

Profesionalni JavaScript

Unapredite svoju karijeru veb developera
moćnim alatima naprednog JavaScript-a

Hugo Di Francesco

Siyuan Gao

Vinicius Isola

Philip Kirkbride



Profesionalni JavaScript

Hugo Di Francesco

Siyuan Gao

Vinicius Isola

Philip Kirkbride

ISBN 978-86-7991-429-3

Autorizovan prevod sa engleskog jezika knjige Professional JavaScript

Original Copyright© 2019. Packt Publishing

Copyright© prevoda, 2020. RAF Računarski fakultet, Beograd, CET Computer Equipment and Trade, Beograd

Sva prava zadržana. Nijedan deo ove knjige ne može biti reprodukovana, snimljen, ili emitovan na bilo koji način: elektronski, mehanički, fotokopiranjem, ili drugim vidom, bez pisane dozvole izdavača. Informacije korišćene u ovoj knjizi nisu pod patentnom zaštitom. U pripremi ove knjige učinjeni su svi naponi da se ne pojave greške. Izdavač i autori ne preuzimaju bilo kakvu odgovornost za eventualne greške i omaške, kao ni za njihove posledice.

Prevod Petar Prvulović

Recenzent Nemanja Radosevljević

Urednik Jovana Ristić

Lektor Marina Đenadić

Prelom Zoran Stanković

Izdavači CET Computer Equipment and Trade
Beograd, Skadarska 45
tel/fax: 011 3243-043, 3235-139, 3237-246
www.cet.rs

Računarski fakultet
Beograd, Knez Mihailova 6/VI
tel: 011 2627-613, 2633-321
www.raf.edu.rs

Za izdavača Dragan Stojanović, direktor

Tiraž 1000

Štampa „Pekograf“, Beograd

Nastavno-naučno veće Računarskog fakulteta na 142. elektronskoj sednici održanoj 27. 4. 2020. godine donelo je odluku da knjiga „Profesionalni JavaScript“ od autora Hugo Di Francesco, Siyuan Gao, Vinicius Isola i Philip Kirkbride bude štampana kao univerzitetski udžbenik.

SADRŽAJ

Predgovor	i
<hr/>	
Poglavlje 1: JavaScript, HTML i DOM	1
<hr/>	
Uvod	2
HTML i DOM	2
Vežba 1: Prolaženje kroz čvorove u dokumentu	4
Razvojni alati	6
Vežba 2: Rukovanje DOM-om iz kartice Elements	9
Vežba 3: Debugovanje koda pomoću kartice Sources	12
Kartica Console	17
Aktivnost 1: Izdvajanje podataka iz stranice	20
Čvorovi i elementi	21
Vežba 4: Obilaženje DOM stabla	24
Specijalni objekti	29
Ispitivanje DOM-a pomoću JavaScript-a	30
Vežba 5: Ispitivanje DOM-a pomoću querySelector metoda	32
Manipulisanje DOM stablom	37
Vežba 6: Filtriranje i pretraga proizvoda	42
Shadow DOM i Web Components	48
Vežba 7: Zamena polja za pretragu veb komponentom	58
Aktivnost 2: Zamena filtera sa oznakama sa veb komponentom	61
Sažetak	62
<hr/>	
Poglavlje 2: Node.js i npm	65
<hr/>	
Uvod	66

Šta je Node.js?	66
Vežba 8: Pokretanje vaših prvih Node.js naredbi	71
Node Version Manager (npm)	75
Vežba 9: Korišćenje npm za upravljanje verzijama	79
Node Package Manager (npm)	82
Vežba 10: Kreiranje konzolnog generatora HTML koda	87
Dependencies – preduslovi	91
npm skriptovi	93
Aktivnost 3: Kreiranje npm paketa za raščlanjivanje HTML-a	97
Sažetak	99

Poglavlje 3: Node.JS API-ji i Web scraping 101

Uvod	102
Globalni elementi	102
Vežba 11: Kreiranje aplikacije podsetnika	112
FileSystem API-ji	116
Vežba 12: Pretraga fajlova u direktorijumu pomoću glob šablona	126
HTTP API	130
Vežba 13: Isporučivanje statičkih fajlova	134
Vežba 14: Isporučivanje dinamičkog sadržaja	142
Šta je Scraping?	148
Preuzimanje i raščlanjivanje veb stranica	150
Vežba 15: Skrejpovanje članaka sa Mediuma	152
Aktivnost 4: Skrejpovanje proizvoda i cena sa kataloga	157
Sažetak	159

