



Osnove web programiranja

Script jezici, JavaScript, BOM, DOM

Termin 10

Sadržaj

1. *Script* jezici
2. *JavaScript* uvod

Dodatno:

1. *JavaScript* jezik nastavak
2. *JavaScript* jezik objekti osnovno
3. BOM (*Browser Object Model*)
4. DOM (*Document Object Model*)

Podsetnik

- u dosadašnjem pristupu u razvoju *web* aplikacija, implementacija na strani servera je upravljala izgledom *web* stranice
- za svaku i **najmanju promenu u prikazu stranice** (npr. poruka o greški), server je morao da generiše **potpuno novu stranicu** sa dodatkom te male promene
- ovakav pristup je sasvim prihvatljiv za aplikacije u kojima ne dolazi do česte promene prikaza
- promene prikaza su do sada bile potrebne tek kada korisnik inicira neku akciju (klik na *link*, klik na *submit* dugme forme i sl.)
- dodatno, server je obavljao validaciju unosa čak i za vrednosti koje su se mogle proveriti i pre slanja (npr. prazno polje za unos)

Script jezici

- da bi **izgled HTML stranice** mogao da se **menja lokalno** kod korisnika u *web browser*-u bez direktnog učešća servera, neophodno je da:
 1. *web browser* može da izvršava naredbe na nekom programskom jeziku koje će da upravljaju tom izmenom
 2. server sa prvim slanjem HTML stranice pošalje i pridruženi program na tom jeziku
- prethodno važi i logiku validacije unosa i eventualne druge namene (interaktivnost)
- da bi takva *web aplikacija* mogla da se izvršava na svim *web browser*-ima i na svim platformama, neophodno je da:
 1. programske naredbe, koje *web browser*-i izvršavaju, budu napisane na istom **standardnom jeziku**
 2. se **naredbe ne prevode u izvršni oblik**, već da se njihova sintaksa evaluiira i izvršava *ad hoc* (da budu interpretirane)

JavaScript



- standardni jezik koga *podržavaju svi web browser-i*
- *interpretirani* jezik
- samo sintaksa liči na sintaksu *Java* programskog jezika i tu se završavaju sličnosti
- *dinamički tipiziran*
- podržava *reference na funkcije*
- identifikatori promenljivih i funkcija su *case-sensitive*

JavaScript

script tag

- navodi se u HTML dokumentima
- ako se navede kao podelement *head tag*-a, tada se u njemu navodi *JavaScript* kod koji će moći da se izvrši na poziv pri učitavanju stranice ili nakon učitavanja stranice (npr. definicije funkcija)
- ako se navede kao podelement *body tag*-a, tada se u njemu navodi *JavaScript* kod koji se izvršava onog momenta kada *web browser* pri učitavanju stranice naiđe na njega

```
<head>
...

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

...
</head>
```

```
<body>
...

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

...
</body>
```

JavaScript

script tag

- umesto u **head tag-u**, *JavaScript* kod koji će moći da se izvrši **na poziv pri učitavanju stranice ili nakon učitavanja stranice** (npr. definicije funkcija) se može izmestiti u eksternu **.js** datoteku
- to omogućuje da se **na različitim HTML stranicama** koriste **iste funkcije** definisane u zajedničkoj **.js** datoteci
- *JavaScript* **biblioteke** se sastoje od jedne ili više **.js** datoteka

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



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



eksterna_datoteka.js

Dodatno JavaScript

JavaScript

Naredbe

- naredba iza koje se ne nalazi ništa više u istoj liniji koda, ne mora da se zatvori znakom ;

```
// naredba dodele  
a = 1  
  
// poziv funkcije  
funkcija()  
  
// uzastopne naredbe  
a = 1; b = 2; funkcija()
```

operator dodele

komentari

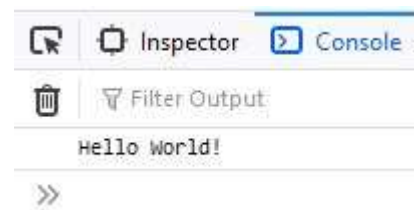
JavaScript

Funkcija *console.log(...)*

- ispisuje vrednost argumenta u konzoli *web browser-a*

```
<body>
  <script type="text/javascript">
    console.log("Hello World!")
  </script>
</body>
```

string literal; zatvara se znacima " ili '



tasterima **Control + Shift + K**
se u web browser-u otvara
Web konzola

JavaScript

Promenljive

- identifikatori promenljivih se navode nakon ključne reči **var**
- ne navodi se tip promenljive
- promenljiva poprima tip onog momenta kada se u nju upiše vrednost

```
var a = "1"      ← string
var b = 2        ← number
var c = false    ← boolean
var d = null
```

konkatenacija



```
console.log("vrednost promenljive a je: " + a)
console.log("vrednost promenljive b je: " + b)
console.log("vrednost promenljive c je: " + c)
console.log("vrednost promenljive d je: " + d)
```

```
vrednost promenljive a je: 1
vrednost promenljive b je: 2
vrednost promenljive c je: false
vrednost promenljive d je: null
```

```
var a = "1", b = 2, c = false, d = null
console.log("vrednost promenljive a je: " + a)
console.log("vrednost promenljive b je: " + b)
console.log("vrednost promenljive c je: " + c)
console.log("vrednost promenljive d je: " + d)
```

```
vrednost promenljive a je: 1
vrednost promenljive b je: 2
vrednost promenljive c je: false
vrednost promenljive d je: null
```

JavaScript

Promenljive

- promenljiva može da menja tip onda kada se u nju upiše vrednost nekog drugog tipa

```
var a = "1"  
console.log("vrednost promenljive a je: " + a)
```

vrednost promenljive a je: 1

```
a = 2  
console.log("vrednost promenljive a je: " + a)
```

vrednost promenljive a je: 2

JavaScript

Aritmetički operatori

<code>console.log(3 + 2)</code>	5
<code>console.log(3 - 2)</code>	1
<code>console.log(3 * 2)</code>	6
<code>console.log(3 / 2)</code>	1.5
<code>console.log(3 % 2) // celobrojni ostatak pri deljenju</code>	1
<code>var a = 3; var b = a++; console.log("a: " + a + ", b: " + b)</code>	a: 4, b: 3
<code>var a = 3; var b = ++a; console.log("a: " + a + ", b: " + b)</code>	a: 4, b: 4
<code>var a = 3; var b = a--; console.log("a: " + a + ", b: " + b)</code>	a: 2, b: 3
<code>var a = 3; var b = --a; console.log("a: " + a + ", b: " + b)</code>	a: 2, b: 2

JavaScript

Operatori dodele

<code>var a = 3; var b = 2; a = b; console.log(a)</code>	2
<code>var a = 3; var b = 2; a += b; console.log(a)</code>	5
<code>var a = 3; var b = 2; a -= b; console.log(a)</code>	1
<code>var a = 3; var b = 2; a *= b; console.log(a)</code>	6
<code>var a = 3; var b = 2; a /= b; console.log(a)</code>	1.5
<code>var a = 3; var b = 2; a %= b; console.log(a)</code>	1

Relazioni operatori

<code>console.log(3 > 2)</code>	<code>true</code>
<code>console.log(3 >= 2)</code>	<code>true</code>
<code>console.log(3 < 2)</code>	<code>false</code>
<code>console.log(3 <= 2)</code>	<code>false</code>
<code>console.log((3 == 2) + ", " + (3 == 3))</code>	<code>false, true</code>
<code>console.log((3 != 2) + ", " + (3 != 3))</code>	<code>true, false</code>
<code>console.log(3 == "3") // poređenje po vrednosti</code>	<code>true</code>
<code>console.log(3 === "3") // poređenje po tipu i vrednosti</code>	<code>false</code>

JavaScript

Logički operatori

```
console.log((true && false) + ", " + (true && true))  
console.log((true || false) + ", " + (true || true))  
console.log(!false + ", " + !true)
```

```
false, true  
true, true  
true, false
```


JavaScript

Izrazi

```
var a = 4
```

```
var ime = "Pera"
```

```
console.log(3 + 2)
```

5

```
console.log("ime: " + ime)
```

ime: Pera

```
console.log(3 + a)
```

7

```
console.log(3 + Math.sqrt(9))
```

6

```
console.log(3 + a + Math.sqrt(9))
```

10

```
console.log(3 + 2 * 2)
```

7

```
console.log((3 + 2) * 2)
```

10

```
console.log((1 == 1)? "tačno": "netačno")
```

tačno

```
console.log((1 != 1)? "tačno": "netačno")
```

netačno

Selekcije

```
var sati = 10
if (sati < 12) {
    console.log("prepodne")
}
```

prepodne

```
var sati = 13
if (sati < 12) {
    console.log("prepodne")
} else {
    console.log("poslepodne")
}
```

poslepodne

```
var sati = 20
if (sati < 10) {
    console.log("Dobro jutro!")
} else if (sati < 18) {
    console.log("Dobar dan!")
} else {
    console.log("Dobro veče!")
}
```

Dobro veče!

