Parsiranje JSON-a

JSON je jezički nezavistan format za predstavljanje podataka u tekstualnom obliku. Ipak, iako je jezički nezavistan, JSON je posebno koristan prilikom kreiranja web aplikacija koje na klijentskom delu koriste JavaScript programski jezik. Stoga će ova i naredna lekcija biti posvećene različitim pristupima koji omogućavaju rukovanje JSON podacima korišćenjem JavaScript programskog koda. Pod manipulacijom, tj. rukovanjem se ovde primarno podrazumeva parsiranje i serijalizacija podataka u JSON formatu.

Kompatibilnost JSON-a i JavaScripta

U prethodnoj lekciji je bio reči o tipovima podataka koji se mogu predstaviti u JSON formatu. Tako ste imali prilike da pročitate da se u JSON formatu mogu predstaviti prosti numerički i tekstualni podaci i objekti i nizovi kao vrste složenih, kompozitnih podataka. Dalje, zbog mogućnosti definisanja konstanti `true` i `false`, može se reći da JSON omogućava i predstavljanje podataka logičkog tipa. Kada se govori o JavaScriptu, sve ovo znači da se bez ikakvih problema u JSON formatu mogu predstaviti sledeći podaci:

- `Object`
- `number`
- `string`
- `boolean`
- `Array`
- `null`

Sa druge strane, JSON podrazumevano ne podržava neke druge JavaScript kompleksne podatke. Pod ovim se prevashodno misli na funkcije, regularne izraze i datume. Ipak, s obzirom na to da je reč o pojmovima koji su u JavaScriptu zapravo objekti, veoma lako se može napisati logika za obavljanje transformisanja takvih podataka u odgovarajući oblik, pre njihove serijalizacije ili deserijalizacije.

Svaki JSON ujedno je i validan JavaScript izraz

Činjenica da se JSON zasniva na ECMAScript specifikaciji omogućava da svaki tekst koji je formiran korišćenjem JSON pravila ilustrovanih u prethodnoj lekciji ujedno bude i validan JavaScript izraz. Šta ovo praktično znači biće ilustrovano jednim primerom. U prethodnoj lekciji je prikazan sledeći JSON:

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

```
            },
            {
                "title": "margarine pat",
                "qty": 1
            }
        ]
    }
```

S obzirom na to da je svaki JSON tekst ujedno i validan JavaScript izraz, moguće je napisati:

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

JSON tekst je sada bez ikakvih izmena prekopiran i postavljen kao vrednost JavaScript promenljive `data`. Na taj način je dobijen jedan JavaScript izraz, odnosno objektni literal, kojim će biti kreiran jedan objekat, a njegova referenca smeštena unutar promenljive `data`. Upravo zbog toga se kaže da je svaki JSON tekst validan JavaScript izraz.

Upravo opisana osobina JavaScripta u potpunosti je postala istinita sa predstavljanjem ECMAScript 2019, odnosno ES10 verzije ove jezičke specifikacije, kada su napravljene dve suptilne izmene na skupu karaktera čije je pojavljivanje u izvornom obliku dozvoljeno unutar JavaScript stringova. Naime, sve do verzije ES10, JavaScript nije dozvoljavao pojavljivanje `U+2028 LINE SEPARATOR` i `U+2029 PARAGRAPH SEPARATOR` karaktera unutar stringova bez navođenja escape karaktera. Sa druge strane, tako nešto je oduvek bilo moguće u JSON-u. Od verzije ES10, ovakva nedoslednost je ispravljena, pa se može reći da je svaki JSON tekst validan JavaScript izraz.

Manipulacija podacima u JSON obliku omogućena je na samom jezičkom nivou JavaScripta. To praktično znači da JavaScript poseduje globalno dostupan objekat `JSON` koji je moguće koristiti za parsiranje i serijalizaciju JSON podataka. U nastavku ove lekcije biće reči o parsiranju JSON-a, dok će se naredna lekcija baviti JSON serijalizacijom.

