Sadržaj ukratko

Uvod Zašto sam napisao ovu knjigu 1

1 „Hello, World!” pisanje prvog programa 5

2 Kako softver radi 23

3 Upoznavanje računara 47

4 Alatke za izgradnju 63

5 Tipovi podataka, strukture podataka, baze podataka 83

6 Regularni izrazi 109

7 if, for, while i when 131

8 Funkcije i metodi 153

9 Programski standardi 173

10 Dokumentacija 185

11 Planiranje 199

12 Testiranje i otklanjanje grešaka 209

13 Naučite da pecate: kako da steknete znanje programiranja za ceo život 233

14 Gradite svoje veštine 243

15 Napredne teme 255

Glosar 273

Indeks 293

Sadržaj

Uvod Zašto sam napisao ovu knjigu 1
Zašto bi trebalo da pročitate ovu knjigu 3
Vaš projekat 3

1 „Hello, World!" pisanje prvog programa 5
Biranje tekst editora 5
Osnovne funkcije 6
Biranje 8
Sublime Text 9
TextMate 9
Notepad++ 9
Gedit 9
Vim 10
Eclipse 10
IntelliJ 11
Xcode 11
Visual Studio 11
Napravite direktorijum za projekat 12
Počnite od malog: napravite fajl za testiranje 12
Kako HTML i JavaScript sarađuju u pretraživaču 13
Značaj malih promena 15
Gradite na svom uspehu 17
Referencirajte svoj JavaScript u fajlu manifest.json 20
Neka radi! 20
Veća moć, veća odgovornost 21
Rezime 21

2 Kako softver radi 23
Šta je to „softver"? 23
Životni ciklus softvera 24
Izvorni kôd – tu sve počinje 25
Skup instrukcija 25
Programski jezici 26
Od izvornog koda do nula i jedinica 31

Prevedeni i interpretirani jezici: kada izvorni kôd postaje binaran? 31
Izvršno okruženje 32
Izvršavanje u procesoru 34
Ulaz i izlaz 34
Korisniji softver (za višekratnu upotrebu), pomoću ulaza 34
Odakle ulaz stiže? 35
Kako softver pribavlja ulaz 36
Vrste izlaza 36
Đubre ušlo, đubre izašlo 37
Stanje 38
Dodavanje stanja u Kittenbook 39
Memorija i promenljive 41
Promenljive 41
Skladištenje promenljive 42
Konačan resurs 44
Curenje memorije 44
Rezime 45

3 Upoznavanje računara 47
Vaš računar je glup 47
Vaš računar je čaroban 48
Stajanje na ramenima džinova 48
Unutrašnjost računara 48
Procesor 48
Kratkoročna memorija 49
Dugoročna memorija 49
Korišćenje računara 50
Sistem fajlova 50
Komandna linija: preuzimanje kontrole 52
Rezime 62

4 Alatke za izgradnju 63
Automatizujte (skoro) sve 63
Instalirajte Node 64
Instalirajte Grunt 66
Softver koji pomaže da se pravi softver 69

Izbegavanje grešaka 70
Raditi brže 70
Zadaci koje treba automatizovati 71
Prevođenje 71
Testiranje 71
Pakovanje 72
Postavljanje 72
Izgradite svoju vlastitu izgradnju 73
Gruntfile.js 73
Dodatni moduli za Grunt 73
Učitavanje dodatnih modula za Grunt 76
Registrovanje zadataka 76
Pazite ovo! 78
Rezime 80

5 Tipovi podataka, strukture podataka, baze podataka 83
Tipovi podataka 83
Zašto postoje različiti tipovi podataka 84
Osnovni tipovi podataka 84
Složeni tipovi podataka 89
Dinamički i statički tipizirani jezici 96
Strukture podataka 96
Skup 99
Stek 99
Stablo 100
Graf 101
Kako izabrati efikasnu strukturu podataka 104
Baze podataka 104
Dugotrajno (postojano) skladištenje 104
Relacione baze podataka 104
Kratak uvod u SQL 106
Rezime 107

6 Regularni izrazi 109
Ctrl+F na steroidima: traženje uzoraka 109
Upotreba regularnih izraza u JavaScriptu 110
Ponavljanje 111

? 111
+ 111
* 112
Specijalni znakovi i izlazne sekvence 112
{1,10}: pravite vlastite super moći 113
Upariti bilo šta, tačka 114
Ne budite gramzivi 114
Uglaste zagrade od [A-Za-z] 115
Liste znakova 115
Rasponi 115
Negacija 116
Šablon za brojeve telefona 116
Treba mi \s 119
Prečice za uglaste zagrade 119
Ograničenja 121
Hvatanje oznake 124
Napredno traženje i zamena 125
Početak i kraj (reda) 126
Indikatori 126
Global 126
Ignore Case 126
Multiline 127
Da li ćete ikad koristiti Regex? 127
grep 127
Menjanje koda 127
Provera ispravnosti 128
Izdvajanje podataka 128
Rezime 129

7 if, for, while i when 131
Operatori 131
Operatori poređenja 131
Logički operatori 132
Unarni operatori 134
Binarni operatori 134
Ternarni operatori 136
Istinitost 139

„Sintaksne poslastice” 140
Petlja nad nizom 142
Petlja nad slikama 142
Ugnežđene petlje 143
Potreban vam je prekid 143
Beskonačne petlje 145
Još jedan prekid 146
Ako ne znate kada da stanete 146
When 147
Događaji 147
Osluškivači 147
Cron zadaci 148
Vremensko odlaganje 149
Catch kada nešto nije u redu 150
Pisanje robustnog koda 151
Rezime 151

8 Funkcije i metodi 153
Struktura funkcije 153
Definicija 154
Pozivanje 154
Argumenti 155
Stek poziva 157
Enkapsuliranje koda 158
Uradi jednu stvar kako treba 158
Podeli pa vladaj 159
Mesto za svaku stvar i svaka stvar na svom mestu 162
Višekratno korišćenje koda 163
Rešavanje opšteg problema 163
Uraditi više sa manje 163
Ne ponavljaj se 165
Opseg 166
Globalne promenljive 167
Lokalne promenljive 168
Kako se promenljive traže 168
Rezime 171

9 Programski standardi 173

Konvencije kodiranja 173

Određivanje standarda 174

Hakerisati ili ne hakerisati 174

Platiti na mostu ili na ćupriji 175

Pisanje održivog koda 175

Formatiranje koda 176

Neka bude konzistentno 177

Beline 178

Neće se desiti samo: uvedite pravila 178

Upotreba tuđeg rada 180

Radite brže 180

Softver otvorenog koda 181

Napravila zajednica 181

Kada da pravite sami 182

Oprobana praksa 182

Dokumentacija 182

Planiranje 183

Testiranje 183

Rezime 183

10 Dokumentacija 185

Dokumentovati nameru 186

Samodokumentovan kôd 186

Ne dokumentujte ono što je očigledno 189

Opasnost od zastarele dokumentacije 190

Pronalaženje grešaka uz pomoć dokumentacije 191

Dokumentujte za sebe 191

Koliko dobro pamtite? 191

Dokumentujte da biste učili 192

Dokumentacija van komentara 192

Dokumentacija za druge 196

Dokumentujte svoje odluke 196

Dokumentujte svoje resurse 197

Dokumentacija kao lekcija 197

Rezime 198

11 Planiranje 199

Razmišljajte pre nego što gradite 199
Napravite specifikaciju 200
Projektovanje arhitekture 200
Crtanje dijagrama 201
Pokušajte da oborite svoj sistem 202
Iterativno planiranje 203
Projektujte proširivanje 203
Koji su vaši prioriteti? 204
Doživljaj korisnika 204
Performanse 204
Bezbednost 205
Skalabilnost 205
Rokovi 205
Kompromisi 206
Utvrdite i postavite ograničenja 206
Saznajte šta možete i šta ne možete 206
Rezime 207

