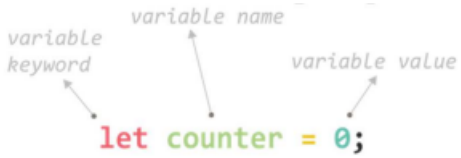
Tipovi podataka

Koncept tipova podataka je nešto što postoji u svakom programskom jeziku, ali se različiti jezici na različite načine odnose prema tipovima. JavaScript je, za razliku od svojih glavnih uzora, slabo tipizirani jezik, odnosno jezik sa dinamičkim tipovima. To praktično znači da JavaScript ne podržava striktno definisanje tipova prilikom deklaracije promenljivih i konstanti. Na primer, deklaracija i inicijalizacija jedne promenljive iz prethodnih lekcija izgledala je kao na sledećoj slici:



Analizom slike može se zaključiti da se, prilikom deklaracije, tip podatka promenljive ni na koji način ne naznačava. Razlog je spomenut nešto ranije – JavaScript poseduje dinamičke tipove. To praktično znači da jedna promenljiva može da prihvati vrednost bilo kog tipa i da će JavaScript izvršno okruženje pokušati da na osnovu vrednosti samostalno utvrdi njen tip.

Tipovi podataka u JavaScriptu

Iako se tipovi podataka eksplicitno ne definišu, to ne znači da oni ne postoje u JavaScript jeziku. Tako JavaScript poznaje sledeće tipove podataka:

- Number
- Bigint
- String
- Boolean
- Null
- Undefined
- Symbol
- Object

Numerički tip podatka – number

Broj je najjednostavniji tipo podatka u JavaScript jeziku. Većina primera do sada ilustrovali su kreiranje jedne promenljive koja je tipa *number*.

Kada se deklariše neka primenljiva i njoj dodeli vrednost nekog broja, može se reći da je kreirana promenljiva tipa *number*.

var x = 10;

U primeru, promenljiva sa nazivom *x* je tipa *number*.

Provera tipa promenljive

JavaScript jezik omogućava veoma laku proveru tipa neke vrednosti, promenljive ili konstante i to korišćenjem operatora **typeof**. Kako bi se proverio i ispisao tip upravo kreirane promenljive, dovoljno je napisati:

```
var x = 10;
document.write(typeof x);
```

Nakon izvršavanja prikazanog koda, unutar HTML dokumenta se dobija sledeći ispis:

```
number
```

Numeričke vrednosti se u JavaScriptu mogu pisati na nekoliko načina: sa decimalama, bez decimala i u eksponencijalnom obliku.

var x1 = 56.00; // with decimals

var x2 = 56; // without decimals

var y = 154e5; // exponential 15400000

var z = 154e-5 // exponential 0.00154

JavaScript poznaje i tri specijalne numeričke vrednosti:

- NaN
- Infinity
- -Infinity

NaN je skraćenica za pojam Not-a-Number, zato što predstavlja vrednost koja se koristi da ukaže da vrednost nije validan broj. Sledeći primer proizvodi *NaN* vrednost:

var x = 0 / 0; // x is NaN

NaN je vrlo karakteristična vrednost, koja je, i pored toga što ukazuje na to da neka vrednost nije broj, i sama tipa *Number*. Takođe, *NaN* je jedina vrednost u JavaScript jeziku koja prilikom poređenja sa samom sobom, daje vrednost *false*.

Vrednosti **Infinity** i **-Infinity** ukazuju da broj teži pozitivnoj ili negativnoj beskonačnosti.

