1. Uvod u JavaScript

JavaScript se koristi na milion *Web* stranica da bi se poboljšalo dizajniranje, proverile forme, detektovali *browser*-i, kreirali *cookie*-ji, i mnogo više.
JavaScript je najpopularniji skript jezik na internetu, i funkcioniše na svim poznatijim *browser*-ima, kao što su *Internet Explorer, Mozilla, Firefox, Netscape, Opera.*

1.1. Šta je JavaScript?

- JavaScript je dizajniran da bi se dodala interaktivnost HTML stranama
- JavaScript je skript jezik (jednostavan programski jezik)
- JavaScript se sastaji od linija koda koji može da se izvršava
- JavaScript se obično direktno ugrađuje u HTML strane
- JavaScript je interpreterski jezik (što znači da se izvršava bez prethodnog kompajliranja)
- JavaScript nije licenciran jezik

1.2. Da li je JavaScript isto što i Java?

NE!
Java i JavaScript su po konceptu i načinu programiranja dva potpuno različita jezika.
Java (koju je razvio *Sun Microsystems*) je kompleksniji programski jezik, u istoj kategoriji kao C and C++.

1.3. Šta JavaScript može?

- **JavaScript daje HTML dizajnerima alat za programiranje** - HTML autori obično nisu programeri, ali JavaScript je skript jezik sa veoma jednostavnom sintaksom.
- **JavaScript može da dinamički ubaci kod u HTML stranu –** sledeća JavaScript naredba: `document.write("<h1>" + name + "</h1>")` može da napiše promenljiv tekst koji se nalazi u promenljivoj `name` na HTML strani
- **JavaScript može da reaguje na događaje** - JavaScript može da se podesi tako da se izvrši kad se nešto desi, npr. kad se strana učita ili kad korisnik klikne na HTML element
- **JavaScript može da pročita ili ispiše HTML elemente** - JavaScript može da pročita i da promeni sadržaj HTML elementa
- **JavaScript može da se koristi za proveru ispravnosti unetih podataka** - JavaScript može da se koristi za proveru ispravnosti podataka unetih u formu, da proveri ispravnost podataka pre nego što se pošalju serveru
- **JavaScript može da se koristi za detektovane *browser*-a korisnika** - JavaScript može da se koristi za detektovanje *browser*-a iu zavisnosti od *browser*-a, učitavanje strane specijalno dizajnirane za taj *browser*
- **JavaScript može da se koristi za kreiranje *cookie*-ja** - JavaScript može da se koristi za čuvanje i vraćanje informacija o računaru posetioca

2. JavaScript kako da ...

HTML tag `<script>` se koristi za umetanje *JavaScript*-a u HTML stranu.

2.1. Primeri

Primer 2.1 Ispisivanje teksta

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

Primer 2.2 Ispisivanje teksta sa formatiranjem

```
<html>
<body>
<script type="text/javascript">
document.write("<h1>Hello World!</h1>")
</script>
</body>
</html>
```

2.2. Kako da ubaciteJavaScript u HTML stranu

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

Ovaj kod će izgledati na strani:

Hello World!

Objašnjenje primera

Da bi ubacili *JavaScript* u HTML stranu, koristimo tag `<script>` (takođe, koristimo atribut `type` da bi definisali skript jezik).
Dakle, `<script type="text/javascript">` i `</script>` definišu gde počinje i završava se *JavaScript*:

```
<html>
<body>
<script type="text/javascript">
...
</script>
</body>
</html>
```

Reči **`document.write`** su standardna JavaScript komanda za pisanje po stranici.
Ako komandu `document.write` ubacite između tagova `<script type="text/javascript">` i `</script>` , *browser* će prepoznati da se radi o JavaScript komandi i izvršiće naredbu. U ovom slučaju browser će napisati tekst Hello World!:

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

Primedba: da nismo uneli tag `<script>`, browser bi celu liniju `document.write("Hello World!")` tretirao kao jednostavan tekst, i jednostavno bi je celu napisao na stranici.

2.3. Završavanje komande tačkom i zarezom?

Tradicionalni programski jezici, kao što je C++ ili Java, svaku instrukciju završavaju tačkom i zarezom. Mnogi programeri nastavljaju ovu naviku, ali je ona opciona u JavaScript-u. Obavezna je samo kad pišete više od jedne naredbe u istoj liniji.

2.4. Kako da radimo sa starijim browser-ima

Browser-i koji ne podržavaju JavaScript, prikazuju ga kao sadržaj strane. Da bi ih u tome sprečili, koristimo tag za umetanje komentara u HTML:

```
<script type="text/javascript">
<!--
document.write("Hello World!")
//-->
</script>
```

Dve kose linije na kraju linije sa komentarom (//) su u JavaScript-u simboli za komentare. Ovim sprečavamo JavaScript kompajler da kompajlira liniju -->.

3. JavaScript gde se stavlja ...

JavaScript u sekciji `body` će biti izvršen dok se strana učitava. JavaScript u sekciji `head` će se izvršiti tek kad bude pozvan.

3.1. Primeri

Primer 3.1 Skript koji sadrži funkcije ide u *Head* delu dokumenta. Ovako smo sigurni da se skript učitava pre poziva funkcije

```
<html>
<head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload event")
}
</script>
</head>
<body onload="message()">
</body>
</html>
```

Primer 3.2 Izvršenje skripta koji se nalazi u *Body* sekciji dokumenta.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write("This message is written when the page loads")
</script>
</body>
</html>
```

Primer 3.3 Pristup eksternom skriptu

```
<html>
<head>
</head>
<body>
<script src="xxx.js">
</script>
<p>
The actual script is in an external script file called "xxx.js".
</p>
</body>
</html>
```

3.2. Gde treba staviti JavaScript

JavaScript se izvršava odmah kad *browser* učitava stranu. Ovo nije baš uvek poželjno, zato što nekad želimo da izvršimo neki skript tek kad korisnik izazove događaj.

Script u sekciji *head:* Skriptovi koji treba da se izvrše kad su pozvani, odnosno kad se izazove neki događaj idu u sekciju *head*. Na ovaj način sigurni smo da je skript učitan pre nego što ga bilo ko pozove.

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
```

Script u sekciji *body*: Skriptovi koji treba da se izvrše u trenutku učitavanja strane idu u sekciju *body*. Ovakav skript generiše sadržaj strane.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
....
</script>
</body>
```

Script u obe sekcije, *body* i *head*: U dokumentu može da postiji neograničeni broj skriptova, tako da možete da ih stavite u obe sekcije.

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
<body>
<script type="text/javascript">
....
</script>
</body>
```

3.3. Korišćenje spoljašnjeg JavaScript-a

Nekada želite da isti skript koristite na više stranica, a da ne morate da ga ponovo pišete na svakoj strani. Da bi ovo olakšali, možete da napišete JavaScript u posebnom fajlu (sa .js ekstenzijom).

Primedba: Eksterni skript ne može da sadrži tag `<script>`!

Da bi koristili eksterni skript, stavite .js fajl za vrednost atributa `src` taga `<script>`:

```
<html>
<head>
<script src="xxx.js"></script>
</head>
<body>
</body>
</html>
```

Napomena: Zapamtite da stavite skript baš tamo gde bi ga normalno stavili!

4. JavaScript promenljive

U promenljivoj se čuvaju podaci.

4.1. Primeri

Primer 4.1 Promenljive

```
<html>
<body>
<script type="text/javascript">
var name = "Hege"
document.write(name)
document.write("<h1>"+name+"</h1>")
</script>
<p>This example declares a variable, assigns a value to it, and then displays the
variable.</p>
<p>Then the variable is displayed one more time, only this time as a heading.</p>
</body>
</html>
```

4.2. Promenljive

U promeljivoj se čuvaju podaci. U toku skripta može da se promeni u toku skripta. Možete da pozovete promenljivu po imenu da bi proverili vrednost, ili da bi proverili da li joj se promenila vrednost. Pravila za zadavanje imena promenljivoj:

- Imena promeljivih su *case sensitive* (mala i velika slova se razlikuju)
- Ime mora da počne slovom, ili znakom _

4.3. Deklarisanje promenljive

Promenljivu možete da kreirate pomoću naredbe `var`

```
var strname = some value
```

Promeljivu možete da kreirate i bez naredbe `var`

```
strname = some value
```

4.4. Dodeljivanje vrednosti promenljivoj

Vrednost promenljivoj možete da dodelite na ovaj način:

```
var strname = "Hege"
```

Ili na ovaj način:

```
strname = "Hege"
```

Ime promenljive mora da bude na levoj strani izraza a vrednost na desnoj strani.

4.5. Životni vek promenljivih

Ako deklarišete promenljivu unutar funkcije, vidljiva je samo unutar funkcije. Kad izađete iz funkcije, promenljiva će biti uništena. Ovo su lokalne promenljive. Možete da imate lokalne promenljive sa istim imeno u različitim funkcijama, zato što postoje samo u funkciji u kojoj su deklarisane.
Ako deklarišete promenljivu van funkcije, sve funkcije na strani mogu da joj pristupe. Životni vek ovakve promenljive počinje u trenutku kad je deklarisana, a završava se kad se strana zatvori.

5. JavaScript if...else naredbe

Uslovne naredbe u JavaScript-u se koriste da bi se izvršile različite akcije u zavisnosti od nekog uslova.

5.1. Primeri

Primer 5.1 If naredba

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time < 10)
{
document.write("<b>Good morning</b>")
}
</script>
<p>
This example demonstrates the If statement.
</p>
<p>
If the time on your browser is less than 10,
you will get a "Good morning" greeting.
</p>
</body>
</html>
```

Primer 5.2 If..else naredba

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time < 10)
{
document.write("<b>Good morning</b>")
}
else
{
document.write("<b>Good day</b>")
}
</script>
<p>
This example demonstrates the If...Else statement.
</p>
<p>
If the time on your browser is less than 10,
you will get a "Good morning" greeting.
Otherwise you will get a "Good day" greeting.
</p>
```

```
</body>
</html>
```

Primer 5.3 If...else if...else naredba

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
{
document.write("<b>Good morning</b>")
}
else if (time>=10 && time<16)
{
document.write("<b>Good day</b>")
}
else
{
document.write("<b>Hello World!</b>")
}
</script>
<p>
This example demonstrates the if..else if...else statement.
</p>
</body>
</html>
```

Primer 5.4 Slučajan link

```
<html>
<body>

<script type="text/javascript">
var r=Math.random()
if (r>0.5)
{
document.write("<a href='http://www.w3schools.com'>Learn Web Development!</a>")
}
else
{
document.write("<a href='http://www.refsnesdata.no'>Visit Refsnes Data!</a>")
}
</script>

</body>
</html>
```

5.2. Uslovne naredbe

U JavaScript-u postoje sledeće uslovne naredbe:

- **if naredba** – ova naredba se koristi ako neki kod želite da izvršite samo ako je uslov ispunjen
- **if...else naredba** – ova naredba se koristi ako neki kod želite da izvršite ako je uslov ispunjen, a u protivnom želite da izvršite neki drugi kod
- **if...else if....else naredba** – ova naredba se koristi kad želite da izvršite jedan, od više mogućih delova koda
- **switch naredba** - ova naredba se koristi kad želite da izvršite jedan, od više mogućih delova koda

5.3. If naredba

Ova naredba se koristi ako neki kod želite da izvršite samo ako je uslov ispunjen.

Sintaksa

