Univerzitet u Beogradu
Matematički fakultet

MASTER RAD

Elektronske lekcije o funkcijama i objektima u programskom jeziku JavaScript

Mentor:
Prof. dr Miroslav Marić

Kandidat:
Marija Đorđević

Februar 2017, Beograd

Sadržaj

1 Uvod

JavaScript je programski skript jezik koji je razvila kompanija Netscape Communications. Danas je to jedan od najpopularnijih skript jezika na internetu i funkcioniše na svim poznatijim pregledačima, kao što su Internet Explorer, Mozilla Firefox, Google Chrome, Opera i sl.

Programski jezik *JavaScript* je, u martu 1996. godine, razvio Brendan Eich, koji je tada bio zaposlen u firmi Netscape (danas Mozilla). Prvobitno ime ovog programskog jezika je bilo *Mocha*, koje je izabrao Marc Andereessen, osnivač kompanije Netscape. Nakon samo pet meseci ime je promenjeno u *LiveScript*. U decembru iste godine programski jezik dobija ime *JavaScript*, koje zadržava sve do danas.

Programski jezik *Java* je u vreme nastanka programskog jezika *JavaScript* bio široko rasprostranjen, pa su osnivači u Netscape-u došli na ideju da i svom izumu daju slično ime. Ovo se pokazalo kao izuzetan marketinški trik, jer se za jezik *JavaScript* brzo pročulo. Dakle, jezici *JavaScript* i *Java* nisu isti, postoje neke sličnosti u sintaksi, ali ne više od toga. U osnovi, to su dva različita programska jezika.

U tabeli 1 predstavljene su verzije jezika *JavaScript*, vreme nastanka, kao i verzije internet pregledača koje ih podržavaju.

Tabela 1: Razvoj i verzije jezika *JavaScript*

Verzija	Vreme nastanka	Jednak sa jezikom	Netscape	Mozilla Firefox	Internet Explorer	Google Chrome
1.0	mart, 1996		2.0		3.0	
1.1	avgust, 1996		3.0			
1.2	jun, 1997		4.0-4.05			
1.3	oktobar, 1998	ECMAScript 1 -2	4..06-4.7x		4.0	
1.5	novembar, 2000	ECMAScript 3	6.0	1.0	5.5-8.0	1.0-10.0.666
1.6	novembar, 2005	ECMAScript for XML		1.5		
1.7	oktobar, 2006			2.0		
1.8	jun, 2008			3.0		
1.8.1	jun, 2009			3.5		
1.8.2	januar, 2010			3.6		
1.8.5	mart, 2011	ECMAScript 5		4	9-10	13.0+

U tabeli 1 nije uključena verzija 1.4, jer je razvijena samo za upotrebu na Netscape serveru.

JavaScript se prvenstveno koristi za dodavanje funkcionalnosti i dinamičkog ponašanja internet stranicama. Najčešće se korsite funkcije koje su uključene ili ugrađene u HTML stranice. Njegova popularnost iz dana u dan raste i proširuje se njegova upotreba. Na početku se koristio isključivo na internetu, dok danas se koristi za razvoj mobilnih i računarskih aplikacija, ali čak i u razvoju igrica. U nastavku rada primarno je prikazan deo *JavaScript* jezika koji se odnosi na rad sa objektnim modelom dokumenata koji se pridružuje HTML stranama.

JavaScript kôd se na HTML stranicu dodaje unutar **script** etiketa, kao u primeru 1.1.

Primer 1.1. *JavaScript* kôd u HTML dokumentu

```
<script>
JavaScript kod se nalazi ovde.
</script>
```

Osim toga *JavaScript* kôd se može preuzeti i iz spoljašnje datoteke čija se putanja zadaje **src** atributom, što se može videti u primeru 1.2.

Primer 1.2. *JavaScript* kôd uvežen iz spoljašnje datoteke „moj_skript.js”

```
<script src="moj_skript.js"> </script>
```

2 Elektronske lekcije o funkcijama i objektima u programskom jeziku *JavaScript*

Postoji mnoštvo knjiga u kojima su detaljno objašnjene funkcije i objekti u programskom jeziku *JavaScript*, ali na našem jeziku ne postoji dovoljno elektronske literature koja bi pratila te sadržaje, zbog čega je korisnicima ograničena mogućnost izbora. Elektronske lekcije treba da predstavljaju interaktivne i interesantne sadržaje koji su saglasni sa teorijom i koji su prikazani korišćenjem modernih veb tehnologija. Elektronske lekcije o funkcijama i objektima u programskom jeziku *JavaScript* kreirane su kako bi korisnik na jednom mestu imao i neophodnu teoriju i primere i mogućnost da testrira naučeno.

2.1 Internet verzija elektronskih lekcija

Internet verzija elektronskih lekcija o funkcijama i objektima u programskom jeziku JavaScript kreiranih za potrebe ovog master rada nalazi se na sledećoj adresi:

http://alas.matf.bg.ac.rs/~mr11032/eskola_veba/#/course-details/js

Na slici 1 prikazana je početna stranica elektronskih lekcija o programskom jeziku *JavaScript* koje su deo projetka *eŠkola veba*. Projekat *eŠkola veba* se razvija na Matematičkom fakultetu u Beogradu [9].

Korisnik sve lekcije može pregledati u padajućem meniju sa leve strane.

Slika 1: Elektronske lekcije o programskom jeziku *JavaScript*

Svaka internet stranica u okviru elektronskih lekcija sastavljena je iz tri dela, što se može videti na slici 2, strana 5. Na levoj strani nalaze se elektronske lekcije, dok se na desnoj strani nalaze dva polja. U gornje polje moguće je uneti

kôd (ili izmeniti), a na donjem polju se prikazuje rezultat rada tog koda. Neke internet stranice sadrže unete primere u gornjem polju.

Ukoliko korisnik želi da pređe na prethodnu ili sledeću lekciju, na dnu internet stranice se nalaze dva dugmeta koja mu to omogućavaju.

Svaka internet stranica na dnu ima i dugme *Pomoć* koje prikazuje uputstva za rad.

Slika 2: Izgled lekcije *Uvod u funkcije*

Za potrebe ovog master rada kreirane su sledeće elektronske lekcije:

- Funkcije - Upoznavanje korisnika sa funkcijama, pravilima prilikom definisanja, načinom pozivanja, kao i rad sa njima;
- Objekti - Upoznavanje korisnika sa objektima, pravila prilikom definisanja i rad sa njima;
- Primeri - Približavanje pojma funkcija i objekta, kao i korisnosti njihove upotrebe;
- Zadaci sa rešenjima - Testiranje stečenog znanja kroz zadatke. Ukoliko želi, korisnik može otvoriti rešnje koje se nalazi u delu *Pomoć*.

2.1.1 Ukratko o *eŠkoli veba*

eŠkola veba je elektronska platforma za interaktivno učenje veb programiranja kroz pažljivo osmišljenje kurseve. Na ovoj platformi lekcije su podeljene u sledeće grupe:

- *HTML*;
- *CSS*;
- *JavaScript*;

- *jQuery*;
- *Bootstrap*;
- *Angular*;
- *PHP*;
- *TypeScript*;
- *Angular 2*;
- *JSON*;
- *Veb*.

Ove lekcije za cilj imaju da učenicima približe savremene tehnologije koje se koriste u svetu veb programiranja. Značaj elektronske platforme za učenje veb programiranja je veliki, posebno kada se uzme u obzir nagli razvoj interneta i popularnost alatki i servisa koje nudi.

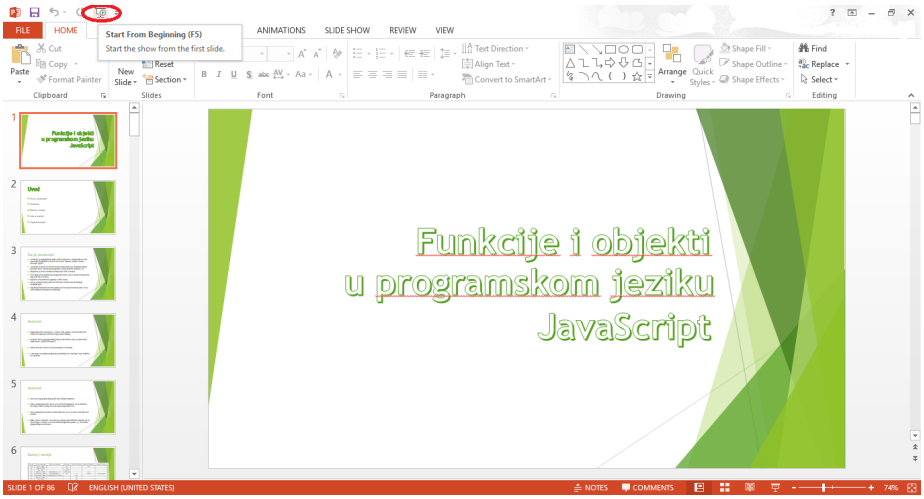
U planu je i proširivanje eŠkole veba novim temama i lekcijama, kao i stalno unapređivanje postojećih lekcija. Krajnjem korsiniku se time nudi jedna celina u kojoj može da uči i uvežbava stečeno znanje, a i može biti siguran da gradivo koje izučava nije zastarelo, niti prevaziđeno.

2.2 Elektronske lekcije u vidu prezentacije

Verzija elektronskih lekcija o funkcijama i objektima u programskom jeziku *JavaScript* u vidu prezentacije se može koristiti bez internet konekcije, a nalazi se na sledećoj adresi:

alas.matf.bg.ac.rs/∼ml10002/MRad/ELOFIOJavaScript.pptx

Dokument *ELOFIOJavaScript.pptx* predstavlja elektronske lekcije u vidu prezentacije, a njegov izgled je prikazan na slici 3. Označeno dugme na slici 3 u gornjem levom uglu pokreće prezentaciju.



