

## Contents

JavaScript.....	1
Sintaksa.....	1
Implementacija JavaScript u HTML.....	1
JavaScript promenljive.....	2
Tipovi podataka.....	3
Celi brojevi.....	3
Brojevi u pokretnom zarezu.....	4
Boolean.....	4
Stringovi.....	4
Operatori.....	4
Aritmetički operatori:.....	4
Operatori dodeljivanja:.....	4
Operatori poredjenja:.....	5
Logički operatori:.....	5
String operator.....	5
Uslovni operator.....	5
Prioriteti operatora.....	6
Nizovi.....	7
Osnovne funkcije.....	8
Opšte JavaScript funkcije.....	8
eval( ).....	8
parseFloat( ).....	8
escape( ).....	9
unescape( );.....	9
length.....	9
JavaScript funkcije nizova.....	10
concat( ).....	10
join( ).....	10
pop( ).....	10
shift( ).....	11
push( ).....	11
unshift( ).....	11
reverse( ).....	11

slice( ).....	12
sort( ).....	12
splice( ).....	12
Tekstualne string funkcije.....	12
String funkcije.....	13
charAt( ).....	14
charCodeAt( ).....	14
fromCharCode( ).....	14
indexOf( ).....	14
slice( ).....	14
split( ).....	15
substr( ).....	15
toLowerCase( ).....	16
toUpperCase( ).....	16
JavaScript regularni izrazi.....	16
Modifikatori:.....	16
Zagrade.....	16
Metakarakter.....	17
Kvantifikatori.....	17
match( ).....	18
replace( ).....	18
search( ).....	18
Matematičke funkcije.....	19
Funkcije za brojeve.....	19
Konstante.....	19
Matematičke funkcije.....	20
• Datum funkcije.....	20
Kontrolne strukture.....	23
JavaScript IF, IF ELSE i SWITCH grananja.....	23
If.....	23
SWITCH.....	24
JavaScript FOR, WHILE i DO WHILE petlje.....	25
For petlja.....	25
while petlja.....	25

<b>Do while</b> .....	26
<b>continue</b> .....	26
<b>Sopstvene funkcije</b> .....	26
<b>Dogadjaji</b> .....	28
<b>Obradivači događaja</b> .....	29
<b>OnLoad</b> .....	30
<b>OnUnload</b> .....	30
<b>OnClick</b> .....	31
<b>OnFocus</b> .....	31
<b>OnBlur</b> .....	31
<b>OnChange</b> .....	32
<b>OnSelect</b> .....	32
<b>OnSubmit</b> .....	33
<b>OnMouseOver</b> .....	33
<b>OnMouseOut</b> .....	34
<b>OnKeyUp</b> .....	34
<b>OnMouseMove</b> .....	35
<b>Browser funkcije</b> .....	35
<b>Alert, confirm i prompt window dijaloški prozori</b> .....	36
<b>CONFIRM</b> .....	37
<b>PROMPT</b> .....	37
<b>Otvaranje stranice u iskaćućem prozoru</b> .....	38
<b>Close(), moveBy(), moveTo(), focus() i blur()</b> .....	39
<b>Close( )</b> .....	39
<b>MoveBy( )</b> .....	40
<b>MoveTo( )</b> .....	40
<b>Focus( )</b> .....	40
<b>Blur( )</b> .....	40
<b>Printanje strane</b> .....	40
<b>SetTimeout(),clearTimeout(),setInterval(), clearInterval()</b> .....	41
<b>SetTimeout( )</b> .....	41
<b>ClearTimeout( )</b> .....	41
<b>SetInterval( ) i clearInterval( )</b> .....	41
<b>ResizeBy() i resizeTo()</b> .....	42

<b>ResizeBy( )</b> .....	42
<b>ResizeTo( )</b> .....	42
<b>ScrollBy() i scrollTo()</b> .....	42
<b>ScrollBy( )</b> .....	42
<b>ScrollTo( )</b> .....	43
<b>Navigatorske funkcije</b> .....	43
<b>JavaEnabled( )</b> .....	43
<b>Ekranske (screen) funkcije</b> .....	44
<b>History funkcije</b> .....	44
<b>History.length</b> .....	44
<b>History.back( ) i history.forward( )</b> .....	45
<b>Go(argument)</b> .....	45
<b>Location funkcije</b> .....	45
<b>HTML DOM objekti</b> .....	46
<b>Document.anchors[]</b> .....	46
<b>document.forms[] i document.images[]</b> .....	47
<b>Document.links[]</b> .....	47
<b>Document.cookie i još neke funkcije</b> .....	49
<b>Document.getElementById(id)</b> .....	50
<b>Anchor funkcije</b> .....	50
<b>Tastatura i miš (EVENT atributi)</b> .....	51
<b>Tabele</b> .....	53
<b>Cells[] i rows[]</b> .....	53
<b>Border, cellPadding, cellSpacing, width i height</b> .....	54
<b>Frame</b> .....	54
<b>Rules</b> .....	55
<b>InsertRow() i insertCell()</b> .....	56
<b>DeleteRow(), deleteCell() i rowIndex</b> .....	56
<b>Align</b> .....	57
<b>ColSpan i rowSpan</b> .....	57
<b>InnerHTML</b> .....	58
<b>Width</b> .....	58
<b>VAlign</b> .....	59
<b>JS i CSS</b> .....	59

JS background CSS funkcije.....	60
JS border CSS funkcije.....	61
JS margin CSS funkcije.....	62
JS padding CSS funkcije.....	63
JS layout CSS funkcije.....	63
JS list CSS funkcije.....	66
JS CSS funkcije za poziciju.....	67
JS tekstualne CSS funkcije.....	68
CANVAS.....	70
CANVAS ugradnja u HTML.....	71
Crtanje pravougaonika.....	72
Crtanje raznih oblika.....	73
Bezijerove krivulje.....	75
Klasične slike.....	77
Stilovi i boje.....	79
Transparentnost boja.....	80
Stil linija.....	81
Nijansiranje boja.....	82
Pattern.....	84
Senke.....	85
Saving i restoring stanja.....	86
Translacije.....	87
Rotacije.....	88
Skaliranje.....	88
Transformacije.....	90
Kompozicije.....	92
Isečci.....	93
Složeniji primeri CANVAS grafike.....	94
JavaScript skripte.....	95
U ovom delu daćemo vrlo praktične skripte koje se često koriste prilikom izrade sajtova.	
Implementacija je Časovnik.....	95
Kalendar.....	96
Povratak na vrh.....	96
Sneg.....	98

Slajd šou.....	100
Animirani tekst.....	106
Harmonika.....	108
Okvir za sadržaj.....	113
Objasnjenje u formi.....	120
Brojanje znakova.....	122
Tekst editor.....	124
Uvecanje slike.....	131
Lepljivi boks.....	134
Skrolovan tekst 1.....	137
Skrolovan tekst 2.....	139
Skrolovanje slike i teksta.....	141
Velicina fonta.....	147

## JavaScript

JavaScript programski jezik unapredjuje HTML dokument u smislu dinamičnosti, realizuje razne grafičke efekte, ali i funkcionalno i interaktivno. Kao takav JavaScript se integriše u HTML kod. Iz ovoga verovatno pretpostavljate da vam je besmisleno učiti JavaScript ukoliko neznate HTML. Medjutim preporučujemo da takodje naučite i CSS pre nego krenete sa učenjem JavaScript-a.

JavaScript se izvršava jedino na klijentskoj strani, odnosno u brauzeru. JavaScript ne može da razmenjuje podatke sa hard diskom.

Evo konkretno na primer, JavaScript može izvršavati odredjene akcije kada se recimo završi učitavanje stranice, ili kada korisnik klikne, ili predje preko nekog HTML elementa-. Takodje JavaScript može napisati neki tekst u zavisnosti od neke akcije koju izvede posetilac na stranici. Može čitati i menjati sadržaj nekog HTML elementa. Može se koristiti za proveru valjanosti podataka koji se unose u HTML forme pre nego što se pošalju na server. To štedi resurse servera. I još mnoge druge korisne stvari možemo obaviti JavaScript-om unutar HTML dokumenta koji sam HTML ne može.

JavaScript je stvorio Brendan Eich na Netscape 1996 godine. JavaScript se i danas usavršava i razvija. Dodatnu popularnost JavaScript jeziku doneo je AJAX koji omogućava JavaScript-u da u svakom trenutku nezavisno od akcije posetioca razmenjuje u oba smera podatke sa serverom. To čini jako dnamičnim internet strane.

I još nešto. JavaScript može manipulisati i odredjenim sadržajima, ali mora se znati da taj sadržaj nije vidljiv pretraživačima, što znači da nikad neće se naći u rezultatima pretrage. E sad sa druge strane ako je nekom i potrebno da se sadržaj sajta ne nadje u rezultatima pretrage onda može sav sadržaj sajta da ide preko JavaScript.

## Sintaksa

U ovom delu razmatraćemo najosnovnije stvari bez kojih je nemoguće bilo šta kreirati vezano za JavaScript. Tu se pre svega misli na JavaScript sintaksu, tipove promenljivih, operatore itd. Poželjno je ovaj deo dobro naučiti kako bi se kasnije lakše razumele kompleksnije stvari.

Primeri su kreirani tako da naj minimalnijim pisanjem dočaravaju najbolje što je moguće vezano za ono što bi trebalo da objasne. Preporučujemo da ni jedan primer ne preskačete. Preporučujemo takodje da svaki primer prepisete u notepad, sačuvate fajl kao HTML dokument i pokrenitega brauzerom, kako bi ste videli rezultat. Ovakvim načinom učenja stećićete rutinu, a i bolje će te zapamtiti sintaksu. Takodje je preporučljivo da sami izvršite razna eksperimentisanja sa primerima.

## Implementacija JavaScript u HTML

Postoje nekoliko načina implementacija, to jest umetanja JavaScript koda unutar HTML koda. Osnovni je pomoću SCRIPT taga, koji ima i početni i krajnji tag, ovako:

```
<script type="text/javascript">
<!--
    Ovde se smešta JavaScript kod
-->
</script>
```

Unutar **SCRIPT** taga koristi se atribut **TYPE** sa vrednošću **text/javascript** kako bi brauzer prepoznao da se radi o JavaScript kodu. Primećujete da smo JavaScript kod takodje umetnuli i izmedju HTML oznake za komentar. Naime, ako postoji neki brauzer koji nema ugrađenu podršku za JavaScript on će sam JavaScript kod prikazatii kao tekst što je efekat koji sigurno ne želimo. Da bi to izbegli, potrebno je sam skript staviti unutar HTML oznake za komentar “<!--” i “-->”.

Po nekad je Javascript kod jako veliki pa da ne opterećujemo HTML fajl možemo ga smestiti u poseban fajl sa ekstenzijom .js . Umetanje takvog JavaScript fajla u HTML kod vrši se ovako:

```
<script type="text/javascript" src="javascriptFajl.js"></script>
```

Kao što vidimo dodali smo atribut **SRC** koji za vrednost ima lokaciju JavaScript fajla. U samom JavaScript fajlu na početku ne stavljamo nikakve posebne tagove ili bilo kakve oznake, već normalno pišemo čist JavaScript kod. Ovi zasebni fajlovi su jako zgodni u slučajevima kad jednom napisan JavaScript kod možemo upotrebljavati u više HTML dokumenata. Obično se JavaScript fajlovi umeću u HEAD delu ali i ne mora.

JavaScript kod, kao što smo rekli, se umeće u HEAD delu HTML koda, i u tom slučaju taj kod se ne izvršava sve dok se ne pozove na izvršenje. U HEAD delu se najčešće smeštaju JavaScript funkcije koje mi pravimo. Takodje JavaScript kod možemo umetati i u BODY delu. Kada se umetne u BODY, onog trenutka kad brauzer naidje na SCRIPT tag u kome je JavaScript kod onda se taj kod i izvršava. Broj umetanja je neograničen, kako u BODY tako i u HEAD delu. Medjutim u HEAD nema potrebe za višestruko umetanje jer se JavaScript kod izvršava tek nakon poziva, što znači da komotno možemo sve JavaScript kodove smestiti u jedan SCRIPT tag.

Medjutim odredjene JavaScript naredbe mogu se naći u HTML kodu i izvan SCRIPT taga. Te naredbe se upotrebljavaju kao **HTML atributi**. Evo primera:

```
<html><head>
<title>Moj prvi JavaScript</title>
<script type="text/javascript" src="javascriptFajl.js"></script>
```

```
</head>
<body onload="nekaJavaScriptFunkcija()">
tekst tela dokumenta
</body></html>
```

ONLOAD je JavaScript naredba, a kao što vidimo unutar HTML koda se tretira kao atribut. Takvi JavaScript HTML atributi za vrednost mogu imati i neke naše kreirane funkcije u JavaScript. U konkretno ovom primeru je to naša `nekaJavaScriptFunkcija()` koja je definisana unutar fajla `javascriptFajl.js` a koji smo umetnuli u HEAD delu SCRIPT tagom. E sad naravno, ne mora samo BODY tag imati takve JavaScript attribute. Gotovo svi HTML tagovi mogu sadržati određene JavaScript naredbe kao svoje attribute. U ovom konkretnom slučaju ONLOAD atribut ima efekat da tek kad se ceo BODY tag realizuje (drugim rečima kad se učitava cela stranica) onda se aktivira vrednost atributa, to jest u konkretnom slučaju pokreće se na izvršenje naša funkcija.

## JavaScript promenljive

U JavaScript razlikujemo formalne, i nazovimo, neformalne tipove podataka. Formalni podaci su oni koje programer definiše u promenljive, a neformalni su oni koji se razmenjuju između JavaScript koda i Brauzera. Na primer ako postoji JavaScript naredba koja kontroliše klikanje na neko dugme, onda kad se klikne brauzer i JavaScript razmenjuju neformalne podatke o klikanju. Nećemo ulaziti u detalje neformalnih podataka ali eto čisto da se napomene.

Podaci koji nas kao JavaScript programere najviše zanimaju su naravno formalni JavaScript podaci. Formalni tipovi podataka su:

- **String**- niz znakova unutar navodnika
- **Brojevi**- bilo koji broj koji nije pod navodnicima
- **Boolean**- logičko istinito(true) ili neistinito(false)
- **Null**- lišeno vrednosti
- **Object**- sva svojstva i metodi koji pripadaju objektu ili nizu
- **Function**- definicija funkcije

Najpogodniji način za rad sa podacima u skriptu je dodeljivanje podataka promenljivima. Promenljiva se u JavaScript-u definiše metodom dodeljivanja. Na primer promenljivoj pod nazivom `brojUcenika` dodelimo vrednost 30:

```
brojUcenika = 30
```

Operator za dodeljivanje vrednosti je znak jednakosti. U JavaScriptu imena promenljivih se sastoje od slova engleske abecede (a-z, A-Z), cifara (0-9) i donje crte ( \_ ). Kao početni znak u nazivu promenljive može biti slovo i donja crta ali ne i broj. Za ime promenljive ne može se uzeti ni jedna JavaScript ključna reč. Promenljiva u nazivu ne može sadržati razmak. Ako ipak postoji potreba za imenom od više reči, postoje dve varijante združivanja reči u jednu. Prva je da se reči razdvoje donjom crtom (`broj_ucenika`), a u druga da početno slovo reči bude veliko slovo (`brojUcenika`). U promenljivama se razlikuju mala i velika slova pa su `brojKilograma` i `BrojKilograma` dve različite promenljive.



Promenljivoj koja je deklarirana unutar funkcije može se pristupiti samo unutar te funkcije. Kada se funkcija napusti, promenljiva prestaje da važi. Ovakve promenljive se nazivaju **lokalne** i može postojati više lokalnih promenljivih sa istim imenom u različitim funkcijama. Ako je promenljiva deklarirana van funkcije tada je slobodna za pristup svim funkcijama sa stranice, ali koje su ispod promenljive u kodu. Oblast važenja ovih promenljivih tzv. **globalnih**, počinje od mesta gde su deklarirane pa sve do kraja koda.

# Tipovi podataka

## Celi brojevi

Celi brojevi u JavaScript-u mogu biti predstavljeni u tri formata: u decimalnom (baza 10), u oktalnom (baza 8) i heksadecimalnom (baza 16). Decimalni celi brojevi se predstavljaju kao niz cifara (0-9) bez vodeće nule. Oktalni celi brojevi se predstavljaju kao niz cifara (0-7) predvođen sa nulom ("0"). Heksadecimalni celi brojevi se predstavljaju kao niz cifara (0-9) i slova (a-f ili A-F) predvođen sa nulom koju sledi slovo x ("0x" ili "0X"). Primer predstavljanja decimalnog celog broja deset (10) u tri brojna sistema je:

- decimalnom: 10.
- oktalnom: 012.
- heksadecimalnom: 0xA.

Broj 0 ispred oktalnih i oznaku 0x ispred heksalnih brojeva treba shvatiti kao oznake kako bi program razlikovao o kojim brojevima se radi.

## Brojevi u pokretnom zarezu

Brojevi u pokretnom zarezu imaju sledeće delove: decimalni ceo broj, decimalnu tačku (".") i deo iza decimalne tačke. A može se napisati i kao eksponent sa oznakama "e" ili "E", praćen decimalnim celim brojem. Primeri brojeva u pokretnom zarezu su sledeći:

- 3.1415
- .1E12
- -3.1E12
- 2E-10

## Boolean

Boolean tip podataka može imati samo dve vrednosti: true (tačno) i false (netačno).

## Stringovi

Stringovi su znakovni podaci smešteni između dvostrukih ili jednostrukih navodnika. Stringovi mogu biti tekstovi, brojevi, ili bilo koji znaci ali isključivo smešteni unutar navodnika. Evo par primera stringova:

- "string"
- "12cc34ccc"
- "PRVA LINIJA druga linija"

U JavaScript stringovima možete koristiti i sledeće specijalne tekst oznake:

- \b - pomeraj za jedno mesto ulevo (backspace),
- \f - pomeraj jedan red dole (form feed),
- \n - pomeraj na početak novog reda (new line),
- \r - povratak (carriage return),
- \t - tabulator (tab).

Takođe je često neophodno koristiti navodnike unutar samih navodnika koji definišu string. Korišćenje navodnika unutar stringa možete realizovati upotrebom **obrnute kose crte ispred navodnika**. Na primer:

```
navod = "<p>JavaScript je vrlo moćno <b>\\"oružje\\"</b>."
```

Primećujete da se kao string mogu koristiti i HTML kodovi !!!

## Operatori

### Aritmetički operatori:

Operator	Naziv	Primer	rezultat
+	sabiranje	2 + 2	4
-	oduzimanje	2 - 2	0
*	množenje	2 * 2	4
/	deljenje	5 / 2	2.5
%	moduo	5 % 2	1
++	uvećanje za 1 (inkrement)	x = 4; x++	5
--	umanjenje za 1 (dekrement)	x = 4; x--	3

### Operatori dodeljivanja:

Operator	Primer	Ekvivalentno
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

### Operatori poredjenja:

Operator	Opis
==	jednako
!=	nije jednako
>	veće

### Logički operatori:

Operator	Opis
&&	AND (logičko I)
	OR (logičko ILI)
!	NOT (logičko NE)

<	manje
>=	veće ili jednako
<=	manje ili jednako

## String operator

String operator spaja dva ili više stringa a znak je običan znak plus +:

```
t1 = "Dobar"
t2 = "dan!"
t3 = t1 + t2
```

Vrednost promenljive t3 sada je "Dobardan!" Ako hoćemo razmak izmedju reči onda pišemo:

```
t1 = "Dobar"
t2 = "dan!"
t3 = t1 + " " + t2
```

Ili da upišemo razmak u jedan od stringova:

```
t1 = "Dobar "
t2 = "dan!"
t3 = t1 + t2
```

## Uslovni operator

JavaScript ima i jedan uslovni operator (ternarni) koji dodeljuje vrednost promenljivoj na osnovu nekog uslova. Sintaksa je sledeća:

```
imePromenljive = (uslov) ? vrednost1 : vrednost2
```

Dakle, ako je uslov zadovoljen promenljivoj **imePromenljive** se dodeljuje vrednost **vrednost1**, a ako nije zadovoljen uslov onda **vrednost2**.

## Prioriteti operatora

Prioritet	Operator	Objašnjenje
1	()	male zagrade
	[]	vrednost indeksa niza
	function( )	bilo koji poziv funkcije
2	!	logičko NOT
	~	NOT nad bitovima
	-	znak za negativni broj
	++	povećanje za 1

	--	umanjenje za 1
	typeof	
	void	
	delete	brisanje elementa niza ili objekta
3	*	množenje
	/	deljenje
	%	modulo
4	+	sabiranje
	-	oduzimanje
5	<<	pomeranja u operacijama nad bitovima
	>	
	>>	
6	<	operatori poredjenja
	<=	
	<	
	>=	
7	==	jednako
	!=	nejednako
8	&	AND nad bitovima
9	^	XOR nad bitovima
10		OR nad bitovima
11	&&	logičko AND
12		logičko OR
13	?	uslovni operator
14	=	operatori dodele
	+=	
	-=	
	*=	
	/=	
	%=	
	<<=	
	>=	
	>>=	
	&=	
	^=	
	=	
15	,	zarez (za razdvajanje parametara)

Veći prioritet imaju oni operatori koji su pod manjim rednim brojem, a koji imaju isti redni broj, onda oni koji su pri vrhu.

# Nizovi

Videli smo kako se definišu promenljive. Medjutom u JavaScript se mogu koristiti i nizovi. Niz je promenljiva koja ima više vrednosti. Jednoj istoj promenljivoj možemo dodeliti više vrednosti u isto vreme tako što svaka vrednost ima svoj indeks. Na primer imamo promenljivu **brojevi** i ajde da toj promenljivoj dodelimo 5 različita broja kao vrednosti:

```
brojevi = Array(2, 4, 12, 43, 9)
```

Ili možemo ovako:

```
brojevi = Array()  
brojevi[0] = 2  
brojevi[1] = 4  
brojevi[2] = 12  
brojevi[3] = 43  
brojevi[4] = 9
```

Ili ovako:

```
brojevi = [2, 4, 12, 43, 9]
```

Ili ovako:

```
brojevi = []  
brojevi[0] = 2  
brojevi[1] = 4  
brojevi[2] = 12  
brojevi[3] = 43  
brojevi[4] = 9
```

Vidite da kad god želimo napraviti neki niz moramo upotrebiti rezervisanu reč **Array** zajedno sa malim zagradama. Pišite tako da uvek prvo slovo "A" bude veliko! E sad ukoliko znamo unapred vrednosti koje želimo dodeliti nizu onda te vrednosti i upisujemo direkto u zagradama, kao u prvom slučaju. Ali možemo i ostaviti praznu zagradu pa dole ispod u nekim izračunavanjima dodeliti vrednosti nizu, na primer kad neznamo unapred koje će se vrednosti dobijati u nekim izračunavanjima. Možemo malo skratiti pisanje ako umesto **Array()** pišemo samo srednje zagrade **[ ]** kao u primeru 3 i 4.

U daljim izračunavanjima možemo pojedinačno koristiti koji god želimo konkretnu vrednost niza. Na primer možemo izračunati i ovako:

```
s = brojevi[2] + brojevi[4]
```

Takodje pošto je niz promenljiva možemo u nastavku skripte i menjati vrednosti kako pojedinačnih vrednosti pojedinih elemenata niza tako i više njih, pa i svih ako se ukaže potreba.

Da rezimiramo, indeksi niza su brojevi **0, 1, 2, 3, 4** a vrednosti su redom **2, 4, 12, 43, 9**, a "**brojevi**" je naziv niza. U JavaScript-u indeksi mogu biti samo brojevi.

# Osnovne funkcije

JavaScript ima izuzetno mnogo ugradjenih funkcija. Sve te ugradjene funkcije imaju neku svrhu, pa tako i ove osnovne. Osnovne funkcije mogu manipulirati brojevima, tekstovima, uredjiviti brojeve, tekstove. Funkcije mogu izvršavati neke ugradjene matematičke radnje, ili kao rezultat da vraćaju vreme u minutima, časovima, danima, itd, itd. Spektar mogućnosti ugradjenih funkcija u JavaScript je izuzetno veliki a upravo to daje velike mogućnosti JavaScript-u. Malte ne sve do kraja ovog kursa mi će mo proučavati u velikoj meri uglavnom ugradjene funkcije, a u ovom delu proučimo one najosnovnije koje se i najmasovnije i najčešće i koriste.

## Opšte JavaScript funkcije

### eval( )

Ako je argument eval funkcije string koji predstavlja neki JavaScript izraz, eval će izvršiti izraz. Ako argument predstavlja jednu ili više JavaScript naredbi, eval će izvršiti te naredbe. Ova funkcija je takodje korisna za transformisanje stringa koji predstavlja numerički izraz u broj, itd. Evo primera i sve će biti jasno:

```
<html><body>
<script type=text/javascript>
a = "3 + 5"
document.write(eval(a))
</script>
</body></html>
```

Rezultat će biti:

8

### parseFloat( )

Ako je string broj ili strin počinje brojem, parseFloat taj broj prebacuje iz string u broj. Evo primer sa više varijacija:

```
<script type="text/javascript">
document.write(parseFloat("10") + ", ");
document.write(parseFloat("10.00") + ", ");
document.write(parseFloat("10.33") + ", ");
document.write(parseFloat("34 45 66") + ", ");
document.write(parseFloat(" 60 ") + ", ");
document.write(parseFloat("40 godina") + ", ");
document.write(parseFloat("Ona ima 40 godina"));
</script>
```

Rezultat:

10, 10, 10.33, 34, 60, 40, NaN

## escape( )

Escape vraća string koji sadrži ASCII kod karaktera koji nisu slova u obliku %xx, gde je xx numerički kod karaktera. Evo primera:

```
<script type="text/javascript">
document.write(escape("Sta volis? COKOLADU!"));
</script>
```

Rezultat:

Sta%20volis%3F%20COKOLADU%21

## unescape( );

Unescape je revizibilna (suprotna) od **escape( )**. Evo primer:

```
<script type="text/javascript">
document.write(unescape("Sta%20volis%3F%20COKOLADU%21"));
</script>
```

Rezultat:

Sta volis? COKOLADU!

## length

Length funkcija u stringu broji karaktere a za nizove broji koliko ima elemenata niza. Brojanje počinje sa 1. Evo primer:

```
<script type="text/javascript">
a = [1, 3, 7, 5, 4]
b = "Ana voli Milovana"
c = a.length
d = b.length
document.write(c + "<br>" + d)
</script>
```

Rezultat:

5  
17

# JavaScript funkcije nizova

## concat( )

concat( ) spaja dva ili više niza. Evo primer:

```
a = [2, 4, "Milan"]
b = [5, 2, 3]
c = [1, 7, 9]
spoj = a.concat(b, c)
```

Niz **spoj** će imati vrednosti: 2, 4, "Milan", 5, 2, 3, 1, 7, 9. A da smo primera radi napisali **b.concat(c, a)** vrednosti bi bile: 5, 2, 3, 1, 7, 9, 2, 4, "Milan". Mogu i više od 3 niza.

## join( )

join( ) funkcija sve elemente niza spaja u jedan string. Evo primera:

```
<html><body>
<script type="text/javascript">
voce = ["Banana", "Jabuka", "Narandza", "Mango"];
document.write(voce.join( ) + "<br>");
document.write(voce.join("+") + "<br>");
document.write(voce.join(" plus "));
</script>
</body></html>
```

Kao rezultat u brauzeru će se pojaviti sledeće:

Banana,Jabuka,Narandza,Mango  
Banana+Jabuka+Narandza+Mango  
Banana plus Jabuka plus Narandza plus Mango

Mislim da je ovde što se tiče same funkcije sve jasno. Medjutim da objasnimo **document.write**. Write je funkcija za ispisivanje na ekran onog što je u zagradi. Medjutim ako napišemo samo write neće se ispisati. Da bi se ispisalo moramo definisati u kom objektu želimo ispisivanje. Ako želimo u sklopu našeg HTML dokumenta onda moramo napisati **document** pa tačka pa onda **write**, i onda u zagradi ono što želimo ispisati.

## pop( )

pop( ) funkcija briše zadnji element u nizu a ujedno i vraća tu vrednost, tako da ako želimo možemo tu izbrisanu vrednost smestiti u neku novu promenljivu. Evo primera:

```
a = [1, 3, 6]
b = a.pop( )
document.write(a + "<br>" + b)
```

Rezultat će biti:

1,3  
6

## shift( )



shift( ) funkcija je ista kao i pop( ) samo što uklanja prvi element i vraća ga ako je potrebno u nekoj promenljivoj.

## push( )

push( ) funkcija dodaje nove elemente na kraj niza a ako nam je potrebno i vraća broj koji je jednak ukupnom broju elemenata niza sa novim elementima koji su dodati.

```
<script type="text/javascript">
voce = ["Banana", "Kivi", "Jabuka", "Mango"];
document.write(voce.push("Kruska") + "<br>");
document.write(voce.push("Limun", "Grozdje") + "<br>");
document.write(voce);
</script>
```

Rezultat:

5  
7

Banana,Kivi,Jabuka,Mango,Kruska,Limun,Grozdje

## unshift( )

unshift( ) funkcija je skoro ista kao i push( ) samo što dodaje nove elemente na početku niza. Naravno i on vraća po potrebi broj elemenata novokomponovanog niza.

## reverse( )

reverse( ) funkcija pravi obrnuti redosle elemenata u nizu. Onaj koji je prvi biće zadnji, drugi, biće predzadnji itd.

```
<script type="text/javascript">
voce = ["Banana", "Kivi", "Jabuka", "Mango"];
document.write(voce.reverse( ));
</script>
```

Rezultat:

Mango,Jabuka,Kivi,Banana

## slice( )

slice( ) funkcija odabira deo niza i vraća u novi niz. Prvi broj u zagradi je od kog broja elementa će se iseći niz a drugi broj je krajnji element. Broji se od 1.

```
<html><body>
<script type="text/javascript">
voce = ["Banana", "Jabuka", "Narandza", "Mango"];
document.write(voce.slice(1, 3 ) + "<br>");
```

```
document.write(voce.slice(1, 2) + "<br>");  
document.write(voce.slice(3));  
</script>  
</body></html>
```

Rezultat:

Jabuka,Narandza  
Jabuka  
Mango

## sort( )

sort( ) funkcija sortira elemente niza po abecednom redosledu, ili po brojčanom. Mislim da ovde ne treba primer ali vi možete da eksperimentišete.

## splice( )

splice( ) funkcija dodaje i uklanja elemente u nizu. Evo primera:

```
<script type="text/javascript">  
voce = ["Banana", "Narandza", "Jabuka", "Mango"];  
voce.splice(2, 0, "Limun");  
document.write(voce);  
</script>
```

Rezultat:

Banana,Narandza,Limun,Jabuka,Mango

# Tekstualne string funkcije

Ovde neću objašnjavati sve pojedinačno jer je lako shvatiti sve funkcije iz sledećeg primera (uz napomenu da ne rade baš sve funkcije u svim brauzerima):

```
<html><body>  
<script type="text/javascript">  
txt="Hello World!";  
document.write("<p>Big: " + txt.big() + "</p>");  
document.write("<p>Small: " + txt.small() + "</p>");  
document.write("<p>Bold: " + txt.bold() + "</p>");  
document.write("<p>Italic: " + txt.italics() + "</p>");  
document.write("<p>Blink: " + txt.blink() + "</p>");  
document.write("<p>Fixed: " + txt.fixed() + "</p>");  
document.write("<p>Strike: " + txt.strike() + "</p>");  
document.write("<p>Fontcolor: " + txt.fontcolor("Red") + "</p>");  
document.write("<p>Fontsize: " + txt.fontsize(16) + "</p>");  
document.write("<p>Subscript: " + txt.sub() + "</p>");
```

```
document.write("<p>Superscript: " + txt.sup() + "</p>");  
document.write("<p>Link: " + txt.link("http://mmance02.on.neobee.net") +  
"</p>");  
</script>  
</body></html>
```

Rezultat:

Big: Hello World!

Small: Hello World!

Bold: **Hello World!**

Italic: *Hello World!*

Blink: Hello World!

Fixed: Hello World!

Strike: ~~Hello World!~~

Fontcolor: Hello World!

Fontsize: **Hello World!**

Subscript: <sub>Hello World!</sub>

Superscript: <sup>Hello World!</sup>

Link: [Hello World!](http://mmance02.on.neobee.net)

## String funkcije

### charAt( )

charAt( ) funkcija vraća znak iz stringa u zavisnosti od položaja. Broji se od 0. Znači prvi znak u stringu je na nultom položaju.

```
<script type="text/javascript">  
str="Hello world!";  
document.write(str.charAt(1), str.charAt(4), str.charAt(6));  
</script>
```

Rezultat:

eow

## charCodeAt( )

charCodeAt( ) funkcija isto kao prethodna samo što vraća ASCII kod znaka.

## fromCharCode( )

fromCharCode( ) funkcija pretvara ASCII kodove u odgovarajuće znakove.

```
document.write(String.fromCharCode(72,69,76,76,79))
```

Rezultat:

HELLO

## indexOf( )

indexOf( ) funkcija vraća broj koji predstavlja poziciju traženog teksta u stringu. Broji se od 0. Ako ne nadje reč vraća -1. Obratite pažnju da je funkcija osetljiva na mala i velika slova.

```
<script type="text/javascript">
str = "Hello world!";
document.write(str.indexOf("Hello") + "<br>");
document.write(str.indexOf("WORLD") + "<br>");
document.write(str.indexOf("world"));
</script>
```

Rezultat:

0  
-1  
6

## slice( )

slice( ) funkcija seče delove stringa i takve ih vraća.

```
<script type="text/javascript">
str="Hello happy world!";
document.write(str.slice(0)+"<br>"); // ne seče ništa
document.write(str.slice(6)+"<br>"); // seče od znaka na poziciji 6
document.write(str.slice(-6)+"<br>"); // seče od pozicije 6 ali od pozadi
document.write(str.slice(0,1)+"<br>");
document.write(str.slice(6,11)+"<br>");
</script>
```

Rezultat:

Hello  
happy  
world!  
H  
happy

happy

world!  
world!

## split( )

split( ) funkcija deli string na osnovu definisanog znaka i dužine, i te delove smešta u niz.

```
<script type="text/javascript">
var str="How are you doing today?";
a=[]; b=[]; c=[]; d=[];
a=str.split();
b=str.split(" ");
c=str.split("");
d=str.split(" ",3);
document.write(a + "<br>" + b + "<br>" + c + "<br>" + d);
</script>
```

Rezultat:

How                      are                      you                      doing                      today?  
How,are,you,doing,today?  
H,o,w,                      ,a,r,e,                      ,y,o,u,                      ,d,o,i,n,g,                      ,t,o,d,a,y,?  
How,are,you

## substr( )

substr( ) funkcija izvlači odredjen broj znakova iz string.

```
<script type="text/javascript">
str="Hello world!";
document.write(str.substr(3)+"<br>");
document.write(str.substr(3, 4));
</script>
```

Rezultati

lo  
lo w

world!

## toLowerCase( )

toLowerCase( ) funkcija prebacuje sve znakove stringa u mala slova

## toUpperCase( )

toUpperCase( ) funkcija prebacuje sve znakove stringa u velika slova

# JavaScript regularni izrazi

Regularni izraz (engl. skr. regexp ili regex, u množini regexps, regexes ili regexen) je string koji opisuje, menja, sparuje, ili pretražuje skup stringova, u skladu sa određenim sintaksnim pravilima.

