**Petlje**

Jedinica: 23 od 30

**Rezime**

* Petlje, pored grananja, predstavljaju osnovne elemente za postizanje kontrole toka.
* Petlje omogućavaju da se određeni kod izvrši više puta.
* For petlja omogućava da se određeni blok koda izvrši tačan broj puta.
* Petlja for…in dizajnirana je specijalno za prolazak kroz sva svojstva neke objektne promenljive.
* Petlja for…of omogućava prolazak kroz vrednosti svojstava jednog objekta.
* While petlja izvršava određeni blok koda, sve dok je ispunjen uslov za izvršavanje petlje.
* Do…while je specijalna vrsta petlje, kod koje se prvo izvršava blok koda, a tek onda proverava i uslov.
* Naredba continue prekida aktuelnu iteraciju i prelazi na sledeću.
* Naredba break napušta petlju.

Petlje, pored grananja, predstavljaju osnovne elemente za postizanje kontrole toka. Ipak, za razliku od uslovnog izvršavanja (grananja), petlje imaju nešto drugačiju funkciju. Da bi se njihova funkcija na najbolji način razumela, biće razmotrena sledeća situacija:

*Na koji način se može postići ponavljanje izvršavanja jedne linije JavaScript koda?*

Na primer, potrebno je 5 puta ispisati jednu istu poruku na stranici. Najjednostavnije, bazično rešenje za postizanje opisanog ponašanja jeste navođenje 5 sukcesivnih linija za ispis:

|  |  |
| --- | --- |
| 1  2  3  4  5 | document.write("Hello");  document.write("Hello");  document.write("Hello");  document.write("Hello");  document.write("Hello"); |

Željeni rezultat je postignut, ali šta da je bilo potrebno umesto 5 puta poruku ispisati 1000 puta? Jasno je da je navođenje linije za linijom vrlo nepraktično i nedopustivo. Ono što je potrebno uraditi kako bi se nastali problem razrešio jeste iznalaženje načina da se interpreteru kaže: *hej, izvrši ovu liniju document.write("Hello"); tačno 1000 puta.*

Upravo su petlje naredbe koje je potrebno koristiti da bi se postiglo nešto ovako. JavaScript poznaje nekoliko načina za kreiranje petlji:

|  |  |
| --- | --- |
| 1  2  3  4  5 | while  do…while  for  for…in  for…of |

Takođe, JS poseduje i dve ključne reči za kontrolu izvršavanja petlji:

|  |  |
| --- | --- |
| 1  2 | break  continue |

Navedene naredbe biće tema lekcije pred vama.

**For petlja**

For petlja omogućava da se određeni blok koda izvrši tačan broj puta. U uvodnom delu ove lekcije postavljeno je pitanje kako postići ispis jedne iste linije koda n (proizvoljan) broj puta. Odgovor upravo leži u korišćenju for petlje.

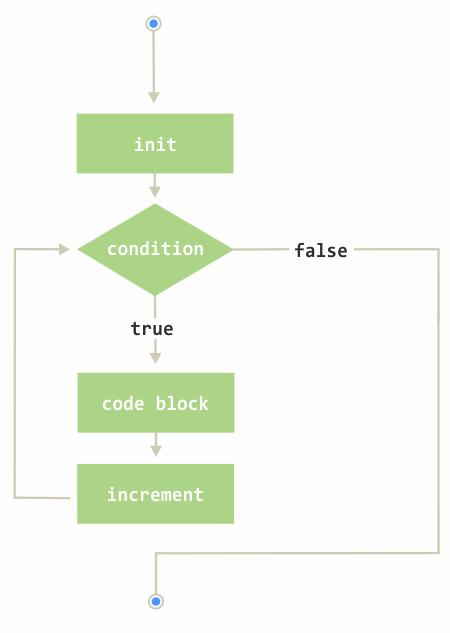
Sintaksa for petlje je sledeća:

|  |  |
| --- | --- |
| 1  2 | for (initialExpression; condition; incrementExpression)    statement |

Izvršavanje for petlje ponavlja se sve dok je uslov (condition) ispunjen. Ipak, pored uslova, analizom sintakse mogu se zapaziti još neki elementi for petlje. Koraci koji se odvijaju prilikom izvršavanja jedne for petlje su sledeći:

1. izvršava se početni izraz – [inicijalizacija](https://www.link-elearning.com/linkdl/opisPojma.php?id=144351) (initialExpression); unutar ovog početnog izraza uglavnom se inicijalizuje jedan ili više brojača petlje;
2. utvrđuje se istinitost uslova; ukoliko je uslov ispunjen, prelazi se na izvršavanje tela petlje;
3. nakon izvršavanja tela petlje, izraz se ažurira izvršavanjem izraza incrementExpression.

