

Fakultet tehničkih nauka, DRA, Novi Sad

Predmet:
Organizacija podataka

dr Vladimir Ivančević

Nikola Todorović

Sanja Špica

Vladimir Jovanović

JavaScript Object Notation JSON

JSON

- JavaScript Object Notation (JSON)
 - format za razmenu i čuvanje podataka
 - nezavisan od programskih jezika
 - minimalna količina dodatnih informacija
 - *minimum overhead*
 - laka računarska obrada podataka u JSON formatu
 - ljudi mogu lako čitati podatke sačuvane u ovom formatu

JSON

- Zasnovan na podskupu JavaScript programskog jezika
 - Standard ECMA-262 3rd Edition – decembar 1999.
- JSON specifikacija
 - <http://www.json.org/>
 - spisak biblioteka za veliki broj programskih jezika
- JSON validator
 - <http://jsonlint.com/>

JSON

- Upotreba
 - komunikacija na Internetu
 - web servisi (REST ...)
 - JSON RPC
 - podaci sa socijalnih mreža
 - *Facebook, Twitter, LinkedIn ...*
 - NoSQL baze podataka
 - konfiguracioni fajlovi

JSON

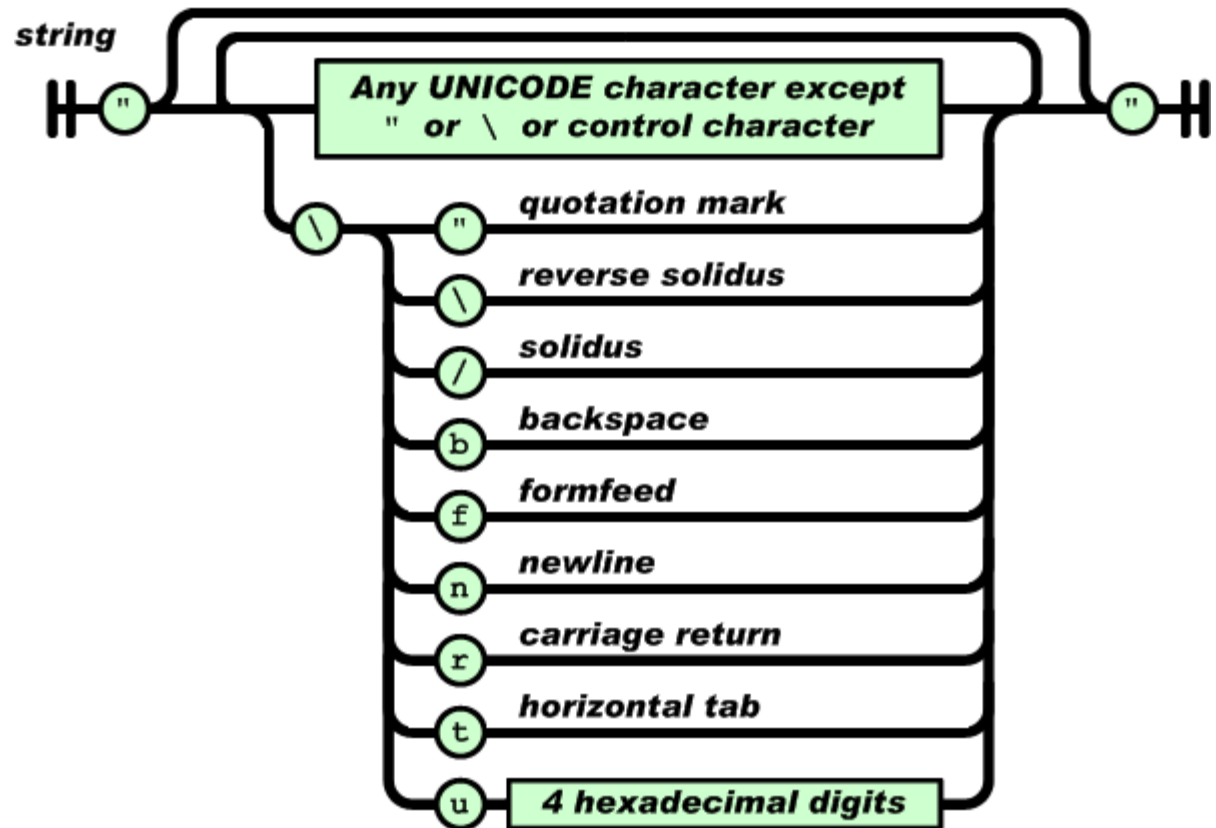
- Sintaksa
 - osnovni element
 - par naziv/vrednost
 - JSON je sgrađen na dve strukture
 - objekat
 - niz

