JSON

Professor Vinícius Costa

Introdução ao JSON

- JSON: **J**ava**S**cript **O**bject **N**otation.
  - Serve para amarazenar e trocar dados
  - É texto escrito com a notação JavaScript

Troca de Dados

- A troca de dados entre navegador e servidor deve ser em formato texto
- É possível converter objetos JavaScript em texto JSON para envia-lo ao servidor
- É possível converter texto JSON recebido do servidor em objeto JavaScript
- Esse processo de texto para objeto utilizando JSON é simples

Armazenamento de Dados

- É possível armazenar textos dentro do navegador, por exemplo, no localStorage
- JSON é texto e pode ser armazenado no localStorage

O que é JSON

- Significa JavaScript Object Notation
- É um formato leve para troca de dados
- É auto descritivo e de fácil entendimento
- É independente de linguagem
  - JSON usa a sintaxe JavaScript, entretanto JSON é texto, sendo assim é possível utiliza-lo com qualquer linguagem de programação

JSON não é JavaScript

Tipos JavaScript	**Diferenças com JSON**
Objetos e Arrays	Nomes de propriedades devem ser strings envoltas em aspas duplas. Virgulas a direita não são permitidas
Números	Não é possível colocar zeros a esquerda. Após o ponto decimal deve existir pelo menos um dígito.
String	Apenas um conjunto de caracteres pode ser escapado; alguns caracteres de controle não são permitidos; as strings não podem estar entre apostrofos.

Por que Usar JSON

- É somente texto
- É facilmente enviado de/para um servidor
- Pode ser usado por qualquer linguagem

JSON versus XML

- Tanto JSON quanto XML servem para armazenar e trocar dados entre aplicações.

JSON versus XML

JSON

```
{
        “nome”: “José”,
        “idade”: 20
}
```

XML

```
<pessoa>
        <nome>José</nome>
        <idade>20</idade>
</pessoa>
```

JSON versus XML

Semelhanças

- Auto Descritivo
- Hierárquico
- Pode ser analisado por diversas linguagens
- Pode ser utilizado com Ajax

Diferenças

- JSON não tem tag de fechamento
- JSON é menor
- JSON é mais rápido para ler e escrever
- JSON permite o uso de vetores

Sintaxe do JSON

- Sua sintaxe é derivada da notação de objetos do JavaScript
- Dado é um par de nome/valor
- Dado é separado por virgula
- Objetos são envolvidos por chaves
- Arrays são envolvidos por colchetes

Dados JSON – Nome e valor

- JSON é escrito com um par de nomes e valores
  - Campo nome, seguido de dois pontos, seguido do valor
  - Exemplo

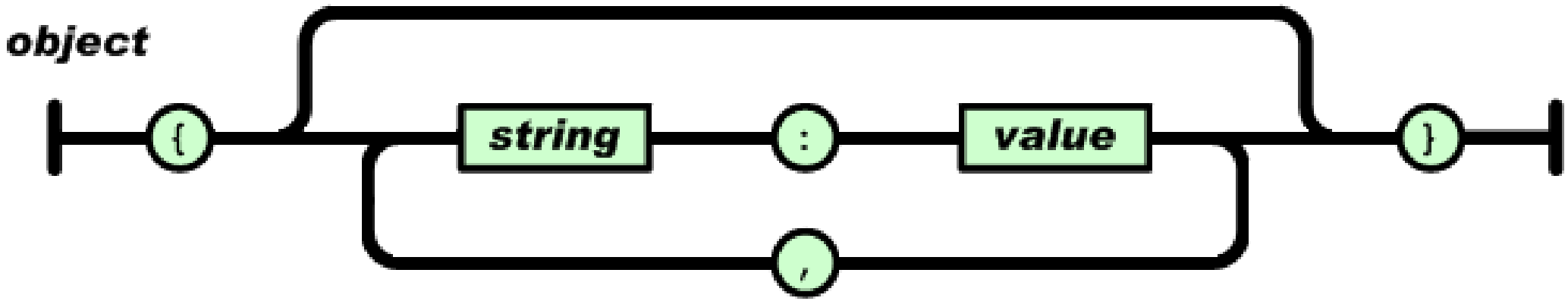
    "nome":"Jose"
  - Observação: os nomes dos campos devem estar entre aspas (em JavaScript não precisa estar entre aspas)

