JSON

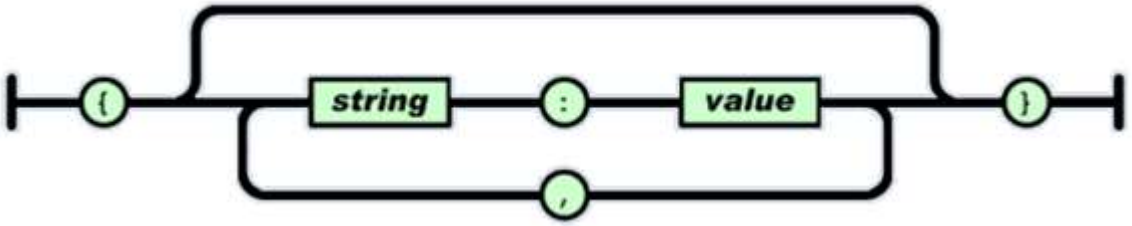
Zoran Ćirović

Uvod

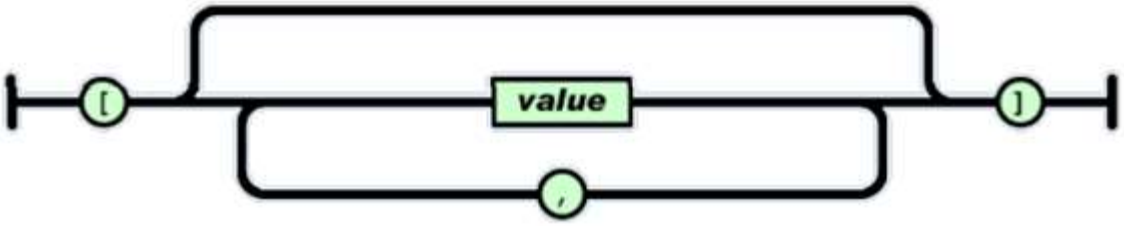
- JSON (engl. JavaScript Object Notation) – format za razmenu podataka. Jednostavan za obradu. Lak za razumevanje i upotrebu. Računari ga lako parsiraju. Bazira se na podskupu JavaScript programskog jezika, standard ECMA-262– decembar 1999.
- Predstavlja tekstualni zapis, kombinaciju `name:value`
- Nezavistan je od korišćenog programskog okruženja i jezika. 2002. pojavljuje se JSON.org a 2005. Predstavlja ključni deo ideje AJAX-a.
- Nije markup language! JSON se bazira na dve programske strukture:
  - • Kolekcija parova ime - vrednost.
  - • Uređena lista. U najvećem broju jezika, ovo se realizuje kao niz, vektor, lista, ili sekvenca.
- Ovo su univerzalne strukture podataka. Svi moderni programski jezici podržavaju ih u nekoj formi. Upravo iz tog razloga i ima smisla da format za razmenu podataka bude baziran na tim strukturama.

Osnovna sintaksa

- Objekat je nesređeni skup **ime/vrednost** parova. Objekat počinje sa „{“ i završava se sa „}“. Svako ime sledi „:“. Ime/vrednost parovi su razdvojeni zarezom.



- Niz je uređena kolekcija vrednosti. Niz počinje sa „[“, a završava se sa „]“. Vrednosti su razdvojene zarezom, kao što je slučaj kod objekata.



JSON objekti

- JSON format je gotovo identičan JavaScript objektima.

- JSON podatak:
- `{ "duzina" : "123.33" }`
- JSON podatak zahteva da naziv takođe bude između znakova navoda (jednostruki ili dvostruki).

- JavaScript to ne zahteva, drugim rečima naziv se podrazumeva da je string.
- `{ duzina : "123.33" }`
- Vrednost može biti jedan od tipova: string, broj, objekat, niz, boolean ili null. Vrednost može biti takođe: funkcija, datum i undefined.

