POGLAVLJE 1

Uvod u JavaScript

JavaScript je programski jezik koji se interpretira, sa objektno orijentisanim (OO) mogućnostima. Po sintaksi, jezgro jezika JavaScript slično je jezicima C, C++ i Java, s programskim konstrukcijama kao što su naredba `if`, petlja `while` i operator `&&`. Međutim, ova sličnost se završava sa sintaksom. JavaScript je slabo tipiziran jezik, što znači da se za promenljive ne mora definisati tip. Objekti u JavaScriptu preslikavaju (mapiraju) imena svojstava u proizvoljne vrednosti svojstava. Zbog toga su sličniji heš tabelama ili asocijativnim nizovima (u Perlu) nego strukturama (u C-u) ili objektima (u jezicima C++ ili Java). Mehanizam OO nasleđivanja u JavaScriptu zasnovan je na prototipovima, kao u malo poznatom jeziku Self. On je potpuno različit od nasleđivanja u jezicima C++ i Java. Kao i Perl, JavaScript je jezik koji se interpretira i nadahnut je Perlom u mnogim oblastima, kakve su regularni izrazi i mogućnosti rada s nizovima.

Jezgro (engl. *core*) jezika JavaScript podržava brojeve, znakovne nizove (engl. *strings*) i logičke vrednosti kao osnovne tipove podataka. Osim toga, ono ima ugrađenu podršku za objekte kao što su nizovi (engl. *arrays*), datumi i regularni izrazi.

JavaScript se najčešće koristi u čitačima Weba (engl. *Web browsers*), pa se jezgro opšte namene proširuje objektima koji omogućavaju skriptovima interakciju s korisnikom, upravljanje čitačem Weba i izmene sadržaja dokumenta koji se pojavljuje unutar prozora čitača. Ova ugrađena (engl. *embedded*) verzija JavaScripta izvršava skriptove koji su ugrađeni u HTML Web stranice. To se obično zove *klijentski* (engl. *client-side*) JavaScript, da bi se naglasilo kako se skriptovi izvršavaju na klijentskom računaru, a ne na Web serveru.

Jezgro jezika JavaScript i njegovi ugrađeni tipovi predmet su međunarodnih standarda, i veoma je dobra kompatibilnost različitih implementacija. Neki delovi klijentskog JavaScripta zvanično su standardizovani, neki su de facto standardi, a ostali predstavljaju proširenja koja zavise od čitača. Kompatibilnost između različitih čitača značajna je za programere koji koriste klijentski JavaScript.

Ovo poglavlje sadrži pregled JavaScripta na visokom nivou i pruža osnovne informacije koje su vam potrebne pre nego što se upustite u proučavanje tog jezika. Kao motivaciju i uvod, ono obuhvata neke jednostavne primere koda klijentskog JavaScripta.

1.1 Šta je JavaScript?

JavaScript izaziva velike zablude i nejasnoće. Pre nego što nastavimo, važno je da razbijemo dva uobičajena i postojana mita o tom jeziku.

1.1.1 JavaScript nije Java

Jedna od najčešćih zabluda o JavaScriptu jeste da je to pojednostavljena verzija Jave, programskog jezika kompanije Sun Microsystems. Osim delimične sličnosti u sintaksi i činjenice da i Java i JavaScript daju sadržaj koji može da se izvršava u čitačima Weba, između ova dva jezika ne postoji nikakva veza. Sličnost naziva je čisto marketinški potez kompanija Netscape i Sun (ovaj jezik je prvobitno nazvan LiveScript i ime mu je u poslednjem trenutku promenjeno u JavaScript). Međutim, JavaScript može da skriptuje Javu (videti poglavlja 12 i 23).

