Pojam promenljivih i konstanti

Šta su promenljive?

Promenljive, kako im i samo ime kaže, služe kao kontejneri za vrednosti koje se koriste prilikom izvršavanja koda. U kreiranju promenljivih mogu da učestvuju specifične ključne reči, identifikatori i vrednosti.

Deklaracija i inicijalizacija

Upravo prikazana linija koda ilustruje kreiranje jedne promenljive. Ipak, vrlo je bitno znati da se navedenim kodom zapravo obavljaju dve operacije:

- Deklaracija
- Inicijalizacija

Deklaracija predstavlja kreiranje promenljive (varijable). Kodom iz primera prikazana je varijabla sa nazivom *counter*.

Inicijalizacija je pojam koji predstavlja dodeljivanje početne vrednosti promenljivoj. Tako, promenljiva *counter* iz primera ima početnu vrednost 0.

Iz navedenog se može zaključiti da je u prikazanoj liniji obavljena i deklaracija i inicijalizacija. Deklaracija i inicijalizacija su odvojene operacije, pa ih je stoga moguće obaviti i u okviru dve izjave. Razdvajanje deklaracije i inicijalizacije na dve izjave ilustrovano je slikom:

Tipovi promenljivih i konstanti

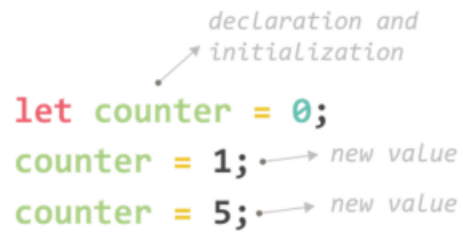
Iz upravo prikazanog primera moguće je videti da se deklaracija promenljivih obavlja korišćenjem specijalne ključne reči *let*. Ipak, to nije jedina ključna reč koju je moguće koristiti za deklarisanje promenljivih. Tabela ilustruje tri različite ključne reči koje je moguće koristiti za obavljanje takvog posla:

Ključna reč	Opis
`var`	deklariše promenljivu
`let`	deklariše lokalnu promenljivu
`const`	deklariše konstantu čiju vrednost je moguće samo čitati

Tabela ukratko definiše osnovne osobine tri različite ključne reči za deklaraciju promenljivih:

- Ključne reči *let* i *var* koriste se za deklaraciju promenljivih
- Ključna reč *const* koristi se za deklaraciju konstanti

Konstante i promenljive su veoma slične, ali kako im i samo ime kaže poseduju neke drugačije osobine. U osnovi samog naziva promenljive jeste mogućnost redefinisanja, odnosno promenljivosti. To praktično znači da se vrednost promenljivih tokom izvršavanja koda može menjati, pa otuda i nazivi promenljiva i varijabla. Primer to ilustruje:



Primer ilustruje proces redefinisanja vrednosti jedne promenljive. U prvoj liniji koda obavljeno je deklarisanje i inicijalizovanje promenljive sa nazivom *counter*. Početna vrednost je postavljena na 0. Već u sledećoj naredbi obavljeno je redefinisanje vrednosti. Stoga je, nakon druge linije koda, nova vrednost *counter* promenljive 1. Na kraju, u trećoj liniji obavlja se još jedno redefinisanje vrednosti. Promenljiva *counter* dobija vrednost 5.

Ispis vrednosti promenljivih

S obzirom na to da u ovoj lekciji prvi put započinjemo rad sa vrednostima korišćenjem JavaScript jezika, u nastavku će biti prikazano nekoliko načina na koje je moguće obaviti ispis takvih vrednosti. Pristupi koji će biti prikazani mogu vam poslužiti za testiranje i lakše upoznavanje osobina promenljivih.

Prvi način za ispis vrednosti podrazumeva njihov ispis direktno unutar `body` dela HTML dokumenta. On se postiže na sledeći način:

```
document.write("Hello World!");
```

Prikazana linija koda će unutar `body` dela HTML dokumenta obaviti ispis teksta Hello World!.

