

Osnovna leksička struktura

Nakon upoznavanja različitih načina za implementiranje JavaScript koda u HTML dokument, kreiranje programske logike može da počne, pod uslovom da se poznaju pravila pisanja jezika, odnosno njegova leksička struktura. Stoga će u narednim redovima biti izneta osnovna pravila pisanja JavaScript koda.

Leksička struktura JavaScript jezika

Leksička struktura nekog programskog jezika predstavlja osnovni skup pravila koji diktira način na koji se programski kôd piše. Leksička struktura se odnosi na pravila pisanja, odnosno na sintaksu, ali i na 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 kôd sačinjen je od izjava (naredbi) koje mogu biti grupisane u blokove. Ove naredbe govore izvršnom okruženju koji efekat je potrebno da programski kôd proizvede. Jedna naredba može se sastojati iz vrednosti (literals), 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 ovog kursa:

```
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;
    }
}
```

Prikazani primer može poslužiti za identifikaciju nekih osnovnih leksičkih elemenata JavaScript jezika (slika 3.1).

```

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

Slika 3.1. Različiti leksički elementi JavaScript jezika, prikazani na uvodnom primeru

Dva osnovna leksička elementa JavaScript jezika su:

- naredbe (izjave),
- blokovi.

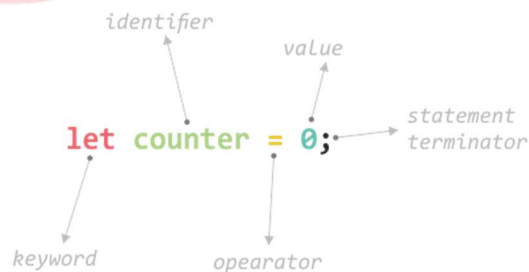
U kreiranju naredbi i blokova dalje mogu učestvovati i razni drugi leksički elementi:

- karakteri za završavanje linija,
- prazna mesta,
- ključne reči,
- identifikatori,
- literali (vrednosti),
- operatori,
- komentari.

O svim ovim pojmovima biće reči u nastavku ove 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 prikazana je slikom 3.2.



Slika 3.2. Primer jedne JavaScript naredbe

Slika 3.2. 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 blok JavaScript jezika, naredbe se koriste za obavljanje velikog broja različitih poslova prilikom JavaScript programiranja. U primeru sa slike 3.2. 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 zapamti tokom izvršavanja JavaScript koda (vrednost 0 dodeljena je 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 (na slici 3.2. predstavljen crnom bojom) **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 3.1. može se primetiti da je u njegovom formiranju iskorišćeno specifično formatiranje koda. Kôd 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 kôd. 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 jedinica 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 kôd kompletnog skripta u jednoj liniji. Ipak, takvo 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 (slika 3.3).

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

Slika 3.3. Dve naredbe u dve linije

Slika 3.3. ilustruje dve naredbe koje su napisane u dve linije JavaScript koda. Ovakve dve naredbe mogu se napisati i kao na slici 3.4.

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

Slika 3.4. Dve naredbe u jednoj liniji

Moguće je otići i korak dalje i napisati nešto kao na slici 3.5.

```
let counter = 0;

const COLOR_1 = "#026873";
```

Slika 3.5. Dve naredbe i različite vrste praznih mesta

Slika 3.5. 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 ovoga kursa (slika 3.2) 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;
    }
}
```

