JavaScript Manipulacija DOM stablom

Autori:
Milan Segedinac
Goran Savić

1. JavaScript i DOM stablo

Iako je JavaScript programski jezik opšte namene i koristi se i za serversko programiranje, ovaj programski jezik se prvenstveno upotrebljava za pisanje programa koji se izvršavaju u pregledaču. Programski kod napisan u JavaScriptu direktnose ugrađuje u HTML stranice u script elemente i izvršava se kada pregledač obrađuje taj element prilikom konstrukcije DOM stabla. Shodno tome, važna namena progmraskog jezika JavaScript je da omogući manipulaciju veb stranicom koja se prikazuje u pregledaču. Ovu namenu programski jezik JavaScript ostvaruje kroz funkcije koje omogućuju *direktnu manipulaciju DOM stablom učitane veb stranice*.

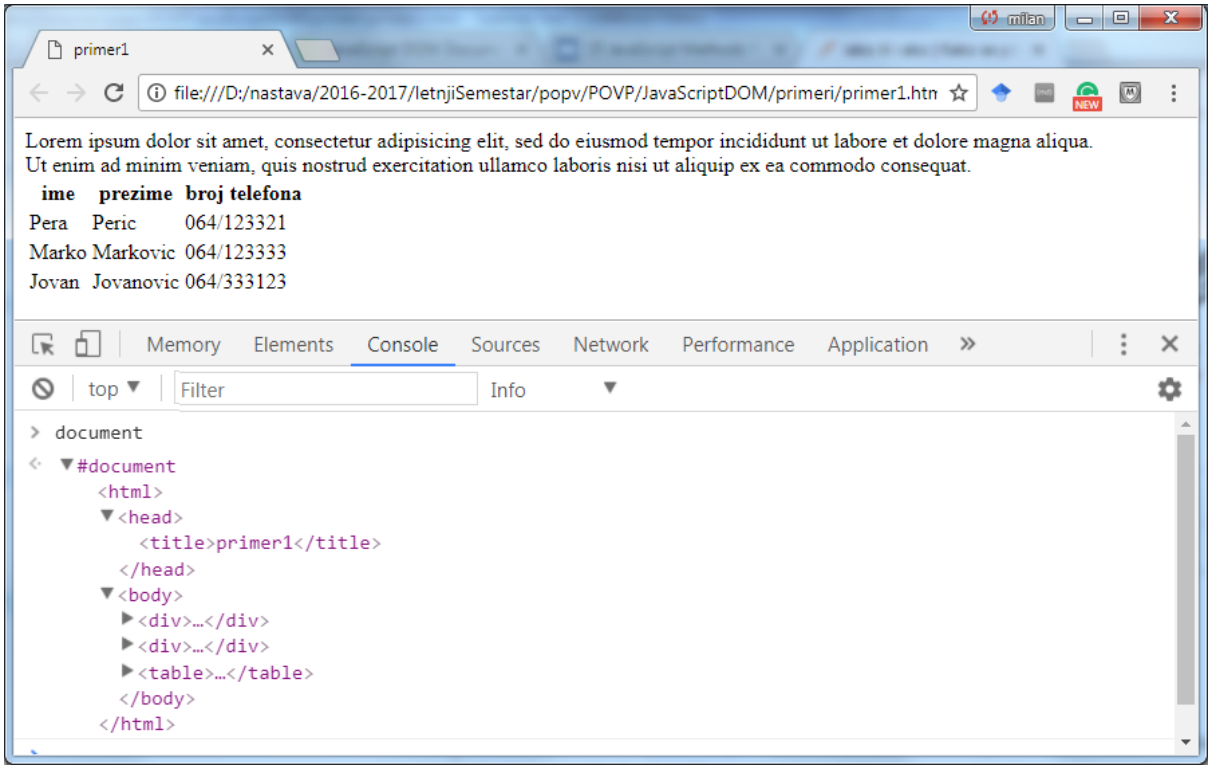
Pronalazak elementa u DOM stablu

U programskom JavaScript objektni model dokumena predstavljen je objektom `document`. U narednim primerima posmatraćemo HTML stranicu datu listingom ispod.

```
<html>
<head>
	<title>primer1</title>
</head>
<body>
	<div>
		Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
	</div>
	<div>
		Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
	</div>
	<table>
		<tr>
			<th>ime</th>
			<th>prezime</th>
			<th>broj telefona</th>
		</tr>
		<tr id="red-1">
			<td class="ime">Pera</td>
			<td>Peric</td>
			<td>064/123321</td>
		</tr>
		<tr class="red">
			<td class="ime">Marko</td>
			<td>Markovic</td>
			<td>064/123333</td>
		</tr>
		<tr class="red">
			<td class="ime">Jovan</td>
			<td>Jovanovic</td>
			<td>064/333123</td>
		</tr>
	</table>
	<script type="text/javascript" src="primer1.js"></script>
</body>
</html>
```

Listing – JavaScript objekat

Na slici ispod prikazan je pristup DOM objektu zadate stranice preko `document` objekta.



Slika – `document` objekat

Ovaj objekat nam omogućuje preuzimanje elemenata sa stranice. Osnovne metode za to su:

- `document.getElementById(id)` – preuzimanje elementa sa zadatim id-om
- `document.getElementsByTagName(name)` – preuzimanje svih elemenata zadatog tag-a (na primer svih div elemenata)
