json);
```

L'exécution du code précédent affiche la chaîne de caractères suivante dans la console du navigateur :
`{"stadium":{"id":18,"label":"Stade de France","fields":[{"id":2,"label":"Terrain de foot","area":200},{"id":5,"label":"Terrain de rugby","area":null}]}}`

Attention

Le format JSON n'acceptant pas de format date à proprement parler, les données au format date de JavaScript sont traduites en chaîne de caractères.

Exemple

```
1 const object = {
2   today: new Date(),
3 };
4
5 const jsonObject = JSON.stringify(object);
6
7 console.log(object.today);
8 console.log(jsonObject);
```

Le résultat de l'exécution du code précédent montre que l'objet `Date` de JavaScript a été traduit en chaîne de caractères lors de sa transcription en JSON.

```
► Date Mon Mar 16 2020 16:50:55 GMT+0100 (heure normale d'Europe centrale)
{"today":"2020-03-16T15:50:55.154Z"}
```

À travers l'exemple précédent, on constate que la fonction `JSON.stringify()` de JavaScript permet d'obtenir une représentation structurée au format JSON des données.

Méthode `JSON.parse()`

L'objet `JSON` de JavaScript expose une fonction permettant de désérialiser une chaîne de caractères respectant un format JSON valide en un type JavaScript classique. Cela signifie que les données peuvent être récupérées en respectant la notation pointée de JavaScript. Cette fonction est `parse()`.

Exemple

```
1 const json = '{"stadium":{"id":18,"label":"Stade de France","fields":[{"id":2,"label":"Terrain de foot","area":200},{"id":5,"label":"Terrain de rugby","area":null}]}';
2
3 const data = JSON.parse(json);
4
5 console.log(data.stadium.id);
```

L'exécution du code précédent affiche la valeur de l'attribut `id` de l'objet contenu dans l'attribut `stadium` de l'objet obtenu, c'est-à-dire 18.

Remarque

La bonne exécution de l'instruction `JSON.parse()` est dépendante de la validité du format JSON de la chaîne de caractères passée en paramètre. En effet, en cas d'invalidité de la structure de données, une exception du type `SyntaxError` est levée. C'est pourquoi l'utilisation de `JSON.parse()` se fait au sein d'un bloc `try/catch` afin de permettre une gestion de l'erreur.

Exemple

```
1 try {
2   const json = '{"stadium":{"id":18,"label":"Stade de France","fields":[{"id":2,"label":"Terrain de foot","area":200},{"id":5,"label":"Terrain de rugby","area":null}]}';
3
4   const data = JSON.parse(json);
5
6   console.log(data.stadium.id);
7 } catch(error) {
8   console.log(error);
```



```
9 }
```

Le code précédent génère l'erreur suivante :

```
1 SyntaxError: Unexpected token i in JSON at position 12
2   at JSON.parse (<anonymous>)
3   at <anonymous>:4:21
```

Syntaxe À retenir

- Le langage JavaScript embarque nativement un objet `JSON` qui expose les fonctions `JSON.stringify()`, pour sérialiser un type JavaScript en une chaîne de caractères respectant le format JSON
- Et `JSON.parse()` pour désérialiser une chaîne de caractères au format JSON valide en un type JavaScript

V. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



Question

[solution n°3 p.13]

Voici un objet JavaScript représentant une liste d'étudiants. Convertissez-le au format JSON valide et vérifiez le JSON produit à l'aide d'un des sites de validation vus précédemment.

```
1 const students = [
2   {
3     name: 'Paul',
4     age: 15,
5     scores: [
6       {matiere: 'Maths', note: 10},
7       {matiere: 'Français', note: 12},
8       {matiere: 'Anglais', note: 14},
9     ]
10  },
11  {
12    name: 'Marie',
13    age: 14,
14    scores: [
15      {matiere: 'Maths', note: 15},
16      {matiere: 'Français', note: 9},
17      {matiere: 'Anglais', note: 10},
18    ]
19  },
20 ]
```

Dans une deuxième étape, désérialisez la chaîne de caractères obtenue et tentez d'afficher le nom du premier étudiant.

1 <https://repl.it/>

VI. Auto-évaluation

A. Exercice final

Exercice 1

[solution n°4 p.15]

Exercice

Que signifie JSON ?

Exercice

Comment peut-on valider une structure au format JSON ?

- ☐ Avec un outil en ligne
- ☐ Avec la méthode `JSON.stringify()`
- ☐ Avec la méthode `JSON.parse()`

Exercice

Il est possible d'écrire des commentaires de code dans un objet au format JSON.

- ☐ Vrai
- ☐ Faux

Exercice

Que retournera la console ?

