**JSON (JavaScript Object Notation),**sua sintaxe simples, leveza e objetividade junto ao suporte de dezenas de linguagens de programação, a torna uma das melhores formas, se não a melhor, para transmitir grande quantidade de dados entre aplicativos diversos com linguagens diversas, sendo assim ótimo para lidar com webservices, comunicação cliente-servidor, servidor-cliente e servidor-servidor. É um modelo tipicamente usado para armazenamento, a partir da entrada do usuário, e transmissão de informações no formato **texto**, ou seja, suas informações dentro do arquivo “.json” são em formato de string.

**Um breve comparativo entre JSON e XML**

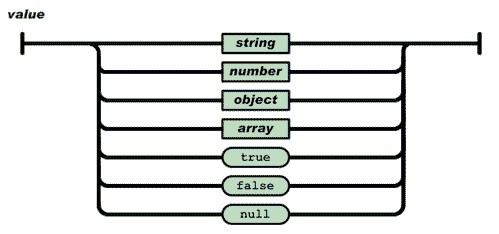
Assim como o JSON, existe o XML, outro modelo para troca de dados entre aplicativos, porém mais complexo de ser entendido e possui uma sintaxe verbosa onde toda tag aberta tem que ser fechada, ao contrário do JSON onde a linguagem é mais compacta, de fácil aprendizado e implementação. Outros pontos podem ser destacados como desvantagens do XML com um processamento mais lento, visto que o JSON provém do JavaScript e a comunicação cliente é feita via browser onde esses possuem JavaScript nativo, além do tamanho do arquivo ser maior, a depender da quantidade de informações que conter. Podemos entender o JSON como uma espécie de “concorrente” da XML na área de troca de informações**.**

Ambos são considerados padrões para representação de dados, o XML é padrão W3C, enquanto JSON foi formalizado na RFC 4627, assim como são independentes de linguagens, dados representados em XML e JSON pode ser acessados por qualquer linguagem de programação através de API’s específicas, ficando mais fácil a utilização do JSON quando a linguagem de acesso é JavaScript.

**Sintaxe**

A ideia utilizada pelo JSON para representar informações é muito simples: para cada valor representado, atribui-se uma chave, onde cada chave é identificada por uma string (entre aspas)**,**que descreve o seu significado(valor). Esta sintaxe é derivada da forma utilizada pelo JavaScript para representar informações. Originalmente, o JSON foi concebido para se trabalhar com javascript, porém não é necessariamente obrigatório o uso de javascript para seu uso, dado que, atualmente, existem muitas API’s que fazem o trabalho de transforma o JSON legível para diversas linguagens.

Os valores podem possuir 3 tipos básicos: **numérico** (inteiro ou real), **booleano** e **string,** alem de alguns outros tipos citados mais a baixo. Os valores do tipo string devem ser representados entre aspas.

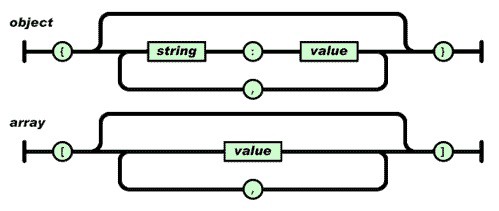


Representando um número real.  
“altura” : 1.72

Representando uma string.  
“site” : “www.google.com”

Representando um valor booleano  
“casado” : true

A partir dos tipos básicos, é possível construir tipos complexos como **array** e **objeto**. O array vai ser sempre dentro de colchetes “[]”, onde vai indicar um conjunto de elementos, sejam variados tipos de dados ou até mesmo objetos e outros arrays.



Array de strings, onde cada elemento é uma string.  
[“RJ”, “SP”, “MG”, “ES”]

Matriz de inteiros, é quando possuímos arrays dentro de arrays.  
[  
[1,5],  
[-1,9],  
[1000,0]  
]

Array com tipos de dados variados  
[  
33,  
“Segunda”,  
TRUE,  
3.14159265,  
NULL,  
“ma&ccedil;&atilde;”  
]

O objeto vai ser sempre dentro de chaves “{}”, onde vai indicar um conjunto de chaves(nome) e valores, no qual esses valores podem ser qualquer dos três tipos básicos como também arrays e outros objetos.

Objeto.

{  
“titulo”: “JSON x XML”,  
“resumo”: “o duelo de dois modelos de representação de informações”,  
 “ano”: 2012,  
“genero”: [“aventura”, “ação”, “ficção”]  
}

Objeto Aninhado, um objeto com duas chaves onde o valor de cada chave é outro objeto.

{  
 “sammy” : {  
 “username” : “SammyShark”,  
 “location” : “Indian Ocean”,  
 “online” : true,  
 “followers” : 987  
 },  
 “jesse” : {  
 “username” : “JesseOctopus”,  
 “location” : “Pacific Ocean”,  
 “online” : false,  
 “followers” : 432  
 }  
}

É possível representar mais de um objeto ou registro de uma só vez.

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”]  
}]

Podemos também declarar o valor da chave como um array de objetos, observado que não há problema se não houver espaço separando os dois pontos das strings.