Objetos JSON

- Deve estar entre chaves
- Cojunto de chaves e valores separados por dois pontos
- As chaves devem estar entre aspas
- Os conjuntos de chaves e valores são separados um do outro por virgula
- Exemplo

  { “nome”: “José”, “idade”: 30 }

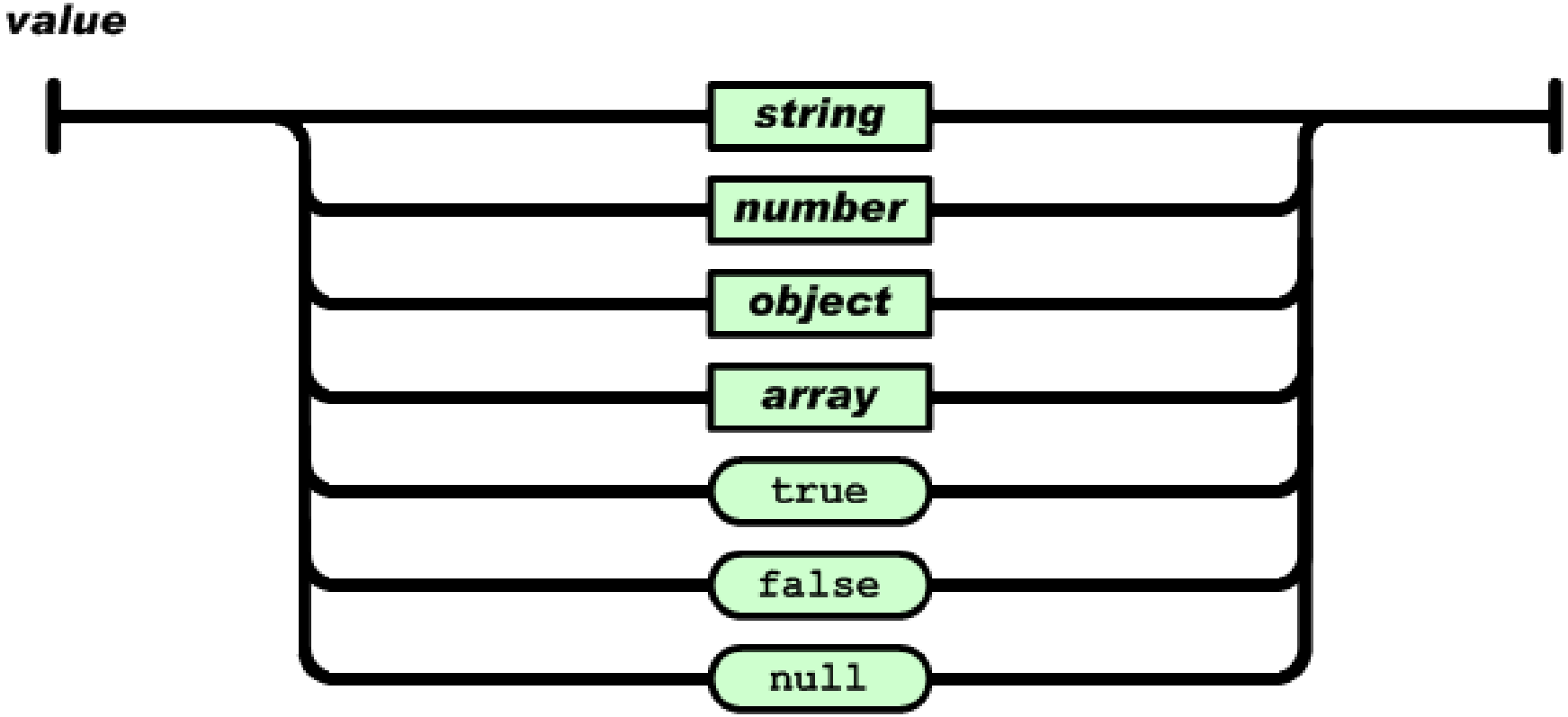