Poglavlje 4: RESTful API u Node.js 161

Uvod	162
------------	-----

Šta je API?	162
Šta je REST?	164
Express.js za RESTful API-je u Node.js	164
Vežba 16: Kreiranje Express projekta sa osnovnom rutom	165
Interakcija sa API-jem kroz HTTP	167
Vežba 17: Kreiranje i učitavanje fajla sa rutama	169
HTTP status kodovi	172
Dizajniranje API-ja	175
Vežba 18: Kreiranje ruta za akcije	178
Nastavak modularizacije	180
Provera tipova i validacija ulaznih vrednosti koje se šalju pristupnim tačkama	182
Vežba 19: Kreiranje rute sa proverom tipova i validacijomž	183
Podrazumevane vrednosti i jednostavniji zahtevi	186
Vežba 20: Pretvaranje trajanja u opcioni parametar	186
Middleware	189
Vežba 21: Postavljanje pristupne tačke koja zahteva autentifikaciju	191
Sadržaj JWT tokena	196
MongoDB	197
Aktivnost 5: Kreiranje API pristupne tačke za bravu sa šifrom za otključavanje	199
Sažetak	200
Poglavlje 5: Modularni JavaScript	203
<hr/>	
Uvod	204
Dependencies i bezbednost	205
Drugi troškovi modularnosti	207
Pregled procesa uvoženja i izvoženja	208
Podela odgovornosti	208

ES6 moduli	209
Vežba 22: Pisanje jednostavnog ES6 modula	210
Objekti u JavaScript-u	212
Prototipovi	212
Vežba 23: Proširivanje prototipa Number	215
ES6 klase	217
Objektno-orijentisano programiranje (OOP)	218
Apstrakcijai	219
Klase i konstruktori	220
Vežba 24: Konvertovanje modula light u klasu	221
Podrazumevani atributi	225
Enkapsulacija	225
WeakMap	226
Vežba 25: WeakMap za enkapsulaciju	227
Geteri i seteri	231
Nasleđivanje	231
Vežba 26: Proširivanje klase	232
Polimorfizam	235
Vežba 27: LightBulb kreator	236
npm paket	240
Naredba npm link	242
Naredba npm publish	243
ESM nasuprot CommonJS	244
Babel	244
webpack	245
Vežba 28: Konvertovanje ES6 i paketa sa webpack i Babel	246
Kompozabilnost i strategije kombinovanja modula	249
Aktivnost 6: Kreiranje sijalice sa fleš režimom rada	250
Sažetak	252

Poglavlje 6: Kvalitet koda 255

Uvod	256
Jasno imenovanje	257
Konvencije	258
Opinionated nasuprot non-opinionated	259
Linting	259
Vežba 29: Podešavanje ESLint i Prettier radi praćenja grešaka u kodu	260
Jedinični (Unit) testovi	263
Vežba 30: Podešavanje Jest test radi testiranja aplikacije kalkulatora	265
Integracioni testovi	267
Vežba 31: Integracioni testovi pomoću Jest	268
Performanse koda na primeru Fibonači sekvence	269
Vežba 32: Obezbeđivanje performansi pomoću Jest	272
End-to-end testiranje	273
Puppeteer	275
Vežba 33: End-to-end testiranje pomoću Puppeteer	277
Git hooks	280
Vežba 34: Postavljanje lokalne Git zakačke	280
Deljenje Git zakački pomoću biblioteke Husky	282
Vežba 35: Postavljanje commit zakačke pomoću Husky	283
Vežba 36: Dohvatanje elemenata tekstom pomoću Puppeteer ...	285
Aktivnost 7: Sastavljanje delova	288
Sažetak	290

