Fakultet informatike u Puli

Programsko inženjerstvo - Vježbe

1. Vrijednosti, tipovi i operatori

(prilagođeno prema knjizi Eloquent Javascript)

> Javascript kod može pokrenuti u konzoli internet preglednika, pomoću `nodejs` interpretera ili na web servisima poput **https://jsconsole.com/**, **https://jsfiddle.net**, **https://js.do**.

Računalo vrši pohranu i obradbu podataka. Svaka informacija koju računalo obrađuje smatramo podatkom, pa čak i sam program koji je najčešće pohranjen kao niz znakova - jest podatak. Podatak je osnovni element pohrane koji reprezentira određene informacije važne za korisnika i/ili nesmetani rad računala.

Javascript podržava nekoliko tipova podataka pojašnjenih u nastavku. Tip podatka određuje način na koji se pohranjeni podatak (najčešće u binarnom zapisu) interpretira.

Cijeli brojevi. Označavaju se prirodno i očekivano: znamenkama pomoću dekadskog brojevnog sustava: `1`, `2`, `3`, ..., `9999`, ... U JavaScriptu se tip podataka koji sadrži brojeve naziva `number` i kako je JavaScript dinamičan, nije ga potrebno eksplicitno navoditi prilikom deklaracije varijabli (univerzalno koristimo ranije spomenuti `let`. JavaScript koristi 64 bita za pohranu brojeva, što znači da podržava 2^{64} različitih brojeva (nisu svi pozitivni, jedan *bit* označava negativne brojeve).

Decimalni brojevi. Decimalni dio broja označava se isključivo decimalnom točkom (`.`), a može se označiti i eksponent. Primjerice broj $3.23 \cdot 10^{55}$ unosimo kao `3.23e55`. U pozadini se koristi IEEE 754 norma (zbog toga je $0.1 + 0.2 = 0.30000000000000004$).

Aritmetika (binarni operatori). Podržani su klasični binarni operatori poput `*`, `+`, `-`, `/` te modulo operator `%`, uz očekivano ponašanje zagrada `(` i `)` koje daju prednost pri izvršavanju matematičkih operacija. Podržane su i specijalni brojevi poput pojma beskonačnosti: `Infinity` i `-Infinity`. Dodatno, oznaka `NaN` označava rezultat koji nije broj (pr. rezultat operacije dijeljenja s nulom). Iako `NaN` nije broj, njezin JavaScript tip je i dalje `number`.

String. Niz znakova (engl. i u daljnjem tekstu - *string*) se u Javascriptu može unositi na tri načina: omeđeno jednostrukim navodnicima (`'`), dvostrukim navodnicima (`"`) ili tzv. *backtick* navodnicima (`` ` ``):

```
"Lie on the ocean"
'Float on the ocean'
`Down on the sea`
```

Nije moguće `string` započeti i završiti različitim znakom od da tri ponuđena. U pravilu se koristi dvostruki navodnik (`"`) osim kada je u samom stringu potrebno unijeti taj znak. Onda imamo dvije mogućnosti: koristiti neki drugi znak za string ili koristiti specijalni znak za *escape* -ing `\`.

```
let s1 = 'Idemo u "McDonalds".';
let s2 = "Idemo u \"McDonalds\".";
```

JavaScript `string` može sadržavati i specijalne znakove poput oznake novog reda `\n` ili *tabulatora* -a `\t`.

Konkatenacija `string` ova se postiže pomoću binarnog operatora `+`: `"a" + "b" + "c"`.

Specijalno, ukoliko je `string` omeđen sa *backtick* navodnicima podržava i mogućnost unošenja Javascript izraza unutar njega, pa je npr. moguće sljedeće:

```
console.log(`Polovica od 100 je ${100/2}`);
```

Unarni operatori. Operatore `typeof` i `-` nazivamo unarnim jer iza njih slijedi samo jedan argument. `typeof` služi za `string` reprezentaciju tipa varijable.

```
console.log(typeof "neki_string"); // → "string"
console.log(typeof 4); // → "number"

console.log(typeof typeof NaN); // ? :)
```

Operator `-` je ujedno i binarni operator ako se koristi kao operacija oduzimanja (prima dva argumenta).