```
1 const student = {
2   name: 'Paul',
3   age: 15,
4   scores: [
5     {matiere: 'Maths', note: 10},
6     {matiere: 'Français', note: 12},
7     {matiere: 'Anglais', note: 14}
8   ]
9 }
10
11 console.log(JSON.stringify(student))
```

- ☐ Une erreur, car il n'est pas possible de sérialiser un objet JavaScript en JSON
- ☐ `["name":"Paul","age":15,"scores": [{"matiere":"Maths","note":10}, {"matiere":"Français","note":12}, {"matiere":"Anglais","note":14}]]`
- ☐ `{ "name": "Paul", "age": 15, "scores": [{ "matiere": "Maths", "note": 10 }, { "matiere": "Français", "note": 12 }, { "matiere": "Anglais", "note": 14 }] }`

Exercice

Comment récupérer la valeur "Groupement de pays" de la clé `name` ?

```
1 const data = JSON.parse(' [{"id":"country","name":"Pays","parents":["country-group"]},
  {"id":"country-group","name":"Groupement de pays","parents":[]},{ "id":"country-
  subset","name":"Sous-ensemble de pays","parents":["country"]},
  {"id":"fr:arrondissement","name":"Arrondissement français","parents":["fr:departement"]},
  {"id":"fr:canton","name":"Canton français","parents":["fr:departement"]},
  {"id":"fr:collectivite","name":"Collectivités d\'outre-mer françaises","parents":
  ["fr:region"]}, {"id":"fr:commune","name":"Commune française","parents":
  ["fr:arrondissement","fr:epci"]}, {"id":"fr:departement","name":"Département
  français","parents":["fr:region"]}, {"id":"fr:epci","name":"Intercommunalité française
  (EPCI)","parents":["country"]}, {"id":"fr:iris","name":"Iris (quartiers INSEE)","parents":
  ["fr:commune"]}, {"id":"fr:region","name":"Région française","parents":["country"]} ]')
```

- ☐ console.log(data[0].name)
- ☐ console.log(data[1].name)
- ☐ console.log(data[0]['name'])
- ☐ console.log(data[1]['name'])

Exercice

Peut-on récupérer une erreur de désérialisation avec la méthode `JSON.parse()` ?

- ☐ Non, on ne peut pas
- ☐ Oui, on peut avec la méthode `JSON.catch()`
- ☐ Oui, on peut avec une structure `try/catch`

Exercice

Un fichier JSON est limité en taille.

- ☐ Vrai
- ☐ Faux

Exercice

Cet objet JSON est dans un format valide.

```
1 { "type": "BMW", "doors": 3, "price": 10000, "colors": ["grey", "blue", "red", "black"], "options": null }
```

- ☐ Vrai
- ☐ Faux

Exercice

Ce code fonctionnera.

```
1 const json = { "hero": { "life": "100", "strong": "23", "defense": "10", "race": "knife" } }
2 console.log(JSON.parse(json))
```

- ☐ Vrai
- ☐ Faux

B. Exercice : Défi

Très souvent les applications web utilisent des API pour récupérer des données au format JSON et les utilisent pour construire leurs sites. Exemple lorsque vous avez un select dans un formulaire qui vous propose le choix de tous les pays d'origine possibles

Dans notre cas, le JSON vous sera fourni dans une variable. Ce JSON est composé de noms de rue.

Question

[solution n°5 p.17]

L'objectif de ce défi est d'utiliser la structure HTML suivante et de permettre de remplir le select dynamiquement avec les données présentes dans le JSON au clic sur le bouton.

