JSON

**JSON:** Javascript Object Notation, é uma sintaxe para armazenar e trocar dados, é um texto escrito com a notação de objeto JavaScript.

Trocando Dados

Ao trocar dados entre um navegador e um servidor, os dados só podem ser texto. JSON é texto, e podemos converter qualquer objeto JavaScript em JSON, e enviar JSON para o servidor. Também podemos converter qualquer JSON recebido do servidor em objetos JavaScript. Desta forma, podemos trabalhar com os dados como objetos JavaScript, sem parsing complicado e traduções.

Envio de dados

Se você tiver dados armazenados em um objeto JavaScript, você pode converter o objeto em JSON e enviá-lo para um servidor: var myObj = { "name":"John", "age":31, "city":"New York" };  
var myJSON = JSON.stringify(myObj);  
window.location = "demo\_json.php?x=" + myJSON;

Recebendo dados

Se você recebe dados no formato JSON, poderá convertê-los em um objeto JavaScript: var myJSON = '{ "name":"John", "age":31, "city":"New York" }';  
var myObj = JSON.parse(myJSON);  
document.getElementById("demo").innerHTML = myObj.name;

Armazenando dados

Ao armazenar dados, os dados têm de ser um determinado formato, e independentemente de onde você escolher para armazená-lo, o texto é sempre um dos formatos legais. O JSON torna possível armazenar objetos JavaScript como texto.

myObj = { "name":"John", "age":31, "city":"New York" };  
myJSON = JSON.stringify(myObj);  
localStorage.setItem("testJSON", myJSON); // Armazenamento de dados

text = localStorage.getItem("testJSON");  
obj = JSON.parse(text);  
document.getElementById("demo").innerHTML = obj.name; // Recuperando dados

O que é JSON?

* JSON significa JavaScript Object Notation
* JSON é um formato de intercâmbio de dados leve
* JSON é "auto-descritivo" e fácil de entender
* JSON é independente de linguagem \*

\*O JSON usa a sintaxe JavaScript, mas o formato JSON é somente texto. Texto pode ser lido e usado como formato de dados por qualquer linguagem de programação.

Por que usar o JSON?

Como o formato JSON é somente texto, ele pode ser facilmente enviado para e de um servidor e usado como um formato de dados por qualquer linguagem de programação. O JavaScript possui uma função incorporada para converter uma string, escrita em formato JSON, em objetos JavaScript nativos:

**JSON.parse()**

Portanto, se você receber dados de um servidor, no formato JSON, poderá usá-lo como qualquer outro objeto JavaScript.

Regras de sintaxe do JSON

A sintaxe JSON é um subconjunto da sintaxe JavaScript. A sintaxe JSON é derivada da sintaxe de notação de objeto JavaScript:

* Os dados estão em pares nome / valor
* Os dados são separados por vírgulas
* Suporte para encapsular objetos
* Suporte a arrays por colchetes
* Dados JSON - um nome e um valor
* Os dados JSON são escritos como pares nome / valor.

Um par nome / valor consiste em um nome de campo (em aspas duplas), seguido por dois pontos, seguido por um valor: "name":"John"

JSON - avalia objetos JavaScript

O formato JSON é quase idêntico a objetos JavaScript. Em JSON, as chaves devem ser strings, escritas com aspas duplas: { "name":"John" }

Em JavaScript, as chaves podem ser cadeias de caracteres, números ou nomes de identificadores: { name:"John" }

Valores JSON

No JSON, os valores devem ser um dos seguintes tipos de dados:

* uma linha
* um número
* um objeto (objeto JSON)
* uma matriz
* um booleano
* null

Em valores JavaScript podem ser todos os acima, além de qualquer outra expressão JavaScript válida, incluindo:

* uma função
* uma data
* undefined

No JSON, os valores de string devem ser escritos com aspas duplas:{ "name":"John" } Em JavaScript, você pode escrever valores de string com aspas duplas ou simples: { name:'John' }

JSON usa sintaxe JavaScript. Como a sintaxe JSON é derivada da notação de objeto JavaScript, muito pouco software adicional é necessário para trabalhar com o JSON dentro do JavaScript. Com o JavaScript você pode criar um objeto e atribuir dados a ele, como este: var person = { "name":"John", "age":31, "city":"New York" };

Você pode acessar um objeto JavaScript como este:   
person.name; // retorna John

Também pode ser acessado da seguinte forma:   
person["name"];// retorna John

Os dados podem ser modificados da seguinte forma: person.name = "Gilbert";

Também pode ser modificado da seguinte forma: person["name"] = "Gilbert";

Arrays JavaScript como JSON

Da mesma forma que os objetos JavaScript podem ser usados como JSON, os arrays JavaScript também podem ser usados como JSON.