Pitanje

Da li je JSON-om izvorno moguće predstaviti JavaScript `boolean` podatke?

a) Da
b) Ne

Objašnjenje

JSON format izvorno omogućava definisanje konstanti true i false, što je moguće iskoristiti za predstavljanje JavaScript boolean podataka.

Parsiranje JSON-a

Parsiranje JSON-a podrazumeva konvertovanje podataka iz JSON tekstualnog formata u JavaScript vrednost ili objekat. Parsiranje se može obaviti korišćenjem metode `parse()` ugrađenog `JSON` objekta.

JSON.parse()

`JSON` objekat poseduje metodu `parse()` kojom je moguće obaviti parsiranje JSON teksta.

Metoda `JSON.parse()` poseduje sledeću sintaksu:

```
JSON.parse(text[, reviver])
```

Metoda `parse()` može da prihvati dva parametra:

- `text` – string koji sadrži JSON tekst; ovo je obavezan parametar
- `reviver` – funkcija koja određuje na koji način će biti obavljeno transformisanje parsiranih podataka; ovo je opcioni parametar

Kada metoda `JSON.parse()` završi svoj posao, kao svoju povratnu vrednost ona emituje JavaScript reprezentaciju podataka koji su definisani JSON tekstom. To može biti podatak nekog prostog tipa (`string`, `number`, `boolean`), ali i `Object` ili `Array`, sve u zavisnosti od JSON teksta koji se parsira.

Bitno je reći da JSON string koji se prosleđuje metodi `JSON.parse()` mora biti u potpunosti validan JSON. Bilo kakvo odstupanje od definisanih pravila za kreiranje JSON formata dovešće do emitovanja izuzetka tipa `SyntaxError`.

U nastavku će biti prikazani različiti primeri parsiranja JSON-a korišćenjem `JSON.parse()` funkcije, počevši od onih jednostavnijih:

```
var json = '{"name": "Ben", "age": 44}';
var obj = JSON.parse(json);
```

Korišćenjem prikazane dve naredbe, obavlja se deklarisanje i inicijalizovanje `json` promenljive sa JSON tekstom, a zatim se obavlja parsiranje takvog JSON-a, upotrebom metode `JSON.parse()`. Pošto je JSON tekstom definisan jedan objekat, povratna vrednost `parse()` metode biće jedan JavaScript objekat, čija referenca će biti smeštena unutar promenljive `obj`. U to se vrlo lako možemo uveriti:

```
console.log(obj.name);
```

Ovakva naredba ispisaće unutar konzole vrednost `Ben`, zato što je korišćenjem dot notacije obavljen pristup svojstvu `name` objekta koji je `parse()` metoda kreirala na osnovu JSON teksta.

Upravo prikazani primer ilustrovao je parsiranje JSON-a kojim je predstavljen jedan objekat. Evo sada primera parsiranja jednog JSON niza:

```
var json = '[13, 14, 15, "Hello World"]';
var arr = JSON.parse(json);
```

JSON-om je sada definisan niz sa četiri člana, pri čemu su prva tri podatka numerička, a poslednji tekstualan. Metoda `parse()` će u ovakvoj situaciji obaviti parsiranje i proizvesti `Array` objekat, čija će referenca biti smeštena unutar promenljive `arr`:

```
console.log(arr[2]);
```

Na ovaj način će unutar konzole biti ispisana vrednost 15.

Iz prethodne lekcije je poznato da validan JSON mogu biti i vrednosti prostih numeričkih i tekstualnih tipova:

```
var json = '13';
var value = JSON.parse(json);
```

Iako se retko koristi za predstavljanje ovako jednostavnih podataka, bitno je znati da je i ovo validan JSON i da će u ovakvoj situaciji metoda `parse()` proizvesti vrednost prostog `number` tipa:

```
console.log(value);  //13
```

Vrednosti koje se u JSON-u formiraju korišćenjem konstanti `true`, `false` i `null` korišćenjem metode `parse()` pretvaraju se u JavaScript podatke odgovarajućeg tipa, odnosno `boolean` i `null`, respektivno:

```
var json = 'true';
var value = JSON.parse(json);

console.log(typeof(value));
```

