**Uslovno izvršavanje**

**00:15:30**

Jedinica: 22 od 30

**+Rezime**

JavaScript jezik omogućava da se više izjava grupiše u jednu logičku celinu koja se naziva blok. Blok koda je osnovni gradivni element mnogih jezičkih konstrukta u JavaScriptu. Bez blokova se ne može zamisliti ostvarivanje kontrole toka ili funkcionalno programiranje. To su sve pojmovi koji će biti objašnjeni u narednim lekcijama. Za sada je dovoljno znati da je više izjava moguće grupisati u jednu celinu.

Blok koda se u JavaScript jeziku formira korišćenjem vitičastih zagrada:

|  |  |
| --- | --- |
| 1  2  3 | {  //this is one block  } |

Primer bloka koda koji uistinu i sadrži neki kod je sledeći:

|  |  |
| --- | --- |
| 1  2  3  4 | {    var x=1;    var y=6;  } |

Blokovi koda otvaraju mogućnost za realizovanje dva veoma bitna aspekta programiranja:

* kontrola toka
* [funkcionalno programiranje](https://www.link-elearning.com/linkdl/opisPojma.php?id=144350)

**Kontrola toka**

Izvršavanje JavaScript koda obavlja se sleva nadesno i pritom se napisane izjave izvršavaju redom, jedna za drugom. Pogledajmo sledeći primer:

|  |  |
| --- | --- |
| 1  2  3 | var x=1;  var y;  y=x; |

U prikazanom primeru definisane su tri JS izjave. One će se izvršiti redom kojim su napisane, tako da će prvo biti deklarisana i inicijalizovana promenljiva x, zatim će biti deklarisana promenljiva y, i na kraju će vrednost promenljive x biti dodeljena promenljivoj y.

Zamisliti sada situaciju u kojoj biste vi mogli da kontrolišete tok izvršavanja prikazanog koda. Na primer, želite da se nakon izvršavanja prve izjave, pređe na izvršavanje treće. Ili, na primer, želite da se izvršavanje sve tri izjave ponovi 10 puta. Svi opisani scenariji spadaju u domen kontrole toka, a jezik JavaScript poznaje različite konstrukte koji omogućavaju da se postigne kontrola toka.

Za kontrolisanje toka izvršavanja koda generalno postoje dve konstrukcije, koje se koriste u zavisnosti od toga šta je potrebno postići:

* grananje (uslovno izvršavanje)
* petlje (ponavljanje)

**Grananje (uslovno izvršavanje)**

Grananje omogućava da se određene naredbe izvrše samo u slučaju zadovoljenja nekog uslova. JavaScript poznaje dve vrste kondicionalnih naredbi:

|  |  |
| --- | --- |
| 1  2 | if...else  switch |

**if...else naredba**

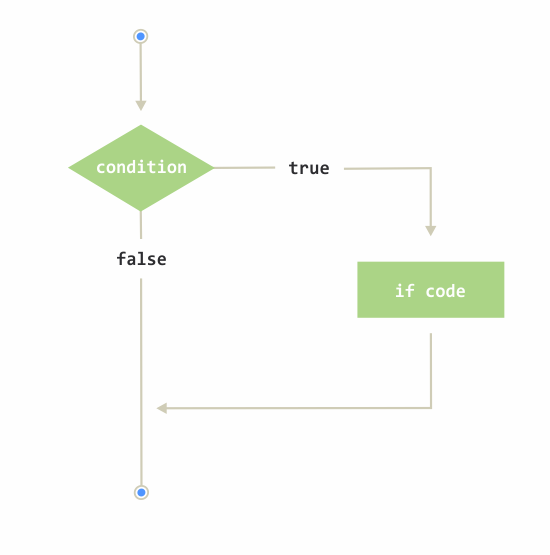
Osnovna naredba grananja u JavaScript jeziku jeste naredba if. Ova naredba poznata je u praktično svim jezicima, a svrha joj je oslikana u samom nazivu: *ako*.