**JSON vs XML**

Tanto o JSON como o XML podem ser usados para receber dados de um servidor web. Exemplo JSON:

{"employees":[  
    { "firstName":"John", "lastName":"Doe" },  
    { "firstName":"Anna", "lastName":"Smith" },  
    { "firstName":"Peter", "lastName":"Jones" }  
]}

Exemplo XML:

<employees>  
    <employee>  
        <firstName>John</firstName> <lastName>Doe</lastName>  
    </employee>  
    <employee>  
        <firstName>Anna</firstName> <lastName>Smith</lastName>  
    </employee>  
    <employee>  
        <firstName>Peter</firstName> <lastName>Jones</lastName>  
    </employee>  
</employees>

JSON é como o XML, porque

* Ambos JSON e XML são "auto-descritivos" (legível por humanos)
* Tanto o JSON como o XML são hierárquicos (valores dentro de valores)
* Tanto o JSON quanto o XML podem ser analisados e usados por muitas linguagens
* Tanto o JSON quanto o XML podem ser obtidos com um XMLHttpRequest

JSON é diferente de XML porque

* JSON não usa tag de encerramento
* JSON é mais curto
* JSON é mais rápido de ler e escrever
* JSON pode usar matrizes

**A maior diferença é:** XML tem de ser analisado com um analisador XML. O JSON pode ser analisado por uma função JavaScript padrão.

Por que o JSON é melhor do que XML

* O XML é muito mais difícil de analisar do que o JSON.
* O JSON é analisado em um objeto JavaScript pronto para usar.

Para aplicações AJAX, o JSON é mais rápido e mais fácil do que o XML:

Usando XML

* Obter um documento XML
* Use o XML DOM para percorrer o documento
* Extrair valores e armazenar em variáveis

Usando o JSON

* Obter uma JSON string
* JSON.Parse a JSON string

**Tipos de dados JSON**

No JSON, os valores devem ser um dos seguintes tipos de dados:

* uma string {"Nome": "João"}
* um número {"Idade": 30}
* Um objeto (objeto JSON) {"Empregado": {"nome": "João", "idade": 30}
* uma array {"Funcionários": ["João", "Ana", "Pedro"]
* um booleano {"Venda": true}
* nulo {"Nomemeio": null}

Valores JSON não podem ser um dos seguintes tipos de dados:

* uma função
* uma data
* undefined
* strings JSON

As seqüências em JSON devem ser escritas entre aspas.

**Objetos JSON**

* Objetos JSON são rodeados por chaves {}.
* Objetos JSON são escritos em pares chave / valor.
* As chaves devem ser strings, e os valores devem ser um tipo de dados JSON válido
* Chaves e valores são separados por dois pontos.
* Cada par chave / valor é separado por uma vírgula.

Acessando Valores de Objeto

Você pode acessar os valores do objeto usando a notação ponto (.): myObj = { "name":"John", "age":30, "car":null };  
x = myObj.name; //retorna John

Você também pode acessar os valores do objeto usando a notação de colchetes([]): myObj = { "name":"John", "age":30, "car":null };  
x = myObj["name"]; //retorna John

Fazendo um loop em um objeto

Você pode percorrer as propriedades do objeto usando o loop for-in: myObj = { "name":"John", "age":30, "car":null };  
for (x in myObj) {  
    document.getElementById("demo").innerHTML += x; } //retorna name age e car

Em um loop for-in, use a notação de suporte para acessar os valores de propriedade: myObj = { "name":"John", "age":30, "car":null };  
for (x in myObj) {  
    document.getElementById("demo").innerHTML += myObj[x];  
} //retorna John 30 e null

Objetos JSON aninhados

Valores em um objeto JSON podem ser outro objeto JSON: myObj = {  
    "name":"John",  
    "cars": {  
        "car1":"Ford",  
        "car2":"BMW",  
        "car3":"Fiat" } }

Você pode acessar objetos JSON aninhados usando a notação de ponto ou a notação de colchetes: x = myObj.cars.car2; //ou: x = myObj.cars["car2"]; //BMW

Modificar Valores

Você pode usar a notação ponto para modificar qualquer valor em um objeto JSON: myObj.cars.car2 = "Mercedes"; //ou myObj.cars["car2"] = "Mercedes";

Excluir Propriedades do Objeto

Utilizar a palavra-chave de exclusão para excluir as propriedades de um objeto JSON: delete myObj.cars.car2; //deleta o carro 2

**JSON Arrays**

Arrays como objetos JSON: [ "Ford", "BMW", "Fiat" ]

Arrays em JSON são quase os mesmos que arrays em JavaScript. Em JSON, valores de arrays devem ser do tipo de string, número, objeto, array, booleana ou nulo.