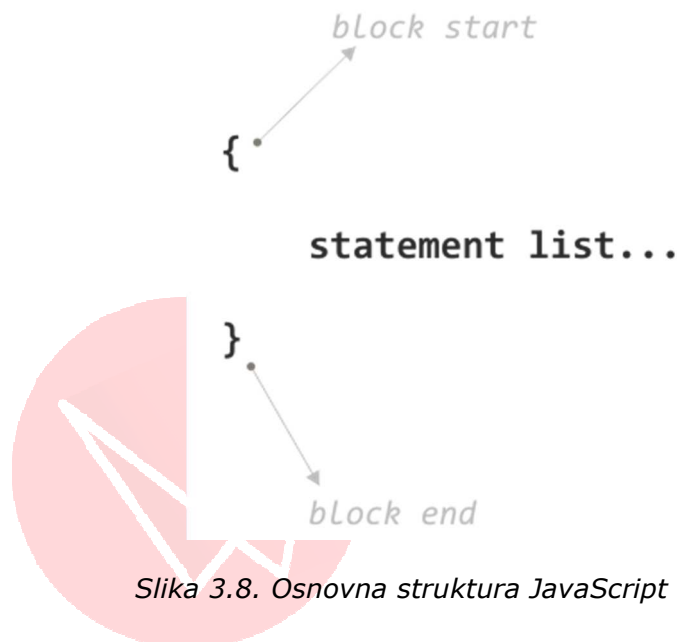
Slika 3.6. Primer jednog bloka

Na slici 3.6. obeležen je jedan blok unutar primera sa početka ove lekcije. Unutar ovog bloka mogu se izdvojiti još dva podbloka (3.7).

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

Slika 3.7. Dva bloka unutar drugog bloka

Sa slika 3.6. i 3.7. se mogu videti najznačajnije sintaksne osobine blokova. Svi blokovi započinju otvorenom, a završavaju se zatvorenom vitičastom zagradom. Tako je osnovni oblik svakog JavaScript bloka kao na slici 3.8.



Slika 3.8. Osnovna struktura JavaScript bloka

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

- kontrola toka (petlje, grananja, obrada izuzetaka),
- funkcionalno programiranje.

Kontroli toka, te kreiranju i korišćenju funkcija biće posvećeno nekoliko lekcija u nastavku ovoga kursa.

Ključne reči

Uvodni primer sa početka ove lekcije (slika 3.1.) poseduje izvestan broj specijalnih reči koje su obeležene crvenom bojom – `let`, `const`, `function`... Reč je 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.

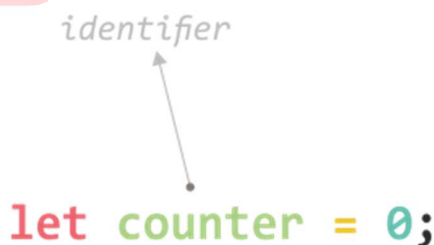
JavaScript jezik poseduje veliki broj ključnih reči. Neke od najkorišćenijih su:

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

Sa različitim ključnim rečima upoznavaćemo se postepeno, tokom trajanja ovog kursa.

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.

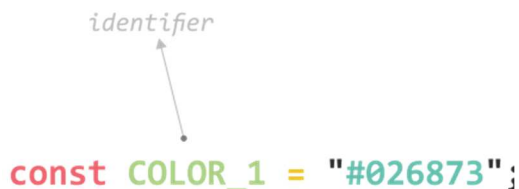


identifier

```
let counter = 0;
```

Slika 3.9. Primer JavaScript identifikatora

Slika 3.9. ilustruje primer jednog identifikatora. Za reč `counter` se kaže da je identifikator zato što identifikuje vrednost 0.



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

Slika 3.10. Primer JavaScript identifikatora (2)

Slika 3.10. prikazuje 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...



```
function toggleBgColor() {  
    ...  
}
```

Slika 3.11. Primer JavaScript identifikatora (3)

Slika 3.11. ilustruje 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 3.1 postoji nekoliko literala, koji su obeleženi plavom bojom.



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

Slika 3.12. Primer JavaScript literala

Slika 3.12. ilustruje primer jednog literala. Literalom se u prikazanom primeru predstavlja jedna tekstualna vrednost, odnosno HEX kôd jedne boje. Iz ovoga primera se može zaključiti da se tekstualne vrednosti, odnosno tekstualni literali navode između znakova navoda, navodnika.



```
let counter = 0;
```

Slika 3.13. Primer JavaScript literala

Još jedan primer literala prikazan je slikom 3.13. 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, koji je ilustrovan slikom 3.1, odnosi se na pojam operatora. Baš kao i u matematici, operatori se koriste za obavljanje određenih operacija nad vrednostima. Tako su operatori zapravo specijalni karakteri koji predstavljaju neku operaciju.


```
counter++;
```

Slika 3.14. Primer jednog JavaScript operatora

Slika 3.14. 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 bio je samo da vas načelno upoznaju 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 se nazivaju **komentari**.