{  
"first\_name" : "Sammy",  
 "last\_name" : "Shark",  
 "location" : "Ocean",  
 "websites" : [{  
 "description" : "work",  
 "URL" : "https://www.digitalocean.com/"  
 },  
 {  
 "desciption" : "tutorials",  
 "URL":"https://www.digitalocean.com/community/tutorials"  
 } ]  
}

**JSON e JavaScript**

Se estivermos usando o objeto JSON em um “.js” ou “.html”, provavelmente veremos esse objeto sendo atribuído a uma variável.

**let** sammy = {“first\_name” : “Sammy”, “last\_name” : “Shark”, “location” : “Ocean”};

O objeto em JavaScript é um pouco diferente do objeto JSON, pois existem algumas limitações no JSON em relação ao objeto JavaScript, como por exemplo em JavaScript é possível atribuir como valor de uma chave, uma função, onde no JSON não é possível, além do mais, objetos em JavaScript só podem ser usados em JavaScript, ao contrário do objeto JSON. Outro ponto é que as chaves do objeto JavaScript não são declarados está entre aspas, diferente do objeto JSON.

**let** user = {  
 first\_name: “Sammy”,  
 last\_name : “Shark”,  
 online : true,  
 full\_name : **function**() {  
 **return** **this**.first\_name + “ “ + **this**.last\_name;  
 }};

Ao atribuirmos o objeto de JavaScript para a variável user, se quisermos acessar os dados no objeto, poderemos usar o[ponto](https://www.digitalocean.com/community/tutorials/how-to-work-with-json-in-javascript#accessing-json-data) junto a variável para chamar o valor da chave desejada, como exemplo user.first\_name; e obter uma string, mas se quisermos acessar o nome completo, precisamos fazer isso chamando a função” user.full\_name();” que vai chamar e concatenar o valor das duas chaves, como mostra o exemplo a cima.

Da mesma forma o objeto JSON é acessado em JavaScript, onde se quisermos acessar o primeiro nome e status,usamossammy.first\_name;e sammy.online;, respectivamente. Existe outra forma de acesso usando o colchete junto a variável passando a chave entre aspas dentro do colchetes, por exemplo, sammy["first\_name"] e sammy["online"];.

**let** sammy = {  
“first\_name” : “Sammy”,  
 “last\_name” : “Shark”,  
 “online” : true  
}

Quando trabalhamos com array como valor de alguma chave, devemos chamar o índice do elemento desse array para retornar o elemento(objeto) por inteiro ou acrescentar o ponto mais a chave para chamar especificamente o valor daquela chave daquele elemento do array, ou seja, temos aqui um array de objetos dentro de outro objeto sendo passado como valor de uma chave.

**let** user\_profile = {  
 "username" : "SammyShark",  
"social\_media" : [{  
 "description" : "twitter",  
 "link" : "https://twitter.com/digitalocean"  
 },  
 {  
 "description" : "facebook",  
 "link" : "https://www.facebook.com/DigitalOceanCloudHosting"  
 },  
 {  
 "description" : "github",  
 "link" : "https://github.com/digitalocean"  
 }]  
}

Para acessar o valor da chave description cuja string é facebook, devemos fazer o seguindo comando, user\_profile.social\_media[1].description, user\_profile é a variável que foi atribuída ao objeto, social\_media[1] é a chave do primeiro objeto que vai conter o array de objetos, 1 é a posição de qual objeto eu quero pegar dentro do array, que é passado dentro dos colchetes e .description é a chave desse objeto que estou pegando, retornando assim o valor dessa última chave.

#### ****JSON.stringify()****

Um método bastante interessante no JavaScript é o JSON.stringify(), passando por parâmetro um objeto JSON que retornar o próprio objeto no formado de string, ou seja, aspas antes da abertura e fechamento das chaves, facilitando, dessa forma, a transmissão de dados entre diferentes aplicativos e linguagens.

**let** sammy = {“first\_name” : “Sammy”, “last\_name” : “Shark”, “location” : “Ocean”};

let obj = JSON.stringify(sammy)

obj == ‘{“first\_name” : “Sammy”, “last\_name” : “Shark”, “location” : “Ocean”}’; //Saida

#### JSON.parse()

Para converter o objeto de string para JSON, é usado o método JSON.parse(), sendo atribuído a uma nova variável, passando por parâmetro a string(objeto), ficando igual a primeira forma.

let aux = JSON.parse(obj)

aux == {“first\_name” : “Sammy”, “last\_name” : “Shark”, “location” : “Ocean”};  
//sammy == aux

#### ****Conclusão****

Podemos concluir que o JSON e JavaScript vão sempre trabalhar muito bem juntos, levando em consideração que a linguagem amplamente utilizada nos navegadores é JavaScript. Ela está sendo bastante utilizada também para troca de informações entre servidores por ser uma linguagem mais simples de manipular.

Referencias:

http://desenvolvimentoparaweb.com/javascript/json-javascript-object-notation/

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

https://tools.ietf.org/html/rfc4627

http://www.json.org/

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global\_Objects/JSON/parse

https://www.digitalocean.com/community/tutorials/an-introduction-to-json

https://www.digitalocean.com/community/tutorials/how-to-work-with-json-in-javascript

https://en.wikipedia.org/wiki/Comparison\_of\_data\_serialization\_formats