12 Testiranje i otklanjanje grešaka 209

Ručno testiranje 209
Testirajte dok radite 210
Probajte nešto ludo 210
Jedite vlastitu pseću hranu 211
Automatizovano testiranje 211
Jedinično testiranje 212
Napravite testove za kittenbook 215
Velika greška! 218
Špijuni kao mi 219
Integraciono testiranje 221
Hvatajte probleme što pre 222
Otklanjanje grešaka 222
Poruke o greškama 223
Evidencije 224
Tačke prekida 224
Inspekcija, osmatranje i konzola 228
Koračanje kroz kôd 229

Stek poziva 231
Pronađite osnovni uzrok 231
Kodirajte, testirajte, otklanjajte greške, pa opet 231
Rezime 232

13 Naučite da pecate: kako da steknete znanje programiranja za ceo život 233

Kako se traži 234
Biranje pravih izraza 235
Tražite unazad 236
Prepoznavanje kvalitetnih resursa 236
Lični blogovi: skrivene dragocenosti 237
Gde, kada i kako postavljati programerska pitanja 237
Gde 237
Kada 240
Kako 241
Učite tako da podučavate 241
Rezime 242

14 Gradite svoje veštine 243

Napravite vlastiti kittenbook 243
Preoblikujte Facebook 243
Dodajte novu funkcionalnost 244
Podelite vašu verziju Kittenbooka 245
Smislite vlastiti projekat 246
Rešite vlastiti problem 246
Budite ambiciozni 246
Tražite pomoć, pružajte pomoć 247
Projekti otvorenog koda 247
GitHub 247
Pronalaženje projekta 248
Različiti načini na koje možete da date svoj doprinos 248
Napravite vlastiti projekat 249
Besplatno onlajn obrazovanje 249
Project Euler 249
Udacity 250
Coursera 250

codecademy 251
Khan Academy 251
Obučavanje 251
Plaćeno obrazovanje 252
Pročitajte knjigu 252
Udacity i Coursera 252
Treehouse 253
Rezime 253

15 Napredne teme 255
Kontrola verzija 255
Zašto da se koristi kontrola verzija? 256
Rad u timu 257
Subversion 260
Git 260
OOP (objektno orijentisano programiranje) 265
Klase 266
Nasleđivanje 266
Primerci 267
Uzorci za projektovanje 268
Pub Sub 268
Mediator 269
Singleton 270
Rezime 270

Glosar 273

Indeks 293

1

„Hello, World!" pisanje prvog programa

Napomena

Projekat: Pravljenje proširenja za Google Chrome.

Dobro došli u svet programiranja! Ja smatram da je najbolji način da naučite kako se programira, da stvarno programirate (ali vi to već znate, pošto nikako niste propustili da pročitate uvod), zato ćemo u ovom poglavlju da napravimo program. Taj program će biti prvi deo većeg projekta koji ćemo da gradimo u celoj knjizi. Sednite pred svoj računar ako to već niste uradili – treba da pišete kôd!

Jedno malo upozorenje, dok niste počeli: u ovom poglavlju ćemo raditi nekoliko stvari koje nećete odmah razumeti. Raditi nešto što sasvim ne razumete predstavlja (bar za mene) veliki deo programiranja računara. Do kraja ove knjige, sve misterije će se razjasniti. Za sada, verujte mi.

Biranje tekst editora

Jedan od najvažnijih delova programiranja je pisanje koda (ali, kao što je opisano u izdvojenom tekstu, kodiranje i programiranje nisu sasvim iste stvari). Kada kažem *kôd*, mislim na instrukcije za računar napisane u nekom programskom jeziku koji računar može da razume. Kôd se piše u tekstualni fajl pomoću tekst editora. Vaš tekst editor je možda vaša najvažnija alatka (kao luk i strela Robina Huda, mač Ekskalibur kralja Artura ili glas Suzan Bojl). Imate mnogo tekst editora koje možete da izaberete, pa birajte mudro.

Programiranje i kodiranje

Razlika između programiranja i kodiranja je možda suptilna, ali je značajna. Kodiranje je samo jedan deo programiranja. U stvari, kodiranje je verovatno *najlakši* deo programiranja. Programiranje obuhvata i zadatke kao što je stvaranje okruženja u kojem vaš kôd može uspešno da se izvršava, organizovanje koda na logičan način, testiranje koda da bi se uočile greške, analiziranje koda da bi se utvrdilo zbog čega dolazi do tih grešaka, rad sa kodom i radnim okvirima koje su drugi napisali i pakovanje koda tako da mogu i drugi da ga koriste. Kada umete da programirate, kodiranje može da bude mnogo zabavnije.

Savet

Kad sam ja učio da programiram, moja prva greška je bio loš izbor tekst editora. Pre toga sam koristio jedino Microsoft Word da bih svoje reči i misli ubacio u računar, pa sam pokušao da programiram u Wordu. Nije dugo potrajalo dok nisam shvatio da to ne vredi. Zatim sam otvorio Notepad i koristio sam ga mesecima dok nisam otkrio neverovatne alternative koje postoje. Notepad ne sadrži nijednu od osnovnih funkcije koje bi trebalo da ima svaki dobar tekst editor. Učite na mojim greškama. Za ime Boga, nemojte da koristite Notepad za programiranje.

Osnovne funkcije

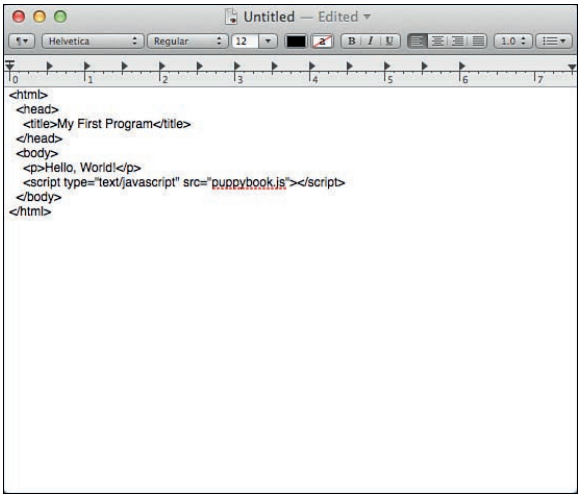
Nekoliko tekst editora je pravljeno upravo za kodiranje. U sledećem odeljku navodim nekoliko najpopularnijih tekst editora i njihove najvažnije osobine. Svi ovi editori imaju nekoliko osnovnih funkcija: monofont, isticanje sintakse, kompletiranje teksta i mogućnost proširivanja.

