JSON

Priča o radu sa podacima i frontend programiranju ne može proći bez još jednog pojma, koji na današnjem webu poseduje ogromnu važnost. Reč je o formatu za predstavljanje podataka koji je specijalno razvijen za potrebe aplikacija koje koriste JavaScript jezik. Upravo zbog toga, pomenuti format nosi naziv *JavaScript Object Notation* ili skraćeno JSON. Reč je o primarnom formatu koji se danas koristi za razmenu podataka između delova web aplikacija, odnosno između prezentacionog sloja (*frontend*) i pozadinske logike (*backend*).

Modul koji je pred vama u potpunosti će biti posvećen upoznavanju sa JSON formatom. Prvo će biti iznete osnovne osobine takvog formata, kao i sintaksa pravila za njegovo kreiranje, a nakon toga će biti prikazani različiti pristupi za programabilno rukovanje podacima u JSON formatu, korišćenjem JavaScript jezika.

Šta je JSON?

JSON je jednostavan, jezički nezavisan tekstualni format za razmenu podataka. Bazira se na delu treće edicije ECMAScript jezičke specifikacije iz 1999. godine. JSON je kreirao Douglas Crockford početkom dvehiljaditih godina, a 2013. organizacija ECMA International ga je standardizovala kao ECMA-404 standard.

JSON je namenjen serijalizaciji strukturiranih podataka. To praktično znači da JSON omogućava da se podaci koji su tokom izvršavanja programa predstavljani korišćenjem oblika tipičnih za većinu programskih jezika (objekti, nizovi, liste...) na veoma lak način pretvore u tekstualni oblik i u toj formi podele.

JSON je kreiran na osnovi dela ECMAScript jezičke specifikacije, koja se tiče pravila za formulisanje objektnih literala. Ipak, JSON je platformski i jezički nezavisan format i može se koristiti prilikom rada sa brojnim programskim jezicima: C, C++, C#, Java, JavaScript, Perl, Python...

Fajlovi sa podacima u JSON formatu imaju ekstenziju **.json**.

Kako izgleda JSON?

Odmah na početku, pre upoznavanja osobina JSON-a, biće prikazano nekoliko primera podataka u JSON formatu. Biće to podaci koji su već korišćeni u jednoj od prethodnih lekcija, u kojoj su predstavljeni korišćenjem XML-a.

Za početak, evo kako u JSON formatu mogu izgledati podaci o jednoj državi:

```
{
      "countryCode": "sr",
      "name": "Serbia",
      "capital": "Belgrade",
      "description": "Serbia is....."
}
```

Dakle, reč je o podacima koje smo već videli u jednoj od prethodnih lekcija, kada su oni bili markirani korišćenjem XML jezika. To su jednostavni podaci o državi, pri čemu se država predstavlja kodom, imenom, glavnim gradom i kratkim opisom, koji je u primeru zbog preglednosti maksimalno uprošćen.

JSON-om je moguće prikazivati i kolekcije podataka, što bi u našem primeru predstavljao veći broj država:

```
[
	{
		"countryCode": "sr",
		"name": "Serbia",
		"capital": "Belgrade",
		"description": "Serbia is....."
	},
	{
		"countryCode": "fr",
		"name": "France",
		"capital": "Paris",
		"description": "France is....."
	}
]
```

Sada su u JSON formatu predstavljani podaci dve države.

Kako biste na pravi način razumeli kako su formirani upravo prikazani primeri, neophodno je da se upoznamo sa osnovnim pravilima za predstavljanje podataka u JSON formatu. Drugim rečima, neophodno je poznavati strukturu i sintaksu JSON formata.

JSON struktura i sintaksa

Iz prikazanih primera odmah se može uvideti sličnost između JSON formata i načina na koji se u JavaScriptu kreiraju objektni literali i literali nizova. Drugim rečima, tekst u JSON formatu jeste zapravo serijalizovan objekat ili niz. Tekstom prvog prikazanog primera serijalizuju se podaci jednog objekta. Tekstom drugog primera serijalizuju se podaci jednog niza objekata. Svaki objekat predstavlja jednu državu, pri čemu objekti koji predstavljaju države poseduju četiri svojstva sa tekstualnim, string vrednostima.