Iznet opis je prilično apstraktan, tako da će stvari postati malo jasnije analizom slike 23.1, koja ilustruje izvršavanje for petlje.



*Slika 23.1 – For petlja*

Konačno, sledi primer koji razrešava pitanje iz uvodnog dela lekcije. Kako ispisati jednu istu liniju određen broj puta?

|  |  |
| --- | --- |
| 1  2  3 | for (var i = 0; i < 5; i++) {  console.log("Hello");  } |

Rezultat prikazanog koda biće:

|  |  |
| --- | --- |
| 1  2  3  4  5 | Hello  Hello  Hello  Hello  Hello |

U primeru se najpre postavlja ključna reč for, čime se započinje petlja. Nakon toga, u zagradama se definišu inicijalizacija, uslov i inkrementacija.

Jednostavnim jezikom, prvo se deklariše i inicijalizuje jedna promenljiva sa nazivom i:

|  |  |
| --- | --- |
| 1 | var i = 0; |

Ova promenljiva ima početnu vrednost 0, a unutar for petlje služi kao brojač.

Nakon inicijalizacije, sledi deo kojim se definiše uslov:

|  |  |
| --- | --- |
| 1 | i < 5; |

Na ovaj način je rečeno da će se telo petlje izvršavati sve dok je vrednost promenljive i manja od 5.

Na kraju, definiše se i inkrementacija:

|  |  |
| --- | --- |
| 1 | i++ |

Svaki put, nakon završetka izvršavanja tela petlje, vrednost brojača se uvećava za jedan.

For petlja se veoma često koristi za prolazak kroz **nizove**:

|  |  |
| --- | --- |
| 1  2  3  4  5 | var arr = ["Belgrade", "Paris", "Berlin"];    for (var i = 0; i < arr.length; i++) {   console.log(arr[i]);  } |

U primeru je deklarisan i inicijalizovan jedan niz sa tri člana. For petlja se koristi za prolazak kroz niz i ispis vrednosti elemenata. Za formiranje uslova je iskorišćeno svojstvo length objekta Array, čime je dobijena dužina niza. Uslov je formiran tako da će se for petlja izvršavati sve dok je vrednost brojača manja od dužine niza. Tako prikazani primer proizvodi sledeći rezultat:

|  |  |
| --- | --- |
| 1  2  3 | Belgrade  Paris  Berlin |

**Prvi segment for petlje odnosi se na:**

 definisanje brojača

 proveru uslova

 korekciju vrednosti brojača

 ništa od navedenog

**for…in petlja**

Za prolazak kroz nizove, kao u prethodnom primeru, JavaScript poseduje još jednu, prikladniju vrstu petlje. Reč je o petlji for…in.

Petlja for…in dizajnirana je specijalno za prolazak kroz sva svojstva neke objektne promenljive. S obzirom da su nizovi objekti, for…in petlja je idealna za korišćenje na nizovima, i uopšte bilo kojim objektima.

Sintaksa for…in petlje je sledeća:

|  |  |
| --- | --- |
| 1  2  3 | for (variable in object) {    statements  } |

Variable označava deklaraciju neke proizvoljno nazvane promenljive, koja će pri svakoj iteraciji preuzimati vrednost koja se odnosi na naziv svojstva objekta. Drugim rečima, for…in petlja dozvoljava prolazak kroz nazive svojstava nekog objekta.

Prethodni primer, u kome je ilustrovan prolazak kroz jedan niz korišćenjem for petlje, može se transformisati na sledeći način:

|  |  |
| --- | --- |
| 1  2  3  4  5 | var arr = ["Belgrade", "Paris", "Berlin"];    for (var city in arr) {   console.log(arr[city]);  } |

Promenljiva sa nazivom city će u primeru, prilikom svake iteracije, dobijati vrednost sledećeg naziva svojstva objekta arr. S obzirom da su nizovi objekti, a nazivi svojstava zapravo indeksi, promenljiva city će prilikom svake iteracije dobijati vrednost indeksa elementa.

