Objekti

U lekciji o tipovima podataka prvi put je spomenut pojam JavaScript objekta. Tada su definisane neke osnovne osobine ovog veoma značajnog leksičkog elementa. Takođe, u lekcijama za nama već je korišćeno nekoliko objekata koji su sastavni deo JavaScript jezika. Ipak, sve do sada objektima nije bila posvećena puna pažnja.

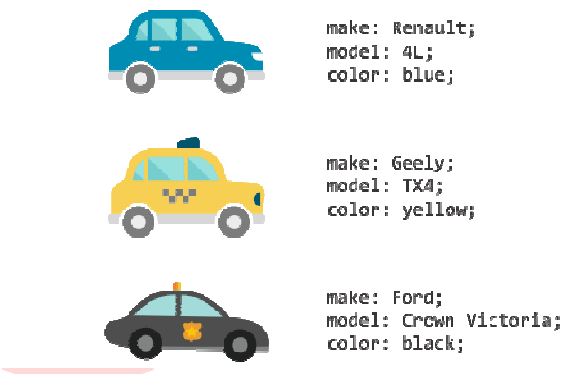
S obzirom na to da je JavaScript objektno orijentisan jezik, a njegova centralna figura objekat, lekcija pred vama biće posvećena osnovnim osobinama JavaScript objekata.

Šta je objekat?

Osnovni gradivni element jezika JavaScript jeste objekat. Objekat je zapravo tip podatka i upravo zbog toga je spomenut u lekciji o tipovima. Ipak, za razliku od primitivnih tipova (`string`, `number`, `boolean`, `null`, `undefined`...), objekat je složeni tip podatka, koji unutar sebe može imati veći broj različitih vrednosti.

Za objekte se veoma često kaže da su programske reprezentacije nekih entiteta. Entiteti su pojave iz realnog sveta – osoba, država, student, olovka, automobil, kompjuter... Stoga su objekti način da se ovakvi i slični entiteti definišu prilikom pisanja JavaScript koda.

Primer jednog entiteta koji se može prevesti u objekat jeste automobil. Automobil može imati razne karakteristike, kao što su proizvođač, model, zapremina motora, težina, boja i slično. Sve su to kandidati za skup vrednosti koje se mogu naći unutar jednog objekta. Različiti objekti entiteta *automobil* prikazani su slikom 8.1.



Slika 8.1. Primeri objekata

Na slici 8.1. prikazano je nekoliko automobila. Svaki od njih ima svoje specifične karakteristike, ali je skup njihovih osobina identičan. Tako je svaki od prikazanih automobila – objekat određen sledećim karakteristikama:

- naziv proizvođača,
- naziv modela,
- boja karoserije.

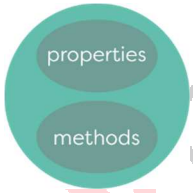
Skup karakteristika, kao što su: *proizvođač*, *naziv*, *boja*, *motor*, *broj vrata*, zajednički se može nazvati šablon ili prototip. Na osnovu takvog prototipa dobijaju se konkretni objekti sa svojim specifičnim karakteristikama (boja žuta, proizvođač Geely, model TX4…).

Objektni članovi

Objekti su skupovi različitih vrednosti koje opisuju jedan isti pojam. *Proizvođač*, *model*, *boja* i *težina* opisuju jedan automobil. Navedene osobine unutar jednog objekta se nazivaju **objektna svojstva** (engl. *object properties*).

Pored svojstava, objekti mogu da enkapsuliraju i različita ponašanja, iskazana kroz funkcije jezika JavaScript. Funkcija koja se nalazi unutar nekog objekta drugačije se naziva **metoda**. Tako objekat koji predstavlja neki automobil može imati metode za pokretanje motora, zaustavljanje, kočenje, promenu brzina.

Može se rezimirati da su dva osnovna elementa JavaScript objekata – svojstva i metode (slika 8.2).



Slika 8.2. Objekti se sastoje iz svojstava i metoda

U nastavku će prvo biti ilustrovani različiti načini za kreiranje objekata sa nekoliko svojstava, dok će nakon toga biti prikazano i kako se objektima dodeljuju funkcije koje postaju objektne metode.