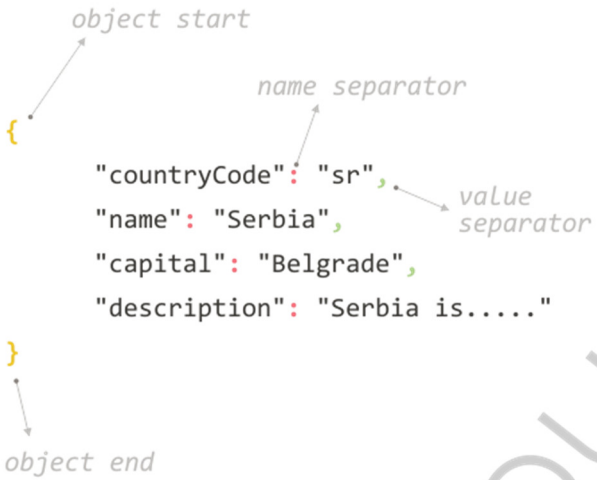
JSON tekst se može sastojati iz sledećih elemenata:

- šest strukturalnih karaktera: `[ ] { } : ,`
- teksta
- brojeva
- tri konstante (false, null, true)

Vitičaste zagrade se koriste za formiranje objekata:

- **{** – početak objekta
- **}** – kraj objekta

Karakter dve tačke (**:**) koristi se za razdvajanje naziva od vrednosti, a karakter zapeta (**,**) za razdvajanje vrednosti od narednog para naziva i vrednosti. Sve ovo ilustruje slika 4.1.



Slika 4.1. Upotreba strukturalnih karaktera za formiranje JSON teksta

Na slici 4.1 žutom bojom su obeležene vitičaste zagrade koje se koriste za kreiranje objekata, crvenom bojom su označeni karakteri za razdvajanje naziva, dok su zelenom bojom obojeni karakteri kojima se vrednosti razdvajaju od narednog para naziva i vrednosti.

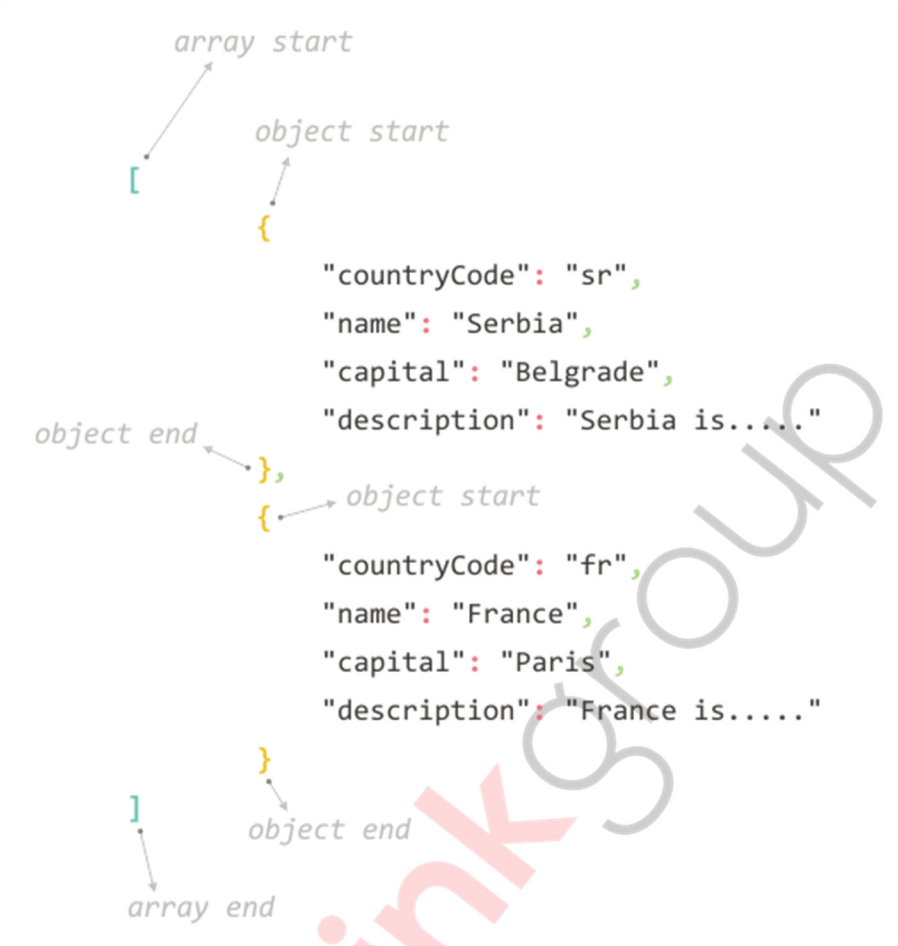
JSON je format u kome se podaci objekata prikazuju u formi parova naziva i vrednosti. Tako je `countryCode` naziv, a njegova vrednost je `sr`. Naziv je `name`, a vrednost `Serbia`, itd. Ovo je osobina koju JSON u potpunosti preuzima od JavaScript jezika, u kome su podaci objekata zapravo parovi svojstava i vrednosti.

