1. **Introdução**

JSON (*JavaScript Object Notation*) é um modelo para armazenamento e transmissão de informações no formato texto. Apesar de muito simples, tem sido bastante utilizado por aplicações *Web* devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML, tornando mais rápido o *parsing* dessas informações. Isto explica o fato de o JSON ter sido adotado por empresas como *Google* e *Yahoo*, cujas aplicações precisam transmitir grandes volumes de dados.

1. **Sintaxe no JSON**

A ideia utilizada pelo **JSON para representar informações é tremendamente simples: para cada valor representado, atribui-se um nome (ou rótulo) que descreve o seu significado**. Esta sintaxe é derivada da forma utilizada pelo JavaScript para representar informações. Por exemplo, para representar o ano de 2012, utiliza-se a seguinte sintaxe:

* **Listagem 1:** Representando o ano de 2012

"ano": 2012

Um par nome/valor deve ser representado pelo nome entre aspas duplas, seguido de dois pontos, seguido do valor. Os valores podem possuir apenas 3 **tipos básicos**: numérico (inteiro ou real), booleano e string. As Listagens 2, 3, 4 e 5 apresentam exemplos. Observe que os valores do tipo string devem ser representados entre aspas.

* **Listagem 2:** Representando um número real

"altura": 1.72

* **Listagem 3:** Representando uma *string*

"site": “www.devmedia.com.br”

* **Listagem 4:** Representando um número negativo

"temperatura": -2

* **Listagem 5:** Representando um valor booleano

"casado": true

A partir dos tipos básicos, é possível construir **tipos complexos**: *array* e objeto. Os ***[arrays](https://www.devmedia.com.br/javascript-arrays/4079" \o "Arrays no JavaScript" \t "_blank)*** são delimitados por colchetes, com seus elementos separados entre vírgulas. As listagens 6 e 7 mostram exemplos.

* **Listagem 6:** *Array* de *Strings*

[“RJ”, “SP”, “MG”, “ES”]

* **Listagem 7:** Matriz de Inteiros

[

  [1,5],

  [-1,9],

  [1000,0]

 ]

Os objetos são especificados entre chaves e podem ser compostos por múltiplos pares nome/valor, por *arrays* e também por outros objetos. Desta forma, um objeto JSON pode representar, virtualmente, qualquer tipo de informação! O exemplo da Listagem 8 mostra a representação dos dados de um filme.

* **Listagem 8:** Objeto

{

  “titulo”: “JSON x XML”,

  “resumo”: “o duelo de dois modelos de representação de informações”,

  “ano”: 2012,

  “genero”: [“aventura”, “ação”, “ficção”]

 }

É possível representar mais de um objeto ou registro de uma só vez. Um exemplo é apresentado na Listagem 9, onde dois filmes são representados em um *array*.

* **Listagem 9:** *Array* de objetos

[

   {

     “titulo”: “JSON x XML”,

     “resumo”: “o duelo de dois modelos de representação de informações”,

     “ano”: 2012,

     “genero”: [“aventura”, “ação”, “ficção”]

    },

   {

     “titulo”: “JSON James”,

     “resumo”: “a história de uma lenda do velho oeste”,

     “ano”: 2012,

     “genero”: [“western”]

    }

]

Por fim, é importante citar que a palavra-chave “*null*” deve ser utilizada para a representação de valores nulos (Listagem 10).

* **Listagem 10:** Representando um valor nulo

"site":null

### JSON x XML

Podemos entender o **JSON como uma espécie de “concorrente” da XML na área de troca de informações**. Nesta seção, apresentamos algumas das principais semelhanças e diferenças entre os dois modelos para a representação de informações.

**Semelhanças:**

* Os dois modelos representam informações no formato texto.
* Ambos possuem natureza auto-descritiva (ou seja, basta “bater o olho” em um arquivo JSON ou em um arquivo XML para entender o seu significado).
* Ambos são capazes de representar informação complexa, difícil de representar no formato tabular. Alguns exemplos: objetos compostos (objetos dentro de objetos), relações de hierarquia, atributos multivalorados, *arrays*, dados ausentes etc.
* Ambos podem ser utilizados para transportar informações em aplicações [**AJAX**](https://www.devmedia.com.br/o-que-e-o-ajax/6702).
* Ambos podem ser considerados padrões para representação de dados. XML é um padrão W3C, enquanto JSON foi formalizado na RFC 4627.
* Ambos são independentes de linguagem. Dados representados em XML e **JSON podem ser acessados por qualquer linguagem de programação, através de API’s específicas**.

**Diferenças:**

* JSON não é uma linguagem de marcação. Não possui *tag* de abertura e muito menos de fechamento!
* JSON representa as informações de forma mais compacta.
* JSON não permite a execução de instruções de processamento, algo possível em XML.
* JSON é tipicamente destinado para a troca de informações, enquanto XML possui mais aplicações. Por exemplo: nos dias atuais existem bancos de dados inteiros armazenados em XML e estruturados em SGBD’s XML nativo.

Nessa aula, apresentamos os conceitos fundamentais sobre o modelo JSON para a representação de informações - **atualmente muito utilizado em aplicações AJAX. Através do JSON podemos representar informações de uma forma mais compacta,** rápida e simples do que na XML. Em contrapartida, a XML é uma linguagem mais rica, que possui um maior número de aplicações e um maior grau de maturidade. Existem inúmeros *parsers* para JSON, disponíveis para praticamente todas as linguagens. O artigo apresentou um exemplo de programa Java que fez uso da biblioteca “*gson*”, desenvolvida pelo *Google*.

Esse conteúdo foi adaptado do DevMedia

<https://www.devmedia.com.br/json-tutorial/25275>