1.1.2 JavaScript nije jednostavan

Pošto se JavaScript interpretira umesto da se prevodi, često se smatra da je to jezik za skriptovanje, a ne pravi programski jezik. Time se nagoveštava da su jezici za skriptovanje jednostavniji i da su to programski jezici za osobe koje nisu programeri. Činjenica da je JavaScript manje strog po pitanju tipova podataka čini ga nešto pristupačnijim neiskusnim programerima. Osim toga, mnogi Web dizajneri mogu da koriste JavaScript za brzo rešavanje manjih programerskih zadataka.

Međutim, ispod svoje tanke glazure jednostavnosti, JavaScript je programski jezik s potpunim skupom mogućnosti, složen kao i bilo koji drugi, a od nekih i složeniji. Programeri koji pokušaju da iskoriste JavaScript za rešavanje netrivijalnih zadataka često otkrivaju da je taj postupak težak, ukoliko dobro ne razumeju ovaj jezik. Ova knjiga iscrpno dokumentuje JavaScript, pa možete da steknete solidno znanje. Ako ste navikli da koristite uputstva u stilu kuvara za JavaScript, možda ćete biti iznenađeni dubinom i detaljnošću narednih poglavlja.

1.2 Verzije JavaScripta

Kao i sve nove tehnologije, JavaScript se brzo razvijao u početku. Prethodna izdanja ove knjige dokumentovala su ovaj napredak, verziju po verziju, tačno objašnjavajući koje su mogućnosti uvedene u kojoj verziji jezika. Međutim, dok je nastajalo ovo izdanje, jezik se ustalio i standardizovalo ga je udruženje European Computer

Manufacturer's Association (ECMA).* Implementacije ovog standarda obuhvataju interpreter za JavaScript 1.5 kompanija Netscape i Mozilla Foundation, kao i Microsoftov interpeter za JScript 5.5. Svaki čitač Weba koji je noviji od Netscapea 4.5 i Internet Explorera 4, podržava najnoviju verziju JavaScripta. U praksi, malo je verovatno da ćete naići na neodgovarajući interpreter.

Zapazite da je zvanično ime jezika ECMAScript, prema standardu ECMA-262. Međutim, ovaj zbunjujući naziv koristi se samo pri eksplicitnom pozivanju na standard. Naziv „JavaScript" se tehnički odnosi samo na implemetacije ovog jezika, koje su dale kompanije Netscape i Mozilla Foundation. U praksi, svi taj jezik zovu JavaScript.

Posle dugog perioda stabilnosti JavaScripta, postoje neki znaci promene. Čitač Weba Firefox 1.5, kompanije Mozilla Foundation, sadrži nov interpreter JavaScripta, čija je verzija 1.6. Ova verzija obuhvata nove (nestandardne) metode za rad s nizovima, opisane u odeljku 7.7.10, i podršku za E4X, što će biti uskoro objašnjeno.

Osim specifikacije ECMA-262, koja standardizuje jezgro jezika JavaScript, organizacija ECMA objavila je još jedan standard koji se odnosi na JavaScript. ECMA-357 standardizuje proširenje JavaScripta, poznato kao E4X ili ECMAScript za XML. To proširenje dodaje ovom jeziku tip podataka XML, zajedno sa operatorima i naredbama za rad s vrednostima tipa XML. Dok je pisana ova knjiga, E4X je primenjen samo u JavaScriptu 1.6 i Firefoxu 1.5. E4X nije formalno dokumentovan u ovoj knjizi, ali poglavlje 21 obuhvata prošireni uvod u obliku uputstva za rad.

Razmatranje predloga za četvrto izdanje specifikacije ECMA-262, kojom bi se standardizovao JavaScript 2.0, odlaže se godinama. U tim predlozima je potpuna revizija jezika, uključujući strogo tipiziranje i pravo nasleđivanje, zasnovano na klasama. Do sada je načinjen mali napredak u pravcu standardizacije JavaScripta 2.0. I pored toga, implementacije koje su zasnovane na nacrtima predloga uključuju Microsoftov jezik JScript.NET, kao i jezike ActionScript 2.0 i ActionScript 3.0 koji se koriste u programu Adobe (prethodno Macromedia) Flash Player. U vreme pisanja ove knjige, postoje znaci da se nastavlja rad na JavaScriptu 2.0 i da se izdavanje JavaScripta 1.6 može smatrati prvim korakom u tom smeru. Naravno, očekuje se da će svaka nova verzija jezika biti kompatibilna sa starijom verzijom koja je ovde opisana. Čak i kada JavaScript 2.0 bude standardizovan, proći će nekoliko godina dok se ne ugradi u sve čitače Weba.

