Ved Antani, Stoyan Stefanov

Objektno-orijentisan JavaScript

III izdanje

Naučite sve što treba da znate o objektno-orijentisanom JavaScriptu pomoću ovog sveobuhvatnog vodiča!
Uđite u svet najsavremenijeg programiranja!

kompjuter biblioteka

Packt>

Ved Antani
Stoyan Stefanov

Objektno-orijentisan JavaScript

III izdanje

kompjuter biblioteka

Packt>

Izdavač:

kompjuter biblioteka

Obalskih radnika 15, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autori: Ved Antani
Stoyan Stefanov

Prevod: Biljana Tešić

Lektura: Miloš Jevtović

Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: Apollo Plus D.O.O.
Beograd

Tiraž: 500

Godina izdanja: 2017.

Broj knjige: 496

Izdanje: Prvo

ISBN: 978-86-7310-519-2

Object-Oriented JavaScript

Third Edition
by Adam Boduch, Jonathan Chaffer and Karl Swedberg

ISBN 978-1-78588-056-8

Fifth edition: May 2017

O AUTORIMA

Ved Antani pravi skalabilne servere i mobilne platforme pomoću jezika JavaScript, Go i Java od 2005. godine. On je pomoćnik potpredsednika u kompaniji „Myntra“, a ranije je bio zaposlen u kompanijama „Electronics Arts“ i „Oracle“. Studirao je kompjuterske nauke i trenutno živi u Bangaloru, u Indiji. Ved je veliki ljubitelj klasične muzike i voli da provodi vreme sa svojim sinom.

Za pisanje ove knjige bilo je potrebno mnogo vremena i zahvaljujem mojim roditeljima i porodici na podršci i ohrabrenju tokom tih dugih dana i vikenda kada sam bio praktično „nevidljiv“.

Stoyan Stefanov je inženjer, autor i govornik, koji objavljuje svoje tekstove na Fejsbuku. Redovno nastupa kao govornik na konferencijama o veb programiranju i piše na svom blogu www.phpied.com. Takođe vodi veliki broj drugih sajtova, među kojima je i JSPatterns.com – sajt koji je namenjen istraživanju JavaScript obrazaca. Stoyan je ranije u kompaniji „Yahoo!“ napravio program YSlow 2.0 i alatku za optimizaciju slika Smash.it.

Stoyan, „građanin sveta“, rođen je i odrastao u Bugarskoj, ali je, takođe, državljanin Kanade, a trenutno živi u Los Anđelesu, u SAD. U slobodno vreme uživa u sviranju gitare, pohađa časove letenja i provodi vreme na plažama Santa Monike sa svojom porodicom.

Ovu knjigu posvećujem supruzi Evi i ćerkama Zlatini i Nataliji.

Hvala vam na strpljenju, podršci i ohrabrenju!

O RECENZENTU

Mohamed Sanaulla je programer softvera sa više od sedam godina iskustva u razvoju poslovnih aplikacija i Java pozadinskih programa aplikacija za elektronsku trgovinu.

Oblasti njegovog interesovanja su razvoj poslovnih softvera, refaktorisanje i redizajniranje aplikacija, dizajniranje i implementacija RESTful veb usluga, otklanjanje grešaka u performansama Java aplikacija i razvoj softvera koji je vođen testovima.

Specijalizovan je za razvoj Java aplikacija, ADF-a (JSF JavaEEE veb okruženja), jezika SQL, ekstenzija PL/SQL, okruženja JUnit, kao i za dizajniranje RESTful veb usluga i razvoj aplikacija Spring, Struts, Elasticsearch i MongoDB. Takođe je programer sa sertifikacijom „Sun Java“ za platformu Java 6 i moderator je sajta JavaRanch.com. Voli da podeli svoja zapažanja na svom blogu ((http://sanaulla.info).www.PacktPub.com.

Kratak sadržaj

Sadržaj

POGLAVLJE 11

POGLAVLJE 12

POGLAVLJE 13

DODATAK A

DODATAK B

DODATAK C

UVOD

JavaScript se pokazao kao jedan od najnaprednijih i najsvestranijih programskih jezika. Moderan JavaScript jezik obuhvata širok spektar proverenih i vrhunskih funkcija. Neke od tih funkcija su polako oblikovale sledeću generaciju Veba i serverskih platformi. Standard ES6 uvodi veoma važne jezičke konstrukcije, kao što su promisi, klase, arrow funkcije i još nekoliko dugo očekivanih funkcija. U ovoj knjizi su detaljno prikazane jezičke konstrukcije i njihova praktična primena. Objašnjene su i osnove JavaScript jezika, pa nije neophodno da posedujete znanje tog jezika. Za one koji poznaju JavaScript jezik ova knjiga će i dalje biti korisna i informativna. A onima koji poznaju ES5 sintaksu ova knjiga može poslužiti kao „bukvar“ o ES6 funkcijama.

ŠTA OBUHVATA OVA KNJIGA?

U Poglavlju 1, *Objektno-orijentisan JavaScript*, ukratko ćemo vam predstaviti istoriju, sadašnjost i budućnost JavaScripta, a zatim ćemo preći na istraživanje osnova objektno-orijentisanog programiranja (OOP). Zatim ćete naučiti kako da podesite okruženje za testiranje (Firebug) da biste mogli da „zaronite“ u jezik, tako što ćete koristiti primere u knjizi kao osnovu.

U Poglavlju 2, *Osnovni tipovi podataka, nizovi, petlje i uslovi*, razmatraćemo osnovne promenljive jezika, osnovne tipove podataka, nizove, petlje i uslovne naredbe.

Poglavlje 3, *Funkcije*, obuhvata funkcije koje se koriste u JavaScript jeziku. Ovladaćete njima i takođe ćete upoznati oblast važenja promenljivih i JavaScript ugrađene funkcije. Zanimljiva jezička funkcija o kojoj se malo zna closure biće demistifikovana na kraju poglavlja.

U Poglavlju 4, *Objekti*, objasnićemo objekte, upotrebu svojstava i metoda, kao i različite načine za kreiranje objekata. U ovom poglavlju ćemo, takođe, objasniti ugrađene objekte, kao što su niz, funkcija, logička promenljiva, broj i string.

U Poglavlju 5, *Iteratori i generatori jezika ES6*, predstavićemo najvažnije funkcije jezika ES6 – iteratore i generatore. Kada ih naučite, preći ćete na detaljan prikaz poboljšanih konstrukcija kolekcija.

Poglavlje 6, *Prototip*, posvećeno je svim važnim konceptima prototipova u JavaScriptu. Osim toga, objasnićemo kako funkcioniše lanac prototipova hasOwnProperty() i predstavićemo neke probleme sa prototipovima.

U Poglavlju 7, *Nasleđivanje*, objasnićemo kako funkcioniše nasleđivanje. Biće reči i o metodu za kreiranje potklasa, kao što su drugi klasični jezici.

U Poglavlju 8, *Klase i moduli*, prikazano je da ES6 standard uvodi važne sintaksičke funkcije koje olakšavaju pisanje klasičnih konstrukcija objektno-orijentisanog programiranja. ES6 sintaksa klase objedinjuje malo složeniju sintaksu jezika ES5. Osim toga, standard ES6 u potpunosti podržava module. U ovom poglavlju će detaljno biti prikazane konstrukcije klasa i modula koje su uvedene u jezik ES6.

