



# Vizualizace a serializace

Způsoby reprezentace strukturovaných dat

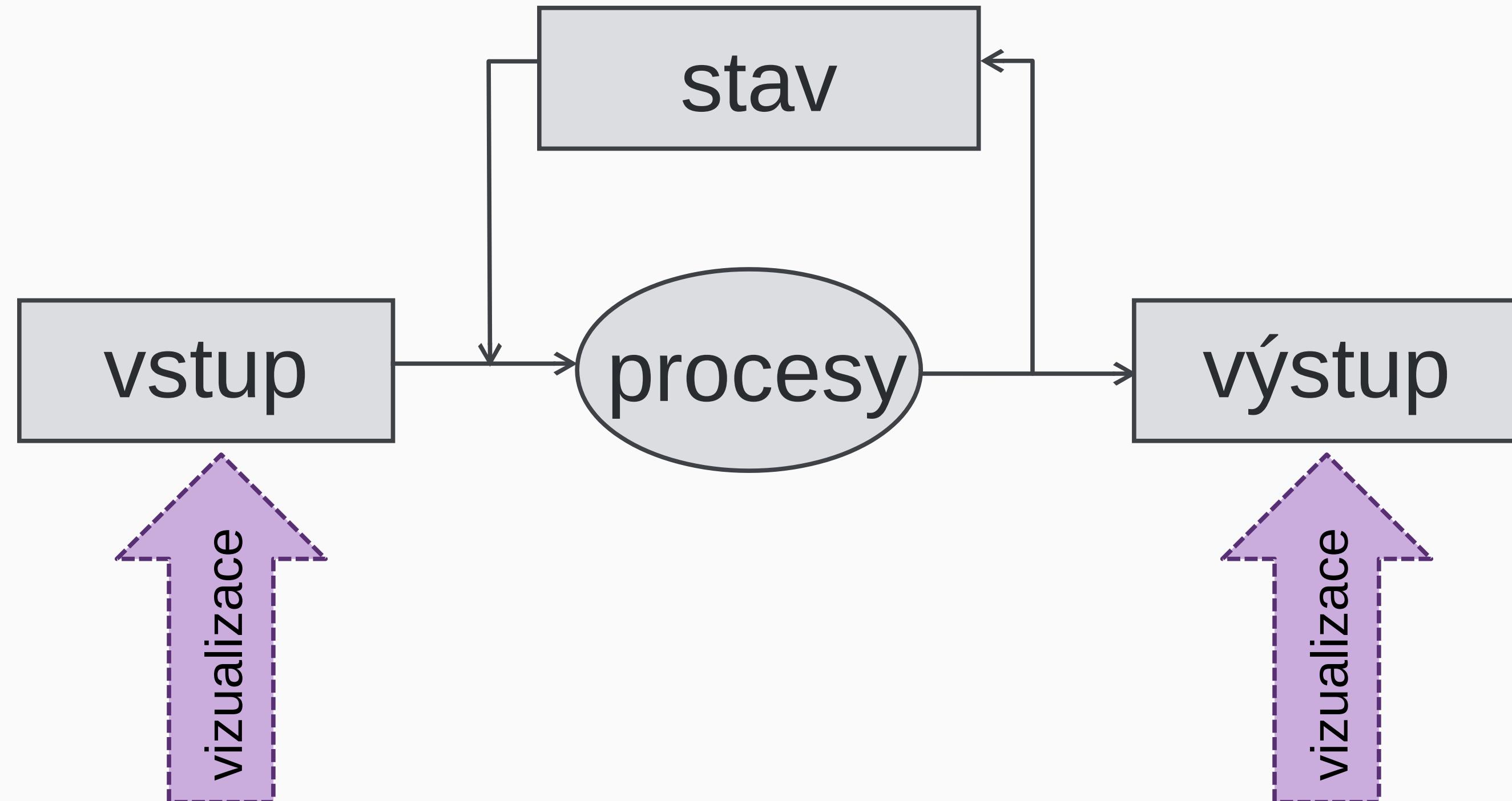
**doc. Ing. Radek Burget, Ph.D.**

[burgetr@fit.vutbr.cz](mailto:burgetr@fit.vutbr.cz)

# Vizualizace a serializace

- Data v IS – jednoduché hodnoty, struktury a kolekce reprezentované pomocí vhodných modelů
- **Vizualizace**
  - 2D reprezentace dat
  - HTML, CSS
  - Pokročilé: JavaScript, XSLT, Grafická prezentace
    - V další přednášce
- **Serializace**
  - 1D reprezentace – sériová forma (řetězec)
  - JSON, XML, další specializované formáty

# Vizualizace v informačním systému



# Zobrazení struktury

```
{  
    "a" : 5,  
    "b" : "obsah b" ,  
    "c" : true ,  
    "d" : 2.5  
}
```

zobrazí se

The diagram illustrates the mapping of a JSON structure to a form. On the left, a JSON object is shown:

```
{  
    "a" : 5,  
    "b" : "obsah b" ,  
    "c" : true ,  
    "d" : 2.5  
}
```

