> Cette page a été traduite à partir de l'anglais par la communauté. Vous pouvez également contribuer en rejoignant la communauté francophone sur MDN Web Docs.

Manipuler des données JSON

Le JavaScript Object Notation (JSON) est un format standard utilisé pour représenter des données structurées de façon semblable aux objets Javascript. Il est habituellement utilisé pour structurer et transmettre des données sur des sites web (par exemple, envoyer des données depuis un serveur vers un client afin de les afficher sur une page web ou vice versa). Comme cette notation est extrêmement courante, cet article a pour but de vous donner les connaissances nécessaires pour travailler avec JSON en JavaScript, vous apprendre à analyser la syntaxe du JSON afin d'en extraire des données et écrire vos propres objets JSON.

Prérequis :	Vocabulaire de base d'informatique, connaissances de base en HTML et CSS, connaissances de base en JavaScript (voir Premiers pas et Les blocs) et en Javascript orienté objets (voir Introduction aux objets (en-US)).
Objectif :	Comprendre comment utiliser les données stockées dans un objet JSON, et créer vos propres objets JSON.

Plus sérieusement, qu'est ce que le JSON ?

JSON est un format de données semblable à la syntaxe des objets JavaScript, qui a été popularisé par Douglas Crockford . Malgré sa syntaxe très similaire à celle des objets littéraux JavaScript, JSON peut être utilisé indépendamment de ce langage et ainsi, de

nombreux autres langages de programmation disposent de fonctionnalités permettant d'analyser la syntaxe du JSON et d'en générer.

Le JSON se présente sous la forme d'une chaîne de caractères —utile lorsque vous souhaitez transmettre les données sur un réseau. Il a donc besoin d'être converti en un objet JavaScript natif lorsque vous souhaitez accéder aux données. Ce n'est pas vraiment un souci puisque le JavaScript fournit un objet global JSON disposant des méthodes pour assurer la conversion entre les deux.

> **Note :** Convertir une chaîne de caractères en un objet natif se nomme **analyse syntaxique (parsage)** tandis que le contraire porte le nom de la **linéarisation (stringification)**.

Un objet JSON peut être stocké dans son propre fichier qui se présente simplement sous la forme d'un fichier texte avec l'extension `.json` et le MIME type `application/json`.

Structure du JSON

Nous disions tout à l'heure qu'un objet JSON n'était ni plus ni moins qu'un objet Javascript tout à fait normal et c'est généralement le cas. Un objet JSON accepte comme valeur les mêmes types de données de base que tout autre objet Javascript — chaînes de caractères, nombres, tableaux, booléens et tout autre objet littéral. Cela vous permet de hiérarchiser vos données comme ceci :

JSON

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": ["Radiation resistance", "Turning tiny", "Radiation blast"]
    },
```

```
    {
      "name": "Madame Uppercut",
      "age": 39,
      "secretIdentity": "Jane Wilson",
      "powers": [
        "Million tonne punch",
        "Damage resistance",
        "Superhuman reflexes"
      ]
    },
    {
      "name": "Eternal Flame",
      "age": 1000000,
      "secretIdentity": "Unknown",
      "powers": [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
        "Interdimensional travel"
      ]
    }
  ]
}
```

Si nous chargeons cet objet dans un fichier Javascript au sein d'une variable appelée `superHeroes` par exemple, nous pouvons accéder à ses données de la même façon que nous l'avons fait dans l'article Les bases de JavaScript orienté objets à l'aide de la notation point / crochets. Par exemple :

JS

```
superHeroes.hometown;
superHeroes["active"];
```

Pour accéder aux données plus profondes de la hiérarchie, vous n'avez qu'à enchaîner à la fois les noms des propriétés et les indexes des tableaux. Par exemple, l'expression suivante pointe vers le troisième superpouvoir du second super héros présent dans la liste :

JS

```
superHeroes["members"][1]["powers"][2];
```

1. D'abord, nous partons de la variable — `superHeroes`
2. À l'intérieur de laquelle nous désirons accéder à la propriété `members`, donc, nous tapons `["members"]`.
3. `members` contient un tableau renfermant des objets. Nous désirons accéder au second de ces objets, donc nous utilisons `[1]`.
4. À l'intérieur de cet objet, nous souhaitons accéder à la propriété `powers`, donc, nous utilisons `["powers"]`.
5. Enfin, à l'intérieur de cette propriété `powers` nous trouvons un nouveau tableau qui contient les super pouvoirs de ce héros. Nous désirons obtenir le troisième, donc nous tapons `[2]`.

> **Note :** L'objet JSON vu ci-dessus est disponible au sein d'une variable dans notre exemple JSONTest.html (voir le code source). Essayez de le charger et d'accéder aux données en utilisant la console Javascript de votre navigateur.

Des tableaux en tant que JSON

Un peu plus haut, nous avons dit qu'un objet JSON n'était ni plus ni moins qu'un objet Javascript tout à fait normal et c'est généralement le cas. La raison pour laquelle nous avons dit "généralement le cas" est qu'un tableau peut également être un objet JSON valide, par exemple :

JSON