JavaScript

Switch

```
var biljka = "banana"

var rezultat;
switch (biljka) {
  case "banana":
  case "pomorandža":
    rezultat = "voće"
    break
  case "krompir":
    rezultat = "povrće"
    break
  default:
    rezultat = "nedefinisana"
}
console.log(biljka + " je " + rezultat)
```

banana je voće

JavaScript

Petlje

```
var granica = 3
```

```
var it = 1;  
while (it <= granica) {  
    console.log("while: " + it)  
    it++  
}
```

```
while: 1  
while: 2  
while: 3
```

```
var it = 1;  
do {  
    console.log("do-while: " + it)  
    it++  
} while (it <= granica)
```

```
do-while: 1  
do-while: 2  
do-while: 3
```

```
for (var it = 1; it <= granica; it++) {  
    console.log("for: " + it)  
}
```

```
for: 1  
for: 2  
for: 3
```

JavaScript

break, continue

- važi za sve vrste petlji

```
var granica = 5
var preskoci = 2
var prekini = 4
for (var it = 1; it <= granica; it++) {
  if (it == preskoci) {
    continue
  }
  if (it == prekini) {
    break
  }
  console.log("for: " + it)
}
```

for: 1
for: 3

JavaScript

Nizovi

```
var brojevi = ["one", "dva", "tri"]  
console.log(brojevi)
```

Array(3) ["one", "dva", "tri"]

```
var brojevi = [] // prazan niz  
brojevi[0] = "one"  
brojevi[1] = "dva"  
brojevi[2] = "tri"  
console.log(brojevi)
```

Array(3) ["one", "dva", "tri"]

```
// da li sadrži element?  
console.log(  
    brojevi.includes("jedan") + ", " +  
    brojevi.includes("one")  
)
```

false, true

```
brojevi[0] = "jedan" // zamena  
brojevi.push("četiri") // dodavanje na kraj  
console.log(brojevi)  
brojevi.pop() // uklanjanje sa kraja  
console.log(brojevi)  
// izdvajanje elemenata u opsegu; vraća modifikovani niz  
brojevi = brojevi.slice(1, 3)  
console.log(brojevi)
```

Array(4) ["jedan", "dva", "tri", "četiri"]

Array(3) ["jedan", "dva", "tri"]

Array ["dva", "tri"]

JavaScript

for-each


```
var brojevi = ["jedan", "dva", "tri"]
for (var it in brojevi) {
    console.log("it: " + it + ", broj: " + brojevi[it])
}
```

```
it: 0, broj: jedan
it: 1, broj: dva
it: 2, broj: tri
```

Funkcije

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Funkcije</title>
  <script type="text/javascript">
    function naslov() {
      console.log("Primer sa funkcijama")
      console.log("-----")
    }
    function saberi(operandA, operandB) {
      return operandA + operandB
    }
  </script>
</head>
<body>
  <script type="text/javascript">
    naslov()

    var a = 3
    var b = 2
    var rezultat = saberi(a, b)
    console.log(a + " + " + b + " = " + rezultat)
  </script>
</body>
</html>
```

A diagram consisting of green arrows. One arrow points from the `naslov()` call in the body script block to the `function naslov()` definition in the head script block. Another arrow points from the `saberi(a, b)` call in the body script block to the `function saberi(operandA, operandB)` definition in the head script block. A third arrow points from the `var a = 3` and `var b = 2` declarations in the body script block to the `naslov()` call, indicating that these variables are used within the function.

Primer sa funkcijama

3 + 2 = 5

Funkcije

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>Funkcije</title>
```

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

```
    var saberi = function(operandA, operandB) {  
      return operandA + operandB  
    }
```

```
    var oduzmi = function(operandA, operandB) {  
      return operandA - operandB  
    }
```

```
    var operacije = [saber, oduzmi] ← niz funkcija
```

```
  </script>
```

```
</head>
```

```
<body>
```

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

```
    var a = 3
```

```
    var b = 2
```

```
    var sabiranje = operacije[0](a, b)  
    console.log(a + " + " + b + " = " + sabiranje)
```

```
    var oduzimanje = operacije[1](a, b)  
    console.log(a + " - " + b + " = " + oduzimanje)
```

```
  </script>
```

```
</body>
```

```
</html>
```

reference na funkcije

3 + 2 = 5

3 - 2 = 1

JavaScript

Ugrađene funkcije

```
// provera da li se string ne može parsirati u broj
console.log(isNaN("nije broj") + ", " + isNaN("1.5"))
// izvršava string kao JavaScript kod
eval("for (var it = 0; it < 3; it++) {console.log(it)}")
```

true, false

0
1
2

```
console.log(parseInt("1")) // parsira string u ceo broj
console.log(parseFloat("1.5")) // parsira string u realan broj
```

1
1.5

```
var kodiraniURL = escape("localhost:8080/URL sa razmakom") // kodira url
console.log(kodiraniURL)
var dekodiraniURL = unescape(kodiraniURL) // dekodira url
console.log(dekodiraniURL)
```

localhost%3A8080/URL%20sa%20razmakom

localhost:8080/URL sa razmakom

JavaScript

String metode i atributi

```
var poruka = "Hello World!"  
console.log(poruka.length)  
console.log(poruka.substring(1, 10))  
console.log(poruka.split(" "))  
console.log(poruka.indexOf("l"))  
console.log(poruka.lastIndexOf("l"))  
console.log(poruka.charAt(6))
```

```
12  
ello Worl  
Array [ "Hello", "World!" ]  
2  
9  
W
```

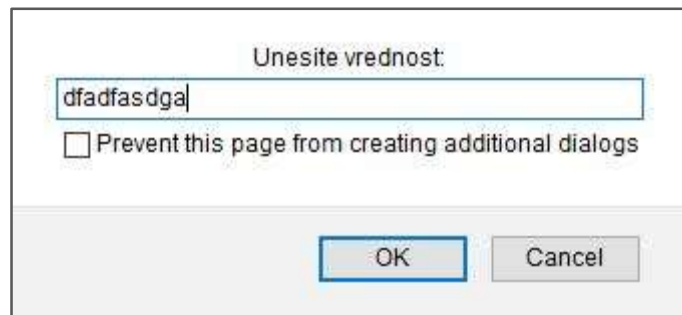
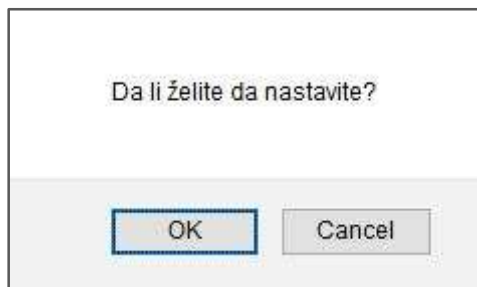
```
var poruka = "$%^ ^&*()"  
console.log(poruka.match("[a-zA-Z0-9]+$") != null)  
var poruka = "1a"  
console.log(poruka.match("[a-zA-Z0-9]+$") != null)
```

```
false  
  
true
```

JavaScript

Poruke

```
// prikazuje pitanje i vraća korisnikov odgovor u vidu boolean rezultata
var nastavak = confirm("Da li želite da nastavite?")
if (nastavak) {
    // traži unos od korisnika i vraća unesenu vrednost
    var vrednost = prompt("Unesite vrednost:")
    // prikazuje poruku
    alert("Uneli ste: " + vrednost)
}
```