Slika 3: Elektronske lekcije u vidu prezentacije

Lekcije su, slično kao i kod internet verzije, podeljene u četiri velike grupe:

- Uvod;
- Funkcije;
- Objekti;
- Zadaci sa rešenjima.

Elektronske lekcije o funkcijama i objektima u vidu prezentacije sadrže uvodnu priču o *JavaScript* programskom jeziku, koja nije uključena u internet verziju elektronskih lekcija o funkcijama i objektima. Na drugom slajdu može se videti podela po oblastima, a ukoliko korisnik želi da pređe na određenu oblast, to može učiniti klikom na njen naziv. Isto tako, ukoliko korsnik u bilo kom trenutku poželi da se vrati na slajd sa sadržajem, u donjem desnom uglu svakog slajda nalazi se dugme koje mu to omogućava.

U okviru lekcija postoji veliki broj primera koji su kreirani kako bi korisnik lakše i brže prihvatio nova saznanja, kao i da bi stekao utisak o samoj implementaciji programa.

Naslov ili naziv svakog primera sadrži link koji vodi do internet stranice na kojoj se može testirati.

Da bi se testirali ovi primeri, neophodna je internet konekcija.

3 Funkcije u programskom jeziku *JavaScript*

Funkcija u programskom jeziku *JavaScript* je izdvojena programska celina ili potprogram. Odnosno, funkcija je blok koda koga čini niz određenih naredbi. Tek kada se eksplicitno pozove, funkcija će uticati na rad programa. Funkcije omogućavaju da kôd bude modularan, odnosno da se lakše održava i menja. Najćešće se koriste kod zadataka koji se često ponavljaju.

U programskom jeziku *JavaScript* funkcija se može definisati i u zaglavlju i u telu HTML dokumenta. Često se navodi na samom kraju HTML dokumenta (u telu dokumenta), jer *JavaScript* kôd najčešće radi nad DOM drvetom, a da bi DOM drvo bilo kreirano ceo HTML sadržaj treba da se učita. Ako se *JavaScript* kôd postavi na kraju dokumenta, DOM će sigurno biti kreiran.

Postoje dve vrste funkcija: ugrađene (primitivne) funkcije i korisničke funkcije (one koje korisnici definišu). U primeru 3.1. definisana je funkcija *prvaFunkcija*, koja poziva ugrađenu funkciju *alert*.

Primer 3.1. Funkcija u programskom jeziku JavaScript

```
function prvaFunkcija() {
        alert(”Obaveštenje”);
}
```

3.1 Ugrađene funkcije u programskom jeziku *JavaScript*

U programskom jeziku *JavaScript* postoji mnoštvo ugrađenih funkcija. U nastavku su navedene neke od njih, one koje imaju široku primenu i čestu upotrebu.

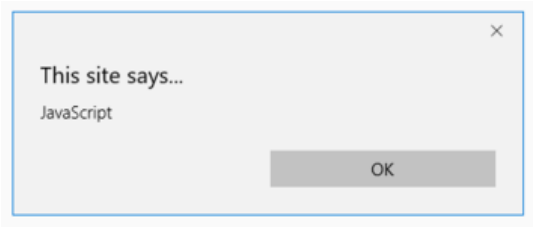
3.1.1 Funkcija alert

Nakon pozivanja funkcije *alert* u internet pregledaču će se prikazati iskačući prozor sa tekstom koji se prilikom poziva funkcije navodi u zagradama kao argument. Način upotrebe funkcije *alert* prikazan je u primeru 3.2.

Primer 3.2. Funkcija *alert*

```
<html>
    <head>
        <title> Funkcija alert </title>
    </head>
    <body>
        <script>
        alert("JavaScript");
        </script>
    </body>
</html>
```

Otvaranjem dokumenta iz primera 3.2. otvara se internet pregledač sa iskačućom porukom koja je prikazana na slici 4.



Slika 4: Rezultat poziva funkcije *alert("JavaScript");*

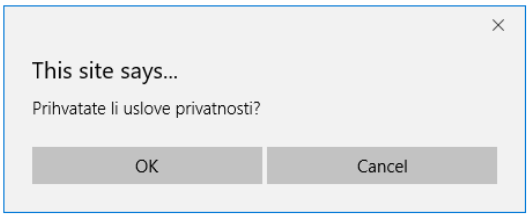
3.1.2 Funkcija confirm

Kada se pozove funkcija *confirm* u internet pregledaču će se prikazati iskačući prozor sa tekstom koji se prilikom poziva funkcije navodi u zagradama kao argument. U primeru 3.3. prikazan je HTML dokument u kome je pozvana funkcija *confirm*.

Primer 3.3. Funkcija *confirm*

```
<html>
    <head>
        <title> Funkcija confirm </title>
    </head>
    <body>
        <script>
        var izaberi=confirm("Prihvatate li uslove privatnosti?");
        </script>
    </body>
</html>
```

Otvaranjem dokumenta iz primera 3.3. otvara se internet pregledač sa iskačućom porukom, koja je prikazana na slici 5.



Slika 5: Rezultat rada funkcije *confirm("Prihvatate li uslove privatnosti?");*

Iskačući prozor sa slike 5 ponudio je dve opcije: „OK" ili „Cancel". Ukoliko korisnik odabere „OK" dugme, funkcija će proslediti povratnu vrednost **true**,

a ukoliko odabere dugme „Cancel”, funkcija prosleđuje vrednost **false**. Osim toga, moguće je promeniti nazive dugmića „OK” i „Cancel”, odnosno prilagoditi ih sopstvenim potrebama.

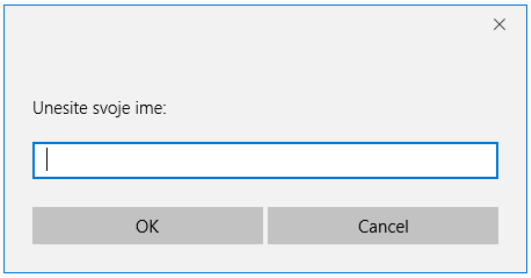
3.1.3 Funkcija prompt

Funkcija *prompt* se poziva tako što se u zagradama navodi tekst koji će se pojaviti u iskačućem prozoru. Jedan od načina pozivanja ove funkcije prikazan je u primeru 3.4.

Primer 3.4. Funkcija *prompt*

```
<html>
    <head>
        <title> Funkcija prompt </title>
    </head>
    <body>
        <script>
        var ime=prompt("Unesite svoje ime:");
        </script>
    </body>
</html>
```

Otvaranjem dokumenta iz primera 3.4. otvara se internet pregledač sa iskačućom porukom prikazanom na slici 6. Iskačuća poruka sadrži tekstualno polje u koje korisnik može upisati vrednost koju želi da funkcija vrati, a koja se prosleđuje odabirom dugmeta „OK”.



Slika 6: Rezultat rada funkcije *prompt(”Unesite svoje ime:”);*

Ukoliko korisnik odabere dugme „Cancel” povratna vrednost funkcije je **null**. Konkretno, u ovom primeru, povratna vrednost se upisuje u promenljivu *ime*, koju korisnik može dalje koristiti, po potrebi.

Napomena: Funkcije alert, confirm i prompt se odnose na internet pregledač, pa se mogu pozivati i sa *window.alert()*, *window.confirm()* ili *window.prompt()*.

3.1.4 Funkcija write

Funkcija *write* se poziva na sledeći način:

document.write();

Reč *document* naglašava da se funkcija primenjuje nad tekućom stranicom koja je učitana od strane pregledača. Između zagrada se navodi tekst koji se ispisuje na internet stranici.

Primer 3.5. Funkcija *write*

```
<html>
    <head>
        <title> Funkcija write </title>
    </head>
    <body>
        <script>
        document.write("Dobar dan!");
        </script>
    </body>
</html>
```

Nakon otvaranja dokumenta iz primera 3.5. u internet pregledaču se prikaže tekst sa slike 7.

Dobar dan!

Slika 7: Rezultat rada funkcije *write(„Dobar dan!”);*

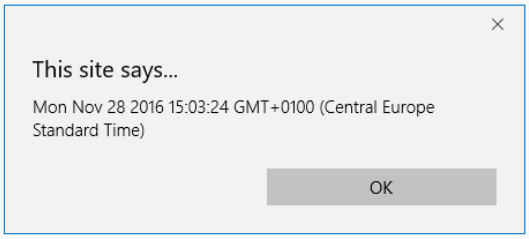
3.1.5 Funkcija Date

Funkcija *Date* kada se pozove generiše objekat koji sadrži datum, vreme i odrednicu vremenske zone odakle je pozvana.

Primer 3.6. Funkcija *Date*

```
<html>
    <head>
        <title> Datum </title>
    </head>
    <body>
        <script>
        var datum=Date();
        alert(datum);
        </script>
    </body>
</html>
```

Otvaranjem dokumenta iz primera 3.6. aktivira se iskačući prozor u čijem je sadržaju trenutni datum, vreme, kao i vremenska zona odakle se poziva ova funkcija. Testiranje primera 3.6. prikazano je na slici 8.



Slika 8: Rezultat rada funkcije *Date();*

3.1.6 Često korišćene metode

U tabeli 2 se nalaze neke od metoda koje se koriste nad numeričkim objektima, kao i rezultat rada ovih metoda, dok se u tabeli 3 nalaze metode koje se koriste nad string objektima.

