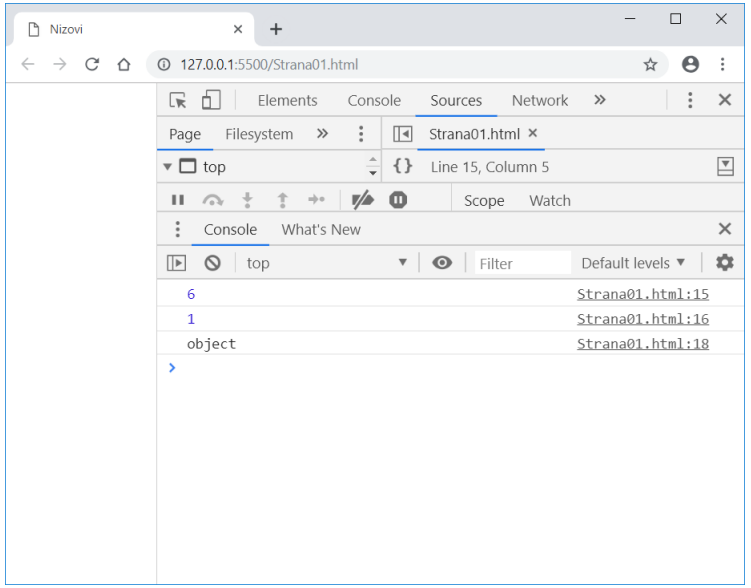
Nizovi i objektu u JavaScript-u

Nizovi

- JavaScript nizovi se koriste za čuvanje više vrednosti u jednoj promenljivoj
- Članovima niza pristupa se na osnovu indeksa koji je baziran na 0
- Nizovi su specijalni tipovi objekata

```
<script>
    var x = [1, 3, 5, 7, 4, 8];
    var brojClanova = x.length;
    var x0 = x[0];

    console.log(brojClanova);
    console.log(x0);

    console.log(typeof(x));
 </script>
```

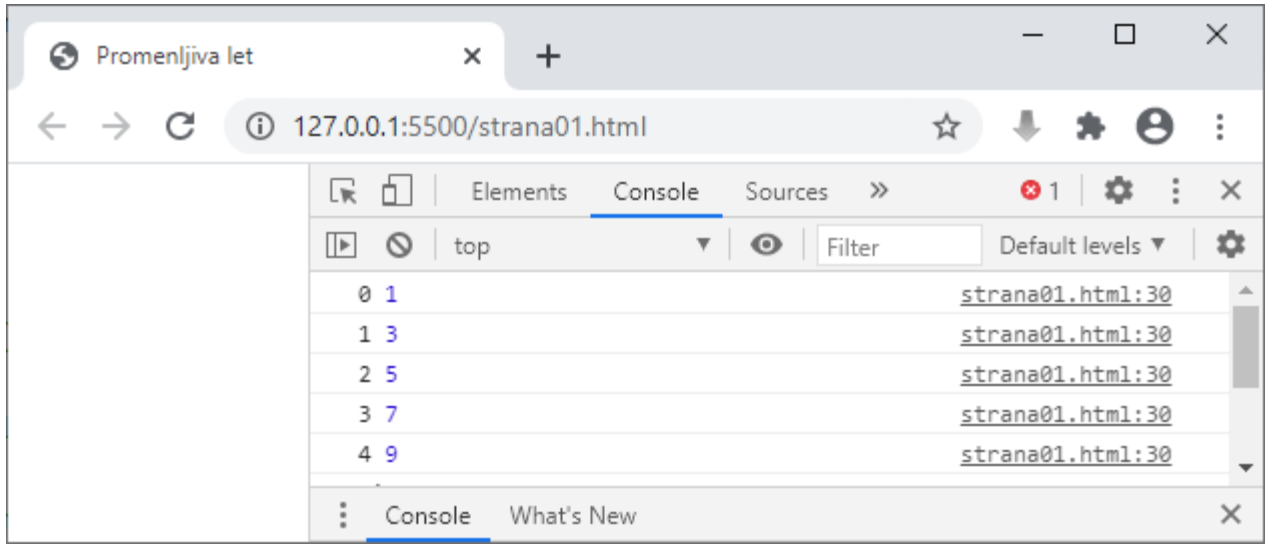


Kreiranje inicijalizovanog niza i prolazak for petljom

```
<script>
    var x = [1, 3, 5, 7, 9];
        for (let i = 0; i < x.length; i++) {
            console.log(x[i]);
        }
</script>
```

Prolazak kroz niz forin petljom

```
<script>
    var x = [1, 3, 5, 7, 9];
        //fin
       for (const i in x) {
           console.log(i, x[i]);
       }
</script>
```

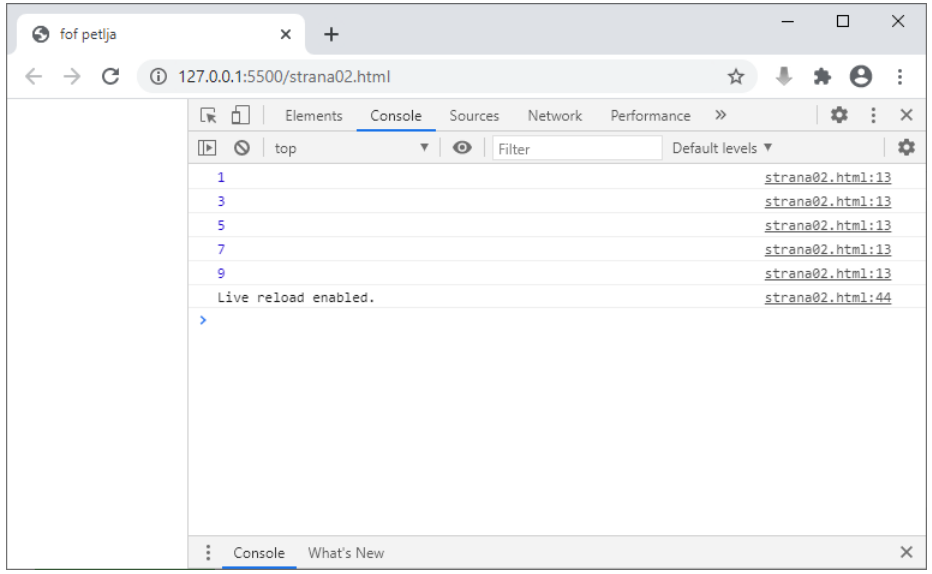


fin code snnipet
Javascript (ES6) code snnipets

i je tipa string!