Poglavlje 7: Napredni JavaScript 293

Uvod	294
Elementi jezika podržani u ES5,ES6,ES7,ES8 i ES9	294

Rad u Node.js REPL	295
Izvršavanje Node.js REPL	296
Rad sa nizovima u JavaScript-u	297
Vežba 37: Kreiranje i menjanje nizova	297
Vežba 38: Dodavanje i uklanjanje elemenata	299
Vežba 39: Dobijanje informacija o elementima u nizu	300
Aktivnost 8: Kreiranje spiska posetilaca	302
Manipulacija objektima u JavaScript-u	302
Vežba 40: Kreiranje i modifikovanje objekata u JavaScript-u	304
JSON.stringify	305
Vežba 41: Kreiranje efikasnog JSON.stringify	307
Dekompozicija nizova i objekata	309
Vežba 42: Upotreba dekompozicione dodele na nizu	309
Vežba 43: Upotreba dekompozicione dodele na objektu	310
Spread operatori	311
Vežba 44: Upotreba operatora proširenja	312
Rest operatori	313
OOP u JavaScript-u	314
Definisanje klase u JavaScript-u	315
Vežba 45: Deklarisanje konstruktora objekta preko funkcije	316
Vežba 46: Kreiranje klase u JavaScript-u	317
Kreiranje jednostavnog keša podataka o korisnicima pomoću objekata	320
Vežba 47: Kreiranje klase Cache za dodavanje/izmenu/brisanje zapisa iz skladišta podataka	320
Nasleđivanje klasa	322
Vežba 48: Implementiranje podklase	322
Privatni i javni metodi	325
Exercise 49: Privatni metodi u klasi Vehicle	326

Ugrađeni metodi nizova i objekata	327
Vežba 50: Upotreba metoda iteracije nad nizovima	330
Vežba 51: Pretraga i filtriranje niza	332
Sortiranje	334
Vežba 52: Sortiranje nizova u JavaScript-u	335
Svođenje niza - reduce	338
Vežba 53: Primena JavaScript metode reduce za računanje vrednosti u korpi	338
Aktivnost 9: Kreiranje evidencije studenata upotrebom JavaScript nizova i klasa	340
Mape i skupovi	342
Vežba 54: Poređenje upotrebe mapa i objekata	342
Vežba 55: Upotreba skupova za čuvanje unikatnih vrednosti	345
Math, Date i String	347
Vežba 56: Upotreba String metoda	348
Math and Date	350
Vežba 57: Math i Date	350
Simboli, iteratori, generatori i proksi	353
Symbol	353
Iterator i Generator	354
Vežba 58: Upotreba simbola i upoznavanje njihovih svojstava	354
Vežba 59: Iteratori i generatori	356
Proksi	358
Vežba 60: Upotreba proksija za kreiranje kompleksnih objekata	360
Refaktorisanje u JavaScript-u	362
Aktivnost 10: Refaktorisanje funkcija kako bi se upotrebile moderne JavaScript funkcional-nosti	363
Sažetak	364

Uvod	368
Kako JavaScript rukuje vremenski zahtevnim operacijama	370
Rukovanje asinhronim operacijama funkcijama povratnog poziva	371
Vežba 61: Vaša prva funkcija povratnog poziva	371
Ciklusi događaja	376
Kako JavaScript izvršava kod	377
Aktivnost 11: Upotreba funkcija povratnog poziva za dobijanje rezultata	379
Callback hell	381
Promises	384
Vežba 62: Promises kao alternativa funkcijama povratnog poziva	385
Ulančavanje	389
Vežba 63: Napredne mogućnosti JavaScript promises	392
Obrada grešaka u promises	398
Vežba 64: Refaktorisanje kalkulatora primenom promises	403
Async i Await	407
Vežba 65: Async i await funkcije	409
Async await konkurentnost	415
Kada koristiti await	418
Vežba 66: Kompleksne async implementacije	420
Aktivnost 12: Refaktorisanje kalkulatora upotrebom async i await	426
Migriranje koda zasnovanog na funkcijama povratnog poziva i promises na async i await	428
Migriranje koda zasnovanog na funkcijama povratnog poziva na async i await	428
Sažetak	430