1.3 Klijentski JavaScript

Kada je interpreter JavaScripta umetnut u čitač Weba, rezultat je klijentski JavaScript. To je, do sada, najčešća varijanta JavaScripta: kada se govori o JavaScriptu, većina ljudi misli na klijentski JavaScript. Ova knjiga dokumentuje klijentski JavaScript, zajedno s jezgrom jezika JavaScript, koje je obuhvaćeno klijentskim JavaScriptom.

* To je standard ECMA-262, verzija 3 (dostupan na Internetu, u datoteci *http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf*).

U klijentskom JavaScriptu kombinuju se mogućnosti interpretera JavaScripta da izvršava skriptove sa objektnim modelom dokumenta (engl. *Document Object Model*, DOM), definisanim u čitaču Weba. Dokumenti mogu da sadrže skriptove napisane na JavaScriptu, a ti skriptovi mogu da koriste DOM za izmenu dokumenta ili upravljanje čitačem Weba koji prikazuje taj dokument. Drugim rečima, možemo kazati da klijentski JavaScript dodaje ponašanje statičkim sadržajima Weba. Klijentski JavaScript čini srž tehnika koje služe za razvoj Web aplikacija, kao što je DHTML (poglavlje 16) i arhitektura kao što je Ajax (poglavlje 20). U uvodu poglavlja 13 dat je pregled mnogih mogućnosti klijentskog JavaScripta.

Kao što specifikacija ECMA-262 definiše standardnu verziju jezgra jezika JavaScript, konzorcijum za upravljanje Webom (engl. *World Wide Web Consortium*, W3C) objavio je specifikaciju modela DOM kojom se standardizuju mogućnosti koje čitač Weba mora da podrži u svom objektnom modelu dokumenta. (Mnogo više o ovom standardu saznaćete u poglavljima 15, 16 i 17.) Osnovni delovi W3C DOM standarda dobro su podržani u svim vodećim čitačima Weba. Jedan značajan izuzetak je Microsoft Internet Explorer, koji ne podržava W3C standard za obradu događaja.

1.3.1 Primeri klijentskog JavaScripta

Kada se čitač Weba proširi interpreterom JavaScripta, on omogućava distribuciju izvršnog sadržaja preko Interneta, u obliku skriptova na JavaScriptu. Primer 1-1 prikazuje kako to izgleda: ovo je jednostavan program na JavaScriptu, ili skript, umetnut u HTML datoteku.

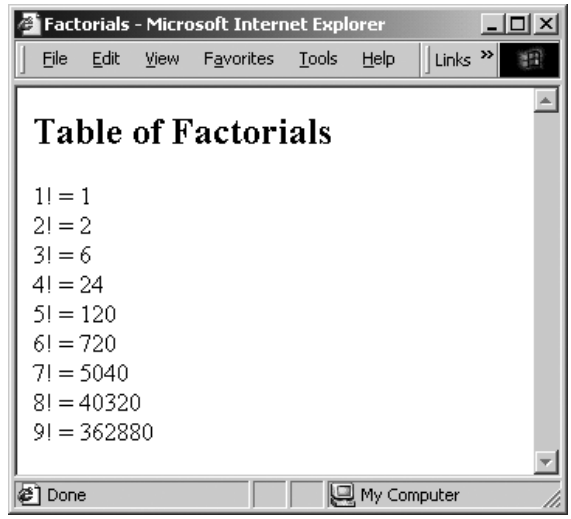
Primer 1-1. Jednostavan program na JavaScriptu

```
<html>
<head><title>Factorials</title></head>
<body>
<h2>Table of Factorials</h2>
<script>
var fact = 1;
for(i = 1; i < 10; i++) {
    fact = fact*i;
    document.write(i + "! = " + fact + "<br>");
}
</script>
</body>
</html>
```

Kada se ovaj skript unese u čitač koji podržava JavaScript, daje izlaz prikazan na slici 1-1.

