Operatori

Operatori su posebni leksički elementi jezika koji omogućavaju izvršavanje operacija nad vrednostima. Iako to možda niste znali, u primerima iz prethodnih lekcija već su korišćeni neki operatori.

Operatori u JavaScript jeziku

JavaScript poznaje nekoliko vrsta operatora:

- Aritmetički operatori
- Operatori dodeljivanja
- Operatori poređenja
- Logički operatori

Aritmetički operatori

Aritmetički operatori su oni koji izvršavaju aritmetičke operacije nad vrednostima. Tabela pokazuje aritmetičke operatore jezika JavaScript.

Operator	Opis	Primer	Vrednost

+	sabiranje	`var x = 5;`
		`var y = 2;`
		`var z = x + y;`
			7

-	oduzimanje	`var x = 5;`
		`var y = 2;`
		`var z = x - y;`
			3

*	množenje	`var x = 5;`
		`var y = 2;`
		`var z = x * y;`
			10

/	deljenje	`var x = 5;`
		`var y = 2;`
		`var z = x / y;`
			2.5

%	ostatak pri deljenju (moduo)	`var x = 5;`
		`var y = 2;`
		`var z = x % y;`
			1

++	uvećanje za jedan	`var x = 5;` `x++;`	6
--	smanjenje za jedan	`var x = 5;` `x--;`	4

Aritmetički operatori nad vrednostima različitih tipova

S obzirom na dinamičku prirodu JavaScript tipova, aritmetičke operatore je moguće koristiti nad vrednostima različitih tipova, a JavaScript će obaviti automatsku konverziju podataka. Iako ovaj pristup omogućava veliku slobodu prilikom rada sa vrednostima, potrebno je poznavati neka osnovna pravila kojih se JavaScript pridržava prilikom obavljanja automatske konverzije.

Nad *number* i *string* vrednostima može se koristiti operator sabiranja, a JavaScript će automatski obaviti konverziju.

x = "Duzina puta je u kilometrina " + 365; // Duzina puta je u kilometrima 365

U primeru se može videti da je JavaScript automatski obavio konverziju broja *365* u *string* tip i da je takvu vrednost nadovezao na tekst sa leve strane izraza. Ista situacija bi se dogodila i u sledećem primeru:

x = 365+ " je duzina puta je u kilometrima"

Iz prikazanog se moze zakljuciti da prilikom upotrebe operatora za sabiranje nad vrednostima *number* i *string* tipova JavaScript automatski konvertuje *number* vrednosti u *string*.

Ipak, u izrazima u kojima se upotrebljavaju ostali aritmeticki operatori JavaScript ne vrsi konverziju *number* vrednosti u *string*.

"45" – 5 // 40

Prilikom korišćenja ostalih aritmetičkih operatora nad tipovima *number* i *string* JavaScript *string* konvertuje u *number* (naravno, ukoliko je to moguće). To se može videti i iz upravo prikazanog primera oduzimanja.

Operatori dodeljivanja

Operatorima dodeljivanja dodeljuje se neka vrednost promenljivoj ili konstanti. JavaScript poznaje nekoliko operatora dodeljivanja, prikazanih tabelom ispod. Osnovni operator dodeljivanja jeste karakter jednako (=). Ipak, JavaScript poznaje i nekoliko drugih operatora ovog tipa, koji predstavljaju kombinaciju operatora dodeljivanja i aritmetičkih operatora.

Operator	Opis	Primer	Značenje
=	dodeljuje vrednost sa desne strane, promenljivoj sa leve	x = y	x = y
+=	uvećava vrednost promenljive x za vrednost y i novodobijenu vrednost dodeljuje promenljivoj x	x += y	x = x y
-=	umanjuje vrednost promenljive x za vrednost y i novodobijenu vrednost dodeljuje promenljivoj x	x -= y	x = x y
*=	množi vrednost promenljive x sa vrednošću y i novodobijenu vrednost dodeljuje promenljivoj x	x *= y	x = x y
/=	deli vrednost promenljive x sa vrednošću y i novodobijenu vrednost dodeljuje promenljivoj x	x /= y	x = x y
%=	računa ostatak deljenja vrednosti x, vrednošću y, i dobijeni rezultat dodeljuje promenljivoj x	x %= y	x = x y

Posebno interesantna može biti kombinacija standardnog operatora dodeljivanja (=) i aritmetičkih operatora za uvećanje i umanjenje.

