# JavaScript

# Skript jezici

- Izvršavaju se u čitaču
- Ugrađuju se u HTML stranice
- Omogućavaju interaktivnost
- Interpretirani jezik
  - Ne kompajliraju se
  - Izvršavaju se momentalno

# Script

- Navođenjem taga **<script>** specificira se script kod koji se pokreće direktno u browseru
- Sve između tagova **<script>** i **</script>** browser smatra elementima skripta
- **<script>** tag se može javiti bilo gde u HTML dokumentu
  - Razlika između **<head>** i **<body>** sekcije
  - Script u **<body>** sekciji se izvršava prilikom iscrtavanja stranice
  - Script u **<head>** sekciji se izvršava prilikom poziva skripta u body sekciji
- Script kod se ne mora nalaziti u HTML datoteci

# Script

- Skript kod se može nalaziti u drugoj datoteci, međutim mora se pozvati iz odgovarajuće HTML datoteke
- Ukoliko atribut **type** u **<script>** tagu ima vrednost „**text/javascript“** tada se radi o JavaScript programskom jeziku

# Primer JavaScript

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

# Primer JavaScript u drugoj datoteci

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

# JavaScript

- Sintaksa slična programskom jeziku Java
  - nije programski jezik Java
- Nema tipove podataka
  - kod deklaracije promenljivih se ne stavlja tip (interpreter).
- Nema kreiranja novih klasa
  - ugrađene funkcije,
  - ugrađeni objekti
- Sistem događaja

# Pozivanje JavaScript-a

- Kao reakciju na neki događaj.
- Unutar <script> taga bilo gde unutar HTML dokumenta
  - Ako koristimo JavaScript funkciju, nju moramo da definišemo unutar <head> taga da bismo mogli da je pozivamo iz bilo kog JavaScript koda.
- Kao adresu unutar <a> taga:

```
<a href="javascript:funkcija('parametar');"> klikni</a>
```

# Tipovi podataka

- Osnovni tipovi podataka:
  - Broj
  - Boolean
  - String (između navodnika: "tekst" ili 'tekst')
  - Objekat (kolekcija svojstava i funkcija)
    - Ugrađeni objekti:
      - String – kako? Zar nije primitivan tip? "tekst" se automatski prevodi u String, ako pozovemo metodu ili pristupimo atributu
      - Object
      - Array
      - Math
      - function – svaka funkcija je objekat
      - ostali
    - Korisnički definisani

# Vrednosti promenljivih

- Primitivni tipovi
  - broj koji uvek se čuva kao double floating point
  - true ili false za boolean tip
  - "tekst" ili 'tekst' za tip String
  - null – kao i u ostalim programskim jezicima
  - undefined – neinicijalizovana promenljiva
- Objekti
  - kolekcija svojstava i funkcija, kojoj se pristupa preko tačke ili preko uglastih zagrada
  - null
  - undefined – neinicijalizovana promenljiva

# Promenljive

- Promenljive sadrže informacije
- Deklaracija promenljivih upotrebom ključne reči **var**
- Primer:

```
var a;
var b = 5;
var c = "Pera";
```

- Ako se izostavi ključna reč **var** kod deklaracije promenljive unutar funkcije, ona postaje globalna

# Promenljive

- Nakon deklaracije, varijabla se može inicijalizovati:

```
var x; // trenutno ima vrednost undefined
x = 5;
```

- Inicijalizacija može i uz deklaraciju:

```
var x = 5;
```

- Varijabla može i da promeni tip:

```
var x = 5;
x = "Mika";
```

# Objekti

- Objekti
  - ugrađeni
  - korisnički definisani
- Korisnički definisani objekti se kreiraju:
  - Object objektom
  - JSON sintaksom
  - pozivom konstruktora

# Nizovi

- Slično Javi, sadrže elemente, atribute i metode
- Atribut length daje veličinu niza
- Metoda push dodaje element na kraj niza

# Nizovi

```
// Kreiranje niza Array objektom
        var niz1 = new Array();
        niz1[0] = 5;
        var niz2 = new Array(1, 2, 3, 4);
// Kreiranje niza "JSON" sintaksom
        var niz3 = [];
        niz3[0] = 6;
        var niz4 = [1, 2, 3, 4];
        alert(niz2.length);
// Dodavanje elementa na kraj niza
        niz1.push(33);
// Uklanjanje jednog elementa na poziciji index
        niz1.splice(index, 1);
```

# Aritmetički i operatori dodele

- Aritmetički: **+ - * / % ++ --**

```
x = 5;
y = x * 4;
z = y % 5;
```

- Dodele: **= += -= *= /= %=**

```
y += 5;      y=y+5;
```

- Operator + ima posebno značenje kada su operandi stringovi:

```
a = "Pera";
b = "Car";
c = a + b;
```

- Kada sabiramo stringove i brojeve, rezultat je string

# Aritmetički operatori

- y = 5;

| Operator | Rezultat |
|---|---|
| x=y+2 | x=7 |
| x=y-2 | x=3 |
| x=y%2 | x=1 |
| x=++y | x=6, y=6 |
| x=y++ | x=5, y=6 |
| x=--y | x=4 |