Objetos JSON



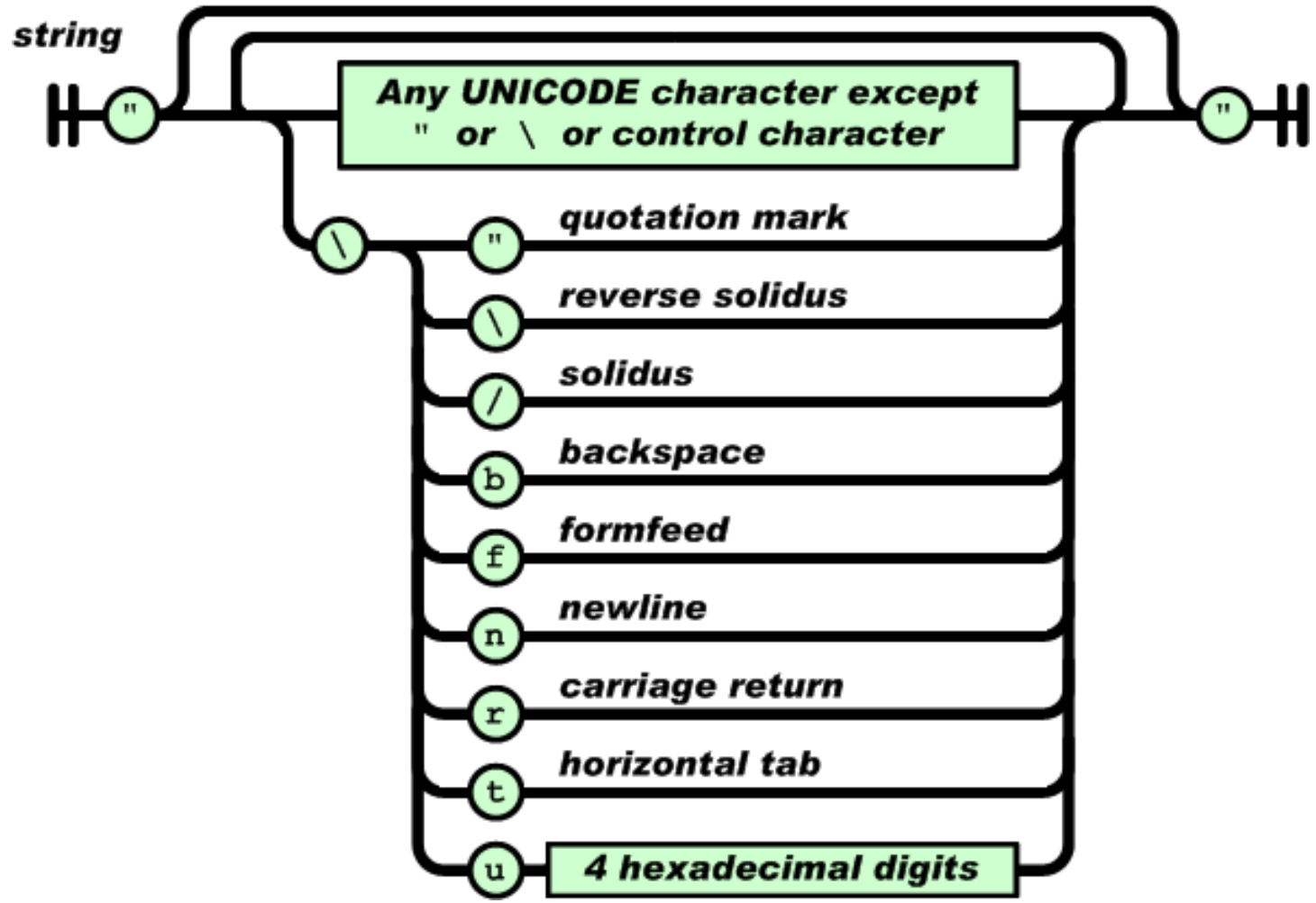
Valores JSON

- Os valores podem ser:
  - String (entre aspas)
  - Numeros
  - Objetos
  - Array
  - Booleano
  - null
- Não pode ser
  - Funções JavaScript
  - Datas
  - undefined
  - String (entre apostrofo)