Prolazak kroz niz forof petljom

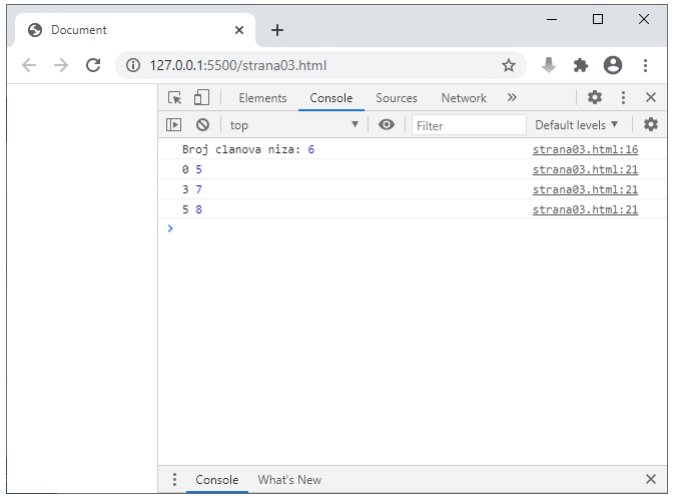
```
<script>
    var x = [1, 3, 5, 7, 9];

    for (const i of x) {
        console.log(i);
    }
</script>
```



fof code snnipet
Javascript (ES6) code snnipets

Kreiranje neinicijalizovanog niza

```
<script>
    //var a = new Array();
    var a = [];
    a[0] = 5;
    a[3] = 7;
    a[5] = 8;

    console.log("Broj clanova niza:", a.length);

    //fin
    for (const i in a) {
        console.log(i, a[i]);
    }
</script>
```

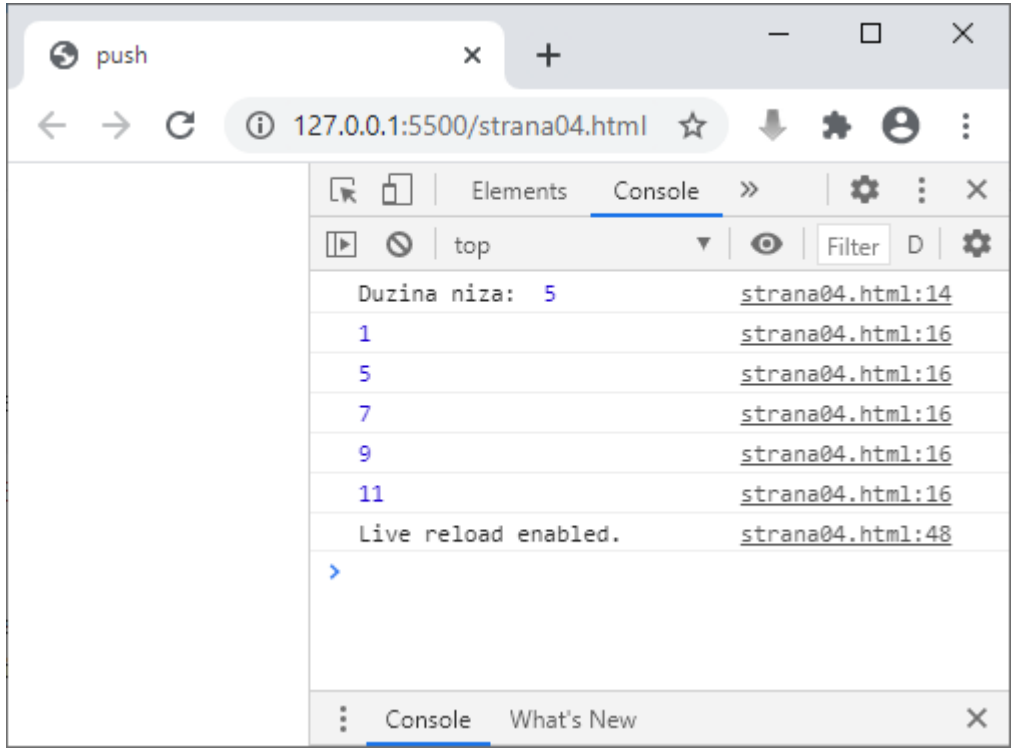


a[1] = undefined

Dodavanje elementa na kraj niza – funkcija push

Funkcija push() dodaje element na kraj niza i vraća broj članova novog niza.

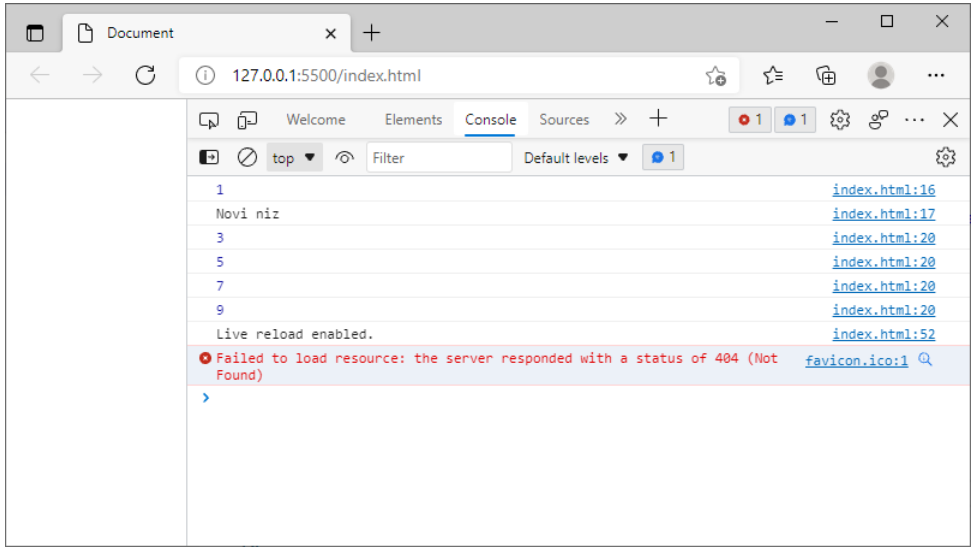
```
<script>
    var a = [1, 5, 7, 9];

    var duzina = a.push(11);

    console.log("Duzina niza: ", duzina);
    for (var i in a) {
        console.log(a[i]);
    }
</script>
```



Funkcija shift()

Uklanja element sa početka niza i vraća taj element.