**for...of petlja**

Najprikladniji način za prolazak kroz jedan niz jeste korišćenje for…of petlje. Kao i for…in petlja, i petlja for…of takođe omogućava prolazak kroz vrednosti svojstava jednog objekta. Sintaksa ove petlje je sledeća:

|  |  |
| --- | --- |
| 1  2  3 | for (variable of object) {    statement  } |

Sintaksa je praktično identična sintaksi for…in petlje. Jedina razlika je u upotrebi ključne reči ofumesto in.

Sada se kod za prolazak kroz niz iz prethodnih primera može napisati ovako:

|  |  |
| --- | --- |
| 1  2  3  4  5 | var arr = ["Belgrade", "Paris", "Berlin"];    for (var city of arr) {   console.log(city);  } |

Ovoga puta nema potrebe za čitanjem elemenata niza na osnovu njihovih indeksa, s obzirom da u svakoj iteraciji promenljiva city dobija vrednost tekućeg elementa niza.

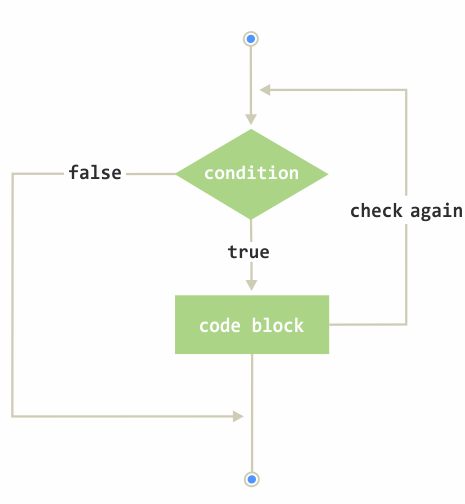
**while petlja**

U prethodnim redovima, prikazana je for petlja, kao i njene varijante for…in i for…of. Pored ovih petlji, JavaScript poznaje još nekoliko načina da se određeni kod izvrši više puta. While petlja je jedan od tih načina.

While petlja izvršava određeni blok koda, sve dok je ispunjen uslov za izvršavanje petlje. To praktično znači da ova vrste petlje nema unapred definisan broj iteracija. Sintaksa while petlje je sledeća:

|  |  |
| --- | --- |
| 1  2  3 | while (condition){    statement  } |

Sve dok je uslov (condition) ispunjen, izvršava se blok koda, odnosno telo petlje. Logika whilepetlje može se ilustrovati slikom 23.2.



*Slika 23.2 – While petlja*

Primer upotrebe while petlje je sledeći:

|  |  |
| --- | --- |
| 1  2  3  4  5 | var x=0;  while(x<5){   console.log("Hello");   x=x+1;  } |

Na početku se deklariše i inicijalizuje promenljiva x sa vrednošću 0. While petlja poseduje uslov i blok. Uslov je:

|  |  |
| --- | --- |
| 1 | x<5 |

Ovo praktično znači da će se petlja izvršavati sve dok je vrednost promenljive x manja od 5.  
Telo petlje definisano je vitičastim zagradama. Unutar tela petlje se vrši ispisivanje poruke *Hello*, ali i uvećavanje vrednosti promenljive x za jedan. Da kojim slučajem ova inkrementacija nije navedena, uslov za izvršavanje ove petlje uvek bi bio ispunjen. Na taj način bi se stvorila jedna beskonačna, odnosno mrtva petlja, koja nikada ne bi prestala da se izvršava.

