JavaScript objekti

Autori:
Milan Segedinac
Goran Savić

1. Objekti

U programskom jeziku JavaScript ima šest primitivnih tipova podataka (`Number`, `String`, `Boolean`, `Null`, `Undefined` i , od verzije ES6 `Symbol`). Svi ostali tipovi su objekti. Objekti u programskom jeziku JavaScript su kolekcije parova atribut-vrednost, i već smo videli kako možemo da postavljamo vrednosti atributa, menjati ih i brisati.

U programskom jeziku JavaScript objekat je moguće kreirati na tri načina:

1. Pomoću objektnog literala
2. Pomoću operatora `new`
3. Pozivom `Object.create` funkcije.

Objektni literali

Objektni literal je izraz koji se interpretira i kao vrednost vraća objekat. Kreiranje objekta pomoću objektnog literala je najjednostavnije: parovi atribut-vrednost se zadaju između vitičastih zagrada, razdvajaju zarezima, i zadaju u formatu atribut:vrednost. Atribut mora da bude string, a vrednost može da bude bilo koji JavaScript izraz. Prilikom kreiranja objekta, izraz se evaluira i dobijena vrednost se postavlja na atribut. Primer ovako kreiranog objekta prikazan je listingom ispod.

```
osoba = {

  ime:"Pera",
  prezime:"Peric",
  godiste:1980,
  mestoRodjenja:{
    naziv: "Novi Sad",
    drzava: "Srbija"
  }
}
```

Listing - JavaScript objekat kreiran objektnim literalom

U ovom slučaju kreirali smo objekat koji reprezentuje podatke o osobi i njegov atribut `ime` ima vrednost `'Pera'`, atribut `prezime` ima vrednost `'Perić'`, a atribut `godiste` ima vrednost `1980`. Ništa nas ne sprečava da vrednost atributa objekta bude takođe objekat. U primeru sa listinga iznad atribut `mestoRodjenja` ima vrednost koja je objekat, koji opet ima svoje atribute (`naziv` i `drzava`) i vrednosti (`'Novi Sad'` i `'Srbija'`).

Prototip

Iako je JavaScript objektno orjentisan programski jezik, ovaj programski jezik ne podržava pisanje klasa. U „klasičnim“ objektno orjetisanim programskim jezicima pomoću klasa se zadaje šablon kreiranja objekata uz mogućnost kreiranja zajedničkih funkcionalnosti (metoda). Pored toga hijerarhija objekata zadavala se pomoću nasleđivanja. Setimo se da smo u programskom jeziku Java pretku objekta mogli da pristupimo pomoću ključne reči `super`. U programskom jeziku JavaScript ovi problemi su rešeni uvođenjem *prototipa*. U

tom programskom jeziku svaki objekat, pored atributa koje sami zadamo ima i atribut `__proto__` čija vrednost je prototip objekta. Svi atributi i metode prototipa direktno su dostupni i u objektu.

Kada objekte kreiramo pomoću objektnih literala prototip za prototip kreiranog objekta se postavlja `Object.prototype`. Nad bilo kojim objektom kreiranim pomoću objektnog literala možemo pozvati funkciju `toString`, jer je ova funkcija definisana u `Object.prototype`, kao što je prikazano na slici ispod.

Slika – Object.prototype

Slika iznad je snapshot konzole. Kreirali smo objekat osoba. Zatim smo pozvali fukciju `toString()` tog objekta. Prikaz atributa `__proto__` ovog objekta pokazuje da je ta metoda definisana u prototipu.

Obratite pažnju da metodu toString nismo morali da pozovemo sa `osoba.__proto__.toString()`, već smo je pozvali sa `osoba.toString()`.

Treba napomenuti da se, prilikom kreiranja niza pomoću literala (uglastih zagrada) za prototip novokreiranog niza postavlja `Array.prototype`, a kada se kreira funkcija pomoću funkcijskog literala njen prototip je `Function.prototype`.

U programskom jeziku JavaScript veza između objekta i njegovog prototipa je po referenci. To znači da, ako nekoliko objekata ima isti prototip, referenciraće isti objekat. Na primer, svi objekti kreirani pomoću objektnih literala na atributu `__proto__` će imati

referencu na isti objekat. U programskom jeziku JavaScript ne postoji zabrana da objekat izmeni svoj prototip. Pošto objekat ima referencu na svoj prototip, ovakva izmena odražava se i na sve ostale objekte koji imaju isti prototip. Primer je dat slikom ispod.

Slika – izmena prototipa

Na slici iznad dat je prikaz konzole u kom smo kreirali objekat `osoba` i objekat `grad`. U objektu `osoba` smo izmenili funkciju `toString` (postavili smo da uvek vraća vrednost `'Hello world'`). Obzirom da objekti `osoba` i `grad` imaju isti prototip (`Object.prototype`) izmena se odrazila i na objekat `grad`, pa je poziv funkcije `toString` ovog objekat vratio string `'Hello world'`.

