Asocijativni nizovi

Prethodne lekcije donele su priču o jednodimenzionalnim i višedimenzionalnim, indeksiranim nizovima, koji se u JavaScript jeziku predstavljaju tipom `Array`. Indeksirani nizovi su oni kod kojih se članovima pristupa korišćenjem numeričkih, celobrojnih vrednosti, koje se drugačije nazivaju indeksi. Indeksiranje nizova započinje od nule, pa tako prvi element niza ima indeks 0.

Pored indeksiranih nizova, u raznim programskim jezicima postoje i nizovi koji se zovu asocijativni. Kod takvih nizova članovi se obeležavaju specijalnim oznakama, koje se nazivaju ključevi. U lekciji pred vama će biti reči o ovom pojmu iz ugla programskog jezika JavaScript.

Pojam asocijativnih nizova

Asocijativni nizovi, za razliku od indeksiranih, omogućavaju obeležavanje elemenata korišćenjem specijalnih odrednica, koje se nazivaju ključevi. Tako asocijativni nizovi nisu ograničeni na celobrojne, numeričke vrednosti za pristupanje elementima nizova. Asocijativni nizovi su zapravo **parovi ključeva i vrednosti**.

Kako bi se na pravi način razumela osnovna osobina asocijativnih nizova, u nastavku će biti prikazane razlike u pristupanju elementima indeksiranih i asocijativnih nizova.

Pristup elementu kod indeksiranih nizova:

```
arr[1]
```

Pristup elementu kod asocijativnih nizova:

```
arr['name']
```

Analizom načina koji se koriste za pristup elementima indeksiranih i asocijativnih nizova moguće je uvideti osnovnu razliku između ove dve vrste nizova. Asocijativni nizovi omogućavaju da se elementima pristupa korišćenjem ključeva (u primeru je to tekstualni ključ `name`).

Zbog čega su nam potrebni asocijativni nizovi?

Asocijativni nizovi mogu biti vrlo korisni prilikom modelovanja tabelarnih podataka (slika 17.1).

0	1	2	3
1	john	04as89...	john@mail.com

id	username	password	email
1	john	04as89...	john@mail.com

Slika 17.1. Jedna ista tabela sa dva različita načina za imenovanje kolona

Na slici 17.1. prikazana je jedna tabela u dve različite varijante. Prvo je navedena tabela sa kolonama koje su označene brojevima 0, 1, 2 i 3. Druga varijanta ove tabele poseduje kolone sa razumljivijim oznakama. Razlike između ove dve varijante jedne iste tabele upravo su i razlike koje postoje između indeksiranih i asocijativnih nizova. Tabela sa numeričkim kolonama asocira na indeksirani niz, a ona sa tekstualnim nazivima kolona na asocijativni.

Iz prikazanog primera potpuno je jasno da je obeležavanje podataka korišćenjem naziva kolona mnogo jednostavnije i razumljivije. Tako su asocijativni nizovi veoma moćan alat za modelovanje podataka u tabelarnom obliku, kada je neophodno podatke imenovati na razumljiv način.

Asocijativni nizovi u JavaScriptu

Kreiranje asocijativnih nizova u JavaScriptu nešto je drugačije nego što je to bio slučaj sa indeksiranim nizovima. Štaviše, asocijativni nizovi u JavaScriptu se ne mogu kreirati korišćenjem objekta `Array`, zato što ovaj objekat omogućava kreiranje isključivo indeksiranih nizova. Upravo zbog toga je za kreiranje asocijativnih nizova u JavaScriptu neophodno koristiti neke druge pristupe, odnosno druge vrste objektnih podataka:

- `Object`
- `Map`.

Upravo su navedena dva tipa objektnih podataka koji postoje u JavaScriptu, a koji se mogu koristiti za kreiranje asocijativnih nizova.

U više navrata do sada je ilustrovano da je objekat centralna figura JavaScript jezika. JavaScript objekti su ništa drugo do parovi ključeva i vrednosti, kod kojih su svojstva ključevi, a vrednosti takvih svojstava ujedno i vrednosti ključeva.

Pored osnovnog objektnog tipa `Object`, za kreiranje asocijativnih nizova u JavaScriptu je moguće koristiti još jedan specijalni objekat – `Map`. Za razliku od objekta `Object`, čija namena je univerzalna, `Map` se isključivo može koristiti za predstavljanje podataka u formi parova ključeva i vrednosti.

O dva upravo navedena pristupa za kreiranje asocijativnih nizova biće reči u redovima koji slede.