Komentari se u JavaScript jeziku mogu formirati na dva načina, u zavisnosti od toga koliko linija teksta obuhvataju (slika 3.15).

```
//a one-line comment
```

```
/*this is a longer,  
multi-line comment*/
```

Slika 3.15. Primer jednolinijskih i višelinijjskih komentara

Slika 3.15. ilustruje primer jednolinijskog i višelinijjskog komentara. Potpuno razumljivo, 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 formiranje komentara – /* za njegov početak i */ za njegov završetak.

Kombinacija komentara i jedne izjave može da izgleda kao na slici 3.16.

```
//this is one statement  
let counter = 0;
```

Slika 3.16. Jednolinijski komentar i izjava

U ovakvoj situaciji, tekst `//this is one statement` nema nikakav efekat po izvršavanje koda. To je informacija koja je namenjena isključivo ljudima, a ne mašinama. Ipak, i JavaScript kôd se može pretvoriti u komentar (slika 3.17).

```
//this is one statement
//let counter = 0;
```

Slika 3.17. Dve pojedinačne linije komentara

Sada se primer sastoji iz dve linije komentara, odnosno kôd iz primera je izgubio svoju funkcionalnost jer je pretvoren u komentar.

Pitanje

Odaberite tačnu tvrdnju:

- **identifikatori mogu da budu sastavljeni iz slova, brojeva i karaktera donja crta i dolar**
- identifikatori smeju početi brojem
- identifikatori nisu osetljivi na velika i mala slova
- ključne reči se mogu koristiti kao identifikatori

Objašnjenje:

Identifikatori mogu da budu sastavljeni iz slova, brojeva i karaktera donja crta i dolar, ali ne mogu početi brojem i ne mogu biti isti kao neki od ključnih reči. Takođe, identifikatori su osetljivi na velika i mala slova.

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. Leksički elementi najnižeg nivoa u najvećem broju slučajeva samostalno imaju veoma malu upotrebnu vrednost. Zbog toga se pribegava njihovom kombinovanju, kako bi se izgradile funkcionalne naredbe koje mogu da proizvedu neki konkretan efekat. Ipak, kombinovanjem osnovnih leksičkih elemenata mogu se dobiti i nešto viši elementi jezika, kao što su funkcije i objekti.

U primeru koji je prikazan na početku ove lekcije korišćena su oba upravo navedena pojma. Funkcije i objekti su pojmovi koji prevazilaze okvire ove lekcije. Njima će biti posvećene kompletne zasebne lekcije u nastavku ovoga kursa. Jednostavno, za njihovo adekvatno razumevanje neophodno je prethodno se upoznati sa svim osnovnim leksičkim elementima nabrojanim na početku ove 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.

Rezime

- Leksička struktura nekog programskog jezika predstavlja osnovni skup pravila koji diktira način na koji se programski kôd piše.
- JavaScript većinu sintakse pozajmljuje od jezika Java (odnosno C), ali poseduje i elemente jezika Awl, Perl i Python.
- JavaScript kôd sačinjen je od izjava (naredbi) koje mogu biti grupisane u blokove.
- Jedna naredba može se sastojati iz vrednosti (literals), operatora, ključnih reči, identifikatora, komentara i raznih drugih elemenata, čijim grupisanjem se dobija entitet razumljiv JavaScript izvršnom okruženju.
- Najmanja jezička jedinica koja može da proizvede neku akciju u JavaScript jeziku, naziva se naredba ili izjava.
- Naredbe se drugačije nazivaju izjave i razdvajaju se karakterom tačka sa zapetom (;).
- Postavljanje karaktera tačka sa zapetom na kraj naredbi, u JavaScriptu nije obavezno, ali se preporučuje.
- Linija koda nije isto što i naredba koda.
- Za JavaScript izvršno okruženje pojam linije, baš kao i pojam praznih mesta, nema nikakvo značenje.
- Blokovi su jezički elementi koji se mogu koristiti za grupisanje većeg broja naredbi u jednu celinu.
- Ključne reči su reči koje su rezervisane od jezika, a za izvršno okruženje imaju posebno značenje.
- Identifikator je nešto što nije unapred definisano jezikom, a služi za označavanje delova koda.
- Literali predstavljaju način za predstavljanje vrednosti unutar skripte.
- Operatori se koriste za obavljanje određenih operacija nad vrednostima.
- JavaScript komentari su tekst koji se ignoriše od izvršnog okruženja, a koristi se za interno obeležavanje koda.