- `document.getElementsByClassName(name)` – preuzimanje svih elemenata zadate klase

Listingom ispod dat je JavaScript kod koji iz zadate stranice preuzima elemente koriseći navedene metode i ispis u konzolu koji je rezultat izvršavanja navedenog programskog koda.

```
>> el = document.getElementById('red-1');
>> console.log('getElementById vrati:',el);
<< getElementById vrati: <tr id="red-1">…</tr>

>> els = document.getElementsByTagName('div');
>> console.log('getElementsByTagName vrati:',els);
<< getElementsByTagName vrati: (2) [div, div]

>> els = document.getElementsByClassName('red');
>> console.log('getElementsByClassName vrati:',els);
<< getElementsByClassName vrati: (2) [tr.red, tr.red]
```

Listing – Preuzimanje elemenata po id, nazivu taga i nazivu klase

Vidimo da metoda `getElementById` prima id elementa koji treba pronaći i da vraća *jedan element* ukoliko je pronađen. To je u skladu sa činjenicom da atribut id treba jednoznačno da identifikuje HTML element. Metoda `getElementsByTagName` prima naziv taga i vraća *listu koja*

sadrži sve elemente tog taga u stranici. Metoda `getElementsByClassName` prima naziv klase i vraća *listu koja sadrži sve elemente te klase u stranici*.

Pored navedenih metoda, moguće je iz DOM stabla preuzeti i elemente koristeći CSS selektore. Za to se koriste sledeće metode:

- `querySelector` – pronalazi prvi element koji odgovara zadatom CSS selektoru
- `querySelectorAll` – pronalazi sve elemente koji odgovaraju zadatom CSS selektoru

Primer korišćenja ove dve funkcije nad posmatranom HTML stranicom dat je listingom ispod.

```
>> el = document.querySelector('table tr td.ime');
>> console.log('querySelector vrati:',el);
<< querySelector vrati: <td class="ime">Pera</td>

>> els = document.querySelectorAll('table tr td.ime');
>> console.log('querySelectorAll vrati:',els);
<< querySelectorAll vrati: (3) [td.ime, td.ime, td.ime]
```

Listing – Preuzimanje elemenata pomoću CSS selektora

Izmena DOM stabla

JavaScript omogućuje *izmenu učitanog DOM stabla* direktnim upisom u HTML tok. Za to je namenjena metoda `write`. Primer korišćenja ove funkcije dat je na slici ispod.