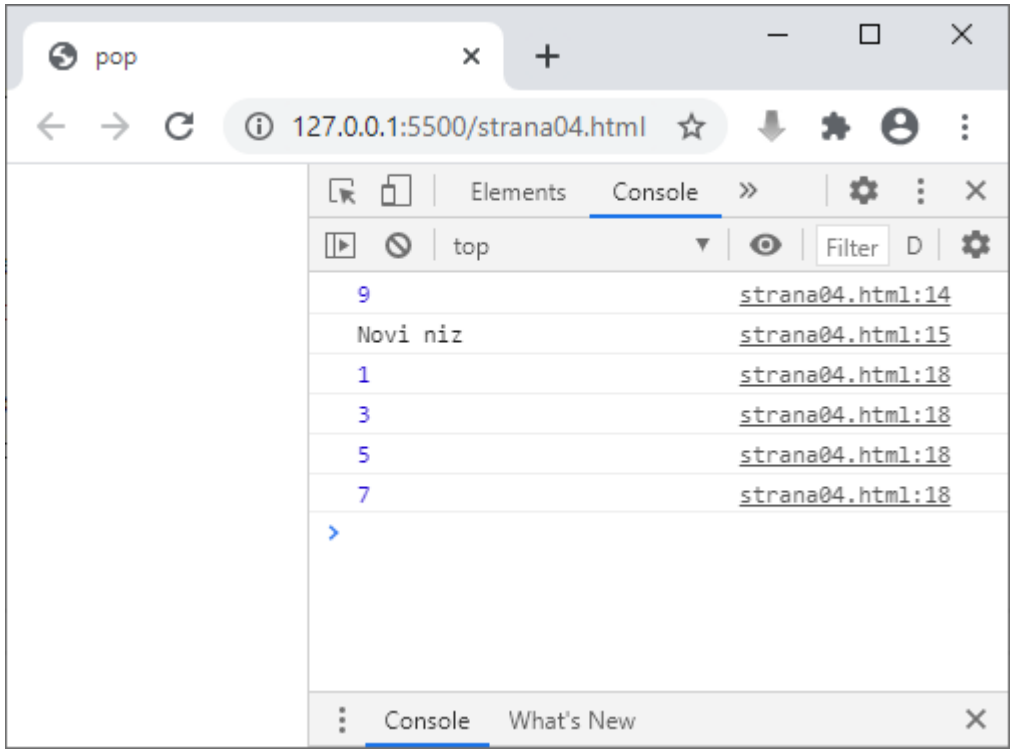
```
<script>

    var x = [1, 3, 5, 7, 9];

    var prvi = x.shift();

    console.log(prvi);
    console.log('Novi niz');

    for (const i of x) {
        console.log(i);
    }

</script>
```



Funkcija pop()

Uklanja element sa kraja niza i vraća taj element.

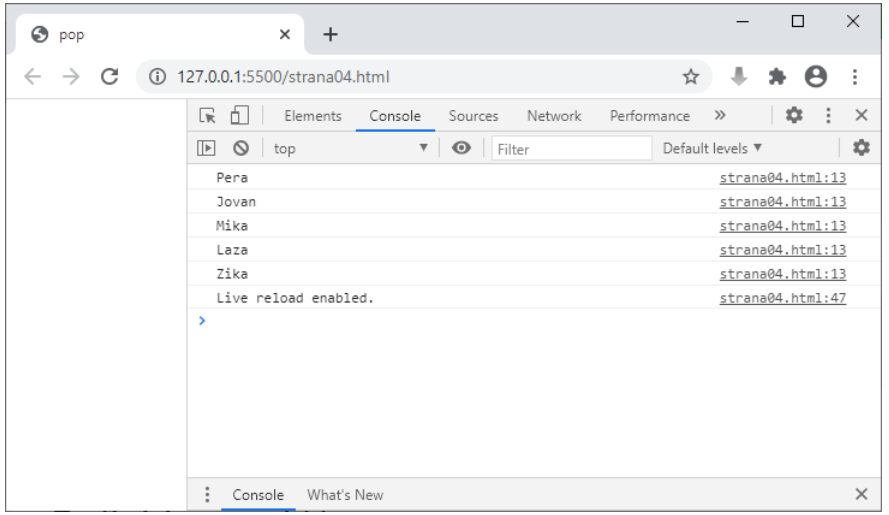
```
<script>
    var x = [1, 3, 5, 7, 9];

    var poslednji =   x.pop();

    console.log(poslednji);
    console.log('Novi niz');

    for (const i of x) {
        console.log(i);
    }
    </script>
```



Funkcija splice()

- Dodaje/uklanja elemente iz niza i vraća uklonjene stavke
- splice(index, brojElemenataZaBrisanje,listaZaDodavanje)

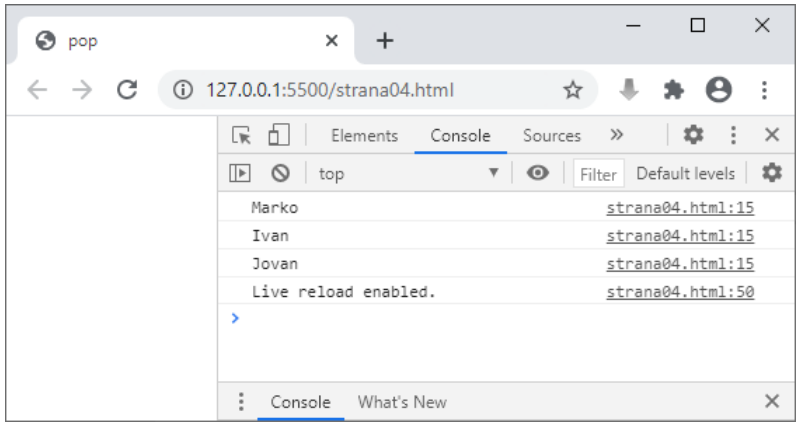
```
<script>
    var imena = ["Pera", "Mika", "Laza", "Zika"];
    imena.splice(1, 0, "Jovan");
    for (const ime of imena) {
        console.log(ime);
    }
</script>
```



Dodavanje na poziciju 1 (drugo mesto u nizu)

Brisanje elementa sa pozicije

```
<script>
    var imena = ["Marko", "Ivan", "Lazar","Jovan"];
    //obrisi sa  pozicije 2
    imena.splice(2,1);

    for (const ime of imena) {
        console.log(ime);
    }
</script>
```