Slika 4.2. JSON objektni članovi su parovi naziva i vrednosti

Bitno je primetiti da nakon poslednjeg para naziva i vrednosti nema karaktera zapeta za razdvajanje, pošto unutar objekta više nema parova naziva i vrednosti.

Iz uvodnog primera, mogli ste da vidite da se u JSON formatu mogu predstavljati i nizovi podataka. Nizovi se formiraju korišćenjem uglastih zagrada:

- **[** – početak niza
- **]** – kraj niza



Slika 4.3. Upotreba strukturalnih karaktera za formiranje JSON teksta (2)

Sada je na slici 4.3 prikazana upotreba strukturalnih karaktera na primeru niza podataka u JSON formatu. Novina je upotreba uglastih zagrada, koje označavaju početak i kraj niza. Niz sadrži dva objekta, od kojih svaki predstavlja po jednu državu. Bitno je primetiti da se karakter za razdvajanje vrednosti sada koristi i za razdvajanje vrednosti koje se nalaze unutar niza. Drugim rečima, između tekstova kojima se prikazuju pojedinačni objekti postavljen je karakter zapeta. Naravno, ni sada nema zapete nakon poslednjeg objekta u nizu.

U dosadašnjem toku ove lekcije imali ste prilike da vidite nekoliko veoma važnih osobina JSON formata:

- JSON je format za serijalizaciju podataka;
- JSON formatom mogu se serijalizovati objekti i nizovi;
- JSON objekti se kreiraju upotrebom vitičastih zagrada;
- JSON nizovi se kreiraju korišćenjem uglastih zagrada;
- podaci objekata predstavljaju se kao parovi naziva i vrednosti;
- nazivi se razdvajaju karakterom dve tačke;
- vrednosti se razdvajaju karakterom zapeta.

U dosadašnjem toku lekcije spomenute su dve vrste podataka koje se mogu predstaviti JSON formatom – objekti i nizovi. Ipak, pored njih JSON omogućava predstavljanje još nekih tipova podataka.

Pitanje

JSON se zasniva na:

a) specifikaciji PHP jezika
b) pravilima XML jezika
c) DTD deklaraciji
d) delu ECMAScript specifikacije

Objašnjenje

JSON je kreiran na osnovi dela ECMAScript jezičke specifikacije, koja se tiče pravila za formulisanje objektnih literala.

JSON tipovi podataka

JSON dozvoljava definisanje podataka sledećih tipova:

- broj
- tekst (string)
- objekat
- niz
- konstante: `true`, `false`, `null`

Često se kaže i da JSON omogućava definisanje vrednosti korišćenjem dva prosta tipa (broj i string) i dva kompleksna, odnosno složena tipa (objekat i niz).

Napomena

Kompletan sadržaj JSON fajla mora predstavljati jedinstvenu vrednost, izraženu upravo navedenim tipovima koje JSON poznaje. Drugim rečima, JSON-om nije moguće prikazati dva objekta, osim ukoliko su oni smešteni unutar nekog drugog objekta ili niza:

```
{
    "countryCode": "sr",
    "name": "Serbia",
    "capital": "Belgrade",
    "description": "Serbia is....."
},
{
    "countryCode": "fr",
    "name": "France",
    "capital": "Paris",
    "description": "France is....."
}
```

Ovakav JSON je sintaksno, odnosno strukturalno netačan, zato što poseduje dva objekta. Ipak, ukoliko bi se oni grupisali nizom, dobio bi se validan JSON:

```
[
    {
        "countryCode": "sr",
        "name": "Serbia",
        "capital": "Belgrade",
        "description": "Serbia is....."
    },
    {
        "countryCode": "fr",
        "name": "France",
        "capital": "Paris",
        "description": "France is....."
    }
]
```

Sada je reč o validnom JSON-u, pošto su objekti grupisani nizom.