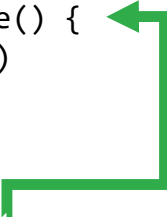


JavaScript

Događaji

- **funkcije**, definisane u *script tag*-u u *head tag*-u HTML dokumenta ili u eksternim datotekama, se ne moraju pozivati samo u *script tag*-ovima u *body tagu* HTML dokumenta, već se mogu **pozivati** i pri korisnički izazvanim akcijama i u drugim posebnim vremenskim trenucima – **događajima** (npr. klik na dugme, pritisak tastera, završetak učitavanja dokumenta i sl.)
- tada se ove funkcije zovu **rukovaoci događajima** (*event handlers*)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Poruke</title>
  <script type="text/javascript">
    function klikNaDugme() {
      alert("onclick")
    }
  </script>
</head>
<body>
  <button onclick="klikNaDugme()">kliknite na dugme</button>
</body>
</html>
```



JavaScript

Događaji

Događaj	Dešava se kada...	Najčešća upotreba
onload	... se stranica učitava, ako se definiše za <i>body</i> element ... se element učitava, ako se definiše za bilo koji drugi element	kada se stranica ili neki element dugo učitava, a neka automatska procedura zahteva da element bude učitavan
onclick	... korisnik klikne na element	validacija unosa forme
onsubmit	... korisnik <i>submit</i> -uje formu	
onmouseover	... korisnik pređe mišem preko elementa	promena izgleda elementa, ispis pojašnjenja, prikaz dodatnih opcija
onfocus	... korisnik uđe u polje za unos	
onblur	... korisnik napusti polje za unos	validacija unosa za celo polje
onkeydown	... korisnik pritisne taster	validacija unosa za svaki karakter
onkeypress	... korisnik pritisne pa otpusti taster, ili drži taster	
onkeyup	... korisnik otpusti taster	
i mnogi drugi...		

JavaScript

Link-ovi

- **href** atribut *link*-a može da sadrži *JavaScript* poziv i tada on definiše ponašanje pri kliku na *link*
- ako funkcija koja se poziva ne vraća ništa, tada se na klikom na *link* izvršava procedura koja definisana u funkciji
- ako funkcija koja se poziva vraća *string* tada se klikom na link prikazuje i interpretira taj *string* kao HTML sadržaj

```
<a href="javascript:funkcija()">kliknite na ovaj link</a>
```

Dodatno JavaScript objekti

JavaScript objekti

JavaScript raspolaže ugrađenim klasama:

- *String*
- *Number*
- *Array*
- *Map*
- *Date*
- *Math*

i mnogim drugim...

JavaScript objekti

String

```
var poruka = new String("Hello World!")
console.log(poruka.length)
console.log(poruka.substring(1, 10))
console.log(poruka.split(" "))
console.log(poruka.indexOf("l"))
console.log(poruka.lastIndexOf("l"))
console.log(poruka.charAt(6))
```

```
12
ello Worl
Array [ "Hello", "World!" ]
2
9
W
```

```
var poruka = new String("$%^ ^&*()")
console.log(poruka.match("[a-zA-Z0-9]+$") != null)
var poruka = new String("1a")
console.log(poruka.match("[a-zA-Z0-9]+$") != null)
```

```
false
true
```

JavaScript objekti

Number

<code>console.log(Number.MAX_VALUE)</code>	<code>1.7976931348623157e+308</code>
<code>console.log(Number.MIN_VALUE)</code>	<code>5e-324</code>
<code>console.log(Number.NaN)</code>	<code>NaN</code>
<code>console.log(Number.isNaN("nije broj"))</code>	<code>false</code>
<code>console.log(Number.isInteger("1.5"))</code>	<code>false</code>
<code>console.log(Number.parseInt("1.5"))</code>	<code>1</code>
<code>console.log(Number.parseFloat("1.5"))</code>	<code>1.5</code>
<code>var broj = new Number("1.532465")</code>	
<code>console.log(broj.toFixed(2))</code>	<code>1.53</code>
<code>// broj cifara iza decimalne tačke</code>	

JavaScript objekti

Array

```
var brojevi1 = Array.of("one", "dva", "tri")
var brojevi1 = new Array("one", "dva", "tri")
console.log(brojevi1)
var brojevi2 = Array.from(brojevi1) // kopiranje
console.log(brojevi2)
```

Array(3) ["one", "dva", "tri"]

Array(3) ["one", "dva", "tri"]

```
var brojevi1 = new Array()
brojevi1[0] = "one"
brojevi1[1] = "dva"
brojevi1[2] = "tri"
console.log(brojevi1)
```

Array(3) ["one", "dva", "tri"]

```
// da li sadrži element?
console.log(brojevi1.includes("jedan") + ", " + brojevi1.includes("one")) false, true
```

```
brojevi1[0] = "jedan" // zamena
brojevi1.push("četiri") // dodavanje na kraj
console.log(brojevi1)
brojevi1.pop() // uklanjanje sa kraja
console.log(brojevi1)
// izdvajanje elemenata u opsegu; vraća modifikovani niz
brojevi1 = brojevi1.slice(1, 3)
console.log(brojevi1)
```

Array(4) ["jedan", "dva", "tri", "četiri"]

Array(3) ["jedan", "dva", "tri"]

Array ["dva", "tri"]

JavaScript objekti

Array

- iteracija

```
var brojevi = ["jedan", "dva", "tri"]
for (let it in brojevi) {
  console.log("it: " + it + ", broj: " + brojevi[it])
}
```

```
it: 0, broj: jedan
it: 1, broj: dva
it: 2, broj: tri
```

JavaScript objekti

Map

```
var proizvodi = new Map()
proizvodi.set("0001", "proizvod1")
proizvodi.set("0002", "proizvod2")
console.log(proizvodi)

// da li sadrži ključ?
console.log(
    proizvodi.has("0001") + ", " +
    proizvodi.has("0003")
)
console.log(proizvodi.get("0001"))

// iteracija po vrednostima
for (let itProizvod of proizvodi.values()) {
    console.log(itProizvod)
}
// iteracija po parovima (ključ, vrednost)
for (let [itSifra, itProizvod] of proizvodi.entries()) {
    console.log(itSifra + ": " + itProizvod)
}

proizvodi.delete("0001")
console.log(proizvodi)
```

Map { 0001 → "proizvod1", 0002 → "proizvod2" }

true, false
proizvod1

proizvod1
proizvod2

0001: proizvod1
0002: proizvod2

Map { 0002 → "proizvod2" }

JavaScript objekti

Date

```
// ISO datum i vreme  
var datumIVreme = new Date("2020-01-01 12:00")  
console.log(datumIVreme)  
var datumIVreme = new Date(Date.now())  
console.log(datumIVreme)
```

Date Wed Jan 01 2020 12:00:00 GMT+0100 (Central European Standard Time)

Date Wed Jun 24 2020 13:11:06 GMT+0200 (Central European Summer Time)

```
// ISO datum i vreme  
console.log(datumIVreme.toISOString())  
// ISO datum  
console.log(datumIVreme.toISOString().split("T")[0])  
// ISO vreme  
console.log(datumIVreme.toISOString().split("T")[1])
```

2020-06-24T11:11:06.063Z

2020-06-24

11:11:06.063Z

JavaScript objekti

Math

<code>console.log(Math.PI)</code>	<code>3.1415926535897939</code>
<code>console.log(Math.pow(3, 2)) // stepenovanje</code>	<code>9</code>
<code>console.log(Math.sqrt(9)) // kvadratni koren</code>	<code>3</code>
<code>console.log(Math.round(1.5)) // zaokruživanje</code>	<code>2</code>
<code>console.log(Math.abs(-1)) // apsolutna vrednost</code>	<code>1</code>
<code>console.log(Math.min(9, 3, 1)) // minimum</code>	<code>1</code>
<code>console.log(Math.max(9, 3, 1)) // maksimum</code>	<code>9</code>
<code>// uniformni pseudoslučajni broj od 0 do 1</code>	
<code>console.log(Math.random())</code>	<code>0.050310997630138354</code>
<code>// uniformni pseudoslučajni broj skaliran na potreban opseg</code>	
<code>console.log(Math.random() * 10) // (0 do 10)</code>	<code>3.264264090401622</code>
<code>// uniformni pseudoslučajni broj sa negativnim vrednostima</code>	
<code>console.log(Math.random() * 20 - 10) // (-10 do 10)</code>	<code>1.524270397187804</code>

JavaScript objekti

- *JavaScript* podržava i korisnički kreirane objekte
- sintaksa koja ovo podržava se zove **JSON** (*Java Script Object Notation*)
- JSON predstavlja standardni tekstualni format kojim se može zapisati proizvoljna struktura podataka