Asocijativni nizovi korišćenjem objekta Object

Najjednostavniji način za dobijanje asocijativnog niza jeste korišćenje objektnog literala u jednoj naredbi:

```
var user = { "id": 1, "username": 'john', "password":
'04as89v4v', "email": 'john@gmail.com' };
```

Na ovaj način podaci o jednom korisniku smešteni su unutar objekta, koji oponaša osobine asocijativnog niza. Podaci su modelovani u obliku parova ključeva i vrednosti, između kojih se nalazi karakter dve tačke (:). Tako je na primer `username` ključ, a `john` vrednost.

Naravno, tip promenljive `user` jeste `Object`, s obzirom na to da je prikazana naredba, klasičan pristup za kreiranje objekata:

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

Rezultat ovakve naredbe jeste ispis tipa unutar konzole – `Object`. S obzirom na to da je ovako kreirani asocijativni niz tipa `Object`, nad njim nije moguće koristiti ugrađena svojstva i metode objekta `Array`, koje su prikazane u prethodnim lekcijama:

```
user.length;
```

Ovakva naredba proizvodi rezultat `undefined`, zato što se svojstvo `length` ne može koristiti nad tipom `Object`, već samo nad tipom `Array`.

Asocijativne nizove tipa `Object` moguće je kreirati i korišćenjem ključne reči `new` i konstruktorske funkcije `Object()`:

```
var user = new Object();
user["id"] = 1;
user["username"] = 'john';
user["password"] = '04as89v4v';
user["email"] = 'john@gmail.com';
```

Sada je asocijativni niz kreiran tako što je prvo obavljeno kreiranje objekta tipa `Object`, a zatim su definisani parovi ključeva i vrednosti.

S obzirom na to da su ovakvi asocijativni nizovi objekti, to praktično znači da su svojstva takvih objekata ujedno i ključevi:

```
var user = new Object();
user.id = 1;
user.username = 'john';
user.password = '04as89v4v';
user.email = 'john@gmail.com';
```

Elementima ovakvih asocijativnih nizova je moguće pristupati na jedan od sledećih načina:

```
console.log(user['username']);
```

ili

```
console.log(user.username);
```

Prolazak kroz asocijativne nizove tipa Object

U prethodnoj lekciji već su ilustrovani različiti pristupi za prolazak kroz indeksirane nizove. Za prolazak kroz upravo ilustrovane asocijativne nizove moguće je koristiti jednu petlju koja je specijalno napravljena za objekte tipa `Object`. Reč je o petlji `for...in`.

```
for (property in object) {
statements
}
```

Ukoliko se objekti posmatraju kao asocijativni nizovi, petlja `for...in` se koristi za prolazak kroz ključeve asocijativnih nizova:

```
var user = { "id": 1, "username": 'john', "password":
'04as89v4v', "email": 'john@gmail.com' };

for (property in user) {
console.log(property);
}
```

Na ovaj način unutar konzole web pregledača biće ispisani nazivi svih svojstava objekta `user`, odnosno nazivi svih ključeva ukoliko se takav objekat posmatra kao asocijativni niz:

```
id
username
password
email
```

Nazivi svojstava, odnosno ključeva, mogu se upotrebiti i za dobijanje vrednosti:

```
var user = { "id": 1, "username": 'john', "password":
'04as89v4v', "email": 'john@gmail.com' };

for (property in user) {
console.log(user[property]);
}
```

Sada su nazivi ključeva iskorišćeni za dobijanje vrednosti. Promenljiva, koja predstavlja ključ u svakoj iteraciji (`property`), postavljena je unutar uglastih zagrada. Tako će prilikom svake iteracije biti odštampana vrednost po jednog svojstva `user` objekta:

```
1
john
04as89v4v
john@gmail.com
```

Asocijativni nizovi korišćenjem Map objekta

U dosadašnjem toku lekcije ilustrovan je jedan pristup za dobijanje asocijativnih nizova u JavaScriptu. Pristup je podrazumevao korišćenje osnovnog JavaScript objekta – `Object`. Ipak, bitno je znati da je prikazani pristup samo način na koji se mogu oponašati osobine asocijativnih nizova, s obzirom na to da je `Object` objekat opšte namene koji nije specijalno namenjen radu sa asocijativnim nizovima. Sa druge strane, JavaScript poseduje i jedan poseban, ugrađeni objekat, koji je moguće koristiti za kreiranje asocijativnih nizova. Reč je o objektu `Map`.

`Map` je jedan od ugrađenih objekata JavaScript jezika, koji omogućava čuvanje parova ključeva i vrednosti. Pri tome je objekat `Map` specijalizovan za takvu namenu, za razliku od objekta `Object`, pa poseduje veliki broj svojstava i metoda koji olakšavaju rad sa asocijativnim nizovima.