```
[
  {
    "name": "Molecule Man",
    "age": 29,
    "secretIdentity": "Dan Jukes",
    "powers": ["Radiation resistance", "Turning tiny", "Radiation blast"]
  },
  {
    "name": "Madame Uppercut",
    "age": 39,
    "secretIdentity": "Jane Wilson",
```

```
    "powers": [
      "Million tonne punch",
      "Damage resistance",
      "Superhuman reflexes"
    ]
  }
]
```

Le code ci dessus est une notation JSON parfaitement valide. Vous n'aurez qu'à accéder aux éléments de votre tableau en commençant avec un index, par exemple : `[0]["powers"][0]`.

Notes diverses

- Un objet JSON est uniquement un format de données — il ne contient que des propriétés mais pas de méthodes.
- La notation JSON nécessite l'usage des guillemets pour être valide. Il est obligatoire d'utiliser des guillemets et non les apostrophes autour des chaînes de caractères et des noms de propriétés.
- Une simple virgule ou un double point mal placé peut rendre votre fichier JSON invalide et non fonctionnel. Soyez très attentif aux données que vous utilisez (bien que le JSON généré automatiquement par un programme sera moins enclin à contenir des erreurs, à partir du moment où le programme est codé correctement). Vous pouvez utiliser une application comme JSONLint pour valider votre code JSON.
- Dans l'absolu, le JSON peut prendre la forme de n'importe quel type de données qui serait valide pour être contenu dans du JSON et non juste des tableaux ou des objets. Ainsi, par exemple, une simple chaîne de caractères ou un nombre serait un objet JSON valide.
- Contrairement au JavaScript dans lequel les propriétés (*keys*) non entourées de guillemets peuvent être utilisées, en JSON, seules les chaînes de caractères entourées de guillemets peuvent être utilisées en tant que propriétés.

Activité : Manipuler le JSON au travers d'un exemple

Allez ! Un petit exemple pour voir comment nous pouvons nous servir de données JSON sur un site web.

Lançons nous

Pour commencer, faites une copie locale de nos fichiers heroes.html et style.css . Le dernier contient simplement quelques instructions CSS pour la mise en forme de notre page alors que le premier n'est ni plus ni moins qu'un squelette HTML de base :

HTML