U Poglavlju 9, *Promisi i proksiji*, objašnjeno je da je JavaScript oduvek bio jezik sa snažnom podrškom za asinhrono programiranje. Sve do pojave standarda ES5, pisanje asinhronih programa se zasnivalo na povratnim pozivima, koji su ponekad dovodili do situacije koja se u JavaScriptu naziva „pakao povratnih poziva“. ES6 promisi su dugo očekivana funkcija koja je uvedena u jezik. Promisi omogućavaju mnogo čistiji način za pisanje asinhronih programa u jeziku ES6. Proksiji se koriste za definisanje prilagođenog ponašanja nekih osnovnih operacija. U ovom poglavlju prikazana je praktična upotreba promisa i proksija u jeziku ES6.

Poglavlje 10, *Okruženje pregledača*, posvećeno je pregledačima. Ono obuhvata i BOM (objektni model pregledača), DOM (objektni model dokumenta koji je definisan W3C standardom), događaje pregledača i AJAX.

U Poglavlju 11, *Obrasci za pisanje programa i dizajn*, „zaronićemo“ u razne jedinstvene JavaScript obrasce i u nekoliko jezički nezavisnih obrazaca za dizajn koji su prevedeni na jezik JavaScript iz knjige koju je napisala grupa od četiri programera („Patterns: Elements of Reusable Object-Oriented Software“), a koja predstavlja najuticajnije delo o obrascima za dizajniranje softvera. U ovom poglavlju će, takođe, biti reči o objektu JSON.

U Poglavlju 12, *Testiranje i otklanjanje grešaka*, videćete da je savremeni JavaScript opremljen alatkama koje podržavaju razvoj softvera koji je vođen testovima (Test Driven Development) i razvoj softvera koji je vođen ponašanjem (Behavior Driven Development). Jasmine je jedna od najpopularnijih alatki koja je trenutno dostupna. U ovom poglavlju razmatramo TDD i BDD razvoj softvera, tako što koristimo Jasmine kao okruženje.

U Poglavlju 13, *Reaktivno programiranje i biblioteka React*, videćete da je sa pojavom standarda ES6 nastalo nekoliko radikalnih ideja. U reaktivnom programiranju se primenjuje veoma različit pristup za upravljanje stanjima, tako što se koriste tokovi podataka. Međutim, React predstavlja okruženje koje se fokusira na deo View arhitekture MVC. U ovom poglavlju ćemo razmatrati reaktivno programiranje i biblioteku React.

Dodatak A, *Rezervisane reči*, sadrži listu rezervisanih reči u JavaScriptu.

Dodatak B, *Ugrađene funkcije*, sadrži listu ugrađenih JavaScript funkcija sa primerima korišćenja.

Dodatak C, *Ugrađeni objekti*, sadrži detaljan pregled i primere korišćenja svakog metoda i svojstava svakog ugrađenog objekta u JavaScriptu.

Dodatak D, *Regularni izrazi*, sadrži listu obrazaca regularnih izraza.

Dodatak E, *Odgovori na pitanja u vežbama*, sadrži rešenja vežbi koje su navedene na kraju svakog poglavlja.

ŠTA VAM JE POTREBNO ZA OVU KNJIGU?

Potrebni su vam moderan pregledač, kao što je Google Chrome ili Firefox, i opciono Node.js podešavanje. Većina kodova iz ove knjige se može izvršiti u kompajleru http://babeljs.io/r epl/ ili http://jsbin.com/. Za uređivanje JavaScripta možete da koristite uređivač teksta po vašem izboru.

ZA KOGA JE OVA KNJIGA?

Ova knjiga je namenjena onima koji počinju da uče JavaScript ili onima koji poznaju JavaScript jezik, ali se ne snalaze dobro u objektno-orijentisanom programiranju. Ako već poznajete funkcije jezika ES5, ova knjiga može da vam posluži kao osnova za jezik ES6.

KONVENCIJE

U ovoj knjizi pronaći ćete više različitih stilova za tekst koje smo upotrebili za različite vrste informacija. Evo nekih primera ovih stilova i objašnjenja njihovog značenja.

Reči koda u tekstu, nazivi tabela baze podataka, nazivi direktorijuma, nazivi fajlova, ekstenzije datoteka, nazivi putanja, kratki URL-ovi, korisnički unos i Twitter identifikatori su prikazani na sledeći način: „Konstruktor `Triangle` preuzima objekte koji se formiraju pomoću tri tačke i dodeljuje ih nizu `this.points` (kolekciji tačaka)“.

Blok koda je postavljen na sledeći način:

```
function sum(a, b) {
var c = a + b;
return c;
}
```

Svaki unos ili ispis komandne linije je napisan na sledeći način:

```
mkdir babel_test
cd babel_test && npm init
npm install --save-dev babel-cli
```

Novi termini i **važne reči** su napisani masnim slovima. Reči koje vidite na ekranu (na primer, u menijima ili u okvirima za dijalog) biće prikazane u tekstu na sledeći način: „Da bi konzola bila prikazana u pregledaču Chrome ili Safari, kliknite desnim tasterom miša bilo gde na stranici i izaberite **Inspect Element**“.

Upozorenja ili važne napomene će biti prikazani u ovakvom okviru.

Saveti i trikovi prikazani su ovako.

POVRATNE INFORMACIJE OD ČITALACA

Povratne informacije od naših čitalaca su uvek dobrodošle. Obavestite nas šta mislite o ovoj knjizi – šta vam se dopalo ili šta vam se možda nije dopalo. Povratne informacije čitalaca su nam važne da bismo kreirali naslove od kojih ćete dobiti maksimum.

Da biste nam poslali povratne informacije, jednostavno nam pošaljite e-mail na adresu informatori@kombib.rs i u naslovu poruke napišite naslov knjige.

ŠTAMPARSKE GREŠKE

Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, greške su moguće. Ako pronađete grešku u nekoj od naših knjiga (u tekstu ili u kodu), bili bismo zahvalni ako biste nam to prijavili. Na taj način možete da poštedite druge čitaoce od frustracije, a nama da pomognete da poboljšamo naredne verzije ove knjige. Ako pronađete neku štamparsku grešku, molimo vas da nas obavestite, tako što ćete posetiti stranicu http://www.packtpub.com/submit – errata, selektovati knjigu, kliknuti na link Errata Submission Form i uneti detalje o grešci koju ste pronašli. Kada je greška verifikovana, vaša prijava će biti prihvaćena i greška će biti aploudovana na naš veb sajt ili dodata u listu postojećih grešaka, pod odeljkom Errata za određeni naslov.

Da biste pogledali prethodno prijavljene greške, posetite stranicu https://www.packtpub.com/books/content/support i unesite naslov knjige u polje za pretragu. Tražena informacija će biti prikazana u odeljku Errata.

PIRATERIJA

Piraterija autorskog materijala na Internetu je aktuelan problem na svim medijima. Mi u „Packtu“ zaštitu autorskih prava i licenci shvatamo veoma ozbiljno. Ako pronađete ilegalnu kopiju naših knjiga, u bilo kojoj formi, na Internetu, molimo vas da nas o tome obavestite i pošaljete nam adresu lokacije ili naziv web sajta da bismo mogli da podnesemo tužbu.

Kontaktirajte sa nama na adresi copyright@packtpub.com i informatori@kombib.rs pošaljite nam link ka sumnjivom materijalu.

1

Objektno-orijentisan JavaScript

Još od najranijih dana Veba postojala je potreba za dinamičnijim i prilagođenim interfejsima. Dobro je pročitati statičke HTML stranice sa tekstom, ali mnogo je bolje kada su te stranice lepo prezentovane pomoću CSS-a i zabavnije je da se povežete sa aplikacijama u pregledačima koje služe za e-poštu, kalendare, bankarstvo, kupovinu, crtanje, igranje igara i uređivanje teksta. Moguće je koristiti sve ove aplikacije, zahvaljujući JavaScriptu – programskom jeziku Veba. JavaScript jezik se na početku sastojao samo od jednostruke linije ugrađene u HTML-u, ali sada se koristi na mnogo sofisticiranije načine. Programeri su iskoristili funkcije objektno-orijentisanog jezika da bi napravili skalabilne arhitekture koda, koje se sastoje od delova koji se mogu ponovo koristiti.