Tako će deljenje pozitivnog broja nulom proizvesti sledeću vrednosti

var x = 5 / 0; // x is Infinity

Deljenje negativnog broja nulom proizvodi negativnu beskonačnu vrednost:

var x = -5 / 0; // x is -Infinity

Numerički tip podatka – bigint

Upravo opisani numerički tip podatka *number* poseduje jedno ograničenje. Korišćenjem njega najveći ceo broj koji je moguće predstaviti je 25^{53} – 1. Drugim rečima, korišćenjem tipa *number* moguće je predstaviti brojeve:

od **-9007199254740991** do **9007199254740991.**

Pokušaj predstavljanja brojeva koji izlaze iz ovog opsega stvara neočekivane rezultate:

var x = 9999999999999999;

document.write(x);

Upravo je kreirana promenljiva sa numeričkom vrednošću koja prevazilazi opseg *number* tipa podataka. Unutar HTML dokumenta se dobija sledeći ispis:

10000000000000000

Može se primetiti da ispisana i inicijalizovana vrednost nisu iste. Razlog je nepredviđeno zaokruživanje vrednosti do koga dolazi zato što vrednost prevazilazi opseg *number* tipa.

Ukoliko je potrebno raditi sa ovako velikim numeričkim vrednostima, potrebno je to eksplicitno i naglasiti:

var x = 9999999999999999n;

document.write(x);

Ispis ovakve vrednosti sada prolazi bez ikakvog problema:

9999999999999999

Da je stvarno reč o tipu koji nije *number* može se veoma lako proveriti:

document.write(typeof x);

Ispis je sledeći:

Bigint

Sada možete napraviti izmenu u kodu, tako što ćete promenljivoj x postaviti vrednost nakon čega ćete moći da uočite kako definisanje vrednosti utiče na prikaz poruke i određivanje tipa promenljive.

Tekstualne vrednosti – string

JavaScript promenljive i konstante mogu kao svoju vrednost imati i neki tekst. Tekstualne vrednosti se, za razliku od numeričkih, pišu između navodnika i na taj način JavaScript izvršno okruženje zna da je reč o tekstu. Dozvoljeno je pisanje tekstualnih vrednosti između jednostrukih i dvostrukih navodnika.

```
var x = 'text 1';
var y = 'text 2;'
```

Na ovaj način kreirane su dve promenljive, *x* i *y,* i obe su tipa **string**.

Navođenje navodnika unutar vrednosti *stringa* je moguće, sve dok se takvi navodnici razlikuju od onih koji su iskorišćeni za definisanje *string* vrednosti:

```
var answer = "It's alright" ;
var answer = "Više nije 'Beograđanka' najviša zgrada u Beogradu";
var answer = 'Sada je "Kula Beograd";'
```

Logičke vrednosti – boolean

Boolean je tip podatka koji poznaje samo dve predefinisane vrednosti: *true* i *false*. Ove vrednosti se obično koriste za kreiranje uslovnih izraza i i kontrolu toka izvršavanja koda. Primer promenljive koja će imati vrednost tipa *boolean* može da izgleda ovako:

```
var x = 5==6;
```

Za vrednost promenljive *x* postavljen je jedan logički izraz unutar koga je upotrebljen operator jednakosti. Jednostavnije rečeno, prikazana linija koda može da se pročita:

Proveri da li je 5 jednako 6 i informaciju o tome upiši u promenljivu x.

S obzirom na to da 5 nije jednako 6, ovakav izraz će da proizvede logičku vrednost *false*. To je vrednost *boolean* tipa.

Undefined

U jednoj od prethodnih lekcija već je prikazano da deklaracijom nove promenljive bez njene inicijalizacije, ona automatski dobija početnu vrednost *undefined*:

var x;

document.write(x);

Kao ispis se dobija:

undefined

Jasno je da je vrednost deklarisane, ali ne i inicijalizovane promenljive *undefined*. Ipak, bitno je znati da je *undefined* ujedno i tip:

var x;

document.write(type of x);

Ispis je

Undefined

Iz upravo prikazanog primera jasno je da je *undefined* tip promenljivih čija vrednost još uvek nije određena. Podrazumevana vrednost *undefined* tipa je takođe *undefined*.

Null

Null je još jedan od osnovnih tipova vrednosti koje JavaScript promenljive mogu imati. Baš kao što je to bio slučaj sa tipom *undefined*, *null* je ujedno i tip i vrednost.