```

1 <code><body>
2
3   <input id="button" type="button" value="Click to Populate SELECT with JSON" />
4
5
6   <!--The SELECT element.-->
7   <select id="select">
8     <option value="">-- Select --</option>
9   </select>
10 </body>
11
12 <script>
13 // JSON.
14
15 const json = '[{"id":33,"name":"Boni Avenue"}, {"id":34,"name":"Shaw Boulevard"},
16 {"id":35,"name":"Rue Madeleine"}, {"id":36,"name":"Shaw Boulevard"}]';
17 </script>
18
19 </code>
20

```

Pensez à utiliser l'id de chaque rue pour l'attribut value de vos options.

Indice :

Pensez à parse le JSON



Solutions des exercices

p.6 Solution n°1

La structure de données fournies représente les données suivantes :

- une clé "quote" contenant un objet :
 - deux clés "id" et un "price" ayant respectivement les valeurs numériques 18 et 245
 - une clé "customer" contenant un autre objet :
 - une clé "id" ayant la valeur numérique 20
 - une clé "phone" contenant une chaîne de caractères "8888888888"
 - une clé "products" contenant un tableau de deux objets :
 - une clé "id" ayant la valeur numérique 8 et une clé "label" contenant la chaîne "Produit 1"
 - une clé "id" ayant la valeur numérique 52 et une clé "label" contenant la chaîne "Produit 10"
 - une clé "address" contenant la valeur null

p.6 Solution n°2

```

1 {
2   "stadium": {
3     "id": 18,
4     "label": "Stade de France",
5     "fields": [
6       {
7         "id": 2,
8         "label": "Terrain de foot",
9         "area": 200
10      },
11      {
12        "id": 5,
13        "label": "Terrain de rugby",
14        "area": null
15      }
16    ]
17  }
18 }
```

p.9 Solution n°3

Pour convertir un objet JavaScript au format JSON, il faut utiliser la méthode `stringify()` de l'objet `JSON`.

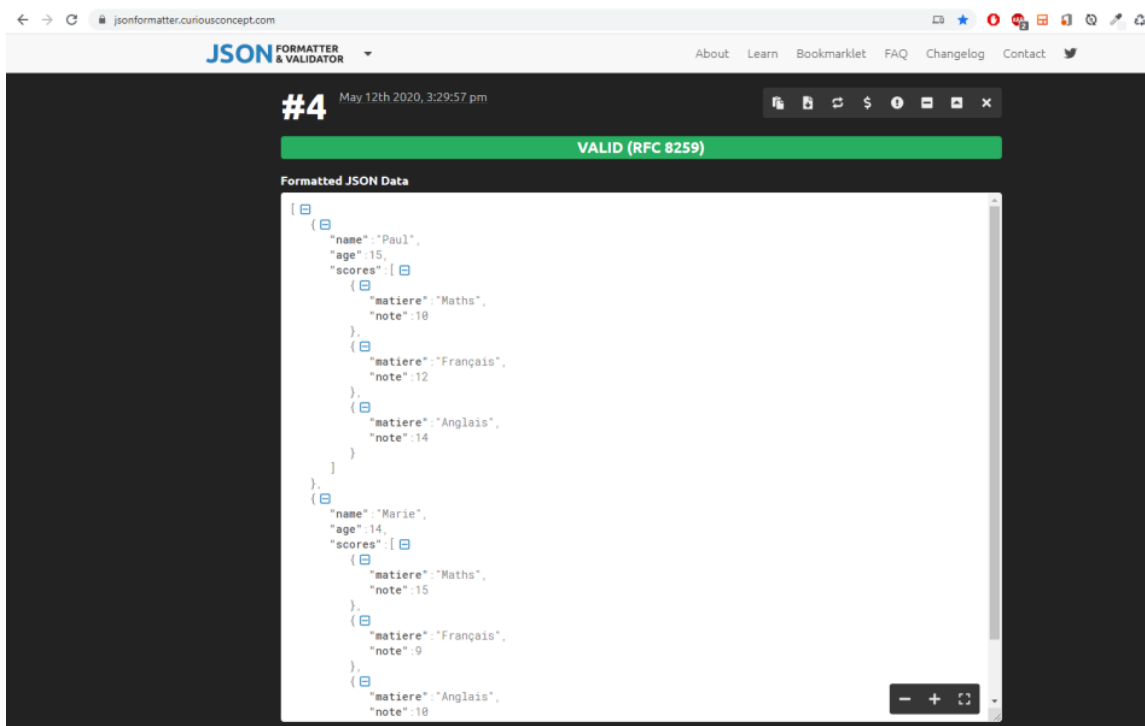
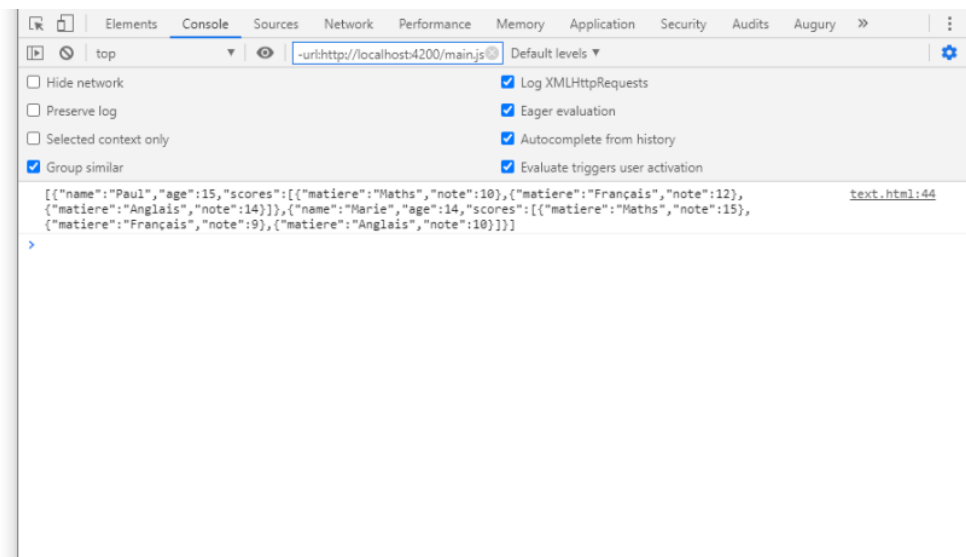
```