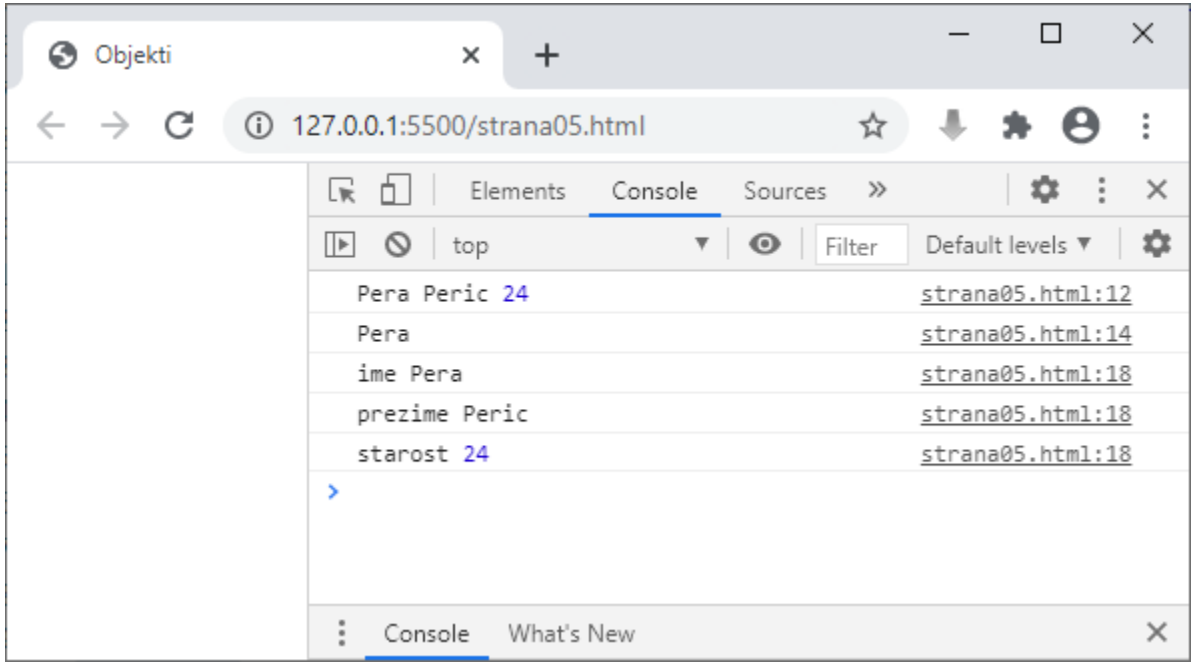


JavaScript objekti

Sadrže imenovane vrednost, tzv. parove (name,value). Parovi se sastoje od svojstva i vrednosti razdvojenih sa :

```
<script>
    var osoba = {ime:"Pera", prezime:"Peric", starost:24 };

    console.log(osoba.ime, osoba.prezime, osoba.starost);

    console.log(osoba["ime"]);

    for (var i in osoba) {
        console.log(i,osoba[i]);
    }
</script>
```

Prikaz sadržaja objekta



Definisanje metode objekta

```
<script>
    var osoba = {
        ime: "Marko",
        prezime: "Petrovic",
        starost: 24,
        PunoIme: function () {
            return this.ime + " " + this.prezime;
        }
    };
    console.log(osoba.PunoIme());
</script>
```

Kreiranje prototipa objekta, konstruktorska funkcija

```
function Osoba(ime, prezime, starost) {
    this.ime = ime;
    this.prezime = prezime;
    this.starost = starost;
}
```

```
var os1 = new Osoba("Marko", "Markovic", 21);
```

Dodavanje funkcije u prototip

```
<script>
    function Osoba(ime, prezime, starost) {
    this.ime = ime;
    this.prezime = prezime;
    this.starost = starost;
    this.Stampaj = function () {
        console.log(this.ime, this.prezime, this.starost);
    };
}

    var os1 = new Osoba("Marko","Markovic",34);
    os1.Stampaj();
</script>
```

Naknadno dodavanje funkcije u prototip

```
<script>
    function Osoba(ime, prezime, starost) {
    this.ime = ime;
    this.prezime = prezime;
    this.starost = starost;
    }
    Osoba.prototype.Stampaj = function () {
        console.log(this.ime, this.prezime, this.starost);
    };

    var os1 = new Osoba("Marko","Markovic",34);
    os1.Stampaj();
</script>
```

JSON

- JavaScript Object Notation
- Browser i web server razmenjuju tekstualne podatke
- JSON je tekst i svaki JavaScript objekat na klijentu se može konvertovati u JSON i poslati do servera
- JSON tekst sa servera koji se šalje klijentu se takođe može konvertovati u JavaScript objekte

Json sintaksa

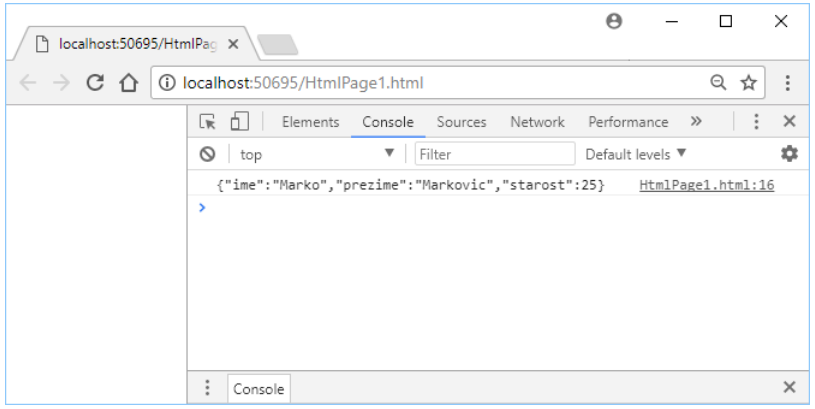
- Podaci su parovi: naziv svojstva i vrednost svojstva
- Naziv svojsta i vrednost svojtva se razdvajaju sa :
- Svi nazivi svojstava se pišu između dvostrukih znakova navoda
- Vrednosti mogu biti:
  - stringovi
  - brojevi
  - logičke vrednosti
  - nizovi
  - objekti
  - null
- Parovi su međusobno razdvojeni zarezom
- Vitičaste zagrade čuvaju objekte
- Uglaste zagrade čuvaju nizove

Tipovi podataka u JSON tekstu

- Stringovi se pišu između dvostrukih navodnika
- Brojevi mogu biti realni ili celi
- Vrednost može biti true ili false
- Vrednost može biti null