Prikazani primer je ilustrovao ispis unapred definisanog teksta. Ipak, prikazani pristup se može koristiti i za ispis vrednosti promenljivih:

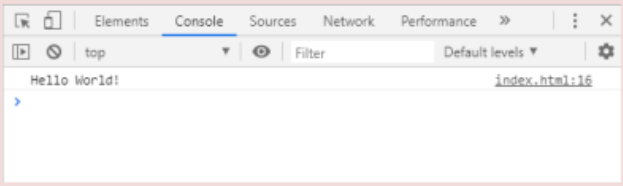
```
let message = "Hello World!";
document.write(message);
```

Sada je prvom linijom koda izvršena deklaracija i inicijalizacija promenljive sa nazivom `message` i vrednošću `Hello Word!`. Zatim je, drugom linijom koda, obavljen ispis takve vrednosti. Oba primera na stranici proizvode identičan efekat.

Postoje još neki pristupi koje je moguće koristiti za ispis vrednosti. Vrednost je moguće ispisati unutar konzole (engl. console) web pregledača. Za obavljanje takvog posla dovoljno je uraditi sledeće:

```
console.log("Hello World!");
```

Na ovaj način, vrednost `Hello World` će biti ispisana unutar konzole (slika 4.4).



Slika 4.4. Ispis vrednosti unutar konzole

Na kraju, vrednost je jednostavno moguće emitovati i korišćenjem funkcije `alert()`. Ona omogućava prikaz vrednosti unutar posebnog modalnog prozora:

```
alert("Hello World!");
```

Već je prikazano da se u JavaScript kodu vrednost promenljivih može redefinisati neograničeni broj puta. Ipak, isto ne važi i za konstante.

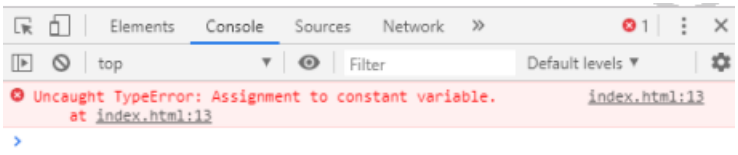
Konstanta je vrednost koja se ne može menjati nakon inicijalizacije. Za njeno deklarisanje koristi se ključna reč *const*. Primer deklarisanja konstante je sledeći:

const COLOR_1 = „#026873“;

Na ovaj način je reklarisana konstanta sa nazivom *COLOR_1* i vrednošću *#026873*. Time je osigurano da su i naziv i vrednost konstante rezervisani. Drugim rečima, ukoliko se, nakon deklarisanja i inicijalizacije konstante, napiše sledeća linija koda, doći će do greške:

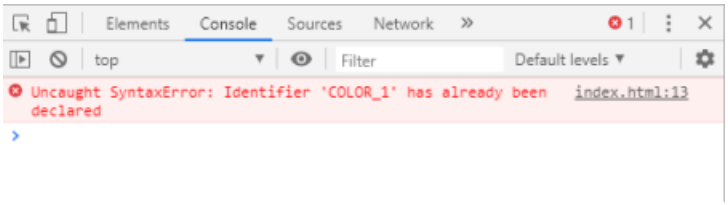
COLOR_1 = „#A9CE83“

Greška se može videti unutar konzole web pregledača



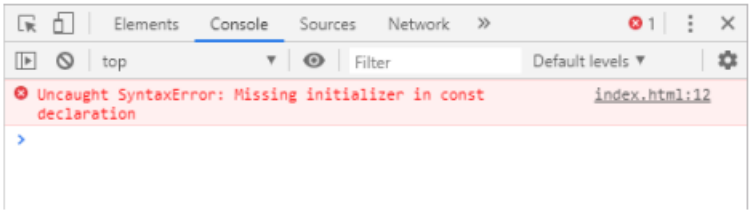
Kada se definiše konstanta određenog imena, zabranjeno je ponovo deklarisati promenljive ili konstante sa istim imenom. Upravo zbog toga će i sledeća linija proizvesti grešku:

const COLOR_1 = „#A9CE83“;



Takođe, konstanta se prilikom deklarisanja mora i inicijalizovati. Stoga deklaracija konstante bez inicijalizacije proizvodi grešku:

const COLOR_1;



Imenovanje promenljivih i konstanti

U prethodnom modulu definisan je pojam identifikatora. Tada je rečeno da se identifikatori koriste za imenovanje različitih jezičkih elemenata, u koje se ubrajaju i promenljive i konstante. Stoga za imenovanje promenljivih važe identična pravila kao i za formulisanje bilo kojih drugih identifikatora:

- Nazivi promenljivih i identifikatora mogu biti sastavljeni iz slova, brojeva i karaktera donja crta i dolar
- Naziv promenljive i konstante ne sme započeti brojem, odnosno mora započeti slovom, donjom crtom (_) ili karakterom dolar ($)
- Nazivi promenljivih i konstanti su osetljivi na mala i velika slova
- Prilikom formulisanja naziva promenljivih i konstanti koristi se UNICODE set karaktera
- Ključne reči ne mogu biti nazivi promenljivih i konstanti

U prethodnim redovima su navedena sintaksna pravila koja se tiču imenovanja promenljivih i konstanti. To su pravila koja se moraju poštovati, kako bi JavaScript uopšte mogao da bude funkcionalan. Pored ovih striktnih pravila, nad imenovanjem promenljivih i konstanti primenjuju se i različite konvencije imenovanja.

Konvencija imenovanja se odnosi na dobru praksu koju je potrebno koristiti prilikom formulisanja naziva promenljivih, konstanti i drugih jezičkih elemenata, a sve kako bi se poboljšala čitljivost i olakšalo održavanje. Konvencija imenovanja je pojam koji je karakterističan za programiranje, bez obzira na jezik koji se koristi. Postoji nekoliko različitih konvencija imenovanja. U JavaScript svetu najpopularnija je takozvana **camelCase** konvencija. Ona nalaže sledeća pravila:

- Nazivi promenljivih započinju malim slovom
- Ukoliko se naziv promenljive sastoji iz više reči, svaka naredna reč započinje velikim slovom
- Reči u nazivu promenljivih se ne razdvajaju nikakvih posebnim karakterom

Primeri naziva promenljivih poštovanjem pravila camelCase notacije:

- myVariable
- primaryColor
- dateOfBirth
- color

Kada je u pitanju imenovanje konstanti, možda ste do sada primetili da je njihovo imenovanje izgledalo nešto drugačije. Preporuka je da se konstante imenuju korišćenjem velikih slova, kako bi bile uočljivije. Tako su u uvodnom primeru nazivi konstanti formulisani na sledeći način:

- COLOR_1
- COLOR_2

S obzirom na to da se za imenovanje konstanti koriste velika slova, u slučaju postojanja naziva sačinjenih iz više reči pojedinačne reči se razdvajaju karakterom donja crta (_).

Napomena

Iako je sintaksno potpuno tačno započeti naziv promenljive karakterom dolar ($), savetuje se izbegavanje takve prakse. Nekada je ovim karakterom započinjao automatski generisan kôd, odnosno kôd koji nije pisao programer. Tako je početni karakter dolar ukazivao da je reč o kodu koji je generisao neki drugi program. Danas mnoge JavaScript biblioteke koriste ovaj karakter interno, kao početak naziva svojih identifikatora. Zbog toga može doći do konflikta između naziva vaših i promenljivih iz biblioteka. Na kraju, karakter dolar se veoma često koristi kao početak naziva promenljivih koje čuvaju jQuery objekat.

Podrazumevane vrednosti

Nešto ranije je bilo reči o dve operacije koje je moguće izvesti prilikom kreiranja promenljivih – deklaracija i inicijalizacija.

Deklaracija jedne promenljive izgleda ovako:

let counter;

S obzirom na to da nije obavljena inicijalizacija, logično je očekivati da promenljiva *counter* ne poseduje nikakvu vrednost. Ipak to nije tako.

document.write(counter);

Ukoliko ispišemo vrednost upravo deklarisane promenljive, dobija se sledeće:

Undefined

Undefined je specijalna vrednost koju imaju promenljive koje su deklarisane, ali čija vrednost još uvek nije inicijalizovana.

Oblast važenja

Kada se govori o promenljivama i konstantama, veoma važna je i oblast njihovog važenja. *Oblast važenja* je pojam koji određuje sa kojih mesta u kodu će neka promenljiva ili konstanta biti vidljiva. Takođe, oblast važenja je jedan od osnovnih pojmova koji unosi razliku između nešto ranije navedenih ključnih reči za deklarisanje promenljivih.