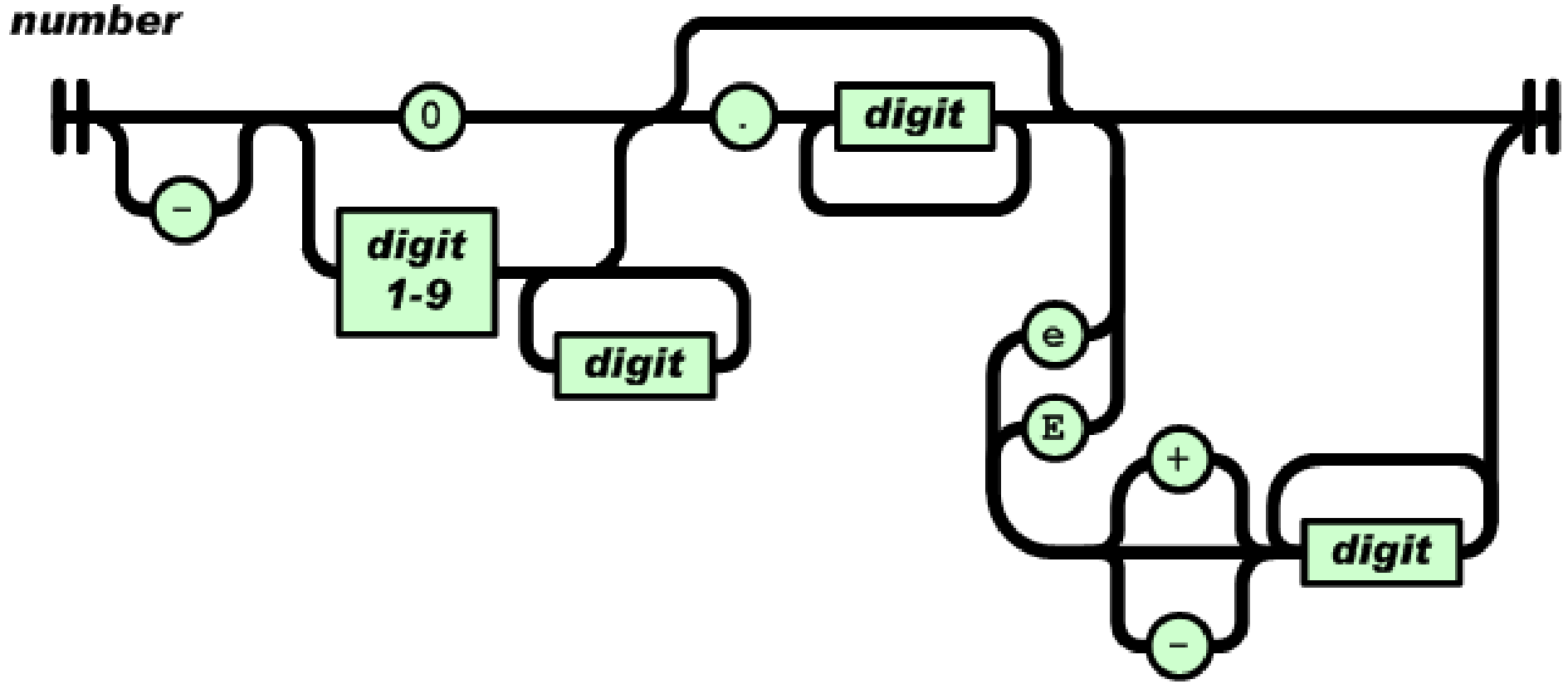
Valores JSON



String JSON


string
"
Any UNICODE character except
" or \ or control character
"
\
"
quotation mark
\
reverse solidus
/
solidus
b
backspace
f
formfeed
n
newline
r
carriage return
t
horizontal tab
u
4 hexadecimal digits

Número JSON



Tipos de Dados

- JSON String
  - {“nome”: “José”}
- JSON Números
  - {“idade”: 18}
- JSON Objeto
  - { “pessoa”: {“nome”: “José”, “idade”: 30} }
- JSON Array
  - { “nomes”: [“Mônica”, “José”, “Pedro”] }
- JSON Booleano
  - {“estudante”: true }
- JSON Nulo
  - {“valor”: null }

JSON com JavaScript

- JSON usa a sintaxe JavaScript
- Exemplo de JavaScript:
  - var pessoa={nome: “José”, idade: 20} //codigo JavaScript
  - console.log(pessoa.nome); //escreve José”
  - console.log(pessoa[“nome”]); //escreve José

  - Observe que isso não é código JSON e sim código JavaScript
  - JSON é inspirado no JavaScript mas não é JavaScript.