Poglavlje 9: Programiranje vođeno događajima i ugrađeni moduli	433
<hr/>	
Uvod	434
Tradicionalni pristup nasuprot programiranju vođenom događajima	435
Eventing – komunikacija pomoću događaja	437
Vežba 67: Jednostavni emiter događaja	439
EventEmitter metodi	442
Uklanjanje slušalaca	442
Uklanjanje svih slušalaca	443
Dodavanje jednokratnog slušaoca	444
Čitanje iz emitera događaja	446
Dobijanje liste događaja na koje su se slušaoci prijavili	448
Maksimalni broj slušalaca	448
Dodavanje slušalaca na početak	449
Konkurentnost u slušaocima događaja	450
Kreiranje sopstvenih emitera događaja	452
Vežba 68: Kreiranje aplikacije za dopisivanje	456
Aktivnost 13: Kreiranje modula vođenog događajima	462
Dobre prakse u programiranju vođenom događajima	463
Node.js ugrađeni moduli	468
path	468
fs	470
Vežba 69: Osnovna upotreba modula fs	474
Rukovanje velikim fajlovima u Node.js	478
Util	479
Timer	481
Aktivnost 14: Kreiranje monitora fajla	486
Zaključak	487

Poglavlje 10: Funkcionalno programiranje i JavaScript 489

Uvod	490
Funkcije – građani prvog reda	490
Funkcije prvog reda – Idiomaticki gradivni elementi JavaScript-a	490
Inverzija kontrole primenom funkcija prvog reda	492
Funkcije koje omogućavaju asinhroni I/O i programiranje vođeno događajima u JavaScript-u	493
JavaScript ugrađeni metodi koji podržavaju funkcije prvog reda ..	496
Vežba 70: Revizija includes, indexOf i join sa some, findIndex i reduce	499
Vežba 71: Računanje cene korpe pomoću map i reduce	502
Komunikacija roditelj-dete komponenti u React-u	504
Aktivnost 15: onCheckout callback prop	509
Vežba 72: Dodavanje proizvoda u korpu	510
Funkcije prvog reda u React render props	512
Vežba 73: Prikazivanje sadržaja korpe pomoću render prop	514
Čiste funkcije	517
Redux reduktori i akcije	518
Vežba 74: Redux	520
Testiranje čistih funkcija	524
Vežba 75:	525
Redux selektori	529
Vežba 76: Implementiranje selektora	529
Aktivnost 16: Testiranje selektora	532
Funkcije višeg reda	532
bind, apply i call	533
Currying i parcijalna primena	537
Upotreba zatvorenja kod React funkcijskih komponenti	539

Kompozicija funkcija	540
Vežba 77: Funkcija komponovanja binarnog u n-arno	542
Kompozicija funkcija u stvarnom svetu primenom jednostavnog BFF	544
Vežba 78: Upotreba Compose kako bi se uprostio korak kreiranja mikroserversa	547
Imutabilnost i sporedni efekti	551
Osvrt na kreatore Redux akcija	551
Vežba 79: Refaktorisanje React/Redux aplikacije tako da koristi kreatore akcija	552
React-Redux mapStateToProps i mapDispatchToProps	555
Vežba 80: Apstrahovanje upravljanja stanjem pomoću funkcije mapDispatchToProps	556
Opširnije o Redux reducers	559
Pretvaranje JavaScript ugrađenih metoda u imutabilni funkcionalni oblik	560
Rukovanje sporednim efektima u React kopčama životnog ciklusa React/Redux aplikacije	566
Rukovanje sporednim efektima u React kopčama React/Redux aplikacije	567
Rukovanje sporednim efektima u Redux-thunk React/Redux aplikacije	568
Uvod u GraphQL šeme i upite	570
Pokretanje ažuriranja pomoću GraphQL mutacija i razrešilaca	572
Vežba 81: Implementiranje BFF mutacija sa micro i GraphQL	573
Aktivnost 17: Dohvatanje trenutne korpe sa BFF	577
Sažetak	578
Dodatak	581
<hr/>	
Index	633
<hr/>	



Predgovor

O poglavlju

Ovo poglavlje ukratko predstavlja autore, kao i teme koje knjiga pokriva, kao i tehničke veštine koje su vam potrebne kako biste mogli da počnete sa učenjem i hardver-a i softver-a, koji su potrebni kako bi se kompletirale sve date aktivnosti i vežbe.

O knjizi

Duboko razumevanje JavaScript-a olakšava učenje celog spektra drugih razvojnih okruženja, uključujući React, Angular i njima bliske alate i biblioteke. Ova knjiga dizajnirana je tako da vam pomogne u savladavanju suštinskih JavaScript koncepta, potrebnih za izgranju modernih aplikacija.