```
{
  korisnici: [
    {korisnickoIme: "a", lozinka: "a", eMail: "a@a.com", pol: "muški", administrator: true},
    {korisnickoIme: "b", lozinka: "b", eMail: "b@b.com", pol: "ženski", administrator: false},
    {korisnickoIme: "c", lozinka: "c", eMail: null, pol: "muški", administrator: false}
  ]
}
```

JavaScript objekti

Kreiranje

- **definicija** JavaScript objekta se navodi između znakova { i }
- navode se parovi **nazivAtributa: vrednostAtributa**
- vrednost atributa može biti primitivnog tipa ili referenca na objekat
- ako atributa ima više, **parovi se odvajaju** znakom ,

```
var film = {  
  id: 1,  
  naziv: "Avengers: Endgame",  
  trajanje: 182  
}
```

```
console.log(film)
```

```
Object { id: 1, naziv: "Avengers: Endgame", trajanje: 182 }
```

```
console.log(JSON.stringify(film)) {"id":1,"naziv":"Avengers: Endgame","trajanje":182}
```

JavaScript objekti

Čitanje i izmena

- atributima (*properties*) se pristupa po nazivu
- izmena se vrši operatorima dodele

```
console.log("id: " + film.id)           id: 1
console.log("id: " + film["id"])        id: 1
console.log("naziv: " + film.naziv)     naziv: Avengers: Endgame
console.log("trajanje: " + film.trajanje) trajanje: 182
```

```
film.id = "2"
film.naziv = "Life"
film.trajanje = 110
console.log(film)
```

```
Object { id: "2", naziv: "Life", trajanje: 110 }
```

JavaScript objekti

Dodavanje i uklanjanje atributa

- atributi se mogu dodati u već kreirani objekat
- atributi se nakon dodavanja mogu ukloniti operatorom **delete**

```
film.zanr = "horor"  
console.log(film)
```

```
Object { id: "2", naziv: "Life", trajanje: 110, zanr: "horor" }
```

```
delete film.zanr  
console.log(film)
```

```
Object { id: "2", naziv: "Life", trajanje: 110 }
```

JavaScript objekti

Reference na objekte

- atributi mogu biti i reference na objekte i čak i kolekcije referenci

```
film.zanr = {id: 4, naziv: "horor"}  
console.log(film)
```

```
{...}
```

```
id: "2"  
naziv: "Life"  
trajanje: 110  
zanr: Object { id: 4, naziv: "horor" }
```

```
film.zanrovi = [  
  {id: 1, naziv: "naučna fantastika"},  
  {id: 4, naziv: "horor"}  
]  
console.log(film)
```

```
{...}
```

```
id: "2"  
naziv: "Life"  
trajanje: 110  
zanrovi: (2) [...]  
  0: Object { id: 1, naziv: "naučna fantastika" }  
  1: Object { id: 4, naziv: "horor" }
```

JavaScript objekti

Metode

- atributi mogu biti i funkcije (metode); i one se mogu dinamički dodavati i uklanjati

```
var film = {  
  // atributi  
  id: 1,  
  naziv: "Avengers: Endgame",  
  trajanje: 182,  
  // metode  
  uvecajTrajanje: function(zaKoliko) {  
    this.trajanje += zaKoliko  
  },  
  toString: function() {  
    return "id: " + this.id + ", naziv: " + this.naziv + ", trajanje: " + this.trajanje  
  }  
}  
film.uvecajTrajanje(8)  
console.log(film.toString())
```

← paziti na , iza funkcije

id: 1, naziv: Avengers: Endgame, trajanje: 190

JavaScript objekti

Getter-i i setter-i

```
var film = {  
  // atributi  
  id: 1,  
  naziv: "Avengers: Endgame",  
  trajanje: 182,  
  // metode  
  get tr() {  
    return this.trajanje  
  },  
  set tr(trajanje) {  
    if (trajanje <= 5) {  
      trajanje = 5  
    }  
    this.trajanje = trajanje  
  }  
}
```

```
film.tr = -5  
console.log(film.tr)
```

JavaScript objekti

Konstruktori

- *getter-i setter-i* se ne mogu dodati na ovaj način; potrebne su klase