JSON.stringfy() funkcija

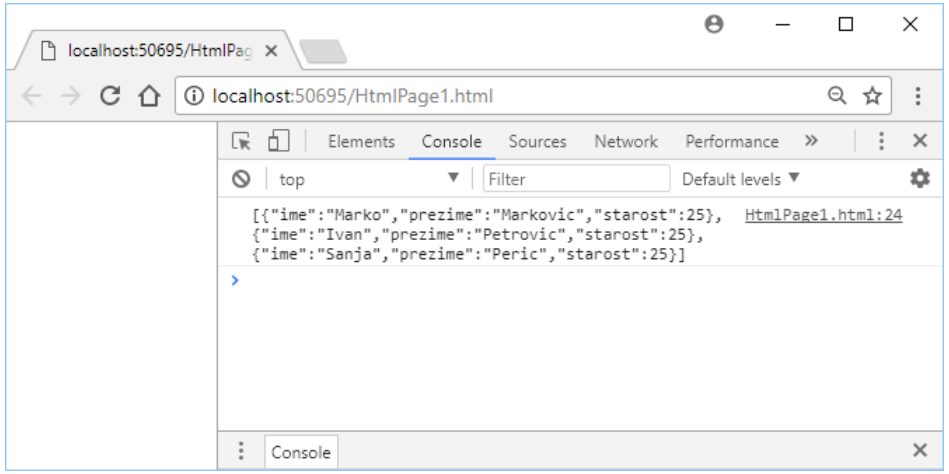
```
<script>
    function Osoba(ime, prezime, starost) {
        this.ime = ime;
        this.prezime = prezime;
        this.starost = starost;
    }
    var osoba = new Osoba("Marko", "Markovic", 25);
    var jsonText = JSON.stringify(osoba);
    console.log(jsonText);
</script>
```



Pretvara javascript objekat ili niz javascript objekata u tekst.
Koristi se pri slanju podataka ka web serveru.

Pretvaranje javascript niza u JSON tekst

```
<script>
function Osoba(ime, prezime, starost) {
    this.ime = ime;
    this.prezime = prezime;
    this.starost = starost;
}
var os1 = new Osoba("Marko", "Markovic", 25);
var os2 = new Osoba("Ivan", "Petrovic", 25);
var os3 = new Osoba("Sanja", "Peric", 25);

var nizOsoba = new Array();
nizOsoba.push(os1);
nizOsoba.push(os2);
nizOsoba.push(os3);

var jsonText = JSON.stringify(nizOsoba);
console.log(jsonText);
</script>
```



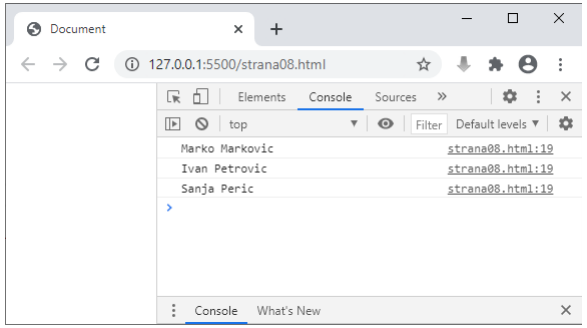
Metoda JSON.parse()-1

Pretvara teks obično dobijen od web servera u javascript objekat ili niz javascript objekata

```
<script>
    var jsonString = `{
            "ime":"Marko",
            "prezime":"Markovic",
            "starost":25
            }`;
    var osoba = JSON.parse(jsonString);
    console.log(osoba.ime, osoba.prezime);

</script>
```

Metoda JSON.parse()-2

```
<script>
    var jsonString = `[
        {"ime":"Marko","prezime":"Markovic","starost":25},
        {"ime":"Ivan","prezime":"Petrovic","starost":25},
        {"ime":"Sanja","prezime":"Peric","starost":25}
        ]`;

    var osobe = JSON.parse(jsonString);

    for (var i in osobe) {
        console.log(osobe[i].ime, osobe[i].prezime);
    }
</script>
```

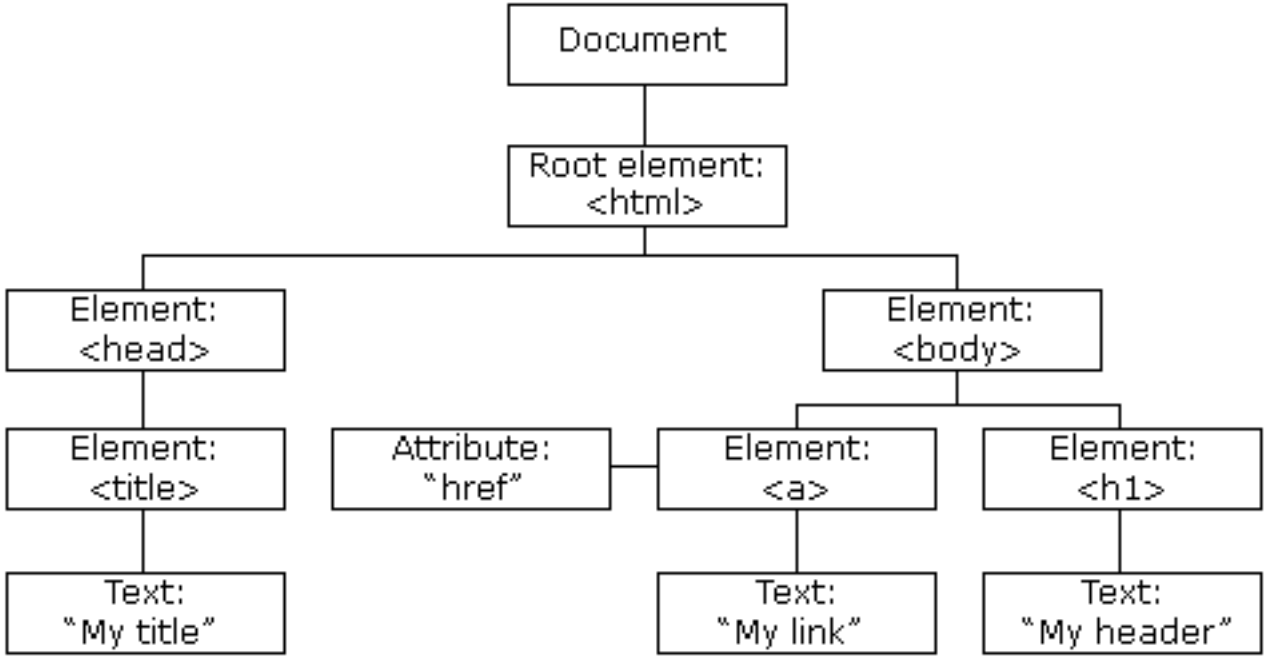


Objektni model dokumenta i objektni model browsera

HTML DOM

- Kada se web strana učita u browser kreira se objektni model strane i objekat **document**
- HTML DOM (document object model) obezbeđuje programski interfejs za pristup i manipulaciju html dokumentima
- HTML elemente definiše kao objekte
- Definiše metode za pristup HTML elementima
- Definiše događaje za sve HTML elemente