Početeće sa učenjem kako da predstavite HTML dokument u Document Object Modelu (DOM). Zatim ćete kombinovati svoje poznavanje DOM-a i Node.js, kako biste kreirali veb skrepper za neke praktične situacije. Kako budete napredovali kroz poglavlja, kreiraćete Node.js RESTful API pomoću biblioteke Express za Node.js. Razumećete i kako se modularni dizajn može koristiti, kako bi se postigla bolja višestruka upotrebljivost koda i kolaboracija više programera na jednom projektu. Kasnija poglavlja će vas voditi kroz kreiranje jediničnih testova, kojima se utvrđuje da se osnove funkcionalnosti programa vremenom ne pokvare. U knjizi se demonstrira i kako konstruktori, `async/await` i događaji mogu da postignu visoke performanse i brzinu aplikacije. Na kraju ćete dobiti koristan uvod u koncepte funkcionalnog programiranja, kao što su imutabilnost, čiste funkcije i funkcije višeg reda.

Do kraja ove knjige steći ćete veštine koje su vam potrebne da se uhvatite u koštac sa stvarnim problemima tokom JavaScript programiranja pomoću modernog pristupa JavaScript-u, kako na klijentskoj, tako i na serverskoj strani.

O autorima

Hugo Di Frančesko je softverski inženjer koji je dosta radio sa JavaScript-om. Ima zvanje mastera matematičkih izračunavanja na University College London (UCL). Koristio je JavaScript kroz celu vertikalnu, kako bi kreirao skalabilne platforme visokih performansi u kompanijama kao što su Canon i Elsevier. Trenutno se bavi problemima u domenu maloprodaje sa Node.js, React i Kubernetes dok istovremeno vodi i istoimeni sajt „Kodirajte sa Hugom”. Pored posla je i internacionalno priznati mačevalac, koji trenira i takmiči se po celom svetu.

Siuan Gao je softverski inženjer u kompaniji Electronic Arts. Diplomirao je računarske nauke na Univerzitetu Purdue. JavaScript i Node.js koristi više od 4 godine, uglavnom u izradi efikasnih pozadinskih rešenja za sisteme visokog nivoa dostupnosti. Takođe je saradnik Node.js Core projekta i objavio je mnogo npm modula. U slobodno vreme uživa u učenju o dizajnu video igara i mašinskom učenju.

Vinicius Isola počeo je programirati još 1999. godine, koristeći Macromedia Flash i ActionScript. Java sertifikacijom se bavi od 2005. godine i specijalizovao se za izgradnju veb i poslovnih aplikacija. Radeći u svim delovima vertikalne projektovanja, JavaScript i veb tehnologije su uvek bili prisutni u njegovim mnogobrojnim poslovima i kompanijama za koje je radio. U slobodno vreme voli da radi na projektima otvorenog koda i da se bavi mentorstvom novih programera.

Philip Kirkbrajd ima preko 5 godina iskustva sa JavaScript-om i baziran je u Montrealu. Završio je tehnički fakultet 2011. godine i od tada se bavi veb tehnologijama u različitim ulogama. Radio je sa 2Klic, IoT kompanijom, koju je Convectair, glavna kompanija za grejanje na struju, unajmila, kako bi projektovali pametne grejače koje pokreće Z-Wave tehnologija. Njegov posao sastojao se u pisanju mikro servisa u Node.js i Bash. Takođe je imao priliku da doprinese projektima otvorenog koda SteemIt (platforma za blogovanje zasnovana na blockchain-u) i DuckDuckGo (pretraživač koji se zasniva na privatnosti).

Ciljevi učenja

Do kraja ove knjige moći ćete da:

- Primenite osnovne koncepte funkcionalnog programiranja.
- Napravite Node.js projekat koji koristi biblioteku Express.js i sadrži API.
- Napravite jedinične testove za Node.js projekat, kako biste proverili njegovu ispravnost.
- Upotrebite biblioteku Cheerio sa Node.js, kako biste napravili jednostavni veb skrejper.
- Razvijete React interfejs kojim možete izraditi tokove obrade.
- Koristite funkcije povratnih poziva, kao osnovni način da vratite kontrolu.