```
function Film(id, naziv, trajanje) {  
  // atributi  
  this.id = id  
  this.naziv = naziv  
  this.trajanje = trajanje  
  // metode  
  this.uvecajTrajanje = function(zaKoliko) {  
    this.trajanje += zaKoliko  
  }  
  this.toString = function() {  
    return "id: " + this.id + ", naziv: " + this.naziv + ", trajanje: " + this.trajanje  
  }  
}  
var film = new Film(1, "Avengers: Endgame", 182)
```

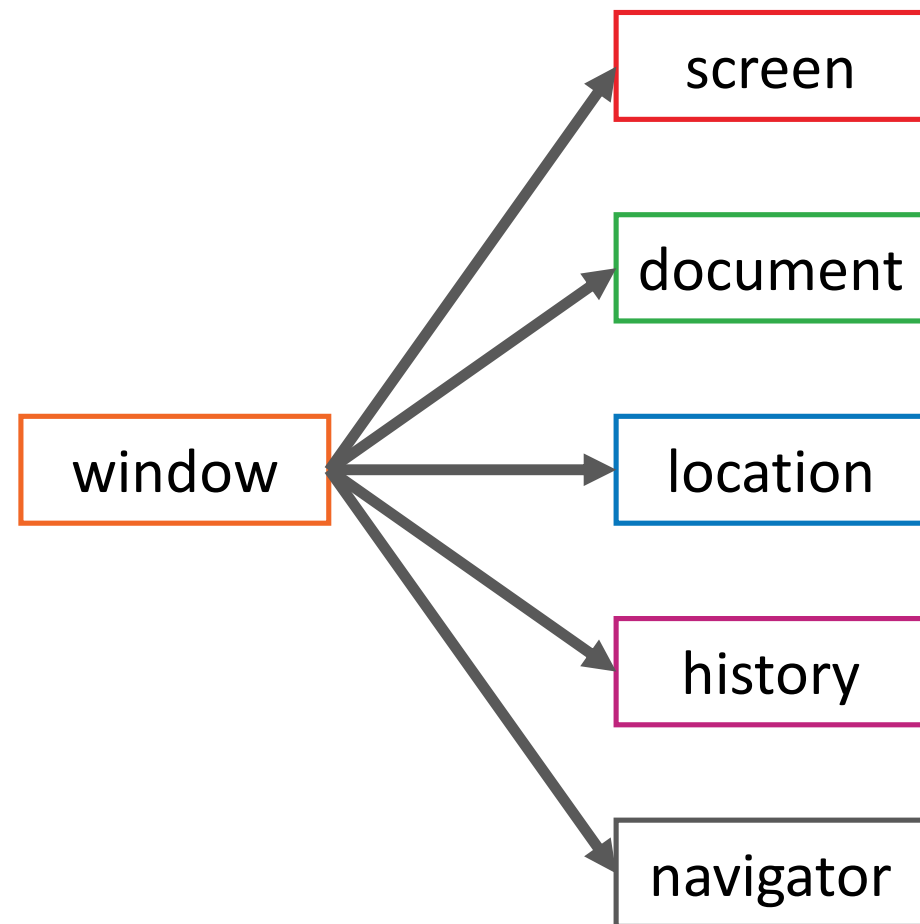
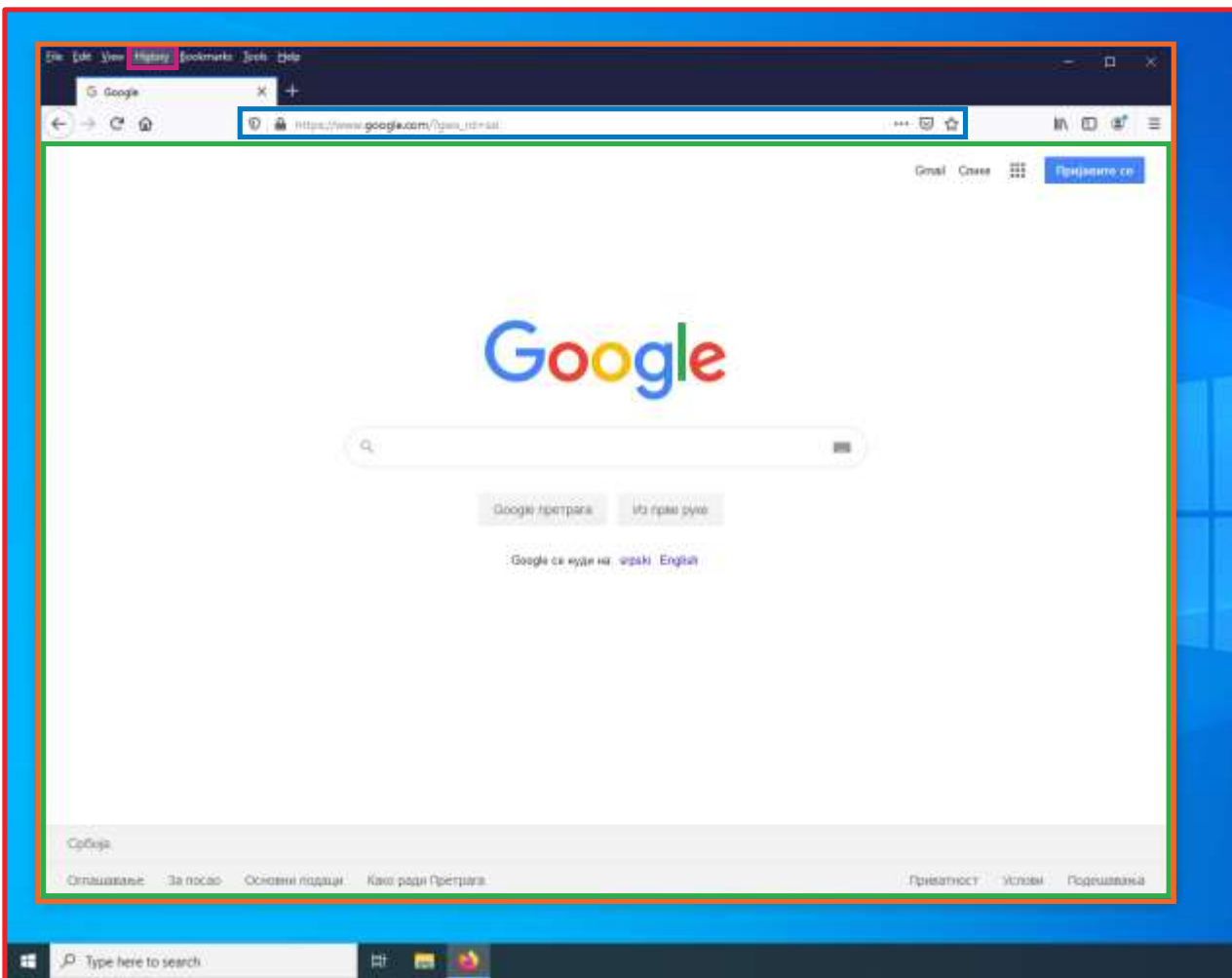
```
film.uvecajTrajanje(8)  
console.log(film.toString())
```

id: 1, naziv: Avengers: Endgame, trajanje: 190

Dodatno BOM

BOM (*Browser Object Model*)

- objektni model prozora *web browser-a* koji omogućuje programsko rukovanje njegovim elementima



BOM (*Browser Object Model*)

Window

	Naziv	Upotreba
metode	<i>alert(...), confirm(...), prompt(...)</i>	prikaz poruka i pitanja korisniku
	<i>back()</i> <i>forward()</i>	povratak na prethodnu stranicu odlazak na sledeću stranicu
	<i>moveBy(...)</i> <i>moveTo(...)</i>	pomeranje prozora
	<i>open(...)</i>	otvara novi prozor sa zadatom adresom
	<i>setTimeout("izraz", timeout)</i>	zadavanje <i>JavaScript</i> izraza koji će se izvršiti nakon određenog vremena
	<i>setInterval("izraz", interval)</i>	zadavanje <i>JavaScript</i> izraza koji će se izvršavati periodično
	i mnoge druge...	
atributi	<i>screen, document, location, history, navigator</i>	pristup podelementima prozora
	<i>screenX, screenY</i>	pozicija prozora na ekranu
	<i>outerWidth, outerHeight</i>	dimenzije prozora
	i mnogi drugi...	

BOM (*Browser Object Model*)

Location



http://localhost:8080/Bioskop/Zanrovi?naziv=a

	Naziv	Upotreba
metode	<i>reload()</i> <i>replace(...)</i>	ponovo učitava tekuću adresu učitava novu adresu
atributi	<i>href</i>	kompletan URL
	<i>protocol</i>	protokol
	<i>host</i>	adresa servera
	<i>port</i>	port
	<i>pathname</i>	putanja do resursa
	<i>search</i>	parametri

BOM (*Browser Object Model*)

History

	Naziv	Upotreba
metode	<i>back()</i>	povratak na prethodnu stranicu
	<i>forward()</i>	odlazak na sledeću stranicu
	<i>go(...)</i>	odlazak na zadatu stranicu iz liste
atributi	<i>current</i>	tekuća stranica
	<i>length</i>	ukupan broj zapamćenih posećenih stranica
	<i>next</i>	sledeća stranica
	<i>previous</i>	prethodna stranica

BOM (*Browser Object Model*)

Navigator

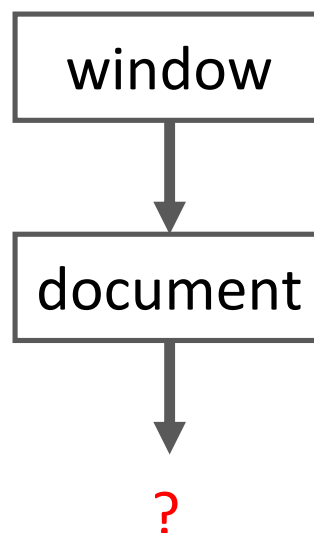
- opisuje *web browser* koji korisnik koristi da bi pristupio stranici

	Naziv	Upotreba
atributi	<i>appName</i>	naziv <i>web browser</i> -a
	<i>appVersion</i>	verzija <i>web browser</i> -a
	<i>cookieEnabled</i>	da li browser podržava <i>cookies</i> ?
	<i>language</i>	jezik <i>web browser</i> -a
	<i>platform</i>	operativni system na kome je pokrenut <i>web browser</i>
	i mnogi drugi...	

Dodatno DOM

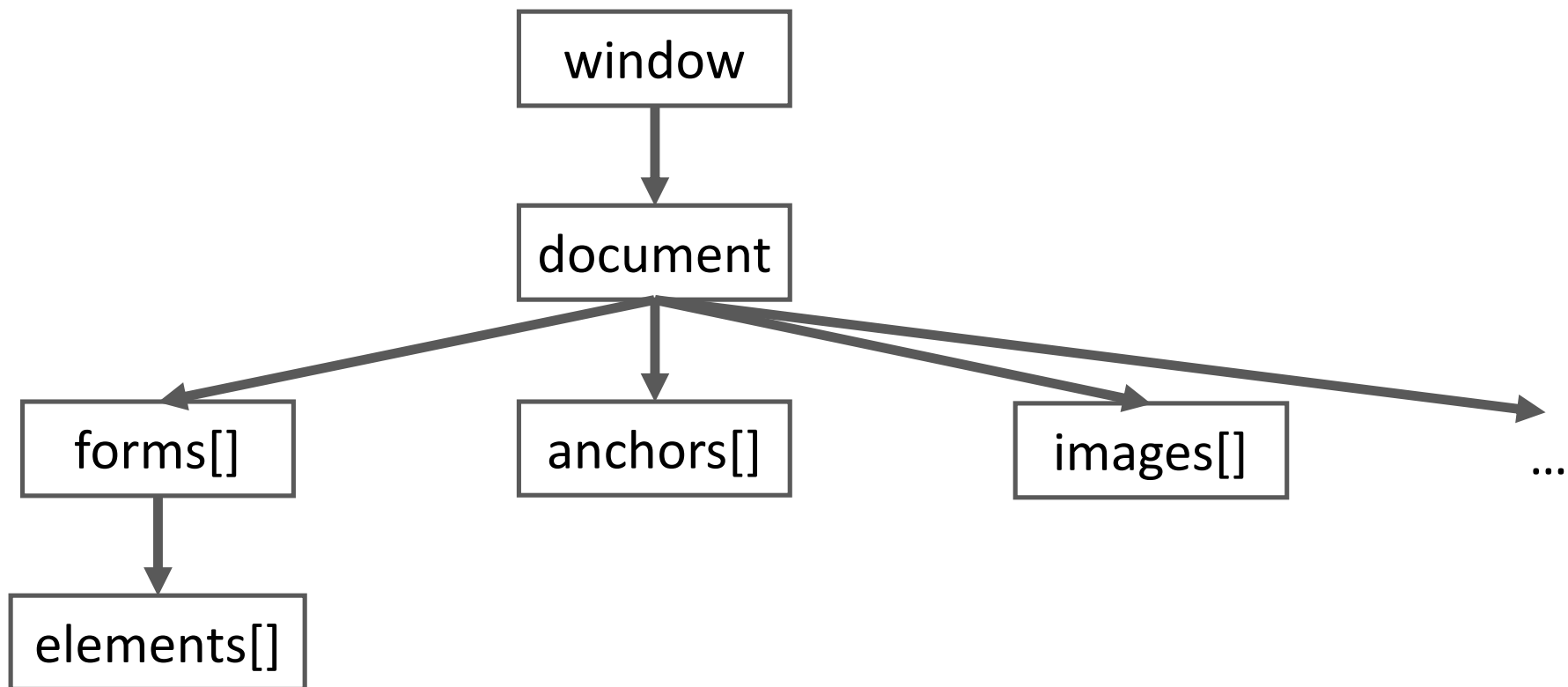
DOM (*Document Object Model*)

- dalje opisuje **objektni model HTML stranice** uz pomoć kog mogu da se čitaju i menjaju njeni elementi
- objektni model je organizovan u **stablo**
- ranije se DOM standard organizovao u generacije (Level 0, Level 1, ...), ali to više nije slučaj



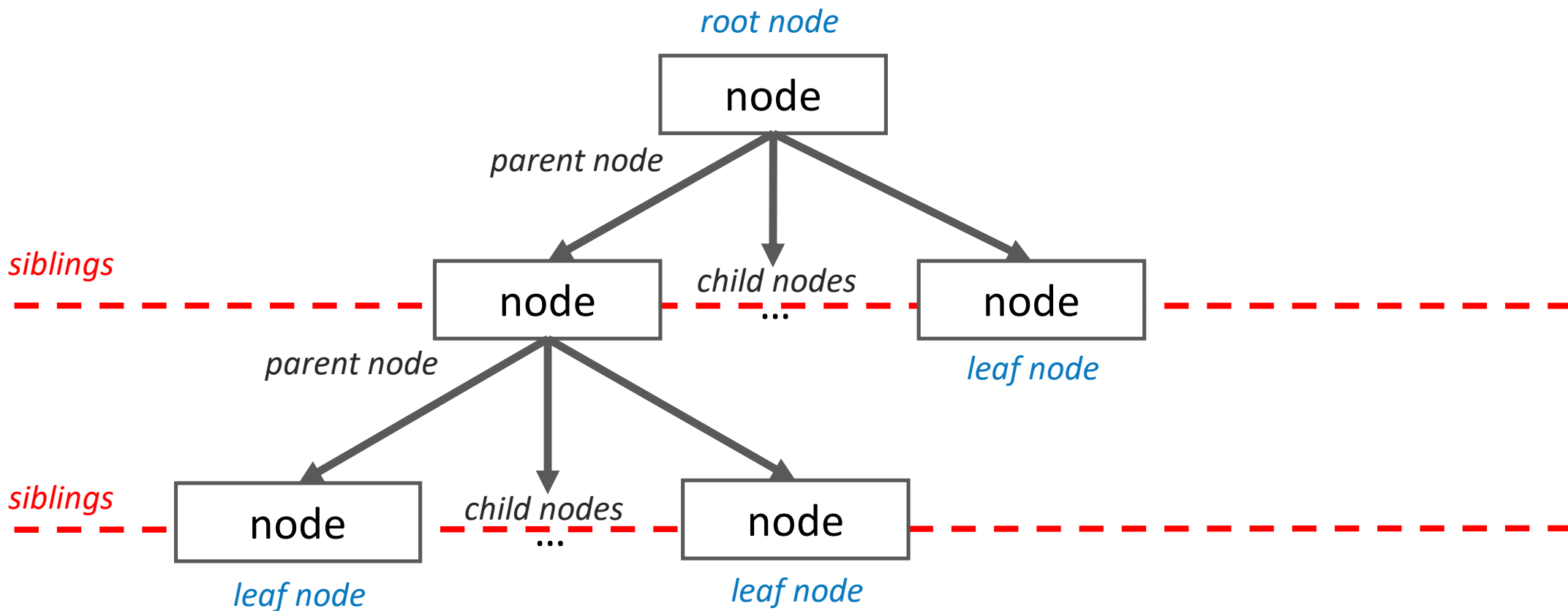
DOM (*Document Object Model*) Level 0

- opisuje predefinisane nizove objekata koji se mogu ili ne moraju javiti na HTML stranici
- nizovi uvek postoje, makar bili i prazni



DOM (*Document Object Model*) Level 1 - 4

- opisuje objektni model proizvoljne HTML stranice u odnosu na njen korenski element



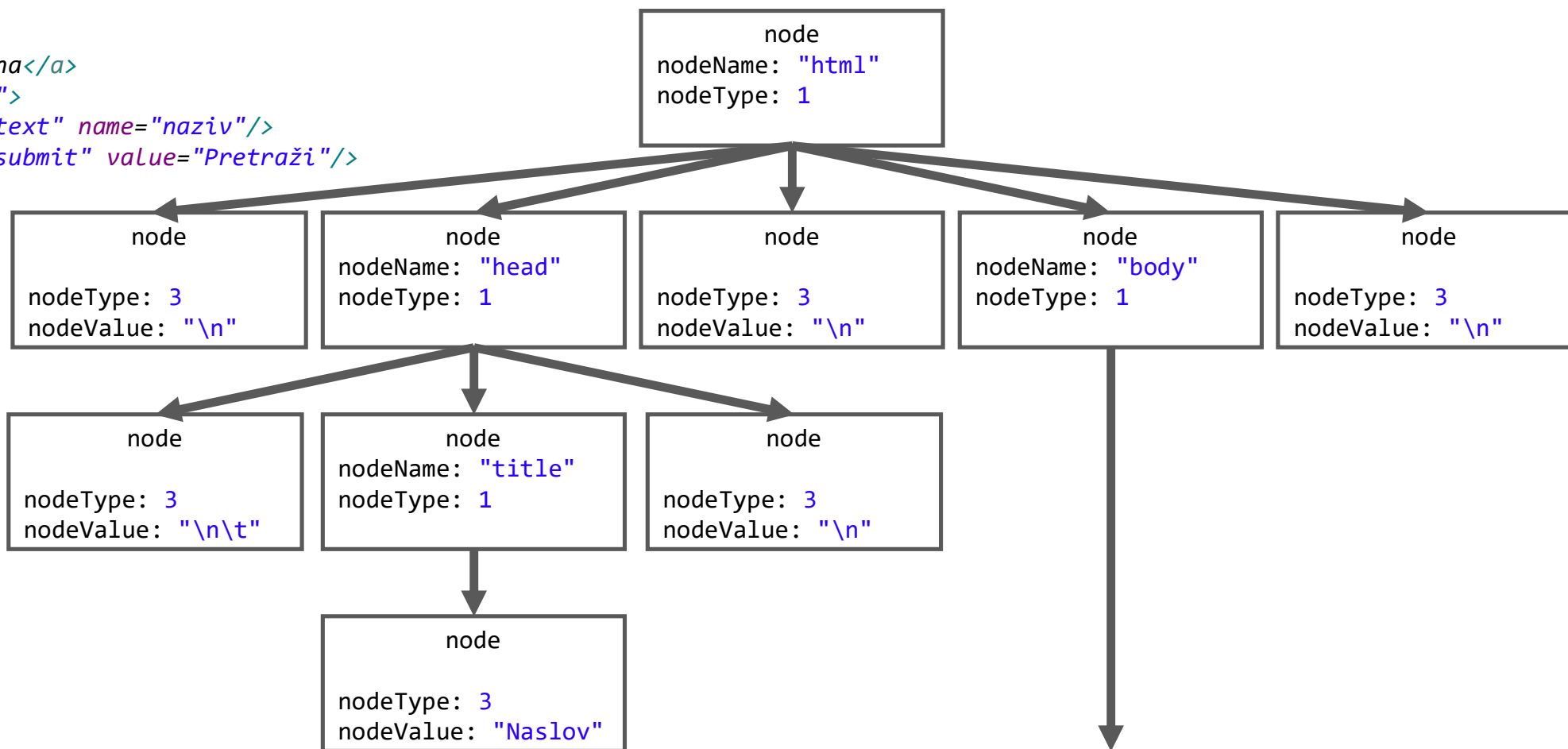
DOM (*Document Object Model*) Level 1 - 4

- opisuje objektni model proizvoljne HTML stranice u odnosu na njen korenski element
- svaki element (čvor) stabla je opisan objektom tipa *node*
- čvor na vrhu stabla se zove korenski čvor (*root node*)
- svaki čvor ima referencu na jedan roditeljski čvor (*parent node*); referenca je prazna za korenski čvor
- svaki čvor ima niz referenci na čvorove potomke (*child nodes*); niz može biti prazan i tada se čvor naziva list
- čvorovi potomci istog roditeljskog čvora se nazivaju *siblings*

DOM (*Document Object Model*) Level 1 - 4