Regularni izrazi proširuju mogućnosti manipulacijom stringova i to pomoću javascript funkcijama kao što su:

- match( )
- replace( )
- search( )

U JavaScript -u string regularnog izraza ima dva dela:

```
regIzraz = /uzorak/modifikatori
```

Znači, između dve kose crte se piše konkretno uzorak regularnog izraza (koji se sastavlja iz tabela "Zagrade" "Metakaraktori" "Kvantifikatori") a modifikator je jedan ili više simbola (mogu i sva tri simbola biti zajedno samo ih pišete jedan pored drugi) iz tablice "Modifikatori".

## Modifikatori:

Modifikator	Opis
i	Ne pravi razliku između mala i velika slova
g	Proći kroz ceo string. A kad je bez " g " onda kad se nađje na prvo podudaranje ili pretragu onda se staje
m	Izvodi podudaranje svih redaka

## Zagrade

Izgled	Opis
[abc]	Pronaći bilo koji znak a, b, c
[^abc]	Pronaći bilo koji znak a da nije a, b, c
[0-9]	Pronaći brojeve od 0 do 9
[a-z]	pronaci sva mala slova
[A-Z]	pronaci sva velika slova
[a-Z]	pronaci sva slova
[belo plavo crno]	pronaci <b>ili</b> reč belo <b>ili</b> reč plavo <b>ili</b> reč crno

## Metakaraktori

Metakarakter	Opis
.	Tačka pronalazi bilo koji znak, ili zamenjuje bilo koji znak
\w	Pronalazi bilo koji word znak. Znači slova, brojevi i podvučeno.
\W	Pronalazi bilo koji znak osim word znakova.
\d	Pronalazi brojeve
\D	Pronalazi bilo koji znak osim brojeve
\s	Pronalazi razmake i tabulatore, ekvivalentno izrazu <code>[\t\n\r\f\v]</code>
\S	Pronalazi znakove a da nisu tabulatori i razmak
\b	Pronadji podudaranje na početku ili/i kraju reči
\B	Pronadji podudaranje ali da nije na početku i kraju reči
\0	Nadji null znak
\n	Pronadji znak novog reda
\f	Pronadji novi red
\r	pronadji povratak
\t	Pronadji tabulator
\v	Pronadji vertikalni tabulator
\xxx	Pronadji znak mimo oktalnog broja xxx
\xdd	Pronadji znak mimo heksadecimalnog broja dd
\uxxxx	Pronači Unicode znak naveden mimo heksadecimalnog broja xxxx

## Kvantifikatori

Kvatifikator	Opis
n+	Pronalazi jednom ili više puta prethodni znak. Na primer <b>bo+</b> će pronaći <b>bo</b> , i <b>boo</b> , ali neće i <b>b</b>
n*	Isto kao gore samo što pronalazi i nula puta, odnosno pronaći će i <b>b</b>
n?	Pronalazi na primer <b>b</b> i <b>bo</b> ali neće <b>boo</b>
n{X}	Pronalazi znak n tačno X puta. Na primer <b>o{2}</b> pronalazi samo <b>oo</b>
n{X,Y}	na primer <b>o{2,4}</b> pronalazi <b>oo</b> , <b>ooo</b> , i samo još <b>oooo</b>
n{X,}	Pronalazi minimum X puta. Na primer <b>o{3}</b> pronalazi <b>ooo</b> , <b>oooo</b> i tako dalje
n\$	Traži znak na kraju reči. Na primer <b>abc\$</b> podudariće se sa reči <b>pegfab</b> <b>c</b>
^n	Traži znak na početku reči. Na primer <b>^abc</b> podudariće se sa reči <b>abc</b> <b>defr</b>

?=n	Pronalazi bilo koji niz koji sledi specifičan niz <b>n</b>
?!=n	Pronalazi bilo koji niz koji ne sledi specifičan niz <b>n</b>

## match( )

match( ) funkcija vraća podudaranja ako ih nadje, na primer izmedju regularnog izraza i stringa koji se kontroliše. Ako ne nadje vraća nulu. Evo primera:

```
<script type="text/javascript">
str="Mali pali za palidrom";
document.write(str.match(/ali/gi));
</script>
```

Rezultat:

ali,ali,ali

## replace( )

replace( ) funkcija traži podudaranje izmedju stringova, ili regularnog izraza i stringa, i ako nadje vrši zamenu tog nadjenog sa novim stringom. Evo primera:

```
<script type="text/javascript">
str="Poseti danas Danicu.";
document.write(str.replace(/dan/gi, "nocas"));
</script>
```

Rezultat:

Poseti nocasas nocasicu.

## search( )

search( ) funkcija traži podudaranje između regularnih izraza i stringa, i vraća položaj podudaranja: Znakove broji od 0. A ako ne nadje vraća -1.

```
<script type="text/javascript">
str="Poseti danas Danicu";
document.write(str.search(/Dan/i));
</script>
```

Rezultat:

7

# Matematičke funkcije



# Funkcije za brojeve

**toFixed( )** - fiksira odredjeni broj decimalnih mesta

```
<script type="text/javascript">
num = 13.3714;
document.write(num.toFixed()+"<br>");
document.write(num.toFixed(1)+"<br>");
document.write(num.toFixed(3)+"<br>");
document.write(num.toFixed(10));
</script>
```

Rezultat:

13  
13.4  
13.371  
13.3714000000

**toPrecision( )** - kao gore samo obuhvata ceo broj

```
<script type="text/javascript">
var num = new Number(13.3714);
document.write(num.toPrecision()+"<br>");
document.write(num.toPrecision(1)+"<br>");
document.write(num.toPrecision(2)+"<br>");
document.write(num.toPrecision(3)+"<br>");
document.write(num.toPrecision(10));
</script>
```

Rezultat:

13.3714  
1e+1  
13  
13.4  
13.37140000

**toString( )** - Prebacuje broj u string

## Konstante

- **E** - Vraća Eulerov broj (approx. 2.718)
- **LN2** - Vraća prirodan logaritam od broja 2 (approx. 0.693)
- **LN10** - Vraća prirodan logaritam od broja 10 (approx. 2.302)
- **LOG2E** - Vraća 1.442
- **LOG10E** - Vraća 0.434

- **PI** - Vraća PI (approx. 3.14159)
- **SQRT1\_2** - Vraća kvadratni koren od 1/2 (approx. 0.707)
- **SQRT2** - Vraća kvadratni koren od 2 (approx. 1.414)

## Matematičke funkcije

**abs( )** - Apsolutna vrednost broja. Evo primera

```
<script type="text/javascript">
document.write(Math.abs(7.25) + "<br>");
document.write(Math.abs(-7.25) + "<br>");
document.write(Math.abs(null) + "<br>");
document.write(Math.abs("Hello") + "<br>");
document.write(Math.abs(7.25-10));
</script>
```

Rezultat:

7.25  
7.25  
0  
NaN  
2.75

Znači piše se **Math** pa tačka pa onda funkcija odgovarajuća. Tako se piše i za ostale matematičke funkcije koje ću dati samo kao spisak.

- **acos(x)** - Vraća arkus sinus od x, u radijanima
- **asin(x)** - Vraća arkus kosinus od x, u radijanima
- **atan(x)** - Vraća arkus tangens od x, u radijanima
- **atan2(y,x)** - Vraća arkus tangens od količnika argumenata
- **ceil(x)** - Vraća veći najbliži celi broj
- **cos(x)** - Vraća kosinus od x u radijanima
- **exp(x)** - Vraća vrednost eksponenta od x
- **floor(x)** - Vraća zaokruženi najniži celi broj
- **log(x)** - Vraća prirodni logaritam broja x
- **max(x,y,z,...,n)** - Vraća najveći broj
- **min(x,y,z,...,n)** - Vraća najmanji broj.
- **pow(x,y)** - Vraća broj koji se dobije kad se x stepenuje na y
- **random( )** - Vraća slučajan decimalni broj iz opsega od 0 do 1

- **round(x)** - Vraća celi broj
- **sin(x)** - Vraća sinus od x
- **sqrt(x)** - Vraća kvadratni koren od x
- **tan(x)** - Vraća tangens od x

## • Datum funkcije

- **getDate( )** - Daje tekući dan u mesecu u broju.

```
<script type="text/javascript">
d = new Date()
danUMesecu = d.getDate();
document.write(danUMesecu);
</script>
```

- Rezultat:

- 6

- Obratite pažnju da smo prvo napravili naš objekat **d** uz pomoć **new Date( )** pa tek onda izvukli informaciju iz **getDate( )** i to tako što smo napisali **d.getDate( )**. Za ostale vremenske funkcije neću više nadalje sve ovako pisati ali znajte da se sve tako radi. Sad ću dati spisak ostalih datumskih funkcija koje su interesantne nama:

Funkcija	Opis
getDay()	Vraća tekući dan u nedelji (opseg 0-6)
getFullYear()	Vraća godinu u formatu 4 broja
getHours()	Vraća sat (0-23)
getMilliseconds()	Vraća milisekundu (0-999)
getMinutes()	Vraća minutu (0-59)
getMonth()	Vraća mesec (0-11)
getSeconds()	sekundu (0-59)
getTime()	Vraća broj milisekundi koji je protekao od ponoći 1. januara, 1970 do sadašnjeg trenutka
getTimezoneOffset()	Vraća

	vremensku razliku između GMT i lokalnog vremena, u minutama
getUTCDate()	Prikazuje dan u mesecu, prema univerzalnom vremenu (1-31)
getUTCDay()	Univerzalno, dan u nedelji (0-6)
getUTCFullYear()	Univerzalna godina u formatu 4 broja
getUTCHours()	Univerzalni časovi (0-23)
getUTCMilliseconds()	Univerzalni milisekundi (0-999)
getUTCMinutes()	Univerzalni minuti (0-59)
getUTCMonth()	Univerzalni mesec (0-11)
getUTCSeconds()	Univerzalne sekunde (0-59)
getYear()	Godina u formatu 4 cifre
toLocaleDateString()	Vraća lokalni string na primer: 16. decembar 2009
toLocaleTimeString()	Vraća sat u formatu: 22:57:22
toLocaleString()	Puno vreme i datum lokalno u formatu: 16. decembar 2009 22:58:05
toTimeString()	U formatu: 22:59:27

	UTC+0100
toUTCString()	U formatu: Wed, 16 Dec 2009 21:59:57 UTC
toString()	Kompletno u formatu: Wed Dec 16 22:58:44 UTC+0100 2009

## Kontrolne strukture

Kontrolne strukture odnose se na upravljanje tokom izvođenja programa. Strukturno programiranje temelji se na tri osnovne kontrolne strukture:

- sekvenca (sled, programski blok),
- selekcija (grananje, odabir) i
- iteracija (petlja).

Nestrukturno programiranje pridodaje još i skokove. Tipična instrukcija skoka je **break** i **continue**.

Sekvenca, poznatija kao programski blok, predstavlja niz instrukcija obrade koje se izvode uzastopno. U JavaScriptu početak sekvence označava se levom vitičastom zagradom { ,a kraj sekvence desnom vitičastom zagradom }. Evo primer:

```
{
  // Telo bloka u kome se smeštaju JavaScript izrazi (instrukcije)
}
```

Svaka leva vitičasta zagrada ima odgovarajuću desnu, odnosno svaki programski blok mora biti zatvoren. Programski blok (sekvenca) može da se nadje u petljama, kontrolnim strukturama kao što su IF - else, funkcije itd. Specijalni slučaj programskog bloka je onaj kada se on sastoji od samo jedne instrukcije obrade. Tad se ne moraju koristiti vitičaste zagrade za označavanje početka i kraja sekvence.

## JavaScript IF, IF ELSE i SWITCH grananja

### If

Najjednostavnija odluka u programu jeste praćenje neke grane ili putanje programa ako je ispunjen određen uslov. Formalna sintaksa za ovu konstrukciju je:

```
if (uslov) {
  // kod koji se izvršava ako je uslov true
}
```

Ako se vrši grananje u oba slučaja, i kad je uslov ispunjen a i kad nije, koristi se onda **if else** konstrukcija.

```
if (uslov) {  
    //kod koji se izvršava ako je uslov true  
} else {  
    //kod koji se izvršava ako je uslov false  
}
```

A možemo pratiti i nekoliko uslova.

```
if (uslov1) {  
    //kod koji se izvršava ako je uslov1 true  
} else if (uslov2) {  
    //kod koji se izvršava ako je uslov2 true  
} else {  
    //kod koji se izvršava ako uslov1 i uslov2 nemaju vrednsot true  
}
```

Evo konkretan primer. Za veće kupovine daje se popust:

- Za manje od 5 kupljenih artikla nema popusta,
- 5 - 10 - popust 10%,
- 11 - 15 - popust 12%,
- 16 i više - popust 15%

Evo kako bi izgledao kod:

```
if (kolicina < 5) procenat = 0;  
else if (kolicina > 4 && kolicina < 11) procenat = 10;  
else if (kolicina > 10 && kolicina < 16) procenat = 12;  
else (kolicina > 15) procenat = 15;
```

Ovde uslovi jedan drugi isključuje pa će se izvršiti samo jedan, ali ako se uslovi preklapaju biće izvršen onaj na koga se prvo naidje a ostali će biti preskočeni.

## SWITCH

Takodje i **SWITCH** odlučuje izmedju više opcija. Na primer, napravimo program koji će brojeve od 1 do 3 imenovati, a ako je u pitanju neki drugi broj izvan skupa od 1 do 3 da ispiše "vrednost promenljive nije iz skupa jedan do tri".

```
switch (a) {  
    case 1:  
        document.write("jedan");  
        break;  
    case 2:  
        document.write("dva");  
        break;  
    case 3:  
        document.write("tri");  
}
```

```
break;
default:
    document.write("vrednost promenljive nije iz skupa od jedan do tri");
break;
}
```

Umesto na primer "case 1:" možemo pisati i neki matematički izraz ako je neophodno, pa tako bi bilo na primer:

```
case (b * 25 + 4);
```

Isto može i neki ovakav program da se pojavi:

```
switch (x) {
    case (y * 4):
    case (9 * 3):
        document.write("Zapamti");
        break;
    default:
        document.write("Zaboravi");
}
```

Dakle, mogu se pojaviti više CASE naredbe sa jednom BREAK naredbom. Naredba DEFAULT obezbeđuje nastavak po putanji izvršavanja kada vrednost izraza ne odgovara ni jednoj oznaci naredbe CASE. Naredba BREAK, koja služi za izlazak iz petlje, ovde ima značajnu ulogu. Naime, ako nije navedeno break posle svake grupe naredbi u case granama, izvršiće se sve naredbe iz svake case grane bez obzira na to da li je nađena odgovarajuća oznaka.

# JavaScript FOR, WHILE i DO WHILE petlje

## For petlja

For petlja omogućava višestruko prolaženje kroz jedan te isti programski blok.

```
for (pocetniIzraz; uslov; korak) {
    // naredbe
}
```

Tri parametra unutar zagrada petlje **for** igraju ključnu ulogu. U petlji **pocetniIzraz** izvršava se samo jednom kada se pokreće petlja. Najčešća primena početnog izraza je dodela imena i početne vrednosti promenljivoj brojača petlje. Izraz **uslov** definiše do koje vrednosti brojač petlje treba da ide pre nego što se kruženje završi. Poslednja izraz **korak** nakon svakog izvršenja petlje uvećava ili smanjuje za određeni korak brojač.

Evo na primer napišimo kod koji ispisuje brojeve od 0 do 10, svaki u posebnom redu.

```
<html><body>
<script language="javascript">
for (var i=0; i<=10; i++) {
    document.write(i + "<br>")
}
```

```
}  
</script></body></html>
```

## while petlja

While petlja izvršava blok koda sve dok je uslov ispunjen. While petlja se koristi kada neznamo unapred broj iteracija. Kada je poznat broj iteracija obično se koristi for petlja jer je brža. Kao primer ispišimo brojeve od 10 do 20:

```
a=10;  
While (a < 21) {  
    document.write(a + "<br>");  
    a++;  
}
```

## Do while

JavaScript nudi još jednu konstrukciju petlje zvanu **do while**. Formalna sintaksa za ovu konstrukciju je sledeća:

```
do {  
    //naredbe  
}  
while (uslov)
```

Razlika između while i do.. while petlje je ta što se u do.. while petlji naredbe izvršavaju bar jednom pre nego što se uslov ispita, dok u petlji while to nije slučaj.

Po nekad petlje nemaju potrebu da nastave kruženje kroz ostatak vrednosti u opsegu brojača petlje. Tad se za izlazak iz petlje koristi naredba **break**. Ona govori JavaScript-u da izađe iz petlje, i izvršavanje skripta se nastavlja odmah iza bloka petlje.

## continue

Naredba **continue** primorava skript da, ako je uslov ispunjen, pređe na sledeći korak tj. poveća vrednost brojača i nastavi petlju od te vrednosti.

```
for(i = 1; i <= 20; i++){  
    if (i == 13) {  
        continue;  
        naredbe  
    ...  
}
```

U slučaju ugnežđenih for petlji, naredba continue utiče na petlju for u čiji trenutni opseg spada i konstrukcija if.

## Sopstvene funkcije



Funkcije koje mi kreiramo predstavljaju jednu od veoma bitnih orudja JavaScript-a. Jednostavno rečeno, funkcija je imenovani blok koda koga sami pravimo jednom a po potrebi upotrebljavamo više puta. Ako se deo nekog koda, ili neko izračunavanje stalno i često ponavlja, onda se taj deo koda može izdvojiti u funkciju. To će vam omogućiti da funkciju koju sami napišete koristite isto kao i neku ugrađenu funkciju, tako što je pozivate neograničen broj puta dodeljivanjem neophodnih parametara. Možete kreirati proizvoljan broj funkcija u programu. Funkcija može vraćati vrednost, bilo promenljivu, objekt, ili da izvrši neku akciju.

Funkcija može smestiti bilo gde u okviru jednog HTML dokumenta, ali obično sve funkcije se grupišu u jedan SCRIPT tag u HEAD tag. Ali možemo ih smestiti u poseban fajl na primer **funkcije.js** i umetnuti ga u HTML dokument.

Unutar funkcije je moguće pozvati druge funkcije definisane unutar istog HTML dokumenta.

Izradu sopstvene funkcije započinjemo rezervisanom rečju **function**. Zatim po našem izboru sledi ime funkcije. Slede u zagradi parametri odvojeni zarezom. I na kraju blok naredbi koji će biti izvršen svaki put kad se funkcija pozove. To izgleda ovako:

```
function formula (parametar1, parametar2, ..., parametarN, "stringN", 10,
...) {
    //telo funkcije
}
```

U telu funkcije inače možete koristite sve vezano za JavaScript što smo do sad naučili i šta će mo naučiti.

U skriptama se ova naša funkcija iz gornjeg primera poziva više puta kad je potrebno na sledeći način:

```
formula (promenljiva1, masa, ..., tezina, "ana voli milovana", 110, ...);
```

Evo na primer nešto konkretno:

```
function stampaj(string){
    document.write(string)
}
```

Funkcija **stampaj ( )** ima argument promenljivu **string**. Unutar tela funkcije možemo sa tom promenljivom raditi šta god želimo. Funkciji se mogu prosledivati promenljive, konstante, ali i obični stringovi u navodnicima ili i obični brojevi. Ukoliko se promenljiva prosledi funkciji, promena njene vrednosti unutar funkcije ne menja njenu vrednost izvan funkcije. Parametri postoje samo za života funkcije. Ukoliko zovete funkciju više puta, svaki put kada zovete funkcije imaćete "nove" parametre, odnosno vrednost koji će parametri imati na kraju prethodnog poziva funkcije neće biti sačuvana.

Na primer, funkciju **stampaj ( )** možemo pozvati sa bilo kojim parametrom:

```
ime = "Tanja";
stampaj (ime);
```

Kada funkcija počne da se izvršava, promenljiva string u funkciji će imati vrednost "Tanja". Isto funkciju **stampaj ( )** možemo pozvati sa konstantom:

```
stampaj ("Tanja");
```

Kada funkcija počne da se izvršava, promenljiva string u funkciji će takođe imati vrednost "Tanja".

Pozivanje JavaScript funkcije se može obaviti na više načina.

1. Iz druge funkcije ili JavaScript programa:

```
<SCRIPT LANGUAGE="JavaScript">
    stampaj("Ovo je tekst koji će se prikazati.")
</SCRIPT>
```

2. Kao reakciju na događaj:

```
<body OnLoad="stampaj(\"Ovo je tekst koji će se prikazati.\")">
```

3. Izborom linka vezanog za događaj:

Kao što je prethodno pomenuto, funkcija može vraćati i rezultate. Rezultat se iz funkcije vraća korišćenjem naredbe **return**. Na primer:

```
function kub(broj) {
    kub = broj * broj * broj;
    return kub;
}
```

naredba **return** vraća vrednost promenljive **kub**. Znači ne vraća promenljivu nego samo njenu VREDNOST !!!! Istu funkciju možemo napisati i na sledeći način, preko vraćanja izraza:

```
function kub(broj) {
    return broj * broj * broj;
}
```

Izraz **broj \* broj \* broj** će se izračunati u jednu vrednost, i ta vrednost se vraća iz funkcije.

Funkcije mogu biti rekurzivne, odnosno mogu pozivati same sebe. Rekurzija funkcija je veoma moćna stvar, ali može biti i veoma opasna. Duboke rekurzije, čak iako nisu beskonačne, mogu prouzrokovati pucanje WWW čitača.

U funkcijama se otvara pitanje o dubini važenja promenljivih. Ako smo promenljivu definisali u funkciju, oblast važnosti promenljive je ta funkcija. Ako je promenljiva definisana van funkcije i to iznad te funkcije, oblast važenja te promenljive je u celoj skripti uključujući i u funkciji. Takva promenljiva je globalna promenljiva. Ako definišemo lokalnu promenljivu unutar funkcije sa istim imenom koju već ima neka globalna promenljiva, u toj funkciji će se kreirati nova verzija promenljive čija je oblast važenja funkcija, dok globalna promenljiva sa istim imenom neće promeniti vrednost.

## Dogadjaji

Kad otvorite neku stranicu u brauzeru vi možete klikovati, linkovati, pisati u raznim formama, i slično. Sve te akcije su u stvari događaji na osnovu kojih u HTML dokumentu JavaScript može aktivirati dalje neka dešavanja na sajtu.

Obradivači događaja su predstavljeni kao specijalni atributi koji modifikuju ponašanje HTML elemenata u okviru kojih se nalaze. Na primer:

```
<body OnLoad=alert("Dobrodošli!")>
```

U gornjem primeru, u body tagu dodat je JavaScript atribut **OnLoad**, koji obrađuje događaj učitavanja HTML dokumenta. U konkretnom slučaju, kada se učitavanje HTML dokumenta završi, izvršiće se naredba JavaScripta dodeljena atributu OnLoad. Kao rezultat u iskaćućem prozorčetu ispisaće se poruka "Dobrodošli".

OnLoad se zove u stvari OBRADJIVAČ DOGADJAJA. Postoje veći broj JavaScript obradivača događaja.

## Obradivači događaja

Daću sve obradivače događaja tabelarno sa objašnjenjima, i mislim da će značenje biti jasno. A kasnije ćemo većinu obradivača događaja ponaosob potkrepiti primerima.

Događaj	Opis	Obradivač događaja
blur	Kada pomeramo fokus sa elementa	onBlur
click	Klik miša na objekat	onClick
dblclick	Dupli klik miša na objekat	onDbClick
change	Događa se kada korisnik promeni vrednost u text, textarea ili select elementima forme.	onChange
focus	Događa se kada element forme dobije ulazni fokus.	onFocus
load	Događa se nakon učitavanja stranice.	onLoad
mouseover	Događa se kada korisnik pomera miša preko objekta	onMouseOver
mouseout	Događa se kada miš pređe preko objekta.	onMouseOut
select	Događa se kada korisnik izabere ulazno polje forme.	onSelect
submit	Događa se kad je forma prosleđena HTTP serveru (kada korisnik klikne na dugme za slanje sadržaja forme).	onSubmit
abort	Događaj koji se aktivira pri prekidu učitavanja slike	onAbort
error	Događaj koji se aktivira kada dodje do greske pri učitavanju slike	onError
keydown	Događaj koji se aktivira kada se pritisne taster	onKeyDown
keypress	Događaj koji se aktivira kada se pritisne taster ili otpusti	onKeyPress
keyup	Događaj koji se aktivira kada se taster otpusti	onKeyUp
mousedown	Događaj koji se aktivira kada se pritisne taster misa	onMouseDown
mousemove	Događaj koji se aktivira kada se pomeri mis	onMouseMove
mouseup	Događaj koji se aktivira kada otpusti taster misa	onMouseUp
reset	Događaj koji se aktivira kada se pritisne taster Reset	onReset
resize	Događaj koji se aktivira kada se promeni velicina prozora	onResize
unload	Događa se kada korisnik napušta HTML dokument, pre nego što počne	onUnload

sa učitavanjem novog HTML dokumenta.

Događaje vezujemo za sledeće HTML elemente:

HTML element	Određivači događaja koji mogu da se primene
Lista za izbor	onBlur, onChange, onFocus
Text element	onBlur, onChange, onFocus, onSelect
Textarea element	onBlur, onChange, onFocus, onSelect
Button element	onClick
Checkbox	onClick
Radio Button	onClick
Hypertext Link	onClick, onMouseOver
Reset Button	onClick
Submit Button	onClick
Document	onLoad, onUnload
Window	onLoad, onUnload
Form	onSubmit

Dosta objekata ima metode koji simuliraju događaje. To može biti posebno korisno u nekim slučajevima ako želimo da simuliramo neke akcije korisnika. U JavaScriptu su raspoloživi sledeći simulatori događaja koje će mo kasnije detaljnije proučiti:

- blur( )
- click( )
- focus( )
- select( )
- submit( )

## OnLoad

OnLoad obradivač događaja se aktivira pošto se HTML dokument (i svi njegovi delovi) kompletno učitava. Dodeljuje se uglavnom elementima <BODY> i <FRAMESET>. Ako je nekad bitno kada će se i u kom roku učitati neka stranica ako koristite OnLoad, možete biti sigurni da su svi ostali okviri učitani.

Evo jedan konkretan primer upotrebe OnLoad

```
<html><head>
<script type=text/javascript>
<!--
function caoo(){
    window.alert("Dobrodošli u svet JavaScripta!");
}
```

```
// -->
</script></head>
<body onLoad="caoo()">
<h3>OnLoad</h3><hr>
</body></html>
```

## OnUnload

Obradivač događaja **OnUnload** je koristan za "čišćenje" posle posete nekoj prezentaciji. Tokom posete nekoj prezentaciji, korisniku se može desiti da ima otvoreno nekoliko prozora kao rezultat rada programa u JavaScriptu. Njih treba zatvoriti, a to se najlakše obalja pomoću OnUnload.

```
<html><head>
<script type=text/javascript>
function dovidjenja(){
    window.alert("Hvala što ste svratili.");
}
</script></head>
<body OnUnload="dovidjenja()">
<h3>OnUnload</h3>
<p>Kada napustite ovu stranicu dobićete pozdravnu poruku.</p>
</body></html>
```

## OnClick

Najčešće korišćeni obradivač događaja uzrokovan akcijom miša je OnClick. Objekti koji prihvataju OnClick događaj su linkovi, check box-ovi i dugmad (uključujući submit, reset i radio buttons).

U sledećem primeru daćemo jednostavnu formu koja odgovara na klik taj komponenti forme.

```
<html><head>
<script type=text/javascript>
function kliktanje(){
    window.alert("Aktiviran je onClick događaj!");
}
</script></head><body>
<a href="javascript:kliktanje()">Standardni link</a>
<form>
<input type="RADIO" OnClick="kliktanje()">RadioButton<br>
<input type="CHECKBOX" OnClick="kliktanje()">CheckBox<br>
<input type="BUTTON" OnClick="kliktanje()" value="Standardno dugme"><br>
<input type="RESET" OnClick="kliktanje()"><br>
<input type="SUBMIT" OnClick="kliktanje()"><br>
</form></body></html>
```

## OnFocus

Događaj Focus se pojavljuje kada objekat postane predmet u fokusu mišem. To se događa kada korisnik klikne mišem na određeni objekat, ili tamo dođe pritiskajući taster tab. Ako korisnik može da unese podatke u objekat (ili da promeni trenutni izbor za slučaj lista izbora), tada objekat ima fokus.

Obrađivač događaja OnFocus se koristi samo sa objektima text, textarea, password i select. U sledećem primeru prikazujemo upotrebu OnFocus sa jednostavnom formom koju čine ulazna polja Ime i Prezime. OnFocus je povezan sa poljem Prezime i prikazuje prozor sa porukom kada korisnik uđe u to polje.

```
<html><head>
<script type=text/javascript>
function fokus(){
window.alert("\"Prezime\" je u fokusu.");
}
</script></head><body>
<form>
Ime: <input type="TEXT"><br>
Prezime: <input type="TEXT" OnFocus="fokus()">
</form></body></html>
```

## OnBlur

Događaj Blur se pojavljuje kada objekt nije više u fokusu. To se može desiti sa prebacivanjem u drugi prozor ili aplikaciju, ili tako što kliknemom mišem na drugi objekt, ili što pomerimo fokus pritiskom na taster tab. Obrađivač događaja OnBlur se koristi samo sa objektima text, textarea, password i select, baš kao i OnFocus.

```
<html><head>
<script language="JavaScript">
function dodaj(){
    a = document.forma.ime.value
    b = document.forma.prezime.value
    document.forma.puno.value = (a + " " + b)
}
</script></head><body>
<form name="forma">
Ime: <input name="ime" type="TEXT" OnBlur="dodaj()"><br>
Prezime: <input name="prezime" type="TEXT" OnBlur="dodaj()"><br><hr>
Puno ime: <input name="puno" type="TEXT">
</form></body></html>
```

Nakon unosa i imena i prezimena kliknite gde god želite u prostoru brauzera da bi se kompletirao sadržaj polja "Puno ime".

## OnChange

Događaj change se aktivira uvek kada objekat izgubi fokus i ako se njegova vrednost promenila. Obrađivač događaja OnChange se koristi samo sa objektima text, textarea, password i select. OnChange može biti veoma koristan kada želite izračunati sumu unetih vrednosti na primer.

```
<html><head>
<script type=text/javascript>
function kalkulacija(){
    a = document.imena.ime1.value
    b = document.imena.ime2.value
    c = document.imena.ime3.value
    document.imena.total.value = a + " " + b + " " + c
}
```

```

}
</script></head><body>
<form name="imena">
Ime: <input name="ime1" type="TEXT" OnChange="kalkulacija()"><br>
Srednje ime: <input name="ime2" type="TEXT" OnChange="kalkulacija()"><br>
Prezime: <input name="ime3" type="TEXT" OnChange="kalkulacija()"> <br><hr>
Total: <input name="total" type="TEXT">
</form></body></html>

```

Nakon unosa svega kliknite gde god želite u prostoru brauzera da bi se kompletirao sadržaj polja "Total".

## OnSelect

Obrađivač događaja OnSelect se odnosi samo na text, textarea i password objekte i izvršava se kada korisnik označi deo teksta u jednom od nabrojanih tipova objekata. Evo i kratkog primera.

```

<html><head>
<script type="text/javascript">
function dodaj(){
    window.alert("selektovao si tekst")
}
</script></head><body>
<form>
<textarea rows="2" cols="20" OnSelect="dodaj()">aaaaaaaaaaaaaaaa</textarea>
</form></body></html>

```

Selektujte malo tekst i pojaviće se reakcija.

## OnSubmit

Obrađivač događaja OnSubmit je povezan sa objektom forme. Najčešće je potrebno da se podaci uneti u polja forme, provere ili iskoriste pre prosleđivanja sadržaja forme HTTP serveru. Na taj način, moguće je smanjiti opterećenje samog HTTP servera, pošto se provera grešaka vrši na strani klijenta, te je redukovano broj izveštaja o grešci koje vraća server. OnSubmit omogućava zaustavljanje prosleđivanja sadržaja forme serveru ako je to neophodno.

```

<html><head>
<script type="text/javascript">
function prover() {
    if (document.forma.num1.value.length < 11) {
        window.alert("Hvala što ste uneli samo 10 karaktera.");
        return true;
    } else {
        window.alert("Unesite samo 10 karaktera, molim Vas.");
        return false;
    }
}
</script>
</head><body>
<form name="forma" OnSubmit="return prover(this)">
Unesite do deset karaktera: <input name="num1" type="TEXT"

```

```
size="10,1"><br><hr>
<INPUT TYPE=SUBMIT value="Pošalji podatke!">
</form></body></html>
```

U ovom primeru proveravamo samo osnovne vrednosti, dužinu unetih podataka. Koristeći string metode u kombinaciji sa regularnim izrazima možemo uraditi mnogo složenije provere sadržaja polja forme pre njenog slanja. Ako unete vrednosti u polje ne bi zadovoljili proveru, sadržaj polja neće biti prosleđen HTTP serveru. Primetite da je u primeru korišćena naredba **this**. Ona se odnosi na tekući objekat i biće detaljnije objašnjena u nekom od narednih poglavlja.

## OnMouseOver

Kod OnMouseOver obradivača događaja je dovoljno da pointer miša bude iznad nekog objekta pa da se događaj aktivira. OnMouseOver se može upotrebiti za objašnjenje linkova ili dugmadi. U primeru OnMouseOver menja tekst u prozoru dobijenom primenom metoda alert().

```
<html><head>
<script type=text/javascript>
function objasni(){
window.alert("Presli ste preko Zdravo kakote");
}
function objasni2(){
window.alert("Presli ste preko dobrosmo");
}
</script>
</head><body>
<a href="#" OnMouseOver="objasni()">Zdravo kakote :</a><br><br>
<a href="#" OnMouseOver="objasni2()">Dobrosmo ;</a>
</body></html>
```

Zavisno od toga preko kog linka pređemo mišem, prozor upozorenja će prikazati drugačiji podatak.

## OnMouseOut

U ovom slučaju kada pointer miša izađe iz područja nekog objekta, događaj je okinut. OnMouseOut se često koristi u sprezi sa događajem OnMouseOver.

```
<html><head>
<script type=text/javascript>
    function objasni(a){
        document.formica.limit.value = a
    }
</script>
</head><body>
<form name="formica">
<input type="button" value="Dugme 1" OnMouseOver="objasni('presio si na dugme 1') " OnMouseOut="objasni('otiso si od dugmeta 1') ">
<input type="button" value="Dugme 2" OnMouseOver="objasni('presio na dugme 2') " OnMouseOut="objasni('otiso si od dugmeta 2') ">
<input type="text" name=limit size=25>
</form></body></html>
```



# OnKeyUp

KeyUp je događaj koji se aktivira kada se taster otpusti. Tako možemo praviti limiter teksta u textbox-u:

```
<html><head>
<script type=text/javascript>
count = "100";
function limiter(){
    tex = document.myform.comment.value;
    len = tex.length;
    if (len > count){
        tex = tex.substring(0,count);
        document.myform.comment.value =tex;
        return false;
    }
    document.myform.limit.value = count - len;
}
</script></head><body>
<form name="myform">
<textarea name=comment rows=3 cols=40 onkeyup=limiter()></textarea><br>
<input type=text name=limit size=4>
</form></body></html>
```

Pišite nešto u textbox -u i videćete kako se smanjuje broj preostalih karaktera za ispis. Kad dodje do 100 karaktera više se ne može pisati.

# OnMouseMove

OnMouseMove reaguje na svako pomeranje miša. Evo primer koji sve dočarava:

```
<html><head>
<script type="text/javascript">
function misPomeraj(event){
    x=event.screenX;
    y=event.screenY;
    document.getElementById("txt1").value=x;
    document.getElementById("txt2").value=y;
}
</script>
</head>
<body onMouseMove="misPomeraj(event)">
<form>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br>
X koordinata kursora: <input type="text" size=10 id="txt1"><br>
Y koordinata kursora: <input type="text" size=10 id="txt2">
</form></body></html>
```

Mislim da je sve jasno što se tiče onMouseMove. Ali u primeru figuriše i EVENT. EVENT je HTML DOM objekat i u tom delu je objašnjen.

