FACAMP
FACULDADES DE CAMPINAS

Linguagens de Programação

Aula 4

Expressões regulares.

Formato JSON.

2º semestre de 2019
Prof José Martins Junior

Expressões Regulares

- É um método formal de especificar um padrão de texto
- Uma composição de símbolos, caracteres com funções especiais, que, agrupados entre si e com caracteres literais, formam uma sequência, uma expressão
  - Essa expressão é interpretada como uma regra, que indicará sucesso se uma entrada de dados qualquer “casar” com essa regra, ou seja, obedecer exatamente a todas as suas condições
- Outras definições
  - Uma maneira de procurar um trecho em posições específicas como no começo ou no fim de uma linha, ou palavra
  - Uma maneira de um programador especificar padrões complexos que podem ser procurados e casados em uma cadeia de caracteres
  - Uma construção que utiliza pequenas ferramentas feita para obter determinada sequência de caracteres de um texto

Metacaracteres

- Ferramentas básicas de uma expressão regular
  - São combinados para representar o padrão de busca desejado
  - Símbolos especiais

Metacaractere	Nome
`.`	Ponto
`[]`	Lista
`[^]`	Lista negada
`?`	Opcional
`*`	Asterisco
`+`	Mais
`{}`	Chaves

Metacaractere	Nome
`^`	Circunflexo
`$`	Cifrão
`\b`	Borda
`\`	Literal
`|`	Ou
`()`	Grupo
`\1`	Retrovisor

Metacaracteres representantes

- Ponto: .
  - Curinga que casa com uma única letra, número, caractere especial (@,#,$,%,...) , TAB, o próprio ponto, entre outros
  - Exemplos

`.ato`	`pato, rato, gato`
`n.o`	`não, nao, nAo`

- Lista: [ ]
  - Lista determina quais caracteres ou símbolos podem ser casados
  - Só pode ser casado um caractere por vez dentro de cada lista
  - Exemplos

`n[aã]o`	`nao, não`
`[pgr]ato`	`pato, gato, rato`
`12[:. ]45`	`12:45, 12.45, 12 45`
`<[BIP]>`	`<B>, <I>, <P>`

  - O ponto em uma lista NÃO é um metacaractere e sim normal

Metacaracteres representantes

- Listas com intervalos
  - Mais de um intervalo pode ser utilizado em uma lista
  - Exemplos

    `[0-9]` `[0123456789]`

    `[a-z]` `[abcdefghijklmnopqrstuvwxyz]`

    `[1-5A-F]` `[12345ABCDEF]`

    `[]a-f-]` `[]abcdef-]`
- Lista negada: [^]
  - Tudo que está na lista não casa com o padrão
  - Exemplos

    `[^0-9]` qualquer símbolo, exceto de 0 a 9

    `[:;,.!?][^ ]` casa com qualquer pontuação que não preceda um espaço em branco

Metacaracteres quantificadores

- Opcional: ?
  - Indica nenhuma ou uma ocorrência do padrão anterior

```
[pgr]atos?  pato, gato, rato, patos, gatos, ratos
casa[r!]?   casa, casar, casa!
```

- Asterisco: *
  - Indica nenhuma ou muitas ocorrências do padrão anterior

```
to*c        tc, toc, tooc, toooc, tooooc   ...
t[oc]*      t, to, tc, toc, tooc, tocc, tococ,...
```

- Mais: +
  - Indica uma ou mais ocorrências do padrão anterior

```
to+c        toc, tooc, toooc, tooooc, ...
t[oc]+      to, tc, toc, tooc, tocc, toccooc, ...
```

Metacaracteres quantificadores

- Chaves: { }
  - Especifica um limite mínimo/máximo de repetições do padrão anterior
  - Possibilidades

`{n,m}`	de n até m
`{n,}`	pelo menos n
`{n}`	exatamente n
`{0,}`	o mesmo que o "*"
`{1,}`	o mesmo que o "+"

  - Exemplos

`to{1,3}c`	toc, tooc, toooc
`to{2,}c`	tooc, toooc, tooooc, ...
`t[oc]{2,}`	tooc, toooc, tco, toc, tcooocoo ...