```
<header></header>

<section></section>
```

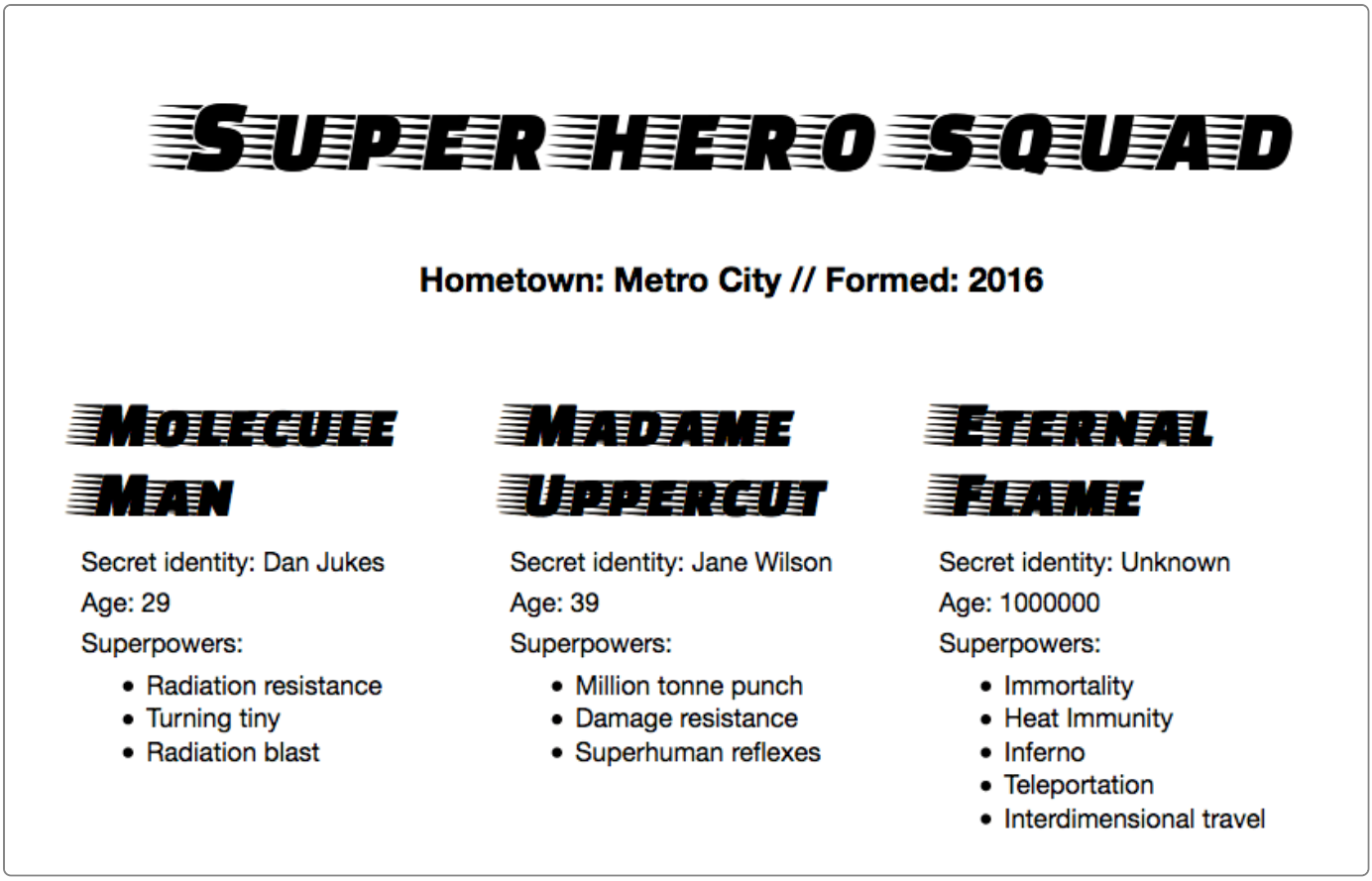
Nous trouvons également un élément `<script>` dans lequel nous écrirons le code Javascript de cet exercice. Pour le moment, il ne contient que deux lignes destinées à récuperer les éléments `<header>` et `<section>` pour les stocker dans des variables :

JS

```
var header = document.querySelector("header");
var section = document.querySelector("section");
```

Nos données JSON sont disponibles sur notre GitHub ici : https://mdn.github.io/learning-area/javascript/oojs/json/superheroes.json .

Nous souhaitons les récupérer et, après quelques manipulations du DOM, les afficher comme ceci :



Chargeons notre JSON

Pour charger nos données JSON, nous allons utiliser l'API `XMLHttpRequest` (qu'on appelle plus couramment **XHR**). Il s'agit d'un objet JavaScript extrêmement utile qui nous permet de construire une requête afin d'interroger un serveur pour obtenir des ressources diverses (images, texte, JSON, ou n'importe quel extrait HTML) le tout en Javascript. En d'autres termes, cela nous permet de mettre à jour de petites sections de contenu sans avoir à recharger notre page toute entière. Ceci conduit à des pages web plus réactives. Mais même si le sujet est très tentant, il dépasse largement l'objet de cet article pour être expliqué plus en détails.

1. Donc, pour commencer, nous allons charger l'URL du fichier JSON que nous voulons récupérer dans une variable. Aussi, ajouter la ligne suivante à votre code Javascript :

   JS

   ```
   var requestURL =
     "https://mdn.github.io/learning-area/javascript/oojs/json/superheroes.json";
   ```

2. Afin de créer une requête, nous avons besoin d'instancier un nouvel objet `XMLHttpRequest` à partir de son constructeur en utilisant le mot clé new. Ajouter la ligne suivante à votre script :

JS