HTML DOM stablo



HTML DOM model se sastoji od stabla objekata

Node objekat

- **Node** je osnovni objekat u DOM-u koja predstavlja bilo koji deo dokumenta
- Node je apstraktna klasa u JavaScriptu
- To može biti element, tekst, komentar ili čak ceo dokument
- Vrste Node objekata:
  - **Element Node**: predstavlja HTML element
  - **Text Node**: predstavlja tekst unutar elementa
  - **Comment Node**: predstavlja HTML komentar
  - **Document Node**: predstavlja celokupni HTML dokument
- Svojstva Node objekta:
  - nodeType (1- element)
  - nodeName (npr div)
  - parentNode – vraća roditeljski čvor
  - childNodes – vraća kolekciju podređenih čvorova

Element objekat

- Nod je osnovni tip, dok je Element njegov podtip
- Element je konkretna klasa koja nasleđuje klasu Nod
- Element se takođe može se smatrati i kao interfejs koji definiše zajedničke karakteristike elemenata
- Element je specifičan tip Node objekta koji predstavlja HTML ili XML element
- Svi Element objekti su Node objekti, ali nije svaki Node objekat i Element objekat
- Svojstva:
  - **id**: ID elementa.
  - **className**: klasa elementa
  - **innerHTML**: HTML sadržaj unutar elementa
  - **attributes**: lista atributa elementa.

Metode Element objekta

- Metoda **appendChild**(): dodaje novi čvor kao poslednji podređeni čvora
- Metoda **removeChild**(): uklanja podređeni čvor
- Metoda **setAttribute**(): postavlja ili menja vrednost atributa

Svojstva objekta HTMLElement

- **HTMLElement** je osnovni objekat u JavaScriptu koji predstavlja HTML element u DOM strukturi
- Nasleđuje svojstva i metode iz Element i Node objekata
- Svi HTML elementi, kao što su , , <p>, <a>, itd., su instance HTMLElement objekta
- element.**id** – čita ili postavlja id atribut elemeta
- element.**style** – čita ili postavlja stilove elementa
- element.**innerHTML** - čita ili postavlja HTML sadržaj unutar elementa
- element.**innerText** - se koristi za dobijanje ili postavljanje vidljivog (tekstualnog) sadržaja HTML elementa
- element. **className** – dohvata ili postavlja klasu elementa
- element.**attributes** vraća kolekciju atributa elementa

Metode objekta HTMLElement

- **appendChild**(childNode): dodaje novi čvor kao poslednji čvor nadređenog čvora
- **removeChild**(childNode): uklanja podređeni čvor iz roditeljskog čvora
- **setAttribute**(name, value) : postavlja vrednost atributa na elementu
- **getAttribute**(name): vraća vrednost atributa sa datim imenom
- **focus**() : postavlja fokus na element (ako je to moguće)
- **getElementsByTagName**(name): vraća kolekciju svih podređenih elemenata sa datim nazivom tag-a
- **getElementsByClassName**(className): vraća kolekciju svih podređenih elemenata sa datom klasom

Objekat document - svojstva

- Objekat document u JavaScriptu predstavlja celokupni HTML dokument koji se trenutno učitava u browseru
- document.title: vraća ili postavlja naslov dokumenta
- document.body: vraća referencu na <body> element dokumenta
- document.head: vraća referencu na <head> element dokumenta
- document.URL: vraća URL trenutnog dokumenta

Objekat document - metode

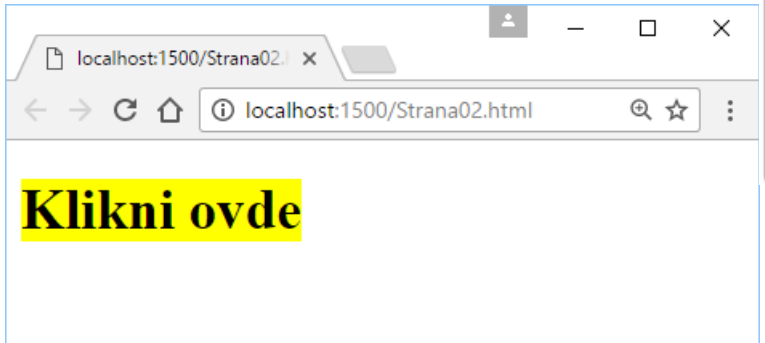
- document.**getElementById**(id): vraća element sa datim ID-om
- document.**getElementsByClassName**(className): vraća HTMLCollection svih elemenata sa datom klasom
- document.**getElementsByTagName**(tagName): vraća HTMLCollection svih elemenata sa datim nazivom tag-a

Izvedene klase iz kalse HTMLElement

- HTMLDivElement
- HTMLSpanElement
- HTMLAnchorElement
- HTMLImageElement
- HTMLButtonElement
- HTMLInputElement
- HTMLFormElement