Ostalih 8 obradivača događaja možete i samo testirati na slične načine, uz malo mašte.

# Browser funkcije

Document Object Model ( DOM ) je plan organizacije objekata na web stranicama. Svaki objekat je jedinstven. Tri najvažnije odlike objekata su: metod (u HTML je to ime taga), svojstva (u HTML su to atributi) i osobine (u HTML su to vrednosti atributa). JavaScript DOM ima sledeću hijerarhiju:

## 1. Čisto JavaScript objekti

- Nizovi
- Date
- Math
- Brojčani
- String
- Regularni izrazi

## 2. Browser objekti

- Window
- Navigator
- Screen
- History
- Location

## 3. HTML DOM objekti

- Document
- Anchor
- Area
- Base
- Body
- Button
- Event
- Form
- Frame
- Frameset
- IFrame

- Image
- Input Button
- Input Checkbox
- Input File
- Input Hidden
- Input Password
- Input Radio
- Input
- Reset
- Input Submit
- Input Text
- Link
- Meta
- Object
- Option
- Select
- Style
- Table
- TableCell
- TableRow
- Textarea

U delu "**ugradjene JavaScript funkcije**" proučili smo čiste JavaScript objekte. U ovom poglavlju u nastavku proučićemo window objekte. A ostale Browser objekte, kao i HTML DOM objekte proučavaćemo u posebnim glavama.

## Alert, confirm i prompt window dijaloški prozori

JavaScript omogućava kreiranje tri "iskakajuća" prozora tj. **popup box-a**:

1. alert box (takozvani "upozoravajući" iskaćući prozor)
2. confirm box (dijalog za potvrdu)
3. prompt box (odzivni okvir za dijalog).

## CONFIRM

ALERT box smo već upoznali, pa da krenemo na confirm box. **CONFIRM** iskaćući prozor prikazuje dva dugmeta: Cancel i OK, i vraća vrednost: true ako korisnik pritisne OK ili false ako pritisne Cancel. Ovaj okvir za dijalog može se koristiti kao podsticaj korisniku da donese odluku o tome kako da se nastavi skript. Izlazna vrednost može se koristiti i kao uslov u nekoj if konstrukciji. CONFIRM funkcija je `confirm()`. Evo primera:

```
<html><head>
<script type="text/javascript">
function izaberi(){
r = confirm("izaberi OK ili CANCEL");
if (r == true) {
document.write("Pritisli ste OK!");
} else {
document.write("Pritisli ste Cancel!");
}
}
</script></head><body>
<input type="button" onclick="izaberi()" value="izaberi OK ili CANCEL">
</body></html>
```

## PROMPT

PROMPT iskaćući prozor prikazuje poruku (pitanje) koju kreiramo, i obezbeđuje polje za unos teksta u koje korisnik unosi odgovor. Dva dugmeta, Cancel i OK, omogućavaju korisniku da ukloni okvir za dijalog sa dva efekta: otkazivanje cele operacije ili prihvatanje unetog. `prompt()` ima dva parametra, prvi je poruka koja se prikazuje korisniku, a drugi je podrazumevani odgovor ukoliko recimo korisnik klikne na OK. Ako nije potreban podrazumevani odgovor navodi se prazan niz- "". Pritiskom na dugme Cancel, metod vraća null (u uslovu neke if konstrukcije smatra se za false); pritiskom na OK vraća se vrednost unetog znakovnog niza, ili podrazumevana vrednost. Evo primera koji sve lepo objašnjava:

```
<html><head>
<script type="text/javascript">
function iskok() {
ime = prompt("Molimo unesite vaše ime", "Harry Potter");
if (ime != null && ime != ""){
document.write("Zdravo " + ime + "! Kako ste danas?");
}else{
document.write("E baš ste pokvareni što ste kliknuli Cancel!!");
}
}
</script></head><body>
<input type="button" onclick="iskok()" value="Unesi ime">
</body></html>
```

## Otvaranje stranice u iskaćućem prozoru

Moguće je otvoriti i čitavu stranicu, bilo da je ona HTML, ili PHP itd, u iskaćućem (popup) prozoru. Funkcija za to je `open()`. Šematski izraz je sledeći:

```
window.open(URL, ime, parametri, zamena)
```

**URL** - parametar u kome se u znacima navoda upisuje link bilo koje web stranice, bilo da je sa našeg sajta ili nekog drugog. To je opcioni parametar, i ako se ne navede možemo na primer mi sami definisati sadržaj tog prozora, što će mo kasnije kroz primer i videti. Ne navodi se tako što se samo stave navodnici a ništa se ne piše izmedju njih

**ime** (neobavezan parametar) - ako ga ne želimo samo stavimo znakove navoda a ništa ne pišemo. Taj parametar određuje naziv stranice ukoliko je prvi parametar samo navodnici. Medjutim tu možemo definisati i na koji način će se otvoriti taj novi prozor. Vrednosti koje se mogu naći u ovom parametru su sledeći:

- **\_blank** - strana se otvara u novom prozoru. To je po defaultu
- **\_parent** - strana se otvara u takozvanom "roditeljskom" okviru, to jest u prozoru u kome smo i inicirali pokretanje te strane
- **\_self** - slično kao prethodno
- **\_top** - vezano za frameset, a za obične strane slično kao prethodna dva
- **Ime** - ime prozora

**Parametri** (takodje neobavezan parametar) - tu se navode parametri koji opisuju kako će izgledati novootvorena stranica. Na primer da li će imati klizače, kolika će biti po dimenzijama i slično. Znači mogu se tu naći i svi ti parametri zajedno samo se odvajaju zarezom, i svi se stavljaju unutar jednog para navodnika. Videćete u ostalom u primeru. E sad tih parametara ima podosta pa da ih damo tabelarno sa objašnjenjima:

Parametar	Objašnjenje
channelmode = yes   no   1   0	Hoće li ili neće prikazati prozor u kompletu. Po defaultu je nula ili <b>no</b>
directories = yes   no   1   0	Hoće ili neće prikazati menju za direktorijume. Po defoltu je 1 ili <b>yes</b> .
fullscreen= yes   no   1   0	Hoće li ili neće prikazati prozor u full-screen modu. Zadana je <b>no</b> ili nula. Prozor u full-screen modu takođe mora biti u kompletu
height = pixels	Visina prozora u pikselima, ali se oznaka za piksele ne piše. Minimalna visina je 100.
width = pixels	Širina prozora u pikselima, ali se oznaka za piksele ne piše. Minimalna širina je 100.
left = pixels	Smešta popup prozor na odredjenom položaju u odnosu na levu ivicu brauzera
top = pixels	Smešta popup prozor na odredjenom položaju u odnosu na gornju ivicu brauzera
location = yes   no   1   0	Da li da prikže adresno polje u brauzeru za taj popup. Po defoltu je yes
menubar = yes   no   1   0	Da li da prikazuje menubar u brauzeru za taj popup. Po defoltu je yes
resizable = yes   no   1   0	Da li da omogući promenu veličine popup prozora. Po defoltu je yes
scrollbars = yes   no   1   0	Da li da prikazuje scrollbar u brauzeru za taj popup. Po defoltu je yes
status = yes   no   1   0	Da li da prikazuje status bar u brauzeru za taj popup. Po defoltu je yes
titlebar = yes   no   1   0	Da li da prikazuje naslov bar u brauzeru za taj popup. Po defoltu je yes
toolbar = yes   no   1   0	Da li da prikazuje toolbar u brauzeru za taj popup. Po defoltu je yes

**Zamena** - je takodje opcioni parametar koga ako želimo možemo potpuno izostaviti. Konkretno parametar može imati jednu od dve vrednosti, a one su **true** ili **false**. Ako je **true**, nova lokacija će zameniti trenutnu stranicu u istoriji brauzera. Neki brauzeri će jednostavno ignorisati ovaj parametar. Evo primera **open()** funkcije:

```
<html><head>
<script type="text/javascript">
function otvori(){
    window.open("http://mmance02.on.neobee.net", "Moja strana", "toolbar=no,
location=no, directories=no, status=no, scrollbars=yes, width=550,
height=600");
}
</script>
</head><body>
<input type=button value="otvori stranicu" onclick="otvori()">
</body></html>
```

A evo primera kad otvaramo novu stranicu i u njoj pišemo nešto:

```
<html><head>
<script type="text/javascript">
function otvori(){
    prozor = window.open("", "Nasa", "width=400, height=300");
    prozor.document.write("Ovo je nas prozor, zar nije lep?");
}
</script></head><body>
<input type="button" value="Otvori našu stranu" onclick="otvori()">
</body></html>
```

# Close(), moveBy(), moveTo(), focus() i blur()

## Close()

Funkcija za zatvaranje otvorenih prozora je **close()**. Evo primera koji sve objašnjava:

```
<html><head>
<script type="text/javascript">
function otvori(){
    prozor=window.open("", "", "width=200,height=100");
    prozor.document.write("Ovo je otvoren prozor");
}
function zatvori(){
    prozor.close();
}
</script>
</head><body><br><br><br><br><br>
<input type="button" value="Otvori novi prozor" onclick="otvori()"><br>
<input type="button" value="zatvori novi prozor" onclick="zatvori()">
</body></html>
```

## MoveBy( )

**MoveBy( )** funkcija pomera otvoren prozor na poziciju koju definišemo u toj funkciji. Sve je prosto, evo primera:

```
<html><head>
<script type="text/javascript">
function otvori(){
    prozor = window.open("", "", "width=200,height=100");
    prozor.document.write("Ovo je naš prozor");
}
</script>
</head><body><br><br><br><br><br><br>
<input type="button" value="Otvori novi prozor" onclick="otvori()"><br>
<input type="button" value="Pomeri otvoreni prozor"
onclick="prozor.moveBy(250,250); prozor.focus()">
</body></html>
```

## MoveTo( )

**MoveTo( )** je potpuno identična funkcija funkciji **MoveBy( )**.

## Focus( )

U primeru takodje figuriše kao što vidimo i funkcija **focus( )**. Na primer napišite i pokrenite gornji program bez **focus( )**, i videćete da će se prozor pomeriti ali biće iza glavnog prozora. Znači da bi ste ponovo uočili pomereni prozor morali bi ste da se spustite u kontrolnoj liniji operativnog sistema i kliknuli na prozor da bi fokusirali pomereni prozor. Sa ovom funkcijom **focus( )** prozor je odma u fokusu.

## Blur( )

Postoji i funkcija **blur( )** koja je suprotnost od **focus( )**. Znači ukoliko postoji neki prozor iza tog prozora on će izaći u fokus a ovaj naš će se ukloniti iz fokusa.

## Printanje strane

Funkcija za printanje veb stranice je **print( )**. Ovom funkcijom će se isprintati sve ono što se vidi u display brauzera preko štampača na papir. Evo primera:

```
<html><head>
<script type="text/javascript">
function printanje(){
    window.print();
}
</script>
</head><body>
<input type="button" value="Printaj stranu" onclick="printanje()">
</body></html>
```

# SetTimeout(),clearTimeout(),setInterval( ), clearInterval()

## SetTimeout( )

SetTimeout( ) izvršava određene naredbe ili funkcije nakon određenog broja milisekundi (1 sekunda = 1000 milisekunda). Prototip te funkcije je:

```
setTimeout("nefunkcija( )", 1000)
```

Znači funkcija nefunkcija( ) će se izvršavati svake sekunde.

## ClearTimeout( )

ClearTimeout( ) - zaustavlja izvršenje funkcije SetTimeout( ). Evo konkretan primer koji objašnjava obe funkcije:

```
<html><head>
<script type="text/javascript">
c=0;
t=0;
function startuj(){
    document.getElementById("txt").value=c;
    c=c+1;
    t = setTimeout("startuj()",100);
}
</script>
</head><body><form>
<input type="button" value="Startuj" onClick="startuj()">
<input type="text" size=10 id="txt">
<input type="button" value="Stopiraj" onClick="clearTimeout(t)">
</form></body></html>
```

## SetInterval( ) i clearInterval( )

Razlika između **setInterval( )** i **setTimeout( )** je ta što **setTimeout( )** na svakih X milisekundi poziva na primer neku funkciju na izvršenje, a **setInterval** izvršava pozvanu funkciju sve dok je ne opozove funkcija **clearInterval( )**. Evo primera za **SetInterval( )** i **ClearInterval( )**:

```
<html><head>
<script type="text/javascript">
int=self.setInterval("clock()",50);
function clock() {
    t=new Date();
    document.getElementById("clock").value=t;
}
</script>
</head><body>
<input type="text" id="clock" size="40">
```



```
<button onclick="int=window.clearInterval(int)">Stop interval</button>
</body></html>
```

## ResizeBy() i resizeTo()

## ResizeBy( )

ResizeBy( ) vrši promenu veličine prozora ali u zavisnosti od postojeće veličine. Znači vrednosti u pikselima koje se unose kao parametri se oduzimaju ili dodaju na postojeću veličinu prozora. Stim ako se unesu negativne vrednosti onda se smanjuje a pozitivne povećavaju prozor. Evo primera koji sve lepo demonstrira:

```
<html><body>
<input type="button" onclick="resizeBy(-100,-100)" value="Resize window">
</body></html>
```

## ResizeTo( )

ResizeTo( ) je slična kao ResizeBy( ) ali se razlikuje po tome što ne uzima u obzir trenutna veličina prozora, već se direktno definiše nova veličina prozora. Evo primer:

```
<html><body>
<input type="button" onclick="resizeTo(400,400)" value="Resize window">
</body></html>
```

## ScrollBy() i scrollTo()

## ScrollBy()

ScrollBy ( ) za broj piksela (koji su kao parametri) pomera (skroluje) u odnosu na trenutnu poziciju. Evo primera:

[illegible]

## ScrollTo( )

ScrollTo( ) skroluje nezavisno od trenutne pozicije u prozoru na tačno definisan položaj. Evo primera:

```

<html><head>
</head><body>
<input type="button" onclick="window.scrollTo(300,300)" value="Scroll"/>
<p>SCROLL SCROLL SCROLL</p><br><b><br><br><br><br><br><br>
<p>SCROLL SCROLL SCROLL</p><br><b><br><br><br><br><br><br>
<p>SCROLL SCROLL SCROLL</p><br><b><br><br><br><br><br><br>
<p>SCROLL SCROLL SCROLL</p><br><b><br><br><br><br><br><br>
<p>SCROLL SCROLL SCROLL</p><br><b><br><br><br><br><br><br>
<p>SCROLL SCROLL SCROLL</p><br><b><br><br><br><br><br><br>
</body></html>

```

## Navigatorske funkcije

Navigacijske funkcije sadrže informacije o pregledniku. Daću primer sa kompletno svim ovim funkcijama, a mislim da on sve objašnjava:

```

<html><body><script type="text/javascript">
x = navigator;
document.write("Ime brauzera=" + x.appName + "<br>");
document.write("Verzija brauzera=" + x.appVersion + "<br>");
document.write("Podverzija brauzera=" + x.appMinorVersion + "<br>");
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
document.write("KOD brauzera=" + x.appCodeName + "<br>");
document.write("Da li su mogući kolacici u pregledniku=" + x.cookieEnabled +
"<br>");
document.write("CPU klasa sistema=" + x.cpuClass + "<br>");
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
document.write("Da li je posetilac online=" + x.onLine + "<br>");
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
document.write("Platforma operativnog sistema=" + x.platform + "<br>");
document.write("User agent=" + x.userAgent + "<br>");
document.write("Jezik brauzera=" + x.browserLanguage + "<br>");
// Ne podržava Firefox (verzija 3.3)
document.write("Jezik sistema=" + x.systemLanguage + "<br>");
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
document.write("Jezik korisnika=" + x.userLanguage + "<br>");
// Ne podržava Firefox (verzija 3.3)
</script></body></html>

```

## JavaEnabled( )

**JavaEnabled( )** - ova funkcija proverava da li brauzer podržava JavaScript. Ako podržava vraća true a ako ne podržava onda ne podržava, ne može ništa da vrati :) Evo primera:

```

<html><body>
<script type="text/javascript">
alert(navigator.javaEnabled());
</script></body></html>

```

## Ekranске (screen) funkcije

Ekranske funkcije daju informacije o ekranskim podacima klijenta. Daću primer sa kompletno svim ovim funkcijama a mislim da on sve objašnjava:

```
<html><body>
<script type="text/javascript">
document.write("Rezolucija ekrana: " + screen.width + "*" + screen.height +
"<br>");
document.write("Rezolucija bez tulbarova: " + screen.availWidth + "*" +
screen.availHeight + "<br>");
document.write("Broj bita boje ekrana: " + screen.colorDepth + "<br>");
document.write("Broj bita boje u baferu: " + screen.bufferDepth + "<br>");
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
document.write("Horizontalni broj piksela po inču: " + screen.deviceXDPI+
"<br>");
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
document.write("Vertikalni broj piksela po inču: " + screen.deviceYDPI +
"<br>");
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
document.write("Horizontalni broj normalnih piksela po inču: " +
screen.logicalXDPI + "<br>");
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
document.write("Vertikalni broj normalnih piksela po inču: " +
screen.logicalYDPI + "<br>");
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
document.write("Omogućeno uredjivanje fontova: " +
screen.fontSmoothingEnabled + "<br>");
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
document.write("Broj bita boje po pixelu : " + screen.pixelDepth + "<br>");
// Ne podržava Internet Explorer 8
document.write("Interval osvežavanja ekrana : " + screen.updateInterval);
// Ne podržavaju Firefox (verzija 3.3) i Opera (verzija 9.6)
</script></body></html>
```

## History funkcije

History funkcije čuvaju URL-ove koje je korisnik posetio.

### History.length

`history.length` - vraća broj stranica na sajtu koje je posetio posetilac. Na primer ako napišemo `document.write(history.length)` ispisaće se broj posećenih strana posetioca. Ovaj podatak može se korisno upotrebiti u nekim statistikama posete sajta.

### History.back( ) i history.forward( )

`history.back( )` i `history.forward( )` - pomoću ovih funkcija možemo kreirati dugme kojim će mo omogućavati posetiocima sajta da se vraćaju (back) na prethodnu posećenu stranicu, ili da idu na sledeću stranicu u odnosu na tu (forward). Evo primera:

```
<html><body>
<input type="button" value="Povratak na prethodnu stranicu">
```

```
onclick="window.history.back() ">
</body></html>
```

## Go(argument)

**go(argument)** - Funkcija koja učitava tačno odredjenu stranicu iz istorije poseta stranica. Argument može biti broj stranice, znači 1, 2, 3 itd, ili pak konkretan URL stranice. Na primer ako hoćemo efekat `back()` funkcije možemo pisati `go(-1)`. Medjutim možemo se vratiti na neku stranicu još u nazad na primer `go(-3)`, itd.

## Location funkcije

**Location** funkcija sadrže informacije o trenutnom URL. Location objekti su deo prozora objekta i pristupa im se putem [window.location](#).

```
<html><body>
<script type="text/javascript">
document.write("Domen sajta: " + location.host + "<br>");
    // na primer ako se stranica otvorila sa google hosta daće informaciju
    www.google.com
document.write("Domen sajta: " + location.hostname + "<br>");
    // potpuno isto kao location.host
document.write("Kompletna adresa otvorene stranice : " + location.href +
"<br>");
document.write("Path stranice: " + location.pathname + "<br>");
    // na primer ako je strana http://nekisajt.com/strane/prva.htm rezultat
    će biti /strane/prva.htm
document.write("Informacije o protokolu stranice: " + location.protocol +
"<br>");
    // na primer da li je http, ftp, i slično
document.write("Query string: " + location.search + "<br>");
    // ako je stranica http://xxxxxx.com/aaa.php?u=5&n=23 daće ?u=5&n=23
</script></body></html>
```

**assign(URL)** - Omogućava "programsko linkovanje". Na primer možete ovom funkcijom ako umesto URL napišete neku adresu bilo koje internet stranice da u tom istom prozoru se otvori ta stranica bez da je posetilac kliknuo na bilo koji link. Naravno može se napraviti i da klikne na neko dugme pa da se otvori, ali onda je to klasično linkovanje koje može da se ostvari i preko HTML anchor elemenata. **ASSIGN** omogućava više. Na primer možete napraviti da posetilac otvori neku stranicu ako samo i predje preko nekog teksta, ili neke slike ili čak preko neke bilo koje površine prazne ili na slici itd. Ili na primer da napravite ako klikne na neku površinu na slici da tako linkuje. Evo primera sa dugmetom:

```
<html><body>
<input type="button" value="Idite na google"
onclick="window.location.assign('http://www.google.com') ">
</body></html>
```

## HTML DOM objekti

HTML DOM je mapa HTML i CSS objekata. JavaScript ima za svaki HTML i CSS objekat odgovarajuću funkciju pomoću koje pristupamo, dodajemo, menjamo ili brišemo taj HTML ili CSS objekat. HTML objekti su u glavnom svi HTML tagovi, a CSS objekti su u glavnom svi CSS atributi. Da bi smo ovim HTML i CSS objektima pristupili moramo uvek ispred odgovarajućih funkcija, odvojeno tačkom, koristiti rezervisanu reč **document**. Document objekat je takođe deo window objekta, te se može pisati **iwindow.document** ali može i bez tog window, pa se zato skoro i nikad ne piše window.

## Document.anchors[]

`document.anchors[ ]` je niz koji čuva HTML elemente svih linkova na stranici, kao što su tekstovi linkova (`innerHTML`), adrese na koje šalju linkovi (`href`), imena linkova ukoliko ih ima (`name`), može da izbaci koliko ima linkova na strnici (`length`). Evo primera:

```
<html><body>
<a href="http://nekisajt.com/strana1.htm" name="link1">Prva strana
sajta</a><br>
<a href="http://nekisajt.com/strana2.htm" name="link2">Druga strana
sajta</a><br>
<a href="http://nekisajt.com/strana3.htm" name="link3">Trecu strana
sajta</a><br>
<script type="text/javascript">
a = document.anchors.length;
for (i = 0; i < a; i++){
document.write("<br>");
document.write("Tekst linka" + i + " je: " +document.anchors[i].innerHTML +
"<br>");
document.write("Adresa linka" + i + " je: " + document.anchors[i].href +
"<br>");
document.write("Ime linka" + i + " je: " + document.anchors[i].name +
"<br>");
}
document.write("<br><br>Na stranici ima ukupno linkova: ",
document.anchors.length);
</script>
</body></html>
```

Rezultat:

Tekst linka0 je: Prva strana sajta  
Adresa linka0 je: <http://nekisajt.com/strana1.htm>  
Ime linka0 je: link1

Tekst linka1 je: Druga strana sajta  
Adresa linka1 je: <http://nekisajt.com/strana2.htm>  
Ime linka1 je: link2

Tekst linka2 je: Trecu strana sajta  
Adresa linka2 je: <http://nekisajt.com/strana3.htm>  
Ime linka2 je: link3

Na stranici ima ukupno linkova: 3

# document.forms[] i document.images[]

**document.forms[ ]** - Potuno ist kao i **anchors[ ]** samo što se odnosi na forme.

```
<html><body>
<form name="formal" method="POST" action="http://nekisajt.com/index.php">
  <p><input type="text" name="tekst1" size="20"></p>
  <p><textarea rows="2" name="tekst areal" cols="20"></textarea></p>
  <p><input type="submit" value="Submit" name="submit dugme1">
  <input type="reset" value="Reset" name="reset dugme1"></p>
</form>
<form name="forma2" method="GET" action="http://nekisajt.com/index.php">
  <p><input type="text" name="tekst2" size="20"></p>
  <p><textarea rows="2" name="tekst area2" cols="20"></textarea></p>
  <p><input type="submit" value="Submit" name="submit dugme2">
  <input type="reset" value="Reset" name="reset dugme2"></p>
</form>
<script type="text/javascript">
document.write("Broj formi na stranici je: " + document.forms.length +
"<br>")
document.write("Ime prve forme na stranici je: " + document.forms[0].name+
"<br>")
document.write("Metoda prenosa podataka druge forme je: "
+document.forms[1].method + "<br>")
document.write("Podaci prve forme se salju na: " +document.forms[0].action +
"<br>")
</script>
</body></html>
```

**document.images[ ]** - Potpuno isto kao i prethodne dve funkcije samo što se u ovom slučaju odnosi na slike.

## Document.links[]

**document.links[ ]** - Potpuno isto kao prethodne funkcije samo što se odnosi na links. Šta je links? Evo primera koji sve otkriva:

```
<html><head>
<script type="text/javascript">
function klik(a){
prozor = window.open("", "", "width=167, height=224");
prozor.document.write("<body bgcolor=\"\#000000\"><img src =\"\" + a + \"\">");
}
</script></head><body>
<img src = "jsslikeplanete/sve.jpg" width="608" height="265" usemap
="#planetmap">
<map name="planetmap">
<area shape="rect" coords="544, 0, 604, 26" href="#"
OnClick=klik("jsslikeplanete/pluton.jpg") alt="Pluton">
<area shape="circle" coords="92, 199, 12" href="#"
OnClick=klik("jsslikeplanete/merkur.jpg") alt="Merkur">
```

```

<area shape="circle" coords="140, 179, 17" href="#"
OnClick=klik("jsslikeplanete/venera.jpg") alt="Venera">
<area shape="circle" coords="189, 152, 16" href="#"
OnClick=klik("jsslikeplanete/zemlja.jpg") alt="Zemlja">
<area shape="circle" coords="221, 121, 14" href="#"
OnClick=klik("jsslikeplanete/mars.jpg") alt="Mars">
<area shape="circle" coords="303, 136, 28" href="#"
OnClick=klik("jsslikeplanete/jupiter.jpg") alt="Jupiter">
<area shape="circle" coords="390, 95, 31" href="#"
OnClick=klik("jsslikeplanete/saturn.jpg") alt="Saturn">
<area shape="circle" coords="455, 60, 21" href="#"
OnClick=klik("jsslikeplanete/uran.jpg") alt="Uran">
<area shape="circle" coords="509, 31, 20" href="#"
OnClick=klik("jsslikeplanete/neptun.jpg") alt="Neptun">
<area shape="poly" coords="56, 264, 0, 264, 0, 0, 33, 15, 73, 134" href="#"
OnClick=klik("jsslikeplanete/sunce.jpg") alt="Sunce">
</map>
<p>Broj area/linkova na stranici je:
<script type="text/javascript">
document.write(document.links.length);
</script></p></body></html>

```

Rezultat



Broj area/linkova na stranici je: 31

Klikćite na planete da bi ste videli HTML efekat MAP taga

## Document.cookie i još neke funkcije

**document.cookie** - Čuva informacije o kolačićima koji su povezani sa internet stranicom. Šta su JavaScript kolačići (cookie)? JavaScript Cookie su tekstualni fajlovi koje brauzer posetioca memoriše u sebe za svaku internet stranicu koju je otvorio. Taj tekstualni fajl se razlikuje od stranice do stranice, što znači da je to u stvari identifikacioni broj stranice (ID). JavaScript (to jest mi kao programeri :) )

pomoću ove COOKIE funkcije definiše cookie fajl. Format koji cookie fajl mora zadovoljiti je (sve ono što je u srednjim zagradama je opciono, to jest nije obavezno):

```
ime = vrednost[;expires=datum][;domain=imeDomena][;path=putanja] [;secure]
[;httpOnly]
```

- ime - ime koje definiše upisani cookie
- vrednost - je upravo informacija koja se želi zapamtiti kako bi se identifikovao posetilac (to je glavna stvari i obavezan je parametar)
- datum - je datum koji definiše do kada cookie ostaje memorisan u brauzeru.
- imeDomena - definiše jedini domen sa kog cookie može da se čita i da mu se menja vrednost
- putanja - definiše jedinu putanju sa koje cookie može da se čita i da mu se menja vrednost.
- secure - upis i čitanje cookie se izvršava preko posebnih, bezbednijih kanala

Atributi kolačića expires, domain, path, secure, httpOnly su opcioni i nije bitan redosled u kom se pojavljuju.

Evo primera:

```
<html><head>
<script type=javascript>
function postavljajCookie() {
    document.cookie="cookie je=" + document.formal.imeCookie.value;
}
function prikazCookie() {
    alert(document.cookie)
}
</script>
</head><body>
<form name="formal">
    <h2>Postavljanje i pregled COOKIE</h2>
    <p>Upiši ime: <input type="text" name="imeCookie" size="20"></p>
    <input type="button" value="upiši" onClick="postavljajCookie()">
    <input type="button" value="Prikaži Cookie" onClick="prikazCookie()">
</form>
</body></html>
```

A rezultat (naravno svaki put je drugačiji) u alert prozoru je na primer:

[TP\\_2215615646=1260887721188; cookie je=Pera Perić](#)

**document.domain** - ispisuje domen

**document.lastModified** - Ispisuje datum kad je zadnji put modifikovana internet stranica

**document.referrer** - Daje kompletan link stranice

**document.title** - Daje naslov internet stranice.

**document.URL** - Potpuno identicno kao i **referrer**. Znači daje kompletan link stranice.



**document.close( )** - Sa funkcijom close( ) smo se već susreli kada smo proučavali WINDOW objekte. Ta funkcija zatvara dokument.

**document.open( )** - I sa funkcijom open() smo se isto susreli već kad smo proučavali WINDOW objekte, i ona otvara novi prozor na primer, to jest novi dokument.

**document.write( )** - Sa funkcijom write() smo se već susreli bezbroj puta i ona omogućava JavaScript ispisivanje na internet stranici.

## Document.getElementById(id)

I sa ovom funkcijom smo se susretali ali sad da je malo bolje objasnimo. Funkcija getElementBy je pokazivač elemenata, na primer HTML tagova. E sad u zavisnosti šta pokazuje imamo getElementById("a1") - pokazuje neki element koji ima identifikator "a1". Ili getElementsByTagName(imeTaga) - pokazuje na sve tagove sa određenim imenom. GetElementsByName(ime) - pokazuje neki elemenat sa specifičnim imenom, na primer na neki element forme, ili sama forma. E sad u nastavku može stajati na primer **innerHTML** - što znači da menja vrednost HTML teksta tog elementa. Ili samo **value** - što znači da menja neku vrednost u nekom form elementu kao što je vrednost nekog text elementa forme. A može da stoji i **length** - što znači da će pokazati broj nekih elemenata. Evo primera:

```
<html><head>
<script type=text/javascript>
function brojZnaka() {
    document.getElementById("b").value =
document.getElementById("a").innerHTML.length
}
function sve() {
    document.getElementById("a").innerHTML="AAAAAA";
    brojZnaka()
}
</script></head><body>
<input type=button value="Klikni za iznenadjenje" onclick=sve()>
<p id="a"><b><font face="Arial" size="7">aaaaaaaaaaaaaaaa</font></b></p>
<p><input type=text id="b"></p>
</body></html>
```

## Anchor funkcije

Ne moramo samo mišem birati linkove. Biranje linka možemo omogućiti i na primer istovremenim kliktnjem Alt + neka dirka. Nakon takvog biranja linka potrebno je samo kliknuti na Enter kako bi i aktivirali taj link. Ovo nam omogućava **accessKey** funkcija. Opera 9 ne podržava ovu funkciju. Evo primera koji sve objašnjava:

```
<html><head>
<script type="text/javascript">
function accesskey() {
    document.getElementById("w1").accessKey="w";
    document.getElementById("w2").accessKey="d";
}
```

```

</script>
</head><body onload="accesskey()">
<a id="w1" href="http://www.nekisajt1.com">Neki sajt 1</a> (Klikni Alt + w
da izabereš link 1, a zatim Enter)<br>
<a id="w2" href="http://www.nekisajt2.com">Neki sajt 2</a> (Klikni Alt + d
da izabereš link 2, a zatim Enter)
</body></html>

```

Kao što znamo nakon linkovanja, stranica se može otvoriti ili u samom jezičku u kome smo kliknuli, ili u posebnom jezičku brauzera. A to se definiše kao što znamo u anshor tagu atributom **target** koji može imati vrednosti **\_blank**, **\_self**, **\_parent**, i **\_top**. E sad mi i preko **target** funkcije u JavaScript-u možemo podešavati anchor target tag po našoj želji. Isto sve važi i za **href** atribut. Zatim sam tekst linka takodje možemo menjati po želji pomoću **innerHTML** funkcije. ID znamo da je identifikacioni broj anchor elementa. Sve ovo možda izgleda nejasno ali naravno sa jednim primerom je sve jasno:

```

<html><head>
<script type="text/javascript">
function promenaLinka(){
    document.getElementById("linki").innerHTML="Poseti Yahoo";
    document.getElementById("linki").href="http://www.yahoo.com";
    document.getElementById("linki").target="_blank";
}
</script>
</head><body>
<a ID = "linki" href="http://www.google.com" target=_top>Poseti
google</a><br><br>
<input type="button" onclick="promenaLinka()" value="Promeni link u Yahoo">
</body></html>

```

## Tastatura i miš (EVENT atributi)

**event.altKey = true | false | 1 | 0** - altKey vraća logičku vrednost true ili 1 ako je ALT dirka na tastaturi pritisnuta, ili false, odnosno 0 ako nije. Evo primera:

```

<html><head>
<script type="text/javascript">
function drzi(event){
    if (event.altKey == 1) {
        alert("Vi držite ALT tipku!");
    }else{
        alert("ALT tipku ne držite trenutno!");
    }
}
</script>
</head><body onmousedown="drzi(event)">
Alert &#263;e vam re&#263;i da li držite tipku ALT ili ne. Ako želite da vam
javi da
držite, morate držati ALT i u isto vreme mišem kliknuti bilo gde na stranici
</body></html>

```

**event.ctrlKey = true | false | 1 | 0** - Potpuno identično kao i altKey samo što je u pitanju CTRL tipka umesto ALT

**event.shiftKey = true | false | 1 | 0** - Potpuno identično kao i altKey samo što je u pitanju SHIFT tipka umesto ALT

**event.button = 0 | 1 | 2** - Određuje koji ste klik mišem izvršili i to tako da 0 znači da ste izvršili levi klik mišem; 1 znači da ste izvršili srednji klik mišem (kod internet explorer je za tu funkciju broj 4); 2 znači da ste izvršili desni klik mišem. Evo primera:

```
<html><head>
<script type="text/javascript">
function klikMis(event){
    if(event.button == 2){
        alert("Vi ste kliknuli desni klik mišem!");
    }
    else{
        alert("Vi ste kliknuli levi klik mišem!");
    }
}
</script>
</head><body onmousedown="klikMis(event)">
Kliknite bilo kojim dugmetom mišem, bilo gde na stranici</p>
</body></html>
```

**event.screenX** i **event.screenY** - ove funkcije određuju x i y koordinate kursora u odnosu na ekransku površinu. Evo primera:

```
<html><head>
<script type="text/javascript">
function misPomeraj(event){
    x=event.screenX;
    y=event.screenY;
    document.getElementById("txt1").value=x;
    document.getElementById("txt2").value=y;
}
</script>
</head>
<body onMouseMove="misPomeraj(event)">
<form>
<br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br>
X koordinata kursora: <input type="text" size=10 id="txt1"><br>
Y koordinata kursora: <input type="text" size=10 id="txt2">
</form></body></html>
```

**event.clientX** i **event.clientY** - Vraća koordinate tačke ekrana u kojoj je kliknuto. Evo primera:

```
<html><head>
<script type="text/javascript">
function koordinate(event){
    x=event.clientX;
    y=event.clientY;
    alert("X koordinata je: " + x + ", Y koordinata je: " + y);
}
</script>
</head><body onmousedown="koordinate(event)">
<p>Klikni bilo gde na stranici, a alert prozor će pokazati koordinate u
```

```
kojoj tačci ste kliknuli.</p>
</body></html>
```

## Tabele

HTML tabele su takodje HTML DOM objekti ali zbog kompleksnosti smo ih izdvojili kao zasebno poglavlje. JavaScript funkcije koje manipulišu HTML tabelama su toliko sveobuhvatne da takoreći mogu iz korena promeniti bilo koju HTML tabelu, to jest promeniti je u svakom mogućem smislu. Pored menjanja te iste finkcije mogu i vraćati sve te elemente tabela, što omogućava upotrebu tih vraćenih informacija u koju god želimo svrhu.