Tipovi podataka u JSON-u

- Broj - JavaScript format u pokretnom zarezu sa dvostrukom preciznošću, zavisi od implementacije.
- String - Unicode format, sa dvostrukim navodnicima, kao izlazna sekvenca se koristi *backslash*
- Boolean - *true* ili *false*
- Niz - uredjena sekvenca vrednosti, odvojena zarezima i uokvirena kockastim zagradama; vrednosti ne moraju biti istog tipa
- Objekat - neuredjena kolekcija ključ:vrednost parova sa ':' karakterom koji razdvaja ključ i vrednost, razdvojeni zarezima i uokvireni vitičastim zagradama; ključevi moraju biti različiti od ostalih ključeva.
- null – prazno.
- Beline nemaju značaj i mogu se slobodno dodati izmedju strukturalnih karaktera (zagrada "{ } [ ]", dve tačke ":" i zareza ",").

JSON objekti različitih tipova u JS

- String
  - `{"ime":"Jovan"}`
- Broj
  - `{"starost":30}`
- Objekat
  - `{"student" : { "ime": "Jovan", "starost": 23, "mesto": "Pančevo" } }`
- Niz
  - `{"imena" : [ "Petar", "Jovan", "Dragan" ] }`
- Boolean
  - `{"diplomirao":true}`
- Null
  - `{nagrade:null}`

JSON i JavaScript objekti

- JSON sintaksa je izvedena iz JavaScript objektne notacije tako da se objekat u JS lako definiše na sličan način.
- `var student = { "ime": "Jovan", starost: 23, "mesto": "Pančevo" };`

- Pristup objektu može da bude kao:
- `var imeStudenta = student.ime;`
- `var mesto = student["mesto"];`
- `mesto = student.mesto;`
- `student.mesto = "Beograd";`

Objekat u objektu - ugnježdavanje objekta

```
obj1={
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,

    "address": {
        "streetAddress": "21 2nd
          Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

Primeri pristupa vrednostima:

```
X = obj1.address.city;
Y = obj1.phoneNumbers[0]["number"];
```

Petlja po svojstvima objekta

Objekat može sadržati unapred nepoznata svojstva ili veći broj svojstava. Petlja koja bi ispisivala svojstva odnosno vrednosti svojstava jednog objekta prikazana je u sledećem primeru:

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>  <meta charset="utf-8" />  <title></title> </head>
<body>
    <div id="div1"></div>
    <script>
        var student = { ime: "Jovan", starost: 23, mesto: "Pančevo" };
        function studentTabela() {
            var rez = "<table>";
            for (var x in student) {
                rez += "<tr>";
                rez += "<td>" + x + "</td>";
                rez += "<td>" + student[x] + "</td>";
                rez += "</tr>";
            }
            rez += "</table>";
            return rez;
        }
        document.getElementById("div1").innerHTML = studentTabela();
    </script>
</body>
</html>
```

Nizovi u JSON-u

- Po prethodno definisanoj sintaksi niz niz stringova je na primer:
- `[ "Pera", "Mika", "Zika" ]`

- Svaki objekat se definiše posebno u samoj definiciji niza ili je ranije definisan. Njihovo grupisanje u niz postiže se uglastim zagradama [ ], a međusobno su razdvojeni zapetom.
- Primer:

```
{ studenti : [
   { "ime":"Pera" , "prezime":"Peric" },
   { "ime ":"Mika" , "prezime":"Mikic" },
   { "ime ":"Zika" , "prezime":"Zikic" }
]}
```

Nizovi kao JSON objekti

Broj elemenata niza se određuje pomoću svojstva `length`, a element niza se referiše pomoću uglastih zagrada sa indeksom koji je pozicija elementa u nizu.

```
var student = { ime: "Jovan", starost: 23, mesto: "Pančevo", predmeti:
["UIT", "IST", 43] };
function sviPredmeti() {
    var rez = "";
    for (i = 0; i < student.predmeti.length; i++)    {
        rez += student.predmeti[i];
    }
    return rez;
}
```