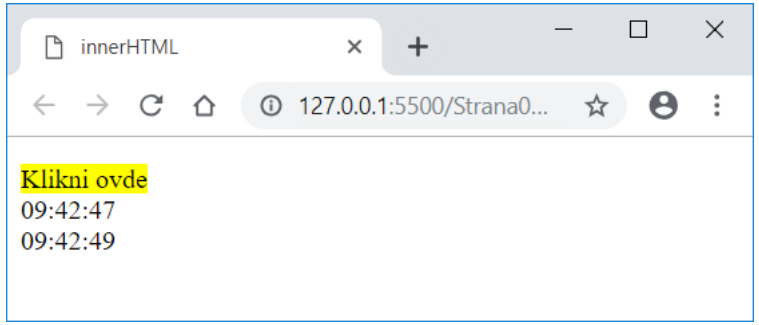
Čitanje html sadržaja nekog elementa

```
<body>
    <h1 id="heder1" onclick="Prikazi()">
        <mark>Klikni ovde</mark>
    </h1>
    <script>
    function Prikazi() {


        var  heder1 = document.getElementById("heder1");
        alert(heder1.innerHTML);

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



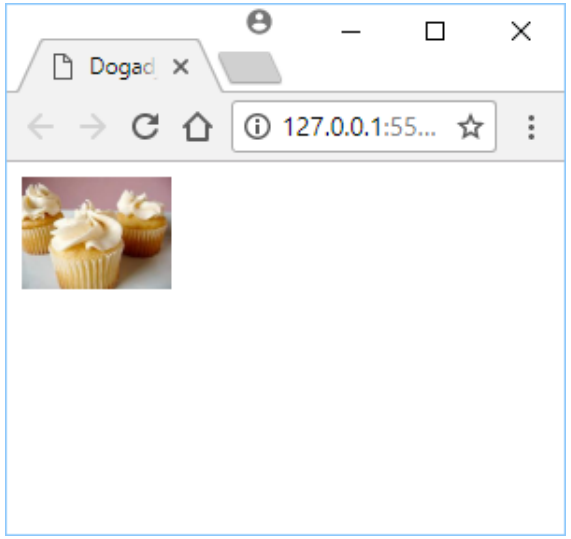
Promena sadržaja paragrafa

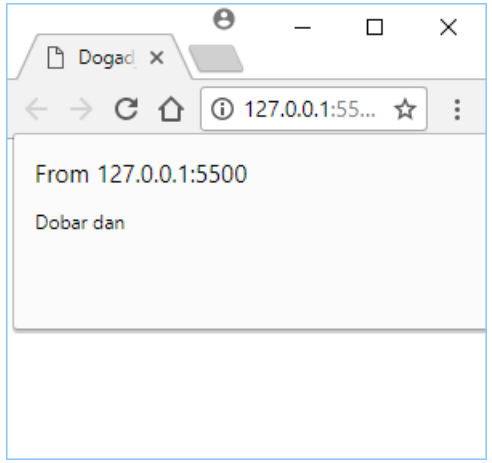
```
<body>
    <p id="p1" onclick="Prikazi()">
        <mark>Klikni ovde</mark>
    </p>

    <script>
    function Prikazi() {
        var p1 = document.getElementById("p1");
        p1.innerHTML += "<br>" + new Date().toLocaleTimeString("sr-Latn-RS");
    }
    </script>

</body>
```



Događaj kao atribut elementa

```
<body>
    <img src="/Slike/1.jpg" width="80" onclick="Pozdrav()">
    <script>
        function Pozdrav() {
            alert("Dobar dan");
        }
    </script>
</body>
```





Pretplata na događaj u JavaScript-u

```
<body>
    <img id="img1" src="/Slike/1.jpg" />

    <script>
            var slika = document.getElementById("img1");
            slika.onclick = function () {
                alert("Pozdrav");
            };
    </script>
</body>
```

HTML DOM addEventListener() metoda

```
<script>

var slika = document.getElementById("img1");

slika.addEventListener("click",Slika_Click);

function Slika_Click() {
    alert("Pozdrav");
}
</script>
```

Korisnički interfejs

```
<style>
    p{
        width: 200px;
        height: 120px;
        background-color :rgb(30, 116, 145);
        padding: 10px;
        color: white;
    }
</style>
```

```
<body>
    <p id="p1">

    </p>
</body>
```

HTML DOM removeEventListener() metoda

```
<script>
        var p1 = document.getElementById("p1");

        p1.addEventListener("mousemove",Prikazi);
        p1.addEventListener("click",Odjava);

        function Prikazi() {
            p1.innerHTML = Math.round(10 * Math.random()) ;
        }

        function Odjava() {
            p1.removeEventListener("mousemove",Prikazi);
            alert("Izvrsena odjava");
        }
</script>
```

Primer upotrebe objekta HTMLElement

```
<body>
    <p id = "p1">
        Paragraf 1
    </p>

    <script>
    var p1 = document.getElementById("p1");

    var id1 = p1.id;
    var id2 = p1.getAttribute("id");

    console.log(id1);
    console.log(id2);

    p1.setAttribute("class","plava");

    p1.style.color = "red";
    p1.style.height = "100px";
    p1.style.padding = "10px";
    </script>
</body>
```

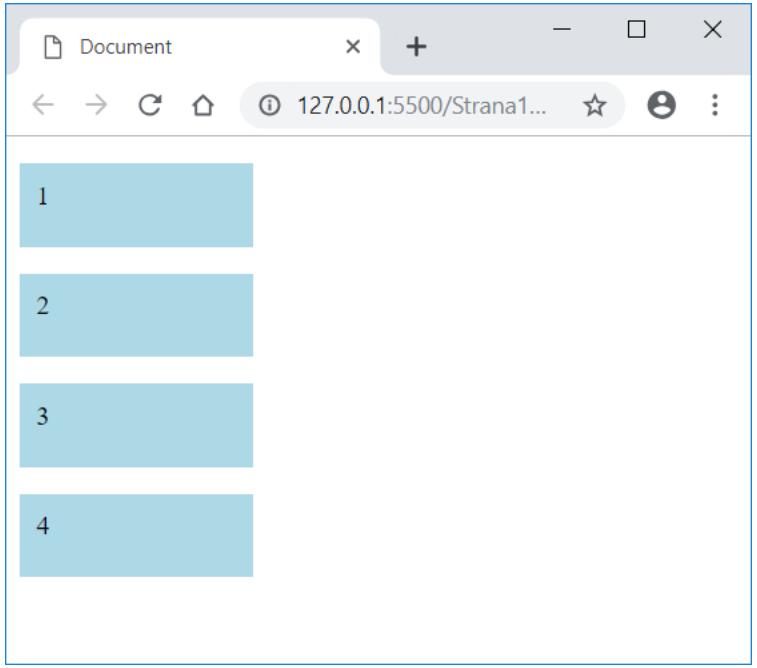
```
<style>
   .plava{
      background-color:#2b72a7;
    }
</style>
```

Metoda getElementsByTagName() objekta document

- Vraća **HTMLCollection** kolekciju elemenata sa odgovarajućim nazivom taga
- **HTMLCollection** je "živa" kolekcija elemenata, što znači da se kolekcija automatski ažurira ako se dodaju ili uklanjaju elementi iz DOM-a
- Koristi se length properti da se odredi broj članova u kolekciji
- Članovima kolekcije se pristupa preko indeksa

Stil paragrafa

```
<style>
    p{
        width: 120px;
        height: 30px;
        background-color: lightblue;
        margin-bottom: 10px;
        padding: 10px;
    }
 </style>
```



getElementsByTagName() - primer