## Cells[] i rows[]

Niz cells[] sadrži elemente svake ćelije u tabeli (znači HTML tagove, vrednosti, znači sve sve), a niz rows[] sadrži elemente svakog reda u tabeli. Ove funkcije mogu i da prikažu sadržaje kolona i redova u tabeli, a isto tako mogu i da ih menjaju. Evo primera:

```
<html><head>
<script type="text/javascript">
function celija(a){
    if(a=="sadrzaj")
alert(document.getElementById("tabla").rows[0].cells[0].innerHTML);
    if(a=="sirina")
alert(document.getElementById("tabla").rows[1].cells[0].width);
    if(a=="polozaj")
alert(document.getElementById("tabla").rows[1].cells[1].align)
    if(a=="celije") alert(document.getElementById("tabla").rows[1].innerHTML)
    if(a=="promeni")
document.getElementById("tabla").rows[1].cells[1].innerHTML = "AAA"
}
</script></head><body>
<table id="tabla" border="1" width="143">
<tr>
<td>celija 1</td>
<td>celija 2</td>
</tr><tr>
<td width="60" height="40">celija 3</td>
<td align="center">celija 4</td>
</tr>
</table><br>
<input type="button" onclick=celija("sadrzaj") value="Sadrzaj prve
&#263;elije"><br>
<input type="button" onclick=celija("sirina") value="Sirina trece
celije"><br>
<input type="button" onclick=celija("polozaj") value="Polozaj teksta u
cetvrtoj celiji"><br>
<input type="button" onclick=celija("celije") value="HTML sadrzaj u drugom
redu"><br>
<input type="button" onclick=celija("promeni") value="Promeni sadrzaj u
cetvrtoj celiji">
</body></html>
```

# Border, cellPadding, cellSpacing, width i height

Sve ove funkcije menjaju ili isčitavaju istoimene tabelarne attribute. Evo primera:

```
<html><head>
<script type="text/javascript">
function celija(a){
    if(a=="border") document.getElementById("tabla").border=10;
    if(a=="padding") document.getElementById("tabla").cellPadding=15;
    if(a=="spacing") document.getElementById("tabla").cellSpacing=20;
    if(a=="sirina") document.getElementById("tabla").width=600
    if(a=="visina") document.getElementById("tabla").height=250
}
</script></head><body>
<table id="tabla" border="1" width="143">
    <tr>
        <td>celija 1</td>
        <td>celija 2</td>
    </tr><tr>
        <td width="60" height="40">celija 3</td>
        <td align="center">celija 4</td>
    </tr>
</table><br>
<input type="button" onclick=celija("border") value="Promeni border table"><br>
<input type="button" onclick=celija("padding") value="Promeni cellPadding tabele"><br>
<input type="button" onclick=celija("spacing") value="Promeni cellSpacing tabele"><br>
<input type="button" onclick=celija("sirina") value="Promeni sirinu tabele"><br>
<input type="button" onclick=celija("visina") value="Promeni visinu tabele">
</body></html>
```

## Frame

**frame = void | above | below | hside | vside | lhs | rhs | box | border** - FRAME postavlja ili vraća spoljne granice tabele u raznim oblicima. Evo primera:

```
<html><head>
<script type="text/javascript">
function celija(a){
    if(a=="above") document.getElementById("tabla").frame="above";
    if(a=="void") document.getElementById("tabla").frame="void";
    if(a=="below") document.getElementById("tabla").frame="below";
    if(a=="hside") document.getElementById("tabla").frame="hside";
    if(a=="vside") document.getElementById("tabla").frame="vside";
    if(a=="lhs") document.getElementById("tabla").frame="lhs";
    if(a=="rhs") document.getElementById("tabla").frame="rhs";
    if(a=="box") document.getElementById("tabla").frame="box";
}
```

```

        if(a=="border") document.getElementById("tabla").frame="border";
    }
</script></head><body>
<table id="tabla">
    <tr>
        <td>celija 1</td>
        <td>celija 2</td>
    </tr><tr>
        <td>celija 3</td>
        <td>celija 4</td>
    </tr>
</table><br>
<input type="button" onclick=celija("above") value="Promeni frame table u
above"><br>
<input type="button" onclick=celija("void") value="Promeni frame table u
void"><br>
<input type="button" onclick=celija("below") value="Promeni frame table u
below"><br>
<input type="button" onclick=celija("hsides") value="Promeni frame table u
hsides"><br>
<input type="button" onclick=celija("vsides") value="Promeni frame table u
vsides"><br>
<input type="button" onclick=celija("lhs") value="Promeni frame table u
lhs"><br>
<input type="button" onclick=celija("rhs") value="Promeni frame table u
rhs"><br>
<input type="button" onclick=celija("box") value="Promeni frame table u
box"><br>
<input type="button" onclick=celija("border") value="Promeni frame table u
border">
</body></html>

```

## Rules

**rules = none | groups | rows | cols | all** - RULES postavlja ili vraća unutrašnje granice tabele. Evo primera:

```

<html><head>
<script type="text/javascript">
function celija(a){
    if(a=="all") document.getElementById("tabla").rules="all";
    if(a=="none") document.getElementById("tabla").rules="none";
    if(a=="groups") document.getElementById("tabla").rules="groups";
    if(a=="rows") document.getElementById("tabla").rules="rows";
    if(a=="cols") document.getElementById("tabla").rules="cols";
}
</script></head><body>
<table id="tabla">
    <tr>
        <td>celija 1</td>
        <td>celija 2</td>
    </tr><tr>
        <td>celija 3</td>
        <td>celija 4</td>
    </tr>

```

```

</tr>
</table><br>
<input type="button" onclick=celija("all") value="Promeni rules table u all"><br>
<input type="button" onclick=celija("none") value="Promeni rules table u none"><br>
<input type="button" onclick=celija("groups") value="Promeni rules table u groups"><br>
<input type="button" onclick=celija("rows") value="Promeni rules table u rows"><br>
<input type="button" onclick=celija("cols") value="Promeni rules table u cols">
</body></html>

```

## InsertRow() i insertCell()

**insertRow( )** i **insertCell( )** - Funkcije za ubacivanje novog reda i novih celija u tabeli. Evo primera:

```

<html><head>
<script type="text/javascript">
function ubaciRed(){
    x = document.getElementById("tabla").insertRow(0);
    x.insertCell(0).innerHTML = "Novi red celija 1";
    x.insertCell(1).innerHTML = "Novi red celija 2";
}
</script>
</head><body>
<table id="tabla" border="1">
    <tr>
        <td>Red 1 celija 1</td>
        <td>Red 1 celija 2</td>
    </tr><tr>
        <td>Red 2 celija 1</td>
        <td>Red 2 celija 2</td>
    </tr>
</table><br>
<input type="button" onclick="ubaciRed()" value="Ubaci novi red u tabelu">
</body></html>

```

## DeleteRow(), deleteCell() i rowIndex

**deleteRow( )**, **deleteCell( )** i **rowIndex** - deleteRow briše red u tabeli, a deleteCell briše ćeliju. RowIndex vraća indeks reda u tabeli. Analogno tome cellIndex vraća indeks ćelije u redu. Evo primera:

```

<html><head>
<script type="text/javascript">
function brisiCeliju(){
    document.getElementById("tr2").deleteCell(1);
}
function brisiRed(r){
    i = r.parentNode.parentNode.rowIndex
    document.getElementById("tabla").deleteRow(i);
}

```

```

}
</script></head><body>
<table id="tabla" border="1">
  <tr id="tr1">
    <td>Red 1</td><td>Red 1</td>
    <td><input type="button" value="Izbrisi red"
onclick=brisiRed(this)></td>
  </tr><tr id="tr2">
    <td>Red 2</td><td>Ova celija se brise</td>
    <td><input type="button" value="Izbrisi celiju"
onclick=brisiCeliju()></td>
  </tr>
</table></body></html>

```

## Align

**align = left | right | center** - vraća ili podešava vodoravno poravnanje teksta u ćelijama ili redova. Evo primer za ćelije, ali analogno tome je i za redove stim što se identifikator postavi u željeni red:

```

<html><head>
<script type="text/javascript">
function poravnanje() {
  document.getElementById("td1").align = "right";
}
</script>
</head><body>
<table border="1" width="181">
  <tr>
    <th>Ime</th>
    <th width="93">Prezime</th>
  </tr><tr>
    <td id="td1">Milan</td>
    <td width="93">Milic</td>
  </tr>
</table>
<form>
<input type="button" onclick="poravnanje()"
value="Poravnaj u desno ime Milan">
</form></body></html>

```

## ColSpan i rowSpan

**colSpan i rowSpan** - colSpan menja ili vraća broj ćelija unutar jednog reda. A rowSpan menja ili vraća broj ćelija po kolonama. Evo primera:

```

<html><head>
<script type="text/javascript">
function promeni(a) {
  if (a=="colspan") document.getElementById("td2").colSpan="2";
  if (a=="rowspan") document.getElementById("td1").rowSpan="2";
}
</script>

```



```

</head><body>
<table border="1" width="226">
  <tr>
    <th>Ime</th>
    <th>Prezime</th>
  </tr><tr>
    <td id="td1">Harry</td>
    <td>Potter</td>
  </tr><tr>
    <td id="td2">Miki</td>
    <td>Milic</td>
  </tr>
</table><br>
<input type="button" onclick=promeni("colspan") value="Promeni colspan">
<input type="button" onclick=promeni("rowspan") value="Promeni rowspan">
</body></html>

```

## InnerHTML

**innerHTML** - menja ili vraća HTML sadržaj ćelije ili reda. Evo primera za ćeliju, ali analogno tome je i za red tabele, stim što se u željeni red postavi identifikator:

```

<html><head>
<script type="text/javascript">
function sadrzaj() {
  document.getElementById("td1").innerHTML="<b><i>BABA</i></b>";
}
</script>
</head><body>
<table border="1" width="226">
  <tr>
    <th>Ime</th>
    <th>Prezime</th>
  </tr><tr>
    <td id="td1">Miki</td>
    <td>Milic</td>
  </tr>
</table><br>
<input type="button" onclick=sadrzaj() value="Promeni ime" />
</body></html>

```

## Width

**width** - Vraca ili menja vodoravnu veličinu ćelija. Evo primer:

```

<html><head>
<script type="text/javascript">
function povecaj() {
  document.getElementById("td1").width="200px";
}
</script>
</head><body>

```

```

<table border="1">
  <tr>
    <th>Ime</th>
    <th>Prezime</th>
  </tr><tr>
    <td id="td1">Miki</td>
    <td>Milic</td>
  </tr>
</table><br>
<input type="button" onclick=povecaj() value="Promeni velicinu celije">
</body></html>

```

## VAlign

**vAlign = top | middle | bottom | baseline** - vraća ili menja vertikalni položaj sadržaja u ćeliji ili redu. Daćemo primer za ćelije ali isti slučaj je i za redove, stim što se identifikator postavi u željeni red. Naravno kad se postavi u red onda se vertikalni položaj sadržaja definiše u svim ćelijama unutar tog reda.

```

<html><head>
<script type="text/javascript">
function topAlign() {
  document.getElementById("td1").vAlign="top";
  document.getElementById("td2").vAlign="middle";
  document.getElementById("td3").vAlign="bottom";
  document.getElementById("td4").vAlign="baseline";
}
</script></head><body>
<table width="500" border="1" height="221">
  <tr>
    <th height="41">tekst0</th><th height="41">TeKst1</th>
    <th height="41">Tekst2</th><th height="41">Tekst3</th>
  </tr><tr>
    <td id="td1">top</td><td id="td2">middle</td>
    <td id="td3">bottom</td><td id="td4">baseline</td>
  </tr></table><br>
<input type="button" onclick="topAlign()" value="Top-align table cells">
</body></html>

```

## JS i CSS

CSS objekti su takodje HTML DOM objekti ali zbog kompleksnosti smo ih izdvojili kao zasebno poglavlje. JavaScript funkcije manipulišu apsolutno svim CSS objektima kako u smislu menjanja tako i u smislu isčitavanja. Ovo drugim rečima znači da preko JavaScript funkcija koje manipulišu CSS objektima mi možemo apsolutno dinamički menjati celokupni izgled stranica sajta.

Opšta JavaScript sintaksa za sve CSS atribut bi izgledala ovako:

```
document.getElementById("id").style.atribut = vrednost
```

Obzirom da ima na desetine CSS atributa, a na stotine vrednosti, mi nećemo ovde svaku ekvivalentnu JavaScript funkciju zasebno objašnjavati, već ćemo dati tabele sa svim tim JavaScript funkcijama i njihovim svim vrednostima uz tekstualno objašnjenje. Na kraju ćemo dati par primera korišćenja nekih od tih JavaScript funkcija, a ostale se skoro identično koriste.

Sve funkcije nisu zastupljene u svim brauzerima i u svim njihovim izdanjima. U tabelama zato imamo kolone u kojima je zapisano od koje verzije određenog brauzera počinje primena odgovarajuće JavaScript funkcije. Sa "O" u tabeli je obeležena Opera, sa "E" Internet Explorer, a sa "F" Firefox. Evo tabela:

## JS background CSS funkcije

Svi znamo šta se definiše background HTML atributom, tako da nije teško shvatiti šta će se definisati JavaScript background funkcijom. Ajde tabelarno da vidimo sve vrednosti koje može imati ova funkcija.

JS Funkcija	Objašnjenje	Vrednosti	Objašnjenje vrednosti	E	F	O
background	Definisanje boje pozadine	red	ime boje, blue, black ...	4	1	9
		rgb(232,123,54)	RGB boja			
		#AB3C23	Heksadecimalni zapis			
	Definisanje slike pozadine	url(URL)	URL adresa slike pozadine			
	Definiše kako će slika pozadine ispunjavati površinu elementa	repeat	Ponavljaće se i po vertikali i po horizontali da ispuni površinu elementa			
		repeat-x	Ponavljaće se po horizontali dok ne ispuni širinu elementa			
		repeat-y	Po vertikali			
		no-repeat	Neće se ponavljati. Pojaviće se na početku elementa jednom			
	Ponašanje pozadine prilikom skrolovanja	scroll	Pomeraće se			
		fixed	Biće fiksna			
	Početni položaj slike pozadine	top left	gornja leva pozicija	4	Ne	Ne
		top center	gornja centralna			
		top right	gornja desna			
		center left	centralna leva			
		center center	centralna centralna			
		center right	centralna desna			
		bottom left	donja leva			
		bottom center	donja centralna			
		bottom right	donja desna			
		X% Y%	Procentualna pozicija u odnosu na veličinu polja elementa			
		Xpos Ypos	Pozicija u pikselima, ili ostalim CSS veličinama			

Primer background funkcije:

```
<html><head>
<script type="text/javascript">
function stil(){
document.body.style.background = "#FFCC80 url(jsslikeplanete/zemlja.jpg)
repeat-y";
}
</script></head><body>
<input type="button" onclick="stil()" value="Podesi novi stil pozadine">
</body></html>
```

## JS border CSS funkcije

JS funkcije	Objašnjenje	Vrednosti	Objašnjenje vrednosti	E	F	O	
border	Definisanje debljine linije	thin	tanka linija	4	1	9	
		medium	srednje debljine				
		thick	debela linija				
		debljina linija u px	Definisanje debljine u pikselima				
borderBottom	Definisanje stila linije	none	bez stila				
		hidden	skriven				
		dotted	tufne				
		dashed	isprekidana linija				
borderLeft		solid	puna linija				
borderRight		double	dupla linija				
borderTop		groove	brazda				
		ridge	greben				
outline		inset	umetak				
		outset	ispupčenje				
		Definisanje boje linije	red				ime boje, blue, black ...
			rgb(123,34,213)				RGB boja
	#ACA445		Heksadecimalna boja				

Funkcija **border** se odnosi za sve 4 linije koje ograničavaju elemente. Medjutim mi možemo definisati i svaku liniju posebno. Tako na primer ako umesto **border** napišemo **borderLeft** onda će mo definisati samo levu ivicu elementa. **BorderRight** definiše desnu ivicu, **borderTop** definiše gornju liniju elementa, a **borderBottom** donju ivicu. I **border**, i **borderLeft**, **borderRight**, **borderBottom** i **borderTop** imaju iste vrednosti za definisanje boje, stila i debljine linija.

Kao što vidimo u tabeli egzistira i **outline** funkcija. Outline u CSS je u stvari linija koja uokviruje i sam **border**! I **outline** takodje ima iste vrednosti za boju, stil linije i debljinu linije, pa da ne bi pravili zasebnu tabelu mi smo i nju smestili u ovoj tabeli sa **border** funkcijama. Outline funkciju ne podržava Internet Explorer.

Evo jednog primera i za border i za outline:

```
<html><head>
<script type="text/javascript">
function element(a){
    if(a == "border") document.getElementById("p1").style.border="5px double
#0000FF";
    if(a == "outline") document.getElementById("p1").style.outline="6px
inset #FF00FF";
}
</script>
</head><body>
<input type="button" onclick=element("border") value="Promeni border"><br>
<input type="button" onclick=element("outline") value="Promeni outline">
<p id="p1">Ovo je paragraf</p>
</body></html>
```

## JS margin CSS funkcije

JS funkcije	Objašnjenje	Vrednosti	Objašnjenje vrednosti	E	F	O
margin	postavlja margine elementa	auto	marginu postavlja sam brauzer	4	1	9
marginBottom		dužina margine u px,mm itd.	definišemo na primer u pikselima			
marginLeft		dužina margine u %	u procentima u odnosu na ukupnu veličinu polja elementa			
marginRight						
marginTop						

Funkcija **margin** definiše sve margine oko elementa, a **marginBottom** donju marginu, **marginLeft** - levu, **marginRight** - desnu i **marginTop** - gornju marginu. Sve one se definišu istim vrednostima. Evo jednad primer:

```
<html><head>
<script type="text/javascript">
function Margine(){
    document.getElementById("p1").style.margin="100px";
}
</script>
</head><body>
<input type="button" onclick="Margine()" value="Promeni margine paragrafa">
<p id="p1">Ovo je paragraf</p>
<script type="text/javascript">
document.getElementById("p1").style.border="5px double #0000FF";
</script>
</body></html>
```

## JS padding CSS funkcije

JS funkcije	Objašnjenje	Vrednosti	Objašnjenje vrednosti	E	F	O
padding	definiše veličinu prostora	dužina u px	Dužina u pikselima	4	1	9

paddingLeft paddingTop paddingRight paddingBottom	između sadržaja elementa i border-a	dužina u %	Dužina u procentima u odnosu na veličinu prostora kojeg zahvata element			
--	-------------------------------------	------------	---	--	--	--

Kao što je i napisano u tabeli, **padding** definiše prostor između sadržaja elementa i border-a. Padding definiše sva 4 prostora, a **paddingLeft** samo levi, **paddingRight** - desni, **paddingTop** - gornji, **paddingBottom** definiše donji prostor. Evo primera:

```
<html><head>
<script type="text/javascript">
function element(){
    document.getElementById("p1").style.padding="40px";
}
</script>
</head><body>
<input type="button" onclick="element()" value="Promeni padding">
<p id="p1">Ovo je paragraf</p>
<script type="text/javascript">
document.getElementById("p1").style.border="5px double #0000FF"
</script>
</body></html>
```

## JS layout CSS funkcije

JS funkcije	Objašnjenje	Vrednosti	Objašnjenje vrednosti	E>	F	O
clip	Seče slike	rect(top, right, bottom, left)	Definiše vidljivi deo	4	1	9
		auto	Po defaultu			
cursor	Definiše izgled kursora	URL	Url slika kursora			
		default	Default kursor (strelica)			
		auto	Brauzer postavlja kursor			
		crosshair	U obliku znaka +			
		pointer	U obliku ruke			
		move	U obliku + a ima i strelice na krajevima			
		e-resize	Minus sa strelicama na krajevima			
		ne-resize	Linija po dijagonali sa strelicama na krajevima			
		nw-resize	isto kao gore samo suprotna dijagonala			
		n-resize	Uspravna linija sa strelicama na krajevima			
		se-resize	isto kao nw-resize			
		sw-resize	isto kao ne-resize			

		s-resize	isto kao n-resize			
		w-resize	isto kao e-resize			
		wait	Peščani sat			
		help	Kursor sa znakom pitanja			
display	Definiše kako će se prikazivati element	none	Element neće biti prikazan. Sakriće se			
		block	Element će biti prikazan kao blok (<div>, <p> itd.)			
		inline	Element će biti prikazan kao inline (<span>, <a>)			
		list-item	Element će biti prikazan kao lista			
		run-in	Element će biti prikazan kao blok ili inline u zavisnosti od konteksta			
		compact	Element će biti prikazan kao blok ili inline da bi se zauzeo što manji prostor			
		marker	Slično kao inline			
		table	Element će biti prikazan kao tablica			
		inline-table	Element će biti prikazan kao inline tablica			
		table-row-group	Element će biti prikazan kao grupa od jednog ili više reda			
		table-header-group	Element će biti prikazan kao zaglavlje tablice			
		table-footer-group	Element će biti prikazan kao podnožje tablice			
		table-row	Element će biti prikazan kao red tablice			
		table-column-group	Element će biti prikazan kao grupa od jednog ili više stupaca			
		table-column	Element će biti prikazan kao stupac ćelija			
		table-cell	element će biti prikazan kao ćelija			
		table-caption	Element će se prikazati kao naslov tablice			
height width minHeight maxHeight minWidth maxWidth	Definiše visinu elementa	auto	Brauzer postavlja visinu	4M	1	9
		visina u pikselima	Definiše se u pikselima, milimetrima i ostalim CSS mernim veličinama			
		visina u %	U procentima u odnosu na visinu bloka u kome se nalazi element			
overflow	Definiše radnje ako sadržaj elementa ne stane u okvir	auto	Brauzer sam određuje radnje	4	1	9
		scroll	Pojavljuje se klizač			

		visible	Proširuje se prostor da bi se sve vidlo			
		hidden	Vidi se samo koliki je prostor a ostatak se ne vidi			
verticalAlign	Definiše vertikalno poravnanje elementa	baseline	Element je postavljen u nivou kao i nadređeni element	4	1	No
		sub	poravnava se kao što je napisan			
		super	poravnava kao natpis			
		top	Poravnava pri vrhu			
		tex-top	Vrh elementa je uskladjen sa vrhom nadređenog elementa			
		middle	Element se nalazi u sredini nadređenog elementa			
		bottom	Element se nalazi u donjem delu nadređenog elementa			
		tex-bottom	Dno elementa je uskladjen sa dna nadređenog elementa			
		dužina u px	Položaj se određuje u pikselima			
		dužina u %	Položaj se određuje u procentima			
visibility	Definiše vidljivost elementa	visible	Po defaultu je element vidljiv	4	1	9
		hidden	Element nije vidljiv ali prostor koji zauzima je zauzet			
		collapse	Element nije vidljiv ali i prostor koji zauzima je slobodan			

Daću samo jedan primer ali vama ako nisu jasna objašnjenja eksperimentišite malo ili odite u CSS delu koji sve pojedinačno objašnjava.

```
<html><head>
<script type="text/javascript">
function funkcija(){
    document.body.style.cursor="help"
}
</script>
</head>
<body onload = funkcija()>
</body></html>
```

## JS list CSS funkcije

JS funkcije	Objašnjenje	Vrednosti	Objašnjenje vrednosti	E	F	O
listStyle	Odredjuje tip markera liste	none	bez markera	4	1	9
		disc	puni kružići			



		circle	prazni kružići	ne	4	
		square	puni kvadratići			
		decimal	na primer 1 2 3...			
		decimal-leading-zero	na primer 01 02 03 ...			
		lower-roman	na primer i ii iii iv	4		
		upper-roman	na primer I II III IV			
		lower-alpha	na primer a b c..			
		upper-alpha	A B C ...			
		lower-greek	grčka slova	ne		
		lower-latin	na primer a b c ...			
		upper-latin	A B C ...			
		hebrew	hebrejski brojevi			
		armenian	armenijski brojevi			
		georgian	gruzijski brojevi			
		cjk-ideographic	cjk simboli			
		hiragana	vijetnamski			
		katakana	indonežanski			
		hiragana-iroha	vijetnamski			
		katakana-iroha	indonežanski			
	Lista je i onako uvučena od ostalog teksta ali može biti i još uvučenija	outside	uvučena normalno	4	1	9
		inside	još više uvučena			
	Odredjuje sliku markera	url(URL)	adresa slike	4	1	ne

Evo primera:

```
<html><head>
<script type="text/javascript">
function lista(){
    document.getElementById("ul").style.listStyle="decimal inside";
}
</script>
</head><body>
<ul id="ul">
    <li>Coffee</li>
    <li>Tea</li>
    <li>Water</li>
    <li>Soda</li>
</ul>
<input type="button" onclick="lista()" value="Promeni izgled liste">
</body></html>
```

## JS CSS funkcije za poziciju

JS funkcije	Objašnjenje	Vrednosti	Objašnjenje vrednosti	E	F	O
bottom top left right	Definiše poziciju elementa u odnosu na prostor u kome se nalazi	auto	Po defaultu	4	1	9
		%	Pozicija se određuje procentualno od donje (bottom), gornje (top), levu (left) i desnu (right) ivicu prostora a u odnosu na veličinu prostora			
		pozicija u px, ili mm itd	Sve isto kao % ali samo što se određuje u px ili ostalim CSS mernim jedinicama			
position	Definiše tip pozicije	static	Po defaultu			
		relative	Pozicija se određuje u zavisnosti od defaultne pozicije			
		absolute	Pozicija se određuje u zavisnosti od nulte pozicije na strani, tom jest od gornjeg kranjeg levog ugla			
		fixed	Odredjuje se kao absolute, ali ostaje fiksna pozicija na ekranu bez obzira što mi pomeramo i klizač			
zIndex	Definiše vidljivos elementa jednog u odnosu na druge	auto	Po defaultu. Onaj element koji je zadnji kreiran on će se i videti.			
		broj	Definišemo koji element će biti vidljiv a koji u pozadini. Vidite detaljnije u CSS edukaciji.			

```

<html><head>
<script type="text/javascript">
function promena() {
    document.getElementById("img1").style.zIndex="-1";
}
</script>
</head><body>
<h1>Pozicija i vidljivost slike</h1>

<script type="text/javascript">
document.getElementById("img1").style.position="absolute"
document.getElementById("img1").style.left="50px";
document.getElementById("img1").style.top="0px";
</script>
<p>Po defoltu z-index je 0. Z-index -1 ima prioritet.</p>
<br><br><br><input type="button" onclick="promena()" value="Povećaj
vidljivost teksta">
</body></html>

```

## JS tekstualne CSS funkcije

JS funkcije	Objašnjenje	Vrednosti	Objašnjenje vrednosti	E	F	O
color	definiše boju teksta	ime	Ime boje, na pr. red, blue black itd	4	1	9
		rgb(1,1,1)	Boja u rgb zapisu			
		#aaaaaa	Boja u heksazapisu			

fontFamily	definiše izgled slova	font	Dfiniše font, na primer arial, itd.
fontSize	definiše veličinu slova	xx-small	Definiše veličinu slova pomoću rezervisanih reči
		x-small	
		small	
		medium	
		large	
		x-large	
		xx-large	
		smaller	Postavlja manja slova u odnosu na nadredjeni element
		larger	Postavlja veća slova u odnosu na nadredjeni element
		veličina u px	Definiše tačno odredjenju veličinu slova u pikselima, milimetrima i drugim CSS jedinicama mere
fontStyle	Definiše stil slova	%	Definiše u procentima u odnosu na veličinu elementa u kome je tekst
		none	Po defaultu
		normal	Po defaultu
		italic	Malo iskošena slova
fontVariant	Definiše varijantu slova	oblique	sto kao i italic
		normal	Po defaultu
		small-caps	Sva slova pretvara u velika slova
fontWeight	Definiše debljinu slova	normal	Po defaultu
		lighter	Tanka
		bold	Podebljana
		bolder	Podebljana
		100	Debljina definisana brojem
		200	
		300	
		400	
		500	
		600	
		700	
		800	
		900	
letterSpacing	Definiše rastojanje izmedju slova u tekstu	normal	Po defaultu
		broj	Na primer -1,-2 itd što znači da će se slova skupljati a ako je pozitivan broj onda će se slova udaljavati

lineHeight	Definiše rastojanje izmedju redova teksta	normal	Po defaultu			
		broj	Rastojanje se definiše kad se broj pomnoži sa veličinom fonta			
		dužina u pikselima na primer	Na primer fiksno u pikselima ili drugim CSS veličinama			
		%	Procentualno u odnosu na velučinu fonta slova			
textAlign	Definiše kako će tekst biti poravnan	left	Biće poravnan prema levoj liniji polja u kome je smešten			
		right	desno			
		center	centrirano			
		justify	poravnan sa svih strana u liniju kao novinski tekst			
textDecoration	Dekoracija teksta	none	po defaultu			
		underline	podvučen tekst			
		overline	nadvučen tekst			
		line-through	precrtan tekst			
		blink	tekst koji treperi			
textIndent	Definiše rastojanje teksta od leve ivice polja u kome se nalazi tekst	dužina u px	Fiksno definisano u pikselima ili u nekim drugim CSS jedinicama			
		%	Procentualno u odnosu na horizontalnu veličinu polja u kome se nalazi tekst			
textTransform	Transformiše slova u velika ili mala itd	none	Po defaultu			
		capitalize	prvi znak svake reči se pretvara u veliko slovo			
		uppercase	sva slova u tekstu se pretvaraju u velika slova			
		lowercase	sva slova u tekstu se pretvaraju u mala slova			
whiteSpace	Definiše lomljenje redova	normal	po defaultu			
		nowrap	ne lomi redove teksta u novi red ukoliko se naidje na kraj ekrana nego pravi vodoravni klizač.			
		pre	Prikazuje se tekst baš onako kako je napisan u HTML kodu			
wordSpacing	Definiše rastojanje izmedju reči u tekstu	normal	Po defaultu			
		dužina	Definiše razmak u pikselima, milimetrima ili ostalim CSS jedinicama mere			

Primer:

```

<html><head>
<style type=text/css>
#aa {
    background-color:#FF9966;
    width: 500px;
    height: 50px;
    border: 5px solid gray;
    padding-top: 10px;
}
</style>
<script type="text/javascript">
c=0;
t=0;
function startuj(){
    document.getElementById("aa").style.textIndent = c;
    if (c==435) c=1
    c=c+1;
    t = setTimeout("startuj()",1);
}
</script>
</head><body>
<input type="button" value="Startuj" onClick="startuj()">
<p id="aa"><b>DEJAN</b></p>
<input type="button" value="Stopiraj" onClick="clearTimeout(t)">
</body></html>

```

## CANVAS

CANVAS (<canvas>) je HTML tag koji se pojavio u HTML5. CANVAS se koristi za crtanje grafike pomoću JavaScript. Može se koristiti na primer za crtanje grafova, foto-kompozicija, ili praviti animacije.

Canvas je prvi put predstavljen u Apple na Mac OS X, a kasnije je implementiran na brauzer Safari 1.3. Takodje ovaj tag podržavaju i Firefox 1.5, Internet Explorer 6, i Opera 9.

Canvas stvara crteže čiju površinu popunjavamo sa jednim ili više renderanim kontekstima. Renderiranje se temelji na OpenGL-u.

U ovom delu ću pokušati da opišem kako ugraditi CANVAS element u HTML stranicu. Takodje ću dati i primere koji bi trebalo da opišu neke ideje šta možete učiniti sa CANVAS.

## CANVAS ugradnja u HTML

CANVAS tag u HTML dokumentu izgleda ovako:

```

<canvas id="graf" width="150" height="150"></canvas>

```

CANVAS kao što vidimo ima samo dva atributa, širinu WIDTH i visinu HEIGHT izraženu u pikselima. Medjutim i širina i visina su opcioni parametri. U ovom našem slučaju platno za crtanje je dimenzije 150 X 150 piksela, a ako nije definisana širina i visina platno je po defaultu dimenzije 300 X 150 piksela.

Obzirom da CANVAS tag podržavaju samo noviji preglednici mi moramo definisati i alternativni sadržaj za te starije preglednike. Na primer možemo pisati unutar CANVAS taga bilo koji HTML sadržaj. Naravno preglednici koji podržavaju CANVAS će ignorisati HTML kod unutar početnog i krajnjeg CANVAS taga. Evo kako na primer možemo pisati za brauzere koji ne podržavaju CANVAS:

```
<canvas id="graf" width="150" height="150">
ovde stoji crtež grafike ali vaš brauzer je star i ne podržava prikaz.
</canvas>
```

Ili:

```
<canvas id="graf" width="150" height="150">

</canvas>
```

OVAKO napisano, preglednici koji podržavaju CANVAS neće prikazati sliku [starBrauzer.png](#) a ni onaj gornji tekst. Da bi brauzeri koji podržavaju CANVAS nešto prikazali treba definisati programibilnu sliku pomoću JavaScript funkcije `getContext ("2d" )`. na primer ovako:

```
<html><head>
<script type="text/javascript">
function crtaj(){
    canvas = document.getElementById("graf");
    if (canvas.getContext){
        ctx = canvas.getContext("2d");
    }
}
</script>
<style type="text/css">
canvas {
border: 1px solid black;
}
</style>
</head><body onload=crtaj()>
<canvas id="graf" width="150" height="150"></canvas>
</body></html>
```

Naravno ovako napisan kod u brauzere koji podržavaju CANVAS daće samo prazno uokvireno platno jer još nismo ništa nacrtali. U ovom kodu takodje primetite vrlo bitnu stvar a to je da se funkcija za crtanje poziva tek nakon učitavanja dokumenta u brauzer. Ako koristimo na primer `setTimeout( )` ili `setInterval( )` možemo prikazivati i neke animacije, itd itd. Na kraju ću dati linkove u kojima možete videti složenije CANVAS animacije, crteže, grafikone, itd.

## Crtanje pravougaonika

Evo jednog jednostavnog primera koji crta dva kvadrata koji se preklapaju:

```
<html><head>
<script type="application/javascript">
function crtaj() {
    canvas = document.getElementById("kocke");
    if (canvas.getContext) {
        ctx = canvas.getContext("2d");
```

```

        ctx.fillStyle = "rgb(200,0,0)";
        ctx.fillRect (10, 10, 55, 50);
        ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
        ctx.fillRect (30, 30, 55, 50);
    }
}
</script>
</head><body onload="crtaj();">
<canvas id="kocke" width="150" height="150"></canvas>
</body></html>

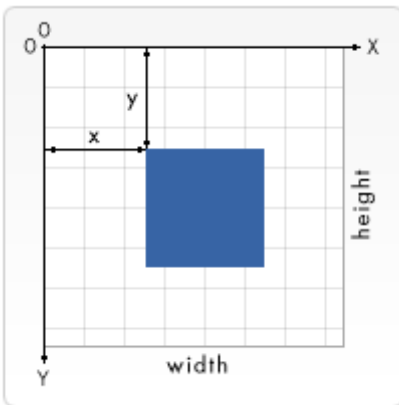
```

Ovde smo crtali dva popunjena i obojena pravougaonika. Postoje tri funkcije koje crtaju pravougaonik:

- **fillRect(x, y, dužina, visina)** - crta ispunjen pravougaonik
- **strokeRect(x, y, dužina, visina)** - crta samo konture pravougaonika
- **clearRect(x, y, dužina, visina)** - briše sve nacrtano unutar kontura ovako definisanog pravougaonika

X i Y su koordinate gornjeg levog temena pravougaonika, a dužina je dužina pravougaonika a visina je visina.

