Web programiranje

JavaScript

Skript jezici

- Obezbeđuju interaktivnost na web stranicama
- "Jednostavni" programski jezici
- Izvršavaju se u čitaču
- Ugrađuju se u HTML stranice
- Interpretirani jezik
  - nema kompajliranja
  - izvršava se momentalno

Skript

- Tag <script> specificira Script kod koji se pokreće direktno u browser-u
- Browser sve između tagova <script> i </script> smatra elementima skripta
- Tag <script> se može javiti bilo gde u HTML dokumentu.
- Ne mora kod da se nalazi u HTML datoteci
  - može i u drugoj datoteci, a da se pozove iz HTML datoteke
- Ako atribut **type** ima vrednost “**text/javascript**”, tada se radi o JavaScript programskom jeziku

Primer

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

Primer skripta u 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)
    - JIT (Just In Time compiler)
- 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>
```

Promenljive

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

```
var a;
var b = 5;
```

Promenljive

- Nakon deklaracije, varijabla se može inicijalizovati:

```
var x;
x = 5;
```

- Inicijalizacija može i uz deklaraciju:

```
var x = 5;
```

- Varijabla može i da promeni tip:

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

PRIMER: 1. Variables

Aritmetički i operatori dodele

- Aritmetički: + - * / % ++ --
- Dodele: = += -= *= /= %=
- Operator + ima posebno značenje kada su operandi stringovi:

```
var a = “Pera”;
var b = “Car”;
var 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)
   // x je jednako 5
```

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

```
if (x === "5")
  // 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) 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

&&	0	1
0	0	0
1	0	1

			0	1
0	0	1
1	1	1

!	
0	1
1	0

Logički operatori

```
x = 6;
y = 3;
```

Operator	Objašnjenje	Primer
&&	konjukcija (and, i)	(x < 10 && y > 1) tačno (true)
||	disjunkcija (or, ili)	(x==5 || y==5) netačno (false)
!	negacija (not, ne)	!(x==y) 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){
   // ...
}else if(uslov_2) {
   // ..
}else {
   // ...
}
```

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;
```

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.

switch

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

PRIMER: 3. switch

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){
}
```

- Važno: izlaz iz petlje na false!

Primer

```
<html>
   <body>
      <script type="text/javascript">
         //racunanje a na n
         i = 1; a = 2; n = 3;
         stepen = 1;
         while (i++ <= n) {
            stepen *= a;
            document.write("a na n je " + stepen);
         }
      </script>
   </body>
</html>
```

PRIMER: 4. while

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) {
   // ...
}
```

PRIMER: 5. for

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.

PRIMER: 6. break

7. continue

Primer – izlaz iz ugnježdene petlje

```
for (…) {
   for (…) {
      ...
      if (uslov) {
        break;
      }
   }
}
```

for ... in petlja

- Za iteriranje kroz nizove
- Opšta sintaksa:

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

PRIMER: 8. for_in

Funkcije

- Opšta sintaksa:

```
function ime_funkcije(arg1, arg2, …) {
     …
     return vrednost;
}
```

- Ako se javascript kod piše u HTML datoteci, funkcije se definišu u head delu.

PRIMER: 9. functions

Događaji

- Događaji se registruju i odrađuju *event handler*-ima
- U skoro svaki element se može staviti atribut tipa događaja koji ima kao vrednost ime funkcije koja će se aktivirati (*event handler*)
- Primer:

```
<body onload="ucitavanje()">
```