Kreiranje objekta iz stringa

- U komunikaciji sa serverom ili čitajući JSON iz nekog dokumenta podaci su uvek u vidu stringa. Kreiranje objekta na osnovu stringa vrši se primenom metode `JSON.parse(obj)`.

```
<body>
    <div id="div1"></div>
<script>
    var tmp = '{"ime":"Jovan", "indeks":"NRT-44218", "godina":4}';
    var obj = JSON.parse(tmp);
    document.getElementById("div1").innerHTML = "indeks=" + obj.indeks;
</script>
```

Izdvajanje posebnih tipova podataka

- JSON sadrži podatke određenih tipova koje smo ranije spomenuli. Moguće je eksplicitno izvršiti konverziju u neki drugi tip. Konverzija se može uraditi nakon učitavanja ili pri samom čitanju podataka. Primer se odnosi na konverziju u tip *Date*.
- U drugom slučaju to se izvodi uključivanjem ***reviver*** funkcije u provere. Funkcija je ugrađena na mesto ge se i koristi.

```
var obj2 = JSON.parse(datasource, function (key,value) {
   if (key == "datumUpisa")
       return new Date(value);
    else
       return value;
});
```

- U nastavku je dat ceo primer (bez HTML zaglavlja).

```
<body>
    Pre konverzije obj1: <div id="div1"></div>
    Nakon konverzije obj1: <div id="div2"></div>
    Konverzija pri učitavanju obj2: <div id="div3"></div>
<script>
    var datasource = '{ "ime":"Jovan", "datumUpisa":"2020-07-10", "skola":"VISER"}';
    var obj1 = JSON.parse(datasource);

    // pre konverzije
    document.getElementById("div1").innerHTML = obj1.datumUpisa;

    // zamena postojećeg svojstva obj1.datumUpisa
    // koji je bio string, u novi istog imena koji je Date
    obj1.datumUpisa = new Date(obj1.datumUpisa);
    document.getElementById("div2").innerHTML = obj1.datumUpisa;

    // konvrerzija pri samom učitavanju podataka
    var obj2 = JSON.parse(datasource, function (key,value) {
        if (key == "datumUpisa")
            return new Date(value);
        else
            return value;
    });
    document.getElementById("div3").innerHTML = obj2.datumUpisa;

</script>
</body>
```

Parsiranje funkcija kroz JSON

- JSON sadrži samo string. Međutim ako se funicija napiše sintaksno ispravno i takav string uključi kao vrednost nekog svojstva, onda se pozivom tog svojstva poziva funkcija. Pogledajmo sledeći primer koji ima svojstvo tSG koje se zamenjuje formiranom funkcijom iz stringa podataka. Formiranje izvršnog koda, u ovom slučaju funkcije, iz stringa može se uraditi pomoću funkcije ***eval***.

```
<script>
    // redosled znakova navoda je bitan
    var strFja = '"function() {' +
    "var sad = new Date();" +
    'return sad.getMonth() < 10 ? sad.getFullYear() - 1 : sad.getFullYear();}"';
    var datasource = '{ "ime":"Jovan", "datumUpisa":"2020-07-10", "tSG":' + strFja +' }';

    var obj1 = JSON.parse(datasource);
    obj1.tSG = eval("("+obj1.tSG+")");

    document.getElementById("div1").innerHTML = obj1.tSG();
</script>
```

Napomena:

- Stariji čitači ne podržavaju JS funkciju `JSON.parse()`. Za njih se može koristiti `eval()` za konverziju JSON teksta u bilo koji JS objekat.
- Potencijalni problem pri konverziji nastaje zbog slobode pisanja brojeva u JSON-a. Brojevi se mogu zapisati kao numerički literali ili nizovi pod navodnicima. Na primer poštanski kod počinje sa nulama (na primer 011 za Beograd). Ako jedan programer piše pod navodnicima, a drugi ne, vodeća nula se može izgubiti prilikom razmene podataka ta 2 sistema.

```
{
"firstName": "John",
"lastName": "Smith",
"age": 25,
"address": { "streetAddress": "21 2nd Street", "city": "New York",
"state": "NY", "postalCode": "10021" },
"phoneNumber": [
{ "type": "home", "number": "212 555-1234" },
{ "type": "fax", "number": "646 555-4567" }
] }
```

```
var p = eval("(" + contact + ")");
var p = JSON.parse(contact);
```