Objekti `Map` i `Object` veoma su slični. Za početak, oba objekta omogućavaju definisanje parova ključeva i vrednosti, što ih kvalifikuje za modelovanje asocijativnih nizova. Ipak, u starijim verzijama JavaScripta, `Map` objekat nije postojao, pa je jedini način za kreiranje asocijativnih nizova bio korišćenje objekta `Object`. Sa pojavom objekta `Map` za takvim nečim više nema potrebe, zato što je ovaj objekat specijalno namenjen za kreiranje asocijativnih nizova. Prednosti korišćenja `Map` objekta su:

- ključevi mogu biti bilo kog tipa (kod tipa `Object` ključevi moraju biti `String` ili `Symbol` tipa),
- objekat `Map` poseduje brojna svojstva i metode koji su specijalno namenjeni radu sa asocijativnim nizovima, na primer kako bi se dobila dužina asocijativnog niza kreiranog `Map` objektom, dovoljno je iskoristiti svojstvo `size`,
- objekat `Map` je iterabilan, što nije slučaj sa objektom `Object`, upravo zbog toga je nad objektima `Map` tipa moguće koristiti `for...of` petlju, a takvu pogodnost ne nudi `Object` tip,
- `Map` objekat obezbeđuje bolje performanse u situacijama čestog dodavanja i uklanjanja elemenata.

Kreiranje Map objekta

Asocijativni niz u obliku `Map` objekta se može kreirati na dva načina. Najjednostavniji način jeste kreiranje u jednoj naredbi, slično definisanju objektnog literala:

```
let map = new Map([
['id', 1],
['username', 'john'],
['password', '04as89v4v'],
 ['email', 'john@gmail.com']
]);
```

Drugi način za kreiranje `Map` asocijativnog niza jeste zasebno kreiranje `Map` objekta, a zatim dodavanje elemenata korišćenjem metode **set()**:

```
let map = new Map();

map.set('id', 1);
map.set('username', 'john');
map.set('password', '04as89v4v');
map.set('email', 'john@gmail.com');
```

Čitanje vrednosti Map objekta

Čitanje vrednosti elemenata se postiže korišćenjem metode **get()**:

```
console.log(map.get('username'));
```

Sasvim očekivano, ovakva naredba unutar konzole proizvodi ispis:

```
John
```

Dobijanje dužine Map objekta

Kako bi se dobila dužina asocijativnog niza predstavljenog `Map` objektom, koristi se svojstvo **size**:

```
console.log(map.size);
```

Na ovaj način se u konzoli dobija ispis:

```
4
```

Pretraga ključeva Map objekta

`Map` objekat poseduje specijalnu metodu za obavljanje pretrage ključeva. Reč je o metodi

has():

```
console.log(map.has('username'));
```

Na ovaj način će biti obavljena pretraga ključa `username`. Ukoliko takav ključ postoji unutar objekta `map`, metoda `has()` vraća `boolean` vrednost `true`. U protivnom, ova metoda emituje `false`. Stoga prikazana naredba proizvodi ispis:

```
true
```

Brisanje elemenata Map objekta

`Map` objekat omogućava i lako brisanje elemenata korišćenjem metode **delete()**:

```
let map = new Map();

map.set('id', 1);
map.set('username', 'john');
map.set('password', '04as89v4v');
map.set('email', 'john@gmail.com');

map.delete('username');

console.log(map.has('username'));
```

U primeru je, nakon kreiranja i popunjavanja `Map` objekta, obavljeno brisanje vrednosti sa ključem `username`. Da li je takva vrednost zaista i obrisana proverava se poslednjom naredbom, kojom se vrši pretraga vrednosti sa obrisanim ključem. Naravno, unutar konzole se dobija vrednost `false`.

Ključevi unutar Map objekta mogu biti bilo kog tipa

Nešto ranije je rečeno da ključevi unutar `Map` objekta mogu biti bilo kojeg tipa. Tako je potpuno legitimno kao ključ postaviti jedan objekat:

```
let user = { "id": 1, "username": 'john', "password":
'04as89v4v', "email": 'john@gmail.com' };

let map = new Map();
map.set(user, 13);

console.log(map.get(user));
```

Sada je prvo kreiran jedan objekat (`user`). Nakon objekta, kreirana je i jedna mapa. Unutar mape smešten je jedan element. Ključ je kreirani objekat (`user`), a vrednost je 13. Poslednja linija ilustruje dolazak do vrednosti 13, korišćenjem objektnog ključa, pa se na kraju unutar konzole dobija ispis:

```
13
```

Pitanje

Kako bi se dobila dužina asocijativnog niza kreiranog Map objektom, dovoljno je iskoristiti svojstvo:

- **size**
- length
- total
- num

Objašnjenje:

Objekat Map poseduje brojna svojstva i metode koje su specijalno namenjene radu sa asocijativnim nizovima. Na primer, kako bi se dobila dužina asocijativnog niza kreiranog Map objektom, dovoljno je iskoristiti svojstvo size.