```
<!DOCTYPE html>
<html>
<head>
  <title>Naslov</title>
```

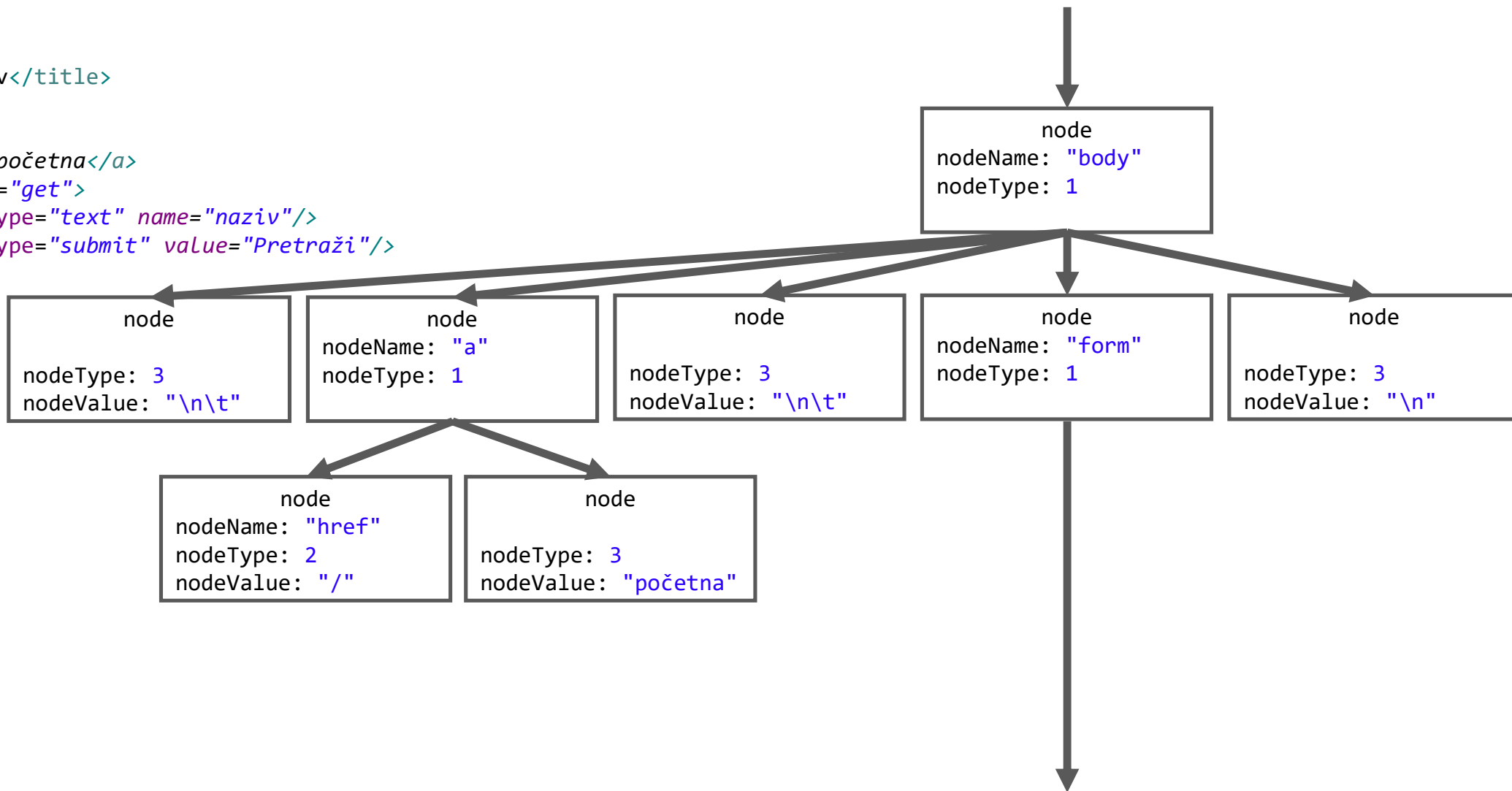
```
</head>
<body>
  <a href="/">početna</a>
  <form method="get">
    <input type="text" name="naziv"/>
    <input type="submit" value="Pretraži"/>
  </form>
</body>
</html>
```



DOM (*Document Object Model*) Level 1 - 4