Tabela 2: Metode koje se koriste nad numeričkim objektima

Metod	Rezultat rada
toString()	String reprezentacija vrednosti broja
toExponential()	Eksponencijalna reprezentacija broja
valueOf()	Vrednost broja
toPrecision()	Definiše koliko ukupno cifara prikazati (u zbiru ispred i iza decimalne zapete)
toFixed()	Prikazaće se određeni broj cifara iza decimalne zapete

Tabela 3: Metode koje se koriste nad string objektima

Metod	Rezultat rada
charAt()	Karakter koji je na određenoj poziciji
indexOf()	Indeks određenog karaktera
lastIndexOf()	Indeks poslednjeg pojavljivanja određenog karaktera
length	Dužina stringa
replace()	Pronalazi karakter i zamenjuje ga određenim karakterom
split()	Podeli string na niz podstringova
substr()	Karakteri u stringu koji počinju na određenoj poziciji, sve do zadanog broja karaktera
toLowerCase()	Svi karakteri stringa se prebacuju u mala slova
toUpperCase()	Svi karakteri stringa se prebacuju u velika slova
valueOf()	Vrednost stringa
parseInt()	Od stringa pravi celobrojnu numeričku promenljivu
parseFloat()	Od stringa pravi realnu numeričku promenljivu

Tabela 4: Metode koje se koriste nad objektima tipa *Date*

Metod	Rezultat rada
Date()	Danasnji datum, vreme i vremenska zona
getDate()	Dan u mesecu za određeni datum
getDay()	Dan u nedelji za određeni datum
getFullYear	Godinu za određeni datum
getHours()	Sat za određeni datum
getMiliseconds()	Milisekunde za određeni datum
getMinutes()	Minute za određeni datum
getMonth()	Mesec za određeni datum
getSeconds()	Sekunde za određeni datum
getTime()	Numerička vrednost u milisekundama počev od 01.01.1970. do trenutka poziva funkcije
setDate()	Postavlja dan u mesecu za određeni datum
setDay()	Postavlja dan u nedelji za određeni datum
setFullYear()	Postavlja godinu za određeni datum
setHours()	Postavlja sat za određeni datum
setMiliseconds()	Postavlja milisekunde za određeni datum
setMinutes()	Postavlja minute za određeni datum
setMonth()	Postavlja mesec za određeni datum
setSeconds()	Postavlja sekunde za određeni datum
setTime()	Postavlja numeričku vrednost u milisekundama pocev od 01.01.1970.
toDateString()	Formatira datum u oblik: „Dan u nedelji”, „Mesec”, „Dan u mesecu”, „Godina”
toString()	String reprezentacija datuma
valueOf()	Primitivna vrednost datuma
Date.parse()	Za uneti datum računa broj milisekundi od 01.01.1970.

U tabeli 4 prikazane su metode koje se koriste nad objektima tipa *Date*, a u tabeli 5 su prikazane neke od matematičkih metoda. Matematičke metode se obavezno pozivaju nad *Math* objektom u formatu *Math.nazivMetode()*.

Tabela 5: Matematičke metode

Metod	Rezultat rada
abs()	Apsolutna vrednost unetog broja
acos()	Arkus kosinus unetog broja (u radijanima)
asin()	Arkus sinus unetog broja (u radijanima)
atan()	Arkus tangens unetog broja (u radijanima)
floor()	Najmanji celi broj koji je veći ili jednak zadatom broju
cos()	Kosinus unetog broja
exp()	Stepenuje Ojlerov broj **E** na zadati stepen
ceil()	Najveći celi broj koji je manji ili jednak zadatom broju
log()	Prirodni logaritam zadatog broja
max()	Najveći od zadatih brojeva
min()	Najmanji od zadatih brojeva
pow()	Stepenuje broj
random()	Slučajno izabrani broj između 0 i 1
round()	Zaokruživanje na najbliži ceo broj
sin()	Sinus unetog broja
sqrt()	Kvadratni koren broja
tan()	Tangens broja
toSource()	String „Math”

3.1.7 Primeri upotrebe nekih metoda

Primer 3.7. Metode nad objektima tipa *Date*

```
<html>
    <head>
        <title> Datum </title>
    <head>
    <body>
        <script>
        var datum = new Date();
        document.write("Ovo je trenutno vreme: <br>")
        document.write(datum+"<br>");
        var datum1=new Date("July 21, 1983 01:15:00");
        document.write("Ovaj datum smo sami podesili: <br>")
        document.write(datum1+"<br>");
        datum.setDate(15);
        document.write("Nakon metode setDate(), danasnji datum je: <br>")
        document.write(datum+"<br>");
        var d=datum1.getDay();
        document.write("Metoda getDay() vratila je: <br>")
        document.write(d+ ", a to je redni broj dana u nedelji.");
        </script>
    </body>
</html>
```

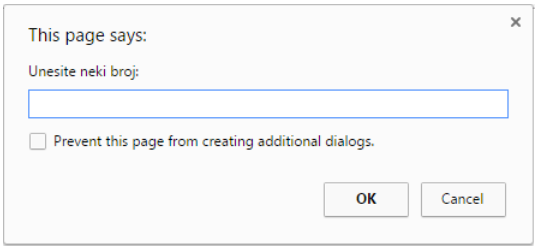
Rezultat rada HTML dokumenta iz primera 3.7. u internet pregledaču prikazan je na slici 9.

Slika 9: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.7.

Primer 3.8. Metode nad numeričkim objektima

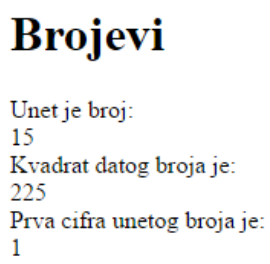
```
<html>
    <head>
        <title> Brojevi </title>
    </head>
    <body>
        <script>
        var broj= prompt("Unesite neki broj:");;
        document.write("Unet je broj: <br>"+broj+"<br>");
        kvBroj=Math.pow(broj,2);
        document.write("Kvadrat datog broja je: <br> "+kvBroj.toString()+" <br>");
        var stringBroj=broj.toString();
        var podString=stringBroj.substr(0,1);
        document.write("Prva cifra unetog broja je: <br>"+podString);
        </script>
    </body>
</html>
```

Otvaranjem dokumenta iz primera 3.8., u internet pregledaču prvo iskoči prozor koji je prikazan na slici 10.



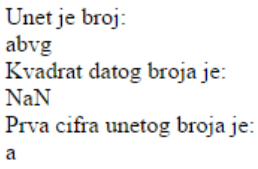
Slika 10: Prozor koji iskoči pri pokretanju HTML dokumenta iz primera 3.8.

Pri unosu broja u tekstualno polje na iskačućem prozoru sa slike 10, u internet pregledaču prikazaće se poruka prikazana slikom 11.



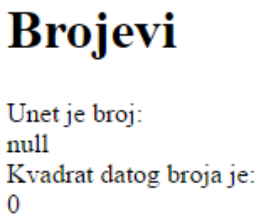
Slika 11: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.8. ukoliko je unet broj

Ako unos nije bio broj, prikazaće se poruka prikazana slikom 12.



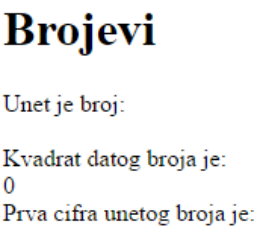
Slika 12: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.8. ukoliko unos nije bio broj

Ukoliko je korisnik izabrao „Cancel" dugme krajnji rezultat biće kao na slici 13.



Slika 13: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.8. ukoliko korisnik izabere dugme „Cancel"

Na slici 14 prikazan je ispis u slučaju da je korisnik zatvorio iskačući prozor.



Slika 14: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.8. ukoliko korisnik zatvori iskačući prozor

3.2 Korisničke funkcije u programskom jeziku *JavaScript*

Korisničke funkcije u programskom jeziku *JavaScript* su one koje korisnik sam definiše. Prilikom definisanja funkcije prvo se navodi ključna reč *function*, zatim razmak, naziv funkcije, obične zagrade i na kraju vitičaste zagrade.

Primer 3.9. Definicija funkcije

```
function naziv_funkcije(niz_argumenata){
    telo_funkcije(niz_naredbi)
}
```

U okviru običnih zagrada mogu, ali i ne moraju, da stoje ulazni argumenti (dodatne informacije koje koristi funkcija). U okviru vitičastih zagrada se nalazi telo funkcije, odnosno, naredbe koje funkcija izvršava.

U zavisnosti od toga da li funkcije u telu sadrze naredbu *return* razlikuju se funkcije sa povratnom vrednošću i funkcije bez povratne vrednosti.

3.2.1 Osobine korisničkih funkcija

Ključna reč *function* je obavezna.

Naziv funkcije je obavezan deo funkcije. Naziv može biti bilo koji niz karaktera, osim rezervisanih reči koje se nalaze u tabeli na slici 15.

abstract	arguments	boolean	break	byte
case	catch	char	class	const
continue	debugger	default	delete	do
double	else	enum	eval	export
extends	false	final	finally	float
for	function	goto	if	implements
import	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	volatile
while	with	yield		

Slika 15: Rezervisane reči

Naziv je često kratak opis onoga što funkcija izvršava. Na primer, funkciji koja računa kvadratni koren nekog broja može se dodeliti naziv „kvadratniKoren”, „kvKoren”, „koren”i sl.

Argumenti funkcije nisu obavezni. Funkciji se prilikom poziva prosleđuju ovi argumenti, koje ona koristi u radu. Ukoliko se funkciji prilikom poziva ne proslede svi ulazni argumenti, oni koji nisu prosleđeni dobijaju vrednost *undefined*. U ovom jeziku moguće je da se prilikom definisanja funkcije ne navedu argumenti, a da se prilikom poziva funkcije unesu.