Konverzija objekta u string

- U tipičnoj komunikaciji sa serverom objekti se moraju prebaciti u string pogodan za razmenu podataka. Konverzija se obavlja pomoću metode `JSON.stringify(obj)`
- Primer:

```
<body>
    student: <div id="div1"></div>
    <script>
    var obj = {ime:"Jovan", indeks:"NRT-44218", godina:4, datum:new Date()};
    document.getElementById("div1").innerHTML = JSON.stringify(obj);
    </script>
</body>
```

  - Kada se datum konvertuje onda se dobija izlaz vrednost datuma.

Serijalizacija funkcija u JSON

- Funkcije nisu dozvoljene u metodi `stringify` i biće preskočeno konvertovanje u string.
- Umesto konverzije na taj način, mogu se konvertovati prethodno u string (pozivom metode `toString()`) zameniti funkcija stringom, a onda se primeniti `stringify`. Na primer:

```
<body>
    student: <div id="div1"></div>
    <script>
        var obj = {
            ime: "Jovan",
            indeks: "NRT-44218",
            godina: 4,
            datum: new Date(),
            semestar: function () { return 4; }
        };
        obj.semestar = obj.semestar.toString();
        document.getElementById("div1").innerHTML = JSON.stringify(obj);
    </script>
</body>
```

JSON i XML

- Najčešće se nudi kao alternativa XML-u
- Sličnosti: Oba su bliska čitanju i tumačenju čoveka
- Oba imaju vrlo prostu sintaksu
- Oba su hijerarhijski organizovana
- Oba su nezavisna od jezika Oba se mogu koristiti za AJAX
- Razlike: Sintaksa je različita: JSON je manje opisan
- JSON može biti parsiran sa JavaScript *eval* metodom
- Name u JSON-u ne sme biti JavaScript rezervisana reč
- XML može biti proverljiv (validated)

Neke prednosti JSON

- XML je teži za prasiranje (ali je značajno i fleksibilniji).
- JSON je jako prilagođen JavaScript objektima.
- Za AJAX aplikacije JSON je brži i lakši za upotrebu od XML:
- Pri upotrebi XML
  - Dohvatiti XML dodokument
  - Koristeći XML DOM kretati se po dokumentu
  - Ekstrakcija i čuvanje vrednosti
- Pri upotrebi JSON
  - Dohvatiti JSON string
  - Primeniti JSON.Parse na JSON string

JSON i XML

```
firstName: John
lastName: Smith
age: 25
address:
   streetAddress: 21 2nd Str
   city: New York
   state: NY
   postalCode: 10021
phoneNumber:
   -
   type: home
   number: 212 555-1234
   -
   type: fax
   number: 646 555-4567
```

```
<person>
  <firstName> John </firstName>
  <lastName> Smith </lastName>
  <age> 25 </age>
  <address>
    <streetAddress> 21 2nd Street </streetAddress>
    <city> New York </city>
    <state> NY </state>
    <postalCode> 10021 </postalCode>
  </address>
  <phoneNumbers>
    <phoneNumber type="home">212 555-1234</phoneNumber>
    <phoneNumber type="fax">646 555-4567</phoneNumber>
  </phoneNumbers>
</person>
```

```
<person firstName="John" lastName="Smith" age="25">
  <address streetAddress="21 2nd Street" city="New York" state="NY" postalCode="10021" />
  <phoneNumbers>
    <phoneNumber type="home" number="212 555-1234"/>
    <phoneNumber type="fax" number="646 555-4567"/>
  </phoneNumbers>
</person>
```