Pre nego krenemo dalje da damo prikaz kako se definiše canvas i kako je postavljen koordinatni sistem, i kako izgleda definicija pravougaonika unutar canvas-a:



Evo primera sa sve tri funkcije gornje:

```

<html><head>
<script type="application/javascript">
function crtaj() {
    canvas = document.getElementById("kocke");
    if (canvas.getContext) {
        ctx = canvas.getContext("2d");
        ctx.fillRect(25,25,100,100);
        ctx.clearRect(45,45,60,60);
        ctx.strokeRect(50,50,50,50);
    }
}
</script>

```

```
</head><body onload="crtaj();">
<canvas id="kocke" width="150" height="150"></canvas>
</body></html>
```

Medjutim kada crtanje pravuganika kombinujemo sa drugim geometrijskim figurama kako bi nacrtali složene crteže koristiti se funkcija:

```
rect(x, y, width, height)
```

Ali `rect( )` se koristi samo u kombinaciji sa funkcijama `stroke()` i `fill()` koje će mo objasniti odma u sledećem naslovu:

## Crtanje raznih oblika

Svaki složeniji crtež mora da sadrži sledeće funkcije:

- `beginPath()`
- `closePath()`
- `stroke()`
- `fill()`

Kao što znamo složeniji crteži se sastoje od raznih geometrijskih oblika, kao što su linije, pravougaonici, krugovi, itd.

**BeginPath( )** je zapravo kontejner koji sadrži definicije svih geometrijskih oblika koji skupa definišu jedan crtež.

**ClosePath( )** zatvara formu crteža. Nije uvek potrebno koristiti `closePath` ali da mnogo ne komplikujemo bolje je uvek koristiti.

**Stroke( )** je funkcija koja se koristi za crtanje kontura, a **fill( )** za crtanje ispunjenih površina. Evo jednog crteža trougla koji demonstrira upotrebu `beginPath`:

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    ctx.beginPath();
    ctx.moveTo(75,50);
    ctx.lineTo(100,75);
    ctx.lineTo(100,25);
    ctx.fill();
  }
}
```

**moveTo( )** je vrlo korisna funkcija, koja zapravo ne definiše nikakv geometrijski oblik, ali može da pomera sve moguće oblike. Princip rada je takav, da na primer imate olovku na stolu, i sad je vi



uzmete i pomerite je sa jednog mesta na drugo. Znači, kad definišemo neki oblik pomoću `beginPath()` funkcijom i kad se pozove ta funkcija onda se crtež crta u koordinati (0, 0). Medjutim mi možemo pomoću `moveTo()` funkcijom da pomeramo taj crtež u koju god želimo koordinatu unutar našeg platna za crtanje. To je jedan korisan način upotrebe ali mi možemo pomerati ne samo ceo crtež, već i pojedine geometrijske oblike unutar same definicije nekog crteža. Evo primera:

```
function crtaj() {
    canvas = document.getElementById("kocke");
    if (canvas.getContext) {
        ctx = canvas.getContext("2d");
        ctx.beginPath();
        ctx.arc(75,75,50,0,Math.PI*2,true); // definisanje spoljnog celog
kruga

        ctx.moveTo(110,75);
        ctx.arc(75,75,35,0,Math.PI,false); // definisanje usta
        ctx.moveTo(65,65);
        ctx.arc(60,65,5,0,Math.PI*2,true); // levo oko
        ctx.moveTo(95,65);
        ctx.arc(90,65,5,0,Math.PI*2,true); // desno oko
        ctx.stroke();
    }
}
```

**lineTo(x, y)** - Funkcija za crtanje linije. Kao što znamo, da bi smo nacrtali liniju potrebno je definisati dve tačke. Funkcija `lineTo()` definiše jednu tačku a druga tačka je koordinatni početak. E sad ako želimo liniju koja ne prolazi kroz koordinatni početak onda koristimo funkciju `moveTo()` za izmeštanje koordinatnog početka. Evo primer korišćenja linija u crtanju dva trougla od kojih je jedan isunjen a drugi samo kontura:

```
function crtaj() {
    canvas = document.getElementById("kocke");
    if (canvas.getContext) {
        ctx = canvas.getContext("2d");
        // Ispunjen trougao
        ctx.beginPath();
        ctx.moveTo(25,25);
        ctx.lineTo(105,25);
        ctx.lineTo(25,105);
        ctx.fill();
        // Kontura trougao
        ctx.beginPath();
        ctx.moveTo(125,125);
        ctx.lineTo(125,45);
        ctx.lineTo(45,125);
        ctx.closePath();
        ctx.stroke();
    }
}
```

**arc(x, y, poluprečnik, početakKrug, krajKrug, smer)** - funkcija za crtanje krugova i isečaka krugova (lukova). X i Y su koordinate tačke centra kruga. Smer crtanja luka je suprotan od smera kretanja na satu ako je **true**, a ako je **false** onda je u smeru kretanja kazaljke na satu. PočetakKrug i KrajKrug su parametri koji definišu početak i kraj luka kruga a mere se u radijanima. Da biste pretvorili uglove u stepenima u uglove u radijanima koristite sledeći JavaScript izraz:

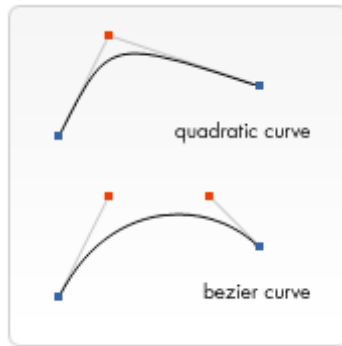
```
radijani = (Math.PI/180) * stepeni
```

Evo primera za više slučajeva krugova:

```
function crtaj() {
canvas = document.getElementById("kocke");
if (canvas.getContext) {
  ctx = canvas.getContext("2d");
  for(var i=0; i<4; i++){
    for(var j=0; j<3; j++){
      ctx.beginPath();
      x = 25 + j*50; // x coordinate
      y = 25 + i*50; // y coordinate
      radius = 20; // Arc radius
      startAngle = 0; // Starting point on circle
      endAngle = Math.PI+(Math.PI*j)/2; // End point on circle
      anticlockwise = i%2==0 ? false : true; // clockwise or anticlockwise
      ctx.arc(x,y,radius,startAngle,endAngle, anticlockwise);
      if (i>1){
        ctx.fill();
      } else {
        ctx.stroke();
      }
    }
  }
}
```

## Bezijerove krivulje

Bezierova krivulja je parametrična krivulja važna u području matematičke numeričke analize, a naročito se primjenjuje u 3D vektorskoj grafici, Bézierove krivulje su važan alat, kojim se služi većina grafičkih programa pri oblikovanju glatkih krivulja, koje se mogu beskonačno skalirati. Svi programi za crtanje i uređivanje slika poput; Adobe Illustratora, Adobe Photoshopa, CorelDrawa služe se "Putanjama" ( engleski: "Paths"), a one su kombinacija Bézierovih krivulja. Bézierove krivulje su široko rasprostanjene i u programima za animaciju poput; Adobe Flasha, Adobe After itd.



Mi će mo ovde te krivulje koristiti da crtamo razne slike. Funkcije su sledeće:

```
quadraticCurveTo(cpx, cpy, x, y)
bezierCurveTo(cpx, cpy, cp2x, cp2y, x, y)
```

X i Y parametri u obe metode su koordinate tačke na kraju. **cp1x** i **cp1y** su koordinate prve kontrolne tačke, a **cp2x** i **cp2y** su koordinate druge kontrolne tačke. Korišćenje kvadratične i kubične Bézierove krivulje može biti prilično zahtevno, jer za razliku od vektorskog crtačkog softvera (poput Adobe Illustrator-a) mi nemamo direktne vizualne povratne informacije o tome šta radimo. To čini prilično teško crtati složene oblike. U sledećem primeru, mi ćemo crtati neke jednostavne oblike, ali ako Vi imate vremena i, najviše od svega, strpljenja, mnogo složenije crteže bi ste mogu kreirati.

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    ctx.beginPath();
    ctx.moveTo(75,25);
    ctx.quadraticCurveTo(25,25,25,62.5);
    ctx.quadraticCurveTo(25,100,50,100);
    ctx.quadraticCurveTo(50,120,30,125);
    ctx.quadraticCurveTo(60,120,65,100);
    ctx.quadraticCurveTo(125,100,125,62.5);
    ctx.quadraticCurveTo(125,25,75,25);
    ctx.stroke();
  }
}
```

Ili na primer ovako nešto:

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    ctx.beginPath();
    ctx.moveTo(75,40);
    ctx.bezierCurveTo(75,37,70,25,50,25);
    ctx.bezierCurveTo(20,25,20,62.5,20,62.5);
    ctx.bezierCurveTo(20,80,40,102,75,120);
    ctx.bezierCurveTo(110,102,130,80,130,62.5);
    ctx.bezierCurveTo(130,62.5,130,25,100,25);
  }
}
```

```
    ctx.bezierCurveTo(85,25,75,37,75,40);
    ctx.fill();
  }
}
```

## Klasične slike

U prostor u kome smo crali pomoću CANVES možemo uvoziti i gotove slike, koje mogu biti kao pozadina našim crtežima, ili pak da pravimo neke dinamičke animacije, itd. Možemo ubacivati bilo koje tipove slika, na primer .jpg, .gif, .png itd.

Ubacivanje slika se vrši u osnovi u dva koraka:

- Pravimo referencu na objekat JavaScript slike ili druge CANVAS elemente kao izvor. Nije moguće jednostavno ubacivanjem URL adrese slika.
- Zatim "crtamo" sliku na platno pomoću **drawImage** funkcije.

**drawImage(image, x, y)** - kao što smo već napisali ovom funkcijom ubacujemo sliku, stim što su X i Y koordinate koje definišu položaj slike i to tako da definišu gornju levu tačku slike. Zapravo drawImage() funkcija ima još dve varijante ali ovo je prva. Evo primera:

```
function crtaj() {
  var ctx = document.getElementById("kocke").getContext("2d");
  var img = new Image();
  img.onload = function() {
    ctx.drawImage(img, 0, 0);
    ctx.beginPath();
    ctx.moveTo(30, 96);
    ctx.lineTo(70, 66);
    ctx.lineTo(103, 76);
    ctx.lineTo(170, 15);
    ctx.stroke();
  }
  img.src = "jsslikeplanete/mars.jpg";
}
```

Medjutim ne moramo definisati sliku u samoj funkciji **img.src="..."**. Ako već u dokumentu postoji slika koju želimo da iskoristimo možemo je pozvati i ovako:

```
function crtaj() {
  var ctx = document.getElementById("kocke").getContext("2d");
  ctx.drawImage(document.getElementById("img1"), 0, 0);
  ctx.beginPath();
  ctx.moveTo(30, 96);
  ctx.lineTo(70, 66);
  ctx.lineTo(103, 76);
  ctx.lineTo(170, 15);
}
```

```
        ctx.stroke();  
    }  
}
```

Druga varijanta funkcije nam dodaje još dva nova parametra koja omogućavaju skaliranje slike, to jest podešavanje veličine slike koja nama odgovara. Funkcija izgleda:

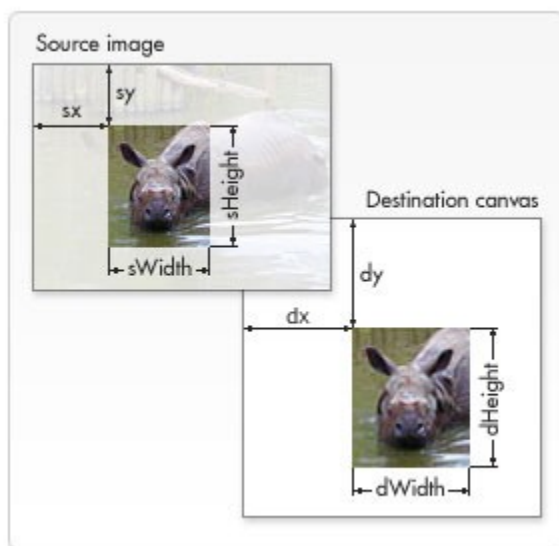
```
drawImage(image, x, y, width, height)
```

Gde su **width** širina slike koju mi želimo, i **height** visina koja nama odgovara, a sve u pikselima naravno. Mislim da je ovo jasno i da ne treba neki poseban primer, jer je sve ostalo isto kao i prethodni primer.

Treća varijanta **drawImage** funkcije je mogućnost sečenja slike. Naime, nekad je neophodno koristiti samo neki deo slike. Funkcija izgleda sad ovako:

```
drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)
```

**sx**, **sy**, **sWidth** i **sHeight** je pravougaonik isečka slike. Znači **sx** i **sy** su kordinate tačke u slici, a koordinatni početak u slici je sam gornji levi ugao slike. **dx**, **dy**, **dWidth** i **dHeight** su u stvari oni naši poznati parametri kao iz gornje dve varijante **drawImage** funkcije. Ovako napisano moguće da je nejasno ali daću sliku koja objašnjava:



Source image je slika koju sečemo, a destinacion canvas je postavljanje tog isečka slike na naše platno. Pošto je ovo verujem jasno a kod je identičan kao u prvoj varijanti drawImage funkcije, onda mislim da nije potreban primer.

## Stilovi i boje

Do sada smo linije crteža, kao i ispune crtali samo bojom koja je po defaultu, a to je crna boja. Ovde će mo videti kako se koriste i sve ostale moguće boje. Postoje dve funkcije za boje a to su:

- **fillStyle = color** - definiše boju ispune
- **strokeStyle = color** - definiše boju kontura

Color je definicija boje a ona može da se definiše na više načina:

```
ctx.fillStyle = "orange";
ctx.fillStyle = "#FFA500";
ctx.fillStyle = "rgb(255,165,0)";
ctx.fillStyle = "rgba(255,165,0,1)";
```

Svi ovi načini su nam poznati, sem zadnjeg **rgba** pa da ga objasnimo. Zapravo je zadnji parametar nepoznanica a ostala tri su definicije boja kao kod **rgb**. Zadnji parametar definiše nivo transparentnosti boje, odnosno nivo providnosti. Ako se stavi broj 1 onda nije providna boja a što se broj više smanjuje do 0 sve je transparentniji. Tako na primer 0.5 je upola providan.

Ako postavite **strokeStyle** i / ili **fillStyle** na kraju onda tako definišete boju za sve linije i površine. Ako neki oblik želite u drugoj boji, onda se **fillStyle** ili **strokeStyle** definiše odmah u narednom redu u odnosu na red u koji ste definisali oblik. Evo nekoliko primera:

Primer1.

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    for (i=0; i<6; i++){
      for (j=0; j<6; j++){
        ctx.fillStyle="rgb(" + Math.floor(255-42.5*i) + "," +
Math.floor(255-42.5*j) + ",0)";
        ctx.fillRect(j*25,i*25,25,25);
      }
    }
  }
}
```

Primer2.

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    for (i=0; i<6; i++){
      for (j=0; j<6; j++){
        ctx.strokeStyle="rgb(0," + Math.floor(255-42.5*i) + "," +
Math.floor(255-42.5*j) + ")";
        ctx.beginPath();
        ctx.arc(12.5+j*25,12.5+i*25,10,0,Math.PI*2,true);
        ctx.stroke();
      }
    }
  }
}
```

```
}  
}  
}  
}
```

## Transparentnost boja

Kao što je već pisano providnost (transparentnost) boja može se praviti pomoću **rgba** definicije za boje, pomoću njenog zadnjeg parametra, koji se kreće od 0 - boja se i neće videti, 0.5 - polutransparentna boja i 1 - kad boja nije nimalo transparentna. Međutim ako su sve boje transparentne podjednako onda ne moramo svuda koristiti **rgba**, već možemo **rgb**, a transparentnost definisati kao globalnu pomoću sledeće funkcije:

```
globalAlpha = vrednostTransparentnosti
```

`vrednostTransparentnosti` se naravno kreće od 0 do 1, sve isto kao kad se definiše transparentnost u zadnjem parametru **rgba** definicije boje. Evo primera:

```
function crtaj() {  
    canvas = document.getElementById("kocke");  
    if (canvas.getContext) {  
        ctx = canvas.getContext("2d");  
        // Crtanje pozadine  
        ctx.fillStyle = "#FD0";  
        ctx.fillRect(0,0,75,75);  
        ctx.fillStyle = "#6C0";  
        ctx.fillRect(75,0,75,75);  
        ctx.fillStyle = "#09F";  
        ctx.fillRect(0,75,75,75);  
        ctx.fillStyle = "#F30";  
        ctx.fillRect(75,75,150,150);  
        ctx.fillStyle = "#FFF";  
        // Definisanje transparentnosti  
        ctx.globalAlpha = 0.2;  
        // Crtanje polutransparentnih krugova  
        for (var i=0;i<7;i++){  
            ctx.beginPath();  
            ctx.arc(75,75,10+10*i,0,Math.PI*2,true);  
            ctx.fill();  
        }  
    }  
}
```

## Stil linija

Postoje sledeće funkcije koje omogućavaju razne stilove linija:

- `lineWidth` = vrednost
- `lineCap` = tip
- `lineJoin` = tip
- `miterLimit` = vrednost

**lineWidth** - definiše debljinu linija. Vrednost mora biti pozitivna, a po defaultu je 1. Evo jednog primera:

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    for (i = 0; i < 10; i++){
      ctx.lineWidth = 1+i;
      ctx.beginPath();
      ctx.moveTo(5+i*14,5);
      ctx.lineTo(5+i*14,140);
      ctx.stroke();
    }
  }
}
```

**lineCap** - definiše krajeve linija. Tipovi mogu biti: **butt**, **round** i **square**. Da ne bih sad mnogo pisao objašnjenje evo primera koji daje sliku ova tri oblika linija:

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    lineCap = ["butt", "round", "square"];
    // Crtanje pomoćnih linija za objašnjenje
    ctx.strokeStyle = "#09f";
    ctx.beginPath();
    ctx.moveTo(10,10);
    ctx.lineTo(140,10);
    ctx.moveTo(10,140);
    ctx.lineTo(140,140);
    ctx.stroke();
    // Crtanje linija
    ctx.strokeStyle = "black";
    for (i=0; i<lineCap.length; i++){
      ctx.lineWidth = 15;
      ctx.lineCap = lineCap[i];
```



```

        ctx.beginPath();
        ctx.moveTo(25+i*50,10);
        ctx.lineTo(25+i*50,140);
        ctx.stroke();
    }
}
}

```

**lineJoin** - definiše u kom obliku povezati dve linije. Postoje tri tipa: **round** (okruglina), **bevel**(kosina), i **miter** (kapa). Po defaultu je miter. Evo primera koji sve objašnjava (obratite pažnju na ivice linija gde se spajaju):

```

function crtaj() {
    canvas = document.getElementById("kocke");
    if (canvas.getContext) {
        ctx = canvas.getContext("2d");
        lineJoin = ["round", "bevel", "miter"];
        ctx.lineWidth = 10;
        for (i =0; i < lineJoin.length; i++){
            ctx.lineJoin = lineJoin[i];
            ctx.beginPath();
            ctx.moveTo(-5,5+i*40);
            ctx.lineTo(35,45+i*40);
            ctx.lineTo(75,5+i*40);
            ctx.lineTo(115,45+i*40);
            ctx.lineTo(155,5+i*40);
            ctx.stroke();
        }
    }
}

```

## Nijansiranje boja

Ovde imamo dve funkcije na raspolaganju:

- **createLinearGradient (x1,y1,x2,y2)** - za linearno gradiranje boje (postepeno) od tačke x1, y1 do tačke x2, y2
- **createRadialGradient (x1,y1,r1,x2,y2,r2)** - za kružno gradiranje od kružnice iz tačke x1, y1 sa poluprečnikom r1, do kružnice u tački x2, y2 sa poluprečnikom r2. Shvatite ovo kružno kao prsten a prsten se gradira bojom.

Medjutim ako mi samo ovako postavimo ove funkcije, ništa se neće desiti jer one samo definišu površinu gde će se vršiti nijansiranje. Moramo sad definisati i koja će biti početna boja a koja krajnja kako bi izmedju nastalo nijansiranje. Početna i krajnja boja se definišu funkcijom:

**addColorStop (pozicija, color)** - color je boja na primer u rgb ili rgba ili hekza zapisu.

E sad obe boje izmedju kojih će se praviti prelaz se definišu istom funkcijom `addColorStop( )`, stim što pozicija određuje da li će se od vrha ka dnu krenuti sa prvom bojom ili sa drugom, odnosno ako je u pitanju radijalni gradijent, da li će se sa prvom bojom krenuti sa unutrašnjosti ili sa spolja ka unutra. Zapravo pozicija može imati vrednosti od 0 do 1 a kad izmedju boja menjate 0 i 1 menjaju se i pozicije koje sam gore objašnjavao. E sad ako stavimo na primer jednu 0.5 a drugu 1 onda će se nijansirati ali malo više u korist one sa 1 itd itd. Malo eksperimentišite sa narednim primerima i videćete o čemu se radi.

Medjutim ako iskoristimo sve dosadašnje funkcije kako treba, i dalje se ništa neće prikazivati na stranici. Mi smo do sada zapravo definisali vrednost za stil bojenja. Znači možemo definisati sledeće korišćenjem naše dobro poznate promenljive `ctx`:

```
nijansiranje = ctx.createLinearGradient(0,0,0,600);
nijansiranje.addColorStop(0, "#00ABEB");
nijansiranje.addColorStop(0.5, "#fff");
ctx.fillStyle = nijansiranje;
```

Kao što vidite sve ovo smo iskoristili kako bi smo definisali jedan stil. E sad taj stil možemo koristiti u funkcijama kako nam je volja, na primer u kreiranju neke površine pravougaone. Evo kompletnog takvog primera:

```
<html><head>
<script type="application/javascript">
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    nijansiranje = ctx.createLinearGradient(0,0,0,600);
    nijansiranje.addColorStop(0, "#00ABEB");
    nijansiranje.addColorStop(1, "#fff");
    ctx.fillStyle = nijansiranje;
    ctx.fillRect(0,0,600,600);
  }
}
</script>
</head><body onload="crtaj();">
<canvas id="kocke" width="600" height="600"></canvas>
</body></html>
```

Mislim da je ovaj primer prilično jasan, kao i sve ove nove funkcije. Evo još jedan primer ali vezan za radijalni gradijent boje:

```
<html><head>
<script type="application/javascript">
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    // Definiše prelaze boja
```

```

radgrad = ctx.createRadialGradient(45,45,10,52,50,30);
radgrad.addColorStop(0, "#A7D30C");
radgrad.addColorStop(0.9, "#019F62");
radgrad.addColorStop(1, "rgba(1,159,98,0)");
radgrad2 = ctx.createRadialGradient(105,105,20,112,120,50);
radgrad2.addColorStop(0, "#FF5F98");
radgrad2.addColorStop(0.75, "#FF0188");
radgrad2.addColorStop(1, "rgba(255,1,136,0)");
radgrad3 = ctx.createRadialGradient(95,15,15,102,20,40);
radgrad3.addColorStop(0, "#00C9FF");
radgrad3.addColorStop(0.8, "#00B5E2");
radgrad3.addColorStop(1, "rgba(0,201,255,0)");
radgrad4 = ctx.createRadialGradient(0,150,50,0,140,90);
radgrad4.addColorStop(0, "#F4F201");
radgrad4.addColorStop(0.8, "#E4C700");
radgrad4.addColorStop(1, "rgba(228,199,0,0)");
// Crta oblike
ctx.fillStyle = radgrad4;
ctx.fillRect(0,0,150,150);
ctx.fillStyle = radgrad3;
ctx.fillRect(0,0,150,150);
ctx.fillStyle = radgrad2;
ctx.fillRect(0,0,150,150);
ctx.fillStyle = radgrad;
ctx.fillRect(0,0,150,150);
}
}
</script>
</head><body onload="crtaj();">
<canvas id="kocke" width="150" height="150"></canvas>
</body></html>

```

# Pattern

U ranijim nekim primerima koristili smo za kreiranje nekih oblika for petlje, ali postoji jednostavniji način korišćenjem sledeće funkcije:

## createPattern (image, type)

Ova funkcija uzima dva argumenta. IMAGE je ili referenca na neku sliku ili nešto što smo mi nacrtali uz pomoć "canves". TYPE je jedna od sledećih vrednosti: **repeat**, **repeat-x**, **repeat-y** i **no-repeat**. Već smo se susreli sa ovim vrednostima tako da ih nećemo objašnjavati, a verovatno sad i shvatate o čemu je sad reč. Evo i primera:

```

<html><head>
<script type="application/javascript">

```

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    // create new image object to use as pattern
    img = new Image();
    img.src = "jsslikeplanete/mars.jpg";
    img.onload = function(){
      // Kreiramo pattern
      ponavljanje = ctx.createPattern(img, "repeat");
      ctx.fillStyle = ponavljanje;
      ctx.fillRect(0,0,600,600);
    }
  }
}
</script>
</head><body onload="crtaj();">
<canvas id="kocke" width="500" height="500"></canvas>
</body></html>
```

# Senke

Za kreiranje senki se koriste sledeće funkcije:

```
shadowOffsetX = vrednost
shadowOffsetY = vrednost
shadowBlur = vrednost
shadowColor = color
```

**shadowOffsetX** i **shadowOffsetY** pokazuju koliko daleko treba proširiti senke od objekta u X i Y smerovima. Mogu se koristiti i negativne vrednosti i tad se senke proširuju prema gore ili ulevo, a pozitivne vrednosti uzrokuju senke prema dole ili udesno. Po defaultu su oba 0.

**shadowBlur** definiše veličinu efekta zamućenja. Vrednost nije u pikselima. Defaultna vrednost je 0.

**shadowColor** definiše boju senke, a boja se definiše klasično pomoću rgb, rgba, heksadecimalno ili navedemo ime boje.

Evo primera:

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    ctx.shadowOffsetX = -5;
    ctx.shadowOffsetY = -5;
    ctx.shadowBlur = 4;
    ctx.shadowColor = "rgba(0, 0, 0, 0.5)";
    ctx.font = "40px Times New Roman";
```

```
    ctx.fillStyle = "red";
    ctx.fillText("Zdravo ljudi, kakoste", 30, 60);
  }
}
```

## Saving i restoring stanja

Da bi smo mogli kreirati složenije crteže moramo odredjena stanja (pozicije crteža) spremati (uslovno rečeno "memorisati") i po nekad obnovljati. U narednim primerima će biti sve jasnije. Funkcije za to su:

```
save( )
restore( )
```

Odredjena CANVAS stanja možemo po želji memorisati `save( )` funkcijom, i takodje po želji možemo ih pozivati `restore( )` funkcijom kako bi smo ih ponovo promenili, i opet prikazali. Tako možemo napraviti vrlo složene crteže sa malo pisanja. Crtanje stanja sastoji se od:

- Transformacija (tj. translacije, rotacije i skaliranje - vidi u nastavku)
- Vrednosti `strokeStyle`, `fillStyle`, `globalAlpha`, `LineWidth`, `lineCap`, `lineJoin`, `miterLimit`, `shadowOffsetX`, `shadowOffsetY`, `shadowBlur`, `shadowColor`, `globalCompositeOperation` svojstva
- Trenutni clipping put, što ćemo videti u sledećem odeljku.

Evo jednog kraćeg primera koji demonstrira `save()` i `restore()` funkcije:

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    ctx.fillRect(0,0,150,150);
    // crtanje pravougaonika
    ctx.save();
    // učitavanje pošetnog stanja crteža
    ctx.fillStyle = "#09F"
    // vršimo nove promene istog crteža, ali imajte
    // na umu da smo prethodno stanje zapamtili
    ctx.fillRect(15,15,120,120);
    // crtamo pravougaonik sa novim dimenzijama
    ctx.save();
    // pamtimo nove promene
    ctx.fillStyle = "#FFF"
    // vršimo nove promene
    ctx.globalAlpha = 0.5;
    ctx.fillRect(30,30,90,90);
    // Crtamo novi pravougaonik
    ctx.restore();
    // vraćamo predjašnje stanje
    ctx.fillRect(45,45,60,60);
    // pravougaonik sa obnovljenim postavkama prepravljamo
    ctx.restore();
    // vraćamo predjašnje originalno stanje
    ctx.fillRect(60,60,30,30);
  }
}
```

```

        // pravougaonik sa obnovljenim originalnim
        // postavkama po drugi put prepravljamo
    }
}

```

Ovaj primer ilustruje kako učitavanjem stanja možemo manjim pisanjem nacrtati veći broj pravougaonika.

## Translacije

Translacija je jedan od metoda transformacija. Ova metoda se koristi za paralelno pomeranje crteža u različitim tačkama na platnu. Funkcija za to je:

```
translate(x, y)
```

Evo jednog primera transformacije translacije:

```

<html><head>
<script type="application/javascript">
function crtaj() {
    canvas = document.getElementById("kocke");
    ctx = canvas.getContext("2d");
    ctx.fillRect(0,0,300,300);
    for (i=0; i<3; i++) {
        for (j=0; j<3; j++) {
            ctx.save();
            ctx.strokeStyle = "#9CFF00";
            ctx.translate(50+j*100,50+i*100);
            crtajSpirograf(ctx,20*(j+2)/(j+1),-8*(i+3)/(i+1),10);
            ctx.restore();
        }
    }
}
function crtajSpirograf(ctx,R,r,O){
    var x1 = R-O;
    var y1 = 0;
    var i = 1;
    ctx.beginPath();
    ctx.moveTo(x1,y1);
    do {
        if (i>20000) break;
        var x2 = (R+r)*Math.cos(i*Math.PI/72) -
(r+O)*Math.cos((R+r)/r)*(i*Math.PI/72)
        var y2 = (R+r)*Math.sin(i*Math.PI/72) -
(r+O)*Math.sin((R+r)/r)*(i*Math.PI/72)
        ctx.lineTo(x2,y2);
        x1 = x2;
        y1 = y2;
        i++;
    } while (x2 != R-O && y2 != 0 );
    ctx.stroke();
}
</script>
</head><body onload="crtaj();">

```

```
<canvas id="kocke" width="600" height="600"></canvas>
</body></html>
```

## Rotacije

Druga metoda transformacija je rotacija. Ona se koristi kada postoji potreba da se neki crtež zarotira za neki ugao. Evo funkcije:

```
rotate(angle)
```

Evo jednog primera:

```
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    ctx.translate(75,75);
    for (i = 1; i < 6; i ++){ // crta prstenove (iznutra ka van)
      ctx.save();
      ctx.fillStyle="rgb("+ (51*i)+", "+ (255-51*i)+", 255)";
      for (j=0; j<i*6; j++){ // crta pojedinačne kružice
        ctx.rotate(Math.PI*2/(i*6));
        ctx.beginPath();
        ctx.arc(0,i*12.5,5,0,Math.PI*2,true);
        ctx.fill();
      }
      ctx.restore();
    }
  }
}
```

## Skaliranje

Jedna od metoda transformacija je i skaliranje. To je takva transformacija kojom se može povećavati ili smanjivati širina ili visina canvas slika, pa time možemo dobiti razne efekte izduživanja ili zbijanja slika, ili pak smanjenje ili povećanje canvas slika ako i visinu i širinu slika smanjujemo odnosno povećavamo proporcionalno.

Osnovna funkcija transformacije skaliranja je:

```
scale(x, y)
```

Ova metoda, kao što vidimo, uzima dva parametra. Parametar X je faktor u vodoravnom smeru a Y u vertikalnom. Oba parametra moraju biti pozitivni brojevi. Vrednosti manja od 1.0 smanjuje a vrednost veća od 1.0 povećava jedinicu veličine. Ako je vrednost X ili Y parametra tačno 1.0 onda ne utiče na veličinu jedinice.

Po defaultu jedna jedinica na platnu je tačno jedan piksel. Ako se primjenjuje, na primer, faktor od 0.5 to rezultira upola smanjenje visine ili širine canvas slike, a ako je faktor 2.0 onda će se širina ili visina duplo povećati.