Kao što vidite u ovom primeru, oznake `<script>` i `</script>` koriste se za umetanje koda na JavaScriptu u HTML datoteku. U poglavlju 13 detaljnije ću opisati oznaku `<script>`. Osnovna mogućnost JavaScripta, koja je prikazana ovim primerom, jeste primena metode `document.write( )`.* Ova metoda se koristi za dinamičko slanje HTML teksta u HTML dokument, dok se on unosi u čitač.

* „Metoda“ je OO termin za funkciju ili proceduru, a njenu primenu ćete videti u ovoj knjizi.



Slika 1-1. Web stranica generisana pomoću JavaScripta

JavaScript može da upravlja sadržajem HTML dokumenata, ali i njihovim ponašanjem. To znači da program na JavaScriptu može na neki način da odgovori kada unesete vrednost u polje za unos podataka ili prelazite mišem preko slike u dokumentu. U JavaScriptu se ovo postiže definisanjem *funkcija za obradu događaja* (engl. *event handlers*) za taj dokument – to su delovi koda na JavaScriptu, koji se izvršavaju kada nastupi određeni događaj, na primer, kada korisnik pritisne dugme. Primer 1-2 prikazuje jednostavan deo HTML koda, s funkcijom za obradu događaja koja se izvršava kada se pritisne dugme.

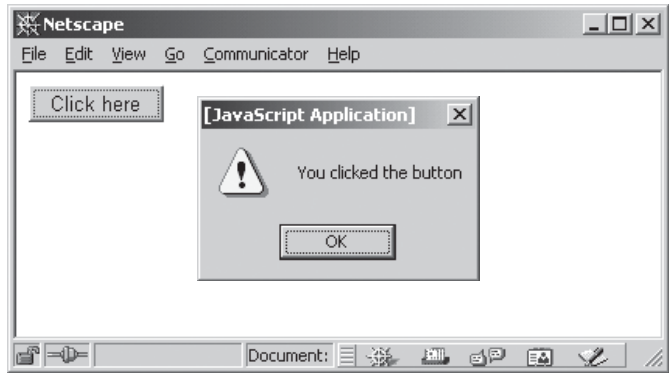
Primer 1-2. HTML dugme s funkcijom za obradu događaja definisanom na JavaScriptu

```
<button onclick="alert('You clicked the button');">
Click here
</button>
```

Na slici 1-2 prikazan je rezultat pritiska na dugme.

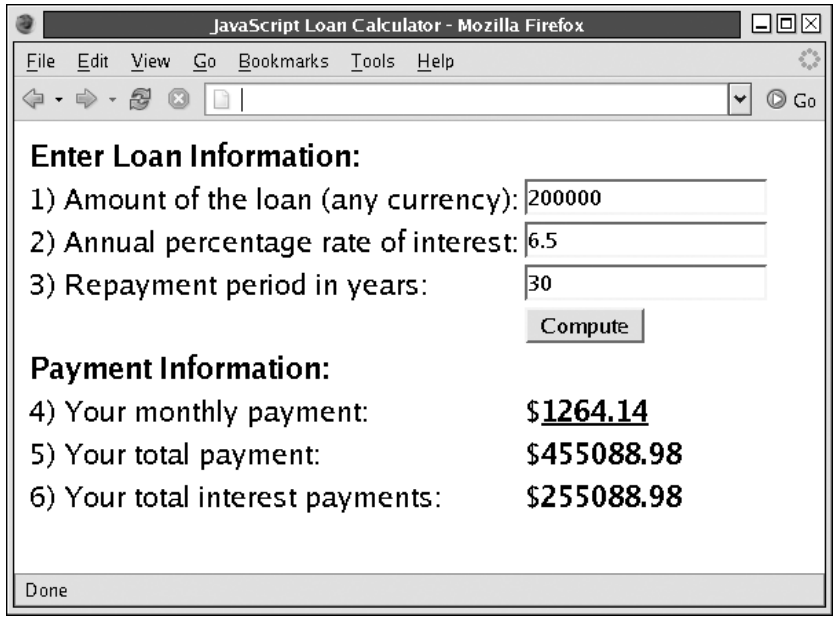
Atribut `onclick`, prikazan u primeru 1-2, sadrži znakovni niz, tj. kôd na JavaScriptu, koji se izvršava kada korisnik pritisne dugme. U ovom slučaju, funkcija za obradu događaja `onclick` poziva funkciju `alert()`. Kao što vidite na slici 1-2, `alert()` otvara okvir za dijalog kako bi se prikazala zadata poruka.