Naime, ove operatore je moguće kombinovati u jednoj naredbi i, u zavisnosti od načina upotrebe, oni mogu stvoriti različit efekat. Unutar radnog okruženja možete testirati pomenuti operator.

```
let x = 45;
let y = x++;
document.write(y);
```

Unutar prikazanog koda prvo se obavlja deklarisanje i inicijalizovanje promenljive *x* sa vrednošću 45. Nakon toga je napisana naredba u kojoj se promenljivoj *y* dodeljuje vrednost promenljive *x*, ali se pri tome obavlja i uvećanje vrednosti promenljive *x* za 1.

Nakon izvršavanja prikazanog koda, unutar HTML dokumenta se ispisuje vrednost primenljive *y*:

45

Iako ste možda očekivali da vrednost promenljive *y* nakon izvršavanja koda bude *46*, ona je zapravo *45*. Već više puta je rečeno da JavaScript kod izvršava naredbu po naredbu sleva nadesno. Upravo zbog toga je prvo dodeljenja vrednost promenljive *x*, promenljivoj *y*, a tek nakon toga je urađeno i njeno uvećanje za 1.

Ukoliko je potrebno postići drugačiji efekat, odnosno prvo obaviti uvećanje, pa tek onda dodeljivanje može se napisati:

let y = ++x;

Operator uvećanja sada je premešten pre naziva promenljive, što će kao rezultat imati uvećanje vrednosti promenljive pre njenog dodeljivanaj. Tako će nakon ove linije promenljiva *y* imati vrednost *46*.

Operatori poređenja

JavaScript poseduje nekoliko operatora koje je moguće koristiti za poređenje dve ili više vrednosti. Poređenjem se stvara nova vrednost, koja ukazuje na rezultat poređenja. Takva vrednost je logičkog tipa (*boolean*), što je jedan od osnovnih JavaScript tipova, obrađenih u prethodnoj lekciji.

Operatori poređenja koriste se u logičkim izjavama kako bi utvrdili istinitost nekog izraza.

Tabela ispod pokazuje operatore poređenja JavaScript jezika.

Operator	Opis	Primer	Rezultat

==	jednakost	`6 == 8`
		`6 == 6`
		`6 == "6"`
			`false`
			`true`
			`true`

===	jednakost po vrednosti i po tipu (striktna jednakost)	`6===6` `6==="6"`	`true` `false`
!=	nejednakost	`6 != 8` `6 != 6`	`true` `false`

!==	nejednakost po vrednosti ili po tipu	`6 !== 6`
		`6 !== "6"`
		`6 !== 8`
			`false`
			`true`
			`true`

>	veće	`6 > 8` `6 > 4`	`false` `true`
<	manje	`6 < 8` `6 < 4`	`true` `false`
>=	veće ili jednako	`6 >= 8` `6 >= 6`	`false` `true`
<=	manje ili jednako	`6 <= 8` `6 <= 6`	`true` `true`

Poređenje vrednosti različitih tipova

Baš kao što je slučaj i sa aritmetičkim operatorima, JavaScript omogućava poređenje vrednosti koje su različitog tipa. U nastavku će biti prikazano nekoliko karakterističnih situacija.

Poređenje teksta i broja:

3 < "13" // true

Poređenjem *string* i *number* vrednosti, JavaScript će tekstualnu vrednost konvertovati u broj:

Vrednost „13“ se konvertuje u broj. Broj 3 je manji od broja 13, pa je rezultat *true*.

Poređenje praznog stringa i broja

‘ ’ == 0 // true

Prazan *string* konvertuje se u broj nula (0). Stoga upravo prikazani izraz proizvodi logičku vrednost *true*.