An arrow labeled "zobrazí se" (will be displayed) points from the JSON object to a form on the right. The form contains four fields:

a	5	b	obsah b
c	✓	d	2.5

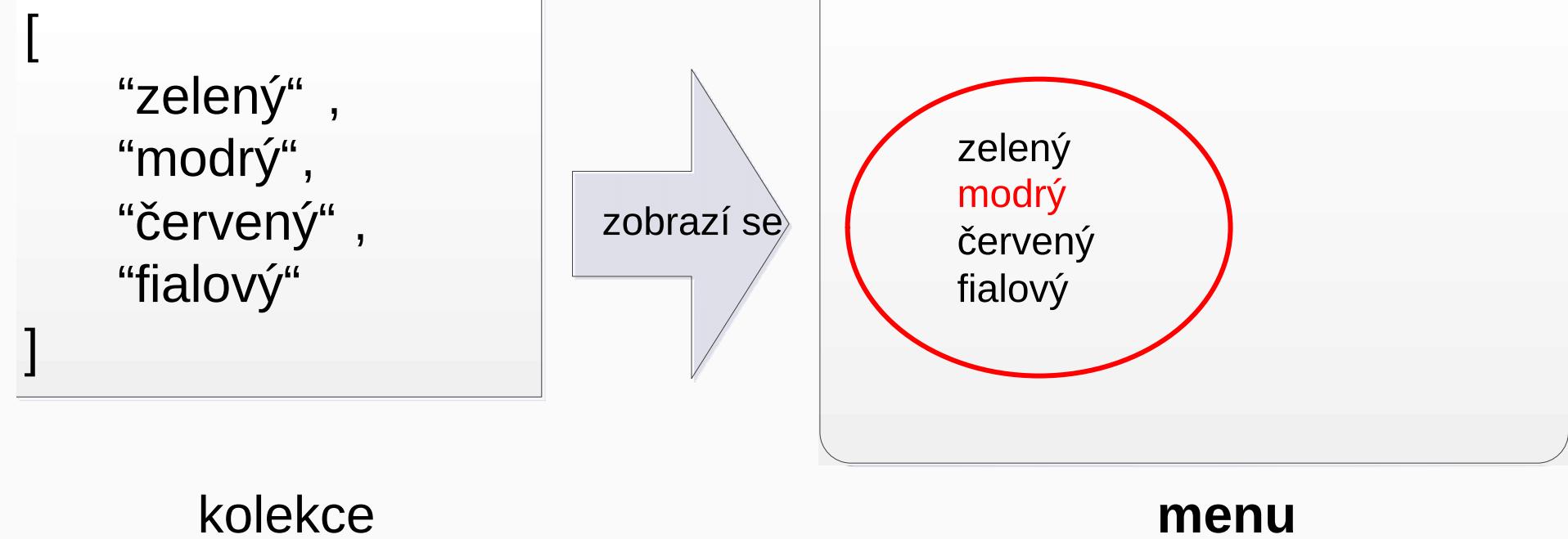
A red oval highlights fields a, b, and d. Field c is positioned below the oval.

struktura

formulář

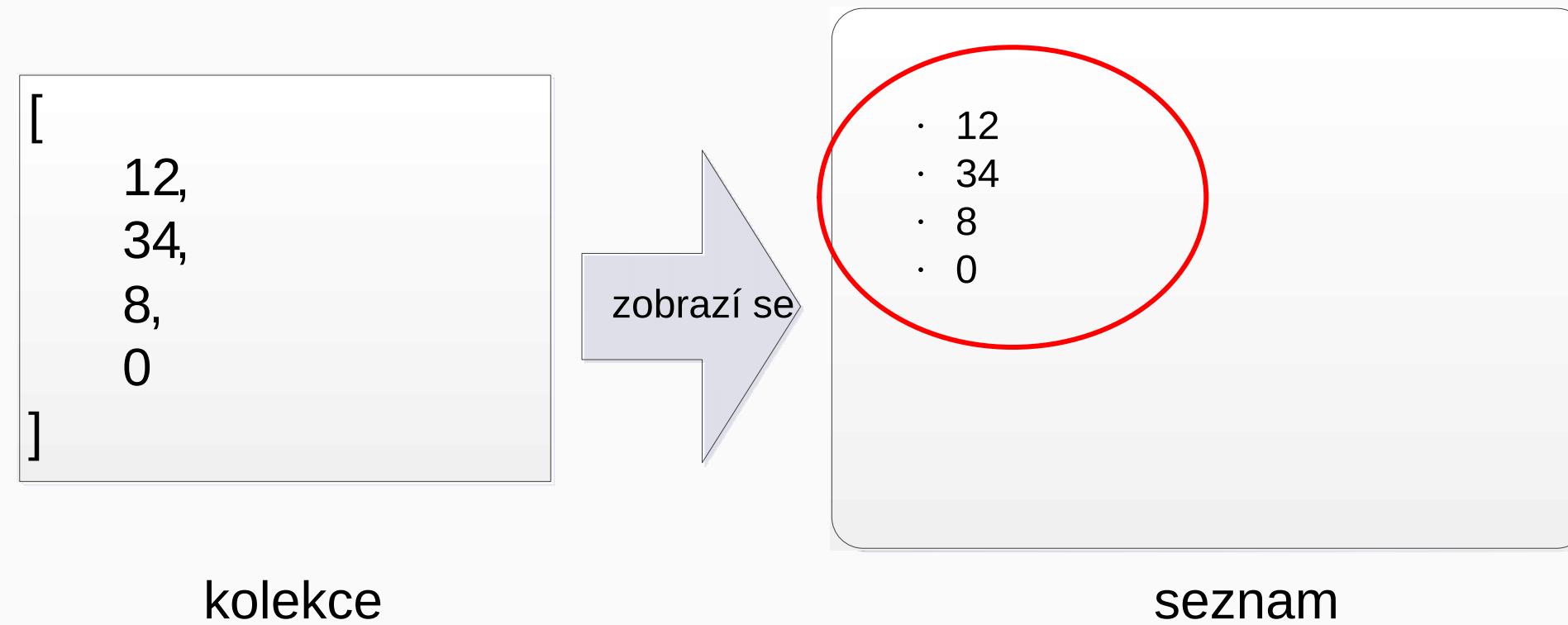
- Počet položek formuláře není závislý na hodnotě struktury