1 const students = [
2   {
3     name: 'Paul',
4     age: 15,
5     scores: [
6       {matiere: 'Maths', note: 10},
7       {matiere: 'Français', note: 12},
8       {matiere: 'Anglais', note: 14},
9     ]
10  },
11  {
```

```

12     name: 'Marie',
13     age: 14,
14     scores: [
15       {matiere: 'Maths', note: 15},
16       {matiere: 'Français', note: 9},
17       {matiere: 'Anglais', note: 10},
18     ]
19   },
20 ]
21
22 const json = JSON.stringify(students)
23 console.log(json)
24
25 const newStudents = JSON.parse(json)
26 console.log(newStudents[0].name)

```



Exercice p. 10 Solution n°4

Exercice

Que signifie JSON ?

JavaScript Object Notation

Exercice

Comment peut-on valider une structure au format JSON ?


- ☒ Avec un outil en ligne
- ☐ Avec la méthode `JSON.stringify()`
Cette méthode convertit un type JavaScript en JSON. Elle ne convient pas ici.
- ☒ Avec la méthode `JSON.parse()`
Cette méthode convertit du JSON en un type JavaScript. Elle convient ici pour vérifier que le JSON est valide.

Exercice

Il est possible d'écrire des commentaires de code dans un objet au format JSON.

☐ Vrai

☒ Faux

 Il n'est pas possible d'écrire du commentaire dans un objet au format JSON.

Exercice

Que retournera la console ?

```
1 const student = {  
2   name: 'Paul',  
3   age: 15,  
4   scores: [  
5     {matiere: 'Maths', note: 10},  
6     {matiere: 'Français', note: 12},  
7     {matiere: 'Anglais', note: 14}  
8   ]  
9 }  
10  
11 console.log(JSON.stringify(student))
```


- ☐ Une erreur, car il n'est pas possible de sérialiser un objet JavaScript en JSON
- ☐ `["name":"Paul","age":15,"scores": [{"matiere":"Maths","note":10}, {"matiere":"Français","note":12}, {"matiere":"Anglais","note":14}]]`
- ☒ `{"name":"Paul","age":15,"scores":[{"matiere":"Maths","note":10}, {"matiere":"Français","note":12}, {"matiere":"Anglais","note":14}]}`

Exercice

Comment récupérer la valeur "Groupement de pays" de la clé `name` ?