Poređenje stringa koji se ne može konvertovati u broj i broja:

3 < “Ben” // false

String vrednost koja se ne može konvertovati u broj dobija vrednost *NaN*. Izrazi poređenja sa *NaN* uvek vraćaju *false*.

Poređenje logičkih vrednosti i brojeva:

false == 0 // true

Prilikom poređenja logičkih vrednosti i brojeva, logičke vrednosti se konvertuju u *0* i *1*.

Vrednost *true* se konvertuje u *1*, a vrednost *false* u *0*. Upravo se zbog toga u prikazanom primeru kao rezultat poređenja dobija *true*.

Ipak, imajte na umu da bi, u slučaju provere striktne jednakosti (===), rezultat bio drugačiji:

false === 0 // false

Poređenje logičkih vrednosti i teksta:

false === ’0’ // false

U slučaju poređenja logičkih vrednosti i teksta obe bivaju konvertovane u brojeve. U prikazanom primeru *false* se konvertuje u broj *0*, a takođe i tekstualna vrednost *‘0’*. Upravo zbog toga izraz proizvodi vrednosti *true*.

Poređenje null i undefined tipova:

null === undefined // true

Poređenjem *null* i *undefined* vrednosti, kao u prikazanom primeru, dobija se vrednost *true*. Ipak, ukoliko se proveri striktna jednakost, dobija se drugačiji rezultat, zbog različitih tipova ove dve vrednosti:

null === undefined // false

Poređenje tekstualnih vrednosti

U JavaScript jeziku moguće je porediti tekstualne vrednosti:

"some text" == "some text" // true

Kao što se može videti, utvrđivanje jednakosti tekstualnih vrednosti je veoma jednostavno.

Ipak, poređenje korišćenjem ostalih operatora funkcioniše na specifičan način:

"3" > "13" // true

Sada je napisan izraz unutar koga se proverava da li je tekst 3 veći od teksta 13. Kao rezultat se dobija *true*. Primer može lako da zavara. Ukoliko se tekstualne vrednosti posmatraju kao brojevi, *3* je svakako manje od *13*, ali ipak, to nije rezultat koji se dobija.

Bitno je znati da je prilikom poređenja dva *stringa* takve vrednosti ne bivaju konvertovane u *number* tip, već se poređenje obavlja na osnovu leksikografičkih pravila (alfabetnog reda), odnosno redosleda karaktera u Unicodeu.

Pravila koja se primenjuju su sledeća:

- Porede se prvi karakteri oba *stringa*,
- Što se karakter ranije pojavljuje unutar leksičke mape, odnosno Unicodea, to je njegova vrednost manja; tako *a* ima manju vrednost od *b*, a karakter *1* ima manju vrednost od *2,*
- Samo u slučaju jednakosti prvih karaktera nastavlja se sa poređenjem preostalih karaktera unutar *stringova*
- Preostali karakteri se porede po istom principu
- Ukoliko su svi karakteri jednaki, onda su i dva *stringa* jednaka,
- Ukoliko su svi karakteri jednaki, ali je jedan od *stringova* duži, veći je *string* sa više karaktera.

Uzimajući u obzir sve što je navedeno, u prikazanom primeru se porede karakteri *3* i *1*. Karakter *3* je u alfabetnom redu veći od karaktera *1*, zato što dolazi nakon njega, pa se na kraju dobija da je tekst *3* veći od teksta *13*.

Logički operatori

Logički operatori se koriste kako bi utvrdili logičku povezanost između promenljivih, odnosno vrednosti. Najčešće se koriste u kombinaciji sa upravo prikazanim operatorima poređenja.

Logičkim operatorima se mogu kreirati različiti složeni logički izrazi, koji kao svoju finalnu vrednost uvek mroaju imati *true* ili *false*.

Operator	Opis	Primer	Rezultat
&&	AND	`(6 < 11 && 5 > 1)` `(6 < 5 && 5 > 1)`	`true` `false`
||	OR	`(6 < 5 || 5 > 1)`	`true`
!	NOT	`!(6==6)`	`false`

Analizom primera upotrebe logičkih operatora može se mnogo toga zaključiti. Operator *AND* zahteva ispunjenje svih navedenih uslova kako bi proizveo vrednost *true*. U protivnom, proizvodi vrednost *false*.