Po istom principu, jedan JSON fajl ne može imati više od jednog niza, ukoliko oni nisu deo nekog drugog objekta ili niza.

Zanimljivo je da je validan JSON i onaj koji poseduje jednu vrednost nekog prostog tipa ili jednu konstantu:

```
"Hello World"
```

Ovo je takođe validan JSON, zato što poseduje samo jednu vrednost. Ipak, dodavanjem još nekih vrednosti, JSON postaje nevalidan:

```
"Hello World", 13
```

Objekti

Objekat je jedan od složenih JSON tipova kojima se predstavljaju strukture podataka, odnosno kolekcije naziva i vrednosti koje u različitim programskim jezicima postoje u različitim oblicima – kao objekti, strukture, rečnici, hash tabele, asocijativni nizovi... Naravno, u JavaScriptu takav tip podatka je objekat.

JSON objekti se kreiraju korišćenjem vitičastih zagrada, unutar kojih se navode objektni članovi. Objektni članovi se definišu kao parovi naziva i vrednosti, pri čemu se naziv i vrednost odvajaju karakterom dve tačke (slika 4.4).

```
{
    "name": "value"
}
```

Slika 4.4. Sintaksa JSON objekta sa jednim članom

Parovi naziva i vrednosti unutar jednog objekta odvajaju se jedni od drugih karakterom zapeta (slika 4.5).

```
{
    "name": "value",
    "name": "value"
}
```

Slika 4.5. Sintaksa JSON objekta sa dva člana

Nazivi se definišu kao tekst (string), isključivo korišćenjem navodnika ("").

Vrednost može biti bilo kojeg JSON tipa (broj, tekst, niz, objekat, true, false, null). Tako vrednost jednog člana objekta može biti i neki drugi objekat, čime se omogućava modelovanje složenih struktura podataka.

Nizovi

Nizovi se u JSON-u formiraju korišćenjem uglastih zagrada, između kojih se može navesti proizvoljan broj vrednosti (nijedna, jedna ili više njih). Vrednosti unutar jednog niza mogu biti bilo koji validan JSON tip (broj, tekst, niz, objekat, true, false, null).

```
[value1, valu2, value3...]
```

Slika 4.6. Sintaksa JSON niza

Primer jednog jednostavnog JSON niza može da izgleda ovako:

```
[13, 14, 15]
```

Potpuno je jasno da se nizovi u JSON-u formiraju na identičan način kao i literali nizova u JavaScript jeziku, što je uostalom i potpuno logično, s obzirom na činjenicu da se JSON zasniva na ECMAScript specifikaciji.

JSON nizovi ne moraju imati članove istog tipa:

```
[13, "Hello", 15, true, null]
```

JSON nizovi mogu kao elemente imati druge nizove, čime se omogućava predstavljanje višedimenzionalnih nizova:

```
[[1, 2, 3], [4, 5, 6]]
```

Na kraju, kao što je to prikazano nešto ranije u primeru podataka država, JSON niz može da sadrži i više objekata:

```
[
    {
        "countryCode": "sr",
        "name": "Serbia",
        "capital": "Belgrade",
        "description": "Serbia is....."
    },
    {
        "countryCode": "fr",
        "name": "France",
        "capital": "Paris",
        "description": "France is....."
    }
]
```

Brojevi

JSON omogućava definisanje numeričkih podataka, i to u nekoliko različitih oblika:

- celi brojevi
- brojevi u eksponencijalnom obliku
- brojevi u decimalnom obliku

Naredni primeri ilustruju validne JSON brojeve:

```
13
35.78
1234e5
1234e-5
```

JSON dozvoljava definisanje negativnih brojeva, navođenjem prefiksa minus (-):