```
if (uslov)
{
kod koji treba da se ispuni ako je uslov ispunjen
}
```

Napomena: `if` se piše malim slovima, ako napišete IF, kompajler će prijaviti grešku.

Primer 1

```
<script type="text/javascript">
//Napiši poruku "Dobro jutro" ako je manje od 10 sati
var d=new Date()
var time=d.getHours()

if (time<10)
{
document.write("<b>Dobro jutro</b>")
}
</script>
```

Primer 2

```
<script type="text/javascript">
//Napiši "Vreme za ručak!" ako je 11 sati
var d=new Date()
var time=d.getHours()

if (time==11)
{
document.write("<b>Vreme za ručak!</b>")
}
</script>
```

Napomena: Kad upoređujete promenljive, obavezno koristite dva znaka jednakost, jedan pored drugog (==)!

5.4. If...else naredba

Ova naredba se koristi ako neki kod želite da izvršite ako je uslov ispunjen, a u protivnom želite da izvršite neki drugi.

Sintaksa

```
if (uslov)
{
kod koji treba da se izvrsi ako je uslov ispunjen
}
else
{
kod koji treba da se izvrsi ako uslov nije ispunjen
}
```

Primer

```
<script type="text/javascript">
//Ako je vreme manje od 10 h
```

```
//dobicete poruku "dobro jutro".
//U protivnom, dobicete poruku "Dobar dan".
var d = new Date()
var time = d.getHours()

if (time < 10)
{
document.write("Dobro jutro!")
}
else
{
document.write("Dobar dan!")
}
</script>
```

5.5. If...else if...else naredba

Ova naredba se koristi kad želite da izvršite jedan, od više mogućih delova koda.

Sintaksa

```
if (uslov)
{
kod koji treba da se izvrsi ako je uslov ispunjen
}
else if (uslov2)
{
kod koji treba da se izvrsi ako je uslov2 ispunjen
}
else
{
kod koji treba da se izvrsi ako ni jedan od uslova nije ispunjen
}
```

Primer

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
{
document.write("<b>Dobro jutro</b>")
}
else if (time>10 && time<16)
{
document.write("<b>Dobar dan</b>")
}
else
{
document.write("<b>Zdravo svima!</b>")
}
</script>
```

5.6. Switch naredba

Ova naredba se koristi kad želite da izvršite jedan, od više mogućih delova koda.

Sintaksa

```
switch(n)
{
case 1:
```

```
  izvrsi blok koda 1
  break
case 2:
  izvrsi blok koda 2
  break
default:
  kod koji treba da se izvrsi ako n nema vrednost 1 ni 2
}
```

Naredba ovako funkcioniše: Prvo imamo jednostavan izraz `n` (veoma često promenljivu), koja se izračunava jednom. Vrednost izraza se zatim upoređuje sa vrednostima za svako case u strukturi. Ako postoji poklapanje, izvršava se odgovarajući blok koda. Koristite naredbu break da bi sprečili naredbu da automatski pokrene sledeći case.

Primer 5.5 Switch naredba

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
theDay=d.getDay()
switch (theDay)
{
case 5:
  document.write("<b>Konacno petak</b>")
  break
case 6:
  document.write("<b>Super subota</b>")
  break
case 0:
  document.write("<b>Pospana nedelja</b>")
  break
default:
  document.write("<b>Zaista jedva cekam vikend!</b>")
}
</script>
<p>Ovaj JavaScript generise razlicitu pozdravnu poruku na osnovu dana. Primetite: Nedelja=0, Ponedeljak=1, Utorak=2, itd.</p>
</body>
</html>
```

6. JavaScript operatori

6.1. Aritmetički operatori

Operator	Opis	Primer	Rezultat
+	Sabiranje	x=2 y=2 x+y	4
-	Oduzimanje	x=5 y=2 x-y	3
*	Množenje	x=5 y=4 x*y	20
/	Deljenje	15/5 5/2	3 2.5

%	Moduo (ostatak pri deljenju)	5%2 10%8 10%2	1 2 0
++	Inkrement	x=5 x++	x=6
--	Dekrement	x=5 x--	x=4

6.2. Operatori dodele

Operator	Primer	Isto kao
=	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

6.3. Operatori poređenja

Operator	Opis	Primer
==	je jednako	5==8 vraca false
===	je jednako (poredi vrednost i tip)	x=5 y="5" x==y vraća true x===y vraća false
!=	nije jednako	5!=8 vraća true
>	veće od	5>8 vraća false
<	manje od	5<8 vraća true
>=	veće ili jednako	5>=8 returns false
<=	manje ili jednako	5<=8 returns true

6.4. Logički operatori

Operator	Opis	Primer
&&	and (i)	x=6 y=3 (x < 10 && y > 1) vraća true
||	or (ili)	x=6 y=3 (x==5 || y==5) vraća false
!	not (ne)	x=6 y=3 !(x==y) vraća true

6.5. String operatori

String je najčešće tekst, na primer "*Hello World*!". Da bi spojili dve ili više string promenljivih, koristimo + operator.

```
txt1="What a very"
txt2="nice day!"
txt3=txt1+txt2
```

Promenljiva `txt3` sada sadrži tekst "*What a verynice day!"*.
Da bi dodali razmak između string promenljivih, ubacite razmak u izraz, ili u jednu od promenljivih.

```
txt1="What a very"
txt2="nice day!"
txt3=txt1+" "+txt2
or
txt1="What a very "
txt2="nice day!"
txt3=txt1+txt2
```

Promenljiva `txt3` sada sadrži *"What a very nice day!"*.

6.6. Uslovni operatori

JavaScript takođe sadrži i uslovne operatore koji dodeljuju vrednost promenljivoj na osnovu nekog uslova.

Sintaksa

```
imepromenljive=(uslov)?vrednost:vrednost2
```

Primer

```
pozdrav=(posetilac=="PREDS")?"Dragi predsednice ":"Dragi "
```

Ako promenljiva `posetilac` ima vrednost `PREDS`, u promenljivu pozdrav stavljamo tekst "Dragi predsednice ". U suprotnom u promenljivu stavljamo tekst "Dragi ".

7. JavaScript poruke (popup box)

U JavaScript-u možemo da kreiramo različite vrste poruka: poruku upozorenja (*Alert box*), poruka za potvrdu (*Confirm box*) i poruke korisniku (*Prompt box*).

7.1. Primeri

Primer 7.1 Okvir sa upozorenjem (*Alert box)*

```
<html>
<head>
<script type="text/javascript">
function disp_alert()
{
alert("Ja sam alert box!!")
}
</script>
</head>
<body>
<form>
<input type="button" onclick="disp_alert()" value="Display alert box">
</form>
</body>
</html>
```

Primer 7.2 Okvir sa upozorenjem sa novim redom

```
<html>
<head>
<script type="text/javascript">
function disp_alert()
{
alert("Zdravo ponovo! Ovako mi" + '\n' + "dodajemo novi red u alert box!")
}
</script>
</head>
<body>
<form>
<input type="button" onclick="disp_alert()" value="Display alert box">
</form>
</body>
</html>
```

Primer 7.3 Okvir sa potvrdom (*Confirm box)*

```
<html>
<head>
<script type="text/javascript">
function disp_confirm()
{
var name=confirm("Pritisni dugme")
if (name==true)
{
document.write("Pritisnuli ste OK dugme!")
}
else
{
document.write("Pritisnuli ste Cancel dugme!")
}
}
</script>
</head>
<body>
<form>
<input type="button" onclick="disp_confirm()" value="Display a confirm box">
</form>
</body>
</html>
```

Primer 7.4 Okvir sa porukom (*Prompt box)*

```
<html>
<head>
<script type="text/javascript">
function disp_prompt()
{
var name=prompt("Molim vas, unesite ime","")
if (name!=null && name!="")
{
document.write("Zdravo " + name + "! Kako ste danas?")
}
}
</script>
</head>
<body>
<form>
<input type="button" onclick="disp_prompt()" value="Display a prompt box">
</form>
</body>
</html>
```

7.2. Alert Box

Alert box se koristi kad želite da budete sigurni da je informacija stigla do korisnika. Kada se pojavi *alert box,* korisnik treba da pritisne dugme "*OK*" da bi nastavio.
Sintaksa:

```
alert("neki tekst")
```

7.3. Confirm Box

Confirm box se koristi kad želite da korisnik potvrdi ili prihvati nešto. Kad se pojavi *confirm box*, korisnik mora da pritisne dugme "*OK*" ili "*Cancel*" da bi nastavio. Ako je pritisnuo dugme "*OK*", povratna vrednost je `true`. Ako je pritisnuo dugme "*Cancel*", povratna vrednost je `false`.
Sintaksa:

```
confirm("neki tekst")
```

7.4. Prompt Box

Prompt box se koristi kad želite da korisnik unese neku vrednost pre nego što pristupi strani. Kad se pojavi *prompt box,* korinik mora da pritisne dugme "*OK*" ili "*Cancel*", pošto je uneo vrednost. Ako je pritisnuo dugme "*OK*", povratna vrednost je `true`. Ako je pritisnuo dugme "*Cancel*", povratna vrednost je `false`.
Sintaksa:

```
prompt("neki tekst","početna vrednost")
```

8. JavaScript funkcije

Funkcija je blok koda koji može da se koristi više puta, izvršava se pri nastupanju nekog događaja ili kad se pozove.

8.1. Primeri

Primer 8.1 Poziv funkcije

```
<html>
<head>
<script type="text/javascript">
function myfunction()
{
alert("HELLO")
}
</script>

</head>
<body>
<form>
<input type="button"
onclick="myfunction()"
value="Call function">
</form>
<p>By pressing the button, a function will be called. The function will alert a
message.</p>
</body>
</html>
```

Primer 8.2 Funkcija sa argumentom

```
<html>
<head>
<script type="text/javascript">
function myfunction(txt)
{
alert(txt)
}
</script>

</head>
<body>
<form>
<input type="button"
onclick="myfunction('Hello')"
value="Call function">
</form>
<p>By pressing the button, a function with an argument will be called. The function will
alert
this argument.</p>
</body>
</html>
```

Primer 8.3 Funkcija sa argumentom 2

```
<html>
<head>
<script type="text/javascript">
function myfunction(txt)
{
alert(txt)
}
</script>
</head>
<body>
<form>
<input type="button"
onclick="myfunction('Good Morning!')"
value="In the Morning">
<input type="button"
onclick="myfunction('Good Evening!')"
value="In the Evening">
</form>
<p>
When you click on one of the buttons, a function will be called. The function will alert
the argument that is passed to it.
</p>
</body>
</html>
```

Primer 8.4 Funkcija koja ima povratnu vrednost

```
<html>
<head>
<script type="text/javascript">
function myFunction()
{
return ("Hello, have a nice day!")
}
</script>

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

```
document.write(myFunction())
</script>
<p>The script in the body section calls a function.</p>
<p>The function returns a text.</p>
</body>
</html>
```

Primer 8.5 Funkcija sa argumentima koja ima povretnu vrednost

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3))
</script>
<p>The script in the body section calls a function with two parameters (4 and 3).</p>
<p>The function will return the product of these two parameters.</p>
</body>
</html>
```

8.2. JavaScript funkcije

Da bi sprečili *browser* da neki skript izvrši kad se učita strana, skript možete da napišete kao funkciju. Funkcija sadrži skript koji će se izvršiti samo pri pozivu ili kad se desi neki događaj. Funkciju možete da pozovete sa bilo kog mesta na strani (ili čak i sa druge strane ako je funkcija deo eksternog.js fajla).
Funkcije se definišu na početku strane, u sekciji `<head>.`

Primer

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!")
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me!"
onclick="displaymessage()" >
</form>
</body>
</html>
```

Da linija: `alert("Hello world!!"),` u gornjem primeru nije bila napisana u funkciji, izvršila bi se kad se stranica učita. Ovako, skript neće biti izvršen pre nego što korisnik pritisne dugme. Dodali smo događaj `onClick` dugmetu koji će izvršiti funkciju function `displaymessage()` kad se pritisne dugme. Više o JavaScript događajima ćete naučiti kasnije u poglavlju 13.