Tako se može reći da upotreba različitih ključnih reči za deklarisanje, određuje oblast važenja:

- **var** – važenje samo u okviru jedne funkcije
- **let** – važenje samo u okviru bilo kojeg bloka
- **const** – važenje samo u okviru bilo kojeg bloka

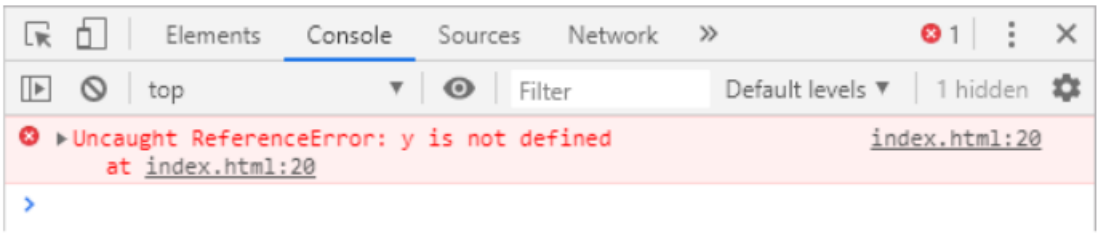
Oblast važenja je najlakše razumeti analizom koda. Stoga će u nastavku biti izneto nekoliko primera.

```
let x = "globalna promenljiva";

{

let y = "blok promenljiva";

}

document.write(x);

document.write(y);
```

Unutar prikazanog koda deklarisane su i inicijalizovane dve promenljive. Deklaracija je obavljena korišćenjem ključne reči *let*. Pored promenljivih, primer poseduje i jedan blok koda. Zapravo, druga promenljiva (y) nalazi se unutar takvog bloka. Na kraju primera napisane su i dve linije za ispis vrednosti promenljivih *x* i *y*. Unutar HTML dokumenta biće ispisano:

globalna promenljiva

Unutar HTML dokumenta biva ispisana vrednost promenljive *x*, ali ne i promenljive *y*. Šta se zapravo dogodilo može se videti unutar konzole web pregledača.



Na slici se može videti greška koja se dobija kada JavaScript pokuša da pristupi promenljivoj *y*. Jasno se stavlja do znanja da takva promenljiva nije definisana. Zapravo, ona jeste definisana, ali unutar bloka, pa je njena vidljivost samo unutar takvog bloka. Izvan bloka u kojem je definisana ona nije vidljiva, pa se stoga kaže da *let* promenljive imaju važenje unutar jednog bloka.

Potpuno je drugačija situacija kada su u pitanju promenljive deklarisane *var* ključnom rečju. Unutar već prikazanog primera dve ključne reči *let* biće zamenjene ključnim rečima *var*:

```
var x = “globalna promenljiva”;
{
var y = “blok promenljiva”;
}
document.write(x);
document.write(y);
```

Ovoga puta unutar HTML dokumenta biće ispisane vrednosti i promenljive *x* i promenljive *y:*

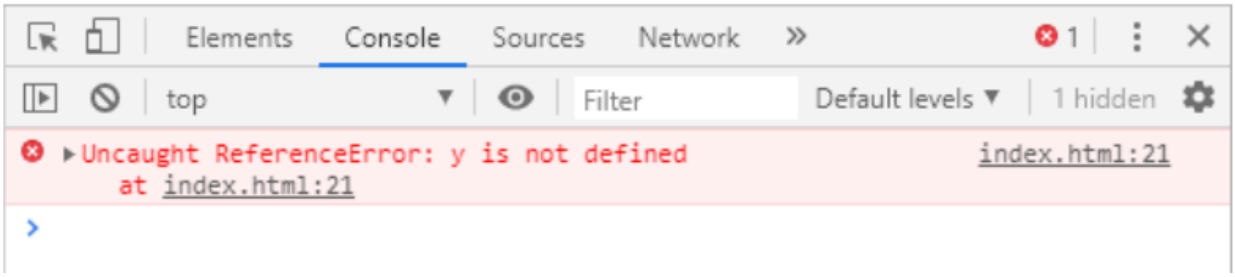
globalna promenljiva blok promenljiva

Razlog za ovakvo ponašanje je i više nego jednostavan. Baš kao što je malopre rečeno, oblast važenja *let* varijabli jeste jedan blok. Upravo zbog toga nešto ranije nije dobijena vrednost *let* varijable definisane unutar bloka. Sa druge strane, kod promenljivih definisanih ključnom rečju *var* ovakvog ponašanja nema, pa i varijable deklarisane unutar bloka postaju globalne. Ipak, i *var* promenljive poseduju oblast važenja – njihova oblast važenja je jedna funkcija.

```
var x = "globalna promenljiva";
function myFunction() {
  var y = "promenljiva u funkciji";
}
document.write(x);
document.write(y);
```