JSON

- Par naziv/vrednost
 - *naziv : vrednost*
 - *naziv*
 - naziv atributa
 - „opis“ vrednosti koja sledi
 - **uvek je tipa String**
 - *vrednost*
 - vrednost atributa
 - **može biti String, broj, *true*, *false*, *null*, objekat ili kolekcija**

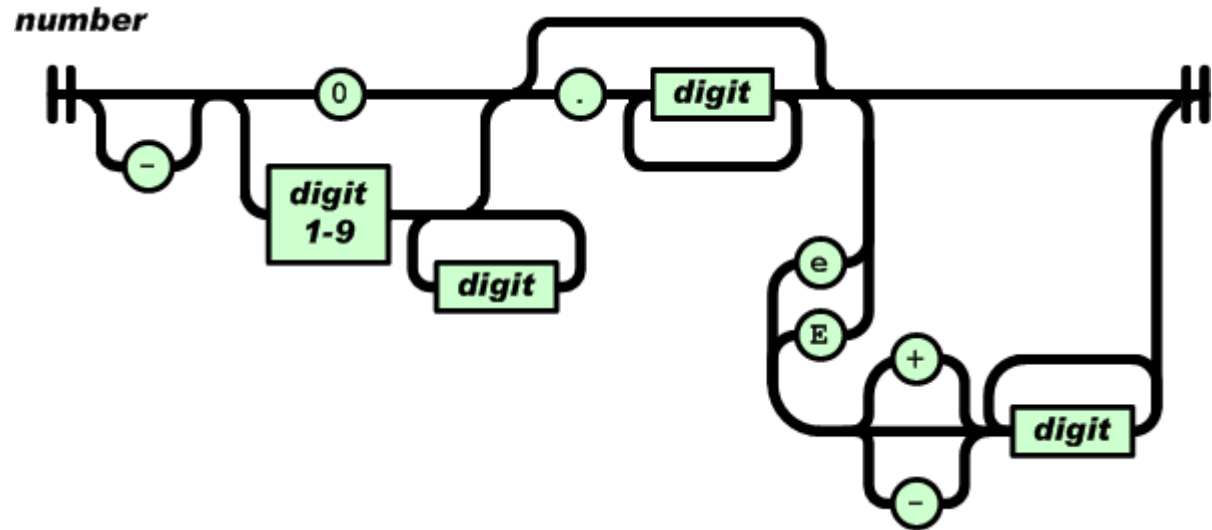
JSON

- String



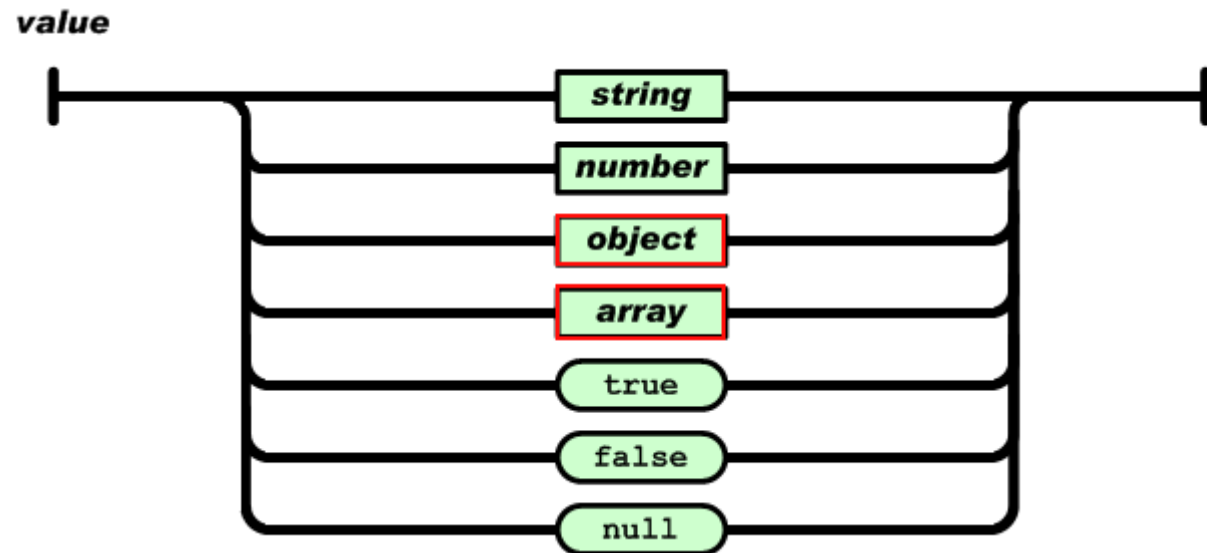
JSON

- Broj



JSON

- Vrednost



JSON

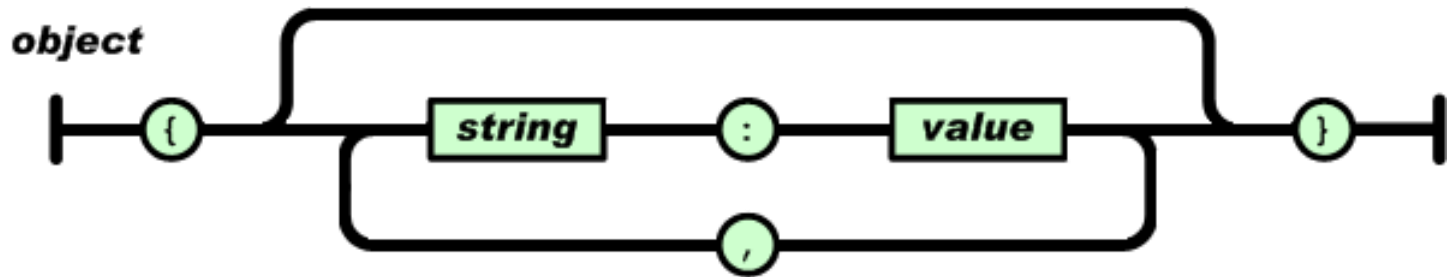
- Par naziv/vrednost
 - osnovni primeri:
 - “NazivKnjizare” : “Moja knjizara”
 - “BrojKnjiga” : 24
 - “BrojKnjiga” : “24”
 - “ClanLancaKnjizara” : true
 - “Direktor” : null

JSON

- **Objekat**
 - **neuređeni skup parova naziv/vrednost**
 - međusobno razdvojeni znakom “,”
 - počinje sa znakom “{”
 - završava se znakom “}”
 - može biti vrednost u paru naziv/vrednost
 - moguće ugneždavanje objekata
 - imenovani objekat
- **JSON objekat (datoteka)**
 - neimenovani objekat