Metacaracteres Âncoras

- Circunflexo: ^
  - Indica que o padrão a seguir deve iniciar uma linha
    - `^[a-z]` linhas que começam com letras minúsculas
    - `^[0-9]` linhas que começam por números
    - `^[^0-9]` linhas que NÃO começam por números
- Cifrão: $
  - Indica que o padrão anterior deve ser considerado no fim da linha
    - `toc$` linhas que terminam com a palavra toc
    - `[0-9]$` linhas que terminam com número
    - `[:-@]$` linhas que terminam com os caracteres :, ;, <, =, >, ?, e @)
- Combinações úteis
    - `^$` linha em branco
    - `...$` últimos três caracteres da linha
    - `^.{15,30}$` linhas que contenham entre 15 e 30 caracteres

Metacaracteres Âncoras

- Borda: \b
  - Indica a borda de uma palavra (início ou fim)

`ana`	`ana, anamaria, mariana, luciana`
`\bana`	`ana, anamaria, analucia`
`ana\b`	`ana, mariana, luciana`
`\bana\b`	`ana`

  - Palavras: sequências de letras, números e o caractere “_”

Outros metacaracteres

- Literal: \
  - Quando colocado antes de um metacaractere o transforma em normal
- Ou alternativo: |
  - Indica um padrão com várias alternativas para uma palavra
    - A lista funciona como um tipo de operador “OU” somente para uma letra

    `Hello | Ola` casa com a palavra `Hello` ou com `Ola`
- Retrovisor: \1 ... \9
  - Retrovisor \1 é uma referência ao texto casado do primeiro grupo ...

```
(quero)-\1                    quero-quero
(lenta)(mente) é \2 \1        lentamente é mente lenta
```

Outros metacaracteres

- Grupo: ( )
  - Agrupamento de caracteres que serão tratados atomicamente

```
(oi!)+                  oi!, oi!oi!, oi!oi!oi!, ...
(\.[0-9]){3}            .3.4.5, .7.2.4, ...
(www\.)?google.com      www.google.com, google.com
(super|hiper)mercado    supermercado, hipermercado
(su|hi)permercado       supermercado, hipermercado
(mini|(su|hi)per)?mercado  supermercado,
                    hipermercado, mercado, minimercado
```

Metacaracteres barra-letra

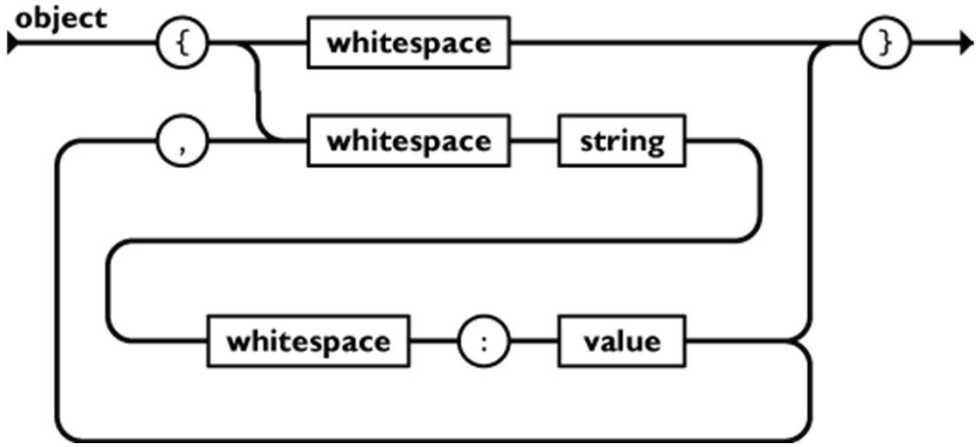
Metacaractere	Função
\d	Dígito
\D	Não Dígito
\w	Palavra (letra, dígito ou _)
\W	Não-palavra
\s	Branco
\S	Não-branco

JSON

- JavaScript Object Notation
  - Subconjunto da ECMA-262 third edition (1999) – European Computer Manufacturers Association
  - Notação aplicada em JavaScript para que seja possível a definição de objetos (estruturas) complexas
    - Mas atualmente, possui parsers em muitas linguagens
  - Sua forma de aplicação assemelha-se a um objeto abstraído do mundo real o qual trabalhamos em POO
    - Fornece a capacidade de atribuir métodos de execução (funções) que estarão diretamente ligadas ao objeto criado
  - Muito utilizado como formato de intercâmbio de dados para aplicações Web
    - Permite representar muitos tipos de dados e coleções
    - Estrutura lembra linguagens derivadas de C