```
<!DOCTYPE html>  
<html>  
<head>  
  <title>Naslov</title>
```

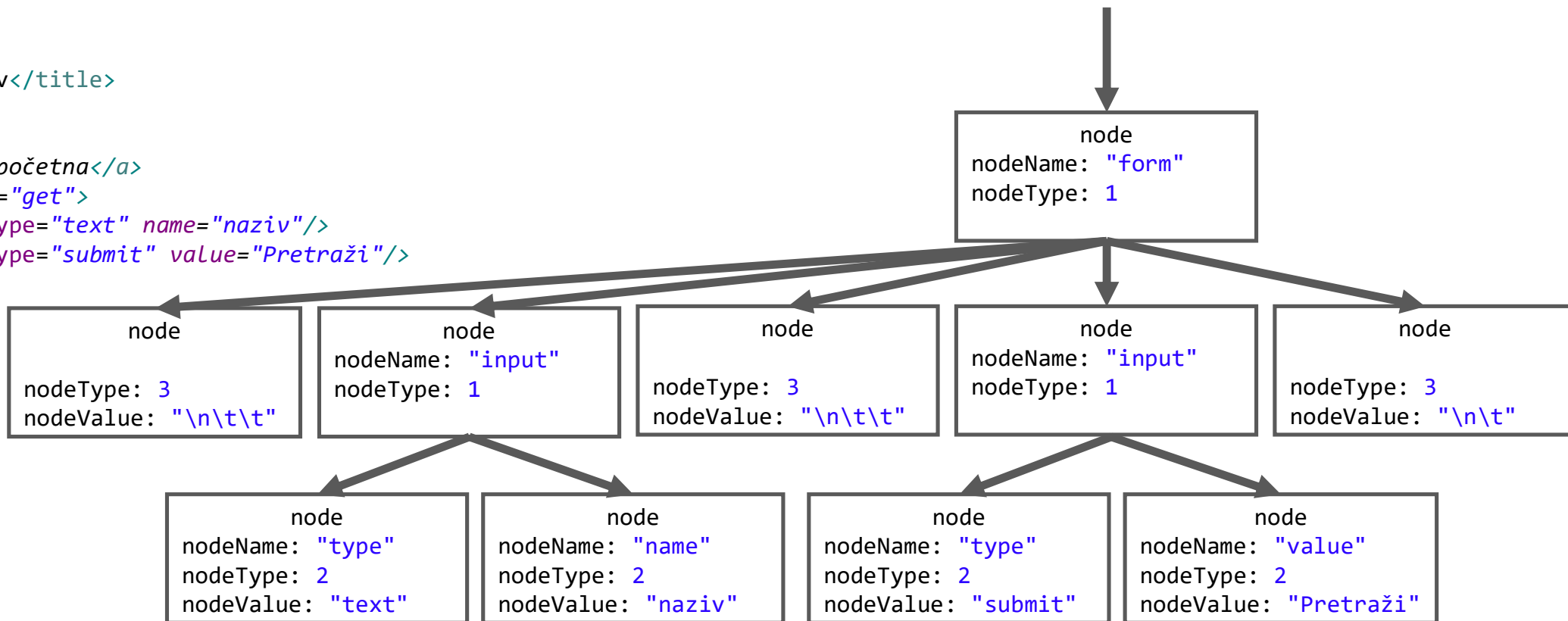
```
</head>  
<body>  
  <a href="/">početna</a>  
  <form method="get">  
    <input type="text" name="naziv"/>  
    <input type="submit" value="Pretraži"/>  
  </form>  
</body>  
</html>
```



DOM (*Document Object Model*) Level 1 - 4