# Zobrazení kolekce jako menu



- Počet řádků seznamu se mění s hodnotou kolekce
- V HTML např. `<select>`, `<input type="radio">`

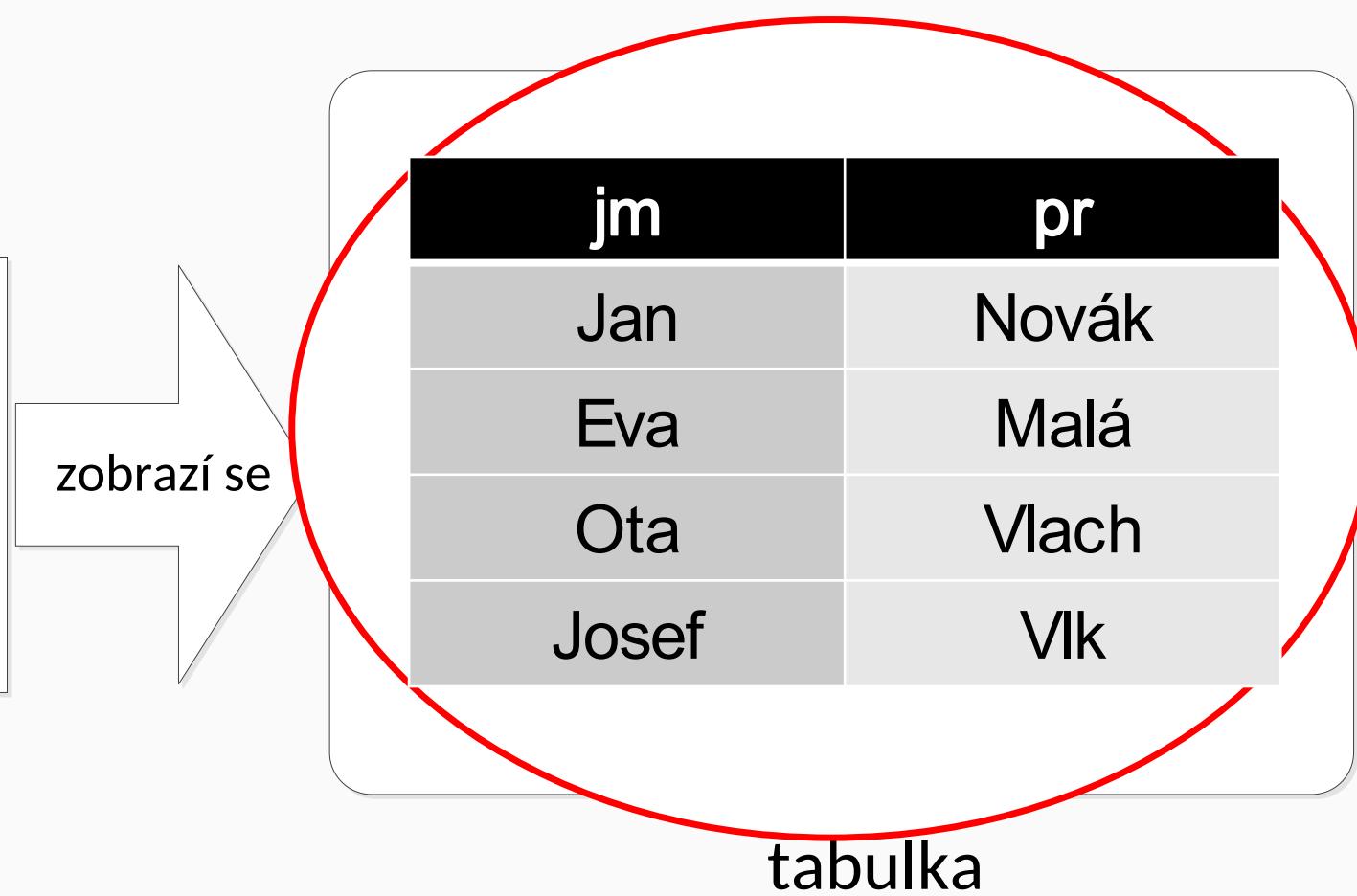
# Zobrazení kolekce jako seznamu



- počet řádků seznamu se mění s hodnotou kolekce
- v HTML např. `<ul>`, `<ol>`, `<li>`, ...

# Zobrazení kolekce jako tabulky

```
[  
  {"jm": "Jan", "pr": "Novák"},  
  {"jm": "Eva", "pr": "Malá"},  
  {"jm": "Ota", "pr": "Vlach"},  
  {"jm": "Josef", "pr": "Vlk"},  
]
```



kolekce (struktur)

tabulka

- počet řádků seznamu se mění s hodnotou kolekce
- nejčastější variantou je tabulka zobrazující kolekci struktur (relaci)

# Vizuální prezentace

- HTML kód generovaný by měl popisovat pouze obsah výsledného dokumentu a jeho strukturu
  - Tabulky, seznamy, formuláře, atd.
- Vizuální prezentace musí být definována zvlášť
  - Často spravována nezávisle
  - Více variant vzhledu (např. desktop, mobil, ...)
  - Chceme udržet serverový kód co nejjednoduší
- Jak definovat vizuální prezentaci odděleně?
  - Cascading Style Sheets – CSS

# Cascading Style Sheets

- Samostatná definice vzhledu – stylový předpis (*style sheet*)
  - Sada CSS pravidel
- Každé pravidlo definuje vlastnosti nějaké množiny HTML elementů
  - Selektor pravidla – vybírá množinu HTML elementů
  - Deklarace vlastností – definuje konkrétní vizuální vlastnosti
- Jedno pravidlo se může týkat většího množství elementů
- Vzhled elementů může být výsledkem aplikace většího množství pravidel