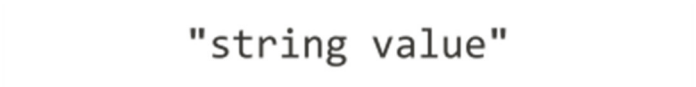
```
-13
-35.78
-1234e5
-1234e-5
```

Najznačajnija pravila za definisanje JSON brojeva su:

- za kreiranje decimalnih brojeva koristi se isključivo karakter tačka;
- brojevi u oktalnom i heksadecimalnom formatu nisu dozvoljeni u JSON-u;
- specijalne numeričke vrednosti koje JavaScript jezik poznaje (`Infinity` i `NaN`) JSON ne podržava;
- JSON ne dozvoljava definisanje brojeva koji započinju nulom; tako su npr. 07 ili 035.78 nevalidni brojevi;
- u JSON-u je nakon decimalne tačke neophodno navesti makar jednu cifru; tako je npr. 13. nevalidan broj.

Stringovi

Pored brojeva, u proste JSON tipove ubraja se i tekst, odnosno string. String vrednosti definišu se isključivo između navodnika ("").



Slika 4.7. Sintaksa JSON stringa

Upotreba apostrofa, odnosno jednostrukih navodnika u JSON-u nije dozvoljena i to je najznačajnija razlika između sintakse JavaScript i JSON stringova.

JSON string vrednost može biti sačinjena iz bilo kojih UNICODE karaktera. Ipak, određeni karakteri se ne mogu direktno pojaviti unutar tekstualnih vrednosti. Tako unutar tekstualne vrednosti nije moguće direktno navesti navodnik:

```
"this is quotation mark: ""
```

Ovo je primer nevalidnog JSON stringa, zato što je unutar tekstualne vrednosti upotrebljen navodnik. S obzirom na to da se navodnici koriste za oivičavanje tekstualnih vrednosti, potpuno je očekivano da bi se na ovaj način zbunio JSON parser, koji ne bi znao gde je kraj tekstualne vrednosti. Stoga se u ovakvim situacijama pribegava operaciji koja se naziva **escape**:

```
"this is quotation mark: \""
```

Escape podrazumeva postavljanje karaktera backslash (\) ispred nekog drugog karaktera. Upravo to je obavljeno i u primeru, kako bi se dobio validan JSON string. Na ovaj način JSON parser će znati da želimo da unutar tekstualne vrednosti prikažemo navodnik, pa će nakon parsiranja tekstualna vrednost izgledati ovako:

```
this is quotation mark: "
```

Sasvim očekivano, ni karakter backslash ne može se direktno naći unutar JSON stringa:

```
"this is backslash: \"
```

I ovo je primer nevalidnog JSON teksta, jer, kao što ste mogli videti u prethodnom primeru, karakter backslash u JSON-u ima specijalnu namenu. Stoga, ukoliko želimo da on stvarno postane deo JSON tekstualne vrednosti, neophodno je obaviti njegov escape:

```
"this is backslash: \\"
```

Sada je dobijen validan JSON string, pri čemu će tekstualna vrednost izgledati ovako:

```
this is backslash: \
```

Escape karakter backslash može se koristiti za kreiranje još nekih specijalnih karaktera:

- `\b` – povratak za jedno mesto unazad (backspace)
- `\f` – form feed
- `\n` – novi red (line feed)
- `\r` – carriage return
- `\t` – tab

Ukoliko je, na primer, potrebno u tekstualnu vrednost uvrstiti prelazak u novi red, može se napisati:

```
"Hello\nWorld"
```

Na ovaj način će biti dobijen tekst:

```
Hello
World
```

Na kraju, u JSON-u nije neophodno obavljati escape slash karaktera:

```
"this is slash: /"
```

Prikazani JSON string je u potpunosti validan. Ipak, većina programskih jezika prilikom generisanja JSON teksta obavlja escape slash karaktera:

```
"this is slash: \/"
```

I ovo je sada validan JSON tekst, pri čemu će escape karakter, kao i u ostalim primerima, biti ignorisan. Ipak, kao što je rečeno, ovako nešto nije obavezno uraditi, već JSON samo ostavlja mogućnost da se ovako nešto obavi. Razlog su vrlo limitirani slučajevi korišćenja u kojima se JSON integriše direktno unutar HTML ili XML koda.