Monofont

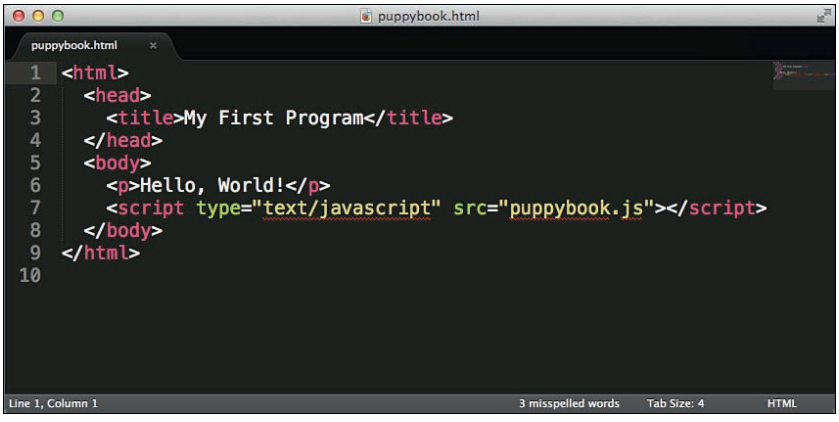
Monofont (engl. *monospace type font*) je takav stil fonta u kojem svi znakovi zauzimaju istu količinu prostora. Drugim rečima, slovo *i* zauzima isti prostor kao *w* i isti prostor kao razmak. Najpre će vam se činiti da je to ružno i nezgodno, ali vremenom ćete naučiti da to tolerišete, potom cenite, a zatim ćete otkriti lepotu koju formatiranje u monofontu daje vašem kodu.

Isticanje sintakse

Isto kao što obični jezici imaju imenice, glagole, prideve i tako dalje, programski jezici se sastoje od različitih delova (kao što su *promenljive*, *rezervisane reči* i *niske*). Dobar tekst editor razlikuje razne delove programskih jezika, obično tako što koristi različite boje teksta. Pogledajte slike 1.1 i 1.2. Čak i kada ne znate šta taj kôd znači, vi vidite da je slika 1.2 mnogo lakša za gledanje. Isticanje sintakse može takođe da vam pomogne da u svom kodu uočite greške kucanja.



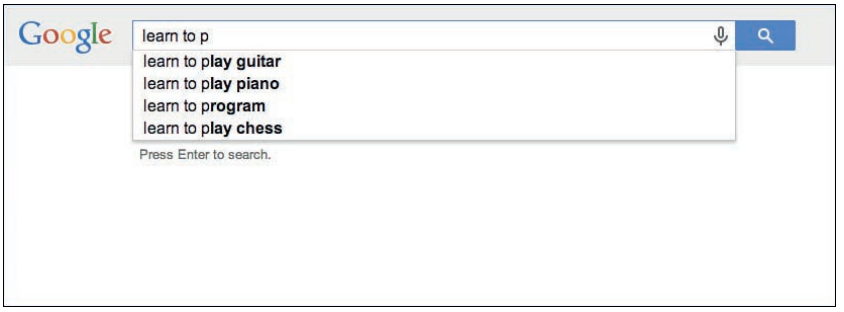
Slika 1.1 Jedan HTML dokument u fontu koji nije monofont.



Slika 1.2 Isti HTML dokument u monofontu. Mnogo se lakše čita (a i lepši je)?

Kompletiranje teksta

Kada pišete kôd, često morate da ponavljate iste reči. Da bi kôd pravilno radio, reči moraju svaki put da budu napisane tačno na isti način (na primer, `myCode` nije isto što i `MyCode` ili `mycode`). Male greške u kucanju teško se pronalaze, a mogu da izazovu velike glavobolje. Svaki dobar tekst editor sadrži neku vrstu kompletiranja teksta – kao što Google može da pogodi šta ćete da tražite, vaš tekst editor može da pogodi šta ćete da napišete (slika 1.3). Kada koristite kompletiranje teksta, lakše ćete brzo da pišete kôd, a i lakše ćete izbeći greške u kucanju kao i druge greške.



Slika 1.3 Dobar tekst editor pogađa šta ste hteli da napišete, isto kao što Google pogađa šta ste hteli da tražite.

Mogućnost proširivanja

Tekst editori koji su namenjeni programerima trebalo bi da tim programerima dozvole pravljenje proširenja (engl. *extension*) i dodatnih modula (engl. *plug-in*), da poboljšaju i izmene ponašanje editora. Na primer, proširenja mogu da izmene izgled editora, da proveravaju da li u kodu ima grešaka, ili da brzo dodaju „odlomke" (engl. *snippets*) koda u fajl. Kao što ćete često videti u ovoj knjizi, mogućnost proširivanja nije važna samo za tekst editore, već za skoro sav softver. Skoro sigurno ima stvari koje se prvobitni autor nije setio da ubaci, koje nije imao vremena da napravi, ili nije ni hteo. Mogućnost proširivanja daje šansu da te stvari napravi svako ko hoće.

Biranje

Koji je najbolji tekst editor za vas zavisi od toga šta više volite (i da li ste spremni da to platite), a zavisi i od projekta na kojem radite. Ja koristim Vim za većinu kodiranja, ali koristim IntelliJ kada pišem Java kôd, a Xcode kada radim na iOS aplikacijama. Sledeći pregled tekst editora trebalo bi da vam pomogne da se odlučite.

> ### Šta je to IDE?
>
> U ovom poglavlju, učite o tome kako u stvari vaš računar izvršava kôd koji vi upišete u svoj tekst editor. Za deo tog procesa potrebno je da imate okruženje koje može da razume i da izvršava vaš kôd. U ovoj knjizi mi koristimo programski jezik koji se zove JavaScript; okruženje koje može da razume i izvršava JavaScript je internet pretraživač (engl. *web browser*). Za neke druge programske jezike, sam tekst editor predstavlja okruženje koje razume i izvršava kôd. Te specijalne vrste tekst editora zovu se integrisana razvojna okruženja (engl. *integrated development environment*), ali ih prijatelji zovu IDE. IDE je moćna alatka za jezike kojima treba IDE. Vaš internet pretraživač može da bude isto tako moćan za pisanje JavaScript koda.

Sublime Text

- Sublime Text ima široku lepezu dodatnih modula koji mogu da pomognu da budete produktivniji. Lako se koristi i ima privlačan (smem li reći subliman?) korisnički interfejs.
- Postoji za Windows, Mac i za Linux.
- Sublime Text je dobar izbor za početnika. Jednostavan je i lako razumljiv i u njemu važe mnoge konvencije na koje ste navikli u programu za obradu teksta.
- Licenca za Sublime Text košta 70 dolara, ali možete besplatno da ga isprobate.
- Sublime Text je jednostavan tekst editor, a ne potpun IDE.
- Sublime možete da preuzmete sa http://www.sublimetext.com/.