Primeri 1-1 i 1-2 ističu samo najjednostavnije mogućnosti klijentskog JavaScripta. Prava snaga JavaScripta na strani klijenta leži u tome što skriptovi imaju pristup sadržaju HTML dokumenata. Primer 1-3 sadrži potpun, netrivijalan program na JavaScriptu. Ovaj program izračunava mesečnu ratu otplate hipoteke na kuću ili nekog drugog zajma, za zadatu visinu kredita, iznos kamate i period otplate. On čita podatke koje korisnik unosi u polje HTML obrasca, obavlja odgovarajuća izračunavanja s tim podacima, a zatim menja dokument kako bi prikazao rezultate te obrade.



Slika 1-2. Odgovor JavaScripta na događaj

Na slici 1-3 vidi se kako program izgleda kada se prikaže u čitaču Weba. Kao što vidite, on se sastoji od HTML obrasca i još nekog teksta. Međutim, ova slika obuhvata samo statički prikaz programa. Dodavanje JavaScript koda čini ga dinamičkim: kad god korisnik promeni iznos zajma, visinu kamate ili broj rata, JavaScript kôd ponovo izračunava mesečnu ratu, ukupan iznos otplate i ukupnu vrednost kamate koja će biti plaćena.



Slika 1-3. Kalkulator za obračun zajma napisan na JavaScriptu

Prva polovina primera 1-3 jednostavan je CSS opis stilova i HTML obrazac, formatiran u jednoj HTML tabeli. Zapazite da elementi obrasca definišu funkcije za obradu događaja `onchange` ili `onclick`. Čitač Weba aktivira ove funkcije za obradu događaja kada korisnik promeni ulazne podatke, odnosno pritisne dugme **Compute** na obrascu. U svakom slučaju, vrednost atributa funkcije za obradu događaja jeste znakovni niz, tj. JavaScript kôd: `calculate()`. Kada se pokrene funkcija za obradu događaja, ona izvršava ovaj kôd koji poziva funkciju `calculate()`.

Funkcija `calculate()` je definisana u drugoj polovini ovog primera, unutar oznake `<script>`. Ta funkcija čita podatke koje je korisnik uneo u obrazac, obavlja izračunavanja koja su potrebna da bi se obračunala otplata zajma i unosi rezultate obrade u dokument, unutar oznaka `<span>` koje su posebno imenovane pomoću atributa `id`.

Primer 1-3 nije kratak, ali je jednostavan i vredi ga pažljivo proučiti. Ne očekujte da ćete u i ovom trenutku razumeti kompletan kôd, ali su HTML, CSS i JavaScript kôd objašnjeni u komentarima i proučavanjem ovog primera steći ćete osećaj za klijentski JavaScript.*