JSON

- Objekat



JSON

- Objekat primer

objekat JSON

objekat Book

objekat Author

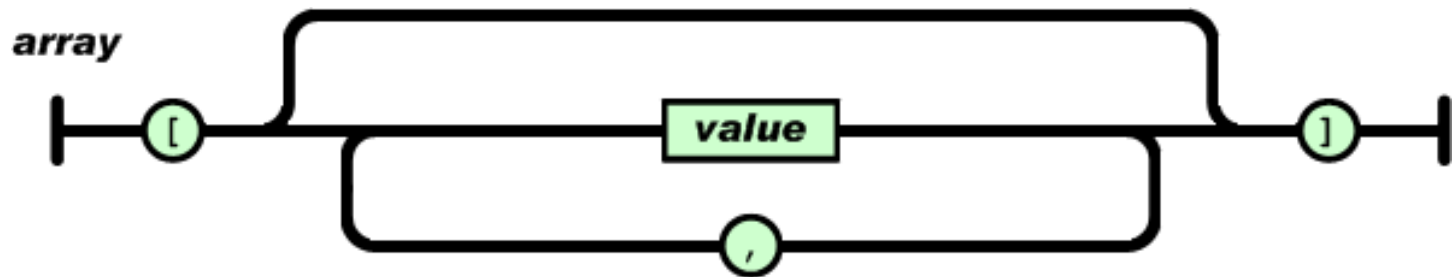
```
{  
  "Book": {  
    "ISBN": "ISBN-0-13-713526-2",  
    "Price": 85,  
    "Edition": 3,  
    "Title": "A First Course in Database Systems",  
    "Author": {  
      "First_Name": "Jeffrey",  
      "Last_Name": "Ullman"  
    }  
  }  
}
```

JSON

- Niz
 - **uređeni skup vrednost**
 - međusobno razdvojeni znakom “,”
 - počinje sa znakom “[”
 - završava se znakom “]”
 - može biti vrednost u paru naziv/vrednost
 - imenovani niz
 - moguće ugneždavanje nizova

JSON

- Niz



JSON

- Niz primer

niz Books

niz Authors

niz Authors

```
{
  "Books": [
    { "ISBN":"ISBN-0-13-713526-2",
      "Price":85,
      "Edition":3,
      "Title":"A First Course in Database Systems",
      "Authors":[ {"First_Name":"Jeffrey", "Last_Name":"Ullman"},
                   {"First_Name":"Jennifer", "Last_Name":"Widom"} ] },
    { "ISBN":"ISBN-0-13-815504-6",
      "Price":100,
      "Remark":"Buy this book bundled with 'A First Course'",
      "Title":"Database Systems:The Complete Book",
      "Authors":[ {"First_Name":"Hector", "Last_Name":"Garcia-Molina"},
                   {"First_Name":"Jeffrey", "Last_Name":"Ullman"},
                   {"First_Name":"Jennifer", "Last_Name":"Widom"} ] }
  ]
}
```

JSON, Eclipse IDE i Java

- JSON Editor Plugin
 - *syntax highlighting* za JSON datoteke
 - preuzeti sa Eclipse *marketplace*-a
- Jackson (v 2.9.10)
 - biblioteka za rad sa formatom podataka JSON u programskom jeziku Java
 - [jackson-core](#)
 - [jackson-databind](#)
 - [jackson-annotations](#)

Jackson

- Tri metode za obradu JSON-a
 - **direktno mapiranje JSON-a na Java objekte**
 - *data binding*
 - *Plain Old Java Object* (POJO)
 - pristup koji je najjednostavniji za korišćenje
 - **inkrementalno parsiranje/generisanje**
 - *streaming API*
 - čita i piše JSON uz pomoć diskretnih događaja
 - za svaki element JSON-a se generiše događaj koji treba obraditi
 - pristup sa najboljim performansama

Jackson

- Tri metode za obradu JSON-a
 - **mapiranje JSON-a na strukturu tipa stabla**
 - *tree model*
 - struktura u radnoj memoriji u koju se smeštaju isprazirani podaci
 - najfleksibilniji pristup

Primer 1

- Napisati *Java* program koji:
 - mapira sadržaj datoteke *book.json* na odgovarajuće *Java* objekte
 - ažurira objekte sa novim podacima
 - čuva izmene u datoteku *Book_changed.json*
- Zadatak uraditi koristeći:
 - Jackson biblioteku
 - **direktno mapiranje JSON-a na Java objekte**