8.3. Kako se definiše funkcija?

Sintaksa za kreiranje funkcije je

```
function imefunkcije(var1,var2,...,varX)
{
neki kod
}
```

`var1, var2`, itd. su promenljive ili vrednosti koje se prenose funkciji. `{` i `}` definišu početak i kraj funkcije.

Napomena: Funkcija bez argumenata mora da uključi zagrade () posle imena funkcije.

```
function imefunkcije()
{
neki kod
}
```

Napomena: Ne zaboravite na važnost velikih slova u JavaScript-u! Reč function mora da bude napisana malim slovima, inače se prijavljuje greška! Takođe, funkciju morate da zovete sa istim slovima kao što stoji u definiciji funkcije.

8.4. Naredba return

Naredba `return` se koristi da bi se specificirala vrednost koju funkcija. Dakle, ako funkcija ima povratnu vrednost, mora da sadrži `return` naredbu.

Primer

Sledeća funkcija vraća proizvod dva broja(a i b):

```
function prod(a,b)
{
x=a*b
return x
}
```

Kad pozovete ovu funkciju, morate da joj prosledite dva parametra:

```
product=prod(2,3)
```

Povratna vrednost prethodnog poziva je 6, i sačuvaće se u promenljivoj `product`.

9. JavaScript for petlja

Petlje se u JavaScript-u koriste da bi se jedan blok naredbi izvršio specificiran broj puta, ili dok je neki uslov ispunjen.

9.1. Primeri

Primer 9.1 *For* petlja

```
<html>
<body>
<script type="text/javascript">
for (i = 0; i <= 5; i++)
{
document.write("The number is " + i)
document.write("<br>")
}
</script>
<p>Explanation:</p>
<p>This for loop starts with i=0.</p>
<p>As long as <b>i</b> is less than, or equal to 5, the loop will continue to run.</p>
```

```
<p><b>i</b> will increase by 1 each time the loop runs.</p>
</body>
</html>
```

Primer 9.2 Petlja koja prolazi kroz HTML naslove

```
<html>
<body>
<script type="text/javascript">
for (i = 1; i <= 6; i++)
{
document.write("<h" + i + ">This is header " + i)
document.write("</h" + i + ">")
}
</script>
</body>
</html>
```

9.2. JavaScript petlje

Više puta, kad pišete kod, želite da je jedan deo koda izvrši više puta za redom. Umesto da sličan kod napišete više puta, za ovo možete da koristite petlje. U JavaScript-u postoje dve vrste petlji:

- ***for*** – petlje koje prolaze kroz deo koda specificiran broj puta
- ***while*** – petlje koje prolaze kroz deo koda sve dok je neki uslov ispunjen

9.3. For petlja

For petlja se koristi kad u napred znate koliko puta neki skript treba da se izvrši.
Sintaksa

```
for (prom=pocetnavrednost;prom<=krajnjavrednost;prom=prom+increment)
{
   kod koji treba da se izvrsi
}
```

Primer 9.3

Objašnjenje: Naredni primer definiše petlju koja počinje sa `i=0`. Petlja se izvršava dok je `i` manje ili jednako 10. `i` se povećava za 1 svaki put kad se petlja izvrši.
Napomena: Parametar increment takođe može da bude i negativan, a <= može da bude bilo koja naredba poređenja.

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

Rezultat:

```
Broj je 0
Broj je 1
Broj je 2
Broj je 3
Broj je 4
```

```
Broj je 5
Broj je 6
Broj je 7
Broj je 8
Broj je 9
Broj je 10
```

10. JavaScript while petlja

10.1. Primeri

Primer 10.1 While petlja

```
<html>
<body>
<script type="text/javascript">
i = 0
while (i <= 5)
{
document.write("Broj je " + i)
document.write("<br>")
i++
}
</script>

<p>Objasnjenje:</p>
<p><b>i</b>  jednako 0.</p>
<p>Dok je <b>i</b> manje , ili jednako, 5, petlja se izvršava.</p>
<p><b>i</b> se povećava za 1 svaki put kad se izvrši petlja.</p>
</body>
</html>
```

Primer 10.2 Do...while petlja

```
<html>
<body>
<script type="text/javascript">
i = 0
do
{
document.write("Broj je " + i)
document.write("<br>")
i++
}
while (i <= 5)
</script>
<p>Objašnjenje:</p>
<p><b>i</b>  jednako 0.</p>
<p>Petlja se izvršava</p>
<p><b>i</b> se povećava za 1 svaki put kad se petlja izvrši.</p>
<p>Dok je <b>i</b> manje ili jednako 5, petljase izvrsava.</p>
</body>
</html>
```

10.2. While petlja

While petlja se koristi za kod koji treba da se izvršava sve dok je neki uslov ispunjen.

```
while (prom<=krajnjaVrednost)
{
```

```
    kod koji treba da se izvrši
}
```

Primedba: <= može da bude bilo koja naredba poređenja.

Primer 10.3

Objašnjenje: Primer definiše petlju u kojoj `i` ima početnu vrednost `i=0`. Petlja nastavlja da se izvršava dok je `i` manje ili jednako 10. U svakom ciklusu izvršenja `i` se povećava za 1.

```
<html>
<body>
<script type="text/javascript">
var i=0
while (i<=10)
{
document.write("Broj je " + i)
document.write("<br />")
i=i+1
}
</script>
</body>
</html>
```

Rezultat

```
Broj je 0
Broj je 1
Broj je 2
Broj je 3
Broj je 4
Broj je 5
Broj je 6
Broj je 7
Broj je 8
Broj je 9
Broj je 10
```

10.3. Do...while petlja

Do...while petlja je varijanta *while* petlje. Ova petlja uvek izvršava kod bar jednom, i ponavlja se sve dok je uslov ispunjen. Zbog toga što se prvo izvrši kod, a zatim ispita uslov, petlja će se izvršiti bar jednom, čak i ako uslov nije ispunjen.

```
do
{
    kod koji treba da se izvrši
}
while (prom<=krajnjaVrednost)
```

Primer 10.4

```
<html>
<body>
<script type="text/javascript">
var i=0
do
{
document.write("Broj je " + i)
document.write("<br />")
i=i+1
}
while (i<0)
</script>
</body>
</html>
```

Rezultat

```
Broj je 0
```

11. Naredbe break i continue

Dve specijalne naredbe mogu da se koriste unutar petlji: `break` i `continue`.

Break

Naredba `break` prekida izvršenje petlje i prelazi na izvršenje koda koji se nalazi nakon petlje (ako postoji).

Continue

Naredba `continue` prekida izvršenje trenutne iteracije petlje i nastavlja sa izvršenjem sledeće iteracije.

11.1. Primeri

Primer 11.1 Break

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
{
if (i==3){break}
document.write("Broj je " + i)
document.write("<br />")
}
</script>
<p>Objašnjenje: Za i=3 se izlazi iz petlje.</p>
</body>
</html>
```

Rezultat

```
Broj je 0
Broj je 1
Broj je 2
```

Primer 11.2 Continue

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
{
if (i==3){continue}
document.write("Broj je " + i)
document.write("<br />")
}
</script>
</table>
<p>Objasnjenje: Petlja izlazi iz trenutne iteracije i nastavlja sa izvršenjem naredne
iteracije za i=3.</p>
</body>
</html>
```

Rezultat

```
Broj je 0
```

```
Broj je 1
Broj je 2
Broj je 4
Broj je 5
Broj je 6
Broj je 7
Broj je 8
Broj je 9
Broj je 10
```

12. Naredba for...in

Naredba *for...in* se koristi da bi se prošlo kroz elemente niza ili kroz atribute objekta. Kod u telu naredbe se izvršava po jednom za svaki element/atribut.
Sintaksa

```
for (promenljiva u objektu)
{
    kod koji treba da se izvrši
}
```

Primer 12.1 For...in za prolaz kroz elemente niza

```
<html>
<body>
<script type="text/javascript">
var x
var mycars = new Array()
mycars[0] = "Saab"
mycars[1] = "Volvo"
mycars[2] = "BMW"
for (x in mycars)
{
document.write(mycars[x] + "<br />")
}
</script>
</body>
</html>
```

13. JavaScript događaji

Događaji su akcije koje JavaScript može da detektuje.

13.1. Events (događaji)

Koristeći JavaScript, možemo da kreiramo dinamičke *web* strane.
Svaki element na *web* strani ima izvesne događaje koji mogu da aktiviraju JavaScript funkcije. Na primer, možemo da koristimo događaj `onClick` za element dugme da bi odredili da će funkcija da se pokrene tek kad korisnik pritisne dugme. Događaje definišemo u HTML tagovima.
Primeri događaja:

- Klik mišem
- Učitavanje *web* strane ili slike
- Prelazak mišem preko određene tačke na slici
- Izbor ulaza u formi
- Slanje (*Submitting*) HTML forme
- Pritisak tastera

Napomena**:** Događaji se obično koriste u kombinaciji sa funkcijama, i funkcije se ne izvršavaju pre nego što se desi događaj.

13.2. onload i onUnload

Događaji `onload` and `onUnload` se dešavaju kad korisnik pristupa strani ili je napušta.
Događaj `onload` se često koristi za proveravanje tipa i verzije *browsera* korisnika, da bi se u zavisnosti od toga učitala odgovarajuća verzija strane.
Oba događaja `onload` i `onUnload` se takođe često koriste za rad sa *cookie*-jima koji treba da se postave kad korisnik pristupa strani, ili je napušta. Na primer, kad neko prvi put pristupa vašoj strani, možete da imate dijalog u koji treba da se unese ime korisnika. Zatim se ime čuva u *cookie*-ju. Sledeći put kad korisnik pristupi vašoj strani, može da se pojavi drugi dijalog, npr. sa tekstom: "*Welcome John Doe*!".