Vrednost *null* je zamišljena kao indikator odsustva vrednosti. Ukoliko je potrebno definisati odsustvo vrednosti neke promenljive, može se napisati:

var x = nll; //   value is null

U JavaScriptu *null* se koristi da predstavi nešto što zapravo ne postoji. Za početnike može biti veoma teško da uvide razliku između tipova *undefined* i *null*. *Undefined* predstavlja promenljivu koja je deklarisana, ali još nije inicijalizovana, dok je sa druge strane *null* tip koji predstavlja odsustvo realnosti.

Napomena

Kako bi stvari bile još komplikovanije, ukoliko pokušate da ispitate tip promenljive sa vrednošću `null`, dobija se veoma zanimljiv efekat:

```
var x = null;
document.write(typeof x);
```

Ispis je sledeći:

```
object
```

Na osnovu dobijenog ispisa, pogrešno se može zaključiti da je tip promenljive sa `null` vrednošću `object`. Reč je o bugu koji u JavaScript jeziku postoji još od njegove prve verzije. Nakon određenog vremena odlučeno je da se greška neće ispravljati, zato što bi mogla narušiti funkcionisanje već kreiranih JavaScript skripti. Stoga je dobro znati i za ovu osobinu `null` tipa i istoimene vrednosti.

Symbol

Najmlađi tip podatka, koji je u JavaScript uvršten 2015. godine, jeste tip *Symbol*. Reč je o tipu koji dozvoljava kreiranje jedinstvenih identifikatora. Kako bi se dobila jedna vrednost *Symbol* tipa, potrebno je uraditi sledeće:

const ID = Symbol();

Iz prikazane naredbe se može videti da se vrednost *Symbol* tipa dobija pozivanjem istoimene funkcije. Bitno je znati da se svakim pozivanjem funkcije *Symbol()* dobija nova, jedinstvena vrednosti.

U ovom trenutku je veoma teško uvideti značaj *Symbol* tipa. Stoga je za sada dovoljno znati da je reč o prilično novom JavaScript tipu koji se koristi za kreiranje jedinstvenih identifikatora.

Object

Svi tipovi koji su navedeni u dosadašnjem toku ove lekcije bili su takozvani **prosti ili primitivni tipovi**. Promenljive sa vrednostima primitivnih tipova podataka mogu sadržati samo jednu vrednost u jednom trenutku:

var car = "Honda".

Ipak, jedan automobil može biti definisan brojnim podacima: proizvođač, model, boja, zapremina motora... Hajde da vidimo kako bismo sve te podatke predstavili tradicionalno korišćenjem JavaScripta:

var make = "Honda";

var model = "S2000";

var color = "black";

var engineDisplacement = 1997;

Zamislite sada koliko korisno bi bilo sve navedene vrednosti smestiti u jednu promenljivu. Takvo nešto je zapravo moguće postići, i to korišćenjem **složenih, odnosno kompozitnih tipova podataka** – objekata.

Objekti se mogu doživeti kao kontejneri za veći broj podataka. Tako je prethodni primer moguće transformisati na sledeći način:

var car = {make:"Honda", model:"S2000", color: "black", engineDisplacement:1997};

Na ovaj način kreiran je jedan objekat, koji sadrži sve informacije o jednom vozilu. Podaci su navedeni u formi parova ključeva i vrednosti. Tako je na primer *make* ključ, a *Honda* vrednost.

Različiti podaci o vozilu, enkapsulirani unutar jednog objekta, mogu se prikazati na sledeći način:

document.write(car.make);

Linija ilustruje pristup za ispis samo jednog podatka o vozilu (*make*). Nakon naziva promenljive koja predstavlja objekat, navodi se tačka i naziv ključa, čiju vrednost je potrebno dobiti.

Promenljive tokom svog života mogu menjati tipove podataka

U JavaScript jeziku moguće je jednoj istoj promenljivoj dodeljivati vrednosti različitih tipova, bez ikakvih problema:

var x = 42;

x = „some text";

U prvoj liniji, promenljivoj *x* dodeljena je numerička vrednost *42*. Na taj način promenljiva *x* dobija tip *number*. U narednoj liniji promenljivoj *x* je dodeljena tekstualna vrednost. Tako promenljiva *x* menja svoj tip u *string*, bez ikakvog problema.