```
<!DOCTYPE html>
<html>
<head>
  <title>Naslov</title>
</head>
```

```
<body>
  <a href="/">početna</a>
  <form method="get">
    <input type="text" name="naziv"/>
    <input type="submit" value="Pretraži"/>
  </form>
</body>
</html>
```



DOM (*Document Object Model*) Level 1 - 4

Node

- opisuje jedan čvor u stablu

	Naziv	Upotreba
atributi	<i>nodeName</i>	naziv elementa ili atributa
	<i>nodeType</i>	tip čvora (1 za HTML tagove, 2 za attribute, 3 za tekstualne čvorove, 8 za komentar, 9 za dokument)
	<i>nodeValue</i>	sadržaj tekstualnog čvora ili vrednost atributa
	<i>innerHTML</i>	kompletan HTML kod podstabla čvora
	<i>id</i>	vrednost <i>id</i> atributa (ako se navede)
	<i>className</i>	vrednost <i>class</i> atributa (ako se navede)
	<i>style</i>	referenca na style objekat
	<i>childNodes</i>	lista čvorova potomaka
	<i>firstChild</i> <i>lastChild</i>	prvi čvor potomak poslednji čvor potomak
	<i>parentNode</i>	roditeljski čvor
	<i>previousSibling</i> <i>nextSibling</i>	prethodni čvor na istom nivou stabla sledeći čvor na istom nivou stabla
	i mnogi drugi...	

DOM (*Document Object Model*) Level 1 - 4

Node

- opisuje jedan čvor u stablu

	Naziv	Upotreba
metode	<i>appendChild(node)</i>	dodaje novi čvor na kraj liste čvorova potomaka pozivajućeg čvora
	<i>insertBefore(referenceNode, insertedNode)</i>	umeće čvor u listu čvorova potomaka pre drugog čvora u podstablu pozivajućeg čvora
	<i>removeChild(node)</i>	uklanja čvor iz liste čvorova potomaka pozivajućeg čvora
	<i>getAttribute(attributeName)</i>	direktno čita vrednost atributa čvora (iako je atribut čvor potomak pozivajućeg čvora)
	<i>setAttribute(attributeName, attributeValue)</i>	direktno postavlja novu vrednost atributa čvora (iako je atribut čvor potomak pozivajućeg čvora)
	<i>removeAttribute(attributeName)</i>	direktno uklanja atribut čvora (iako je atribut čvor potomak pozivajućeg čvora)
	<i>hasAttributes()</i>	vraća <i>true</i> ako pozivajući čvor u svom podstablu ima attribute
	i mnoge druge...	

DOM (*Document Object Model*) Level 1 - 4

Document

- opisuje HTML dokument

	Naziv	Upotreba
metode	<i>write(...)</i>	dopisuje sadržaj na kraj HTML dokumenta za vreme učitavanja stranice
	<i>getElementById(id)</i> <i>getElementsByName(name)</i> <i>getElementsByClassName(name)</i> <i>getElementsByTagName(name)</i>	pronalazi element po vrednosti <i>id</i> atributa vraća niz elementa pronađenih po vrednosti <i>name</i> atributa vraća niz elementa pronađenih po vrednosti <i>class</i> atributa vraća niz elementa pronađenih po nazivu <i>tag</i> -a
	<i>createElement(...)</i> <i>createTextNode(...)</i> <i>createAttribute(...)</i>	kreira čvor koji predstavlja element kreira tekstualni čvor kreira čvor koji predstavlja atribut
	i mnoge druge...	
atributi	<i>head, body</i>	referenca na čvor <i>head</i> , odnosno <i>body</i> elementa
	<i>forms, anchors, images</i>	reference na nizove definisane u DOM Level 0
	<i>title</i>	naslov dokumenta
	<i>URL</i>	URL dokumenta
	i mnogi drugi...	

DOM (*Document Object Model*) Level 1 - 4

Form

- opisuje formu

	Naziv	Upotreba
metode	<i>reset()</i>	<i>reset</i> -uje sva polja forme
	<i>submit()</i>	programski klik na <i>submit</i> dugme
atributi	<i>method</i>	vrednost <i>method</i> atributa
	<i>action</i>	vrednost <i>action</i> atributa
	<i>name</i>	vrednost <i>name</i> atributa
	<i>length</i>	ukupan broj <i>input</i> elemenata u formi
	<i>elements</i>	niz <i>input</i> elemenata u formi
	i mnogi drugi...	

DOM (*Document Object Model*) Level 1 - 4

Input

- opisuje *input* element forme

	Naziv	Upotreba
atributi	<i>value</i>	vrednost unosa
	<i>defaultValue</i>	početna vrednost
	<i>type</i>	vrednost type atributa
	<i>name</i>	vrednost name atributa
	<i>form</i>	referenca na formu kojoj <i>input</i> pripada
	i mnogi drugi...	

DOM (*Document Object Model*) Level 1 - 4

Metoda *document.write(...)*

- ako se pozove za vreme učitavanja stranice dopisuje sadržaj na kraj HTML dokumenta
- ako se pozove nakon učitavanja stranice, zamenjuje kompletan sadržaj HTML dokumenta

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Funkcija write</title>
  <script type="text/javascript">
    function generisiPasus(tekst) {
      document.write("<p>" + tekst + "</p>")
    }
  </script>
</head>
<body>
  <p>običan pasus</p>
  <script type="text/javascript">
    generisiPasus("pasus generisan za vreme učitavanja")
  </script>
  <p>običan pasus</p>
</body>
```

običan pasus

pasus generisan za vreme učitavanja

običan pasus

DOM (*Document Object Model*) Level 1 - 4

Style

- objekat uz pomoć kog se upravlja CSS atributima čvora

	Naziv	CSS atribut
atributi	<i>display</i>	<i>display</i>
	<i>color</i>	<i>color</i>
	<i>backgroundColor</i>	<i>background-color</i>
	<i>width</i>	<i>width</i>
	<i>borderStyle</i>	<i>border-style</i>
	<i>borderColor</i>	<i>border-color</i>
	<i>borderWidth</i>	<i>border-width</i>
	i mnogi drugi...	

Dodatni materijali

- <https://www.w3schools.com/js/>
- https://www.w3schools.com/js/js_object_definition.asp