Reč if (*ako*), sama za sebe, nema baš mnogo logike u retorici, a tako je i u programiranju. Naime, pored ove ključne reči if, potrebno je postaviti i neki uslov od koga će zavisiti ishod ove naredbe. Tako opšta sintaksa if naredbe izgleda ovako:

|  |  |
| --- | --- |
| 1  2  3 | if(condition){            //code to execute if condition is met  } |

Za formiranje if naredbe koristi se ključna reč if, nakon koje se u zagradama navodi uslov. U slučaju ispunjenja uslova, izvršava se blok koda if naredbe.

Slika ilustruje logiku izvršavanja programa kada se upotrebljava if naredba.



*Slika 22.1 – If naredba*

Kao što se sa slike 22.1 može videti, izvršavanje programa dolazi do dela nazvanog *condition* (uslov). U ovom delu se proverava definisani uslov, i ukoliko je on ispunjen, izvršava se kod if bloka. U protivnom, if blok koda se preskače, a izvršavanje se nastavlja na prvoj sledećoj liniji. Primer koji ilustruje upotrebu if naredbe je sledeći:

|  |  |
| --- | --- |
| 1  2  3  4 | var speed = 9;  if (speed < 10) {  document.write("Too slow...");  } |

Na početku je deklarisana i inicijalizovana promenljiva sa nazivom speed i vrednošću 9. Nakon ove linije, sledi if naredba. Uslov je definisan tako da se proverava da li je vrednost promenljive speedmanja od 10. Ukoliko je ovakav uslov ispunjen, izvršava se blok koda između vitičastih zagrada, koji se u primeru sastoji iz jedne linije koda.

Da je kojim slučajem vrednost promenljive speed bila veća ili jednaka broju 10, blok koda iz primera se ne bi izvršio.

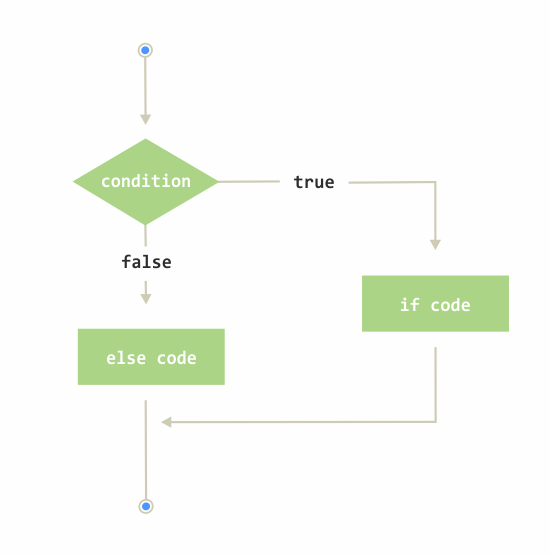
U ovom trenutku se može postaviti jedno pitanje: šta ukoliko je potrebno izvršiti određeni blok koda i kada uslov nije ispunjen?

Kao odgovor na ovo pitanje, na scenu stupa ključna reč else, a naredba if postaje naredba if…else.

**if...else** naredba koristi se za kontrolu toka, tako što omogućava da se definiše logika koja će se izvršiti i ukoliko je uslov ispunjen, ali i ukoliko nije. Sintaksa if…else naredbe je sledeća:

|  |  |
| --- | --- |
| 1 | if (condition) {  statement\_1; } else {  statement\_2; } |

U zagradama nakon if ključne reči navodi se uslov (condition). U slučaju ispunjenja ovakvog uslova, izvršava se prva izjava (statement\_1), a u protivnom druga (statement\_2).  
Sve ovo ilustrovano je slikom 22.2.



*Slika 22.2 – If…else naredba*

Sa slike 22.2 se može videti da u ovoj situaciji postoje dva bloka koda: *if code i else code*. U zavisnosti od ispunjenja uslova, izvršava se jedan od ova dva bloka koda. Interesantno je napomenuti da se jedan blok koda mora izvršiti.

Primer ilustruje upotrebu if..else naredbe:

|  |  |
| --- | --- |
| 1  2  3  4  5 | if (1==1) {   alert("true");  } else {   alert("false");  } |

U slučaju da je 1 jednako 1, na stranici će se pojaviti popup sa porukom true, u protivnom, popup sa porukom false. Naravno, s obzirom da je uslov ispunjen, izvršiće se kod unutar if bloka, odnosno, na stranici će se pojaviti popup sa porukom true.