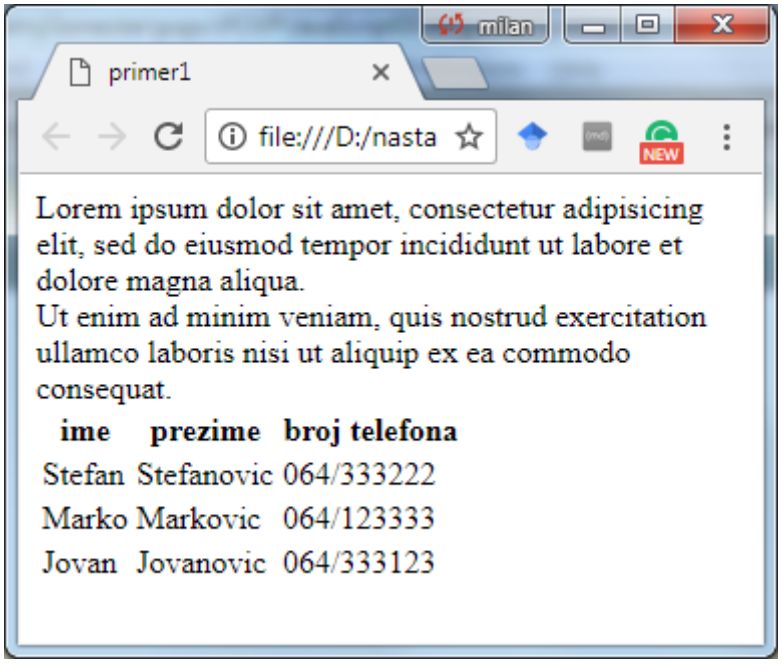
Slika – korišćenje `write` funkcije

Izvršavanja prve `write` funkcije dodalo je `<h1>` element sa tekstom „Izmenjeni HTML“. Izvršavanje druge `write` funkcije dodalo je novi `<div>` element sa tekstom „Funkcija write direktno piše u HTML tok“. Sav sadržaj koji je prethodno postojao u HTML stranici pri tome se izgubio.

Najčešće nam ne treba kompletno prepisivanje čitave HTML stranice, što postižemo korišćenjem `write` funkcije. Češći je slučaj da nam treba samo izmena sadržaja elemenata. Kada smo pruzeli element iz DOM stabla možemo da pristupimo njegovom *unutrašnjem HTML sadržaju* pomoću atributa `innerHTML`. Izmenom vrednosti ovog atributa dobijemo izmenu sadržaja elementa. Primer izmene sadržaja elementa dat je listingom i slikom ispod.

```
el = document.querySelector('#red-1');
el.innerHTML = '<td class="ime">Stefan</td><td>Stefanovic</td><td>064/333222</td>';
```

Listing – Izmena sadržaja selektovanog elementa



Slika – Prikaz izmenjenog HTML

Prvom linijom koda preuzet je element sa `id="red-1"`. Druga linija koda postavila je novu vrednost za sadržaj ovog elementa. Rezultat izvršavanja prikazan je slikom ispod.

Za selektovani element moguće je preuzeti i postaviti vrednost atributa. Za to se koriste metoda `getAttribute` i `setAttribute` koje se pozivaju nad preuzetim elementom.

2. Rukovanje događajima

JavaScript kod koji smo do sada pisali izvršavao se prilikom učitavanja HTML stranice, kada se u konstrukciji DOM stabla počne obrađivati `<script>` element. Pored toga, JavaScript kod može da se izvršava kao reakcija na događaje. Na primer, možemo napisati JavaScript kod koja se izvršava kada kliknemo na neki HTML element.

Prilikom zadavanja reakcije na događaj potrebno je zadati:

- Na koji element se događaj odnosi
- Koji događaj se posmatra
- JavaScript funkciju koja je reakcija na događaj.