Kreiranje objekata

U JavaScript jeziku objekat može biti kreiran na nekoliko različitih načina:

- korišćenjem objektnog literala,
- korišćenjem konstruktorske funkcije `Object()`,
- korišćenjem proizvoljne konstruktorske funkcije.

Kreiranje objekata korišćenjem objektnog literala

Najjednostavniji način za kreiranje objekta i definisanje njegovih svojstava jeste korišćenje objektnog literala. Sledeći primer ilustruje kreiranje objekta subaryLegacy na ovaj način:

```
var subaryLegacy = { make: "Subary", model: "Legacy", weight: 1560,
color: "black" };
```

Primer može da potvrdi konstataciju da su objekti skupovi više svojstava koji opisuju jedan pojam. Pri tome se takva svojstva navode kao parovi ključeva i vrednosti. Ključ je naziv svojstva, dok je vrednost svojstva – vrednost koja se dobija navođenjem ključa.

Kreiranje objekata korišćenjem konstruktorske funkcije Object()

JavaScript omogućava kreiranje objekata korišćenjem specijalne ključne reči **new** u kombinaciji sa posebnom konstruktorskom funkcijom `Object()`, koja je automatski dostupna u JavaScript jeziku.

```
var myCar = new Object();
myCar.make = "Subary";
myCar.model = "Legacy";
myCar.weight = 1560;
myCar.color = "black";
```

Iz primera se može videti da je u prvoj liniji upotrebljena ključna reč `new` ispred posebne konstruktorske funkcije `Object()`, pa je na taj način kreiran objekat. Tako prva naredba prikazanog primera kreira jednu promenljivu tipa `object`, bez nekih posebnih osobina. Može se reći da je prvom naredbom kreiran jedan prazan objekat. Ipak, narednim naredbama kreiranom objektu se dodaju određene osobine, definisanjem svojstava.

U prikazanom primeru se prvi put koristi takozvana **Dot notacija**. Nakon naziva promenljive se navodi tačka, a zatim i naziv svojstva. Na ovaj način je objektima moguće dodeliti nova svojstva, kao što je to urađeno u primeru.

Ilustrovani pristup (dot notaciju) je moguće koristiti i za čitanje vrednosti postojećih svojstava:

```
console.log(myCar.make + " " + myCar.model);
```

Unutar konzole se dobija:

```
Subary Legacy
```

Kreiranje objekata korišćenjem proizvoljne konstruktorske funkcije

Prethodni primer ilustrovao je kreiranje objekta korišćenjem ugrađene konstruktorske funkcije, čime se dobija prazan objekat bez ikakvih predefinisanih osobina. Ipak, veoma često se može javiti potreba za kreiranjem većeg broja objekata koji dele identičan skup karakteristika (na primer, kada je potrebno kreirati veći broj automobila različitih karakteristika). U takvim situacijama je moguće otići korak dalje i u potpunosti samostalno definisati konstruktorsku funkciju za kreiranje objekata.

Korišćenjem samostalno definisane konstruktorske funkcije moguće je kreirati proizvoljan broj objekata koji dele osnovna svojstva. Na primer, svi automobili poseduju identična svojstva koja ih opisuju. Sve je proizveo neki proizvođač, imaju svoje ime, boju, težinu i slično. Ipak, svaki automobil uglavnom ima različite karakteristike, definisane vrednostima zajedničkih svojstava. Jedan automobil je crvene boje, drugi crne, svaki od njih ima svoju težinu, jedinstveno ime i slično.

Kada je potrebno modelovati ovakvu situaciju, odnosno, kada je potrebno napraviti više objekata sa istim svojstvima, najprikladnije je za kreiranje objekata koristiti proizvoljnu konstruktorsku funkciju:

```
function Car(make, model, weight, color) {
this.make = make;
this.model = model;
this.weight = weight;
this.color = color;
}
```

Funkcija sa nazivom `Car` je najobičnija JavaScript funkcija. Ona prihvata četiri (4) ulazna parametra i ne poseduje povratne vrednosti. Ipak, bitno je zapaziti da je ova funkcija imenovana korišćenjem velikog početnog slova. Takvo nešto nije obaveza, ali predstavlja veoma korisnu notaciju, kojom se konstruktorske funkcije distanciraju od *običnih* funkcija.
I unutar tela prikazane konstruktorske funkcije postoji jedna novina. Reč je o upotrebi jedne posebne ključne reči JavaScript jezika – `this`.