Telo funkcije predstavlja niz naredbi. Podrazumeva se da je telo obavezan deo funkcije jer funkcija ništa neće izvršavati ukoliko se telo izostavi. U telu funkcije mogu se definisati lokalne promenljive, koristiti globalne promenljive, ali i pozivati postojeće funkcije, ukoliko se ukaže takva potreba. Promenljive koje se definišu u telu funkcije „žive" samo u toku rada funkcije. Kada funkcija završi sa radom, njima nije moguće pristupiti.

3.2.2 Primeri funkcija sa povratnom vrednošću

Primer 3.10. Funkcija sa povratnom vrednošću bez ulaznih argumenata

```
function najvecaOcena(){
    return 5;
}
```

Funkcija prikazana u primeru 3.10. nema ulaznih argumenata, ali ima povratnu vrednost. Svaki put kada se pozove ova funkcija povratna vrednost biće broj **5**.

Primer 3.11. Funkcija sa povratnom vrednošću i sa jednim ulaznim argumentom

```
function duplirajBroj(x){
    return 2*x;
}
```

Funkcija iz primera 3.11. očekuje da joj se prilikom poziva obezbedi jedan ulazni argument, broj x, a potom ona vraća dvostruku vrednost tog broja. Ukoliko se ne prosledi ulazni argument, ova funkcija vratiće vrednost **NaN** (not a number - nije broj).

Funkcija prikazana u primeru 3.12. ima tri ulazna argumenta, a potom računa zbir unetih brojeva.

Primer 3.12. Funkcija sa povratnom vrednošću i sa tri ulazna argumenta

```
function zbirbrojeva1(a,b,c){
    return a+b+c;
}
```

Ukoliko neki od argumenta u primeru 3.12. nije unet povratna vrednost funkcije će biti **NaN**.

Primer 3.13. Funkcija sa povratnom vrednošću, sa dva ulazna argumenta i novom promenljivom u telu funkcije

```
function zbirbrojeva2(a,b){
    var zbir=a+b;
    return zbir;
}
```

Funkcija prikazana u primeru 3.13. ima dva ulazna argumenta, a u telu funkcije je definisana nova promenljiva *zbir*, koja čuva zbir ulaznih argumenata. Potom, funkcija vraća vrednost koja je sačuvana u promenljivoj *zbir*.

Program kada naiđe na naredbu *return* odmah izlazi iz funkcije, tj. ukoliko u funkciji postoji neka naredba nakon naredbe *return*, ona nikada neće biti izvršena.

Primer 3.14. Funkcija sa povratnom vrednošću sa naredbom nakon naredbe *return*

```
function zbirbrojeva3(a,b){
    var zbir=a+b;
    return zbir;
    alert("Ovo se nikada nece izvrsiti");
}
```

3.2.3 Primeri funkcija bez povratne vrednosti

Primer 3.15. Funkcija bez povratne vrednosti i bez ulaznih argumenata

```
function pozdrav(){
    document.write("Zdravo");
}
```

Funkcija prikazana u primeru 3.15. nema ulaznih argumenata, a pri pozivu na internet stranici se ispisuje tekst **Zdravo**.

Primer 3.16. Funkcija bez povratne vrednosti sa jednim ulaznim argumetnom

```
function kvadratBroja(x){
    document.write(x*x);
}
```

Funkcija iz primera 3.16. ima jedan ulazni argument, broj x, a potom ispisuje kvadrat tog broja.

3.2.4 Pozivanje funkcije

U primerima je prikazano kako se pozivaju funkcije i šta je rezultat njihovog rada.

U primeru 3.17. prikazan je HTML dokument u kome se poziva funkcija *kvadratBroja*.

Primer 3.17. Pozivanje funkcije

```
<html>
    <head>
        <title> Kvadrat broja </title>
        <script>
        function kvadratBroja(x){
            return x*x;
        }
        </script>
    </head>
    <body>
        <h1> Kvadrat broja </h1>
        <script>
        document.write(kvadratBroja(5));
        </script>
    </body>
</html>
```

Nakon otvaranja dokumenta iz primera 3.17. u internet pregledaču se ispisuje poruka sa slike 16.

Slika 16: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.17.

Dakle, funkcija *kvadratBroja* iz primera 3.17. ima jedan ulazni argument i računa kvadrat tog broja. Pri pozivu ove funkcije prosleđen je broj **5** kao ulazni argument, a u internet pregledaču se upisuje kvadrat tog broja, odnosno broj **25**.

U primeru 3.18. prikazano je kako funkcija može u telu pozvati drugu funkciju.

Primer 3.18. Pozivanje funkcije iz tela druge funkcije

```
<html>
    <head>
        <title> Površina kvadrata </title>
        <script>
        function kvadratBroja(x){
            return x*x;
        }
        function povrsinaKvadrata(x){
            var povrsina=kvadratBroja(x);
            return povrsina;
        }
        </script>
    </head>
    <body>
        <h1> Površina kvadrata </h1>
        <script>
        document.write(kvadratBroja(5));
        </script>
    </body>
</html>
```

Otvaranjem HTML dokumenta iz primera 3.18. u internet pregledaču ispisuje se poruka prikazana na slici 17.

Slika 17: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.18.

U primeru 3.18. funkcija *povrsinaKvadrata* poziva funkciju *kvadratBroja* koja računa kvadrat prosleđenog argumenta. Prosleđeni argument je broj **5**, a nakon izvršenja funkcije prikazuje se broj **25**.

3.2.5 *Callback* funkcije

Callback je mehanizam, poznat i u drugim jezicima, koji omogućava da se funkcija prosledi kao argument drugim funkcijama.

Ukoliko se programira složenija aplikacija na vebu, često se dolazi u poziciju da je neophodno sačekati rezultat nekog izračunavanja ili podatke sa servera u toku izvršavanja samog programa. Rad programa se tada zaustavlja i nastavlja se tek kada svi podaci pristignu. U momentu kada server odgovori, poziva se *callback* funkcija koja služi kao početna tačka za nastavak rada.

Primer *callback* funkcije je zadavanje tajmera koji poziva *callback* funkciju posle određenog zadatog vremena.

Primer 3.19. *Callback* funkcija

```
function callbackFunkcija(){
  document.body.style.backgroundColor = "#003";
}
window.setTimeout(callbackFunkcija, 4000);
```

U primeru 3.19. definisana je *callback* funkcija *callbackFunkcija()* i postavljen je tajmer koji je poziva nakon četiri sekunde. Konkretno, funkcija *callbackFunkcija()* menja boju pozadine u tamno plavu.

Specifično je i navođenje *callback* funkcije u metodi *setTimeout()*. Navodi se samo naziv funkcije bez zagrada.

Callback funkcije se često koriste kao funkcije za upravljanje događajima, odnosno aktiviraju se kao posledica neke aktivnosti korisnika.

Primer 3.20. *Callback* funkcija

```
var niz = [14, 5, 6, -2, 7];
if (niz.every(veci)){
  console.log("Svi elementi su veći od nule.");
}
function veci(a){
  return a > 0;
}
```

U primeru 3.20. funkcija *veci()* je u ulozi *callback* funkcije i zadata je kao parametar metodi *every()*. Ovaj program će u konzoli upisati tekst: „*Svi elementi su veći od nule*”.

3.2.6 Zatvorenja funkcije

Program kada izađe iz funkcije sve lokalne promenljive prestaju da postoje i njima nije moguće pristupiti. Logičan zaključak je da se ne može pristupiti unutrašnjoj funkciji ukoliko program izađe iz nadfunkcije. Međutim, u programskom jeziku *JavaScript* ovo nije slučaj. Moguće je pristupiti unutrašnoj funkciji i kada se završi sa radom sa nadfunkcijom. Ovakva unutrašnja funkcija sme da koristi promenljive svoje nadfunkcije i zovemo je **zatvorenje**.

Dakle, zatvorenje je unutrašnja funkcija koja se poziva van svoje nadfunkcije koja i dalje ima pristup lokalnim promenljivama nadfunkcije.

Najčešća upotreba zatvorenje funkcija je kada se upotrebljava *callback* mehanizam i kada se kreiraju funkcije koje obrađuju događaje.

Primer 3.21. Zatvorenje funkcija

```
function nadfunkcija(){
  var broj = 0;
  function unutrasnjafunkcija(){
    broj++;
    console.log("BROJ: " + broj);
  }
  window.setTimeout(unutrasnjafunkcija, 5000);
  unutrasnjafunkcija();
  console.log("Nadfunkcija je završena!");
}
nadfunkcija();
```

U primeru 3.21. definisana je funkcija *nadfunkcija()*. U njoj je prvo definisana lokalna promenljiva *broj*, a zatim i unutrašnja funkcija *unutrasnjafunkcija()* koja povećava promenljivu broj i ispisuje je. Sledeća u nizu naredba je *setTimeout(unutrasnjafunkcija, 5000)*, ali ona će se izvršiti tek za pet sekundi, odnosno prvo će se izvršiti naredbe koje se nalaze ispod nje. Poziva se funkcija *unutrasnjafunkcija()* koja u konzoli upisuje broj **1**, a zatim se izvršava sledeća naredba, tj. u konzoli se upisuje tekst: „*Nadfunkcija je završena!*".