# Více o CSS

- Jednoduchý příklad
  - <https://github.com/DIFS-Teaching/basic-demos/blob/master/php-forms-pdo/style.css>
- Přednášky ITW
  - <https://www.fit.vutbr.cz/study/courses/ITW/private/prednasky/>

# CSS Frameworky

- Hotová CSS řešení pokrývající nejčastější prvky uživatelského rozhraní
- Využití nejčastěji pomocí přidání různých tříd HTML elementům (standardní atribut class)
- Vhodné pro rychlý návrh aplikace
  - Avšak značná část specifikace vzhledu se přesouvá do HTML, což úplně nechceme
  - Možnost kombinovat s vlastními CSS předpisy a problém alespoň částečně eliminovat
- Např. Bootstrap

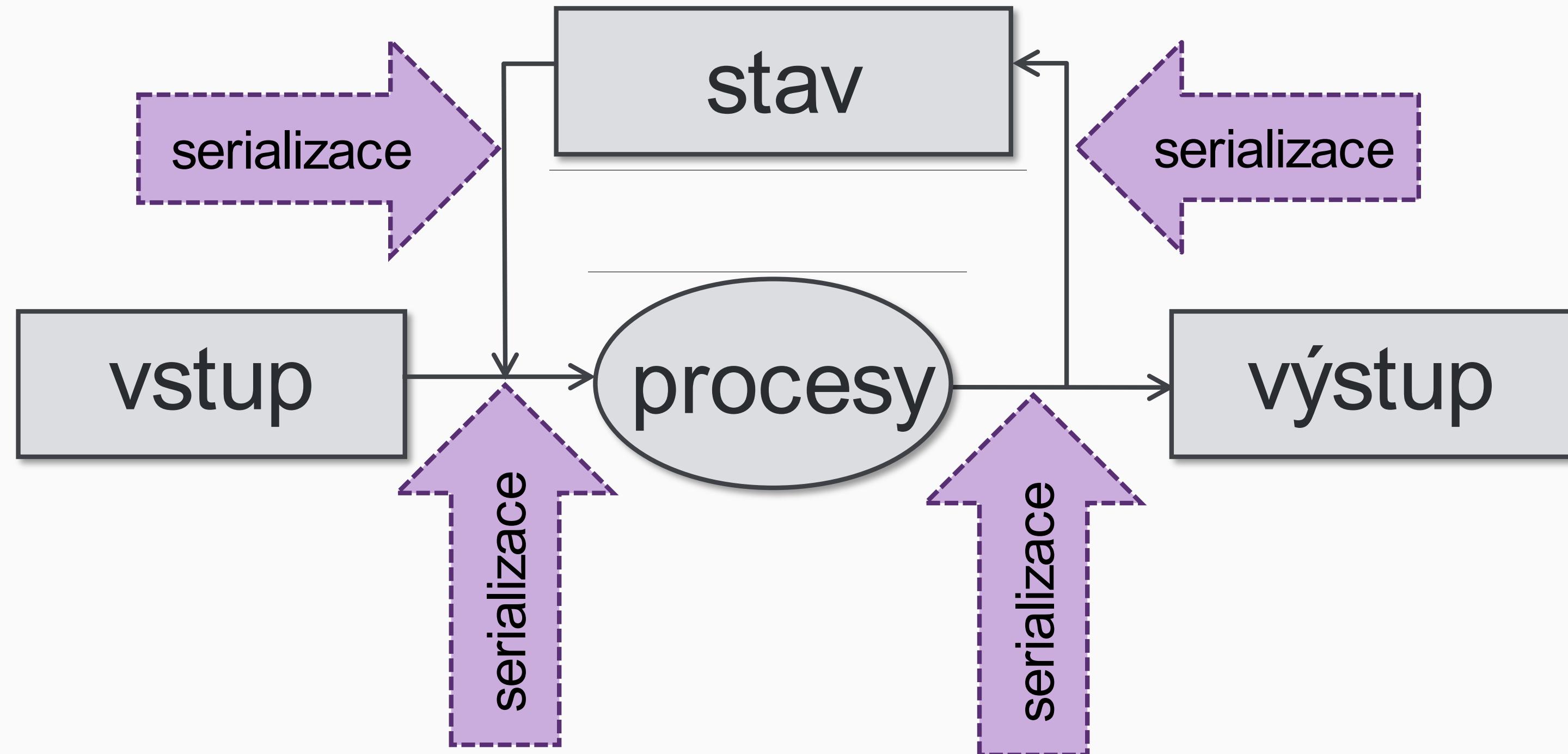
# Serializace

Řetězcová reprezentace strukturovaných dat

# Serializace

- *Serializace (marshalling)* je proces konvertování datových struktur nebo stavů objektů do formátu, který může být
  - uložen (např. textový soubor) nebo
  - přenášen síťovým přenosem
- *Deserializace (unmarshalling)* – rekonstrukce hodnoty na tentýž nebo jiný použitelný i netextový formát (vytvoření sémanticky ekvivalentního klonu původní datové struktury)
- Tento proces je složitý zejména v případě použití referujících hodnot (vztahů) a u objektů také u *metod*.