Primer je sada neznatno izmenjen u odnosu na prethodni. Blok koji je do sada korišćen zamenjen je jednom posebnom vrstom bloka – funkcijom. S obzirom na to da je oblast važenja *var* promenljivih jedna funkcija, ovakav kod proizvešće sledeći rezultat:

globalna promenljiva



Jasno je da se *var* promenljivoj definisanoj unutar funkcije ne može pristupiti izvan takve funkcije.

Napomena: Konstante koje se deklarišu korišćenjem ključne reči *const* imaju identičnu oblast važenja kao i promenljive definisane korišćenjem ključne reči *let*.

Podizanje varijabli na početak dokumenta

Do sada je nekoliko puta rečeno da JavaScript kod izvršava naredbu po naredbu, odozgo nadole i sleva nadesno:

document.write(x);

var x;

Prikazane su dve linije koda koje smo do sada već viđali: linija za ispis vrednosti promenljive i linija za deklaraciju. Ipak, ovoga puta su one napisane drugačijim redosledom.

Prvo je navedena naredba za ispis vrednosti, a tek onda naredba za deklaraciju. U ovakvoj situaciji je potpuno očekivano da dođe do pojave greške. Ipak, do pojave greške ne dolazi:

Undefined

Unutar HTML dokumenta dobija se ispis *undefined*, što je podrazumevana vrednost promenljivih. Iz ovoga se može zaključiti da, iako je deklarisana nakon naredbe za ispis, promenljiva poseduje podrazumevanu vrednost. Ovakva pojava se naziva *hoisting*, odnosno podizanje promenljivih na vrh tekućeg bloka ili dokumenta.

Hoisting je pojam koji se odnosi na podizanje deklaracije promenljivih na vrh tekućeg bloka ili kompletne skripte, ukoliko je promenljiva definisana izvan bilo kakvog bloka. To praktično znači da web pregledači nekoliko puta prolaze kroz JavaScript koda pre nego što započne njegovo izvršavanje. U prvom prolasku se analiziraju promenljive i obavlja se njihovo podizanje na vrh tekućeg bloka. Tek nakon toga se obavlja izvršavanje konkretnih naredbi. Tako je *hoisting* razlog zbog kog se unutar upravo prikazanog primera ne dobija greška, već podrazumevana vrednost promenljive.

Veoma je bitno napomenuti da se u procesu podizanja promenljivih na vrh dokumenta ili bloka isključivo obavlja podizanje deklaracije, ali ne i inicijalizacije:

document.write(x);

var x = "vrednost promenljive";

Sada je, pored deklaracije, obavljeno i inicijalizovanje promenljive *x*. Ipak, unutar HTML dokumenta i dalje se dobija:

Undefined

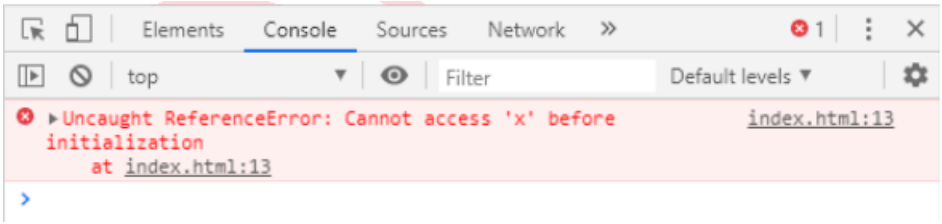
Ispis unutar dokumenta je dokaz da *hoisting* podrazumeva podizanje samo deklaracije, ali ne i inicijalizacije.