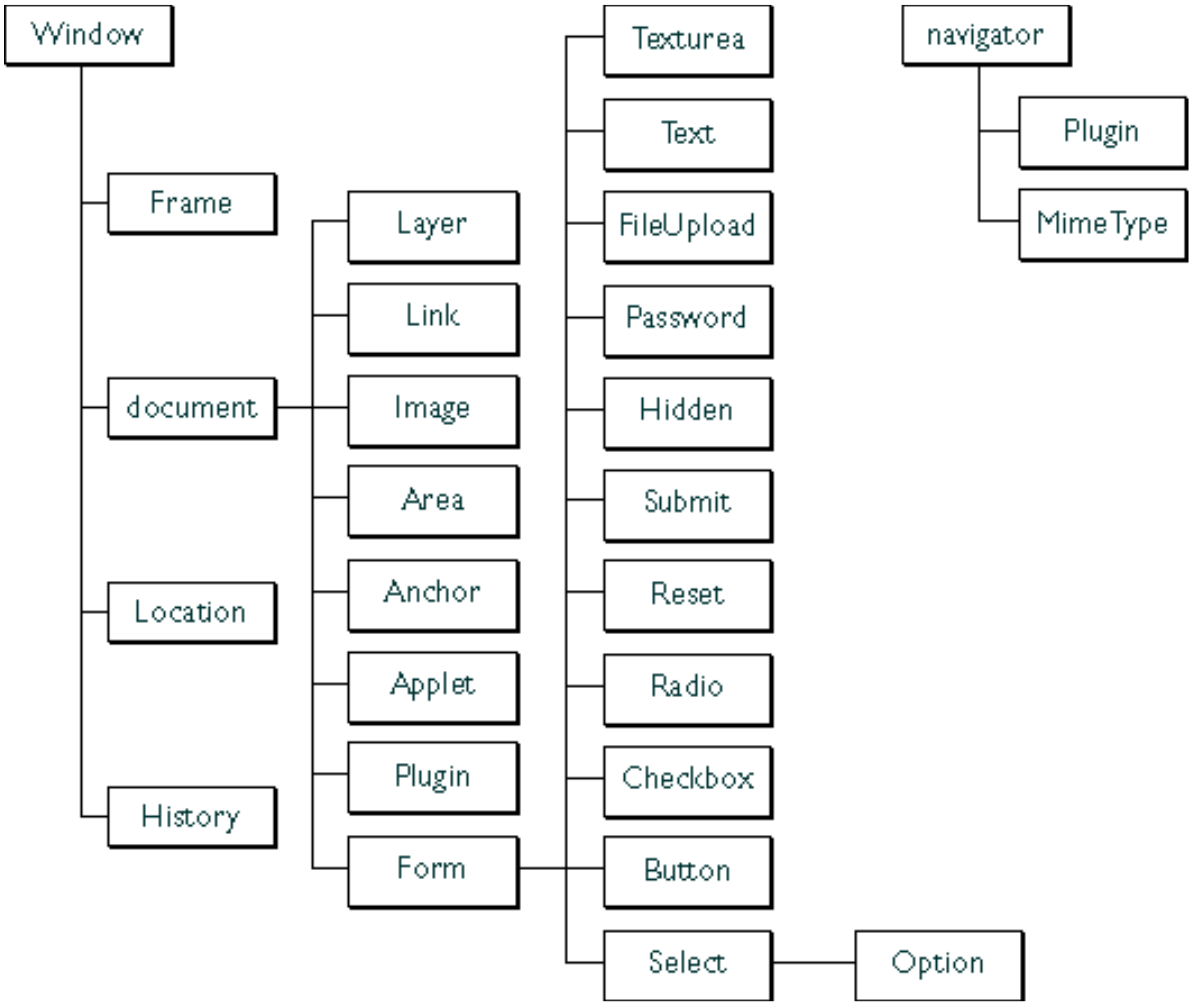
PRIMER: 10. events

Ugrađene funkcije

- Sistemske funkcije:
  - `isNaN()` – vraća true ako prosleđeni string nije broj,
  - *`eval`*`()` – interpretira prosleđeni string kao JavaScript kod,
  - `parseInt()` – parsira string u celi broj,
  - `parseFloat()` – parsira string u decimalni broj,
  - `console.log()` - ispisuje zadani tekst u konzolu browsera.
  - U `Math` biblioteci se mogu naći korisne matematičke funkcije.

PRIMER: 11. Math

Hijerarhija objekata



Window objekt

- Omogućuje manipulaciju prozorima
- Sadrži informacije o tekućem prozoru
- Metode:
  - `alert()`, `confirm()`, `prompt()` - poruka u prozoru (MessageBox)
  - `back()`, *`forward`*`()` - povratak na prethodnu stranicu/odlazak na sledeću iz istorije.
  - `moveBy()`, `MoveTo()` - pomera prozor
  - `open()` - otvara nov prozor
  - `setTimeout()` / `clearTimeout()` – podešava/isključuje kod koji će se izvršavati kada istekne timeout
  - `setInterval()` / `clearInterval()` – zadaje funkciju koja će se periodično izvršavati
- Atributi:
  - `history` - istorija odlazaka na stranice,
  - `document` - tekući HTML dokument,
  - `frames` - niz svih frejmova u prozoru,
  - `location` – kompletan URL tekuće stranice,
  - `statusbar` - statusna linija na dnu ekrana

PRIMER: 12. window

Location objekt

- Reprezentuje URL stranice koja je učitana u navigator:

  `location = "http://www.google.com"`
- Sadrži informacije o tekućem dokumentu
- Metode:
  - `reload()` - ponovno učitavanje tekućeg prozora
  - `replace()` - učitava novi URL sa novom istorijom.
  - `assign()` - učitava novi URL i ažurira postojeću istoriju.
- Atributi:
  - `href` – pun URL do stranice:
  - `protocol` – protokol iz URL-a
  - `host` – adresa servera iz URL-a
  - `port` – port iz URL-a
  - `pathname` – putanja do resursa
  - `search` – parametri forme