```
<body>
    <p></p>
    <p></p>
    <p></p>
    <p></p>

    <script>

    var listaParagrafa = document.getElementsByTagName("p");

    for (const i in listaParagrafa) {
        listaParagrafa[i].innerHTML = parseInt(i) +1;
    }

    </script>
</body>
```

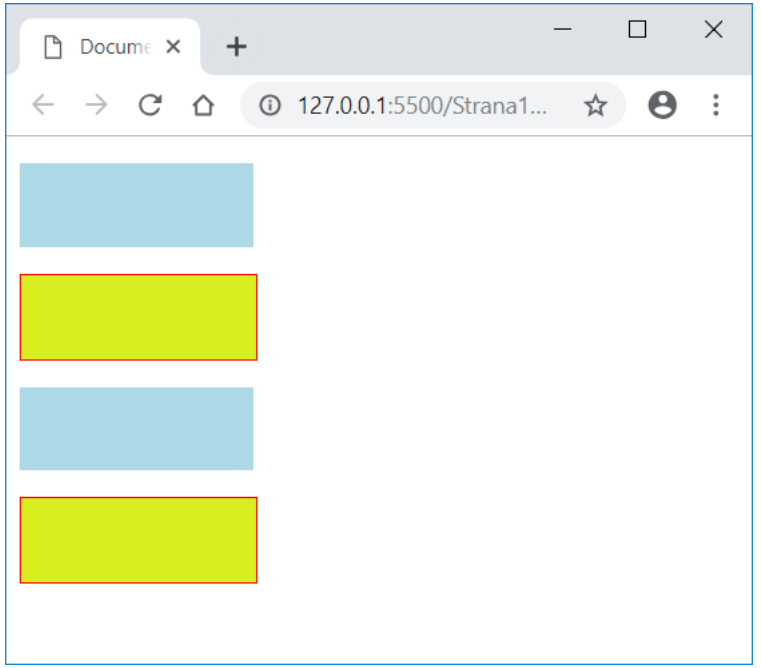
Metoda getElementsByClassName () objekta document

- Vraća **HTMLCollection** kolekciju elemente sa specificiranim imenom klase

```
<style>
        p {
            width: 120px;
            height: 30px;
            background-color: lightblue;
            margin-bottom: 10px;
            padding: 10px;
        }

        p.p1 {

            background-color: rgb(217, 238, 32);
        }
 </style>
```

Primer

```
<body>
    <p></p>
    <p class="p1"></p>
    <p></p>
    <p class="p1"></p>

    <script>
     var listaParagrafa = document.getElementsByClassName("p1");

     for (const i in listaParagrafa) {
         listaParagrafa[i].style.border = "1px solid red";
     }
    </script>
</body>
```



Pronalaženje HTML elemenata na osnovu css selektora

- Metoda **querySelector**() je metoda koja vraća prvi element u roditeljskom elementu koji odgovara određenom CSS selektoru
- Metoda **qerySelectorAll**() je metoda koja vraća sve elemente u roditeljskom element (NodeList) koji odgovaraju određenom CSS selektoru
- **NodeList** podseća na niz (array), ali nije prava JavaScript Array struktura
- NodeList je statička kolekcija i ne ažurira se automatski kada se DOM menja
- U NodeList strukturi može se pristupiti pojedinačnim elementima koristeći indekse (kao u nizu) i ima svojstvo length koje pokazuje koliko elemenata ima u listi.

Metode querySelector() i qerySelectorAll()

```
let element = parentNode.querySelector(selector);
```

```
let nodeList = document.querySelectorAll(selector);
let elements = Array.from(nodeList);
```

Metode querySelector() i qerySelectorAll()

```
<p class="p1">Prvi paragraf.</p>
<p class="p1">Drugi paragraf.</p>
```

```
    <script>
    document.querySelector("p.p1").style.backgroundColor = "red";
    </script>
```

```
<script>
    let nodeList = document.querySelectorAll(".p1");
    for (let i = 0; i < nodeList.length; i++) {
    const element = nodeList[i];
    element.style.backgroundColor = "red";
    }
</script>
```

Rad sa atributima elementa

- Koristi se interfejs **Element** za rad sa atributima
- **getAttribute**(name): vraća vrednost atributa sa zadatim imenom
- **setAttribute**(name, value): postavlja ili menja vrednost atributa sa zadatim imenom
- **removeAttribute**(name): uklanja atribut sa zadatim imenom
- **hasAttribute**(name): vraća true ako element ima atribut sa zadatim imenom, u suprotnom false
- **attributes**: vraća **NamedNodeMap** svih atributa elementa
- Kolekcija NemedNodeMap takođe predstavlja živu kolekciju

Rad sa atributima elementa

```
<body>
    <img id="img1" src="image.jpg" alt="Slika"  width="300">
    <script>

        const image = document.getElementById("img1");

        // Pristup atributima pomoću "attributes" svojstva
        const attributes = image.attributes;

        // Ispisivanje svih atributa
        console.log("Svi atributi slike:");
        for (let i = 0; i < attributes.length; i++) {
            console.log(`${attributes[i].name}: ${attributes[i].value}`);
        }


    </script>
</body>
```

Pitanje 1

Deklarisanje i inicijalizacija javascript niza je ispravno izvršena u primeru pod opcijom:

a. var a = [1, 2, 3, 4];
b. var b = {1,2,3,4};
c. var c= (1,2,3,4);

Odgovor: a

Pitanje 2

Šta se ispisuje na konzoli nakon izvršavanja sledećeg javascript koda:

```
<script>
    var osoba = { ime: "Pera", prezime: "Peric", starost: 30 };
    var s = "";
    for (var i in osoba) {
        s += i + " ";
    }
    console.log(s);
</script>
```

a. Pera Peric 30
b. ime prezime starost
c. ime:Pera prezime:Peric starost:30

Odgovor: b

Pitanje 3

Koja od sledećih sintaksi ispravno kreira JavaScript objekat?

a. var osoba = (ime: "Marko", starost: 30);
b. var osoba = [ime: "Marko", starost : 30];
c. var osoba = { ime: "Marko", starost : 30 };

Odgovor: c

Pitanje 4

Koja od sledećih klasa nasleđuje klasu Element i predstavlja osnovu za sve HTML elemente?

a. Node
b. HTMLElement
c. Document

Odgovor: b

Pitanje 5

Koja od sledećih opcija se koristi za pristup svim atributima HTML elementa u DOM-u?

a. getAttribute()
b. attributes
c. getAttributes()

Odgovor: b