JSON com JavaScript

- JSON usa a sintaxe JavaScript
- Exemplo de JSON com JavaScript:
  - var pessoa={“nome”:“José”, “idade”:20} //codigo JavaScript com a sintaxe do JSON
  - console.log(pessoa.nome); //escreve José”
  - console.log(pessoa[“nome”]); //escreve José

  - Observe que isso é código JavaScript utilizando a sintaxe do JSON, que continua sendo a sintaxe do JavaScript

Laço em um Objeto

- Para percorrer as propriedades de um objeto em JavaScript utilize o laço *for-in*
- Exemplo:

```
var pessoa={ “nome”: “José”, “idade”: 18 };
for ( chave in pessoa)
{
    console.log(chave+": "+pessoa[chave]);
}
```

Deletar propriedades

- Para deletar propriedades de um objeto utilize a palavra chave: *delete.*
- Exemplo

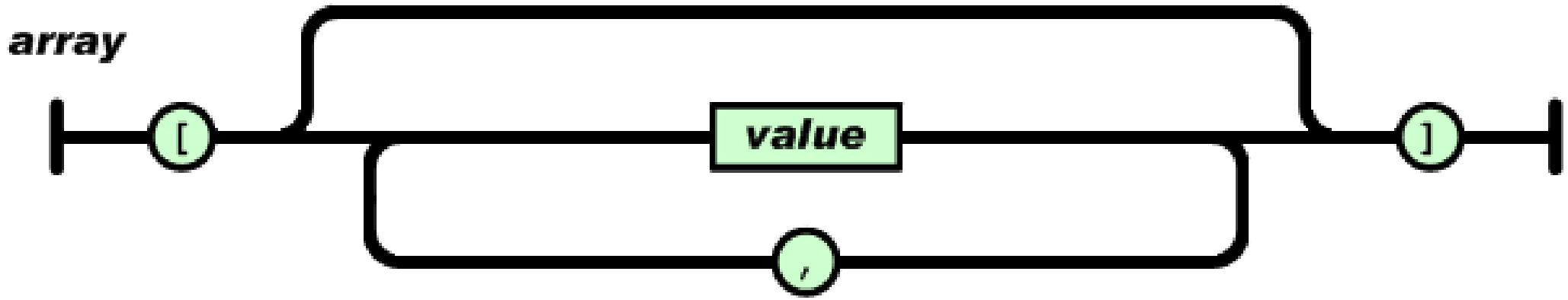
```
delete pessoa.idade;
```

Arrays com JSON

- Arrays no JSON são semelhantes aos arrays em JavaScript
- Exemplo:

  [8.9 , 10 , 9.5 , 7.5]
- Os valores de um array JSON podem ser: texto, número, objetos, array, booleano ou null.

Array JSON



Acessando Valores do Array

- Utiliza-se um indice númerico, iniciado em 0 (zero), entre os colchetes.
- Exemplo

```
var aluno ={ “nome”: “José”, “notas”:[8.9 , 10 , 9.5 , 7.5]};
console.log(alunos.notas[0]);
console.log(alunos.notas[1]);
console.log(alunos.notas[2]);
console.log(alunos.notas[3]);
```

Laço em um Array

- Laço for-in

```
for (i in aluno.notas)
    console.log(aluno.notas[i]);
```

- Laço for

```
for(i=0; i <aluno.notas.length;  i++)
    console.log(alunos.notas[i]);
```

Deletar Itens de um Array

- Para deletar itens de um array utilize a palavra chave: *delete.*

- Exemplo

```
delete aluno.notas[0];
```

Sintaxe Completa do JSON

JSON = null
or true or false
or JSONNumber
or JSONString
or JSONObject
or JSONArray