Primer 1-3. Obračun otplate zajma pomoću JavaScripta

```
<html>
<head>
<title>JavaScript Loan Calculator</title>
<style>
/* Ovo je CSS opis stila: on dodaje stil rezultatu izvršavanja programa */
.result { font-weight: bold; }  /* Za elemente, za koje je class="result" */
#payment { text-decoration: underline; } /* Za elemente, za koje je id="payment" */
</style>
</head>
<body>
<!--
  Ovo je HTML obrazac koji omogućava korisniku da unese podatke i omogućava da
  JavaScript prikaže izračunate rezultate korisniku. Elementi obrasca su umetnuti
  u tabelu da bi lepše izgledali.
  Obrazac je nazvan "loandata" (podaci o zajmu), a poljima obrasca dodeljena su
  imena poput "interest" (kamata) i "years" (godine). Ova imena polja koriste se
  u JavaScript kodu koji prati obrazac.
  Zapazite da su za neke elemente obrasca definisane funkcije za obradu događaja
  "onchange" ili "onclick". One definišu delove JavaScript koda koji se izvršava
  kada korisnik unese podatak, odnosno pritisne dugme.
-->
<form name="loandata">
  <table>
    <tr><td><b>Enter Loan Information:</b></td></tr>
    <tr>
      <td>1) Amount of the loan (any currency):</td>
      <td><input type="text" name="principal" onchange="calculate();"></td>
    </tr>
    <tr>
```

* Ako vam intuicija govori da je loša ideja kombinovati HTML oznake, CSS stilove i kôd na JavaScriptu, niste usamljeni. Mnogi ljudi koji programiraju na JavaScriptu i projektuju za Web, teže da sadržaj, način prikazivanja i ponašanje čuvaju u odvojenim datotekama. U odeljku 13.1.5, objašnjeno je kako se to postiže.

Primer 1-3. Obračun otplate zajma pomoću JavaScripta (nastavak)

```
      <td>2) Annual percentage rate of interest:</td>
      <td><input type="text" name="interest" onchange="calculate();"></td>
    </tr>
    <tr>
      <td>3) Repayment period in years:</td>
      <td><input type="text" name="years" onchange="calculate();"></td>
    </tr>
    <tr><td></td>
      <td><input type="button" value="Compute" onclick="calculate();"></td>
    </tr>
    <tr><td><b>Payment Information:</b></td></tr>
    <tr>
      <td>4) Your monthly payment:</td>
      <td>$<span class="result" id="payment"></span></td>
    </tr>
    <tr>
      <td>5) Your total payment:</td>
      <td>$<span class="result" id="total"></span></td>
    </tr>
    <tr>
      <td>6) Your total interest payments:</td>
      <td>$<span class="result" id="totalinterest"></span></td>
    </tr>
  </table>
</form>

<script language="JavaScript">
/*
 * Ovo je funkcija JavaScripta koja omogućava da navedeni primer radi. Zapazite da
 * je u skriptu definisana funkcija calculate( ) koja se poziva iz funkcija za obradu
 * događaja u obrascu. Ta funkcija čita vrednosti iz <input> polja obrasca, koristeći
 * imena definisana u prethodnom HTML kodu. Ona unosi svoje rezultate u imenovane
 * <span> elemente.
 */
function calculate( ) {
    // Uzima iz obrasca podatke koje unosi korisnik. Pretpostavlja se da su ispravni.
    // Konvertuje kamatu iz procentualnog u decimalni zapis i konvertuje iz godišnjeg
    // u mesečni iznos. Konvertuje period otplate (u godinama) u broj mesečnih rata.

    var principal = document.loandata.principal.value;
    var interest = document.loandata.interest.value / 100 / 12;
    var payments = document.loandata.years.value * 12;

    // Izračunava mesečni iznos, primenjujući čudne formule.
    var x = Math.pow(1 + interest, payments);
    var monthly = (principal*x*interest)/(x-1);

    // Uzima imenovane <span> elemente iz obrasca.
    var payment = document.getElementById("payment");
    var total = document.getElementById("total");
    var totalinterest = document.getElementById("totalinterest");
```