# Serializace v informačním systému



# Syntaxe serializačních formátů

- Formálněji řečeno je serializace vyjádřením hodnoty struktury v nějakém **formálním jazyce**
- Hodnoty struktury a kolekce mají tvar
  - hodnota typu struktura = uspořádaná n-tice  $\$(a_1, a_2, \dots, a_n)\$$
  - hodnota typu kolekce = uspořádaná množina  $\$\{b_1, b_2, \dots, b_n\}\$$
- Na úrovni hodnoty = výskytu jsou stejné
- Jde o závorkované zanořené věty tvaru  $\$a^n \dots b^n\$$ , formální jazyk tudíž musí být minimálně **bezkontextový**

# Syntaxe serializačních formátů

- Vyhovuje libovolný bezkontextový jazyk mající možnost zápisu literálů strukturovaných hodnot (např. JavaScript – JSON)
- případně speciální bezkontextové jazyky (např. značkovací XML)

# Serializační formáty

- Často proprietární formáty jednotlivých aplikací
  - Závislost na architektuře (např. dump paměti)
  - Nemožnost sdílet existující nástroje (parser, ...)
- 1987 zavedla firma Sun Microsystems *External Data Representation* (XDR)
- 1990 XML
- 2001 JSON
- 2001 YAML
- mnoho dalších

- eXtensible Markup Language
- Původně značkovací jazyk primárně pro tvorbu dokumentů
  - Vývoj ze standardu SGML
- Snadno rozšířitelný pro další aplikace
  - Možnost formální definice vlastního schématu, validace, transformací, objektová reprezentace, ...
- Proto ve své době první kandidát pro reprezentaci dat
  - Avšak v mnoha situacích poměrně těžkopádný

# JSON

- **Java Script Object Notation**
- Navržen jako jazyk pro přenos dat mezi serverem a prohlížečem
- Využívá syntaxi pro zápis literálů v JavaScript
  - Snadné zpracování zejména v JS aplikacích
- Avšak je nezávislý na platformě
  - Podpora na všech hlavních serverových i klientských platformách

# YAML

- YAML - Ain't Markup Language
- YAML obsahuje rysy pro efektivní serializaci uživatelsky příjemnou a potenciálně kompaktnější.
- Jsou to:
  - podpora ne-hierarchických struktur
  - strukturování dat indentací-zanořením (nebere tabelátory)

# YAML příklad

```
invoice: 34843
date   : 2001-01-23
bill-to: &id001
  given  : Chris
  family : Dumars
  address:
    lines: 458 Walkman Dr.
    city   : Royal Oak
    state  : MI
    postal : 48046
ship-to: *id001
```

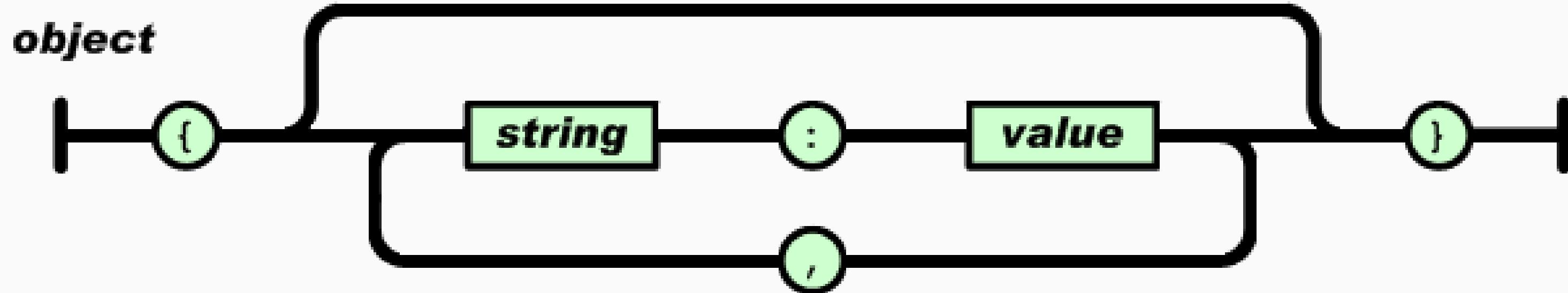
# YAML příklad

```
product:
  - sku          : BL394D
    quantity    : 4
    description : Basketball
    price       : 450.00
  - sku          : BL4438H
    quantity    : 1
    description : Super Hoop
    price       : 2392.00
tax  : 251.42
total: 4443.52
comments: >
```

# JSON

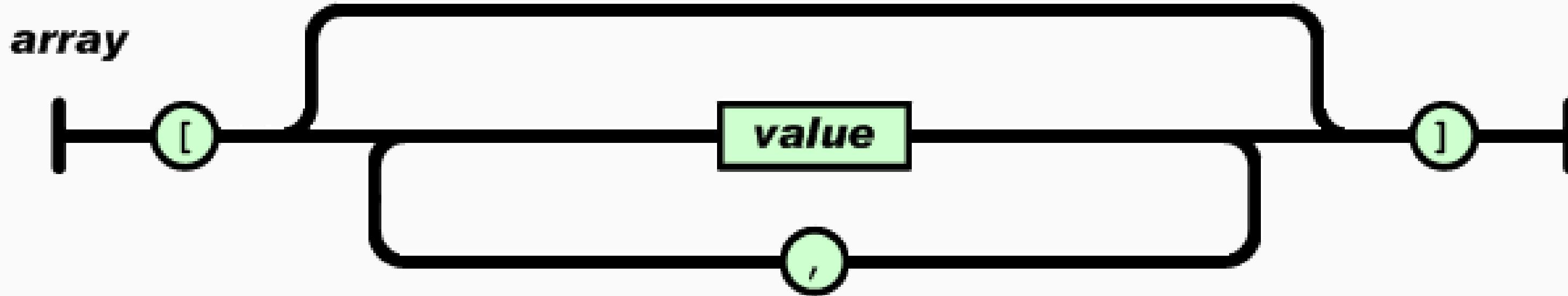
- **JSON** (JavaScript Object Notation)
- Je založen na podmnožině programovacího jazyka JavaScript, Standard ECMA-262 3rd Edition - December 1999.
- Zápis v JSON je literálem v jazyce JavaScript. Není tudíž třeba speciálního analyzátoru
  - Všechny prohlížeče a klientské JavaScripty implicitně analyzují JSON
  - Ale v praxi se analyzátor používá i v JavaScriptu (bezpečnost)