XML fajl

```
<?xml version='1.0' encoding='UTF-8'?>
<card>
   <fullname>Sean Kelly</fullname>
   <org>SK Consulting</org>
   <emailaddrs>
      <address type='work'>kelly@seankelly.biz</address>
      <address type='home' pref='1'>kelly@seankelly.tv</address>
   </emailaddrs>
   <telephones>
      <tel type='work' pref='1'>+1 214 555 1212</tel>
      <tel type='fax'>+1 214 555 1213</tel>
      <tel type='mobile'>+1 214 555 1214</tel>
   </telephones>
   <addresses>
      <address type='work' format='us'>1234 Main StSpringfield</address>
      <address type='home' format='us'>5678 Main StSpringfield</address>
   </addresses>
   <urls>
      <address type='work'>http://seankelly.biz/</address>
      <address type='home'>http://seankelly.tv/</address>
   </urls>
</card>
```

JSON fajl

```
{
   "fullname": "Sean Kelly",
   "org": "SK Consulting",
   "emailaddrs": [
      {"type": "work", "value": "kelly@seankelly.biz"},
      {"type": "home", "pref": 1, "value": "kelly@seankelly.tv"}
   ],
   "telephones": [
      {"type": "work", "pref": 1, "value": "+1 214 555 1212"},
      {"type": "fax", "value": "+1 214 555 1213"},
      {"type": "mobile", "value": "+1 214 555 1214"}
   ],
   "addresses": [
      {"type": "work", "format": "us", "value": "1234 Main
      StnSpringfield"},
      {"type": "home", "format": "us","value": "5678 Main
      StnSpringfield"}
   ],
   "urls": [
      {"type": "work", "value": "http://seankelly.biz/"},
      {"type": "home", "value": "http://seankelly.tv/"}
   ]
}
```

JS sa XML podacima

```
function myHandler()
{
    if (req.readyState == 4 /*complete*/)
    {
        var addrField =
        document.getElementById('addr');
        var root = req.responseXML;
        var addrsElem =
        root.getElementsByTagName('addresses')[0];
        var firstAddr =
        addrsElem.getElementsByTagName('address')[0];
        var addrText = fistAddr.firstChild;
        var addrValue = addrText.nodeValue;
        addrField.value = addrValue;
    }
}
```

JS sa JSON podacima

```
function myHandler()
{
  if (req.readyState == 4 /*complete*/)
  {
    var addrField = document.getElementById('addr');
    var card = eval('(' + req.responseText + ')');
    addrField.value = card.addresses[0].value;
  }
}
```

Šema

- JSON Šema je specifikacija za formate zasnovane na JSON-u za definisanje strukture JSON podataka. JSON shema pruza garancije za kakvi su JSON podaci potrebni u datoj aplikaciji i kako se mogu menjati.
- JSON shema obezbeđuje validaciju i kontrolu sa JSON podacima. JSON shema je bazirana na konceptima XML Schema, ali je namenjena na JSON-u. Isti alati za serijalizaciju/deserijalizaciju se mogu koristiti za šemu i podatke.

```
{
    "name": "Product",
    "properties": {
        "id": {
            "type": "number",
            "description": "Product identifier",
            "required": true
        },
        "name": {
            "type": "string",
            "description": "Name of the product",
            "required": true
        },
        "price": {
            "type": "number",
            "minimum": 0,
            "required": true
        },
}
```

```
"tags": {
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "stock": {
            "type": "object",
            "properties": {
                "warehouse": {
                    "type": "number"
                },
                "retail": {
                    "type": "number"
                }
            }
        }
    }
}
```

JSON šema iznad se moze koristiti za testiranje validnosti sledećeg JSON koda:

```
{
    "id": 1,
    "name": "Foo",
    "price": 123,
    "tags": [ "Bar", "Eek" ],
    "stock": {
        "warehouse": 300,
        "retail": 20
    }
}
```