Publika

Ako želite da napredujete od frontend programera do full-stack programera i naučite kako Node.js može da se koristi za pokretanje full-stack aplikacija, ovo je idealna knjiga za vas. Nakon čitanja ove knjige, moći ćete da napišete bolji JavaScript kôd i naučite o najnovijim trendovima jezika. Da biste lako shvatili ovde opisane koncepte, treba da znate osnovnu JavaScript sintaksu i trebalo bi da ste već radili sa popularnim bibliotekama, kao što je jQuery. Trebalo bi i da ste koristili JavaScript sa HTML i CSS, ali ne nužno i Node.js.

Pristup

Svaki odeljak ove knjige posebno je dizajniran da vas angažuje i stimuliše tako da ono što naučite možete zapamtiti i primeniti u praksi na najefektivniji mogući način. Naučićete kako da se nosite sa intelektualno stimulativnim izazovima programiranja, koji će vas kroz funkcionalno programiranje i razvoj vođen testiranjem pripremiti za realne teme. Svako poglavlje je pažljivo dizajnirano, tako da se nadograđuje na JavaScript kao osnovni jezik.

Hardverski zahtevi

Za optimalno iskustvo preporučujemo sledeću konfiguraciju

- Procesor: Intel Core i5 ili ekvivalentan;
- Memorija: 4 GB RAM
- Disk: 5 GB slobodnog prostora.

Softverski zahtevi

Preporučujemo i da imate spreman i instaliran sledeći softver:

- Git, najnovija verzija;
- Node.js 10.16.3 LTS (<https://nodejs.org/en/>)

Konvencije

Kodne reči u tekstu, imena tabela u bazi podataka, nazivi foldera, fajlova, ekstenzija fajlova, putanje, primeri URL-a, korisnički input i Twitter kopče su formatirani na sledeći način:

„ES6 funkcija **import** omogućava vam i da učitate podsekciju modula, umesto da učitate sve. Ovo je jedna od prednosti koje ES6 import ima u odnosu na Node.js funkciju **require**. SUSE”

Blok koda formatiran je ovako:

```
let myString = "hello";
console.log(myString.toUpperCase()); // vraća HELLO
console.log(myString.length); // vraća 5
```

Instalacija i podešavanje

Pre nego što počnemo da radimo neke super stvari sa podacima, treba da pripremimo produktivno okruženje. U ovom kratkom poglavlju videćete kako to da uradite.

Instaliranje Node.js i npm

Instalacije Node.js dolaze sa npm (Node.js podrazumevanim menadžerom paketa).

Instaliranje Node.js na Windows:

1. Pronađite željenu verziju Node.js na zvaničnoj stranici sa instalacijama na <https://nodejs.org/en/download/current/>.
2. Utvrdite da ste izabrali Node.js 12 (tekuću verziju).

3. Utvrdite da li ste instalirali na pravu arhitekturu vašeg računara; tj. 32-bitno, ili 64-bitno. Možete pronaći ove informacije u System Properties prozoru vašeg sistema.
4. Nakon što preuzmete instaler, kliknite dva puta na fajl i pratite instrukcije na ekranu.

Instaliranje Node.js i npm na Linux:

Za instalaciju Node.js na Linux imate nekoliko dobrih opcija:

- Za instaliranje Node.js kroz Linux menadžere paketa na sistemu koji nije naveden u nastavku, pogledajte <https://nodejs.org/en/download/package-manager/>.
- Za instaliranje Node.js na Ubuntu pokrenite ovo (više informacija i instrukcije za ručnu instalaciju možete naći na <https://github.com/nodesource/distributions/blob/master/README.md#installation-instructions>):
 - `curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -`
 - `sudo apt-get install -y nodejs`
- Za instaliranje Node.js na Debian distribucijama (više informacija i instrukcije za ručnu instalaciju možete naći na <https://github.com/nodesource/distributions/blob/master/README.md#installation-instructions>):
 - `# As root`
 - `curl -sL https://deb.nodesource.com/setup_12.x | bash -`
 - `apt-get install -y nodejs`
- Zvanična Node.js stranica za instalaciju ima još opcija za instalaciju na neke od Linux sistema: <https://nodejs.org/en/download/current/>.

Instaliranje Node.js i npm na macOS:

Slično Linux-u, postoji nekoliko načina da se Node.js i npm instaliraju na Mac.

