

## Osnovna leksička struktura

Leksička struktura nekog programskog jezika predstavlja osnovni skup pravila koji diktira način na koji se programski kod piše. Leksička struktura se odnosi na pravila pisanja, odnosno na sintaksu, ali i način na koji se različiti jezički izrazi kombinuju i strukturiraju kako bi se dobio željeni efekat.

Kao i većina modernih jezika, JavaScript najveći deo leksičke strukture pozajmljuje od jezika Java (odnosno C), ali i nešto modernijih jezika kao što su Perl i Python.

JavaScript kod je sačinjen od izjava (naredbi) koje mogu biti grupisane u blokove. Ove naredbe govore izvršnom okruženju koji efekat je potrebno da programski kod proizvede. Jedna naredba može se sastojati iz vrednosti (litala), operatora, ključnih reči, identifikatora, komentara i raznih drugih elemenata, čijim grupisanjem se dobija entitet razumljiv JavaScript izvršnom okruženju.

Upravo opisana leksička struktura JavaScript jezika biće razmotrena na uvodnom primeru iz prve lekcije:

Prikazani primer može poslužiti za identifikaciju nekih osnovnih leksičkih elemenata JavaScript jezika.

```
const COLOR_1 = "#026873";
const COLOR_2 = "#F29484";





let containerElement = document.getElementById("main-container");

document.getElementById("button").addEventListener("click", toggleBgColor);

let counter = 0;

function toggleBgColor() {
    counter++;

    if (counter % 2 == 1) {
        containerElement.style.backgroundColor = COLOR_1;
    }
    else {
        containerElement.style.backgroundColor = COLOR_2;
    }
}
```

 reserved words (keywords)	 operators
 identifiers	 literals

Dva osnovna leksička elementa JavaScript jezika su:

- Naredbe (izjave)
- Blokovi

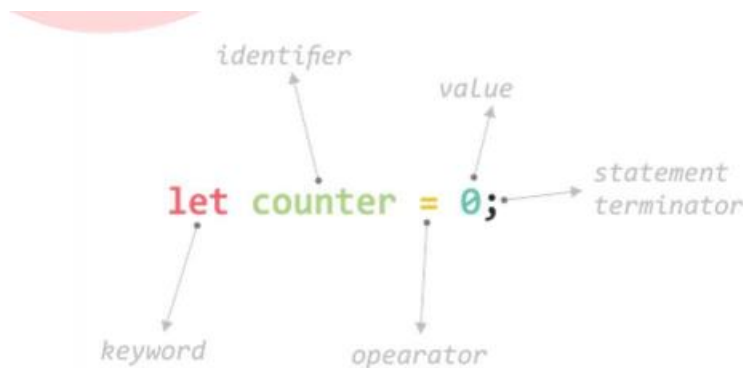
U kreiranju naredbi i blokova dalje mogu učestvovati i razni drugi leksički elementi:

- Karakteri za završavanje linija,
- Prazna mesta,
- Ključne reči,
- Identifikatori,
- Literali (vrednosti),
- Operatori
- Komentari

O svim ovim pojmovima biće reči u nastavku lekcije.

### **NAREDBE (izjave)**

Najmanja jezička jedinica koja može da proizvede neku akciju u JavaScript jeziku naziva se naredba ili izjava. U uvodnom primeru sa početka ove lekcije postoji nekoliko naredbi. Na primer, jedna od njih je prikazana sledećom slikom:



Slika ilustruje različite segmente od kojih je sastavljena jedna JavaScript izjava (naredba) – ključna reč, identifikator, operator, literal... Različiti leksički elementi od kojih mogu biti sastavljene naredbe biće objašnjeni nešto kasnije, dok ćemo se sada posvetiti pojmu naredbe kao celine.

S obzirom na to da su osnovni gradivni blokovi JavaScript jezika, naredbe se koriste za obavljanje velikog broja različitih poslova prilikom JavaScript programiranja. U primeru sa slike prikazanom naredbom se obavlja deklaracija i inicijalizacija jedne promenljive.

Sam pojam promenljive za nas je u ovom trenutku još uvek nepoznanica, pa je tako prikazanu naredbu dovoljno razumeti kao način da se određena vrednost zamapti tokom izvršavanja JavaScript koda (vrednost 0 je dodeljena identifikatoru *counter*).