Najjednostavniji nači zadavanja reakcije na događaj je pomoću atributa elemenata. Listingom ispod dat je primer ovakvkog definisanja događaja.

```
<html>
<head>
	<title>primer 4</title>
	<script type="text/javascript">
		function sayHello(){
			alert('Hello world!');
		}
	</script>
</head>
<body>
	<div onclick="sayHello()">
		Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
	</div>
</body>
</html>
```

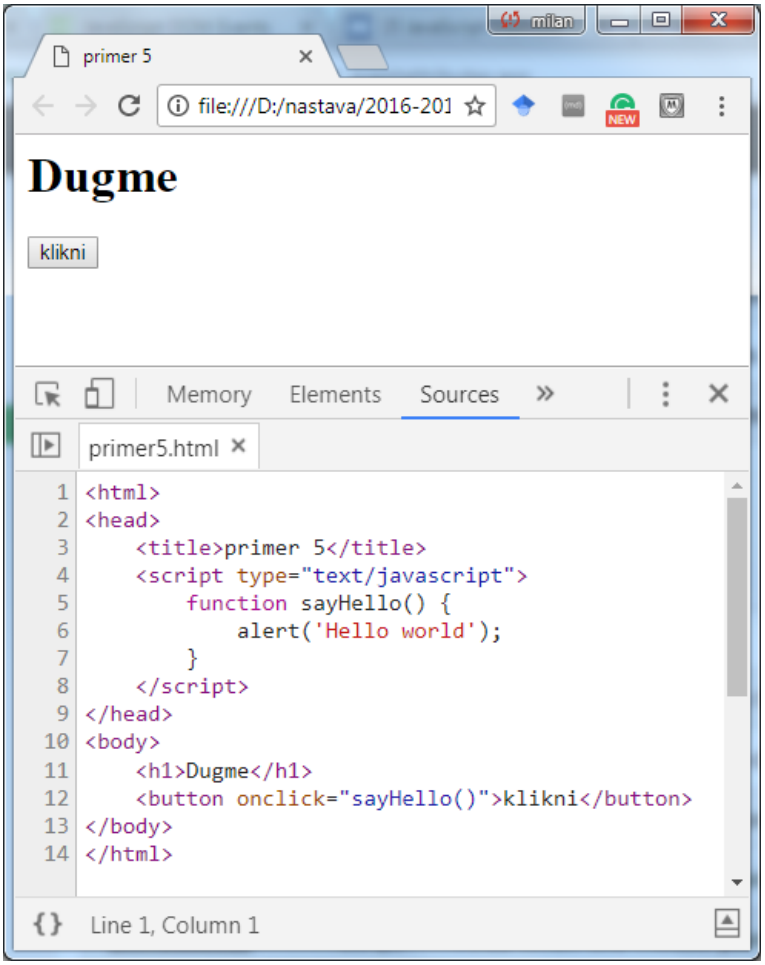
Listing – Zadavanje reakcije na događaj preko atributa elementa

Vidimo da smo definisali funkciju `sayHello` koja prikazuje alert sa tekstom „Hello world!“ u `<script>` elementu. U elementu `<div>` postavili smo vrednost atributa onclick da bude `sayHello()`. Time smo specificirali da će se, svaki put kada korisnik klikne na sadržaj tog `<div>` elementa izvršiti funkcija `sayHello`.

Pored `onclick` događaja, često se koriste i sledeći događaji miša:

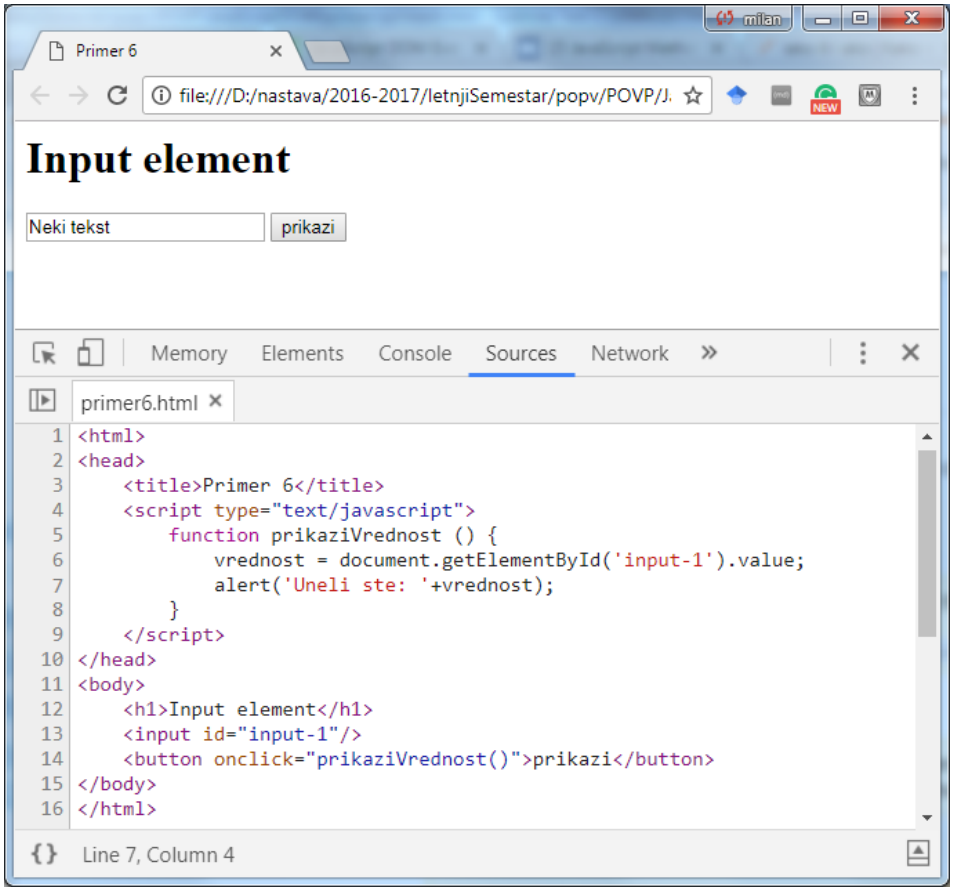
- `ondblclick` – dvoklik
- `onmouseover` – prelazak kurzorom preko elementa
- `onmouseout` – izlazak kurzora iz oblasti elementa

Ovi događaji se mogu vezati za bilo koji HTML element, ali poseban html element za koji se ti događaji najčešće vezuju je `<button>`. Sadržaj ovog elementa bitće prikazan kao tekst na dugmetu. Primer korišćenja ovog elementa prikazan je na slici ispod.



Slika – HTML dugme

Vidmo da uvođenjem JavaScript-a u HTML stranice, HTML prestaje da bude samo format za predstavljanje dokumenata i omogućuje interakcije sa korisnicima. Pored dugmeta, HTML ima i druge elemente korisničkog interfejsa od koji je najznačajniji `<input>` element. Ovaj element je polje za unos teksta i omogućuje korisniku da unosi vrednosti. Primer korišćenja ovog elementa prikazan je na slici ispod.