Za instalaciju Node.js na macOS X uradite sledeće:

1. Otvorite Terminal za Max tako što ćete pritisnuti `cmd + Spacebar`, upisati **terminal** u polje za pretragu koje se otvori i pritisnuti `Enter`.
2. Instalirajte Xcode kroz komandnu liniju pokretanjem `xcode-select --install`.
3. Najjednostavniji način da se instalira Node.js i npm je pomoću Homebrew, koji se instalira kroz komandnu liniju pokretanjem `ruby -e "$(curl -fsSL (https://raw.githubusercontent.com/Homebrew/install/master/install))"`.

4. Konačni korak je da se instaliraju Node.js i npm. Na komandnoj liniji pokrenite **brew install node**.
5. Opet, možete instalirati Node.js i npm i pomoću instalera koji je dostupan na <https://nodejs.org/en/download/current/>.

Installing Git-a

Da biste instalirali git idite na <https://git-scm.com/downloads> i pratite instrukcije za vašu platformu.

Dodatni resursi

Paket koda koji prati ovu knjigu smešten je i na GitHub na <https://github.com/TrainingByPackt/Professional-JavaScript>. Imamo dostupne i pakete koda drugih knjiga i video-materijala iz našeg bogatog kataloga, dostupne na <https://github.com/PacktPublishing/>. Pogledajte čega tu ima!

1

JavaScript, HTML i DOM

Ciljevi i zadaci

Do kraja ovog poglavlja moći ćete da:

- Opišete HTML objektni model dokumenta (**DOM - Document Object Model**)
- Istražujete DOM veb stranice pomoću Chrome DevTools
- Napravite JavaScript koji čita i menja DOM
- Kreirate sopstvene komponente pomoću Shadow DOM

U ovom poglavlju naučićemo šta je DOM i kako da mu pristupimo i menjamo pomoću JavaScript-a. Naučićemo i kako da pravimo dinamičke aplikacije pomoću sopstvenih komponenti.

Uvod

HTML je zamišljen kao jezik za opisivanje statičkih dokumenata, koji bi bio jednostavan za upotrebu i mogao da se piše upotrebom bilo kog editora teksta. Kada je JavaScript postao bitan faktor u svetu interneta javila se potreba da se HTML dokumenti načine dostupnim JavaScript platformi. Tako je nastao DOM. DOM je zapravo HTML sadržaj koji je mapiran na stablo objekata, koji se mogu dohvatiti i menjati pomoću JavaScript-a.

U ovom poglavlju naučićete šta je DOM i kako da koristimo JavaScript da mu pristupimo. Naučićete kako da pronađete elemente i podatke u dokumentu, kako da menjate stanja elemenata i kako da im menjate sadržaj. Naučićete i kako da kreirate nove DOM elemente i dodajete ih u stranicu.

Kada završimo sa učenjem šta je DOM i kako se koristi, napravićete dinamičku aplikaciju na datom primeru. Konačno, naučićete kako da napravite prilagođene HTML elemente i kako da od njih formirate svoje komponente upotrebom Shadow DOM.