Evo primera koji će sve lepo demonstrirati:

```
<html><head>
<script type="application/javascript">
function crtaj() {
    canvas = document.getElementById("kocke");
    if (canvas.getContext) {
        ctx = canvas.getContext("2d");
        ctx.strokeStyle="#fc0";
        ctx.lineWidth=1.5;
        ctx.fillRect(0,0,300,300);

        ctx.save();
        ctx.translate(50,50);
        drawSpirograph(ctx,22,6,5);

        ctx.translate(100,0);
        ctx.scale(0.75,0.75);
        drawSpirograph(ctx,22,6,5);

        ctx.translate(133.333,0);
        ctx.scale(0.75,0.75);
        drawSpirograph(ctx,22,6,5);
        ctx.restore();

        ctx.strokeStyle="#0cf";
        ctx.save();
        ctx.translate(50,150);
        ctx.scale(1,0.75);
        drawSpirograph(ctx,22,6,5);

        ctx.translate(100,0);
        ctx.scale(1,0.75);
        drawSpirograph(ctx,22,6,5);

        ctx.translate(100,0);
        ctx.scale(1,0.75);
        drawSpirograph(ctx,22,6,5);
        ctx.restore();

        ctx.strokeStyle="#cf0";
        ctx.save();
        ctx.translate(50,250);
        ctx.scale(0.75,1);
        drawSpirograph(ctx,22,6,5);

        ctx.translate(133.333,0);
        ctx.scale(0.75,1);
        drawSpirograph(ctx,22,6,5);

        ctx.translate(177.777,0);
        ctx.scale(0.75,1);
        drawSpirograph(ctx,22,6,5);
        ctx.restore();
    }
}
```



```

function drawSpirograph(ctx,R,r,O){
    x1 = R-O;
    y1 = 0;
    i = 1;
    ctx.beginPath();
    ctx.moveTo(x1,y1);
    do {
        if (i>20000) break;
        x2 = (R+r)*Math.cos(i*Math.PI/72) -
(r+O)*Math.cos(( (R+r)/r)*(i*Math.PI/72))
        y2 = (R+r)*Math.sin(i*Math.PI/72) -
(r+O)*Math.sin(( (R+r)/r)*(i*Math.PI/72))
        ctx.lineTo(x2,y2);
        x1 = x2;
        y1 = y2;
        i++;
    } while (x2 != R-O && y2 != 0 );
    ctx.stroke();
}
</script>
</head>

<body onload="crtaj();">
<canvas id="kocke" width="500" height="500"></canvas>
</body></html>

```

# Transformacije

Nakon svih gornjih transformacija, na primer rotacije, translacije i skaliranja možemo na kraju napraviti matricu svih transformacija, odnosno matricu transformacije. Funkcija za to je:

```
transform (m11, m12, m21, m22, dx, dy)
```

Naravno kao i svaka matrica, i ova se može predstaviti šematski:

m11	m21	dx
m12	m22	dy
0	0	1

Postoji i funkcija:

```
setTransform (m11, m12, m21, m22, dx, dy)
```

Evo sad jednog primera koji obuhvata setTransforms() funkciju ali i skoro sve ostale funkcije za transformacije:

```

<html><head>
<script type="text/javascript">
i=0;
function crtaj(){
    img = new Image();
    img.src = "jsslikeplanete/neptun.jpg";

```

```

img.onload = function(){
canvas = document.getElementById("tutorial");
if (canvas.getContext){
    ctx = canvas.getContext("2d");
    crtaj()
}
function crtaj(){
    brzinal=5
    brzina2=1
    ctx.save()
    ctx.clearRect(0,0,750,750); // briše prethodno pre crtanja novog
    ctx.setTransform (-.5,.5,-1,.25,250,img.width*.5)
    ctx.rotate(Math.PI*2*(i/360)*brzinal )
    ctx.drawImage(img,img.width*-.5,-48);
    ctx.restore();
    ctx.save()
    ctx.setTransform(-.5,.5,-1,.01,250,img.width)
    ctx.rotate(Math.PI*2*(i/360)*brzina2 )
    ctx.drawImage(img,img.width*-.5,-48);
    ctx.restore();
    font="Times new roman"
    size= 40 // velicina teksta
    /*_____U navodnicima pišemo tekst po želji_____*/
    text="3D Canvas Tekst"
    /*U nastavku sledi podešavanje teksta*/
    ctx.save()
    debljina =20
    for (j=1; j<debljina; j++){
        ctx.setTransform(-.5,.5,-1,.01,250,img.width*.5-j)
        ctx.rotate(Math.PI*2*(i/360)*brzina2 )
        ctx.translate(-img.width*.25, img.height*.25);
        ctx.fillStyle="rgba(255,190,100,.3)"
        ctx.mozTextStyle = size + "pt " + font ;
        ctx.mozDrawText(text);
    }
    ctx.restore()
    ctx.save()
    ctx.setTransform(-.5,.5,-1,.01,250,img.width*.5-debljina)
    ctx.rotate(Math.PI*2*(i/360)*brzina2 )
    ctx.translate(-img.width*.25, img.height*.25);
    gradBase = ctx.createLinearGradient(0,-img.height*.25,0,img.height*.25);
    gradBase.addColorStop(0 ,"rgb(255,220,150)");
    gradBase.addColorStop(0.2,"rgb(220,180,90)");
    gradBase.addColorStop(0.5,"rgb(185,155,70)");
    gradBase.addColorStop(1 ,"rgb( 140, 100,50)");
    ctx.mozTextStyle = size + "pt " + font ;
    ctx.fillStyle=gradBase
    ctx.mozDrawText(text);
    ctx.restore()
    gradGloss = ctx.createLinearGradient(150,150,250,250);
    gradGloss.addColorStop(0 ,"rgba(255,255,255,.4)");
    gradGloss.addColorStop(0.35,"rgba(225,225,225,.4)");
    gradGloss.addColorStop(0.65,"rgba(180,180,180,.3)");
    gradGloss.addColorStop(1 ,"rgba( 120, 120,120,.1)");
    ctx.save()
    ctx.setTransform(-.5,.5,-1,.01,250,img.width*.5-debljina)
    ctx.rotate(Math.PI*2*(i/360)*brzina2 )

```

```

    ctx.translate(-img.width*.25, img.height*.25);
    ctx.mozTextStyle = size + "pt " + font ;
    ctx.fillStyle=gradGloss
    ctx.mozDrawText(text);
    ctx.restore()
    i+=1;
    if (i==360) i=i%360
    setTimeout(crtaj, 50);
  }
}
}
</script>
<style type="text/css">
canvas {
border: 1px solid black;
background-color: black;
}
</style></head>
<body onload="crtaj()">
<canvas id="tutorial" width="500" height="300"></canvas>
</body></html>





```

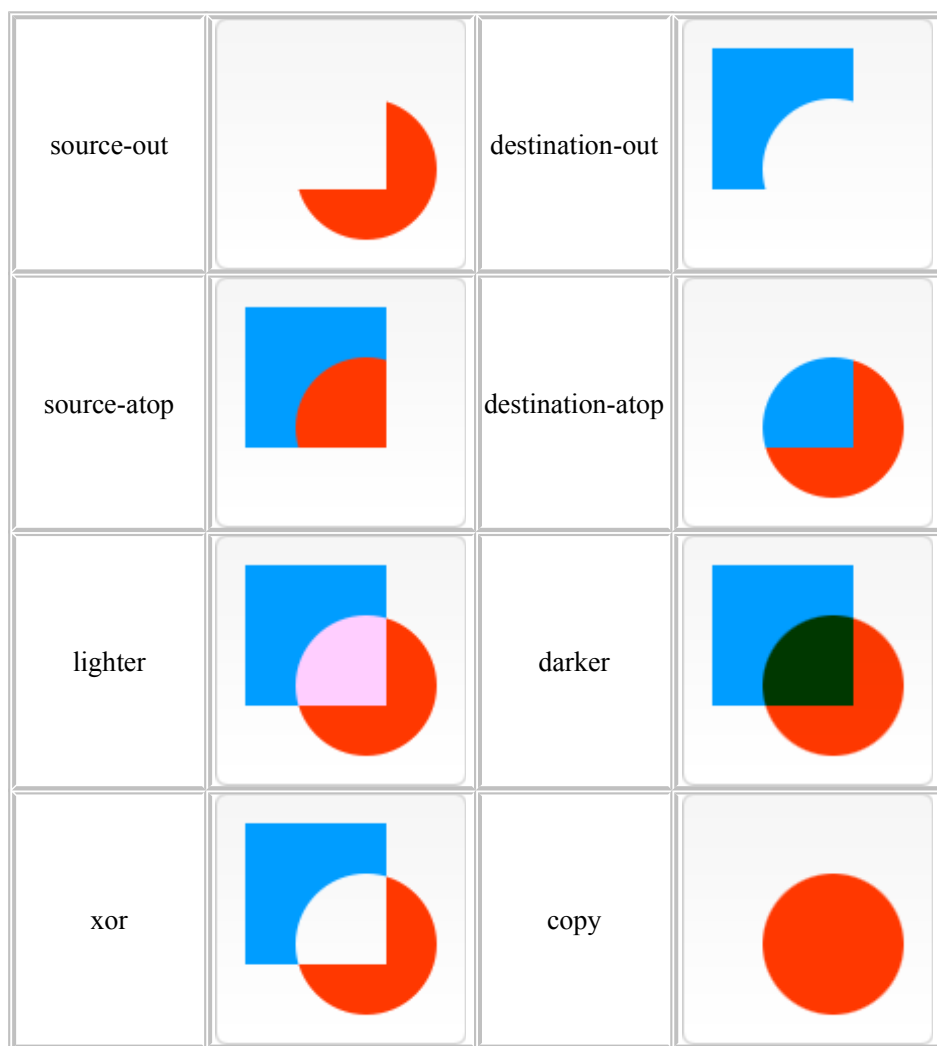
## Kompozicije

Ovde će mo razmatrati varijacije u slučajevima kada se preklapaju oblici u nekim njihovim delovima. Funkcija za to je:

```
globalCompositeOperation = tip
```

Gde tip može biti sledeće rezervisane reči:

Tip	Demonstracija	Tip	Demonstracija
source-over (default)		destination-over	
source-in		destination-in	



## Isečci

Možemo i sakrivati neželjene delove crteža, i tako postizati razne efekte. Funkcija za to je:

**clip()**

Evo primera:

```
<html><head>
<script type="application/javascript">
function crtaj() {
  canvas = document.getElementById("kocke");
  if (canvas.getContext) {
    ctx = canvas.getContext("2d");
    ctx.fillRect(0,0,150,150);
    ctx.translate(75,75);
    // Kreiranje kružnog isečka
    ctx.beginPath();
    ctx.arc(0,0,60,0,Math.PI*2,true);
  }
}
```

```

    ctx.clip();
    // Crtanje pozadina
    lingrad = ctx.createLinearGradient(0,-75,0,75);
    lingrad.addColorStop(0, "#232256");
    lingrad.addColorStop(1, "#143778");
    ctx.fillStyle = lingrad;
    ctx.fillRect(-75,-75,150,150);
    // crtanje zvezda
    for (j = 1; j < 50; j++){
        ctx.save();
        ctx.fillStyle = "#fff";
        ctx.translate(75-Math.floor(Math.random()*150), 75-
Math.floor(Math.random()*150));
        drawStar(ctx,Math.floor(Math.random()*4)+2);
        ctx.restore();
    }
}
function drawStar(ctx,r){
    ctx.save();
    ctx.beginPath()
    ctx.moveTo(r,0);
    for (i = 0; i < 9; i++){
        ctx.rotate(Math.PI/5);
        if (i%2 == 0) {
            ctx.lineTo((r/0.525731)*0.200811,0);
        } else {
            ctx.lineTo(r,0);
        }
    }
    ctx.closePath();
    ctx.fill();
    ctx.restore();
}
</script>
</head><body onload="crtaj();">
<canvas id="kocke" width="600" height="600"></canvas>
</body></html>

```

## Složeniji primeri CANVAS grafike

Evo linkova na kojima možete naći složenije primere upotrebe JavaScript CANVAS grafike:

<http://www.benjoffe.com/code/demos/canvascape/>

<http://andrewwooldridge.com/canvas/canvasgame001/canvasgame002.html>

<http://igrapher.com/#>

<http://arapehlivanian.com/wp-content/uploads/2007/02/canvas.html>

<http://www.blobsallad.se/>

<http://www.gradius-js.com/>

<http://billmill.org/static/canvastutorial/>

<http://www.canvasdemos.com/>

# JavaScript skripte

U ovom delu daćemo vrlo praktične skripte koje se često koriste prilikom izrade sajtova.

## Implementacija je **Časovnik**

Ova skriptica, kako joj i samo ime kaže, služi za prikaz "digitalnog" časovnika na web stranici. Pogledajte [DEMONSTRACIJU](#).

U <HEAD> delu stranice na kojoj želite da se prikaže časovnik pišete sledeći kod:

```
<script language="JavaScript">
function show5(){
    if (!document.layers&&!document.all&&!document.getElementById) return

    var Digital=new Date()
    var hours=Digital.getHours()
    var minutes=Digital.getMinutes()
    var seconds=Digital.getSeconds()
    var dn="AM"

    if (hours>12){
        dn="PM"
        hours=hours-12
    }

    if (hours==0) hours=12
    if (minutes<=9) minutes="0"+minutes
    if (seconds<=9) seconds="0"+seconds
    myclock="<font size='5' face='Arial' ><b><font size='4'>Trenutno
vreme:</font></br>" +hours+":"+minutes+":"+seconds+" "+dn+ "</b></font>"

    if (document.layers){
        document.layers.liveclock.document.write(myclock)
        document.layers.liveclock.document.close()
    }
    else if (document.all) liveclock.innerHTML=myclock
    else if (document.getElementById)
document.getElementById("liveclock").innerHTML=myclock

    setTimeout("show5()",1000)
}
</script>
```

U <BODY> delu pišete sledeće:

```
<body onLoad="show5()">
  <span id="liveclock" style="position:absolute;left:0;top:0;"></span>
</body>
```

Dakle obratite pažnju da u BODY tag obaveyno piše kako je napisano. Pozicija sata je u krajnjem levom gornjem uglu stranice ali naravno vi možete menjati poziciju menjajući STYLE.

vrlo jednostavna i mogu se malte ne bez ikakvih izmena koristiti.

## Kalendar

Skripta služi za prikaz kalendara u iskaćućem prozorčetu preko koga odabirate datum. Odabrani datum se upisuje u text box polje i automatski se zatvara kalendar. To je jako pogodno recimo za forme, za unos nekih datuma, godine rođenja ili slično. Može se iskoristiti i za neke druge stvari. Kako to izgleda kliknite na [DEMONSTRACIJU](#).

U <HEAD> delu stranice na kojoj želite da se prikaže kalendar pišete sledeći kod: [KLIKNI](#)

U <BODY> delu pišete sledeće:

```
<form name="sampleform">
<input type="text" name="firstinput" size=20> <small><a
href="javascript:showCal('Calendar1')">Selektuj datum</a></small>
</form>
```

## Povratak na vrh

Skripta služi da se klikanjem na pokretnu ikonicu vratite na vrh strane ukoliko stranica ima jako veliki sadržaj. Naravno ikonica se svo vreme kreće kako silazite naniže. Kako to izgleda kliknite na [DEMONSTRACIJU](#).

U <HEAD> delu stranice pišete sledeće:

```
<script language="JavaScript">
var displayed="<nobr><img border=0 src=up.png width=48 height=48></nobr>"

var logolink='javascript:window.scrollTo(0,0) '
var ns4=document.layers
var ie4=document.all
var ns6=document.getElementById&&!document.all

function regenerate(){
window.location.reload()
}
function regenerate2(){
if (ns4)
setTimeout("window.onresize=regenerate",400)
}

if (ie4||ns6)
```

```

document.write('<span id="logo" style="position:absolute;top:-300;z-index:100">'+displayed+'</span>')

function createtext(){
staticimage=new Layer(5)
staticimage.left=-300
staticimage.document.write('<a href="'+logolink+'">'+displayed+'</a>')
staticimage.document.close()
staticimage.visibility="show"
regenerate2()
staticitns()
}

function staticit(){
var w2=ns6? pageXOffset+w : document.body.scrollLeft+w
var h2=ns6? pageYOffset+h : document.body.scrollTop+h
crosslogo.style.left=w2
crosslogo.style.top=h2
}

function staticit2(){
staticimage.left=pageXOffset+window.innerWidth-staticimage.document.width-28
staticimage.top=pageYOffset+window.innerHeight-staticimage.document.height-10
}

function inserttext(){
if (ie4)
crosslogo=document.all.logo
else if (ns6)
crosslogo=document.getElementById("logo")
crosslogo.innerHTML='<a href="'+logolink+'">'+displayed+'</a>'
w=ns6? window.innerWidth-crosslogo.offsetWidth-20 :
document.body.clientWidth-crosslogo.offsetWidth-10
h=ns6? window.innerHeight-crosslogo.offsetHeight-15 :
document.body.clientHeight-crosslogo.offsetHeight-10
crosslogo.style.left=w
crosslogo.style.top=h
if (ie4)
window.onscroll=staticit
else if (ns6)
startstatic=setInterval("staticit()",100)
}

if (ie4||ns6){
window.onload=inserttext
window.onresize=new Function("window.location.reload()")
}
else if (ns4)
window.onload=createtext

function staticitns(){
startstatic=setInterval("staticit2()",90)
}
</script>

```



U <BODY> delu pišite šta god želite samo da bude dovoljno sadržaja kako bi se pojavio klizač, kako bi mogli da se vratite na vrh stranice. Na primer demonstracije radi možete pisati sledeće::

```
p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>
p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>
p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p<br>p
```



I naravno ikonica može biti po vašoj volji, ali možete koristiti i našu:

## Sneg

Skripta ostvaruje kao da proverjava sneg na stranici. Naravno umesto ikonice za sneg možete koristiti i na primer neku slikicu kao što je na primer list drveta pa će biti efekat kao da pada lišće, itd. Kako to izgleda kliknite na [DEMONSTRACIJU](#).

U <HEAD> delu stranice pišete sledeće:

```
<script type="text/javascript">

var snowsrc="http://www.bubaj.com/skripte/javascript/sneg/snow.gif"
var no = 10;
var hidesnowtime = 0;
var snowdistance = "pageheight";

var ie4up = (document.all) ? 1 : 0;
var ns6up = (document.getElementById&&!document.all) ? 1 : 0;

function iecompattest(){
return (document.compatMode && document.compatMode!="BackCompat")?
document.documentElement : document.body
}

var dx, xp, yp;
var am, stx, sty;
var i, doc_width = 800, doc_height = 600;

if (ns6up) {
doc_width = self.innerWidth;
doc_height = self.innerHeight;
} else if (ie4up) {
doc_width = iecompattest().clientWidth;
doc_height = iecompattest().clientHeight;
}

dx = new Array();
xp = new Array();
yp = new Array();
am = new Array();
stx = new Array();
sty = new Array();
```

```

snowsrc=(snowsrc.indexOf("dynamicdrive.com")!=-1)?
"http://www.bubaj.com/skripte/javascript/sneg/snow.gif" : snowsrc
for (i = 0; i < no; ++ i) {
dx[i] = 0;
xp[i] = Math.random()*(doc_width-50);
yp[i] = Math.random()*doc_height;
am[i] = Math.random()*20;
stx[i] = 0.02 + Math.random()/10;
sty[i] = 0.7 + Math.random();
if (ie4up||ns6up) {
if (i == 0) {
document.write("<div id=\"dot\"+ i +\"\" style=\"POSITION: absolute; Z-INDEX:
"+ i +\"; VISIBILITY: visible; TOP: 15px; LEFT: 15px;\"><a
href=\"http://dynamicdrive.com\"><img src='"+snowsrc+"'
border=\"0\"></a></div>");
} else {
document.write("<div id=\"dot\"+ i +\"\" style=\"POSITION: absolute; Z-INDEX:
"+ i +\"; VISIBILITY: visible; TOP: 15px; LEFT: 15px;\"><img
src='"+snowsrc+"' border=\"0\"></div>");
}
}
}

function snowIE_NS6() {
doc_width = ns6up?window.innerWidth-10 : iecompattest().clientWidth-10;
doc_height=(window.innerHeight && snowdistance=="windowheight"?
window.innerHeight : (ie4up && snowdistance=="windowheight"?
iecompattest().clientHeight : (ie4up && !window.opera &&
snowdistance=="pageheight"? iecompattest().scrollHeight :
iecompattest().offsetHeight);
for (i = 0; i < no; ++ i) {
yp[i] += sty[i];
if (yp[i] > doc_height-50) {
xp[i] = Math.random()*(doc_width-am[i]-30);
yp[i] = 0;
stx[i] = 0.02 + Math.random()/10;
sty[i] = 0.7 + Math.random();
}
dx[i] += stx[i];
document.getElementById("dot"+i).style.top=yp[i]+"px";
document.getElementById("dot"+i).style.left=xp[i] + am[i]*Math.sin(dx[i])
+"px";
}
snowtimer=setTimeout("snowIE_NS6()", 10);
}

function hidesnow(){
if (window.snowtimer) clearTimeout(snowtimer)
for (i=0; i<no; i++)
document.getElementById("dot"+i).style.visibility="hidden"
}

if (ie4up||ns6up){
snowIE_NS6();
if (hidesnowtime>0)

```

```
setTimeout("hidesnow()", hidesnowtime*1000)
}

</script>
```

U <BODY> ništa ne pišete vezano za efekat snega već samo vaš neki sadržaj:



I naravno ikonica može biti po vašoj volji, ali možete koristiti i našu:

## Slajd šou

Skripta služi da prikaže slike. Da ne bih mnogo pisao evo vidite [DEMONSTRACIJU](#).

U <HEAD> delu stranice pišete sledeći JavaScript KOD:

```
<script language="JavaScript">
var featuredcontentslider={

ajaxloadingmsg: '<div style="margin: 20px 0 0 20px"> Fetching slider Contents. Please wait...</div>',
bustajaxcache: true,
enablepersist: true,

settingcaches: {},

jumpTo:function(fcsid, pagenumber){
this.turnpage(this.settingcaches[fcsid], pagenumber)
},

ajaxconnect:function(setting){
var page_request = false
if (window.ActiveXObject){
try {
page_request = new ActiveXObject("Msxml2.XMLHTTP")
}
catch (e){
try{
page_request = new ActiveXObject("Microsoft.XMLHTTP")
}
catch (e){}
}
}
else if (window.XMLHttpRequest)
page_request = new XMLHttpRequest()
else
return false
var pageurl=setting.contentsource[1]
page_request.onreadystatechange=function(){
featuredcontentslider.ajaxpopulate(page_request, setting)
```

```

}
document.getElementById(setting.id).innerHTML=this.ajaxloadingmsg
var bustcache=(!this.bustajaxcache)? "" : (pageurl.indexOf("?")!=-1)?
"&" + new Date().getTime() : "?" + new Date().getTime()
page_request.open('GET', pageurl+bustcache, true)
page_request.send(null)
},

ajaxpopulate:function(page_request, setting){
if (page_request.readyState == 4 && (page_request.status==200 ||
window.location.href.indexOf("http")==-1)){
document.getElementById(setting.id).innerHTML=page_request.responseText
this.buildpaginate(setting)
}
},

buildcontentdivs:function(setting){
var alldivs=document.getElementById(setting.id).getElementsByTagName("div")
for (var i=0; i<alldivs.length; i++){
if (this.css(alldivs[i], "contentdiv", "check")){
setting.contentdivs.push(alldivs[i])
alldivs[i].style.display="none"
}
}
},

buildpaginate:function(setting){
this.buildcontentdivs(setting)
var sliderdiv=document.getElementById(setting.id)
var pdiv=document.getElementById("paginate-"+setting.id)
var phtml=""
var toc=setting.toc
var nextprev=setting.nextprev
if (typeof toc=="string" && toc!="markup" || typeof toc=="object"){
for (var i=1; i<=setting.contentdivs.length; i++){
phtml+=''<a href="#" + i + '' class="toc">' + (typeof toc=="string"?
toc.replace(/#increment/, i) : toc[i-1]) + '</a> '
}
phtml=(nextprev[0]!=''? '<a href="#"prev" class="prev">' + nextprev[0] + '</a>
' : '') + phtml + (nextprev[1]!=''? '<a href="#"next"
class="next">' + nextprev[1] + '</a>' : '')
pdiv.innerHTML=phtml
}
var pdivlinks=pdiv.getElementsByTagName("a")
var toclinkscount=0
for (var i=0; i<pdivlinks.length; i++){
if (this.css(pdivlinks[i], "toc", "check")){
if (toclinkscount>setting.contentdivs.length-1){
pdivlinks[i].style.display="none"
continue
}
pdivlinks[i].setAttribute("rel", ++toclinkscount)
pdivlinks[i][setting.revealtype]=function(){
featuredcontentslider.turnpage(setting, this.getAttribute("rel"))
return false
}
}
}

```

```

setting.toclinks.push(pdivlinks[i])
}
else if (this.css(pdivlinks[i], "prev", "check") || this.css(pdivlinks[i],
"next", "check")){
pdivlinks[i].onclick=function(){
featuredcontentslider.turnpage(setting, this.className)
return false
}
}
}
this.turnpage(setting, setting.currentpage, true)
if (setting.autorotate[0]){
pdiv[setting.revealtype]=function(){
featuredcontentslider.cleartimer(setting, window["fcsautorun"+setting.id])
}
sliderdiv["onclick"]=function(){
featuredcontentslider.cleartimer(setting, window["fcsautorun"+setting.id])
}
setting.autorotate[1]=setting.autorotate[1]+(1/setting.enablefade[1]*50)
this.autorotate(setting)
},

urlparamselect:function(fcsid){
var result=window.location.search.match(new RegExp(fcsid+"=(\d+)", "i"))
return (result==null)? null : parseInt(RegExp.$1)
},

turnpage:function(setting, thepage, autocall){
var currentpage=setting.currentpage
var totalpages=setting.contentdivs.length
var turntopage=(/prev/i.test(thepage))? currentpage-1 :
(/next/i.test(thepage))? currentpage+1 : parseInt(thepage)
turntopage=(turntopage<1)? totalpages : (turntopage>totalpages)? 1 :
turntopage
if (turntopage==setting.currentpage && typeof autocall=="undefined")
return
setting.currentpage=turntopage
setting.contentdivs[turntopage-1].style.zIndex=++setting.topzindex
this.cleartimer(setting, window["fcsfade"+setting.id])
setting.cacheprevpage=setting.prevpag
if (setting.enablefade[0]==true){
setting.cuopacity=0
this.fadeup(setting)
}
if (setting.enablefade[0]==false){
setting.contentdivs[setting.prevpag-1].style.display="none"
setting.onChange(setting.prevpag, setting.currentpage)
}
setting.contentdivs[turntopage-1].style.visibility="visible"
setting.contentdivs[turntopage-1].style.display="block"
if (setting.prevpag<=setting.toclinks.length)
this.css(setting.toclinks[setting.prevpag-1], "selected", "remove")
if (turntopage<=setting.toclinks.length)
this.css(setting.toclinks[turntopage-1], "selected", "add")
setting.prevpag=turntopage

```

```

if (this.enablepersist)
this.setCookie("fcspersist"+setting.id, turntopage)
},

setopacity:function(setting, value){
var targetobject=setting.contentdivs[setting.currentpage-1]
if (targetobject.filters && targetobject.filters[0]){
if (typeof targetobject.filters[0].opacity=="number")
targetobject.filters[0].opacity=value*100
else
targetobject.style.filter="alpha(opacity="+value*100+)"
}
else if (typeof targetobject.style.MozOpacity!="undefined")
targetobject.style.MozOpacity=value
else if (typeof targetobject.style.opacity!="undefined")
targetobject.style.opacity=value
setting.cuopacity=value
},

fadeup:function(setting){
if (setting.cuopacity<1){
this.setopacity(setting, setting.cuopacity+setting.enablefade[1])
window["fcsfade"+setting.id]=setTimeout(function()
{featuredcontentslider.fadeup(setting)}, 50)
}
else{
if (setting.cacheprevpage!=setting.currentpage)
setting.contentdivs[setting.cacheprevpage-1].style.display="none"
setting.onChange(setting.cacheprevpage, setting.currentpage)
}
},

cleartimer:function(setting, timervar){
if (typeof timervar!="undefined"){
clearTimeout(timervar)
clearInterval(timervar)
if (setting.cacheprevpage!=setting.currentpage){
setting.contentdivs[setting.cacheprevpage-1].style.display="none"
}
}
},

css:function(el, targetclass, action){
var needle=new RegExp("(^|\\s+)" +targetclass+"(\\s+)", "ig")
if (action=="check")
return needle.test(el.className)
else if (action=="remove")
el.className=el.className.replace(needle, "")
else if (action=="add")
el.className+=" "+targetclass
},

autorotate:function(setting){
window["fcsautorun"+setting.id]=setInterval(function()
{featuredcontentslider.turnpage(setting, "next")}, setting.autorotate[1])
},

```

```

getCookie:function(Name) {
var re=new RegExp(Name+"=[^;]+", "i");
if (document.cookie.match(re))
return document.cookie.match(re)[0].split("=")[1]
return null
},

setCookie:function(name, value){
document.cookie = name+"="+value

},

init:function(setting) {
var persistedpage=this.getCookie("fcspersist"+setting.id) || 1
var urlselectedpage=this.urlparamselect(setting.id)
this.settingcaches[setting.id]=setting
setting.contentdivs=[]
setting.toclinks=[]
setting.topzindex=0
setting.currentpage=urlselectedpage || ((this.enablepersist)?
persistedpage : 1)
setting.prevpag=setting.currentpage
setting.revealtype="on"+(setting.revealtype || "click")
setting.cuopacity=0
setting.onChange=setting.onChange || function(){}
if (setting.contentsource[0]=="inline")
this.buildpaginate(setting)
if (setting.contentsource[0]=="ajax")
this.ajaxconnect(setting)
}

}
</script>

```

Takodje u <HEAD> delu pišete sledeći CSS KOD:

```

<style type="text/css">
.sliderwrapper{
position: relative;
overflow: hidden;
border: 10px solid navy;
border-bottom-width: 6px;
width: 400px;
height: 250px;
}

.sliderwrapper .contentdiv{
visibility: hidden;
position: absolute;
left: 0;
top: 0;
padding: 5px;

```

```

background: white;
width: 390px;
height: 100%;
filter:progid:DXImageTransform.Microsoft.alpha(opacity=100);
-moz-opacity: 1;
opacity: 1;
}

.pagination{
width: 400px;
text-align: right;
background-color: navy;
padding: 5px 10px;
}

.pagination a{
padding: 0 5px;
text-decoration: none;
color: #00007D;
background: white;
}

.pagination a:hover, .pagination a.selected{
color: #000;
background-color: #FEE496;
}
</style>

```

U <BODY> delu pišite pišete sledeće:

```

<div id="slider2" class="sliderwrapper">

<div class="contentdiv">

</div>

<div class="contentdiv">

</div>

<div class="contentdiv">

</div>

</div>

<div id="paginate-slider2" class="pagination">

<a href="#" class="toc">Prva</a>
<a href="#" class="toc anotherclass">Druga</a>
<a href="#" class="toc">Treci</a>
<a href="#" class="prev" style="margin-left: 10px"> < </a>
<a href="#" class="next"> > </a>

</div>

```



```

<script type="text/javascript">

featuredcontentslider.init({
id: "slider2",
contentsource: ["inline", ""],
toc: "markup",
nextprev: ["Previous", "Next"],
revealtype: "click",
enablefade: [true, 0.2],
autorotate: [false, 3000],
onChange: function(previndex, curindex){}
})

</script>

```

## Animirani tekst

Vrlo zanimljiva skriptica. Može se iskoristiti za razne stvari, evo kliknite da vidite o čemu se radi: [DEMONSTRACIJU](#).

U <HEAD> delu stranice pišete sledeći KOD: [JavaScript KOD](#).

Takodje u <HEAD> delu stranice smestite sledeći KOD, ali obavezno pre gornjeg KOD-a:

```

<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js"></scri
pt>

```

U <BODY> smestite sledeće:

```

<b>
<a href="javascript:animatedcollapse.show(['jason', 'kelly',
'michael'])">Otvori primere 1, 2, 3</a>
|
<a href="javascript:animatedcollapse.hide(['jason', 'kelly',
'michael'])">Zatvori primere 1, 2, 3</a>
</b>

<p>
<b>Primer 1 (individualno):</b>
</p>

<a href="javascript:animatedcollapse.toggle('jason')">

</a>
<a href="javascript:animatedcollapse.show('jason')">Otvori</a>
||
<a href="javascript:animatedcollapse.hide('jason')">Zatvori</a>

<div id="jason" style="width: 300px; background: #FFFFCC; display:none">
<b>Sadrzaj unutar DIV!</b><br />
<b>Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde
neki tekst. Ovde neki tekst</b><br />

```

```

</div>

<p>
<b>Primer 2 (individualni):</b>
</p>

<a href="javascript:animatedcollapse.toggle('kelly')">

</a>
<a href="javascript:animatedcollapse.show('kelly')">Otvori</a>
||
<a href="javascript:animatedcollapse.hide('kelly')">Zatvori</a>

<div id="kelly" style="width: 300px; background: #D2FBFF; display:none">
<b>Sadrzaj unutar DIV!</b><br />
<b>Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde
neki tekst. Ovde neki tekst</b><br />
</div>

<p><b>Primer 3 (individualni):</b></p>

<a href="javascript:animatedcollapse.toggle('michael')">

</a>
<a href="javascript:animatedcollapse.show('michael')">Otvori</a>
||
<a href="javascript:animatedcollapse.hide('michael')">Zatvori</a>

<div id="michael" style="width: 300px; background: #E7FFCC; display:none">
<b>Sadrzaj unutar DIV!</b><br />
<b>Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde
neki tekst. Ovde neki tekst</b><br />
</div>

<hr style="margin: 1em 0" />

<p><b>Primer 4 (grupno zavisno):</b></p>

<a href="#" rel="toggle[cat]" data-openimage="collapse.jpg" data-
closedimage="expand.jpg">

</a>

<a href="javascript:animatedcollapse.show('cat')">Otvori</a>
||
<a href="javascript:animatedcollapse.hide('cat')">Zatvori</a>

<div id="cat" style="width: 400px; background: #BDF381;">
<b>Sadrzaj unutar DIV!</b><br />
<b>Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde

```

```

neki tekst. Ovde neki tekst</b><br />
</div>

<p><b>Primer 5 (grupno zavisno):</b></p>

<a href="#" rel="toggle[dog]" data-openimage="collapse.jpg" data-
closedimage="expand.jpg">

</a>

<a href="javascript:animatedcollapse.show('dog')">Otvori</a>
||
<a href="javascript:animatedcollapse.hide('dog')">Zatvori</a>

<div id="dog" style="width: 400px; background: #BDF381;">
<b>Sadrzaj unutar DIV!</b><br />
<b>Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde
neki tekst. Ovde neki tekst</b><br />
</div>

<p><b>Primer 6 (grupno zavisno):</b></p>

<a href="#" rel="toggle[rabbit]" data-openimage="collapse.jpg" data-
closedimage="expand.jpg">

</a>

<a href="javascript:animatedcollapse.show('rabbit')">Otvori</a>
||
<a href="javascript:animatedcollapse.hide('rabbit')">Zatvori</a>

<div id="rabbit" style="width: 400px; background: #BDF381;">
<b>Sadrzaj unutar DIV!</b><br />
<b>Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde neki tekst. Ovde
neki tekst. Ovde neki tekst</b><br />
</div>

```

I naravno ikonice mogu biti po vašoj volji, ali možete koristiti i naše:

ZATVORI

OTVORI

Dugme za prikaz

## Harmonika

Takođe zanimljiva skriptica. Može se iskoristiti za razne stvari, evo kliknite da vidite o čemu se radi: [DEMONSTRACIJU](#).