Em JavaScript, valores da matriz pode ser todos os itens acima, além de qualquer outra expressão JavaScript válido, incluindo funções, datas e indefinido.

Arrays em objetos JSON

Arrays podem ser valores de uma propriedade do objeto: {  
"name":"John",  
"age":30,  
"cars":[ "Ford", "BMW", "Fiat" ]  
}

Acessando valores de arrays

Você acessa os valores do array usando o número de índice: x = myObj.cars[0]; //Ford

Looping através de uma matriz

Você pode acessar valores de matriz usando um loop for-in: for (i in myObj.cars) {  
    x += myObj.cars[i];  
} // retorna Ford BMW Fiat

Ou você pode usar um loop for: for (i = 0; i < myObj.cars.length; i++) {  
    x += myObj.cars[i];  
} // retorna Ford BMW Fiat

Arrays aninhados em objetos JSON

Valores em um array também podem ser outro array, ou mesmo outro objeto JSON: myObj = {  
    "name":"John",  
    "age":30,  
    "cars": [  
        { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },  
        { "name":"BMW", "models":[ "320", "X3", "X5" ] },  
        { "name":"Fiat", "models":[ "500", "Panda" ] }  
    ]  
 }

Para acessar matrizes dentro de arrays, use um loop for-in para cada array: for (i in myObj.cars) {  
    x += "<h1>" + myObj.cars[i].name + "</h1>";  
    for (j in myObj.cars[i].models) {  
        x += myObj.cars[i].models[j];  
    }  
}

Modificar valores de matriz

Use o número de índice para modificar uma matriz: myObj.cars[1] = "Mercedes";

Excluir itens de matriz

Use a palavra-chave delete para excluir itens de uma matriz: delete myObj.cars[1];

# JSON.parse()

Um uso comum do JSON é trocar dados de / para um servidor web. Ao receber dados de um servidor web, os dados são sempre uma seqüência de caracteres. Analise os dados com JSON.parse (), e os dados se tornam um objeto JavaScript. Exemplo - Parsing JSON. Imagine que recebemos este texto de um servidor web: '{ "name":"John", "age":30, "city":"New York"}'

Use a função JavaScript JSON.parse () para converter texto em um objeto JavaScript: var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');

Certifique-se de que o texto está escrito em formato JSON, ou então você receberá um erro de sintaxe. Use o objeto JavaScript em sua página: <p id="demo"></p>   
<script>  
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;   
</script> //retorna John, 30

JSON a partir do servidor

Você pode solicitar JSON do servidor usando um pedido AJAX. Contanto que a resposta do servidor seja escrita em formato JSON, você pode analisar a seqüência de caracteres em um objeto JavaScript. Use o XMLHttpRequest para obter dados do servidor: var xmlhttp = new XMLHttpRequest();  
xmlhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
        myObj = JSON.parse(this.responseText);  
        document.getElementById("demo").innerHTML = myObj.name;  
    }  
};  
xmlhttp.open("GET", "json\_demo.txt", true);  
xmlhttp.send(); //retorna John

Array como JSON

Ao usar o JSON.parse () em um JSON derivado de um array, o método retornará um array JavaScript, em vez de um objeto JavaScript. O JSON retornado do servidor é um array: var xmlhttp = new XMLHttpRequest();  
xmlhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
        myArr = JSON.parse(this.responseText);  
        document.getElementById("demo").innerHTML = myArr[0];  
    }  
};  
xmlhttp.open("GET", "json\_demo\_array.txt", true);  
xmlhttp.send(); //retorna Ford

Exceções

Parsing Datas. Objetos de data não são permitidos no JSON. Se você precisar incluir uma data, escreva-a como uma string. Você pode convertê-lo novamente em um objeto de data mais tarde: var text = '{ "name":"John", "birth":"1986-12-14"}';  
var obj = JSON.parse(text);  
obj.birth = new Date(obj.birth); //muda o valor da data  
document.getElementById("demo").innerHTML = obj.name + ", " + obj.birth; //retorna John + data de nascimento