Primer 2

- Napisati *Java* program koji:
 - generiše novi JSON sa podacima o autoru i njegovoj knjizi
 - generiše datoteku *Book_generated.json*
 - učitava autora i naslov njegove knjige
- Zadatak uraditi koristeći:
 - Jackson biblioteku
 - **inkrementalno parsiranje/generisanje**

Primer 3

- Napisati *Java* program koji:
 - učitava autora i naslov njegove knjige
 - ažurira naslov knjige
 - sačuvava izmene u datoteci *Book_changed.json*
- Zadatak uraditi koristeći:
 - Jackson biblioteku
 - **mapiranje JSON-a na strukturu tipa stabla**

Zadatak 1

- Napisati *Java* program koji iz JSON datoteke *bookstore.json* učitava sve knjige i časopise i ispisuje ih na standardni izlaz.

Zadatak 2

- Napisati Java program koji:
 - učitava sve podatke iz CSV datoteke *countries_cities.csv*
 - grupiše države po kontinentu kojem pripadaju
 - za svaki kontinent snima po jednu JSON datoteku koja sadrži sve podatke o državama koje se nalaze na tom kontinentu

Zadatak 3

- Napisati Java program koji vrši pretragu po JSON datoteci sa podacima preuzetim sa *Twitter*-a. Program treba da omogući:
 - čuvanje id-a korisnika i teksta njegovog *tweet*-a
 - čuvanje ukupnog osećanja nekog *tweet*-a
 - čuvanje ukupnog osećanja datog korisnika na osnovu reči u svim njegovim *tweet*-ovima
 - prikaz odnosa pozitivnih i negativnih *tweet*-ova

Zadatak 3

- Date su dve datoteke sa *tweet*-ovima*
 - output_1.txt
 - *tweet*-ovi su sakupljani 1 minut
 - za inicijalnu analizu podataka
 - output_10.txt
 - *tweet*-ovi su sakupljani 10 minuta
 - za detaljniju analizu podataka
- Značenje JSON atributa iz datih datoteka
 - <https://dev.twitter.com/overview/api/tweets>

*Datoteke se mogu preuzeti [ovde](#).

Zadatak 3

- Data je datoteka sa engleskim rečima i osećanjem koju svaka reč nosi
 - *AFIN-111.txt*
 - svaka reč i njeno osećanje su u posebnom redu
 - reč i jačina osećanja su razdvojeni tabom
 - jačina osećanja je predstavljena celim brojem u intervalu od -5 do 5
 - -5 veoma negativno osećanje
 - 5 veoma pozitivno osećanje
 - preuzeto sa http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010

Zadatak 4

- Napraviti Java program koji za bilo koju ulaznu JSON datoteku proverava da li je sintaksa ispravna (da li je datoteka dobro formirana).

JSON Schema

- Opisuje strukturu JSON dokumenata
- Takođe je JSON dokument
 - mogu se koristiti isti alati za učitavanje ovog dokumenta
- Upotreba
 - dokumentacija
 - automatizacija rada sa JSON datotekama
 - generisanje koda

JSON Schema

- JSON Schema specifikacija
 - <http://json-schema.org/latest/json-schema-core.html>
- JSON Schema validator
 - <http://jsonschemalint.com/>
- Primer
 - schema u datotece *Book2Schema.json*
 - podaci u datoteci *Book2.json*

JSON Schema

- *Book2.json*

```
1 {  
2   "ISBN": "ISBN-0-13-713526-2",  
3   "Price": 85,  
4   "Edition": 3,  
5   "Title": "A First Course in Database Systems",  
6   "Author": {  
7     "First_Name": "Jeffrey",  
8     "Last_Name": "Ullman"  
9   },  
10  "tags": ["Databases", "Data", "Organization"]  
11 }  
--
```


JSON Schema

- Korenski element je objekat i sadrži informacije o samoj schemi
- *metadata* ključne reči
 - “*title*”
 - naziv elementa
 - “*description*”
 - opis elementa

JSON Schema

- “*type*”
 - tip podataka JSON elementa
 - dovoljene vrednosti
 - array, boolean, integer, number, null, object, string
 - **za korenski element mora biti *object***