Kod unutar konzole proizvodi:

```
boolean
```

JSON tekst kao vrednost string JavaScript promenljive

Iz prikazanih primera možete da vidite da je JSON tekst definisan kao vrednost JavaScript `string` promenljivih. U realnim okolnostima, odnosno u realnim JavaScript aplikacijama, JSON će uglavnom dolaziti sa nekog servera. Ipak, s obzirom na to da u ovom kursu još nisu obrađeni pristupi za komunikaciju sa serverom, za sada će JSON biti definisan na ovaj način. U jednoj od narednih lekcija biće obrađeni i pristupi za komunikaciju sa serverom, pa će biti prikazani i primeri parsiranja JSON teksta koji se neće nalaziti direktno unutar aplikativne JavaScript logike. Ipak, do tada, neophodno je razumeti neke specifične osobine ovakvog načina definisanja JSON teksta.

JSON tekst kao vrednost `string` JavaScript promenljive najčešće se definiše između apostrofa, odnosno jednostrukih navodnika. Razlog je vrlo jednostavan – JSON nalaže upotrebu dvostrukih navodnika za definisanje naziva ključeva i string vrednosti, pa se onda korišćenje apostrofa za oivičavanje string literala čini potpuno logičnim.

JSON tekst kao vrednost `string` JavaScript promenljive mora zadovoljiti sintaksna pravila kako JSON formata tako i JavaScript stringova. Za početak, to znači da je neophodno navesti escape karakter ispred navodnika, ne zbog validnosti JSON-a već zbog validnosti JavaScript `string`-a:

```
var json = '"This is single quote: \'"';
```

Dalje, kada se obavlja escape nekih specijalnih karaktera čije pojavljivanje nije dozvoljeno u izvornom obliku unutar JSON-a, neophodno je navesti dva karaktera za escape:

```
var json = '"This is double quote: \\""';
```

Sada se unutar JSON stringa definiše karakter navodnik ("). Njega je po pravilima JSON-a neophodno napisati sa prefiksom `\`. Ipak, s obzirom na to da se karakter `\` koristi kao escape karakter i unutar JavaScript `string`-ova, neophodno je obaviti još jedan escape. Zbog toga na kraju u primeru postoje dva karaktera `\`, jedan za JavaScript `string`, a drugi za JSON string.

Bitno je razumeti da je upravo opisane zahvate neophodno obavljati samo kada se JSON tekst direktno smešta unutar JavaScript promenljive. Kada se on dobija od servera, korišćenjem pristupa koji će biti prikazani u jednoj od narednih lekcija, ovako nešto se ne primenjuje.

Primer – Parsiranje i prikaz JSON podataka država

Nakon upoznavanja osnovnih osobina metode `JSON.parse()`, moguće je prikazati prvi realan primer parsiranja JSON podataka. Tako će u nastavku biti obavljeno parsiranje JSON podataka o državama, a zatim i prikaz takvih podataka unutar HTML stranice:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <title>JSON parse example - countries</title>
    </head>
    <body>
        <script>
            var json =
'{"countries":[{"countryCode":"sr","name":"Serbia","capital":"Belgrade
","description":"Serbia
is....."},{"countryCode":"fr","name":"France","capital":"Paris","descr
iption":"France is....."}]}';

            var data = JSON.parse(json);
            for (let i = 0; i < data.countries.length; i++) {
                document.body.innerHTML += '<h2>' +
data.countries[i].name + '</h2>'
                document.body.innerHTML += '<p>Capital: ' +
data.countries[i].capital + '</p>'
                document.body.innerHTML += '<p>Description: ' +
data.countries[i].description + '</p>'
            }