TextMate

- TextMate i Sublime Text imaju mnoga ista proširenja – proširenje za jedan od njih obično funkcioniše i u drugom. TextMate se lako uči i lako se koristi.
- Postoji samo za Mac.
- Kao i Sublime Text, TextMate je dobar izbor za početnika. Možete da počnete i da budete produktivni, bez potrebe da išta novo naučite.
- Licenca za TextMate košta 55 dolara, ali možete besplatno da ga isprobate.
- TextMate je jednostavan tekst editor, a ne potpun IDE.
- TextMate možete da preuzmete sa http://macromates.com/.

Notepad++

- Notepad++ je odlična opcija ako koristite Windows. Nastavak „++" (koji se čita „plus plus") potiče od programskog jezika C++; zamisao je da C++ liči na C, ali je bolji – dakle, Notepad++ liči na Notepad, ali je bolji.
- Postoji samo za Windows.
- Notepad++ se lako koristi i odličan je za početnike.
- Notepad++ je besplatan.
- Notepad++ je jednostavan tekst editor, a ne potpun IDE.
- Notepad++ možete da preuzmete sa http://notepad-plus-plus.org/download/.

Gedit

- Gedit je dobar osnovni editor, ali nije vizuelno privlačan kao neki od ostalih editora. Ako koristite Linux, verovatno je već instaliran.

- Postoji za Windows, Mac i za Linux.
- Gedit se lako koristi i lako se uči.
- Gedit je besplatan.
- Gedit je jednostavan tekst editor, a ne potpun IDE.
- Gedit možete da preuzmete sa https://wiki.gnome.org/Apps/Gedit/.

Vim

- Vim treba neko vreme da se uči, ali kad ga jednom naučite, možete brzo da radite, sa raznovrsnim prečicama i dodatnim modulima. Osim toga, Vim je kompatibilan sa skoro svim operativnim sistemima i često je već instaliran, pa je rad na nepoznatom operativnom sistemu manje neobičan ako koristite Vim. Nekad sam se plašio Vima, ali je on sada moj omiljeni editor.
- Raspoloživ je na skoro svakom operativnom sistemu koji postoji (unapred je instaliran na Mac OS X-u i Linuxu, ali na Windowsu morate sami da ga instalirate). Vim je kao bubašvaba: može da preživi skoro svuda i verovatno će i dalje postojati kad svih nas više ne bude.
- Vim se ne uči lako i verovatno nije najbolji editor za početnike. Međutim, ako hoćete da naučite Vim, naći ćete mnoge resurse, uključujući ugrađeni program za podučavanje (upišite `vimtutor` na komandnu liniju).
- Vim je besplatan.
- Vim je jednostavan tekst editor, a ne potpun IDE.
- Vim možete da preuzmete sa www.vim.org/download.php, ako nije već instaliran.

Eclipse

- Eclipse je potpun IDE koji se obično koristi za Java programiranje. Ako radite na projektu u Javi, ovo je odličan izbor.
- Postoji za Windows, Mac i za Linux.
- Eclipse nije naročito lak za početnike, zato što to nije jednostavan tekst editor. Međutim, učiti programiranje u Javi je mnogo lakše ako koristite Eclipse ili IntelliJ nego neki od ranije pomenutih jednostavnih tekst editora.
- Eclipse je besplatan.
- Eclipse je IDE fokusiran na Java programiranje.
- Eclipse možete da preuzmete sa www.eclipse.org/downloads/

IntelliJ

- IntelliJ je potpun IDE za Javu i nešto je laganiji (i, verovatno, lepši) nego Eclipse. IntelliJ podržava i druge jezike, kao što su Scala i JavaScript.
- Postoji za Windows, Mac i za Linux.
- IntelliJ se uči podjednako lako kao i Eclipse.
- IntelliJ ima besplatno izdanje Community Edition. Licenca za izdanje Ultimate Edition košta 199 dolara.
- IntelliJ je potpun IDE.
- IntelliJ možete da preuzmete sa www.jetbrains.com/idea/.

Xcode

- Ako pišete aplikaciju za iOS ili Mac OS X, Xcode je pravi IDE za vas. Xcode je IDE koji je Apple napravio da bi se pravio softver za Apple platforme.
- Postoji samo za Mac.
- Xcode nije lako naučiti, ali Apple ima opširnu dokumentaciju, a možete da nađete i mnogo podrške od zajednice. Ako želite da naučite kako se prave aplikacije za iPhone, morate da naučite kako se koristi Xcode.
- Xcode je besplatan.
- Xcode možete da preuzmete sa https://developer.apple.com/xcode/ ili sa Mac App Store.

Visual Studio

- Visual Studio je potpun IDE koji se najviše koristi za .NET razvoj (C#, Visual Basic, i tako dalje), ali može takođe da se koristi za druge jezike.
- Postoji samo za Windows.
- Visual Studio se uči podjednako lako kao i bilo koji drugi potpun IDE – ne previše lako.
- Visual Studio ima besplatno izdanje Express Edition, koje je i dobro i upotrebljivo.
- Plaćena izdanja Visual Studia koštaju od 1.200 dolara do 13.300 dolara.
- Visual Studio možete da preuzmete ili da kupite sa www.visualstudio.com/downloads/ download-visual-studio-vs.

Nisam ovde pomenuo sve editore, ali ovaj spisak bi trebalo da vam bude dovoljan za početak. Za potrebe ove knjige, ja preporučujem Sublime Text (osim ako već imate iskustvo sa nekim od drugih editora). Sublime Text je više nego dovoljan za naše potrebe i lako se odmah koristi.

Napravite direktorijum za projekat

Pre nego što počnemo da pišemo kôd, treba nam mesto na koje ćemo da ga stavimo. Računarski programi se obično sastoje od više fajlova koji sarađuju, zato je dobro da se svi fajlovi potrebni za projekat grupišu u jedan folder (folderi se često zovu direktorijumi – u ovoj knjizi naizmenično koristim izraze *folder* i *direktorijum*). Ja imam jedan direktorijum koji se zove `projects` u kojem grupišem sve takve direktorijume projekata. Napravite negde na svom disku direktorijum po imenu `kittenbook`. Nov direktorijum možete da napravite pomoću čitača fajlova (Finder za Mac, Windows Explorer za Windows, a Nautilus za Linux) pa pritisnite File → New Folder.