Na kraju, JSON omogućava i da se korišćenjem escape karaktera definiše i bilo koji karakter korišćenjem UTF-16 heksadecimalnog formata (slika 4.8).

Slika 4.8. Sintaksa za prikaz Unicode karaktera u heksadecimalnom UTF-16 formatu

Ukoliko, na primer, želimo da unutar tekstualne JSON vrednosti ispišemo *copyright* karakter, dovoljno je uraditi sledeće:

```
"\u00A9"
```

Parser će ovakvu vrednost pretvoriti u:

```
©
```

Konstante true, false i null

Pored već navedenih tipova podataka, JSON omogućava i upotrebu tri konstante:

- `true`
- `false`
- `null`

Stoga je moguće napisati ovako nešto:

```
{
    "isValid": true,
    "value": null,
    "required": false
}
```

Prikazanim JSON tekstom, predstavljen je jedan objekat sa tri člana, odnosno tri para naziva i vrednosti. Članovi imaju vrednosti koje nisu ni tekst, zato što se ne navode između navodnika, ali ni brojevi, zato što ne predstavljaju validne brojeve. Navedene vrednosti su, zapravo, tri upravo spomenute konstante.

Konstante `true`, `false` i `null` moraju se pisati malim slovima, baš kao što je to urađeno u primeru (na primer, definisanje vrednosti `True` nije validno).

U JSON formatu nije moguće definisati vrednost `undefined`.

Prazna mesta

Iz dosadašnjeg toka lekcije mogli ste da vidite da JSON omogućava definisanje proizvoljne količine praznih mesta pre i nakon šest strukturalnih karaktera. Upravo ta osobina nam je omogućila da JSON tekst jednog objekta formatiramo na sledeći način:

```
{
      "countryCode": "sr",
      "name": "Serbia",
      "capital": "Belgrade",
      "description": "Serbia is....."
}
```

U prikazanom JSON tekstu, nakon strukturalnog karaktera za definisanje početka objekta naveden je prelazak u novi red. Takođe, svaki objektni član je napisan u zasebnom redu, a nakon karaktera dve tačke, svuda je naveden razmak. Svakako, identičan efekat je mogao biti postignut i bez ikakvih praznih mesta:

```
{"countryCode":"sr","name":"Serbia","capital":"Belgrade","description":"
Serbia is....."}
```

Može se zaključiti da JSON parser ignoriše prazna mesta iz upravo prikazanih primera.

Prazna mesta je moguće postavljati i pre i posle numeričkih i tekstualnih vrednosti i konstanti `true`, `false` i `null`. Tako je moguće napisati:

```
[13,        14,        15]
```

Kao i u prethodnom primeru, JSON parseri ignorišu ovakva prazna mesta, pa je ovakav JSON praktično identičan sledećem:

```
[13,14,15]
```

JSON i XML

U dosadašnjem toku ovoga kursa, prikazana su dva različita formata koja se koriste za predstavljanje podataka u tekstualnom obliku: XML i JSON. XML je jezik za obeležavanje koji je nastao na osnovi SGML jezika. Zbog toga XML poznaje pojmove tagova, elemenata i atributa. Sa druge strane, JSON je notacija, odnosno format koji, kao što ste mogli da vidite, funkcioniše po drugačijem principu, zasnovanom na pravilima ECMAScript specifikacije koja se odnose na kreiranje objektnih literala.

JSON format je nešto kompaktniji i lakši za čitanje i pisanje. Takođe, JSON je značajno upotrebljiviji kada je potrebno serijalizovati proizvoljne objekte koji mi samostalno kreiramo tokom razvoja JavaScript aplikacija.

Na kraju, ukoliko XML i JSON posmatramo isključivo iz ugla JavaScript programskog jezika, rad sa JSON podacima je podržan na samom jezičkom nivou, što nije slučaj sa XML-om, za čije rukovanje je neophodno koristiti funkcionalnosti koje izlažu web pregledači.