        </script>
    </body>
</html>
```

Ovakav HTML dokument, unutar web pregledača stvoriće prikaz kao na slici 5.1.

Slika 5.1. Parsirani JSON podaci, prikazani na stranici

Parsiranjem prikazanog JSON-a dobija se jedan objekat sa svojstvom `countries`. Ovo svojstvo sadrži niz sa objektima koji predstavljaju države.

Novina u prikazanom primeru jeste kod koji je definisan nakon parsiranja JSON-a. Unutar jedne for petlje obavlja se prolazak kroz sve članove niza `data.countries`. Svakom iteracijom obavlja se dinamičko generisanje HTML koda. Naziv države se smešta unutar `h2` elementa, a kod, glavni grad i opis unutar pojedinačnih paragrafa.

Parametar reviver

Nešto ranije, prilikom prikaza sintakse metode `JSON.parse()`, mogli ste videti da je njoj, pored JSON teksta, moguće proslediti još jedan parametar. Reč je o parametru kojim je moguće obaviti transformaciju parsiranih vrednosti, pre nego što one budu isporučene kao deo povratne vrednosti.

Naredni primer ilustrovaće korišćenje reviver parametra, a pored toga i još jedan veoma značajan segment rada sa podacima u JSON formatu – rukovanje datumima. Naime, s obzirom na to da JSON izvorno ne poznaje poseban tip za prikaz datuma, u nastavku će biti prikazano kako se korišćenjem reviver parametra može obaviti transformisanje tekstualnih podataka u JavaScript `Date` objekte.

Za realizaciju opisanog primera, biće korišćen sledeći JSON:

```
[
    {
        "username": "user1",
        "password": "gf^^$%bb%4",
        "dateTimeCreated": "1587290213832"
    },
    {
        "username": "user2",
        "password": "*(&%$$fff33ff",
        "dateTimeCreated": "1382290314831"
    }
]
```

Reč je o JSON tekstu kojim se prikazuje niz korisničkih naloga. Svaki korisnički nalog određen je korisničkim imenom (`username`), lozinkom (`password`) i vremenom kreiranja naloga (`dateTimeCreated`). Podaci su striktno demonstrativni i maksimalno uprošćeni, pa se tako lozinke prikazuju u izvornom obliku, kao običan tekst, što se u produkciji nikada ne praktikuje.

Za nas je u ovom trenutku posebno značajan podatak `dateTimeCreated`, koji čuva vreme kreiranja naloga i to korišćenjem Unix vremena. Broj sekundi od početka Unix epohe u JSON formatu je prikazan kao običan tekst. Mi ćemo u nastavku kreirati funkcionalnost koja će vreme u ovakvom formatu prilikom parsiranja pretvoriti u objektni `Date` JavaScript oblik:

```
var json =
'[{"username":"user1","password":"gf^^$%bb%4","dateTimeCreated":"1587290
213334"},{"username":"user2","password":"*(&%$$fff33ff","dateTimeCreated
":"1382290314543"}]';