Slika – Input element

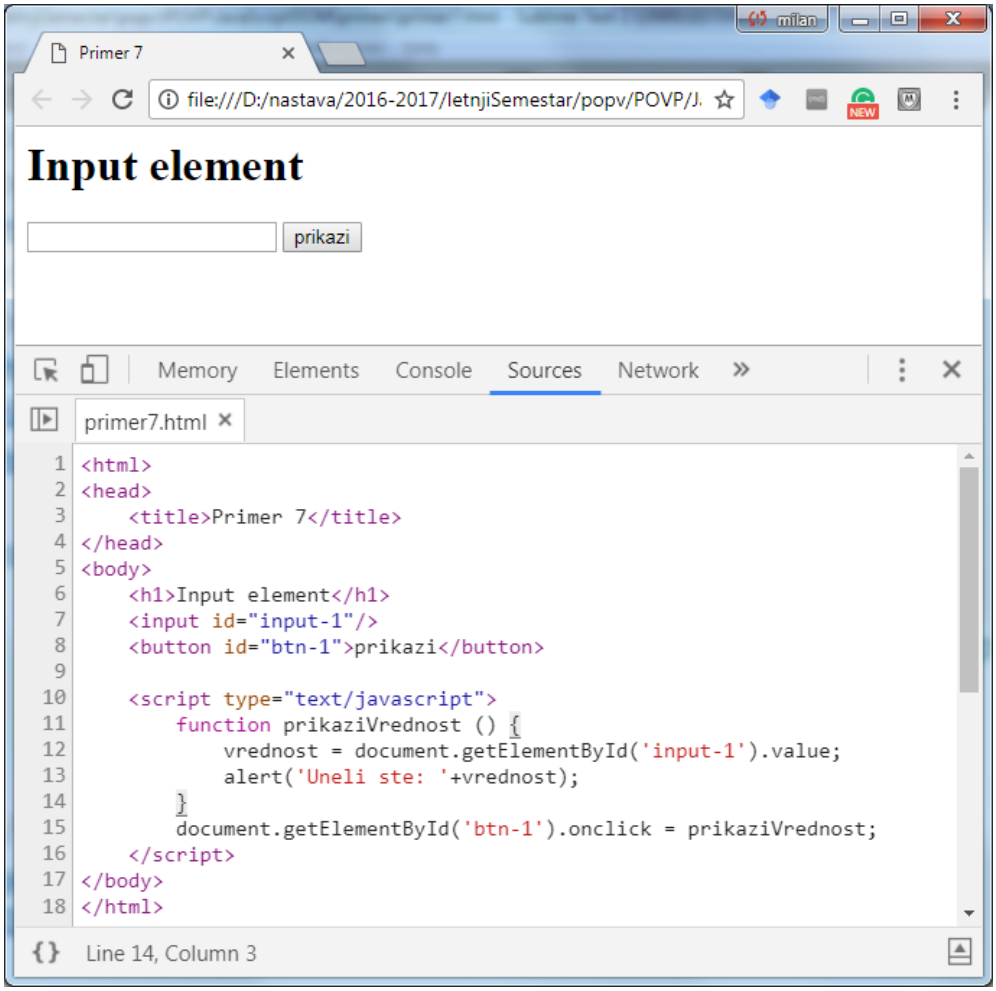
Napravljen je jedan `<input>` element koji ima `id="input-1"`. Na klik na dugme „`prikazi`" dodat je poziv funkcije `prikaziVrednost`. U toj funkciji vidimo kako se preuzima vrednost iz `<input>` elementa: selektujemo taj element i zatim pristupimo njegovom atributu `value`. Nakon preuzimanja vrednosti, vrednost je prikazana u alertu.

I za element `<input>` postoje karakteristični događaji:

- `onfocus` – kada se kurzor postavi na taj element
- `onblur` – kada kurzor napusti element
- `onkeypress` – kada korisnik unese jedan karakter sa tastature

Ovo je tek mali podskup DOM događaja. Kompletan skup DOM događaja dostupan je na https://developer.mozilla.org/en-US/ocs/Web/Events.

Važno je napomenuti da je reakcije na događaje moguće postaviti i iz JavaScript koda. Za to je potrebno selektovati jedan element na koji se vezuje događaj i postaviti njegov odgovarajući `on` atribut. Primer takvog postavljanja reakcije na događaj prikazan je na slici ispod.



Slika – Postavljanje reakcije na događaj iz JavaScript koda

U script elementu u HTML stranici kreirana je funkcija `prikaziVrednost`. Zatim je selektovan element sa `id="btn-a"`. Funkcija `prikaziVrednost` je postavljena kao vrednost atributa `onclick` selektovanog elementa. Obratite pažnju da činjenica da JavaScript podržava first-class funkcije dozvoljava da *funkcija bude postavljena na atribut*.