Ključna reč this

Ključna reč `this` uvek se odnosi na objekat koji je vlasnik koda u kome se takva reč nalazi. Kada se `this` navede unutar konstruktorske funkcije, kao u primeru, odnosi se na konkretan objekat koji će korišćenjem takve funkcije biti kreiran. Upravo zbog toga vrednosti prosleđene konstruktorskoj funkciji bivaju upisane u svojstva upravo tog (`this`) objekta.

Nakon kreiranja konstruktorske funkcije, nju je veoma lako moguće iskoristiti za kreiranje proizvoljnog broja različitih objekata koji poseduju osobine definisane konstruktorskom funkcijom (*marka*, *model*, *težina* i *boja*):

```
var car1 = new Car("Subary", "Legacy", 1563, "black");
var car2 = new Car("Ford", "Taurus", 1876, "blue");
var car3 = new Car("Porsche", "Panamera", 1963, "grey");
```

U primeru se kreiraju tri objekta, sa identičnim skupom svojstava, ali s različitim vrednostima. Za ovakva tri objekta se može reći da su *istog tipa*. Naravno, samo fiktivno, pošto su svi oni zapravo tipa `Object`.

Osnovna stvar na koju je potrebno obratiti pažnju u prikazanom primeru jeste to da se konstruktorska funkcija `Car()` ne poziva na tradicionalan način, već njenom pozivanju prethodi ključna reč `new`. Na ovaj način izvršnom okruženju je stavljeno do znanja da je potrebno da funkciju tretira kao konstruktorsku. Stoga će, svakim pozivanjem konstruktorske funkcije na ovaj način, biti kreiran novi objekat koji će biti dodeljen odgovarajućoj promenljivoj (`car1`, `car2`…).

Koji pristup za kreiranje objekata treba koristiti?

Potpuno je očekivano da se nakon prethodnih redova pitate koji pristup za kreiranje objekata treba koristiti. Ukoliko je potrebno brzo i jednostavno napraviti objekat, a da je pri tome poznato da neće biti potrebe za kreiranjem većeg broja objekata identičnog skupa karakteristika, najbolje je koristiti objektni literal. Kada unapred znate da će biti potrebe za kreiranjem većeg broja objekata identičnog skupa karakteristika, najbolje je napraviti sopstvenu konstruktorsku funkciju. Korišćenje ugrađene konstruktorske funkcije `Object()` nema neku posebnu upotrebnu vrednost.

Metode

Objektni članovi mogu biti svojstva i metode. Primeri objekata koji su kreirani do sada oslikavali su objekte koji nisu sadržali metode, već isključivo svojstva. Metoda je ništa drugo do funkcija koja se nalazi unutar nekog objekta. I kao što svojstva opisuju objekat i njegove karakteristike, metode govore šta objekat može da uradi i definišu logiku kojom se takvo nešto postiže.

Kako bi se jednom objektu dodala metoda, potrebno je kreirati funkciju unutar takvog objekta i nju dodeliti novoj promenljivoj:

```
var Car = function(make, model, weight, color) {
this.make = make;
this.weight = weight;
this.color = color;

this.startEngine = function(){
return "Engine of " + this.make + " " + this.model + " is
started.";
}
```

Konstruktorskoj funkciji dodata je funkcija koja je dodeljena promenljivoj `startEngine`. Funkcija kao rezultat vraća poruku o tome da je motor određenog automobila startovan.

Sada je moguće kreirati jedan objekat korišćenjem ovakve konstruktorske funkcije i pozvati kreiranu metodu:

```
var car1 = new Car("Subary", "Legacy", 1563, "black");
console.log(car1.startEngine());
```

Kôd proizvodi sledeći efekat:

```
Engine of Subary Legacy is started.
```

Ugrađeni JavaScript objekti

Sve do sada, u ovoj lekciji, bavili smo se korisnički definisanim objektima. Pored korisnički definisanih, JavaScript poznaje i skup ugrađenih objekata koji su na raspolaganju programeru.