```
1 const data = JSON.parse('["id":"country","name":"Pays","parents":["country-group"]},
  {"id":"country-group","name":"Groupement de pays","parents":[]},{id:"country-
  subset","name":"Sous-ensemble de pays","parents":["country"]},
  {"id":"fr:arrondissement","name":"Arrondissement français","parents":["fr:departement"]},
  {"id":"fr:canton","name":"Canton français","parents":["fr:departement"]},
  {"id":"fr:collectivite","name":"Collectivités d'outre-mer françaises","parents":
  ["fr:region"]}, {"id":"fr:commune","name":"Commune française","parents":
  ["fr:arrondissement","fr:epci"]}, {"id":"fr:departement","name":"Département
  français","parents":["fr:region"]}, {"id":"fr:epci","name":"Intercommunalité française
  (EPCI)","parents":["country"]}, {"id":"fr:iris","name":"Iris (quartiers INSEE)","parents":
  ["fr:commune"]}, {"id":"fr:region","name":"Région française","parents":["country"]}']')
```

- ☐ console.log(data[0].name)
- ☒ console.log(data[1].name)
- ☐ console.log(data[0]['name'])
- ☒ console.log(data[1]['name'])

 Deux syntaxes sont possibles, mais on préférera souvent la notation `data[1].name` lorsque l'on manipule un objet.

Exercice


Peut-on récupérer une erreur de désérialisation avec la méthode `JSON.parse()` ?

- ☐ Non, on ne peut pas
- ☐ Oui, on peut avec la méthode `JSON.catch()`
- ☒ Oui, on peut avec une structure `try/catch`

Exercice

Un fichier JSON est limité en taille.

- ☐ Vrai
- ☒ Faux


 Un fichier JSON n'est pas limité en taille. Toutefois, il peut arriver que, lors de vos développements, vous soyez confronté à une erreur serveur indiquant que la taille des données est trop importante. Il faudra alors vérifier que la configuration du serveur ne limite pas la taille des données transférées.

Exercice

Cet objet JSON est dans un format valide.

```
1 {"type":"BMW","doors":3,"price":10000,"colors":["grey","blue","red","black"],"options":null}
```

- ☐ Vrai
- ☒ Faux

 Cet objet n'est pas dans un format JSON valide. Pour tester du JSON, utilisez un convertisseur en ligne ou vérifiez dans votre code que `console.log(JSON.parse('objet'))` n'affiche pas d'erreur en console.

L'erreur se situe au niveau de la propriété `colors`, il manque les `"`.

Exercice

Ce code fonctionnera.

```
1 const json = {"hero":{"life":"100","strong":"23","defense":"10","race":"knife"}}
2 console.log(JSON.parse(json))
```


☐ Vrai

☒ Faux

Q Pour qu'un objet représenté au format JSON puisse être désérialisé en objet JavaScript, il faut qu'il soit stocké sous forme de chaînes de caractères. Attention au copier/coller lorsque vous utilisez un convertisseur de JSON ! Il ne faut pas oublier d'englober l'objet avec des *simple quotes* :

```
1 const json = '{"hero":{"life": "100","strong": "23","defense": "10","race": "knife"}}'
```

p. 12 Solution n°5

```
1 <code>
2 <html>
3 <head>
4   <title>Bind SELECT Dropdown with JSON using JavaScript</title>
5 </head>
6 <body>
7
8   <input id="button" type="button" value="Click to Populate SELECT with JSON" />
9
10
11   <!--The SELECT element.-->
12   <select id="select">
13     <option value="">-- Select --</option>
14   </select>
15 </body>
16
17 <script>
18 // THE JSON ARRAY.
19
20 const json = '[{"id":33,"name":"Boni Avenue"}, {"id":34,"name":"Shaw Boulevard"},
21 {"id":35,"name":"Rue Madeleine"}, {"id":36,"name":"Shaw Boulevard"}]';
22
23 const select = document.getElementById('button')
24
25 const populateSelect = () => {
26   const data = JSON.parse(json);
27   let ele = document.getElementById('select');
28   for (let i = 0; i < data.length; i++) {
29     // POPULATE SELECT ELEMENT WITH JSON.
30     let option = document.createElement('option')
31     option.value = data[i]['id']
32     option.innerText = data[i]['name']
33     ele.append(option)
34   }
35 }
36
37 select.addEventListener('click', populateSelect)
38
39 </script>
40 </html>
41
42 </code>
```