Do sada je *hoisting* ilustrovan samo na primeru *var* promenljivih. Kod promenljivih deklarisanih ključnom rečju *let* situacija je nešto drugačija:

document.write(x);

let x = "vrednost promenljive";

Sada je ključna reč *var* zamenjena ključnom rečju *let*. Efekat će biti kao na slici:



Sa slike se može videti da je ponašanje *let* varijabli nešto drugačije. Kada se pokuša pristupiti *let* varijabli pre njenog deklarisanja, dobija se greška. Ipak, to ne znači da se podizanje ne primenjuje nad *let* promenljivim. Kako bi upravo izrečeno bilo dokazano, dat je sledeći primer:

```
let x = "izvan bloka";

{

  document.write(x);

}
```

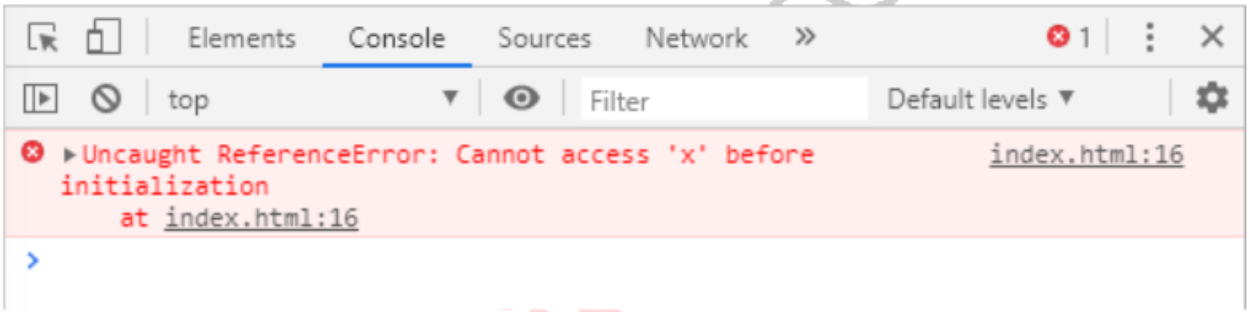
Promenljiva je deklarisana izvan bloka, dok se unutar bloka nalazi linija za ispis njene vrednosti. Rezultat će biti očekivan:

izvan bloka

Sada će unutar bloka biti dodata još jedna naredba sa deklaracijom i inicijalizacijom promenljive:

```
let x = "izvan bloka";

{

  document.write(x);

  let x = 'unutar bloka';

}
```

Unutar bloka sada je, nakon naredbe za ispis vrednosti promenljive x, postavljena naredba za njeno redefinisanje. Rezultat je ilustrovan slikom:



Umesto da se na izlazu dobije vrednost već deklarisane *x* promenljive izvan bloka, dolazi do emitovanja greške koja je već viđena u prethodnom primeru. Drugim rečima, izvršno okruženje je svesno postojanja lokalne promenljive *x*, koja je deklarisana ispod naredbe za ispis i to upravo zbog procesa podizanja. Tako se na kraju može rezimirati da se nad *let* i *var* promenljivama primenjuje postupak podizanja, ali da je efekat koji proizvodi pokušaj njihovom pristupanju ranije u kodu nešto drugačiji.

Napomena: konstante definisane ključnom rečju *const* ponašaju se na identičan način kao i promenljive *let* kada je u pitanju hoisting.

Rezime:

- Promenljive služe kao kontejneri za vrednosti koje se koriste prilikom izvršavanja koda.
- Deklaracija predstavlja kreiranje promenljive (varijable).
- Inicijalizacija je dodeljivanje početne vrednosti promenljivoj.
- JavaScript poseduje tri specifične ključne reči: *var* i *let* za deklaraciju promenljivih i *const* za deklaraciju konstanti.
- Konstanta je vrednost koja se ne može menjati nakon inicijalizacije.
- JavaScript konstante imaju sledeće osobine: prilikom deklaracije se moraju inicijalizovati, njihova vrednost se ne može redefinisati i nije moguće definisati neku drugu konstantu ili promenljivu sa istim nazivom.
- camelCase konvencija imenovanja promenljivih nalaže da naziv promenljivih započinju malim slovom, da svaka reč unutar naziva započinje velikim slovom i da se reči u nazivu ne razdvajaju nikakvim posebnim karakterom.
- *Undefined* je specijalna vrednost koju imaju promenljive koje su deklarisane, ali čija vrednost još nije inicijalizovana.
- Oblast važenja *let* promenljivih i *const* konstanti jeste bilo koji blok

- Oblast važenja *var* promenljivih je jedna funkcija
- Hoisting je pojam koji se odnosi na podizanje promenljivih na vrh tekućeg bloka ili kompletne skripte.