Ou, você pode usar o segundo parâmetro, da função JSON.parse (), chamado reviver. O parâmetro reviver é uma função que verifica cada propriedade, antes de retornar o valor:

var text = '{ "name":"John", "birth":"1986-12-14"}';  
var obj = JSON.parse(text, function (key, value) {  
    if (key == "birth") {  
        return new Date(value);  
    } else {  
        return value;  
    }});  
document.getElementById("demo").innerHTML = obj.name + ", " + obj.birth; //retorna John + data de nascimento

Funções Parsing

As funções não são permitidas no JSON. Se você precisar incluir uma função, escreva-a como uma string. Você pode convertê-lo novamente em uma função mais tarde. Converter uma string em uma função: var text = '{ "name":"John", "age":"function () {return 30;}", "city":"New York"}';  
var obj = JSON.parse(text);  
obj.age = eval("(" + obj.age + ")");  
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age();

Você deve evitar usar funções no JSON, as funções perderão seu escopo, e você teria que usar eval () para convertê-las de volta em funções.

**JSON.stringify()**

Um uso comum do JSON é trocar dados de / para um servidor web. Ao enviar dados para um servidor web, os dados devem ser uma string. Converta um objeto JavaScript em uma seqüência de caracteres com JSON.stringify ().

Stringify um objeto JavaScript

Imagine que temos esse objeto em JavaScript: var obj = { "name":"John", "age":30, "city":"New York"};

Use a função JavaScript JSON.stringify () para convertê-la em uma seqüência de caracteres: var myJSON = JSON.stringify(obj);

O resultado será uma string seguindo a notação JSON. MyJSON agora é uma string, e pronto para ser enviado para um servidor: var obj = { "name":"John", "age":30, "city":"New York"};  
var myJSON = JSON.stringify(obj);  
document.getElementById("demo").innerHTML = myJSON; //retorna {"name":"John","age":30,"city":"New York"}

Stringify um array de JavaScript

Também é possível encadear conjuntos de códigos JavaScript. Imagine que temos esse array em JavaScript: var arr = [ "John", "Peter", "Sally", "Jane" ];

Use a função JavaScript JSON.stringify () para convertê-la em uma seqüência de caracteres: var myJSON = JSON.stringify(arr);

O resultado será uma string string seguindo a notação JSON. MyJSON é uma string, e pronto para ser enviado para um servidor: var arr = [ "John", "Peter", "Sally", "Jane" ];  
var myJSON = JSON.stringify(arr);  
document.getElementById("demo").innerHTML = myJSON; //retorna ["John","Peter","Sally","Jane"]

Stringify uma matriz de JavaScript

Também é possível encadear conjuntos de códigos JavaScript. Imagine que temos essa matriz em JavaScript: var arr = [ "John", "Peter", "Sally", "Jane" ];

Use a função JavaScript JSON.stringify () para convertê-la em uma seqüência de caracteres: var myJSON = JSON.stringify(arr);

O resultado será uma string seguindo a notação JSON. MyJSON agora é uma string, e pronto para ser enviado para um servidor: var arr = [ "John", "Peter", "Sally", "Jane" ];  
var myJSON = JSON.stringify(arr);  
document.getElementById("demo").innerHTML = myJSON; //retorna ["John","Peter","Sally","Jane"]

**Exceções**

Datas Stringify

No JSON, objetos de data não são permitidos. A função JSON.stringify () converterá quaisquer datas em seqüências de caracteres: var obj = { "name":"John", "today":new Date(), "city":"New York"};  
var myJSON = JSON.stringify(obj);  
document.getElementById("demo").innerHTML = myJSON; //retorna {"name":"John","today":"(dia e horário)","city":"New York"}

Você pode converter a seqüência de caracteres de volta em um objeto de data no receptor.

Stringify Funções

No JSON, as funções não são permitidas como valores de objeto. A função JSON.stringify () removerá todas as funções de um objeto JavaScript, tanto a chave como o valor: var obj = { "name":"John", "age":function() {return 30;}, "city":"New York"};  
var myJSON = JSON.stringify(obj);  
document.getElementById("demo").innerHTML = myJSON; //retorna {"name":"John","city":"New York"}

JSON.stringify removerá todas as funções de um objeto.