Objetos em JSON

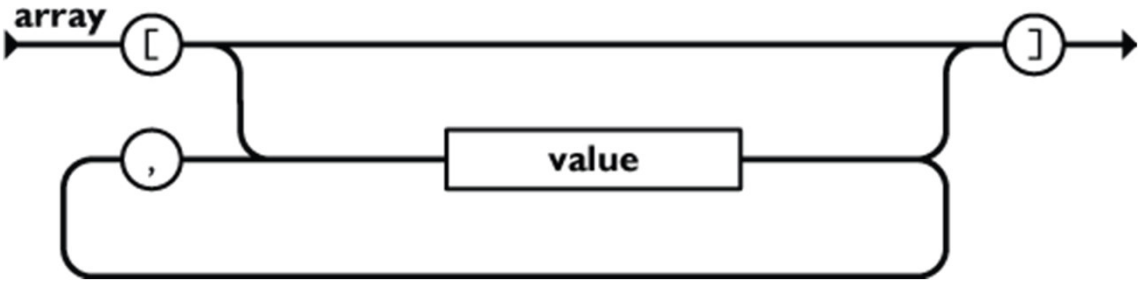
- Um objeto é um conjunto não ordenado de pares nome/valor
  - Começa com { e termina com }
  - Entre cada nome e valor coloca-se um :
  - Cada par é separado por uma ,




```
{
 "titulo": "JASON X",
 "resumo": "décima sequência da série Sexta-Feira 13",
 "ano": 2001,
 "genero": "terror"
}
```

Arrays em JSON

- Um array é uma coleção de valores ordenados
  - Começa com [e termina com]
  - Valores são separados por ,

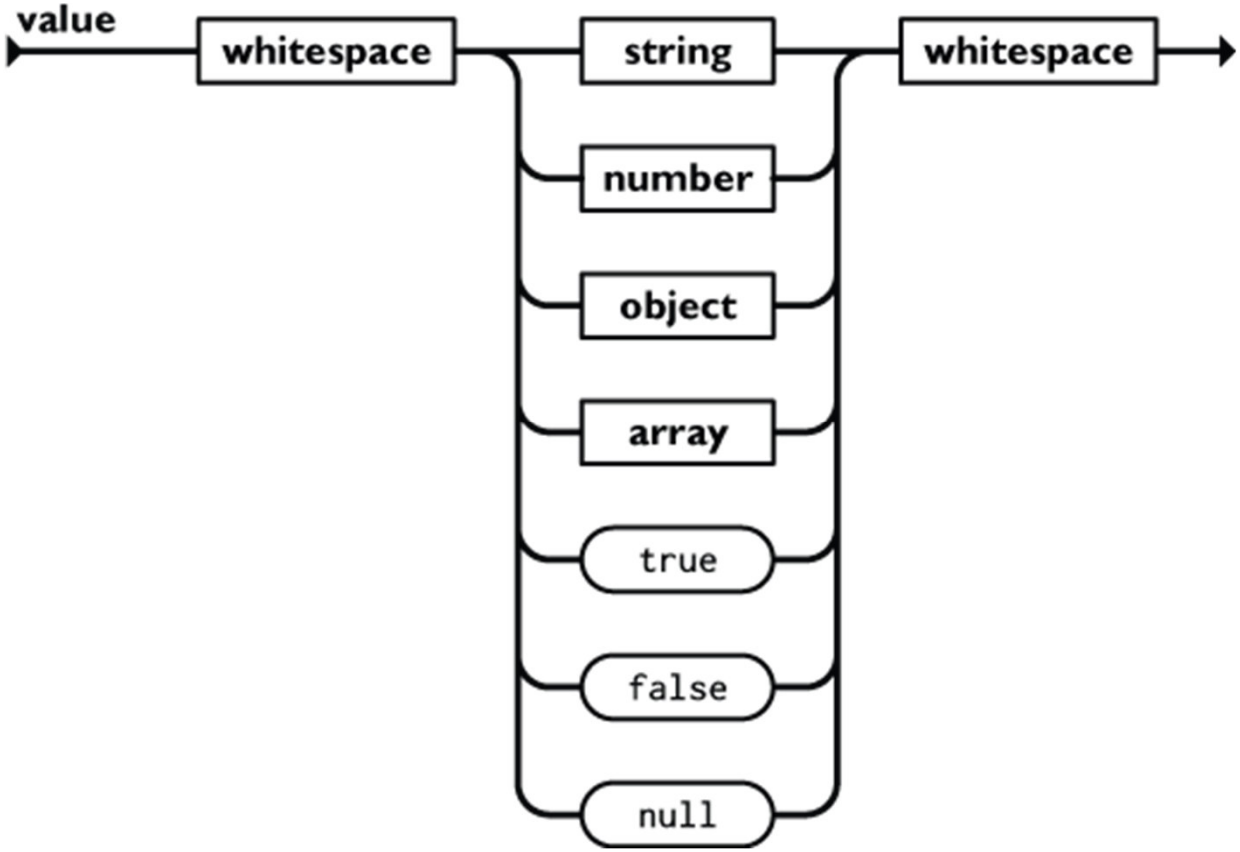



```
["RJ", "SP", "MG", "ES"]
```

```
[
  [1,5],
  [-1,9],
  [1000,0]
]
```