13.3. onFocus, onBlur i onChange

Događaji `onFocus`, `onBlur` i `onChange` se često koriste u kombinaciji sa validacijom polja forme.
Sledi primer za korišćenje događaja `onChange`. Funkcija `checkEmail()` će biti pozvana svaki put kad se promeni sadržaj polja:

```
<input type="text" size="30"
id="email" onchange="checkEmail()">;
```

13.4. onSubmit

Događaj `onSubmit` se koristi za validaciju SVIH polja forme, pre njihovog slanja.
Sledi primer korišćenja događaja `onSubmit`. Funkcija `checkForm()` se poziva kad korisnik pritisne dugme *submit* u formi. Ako vrednosti polja nisu prihvaćene, predaja će biti obustavljena. Funkcija `checkForm()` vraća `true` ili `false`. Ako vrati `true` forma forma će biti predata, ako ne, slanje forme ce biti obustavljeno:

```
<form method="post" action="xxx.htm"
onsubmit="return checkForm()">
```

13.5. onMouseOver i onMouseOut

Događaji `onMouseOver` i `onMouseOut` se često koriste za kreiranje animiranih dugmadi.
Sledi primer korišćenja događaja `onMouseOver`. Poruka upozorenja se pojavljuje kad se detektuje prelazak miša (događaj `onMouseOver`):

```
<a href="http://www.w3schools.com"
onmouseover="alert('An onMouseOver event');return false">
<img src="w3schools.gif" width="100" height="30">
</a>
```

14. JavaScript try...catch naredba

Naredba *try...catch* dozvoljava da testiramo blok koda na postojanje grešaka.

14.1. JavaScript – hvatanje grešaka

Svima nam se već desilo, dok otvaramo strane na internetu da se pojavi poruka da postoji *runtime error* i pitanjem: "*Do you wish to debug*?". Poruke o greškama poput ove mogu da budu korisne za onog koji piše stranu, ali ne i za korisnika. Kad korinik vidi da postoji greška, često napušta stranu. Ovo poglavlje vas uči

kako da hvatate greške i kako da ih obrađujete, tako da ne izgubite posetioce. Postoje dva načina na koje možete da uhvatite grešku na strani:

- Korišćenjem naredbe ***try...catch*** (funkcioniše kod *IE5+, Mozilla 1.0, iNetscape 6*)
- Korišcenjem događaja ***onerror***. Ovo je staro rešenje za hvatanje grešaka (dostupno još od *Netscape 3*)

14.2. Try...Catch naredba

Naredba *try...catch* dozvoljava da testirate blok koda na greške. Blok *try* sadrži blok koda koji treba da se izvrši, a *catch* blok koda koji treba da se izvrši ako se desi greška.

Sintaksa

```
try
{
//izvršenje nekog koda
}
catch(greška)
{
//obrada greške
}
```

Primer 14.1

Sledeći primer sadrži skript koji treba da prikaže "Dobrodošli!" kad pritisnete dugme. Ali, postoji greška u kucanju u funkciji `message()`. `alert()` je pogrešno napisano kao `adddlert()`. Javlja se JavaScript greška:

```
<html>
<head>
<script type="text/javascript">
function message()
{
adddlert("Dobrodošli!")
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()" />
</body>

</html>
```

Da bi preduzeli odgovarajuću akciju kad se desi greška, možete da dodate a *try...catch.*
Sledeći primer je napisan tako da koristi ovu naredbu. Pošto je `alert()` napisano pogrešno javlja se greška. Ipak, ovog puta blok `catch` hvata grešku i izvršava kod koji obrađuje ovu grešku (prikazuje drugu poruku kojom obaveštava korisnika o tome šta se desilo:

```
<html>
<head>
<script type="text/javascript">
var txt=""
function message()
{
try
  {
  adddlert("Dobrodošli!")
  }
catch(err)
  {
```

```
  txt="Na ovoj strani se desila greška.\n\n"
  txt+="Opis greške: " + err.description + "\n\n"
  txt+="Kliknite OK da bi nastavili.\n\n"
  alert(txt)
  }
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()" />
</body>

</html>
```

Primer 14.2

Sledeći primer koristi dijalog sa potvrdom da obavesti korisnika o tome da je nastupila greška i da može da pritisnu dugme *OK* da bi nastavio da gleda stranu, ili dugme *Cancel* da bi se vratio na matičnu stranu (vrši se redirekcija korisnika).

```
<html>
<head>
<script type="text/javascript">
var txt=""
function message()
{
try
  {
  adddlert("Dobrodosli!")
  }
catch(err)
  {
  txt="Na ovoj strani se desila greska.\n\n"
  txt+="Kliknite OK da bi nastavili da posmatrate stranu,\n"
  txt+="ili Cancel da bi se vratili na maticnu stranu.\n\n"
  if(!confirm(txt))
    {
    document.location.href="http://www.w3schools.com/"
    }
  }
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

14.3. JavaScript throw naredba

Naredba `throw` vam dozvoljava da kreirate izuzetak. Ako ovu naredbu koristite zajedno sa naredbom `try...catch`, možete da kontrolišete tok programa i da generišete precizne poruke o greškama.

Sintaksa

```
throw(izuzetak)
```

Izuzetak može da bude string, karakter, objekat ili Boolean.

Primer 14.3 Throw

```
<html>
<body>
```

```
<script type="text/javascript">
var x=prompt("Unesite broj izmedju 0 i 10:","")
try
{
if(x>10)
  throw "Err1"
else if(x<0)
  throw "Err2"
else if(isNaN(x))
  throw "Err3"
}
catch(er)
{
if(er=="Err1")
  alert("Greska! Broj je prevelik")
if(er == "Err2")
  alert("Greska! Broj je premali")
if(er == "Err3")
  alert("Greska! Znak koji ste uneli nije broj")
}
</script>
</body>
</html>
```

14.4. Događaj onerror

Upravo smo objasnili kako da koristimo *try...catch* naredbu da bi uhvatili grešku na strani. Sada ćemo da objasnimo kako da za iste potrebe koristimo događaj `onerror`. Događaj `onerror` se dešava uvek kad se desi greška u skriptu na strani.
Da bi koristili događaj `onerror`, morate da kreirate funkciju koja će da obradi taj događaj. Funkcija za obradu događaja se poziva sa tri argumenta: `msg` (poruka o grešći), `url` (url strane koja je izazvala grešku) i `line` (linija koda u kojoj je nastupila greška).

Sintaksa

```
onerror=handleErr
function handleErr(msg,url,l)
{
//Ovde obrađujemo grešku
return true or false
}
```

Vrednost koju vraća `onerror` određuje da li će *browser* da prikaže standardnu *error* poruku. Ako vratite `false`, *browser* prikazuje standardnu *error* poruku JavaScript consoli. Ako vratite `true`, *browser* ne prikazuje standardnu *error* poruku.

Primer 14.4

Sledeći primer pokazuje kako da uhvatite grešku pomoću:

```
<html>
<head>
<script type="text/javascript">
onerror=handleErr
var txt=""
function handleErr(msg,url,l)
{
txt="Na ovoj strani postoji greška.\n\n"
txt+="Greška: " + msg + "\n"
txt+="URL: " + url + "\n"
txt+="Linija: " + l + "\n\n"
txt+="Pritisnite OK da bi nastavili.\n\n"
```

```
alert(txt)
return true
}
function message()
{
adddlert("Dobrodosli!")
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

15. JavaScript specijalni karakteri

U JavaScript možete da u tekst dodate specijalne karaktere korišćenjem znaka \.

15.1. Ubacivanje specijalnih karaktera

Znak (\) se koristi da bi u tekstualni string ubacili specijalne znake, kao što su apostrofi, novi red, navodnici,...
Obratite pažnju na sledeći kod:

```
var txt="We are the so-called "Vikings" from the north."
document.write(txt)
```

U JavaScript-u string počinje i završava se navodnicima (’ ili „). To znači da će prethodni tekst da bude odsečen na: *We are the so-called*
Da bi rešili ovaj prblem, moramo da stavimo znak (\) pre oba navodnika u reči "Viking". Ovo oba navodnika pretvara u deo stringa.:

```
var txt="We are the so-called \"Vikings\" from the north."
document.write(txt)
```

JavasScript će sada da prikaže pravilan tekst: *We are the so-called "Vikings" from the north.*
Evo još jednog primer:

```
document.write ("You \& me are singing!")
```

Prethodni primer daje sledeći izlaz:

```
You & me are singing!
```

U sledećoj tabeli su dati specijalni znaci za koje morate da koristite \ prilikom dodavanja u string:

Code	**Izlaz**
\'	apostrof
\"	navodnik
\&	*ampersand*
\\	*backslash*
\n	novi red
\r	*carriage return*
\t	*tab*
\b	*backspace*
\f	*form feed*

16. JavaScript saveti

Još neke važne stvari koje treba da znate o JavaScript-u.

16.1. JavaScript je *case sensitive*

Funkcija `myfunction` nije isto što i funkcija `myFunction` i promenljiva `myVar` nije isto što `myvar`. JavaScript je *case sensitive* (pravi razliku između velikih i malih slova).

16.2. Razmaci

JavaScript ignoriše dodatne razmake. Slobodno dodajte razmake da bi dobili pregledniji kod. Sledeće linije su ekvivalentne:

```
name="Hege"
name = "Hege"
```

16.3. Novi red u liniji koda

Tekst u stringu možete da prenesete u novi red pomoću znaka \. Sledeći primer će biti pravilno prikazan.

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

Ipak, ne možete da prenesete kod u novi red na sledeći način:

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

16.4. Komentari

Komentare možete da unesete u skript pomoću //:

```
//ovo je komentar
document.write("Hello World!")
```

Ili korišćenjem /* i */ (kad imate komentar u više linija):

```
/* Ovo je duži komentar.
Sastoji se iz
vise linija */
document.write("Hello World!")
```

17. Uvod u JavaScript objekte

JavaScript je objektno orijentisani programski jezik (OOP).
OOP jezik dozvoljava da definišete objekte i sopstvene tipove podataka.
Ipak, kreiranje sopstvenih objekata će biti objašnjeno kasnije u Poglavlju 24. Za sada ćemo da se bavimo postojećim JavaScript objektima.
Napomena: objekat je specijalna vrsta podataka. Sadrži atribute i metode.

Atributi (properties)

Atributi su vrednosti vezane za objekat. U sledećem primeru koristimo atribut `length` objekta `String` da bi vratili dužinu stringa:

```
<script type="text/javascript">
var txt="Hello World!"
document.write(txt.length)
</script>
```

Rezultat ovog koda bi bio:

```
12
```

Metodi

Metodi su akcije koje mogu da se preduzmu nad objektima. U sledećem primeru koristimo metod `toUpperCase()` objekta `String` da bi prikazali string velikim slovima:

```
<script type="text/javascript">
var str="Hello world!"
document.write(str.toUpperCase())
</script>
```

Izlaz ovog koda bi bio:

```
HELLO WORLD!
```

17.1. JavaScript String objekat

Objekat tipa `String` se koristi za rad sa tekstualnim podacima.