Poslednja u nizu se izvršava funkcija *unutrasnjafunkcija()* koja je pozvana *callback* mehanizmom. Na ovaj način će se unutrašnja funkcija, nakon završetka rada nadfunkcije, još jednom izvršiti. Promenljivoj *broj* se ponovo pristupa iako je program izašao iz nadfunkcije, a u konzoli će se upisati broj **2**. Dakle, preko funkcije *unutrasnjafunkcija()* ostalo je „živo" celo njeno okruženje, odnosno sve lokalne promenljive iz funkcije *nadfunkcija()*. Isto bi se dogodilo i kada bi program imao više nivoa ugnježdenih funkcija.

3.2.7 Objekat *this*

Objekat *this* predstavlja kontekst u kome se izvršava funkcija.

Na objekat *this* se najčešće nailazi kada se kreiraju funkcije događaja, na primer *this* može predstavljati element na koji je kliknuto.

U programskom jeziku *JavaScript* objekat *this* se formira tokom izvršavanja funkcije. Svaka funkcija ima svoj *this* objekat. *This* može biti pet različitih objekata. Na programerima je da predvide šta će *this* predstavljati u funkciji.

Primer 3.22. Standardni poziv funkcije

```
function fun() {}
fun();
```

U ovom kontekstu *this* se odnosi na *window* objekat.

Primer 3.23. Funkcija kao metod objekta

```
obj.metod = function() {}
obj.metod();
```

Kada je funkcija pozvana na način prikazan u primeru 3.23., *this* je tada objekat iz koga je funkcija pozvana. Odnosno, u funkciji se kreira lokalna promenljiva *this* koja predstavlja referencu na objekat čiji je ta funkcija metod. Ako je jedna ista funkcija metod različitih objekata, *this* će se razlikovati zavisno od toga iz kog objekta je pozvana funkcija.

Primer 3.24. Funkcija kao konstruktor

```
function Klasa() {}
var obj = new Klasa();
```

U primeru 3.24. operator *new* kreira novi objekat i izvršava konstruktorsku funkciju u kontekstu tog objekta (kao da je pozvana iz njega). Objekat *this* u funkciji konstruktora će referencirati na novi objekat.

Primer 3.25. Poziv funkcije preko metode *call()*

```
function fun(p1, p2, p3...) {}
fun.call(obj, v1,v2,v3...)
```

Primer 3.25. prikazuje način pozivanja funkcije metodom *call()*. Metod *call()* uvek ima jedan parametar više od same funkcije. To je zato što prvi parametar predstavlja objekat koji se prosleđuje funkciji kao *this*.

Metod *apply()* koji je prikazan u primeru 3.26., radi isto kao i metod *call()*, ali malo drugačijim stilom. Ovaj metod ima samo dva parametra. Prvi parametar je objekat koji će u funkciji predstavljati *this*, a drugi parametar je niz koji sadrži vrednosti za parametre funkcije.

Primer 3.26. Poziv funkcije preko metode *apply()*

```
function fun(p1, p2, p3...) {}
fun.apply(obj, [v1,v2,v3...]);
```

Primer 3.27. Upotreba *this* objekta

```
function fun(){
    console.log(this.id ? this.id : "Nepoznat ID");
}
var prvi = {id:"PRVI"};
var drugi = {id:"DRUGI"};
prvi.metod = fun;
drugi.poziv = fun;
prvi.metod();
drugi.poziv();
fun();
document.getElementById("blok").onclick = fun;
```

U primeru 3.27. prikazana je upotreba *this* objekta. Prvo je definisana funkcija *fun()*. Ona ima objekat *this*, ali mora se proveriti da li taj objekat ima svojstvo *id*. Ako ima, ispisaće se sadržaj svojstva *id*, ako ne, dobiće se poruka o tome.

Zatim su kreirana dva objekta: *prvi* i *drugi*. Ovim objektima je definisano *id* svojstvo kao string u kome se nalazi njihov naziv. Zatim se tim objektima dodaje po jedan metod, koji predstavlja referencu na funkciju *fun*. Pozivom metoda tih objekata dobijaju se različite vrednosti.

Naredba *prvi.metod();* vratiće string **PRVI**, dok će naredba *drugi.poziv();* vratiti string **DRUGI**. Nakon toga se na standardan način poziva funkcija *fun();*, a tada je objekat *this* referenca na *window*, koji ne poseduje svojstvo *id*.

Na kraju, pretpostavljeno je da u HTML dokumentu postoji neki element, npr. ... . Tom elementu dodata je funkcija događaja koja kada korisnik klikne na njega poziva funkciju *fun()*. Rezultat koji se u ovom slučaju dobija je **blok**, jer objekat *this* će biti referenca na element na koji je kliknuto, a on ima identifikator sa vrednoscu **blok**.

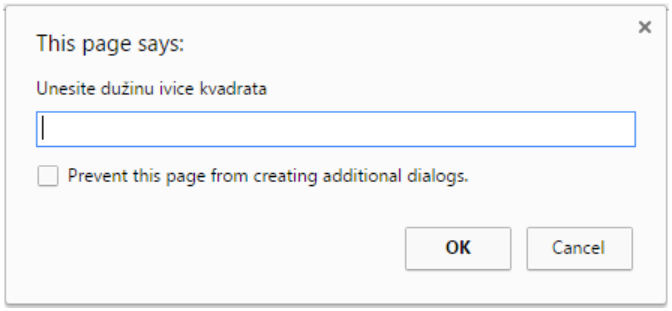
3.3 Primeri

Sledeći primeri objedinjuju pozivanje ugrađenih i korisničkih funkcija ilustrujući njihovu upotrebu.

Primer 3.28. Površina kvadrata (verzija 1)

```
<html>
    <head>
        <title> Površina kvadrata </title>
        <script>
        function kvadratBroja(x){
            return x*x;
        }
        function povrsinaKvadrata(x){
            var povrsina=kvadratBroja(x);
            return povrsina;
        }
        </script>
    </head>
    <body>
        <h1> Površina kvadrata </h1>
        <script>
        var x=prompt("Unesite dužinu ivice kvadrata:");
        document.write("Površina kvadrata je: "+povrsinaKvadrata(x));
        </script>
    </body>
</html>
```

Nakon otvaranja dokumenta iz primera 3.28. funkcija *prompt()* pokreće iskačući prozor prikazan na slici 18.



Slika 18: Prozor koji iskoči prilikom pregledanja dokumenta iz primera 3.28.

Ukoliko je unos u tekstualno polje koje se nalazi u iskačućem prozoru prikazanom na slici 18 validan (ukoliko je unet broj), u internet pregledaču se ispisuje poruka data na slici 19.

Slika 19: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.28. pri validnom unosu

U slučaju koji je prikazan na slici 19, unet je bio broj **7**.

Ukoliko je korisnik kliknuo na dugme „Cancel" ili na dugme „x" kada se pojavio iskačući prozor sa slike 18, ispis u internet pregledaču biće kao na slici 20.

Slika 20: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.28. kada nije bilo unosa

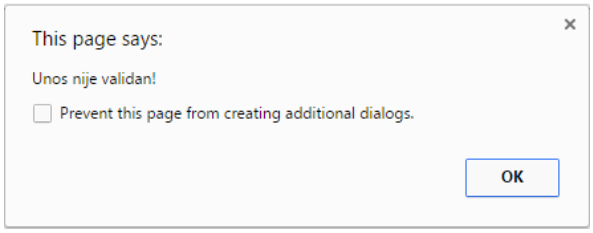
Ako je u iskačući prozor sa slike 18 uneta neka vrednost koja nije broj, rezultat će biti **NaN**.

Funkcija *povrsinaKvadrata()* će izračunavati površinu kada korisnik unese ma koji broj, što nije poželjno, jer dužina ne može biti negativan broj. Iz tog razloga, potrebno je zabraniti unos negativnih brojeva. Ova greška je ispravljena u primeru 3.29.

Primer 3.29. Površina kvadrata (verzija 2)

```
<html>
    <head>
        <title> Površina kvadrata </title>
        <script>
        function kvadratBroja(x){
            return x*x;
        }
        function povrsinaKvadrata(x){
            if(x>=0){
                var povrsina=kvadratBroja(x);
                return povrsina;
            }
            else
            alert("Unos nije validan!");
        }
        </script>
    </head>
    <body>
        <h1> Površina kvadrata </h1>
        <script>
        var x=prompt("Unesite dužinu ivice kvadrata:");
        document.write("Površina kvadrata je: "+povrsinaKvadrata(x));
        </script>
    </body>
</html>
```

Ukoliko unos u iskačući prozor koji aktivira funkcija *prompt* iz primera 3.29. ne bude regularan, odnosno, ukoliko nije pozitivan broj, aktivira se funkcija *alert()* koja pokreće iskačući prozor prikazan na slici 21.



Slika 21: Prozor koji iskoči prilikom pregledanja dokumenta iz primera 3.29.

Kada se ukloni iskačući prozor klikom na dugme „OK" ili „x", funkcija nastavlja sa radom i ispisuje se tekst sa slike 22.

Slika 22: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.29. kada nije bilo unosa

Kako nije bilo validnog unosa ne treba biti ni ispisivanja u dokument, pa je ova greška ispravljena u primeru 3.30.

Primer 3.30. Površina kvadrata (verzija 3)

```
<html>
    <head>
        <title> Površina kvadrata </title>
        <script>
        function kvadratBroja(x){
            return x*x;
        }
        function povrsinaKvadrata(x){
            if(x>=0){
                var povrsina=kvadratBroja(x);
                return povrsina;
            }
            else
            alert("Unos nije validan!");
        }
        </script>
    </head>
`<body>
        <h1> Površina kvadrata </h1>
        <script>
        var x=prompt("Unesite dužinu ivice kvadrata:");
        if(x>=0){
            document.write("Površina kvadrata je: "+povrsinaKvadrata(x));
        }
        else{
            document.write("Površina nije izračunata.");
        }
        </script>
    </body>
</html>
```

Pri nevalidnom unosu ispis u internet pregledaču će biti kao na slici 23.