Alati za rad sa JSON-om

Prilikom rada sa JSON formatom veoma korisni mogu biti i različiti alati:

- **JSON Validator** – validira ispravnost podataka u JSON formatu
- **JSON Minifier** – kreira optimizovani JSON tekst, izbacivanjem svih nepotrebnih praznih mesta, čime se štedi količina memorije potrebne za prikaz podataka u JSON formatu; procesom umanjenja dobija se JSON tekst u jednoj liniji, bez obzira na dužinu JSON-a, što može biti i više nego korisno kada je JSON tekst potrebno postaviti za vrednost JavaScript promenljive, što će biti slučaj u sledećoj lekciji
- **JSON Beautifier** – formatira JSON tekst tako da se maksimalno poboljša čitljivost, dodavanjem praznih mesta i prelazaka u novi red; *Beautifier* i *Minifier* zapravo obavljaju dve suprotne operacije

Na web-u postoji veliki broj besplatnih alata za obavljanje upravo opisanih, ali i mnogih drugih zahvata nad JSON podacima. Jedan od skupova takvih alata može se pronaći na sledećoj adresi:

https://jsonbeautifier.org/

Za kraj ove lekcije biće prikazano nekoliko primera podataka u JSON formatu.

Primer 1 – Predstavljanje država korišćenjem JSON-a

Kao prvi primer, biće prikazan JSON sličan već viđenom na početku ove lekcije:

```
{
    "countries": [
        {
            "countryCode": "sr",
            "name": "Serbia",
            "capital": "Belgrade",
            "description": "Serbia is....."
        },
        {
            "countryCode": "fr",
            "name": "France",
            "capital": "Paris",
            "description": "France is....."
        }
    ]
}
```

Prikazani JSON sadrži objekat sa jednim parom imena i vrednosti, ili, drugim rečima, sa jednim svojstvom. Vrednost svojstva `countries` je jedan niz, unutar koga se nalaze dva objekta. Oba objekta poseduju po četiri para naziva i vrednosti. Sve vrednosti unutar objekata su tipa string.

Primer 2 – Predstavljanje jednog recepta u JSON formatu

Drugi JSON primer podrazumeva predstavljanje podataka jednog kulinarskog recepta koji je u jednoj od prethodnih lekcija opisan korišćenjem XML-a. Ovoga puta će biti urađeno isto to, ali korišćenjem JSON formata, pa ćete na direktnom primeru moći da vidite razlike između XML-a i JSON-a:

```
{
    "title": "Grilled Cheese Sandwich",
    "ingredients": [
        {
            "title": "bread slice",
            "qty": 2
        },
        {
            "title": "cheese slice",
            "qty": 1
        },
        {
            "title": "margarine pat",
            "qty": 1
        }
    ]
}
```

Prikazanim JSON-om predstavlja se jedan objekat, sa dva svojstva: `title` i `ingredients`. Vrednost `title` svojstva je naziv recepta u tekstualnom formatu. Vrednost svojstva `ingredients` je niz od tri objekta, pri čemu svaki objekat predstavlja po jedan sastojak recepta. Objekti koji predstavljaju sastojke poseduju po dva para naziva i vrednosti, `title` sa tekstualnom vrednošću koja predstavlja naziv sastojka i `qty` sa numeričkom vrednošću koja predstavlja količinu sastojka.

Predstavljanje identičnih podataka u jednoj od prethodnih lekcija je korišćenjem XML-a obavljeno na sledeći način:

```
<recipe>
    <title>Grilled Cheese Sandwich</title>
    <ingredients>
        <ingredient qty="2">bread slice</ingredient>
        <ingredient>cheese slice</ingredient>
        <ingredient qty="2">margarine pat</ingredient>
    </ingredients>
</recipe>
```

Primer 3 – Podaci jednog menija u JSON formatu

Naredni primer podrazumevaće predstavljanje podataka jednog menija u JSON formatu. JSON će izgledati ovako:

```
{
    "menu": {
        "id": "file",
        "text": "File",
        "icon": "file.svg",
        "popup": {
            "menuitem": [
                {
                    "text": "New",
                    "onclick": "createNewDoc()"
                },
                {
                    "text": "Open",
                    "onclick": "openDoc()"
                },
                {
                    "text": "Close",
                    "onclick": "closeDoc()"
                }
            ]
        }
    }
}
```

Prikazani JSON čini jedan objekat sa svojstvom `menu`. Njegova vrednost je opet jedan objekat sa svojstvima `id`, `text`, `icon` i `popup`. Svojstva `id`, `text` i `icon` imaju string vrednosti, dok svojstvo `popup` kao svoju vrednosti ima jedan objekat. On poseduje jedno svojstvo `menuitem`, čija vrednost je niz, sačinjen iz tri objekta. Svaki od objekata unutar niza poseduje svojstva `text` i `onclick`, sa odgovarajućim string vrednostima.