Primer 17.1 Vraćanje dužine stringa

```
<html>
<body>
<script type="text/javascript">
var txt="Hello World!"
document.write(txt.length)
</script>
</body>
</html>
```

Primer 17.2 Stil stringa

```
<html>
<body>
<script type="text/javascript">
var 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() + " (does not work in IE)</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>Lowercase: " + txt.toLowerCase() + "</p>")
document.write("<p>Uppercase: " + txt.toUpperCase() + "</p>")
document.write("<p>Subscript: " + txt.sub() + "</p>")
document.write("<p>Superscript: " + txt.sup() + "</p>")
document.write("<p>Link: " + txt.link("http://www.w3schools.com") + "</p>")
</script>
</body>
</html>
```

Primer 17.3 Metod indexOf()

```
<html>
<body>
<script type="text/javascript">
var str="Hello world!"
document.write(str.indexOf("Hello") + "<br />")
document.write(str.indexOf("World") + "<br />")
document.write(str.indexOf("world"))
</script>
</body>
</html>
```

Primer 17.4 Metod match()

```
<html>
<body>
<script type="text/javascript">
var str="Hello world!"
document.write(str.match("world") + "<br />")
document.write(str.match("World") + "<br />")
document.write(str.match("worlld") + "<br />")
document.write(str.match("world!"))
</script>
</body>
</html>
```

Primer 17.5 Zamena karaktera u stringu

```
<html>
<body>
<script type="text/javascript">
var str="Visit Microsoft!"
document.write(str.replace(/Microsoft/,"W3Schools"))
</script>
</body>
</html>
```

Primeri upotrebe:

Sleećі primer koristi metod `toUpperCase()` da bi konvertovao string u string napisan velikim slovima.

```
var txt="Hello world!"
document.write(txt.toUpperCase())
```

Prethodni kod daje sledeći izlaz:

```
HELLO WORLD!
```

17.2. Objekat Date

Objekat tipa `Date` se koristi za rad sa datumom i vremenom.

Primer 17.6 Vraćanje trenutnog datuma i vremena

```
<html>
<body>
<script type="text/javascript">
document.write(Date())
</script>
</body>
</html>
```

Primer 17.7 getTime()

```
<html>
<body>
<script type="text/javascript">
var minutes = 1000*60
var hours = minutes*60
var days = hours*24
var years = days*365
var d = new Date()
var t = d.getTime()
var y = t/years
document.write("It's been: " + y + " years since 1970/01/01!")
</script>
</body>
</html>
```

Primer 17.8 setFullYear()

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
d.setFullYear(1992,10,3)
document.write(d)
</script>
</body>
</html>
```

Primer 17.9 toUTCString()

```
<html>
<body>
<script type="text/javascript">
var d = new Date()
document.write (d.toUTCString())
</script>
</body>
</html>
```

Primer 17.10 getDay()

```
<html>
<body>
<script type="text/javascript">
var d=new Date()
var weekday=new Array(7)
weekday[0]="Sunday"
weekday[1]="Monday"
weekday[2]="Tuesday"
weekday[3]="Wednesday"
weekday[4]="Thursday"
weekday[5]="Friday"
weekday[6]="Saturday"
document.write("Today it is " + weekday[d.getDay()])
</script>
</body>
</html>
```

Primer 17.11 Prikazivanje sata

```
<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date()
var h=today.getHours()
var m=today.getMinutes()
var s=today.getSeconds()
// add a zero in front of numbers<10
m=checkTime(m)
s=checkTime(s)
document.getElementById('txt').innerHTML=h+":"+m+":"+s
t=setTimeout('startTime()',500)
}
function checkTime(i)
{
if (i<10)
  {i="0" + i}
  return i
}
```

```
</script>
</head>
<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

Objekat tipa `Date` definišemo pomoću ključne reči `new`. Sledeća linija definiše objekat `Date` koji se zove `myDate`:

```
var myDate=new Date()
```

Napomena: `Date` objekat automatski sadrži u svojoj inicijalnoj vrednosti trenutno vreme i datum.

Datumom jednostavno manipulišemo koristeći metode dostupne za objekat `Date`. U sledećem primeru postavljamo objekat `Date` na određeni datum (14. Januar 2010):

```
var myDate=new Date()
myDate.setFullYear(2010,0,14)
```

A u sledećem primeru postavljamo datum na dan za 5 dana u budućnosti:

```
var myDate=new Date()
myDate.setDate(myDate.getDate()+5)
```

Primedba: Ako prethodno sabiranje promeni mesec ili godinu, objekat `Date` to sam prepozna i obradi.

Objekat tipa `Date` takođe može i da uporedi dva datuma. Sledeći primer upoređuje današnji datum sa datumom 14. Januar 2010:

```
var myDate=new Date()
myDate.setFullYear(2010,0,14)
var today = new Date()
if (myDate>today)
  alert("Danas je pre 14. Januara 2010")
else
  alert("Today is 14. Januara 2010")
```

17.3. Objekat tipa Array

Objekat tipa `Array` se koristi za čuvanje skupa vrednosti u jednj promenljivoj (polju).

Primer 17.12

```
<html>
<body>
<script type="text/javascript">
var mycars = new Array()
mycars[0] = "Saab"
mycars[1] = "Volvo"
mycars[2] = "BMW"
for (i=0;i<mycars.length;i++)
{
document.write(mycars[i] + "<br />")
}
</script>
</body>
</html>
```

Primer 17.13 For...In

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

```
var x
var mycars = new Array()
mycars[0] = "Saab"
mycars[1] = "Volvo"
mycars[2] = "BMW"
for (x in mycars)
{
document.write(mycars[x] + "<br />")
}
</script>
</body>
</html>
```

Primer 17.14 Spajanje dva polja – concat()

```
<html>
<body>
<script type="text/javascript">
var arr = new Array(3)
arr[0] = "Jani"
arr[1] = "Tove"
arr[2] = "Hege"
var arr2 = new Array(3)
arr2[0] = "John"
arr2[1] = "Andy"
arr2[2] = "Wendy"
document.write(arr.concat(arr2))
</script>
</body>
</html>
```

Primer 17.15 Stavljanje elemenata polja u string – join()

```
<html>
<body>
<script type="text/javascript">
var arr = new Array(3)
arr[0] = "Jani"
arr[1] = "Hege"
arr[2] = "Stale"
document.write(arr.join() + "<br />")
document.write(arr.join("."))
</script>
</body>
</html>
```

Primer 17.16 Polja literala sort()

```
<html>
<body>
<script type="text/javascript">
var arr = new Array(6)
arr[0] = "Jani"
arr[1] = "Hege"
arr[2] = "Stale"
arr[3] = "Kai Jim"
arr[4] = "Borge"
arr[5] = "Tove"
document.write(arr + "<br />")
document.write(arr.sort())
</script>
</body>
</html>
```

Primer 17.17 Numerička polja sort()

```
<html>
<body>
<script type="text/javascript">
function sortNumber(a, b)
{
return a - b
}
var arr = new Array(6)
arr[0] = "10"
arr[1] = "5"
arr[2] = "40"
arr[3] = "25"
arr[4] = "1000"
arr[5] = ""
document.write(arr + "<br />")
document.write(arr.sort(sortNumber))
</script>
</body>
</html>
```

Objekat tipa `Array` definišemo pomoću ključne reči `new`. Sledeća linija koda definiše objekat tipa `Array` koji se zove `myArray`:

```
var myArray=new Array()
```

Postoje dva načina dodeljivanja vrednosti polju (možete da dodate koliko hoćete vrednosti da bi definisali onoliko promenljivih koliko vam je potrebno).

1:

```
var mycars=new Array()
mycars[0]="Saab"
mycars[1]="Volvo"
mycars[2]="BMW"
```

Takođe možete da prosledite integer argument kad kreirate polje da bi definisali njegovu veličinu:

```
var mycars=new Array(3)
mycars[0]="Saab"
mycars[1]="Volvo"
mycars[2]="BMW"
```

2:

```
var mycars=new Array("Saab","Volvo","BMW")
```

Napomena: Ako kao vrednosti članova polja navedete vrednosti `true/false` tip promenljivih će biti `numeric` ili `Boolean` a ne `String`.

Možete da se referencirate na konkretan element niza navođenjem imena niza i indeksa. Indeksi počinju od 0.

Sledeća linija koda:

```
document.write(mycars[0])
```

ima sledeći rezultat:

```
Saab
```

Da bi promenili vrednosti u postojećem nizu, samo dodelite novu vrednost odgovarajućem elementu niza:

```
mycars[0]="Opel"
```

Sada, sledeća linija niza:

```
document.write(mycars[0])
```

ima sledeći izlaz:

```
Opel
```

17.4. Boolean objekti

Objekat tipa `Boolean` se koristi da bi konvertovali druge vrednosti u `Boolean` vrednosti (`true` ili `false`).

Primer 17.18 Provera boolean vrednosti

```
<html>
<body>
<script type="text/javascript">
var b1=new Boolean( 0)
var b2=new Boolean(1)
var b3=new Boolean("")
var b4=new Boolean(null)
var b5=new Boolean(NaN)
var b6=new Boolean("false")
document.write("0 is boolean "+ b1 +"<br />")
document.write("1 is boolean "+ b2 +"<br />")
document.write("An empty string is boolean "+ b3 + "<br />")
document.write("null is boolean "+ b4+ "<br />")
document.write("NaN is boolean "+ b5 +"<br />")
document.write("The string 'false' is boolean "+ b6 +"<br />")
</script>
</body>
</html>
```

Objekat tipa `Boolean` definišemo pomoću ključne reči `new`. Sledeći kod definiše objekat tipa `Boolean` koji se zove `myBoolean`:

```
var myBoolean=new Boolean()
```

Napomena: Ako objekat tipa `Boolean` nema inicijalnu vrednost, ili je ta vrednost `0, -0, null, "", false,` nedefinisana ili `NaN`, vrednost objekta je `false`. U svim drugim slučajevima, vrednost je `true` (čak i za string `"false"`)!

Sve linije u narednom kodu kreiraju `Boolean` objekat sa inicijalnom vrednošću `false`:

```
var myBoolean=new Boolean()
var myBoolean=new Boolean(0)
var myBoolean=new Boolean(null)
var myBoolean=new Boolean("")
var myBoolean=new Boolean(false)
var myBoolean=new Boolean(NaN)
```

Sve linije narednog koda kreiraju `Boolean` objekat sa inicijalnom vrednošću `true`:

```
var myBoolean=new Boolean(true)
var myBoolean=new Boolean("true")
var myBoolean=new Boolean("false")
var myBoolean=new Boolean("Richard")
```

17.5. Objekat tipa Math

Objekat tipa `Math` vam omogućava da izvršavate česte matematičke zadatke.