Ako pogledate starije i novije popularne izraze u veb programiranju, kao što su DHTML, Ajax, Web 2.0 i HTML5, videćete da HTML označava **sadržaj**, CSS **prezentaciju**, a JavaScript **ponašanje**. Drugim rečima, JavaScript je „lepak“ koji spaja sve elemente, tako da možemo da napravimo bogate veb aplikacije.

Međutim, tu nije kraj. JavaScript se ne koristi samo za Veb.

JavaScript programi funkcionišu unutar okruženja hosta. Veb pregledač je uobičajeno okruženje, ali ne i jedino. Pomoću JavaScripta možete da kreirate sve vrste vidžeta, ekstenzije aplikacija i druge delove softvera, kao što ćete ubrzo videti. Vreme koje ćete uložiti u učenje JavaScripta je pametna investicija – naučite jedan jezik i onda možete da pišete različite vrste aplikacija koje funkcionišu na više platformi, uključujući mobilne i serverske aplikacije. Danas sa sigurnošću možemo reći da je JavaScript prisutan svuda.

Ova knjiga počinje „od nule“ i nije vam potrebno znanje o programiranju, već samo osnovno poznavanje HTML jezika. Jedno poglavlje je posvećeno okruženju veb pregledača, a ostatak knjige sadrži opšte informacije o JavaScriptu, tako da se može primeniti u svim okruženjima.

Počnimo od sledećeg:

- kratak uvod o nastanku JavaScripta
- osnovni koncepti sa kojima ćete se susresti u objašnjenjima objektno-orijentisanog programiranja

KRATKA ISTORIJA

Na početku je Veb bio samo niz naučnih publikacija u obliku statičkih HTML dokumenata koji su bili povezani hiperlinkovima. Verovali ili ne, tada nije postojao način da se slike postave na stranicu, ali to se ubrzo promenilo. Kako je rasla popularnost i veličina Veba, tako su administratori veb sajtova koji su kreirali HTML stranice shvatili da im treba nešto bolje. Hteli su da kreiraju bogatije korisničke interakcije, uglavnom zbog želje da uštede prostor na serveru za jednostavne zadatke koje je trebalo obaviti više puta, kao što je validacija obrazaca. Nastale su dve opcije – Java apleti i LiveScript jezik, koji je osmislio Brendan Eich u kompaniji *Netscape* 1995. godine, a koji je kasnije uveden u pregledač Netscape 2.0 pod nazivom JavaScript.

Apleti nisu masovno prihvaćeni, ali JavaScript jeste. Zajednica administratora veb sajtova je prihvatila mogućnost upotrebe kratkih isečaka koda koji su ugrađeni u HTML dokumenta i izmenu statičkih elemenata veb stranice. Zatim je konkurentski dobavljač pregledača „Microsoft“ objavio pregledač **Internet Explorer (IE)** 3.0 sa JScript jezikom, koji je bio obrnuta inženjerska verzija JavaScripta sa nekim IE funkcijama. Na kraju, uložen je trud da se standardizuju različite implementacije jezika i tako je nastao jezik ECMAScript. **Evropska socijacija proizvođača računara (ECMA)** je kreirala standard ECMA-262, u kome su opisani osnovni delovi JavaScript programskog jezika, bez pregledača i funkcije za veb stranice.

JavaScript možemo da posmatramo kao pojam koji obuhvata sledeća tri dela:

- **ECMAScript**: osnovni jezik – promenljive, funkcije, petlje i tako dalje. ECMAScript jezik nije povezan sa pregledačem i može da se koristi u mnogim drugim okruženjima.
- **objektni model dokumenta (DOM) -** Ovaj model obezbeđuje načine za korišćenje HTML i XML dokumenata. Na početku je JavaScript omogućavao ograničeni pristup onim elementima koji se mogu skriptovati na stranici, uglavnom obrascima, linkovima i slikama. Kasnije je JavaScript jezik proširen da bi svi elementi mogli da se skriptuju. Zbog toga je **World Wide Web konzorcijum (W3C)** kreirao jezički nezavisan (koji više nije povezan sa JavaScriptom) DOM standard za manipulisanje strukturiranim dokumentima.
- **objektni model pregledača (BOM) -** Ovo je skup objekata koji je povezan sa okruženjem pregledača i nikada nije bio deo ni jednog standarda, sve dok HTML5 nije započeo standardizaciju nekih uobičajenih objekata koji postoje u pregledačima.

U ovoj knjizi jedno poglavlje je posvećeno pregledačima DOM i BOM, ali u većem delu knjige opisan je osnovni jezik i naučićete veštine koje možete da koristite u svakom okruženju u kojem funkcionišu JavaScript programi.

„Ratovi“ zbog pregledača i preporod

Brzu popularnost JavaScript je stekao tokom prvog „rata“ zbog pregledača (u periodu od 1996. do 2001. godine). To su bila vremena početne ekspanzije Interneta, kada su se dve velike kompanije koje su razvile pregledače „Nestcape“ i „Microsoft“ nadmetale u osvajanju tržišta. Obe su konstantno dodavale nove opcije u svoje pregledače i u verzije JavaScripta i u modelima DOM i BOM, što je, naravno, dovelo do mnogo nekonzistentnosti. Prilikom dodavanja novih funkcija dobavljači pregledača nisu uspeli da obezbede odgovarajuće alatke za programiranje i otklanjanje grešaka, kao ni odgovarajuću dokumentaciju. Često je programiranje predstavljalo težak posao – trebalo je da napišemo skript u jednom pregledaču i nakon završetka programiranja da ga testiramo u drugom pregledaču, a onda bismo otkrili da skript jednostavno ne funkcioniše zbog nekog nepoznatog razloga i bila bi prikazana samo poruka o prekidu operacije iz koje nije mogao da se sazna razlog greške.

Zbog nekonzistentne implementacije i nedostatka dokumentacije i odgovarajućih alatki, JavaScript je predstavljen „u lošem svetlu“, pa su mnogi programeri jednostavno odbijali da ga koriste.

Sa druge strane, programeri koji su pokušali da eksperimentišu sa JavaScriptom su malo preterali, jer su dodavali previše specijalnih efekata na stranice, a da pri tom nisu obraćali pažnju da li će krajnji rezultati biti upotrebljivi. Programeri su bili nestrpljivi da iskoriste svaku novu mogućnost u pregledačima, pa su na kraju poboljšavali svoje veb stranice animacijama u statusnoj traci, jarkim bojama, trepćućim tekstovima, objektima koji su pratili kursor i mnogim drugim inovacijama, koje su, u stvari, pravile problem korisnicima. Zloupotreba JavaScripta na različite načine je vremenom prestala, ali zbog nje je on imao lošu reputaciju. Mnogi ozbiljni programeri smatrali su da je jezik JavaScript samo „igračka“ koju koriste dizajneri i da nije pogodan za upotrebu u ozbiljnim aplikacijama. Negativne kritike o JavaScriptu dovele su do toga da se u potpunosti zabrani klijentsko programiranje u nekim veb projektima i da se veruje samo predvidivom i strogo kontrolisanom serveru. I, zaista, zašto biste uložili duplo više vremena da završite proizvod, a da onda provedete dodatno vreme u otklanjanju problema u različitim pregledačima?