Počnite od malog: napravite fajl za testiranje

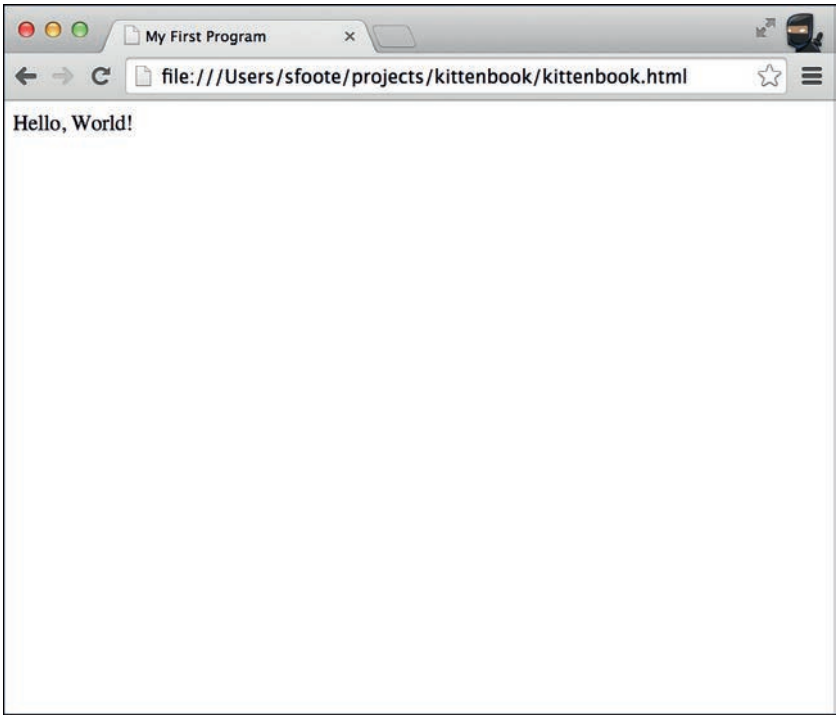
Pošto sada imate direktorijum projekta, počnimo sa programiranjem. Prva stvar koju ćete u vašem direktorijumu `kittenbook` da napravite je HTML fajl koji ćete nazvati `kittenbook.html`. Novi fajl možete da napravite na više načina, ali je najlakše da otvorite odabrani tekst editor, napravite nov fajl (pritisnite File → New File), a zatim sačuvate fajl kao `kittenbook.html` (File → Save As, zatim pronađite svoj direktorijum `kittenbook`, upišite `kittenbook.html` kao ime fajla, pa pritisnite Save). Sada popunite `kittenbook.html` kodom navedenim kao listing 1.1.

Listing 1.1 **`kittenbook.html`**

```
<html>
    <head>
        <title>My First Program</title>
    </head>
    <body>
        <p>Hello, World!</p>
    </body>
</html>
```

Sačuvajte fajl a zatim ga otvorite u veb pretraživaču tako da ga dva puta pritisnete u svom čitaču fajlova. Trebalo bi da ugledate nešto kao na slici 1.4.

Odmorite se da biste uživali u ovom trenutku. Pogledajte ponovo svoj kôd i uporedite ga sa onim što vidite u veb pretraživaču. Trebalo bi da vidite `Hello, World!` u glavnom prozoru, a `My First Program` na jezičku kartice. HTML (Hypertext Markup Language) se sastoji od elemenata. Element se sastoji od početne oznake (engl. *opening tag*), od opcionog tela (engl. *body*) i od završne oznake (engl. *closing tag*). Na primer `<p>` je početna oznaka za element pasus (engl. *paragraph*), `Hello, World` je sadržaj, odnosno telo, a `</p>` je završna oznaka. Telo elementa može da sadrži i druge elemente. Na primer, pogledajte listing 1.2.



Slika 1.4 Naredili ste računaru da nešto uradi i uspelo je!

Listing 1.2 **Oznaka `<head>` za `kittenbook.html`**

```
<head>
    <title>My First Program</title>
</head>
```

Element `head` sadrži element naslova `<title>`. On obaveštava veb pretraživač da je to naslov HTML stranice, a većina veb pretraživača prikazuje naslov u jezičku kartice. Još ćete mnogo učiti o HTML-u, ali za sada ostajemo na ovome.

Kako HTML i JavaScript sarađuju u pretraživaču

Upravo ste napravili svoju prvu veb stranicu. Hajde sada da ta stranica bude malo interaktivnija. Napravite u svom direktorijumu `kittenbook` još jedan fajl po imenu `kittenbook.js`. To će biti jedan JavaScript fajl, pa će vaša veb stranica biti zanimljivija.

Otvorite `kittenbook.js` pomoću tekst editora i popunite fajl kodom iz listinga 1.3.

Listing 1.3 **Hello, prijatelju!**

```
alert('Hello, [vaše ime]!');
```

Umesto `[vaše ime]` napišite svoje ime. Za mene, to je bilo `alert('Hello, Steven!')`. Sada ponovo otvorite svoju veb stranicu (ili je ponovo učitajte) i pogledajte šta se dešava. To je trik! Stranica bi trebalo da izgleda isto kao i ranije. HTML ne zna da naš JavaScript postoji, a to je zato što mi nismo HTML-u ništa rekli za taj JavaScript. Ažurirajte `kittenbook.html` tako što ćete red iz listinga 1.4 dodati u element `<body>`.

Listing 1.4 **Dodajte vaš JavaScript u `kittenbook.html`**

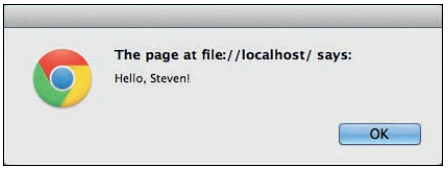
```
<script type="text/javascript" src="kittenbook.js"></script>
```

Sada bi `kittenbook.html` trebalo da izgleda kao listing 1.5.

Listing 1.5 **`kittenbook.html` sa JavaScript-om**

```
<html>
    <head>
        <title>My First Program</title>
    </head>
    <body>
        <p>Hello, World!</p>
        <script type="text/javascript" src="kittenbook.js"></script>
    </body>
</html>
```

Ovog puta, trebalo bi da ugledate nešto kao na slici 1.5.



Slika 1.5 Hello, napisali ste pravi program!

To je bilo fenomenalno! Izazvali ste pojavljivanje prozora sa svojim imenom. Mogli ste da napišete što god hoćete i to bi se pojavilo u tom prozoru. Govorićemo o tome šta ste upravo uradili; zatim ćemo preći na sledeći nivo.

JavaScript koji ste napisali je samo jedna instrukcija za vaš računar. Instrukcija `alert` je nešto što se zove funkcija, a ona naređuje veb pretraživaču da otvori prozor koji sadrži dugme OK. U taj prozor možete da dodate neki tekst tako da funkciji `alert` dodate

argument. Funkcija je deo koda koji može da se pozove da bi izvršio neki zadatak. Argument može da menja način na koji funkcija vrši taj zadatak. Znak tačka i zarez na kraju reda obaveštava JavaScript da je to kraj instrukcije. Programi se obično sastoje od mnogo instrukcija, zato se instrukcije razdvajaju znakom tačka i zarez.

Značaj malih promena

Sada ćemo da obogatimo svoj program tako da može da pozdravi bilo koga, umesto da kaže samo „Hello, World!" Kada unapređujete svoj program, najbolje je da pravite male promene. Zatim isprobajte te male promene pre dodavanja drugih promena. Na početku ovaj proces može da izgleda dosadan i nepotreban. Međutim, ako pre testiranja unesete mnogo promena, pa utvrdite da vaš program ne radi, biće vam teško da utvrdite koja od promena je pokvarila program. Ako pre testiranja unesete jednu malu promenu, pa vidite da se vaš program pokvario, znaćete tačno šta ste uradili da ga pokvarite.