U <HEAD> delu stranice pišete sledeći KOD:

```

<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js"></scri
pt>

<style type="text/css">

```

```

.haccordion{
padding: 0;
}

.haccordion ul{
margin: 0;
padding: 0;
list-style: none;
overflow: hidden;
}

.haccordion li{
margin: 0;
padding: 0;
display: block;
width: 100%;
height: 200px;
overflow: hidden;
float: left;
}

.haccordion li .hpanel{
width: 100%;
height: 200px;
}

#hc1 li{
margin:0 3px 0 0;
}

#hc1 li .hpanel{
padding: 5px;
background: lightblue;
}

#hc2 li{
margin:0 0 0 0;
border: 12px solid black;
}

#hc2 li .hpanel{
padding: 5px;
background: #E2E9FF;
cursor: hand;
cursor: pointer;
}
</style>

<script type="text/javascript">
var haccordion={
ajaxloadingmsg: '<div style="margin: 1em; font-weight: bold"></div>',

accordioninfo: {},

expandli:function(accordionid, targetli){
var config=haccordion.accordioninfo[accordionid]

```

```

var $targetli=(typeof targetli=="number")? config.$targetlis.eq(targetli) :
(typeof targetli=="string")? jQuery('#'+targetli) : jQuery(targetli)
if (typeof config.$lastexpanded!="undefined")
config.$lastexpanded.stop().animate({width:config.paneldimensions.peekw},
config.speed)
$targetli.stop().animate({width:$targetli.data('hpaneloffsetw')},
config.speed)
config.$lastexpanded=$targetli
},

urlparamselect:function(accordionid){
var result=window.location.search.match(new RegExp(accordionid+"=(\d+)",
"i"))
if (result!=null)
result=parseInt(RegExp.$1)+""
return result
},

getCookie:function(Name){
var re=new RegExp(Name+"=[^;]+", "i")
if (document.cookie.match(re))
return document.cookie.match(re)[0].split("=")[1]
return null
},

setCookie:function(name, value){
document.cookie = name + "=" + value + "; path=/"
},

loadexternal: function($, config){
var $hcontainer=$( '#'+config.ajaxsource.container ).html(this.ajaxloadingmsg)
$.ajax({
url: config.ajaxsource.path,
async: true,
error:function(ajaxrequest){
$hcontainer.html('Error fetching content.<br />Server Response:
'+ajaxrequest.responseText)
},
success:function(content){
$hcontainer.html(content)
haccordion.init($, config)
}
})
},

init:function($, config){
haccordion.accordioninfo[config.accordionid]=config
var $targetlis=$( '#'+config.accordionid ).find('ul:eq(0) > li')
config.$targetlis=$targetlis
config.selectedli=config.selectedli || []
config.speed=config.speed || "normal"
$targetlis.each(function(i){
var $target=$(this).data('pos', i)
$target.data('hpaneloffsetw', $target.find('.hpanel:eq(0)').outerWidth())
$target.mouseenter(function(){
haccordion.expandli(config.accordionid, this)
config.$lastexpanded=$(this)

```

```

    })
    if (config.collapsecurrent){
    $target.mouseleave(function(){
    $(this).stop().animate({width:config.paneldimensions.peakw}, config.speed)
    })
    }
    })
    var selectedli=haccordion.urlparamselect(config.accordionid) ||
    ((config.selectedli[1] && haccordion.getCookie(config.accordionid))?
    parseInt(haccordion.getCookie(config.accordionid)) : config.selectedli[0])
    selectedli=parseInt(selectedli)
    if (selectedli>=0 && selectedli<config.$targetlis.length){
    config.$lastexpanded=$targetlis.eq(selectedli)
    config.$lastexpanded.css('width', config.
    $lastexpanded.data('hpaneloffsetw'))
    }
    $(window).bind('unload', function(){
    haccordion.uninit($, config)
    })
    },

    uninit:function($, config){
    var $targetlis=config.$targetlis
    var expandedliindex=-1
    $targetlis.each(function(){
    var $target=$(this)
    $target.unbind()
    if ($target.width()==$target.data('hpaneloffsetw'))
    expandedliindex=$target.data('pos')
    })
    if (config.selectedli[1]==true)
    haccordion.setCookie(config.accordionid, expandedliindex)
    },

    setup:function(config){
    document.write('<style type="text/css">\n')
    document.write('#'+config.accordionid+' li{width:
    '+config.paneldimensions.peakw+';\nheight:
    '+config.paneldimensions.h+';\n}\n')
    document.write('#'+config.accordionid+' li .hpanel{width:
    '+config.paneldimensions.fullw+';\nheight:
    '+config.paneldimensions.h+';\n}\n')
    document.write('</style>')
    jQuery(document).ready(function($){
    if (config.ajaxsource)
    haccordion.loadexternal($, config)
    else
    haccordion.init($, config)
    })
    }

    }

    haccordion.setup({
    accordionid: 'hcl',
    paneldimensions: {peakw:'50px', fullw:'400px', h:'158px'},
    selectedli: [0, true],

```

```

collapsecurrent: false
})

haccordion.setup({
  accordionid: 'hc2',
  paneldimensions: {peekw:'30px', fullw:'450px', h:'150px'},
  selectedli: [-1, true],
  collapsecurrent: true
})
</script>

```

U <BODY> smestite sledeće:

```

<h1>Znamenitosti</h1>

<div id="hcl" class="haccordion">
<ul>

<li>
<div class="hpanel">
Andaman obala je Tajlandski najdragocjeniji prirodni resurs.
Tu su najpopularnija i najluksuznija naselja u Aziji.
</div>
</li>

<li>
<div class="hpanel">
Japan je ustavna monarhija ali careva moć je izuzetano
ograničena.
</div>
</li>

<li>
<div class="hpanel">
Malezija je južnoazijska zemlja bogata prirodnim resursima u
poljoprivredi, šumarstvu i mineralima.
</div>
</li>

<li>
<div class="hpanel">
Poljoprivreda je dugo bila najvažniji privredni sektor
Kambodžije privrede gospodarstva.
</div>
</li>

<li>
<div class="hpanel">
Langkawi je posebno poznat po svojim plažama koje su
među najboljima u Maleziji.
</div>

```


```

</li>

</ul>
</div>

<p style="clear:left">
<a href="javascript:haccordion.expandli('hc1', 0)">Andaman</a>
|
<a href="javascript:haccordion.expandli('hc1', 1)">Japanski car</a>
|
<a href="javascript:haccordion.expandli('hc1', 2)">Resursi Malezije</a>
|
<a href="javascript:haccordion.expandli('hc1', 3)">Poljoprivreda
Kambodže</a>
|
<a href="javascript:haccordion.expandli('hc1', 4)">Plaže u Maleziji</a>
</p>

```

I naravno ikonicu ya učitavanje može biti po vašoj volji, ali možete koristiti i našu: 

## Okvir za sadrzaj

Takođe zanimljiva skriptica. Može se iskoristiti za razne stvari, evo kliknite da vidite o čemu se radi: [DEMONSTRACIJA](#).

U <HEAD> delu stranice pišete sledeći KOD:

```

<style type="text/css">
div.dropinbox{
width:400px;
height:230px;
border:1px solid black;
padding:5px;
background:#e7e7e7;
}

div.dropinboxaltstyle{
width:300px;
height:200px;
padding:10px;
background:#c7edac;
-webkit-border-radius:8px;
-moz-border-radius:8px;
border-radius:8px;
}

.standardshadow{
-webkit-box-shadow: 5px 5px 10px rgba(95, 95, 95, 0.5);
-moz-box-shadow: 5px 5px 10px rgba(95, 95, 95, 0.5);
box-shadow: 5px 5px 10px rgba(95, 95, 95, 0.5);
}

```



```

.drop-shadow{
-webkit-box-shadow:0 1px 3px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1)
inset;
-moz-box-shadow:0 1px 3px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1)
inset;
box-shadow:0 1px 3px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
}

.drop-shadow:before, .drop-shadow:after{
content:"";
position:absolute;
z-index:-2;
}

.lifted{
-moz-border-radius:4px;
border-radius:4px;
}

.lifted:before, .lifted:after{
bottom:15px;
left:10px;
width:50%;
height:20%;
max-width:300px;
-webkit-box-shadow:0 15px 10px rgba(0, 0, 0, 0.7);
-moz-box-shadow:0 15px 10px rgba(0, 0, 0, 0.7);
box-shadow:0 15px 10px rgba(0, 0, 0, 0.7);
-webkit-transform:rotate(-3deg);
-moz-transform:rotate(-3deg);
-ms-transform:rotate(-3deg);
-o-transform:rotate(-3deg);
transform:rotate(-3deg);
}

.lifted:after{
right:10px;
left:auto;
-webkit-transform:rotate(3deg);
-moz-transform:rotate(3deg);
-ms-transform:rotate(3deg);
-o-transform:rotate(3deg);
transform:rotate(3deg);
}
</style>

<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.5.2/jquery.min.js"></scri
pt>

<script>
jQuery.easing['jswing'] = jQuery.easing['swing'];

jQuery.extend( jQuery.easing,
{

```

```

def: 'easeOutQuad',
swing: function (x, t, b, c, d) {
return jQuery.easing[jQuery.easing.def](x, t, b, c, d);
},
easeInQuad: function (x, t, b, c, d) {
return c*(t/=d)*t + b;
},
easeOutQuad: function (x, t, b, c, d) {
return -c *(t/=d)*(t-2) + b;
},
easeInOutQuad: function (x, t, b, c, d) {
if ((t/=d/2) < 1) return c/2*t*t + b;
return -c/2 * ((--t)*(t-2) - 1) + b;
},
easeInCubic: function (x, t, b, c, d) {
return c*(t/=d)*t*t + b;
},
easeOutCubic: function (x, t, b, c, d) {
return c*((t=t/d-1)*t*t + 1) + b;
},
easeInOutCubic: function (x, t, b, c, d) {
if ((t/=d/2) < 1) return c/2*t*t*t + b;
return c/2*((t-=2)*t*t + 2) + b;
},
easeInQuart: function (x, t, b, c, d) {
return c*(t/=d)*t*t*t + b;
},
easeOutQuart: function (x, t, b, c, d) {
return -c * ((t=t/d-1)*t*t*t - 1) + b;
},
easeInOutQuart: function (x, t, b, c, d) {
if ((t/=d/2) < 1) return c/2*t*t*t*t + b;
return -c/2 * ((t-=2)*t*t*t - 2) + b;
},
easeInQuint: function (x, t, b, c, d) {
return c*(t/=d)*t*t*t*t + b;
},
easeOutQuint: function (x, t, b, c, d) {
return c*((t=t/d-1)*t*t*t*t + 1) + b;
},
easeInOutQuint: function (x, t, b, c, d) {
if ((t/=d/2) < 1) return c/2*t*t*t*t*t + b;
return c/2*((t-=2)*t*t*t*t + 2) + b;
},
easeInSine: function (x, t, b, c, d) {
return -c * Math.cos(t/d * (Math.PI/2)) + c + b;
},
easeOutSine: function (x, t, b, c, d) {
return c * Math.sin(t/d * (Math.PI/2)) + b;
},
easeInOutSine: function (x, t, b, c, d) {
return -c/2 * (Math.cos(Math.PI*t/d) - 1) + b;
},
easeInExpo: function (x, t, b, c, d) {
return (t==0) ? b : c * Math.pow(2, 10 * (t/d - 1)) + b;
},
easeOutExpo: function (x, t, b, c, d) {

```

```

return (t==d) ? b+c : c * (-Math.pow(2, -10 * t/d) + 1) + b;
},
easeInOutExpo: function (x, t, b, c, d) {
if (t==0) return b;
if (t==d) return b+c;
if ((t/=d/2) < 1) return c/2 * Math.pow(2, 10 * (t - 1)) + b;
return c/2 * (-Math.pow(2, -10 * --t) + 2) + b;
},
easeInCirc: function (x, t, b, c, d) {
return -c * (Math.sqrt(1 - (t/=d)*t) - 1) + b;
},
easeOutCirc: function (x, t, b, c, d) {
return c * Math.sqrt(1 - (t=t/d-1)*t) + b;
},
easeInOutCirc: function (x, t, b, c, d) {
if ((t/=d/2) < 1) return -c/2 * (Math.sqrt(1 - t*t) - 1) + b;
return c/2 * (Math.sqrt(1 - (t-=2)*t) + 1) + b;
},
easeInElastic: function (x, t, b, c, d) {
var s=1.70158;var p=0;var a=c;
if (t==0) return b; if ((t/=d)==1) return b+c; if (!p) p=d*.3;
if (a < Math.abs(c)) { a=c; var s=p/4; }
else var s = p/(2*Math.PI) * Math.asin (c/a);
return -(a*Math.pow(2,10*(t-=1)) * Math.sin( (t*d-s)*(2*Math.PI)/p )) + b;
},
easeOutElastic: function (x, t, b, c, d) {
var s=1.70158;var p=0;var a=c;
if (t==0) return b; if ((t/=d)==1) return b+c; if (!p) p=d*.3;
if (a < Math.abs(c)) { a=c; var s=p/4; }
else var s = p/(2*Math.PI) * Math.asin (c/a);
return a*Math.pow(2,-10*t) * Math.sin( (t*d-s)*(2*Math.PI)/p ) + c + b;
},
easeInOutElastic: function (x, t, b, c, d) {
var s=1.70158;var p=0;var a=c;
if (t==0) return b; if ((t/=d/2)==2) return b+c; if (!p) p=d*(.3*1.5);
if (a < Math.abs(c)) { a=c; var s=p/4; }
else var s = p/(2*Math.PI) * Math.asin (c/a);
if (t < 1) return -.5*(a*Math.pow(2,10*(t-=1)) * Math.sin( (t*d-
s)*(2*Math.PI)/p )) + b;
return a*Math.pow(2,-10*(t-=1)) * Math.sin( (t*d-s)*(2*Math.PI)/p )*.5 + c +
b;
},
easeInBack: function (x, t, b, c, d, s) {
if (s == undefined) s = 1.70158;
return c*(t/=d)*t*((s+1)*t - s) + b;
},
easeOutBack: function (x, t, b, c, d, s) {
if (s == undefined) s = 1.70158;
return c*((t=t/d-1)*t*((s+1)*t + s) + 1) + b;
},
easeInOutBack: function (x, t, b, c, d, s) {
if (s == undefined) s = 1.70158;
if ((t/=d/2) < 1) return c/2*(t*t*(((s*=(1.525))+1)*t - s)) + b;
return c/2*((t-=2)*t*(((s*=(1.525))+1)*t + s) + 2) + b;
},
easeInBounce: function (x, t, b, c, d) {
return c - jQuery.easing.easeOutBounce (x, d-t, 0, c, d) + b;
}

```

```

},
easeOutBounce: function (x, t, b, c, d) {
if ((t/=d) < (1/2.75)) {
return c*(7.5625*t*t) + b;
} else if (t < (2/2.75)) {
return c*(7.5625*(t-=(1.5/2.75))*t + .75) + b;
} else if (t < (2.5/2.75)) {
return c*(7.5625*(t-=(2.25/2.75))*t + .9375) + b;
} else {
return c*(7.5625*(t-=(2.625/2.75))*t + .984375) + b;
}
},
easeInOutBounce: function (x, t, b, c, d) {
if (t < d/2) return jQuery.easing.easeInBounce (x, t*2, 0, c, d) * .5 + b;
return jQuery.easing.easeOutBounce (x, t*2-d, 0, c, d) * .5 + c*.5 + b;
}
});

function dropincontentbox(options){
this.closebutton='<div
style="position:absolute;top:4px;right:4px;cursor:pointer"></div>'
this.s=jQuery.extend({fx:'easeOutBounce', fxtime:500, freq:'always',
showduration:0, pos:['center','center'], deferred:0.5}, options)
var thisbox=this
this.s.source=(!$.isArray(this.s.source))? [this.s.source] : this.s.source
this.$closebutton=$(this.closebutton).hide().click(function()
{thisbox.hide()})
var loadbox=(this.s.deferred=="fullon"? false: true
this.s.freqispersist=!isNaN(parseInt(this.s.freq))
if ((this.s.freq=="session" || this.s.freqispersist) &&
dropincontentbox.routines.getCookie(this.s.source[0])){
loadbox=false
if (dropincontentbox.routines.getCookie(this.s.source[0]+'_freq')!
=this.s.freq){
dropincontentbox.routines.setCookie(this.s.source[0], '', -1)
loadbox=true
}
}
jQuery(function($){
thisbox.init($, thisbox.s, loadbox)
})
}

dropincontentbox.prototype={

show:function(pos) {
var $=jQuery, $contentbox=this.$contentbox.css({display:'block'}), s=this.s
if (typeof pos=="undefined")
var pos=s.pos
var winmeasure={w:$(window).width(), h:$(window).height(), left:$(
document).scrollLeft(), top:$(document).scrollTop()}
var boxmeasure={w:$contentbox.outerWidth(), h:$contentbox.outerHeight()}
var finalpos=[]
$.each(pos, function(i, val){
if (val<0){
finalpos[i]=(i==0)? winmeasure.left+winmeasure.w-boxmeasure.w+val :

```

```

winmeasure.top+winmeasure.h-boxmeasure.h+val
}
else if (val=="center"){
finalpos[i]=(i==0)? winmeasure.left+winmeasure.w/2-boxmeasure.w/2 :
winmeasure.top+winmeasure.h/2-boxmeasure.h/2
}
})
$contentbox.css({left:finalpos[0], top:winmeasure.top-boxmeasure.h-10,
visibility:'visible'}).animate({top:finalpos[1]}, s.fxduration, s.fx)
},

hide:function(){
this.$contentbox.hide()
this.$closebutton.hide()
},

init:function($, s, loadcheck){
var thisbox=this
this.$contentbox=(s.source.length==1)? $
(s.source[0]).css({position:'absolute', visibility:'hidden',
top:0}).addClass(s.cssclass) : ""
function selectiveshow(){
if (loadcheck==false)
return
thisbox.$contentbox.append(thisbox.$closebutton).hover(
function(){
thisbox.$closebutton.stop(true,true).fadeIn()
},
function(){
thisbox.$closebutton.stop(true,true).fadeOut()
}
)
function selectivesetcookie(){
if (s.freq=="session" || s.freqispersist){
dropincontentbox.routines.setCookie(s.source[0], 'yes', s.freq)
dropincontentbox.routines.setCookie(s.source[0]+'_freq', s.freq, s.freq)
}
}
if (s.deferred>0)
setTimeout(function(){thisbox.show(s.pos); selectivesetcookie()},
s.deferred*1000)
else if (s.deferred==0){
thisbox.show(s.pos)
selectivesetcookie()
}
if (s.showduration>0)
setTimeout(function(){thisbox.hide()}, s.deferred*1000+s.showduration*1000)
}
if (s.source.length==2){
$.ajax({
url: s.source[1].replace(/^http:\/\/\[^\]\/\]/i,
"http://"+window.location.hostname+"/"),
dataType:'html',
error:function(ajaxrequest){
alert('Error fetching Ajax content\nServer Response:
'+ajaxrequest.responseText)
},

```

```

success:function(content) {
thisbox.$contentbox=$
(content).addClass(s.cssclass).css({position:'absolute',
visibility:'hidden', top:0}).appendTo(document.body)
selectiveshow()
}
})

}
else{
selectiveshow()
}
}
}

dropincontentbox.routines={

getCookie:function(Name) {
var re=new RegExp(Name+"=[^;]+", "i");
if (document.cookie.match(re))
return document.cookie.match(re)[0].split("=")[1]
return null
},

setCookie:function(name, value, duration){
var expirestr='', expiredate=new Date()
if (typeof duration!="undefined"){
var offsetmin=parseInt(duration) * (/hr/i.test(duration)? 60 :
/day/i.test(duration)? 60*24 : 1)
expiredate.setMinutes(expiredate.getMinutes() + offsetmin)
expirestr="; expires=" + expiredate.toUTCString()
}
document.cookie = name+"="+value+"; path=/"+expirestr
}
}

var dropinbox2=new dropincontentbox({
source:'#reminder',
cssclass:'dropinbox dropinboxaltstyle drop-shadow lifted',
fx:'easeInExpo',
pos:[-20, -20],
deferred:1
})

</script>

```


U <BODY> smestite sledeće:

```

<div id="reminder">
Ovo je sadržaj okvira koji pada. Definisan je na stranici unutar DIV sa id =
"reminder". Ovaj DIV pada pomoću "easeInExpo" animacije.
Okvir neće nestati sve dok korisnik ne klikne na ikonicu koja služi za
to.
</div>

```

```
<a href="javascript:dropinbox2.show()">Neka pdne sadrzaj opet</a><br>
<a href="javascript:dropinbox2.hide()">Ukolni padajuci sadrzaj</a>
```

I naravno ikonica za brisanje može biti po vašoj volji, ali možete koristiti i našu: 

## Objasnjenje u formi

Jako zanimljiva skriptica, gotovo nezamenjljiva u formama. Evo kliknite da vidite o čemu se radi: [DEMONSTRACIJA](#).

U <HEAD> delu stranice pišete sledeći KOD:

```
<style type="text/css">
#hintbox{
position:absolute;
top: 0;
background-color: lightyellow;
width: 150px;
padding: 3px;
border:1px solid black;
font:normal 11px Verdana;
line-height:18px;
z-index:100;
border-right: 3px solid black;
border-bottom: 3px solid black;
visibility: hidden;
}

.hintanchor{
font-weight: bold;
color: navy;
margin: 3px 8px;
}
</style>

<script type="text/javascript">

var horizontal_offset="9px"

var vertical_offset="0"
var ie=document.all
var ns6=document.getElementById&&!document.all

function getposOffset(what, offsettype){
var totaloffset=(offsettype=="left"? what.offsetLeft : what.offsetTop;
var parentEl=what.offsetParent;
while (parentEl!=null){
totaloffset=(offsettype=="left"? totaloffset+parentEl.offsetLeft :
totaloffset+parentEl.offsetTop;
parentEl=parentEl.offsetParent;
}
return totaloffset;
}
```

```

function iecompattest(){
return (document.compatMode && document.compatMode!="BackCompat")?
document.documentElement : document.body
}

function clearbrowseredge(obj, whichedge){
var edgeoffset=(whichedge=="rightedge"? parseInt(horizontal_offset)*-1 :
parseInt(vertical_offset)*-1
if (whichedge=="rightedge"){
var windowedge=ie && !window.opera?
iecompattest().scrollLeft+iecompattest().clientWidth-30 :
window.pageXOffset+window.innerWidth-40
dropmenuobj.contentmeasure=dropmenuobj.offsetWidth
if (windowedge-dropmenuobj.x < dropmenuobj.contentmeasure)
edgeoffset=dropmenuobj.contentmeasure+obj.offsetWidth+parseInt(horizontal_of
fset)
}
else{
var windowedge=ie && !window.opera?
iecompattest().scrollTop+iecompattest().clientHeight-15 :
window.pageYOffset+window.innerHeight-18
dropmenuobj.contentmeasure=dropmenuobj.offsetHeight
if (windowedge-dropmenuobj.y < dropmenuobj.contentmeasure)
edgeoffset=dropmenuobj.contentmeasure-obj.offsetHeight
}
return edgeoffset
}

function showhint(menucontents, obj, e, tipwidth){
if ((ie||ns6) && document.getElementById("hintbox")){
dropmenuobj=document.getElementById("hintbox")
dropmenuobj.innerHTML=menucontents
dropmenuobj.style.left=dropmenuobj.style.top=-500
if (tipwidth!=""){
dropmenuobj.widthobj=dropmenuobj.style
dropmenuobj.widthobj.width=tipwidth
}
dropmenuobj.x=getposOffset(obj, "left")
dropmenuobj.y=getposOffset(obj, "top")
dropmenuobj.style.left=dropmenuobj.x-clearbrowseredge(obj, "rightedge")
+obj.offsetWidth+"px"
dropmenuobj.style.top=dropmenuobj.y-clearbrowseredge(obj, "bottomedge")+"px"
dropmenuobj.style.visibility="visible"
obj.onmouseout=hidetip
}
}

function hidetip(e){
dropmenuobj.style.visibility="hidden"
dropmenuobj.style.left="-500px"
}

function createhintbox(){
var divblock=document.createElement("div")
divblock.setAttribute("id", "hintbox")
document.body.appendChild(divblock)
}

```



```

}

if (window.addEventListener)
window.addEventListener("load", createhintbox, false)
else if (window.attachEvent)
window.attachEvent("onload", createhintbox)
else if (document.getElementById)
window.onload=createhintbox

</script>

```

U <BODY> smestite sledeće:

```

<form>
<b>Nadimak:</b>
<input type="text" class="test" />
<a href="#" class="hintanchor" onMouseover="showhint('Odaberite
korisni&#269;ko ime. Trebalo bi se sastojati samo od alfanumeri&#269;kih
znakova.', this, event, '150px')">[?]</a><br />
<b>Lozinka:</b>
  <input type="text" class="test" />
<a href="#" class="hintanchor" onMouseover="showhint('Unesite zeljenu
lozinku. <b>Mora</b> biti duzine 8 znaka ili duze.', this, event,
'200px')">[?]</a><br />
</form>

```

## Brojanje znakova

Takođe jako zanimljiva skriptica za unos teksta jer broji koliko je znakova ostalo za unos u formu. Evo kliknite da vidite o čemu se radi: [DEMONSTRACIJA](#).

U <HEAD> delu stranice pišete sledeći KOD:

```

<script type="text/javascript">

var fieldlimiter={

defaultoutput: "<b>[int]</b> Preostali broj slova za upis",

uncheckedkeycodes: /(8)|(13)|(16)|(17)|(18)/,

limitinput:function(e, config){
var e=window.event || e
var thefield=config.thefield
var keyunicode=e.charCode || e.keyCode
if (!this.uncheckedkeycodes.test(keyunicode)){
if (thefield.value.length>=config.maxlength){
if (e.preventDefault())
e.preventDefault()
return false
}
}
},

```

```

showlimit:function(config){
var thefield=config.thefield
var statusids=config.statusids
var charsleft=config.maxlength-thefield.value.length
if (charsleft<0)
thefield.value=thefield.value.substring(0, config.maxlength)
for (var i=0; i<statusids.length; i++){
var statusdiv=document.getElementById(statusids[i])
if (statusdiv)
statusdiv.innerHTML=this.defaultoutput.replace("[int]", Math.max(0,
charsleft))
}
config.onkeypress.call(thefield, config.maxlength, thefield.value.length)
},

cleanup:function(config){
for (var prop in config){
config[prop]=null
}
},

addEvent:function(targetarr, functionref, tasktype){
if (targetarr.length>0){
var target=targetarr.shift()
if (target.addEventListener)
target.addEventListener(tasktype, functionref, false)
else if (target.attachEvent)
target.attachEvent('on'+tasktype, function(){return functionref.call(target,
window.event)})
this.addEvent(targetarr, functionref, tasktype)
}
},

setup:function(config){
if (config.thefield){
config.onkeypress=config.onkeypress || function(){}
config.thefield.value=config.thefield.value
this.showlimit(config)
this.addEvent([window], function(e){fieldlimiter.showlimit(config)}, "load")
this.addEvent([window], function(e){fieldlimiter.cleanup(config)}, "unload")
this.addEvent([config.thefield], function(e){return
fieldlimiter.limitinput(e, config)}, "keypress")
this.addEvent([config.thefield], function(){fieldlimiter.showlimit(config)},
"keyup")
}
}
}

</script>

```

U <BODY> smestite sledeće:

```

<form name="sampleform">

<input type="text" name="george" style="width:250px" />
<div id="george-status"></div>

<p>
<textarea name="johndoe" id="johndoe" cols="30" rows="5"
style="width:250px"></textarea>
<div id="john-status"></div>

</form>

<script type="text/javascript">

fieldlimiter.setup({
thefield: document.sampleform.george,
maxlength: 10,
statusids: ["george-status"],
onkeypress:function(maxlength, curlength){
if (curlength<maxlength)
this.style.border="2px solid gray"
else
this.style.border="2px solid red"
}
})

fieldlimiter.setup({
thefield: document.getElementById("johndoe"),
maxlength: 30,
statusids: ["john-status"],
onkeypress:function(maxlength, curlength){
}
})

</script>

```

## Tekst editor

Ova skriptica služi za unos teksta ali koja omogućava stilizovanje i razno editovanje istog. Evo kliknite da vidite o čemu se radi: [DEMONSTRACIJA](#).

U <HEAD> delu stranice pišete sledeći KOD:

```

<script type="text/javascript">
// +-----+
// | Tekst editor V1.0                               |
// | Autor: Stankov Dejan [www.bubaj.com]             |
// | Besplatno dokle god su autorska prava netaknuta |
// +-----+

function SimpleTextEditor(id, objectId) {
if (!id || !objectId) { alert("SimpleTextEditor.constructor(id, objectId)
failed, two arguments are required"); }

```

```

var self = this;
this.id = id;
this.objectId = objectId;
this.frame;
this.viewSource = false;

this.path = "";
this.cssFile = "";
this.charset = "iso-8859-1";

this.editorHtml = "";
this.frameHtml = "";

this.textareaValue = "";

this.browser = {
  "ie": Boolean(document.body.currentStyle),
  "gecko" : (navigator.userAgent.toLowerCase().indexOf("gecko") != -1)
};

this.init = function() {
  if (document.getElementById && document.createElement && document.designMode
    && (this.browser.ie || this.browser.gecko)) {
    if (!document.getElementById(this.id)) { alert("SimpleTextEditor
    "+this.objectId+".init() failed, element '"+this.id+"' does not exist");
    return; }
    this.textareaValue = document.getElementById(this.id).value;
    var ste = document.createElement("div");
    document.getElementById(this.id).parentNode.replaceChild(ste,
    document.getElementById(this.id));
    ste.id = this.id+"-ste";
    ste.innerHTML = this.editorHtml ? this.editorHtml : this.getEditorHtml();
    var buttons = ste.getElementsByTagName("td");
    for (var i = 0; i < buttons.length; ++i) {
      if (buttons[i].className == "button") {
        buttons[i].id = this.id+'-button-'+i;
        buttons[i].onmouseover = function() { this.className = "button-hover"; }
        buttons[i].onmouseout = function() { this.className =
        this.className.replace(/button-hover(\s)?/, "button"); }
        buttons[i].onclick = function(id) { return function() { this.className =
        "button-hover button-click"; setTimeout(function()
        { document.getElementById(id).className =
        document.getElementById(id).className.replace(/(\s)?button-click/, ""); },
        100); } } (buttons[i].id);
      }
    }
    if (this.browser.ie) {
      this.frame = frames[this.id+"-frame"];
    } else if (this.browser.gecko) {
      this.frame = document.getElementById(this.id+"-frame").contentWindow;
    }
    this.frame.document.designMode = "on";
    this.frame.document.open();
    this.frame.document.write(this.frameHtml ? this.frameHtml :
    this.getFrameHtml());
    this.frame.document.close();
    insertHtmlFromTextarea();
  }
}

```

```

};

function lockUrls(s) {
if (self.browser.gecko) { return s; }
return s.replace(/href=["']([^\']*)*["']/g,
'href="simpletexteditor://simpletexteditor/$1"');
}

function unlockUrls(s) {
if (self.browser.gecko) { return s; }
return s.replace(/href=["']simpletexteditor:\\/\\/simpletexteditor\\/([^\']*)*
["']/g, 'href="$1"');
}

function insertHtmlFromTextarea() {
try { self.frame.document.body.innerHTML = lockUrls(self.textareaValue); }
catch (e) { setTimeout(insertHtmlFromTextarea, 10); }
}

this.getEditorHtml = function() {
var html = "";
html += '<input type="hidden" id="'+this.id+'" name="'+this.id+'"
value="">';
html += '<table class="ste" cellspacing="0" cellpadding="0">';
html += '<tr><td class="bar"><table id="'+this.id+'-buttons" cellspacing="0"
cellpadding="0"><tr>';
html += '<td><select
onchange="'+this.objectId+'.execCommand(\`formatblock\`,
this.value);this.selectedIndex=0;"><option value=""></option><option
value="<h1>">Heading 1</option><option value="<h2>">Heading
2</option><option value="<h3>">Heading 3</option><option
value="<p>">Paragraph</option><option
value="<pre>">Preformatted</option></select></td>';
html += '<td><div class="separator"></div></td>';
html += '<td class="button"></td>';
html += '<td class="button"></td>';
html += '<td class="button"></td>';
html += '<td><div class="separator"></div></td>';
html += '<td class="button"></td>';
html += '<td class="button"></td>';
html += '<td class="button"></td>';
html += '<td><div class="separator"></div></td>';
html += '<td class="button"></td>';
html += '<td class="button"></td>';
html += '<td><div class="separator"></div></td>';
html += '<td class="button"></td>';
html += '<td class="button"></td>';
html += '<td><div class="separator"></div></td>';
html += '<td class="button"></td>';
html += '<td class="button"></td>';
html += '<td><div class="separator"></div></td>';
html += '<td class="button"></td>';
html += '</tr></table></td></tr>';
html += '<tr><td class="frame"><iframe id="' + this.id + '-frame"
frameborder="0"></iframe></td></tr>';
html += '<tr><td class="source"><input id="' + this.id + '-viewSource"
type="checkbox" onclick="' + this.objectId + '.toggleSource()"> View
Source</td></tr>';
html += '</table>';
return html;
};

```

```

this.getFrameHtml = function() {
var html = "";
html += '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">';
html += '<html><head>';
html += '<meta http-equiv="Content-Type" content="text/html;
charset=' + this.charset + '">';
html += '<title>SimpleTextEditor frame</title>';
html += '<style type="text/css">pre { background-color: #eeeeee; padding:
0.75em 1.5em; border: 1px solid #dddddd; }</style>';
if (this.cssFile) { html += '<link rel="stylesheet" type="text/css"
href="' + this.cssFile + '">'; }
html += '<style type="text/css">html,body { cursor: text; } body { margin:
0.5em; padding: 0; }</style>';
html += '</head><body></body></html>';
return html;
};

```

```

this.openWindow = function(url, width, height) {
var x = (screen.width/2-width/2);
var y = (screen.height/2-height/2);
window.open(url, "",
"scrollbars=yes,width="+width+",height="+height+",screenX="+x)
+",screenY="+y+",left="+x+",top="+y);

```

```

};

this.toggleSource = function() {
var html, text;
if (this.browser.ie) {
if (!this.viewSource) {
html = this.frame.document.body.innerHTML;
this.frame.document.body.innerText = unlockUrls(html);
document.getElementById(this.id+"-buttons").style.visibility = "hidden";
this.viewSource = true;
} else {
text = this.frame.document.body.innerText;
this.frame.document.body.innerHTML = lockUrls(text);
document.getElementById(this.id+"-buttons").style.visibility = "visible";
this.viewSource = false;
}
} else if (this.browser.gecko) {
if (!this.viewSource) {
html = document.createTextNode(this.frame.document.body.innerHTML);
this.frame.document.body.innerHTML = "";
this.frame.document.body.appendChild(html);
document.getElementById(this.id+"-buttons").style.visibility = "hidden";
this.viewSource = true;
} else {
html = this.frame.document.body.ownerDocument.createRange();
html.selectNodeContents(this.frame.document.body);
this.frame.document.body.innerHTML = html.toString();
document.getElementById(this.id+"-buttons").style.visibility = "visible";
this.viewSource = false;
}
}
document.getElementById(this.id+"-viewSource").checked = this.viewSource ?
"checked" : "";
document.getElementById(this.id+"-viewSource").blur();
};

this.execCommand = function(cmd, value) {
if (cmd == "createlink" && !value) {
var url = prompt("Enter URL:", "");
if (url) {
this.frame.focus();
this.frame.document.execCommand("unlink", false, null);
if (this.browser.ie) this.frame.document.execCommand(cmd, false,
"simpletexteditor://simpletexteditor/"+url);
else if (this.browser.gecko) this.frame.document.execCommand(cmd, false,
url);
this.frame.focus();
}
} else if (cmd == "insertimage" && !value) {
var imageUrl = prompt("Enter Image URL:", "");
if (imageUrl) {
this.frame.focus();
this.frame.document.execCommand(cmd, false, imageUrl);
this.frame.focus();
}
} else {
this.frame.focus();
}
};

```

```

this.frame.document.execCommand(cmd, false, value);
this.frame.focus();
}
};

this.isOn = function() {
return Boolean(this.frame);
};

this.getContent = function() {
try { return unlockUrls(this.frame.document.body.innerHTML); } catch(e)
{ alert("SimpleTextEditor "+this.objectId+".getContent() failed"); }
};

this.submit = function() {
if (this.isOn()) {
if (this.viewSource) { this.toggleSource(); }
document.getElementById(this.id).value = this.getContent();
}
};
}
</script>

<style type="text/css">
.ste .bar { background: #ECE9D8; padding: 3px; border: #ACA899 1px; border-
style: solid solid none solid; }
.ste .frame { border: 1px solid; border-color: #716F64 #ECE9D8 #ECE9D8
#716F64; }
.ste .frame iframe { width: 500px; height: 300px; }
.ste img { border: 0; }
.ste .button { padding: 1px; border: #ECE9D8 1px solid; }
.ste .button-hover { padding: 1px; border: 1px solid; border-color: #ffffff
#ACA899 #ACA899 #ffffff; }
.ste .button-click { padding: 1px; border: 1px solid; border-color: #ACA899
#ffffff #ffffff #ACA899; }
.ste .separator { width: 0px; height: 18px; border-left: #aca899 1px solid;
border-right: #ffffff 1px solid; margin: 0 5px; }
.ste .source { padding-top: 5px; }
</style>

```

U <BODY> smestite sledeće:

```

<form action="" method="post">

<textarea id="body" name="body" cols="60" rows="10">
<h1>Heading One</h1> <h2 style="margin-right: 0px;"
dir="ltr">Heading Two</h2> <h3>Heading Three</h3>
<pre>Preformatted <strong>bold</strong>
<em>italic</em> <span style="text-decoration:
underline;">underline</span> <a
href="test2.php">test</a> test<br></pre>
<p>Paragraph</p> <p style="margin-right: 0px; text-align:
left;" dir="ltr">Normal </p> <ul> <li>List
1</li><li>List 2</li><li>List 3</li>
</ul> <ol> <li>Ordered List</li><li>Ordered
List</li><li>Ordered List</li><li>Ordered

```



```

List<li></li></ol></li></ol>
</textarea>

<script type="text/javascript">
var ste = new SimpleTextEditor("body", "ste");
ste.init();
</script>

<input type="submit" value="submit" onclick="ste.submit();">

</form>