Slika 23: Sadržaj internet stranice nakon otvaranja HTML dokumenta iz primera 3.30. pri nevalidnom unosu

Zadatak je moguće rešiti na više načina. Ovo je samo jedno od rešenja koje nije ni potpuno ispravno jer postoji deo koda koji se nikada ne izvršava. Na slici 24 označen je deo koda koji se nikada ne izvršava.

```
<html>
    <head>
        <title> Površina kvadrata </title>
        <script>
        function kvadratBroja(x){
            return x*x;
        }
        function povrsinaKvadrata(x){
            if(x>=0){
                var povrsina=kvadratBroja(x);
                return povrsina;
            }
            else
            alert("Unos nije validan!");
        }
        </script>
    </head>
`<body>
        <h1> Površina kvadrata </h1>
        <script>
        var x=prompt("Unesite dužinu ivice kvadrata:");
        if(x>=0){
            document.write("Površina kvadrata je: "+povrsinaKvadrata(x));
        }
        else{
            document.write("Površina nije izračunata.");
        }
        </script>
    </body>
</html>
```

Slika 24: Deo koda HTML dokumenta iz primera 3.30. koji se nikada neće izvršiti

Ovaj deo se ne izvršava jer funkciji se pristupa samo ukoliko je $x \geq 0$, tako da pitanje da li je $x \geq 0$ unutar funkcije je suvišno i odgovor na to pitanje je uvek potvrdan.

Primer 3.31. Palindrom

```
function ispitajPalindrom(rec){
    var novaRec = rec.split("").reverse().join("");
    return rec == novaRec;
}
function najduziPalindrom(rec){
    var najvecaDuzina = 0,
    najveciPalindrom = '';
    for(var i=0; i < rec.length; i++){
        var deoReci = rec.substr(i, rec.length);
        for(var j=deoReci.length; j>=0; j--){
            var deoDelaReci = deoDelaReci(0, j);
            if (deoDelaReci.length <= 1)
                continue;
            if (ispitajPalindrom(deoDelaReci)){
                if (deoDelaReci.length > najvecaDuzina){
                    najvecaDuzina = deoDelaReci.length;
                    najveciPalindrom = deoDelaReci;
                }
            }
        }
    }
    return najveciPalindrom;
}
```

Primer 3.31. prikazuje funkciju tj. kombinaciju funkcija koje u prosleđenoj reči pronalaze najduži palindrom. Palindromi su reči, izrazi, rečenice, brojevi ili drugi nizovi znakova i simbola koji imaju isto značenje bilo da se čitaju unapred ili unazad (npr. pop, 1001, 1+1, "ana voli milovana"itd.).

Funkcija *ispitajPalindrom* proverava da li je prosleđena reč palindrom. Prvo se string metodom *split()* prosleđena reč pretvori u niz karaktera, zatim metodom *reverse()* se tako dobijen niz preokrene, a na kraju se metodom *join()* preokrenut niz pretvara u string. Rezultat je string preokrenut u odnosu na početni koji je sačuvan i promenljivoj *novaRec*. Funkcija na kraju proverava da li su jednaki početni string i novonastali preokrenuti string i ukoliko je odgovor potvrdan vraća vrednost **true**, u suprotnom vraća vrednost **false**.

Funkcija *najduziPalindrom* prosleđeni string razlaže na podstringove i proverava svaki da li je palindrom koristeći se funkcijom *ispitajPalindrom*. U promenljivoj *najveciPalindrom* će se sačuvati najduži palindrom koji sadrzi prosleđeni string, a u promenljivoj *najvecaDuzina* će se sačuvati dužina tog palindroma. Prva *for* petlja prolazi kroz string od početka ka kraju, dok druga *for* petlja od pozadi skraćuje podstringove koji se proveravaju.

Na primer, ukoliko se proverava reč *banana*, provera će ići sledećim tokom:

1. banana ($i = 0, j = 6$);
2. banan ($i = 0, j = 5$);

3. bana $(i = 0, j = 4)$;
4. ban $(i = 0, j = 3)$;
5. ba $(i = 0, j = 2)$;
6. b $(i = 0, j = 1)$;
7. anana $(i = 1, j = 6)$;
8. anan $(i = 1, j = 5)$;
9. ana $(i = 1, j = 5)$;

   . . .

Provera se nastavlja istim tokom, ne prekida se iako je pronađena reč koja jeste najduži palindrom. Prvi pronađeni palindrom je slovo **b** i tada promenljiva *najvecaDuzina* uzima vrednost **1**. Sledeća reč *anana* je takođe palindrom, dužine **5**, pa promenljiva *najvecaDuzina* uzima vrednost **5**, jer je **5**>**1**. Sledeći palindrom na koji funkcija naiđe je *ana*, ali on je kraće dužine, pa se vrednost promenljive *najvecaDuzina* neće promeniti. Funkcija će još jednom naići na reč *ana* koja je palindrom, koja neće promeniti vrednost promenljive *najvecaDuzina*.

Kada funkcija izađe iz obe *for* petlje vraća se vrednost sačuvana u promenljivoj *najvecaDuzina* koja bi u navedenom primeru bila broj **5**.

Primer 3.32. Najmanji zajednički sadržalac i najveći zajednički delilac

```
function NZS(x, y){
  return Math.abs((x * y) / NZD(x, y));
}

function NZD(x, y){
  x = Math.abs(x);
  y = Math.abs(y);
  while(y){
    var t = y;
    y = x % y;
    x = t;
  }
  return x;
}
```

Funkcije prikazane u primeru 3.32. pronalaze najmanji zajednički sadržalac i najveći zajednički delilac dva broja.

Funkcija *NZS* pronalazi najmanji zajednički sadržalac prosleđenih brojeva tako što apsolutnu vrednost proizvoda tih brojeva podeli sa njihovim najvećim zajedničkim deliocem.

Funkcija *NZD* pronalazi najveći zajednički delilac prosleđenih brojeva. Prvo se oba broja pretvore u njihovu apsolutnu vrednost, odnosno ukoliko je neki broj bio pozitivan, njegova vrednost se neće promeniti. Zatim se *while* petljom

primenjuje Euklidov algoritam za pronalaženje najvećeg zajedničkog delioca sve dok je druga promenljiva veća od nule (vrednosti prve i druge promenljive se menjaju prolaskom kroz petlju). Kada se izađe iz *while* petlje, odnosno, kada vrednost druge promenljive postane **0**, vratiće se vrednost prve promenljive koja predstavlja najveći zajednički delilac brojeva.

4 Objekti u programskom jeziku *JavaScript*

JavaScript je jezik zasnovan na objektima. Skoro sve *JavaScript* mogućnosti su, u većoj ili manjoj meri, implementirane korišćenjem objekata.

Objekat u *JavaScript* jeziku predstavlja kolekciju svojstava. Ova svojstva mogu biti primitivni tipovi podataka ili drugi objekti. Drugim rečima, objekat je neuređen skup svojstava od kojih svako svojstvo ima ime i vrednost.

Svojstva objekta se ponašaju kao promenljive, u njih se mogu upisivati vrednosti i čitati odatle. Postoji mogućnost da se svojstva *JavaScript* objekata proglase nepromenljivim tj. da se u njih ne mogu upisivati vrednosti. Svojstvima objekta se može pristupiti tako što se navede objekat, potom tačka i naziv svojstva.

4.1 Definisanje objekta

U primeru 4.1. prikazan je način na koji se u programskom jeziku *JavaScript* definišu objekti.

Primer 4.1. Definisanje objekta

```
var auto={
    vrsta: "limuzina",
    brojVrata: 5,
    motor:{
        snaga: '185hp',
        zapremina: 1.5
    },
    potrosnja: function(a,b){
            return a+b;
            }
    };
```

Prilikom definisanja objekta, prvo se navodi ključna reč *var* koja označava promenljivu, a zatim se navodi njegovo ime. U ovom slučaju za ime je izabrana reč **auto**. Učestalo je da ime bliže opisuje objekat kojem se pridružuje. Ime je kombinacija karaktera, ali nikako ne sme biti neka od rezervisanih reči koje su prikazane u tabeli na slici 15.

Nakon imena objekta se navodi znak jednakosti („=”), a potom vitičaste zagrade unutar kojih se navode svojstva koja taj objekat poseduje. Navedeni objekat poseduje četiri svojstva: *vrsta, brojVrata, motor* i *potrošnja*. Za imena svojstava, tj. osobine nekog objekta važe ista pravila kao i za ime samog objekta. Nakon navedenog imena osobine navode se dve tačke („:”), a potom vrednost koja joj je dodeljena.

Vrsta je svojstvo tipa *string*, brojVrata je svojstvo tipa *int*, motor je svojstvo tipa objekat (ima svoja dva svojstva koja su tipova *string* i *int*), a potrošnja je funkcijsko svojstvo (prilikom poziva ona računa zbir dva prosleđena argumenta).

Promenljivama objekta iz primera 4.1. pristupa se na sledeći način:

- auto.vrsta - vratiće string **limuzina**;
- auto["vrsta"] - vratiće string **limuzina**;
- auto.motor.snaga - vratiće string **185hp**;
- auto.potrosnja(8,100) - vratiće broj **108**.

Ukoliko se pojavi potreba da se kasnije dodefiniše neko svojstvo, to je u ovom jeziku dozvoljeno i izvodi se na način prikazan u primeru 4.2.