Konstruktorske funkcije

Konstruktorska funkcija nam omogućuju da kreiramo objekat sa zadatim prototipom. Pozivu konstruktorske funkcije prethodi ključna reč `new`. Pozivom ove funkcije kreira se objekat čija vrednost atributa `__proto__` postaje objekat postavljen na atribut `prototype` konstruktorske funkcije. Kod konstruktorskih funkcija susrećemo se sa još jednim veoma važnim konceptom u programskom jeziku JavaScript: varijablom `this`, koja je u konstruktorskoj funkciji referenca na kreirani objekat. Primer konstruktora je dat ispod.

```
var Brojac = function (startnaVrednost) {

      this.vrednost = startnaVrednost;
};
Brojac.prototype.uvecaj = function(){
      this.vrednost += 1;
};

var b1 = new Brojac(5);
var b2 = new Brojac(10);
b1.uvecaj();
b2.uvecaj();
console.log('b1.vrednost: '+b1.vrednost);//6
console.log('b2.vrednost: '+b2.vrednost);//11
```

Listing - konstruktorska funkcija

Na listingu iznad kreirali smo konstruktorsku funkciju `Brojac` i za atribut `vrednost` objekat koji će ona kreirati postavili prosleđenu vrednost (`startnaVrednost`). U prototip konstruktorske funkcije `Brojac` postavili smo funkciju koja atribut `vrednost` uvećava za jedan. Zatim smo kreirali dva objekta za različite stratne vrednosti i nad njima pozvali funkcije `uvecaj`. Na kraju smo u konzolu ispisali uvećane vrednosti dva brojača.

Obratite pažnju da nigde nismo specificirali da je funkcija konstruktorska. U stvari, razlika između konstruktorskih i „običnih“ funkcija je samo u načinu invokacije: konstruktorske funkcije pozivamo sa ključnom rečju `new` ispred poziva funkcije, ali tako možemo pozvati i bilo koju „običnu“ funkciju. Ako bismo to uradili, kreirao bi se novi objekat, `this` bi u funkciji ima vrednost novokreiranog objekta i atribut `__proto__` novokreiranog objekat dobio bi vrednost atributa `prototype` funkcije. Činjenica da ne postoji sintaktička razlika između konstruktorskih i „običnih“ funkcija čest je izvor veoma ozbiljnih bugova. Zbog toga se preporučuje pravilo da se nazivi konstruktorskih funkcija počinju velikim slovom, a „običnih“ funkcija malim.

Object.create

U programskom jeziku JavaScript objekte je moguće kreirati i pozivom `Object.create` funkcije. Ova funkcija kreira i vraća objekat koji kao prototip ima vrednost prosleđenu ovoj funkciji. Primer kreiranja objekta pomoću `Object.create` funkcije dat je listingom ispod.

```
peraPeric = {

      ime:'Pera',
      prezime:'Peric',
      brojLicneKarte:123321
}
studentPeraPeric = Object.create(peraPeric);
studentPeraPeric.brojIndeksa = 123;
studentPeraPeric.godinaStudija = 2;
```

Listing - Object.create

Na listingu iznad kreiran je objekat `peraPeric` koji ima atribute `ime`, `prezime` i `brojLicneKarte`. Zatim je kreiran novi objekat `studentPeraPeric` koji kao prototip ima objekat `peraPeric`, a na atributima `brojIndeksa` i `godinaStudija` ima vrednosti `123` i `2`.

Obratite pažnju da pomoću `Object.create` možemo da pravimo lance prototipova. Objekat `peraPeric` kao svoj prototip ima `Object.prototype` (jer je kreiran pomoću objektnog literala), a objekat `studentPeraPeric` kao prototip ima objekat `peraPeric`.

Šta se dešava kada pokušamo da pristupimo atributu objekta? Kada pristupamo atributu objekta provo se proveri da li postoji u samom objektu. Ako ne postoji, proveri se da li postoji u prototipu. Ako ne postoji proveri se da li postoji u prototipu prototipa, i tako do kraja lanca prototipova. Tek ako ne postoji nigde u lancu prototipova, konstatuje se da je nedefinisan.

Šta se dešava kada pokušamo da izmenimo vrednost atributa? Ako pristupimo atributu objekta, ako postoji, izmeni se vrednost, a ako ne postoji postavi se nova vrednost tog atributa objekta. Ako želimo da menjano vrednost u prototipu moramo eksplicitno da pristupimo atributu `__proto__`.

U programskom jeziku Java sve klase su naslednice klase `Object`. U JavaScriptu možemo i da kreiramo objekte koji nemaju prototip - prosto bismo funkciji `Object.create` prosledili `null`.