Mnogo štošta se promenilo u godinama posle kraja prvog „rata“ zbog pregledača. Veliki broj događaja je ponovo oblikovao veb programiranje na pozitivan način. Ovo su neki od tih događaja:

- Kompanija „Microsoft“ je dobila „rat“ kada je uvela IE6, koji je tada bio najbolji pregledač, pa već godinama nije razvijala Internet Explorer. Ovo je omogućilo drugim kompanijama da unaprede svoje pregledače, pa, čak, i da prestignu mogućnosti IE pregledača.
- Programeri i kompanije koje su razvile pregledače su jednako prihvatili napredak veb standarda. Naravno, programerima se nije dopadalo to što kod moraju da pišu dva (ili više) puta da bi ga prilagodili različitim pregledačima, pa im se, stoga, dopala ideja da se postigne dogovor o standardima koje bi svi programeri pratili.
- Programeri i tehnologije su „sazreli“ i sve više ljudi je počelo da obraća pažnju na upotrebljivost, napredne tehnike, poboljšanja i pristupačnost. Alatke poput Firebuga omogućile su programerima da postanu produktivniji i da pojednostave programiranje.

U ovom zdravijem okruženju programeri su počeli da otkrivaju nove i bolje načine za upotrebu alatki koje su im bile dostupne. Nakon objave aplikacija Gmail i Google Maps, koje su bogate programiranjem na strani klijenta, postalo je jasno da je JavaScript svestran, na određeni način jedinstven i moćan jezik prototipskog objektno-orijentisanog programiranja. Najbolji primer ponovnog „otkrića“ JavaScripta je široko prihvaćena funkcija koju omogućava objekat `XMLHttpRequest`, koji je ranije bio samo inovacija pregladača IE, ali je, zatim, implementiran u većini drugih pregledača. Objekat `XMLHttpRequest` omogućava JavaScriptu da šalje HTTP zahteve i da dobija najnoviji sadržaj sa servera da bi neki delovi stranice mogli da se ažuriraju bez potpunog učitavanja stranice. Zbog raspostranjene upotrebe objekta `XMLHttpRequest`, nastala je nova vrsta veb aplikacija nalik na desktop aplikacije, pod nazivom Ajax.

Sadašnjost

Zanimljivo je da se za pokretanje JavaScripta uvek koristi okruženje hosta. Veb pregledač je samo jedan od dostupnih hostova. JavaScript može, takođe, da funkcioniše na serveru, na desktopu i na mobilnim uređajima. U današnje vreme možete da koristite JavaScript da biste:

- kreirali bogate i moćne veb aplikacije (vrste aplikacija koje funkcionišu u veb pregledaču). Zahvaljujući dodacima za jezik HTML5, kao što su keš memorija aplikacije, klijentska memorija i baze podataka, programiranje pregledača postaje sve moćnije, kako za aplikacije na mreži, tako i za aplikacije van mreže. Moćni dodaci za Chrome WebKit, takođe, sadrže podršku za radne skriptove i prosleđena obaveštenja pregledača.

- napisali kod na strani klijenta pomoću alatke Node.js, kao i kod koji može da funkcioniše pomoću Rhinoa (JavaScript virtuelne mašine koja je napisana u Java jeziku)
- napravili mobilne aplikacije. Možete da kreirate aplikacije za iPhone, Android i druge telefone i tablične računare u JavaScriptu pomoću alatke **PhoneGap** ili **Titanium**. Osim toga, aplikacije za Firefox OS za mobilne telefone su u potpunosti napisane u JavaScriptu, HTML-u i CSS-u. React Native kompanije „Fejsbuk“ predstavlja novi, uzbudljivi način za razvoj izvornih iOS, Android i Windows (Experimental) aplikacija pomoću JavaScripta.
- kreirali bogate multimedijalne aplikacije, kao što su Flash ili Flex, pomoću ActionScripta koji se zasniva na ECMAScriptu
- napisali alatke za komandnu liniju i skriptove koji automatizuju administrativne zadatke na vašem desktopu pomoću okruženja **Windows Scripting Host (WSH)** ili WebKit **JavaScriptCore,** koje je dostupno na svim Mac računarima
- napisali ekstenzije i pluginove za veliki broj desktop aplikacija, kao što su Dreamweaver i Photoshop, i za mnoge druge pregledače
- kreirali višeplatformske desktop aplikacije pomoću Mozilla alatki **XULRunner** i **Electron**. Electron se koristi za izradu nekih najpopularnijih aplikacija na desktopu, kao što su Slack, Atom i Visual Studio Code. Sa druge strane, kompajler **Emscripten** omogućava da se kod koji je napisan u jeziku C/C++ kompajlira u format `asm.js` da bi, zatim, mogao da se koristi u pregledaču.
- testirali okruženja, kao što je **PhantomJS**, koja su programirana pomoću JavaScripta.
- Korišćenje JavaScript jezika je započeto unutar veb stranica, ali danas možemo sigurno reći da je on prisutan svuda. Osim toga, kompanije koje su razvile pregledače sada koriste brzinu kao konkurentsku prednost i utrkuju se u kreiranju najbržih JavaScript virtuelnih mašina, što je dobro i za korisnike i programere, jer se time otvaraju „vrata“ čak i još moćnijoj upotrebi JavaScripta u novim oblastima, kao što su obrada slika, obrada audio i video zapisa i razvoj igara.

Budućnost

Možemo samo da nagađamo kakva će biti budućnost, ali prilično je sigurno da će u njoj biti JavaScript. Ovaj jezik je možda potcenjen i nije dovoljno iskorišćen (ili možda je previše korišćen na pogrešne načine), ali svakodnevno smo svedoci pojave novih JavaScript aplikacija koje su sve zanimljivije i kreativnije. Na početku se JavaScript sastojao samo od jednostrukih linija, koje su ugrađivane u HTML atribute tagova, kao što je `onclick`. Sada programeri prave sofisticirane, dobro dizajnirane i strukturirane proširive aplikacije i biblioteke, koje podržavaju više platformi pomoću jednog osnovnog koda. JavaScript jezik

je zaista shvaćen ozbiljno i programeri počinju ponovo da otkrivaju njegove jedinstvene funkcije u kojima sve više uživaju.

U današnje vreme je poznavanje JavaScripta često odlučujući faktor prilikom zapošljavanja veb programera. Na intervjuu za posao često ćete čuti pitanja „Da li je JavaScript objektno-orijentisan jezik?“ i „Kako se nasleđivanje implementira u JavaScriptu?“. Nakon što pročitate ovu knjigu, bićete spremni za intervju za posao JavaScript programera i impresioniraćete poslodavce nekim informacijama koje, možda, nisu znali.

ECMASCRIPT 5

Poslednja najvažnija prekretnica u revizijama ECMAScripta je uvođenje standarda **ECM Script 5 (ES5)**, koji je zvanično prihvaćen u decembru 2009. godine. Standard ECMAScript 5 je implementiran i podržan na svim većim pregledačima i u serverskim tehnologijama.

Standard ES5 je predstavljao veliku reviziju, jer je pored nekoliko važnih sintaksnih promena i dodataka u standardnim bibliotekama, uveo i nekoliko novih jezičkih konstrukcija.

Na primer, standard ES5 je uveo neke nove objekte i svojstva, ali i takozvani režim **strict**, podskup jezika koji isključuje zastarele funkcije. Režim strict je opcioni i nije neophodan. To znači da ako želite da vaš kod funkcioniše u tom režimu, deklarisaćete ono što želite da uradite (jednom za svaku funkciju ili jednom za ceo program) pomoću sledećeg stringa:

```
"use strict";
```

Ovo je samo JavaScript string i neće predstavljati problem ako stringovi nisu dodeljeni ni jednoj promenljivoj. Kao rezultat toga, stariji pregledači koji ne prepoznaju jezik ES5 će jednostavno zanemariti taj string, pa je ovaj režim strict kompatibilan sa prethodnim verzijama i ne može naškoditi starijim pregledačima.