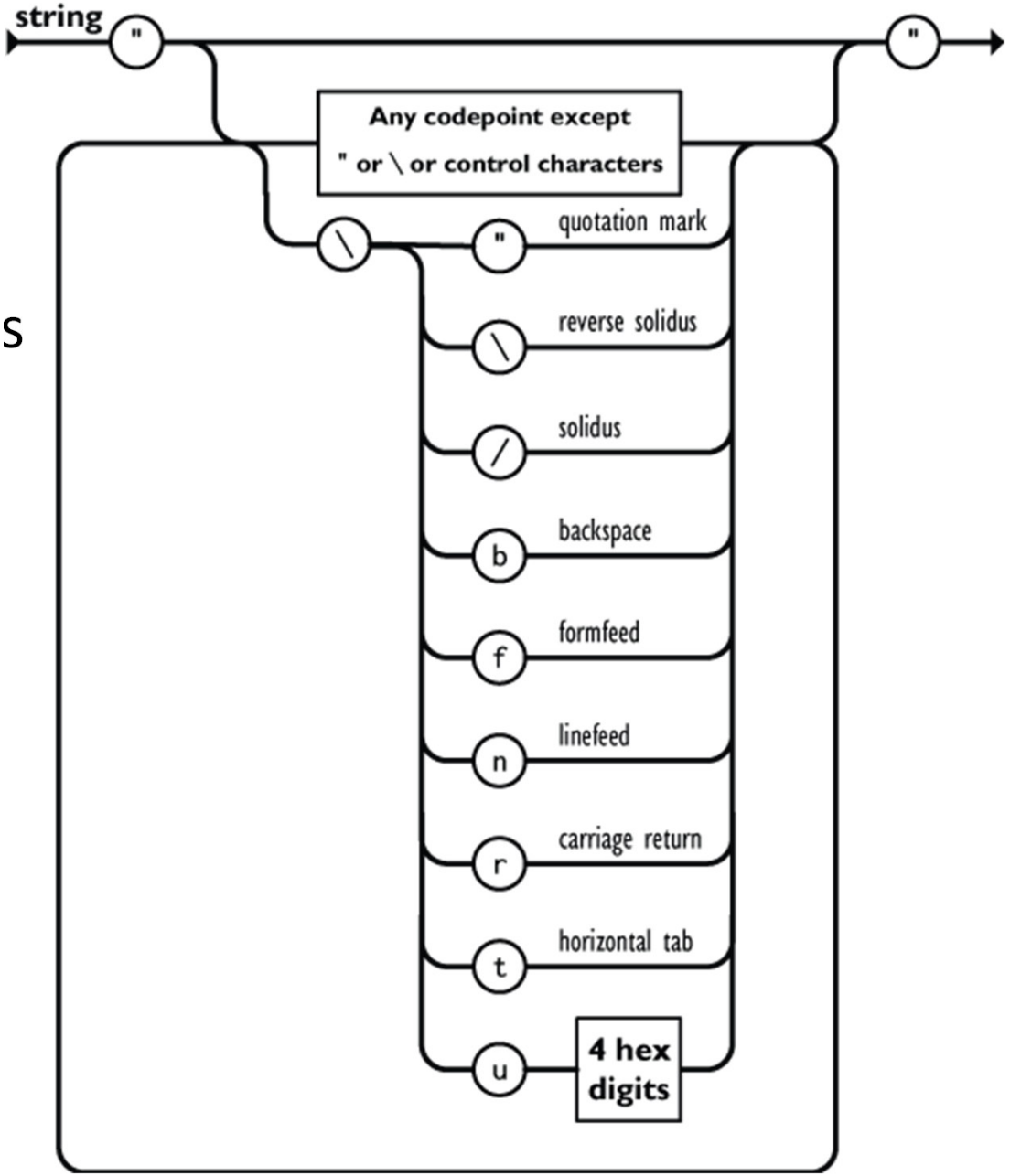
Valores em JSON

- Um valor pode ser uma string entre aspas, um número, true/false, null, um objeto ou um array
  - Tais estruturas podem ser aninhadas