Primer 4.2. Naknadno definisanje svojstava objekta

```
var auto={
    vrsta: "limuzina",
    brojVrata: 5
    };
auto.nov=true;
```

U primeru 4.2. može se videti da je objekat auto isprva definisan sa dva svojstva, *vrsta* i *brojVrata*, a potom mu se dodaje svojstvo *nov*, tipa *boolean*, čija se vrednost postavlja na **true**.

Primer 4.3 ilustruje menjanje osobina postojećih objekata.

Primer 4.3. Promena vrednosti osobina objekta

```
var auto={
    vrsta: "limuzina",
    brojVrata: 5
    };
auto.brojVrata=3;
```

Objekat u primeru 4.3. ima dva svojstva, *vrsta* i *brojVrata*, čije su vrednosti, redom, **limuzina** i **5**. Potom je svojstvu *brojVrata* promenjena vrednost na **3**.

Moguće je formirati i potpuno prazan objekat, bez svojstava, kome će se naknadno dodavati neophodna svojstva. Definisanje praznog objekta prikazano je u primeru 4.4.

Primer 4.4. Definisanje praznog objekta

```
var prazanObjekat = {}
```

4.2 Primeri

U nastavku su prikazani primeri definisanja objekata.

Primer 4.5. Objekat *osoba*

```
var osoba = {
    ime:"Petar",
    prezime:"Petrović",
    starost:50,
    bojaOciju:"plave"
};
```

Objekat *osoba* definisan u primeru 4.5. ima četiri svojstva i sva su primitivnih tipova.

Primer 4.6. Objekat *korisnik*

```
var korisnik= {
    ime: "Gost",
    promeniIme: function() {
        this.ime = prompt("Vaše ime je?")
    },
    pozdrav: function() {
        alert('Zdravo, moje ime je: '+this.ime)
    }
}
```

Objekat *korisnik* u primeru 4.6. definisan je za potrebe registracije korisnika na jednom forumu. On na početku svome svojstvu *ime* dodeljuje vrednost **Gost**, a potom se može funkcijskim svojstvom *promeniIme* promeniti vrednost ovog svojstva. Osim toga, definisano je još jedno funkcijsko svojstvo *pozdrav*, koje kada se pozove aktivira iskačući prozor sa porukom: **'Zdravo, moje ime je: '+this.ime**, gde deo *this.ime* prikazuje vrednost sačuvanu u svojstvu *ime* tog korisnika.

Primer 4.7. Objekat *stepeniste*

```
var stepeniste = {
  stepenik: 0,
  gore: function() {
    this.stepenik++
  },
  dole: function() {
    this.stepenik--
  },
  prikaziStepenik: function() {
    alert(this.stepenik)
  }
}
```

Objekat *stepeniste* u primeru 4.7. je simulacija pravog stepeništa. On sadrži numeričko svojstvo *stepenik* u kome se čuva informacija o tome na kom se stepeniku korisnik trenutno nalazi, zatim dva funkcijska svojstva *gore* i *dole* kojim se korisnik može kretati po stepeništu i funkcijsko svojstvo *prikaziStepenik* kojim se u iskačućem prozoru ispisuje numerička vrednost stepenika na kom se korisnik nalazi.

Primer 4.8. Objekat *zaposlen*

```
var zaposlen = new Object();
    zaposlen.ime ="Nikola";
    zaposlen.prezime = "Nikolić";
    zaposlen.zanimanje = "Programer";
    zaposlen.prikaziZanimanje = function(){
        alert(this.zanimanje);
    }
    zaposlen.prikaziZanimanje();
```

Objekat *zaposlen* u primeru 4.8. predstavlja zaposlenog u nekom preduzeću. On sadrži četiri svojstva, od kojih su prva tri niske, dok je poslednje funkcijsko svojstvo, kojim se prikazuje zanimanje zaposlenog.

5 Zadaci sa rešenjima

U nastavku su prikazani zadaci koji objedinjuju upotrebu funkcija i objekata u programskom jeziku *JavaScript*. Navedena rešenja zadataka nisu jedinstvena, već samo jedno od mogućih rešenja postavljenih problema.

Zadatak 1. Napraviti internet stranicu koja računa iznos poreza na imovinu. Unete podatke sačuvati u objektu *imovina*, a zatim ispisati sve podatke tog objekta, kao i visinu poreza. Koristiti formulu: porez=površinaImovine · 30.000,00 RSD-0,08 · godinePosedovanjaImovine · 30.000,00 RSD. Očekuje se da će korisnik korektno uneti vrednosti.

Rešenje:

```
<script>
//Definisanje objekta
var imovina = {
    povrsinaImovine: 0,
    godinePosedovanjaImovine: 0,
    porez:function(){
        //Porez se računa na osnovu date formule
        var porez=this.povrsinaImovine*30000.00-0.08*
        this.godinePosedovanjaImovine*30000.00;
        return porez;
    }
}
//Zahteva se od korisnika unos površine imovine
imovina.povrsinaImovine=prompt("Povrsina vaše imovine
za koju želite da izračunate visinu poreza je: ");
//Ispisuje se unešena površina imovine
document.write("Površina vaše imovine je: "
+imovina.povrsinaImovine+" m<sup>2</sup><br>");
//Zahteva se od korisnika unos godina posedovanja
imovina.godinePosedovanjaImovine=prompt("Koliko
godina posedujete navedenu imovinu?");
//Ispisuje se unešen broj godina posedovanja
document.write("Imovinu posedeujete "
+imovina.godinePosedovanjaImovine+" godina. <br>");
//Ispisuje se visina poreza pozivanjem funkcije
document.write("Visina poreza na vašu imovinu je "
+imovina.porez()+" dinara.");
</script>
```

Slika 25: Rešenje zadatka 1

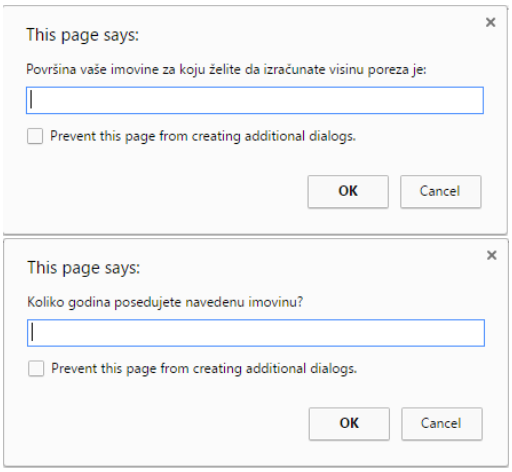
Po uslovima zadatka prvo je definisan objekat *imovina*, koji ima osobine:

- *povrsinaImovine* u kojoj je sačuvana numerička vrednost površine imovine za koju korisnik želi da izračuna visinu poreza;

- *godinePosedovanjaImovine* u kojoj je sačuvana numerička vrednost godina koliko je korisnik vlasnik te imovine;
- *porez* koja je funkcijska osobina i izračuvana visinu poreza po formuli koja je zadata u tekstu zadatka.

Potom, na internet stranici se funkcijom *prompt* zahteva unos neophodnih podataka, koji se, redom, ispisuju funkcijom *write*. Na kraju, ispisuje se i vrednost funkcijskog svojstva objekta *imovina*.

Nakon otvaranja HTML dokumenta sa slike 25 pojavljuju se dva iskačuća prozora prikazana na slici 26, jedan za drugim.



Slika 26: Prozori koji se pojave nakon otvaranja HTML dokumenta sa slike 25

Nakon unosa podataka **50** i **10** redom, na internet stranici se ispisuje tekst sa slike 27.

Porez na imovinu

Površina vaše imovine je: 50 m^2
Imovinu posedujete 10 godina.
Visina poreza na vašu imovinu je: 1476000 dinara.

Slika 27: Sadržaj internet stranice nakon otvaranja HTML dokumenta sa slike 25

Zadatak 2. Napraviti internet stranicu koja u zavisnosti od toga da li je dan ili noć boju pozadine postavi na žutu ili plavu (dan je od 00h do 20h, a noć od 20h do 00h).

Rešenje: U okviru funkcije *oboji* definisana je nova promenljiva *datum* u kojoj se čuva rezultat poziva funkcije *Date*. Funkcijom *oboji* menja se boja pozdine na žutu ili plavu. Da li će boja pozadine biti plave ili žute boje određuje naredba *if* koja ispituje koliko je sati u promenljivoj *datum*.

```
<html>
    <head>
        <title> Zadatak 2 </title>
    </head>
    <body>
        <script>
        //Definisanje funkcije koja će obojiti pozadinu
        function oboji(){
        //Kreiranje promenljive sa trenutnim datumom
        var datum=new Date();
        if(datum.getHours()<20)
        document.body.style.backgroundColor = "yellow";
        else
        document.body.style.backgroundColor = "blue";
        }
        //Pozivanje funkcije koja će obojiti pozadinu
        oboji();
        </script>
    </body>
</html>
```

Slika 28: Rešenje zadatka 2

Kada korisnik testira dokument sa slike 28 boja pozadine se oboji u plavo ili žuto u zavisnosti od vremena kada se dokument pregleda.

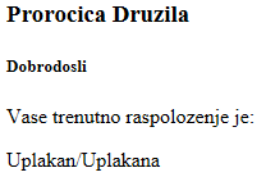
Zadatak 3. Napraviti internet stranicu proročice Druzile, koja pogađa raspoloženje korisnika u trenutku pristupanja toj stranici.