Primer 1-3. Obračun otplate zajma pomoću JavaScripta (nastavak)

```
    // Proverava da li je rezultat konačan broj. Ako jeste, prikazuje
    // rezultate, definišući HTML sadržaj svakog <span> elementa.
    if (isFinite(monthly)) {
        payment.innerHTML = monthly.toFixed(2);
        total.innerHTML = (monthly * payments).toFixed(2);
        totalinterest.innerHTML = ((monthly*payments)-principal).toFixed(2);
    }
    // U suprotnom, korisnik je verovatno uneo pogrešnu vrednost, pa se ne prikazuje
    // ništa.
    else {
        payment.innerHTML = "";
        total.innerHTML = "";
        totalinterest.innerHTML = "";
    }
}
</script>
</body>
</html>
```

1.4 JavaScript u drugim kontekstima

JavaScript je programski jezik opšte namene i ne koristi se samo u čitačima Weba. JavaScript može da bude umetnut u bilo koju aplikaciju i da obezbedi izvršavanje skriptova. U stvari, od prvih dana su Netscapeovi Web serveri uključivali interpreter JavaScripta, tako da su se serverski skriptovi mogli pisati na JavaScriptu. Slično, Microsoft koristi svoj interpreter za JScript u svom IIS Web serveru i u svom proizvodu Windows Scripting Host, pored toga što ga koristi u Internet Exploreru. Adobe koristi jezik izveden iz JavaScripta, za izvršavanje skriptova u programu Flash Player. Kompanija Sun dodaje interpeter za JavaScript prilikom distribuiranja svoje Jave 6. 0, tako da se mogućnosti izvršavanja skriptova lako mogu dodati svakoj aplikaciji napisanoj na Javi (u poglavlju 12 objašnjeno je kako se to radi.)

I Netscape i Microsoft učinili su svoje interpretere za JavaScript dostupnim kompanijama i programerima koji hoće da ih uključe u svoje aplikacije. Netscapeov interpreter je objavljen u vidu otvorenog koda i sada je dostupan preko organizacije Mozilla (videti na Internet adresi *http://www.mozilla.org/js/*). Mozilla zapravo daje dve različite verzije interpretera za JavaScript 1.5. Jedna verzija je napisana na jeziku C i zove se SpiderMonkey. Druga je napisana na Javi i, u jednom laskavom pozivanju na ovu knjigu, zove se Rhino (nosorog).

Ako pišete skriptove za aplikaciju koja ima ugrađen interpreter za JavaScript, od koristi će vam biti prva polovina ove knjige, koja dokumentuje jezgro jezika. Poglavlja koja se odnose na određene čitače Weba, verovatno neće biti primenljiva na vaše skriptove.

1.5 Istraživanje JavaScripta

Nov programski jezik najbolje ćete naučiti ako na njemu pišete programe. Dok čitate ovu knjigu, preporučujem vam da isprobate mogućnosti JavaScripta o kojima učite. Postoji veliki broj tehnika koje olakšavaju eksperimentisanje s JavaScriptom.

Najočigledniji način istraživanja JavaScripta jeste pisanje jednostavnih skriptova. Jedna od prednosti JavaScripta jeste to što svako ko ima čitač Weba i jednostavan program za uređivanje teksta (editor) poseduje potpuno razvojno okruženje: nije potrebno da se kupuje ili preuzima neki softver posebne namene kako bi se počelo pisanje skriptova na JavaScriptu.

Recimo, možete da izmenite primer 1-1 tako da prikazuje Fibonačijeve brojeve umesto faktorijela:

```
<script>
document.write("<h2>Tabela Fibonačijevih brojeva</h2>");
for (i=0, j=1, k=0, fib =0; i<50; i++, fib=j+k, j=k, k=fib){
    document.write("Fibonači ("  + i +  ") = " + fib);
    document.write("<br>");
}
</script>
```