Pored rada sa promenljivama, korišćenjem naredbi u JavaScript jeziku obavlja se i veliki broj drugih poslova – pozivanje funkcija, kreiranje objekata, definisanje uslova i petlji itd. Sve su to pojmovi koji će biti obrađeni u lekcijama koje slede.

JavaScript izjave završavaju se karakterom tačka sa zapetom (;). Tako se može reći da je ovaj karakter **terminator jedne izjave**. JavaScript poseduje funkcionalnost automatskog ubacivanja karaktera tačka sa zapetom na kraj izjava, što praktično znači da on nije obavezan, te da ga je moguće izostaviti. Ipak, uvek se savetuje ručno pisanje ovog karaktera, jer se time poboljšava preglednost koda i smanjuje mogućnost greške.

## PRAZNA MESTA I LINIJE

Analizom uvodnog primera koji je prikazan slikom može se primetiti da je u njegovom formiranju iskorišćeno specifično formatiranje koda. Kod je smešten u određeni broj novih redova (linija), neki semantički elementi su razdvojeni praznim mestima, dok su određene linije uvučene u odnosu na ostatak koda. Sve ovo je urađeno kako bi se dobio pregledniji i čitljiviji kod. Prazna mesta (engl. white spaces) koja su u te svrhe korišćena su:

- Razmaci
- Tabulatori
- Prelasci u novi red

Primenom specifičnog formatiranja koje poboljšava čitljivost koda dobijaju se linije. Naime, veoma često se u programiranju broj linija koristi kao jedincia za izražavanje količine nekog koda. Tako se kaže da ova skripta poseduje više od 1000 linija koda ili na 323. liniji nalazi se greška. Veoma je bitno znati da za JavaScript izvršno okruženje pojam linije, baš kao i pojam praznih mesta, nema nikakvo značenje. Naime, potpuno je legitimno napisati kod kompletnog skripta u jednoj liniji. Ipak, tako nešto se ne praktikuje, jer veoma loše utiče na čitljivost.

Uzimajući u obzir sve što je rečeno, u nastavku će biti prikazana tri primera. Prvi primer će ilustrovati dve naredbe razdvojene prelaskom u novi red:

```
let counter = 0;  
const COLOR_1 = "#026873";
```

Slika 1

Slika 1 ilustruje dve naredbe koje su napisane u dve linije JavaScript koda. Ovakve dve naredbe se mogu napisati i kao na slici 2:

```
let counter = 0;const COLOR_1 = "#026873";
```

Slika 2

Moguće je otići i korak dalje i napisati nešto kao na slici 3:

```
let counter = 0;

const COLOR_1 = "#026873";
```

Slika 3

Slika 3 ilustruje dve naredbe u posebnim linijama, razdvojene još jednom potpuno praznom linijom, pri čemu je druga naredba korišćenjem tabulatora (tab) uvučena u odnosu na prvu naredbu.

Na kraju, potrebno je razumeti da **svi primeri iz ovog poglavlja proizvode identičan efekat**, zato što JavaScript izvršno okruženje ignoriše prazna mesta.

## BLOKOVI

Na početku ove lekcije rečeno je da su, pored naredbi, osnovni leksički elementi JavaScript jezika i blokovi. Blok se veoma često naziva i blok naredbi, odnosno izjava (engl. block statement), pa stoga nije teško zaključiti da je reč o jezičkom elementu koji se može koristiti za grupisanje većeg broja naredbi u jednu celinu. U uvodnom primeru postoji nekoliko blokova, koji se mogu prepoznati po otvorenim i zatvorenim vitičastim zagradama ( { } ).

```
const COLOR_1 = "#026873";
const COLOR_2 = "#F29484";

let containerElement = document.getElementById("main-container");
document.getElementById("button").addEventListener("click", toggleBgColor);

let counter = 0;

function toggleBgColor() {
  counter++;
  if (counter % 2 == 1) {
    containerElement.style.backgroundColor = COLOR_1;
  }
  else {
    containerElement.style.backgroundColor = COLOR_2;
  }
}
```

Na slici je obeležen jedan blok unutar primera sa početka ove lekcije. Unutar ovog bloka mogu se izdvojiti još dva podbloka:

```
function toggleBgColor() {  
  
    counter++;  
  
    if (counter % 2 == 1) {  
        containerElement.style.backgroundColor = COLOR_1;  
    }  
  
    else {  
        containerElement.style.backgroundColor = COLOR_2;  
    }  
}
```

Sa ove dve slike se mogu videti najznačajnije sintaksne osobine blokova. Svi blokovi započinju otvorenom, a završavaju se zatvorenom vitičastom zagradom.