var data = JSON.parse(json, (key, value) => {
    if (key === "dateTimeCreated") {
        value = new Date(parseInt(value));
    }
    return value;
});
```

Sada je prilikom pozivanja metode `parse()` njoj prosleđen i drugi parametar, odnosno referenca na jednu Arrow funkciju. Reč je o *reviver funkciji* kojom se unutar parsiranih podataka pronalaze sva svojstva `dateTimeCreated` i njihova vrednost se prevodi u `Date` objekte.

Osobine reviver funkcije

Funkcija reviver može da prihvati dva ulazna parametra – `key` i `value`. Ovi parametri se automatski popunjavaju vrednostima, pri čemu se `key` odnosi na naziv svojstva, a `value` na njegovu vrednosti.

Reviver funkcija aktivira se nakon parsiranja, ali pre nego što metoda `parse()` isporuči svoju povratnu vrednost. Takođe, funkciji reviver prosleđuje se svaka prosta vrednost (`number`, `string`, `boolean`) koja se dobije parsiranjem. Ukoliko je reč o vrednosti nekog objektnog svojstva, parametrom `key` dostavlja se i naziv takvog svojstva. Ukoliko se radi o vrednosti koja predstavlja član niza, parametar `key` će imati `string` vrednost indeksa na kome se takva vrednost nalazi. Na kraju, ukoliko je reč o prostoj vrednosti koja se ne nalazi ni unutar objekta, ali ni unutar niza, vrednost parametra `key` će biti prazan `string`.

Unutar reviver funkcije moguće je obaviti transformaciju prosleđene vrednosti, na osnovu sopstvenih potreba. U prikazanom primeru, takva transformacija podrazumeva konvertovanje Unix vremena iz `string` oblika u JavaScript `Date` format.

Nakon obavljanja transformacija, veoma je bitno obaviti emitovanje transformisane vrednosti kao povratne vrednosti reviver funkcije. Ukoliko se ne obavi emitovanje povratne vrednosti, ili se kao povratna vrednost izbaci `undefined`, vrednost će zajedno se eventualnim pripadajućim svojstvom biti uklonjena iz finalnog skupa vrednosti. Stoga je veoma bitno obaviti emitovanje čak i onih vrednosti koje se ne transformišu. U primeru su to sve vrednosti osim onih koje pripadaju ključevima `dateTimeCreated`.

Upravo prikazanim primerom, koji je podrazumevao korišćenje reviver funkcije, demonstrirano je obavljanje konverzije podataka iz `string` oblika u objektni JavaScript tip `Date`. Tako će svojstvo `dateTimeCreated`, koje će nakon parsiranja postojati unutar objekata koji predstavljaju korisnike, čuvati referencu na podatak tipa `Date`. Da reviver funkcija nije navedena, takvi podaci bi bili u `string` obliku.

Rad sa datumom i vremenom prilikom korišćenja JSON-a

Već više puta je rečeno da JSON izvorno ne poznaje pojam datuma i vremena. Stoga je datum i vreme u JSON-u neophodno prikazati korišćenjem jednog od dva prosta tipa podataka, odnosno kao tekst ili kao broj. Naravno, datum i vreme je tokom parsiranja neophodno na neki način konvertovati u objektni oblik; za obavljanje takvog posla, moguće je koristiti pristup koji je prikazan u prethodnom primeru (reviver funkciju).

Prilikom obavljanja serijalizacije, datum i vreme je iz objektnog oblika neophodno konvertovati u tekst ili broj. Za obavljanje takvog posla `Date` objekti poseduju ugrađenu metodu `toJSON()`, koja emituje tekstualnu reprezentaciju `Date` objekta, koja je pogodna za prikaz u JSON formatu. To ćete, između ostalog, moći da vidite u narednoj lekciji, u kojoj ćemo se baviti procesom serijalizacije JavaScript objekata u JSON format.

Rezime

- U JSON formatu se mogu predstaviti sledeći JavaScript tipovi podataka: `Object`, `number`, `string`, `boolean`, `Array` i `null`.
- JSON podrazumevano ne podržava neke JavaScript kompleksne podatke, kao što su funkcije, regularni izrazi i datumi.
- Svaki tekst koji je formiran korišćenjem JSON pravila ujedno je i validan JavaScript izraz.
- Manipulacija podacima u JSON obliku omogućena je na samom jezičkom nivou JavaScripta.
- JavaScript poseduje globalno dostupan objekat `JSON`, koji je moguće koristiti za parsiranje i serijalizaciju JSON podataka.
- Parsiranje JSON-a podrazumeva konvertovanje podataka iz JSON tekstualnog formata u JavaScript vrednost ili objekat.
- `JSON` objekat poseduje metodu `parse()`, kojom je moguće obaviti parsiranje JSON teksta.
- Ukoliko `string` koji se prosleđuje metodi `JSON.parse()` nije validan JSON, ona emituje izuzetak tipa `SyntaxError`.
- Metoda `JSON.parse()` kao svoj drugi parametar može da prihvati reviver funkciju, kojom se definiše logika za transformisanje parsiranih vrednosti.
- Reviver funkcija aktivira se nakon parsiranja, ali pre nego što metoda `parse()` isporuči svoju povratnu vrednost.