Tabela ispod ilustruje različite situacije upotrebe logičkog operatora AND.

Izraz	Rezultat
`true && true`	`true`
`false && true`	`false`
`true && false`	`false`
`false && false`	`false`

Sa druge strane, OR operator zahteva i ispunjenje samo jednog od uslova, kako bi kao svoju finalnu vrednost proizveo *true*

Izraz	Rezultat
`true || true`	`true`
`false || true`	`true`
`true || false`	`true`
`false || false`	`false`

Na kraju, operator NOT je operator negacije, koji *true* vrednost pretvara u *false*, a *false* u *true*, što je ilustrovano tabelom ispod.

Izraz	Rezultat
`!true`	`false`
`!false`	`true`

Ternarni operator

Ternarni operator je obavezni operator svakog popularnijeg jezika. Ni JavaScript nije izuzetak. Ovaj operator je nešto teži za razumevanje, ali je u osnovi veoma jednostavan i posebno koristan kada se jednom ovlada njegovom sintaksom.

Ternarni operator se koristi za dodelu vrednosti promenljivoj, ali na osnovu nekog predefinisanog uslova. Sintaksa ternarnog operatora je sledeća:

```
myVariable = (condition) ? value1 : value2
```

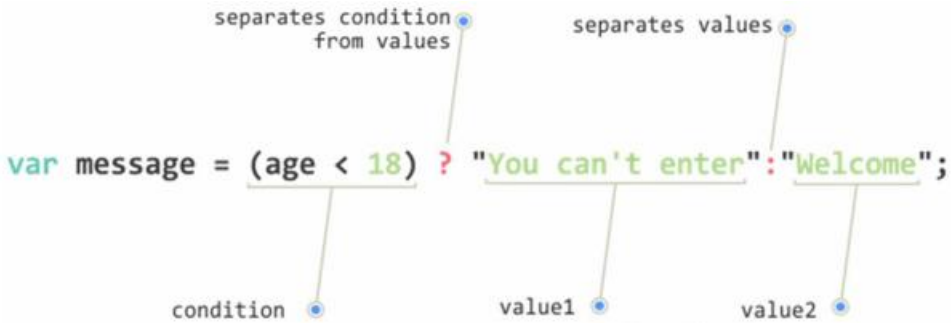
Napisano prostim jezikom, značenje navedene sintakse bi bilo: ukoliko je *condition* ispunjen, promenljivoj *myVariable* dodeli *value1*, a u protivnom *value2*.

Ternarni operator se gradi korišćenjem karaktera ***?*** i ***:*** . Karakter upitnik (?) razdvaja uslov od mogućih vrednosti, a karakter dve tačke (:) razdvaja ponuđene vrednosti,

Primer upotrebe ternarnog operatora je sledeći:

```
var message = (age < 18) ? "You can't enter":"Welcome";
```

Prikazana naredba, koja predstavlja ternarni operator, može se raščlaniti na elemente prikazane slikom ispod.



Ukoliko je vrednost promenljive *age* manja od 18, promenljiva *message* dobija vrednost *You can't enter*. U protivnom promenljiva *message* dobija vrednost *Welcome*.

Nadovezivanje stringova

Aritmetički operator sabiranja (+) može se u JavaScriptu koristiti i za nadovezivanje *string* vrednosti. Nadovezivanje *stringova* se drugačije naziva **konkatenacija**. Primer ispod ilustruje nadovezivanje stringova.

```
let txt1 = "Java";
let txt2 = "Script";
let txt3 = txt1 + txt2;

document.write(txt3);
```

Rezultat prikazanog koda je sledeći:

JavaScript

Pokušajte da unutar radnog okruženja izmenite kod kako bi dodali i skraćenicu za JavaScript. Rezultat koda bi trebalo da je sledeći:

JavaScript (JS)