```
var request = new XMLHttpRequest();
```

3. Maintenant, nous avons besoin d'ouvrir une nouvelle requête grâce à la méthode `open()` . Ajoutez la ligne suivante :

   JS

   ```
   request.open("GET", requestURL);
   ```

   Cette méthode prend au moins deux paramètres — il y a d'autres paramètres optionnels disponibles. Deux suffiront pour notre exemple :

   - La méthode HTTP à utiliser sur le réseau pour notre requête. Dans notre cas, la méthode `GET` est appropriée dans la mesure où nous voulons simplement récupérer quelques données.
   - L'URL où adresser notre requête — il s'agit de l'URL du fichier JSON dont nous parlions tout à l'heure.

4. Ensuite, ajoutez les deux lignes suivantes — ici, nous attribuons la valeur `'json'` à `responseType` (en-US), signalant ainsi au serveur que nous attendons une réponse au format JSON. Puis, nous envoyons notre requête à l'aide de la méthode `send()` :

   JS

   ```
   request.responseType = "json";
   request.send();
   ```

5. La dernière partie de cette section concerne la réponse du serveur et son traitement. Ajoutez les lignes suivantes à la fin de votre code :

   JS

   ```
   request.onload = function () {
     var superHeroes = request.response;
     populateHeader(superHeroes);
     showHeroes(superHeroes);
   };
   ```

Ici, nous stockons la réponse à notre requête (disponible au travers de la propriété `response`) dans la variable `superHeroes` ; cette variable contiendra désormais l'objet JavaScript basé sur le JSON ! Nous passerons ensuite cet objet en paramètre à deux fonctions — la première remplira le `<header>` avec les données correspondantes tandis

que la seconde créera une carte d'identité pour chaque héros de l'équipe et l'ajoutera dans la `<section>`.

Nous avons encapsulé ce code dans un gestionnaire d'évènements qui s'exécutera quand l'évènement load sera déclenché sur l'objet request (voir `onload` (en-US)) — simplement parce que l'évènement load est déclenché quand une réponse a été renvoyée avec succès ; en procédant de la sorte, nous serons certains que la propriété `request.response` sera disponible au moment où nous essayerons d'en faire quelque chose.

Remplissage de l'en-tête

Maintenant que nous avons récupéré et converti en objet JavaScript nos données JSON, il est temps d'en faire bon usage : implémentons donc les deux fonctions évoquées ci-dessus. Avant tout, ajoutons les lignes suivantes en dessous de notre code :

JS

```
function populateHeader(jsonObj) {
  var myH1 = document.createElement("h1");
  myH1.textContent = jsonObj["squadName"];
  header.appendChild(myH1);

  var myPara = document.createElement("p");
  myPara.textContent = "Hometown: " + jsonObj["homeTown"] + jsonObj["formed"];
  header.appendChild(myPara);
}
```

Nous avons appelé le paramètre de cette fonction `jsonObj` afin de garder en tête que cet objet JavaScript provient du JSON. Ici, nous créons tout d'abord un élément `<h1>` à l'aide de `createElement()`, nous fixons son `textContent` à la valeur de la propriété `squadName` de l'objet, puis nous l'ajoutons à l'en-tête en utilisant `appendChild()`. Ensuite, nous faisons quelque chose de relativement similaire avec un élément paragraphe : nous le créons, fixons son contenu et l'ajoutons à l'en-tête. La seule différence est que pour son contenu, nous avons concaténé la chaîne de caractère `homeTown` et la propriété `formed` de l'objet.

Création des fiches des héros

Maintenant, ajoutons la fonction suivante qui crée et affiche les fiches de nos super-héros en dessous de notre code :

JS

```
function showHeroes(jsonObj) {
  var heroes = jsonObj["members"];

  for (var i = 0; i < heroes.length; i++) {
    var myArticle = document.createElement("article");
    var myH2 = document.createElement("h2");
    var myPara1 = document.createElement("p");
    var myPara2 = document.createElement("p");
    var myPara3 = document.createElement("p");
    var myList = document.createElement("ul");

    myH2.textContent = heroes[i].name;
    myPara1.textContent = "Secret identity: " + heroes[i].secretIdentity;
    myPara2.textContent = "Age: " + heroes[i].age;
    myPara3.textContent = "Superpowers:";

    var superPowers = heroes[i].powers;
    for (var j = 0; j < superPowers.length; j++) {
      var listItem = document.createElement("li");
      listItem.textContent = superPowers[j];
      myList.appendChild(listItem);
    }

    myArticle.appendChild(myH2);
    myArticle.appendChild(myPara1);
    myArticle.appendChild(myPara2);
    myArticle.appendChild(myPara3);
    myArticle.appendChild(myList);