JSON Schema

- *“properties”*
 - specifikacija atributa nekog objekta
 - sam po sebi JSON objekat
 - svaka specifikacija objekta u schemi mora imati *properties* objekat
 - specifikacija parova naziv/vrednost

JSON Schema

```
1 {
2   "title": "Book schema",
3   "description": "A schema for the book JSON object from the Data organization course",
4   "type": "object",
5   "properties": {
6     "ISBN": {
7       "description": "The unique identifier for a book",
8       "type": "string"
9     },
10    "Price": { "type": "integer" },
11    "Edition": { "type": "integer" },
12    "Title": { "type": "string" },
13    "Author": {
14      "type": "object",
15      "properties": {
16        "First_name": { "type": "string" },
17        "Last_name": { "type": "string" }
18      }
19    },
20    "tags": {
21      "type": "array",
22      "items": {
23        "type": "string"
24      }
25    }
26  }
27 }
```

JSON Schema

- ograničenja
 - za svaki element mogu se napisati ograničenja koja važe za vrednosti tog elementa
 - postoji skup opštih elemenata za kontrolu vrednosti
 - za svaki tip postoji skup predefinisanih funkcija

JSON Schema

- globalna ograničenja
 - važe za element bilo kog tipa
 - *“optional” : true/false*
 - za svaki element može se reći da li je obavezan ili ne
 - *“enum” : lista vrednosti*
 - lista dozvoljenih vrednosti
 - *“allOf”, “anyOf”, “oneOf” : lista vrednosti*
 - lista schema koje vrednosti moraju zadovoljiti
 - *“not” : lista vrednosti*
 - lista schema koje vrednosti ne smeju zadovoljiti

JSON Schema

- ograničenja za *number* i *integer* tipove
 - “*multipleOf*” : broj
 - proverava da li je vrednost elementa deljiva sa brojem specificiranim u ograničenju
 - “*maximum*” : broj
 - “*exclusiveMaximum*” : true/false
 - “*minimum*” : broj
 - “*exclusiveMinimum*” : true/false

JSON Schema

- ograničenja za *string* tip
 - “*maxLength*” : broj
 - “*minLength*” : broj
 - “*pattern*” : *string*
 - vrednost ovog elementa mora biti ispravan regularni izraz

JSON Schema

- ograničenja za *array* tip
 - “*items*” : *string*
 - vrednost označava podschemu za opis vrednosti u listi
 - “*additionalItems*” : *true/false* ili *objekat*
 - “*maxItems*” : *broj*
 - “*minItems*” : *broj*
 - “*uniqueItems*” : *true/false*

JSON Schema

- ograničenja za *object* tip
 - “*maxProperties*” : broj
 - “*minProperties*” : broj
 - “*required*” : lista vrednosti
 - “*additionalProperties*” : *true/false* ili objekat
 - “*patternProperties*” : objekat

JSON Schema

```
1 {
2   "title": "Book schema",
3   "description": "A schema for the book JSON object from the Data organization course",
4   "type": "object",
5   "properties": {
6     "ISBN": {
7       "description": "The unique identifier for a book",
8       "type": "string",
9       "pattern": "ISBN*",
10      "optional": false
11    },
12    "Price": {
13      "type": "integer",
14      "minimum": 0,
15      "maximum": 100000
16    },
17    "Edition": { "type": "integer" },
18    "Title": { "type": "string" },
19    "Author": {
20      "type": "object",
21      "properties": {
22        "First_name": { "type": "string" },
23        "Last_name": { "type": "string" }
24      }
25    },
26    "tags": {
27      "type": "array",
28      "items": {
29        "type": "string"
30      },
31      "minItems": 1,
32      "uniqueItems": true
33    }
34  }
35 }
```

Zadatak 5

- Napisati JSON Schema dokument za podatke koji se nalaze u datoteci *Bookstore.json*

Zadatak 6

- Napisati Java program koji validira zadatu JSON datoteku i proverava njenu sintaksnu i semantičku tačnost u odnosu na zadatu schemu.