```

Takodje u folderu gde ste smestili HTML fajl sa gornjim BODY tagom smeštate i HTML fajl pod nazivom **help.html** a koji ima sledeći KOD:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Help</title>
<style type="text/css">
body { background: #ECE9D8; margin: 1em; padding: 0; }
h1 { font-size: 125%; }
h2 { font-size: 100%; }
p, ul { margin: 0.5em 0; padding: 0; }
ul { list-style-type: none; }
</style>
</head>
<body>

<h1>Help</h1>

<h2>Dugmi<i>:</i></h2>

<ul>
<li>Ubaci link - radi samo nakon selektovanja teksta</li>
</ul>

<h2>Pre<i>:ice na tastaturi:</i></h2>

<ul>
<li>ctrl+c - copy (kopiranje)</li>
<li>ctrl+v - paste (lepljenje)</li>
<li>ctrl+x - cut (rezanje)</li>
<li>ctrl+z - undo (poni<i>tavanje)</i></li>
<li>ctrl+shift+z - redo (prepraviti)</li>
<li>shift+enter - break (pauza/lomiti)</li>
</ul>

<hr>




<p align="center">
<!-- DO NOT REMOVE THIS -->
(c) 2011 Stankov Dejan <br>

```

```
<a href="http://www.bubaj.com">Tekst editor</a>
</p>

</body>
</html>
```

Ikonice smeštate u folder **images** a ikonice su sledeće:

**B**    *I*       U

## Uvecanje slike

Ova skriptica služi za uvećane slike sa efektom. Evo kliknite da vidite o čemu se radi: [DEMONSTRACIJA](#).

U <HEAD> delu stranice pišete sledeći KOD:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js"></scri
pt>

<script type="text/javascript">
jQuery.noConflict()

jQuery.imageMagnify={
dsettings: {
magnifyby: 3, //faktor uvecanja slike
duration: 500, //trajanje animacije u milisekundama
imgopacity: 0.2
},
cursorcss: '',
zIndexcounter: 100,

refreshoffsets:function($window, $target, warpsell){
var $offsets=$target.offset()
var winattrs={x:$window.scrollLeft(), y:$window.scrollTop(), w:
$window.width(), h:$window.height()}
warpsell.attrs.x=$offsets.left
warpsell.attrs.y=$offsets.top
warpsell.newattrs.x=winattrs.x+winattrs.w/2-warpsell.newattrs.w/2
warpsell.newattrs.y=winattrs.y+winattrs.h/2-warpsell.newattrs.h/2
if (warpsell.newattrs.x<winattrs.x+5){
warpsell.newattrs.x=winattrs.x+5
}
else if (warpsell.newattrs.x+warpsell.newattrs.w > winattrs.x+winattrs.w){
warpsell.newattrs.x=winattrs.x+5
}
if (warpsell.newattrs.y<winattrs.y+5){
warpsell.newattrs.y=winattrs.y+5
}
},

magnify:function($, $target, options){
```

```

var setting={}
var setting=jQuery.extend(setting, this.dsettings, options)
var attrs=(options.thumbdimensions)? {w:options.thumbdimensions[0],
h:options.thumbdimensions[1]} : {w:$target.outerWidth(), h:
$target.outerHeight()}
var newattrs={}
newattrs.w=(setting.magnifyto)? setting.magnifyto :
Math.round(attrs.w*setting.magnifyby)
newattrs.h=(setting.magnifyto)? Math.round(attrs.h*newattrs.w/attrs.w) :
Math.round(attrs.h*setting.magnifyby)
$target.css('cursor', jQuery.imageMagnify.cursorcss)
if ($target.data('imgshell')){
$target.data('imgshell').$clone.remove()
$target.css({opacity:1}).unbind('click.magnify')
}
var $clone=$target.clone().css({position:'absolute', left:0, top:0,
visibility:'hidden', border:'1px solid gray',
cursor:'pointer'}).appendTo(document.body)
$clone.data('$relatedtarget', $target)
$target.data('imgshell', {$clone:$clone, attrs:attrs, newattrs:newattrs})
$target.bind('click.magnify', function(e){
var $this=$(this).css({opacity:setting.imgopacity})
var imageinfo=$this.data('imgshell')
jQuery.imageMagnify.refreshoffsets($(window), $this, imageinfo)
var $clone=imageinfo.$clone
$clone.stop().css({zIndex:++jQuery.imageMagnify.zIndexcounter,
left:imageinfo.attrs.x, top:imageinfo.attrs.y, width:imageinfo.attrs.w,
height:imageinfo.attrs.h, opacity:0, visibility:'visible', display:'block'})
.animate({opacity:1, left:imageinfo.newattrs.x, top:imageinfo.newattrs.y,
width:imageinfo.newattrs.w, height:imageinfo.newattrs.h}, setting.duration,
function(){
})
})
$clone.click(function(e){
var $this=$(this)
var imageinfo=$this.data('$relatedtarget').data('imgshell')
jQuery.imageMagnify.refreshoffsets($(window), $this.data('$relatedtarget'),
imageinfo)
$this.stop().animate({opacity:0, left:imageinfo.attrs.x,
top:imageinfo.attrs.y, width:imageinfo.attrs.w, height:imageinfo.attrs.h},
setting.duration,
function(){
$this.hide()
$this.data('$relatedtarget').css({opacity:1})
})
})
}
};

jQuery.fn.imageMagnify=function(options){
var $=jQuery
return this.each(function(){
var $imgref=$(this)
if (this.tagName!="IMG")
return true
if (parseInt($imgref.css('width'))>0 && parseInt($imgref.css('height'))>0 ||

```

```

options.thumbdimensions){
jQuery.imageMagnify.magnify($, $imgref, options)
}
else if (this.complete){
jQuery.imageMagnify.magnify($, $imgref, options)
}
else{
$(this).bind('load', function(){
jQuery.imageMagnify.magnify($, $imgref, options)
})
}
})
};

jQuery.fn.applyMagnifier=function(options){
var $=jQuery
return this.each(function(){
var $imgref=$(this)
if (this.tagName!="IMG")
return true

}))

};

jQuery(document).ready(function($){
var $targets=$('.magnify')
$targets.each(function(i){
var $target=$(this)
var options={}
if ($target.attr('data-magnifyto'))
options.magnifyto=parseFloat($target.attr('data-magnifyto'))
if ($target.attr('data-magnifyby'))
options.magnifyby=parseFloat($target.attr('data-magnifyby'))
if ($target.attr('data-magnifyduration'))
options.duration=parseInt($target.attr('data-magnifyduration'))
$target.imageMagnify(options)
})
var $triggers=$('a[rel^="magnify[\"']')
$triggers.each(function(i){
var $trigger=$(this)
var targetid=$trigger.attr('rel').match(/\[.+\\]/)[0].replace(/\[\\\"'\]/g, '')
$trigger.data('magnifyimageid', targetid)
$trigger.click(function(e){
$('#'+$(this).data('magnifyimageid')).trigger('click.magnify')
e.preventDefault()
})
})
})
})
</script>

```

U <BODY> smestite sledeće:

```




```

```
<p>
Klikni na sliku za uvećanje!
</p>
```

## Lepljivi boks

Ova skriptica služi da bi se pojavio recimo boks sa objašnjenjem kad se predje mišem preko nekog polja. Evo kliknite da vidite o čemu se radi: [DEMONSTRACIJA](#).

U <HEAD> delu stranice pišete sledeći KOD:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js"></scri
pt>

<script type="text/javascript">
var stickytooltip={
tooltipoffsets: [20, -30],
fadeinspeed: 200,
rightclickstick: true,
stickybordercolors: ["black", "darkred"],
stickynotice1: ["Pritisni \"s\"", "ili desni klik misa", "za poruku"],
stickynotice2: "Klikni da se sakrije prozorke",

isdocked: false,

positiontooltip:function($, $tooltip, e){
var x=e.pageX+this.tooltipoffsets[0], y=e.pageY+this.tooltipoffsets[1]
var tipw=$tooltip.outerWidth(), tiph=$tooltip.outerHeight(),
x=(x+tipw>$ (document).scrollLeft()+$(window).width())? x-tipw-
(stickytooltip.tooltipoffsets[0]*2) : x
y=(y+tiph>$ (document).scrollTop()+$(window).height())? $
(document).scrollTop()+$(window).height()-tiph-10 : y
$tooltip.css({left:x, top:y})
},

showbox:function($, $tooltip, e){
$tooltip.fadeIn(this.fadeinspeed)
this.positiontooltip($, $tooltip, e)
},

hidebox:function($, $tooltip){
if (!this.isdocked){
$tooltip.stop(false, true).hide()
$tooltip.css({borderColor:'black'}).find('.stickystatus:eq(0)').css({backgro
und:this.stickybordercolors[0]}).html(this.stickynotice1)
}
},

docktooltip:function($, $tooltip, e){
this.isdocked=true
$tooltip.css({borderColor:'darkred'}).find('.stickystatus:eq(0)').css({backg
round:this.stickybordercolors[1]}).html(this.stickynotice2)
},
```

```

init:function(targetselector, tipid){
jQuery(document).ready(function($){
var $targets=$(targetselector)
var $tooltip=$('#'+tipid).appendTo(document.body)
if ($targets.length==0)
return
var $alltips=$tooltip.find('div.atip')
if (!stickytooltip.rightclickstick)
stickytooltip.stickynotice1[1]=''
stickytooltip.stickynotice1=stickytooltip.stickynotice1.join(' ')
stickytooltip.hidebox($, $tooltip)
$targets.bind('mouseenter', function(e){
$alltips.hide().filter('#'+$(this).attr('data-tooltip')).show()
stickytooltip.showbox($, $tooltip, e)
})
$targets.bind('mouseleave', function(e){
stickytooltip.hidebox($, $tooltip)
})
$targets.bind('mousemove', function(e){
if (!stickytooltip.isdocked){
stickytooltip.positiontooltip($, $tooltip, e)
}
})
$tooltip.bind("mouseenter", function(){
stickytooltip.hidebox($, $tooltip)
})
$tooltip.bind("click", function(e){
e.stopPropagation()
})
$(this).bind("click", function(e){
if (e.button==0){
stickytooltip.isdocked=false
stickytooltip.hidebox($, $tooltip)
}
})
$(this).bind("contextmenu", function(e){
if (stickytooltip.rightclickstick && $
(e.target).parents().andSelf().filter(targetselector).length==1){
stickytooltip.docktooltip($, $tooltip, e)
return false
}
})
$(this).bind('keypress', function(e){
var keyunicode=e.charCode || e.keyCode
if (keyunicode==115){ //za "s" dugme
stickytooltip.docktooltip($, $tooltip, e)
}
})
})
}
}

stickytooltip.init("[data-tooltip]", "mystickytooltip")
</script>

<style type="text/css">

```

```

.stickytooltip{
box-shadow: 5px 5px 8px #818181; /*shadow for CSS3 capable browsers.*/
-webkit-box-shadow: 5px 5px 8px #818181;
-moz-box-shadow: 5px 5px 8px #818181;
display:none;
position:absolute;
display:none;
border:5px solid black; /*Border around tooltip*/
background:white;
z-index:3000;
}

.stickytooltip .stickystatus{ /*Style for footer bar within tooltip*/
background:black;
color:white;
padding-top:5px;
text-align:center;
font:bold 11px Arial;
}
</style>

```

U <BODY> delu recimo možete pisati nešto ovako:

```

<p>Neki sadržaj...</p>

<p><a href="http://en.wikipedia.org/wiki/Thailand" data-
tooltip="sticky1">Tajland</a></p>
<p><a href="http://en.wikipedia.org/wiki/British_columbia" data-
tooltip="sticky2">Britanska Kolumbija</a></p>
<p></p>

<p>Neki sadržaj...</p>

<div id="mystickytooltip" class="stickytooltip">
<div style="padding:5px">

<div id="sticky1" class="atip" style="width:200px">
<br />
Tajland se ponosi sa ovim najluksuznijim odmaralistima u Aziji
</div>

<div id="sticky2" class="atip" style="width:262px">
<br />
BC je najzapadnija Kanadska provincija i poznata je po svojim prirodnim
lepotama.<b>
<a href="http://en.wikipedia.org/wiki/Vancouver">Vankuver</a></b>
je BC najveći grad.
</div>

<div id="sticky3" class="atip">

</div>

</div>

```

```
<div class="stickystatus"></div>
</div>
```

# Skrolovan tekst 1

Ova skriptica služi da bi se pojavio recimo boks sa objašnjenjem kad se predje mišem preko nekog polja. Evo kliknite da vidite o čemu se radi: [DEMONSTRACIJA](#).

U <HEAD> delu stranice pišete sledeći KOD:

```
<style type="text/css">
#pscroller1{
width: 200px;
height: 100px;
border: 1px solid black;
padding: 5px;
background-color: lightyellow;
}

#pscroller2{
width: 350px;
height: 20px;
border: 1px solid black;
padding: 3px;
}

#pscroller2 a{
text-decoration: none;
}

.someclass{ //class to apply to your scroller(s) if desired
}
</style>

<script type="text/javascript">

var pausecontent=new Array()
pausecontent[0]='<a href="http://www.bubaj.com">web izrada</a><br />Najbolji
sajt za edukaciju u vezi izrade sajtova!'
pausecontent[1]='<a href="http://www.bubaj.com">Tematski forum</a><br
/>Forum u vezi svih pitanja za izradu sajtova'
pausecontent[2]='<a href="http://www.bubaj.com" target="_new">CSS
edukacija</a><br />Sve o CSS do najsitnijih detalja'

var pausecontent2=new Array()
pausecontent2[0]='<a href="http://www.bubaj.com">Najbolji sajt na
Srpskom</a>'
pausecontent2[1]='<a href="http://www.bubaj.com">Uvek sveze informacije</a>'
pausecontent2[2]='<a href="http://www.bubaj.com">Kompletne informacije
primenljive</a>'

function pausescroller(content, divId, divClass, delay){
```



```

this.content=content
this.tickerid=divId
this.delay=delay
this.mouseoverBol=0
this.hiddendivpointer=1
document.write('<div id="'+divId+'" class="'+divClass+'" style="position:
relative; overflow: hidden"><div class="innerDiv" style="position: absolute;
width: 100%" id="'+divId+'1">'+content[0]+'</div><div class="innerDiv"
style="position: absolute; width: 100%; visibility: hidden"
id="'+divId+'2">'+content[1]+'</div></div>')
var scrollerinstance=this
if (window.addEventListener)
window.addEventListener("load", function(){scrollerinstance.initialize()},
false)
else if (window.attachEvent)
window.attachEvent("onload", function(){scrollerinstance.initialize()})
else if (document.getElementById)
setTimeout(function(){scrollerinstance.initialize()}, 500)
}

pausescroller.prototype.initialize=function(){
this.tickerdiv=document.getElementById(this.tickerid)
this.visiblediv=document.getElementById(this.tickerid+"1")
this.hiddendiv=document.getElementById(this.tickerid+"2")
this.visibledivtop=parseInt(pausescroller.getCSSpadding(this.tickerdiv))

this.visiblediv.style.width=this.hiddendiv.style.width=this.tickerdiv
.offsetWidth-(this.visibledivtop*2)+"px"
this.getinline(this.visiblediv, this.hiddendiv)
this.hiddendiv.style.visibility="visible"
var scrollerinstance=this
document.getElementById(this.tickerid).
onmouseover=function(){scrollerinstance.mouseoverBol=1}
document.getElementById(this.tickerid).
onmouseout=function(){scrollerinstance.mouseoverBol=0}
if (window.attachEvent)
window.attachEvent("onunload", function(){scrollerinstance.tickerdiv.
onmouseover=scrollerinstance.tickerdiv.onmouseout=null})
setTimeout(function(){scrollerinstance.animateup()}, this.delay)
}

pausescroller.prototype.animateup=function(){
var scrollerinstance=this
if (parseInt(this.hiddendiv.style.top)>(this.visibledivtop+5)){
this.visiblediv.style.top=parseInt(this.visiblediv.style.top)-5+"px"
this.hiddendiv.style.top=parseInt(this.hiddendiv.style.top)-5+"px"
setTimeout(function(){scrollerinstance.animateup()}, 50)
}
else{
this.getinline(this.hiddendiv, this.visiblediv)
this.swapdivs()
setTimeout(function(){scrollerinstance.setmessage()}, this.delay)
}
}

pausescroller.prototype.swapdivs=function(){

```

```

var tempcontainer=this.visiblediv
this.visiblediv=this.hiddendiv
this.hiddendiv=tempcontainer
}

pausescroller.prototype.getinline=function(div1, div2){
div1.style.top=this.visibledivtop+"px"
div2.style.top=Math.max(div1.parentNode.offsetHeight, div1.offsetHeight)
+"px"
}

pausescroller.prototype.setmessage=function(){
var scrollerinstance=this
if (this.mouseoverBol==1)
setTimeout(function(){scrollerinstance.setmessage()}, 100)
else{
var i=this.hiddendivpointer
var ceiling=this.content.length
this.hiddendivpointer=(i+1>ceiling-1)? 0 : i+1
this.hiddendiv.innerHTML=this.content[this.hiddendivpointer]
this.animateup()
}
}

pausescroller.getCSSpadding=function(tickerobj){
if (tickerobj.currentStyle)
return tickerobj.currentStyle["paddingTop"]
else if (window.getComputedStyle)
return window.getComputedStyle(tickerobj, "").getPropertyValue("padding-top")
else
return 0
}
</script>

```

U <BODY> delu recimo možete pisati nešto ovako:

```

<script type="text/javascript">
new pausescroller(pausecontent, "pscroller1", "someclass", 3000)
document.write("<br />")
new pausescroller(pausecontent2, "pscroller2", "someclass", 2000)
</script>

```

## Skrolovan tekst 2

Ova skriptica služi za skrolovanje teksta. Evo kliknite da vidite o čemu se radi: [DEMONSTRACIJA](#).

U <BODY> delu pišete sledeći KOD:

```

<script language="JavaScript1.2">

var marqueewidth="300px" //margine duzina

```

```

var marqueeheight="25px" //margine sirina
var marqueespeed=2 //brzina skrolovanja (1-10)
var marqueebgcolor="#DEFDD9" //boja pozadine dela gde se skroluje
var pauseit=1 //Pauza ako se dodje misem iznad polja (0=ne, 1=da)
var marqueecontent='<nobr>Hvala sto posecujete nas sajt</nobr>' //Poruka
koja se skroluje

marqueespeed=(document.all)? marqueeSpeed : Math.max(1, marqueeSpeed-1)
var copyspeed=marqueespeed
var pausespeed=(pauseit==0)? copyspeed: 0
var iedom=document.all||document.getElementById
if (iedom)
document.write('<span id="temp"
style="visibility:hidden;position:absolute;top:-100px;left:-9000px">'
+marqueecontent+'</span>')
var actualwidth=''
var cross_marquee, ns_marquee

function populate(){
if (iedom){
cross_marquee=document.getElementById?
document.getElementById("iemarquee") : document.all.iemarquee
cross_marquee.style.left=parseInt(marqueeWidth)+8+"px"
cross_marquee.innerHTML=marqueecontent
actualwidth=document.all? temp.offsetWidth :
document.getElementById("temp").offsetWidth
}
else if (document.layers){
ns_marquee=document.ns_marquee.document.ns_marquee2
ns_marquee.left=parseInt(marqueeWidth)+8
ns_marquee.document.write(marqueecontent)
ns_marquee.document.close()
actualwidth=ns_marquee.document.width
}
leftTime=setInterval("scrollmarquee()",20)
}
window.onload=populate

function scrollmarquee(){
if (iedom){
if (parseInt(cross_marquee.style.left)>(actualwidth*(-1)+8))
cross_marquee.style.left=parseInt(cross_marquee.style.left)-copyspeed+"px"
else
cross_marquee.style.left=parseInt(marqueeWidth)+8+"px"
}
else if (document.layers){
if (ns_marquee.left>(actualwidth*(-1)+8))
ns_marquee.left-=copyspeed
else
ns_marquee.left=parseInt(marqueeWidth)+8
}
}

if (iedom||document.layers){
with (document){
document.write('<table border="0" cellspacing="0" cellpadding="0"><td>')

```

```

if (iedom){
write('<div style="position:relative;width:'+marqueewidth+';height:'+
marqueeheight+';overflow:hidden">')
write('<div style="position:absolute;width:'+marqueewidth+';height:'+
marqueeheight+';background-color:'+marqueebgcolor+'"
onMouseover="copyspeed=pausespeed" onMouseout="copyspeed=marqueespeed">')
write('<div id="iemarquee"
style="position:absolute;left:0px;top:0px"></div>')
write('</div></div>')
}
else if (document.layers){
write('<ilayer width='+marqueewidth+' height='+marqueeheight+'
name="ns_marquee" bgColor='+marqueebgcolor+'>')
write('<layer name="ns_marquee2" left=0 top=0
onMouseover="copyspeed=pausespeed"
onMouseout="copyspeed=marqueespeed"></layer>')
write('</ilayer>')
}
document.write('</td></table>')
}
}
</script>

```

## Skrolovanje slike i teksta

Ova skriptica služi slika ili teksta. Evo kliknite da vidite o čemu se radi: [DEMONSTRACIJA](#).

U <HEAD> delu stranice pišete sledeći KOD:

```

<script type="text/javascript">
function marqueeInit(config){
if(!document.createElement) return;
marqueeInit.ar.push(config);
marqueeInit.run(config.uniqueid);
}

(function(){
if(!document.createElement) return;
marqueeInit.ar = [];
document.write('<style type="text/css">.marquee{white-
space:nowrap;overflow:hidden;visibility:hidden;}' +
'#marq_kill_marg_bord{border:none!important;margin:0!important;}</style>');
var c = 0, tTRE = [/^\s*$/, /\s*$/, /\s*$/, /^[^\s]+$/],
req1 = {'position': 'relative', 'overflow': 'hidden'}, defaultconfig = {
style: { //default style object for marquee containers without configured
style
'margin': '0 auto'
},
direction: 'left',
inc: 2, // Brzina skrolovanja
mouse: 'pause' // definisanje sta se radi kad predjemo misem preko
skrolovanja ('pause' 'cursor driven' or false)
}, dash, ie = false, oldie = 0, ie5 = false, iever = 0;

```

```

if(!ie5){
dash = /-(.)/g;
function toHump(a, b){return b.toUpperCase();};
String.prototype.encamel = function(){return this.replace(dash, toHump);};
}

if(ie && iever < 8){
marqueeInit.table = [];
window.attachEvent('onload', function(){
marqueeInit.OK = true;
for(var i = 0; i < marqueeInit.table.length; ++i)
marqueeInit.run(marqueeInit.table[i]);
});
}

function intable(el){
while((el = el.parentNode))
if(el.tagName && el.tagName.toLowerCase() === 'table')
return true;
return false;
};

marqueeInit.run = function(id){
if(ie && !marqueeInit.OK && iever < 8 &&
intable(document.getElementById(id))){
marqueeInit.table.push(id);
return;
}
if(!document.getElementById(id))
setTimeout(function(){marqueeInit.run(id);}, 300);
else
new Marq(c++, document.getElementById(id));
}

function trimTags(tag){
var r = [], i = 0, e;
while((e = tag.firstChild) && e.nodeType === 3 && tTRE[0].test(e.nodeValue))
tag.removeChild(e);
while((e = tag.lastChild) && e.nodeType === 3 && tTRE[0].test(e.nodeValue))
tag.removeChild(e);
if((e = tag.firstChild) && e.nodeType === 3)
e.nodeValue = e.nodeValue.replace(tTRE[1], '');
if((e = tag.lastChild) && e.nodeType === 3)
e.nodeValue = e.nodeValue.replace(tTRE[2], '');
while((e = tag.firstChild))
r[i++] = tag.removeChild(e);
return r;
}

function randthem(tag){
var els = oldie? tag.all : tag.getElementsByTagName('*'), i = els.length -
1, childels = [], newels = [];
for (i; i > -1; --i){
if(els[i].parentNode === tag){
childels.push(els[i]);
newels.push(els[i].cloneNode(true));
}
}
}

```

```

}
newels.sort(function(){return 0.5 - Math.random();});
i = childels.length - 1;
for (i; i > -1; --i){
tag.replaceChild(newels[i], childels[i]);
}
}

function Marq(c, tag){
var p, u, s, a, ims, ic, i, marqContent, cObj = this;
this.mq = marqueeInit.ar[c];
if(this.mq.random){
randthem(tag);
}
for (p in defaultconfig)
if((this.mq.hasOwnProperty && !this.mq.hasOwnProperty(p)) || (!
this.mq.hasOwnProperty && !this.mq[p]))
this.mq[p] = defaultconfig[p];
this.mq.style.width = !this.mq.style.width ||
isNaN(parseInt(this.mq.style.width))? '100%' : this.mq.style.width;
if(!tag.getElementsByTagName('img')[0])
this.mq.style.height = !this.mq.style.height ||
isNaN(parseInt(this.mq.style.height))? tag.offsetHeight + 3 + 'px' :
this.mq.style.height;
else
this.mq.style.height = !this.mq.style.height ||
isNaN(parseInt(this.mq.style.height))? 'auto' : this.mq.style.height;
u = this.mq.style.width.split(/\d/);
this.cw = this.mq.style.width? [parseInt(this.mq.style.width), u[u.length -
1]] : ['a'];
marqContent = trimTags(tag);
tag.className = tag.id = '';
tag.removeAttribute('class', 0);
tag.removeAttribute('id', 0);
if(ie)
tag.removeAttribute('className', 0);
tag.appendChild(tag.cloneNode(false));
tag.className = ['marquee', c].join('');
tag.style.overflow = 'hidden';
this.c = tag.firstChild;
this.c.appendChild(this.c.cloneNode(false));
this.c.style.visibility = 'hidden';
a = [[req1, this.c.style], [this.mq.style, this.c.style]];
for (i = a.length - 1; i > -1; --i)
for (p in a[i][0])
if((a[i][0].hasOwnProperty && a[i][0].hasOwnProperty(p)) || (!a[i]
[0].hasOwnProperty))
a[i][1][p.encamel()] = a[i][0][p];
this.m = this.c.firstChild;
if(this.mq.mouse === 'pause'){
this.c.onmouseover = function(){cObj.mq.stopped = true;};
this.c.onmouseout = function(){cObj.mq.stopped = false;};
}
this.m.style.position = 'absolute';
this.m.style.left = '-10000000px';
this.m.style.whiteSpace = 'nowrap';
if(ie5) this.c.firstChild.appendChild((this.m =

```

```

document.createElement('nobr')));
if(!this.mq.noAddedSpace)
this.m.appendChild(document.createTextNode('\xa0'));
for(i = 0; marqContent[i]; ++i)
this.m.appendChild(marqContent[i]);
if(ie5) this.m = this.c.firstChild;
ims = this.m.getElementsByTagName('img');
if(ims.length){
for(ic = 0, i = 0; i < ims.length; ++i){
ims[i].style.display = 'inline';
if(!ims[i].alt && !this.mq.noAddedAlt){
ims[i].alt = (tTRE[3].exec(ims[i].src)) || ('Image #' + [i + 1]);
if(!ims[i].title){ims[i].title = '';}
}
ims[i].style.display = 'inline';
ims[i].style.verticalAlign = ims[i].style.verticalAlign || 'top';
if(typeof ims[i].complete === 'boolean' && ims[i].complete)
ic++;
else {
ims[i].onload = ims[i].onerror = function(){
if(++ic === ims.length)
cObj.setup(c);
};
}
if(ic === ims.length)
this.setup(c);
}
}
else this.setup(c)
}

Marq.prototype.setup = function(c){
if(this.mq.setup) return;
this.mq.setup = this;
var s, w, cObj = this, exit = 10000;
if(this.c.style.height === 'auto')
this.c.style.height = this.m.offsetHeight + 4 + 'px';
this.c.appendChild(this.m.cloneNode(true));
this.m = [this.m, this.m.nextSibling];
if(this.mq.mouse === 'cursor driven'){
this.r = this.mq.neutral || 16;
this.sinc = this.mq.inc;
this.c.onmousemove = function(e){cObj.mq.stopped = false;
cObj.directspeed(e)};
if(this.mq.moveatleast){
this.mq.inc = this.mq.moveatleast;
if(this.mq.savedirection){
if(this.mq.savedirection === 'reverse'){
this.c.onmouseout = function(e){
if(cObj.contains(e)) return;
cObj.mq.inc = cObj.mq.moveatleast;
cObj.mq.direction = cObj.mq.direction === 'right'? 'left' : 'right';}};
} else {
this.mq.savedirection = this.mq.direction;
this.c.onmouseout = function(e){
if(cObj.contains(e)) return;

```

```

cObj.mq.inc = cObj.mq.moveatleast;
cObj.mq.direction = cObj.mq.savedirection;};
}
} else
this.c.onmouseout = function(e){if(!cObj.contains(e)) cObj.mq.inc =
cObj.mq.moveatleast;};
}
else
this.c.onmouseout = function(e){if(!cObj.contains(e)) cObj.slowdeath();};
}
this.w = this.m[0].offsetWidth;
this.m[0].style.left = 0;
this.c.id = 'marq_kill_marg_bord';
this.m[0].style.top = this.m[1].style.top = Math.floor((this.c.offsetHeight
- this.m[0].offsetHeight) / 2 - oldie) + 'px';
this.c.id = '';
this.c.removeAttribute('id', 0);
this.m[1].style.left = this.w + 'px';
s = this.mq.moveatleast? Math.max(this.mq.moveatleast, this.sinc) :
(this.sinc || this.mq.inc);
while(this.c.offsetWidth > this.w - s && --exit){
w = isNaN(this.cw[0])? this.w - s : --this.cw[0];
if(w < 1 || this.w < Math.max(1, s)){break;}
this.c.style.width = isNaN(this.cw[0])? this.w - s + 'px' : --this.cw[0] +
this.cw[1];
}
this.c.style.visibility = 'visible';
this.runit();
}

Marq.prototype.slowdeath = function(){
var cObj = this;
if(this.mq.inc){
this.mq.inc -= 1;
this.timer = setTimeout(function(){cObj.slowdeath();}, 100);
}
}

Marq.prototype.runit = function(){
var cObj = this, d = this.mq.direction === 'right'? 1 : -1;
if(this.mq.stopped || this.mq.stopMarquee){
setTimeout(function(){cObj.runit();}, 300);
return;
}
if(this.mq.mouse != 'cursor driven')
this.mq.inc = Math.max(1, this.mq.inc);
if(d * parseInt(this.m[0].style.left) >= this.w)
this.m[0].style.left = parseInt(this.m[1].style.left) - d * this.w + 'px';
if(d * parseInt(this.m[1].style.left) >= this.w)
this.m[1].style.left = parseInt(this.m[0].style.left) - d * this.w + 'px';
this.m[0].style.left = parseInt(this.m[0].style.left) + d * this.mq.inc +
'px';
this.m[1].style.left = parseInt(this.m[1].style.left) + d * this.mq.inc +
'px';
setTimeout(function(){cObj.runit();}, 30 + (this.mq.addDelay || 0));
}

```



```

Marq.prototype.directspeed = function(e) {
e = e || window.event;
if(this.timer) clearTimeout(this.timer);
var c = this.c, w = c.offsetWidth, l = c.offsetLeft, mp = (typeof e.pageX
=== 'number'?
e.pageX : e.clientX + document.body.scrollLeft +
document.documentElement.scrollLeft) - l,
lb = (w - this.r) / 2, rb = (w + this.r) / 2;
while((c = c.offsetParent)) mp -= c.offsetLeft;
this.mq.direction = mp > rb? 'left' : 'right';
this.mq.inc = Math.round((mp > rb? (mp - rb) : mp < lb? (lb - mp) : 0) / lb
* this.sinc);
}

Marq.prototype.contains = function(e) {
if(e && e.relatedTarget) {var c = e.relatedTarget; if(c === this.c) return
true;
while ((c = c.parentNode)) if(c === this.c) return true;}
return false;
}

function resize(){
for(var s, w, m, i = 0; i < marqueeInit.ar.length; ++i){
if(marqueeInit.ar[i] && marqueeInit.ar[i].setup){
m = marqueeInit.ar[i].setup;
s = m.mq.moveatleast? Math.max(m.mq.moveatleast, m.sinc) : (m.sinc ||
m.mq.inc);
m.c.style.width = m.mq.style.width;
m.cw[0] = m.cw.length > 1? parseInt(m.mq.style.width) : 'a';
while(m.c.offsetWidth > m.w - s){
w = isNaN(m.cw[0])? m.w - s : --m.cw[0];
if(w < 1){break;}
m.c.style.width = isNaN(m.cw[0])? m.w - s + 'px' : --m.cw[0] + m.cw[1];
}
}
}
}

if (window.addEventListener)
window.addEventListener('resize', resize, false);
else if (window.attachEvent)
window.attachEvent('onresize', resize);

})();
</script>

```

U <BODY> delu recimo možete pisati nešto ovako:

```

<div class="marquee" id="mycrawler">
Zelite napraviti sajt? Onda je pravo mesto na internetu za vas
www.bubaj.com!!
</div>

<script type="text/javascript">
marqueeInit({

```

```

uniqueid: 'mycrawler',
style: {
  'padding': '5px',
  'width': '450px',
  'background': 'lightyellow',
  'border': '1px solid #CC3300'
},
inc: 8, //brzina skrolovanja
mouse: 'cursor driven', //"mouseover" ('pause' 'cursor driven' or false)
moveatleast: 4,
neutral: 150,
savedirection: true
});
</script>

<div class="marquee" id="mycrawler2">
  

</div>

<script type="text/javascript">
marqueeInit({
uniqueid: 'mycrawler2',
style: {
  'padding': '2px',
  'width': '600px',
  'height': '180px'
},
inc: 5, //brzina skrolovanja
mouse: 'cursor driven', //"mouseover" ('pause' 'cursor driven' or false)
moveatleast: 2,
neutral: 150,
savedirection: true,
random: true
});
</script>

```

## Velicina fonta

Ova skriptica služi da automatski uvecava sav tekst na stranici ili da smanjuje. Evo kliknite da vidite o čemu se radi: [DEMONSTRACIJA](#).

U <HEAD> delu stranice pišete sledeći KOD:

```

<script type="text/javascript">
var tgs = new Array( 'div','td','tr'); //Tagovi u kojima se javljaju efekti.
Mogu se dodavati novi tagovi ili da se brisu.

var szs = new Array( 'xx-small','x-small','small', 'medium','large','x-
large','xx-large' ); //Spektar velicine fonta:
var startSz = 2;

function ts(trgt, inc) {
if (!document.getElementById) return

```

```

var d = document, cEl = null, sz = startSz, i, j, cTags;

sz += inc;
if ( sz < 0 ) sz = 0;
if ( sz > 6 ) sz = 6;
startSz = sz;

if ( !( cEl = d.getElementById( trgt ) ) ) cEl =
d.getElementsByTagName( trgt )[ 0 ];

cEl.style.fontSize = szs[ sz ];

for ( i = 0 ; i < tgs.length ; i++ ) {
cTags = cEl.getElementsByTagName( tgs[ i ] );
for ( j = 0 ; j < cTags.length ; j++ ) cTags[ j ].style.fontSize =
szs[ sz ];
}
}
</script>

```

U <BODY> delu recimo možete pisati nešto ovako:

```

<b><font size="4">Font: </font>
<a href="javascript:ts('body', 1)"><font size="6">+</font></a>
<a href="javascript:ts('body', -1)"><font size="6">--</font></a>
</b>

<br><br>
Slobodno klikci vise puta na minus za povecanje fonta, ili znak minus za
smanjenje!

```