PRIMER: 13. location

History objekt

- Omogućuje kontrolu pristupa već viđenim stranicama
- Sadrži listu adresa posećenih stranica
- Metode:
  - `back()` - učitava prethodnu stranicu iz liste
  - `forward()` – učitava sledeću stranicu iz liste
  - `go()` - učitava zadatu adresu iz liste
- Atributi:
  - `current` – trenutno učitana adresa
  - `length` – broj stavki u history listi
  - `next` – zadavanje sledećeg elementa
  - `previous` – zadavanje prethodnog elementa

Document objekt

- Omogućuje ispis HTML-a na ekran
- Sadrži informacije o tekućem dokumentu
- Metode:
  - `write()` - ispisuje na ekran tekst
- Atributi:
  - `forms` - niz svih formi u dokumentu
  - `links` - niz svih linkova u dokumentu
  - `applets` - niz svih apleta u dokumentu
  - `title` - sadrzaj **title** taga

String objekt

- Reprezentuje string
  - string konstanta “tekst” reprezentuje string
- Metode:
  - `substring()` – vraća deo stringa
  - *`split`*`()` – vraća niz stringova kao rezultat “razbijanja” stringa
  - `indexOf()`, `lastIndexOf()` – vraća poziciju nekog podstringa
  - `charAt()` – vraća karakter sa zadate pozicije
- Atributi:
  - `length` – dužina stringa

Forme

- Reprezentovane **form** objektom.
- Metode:
  - `submit()` - šalje podatke iz forme na odredište definisano `action` atributom `form` taga.
  - `reset()` - simulira pritisak na Reset dugme forme.
- Atributi:
  - `elements` - niz elemenata forme. Svaki element ima **value** atribut za pristup sadržaju,
  - `length` - broj elemenata na formi.
  - `action` - sadržaj action atributa.

PRIMER: 11. forms

Document Object Model (DOM)

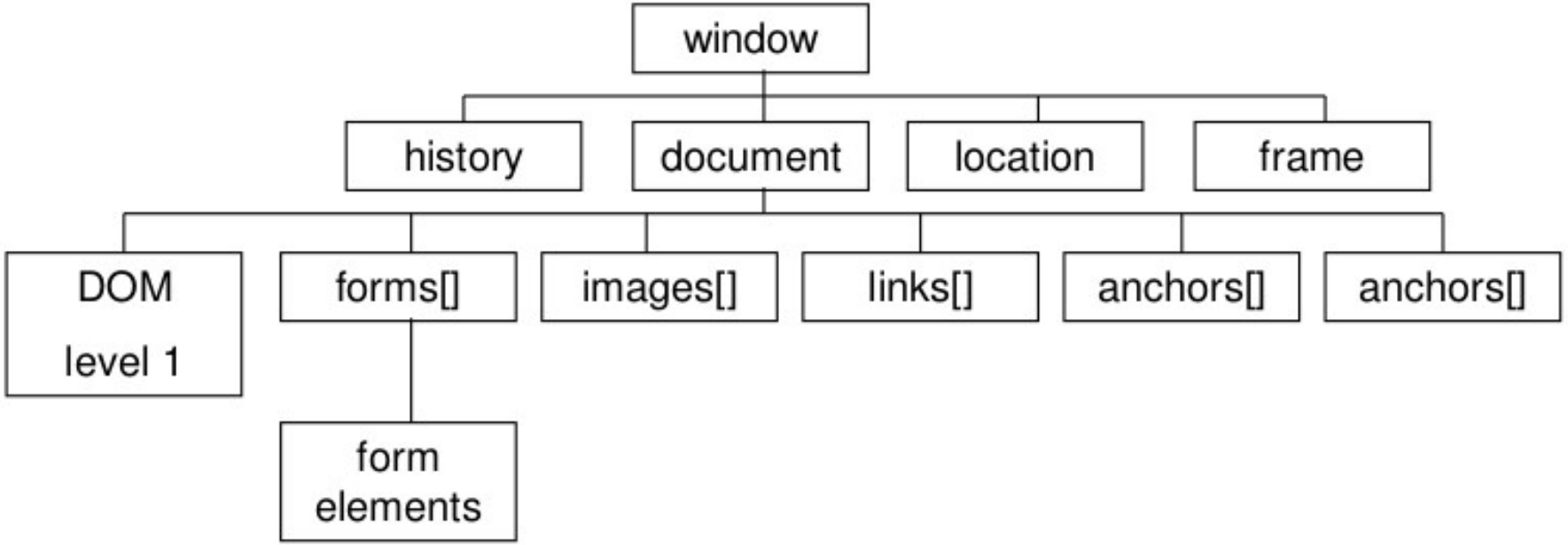
- DOM predstavlja objektnu reprezentaciju XML dokumenta.
- JavaScript poseduje skup funkcija za rad sa DOM objektima.
- Postoji više nivoa reprezentacije:
  - DOM Level 0 i
  - DOM Level 1,
  - DOM Level 2,
  - DOM Level 3.