Često se unutar if…else naredbe, unutar uslova, proverava jednakost nekih promenljivih:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var x=10;  var z=13;    if (x==z) {   alert("true");  } else {   alert("false");  } |

Jedna if…else naredba može imati samo jedan if i samo jedan else blok. Uslov se uvek definiše samo na if bloku.

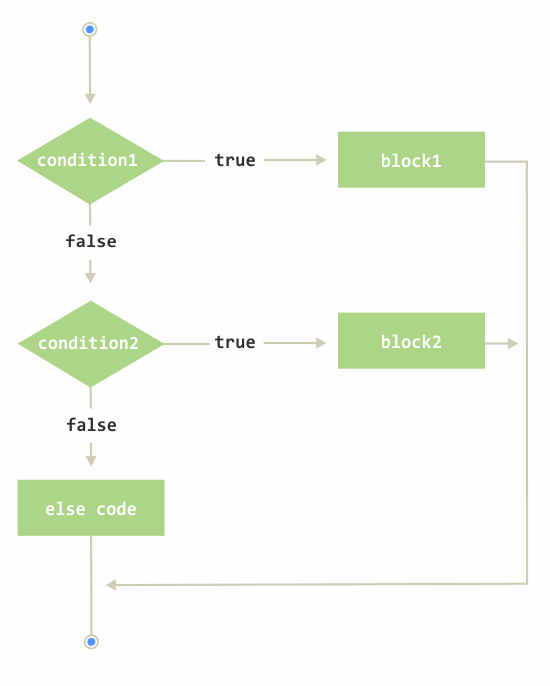
Šta se dešava ukoliko je potrebno definisati dodatne blokove koda sa sopstvenim uslovima? Kao odgovor na ovo pitanje, na scenu stupa rezervisan skup reči else if.

**Else if** omogućava da se prilikom kreiranja if…else naredbe definišu dodatni uslovi, tako da naredba if…else postaje if…else if…else naredba.

Sintaksa ovakve naredbe je sledeća:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | if (condition\_1) {    statement\_1;  } else if (condition\_2) {    statement\_2;  } else if (condition\_n) {    statement\_n;  } else {    statement\_last;  } |

U ovom primeru, pored if i else blokova, postoje i dva else if bloka. Ukoliko je prvi uslov (condition\_1) ispunjen, izvršiće se izjava statement\_1. Ukoliko je uslov condition\_2ispunjen, izvršiće se izjava statement\_2. Ukoliko je uslov condition\_n ispunjen, izvršiće se naredba statement\_n. Na kraju, ukoliko nijedan uslov nije ispunjen, izvršiće se izjava statement\_last. Sve ovo je ilustrovano slikom 22.3.



*Slika 22.3 – If…else if…else naredba*

Izvršavanje koda dolazi do prvog uslova. Ukoliko je uslov ispunjen, izvršava se block1, i nakon toga se izlazi iz if…else if…else konstrukcije. Ukoliko prvi uslov (condition1) nije ispunjen, ispituje se tačnost drugog uslova (condition2). Ukoliko je ispunjen, izvršava se block2. Na kraju, ukoliko ni prvi ni drugi uslov nisu ispunjeni, izvršava se else blok koda.

Primer korišćenja if…else if…else naredbe je sledeći:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | var speed = 50;  if (speed < 10) {  document.write("Too slow...");  }else if(speed <= 80){  document.write("Regular speed.");  }else if(speed < 100){  document.write("Too fast!");  }else{  document.write("Incorrect variable");  } |

Na početku koda, u prvoj liniji, deklariše se i inicijalizuje promenljiva speed sa vrednošću 50. If…else if…else naredba poseduje tri uslova. Ukoliko je vrednost promenljive speed manja od 10, izvršava se prvi blok koda. S obzirom da to nije slučaj, ispituje se istinitost drugog uslova. Drugi uslov je istinit ukoliko je vrednost promenljive speed manja ili jednaka broju 80. Ovaj uslov je ispunjen, s obzirom da je 50 manje od 80, pa izvršavanje programa ulazi u prvi else if blok i ispisuje se poruka *regular speed*.