Prolazak kroz Map objekat

Nešto ranije je već rečeno da je `Map` jedan od iterabilnih tipova. Takav tip je i `Array`, što je upravo osobina koja ovim tipovima omogućava da budu korišćeni unutar `for...of` petlji. Stoga, ukoliko je potrebno proći kroz sve elemente jednog `Map` objekta, dovoljno je napisati:

```
let map = new Map();

map.set('id', 1);
map.set('username', 'john');
map.set('password', '04as89v4v');
map.set('email', 'john@gmail.com');

for (let [key, value] of map) {
console.log(key + ': ' + value)
}
```

Na ovaj način se unutar konzole dobija:

```
id: 1
username: john
password: 04as89v4v
email: john@gmail.com
```

Za ispis svih ključeva i vrednosti u prikazanom primeru je iskorišćena `for...of` petlja, unutar koje je obavljena deklaracija dve promenljive – `key` i `value`. Deklaracija ovih promenljivih je obavljena unutar uglastih zagrada, te su na taj način unutar tela petlje dostupni i ključevi i vrednosti. Ovakav pristup za prolazak kroz asocijativni niz moguć je zato što je `Map` jedan od iterabilnih tipova.

Prikazani pristup za dobijanje ključeva i vrednosti `Map` objekta zapravo je skraćeni oblik, koji podrazumeva korišćenje metode **entries():**

```
let map = new Map();

map.set('id', 1);
map.set('username', 'john');
map.set('password', '04as89v4v');
map.set('email', 'john@gmail.com');

for (let [key, value] of map.entries()) {
console.log(key + ': ' + value)
}
```

Sada je bitno da primetite da su parovi ključeva i vrednosti dobijeni korišćenjem metode `entries()`. Efekat je potpuno identičan prethodnom primeru, tako da je bez problema moguće izostaviti poziv ove metode, jer se ona svakako poziva kada se navede naziv promenljive koja predstavlja `Map` objekat.

Pored metode `entries()`, `Map` objekat poseduje još dve metode koje je moguće koristiti za dobijanje ključeva i vrednosti, pojedinačno. Za dobijanje ključeva koristi se metoda **keys()**:

```
let map = new Map();

map.set('id', 1);
map.set('username', 'john');
map.set('password', '04as89v4v');
map.set('email', 'john@gmail.com');

for (let key of map.keys()) {
console.log(key)
}
```

Ovoga puta je prilikom formiranja `for...of` petlje iskorišćena metoda `keys()`, kojom se dobijaju svi ključevi `Map` objekta. Izmenjena je i deklaracija, pa sada postoji samo jedna promenljiva sa nazivom `key`, koja će u svakoj iteraciji poprimati vrednosti narednog ključa:

```
id
username
password
email
```

Za dobijanje svih vrednosti moguće je koristiti metodu **values()**:

```
let map = new Map();

map.set('id', 1);
map.set('username', 'john');
map.set('password', '04as89v4v');
map.set('email', 'john@gmail.com');

for (let value of map.values()) {
console.log(value)
}
```

Poziv metode `keys()` zamenjen je metodom `values()`, koja isporučuje niz vrednosti `Map` objekta. Takođe, naziv promenljive je promenjen u `values`, a unutar konzole se dobija sledeće:

```
1
john
04as89v4v
john@gmail.com
```

Rezime

- Indeksirani nizovi su oni kod kojih se članovima pristupa korišćenjem numeričkih, celobrojnih vrednosti, koje se drugačije nazivaju indeksi.
- Asocijativni nizovi, za razliku od indeksiranih, omogućavaju obeležavanje elemenata korišćenjem specijalnih odrednica koje se nazivaju ključevi.
- Asocijativni nizovi su parovi ključeva i vrednosti.
- Asocijativni nizovi u JavaScriptu se ne mogu kreirati korišćenjem objekta `Array`.
- Asocijativni nizovi se u JavaScriptu mogu kreirati korišćenjem objekata `Object` i `Map`.
- Za prolazak kroz asocijativne nizove tipa `Object` moguće je koristiti petlju `for...in`.
- `Map` je jedan od ugrađenih objekata JavaScript jezika, koji omogućava čuvanje parova ključeva i vrednosti.
- `Map` objekat je, za razliku od objekta `Object`, specijalno namenjen za predstavljanje parova ključeva i vrednosti.
- `Map` je iterabilan objekat, pa ga je moguće koristiti u kombinaciji sa `for...of` petljom.