# Operatori dodele

x = 10;

y = 5;

| Operator | Isto kao | Rezultat |
|---|---|---|
| x=y | | x=5 |
| x+=y | x=x+y | x=15 |
| x-=y | x=x-y | x=5 |
| x*=y | x=x*y | x=50 |
| x/=y | x=x/y | x=2 |
| x%=y | x=x%y | x=0 |

# Relacioni operatori

- Relacioni: **==  ===  !=  <  <=  >  >=**

```
x = 5;
if (x == 5)
   document.write("x je jednako 5");
```

- Operator === će porediti i vrednost i tip:

```
var x = 5;
if (x === "5") // == bi vratilo true
   document.write("x je string sa sadržajem 5");
```

- Rezultat relacionih operatora je logička vrednost tačno (true) ili netačno (false)

# Relacioni operatori

- x = 5;

| Operator | Rezultat |
|---|---|
| == | x == 8 je netačno (false) |
| === | x === 5 je tačno (true)<br>x === "5" je netačno (false) |
| != | x != 8 je tačno (true) |
| > | x > 8 je netačno (false) |
| < | x < 8 je tačno (true) |
| >= | x >= 8 je netačno (false) |
| <= | x <= 8 je tačno (true) |

# Logički operatori

- Logički: **&&  ||  !**
- Rezultat logičkih operatora je tačno (true) ili netačno (false)
- Operandi logičkih operatora su logički izrazi

| && | false | true |
|---|---|---|
| false | false | false |
| true | false | true |

| \|\| | false | true |
|---|---|---|
| false | false | true |
| true | true | true |

| ! | |
|---|---|
| false | true |
| true | false |

# Logički operatori

x = 6;

y = 3;

| Operator | Objašnjenje | Primer |
|---|---|---|
| && | konjukcija (and, i) | (x < 10 && y > 1)<br>tačno (true) |
| \|\| | disjunkcija (or, ili) | (x==5 \|\| y==5)<br>netačno (false) |
| ! | negacija (not, ne) | !(x==y)<br>tačno (true) |

# Uslovni operator

- Sintaksa

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

- To je kao:

```
if (uslov)
   promenljiva = vrednost1;
else
   promenljiva = vrednost2;
```

- Primer:

```
x = (y>3)?5:6;
```

# Kontrola toka

- **`if else`**
- **`switch`**
- **`for`**
- **`while`**
- **`do while`**
- **`break`**
- **`continue`**

# if else

- Opšta sintaksa:

```
if(uslov_1)
   telo_1
else if(uslov_2)
   telo_2
else
   telo_3
```

# Primer

```
if (poeni > 94)
  ocena = 10;
else if (poeni > 84)
  ocena = 9;
else if (poeni > 74)
  ocena = 8;
else if (poeni > 64)
  ocena = 7;
else if (poeni > 54)
  ocena = 6;
else ocena = 5;
```

# Primer

```
<script type="text/javascript">
var d = new Date();
var time = d.getHours();

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

# switch

- Izraz u **`switch()`** izrazu mora da proizvede celobrojnu vrednost.
- Ako ne proizvodi celobrojnu vrednost, ne može da se koristi **`switch()`**, već **`if()`**!
- Ako se izostavi **`break`**; propašće u sledeći **`case`**.
- Kod **`default`** izraza ne mora **`break`** - to se podrazumeva.

# Primer

```
switch (a)
{
  case 1:
  case 2: i = j + 6;
          break;
  case 3: i = j + 14;
          break;
  default: i = j + 8;
}
```

# while

- Za cikličnu strukturu kod koje se samo zna uslov za prekid.
- Telo ciklusa ne mora ni jednom da se izvrši
- Opšta sintaksa:

```
while (uslov)
   telo
```

- Važno: izlaz iz petlje na false!

# Primer

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

# do while

- Za cikličnu strukturu kod koje se samo zna uslov za prekid
- Razlika u odnosu na while petlju je u tome što se telo ciklusa izvršava makar jednom.
- Opšta sintaksa:

```
do
  telo
while (uslov);
```

- Važno: izlaz iz petlje na false!

# Primer

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

# for

- Za organizaciju petlji kod kojih se unapred zna koliko puta će se izvršiti telo ciklusa.
- Petlja sa početnom vrednošću, uslovom za kraj i blokom za korekciju.
- Opšta sintaksa:

```
for (inicijalizacija; uslov; korekcija)
 telo
```

# Primer

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

# for ... in petlja

- Za iteriranje kroz nizove
- Opšta sintaksa:

```
for (promenljiva in niz) {
   ...
}
```

# Primer

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

# break i continue

- **`break`** – prekida telo tekuće ciklične strukture (ili **`case`** dela) i izlazi iz nje.
- **`continue`** – prekida telo tekuće ciklične strukture i otpočinje sledeću iteraciju petlje.

# break i continue

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0; i <= 10; i++)
{
  if (i==3)
  {
   break;
  }
  document.write("The number is " + i);
  document.write("<br />");
}
</script>
</body>
</html>
```