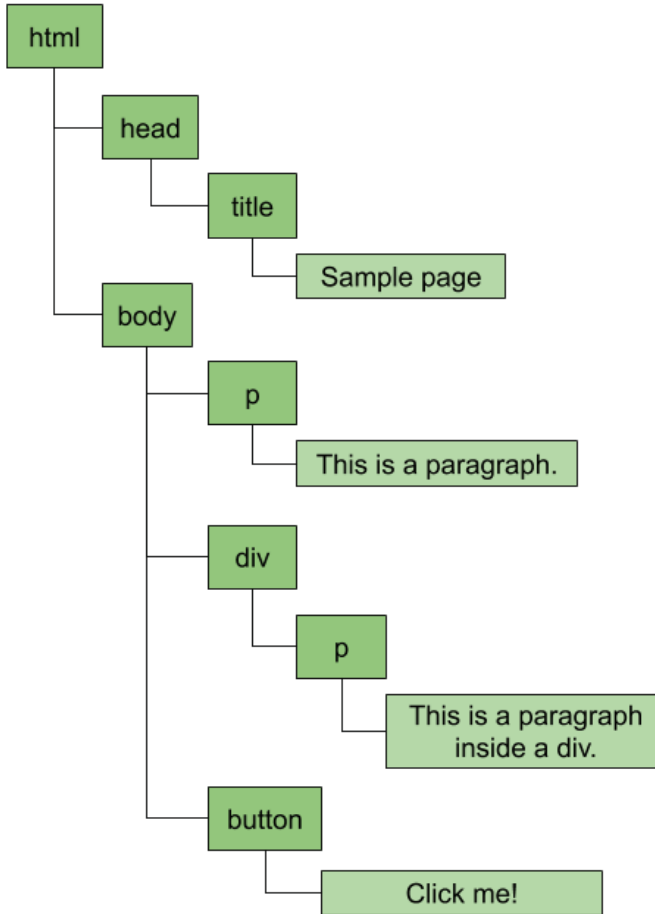
HTML i DOM

Kada veb pretraživač učita HTML stranicu kreira i strukturu stabla koje predstavlja tu stranicu. Ovo stablo bazirano je na DOM specifikaciji i koristi HTML oznake kako bi se odredilo gde koji čvor počinje i gde se završava.

Pogledajmo ovaj isečak HTML koda:

```
<html>
  <head>
    <title>Sample Page</title>
  </head>
  <body>
    <p>This is a paragraph.</p>
    <div>
      <p>This is a paragraph inside a div.</p>
    </div>
    <button>Click me!</button>
  </body>
</html>
```

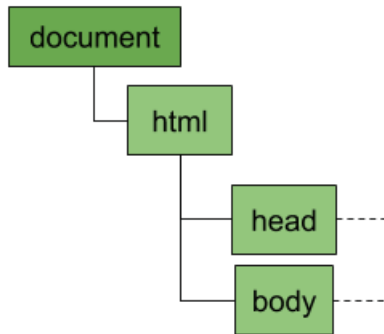

Pretraživač će kreirati ovakvu hijerarhiju čvorova:



Slika 1.1: Čvor paragrafa sadrži čvor sa tekstom

Sve postaje čvor, tekst, elementi i komentari pa sve do korenog elementa stabla. Ovakvo stablo koristi se da se elementima pridruže CSS stilovi da se formira prikaz stranice na ekranu. Pored toga, transformišu se u objekat i postaje dostupno JavaScript izvršnoj platformi.

Ali zašto se zove baš objektni model dokumenta? Zato što je HTML izvorno zamišljen za distribuciju dokumenata a ne za kreiranje dinamičkih aplikacija koje danas imamo. Otud, svaki HTML DOM počinje sa elementom dokumenta, na koji su zakačeni svi ostali elementi. Imajući to u vidu, prethodna ilustracija DOM stabla zapravo postaje ovakva:



Slika 1.2: Svako DOM stablo ima document element kao svoj koren

Šta znači kada kažem da pretraživač čini DOM dostupnim JavaScript platformi? Znači da ako pišete neki JavaScript kod u vašoj HTML stranici, možete da pristupite tom stablu i da radite neke prilično interesantne stvari sa njim. Na primer, možete da prilično lako pristupite korenom elementu dokumenta i pristupite svim čvorovima na stranici, i upravo to ćemo i uraditi u sledećoj vežbi.

Vežba 1: Prolaženje kroz čvorove u dokumentu

U ovoj vežbi napisaćemo JavaScript kod koji traži od DOM-a da pronade i dohvati dugme i doda mu slušaoca događaja (event listener) tako da možemo da izvršimo neki kod kada korisnik klikne na to dugme. Kada se desi događaj, dohvaćićemo sve elemente paragrafa, prebrojati ih i zapamtiti njihov sadržaj i na kraju prikazati obaveštenje. Fajlove sa kodom ove vežbe možete naći na <https://github.com/TrainingByPackt/Professional-JavaScript/tree/master/Lesson01/Exercise01>.

Pratite sledeće korake kako biste uradili vežbu:

1. Otvorite svoj omiljeni editor teksta, kreirajte novi fajl pod imenom `alert_paragraphs.html` i u njega smestite primer HTML koda iz prethodnog poglavlja (možete ga naći i na GitHub-u: <https://bit.ly/2maW0Sx>):

```
<html>
  <head>
    <title>Sample Page</title>
```