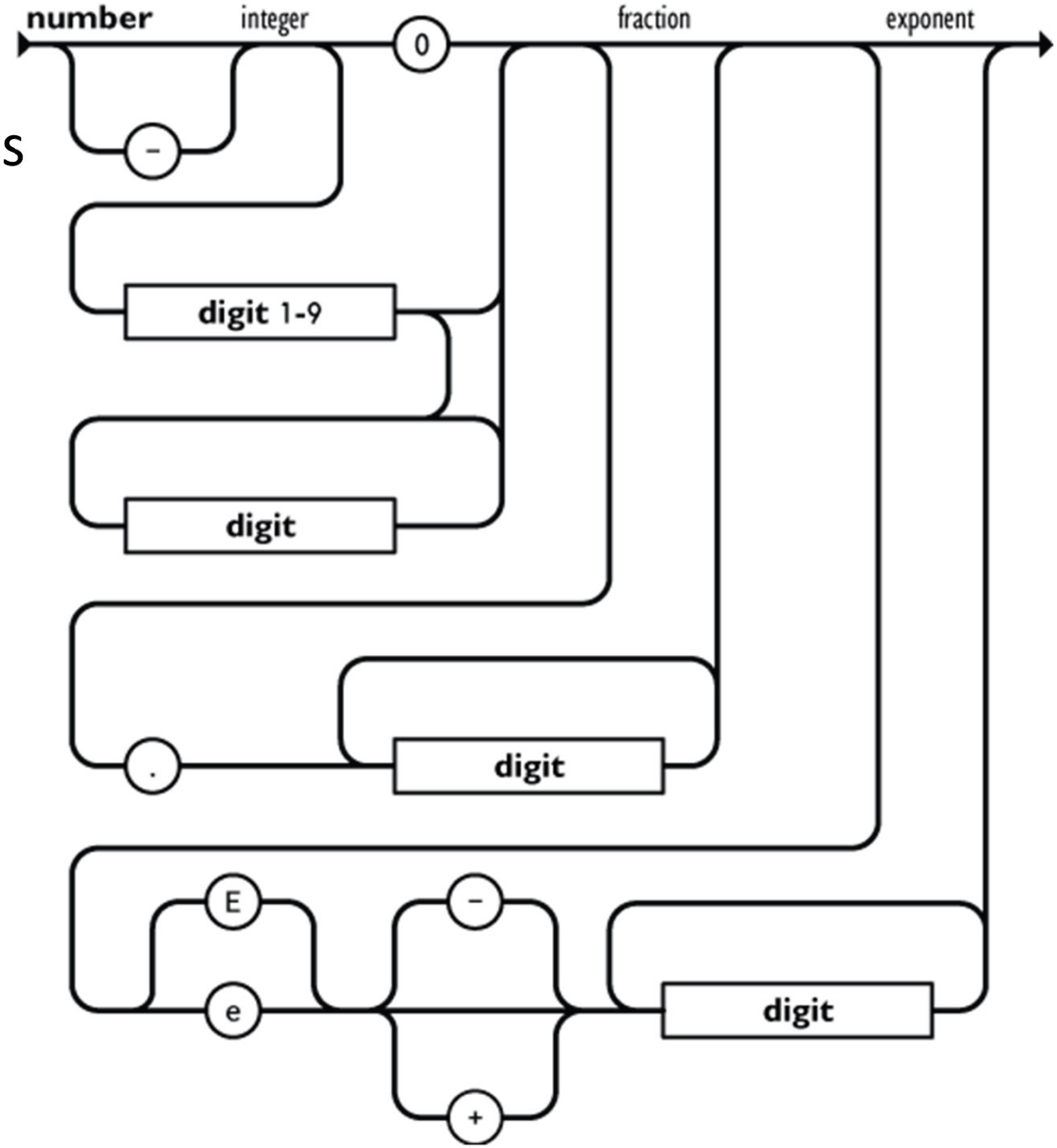
Strings em JSON

- Uma string é uma sequência de 0 ou mais caracteres Unicode colocados entre aspas
  - Permite representação com escape \
  - Um caractere é representado como uma string unitária
  - Como strings em C e Java



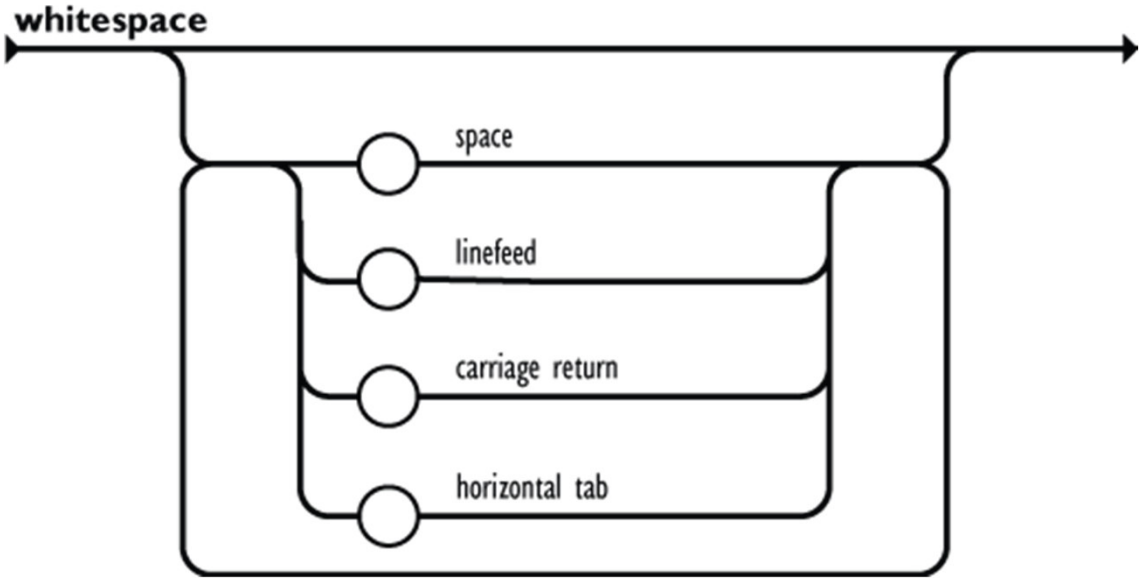
Números em JSON

- Representa valores numéricos
  - Muito parecido com C e Java
  - Exceto que não usa formatos octal e hexa



Espaços em JSON

- Espaços em branco podem ser colocado entre pares de tokens
  - Sua formalização não é complexa
  - Parecida com a utilizada em outras linguagens



Exemplo Javascript com JSON

teste.html

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <title>Testen JSON</title>
        <meta charset="utf-8">
        <script type="text/javascript" src="teste.js"></script>
    </head>
    <body>
        Teste JSON
    </body>
</html>
```

teste.js

```
var json = '{"nome": "Joao", "idade": 30, "cidade": "Campinas"}';
var obj = JSON.parse(json);
alert("nome: " + obj.nome
       + "\nidade: " + obj.idade
       + "\ncidade: " + obj.cidade);
```

Bibliografia

- SEBESTA, R. W. Conceitos de Linguagens de Programação. Porto Alegre: Bookman, 2011. 792p.
- DEVMEDIA. JSON Tutorial. Em: https://www.devmedia.com.br/json-tutorial/25275
- JSON.ORG. Introducing JSON. Em: https://www.json.org/