Boolean vrijednosti. Tip podataka `boolean` sadrži samo dvije moguće vrijednosti: `true` i `false` koji su ujedno i rezultat logičke usporedbe pomoću operatora: `>`, `<`, `>=`, `<=`, `==` i `===`. O razlici između `==` i `===` kasnije. Postoji i logički operatori *AND* koji se označava sa `&&`, OR (`||`) i negacija `!`.

```
console.log(3 > 2) → true
console.log("b" > "a") → true
console.log(true || false) → true
console.log(false && true) → false
```

Postoji i ternarni operator (koji izražava odnos između tri vrijednosti) koji koristi `?` i `:` te predstavlja biranje vrijedsnosti u zavisnosti o istinitosti uvjeta.

```
console.log(true ? 1 : 2) → 1
console.log(false ? 1 : 2) → 2
console.log(1234 > 23 ? "a" : "b") → "a"
```

Automatska promjena tipova

Javascript je jedan od jezika kod kojeg postoji automatska promjena tipova između gotovo svake vrste tipova. Kao što je ranije napomenuto, ponekad se ta konverzija ne ponaša u skladu sa očekivanom pa valja upoznati na koji način funkcionira. Ponekad je to ponašanje i nekonzistentno,, primjerice:

```
console.log(12 * null) → 0
console.log("5" - 4) → 1
console.log("5" + 1) → "51"
console.log(5 + "1") → "51"
console.log("five" * 2) → NaN
console.log(false == 0) → true
```

Ako neki operator primijenimo na krivi tip podataka, Javascript pokušava konvertirati vrijednosti slijedeći niz ugrađenih pravila, što nazivamo `type coercion`.

U prvom izrazu `null` se pretvara u `0`. U drugom izrazu `"5"` se pretvara u `5`, dok se pak za operator `+` 1 pretvara u `"1"` što za rezultat daje da se `string` ovi konkateniraju koristeći operator `+`. To je zbog toga što je prvi argument `string`, te postoji operator `+` koji funkcionira na `string` ovima. U petom primjeru ne postoji operator `*` definiran nad `string` ovima pa Javascript pokušava vrijednost `"five"` pretvoriti u `number`. Kako to ne uspijeva, rezultat te operacije je `NaN` što pomnoženo sa 2 daje `NaN`.

Da se `"five"` pretvara u `NaN` možemo se uvjeriti koristeći ugrađenu funkciju za pretvorbu u `number`:

```
console.log(Number("five")) → NaN
```

Javascript također automatski prevodi određene vrijednosti u `true` ili `false` ukoliko je to potrebno (primjerice prilikom logičkih operatora ili logičke usporedbe `if` om). Vrijednost `""` se pretvara u `false` kao i vrijednost `0` što se može provjeriti dvostrukom operacijom negacije:

```
console.log(!!"") → false
console.log(!!0) → false
console.log(!!"nešto, bilo što") → true
console.log(!!1) → true
console.log(!!undefined) → false
console.log(!!{}) → true // iako je prazan objekt
console.log(!!NaN) → false
```

Ukoliko ne želimo da se prilikom provjere jednakosti između vrijednost korištenjem operatora `==` automatski promjene tipovi, možemo koristiti operator `===`. Primjerice:

```
console.log(1 == "1") → true
console.log(1 === "1") → false
```

Povratni tip kod logičkih operatora. Logički operatori `&&` i `||` imaju dodatno definirano ponašanje, a to je da ne vraćaju strogo `true` ili `false` (kako bi očekivali).

Operator `||` vraća vrijednost s lijeve strane ukoliko se ona može pretvoriti u `true`, a u protivnom vrijednost s desne strane (neovisno što je).

Operator `&&` vraća vrijednost s lijeve strane ukoliko se ona pretvara u `false`, a u protivnom vraća vrijednost s desne strane (neovisno).

Primjer:

```
if (null || 0) {
  console.log("OK")
}
else {
  console.log("čudno...")
}

if (null || 0) {
  console.log("OK")
}
else {
  console.log("čudno...")
}
```

Zadaci za vježbu

1. **(JS-101)** Nadopuni kod da ispravno ispisuje.

```
let a = prompt("Unesi prvi broj");
let b = prompt("Unesi drugi broj");
let c = prompt("Unesi treći broj");
let d = prompt("Unesi četvrti broj");
let maks;

// tvoj kod

console.log("Najveći između njih je: " + maks);
```

2. **(JS-102)** Napiši program koji prima broj bodova sa kolegija i ispisuje ocjenu (prema previlniku o ocjenjivanju). Ukoliko je ocjena pozitivna ispisati teskt sa čestitkom i ocjenom.

```
let bodovi = prompt("Unesi broj bodova");
let ocjena;
let poruka;

// tvoj kod

console.log(poruka);
```

3. Instaliraj na svoje računalo