Naša prva mala promena biće da zatražimo ime. Personalizovan pozdrav ništa ne vredi ako ne sadrži ime. Za ovaj primer, upotrebićemo funkciju koja se zove `prompt` (podsticanje na odgovor) zato što je jednostavna i zato što ovde dobro funkcioniše. Međutim, preklinjem vas da ne koristite `prompt` (ili `alert`) na pravom veb sajtu. Mada su `prompt` i `alert` zgodni za testiranje i učenje, oni proizvode jadan i zastareo korisnički doživljaj. Alternative obrađujemo u poglavlju 8, „Funkcije i metodi". Za sada, ažurirajte svoj `kittenbook.js` tako da pozove `prompt` umesto `alert`, a poruku promenite tako da glasi „Hello, what\'s your name?" („Hello, kako je vaše ime?") (pogledajte listing 1.6).

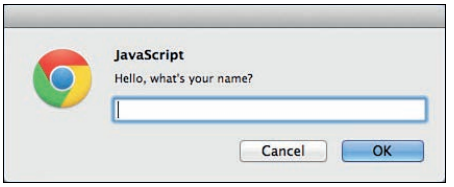
Listing 1.6 **Podsticanje na unos imena**

```
prompt('Hello, what\'s your name?');
```

Možda se pitate zbog čega u kodu stoji `what\'s` a ne `what's`. Grupa znakova kao što je `'Hello, World!'` zove se niska (engl. *string*). Niska počinje znakom navoda i završava se znakom navoda. Ako napišem `prompt('Hello, what's your name?');`, onda imam tri znaka navoda (iako je jedan od njih u stvari apostrof), a to će zbuniti računar. Računar vidi nisku `'Hello, what'`, zatim vidi neke druge znakove koje ne razume, a zatim vidi još jednu nisku: `');'`. Ja sam upotrebio obrnutu kosu crtu ispred znaka navoda u `what\'s` da bih obavestio računar da ne želim da taj apostrof označi kraj niske. Obrnuta kosa crta se naziva izlazna sekvenca (engl. *escape character*).

Ako sada osvežite svoju stranicu, trebalo bi da ugledate prozor sa prostorom u koji treba da upišete svoje ime (slika 1.6).

Primetićete da kada upišete svoje ime i pritisnete OK, stranica još uvek odgovara `Hello, World!` U čemu je štos? Zašto se traži ime ako se posle ne pokaže? E pa, mi nismo rekli računaru šta da uradi sa tim imenom. Napravili smo malu promenu: zamenili smo prozor `alert` prozorom `prompt`. Uspelo je, dakle sada možemo da kažemo računaru šta da uradi sa tim imenom.



Slika 1.6 Kako je vaše ime?

Jedna od najvažnijih alatki u programiranju je promenljiva (engl. *variable*). Promenljiva je mesto za čuvanje nekog podatka koji utiče na to kako će program da radi – u poglavlju 5, „Tipovi podataka, strukture podataka, baze podataka" detaljno se opisuju promenljive. U ovom slučaju, mi hoćemo da sačuvamo ime iz prozora prompt. (Pogledajte listing 1.7.)

Listing 1.7 **Vaša prva promenljiva**

```
var userName = prompt('Hello, what\'s your name?');
```

Sve promenljive imaju imena, a mi smo našu promenljivu nazvali `userName`. Da bismo obavestili računar da pravimo promenljivu, koristimo rezervisanu reč: `var`. Rezervisana reč je reč koja ima specijalno značenje za neki programski jezik. Ona je rezervisana, pa ne možete tu reč da upotrebite kao ime promenljive; ne biste želeli da zbunite računar.

Pošto smo sada sačuvali svoju promenljivu, osvežite stranicu da proverite da li sve još uvek radi. Veb stranica bi trebalo da izgleda isto kao kada ste je prethodni put osvežili. Sačuvali smo ime korisnika, ali nismo još rekli računaru da nešto uradi s njim. Sledeći korak je da ubacimo ime korisnika na veb stranicu (pogledajte listing 1.8).

Listing 1.8 **Uposlite svoju promenljivu**

```
var userName = prompt('Hello, what\'s your name?');
document.body.innerHTML = 'Hello, ' + userName + '!';
```

Dodajemo jednu novu instrukciju da promenimo `body` (telo) HTML `documenta`. Sećate se, element `<body>` iz `kittenbook.html`? Ovo `document.body.innerHTML` odnosi se na sve, od početne oznake pa do završne oznake u elementu `<body>`. To `document.body.innerHTML = 'Hello, ' + userName + '!';` u suštini menja HTML pa će on izgledati kao u listingu 1.9 (pod pretpostavkom da ja upišem `Steven` kada se od mene zatraži moje ime).

Listing 1.9 **kittenbook.html pošto vaš JavaScript obavi svoj posao**

```
<html>
    <head>
    <title>My First Program</title>
    </head>
    <body>
```

```
        Hello, Steven!
    </body>
</html>
```

Čestitam! Upravo ste napravili pravi program. Priznajem, ovo nije najkorisniji program koji je ikad napisan, ali svakako nije *najmanje* koristan – a pri tom ste prilično naučili.

Gradite na svom uspehu

Sada ćemo uzeti program koji ste upravo napravili i pretvorićemo ga u proširenje (engl. *extension*) za Chrome. Proširenje za Chrome je mali program koji može da se instalira u Google Chrome da bi se unapredio doživljaj korisnika. Proširenje za Chrome može da bude moćno; nekim kompanijama je glavni proizvod neko proširenje za Chrome. Proširenje koje započinjemo u ovom poglavlju poslužiće kao osnova za projekat u ostatku knjige.

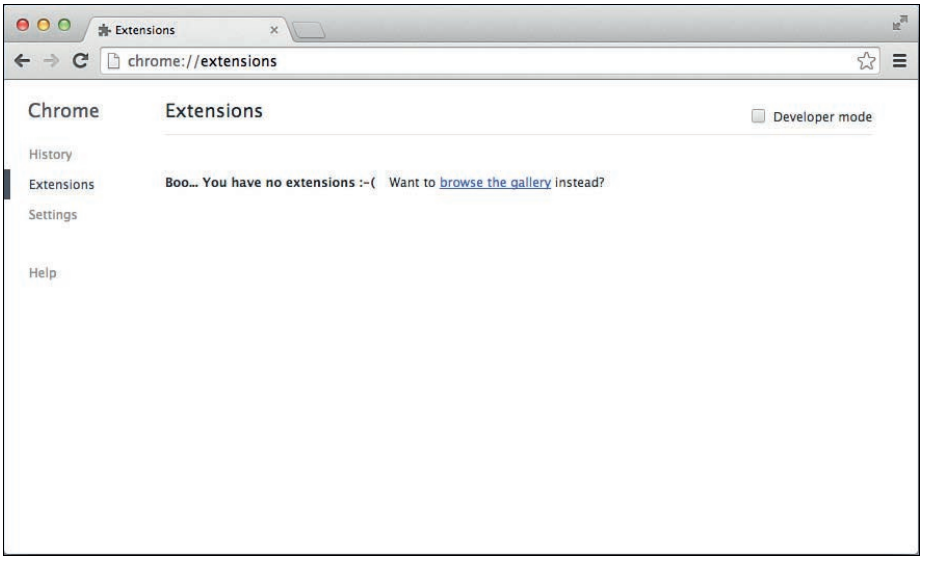
Prvi korak je da se napravi novi fajl po imenu `manifest.json`. Ovaj fajl sadrži informacije za Chrome o proširenju i o tome kako ono funkcioniše. JSON je tip dokumenta u kojem se podaci čuvaju na jedan efikasan način koji lako mogu da čitaju i računari i ljudi. Popunite svoj fajl `manifest.json` tako da izgleda kao listing 1.10 (ne zaboravite da je svaki znak važan, uključujući i neobične znakove u prvom i poslednjem redu, koji se zovu vitičaste zagrade).