Primer 17.19 round()

```
<html>
<body>
<script type="text/javascript">
document.write(Math.round(0.60) + "<br />")
document.write(Math.round(0.50) + "<br />")
document.write(Math.round(0.49) + "<br />")
document.write(Math.round(-4.40) + "<br />")
document.write(Math.round(-4.60))
</script>
</body>
```

```
</html>
```

Primer 17.20 random()

```
<html>
<body>
<script type="text/javascript">
document.write(Math.random())
</script>
</body>
</html>
```

Primer 17.21

```
<html>
<body>
<script type="text/javascript">
document.write(Math.max(5,7) + "<br />")
document.write(Math.max(-3,5) + "<br />")
document.write(Math.max(-3,-5) + "<br />")
document.write(Math.max(7.25,7.30))
</script>
</body>
</html>
```

Primer 17.22 min()

```
<html>
<body>
<script type="text/javascript">
document.write(Math.min(5,7) + "<br />")
document.write(Math.min(-3,5) + "<br />")
document.write(Math.min(-3,-5) + "<br />")
document.write(Math.min(7.25,7.30))
</script>
</body>
</html>
```

Objekat tipa `Math` uključuje više matematičkih vrednosti i funkcija. Pre korišćenja `Math` objekta, potrebno je da ga definišete.

JavaScript obezbeđuje osam matematičkih vrednosti (konstanti) koje možete da koristite pomoću `Math` object. To su: E, PI, kvadratni koren od 2, kvadratni koren od 1/2, prirodni logaritam od 2, prirodni logaritam od 10, logaritam sa osnovom 2 od E i logaritam sa osnovom 10 od E.

Ove vrednosti možete da koristite u JavaScript-u na sledeći način:

```
Math.E
Math.PI
Math.SQRT2
Math.SQRT1_2
Math.LN2
Math.LN10
Math.LOG2E
Math.LOG10E
```

Pored matematičkih vrednosti u `Math` objektu su definisane i neke funkcije (metodi).

Primer funkcije (metoda):

Sledeći primer koristi metod `round()` `Math` objekta da bi dobio brom zaokružen na najbližu celobrojnu vrednost:

```
document.write(Math.round(4.7))
```

Rezultat ovog koda bi bio:

```
5
```

Sledeći primer koristi metod `random()` `Math` objekta da bi vratio slučajnu vrednost između 0 i 1:

```
document.write(Math.random())
```
Ovaj kod bi mogao da ima sledeći rezultat:
```
0.306122139905322
```

Sledeći primer koristi metode `floor()` i `random()` `Math` objekta da bi vratio slučajan broj između 0 i 10:
```
document.write(Math.floor(Math.random()*11))
```
Prethodni kod bi mogao da ima ovakav izlaz:
```
0
```

18. Detekcija browser-a

JavaScript objekat tipa `Navigator` sadrži podatak o *browser*-u posetioca.

18.1. Primeri

Primer 18.1 Detektovanje browser-a i verzije browser-a posetioca

```
<html>
<body>
<script type="text/javascript">
var browser=navigator.appName
var b_version=navigator.appVersion
var version=parseFloat(b_version)
document.write("Browser name: "+ browser)
document.write("<br />")
document.write("Browser version: "+ version)
</script>
</body>
</html>
```

Primer 18.2 Više detalja o browser-u posetioca

```
<html>
<body>
<script type="text/javascript">
document.write("<p>Browser: ")
document.write(navigator.appName + "</p>")
document.write("<p>Browserversion: ")
document.write(navigator.appVersion + "</p>")
document.write("<p>Code: ")
document.write(navigator.appCodeName + "</p>")
document.write("<p>Platform: ")
document.write(navigator.platform + "</p>")
document.write("<p>Cookies enabled: ")
document.write(navigator.cookieEnabled + "</p>")
document.write("<p>Browser's user agent header: ")
document.write(navigator.userAgent + "</p>")
</script>
</body>
</html>
```

Primer 18.3 Svi detalji o browser-u posetioca

```
<html>
<body>
<script type="text/javascript">
var x = navigator
document.write("CodeName=" + x.appCodeName)
```

```
document.write("<br />")
document.write("MinorVersion=" + x.appMinorVersion)
document.write("<br />")
document.write("Name=" + x.appName)
document.write("<br />")
document.write("Version=" + x.appVersion)
document.write("<br />")
document.write("CookieEnabled=" + x.cookieEnabled)
document.write("<br />")
document.write("CPUClass=" + x.cpuClass)
document.write("<br />")
document.write("OnLine=" + x.onLine)
document.write("<br />")
document.write("Platform=" + x.platform)
document.write("<br />")
document.write("UA=" + x.userAgent)
document.write("<br />")
document.write("BrowserLanguage=" + x.browserLanguage)
document.write("<br />")
document.write("SystemLanguage=" + x.systemLanguage)
document.write("<br />")
document.write("UserLanguage=" + x.userLanguage)
</script>
</body>
</html>
```

Primer 18.4 Upozorenje za korisnika, zavisno od browser-a

```
<html>
<head>
<script type="text/javascript">
function detectBrowser()
{
var browser=navigator.appName
var b_version=navigator.appVersion
var version=parseFloat(b_version)
if ((browser=="Netscape"||browser=="Microsoft Internet Explorer") && (version>=4))
  {alert("Your browser is good enough!")}
else
  {alert("It's time to upgrade your browser!")}
}
</script>
</head>
<body onload="detectBrowser()">
</body>
</html>
```

18.2. Detekcija browser-a

Skoro sve naredbe iz ovog dokumenta rade na svim *browser*-ima koji podržavaju JavaScript. Ipak, neke stvari jednostavno ne rade na određenim *browser*-ima, posebno na starijim *browser*-ima. Dakle, ponekad može da bude posebno korisno da se detektuje tip i verzija *browser*-a. Najbolji način da se ovo uradi je da se napravi takva stranica koja ima prilagođen izgled za većinu *browser*-a. JavaScript sadrži objekat `Navigator`, koji može da se koristi za ove potrebe. Objekat `Navigator` sadrži podatke vezane za ime *browser*-a, verziju i još neke druge.

18.3. Objekat Navigator

`Navigator` objekat sadrži sve podatke o *browser*-u posetioca. Mi ćemo se baviti sledećim atributima ovog objekta:

- `appName` - sadrži ime *browser*-a
- `appVersion` – sadrži, pored ostalog, i verziju *browser*-a

Primer 18.5

```
<html>
<body>
<script type="text/javascript">
var browser=navigator.appName
var b_version=navigator.appVersion
var version=parseFloat(b_version)
document.write("Browser name: "+ browser)
document.write("<br />")
document.write("Browser version: "+ version)
</script>
</body>
</html>
```

Promenljiva `browser` u prethodnom primeru sadrži ime *browser*-a, npr. "*Netscape*" ili "*Microsoft Internet Explorer*".
Atribut `appVersion` vraća string koji sadrži mnogo više informacija od samo broja verzije, ali smo mi za sada zainteresovani samo za ovaj podatak. Da bi izvukli broj verzije iz ovog stringa koristimo funkciju `parseFloat(),` koja izvlači prvu stvar koja izgleda kao decimalni broj iz stringa.
VAŽNA NAPOMENA! Broj verzije u IE 5.0 i kasnije je POGREŠAN! Microsoft počinje string `appVersion` brojeviima 4.0. u IE 5.0 i IE 6.0!!! Ipak, JavaScript je isti u verzijama IE6, IE5 i IE4, tako da je za većinu skriptova sve u redu.

Primer 18.6 Prikazivanje različitog upozorenja, zavisno od verzije browser-a:

```
<html>
<head>
<script type="text/javascript">
function detectBrowser()
{
var browser=navigator.appName
var b_version=navigator.appVersion
var version=parseFloat(b_version)
if ((browser=="Netscape"||browser=="Microsoft Internet Explorer")
&& (version>=4))
  {alert("Your browser is good enough!")}
else
  {alert("It's time to upgrade your browser!")}
}
</script>
</head>
<body onload="detectBrowser()">
</body>
</html>
```

19. Cookie

Cookie se često koristi za identifikovanje korisnika.

19.1. Primeri

Primer 19.1 Kako se kreira cookie za dobrodošlicu

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

```
function getCookie(c_name)
{
if (document.cookie.length>0)
{
c_start=document.cookie.indexOf(c_name + "=")
if (c_start!=-1)
{
c_start=c_start + c_name.length+1
c_end=document.cookie.indexOf(";",c_start)
if (c_end==-1) c_end=document.cookie.length
return unescape(document.cookie.substring(c_start,c_end))
}
}
return null
}
function setCookie(c_name,value,expiredays)
{
var exdate=new Date()
exdate.setDate(expiredays)
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : "; expires="+exdate)
}

function checkCookie()
{
username=getCookie('username')
if (username!=null)
  {alert('Welcome again '+username+'!')}
else
  {
  username=prompt('Please enter your name:',"")
  if (username!=null||username!="")
    {
    setCookie('username',username,365)
    }
  }
}
</script>
</head>
<body onLoad="checkCookie()">
</body>
</body>
</html>
```

19.2. Šta je cookie?

Cookie je promenljiva koja se čuva u računaru posetioca. Svaki put kad *browser* sa nekog računara zatraži stranu, on će poslati i *cookie*. Pomoću JavaScript-a, možete da kreirate i vratite vrednost *cookie*-ja.
Primeri za *cookie*:

- *Cookie* za ime – Prvi put kad posetilac poseti vašu stranu, mora da unese svoje ime. Ime se čuva u *cookie*-ju. Sledeći put kad vas poseti, možete da mu pošaljete pozdravnu poruku tipa "Dobrodošli John Doe!" Ime dobijate od sačuvanog *cookie*-ja
- *Cookie* za šifru - Prvi put kad posetilac poseti vašu stranu, mora da unese šifru. Šifra se čuva u *cookie*-ju. Sledeći put kad vas poseti, šifru dobijate od sačuvanog *cookie*-ja
- *Cookie* za datum - Prvi put kad posetilac poseti vašu stranu, možete da sačuvate trenutni datum u *cookie*-ju. Sledeći put kad vas poseti, možete da mu pošaljete pozdravnu poruku tipa "Vaša poslednja poseta je u utorak, 11. avgusta, 2005!"

19.3. Kreiranje i čuvanje cookie-ja

U ovom primeru kreiramo *cookie* koji čuva ime posetioca. Prvi put kad posetilac poseti našu stranu, dobiće upit o imenu. Ime se čuva u *cookie*-ju. Sledeći put kad poseti istu stranu, dobiće pozdravnu poruku.

Prvo, kreiramo funkciju koja čuva ime posetioca u *cookie* promenljivoj:

```
function setCookie(c_name,value,expiredays)
{
var exdate=new Date()
exdate.setDate(expiredays)
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : ";expires="+exdate)
}
```

Parametri funkcije sadrže ime za *cookie*, vrednost za *cookie* i broj dana za koji *cookie*-ju ističe rok važnosti.
U prethodnoj funkciji smo konvertovali broj dana u validan datum. Nakon toga smo sačuvali ime za *cookie*, vrednost i datum isticanja u objektu `document.cookie`.