# Struktura v JSON



Je serializací *neuspořádané struktury*

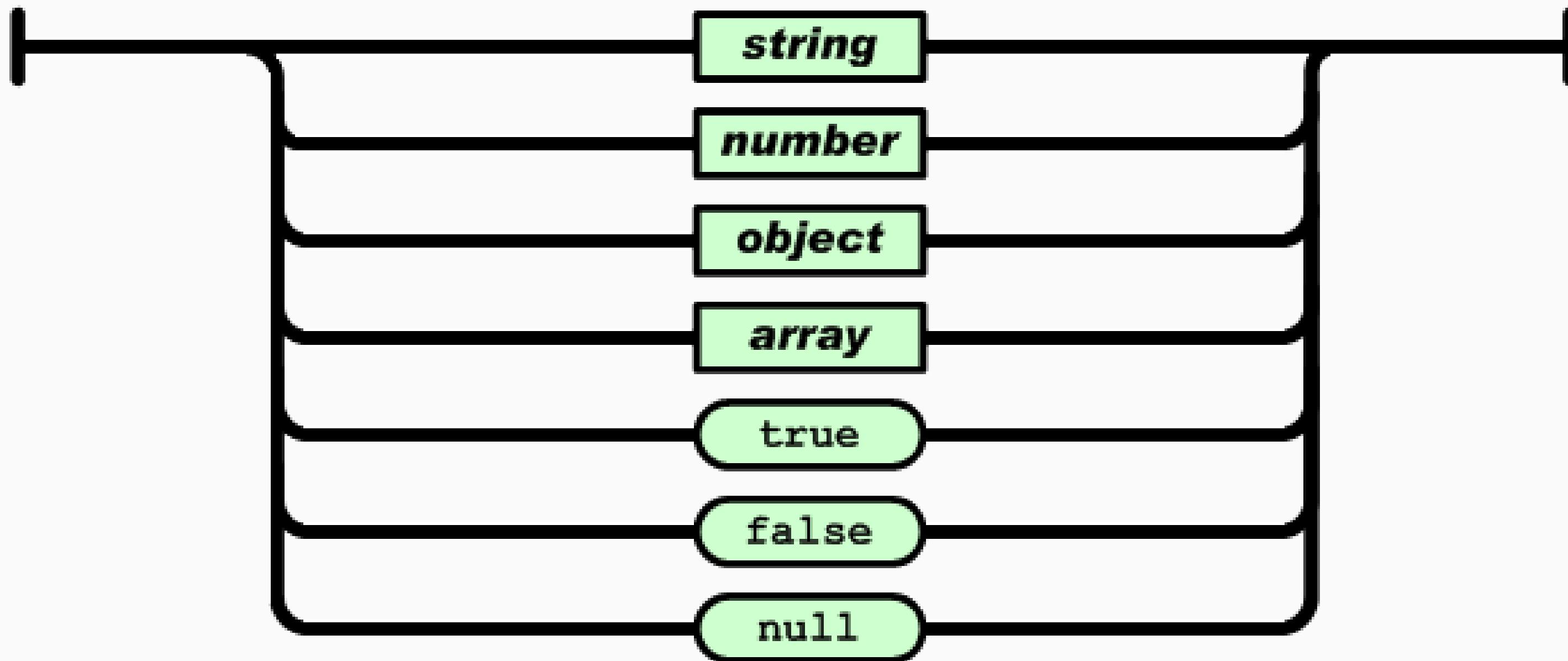
# Kolekce v JSON



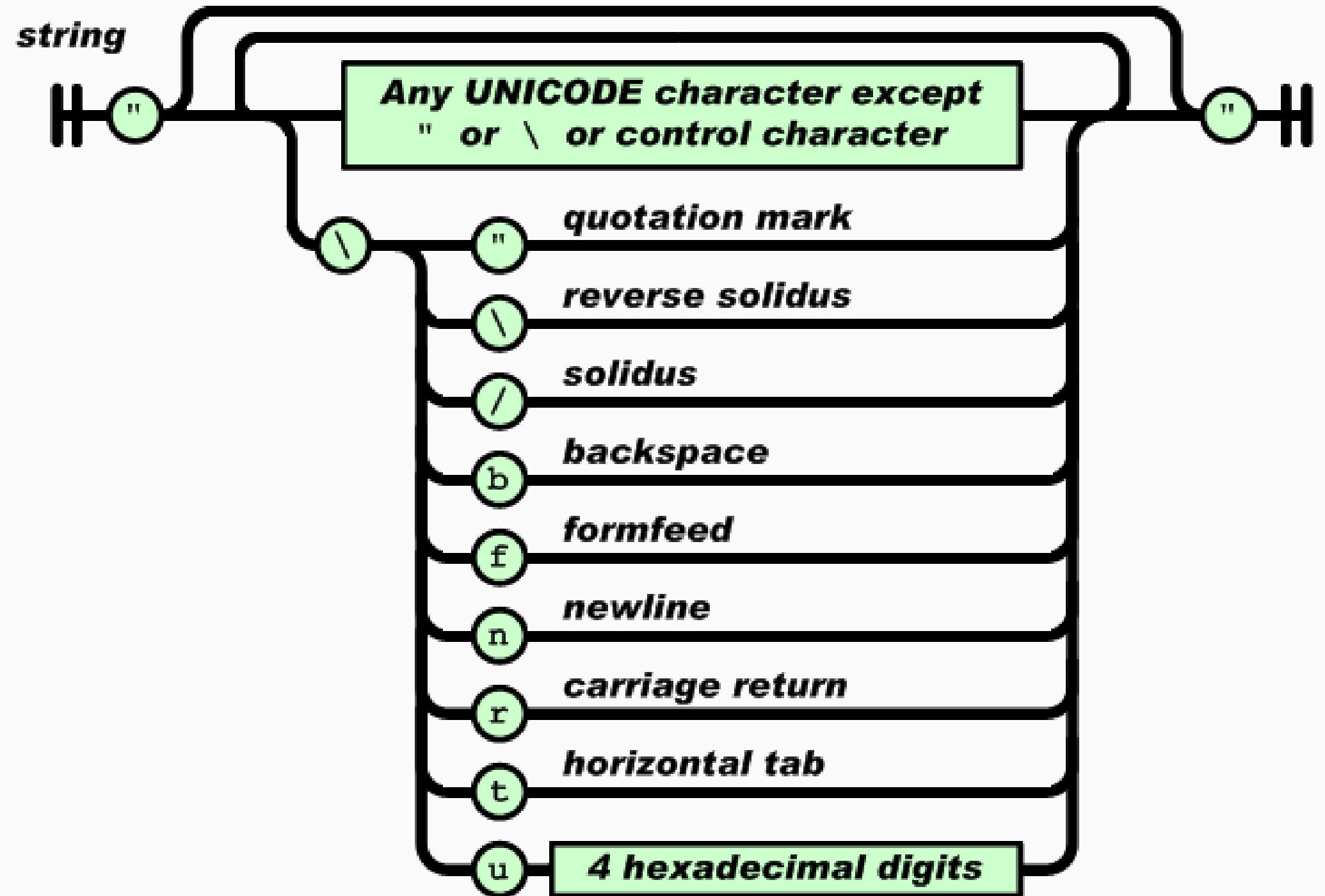
Je serializací ***uspořádané kolekce – seznamu***

