JSON est l'acronyme de *JavaScript Object Notation*. Comme cette expression le suggère, il a été initialement créé pour la sérialisation et l'échange d'objets Javascript entre deux applications.

Le scénario le plus courant est sans doute celui des applications Ajax dans lesquelles le serveur (Web) et le client (navigateur) échangent des informations codées en JSON. Cela dit, JSON est un format texte indépendant du langage de programmation utilisé pour le manipuler, et se trouve maintenant utilisé dans des contextes très éloignés des applications Web.

C'est le format de données principal que nous aurons à manipuler. Il est utilisé comme modèle de données natif dans des systèmes NoSQL comme MongoDB ou CouchDB, et comme format d'échange sur le Web par d'innombrables applications, notamment celles basées sur l'architecture REST.

La syntaxe est très simple et a déjà été en grande partie introduite précédemment. Elle est présentée ci-dessous, mais vous pouvez aussi vous référer à des sites comme http://www.json.org/.

Valeurs simples

La structure de base est la paire (clé, valeur) (*key-value pair*).

```
"title": "The Social network"
```

Qu'est-ce qu'une valeur? On distingue les valeurs *atomiques* et les valeurs *complexes* (construites). Les valeurs atomiques sont:

- les chaînes de caractères (entourées par les classiques apostrophes doubles anglais (droits)),
- les nombres (entiers, flottants)
- les valeurs booléennes (true ou false).

Voici une paire (clé, valeur) où la valeur est un entier (NB: pas d'apostrophes).

```
"year": 2010
```

Et une autre avec un Booléen.

```
"oscar": false
```

Valeurs complexes

Les valeurs complexes sont soit des objets (agrégats de paires clé-valeur) soit des tableaux (séquences de valeurs). Un *objet* est un ensemble de paires clé-valeur dans lequel chaque clé n'apparaît qu'une fois.

```
{"last_name": "Fincher", "first_name": "David"}
```

Un objet est une *valeur* complexe et peut être utilisé comme valeur dans une paire clé-valeur.

```
"director": {
  "last_name": "Fincher",
  "first_name": "David",
  "birth_date": 1962
}
```

Un *tableau* (*array*) est une liste de valeurs dont les types peuvent varier: Javascript est un langage non typé et les tableaux peuvent contenir des éléments hétérogènes, même si ce n'est sans doute pas recommandé. Un tableau est une valeur complexe, utilisable dans une paire clé-valeur.

```
"actors": ["Eisenberg", "Mara", "Garfield", "Timberlake"]
```

L'imbrication est sans limite : on peut avoir des tableaux de tableaux, des tableaux d'objets contenant eux-mêmes des tableaux, etc.

Pour représenter un *document* avec JSON, nous adopterons simplement la contrainte que le constructeur de plus haut niveau soit un objet (en bref, *document* et *objet* sont synonymes).

```
{
  "title": "The Social network",
  "summary": "On a fall night in 2003, Harvard undergrad and \n
     programming genius Mark Zuckerberg sits down at his \n
     computer and heatedly begins working on a new idea. (...)",
  "year": 2010,
  "director": {"last_name": "Fincher",
               "first_name": "David"},
  "actors": [
    {"first_name": "Jesse", "last_name": "Eisenberg"},
    {"first_name": "Rooney", "last_name": "Mara"}
  ]
}
```

Exercices

Pour manipuler ou créer des documents JSON, il existe de nombreuses ressources sur le Web.

- Un validateur de documents JSON, bien utile pour détecter les fautes: http://jsonlint.com/
- Un éditeur en ligne montrant les versions sérialisées et arborescentes: http://jsoneditoronline.org/

Exercice : produire un jeu de documents JSON volumineux

Pour tester des systèmes avec un jeu de données de taille paramétrable, nous pouvons utiliser des générateurs de données. Voici quelques possibilités qu'il vous est suggéré d'explorer.

- le site http://generatedata.com/ est très paramétrable mais ne permet malheureusement pas (aux dernières nouvelles) d'engendrer des documents imbriqués; à étudier quand même pour produire des tableaux volumineux;
- https://github.com/10gen-labs/ipsum est un générateur de documents JSON spécifiquement conçu pour fournir des jeux de test à MongoDB.

Ipsum produit des documents JSON conformes à un schéma (http://json-schema.org). Un script Python (vous devez avoir un interpréteur Python installé sur votre machine) prend ce schéma en entrée et produit un nombre paramétrable de documents. Voici un exemple d'utilisation.

```
python ./pygenipsum.py --count 1000000 schema.jsch > bd.json
```

Lisez le fichier README pour en savoir plus. Vous êtes invités à vous inspirer des documents JSON produits par le site WebScope pour créer un schéma et engendrer un base de films avec quelques millions de documents. Pour notre base movies, (Suggestion: allez jeter un œil à http://www.jsonschema.net/).

```
{
   "type":"object",
   "$schema": "http://json-schema.org/draft-03/schema",
   "properties":{
      "_id": {"type":"string", "ipsum": "id"},
      "actors": {
            "type":"array",
            "items":
                  {
                        "type":"object",
                        "required":false,
                         "minItems": 2,
          "maxItems": 6,
                        "properties":{
                              "_id": {"type":"string", "ipsum": "id"},
                              "birth_date": {"type":"string", "format": "date"},
                              "first_name": {"type":"string", "ipsum": "fname"},
                              "last_name": {"type":"string", "ipsum": "fname"},
                              "role": {"type":"string"}
                        }
                  }
      },
      "genre": {"type":"string",
             "enum": [ "Drame", "Comédie", "Action", "Guerre", "Science-Fiction"]},
      "summary": {"type":"string", "required":false},
      "title": {"type":"string"},
      "year": {"type":"integer"},
      "country": {"type":"string",  "enum": [ "USA", "FR", "IT"]},
      "director": {
            "type":"object",
            "properties":{
                  "_id":{"type":"string"},
                  "birth_date": {"type":"string"},
                  "first_name":  {"type":"string", "ipsum": "fname"},
                  "last_name":{"type":"string", "ipsum": "fname"}
```

```
        }
    }
}
```