Zatim, kreiramo drugu funkciju koja proverava da li je *cookie* postavljen:

```
function getCookie(c_name)
{
if (document.cookie.length>0)
  {
  c_start=document.cookie.indexOf(c_name + "=")
  if (c_start!=-1)
    {
    c_start=c_start + c_name.length+1
    c_end=document.cookie.indexOf(";",c_start)
    if (c_end==-1) c_end=document.cookie.length
    return unescape(document.cookie.substring(c_start,c_end))
    }
  }
return null
}
```

Prethodna funkcija prvo proverava da li je *cookie* sačuvan u objektu `document.cookie`. Ako objekat `document.cookie` sadrži neki *cookie*-je, proverava se da li je naš *cookie* sačuvan. Ako se pronađe naš *cookie*, vraća se njegova vrednost, u suprotnom, vraća se `null`.

Na kraju, kreiramo funkciju koja prikazuje pozdravnu poruku ako je *cookie* postavljen, a ako nije, prikazuje dijalog u kome traži da se unese ime korisnika:

```
function checkCookie()
{
username=getCookie('username')
if (username!=null)
  {alert('Welcome again '+username+'!')}
else
  {
  username=prompt('Please enter your name:',"")
  if (username!=null||username!="")
    {
    setCookie('username',username,365)
    }
  }
}
```

Primer 19.2 Sve zajedno izgleda ovako:

```
<html>
<head>
<script type="text/javascript">
function getCookie(c_name)
{
if (document.cookie.length>0)
  {
  c_start=document.cookie.indexOf(c_name + "=")
  if (c_start!=-1)
    {
    c_start=c_start + c_name.length+1
    c_end=document.cookie.indexOf(";",c_start)
    if (c_end==-1) c_end=document.cookie.length
    return unescape(document.cookie.substring(c_start,c_end))
    }
  }
return null
}
function setCookie(c_name,value,expiredays)
{
var exdate=new Date()
exdate.setTime(exdate.getTime()+(expiredays*24*3600*1000))
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : "; expires="+exdate)
}
function checkCookie()
{
username=getCookie('username')
if (username!=null)
  {alert('Welcome again '+username+'!')}
else
  {
  username=prompt('Please enter your name:',"")
  if (username!=null||username!="")
    {
    setCookie('username',username,365)
    }
  }
}
</script>
</head>
<body onLoad="checkCookie()">
</body>
</html>
```

Prethodni primer poziva funkciju `checkCookie()` prilikom učitavanja strane.

20. JavaScript validacija obrazaca

JavaScript može da se koristi za proveru (validaciju) ulaznih podataka u HTML obrascu pre slanja njihovog sadržaja serveru. Najčešće se vrše sledeće provere:

- da li je korisnik popunio sva potrebna polja?
- da li je korisnik uneo ispravnu e-mail adresu?
- da li je korisnik uneo validan datum?
- da li je korisnik uneo tekst u polje za broj?

20.1. Obavezna polja

Sledeća funkcija proverava da li je popunjeno obavezno polje. Ako nije popunjeno, pojavljuje se poruka upozorenja i funkcija vraća vrednost `false`. Ako je vrednost unešena, funkcija vraća `true` (što znači da su podaci ispravni):

```
function validate_required(field,alerttxt)
{
with (field)
{
if (value==null||value=="")
  {alert(alerttxt);return false}
else {return true}
}
}
```

Primer 20.1 Ceo skript, sa HTML formom može da izgleda ovako:

```
<html>
<head>
<script type="text/javascript">
function validate_required(field,alerttxt)
{
with (field)
{
if (value==null||value=="")
  {alert(alerttxt);return false}
else {return true}
}
}
function validate_form(thisform)
{
with (thisform)
{
if (validate_required(email,"Email must be filled out!")==false)
  {email.focus();return false}
}
}
</script>
</head>
<body>
<form action="submitpage.htm"
onsubmit="return validate_form(this)"
method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

20.2. Provera *e-mail*-a

Sledeća funkcija proverava da li sadržaj ima sintaksu *email* adrese. Ovo znači da ulazni podatak mora da bar sadrži znakove @ i tačku (.). Takođe, @ ne sme da bude prvi znak adrese, i poslednja tačka mora da bude udaljena najmanje jedan znak od znaka @:

```
function validate_email(field,alerttxt)
{
with (field)
{
apos=value.indexOf("@")
dotpos=value.lastIndexOf(".")
if (apos<1||dotpos-apos<2)
```

```
  {alert(alerttxt);return false}
else {return true}
}
}
```

Primer 20.2 Ceo skript, sa HTML formom može da izgleda ovako:

```
<html>
<head>
<script type="text/javascript">
function validate_email(field,alerttxt)
{
with (field)
{
apos=value.indexOf("@")
dotpos=value.lastIndexOf(".")
if (apos<1||dotpos-apos<2)
  {alert(alerttxt);return false}
else {return true}
}
}
function validate_form(thisform)
{
with (thisform)
{
if (validate_email(email,"Not a valid e-mail address!")==false)
  {email.focus();return false}
}
}
</script>
</head>
<body>
<form action="submitpage.htm"
onsubmit="return validate_form(this)"
method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

21. JavaScript animacije

Uz pomoć JavaScript-a možemo da kreiramo animirane slike.

21.1. Primeri

Primer 21.1 Animacija dugmeta

```
<html>
<head>
<script type="text/javascript">
function mouseOver()
{
document.b1.src ="b_blue.gif"
}
function mouseOut()
{
document.b1.src ="b_pink.gif"
}
</script>
</head>
```

```
<body>
<a href="http://www.w3schools.com" target="_blank"
onmouseover="mouseOver()"
onmouseout="mouseOut()">
<img border="0" alt="Visit W3Schools!" src="b_pink.gif" name="b1" width="26" height="26"
/></a>
</body>
</html>
```

21.2. JavaScript animacije

Trik je dozvoliti da JavaScript prikazuje različite slike u zavisnosti od događaja. U narednim primerima ćemo dodati sliku koja treba da se ponaša kao link dugme na strani. Onda ćemo dodati događaje `onMouseOver` i `onMouseOut` koji će pokretati dve JavaScript funkcije koje će menjati slike.

HTML kod izgleda ovako:

```
<a href="http://www.w3schools.com" target="_blank"
onmouseOver="mouseOver()"
onmouseOut="mouseOut()">
<img border="0" alt="Visit W3Schools!"
src="b_pink.gif" name="b1" />
</a>
```

Primetite, slici smo dali ime da bi JavaScript mogao da je kasnije adresira.
Događaj `onMouseOver` kaže *browser*-u da kad miš pređe preko slike, treba da se izvrši funkcija koja će da sliku zameni drugom slikom.
Događaj `onMouseOut` kaže browseru da kad se miš skloni sa slike, druga JavaScript funkcija treba da se izvrši. Ova funkcija vraća staru sliku.
VAŽNO! Događaj miša je dodat tagu `<a>` a ne tagu `<img>.` Na žalost, *browser*-i ne podržavaju događaje miša na slikama!

Zamenu slika vrši sledeći JavaScript:

```
<script type="text/javascript">
function mouseOver()
{
document.b1.src ="b_blue.gif"
}
function mouseOut()
{
document.b1.src ="b_pink.gif"
}
</script>
```

Funkcija `mouseOver()` postavlja sliku "b_blue.gif".
Funkcija `mouseOut()` postavlja sliku "b_pink.gif".

Primer 21.2 Ceo kod izgleda ovako:

```
<html>
<head>
<script type="text/javascript">
function mouseOver()
{
document.b1.src ="b_blue.gif"
}
function mouseOut()
{
document.b1.src ="b_pink.gif"
}
</script>
```

```
</head>

<body>
<a href="http://www.w3schools.com" target="_blank"
onmouseOver="mouseOver()"
onmouseOut="mouseOut()">
<img border="0" alt="Visit W3Schools!"
src="b_pink.gif" name="b1" />
</a>
</body>
</html>
```

22. JavaScript slike-mape

Slika-mapa je slika sa regionima na koje možete da kliknete.

22.1. Primeri

Primer 22.1 Jednostavna HTML slika mapa

```
<html>
<body>
<img src ="planets.gif" width ="145" height ="126" alt="Planets" usemap="#planetmap" />
<map id ="planetmap" name="planetmap">
<area shape ="rect" coords ="0,0,82,126"
href ="sun.htm" target ="_blank" alt="Sun" />
<area shape ="circle" coords ="90,58,3"
href ="mercur.htm" target ="_blank" alt="Mercury" />
<area shape ="circle" coords ="124,58,8"
href ="venus.htm" target ="_blank" alt="Venus" />
</map>
</body>
</html>
```

Primer 22.2 Slika mapa sa dodatim JavaScript-om

```
<html>
<head>
<script type="text/javascript">
function writeText(txt)
{
document.getElementById("desc").innerHTML=txt
}
</script>
</head>
<body>
<img src ="planets.gif" width ="145" height ="126" alt="Planets" usemap="#planetmap" />
<map id ="planetmap" name="planetmap">
<area shape ="rect" coords ="0,0,82,126"
onMouseOver="writeText('The Sun and the gas giant planets like Jupiter are by far the
largest objects in our Solar System.')"
href ="sun.htm" target ="_blank" alt="Sun" />
<area shape ="circle" coords ="90,58,3"
onMouseOver="writeText('The planet Mercury is very difficult to study from the Earth
because it is always so close to the Sun.')"
href ="mercur.htm" target ="_blank" alt="Mercury" />
<area shape ="circle" coords ="124,58,8"
onMouseOver="writeText('Until the 1960s, Venus was often considered a twin sister to the
Earth because Venus is the nearest planet to us, and because the two planets seem to
share many characteristics.')"
href ="venus.htm" target ="_blank" alt="Venus" />
</map>
```

```
<p id="desc"></p>
</body>
</html>
```

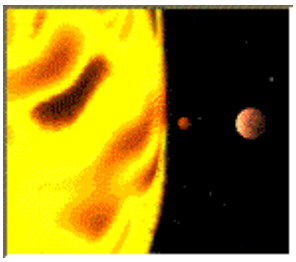
22.2. HTML slike-mape

U HTML skripti smo naučili da je slika-mapa slika sa regionim na koje možete da kliknete. Normalno, svaki region je povezan sa nekim linkom. Ako kliknete na region, idete na odgovarajući link.

Primer 22.3 Primer prikazuje kako da kreirate sliku-mapu. Svaki od regiona je link.

```
<img src ="planets.gif"
width ="145" height ="126"
alt="Planets"
usemap ="#planetmap" />
<map id ="planetmap"
name="planetmap">
<area shape ="rect" coords ="0,0,82,126"
  href ="sun.htm" target ="_blank"
  alt="Sun" />
<area shape ="circle" coords ="90,58,3"
  href ="mercur.htm" target ="_blank"
  alt="Mercury" />
<area shape ="circle" coords ="124,58,8"
  href ="venus.htm" target ="_blank"
  alt="Venus" />
</map>
```

Rezultat



Možemo da dodamo događaje (koji pozivaju JavaScript) tagovima `<area>` unutar slike-mape. Tag `<area>` podržava događaje `onClick`, `onDblClick`, `onMouseDown`, `onMouseUp`, `onMouseOver`, `onMouseMove`, `onMouseOut`, `onKeyPress`, `onKeyDown`, `onKeyUp`, `onFocus`, i `onBlur`.