Rešenje: Definisan je niz stringova pod nazivom *raspolozenje* u kome su sačuvana različita raspoloženja koja osoba može imati. Funkcija *pogodi* slučajno bira celi broj iz intervala $[0, 9]$, a potom se na internet stranici ispisuje string iz niza *raspolozenje*, koji stoji na slučajno odabranom mestu.

```
<html>
    <head>
        <title> Zadatak 3 </title>
    </head>
    <body>
        <h3> Proročica Druzila </h3>
        <p> Vaše trenutno raspoloženje je: </p>
        <script>
        //Definisanje niza stringova od kojih će slučajno biti izabran
        //jedan član, a koji će predstavljati predviđanje proročice Druzile
        var raspolozenje=["Tuzan/Tuzna", "Uplakan/Uplakana",
                          "Srecan/Srecna", "Radostan/Radosna",
                          "Veseo/Vesela", "Isfrustriran/Isfrustrirana",
                          "Zaljubljen/Zaljubljena", "Namrgodjen/Namrgodjena",
                          "Ljubomoran/Ljubomorna", "Nasmejan/Nasmejana"];
        //Definisanje funkcije koja slučajno
        //bira jedan ceo broj iz intervala [0,9]
        function pogodi(){
            var rnd=Math.random();
            return Math.floor(rnd*10);
        };
        //Ispisivanje člana niza "raspoloženje"
        //koji stoji na slučajno odabranom mestu
        document.write(raspolozenje[pogodi()]);
        </script>
    </body>
</html>
```

Slika 29: Rešenje zadatka 3

Nakon otvaranja dokumenta sa slike 29 na internet stranici se pojavi poruka prikazana slikom 30.



Slika 30: Sadržaj internet stranice nakon otvaranja dokumenta sa slike 29

Na slici 30 prikazano je jedno od proricanja koja su testirana.

Zadatak 4. Napraviti simulaciju izvlačenja brojeva za igru na sreću „Loto”.

Rešenje: Na slici 31 prikazan je HTML dokument, koji sadrži dve funkcije.

```
<script>
//Funkcija koja slučajnim izborom bira celi broj iz intervala [1,39]
function izvuciBroj(){
    var rnd=Math.random();
    return Math.ceil(rnd*39);
};
//Funkcija koja izvlači sedam brojeva
function svihSedamBrojeva(){
    var izvuceni=[], novBroj=0, ind=0;
    for(i=0;i<7;i++){
        ind=0;
        novBroj=izvuciBroj();
        //Provera da li se broj ponovio
        for(j=0;j<i;j++){
            if(novBroj==izvuceni[j]){
            ind=1;
            }
        }
        if(ind==0){
        //Ukoliko se nije ponovio dodajemo ga u niz izvučenih brojeva
        izvuceni[i]=novBroj;
        }
        //Ukoliko već postoji broj u nizu ponavljamo izvlačenje
        else if(ind==1){
        i--;
        }
    }
    return izvuceni;
}
document.write(svihSedamBrojeva());
</script>
```

Slika 31: Rešenje zadatka 4

Funkcija *izvuciBroj()* pri pozivu određuje jedan slučajno odabrani celi broj u intervalu brojeva $[1, 39]$. Nakon toga funkcija *svihSedamBrojeva()* najmanje sedam puta poziva funkciju *izvuciBroj()* u prvoj *for* petlji. Drugom *for* petljom se proverava da li je broj ponovljen prilikom izvlacenja - ako je to slučaj, preskače se i izvlačenje se ponavlja. Na kraju, rezultat se ispisuje na internet stranici.

Loto

Ko igra, taj i dobija

26,37,1,27,2,25,3

Slika 32: Sadržaj internet stranice nakon otvaranja HTML dokumenta sa slike 31

Na slici 32 prikazano je jedno od izvlačenja koja su testirana.

Zadatak 5. Napraviti internet stranicu koja proračunava da li je određena osoba pravi prijatelj, odnosno da li joj se može verovati. Kalkurator proračuna prijateljstva bazirati na string metodi *localCompare()* upoređujući ime i prezime osobe.

Rešenje: HTML dokument koji predstavlja rešenje zadatka 5 prikazan je na slici 33.

```
<html>
    <head>
        <title> Zadatak 5 </title>
    </head>
    <body>
        <h3> (Ne)Prijatelj </h3>
        <h5> Da li da ti verujem? </h5>
        <script>
        //Funkcija koja zahteva unos imena
        function ime(){
            return prompt("Unesite ime osobe:");
        }
        //Funkcija koja zahteva unos prezimena
        function prezime(){
            return prompt("Unesite prezime osobe:");
        }
        //Funkcija koja upoređuje ime i prezime osobe
        function proracunPrijateljstva(i, p){
            if(i.localeCompare(p)<0){
                return "Ne";
            }
            else if(i.localeCompare(p)>0){
                return "Reci ove osobe uzimajte sa rezervom";
            }
            else{
                return "Da, potpuno";
            }
        }
        //Pozivanje funkcije "ime" čija se vrednost čuva u promenjivoj "imeOsobe"
        var imeOsobe=ime();
        //Pozivanje funkcije "prezime" čija se vrednost čuva u promenjivoj "prezimeOsobe"
        var prezimeOsobe=prezime();
        //Ispisivanje vrednosti funkcije "proracunPrijateljstva"
        document.write(proracunPrijateljstva(imeOsobe, prezimeOsobe));
        </script>
    </body>
</html>
```

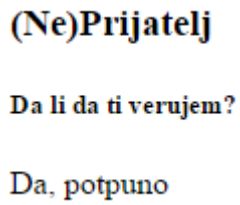
Slika 33: Rešenje zadatka 5

Definisane su tri funkcije:

- *ime* - koja od korisnika traži da se unese ime osobe;
- *prezime* - koja od korisnika traži da se unese prezime osobe;
- *proracunPrijateljstva* - koja poredi prosleđeno ime i prezime i u zavisnosti od rezultata vraća string *„Ne”*, *„Reci ove osobe uzimajte sa rezervom”* ili *„Da, potpuno”*.

Ime i prezime se porede string metodom *localCompare*, koja ispituje da li prvi string pri poređenju stoji pre drugog (u tom slučaju povratna vrednost ove funkcije je **-1**), nakon drugog (u tom slučaju povratna vrednost ove funkcije je **1**), ili su stringovi jednaki (u tom slučaju povratna vrednost je **0**).

Internet stranica nakon otvaranja dokumenta sa slike 33 i unošenja neophodnih podataka ispisuje tekst prikazan slikom 34.



Slika 34: Sadržaj internet stranice nakon otvaranja HTML dokumenta sa slike 33

6 Zaključak

Elektronske lekcije o funkcijama i objektima u programskom jeziku *JavaScript*, napravljene za potrebe ovog master rada, prevashodno su namenjene učenicima osnovnih i srednih škola, ali i svima onima koji žele da steknu ili prošire svoja znanja o ovom skript jeziku.

U školama se na časovima informatike aktivno izučavaju programski jezici *C, C++, Pascal, C#, Delphi* itd. Učenici koji žele da znaju više često uče i programski jezik *JavaScript*. Elektronske lekcije o funkcijama i objektima o programskom jeziku *JavaScript* će im u mnogome pomoći, zato što učenici lakše prihvataju znanja stečena kroz primere. Učenici i ostali korisnici mogu odmah da testiraju stečeno znanje u delu za unos koda.

Značaj elektronskih lekcija o funkcijama i objektima u programskom jeziku *JavaScript* je višestruki jer su stalno dostupne za bilo koji vid podsećanja ili ponovnog učenja i sadrže veliki broj primera i zadataka sa rešenjima. Elektronske lekcije o funkcijama i objektima u programskom jeziku *JavaScript* imaju veliku prednost u odnosu na štampane verzije. Ukoliko dođe do unapređenja jezika, knjige je neophodno ponovo štampati. S druge strane, elektronske lekcije je potrebno samo izmeniti, unaprediti ili proširiti i odmah će biti spremne za upotrebu. Shodno tome, elektronske lekcije na duži period štede vreme, novac i prirodne resurse. Odlučujući faktor pri izboru literature je kvalitet i kvanititet znanja koje se stiče i lakoća savladavanja gradiva, zbog čega je prilikom izrade elektronskih lekcija o funkcijama i objektima u programskom jeziku *JavaScript* uložen dodatan trud i rad.

Literatura

[1] J. N. Robbins, Learning Web Design, Fourth Edition, 2012

[2] J. Duckett, Beginning HTML, XHTML, CSS, and JavaScript, 2010

[3] S. Pantić, Uvod u JavaScript, 1997

[4] L. Lemay, R. Colburn, J. Kyrnin, HTML5, CSS3 I JavaScript za razvoj veb strana, 2016

[5] J. McPeak, JavaScript 24-asovna obuka, 2012

[6] K. Simpson, Nauite JavaScript, 2016

[7] D. Crockford, JavaScript: The Good Parts, 2008

[8] D. Herman, Effective JavaScript, 2013

[9] N. Jurić, eŠkola veba, sedmi simpozijum „Matematika i primene", 2016

[10] JavaScript Tutorials, www.htmldog.com/guides/javascript pristupljeno septembra 2016.

[11] JavaScript, www.codecademy.com/learn/javascript pristupljeno oktobra 2016.

[12] JavaScript, www.javascript.com pristupljeno novembra 2016.

[13] JavaScript Tutorial, www.w3schools.com/js pristupljeno novembra 2016.

[14] The State of JavaScript: Language Versions, www.devproconnections.com/javascript/state-javascript-language-versions pristupljeno novembra 2016.

[15] Zašto JavaScript programski jezik?: www.almirvuk.blogspot.rs/2016/05/zasto-javascript-programski-jezik.html pristupljeno novembra 2016.