Listing 1.10 **Primer fajla `manifest.json` za Chrome proširenje**

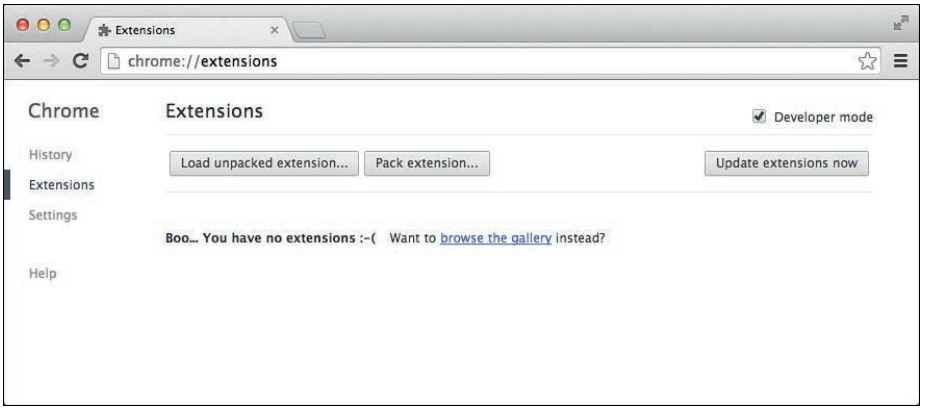
```
{
    "manifest_version": 2,
    "name": "kittenbook",
    "description": "Replace photos on Facebook with kittens",
    "version": "0.0.1"
}
```

Sada, pošto imate fajl `manifest.json`, vi imate sve što je potrebno da biste napravili proširenje za Chrome. U ovom trenutku, vaše proširenje ništa ne radi, ali ipak možete da ga dodate u Chrome da biste proverili da li postoji neki problem sa vašim fajlom `manifest.json`. Da biste svoje proširenje dodali u Chrome, treba da otvorite veb pretraživač Chrome (preuzmite ga, ako već niste) a zatim u adresnu liniju upišite **`chrome://extensions`**. Trebalo bi da ugledate nešto kao na slici 1.7.

Potvrdite polje za potvrdu režima Developer Mode (pošto ste vi sada razvijalac (engl. *developer*)!) da biste mogli da dodate svoje proširenje. Sada bi trebalo da se vidi dugme na kojem piše Load Unpacked Extension (učitati nespakovano proširenje) (slika 1.8).



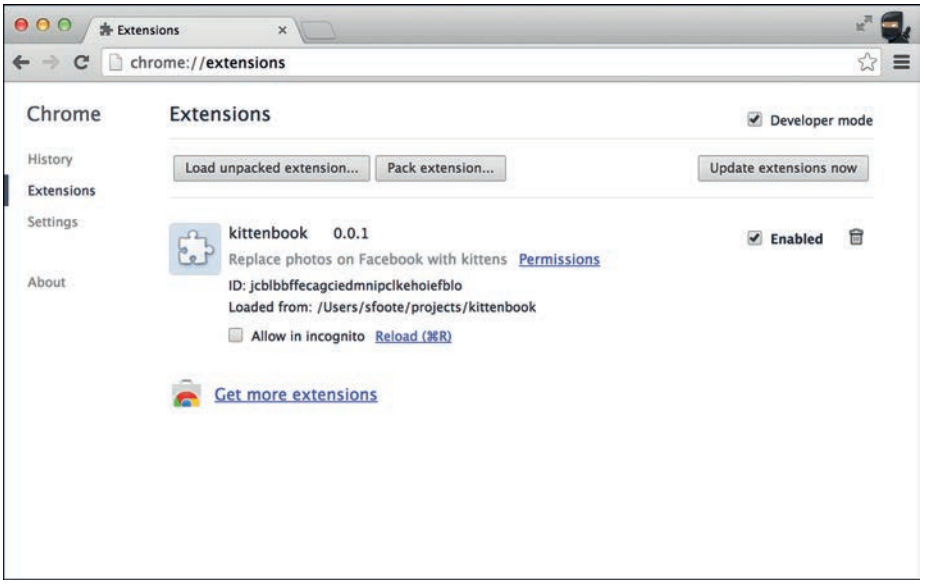
Slika 1.7 Stranica Chrome Extensions



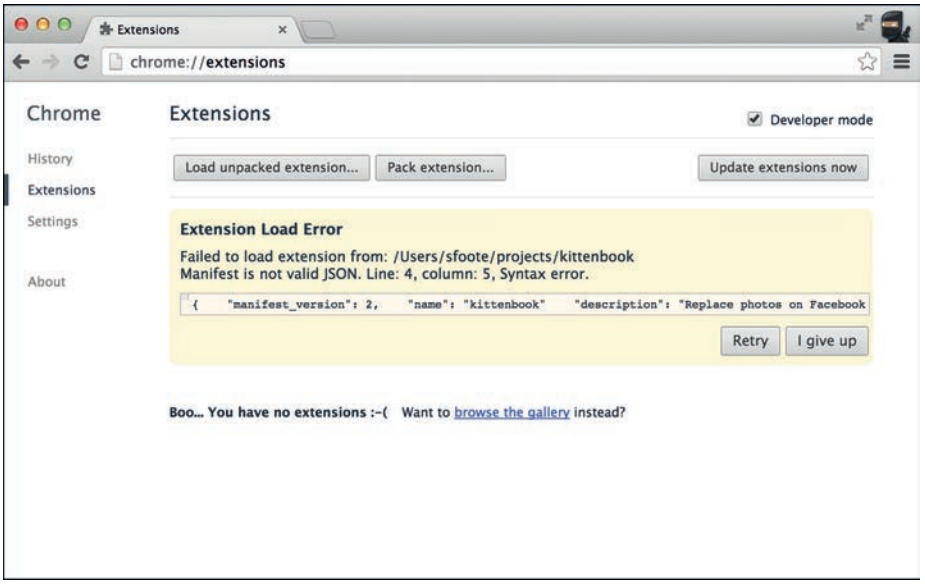
Slika 1.8 Stranica Chrome Extensions u režimu Developer Mode

Kada pritisnete to dugme, treba da selektujete ceo svoj direktorijum projekta (a ne samo jedan fajl u direktorijumu projekta). Ako ste selektovali pravilan direktorijum, a sa fajlom `manifest.json` nema nikakvih problema, vaše proširenje bi trebalo da se pojavi u spisku proširenja (slika 1.9).