Skup osnovnih ugrađenih JavaScript objekata prikazan je tabelom 8.1.

Objekat	Opis
`Object`	osnovni objekat u jeziku JavaScript
`Function`	globalni objekat za kreiranje funkcija
`Boolean`	objektni omotač za boolean prost tip
`Number`	objektni omotač za number prost tip
`Math`	objekat koji sadrži matematičke funkcije i konstante
`Date`	objekat koji se koristi za predstavljanje jednog trenutka u vremenu
`String`	objekat za rad sa tekstom
`RegExp`	objekat za kreiranje regularnih izraza
`Array`	objekat za kreiranje nizova
`Map`	objekat koji predstavlja kolekciju ključeva i vrednosti
`Set`	objekat koji predstavlja kolekciju jedinstvenih vrednosti
`JSON`	objekat koji sadrži logiku za parsiranje JSON-a i konverziju vrednosti u JSON

Tabela 8.1. Osnovni ugrađeni objekti jezika JavaScript

Funkcije su objekti

U JavaScript jeziku, funkcije su još jedna vrsta objekata, tako da se kreiranjem funkcije zapravo kreira objekat. Reč je o objektu koji se zove `Function` i može se kreirati korišćenjem istoimenog konstruktora:

```
var myFunction = new Function('a', 'b', 'return a + b');
```

Ovoga puta funkcija je kreirana u objektnom maniru, korišćenjem konstruktora i ključne reči `new`. Konstruktoru su prosleđeni nazivi promenljivih, koji će predstavljati ulazne parametre funkcije i kôd koji je potrebno izvršiti kada se funkcija pozove. Prva dva parametra se odnose na nazive parametara funkcije, a poslednji parametar na telo funkcije. Ovakvu funkciju je moguće pozvati na sledeći način:

```
myFunction(10, 15);
```

Naravno, povratna vrednost funkcije će biti 25.

Kreiranje funkcija korišćenjem konstruktora i bez korišćenja konstruktora proizvodi identičan efekat kada se govori o logici funkcije. Ipak, funkcije koje se kreiraju korišćenjem objektnog konstruktora parsiraju se kada se funkcija kreira.

Pitanje

U JavaScript jeziku sve je zapravo:

- **objekat**
- funkcija
- procedura
- klasa

Objašnjenje:

JavaScript je jezik kod koga se objekti zasnivaju na drugim objektima, a na vrhu takve hijerarhije se nalazi objekat Object.

Rezime

- Osnovni gradivni element jezika JavaScript jeste objekat.
- Za razliku od primitivnih tipova, objekat je složeni tip podatka koji unutar sebe može imati veći broj različitih vrednosti.
- Objekti su programske reprezentacije entiteta iz realnog sveta.
- Objekti mogu da sadrže svojstva i metode.
- Funkcija koja se nalazi unutar nekog objekta drugačije se naziva metoda.
- Najjednostavniji način za kreiranje objekta i definisanje njegovih svojstava jeste korišćenje objektnog literala.
- JavaScript omogućava kreiranje objekata korišćenjem specijalne ključne reči `new` u kombinaciji sa posebnom konstruktorskom funkcijom `Object()` koja je automatski dostupna u JavaScript jeziku.
- Kreiranje novih svojstava ili pristup postojećim svojstvima moguće je obaviti korišćenjem Dot notacije – nakon naziva promenljive se navodi tačka, a zatim i naziv svojstva.
- Korišćenjem samostalno definisane konstruktorske funkcije moguće je kreirati proizvoljan broj objekata koji dele ista svojstva.
- Nezvanična notacija nalaže da konstruktorske funkcije započinju velikim slovom.
- Ključna reč `this` uvek se odnosi na objekat koji je vlasnik koda u kome se takva reč nalazi.
- U JavaScript jeziku, funkcije su jedna vrsta objekata, tako da se kreiranjem funkcije zapravo kreira objekat.
- Pored korisnički definisanih, JavaScript poznaje i skup ugrađenih objekata, koji su na raspolaganju programeru, a obavljaju različite funkcije.