## KLJUČNE REČI

Uvodni primer sa početka ove lekcije poseduje izvestan broj specijalnih reči koje su obeležene crvenom bojom – *let*, *const*, *function*... Reč je o leksičkim elementima jezika koji se nazivaju ključne reči.

Ključne reči su reči koje su rezervisane od jezika, pa tako za JavaScript izvršno okruženje imaju posebno značenje. Tako na primer, kada čitanje HTML dokumenta dođe do reči *const* prevodilac zna da je potrebno da rezerviše deo memorije i u nju upiše određenu vrednost.

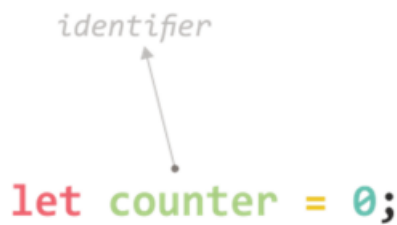
JavaScript jezik poseduje veliki broj ključnih reči. Neke od najkorišćenijih su:

- break
- case
- catch
- class
- const
- continue
- default
- do
- else
- extends
- switch
- finally
- this
- throw
- try
- typeof
- var
- void
- while
- with
- for
- function
- if
- import
- in
- instanceof
- let
- new
- return
- super

Sa različitim ključnim rečima upoznavaćemo se postepeno.

## IDENTIFIKATORI

U prethodnim redovima definisan je pojam ključnih reči JavaScript jezika, odnosno reči čiji su oblik i značenje unapred utvrđeni pravilima jezika. Prilikom pisanja JavaScript koda, programer ima mogućnost i da samostalno formira nazive određenih jezičkih elemenata. To se postiže korišćenjem identifikatora.



The diagram shows the code `let counter = 0;` with the word `counter` highlighted in green. An arrow labeled "identifier" points to the green text.

Slika ilustruje primer jednog identifikatora. Za reč *counter* se kaže da je identifikator zato što identifikuje vrednost 0.



The diagram shows the code `const COLOR_1 = "#026873";` with the text `COLOR_1` highlighted in green. An arrow labeled "identifier" points to the green text.

Slika ilustruje još jedan identifikator. U pitanju je *COLOR\_1*. Ovim identifikatorom se u nastavku skripte predstavlja vrednost jedne boje u heksadecimalnom obliku.

Identifikatori nisu ograničeni na obeležavanje nekih vrednosti, već se upotrebljavaju i za *identifikovanje* funkcija, objekata, klasa...



The diagram shows the code `function toggleBgColor() {` with the text `toggleBgColor` highlighted in green. An arrow labeled "identifier" points to the green text.

Slika prikazuje primer identifikatora prilikom imenovanja jedne funkcije. Tekst *toggleBgColor* je identifikator kojim se imenuje jedna funkcija.

## Pravila za definisanje identifikatora

Bitno je razumeti da identifikatori nisu nešto što je unapred definisano jezikom. Drugim rečima, programer sam smišlja njihove nazive. Zato je veoma bitno znati da je prilikom definisanja identifikatora potrebno poštovati nekoliko pravila za njihovo pisanje:

- identifikatori mogu biti sastavljeni iz slova, brojeva i karaktera donja crta i dolar,
- identifikatori ne smeju početi brojem,
- identifikatori su osetljivi na velika i mala slova, tako da `color` i `Color` nije isto,
- ključne reči se ne mogu koristiti kao identifikatori.

Pravila za definisanje identifikatora su veoma značajna, s obzirom na to da se identifikatori koriste za imenovanje brojnih jezičkih elemenata, o kojima će biti reči u lekcijama koje slede – promenljive, konstante, funkcije...