Svi primeri iz ove knjige funkcionišu u jeziku ES3, radi kompatibilnosti sa prethodnim verzijama pregledača, ali svi kodovi su napisani tako da će bez ikakvih problema funkcionisati i u režimu strict jezika ES5. Osim toga, svi delovi jezika ES5 će biti jasno označeni. Dodatak C, *Ugrađeni objekti*, sadrži detaljan opis novih dopuna u jeziku ES5.

Režim strict u jeziku ES6

Režim strict nije neophodan u jeziku ES5, ali ES6 moduli i klase su podrazumevano u režimu strict. Kao što ćete ubrzo videti, veći deo koda koji pišemo u jeziku ES6 se nalazi u modulu, zbog čega se režim strict podrazumevano primenjuje. Međutim, važno je da razumemo da u svim drugim konstruktorima ne postoji implicitna primena režima strict. Bilo je pokušaja da se naprave novije konstrukcije, kao što su arrow i generator funkcije, da bi bio primenjen režim strict, ali se kasnije došlo do zaključka da bi te nove konstrukcije dovele do veoma podeljenih jezičkih pravila i kodova.

ECMASCRIPT 6

Bilo je potrebno dugo vremena da se ECMAScript 6 revizija završi i konačno je prihvaćena 17. juna 2015. godine. ES6 funkcije polako postaju deo većine pregledača i serverskih tehnologija. Za kompajliranje jezika ES6 u jezik ES5 možete da koristite transpajlere, a možete da koristite kod i u okruženjima koja još uvek ne podržavaju ES6 u potpunosti (transpajlere ćemo razmatrati kasnije).

Standard ES6 je značajno unapredio JavaScript jezik i uvodi veoma zanimljive sintaksne promene i jezičke konstruktore.Uopšteno govoreći, postoje dve vrste osnovnih promena u ovoj reviziji ECMAScripta:

- poboljšana sintaksa postojećih funkcija i izdanja standardne biblioteke – na primer, klase i promisi
- nove jezičke funkcije – na primer, generatori

ECS6 omogućava da razmišljamo o kodu na drugačiji način. Nove sintaksne promene omogućavaju da napišemo kod koji je jasniji, lakši za održavanje i za koji nisu potrebni neki posebni trikovi. Sam jezik sada podržava nekoliko konstruktora za koje su ranije bili potrebni posebni moduli. Jezičke promene koje su uvedene u standardu ES6 su zaista izmenile način na koji pišemo kod u JavaScriptu.

Napomene u nomenklaturi o jezicima ECMAScript 6, ES6 i ECMAScript 2015 su iste, ali se koriste naizmenično.

Podrška za jezik ES6 u pregledačima

Većina pregledača i serverskih okruženja su „na putu“ ka implementaciji funkcija jezika ES6. Možete da pogledate koje su funkcije podržane tako što ćete posetiti adresu `http://kangax.github.io/c ompat-table/es6/.`

Iako jezik ES6 nije potpuno podržan u svim pregledačima i serverskim okruženjima, možemo da počnemo da koristimo sve funkcije jezika ES6 pomoću **transpajlera.** Transpajleri su source-to-source (izvor-u-izvor) kompajleri. ES6 transpajleri omogućavaju da napišete kod u ES6 sintaksi i da ga kompajlirate/pretvorite u ekvivalentnu ES5 sintaksu, koja može, zatim, da se koristi u pregledačima koji ne podržavaju sve ES6 funkcije.

U stvari, ES6 transpajler koji je dostupan u ovom trenutku je Babel. U ovoj knjizi ćemo ga koristiti za pisanje i testiranje naših primera.

Babel

Alatka Babel podržava skoro sve uobičajene ES6 funkcije i funkcije sa prilagođenim plaginovima. Može da se koristi u velikom broju sistema za izradu aplikacija, okruženja i jezika, kao i u šablonskim sistemima (template engine), a sadrži i dobru komandnu liniju i ugrađenu **petlju čitanja, evaluacije i ispisivanja (REPL – read-eval-print loop)**.

Da biste shvatili kako Babel transpajlira ES6 kod u ekvivalentni ES6 oblik, pređite na alatku Babel REPL (`http://babeljs.io/repl/`).

Babel REPL omogućava da brzo testirate kratke isečke koda jezika ES6. Kada otvorite alatku Babel u pregledaču, videćete deo ES6 koda koji je u njoj podrazumevano smešten. Uklonite kod u levom oknu i ukucajte sledeći tekst:

```
var name = "John", mood = "happy";
console.log(`Hey ${name}, are you feeling ${mood} today?`)
```

Kada unesete ovaj tekst i zatvorite levu stranu okna, videćete da REPL transpajlira ovaj ES6 kod u sledeći kod:

```
"use strict";
var name = "John",
  mood = "happy";
console.log("Hey " + name + ",
  are you feeling " + mood + " today?");
```

Ovaj kod je ekvivalent kodu jezika ES5 koji smo ranije napisali u levom oknu. Možete da vidite da je kod koji je dobijen kompajliranjem u desnom oknu poznati jezik ES5. Kao što smo rekli, Babel REPL je dobra alatka u kojoj možete eksperimentisati sa različitim ES6 konstruktorima. Međutim, treba da pomoću Babela automatski transpajliramo ES6 kod u ES5, pa ćemo, zato, Babel da uključimo u sisteme za izradu aplikacija ili u okruženja.

Prvo ćemo instalirati Babel kao alatku za komandnu liniju. Pretpostavićemo da poznajete alatke Node i **Node Package Manager (npm)**. Instaliranje Babela pomoću alatke `npm` je lako. Prvo ćemo kreirati direktorijum u kojem ćemo instalirati Babel kao modul i ostatak izvornog koda. Na Mac računaru ću upotrebiti sledeće komande za kreiranje direktorijuma pod nazivom `babel_test` – inicijalizovaću projekat pomoću komande `npm init` i instaliraću Babel komandnu liniju pomoću komande `npm`:

```
mkdir babel_test
cd babel_test && npm init

npm install --save-dev babel-cli
```

Ako poznajete `npm`, možda ćete pokušati da instalirate Babel globalno. Međutim, instaliranje Babela kao globalnog modula nije baš dobra ideja. Nakon instaliranja Babela u vaš projekat, datoteka `package.json` će izgledati slično kao sledeći blok koda:

```
{
  "name": "babel_test",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo "Error: no test specified" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "babel-cli": "^6.10.1"
  }
}
```

U prethodnom bloku koda možete da vidite skript devDependencies, koji je kreiran u Babel verziji 6.10.1. Da biste transpajlovali kod, možete aktivirati Babel iz komandne linije ili kao deo koraka za izradu projekta. Za svaki teži posao moraćete ponovo da aktivirate Babel. Da biste aktivirali Babel kao deo koraka za izradu projekta, možete da korak za izradu, koji će aktivirati Babel, dodate unutar taga skripta u datoteci `package.json`. Na primer:

```
"scripts": {
  "build": "babel src –d lib"
},
```

Kada koristite `npm` kao alatku za izradu aplikacija, Babel će biti aktiviran na vašem `src` direktorijumu i transpajlirani kod će biti smešten unutar `lib` direktorijuma. Druga mogućnost je da ručno pokrenete Babel, tako što ćete napisati sledeću komandu:

```
$ ./node_modules/.bin/babel src –d lib
```

Različite opcije Babela i plaginove ćemo razmatrati kasnije u knjizi. Ovaj odeljak će vam pomoći da počnete da istražujete jezik ES6.