**do…while petlja**

Upravo opisana while petlja ni na koji način ne garantuje da će se njeno telo uopšte i izvršiti. Jednostavno, ukoliko uslov nije ispunjen, telo petlje se neće izvršiti nijednom:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var x=6;  while(x<5)  {   console.log("Hello");   x=x+1;  } |

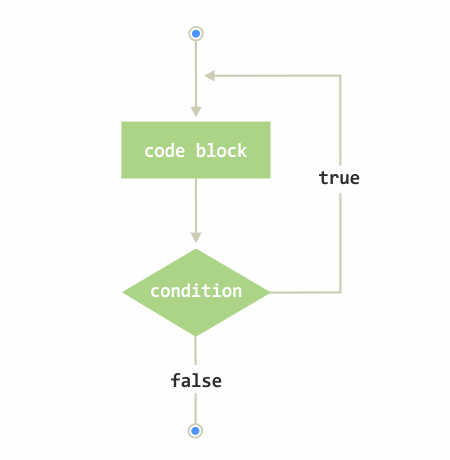
U primeru, vrednost promenljive x je veća od 5, tako da uslov za izvršavanje while petlje nije ispunjen, te se kod unutar tela while petlje neće izvršiti nijednom. Međutim, šta ukoliko je potrebno osigurati da se petlja izvrši makar jednom, bez obzira na ispunjenje uslova?

Tako nešto moglo bi se postići na taj način što bi se provera ispunjenosti uslova za izvršavanje smestila nakon bloka koda koji je potrebno izvršiti. Upravo to je osobina petlje do…while.

Sintaksa do…while petlje je sledeća:

|  |  |
| --- | --- |
| 1  2  3 | do{     // statements  }while(condition); |

Na početku se postavlja ključna reč do, a nakon nje blok koda koji je oivičen vitičastim zagradama. Nakon bloka, koristi se ključna reč while za definisanje uslova izvršenja petlje. Sve ovo se može ilustrovati slikom 23.3.



*Slika 23.3 – Do while petlja*

Sada se prethodni primer može transformisati na sledeći način:

|  |  |
| --- | --- |
| 1  2  3  4  5 | var x=6;  do{   console.log("Hello");   x=x+1;  }while(x<5); |

Rezultat je sledeći:

|  |  |
| --- | --- |
| 1 | Hello |

Zaključak je da se telo petlje izvršilo jednom zato što se provera ispunjenosti uslova sprovodi nakon izvršenja petlje.

**Naredbe break i continue**

Ponekad je potrebno kontrolisati izvršavanje petlje na osnovu nekog uslova i to unutar tela petlje. U takvim situacijama je moguće koristiti naredbe break i continue.

Naredba continue prekida aktuelnu iteraciju u prelazi na sledeću.

Naredba break napušta petlju.

Sledeći primer ilustruje upotrebu naredbe continue:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | for(var i=0;i<100;i++){   if(i>50){    continue;   }    console.log(i);  }  console.log(i); |

U primeru je kreirana jedna for petlja, koja će imati 100 iteracija (od 0 do 99). Ipak, unutar for petlje je definisan jedan uslov, kojim se proverava da li je brojač veći od 50. Ukoliko je brojač veći od 50, poziva se naredba continue. S obzirom da ova naredba odmah prelazi na izvršavanje sledeće iteracije, linija za ispis vrednosti brojača se neće izvršiti u slučaju da je brojač veći od 50. Sve ovo proizvodi sledeći rezultat:

|  |  |
| --- | --- |
| 1  2 | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50  100 |

Analizom izlaza se može utvrditi da se linija za ispis vrednosti brojača unutar tela petlje izvršava do vrednosti 50. Petlja nastavlja da se izvršava do kraja, ali ne ispisuje vrednosti, u šta se možemo uveriti poslednjim ispisanim brojem. To je broj 100, koji je ispisan od strane linije koja se nalazi izvan petlje. Ovo praktično znači da je vrednost brojača nakon završetka petlje 100, što dokazuje da se petlja izvršila u celosti.

Za razliku od naredbe continue, koja preskače jednu iteraciju, naredba break preskače sve iteracije i izlazi iz petlje. Drugim rečima, naredba break završava petlju. Sledeći primer ilustruje upotrebu naredbe break:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | for(var i=0;i<100;i++){   if(i>50){    break;   }    console.log(i);  }  console.log(i); |

Ovoga puta kod proizvodi sledeći izlaz:

|  |  |
| --- | --- |
| 1  2 | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50  51 |

Sada se jasno može videti razlika između naredbi continue i break. Kada brojač dostigne vrednost 50, napušta se izvršavanje petlje. Brojač se još jednom, poslednji put uvećava i dobija vrednost 51. Može se zaključiti da se petlja ne izvršava u celosti.