DOM reprezentacija

- HTML dokument se posmatra kao stablo koje se sastoji iz elemenata
- Koren stabla je `<html>` tag
- Svaki HTML tag je čvor tipa element u stablu
- Svaki atribut je čvor tipa atribut u stablu
- Svaki tekst je čvor tipa tekst (tekstualni čvor) u stablu
- Svaki komentar je čvor tipa komentar u stablu

DOM Level 0

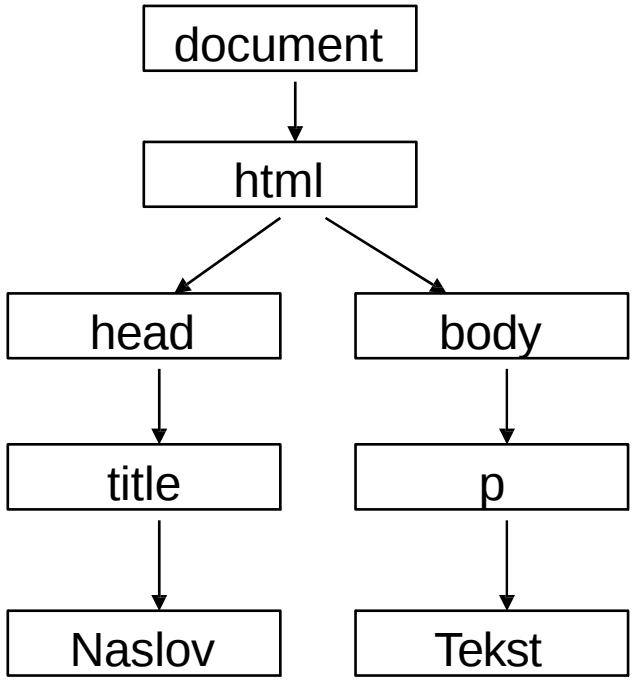
- DOM Level 0 omogućuje pristup elementima stranice preko predefinisanih objekata.



DOM Level 1-3

- DOM nivoi 1-3 predstavljaju objektnu reprezentaciju sadržaja HTML dokumenta
- Primer:

```
<html>
   <head>
       <title>Naslov</title>
   </head>
   <body>
       <p>Tekst</p>
   </body>
</html>
```



DOM Stablo

- Svaki HTML dokument se posmatra kao DOM stablo
- Čvor na vrhu se zove korenski čvor
- Svaki čvor osim korenskog ima jednog roditelja (čvor iznad)
- Svaki čvor može da ima potomke (decu – čvorovi ispod)
- List je čvor bez dece
- Čvorovi istog nivoa (sibiling) su čvorovi sa istim roditeljem.

DOM i JavaScript

- DOM objektima se može pristupiti jedino iz skripta.
- JavaScript poseduje atribute i metode za pristup DOM elementima.
- Osnovni element je document objekat.
  - On sadrži sve čvorove DOM stabla koji reprezentuju HTML stranicu
- Obično se elemente HTML stranice u DOM stablu pronalazi pomoću `id` atributa.

Objekat tipa čvor (node)

- Atributi:
  - `nodeName` – ime čvora
  - `nodeType` – tip čvora (1 za HTML tagove, 2 za atribute, 3 za tekstualne čvorove, 8 za komentar, 9 za dokument)
  - `nodeValue` – sadržaj tekstualnog čvora
  - `innerHTML` – sadržaj čvora kao HTML
  - `id` – ID čvora
  - `firstChild`, `lastChild` – prvi/poslednji čvor ispod u hijerarhiji
  - `childNodes` – niz čvorova koji su u prvom nivou ispod, u hijerarhiji
  - `parentNode` – objekat koji sadrži tekući čvor
- Atributi stila – svaki čvor ima atribut stila – `style`:
  - *`cvor.style.top = 10`* – stil {top:10}
  - *`cvor.style.visibility`*=“`visible`” – stil {visibility:visible}
- Ako je naziv stila sa crticom, u JavaScriptu se spaja i koristi veliko slovo:
  - *`cvor.style.borderWidth = 0`*

Objekat tipa čvor (node)

- Metode:
  - `appendChild()` – dodaje tekućem čvoru novi čvor, na kraj prvog nivoa ispod u hijerarhiji
  - `insertBefore()` – ubacuje zadati čvor ispred drugog čvora
  - `removeChild()` – uklanja zadati čvor iz stabla
  - `getAttribute()` – vraća vrednost zadatog atributa
  - `setAttribute()` – postavlja vrednost atributa
  - `removeAttribute()` – uklanja zadati atribut
  - `hasAttributes()` – vraća true ako tekući čvor ima atribute