Evo prethodnog primera, sa dodatim JavaScript-om:

```
<html>
<head>
<script type="text/javascript">
function writeText(txt)
{
document.getElementById("desc").innerHTML=txt
}
</script>
</head>
<body>
<img src="planets.gif" width="145" height="126"
alt="Planets" usemap="#planetmap" />

<map id ="planetmap" name="planetmap">
<area shape ="rect" coords ="0,0,82,126"
```

```
onMouseOver="writeText('The Sun and the gas giant
planets like Jupiter are by far the largest objects
in our Solar System.')"
href ="sun.htm" target ="_blank" alt="Sun" />

<area shape ="circle" coords ="90,58,3"
onMouseOver="writeText('The planet Mercury is very
difficult to study from the Earth because it is
always so close to the Sun.')"
href ="mercur.htm" target ="_blank" alt="Mercury" />

<area shape ="circle" coords ="124,58,8"
onMouseOver="writeText('Until the 1960s, Venus was
often considered a twin sister to the Earth because
Venus is the nearest planet to us, and because the
two planets seem to share many characteristics.')"
href ="venus.htm" target ="_blank" alt="Venus" />
</map>

<p id="desc"></p>

</body>
</html>
```

23. JavaScript vremenski događaji

Pomoću JavaScript-a, moguće je da se neki kod izvrši ne odmah nakon što je funkcija pozvana, već nakon nekog navedenog vremenskog perioda. Ovo su vremenski događaji.

23.1. Primeri

Primer 23.1 Jednostavan tajming

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
{
var t=setTimeout("alert('5 seconds!')",5000)
}
</script>
</head>
<body>
<form>
<input type="button" value="Display timed alertbox!" onClick = "timedMsg()">
</form>
<p>Click on the button above. An alert box will be displayed after 5 seconds.</p>
</body>
</html>
```

Primer 23.2 Jednostavan tajming 2

```
<html>
<head>
<script type="text/javascript">
function timedText()
{
var t1=setTimeout("document.getElementById('txt').value='2 seconds!'",2000)
var t2=setTimeout("document.getElementById('txt').value='4 seconds!'",4000)
var t3=setTimeout("document.getElementById('txt').value='6 seconds!'",6000)
}
```

```
</script>
</head>
<body>
<form>
<input type="button" value="Display timed text!" onClick="timedText()">
<input type="text" id="txt">
</form>
<p>Click on the button above. The input field will tell you when two, four, and six
seconds have passed.</p>
</body>
</html>
```

Primer 23.3 Vremenski događaj sa beskonačnom petljom

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c
c=c+1
t=setTimeout("timedCount()",1000)
}
</script>
</head>
<body>
<form>
<input type="button" value="Start count!" onClick="timedCount()">
<input type="text" id="txt">
</form>
<p>Click on the button above. The input field will count for ever, starting at 0.</p>
</body>
</html>
```

Primer 23.4 Vremenski događaj sa beskonačnom petljom sa Stop dugmetom

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c
c=c+1
t=setTimeout("timedCount()",1000)
}
function stopCount()
{
clearTimeout(t)
}
</script>
</head>
<body>
<form>
<input type="button" value="Start count!" onClick="timedCount()">
<input type="text" id="txt">
<input type="button" value="Stop count!" onClick="stopCount()">
</form>
<p>Click on the "Start count!" button above to start the timer. The input field will
count for ever, starting at 0.
Click on the "Stop count!" button to stop the counting.</p>
</body>
```

```
</html>
```

Primer 23.5 Sat kreiran pomoću vremenskih događaja

```
<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date()
var h=today.getHours()
var m=today.getMinutes()
var s=today.getSeconds()
// add a zero in front of numbers<10
m=checkTime(m)
s=checkTime(s)
document.getElementById('txt').innerHTML=h+":"+m+":"+s
t=setTimeout('startTime()',500)
}
function checkTime(i)
{
if (i<10)
  {i="0" + i}
  return i
}
</script>
</head>
<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

23.2. JavaScript vremenski događaji

Za kreiranje vremenskih događaja se koriste dve metode:

- `setTimeout()` – izvršava neki kod nekad u budućnosti
- `clearTimeout()` – obustavlja funkciju setTimeout()

Napomena: `setTimeout()` i `clearTimeout()` su metodi HTML DOM Window objekta.

23.3. setTimeout()

Sintaksa:

```
var t=setTimeout("javascript naredba",broj milisekundi)
```

Metod `setTimeout()` vraća vrednost – u prethodnoj naredbi, vrednost se čuva u promenljivoj `t`. Ako želite da obustavite metod `setTimeout()`, referencirate ga koristeći ime promenljive.
Prvi parametar funkcije `setTimeout()` je string koji sadrži JavaScript naredbu. Ovo može da bude naredba poput "`alert('5 seconds!')`" ili poziv funkcije, kao "`alertMsg()`".
Drugi parametar navodi za koliko sekundi želite da odložite izvršenje prethodne naredbe.
Napomena: Jedna sekunda sadrži 1000 milisekundi.

Primer 23.6 Kad pritisnete dugme, prikazuje se poruka nakon 5 sekundi.

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
{
var t=setTimeout("alert('5 seconds!')",5000)
}
```

```
</script>
</head>
<body>
<form>
<input type="button" value="Display timed alertbox!"
onClick="timedMsg()">
</form>
</body>
</html>
```

Primer 23.7 – Beskonačna petlja

Da bi naterali tajmer da radi u beskonačnoj petlji, morate da napišete funkciju koja poziva sama sebe. U sledećem primeru, kad se pritisne dugme, ulazno polje počinje da broji od 0 (beskonačno):

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c
c=c+1
t=setTimeout("timedCount()",1000)
}
</script>
</head>
<body>
<form>
<input type="button" value="Start count!"
onClick="timedCount()">
<input type="text" id="txt">
</form>
</body>
</html>
```

23.4. clearTimeout()

Sintaksa

```
clearTimeout(setTimeout_promenljiva)
```

Primer 23.8 Sledeći primer je ista beskonačna petlja iz prethodnog primera. Jedina razlika je što smo dodali dugme"Stop Count!" koje zaustavlja tajmer:

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c
c=c+1
t=setTimeout("timedCount()",1000)
}
function stopCount()
{
clearTimeout(t)
}
</script>
</head>
<body>
```

```
<form>
<input type="button" value="Start count!"
onClick="timedCount()">
<input type="text" id="txt">
<input type="button" value="Stop count!"
onClick="stopCount()">
</form>
</body>
</html>
```

24. JavaScript kreiranje sopstvenih objekata

Objekti su korisni za organizovanje podataka.

24.1. Primeri

Primer 24.1 Kreiranje direktne instance objekta

```
<html>
<body>
<script type="text/javascript">
personObj=new Object()
personObj.firstname="John"
personObj.lastname="Doe"
personObj.age=50
personObj.eyecolor="blue"
document.write(personObj.firstname + " is " + personObj.age + " years old.")
</script>
</body>
</html>
```

Primer 24.2 Kreiranje šablona (template) za objekat

```
<html>
<body>
<script type="text/javascript">
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname
this.lastname=lastname
this.age=age
this.eyecolor=eyecolor
}
myFather=new person("John","Doe",50,"blue")
document.write(myFather.firstname + " is " + myFather.age + " years old.")
</script>
</body>
</html>
```

24.2. JavaScript objekti

U poglavlju 17 smo videli da JavaScript ima nekoliko ugrađenih objekata, kao što su String, Date, Array, i drugi. Pored ovih ugrađenih objekata, možete i da kreirate sopstvene objekte.
Objekat je specijalan vid podataka, sa skupom atributa i metoda.
Primer za ovo bi bio: Osoba je objekat. Atributi su vrednosti povezane sa objektom. Atributi objekta osoba mogu da budu ime, visina, težina, starost, boja kože, boja očiju... Sve osobe imaju ove atribute, ali se vrednosti atributa razlikuju od osobe do osobe. Objekat sadrži i metode. Metodi su akcije koje mogu da se izvrše nad objektom. Metodi objekta osoba bi mogli da budu hranjenje(), spavanje(), rad(), igra()...

Atributi

Sintaksa za pristup atributima objekta je:

```
imeObjekta.imeAtributa
```

Atribut može da se doda objektu, tako što mu se dodeli vrednost. Pretpostavimo da objekat `personObj` već postoji, možete da mu dodate atribut `firstname, lastname, age`, i `eyecolor` na sledeći način:

```
personObj.firstname="John"
personObj.lastname="Doe"
personObj.age=30
personObj.eyecolor="blue"
document.write(personObj.firstname)
```

Prethodni kod daje sledeći izlaz:

```
John
```

Metodi

Objekat može da sadrži i metode.
Metod se poziva na sledeći način:

```
imeObjekta.imeMetoda()
```

Napomena: Parametri koje zahteva metod mogu da se navedu između zagrada.
Metod `sleep()` za objekat `personObj` se poziva na sledeći način:

```
personObj.sleep()
```

24.3. Kreiranje novog objekta

Postoje različiti načini za kreiranje novog objekta:

1. Kreiranje direktne instance objekta
Sledeći kod kreira instancu objekta i dodaje joj četiri atributa:

```
personObj=new Object()
personObj.firstname="John"
personObj.lastname="Doe"
personObj.age=50
personObj.eyecolor="blue"
```

Dodavanje metoda objektu `personObj` je takođe jednostavno. Sledeći kod dodaje motod `eat()` objektu `personObj`:

```
personObj.eat=eat
```

2. Kreiranje objekta-šablona (template)
Šablon definiše strukturu objekta:

```
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname
this.lastname=lastname
this.age=age
this.eyecolor=eyecolor
}
```

Primetite da je šablon u stvari funkcija. Unutar funkcije dodeljujete vrednosti atributima objekta `this`. Razlog korišćenja reči "`this`" je taj što u jednom trenutku može da postoji više različitih osoba. Reč "`this`" se odnosi na konkretnu instancu objekta `person` (na konkretnu osobu).
Kad ste jednom definisali šablon, nove primerke (instance) objekta kreirate na sledeći način:

```
myFather=new person("John","Doe",50,"blue")
myMother=new person("Sally","Rally",48,"green")
```

Takođe možete da dodate i neko metode objektu `person`. Ovo se takođe radi unutar šablona:

```
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname
this.lastname=lastname
this.age=age
this.eyecolor=eyecolor
this.newlastname=newlastname
}
```

Primetite da su metodi samo funkcije dodate objektima. Nakon ovoga, treba da napišemo funkciju `newlastname()`:

```
function newlastname(new_lastname)
{
this.lastname=new_lastname
}
```

Funkcija `newlastname()` definiše novo prezime osobe i dodeljuje ga osobi. JavaScript zna na koju osobu mislite, zato što koristimo reč `this`. Sada, možemo da napišemo kod: `myMother.newlastname("Doe")`.