# Základní typy v JSON

**value**

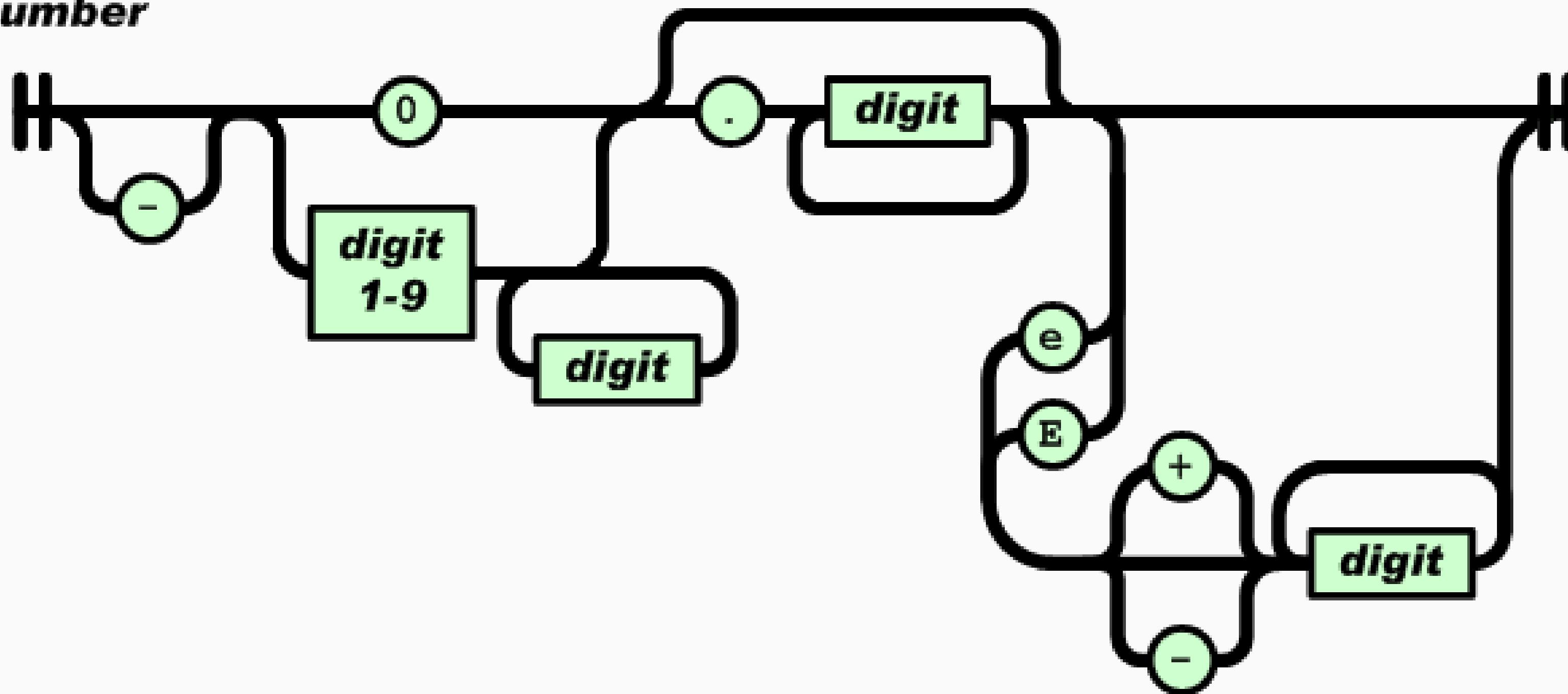


# Textové řetězce v JSON



# Čísla v JSON

*number*



# Příklad JSON

```
{  
    "firstName": "John",  
    "lastName": "Smith",  
    "address": {  
        "streetAddress": "21 2. street",  
        "city": "New York",  
        "state": "NY",  
        "postalCode": 10021  
    },  
    "phoneNumbers": [  
        "212 555-1234",  
        "646 555-4567"  
    ]  
}
```

- V PHP funkce `json_encode($pole)` a `json_decode($retezec)`.