    section.appendChild(myArticle);
  }
}
```

Pour commencer, on stocke la propriété `members` de l'objet JavaScript dans une nouvelle variable. Ce tableau contient plusieurs objets contenant les informations relatives à chaque héros.

Maintenant, on utilise une boucle for pour parcourir chaque object du tableau. Pour chaque cas, il faut :

1. Créer plusieurs nouveaux éléments : un `<article>` , un `<h2>` , trois `<p>` s, et un `<ul>` .

2. Mettre le `name` du héros actuel dans le `<h2>`.
3. Remplir les trois paragraphes avec leur `secretIdentity`, leur `age`, et une ligne nommée "Superpowers:" pour présenter la liste des super-pouvoirs.
4. Stocker la propriété `powers` dans une nouvelle variable nommée `superPowers` contenant un tableau listant les super-pouvoirs du héros actuel.
5. Utiliser une autre boucle `for` pour parcourir les super-pouvoirs du héros actuel — créer pour chacun d'entre eux un élément `<li>`, y placer le super-pouvoir et placer le `listItem` dans l'élément `<ul>` (`myList`) en utilisant `appendChild()`.
6. Pour finir, on ajoute `<h2>`, les `<p>`s et `<ul>` à `<article>` (`myArticle`), et on ajoute `<article>` à `<section>`. L'ordre d'ajout est important, c'est l'ordre dans lequel les éléments seront affichés dans le HTML.

> **Note :** Si vous ne parvenez pas à faire fonctionner l'exemple, consultez notre code source heroes-finished.html (ou regardez-le en action .)

> **Note :** Si vous comprenez difficilement la notation avec un point/une accolade utilisée pour accéder au JSON, ouvrez le fichier superheroes.json dans un nouvel onglet ou dans votre éditeur de texte et consultez-le pendant la lecture de notre code Javascript. Vous pouvez également vous reporter à notre article Les bases du JavaScript orienté objet pour obtenir plus de détails sur la notation avec un point et avec une accolade.

Conversion entre objets et textes

Dans l'exemple ci-dessus, accéder au JSON est simple, il suffit de définir la requête XHR pour renvoyer la réponse au format JSON en écrivant :

JS

```
request.responseType = "json";
```

Mais on n'a pas toujours cette chance — il est possible de recevoir la réponse JSON sous

convertir en JSON (une chaîne de caractères). Heureusement, ces deux problèmes sont tellement communs dans le développement web qu'un objet JSON interne a été ajouté aux navigateurs depuis longtemps, contenant les deux méthodes suivantes :

- `parse()` qui accepte un objet JSON sous la forme d'une chaîne de caractères en paramètre et renvoie l'objet JavaScript correspondant.
- `stringify()` qui accepte un objet JavaScript en paramètre et renvoie son équivalent sous la forme d'une chaîne de caractères JSON.

Vous pouvez voir la première méthode en action dans notre exemple heroes-finished-json-parse.html (voir le code source) — C'est la même chose que pour l'exemple que nous avons écrit un peu plus tôt, à ceci près qu'on indique à la requête XHR de renvoyer la réponse en JSON sous forme de texte avant d'utiliser la méthode `parse()` pour la convertir en objet JavaScript. La partie du code correspondante se trouve ci-dessous :

JS

```
request.open("GET", requestURL);
request.responseType = "text"; // now we're getting a string!
request.send();

request.onload = function () {
  var superHeroesText = request.response; // get the string from the response
  var superHeroes = JSON.parse(superHeroesText); // convert it to an object
  populateHeader(superHeroes);
  showHeroes(superHeroes);
};
```

Comme vous pouvez le deviner, `stringify()` fait exactement le contraire. Essayez d'entrer les lignes ci-dessous une par une dans la console Javascript de votre navigateur pour voir la méthode en action :

JS

```
var myJSON = { name: "Chris", age: "38" };
myJSON;
var myString = JSON.stringify(myJSON);
myString;
```

On commence par créer un objet JavaScript puis on vérifie son contenu avant de le convertir en chaîne de caractères JSON avec `stringify()` — en sauvegardant au passage le résultat dans une nouvelle variable avant d'effectuer à nouveau une vérification du contenu.

Résumé

Dans cet article, nous vous donnons un manuel simple pour utiliser le JSON dans vos programmes, incluant les méthodes de création et d'analyse syntaxique (parsage) du JSON et d'accès aux données qu'il contient. Dans le prochain article, nous débuterons l'apprentissage du Javascript orienté objet.

Voir aussi

- La page de référence sur l'objet JSON
- La page de référence sur l'objet XMLHttpRequest
- Utiliser XMLHttpRequest
- Les méthodes de requêtes HTTP
- Le site web officiel avec un lien vers les normes de l'ECMA

This page was last modified on 3 août 2023 by MDN contributors.