Ovakav JSON se bez problema može upotrebiti za generisanje jednog HTML menija.

Primer 4 – Podaci jedne galerije u JSON formatu

Naredni primer predstaviće podatke jedne galerije u JSON formatu. Reč je o galeriji koja se može naći unutar nekog web sajta. Podaci u JSON formatu koji će u nastavku biti prikazani mogu se, na primer, dobiti od servera, a zatim se na osnovu njih može obaviti konstruisanje strukture same galerije unutar HTML dokumenta. JSON može da izgleda ovako:

```
{
    "name": "Main Gallery",
    "images": [
        {
            "id": 1,
            "width": 800,
            "height": 600,
            "title": "View from 15th Floor",
            "thumbnail": {
                "url": "http://www.example.com/image/481989943",
                "height": 125,
                "width": 100
            },
            "show": true
        },
        {
            "id": 2,
            "width": 800,
            "height": 600,
            "title": "Garden View",
            "thumbnail": {
                "url": "http://www.example.com/image/28wqeqw2345",
                "height": 125,
                "width": 100
            },
            "show": true
        }
    ]
}
```

Zbog jednostavnosti, galerija je ograničena na dve slike.

JSON predstavlja jedan objekat sa dva svojstva – `name` i `images`. Svojstvo `name` ima string vrednost koja predstavlja naziv galerije. Svojstvo `images` kao svoju vrednosti ima niz objekata, pri čemu svaki objekat predstavlja po jednu sliku galerije. Svaka slika u galeriji predstavljena je sledećim podacima:

- `id` – identifikaciona vrednost u numeričkom formatu
- `width` – širina slike u numeričkom formatu
- `height` – visina slike u numeričkom formatu
- `title` – tekstualni naziv slike koji može poslužiti kao vrednost `title` HTML atributa
- `thumbnail` – objekat koji sadrži podatke o umanjenoj (*thumbnail*) verziji slike; svaki
- `thumbnail` objekat poseduje:

- `url` – tekstualna vrednost koja ukazuje na putanju na kojoj se nalazi thumbnail slika
- `height` – numerička vrednost koja predstavlja visinu thumbnail slike
- `width` – numerička vrednost kojom se predstavlja širina thumbnail slike
- `show` – svojstvo koje ukazuje da li sliku treba prikazati u galeriji, te stoga može imati vrednosti JSON konstanti `true` ili `false`

Rezime

- JSON je jednostavan, jezički nezavisan tekstualni format za razmenu podataka.
- JSON je kreiran na osnovi dela ECMAScript jezičke specifikacije, koja se tiče pravila za formulisanje objektnih literala.
- JSON je platformski i jezički nezavisan format i može se koristiti prilikom rada sa brojnim programskim jezicima: C, C++, C#, Java, JavaScript, Perl, Python...
- Fajlovi sa podacima u JSON formatu imaju ekstenziju `.json`.
- JSON tekst se može sastojati iz šest strukturalnih karaktera ([ ] { } : ,), teksta, brojeva i tri konstante (false, null i true).
- Unutar JSON-a vitičaste zagrade se koriste za formiranje objekata, a uglaste zagrade za formiranje nizova.
- JSON je format u kome se podaci objekata prikazuju u formi parova naziva i vrednosti.
- Karakter dve tačke (**:**) koristi se za razdvajanje naziva od vrednosti, a karakter zapeta (**,**) za razdvajanje vrednosti od narednog para naziva i vrednosti.
- JSON dozvoljava definisanje brojeva, teksta, objekata, nizova i konstanti true, false i null.
- Nazivi svojstava i string vrednosti se u JSON-u definišu isključivo korišćenjem navodnika ("").
- Escape je operacija kojom se unutar JSON teksta definišu specijalni karakteri, a podrazumeva postavljanje karaktera backslash (\) ispred tih karaktera.