OBJEKTNO-ORIJENTISANO PROGRAMIRANJE

Pre nego što „zaronimo“ u JavaScript, razmotrićemo šta znači termin objektno-orijentisan i koje su glavne funkcije objektno-orijentisanog stila programiranja. Ovo je lista koncepata koji se najčešće koriste u **objektno-orijentisanom progamiranju (OOP)**:

- objekat, metod i svojstvo
- klasa
- enkapsulacija
- agregacija
- ponovna upotrebljivost/nasleđivanje
- polimorfizam

Pogledajte malo bolje svaki od ovih koncepata. Ako ste početnik u objektno-orijenitisanom programiranju, ovi koncepti mogu vam izgledati previše teoretski i možete imati poteškoća da ih odmah shvatite ili zapamtite. Ne brinite - potrebno je da objašnjenja pročitate nekoliko puta, a tema može biti i malo „suvoparna“ na konceptualnom nivou. Međutim, razmotrićemo mnogo primera koda u nastavku knjige i videćete da je objektno-orijentisano programiranje mnogo jednostavnije u praksi.

Objekti

Kao što se može zaključiti iz samog naziva objektno-orijentisan, objekti su važni. Objekat predstavlja nešto ili nekoga, a prikaz objekta se izražava pomoću programskog jezika. Objekat može da bude bilo šta – objekat iz stvarnog života ili malo komplikovaniji koncept. Na primer, videćete da uobičajeni objekat, kao što je mačka, ima određene karakteristike – boju, ime, težinu i tako dalje, i može da obavlja neke radnje – da mijauče, da spava, da se skriva, da beži i tako dalje. Karakteristike objekta se nazivaju svojstva u objektno-orijentisanom programiranju, a radnje se nazivaju metodi.

Analogija sa govornim jezikom je sledeća:

- Objekti su najčešće imenice, kao što su knjiga, osoba i tako dalje.
- Metodi su glagoli - na primer, čitati, trčati i tako dalje.
- Vrednosti svojstava su pridevi.

Uzmimo kao primer rečenicu: „Crna mačka spava na otiraču“. „Mačka“ (imenica) je objekat, „crna“ (pridev) je vrednost svojstva boje i „spava“ (glagol) je radnja ili metod u OOP-u. Radi analogije, možemo ići korak dalje i reći da „na otiraču“ označava nešto što je povezano sa radnjom „spavati“, pa se „otirač“ ponaša kao parametar koji je prosleđen metodu `sleep`.

Klase

U stvarnom životu slični objekti se mogu grupisati na osnovu nekih kriterijuma. Kolibri i orao su ptice, pa se mogu klasifikovati kao da pripadaju nekoj izmišljenoj klasi `Birds`. U OOP-u klasa je nacrt ili recept za objekat. Drugi naziv za objekat je instanca, tako da možemo reći da je orao stvarna instanca opšte klase `Birds`. Možete da kreirate različite objekte, tako što ćete koristiti istu klasu, zato što je klasa samo šablon, dok su objekti stvarne instance koje se zasnivaju na šablonu.

Postoji razlika između JavaScripta i klasičnih OO jezika, kao što su C++ i Java. Treba odmah na početku da znate da u JavaScriptu ne postoje klase, već se svi elementi zasnivaju na objektima. U JavaScriptu se koriste prototipovi, koji su, takođe, objekti (o njima će biti reči kasnije). U klasičnom OO jeziku reći ćete: „Kreiraj novi objekat pod nazivom `Bob`, koji pripada klasi `Person`.“, a u prototipskom OO jeziku: „Uzeću objekat pod nazivom „Bob’s dad“ koji leži (na kauču ispred televizora) i ponovo ga upotrebiti kao prototip za novi objekat koji ću nazvati `Bob`.“

Enkapsulacija

Enkapsulacija je još jedan koncept povezan sa objektno-orijentisanim programiranjem, koji ukazuje da objekat sadrži (enkapsulira):

- podatke (koji su uskladišteni u svojstvima)
- načine na koje će biti iskorišćeni podaci (pomoću metoda)

Još jedan pojam koji je povezan sa enkapsulacijom je „sakrivanje informacija“. Ovo je širok pojam i može da znači mnogo štošta, ali hajde da vidimo šta znači kada se koristi u objektno-orijentisanom programiranju.

Zamislite jedan objekat – na primer, MP3 plejer. Kao korisnik objekta, možete da koristite interfejs, kao što su tasteri, ekran i tako dalje. Upotrebićete interfejs da bi objekat uradio nešto korisno za vas, kao što je reprodukovanje pesme. Vi ne znate kako uređaj funkcioniše u unutrašnjosti, a najčešće vas to i ne zanima. Drugim rečima, implementacija interfejsa je sakrivena od vas. Isto se dešava u OOP-u kada kod koristi objekat pozivanjem metoda. Bez obzira da li ste sami napisali kod objekta ili ste ga dobili iz neke nezavisne biblioteke, vaš kod ne mora da „zna“ kako metodi funkcionišu interno. U kompajliranom jeziku možete, u stvari, da pročitate kod pomoću kojeg objekat funkcioniše. U JavaScriptu možete da vidite kod, zato što je to interpetirani jezik, ali koncept je i dalje isti – koristite interfejs objekta, a da, pri tom, ne brinete o implementaciji.

Drugi aspekt sakrivanja informacija je vidljivost metoda i svojstava. U nekim jezicima objekat može da ima `public`, `private` i `protected` metode i svojstva. Ova kategorizacija definiše nivo pristupa koji korisnici objekta mogu imati. Na primer, samo metodi istog objekta imaju pristup metodima `private`, dok svako može pristupiti metodima `public`. U JavaScriptu svi metodi i svojstva su javni, ali videćete da postoje načini za zaštitu podataka unutar objekta i ostvarivanja privatnosti.

Agregacija

Spajanje nekoliko objekata u jedan objekat je poznato kao agregacija ili kompozicija. To predstavlja moćan način za deljenje problema na manje delove ili na delove kojima se može upravljati. Kada je opseg problema toliko složen da je o njemu nemoguće razmišljati na detaljnom nivou, možete problem da podelite na nekoliko manjih oblasti i onda svaku oblast na još manje delove. To vam omogućava da razmišljate o problemu na nekoliko nivoa apstrakcije.

Na primer, lični računar je kompleksan objekt. Ne možete razmišljati o svemu onome što može da se desi kada ga uključite. Međutim, možete da apstraktujete problem, tako što ćete inicijalizovati sve razdvojene objekte od kojih se sastoji objekat `Computer` – objekte `Monitor`, `Mouse`, `Keyboard` i tako dalje. Sklapanjem delova koji se mogu ponovo koristiti možete sastaviti kompleksne objekte.

Druga mogućnost je da u objekat `Book` smestite (spojite) jedan ili više objekata `Author`, objekat `Publisher`, nekoliko objekata `Chapter`, `TOC` (tabelu sadržaja) i tako dalje.

Nasleđivanje

Nasleđivanje predstavlja elegantan način za ponovnu upotrebu postojećeg koda. Na primer, možete koristiti generički objekat `Person`, koji ima svojstva, kao što su `name` i `date_of_birth` `npm`, i koji, takođe, sadrži funkcije `walk`, `talk`, `sleep` i `eat`. Zatim će vam biti potreban još jedan objekat pod nazivom `Programmer`. Možete ponovo implementirati sve metode i svojstva koji imaju objekat `Person`, ali pametnije je da upotrebite nasleđivanje - objekat `Programmer` nasleđuje objekat `Person` i imaćete manje posla. Objekat `Programmer` će ponovo koristiti sve funkcije objekta `Person` i tako implementirati specifičnije funkcije, kao što je metod `writeCode`.

U klasičnom OOP-u klase se nasleđuju od drugih klasa, ali u JavaScriptu ne postoje klase, pa se objekti nasleđuju od drugih objekata.