# Omezené serializační formáty

Pro speciální použití

# Omezené serializační formáty

- Omezují hloubku zanoření a typy položek strukturovaných hodnot
- Použití například pro předávání obsahu formulářů v HTML protokolem HTTP

# application/x-www-form-urlencoded

- Serializace struktury s položkami textového a booleovského typu
- Mezery se převádějí na plus a nealfanumerické znaky na %*hh*, kde *hh* je hexadecimální kód znaku.
- Hodnoty položek jsou odděleny &. Bývá položka pro tlačítko Submit a jeho jméno, pro indikaci, které tlačítko užil uživatel pro odeslání.

# application/x-www-form-urlencoded

- Pokud uživatel vyplní formulář se vstupními polí **Jmeno** a **Chut**:

```
Jmeno=Hru%9Aka&Chut=Vanilkova
```

- Používá se pro přenos hodnot z prvku **<form>** jazyka HTML
- Metody GET i POST

# multipart/form-data

- Serializace struktury s položkami textového a booleovského typu
- pouze pro metodu POST. Každé odeslané pole zde má svůj vlastní oddíl, oddělený oddělovačem (boundary string) zvoleným tak, aby nekolidoval s odesílanými daty.
- Neprovádí se žádné kódování hodnot. Je tedy možné přenášet **binární data**.
- Používá se pro přenos hodnot z prvku `<form>` jazyka HTML
  - Nutno specifikovat pomocí atributu enctype
  - Potřebné pro přenos souborů `<input type="file">`

# Příklad formuláře s MIME typem

```
<form action="upload.php" method="post"
      enctype="multipart/form-data">
    Select file to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>
```

- PHP automaticky převeze uploadovaný soubor a uloží do dočasného souboru na serveru
- Údaje jsou v `$_FILES['fileToUpload']['...']`
  - name – jméno originálního souboru
  - tmp\_name – jméno dočasného souboru
  - type – MIME typ souboru

# Příklad serializovaného formuláře

```
-----7d627e30307c4
Content-Disposition: form-data; name="Jmeno"

hruška
-----7d627e30307c4
Content-Disposition: form-data; name="Chut"

Vanilkova
-----7d627e30307c4
Content-Disposition: form-data; name="thefile"; filename="a.jpg"
Content-Type: application/octet-stream
```

# XML – Extensible Markup Language

- Serializační formát vycházející ze starších značkovacích jazyků
- Orientovaný na dokumenty
  - Odlišný způsob zápisu
- Celá řada souvisejících technologií pro zpracování XML

# XML dokumenty

- Syntakticky obdobné jako v HTML
- Hierarchické zanořování XML prvků (**elementů**)
  - Každý element má *jméno* a případně *atributy*
  - Jeden **kořenový element**
- Syntaktické rozdíly oproti HTML
  - XML ne definuje jména ani význam elementů a atributů – vše je dáno konkrétní aplikací
  - Proto všechny značky jsou párové (parser nezná jejich význam)
  - Hodnoty všech atributů musí být v uvozovkách

# Prázdný prvek

- Je-li element prázdný, tj. nemá žádný obsah (ale může mít atributy), bude mít tvar

```
<jméno  případné atributy />
```

- Což je ekvivalentní

```
<jméno  případné atributy></jméno>
```

# Příklad XML

```
{  
  "firstName": "John",  
  "lastName": "Smith",  
  "address": {  
    "streetAddress": "21 2. street",  
    "city": "New York",  
    "state": "NY",  
    "postalCode": 10021  
  },  
  "phoneNumbers": [  
    "212 555-1234",  
    "646 555-4567"
```

```
1 <?xml version="1.0" encoding="UTF-  
2 <person>  
3   <address>  
4     <city>New York</city>  
5     <postalCode>10021</postalCode>  
6     <state>NY</state>  
7     <streetAddress>21 2. street</streetAddress>  
8   </address>  
9   <firstName>John</firstName>  
10  <lastName>Smith</lastName>  
11  <phoneNumbers>  
12    <item>212 555-1234</item>
```

- Záhlaví XML není nutné pro kódování UTF-8
- Navíc kořenový element a elementy pro prvky kolekce

# Příklad XML – Alternativně

```
{  
  "firstName": "John",  
  "lastName": "Smith",  
  "address": {  
    "streetAddress": "21 2. street",  
    "city": "New York",  
    "state": "NY",  
    "postalCode": 10021  
  },  
  "phoneNumbers": [  
    "212 555-1234",  
    "646 555-4567"]
```

```
1 <?xml version="1.0" encoding="UTF-  
2 <person>  
3   <address  
4     city="New York"  
5     postalCode="10021"  
6     state="NY"  
7     streetAddress="21 2. street"  
8   <firstName>John</firstName>  
9   <lastName>Smith</lastName>  
10  <phoneNumbers>  
11    <item>212 555-1234</item>  
12    <item>646 555-4567</item>
```

- Atributy místo elementů
- Prázdný element `<address />`

# JSON nebo XML?

- **JSON**
  - Většinou komunikace specifická pro aplikaci
  - Ad-hoc formát
  - Snadné čtení i zápis, mapování na objekty jazyka
- **XML**
  - Dokumenty spíše orientované na text
  - Často veřejně sdílená data
  - Celý ekosystém navazujících technologií – *v dalších přednáškách*
    - Adresování, transformace, vizualizace, ...
- Často je volba zbytečně subjektivní :-)

A to je vše!  
Dotazy?