JSONNumber = - PositiveNumber
or PositiveNumber
PositiveNumber = DecimalNumber
or DecimalNumber . Digits
or DecimalNumber . Digits ExponentPart
or DecimalNumber ExponentPart

DecimalNumber = 0
or OneToNine Digits
ExponentPart = e Exponent
or E Exponent
Exponent = Digits
or + Digits
or - Digits
Digits = Digit
or Digits Digit
Digit = 0 through 9
OneToNine = 1 through 9

Sintaxe Completa do JSON

JSONString = ""

or " StringCharacters "

StringCharacters = StringCharacter

or StringCharacters StringCharacter

StringCharacter = any character

except " or \ or U+0000 through U+001F

or EscapeSequence

EscapeSequence = \" or \/ or \\ or \b or \f or \n or \r or \t

or \u HexDigit HexDigit HexDigit HexDigit

HexDigit = 0 through 9

or A through F

or a through f

JSONObject = { }

or { Members }

Members = JSONString : JSON

or Members , JSONString : JSON

JSONArray = [ ]

or [ArrayElements]

ArrayElements = JSON

or ArrayElements , JSON

Parse - Analisar

- Os dados recebidos de um servidor por um cliente Web estão sempre no formato de texto
- Para transformar os dados no formato texto em objetos JavaScript utiliza-se: JSON.parse( ).
- Exemplo:

```
var texto= '{ “nome”: “José”, “idade”: 20 }' ;
var pessoa= JSON.parse(texto);
console.log(pessoa.nome  + “ “ + pessoa.idade);
```

- A variável texto poderia vir do servidor em uma requisiçao realizada pelo Ajax.

Parâmetro: Função *reviver*

- A função JSON.parse( ) tem um segundo parâmetro opcional
- Este parâmetro é uma função que avalia cada propriedade antes de retornar seu valor
- Exemplo

```
var texto='{ “nome”: “josé”, “idade”: 18 }
var pessoa=JSON.parse(texto, function(chave, valor)
{
    if(chave == “nome”)
        return valor[0].toUpperCase( )+value.substring(1);
    else return valor;
});
```

Stringify - Converter em Texto

- Os dados enviados para o servidor por um cliente Web estão sempre no formato de texto
- Para transformar os objetos JavaScript em formato texto utiliza-se: JSON.stringify( ).
- Exemplo:

```
var pessoa= { nome: “José”, idade: 20 };
var texto= JSON.stringify(pessoa);
console.log(texto);
```

- A variável texto poderia ser enviada para o servidor em uma requisiçao realizada pelo Ajax.

Parâmetro: replacer

- A função JSON.stringify aceita um segundo parâmetro opcional, que pode ser uma função ou um array de strings e números.
- Array replacer
  - Um array que especifica quais propriedades serão serealizadas.
  - Exemplo

```
var pessoa={“nome”: “José”, “idade”:18, “telefone”: “99999999”};
var texto=JSON.stringify(pessoa,[“nome”,”telefone”]);
console.log(texto); //será mostrado apenas o nome e o telefone da
                    //pessoa a propriedade idade foi ignorada pelo
    //JSON.stringify
```

Parâmetro: replacer

- Função replacer pode manipular a forma como os dados são serealizados;
- Esta função recebe iterativamente a chave de todas as propriedades encontradas.
- Exemplo

```
var pessoa={“nome”: “José”, “idade”:18, “telefone”: “99999999”};
var texto=JSON.stringify(pessoa,
    function(chave,valor)
        if(chave== “idade”) return undefined;
        else return valor;
);
console.log(texto);
```

Parâmetro: space

- Terceiro parâmetro, opcional, da função JSON.stringify;
- Indica como será a identação do texto serealizado;
- Pode ser um número indicando quantos espaços cada subnivel tem ou pode ser uma string que será colocada como identação, por exemplo “\n”.
- Exemplo

```
JSON.stringify(pessoa,null,4);
```

ou

```
JSON.stringify(pessoa,null,“\t”);
```

Bibliografia

- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/JSON
- https://www.w3schools.com/js/js_json_intro.asp
- https://www.json.org/json-pt.html
- Smith, Ben. JSON Básico: Conheça o Formato de Dados Preferido da Web. São Paulo: Novatec, 2015.