Kada se objekat nasleđuje od drugog objekta, u nasleđene objekte se, obično, dodaju novi metodi, pa se tako stari objekat proširuje. Često se sledeće fraze mogu koristiti naizmenično – B nasleđuje od A i B proširuje A. Osim toga, objekat koji nasleđuje može da izabere jedan ili dva metoda, koja može da redefiniše i prilagodi svojim potrebama. Stoga, interfejs i naziv metoda se ne menjaju, ali kada se poziva na novi objekat, metod se ponaša drugačije. Redefinisanje načina rada metoda poznato je kao **izmena**.

Polimorfizam

U prethodnom primeru objekat `Programmer` je nasledio sve metode podređenog objekta `Person`. To znači da oba objekta sadrže metod `talk`. Sada zamislite da negde u vašem kodu postoji promenljiva pod nazivom `Bob`, ali da vi ne znate da li je `Bob` objekat `Person` ili objekat `Programmer`. Još uvek možete da pozovete metod `talk` na objekat `Bob` i kod će funkcionisati. Ova mogućnost pozivanja istog metoda na različite objekte koji reaguju na sebi svojstven način naziva se polimorfizam.

REZIME OOP-A

Evo kratke tabele koja sadrži rezime koncepata koje smo do sada razmatrali:

FUNKCIJA	PRIKAZ KONCEPTA
Bob je čovek (objekat).	objekat
Bobov datum rođenja je 1. jun 1980. godine, pol – muško i kosa – crna.	svojstva
Bob može da jede, spava, pije, sanja, razgovara i da izračuna svoje godine.	metodi
Bob je instanca klase `Programmer`.	klasa (u klasičnom OOP-u)
Bob se zasniva na drugom objektu pod nazivom `Programmer`.	prototip (u prototipskom OOP-u)
Bob sadrži podatke, kao što je `birth_date`, i metode koje funkcionišu pomoću podataka, kao što je `calculateAge()`.	enkapsulacija
Ne morate da znate kako metod izračunavanja funkcioniše interno. Objekat može da sadrži neke podatke koji nisu javni, kao što je broj dana u februaru prestupne godine. Ne znate, niti želite da znate te podatke.	sakrivanje informacija
Bob je deo objekta `WebDevTeam` sa Jillom, objektom `Designer` i Jackom, objektom `ProjectManager`.	agregacija i kompozicija

`Designer`, `ProjectManager` i `Programmer` su zasnovani na objektu `Person` koji proširuju.	nasleđivanje
Možete da pozovete metode `Bob.talk()`, `Jill.talk()` i `Jack.talk()` i oni će funkcionisati kako treba, iako će dati različite rezultate. Bob će, verovatno, više pričati o performansama, Jill o lepoti, a Jack o rokovima. Svaki objekat je nasledio metod `talk` od objekta `Person` i prilagodio ga.	polimorfizam i izmena metoda

PODEŠAVANJE OKRUŽENJA ZA TESTIRANJE

Ova knjiga se zasniva na principu „uradi sam“ kada je reč o pisanju koda, zato što čvrsto verujem da ćete programski jezik zaista najbolje naučiti kada pišete kod. Ne postoje preuzimanja koda koja možete samo kopirati i nalepiti na vaše stranice. Naprotiv, od vas se očekuje da ukucate kod, da vidite kako on funkcioniše i zatim da ga prilagodite i uvežbate. Kada isprobavate primere koda, preporučuje se da kod unesete u JavaScript konzolu. Sada ćemo pogledati kako se unosi kod.

Kao programer, verovatno ste već instalirali nekoliko pregledača na vaš sistem, kao što su Firefox, Safari, Chrome ili Internet Explorer. Svi moderni pregledači sadrže funkciju JavaScript Console, koju ćete, čitajući ovu knjigu, koristiti kao pomoć u učenju i eksperimentisanju sa jezikom JavaScript. Preciznije rečeno, u ovoj knjizi se koristi WebKit konzola, koja je dostupna u pregledačima Safari i Chrome, ali primeri bi trebalo da funkcionišu u svim drugim konzolama.

WebKit funkcija Web Inspector

U ovom primeru je prikazano kako da koristite konzolu da biste ukucali kod koji će zameniti logotip na početnoj stranici google.com slikom po vašem izboru. Kao što vidite, možete uživo da testirate svoj JavaScript kod na bilo kojoj stranici:

Da bi konzola bila prikazana u pregledaču Chrome ili Safari, kliknite desnim tasterom miša bilo gde na stranici i izaberite **Inspect Element**. Dodatni prozor koji će biti prikazan predstavlja funkciju Web Inspector. Izaberite karticu **Console** i spremni ste da unesete kod.

Ukucajte kod direktno u konzolu i kada pritisnete taster *Enter*, kod će biti izvršen. Vraćena vrednost koda se ispisuje u konzolu. Kod se izvršava u trenutno učitanoj stranici, pa, na primer, kada ukucate `location.href`, vratiće se URL adresa aktuelne stranice.

Konzola sadrži, takođe, funkciju automatskog popunjavanja. Ova funkcija se koristi na isti način kao uobičajeni zahtev za unos u komandnu liniju u vašem operativnom sistemu ili kao funkcija automatskog popunjavanja u kompletno integrisanim razvojnim okruženjima. Ako, na primer, ukucate `docu` i pritisnete taster *Tab* ili taster sa strelicom nadesno, `docu` će biti automatski dodat u dokument. Zatim, ako ukucate . (tačku), možete da obavite iteraciju svih dostupnih svojstava i metoda koje možete da pozovete na objekat `document`.

Pomoću tastera sa strelicom nagore i nadole možete da se krećete kroz listu komandi koje su izvršene i da prikažete te komande u konzoli.

Konzola sadrži samo jednu liniju za unos, ali možete da izvršite nekoliko JavaScript iskaza, tako što ćete ih odvojiti znakom tačka-zarez. Ako vam treba više linija, možete da pritisnete *Shift* + *Enter* da biste prešli na novu liniju, a da, pri tom, ne morate još uvek da dobijete rezultat.

JavaScriptCore na Mac računaru

Na Mac računaru vam, u stvari, nije potreban pregledač - možete da istražujete JavaScript direktno iz aplikacije **Terminal** komandne linije.

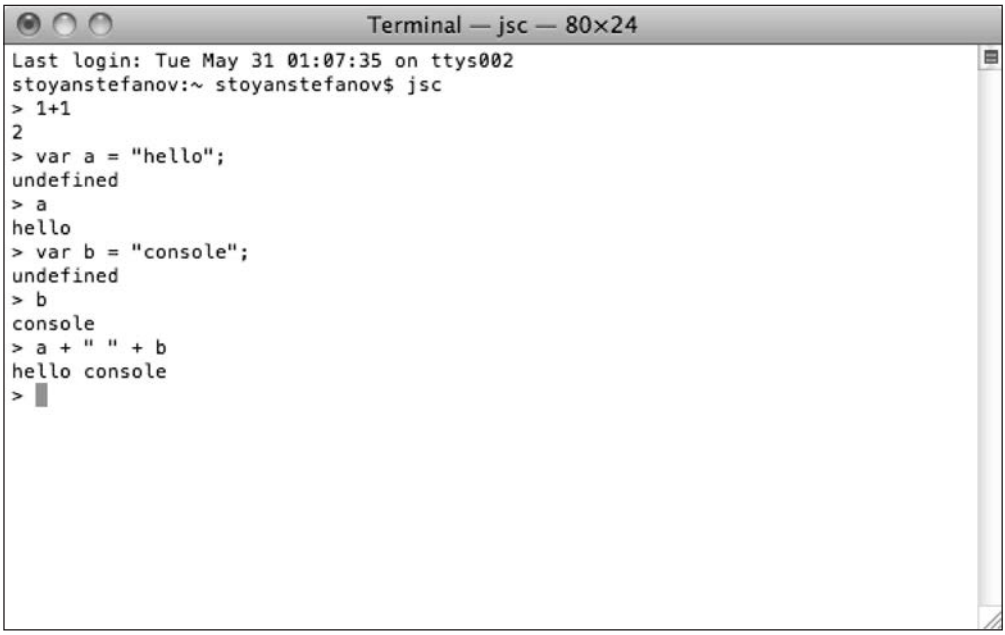
Ako nikada ranije niste koristili aplikaciju **Terminal**, možete jednostavno da je pronađete pomoću opcije **Spotlight search**. Kada pokrenete **Terminal,** unesite sledeće komandu:

```
alias jsc='/System/Library/Frameworks/JavaScriptCore.framework/
  Versions/Current/Resources/jsc'
```

Ova komanda će skratiti naziv male aplikacije JavaScriptCore, koja je deo WebKit virtuelne mašine, u `jsc`. JavaScriptCore se isporučuje sa operativnom sistemom Mac.