## LITERALI

Pored identifikatora, programer ima mogućnost da proizvoljno, odnosno samostalno definiše još jedan jezički element – reč je o literalima, odnosno vrednostima. Literali predstavljaju način za predstavljanje vrednosti unutar skripte. U uvodnom primeru sa slike postoji nekoliko literala, koji su obeleženi plavom bojom.



The diagram shows a code snippet: `const COLOR_1 = "#026873";`. The word `const` is red, `COLOR_1` is green, and the string `"#026873"` is blue. An arrow labeled "Literal" points to the blue string.

Slika ilustruje primer jednog literala. Literalom se u prikazanom primeru predstavlja jedna tekstualna vrednost, odnosno HEX kod jedne boje. Iz ovog primera se može zaključiti da se tekstualne vrednosti, odnosno tekstualni literali navode između znakova navodnika.



The diagram shows a code snippet: `let counter = 0;`. The word `let` is red, `counter` is green, and the number `0` is blue. An arrow labeled "Literal" points to the blue number.

Još jedan primer literala je prikazan slikom iznad. Ovoga puta je reč o literalu koji predstavlja brojčanu (numeričku) vrednost – 0. S obzirom na to da je reč o numeričkom literalu, njegova vrednost se ne piše između navodnika.

Pojam literala je tesno povezan sa pojmom promenljivih i tipovima podataka. Stoga će u lekcijama koje slede sa posebnom pažnjom biti nastavljena priča koja je započeta u prethodnim redovima.

## OPERATORI

Još jedan jezički, leksički element JavaScripta, se odnosi na pojam operatora. Baš kao i u matematici, operatori se koriste za obavljanje određenih operacija nad vrednostima. Tako su operatori upravo specijalni karakteri koji predstavljaju neku operaciju.



Slika ilustruje primer jednog operatora. Reč je o dva karaktera ++ koji su obojeni žutom bojom. Ovakav operator se u JavaScript jeziku naziva operator uvećanja, zato što uvećava vrednost za jedan.

Cilj prethodnih redova je bio samo da vas načelno upozna sa pojmom operatora. I to je tema koja će detaljno biti obrađena u lekcijama koje slede.

## KOMENTARI

Prilikom pisanja JavaScript koda moguće je napisati i tekst koji nema nikakvo značenje za izvršno okruženje. Na taj način je moguće napisati neku dodatnu informaciju o kodu koja će poslužiti kao uputstvo ili naznaka za lakše snalaženje programera. Takve dodatne informacije nazivaju se **komentari**.

Komentari u JavaScript jeziku mogu se formirati na dva načina, u zavisnosti od toga koliko linija teksta obuhvataju.

**// jednolinijski komentar**

**/\* ovo je duži,**

**Višelinijski komentar \*/**

Jednolinijski komentar je onaj koji obuhvata jednu liniju teksta. Formira se korišćenjem dva karaktera kosa crta. Ukoliko je pod komentar potrebno postaviti više linija teksta, koristi se nešto drugačiji pristup za formatiranje komentara - /\* za njegov početak i \*/ za njegov završetak.



## FUNKCIJE I OBJEKTI

U dosadašnjem toku ove lekcije ilustrovani su osnovni gradivni blokovi JavaScript koda. Kada se kaže osnovni, misli se na leksičke elemente najnižeg nivoa.

Funkcije i objekti su pojmovi koji prevazilaze okvire ove lekcije. Njima će biti posvećene kompletne zasebne lekcije.

Ipak, dobro je znati da uvodni primer koristi i objekte i funkcije. Na primer, *document* je jedan od objekata koji automatski postoje unutar JavaScripta. Dalje, *getElementById()* i *addEventListener()* su funkcije koje takođe automatski postoje unutar jezika. Primer sadrži i jednu funkciju koju smo samostalno kreirali – *toggleBgColor*.

U narednim lekcijama funkcije će na nekim mestima možda biti oslovljavane i kao metode. Reč je o dva pojma – funkcija i metoda, koje u ovom trenutku možete smatrati sinonimima, dok će razlike između njih biti razjašnjene u lekciji o objektima.