Još jedna situacija je vrlo zanimljiva. Ukoliko se pogleda sledeći uslov (speed < 100), može se zaključiti da je i on ispunjen, s obzirom da je 50 manje od 100. Ipak, unutar naredbe if…else if…else, izvršava se uvek samo jedan blok koda, i to prvi čiji uslov se pokaže kao istinit.

**Unutar kog bloka se ne može definisati uslov?**

 If

 Else

 else if

**Switch**

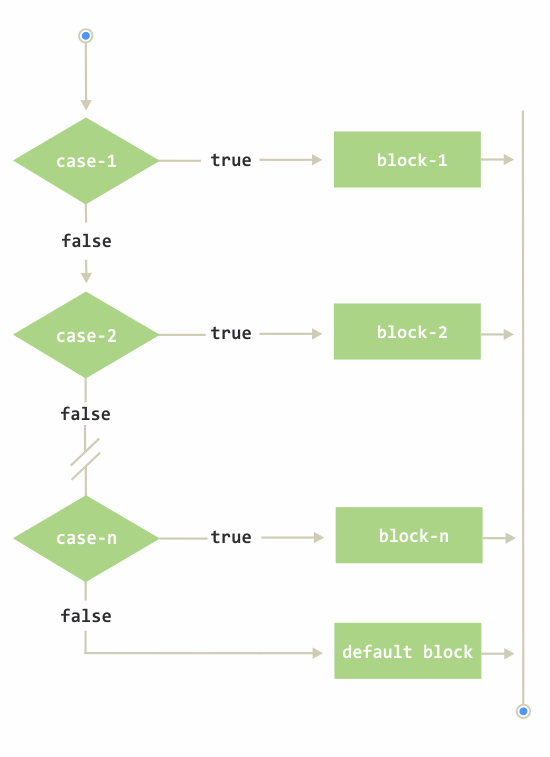
Prethodni primer može se značajno lakše i preglednije napisati korišćenjem naredbe switch. Switch naredba omogućava postizanje kontrole toka, na osnovu podudarnosti vrednosti. Definicija nije preterano slikovita, ali sve postaje jasnije analizom sintakse switch naredbe:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | switch (expression) {    case label\_1:      statements\_1      break;    case label\_2:      statements\_2      break;      ...    default:      statements\_def      break;  } |

Kod switch naredbe, JavaScript poredi expression vrednost sa vrednostima case klauzula. U primeru koda, vrednosti case klauzula su obeležene sa label\_1, label\_2 itd.

Kada se utvrdi podudaranje expression vrednosti sa nekom od case klauzula, izvršavanje programa se preusmerava na taj blok. U slučaju da se ne pronađu podudaranja, izvršava se opcioni default blok.

Logika naredbe ilustrovana je slikom 22.4.



*Slika 22.4 – Switch naredba*

Praktičan primer upotrebe switch naredbe:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | var x = 1;      switch(x){          case 0:              document.write("x is 0");              break;          case 1:              document.write("x is 1");              break;          default:              document.write("Unknown value.");    break;      } |

Na početku primera, deklariše se i inicijalizuje promenljiva x sa vrednošću 1. Zatim je definisana jedna switch naredba, sa dva case bloka i jednim podrazumevanim default blokom. S obzirom da je vrednost promenljive x jednaka 1, izvršava se središnji blok koda, odnosno na izlazu se ispisuje x is 1. Svaki od blokova koda unutar switch naredbe (ima ih 3) na kraju poseduje ključnu reč break. Uloga ključne reči break je da izvede izvršavanje koda izvan switch naredbe. Drugim rečima, kada interpreter naiđe na ovu liniju, završava se izvršavanje switch naredbe.

|  |
| --- |
| **Napomena**  *Default blok koda se, po konvenciji, postavlja na kraj switch naredbe, ali to ne mora biti pravilo. Ovaj deo koda je moguće postaviti bilo gde.* |

**Višestruko poklapanje kod switch naredbe**

U svim dosadašnjim primerima izvršavao se blok koda kao rezultat jednostrukog poklapanja. Na primer, ukoliko je x jednako 1 i samo 1, izvršava se određeni blok kod. Ukoliko je x jednako 2 i samo 2, izvršava se drugi blok koda. A šta ukoliko je potrebno da se neki blok koda izvrši ukoliko x ima bilo koju od vrednosti 1 ili 2? U takvoj situaciji mora se pribeći višestrukom poklapanju.