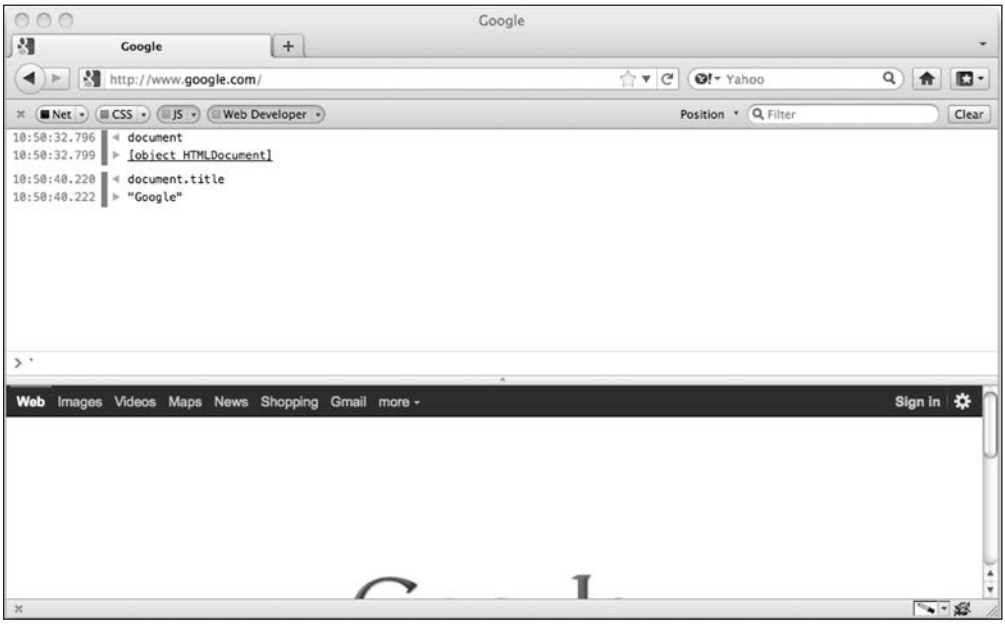
Možete da dodate liniju sa skraćenim nazivom, koja je prethodno prikazana, u datoteku `~/.profile`, tako da vam `jsc` uvek bude pri ruci kada je potrebno.

Da biste pokrenuli interaktivno komandno okruženje, sada ćete jednostavno uneti `jsc` iz bilo kog direktorijuma. Zatim, možete uneti JavaScript iskaze, a kada pritisnete Enter, videćete rezultat iskaza. Pogledajte sledeći snimak ekrana:



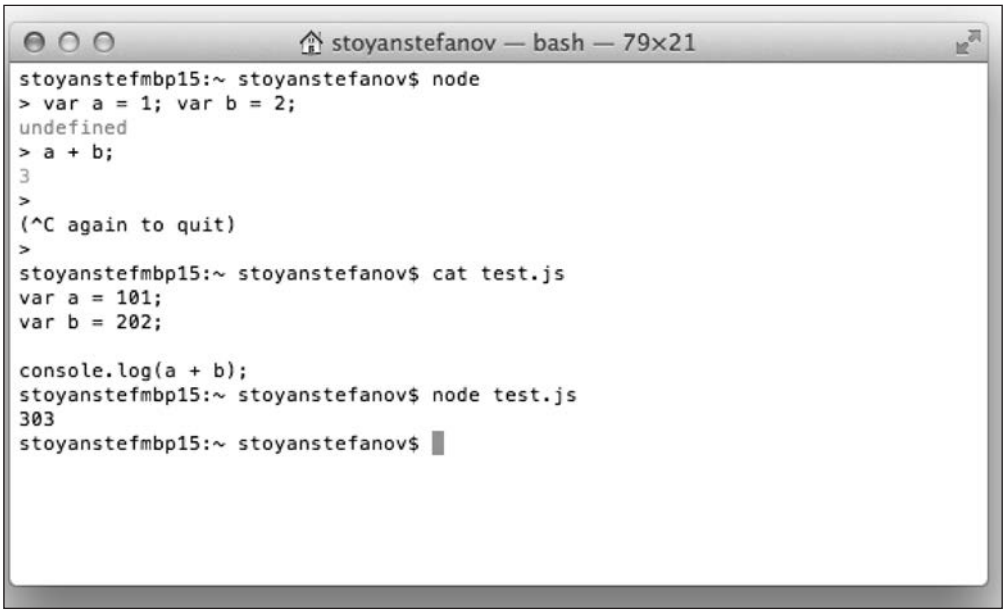
Druge konzole

Svi moderni pregledači imaju ugrađene konzole. Ranije ste videli konzolu Chrome /Safari. U svakoj verziji Firefoxa možete da instalirate Firebug ekstenziju, koja sadrži konzolu. Osim toga, u novijim Firefox verzijama postoji ugrađena konzola, koja je dostupna u meniju **Tools | Web Developer | Web Console.**



Pregledač Internet Explorer, još od verzije 8, ima F12 Developers Tools funkciju, koja sadrži konzolu u kartici **Script.**

Dobro bi bilo da upoznate alatku `Node.js`, a to možete učiniti tako što ćete isprobati njenu konzolu. Instalirajte alatku `Node.js` sa adrese http://nodejs.org i isprobajte konzolu u komandnoj liniji (terminalu):



Kao što vidite, možete da upotrebite `Node.js` konzolu da biste isprobali kratke primere. Međutim, možete, takođe, da pišete duže komandne skriptove (kao što je `test.js` na snimku ekrana) i da ih pokrenete u čvoru `scriptname.js`.

Node REPL je moćna programerska alatka. Kada ukucate „node“ u komandnu liniju, aktiviraćete petlju REPL. Možete da isprobate JavaScript u ovoj petlji REPL:

```
node
> console.log("Hellow World");
Hellow World
undefined
> a=10, b=10;
10
> console.log(a*b);
100 undefined
```

REZIME

U ovom poglavlju ste naučili kako je nastao jezik JavaScript i koliko je napredovao do danas. Takođe smo predstavili koncepte objektno-orijentisanog programiranja i videli ste kako se koristi okruženje za testiranje – JavaScript konzola. Sada ste spremni da „zaronite“ u JavaScript i da naučite kako se koriste moćne OO funkcije. Međutim, počnimo od početka.

U sledećem poglavlju ćemo razmatrati tipove podataka u JavaScriptu (postoji samo nekoliko tipova) – uslove, petlje i nizove. Ako smatrate da znate ove tipove podataka, slobodno preskočite sledeće poglavlje, ali prvo se uverite da možete da uradite nekoliko kratkih vežbi na kraju poglavlja.