Ovaj kôd je možda zapetljan (ne brinite ako ga još uvek ne razumete), ali je suština sledeća: kada hoćete da eksperimentišete s kratkim programima poput ovog, možete ih otkucati i isprobati u svom čitaču Weba, koristeći sintaksu `file:` URL adresa. Zapazite da se u navedenom kodu koristi metoda `document.write( )` za prikaz HTML izlaza (rezultata) metode, pa možete da vidite rezultate izračunavanja. Ovo je važan način eksperimentisanja s JavaScriptom. Druga mogućnost je da koristite metodu `alert( )` kako biste prikazali tekstualni rezultat u okviru za dijalog:

```
alert("Fibonači ("  + i +  ") = " + fib);
```

Zapazite i sledeće: u jednostavnim eksperimentima s JavaScriptom, poput ovoga, iz svoje HTML datoteke možete da izostavite oznake `<html>`, `<head>` i `<body>`.

Za još jednostavnije eksperimente s JavaScriptom, nekada možete koristiti pseudoprotokol `javascript:` URL adresa, kako biste izračunali neki izraz pomoću JavaScripta i prikazali rezultat. URL adresa se u JavaScriptu sastoji od specifikatora protokola `javascript:` iza koga sledi proizvoljan kôd na JavaScriptu (naredbe su međusobno odvojene znakom tačka i zarez). Kada čitač pročita takvu URL adresu, on izvršava JavaScript kôd. Vrednost poslednjeg izraza na toj URL adresi konvertuje se u znakovni niz koji se prikazuje u čitaču Weba kao novi dokument. Na primer, možete da unesete sledeće URL adrese JavaScripta u polje **Location** svog čitača Weba, kako biste proverili da li razumete neke operatore i naredbe JavaScripta:

```
javascript:5%2
javascript:x = 3; (x < 5)? "x je manje": "x je veće"
javascript:d = new Date( ); typeof d;
javascript:for(i=0,j=1,k=0,fib=1; i<5; i++,fib=j+k,k=j,j=fib) alert(fib);
javascript:s=""; for(i in navigator) s+=i+":"+navigator[i]+"\n"; alert(s);
```

U čitaču Firefox takođe možete da eksperimentišete s jednim redom koda u konzoli JavaScripta, kojoj pristupate preko menija **Tools**. Samo otkucajte izraz koji hoćete da izračunate ili naredbu koju hoćete da izvršite. Kada umesto polja za URL adresu koristite konzolu, izostavite prefiks `javascript:` iz adrese.

Dok istražujete JavaScript, verovatno ćete pisati kôd koji ne radi onako kako očekujete i želećete da pronađete grešku. Osnovni način za pronalaženje grešaka u JavaScriptu sličan je onom u mnogim drugim jezicima: u svoj kôd dodajte naredbe za prikazivanje vrednosti značajnih promenljivih, tako da možete pokušati da otkrijete šta se stvarno dešava. Kao što je prethodno prikazano, u tu svrhu možete koristiti metode `document.write()` ili `alert()`. (U primeru 15-9 opisan je napredniji način za evidentiranje poruka koje služe za otkrivanje grešaka.)

Petlja `for/in` (opisana u poglavlju 6) takođe je korisna prilikom pronalaženja grešaka. Na primer, možete da je primenite, uz metodu `alert()`, kako biste prikazali listu naziva i vrednosti svih svojstava nekog objekta. Ova vrsta funkcije može da bude pogodna kada istražujete jezik ili pokušavate da otklonite greške iz koda.

Ako su greške u programima na JavaScriptu uporne i ako se pogoršavaju, možda će vam zatrebati neki pravi program za njihovo pronalaženje i otklanjanje. U Internet Exploreru možete koristiti Microsoft Script Debugger. U Firefoxu možete da upotrebite dodatni program za pronalaženje grešaka, pod nazivom Venkman. Obe alatke su izvan opsega ove knjige, ali ćete ih pronaći na Internetu. Još jedno korisno sredstvo, koje nije isključivo program za pronalaženje grešaka, jeste verifikator koda *jslint*, koji traži uobičajene probleme u kodu napisanom na JavaScriptu (videti *http://jslint.com*).