*Usecase* primer biće sledeći:

*Potrebno je napraviti switch strukturu koja će testirati vrednost promenljive x. Ako x ima vrednost 1 ili 2, na strani napisati „YES“, dok za svaku drugu vrednost promenljive x treba ispisati „NOT“.*

Naravno, ponavljanje istog bloka koda za različite vrednosti ne dolazi u obzir.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | var x = 1;  switch(x){  case 1:  case 2:  document.write("YES");  break;  default:  document.write("NOT");  break;      } |

Na ovaj način, prvi blok koda će se izvršiti kada promenljiva x ima vrednosti 1 ili 2. Za sve ostale vrednosti, izvršava se default blok koda.

Drugi način da bi se nešto ovako postiglo:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var x = 1;  switch (true) {  case x >= 1 && x < 3:  document.write('YES');  break;  default:  document.write('NOT');  } |

Za definisanje uslova case klauzule može se koristiti znanje stečeno u jednoj od prethodnih lekcija, kada je bilo reči o operatorima poređenja i logičkim operatorima. Ovoga puta je kao vrednost caseklauzule postavljen jedan logički izraz, koji će evaluirati u true ili false vrednost. Zato je u samom switch uslovu i postavljena vrednost true. To praktično znači da će se izvršiti onaj blok koda čiji se izraz transformiše u true. Da li će se izraz case klauzule transformisati u true ili false, zavisi od vrednosti  promenljive x. Drugim rečima, ukoliko je x veće ili jednako od 1, a manje od 3, uslov će biti zadovoljen, pa će se izvršiti prvi blok koda. U protivnom se izvršava defaultbloka koda.

**Ternarni operator**

Ternarni operator predstavljen je u prethodnoj lekciji. Tada je rečeno da je ternarni operator veoma koristan u situacijama kada je potrebno ostvariti kontrolu toka. Da budemo precizniji, korišćenjem ternarnog operatora, moguće je promenljivoj dodeliti vrednost na osnovu nekog uslova.

Logika koja se postiže ternarnim operatorom može se zameniti jednostavnom if…else naredbom. Dat je sledeći primer:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var a=10;  var b;    if(a==10){  var b=a;  }else{  var b=150;  } |

Prikazani kod ima sledeće značenje:

*Vrednost promenljive a je 10. Ukoliko je vrednost promenljiva a 10, dodeli vrednost promenljive a promenljivoj b. U svim ostalim situacijama, promenljivoj b dodeli vrednost 150.*

Prikazani primer se korišćenjem ternarnog operatora transformiše:

|  |  |
| --- | --- |
| 1  2 | var a=10;  var b=(a==10) ? a : 150; |

S obzirom da ternarni operator poznajemo od ranije, sve je jasno. Uslov je (a==10), moguće vrednosti koje bi mogle da budu dodeljene promenljivoj b su a i 150. Ukoliko je a jednako 10, bdobija vrednost a. Ukoliko a nije jednako 10, b dobija vrednost 150.

Prikazani pristup je veoma koristan kada je potrebno postaviti podrazumevane vrednosti nekih promenljivih, u slučaju da njihove vrednosti nisu nigde drugde navedene. Na primer, neka vrednost može doći sa korisničke forme na stranici. Korisnik može da takvu vrednost ne unese, a ternarni operator će se pobrinuti da takva promenljiva dobije svoju podrazumevanu vrednost:

|  |  |
| --- | --- |
| 1 | b=(a!=null) ? a : 120; |

Ukoliko a nije definisano, b dobija podrazumevanu vrednost 120. Ukoliko je a definisano, b dobija vrednost a.