Ako postoji neki problem sa vašim fajlom `manifest.json`, javiće se poruka kao ona na slici 1.10. Ako ugledate takvu poruku, treba da iskopirate sadržaj fajla `manifest.json` i prenesete ga u neki JSON validator kao što je http://jsonlint.com/, koji će vam pokazati tačno gde se nalazi vaša greška i kako da je ispravite.



Slika 1.9 Kittenbook se prvi put javlja u Chrome



Slika 1.10 Kada `manifest.json` nije u redu

Referencirajte svoj JavaScript u fajlu manifest.json

Pošto se proširenje kittenbook uspešno instalira, spremni ste da ono zaista nešto i uradi. Već ste napisali JavaScript fajl, pa vam ostaje jedino da taj JavaScript stavite na raspolaganje na Facebook.com. Ovaj isti problem smo imali i ranije kada smo pokušali da dodamo JavaScript fajl u `kittenbook.html`. Da bismo rešili taj problem, morali smo da obavestimo `kittenbook.html` o JavaScript fajlu. U ovom slučaju, moramo da obavestimo `manifest.json` o JavaScript fajlu. Takođe moramo da obavestimo `manifest.json` o veb sajtu u koji naš JavaScript fajl treba da se doda. To možemo da uradimo pomoću jednog svojstva po imenu `content_scripts` (pogledajte listing 1.11). Skriptovi sadržaja (engl. *content scripts*) su JavaScript fajlovi koje treba dodati u sadržaj date veb stranice ili skupa veb stranica.

Listing 1.11 **`manifest.json` sa skriptovima sadržaja**

```
{
    "manifest_version": 2,
    "name": "kittenbook",
    "description": "Replace photos on Facebook with kittens",
    "version": "0.0.1",
    "content_scripts": [
        {
            "matches": ["*://www.facebook.com/*"],
            "js": ["kittenbook.js"]
        }
    ]
}
```

Sa ovim dodatkom u listingu 1.11, naše proširenje sada dodaje `kittenbook.js` (`"js": ["kittenbook.js"]`) u svaku stranicu koja u svom URL-u sadrži www.facebook.com (`"matches": ["*://www.facebook.com/*"]`). Da bi vaše proširenje prihvatilo promene koje ste napravili u `manifest.json`, morate ponovo da učitate svoje proširenje, a to ćete uraditi pritiskom na link Reload koji vidite na slici 1.9.

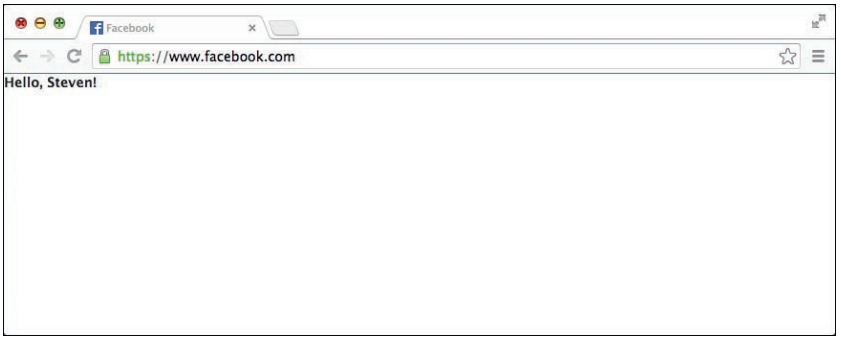
Neka radi!

Ako se vaše proširenje uspešno ponovo učitalo, možete da napravite još jednu, poslednju, promenu u `kittenbook.js` da bi pozdrav izgledao malo lepše (pogledajte listing 1.12).

Listing 1.12 **Izmenjen `kittenbook.js` za primenu u proširenju za Chrome**

```
var userName = prompt('Hello, what\'s your name?'); document.body.innerHTML =
'<h1>Hello, ' + userName + '!</h1>';
```

Dodavanjem oznake `<h1>` pozdrav se menja u stil naslova (engl. *heading*) (veći i podebljan tekst). Sada ponovo učitajte proširenje i otvorite Facebook da vidite šta će se desiti. Trebalo bi da se pojavi vaš prozor u kojem se od vas traži ime, a zatim nešto kao na slici 1.11.



Slika 1.11 Hello, Facebook!

To je bilo neverovatno – promenili ste Facebook. *Ovaj* Facebook prikazuje *vaš* pozdrav. To je stvarno super, ali nije uopšte korisno. U stvari, proširenje koje smo upravo napravili će sprečiti prikazivanje bilo koje prave Facebook stranice. To je stvarno neprijatno. Dok sam pravio ovo proširenje, moja žena je htela da pogleda Facebook na mom računaru; nije baš bila oduševljena pozdravom koji je preuzeo njenu Facebook stranicu. Kada ne radite na svom proširenju, možete da ga onemogućite tako da iz polja Enabled obrišete znak za potvrdu (pogledajte sliku 1.9).

Veća moć, veća odgovornost

Čitajući ovo poglavlje, počeli ste da stičete značajnu moć. Vi počinjete da shvatate da možete svom računaru da dajete instrukcije i da će on sve da uradi onako kako vi kažete. Softver nije više nešto što prosto kupite; možete i da ga *napravite*.

Proširenje za Chrome koje smo započeli u ovom poglavlju biće odlična alatka uz čiju pomoć ćete učiti o mnogim različitim konceptima programiranja. Međutim, kako sada stoji, proširenje više smeta nego što koristi. Ne bi bilo u redu da ga nudite svojim prijateljima (pogotovo ako im ne kažete kako da ga onemogućе). Sada imate moć da pravite korisne i zabavne programe koji mogu da budu od pomoći, ali takođe imate moć da pravite dosadne, štetne i zlonamerne programe. Uvek koristite svoju moć za dobre namene.

Rezime

U ovom poglavlju ste učili o:

- Tekst editorima, mestu gde se prave računarski programi
- Direktorijumima projekta da bi vam programi bili organizovani
- HTML i JavaScript-u i o tome kako oni sarađuju
- Malim izmenama da bi se problemi što pre uočili
- Promenljivima i izlaznim sekvencama

O, još ste i napravili pravi program koji funkcioniše. Zatim ste taj program pretvorili u proširenje za Chrome koje stvarno menja Facebook.com. Samo pomislite, pre svega 40 minuta niste čak ni znali šta je to promenljiva. Trebalo bi da ste ponosni – a možda i malo umorni. Sada bi bio dobar trenutak da se malo odmorite.

U sledećem poglavlju (nakon pauze), učićete o:

- tome kako softver radi
- prevedenom softeru
- interpretiranom softveru
- ulazu i izlazu
- memoriji i promenljivima