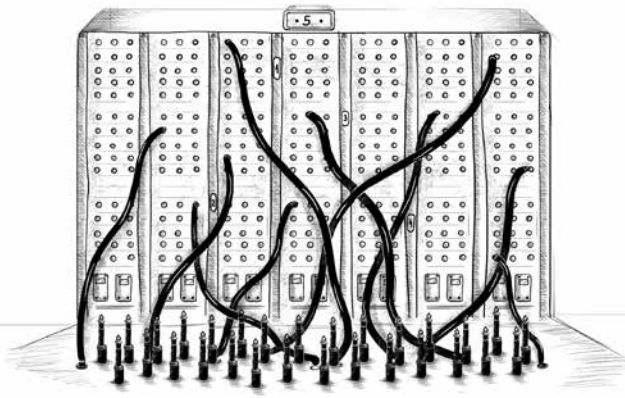
DEO II

ČITAČ VEBA

„San u osnovi veba jeste san o zajedničkom informacionom prostoru u kojem komuniciramo razmenjivanjem informacija. Njegova univerzalnost je neophodna: činjenica da hiperveza može da pokaže bilo šta, bilo to lično, lokalno ili globalno, bilo to samo nacrt ili savršeno uglancano."

– Tim Berners-Lee,
The World Wide Web: A very short personal history



13

JAVASCRIPT I ČITAČ VEBA

Naredna poglavlja ove knjige govoriće o čitačima veba. Bez njih ne bi bilo ni JavaScripta. A čak i kada bi ga bilo, niko na njega ne bi obraćao pažnju.

Veb tehnologija je bila decentralizovana od samog početka, ne samo tehnički već i po načinu na koji se razvijala. Razni proizvođači čitača dodavali su nove funkcionalnosti, *ad hoc* i ponekad slabo promišljeno, što su onda, ponekad, prihvatali ostali – i na kraju je bilo zapisano kao standard.

To je istovremeno i blagoslov i prokletstvo. S jedne strane, podsticajno je da ne postoji jedna glavna strana koja upravlja sistemom, već ga unapređuju razne strane koje rade u labavoj saradnji (ili, povremeno, u otvorenom neprijateljstvu). S druge strane, opasan način na koji se veb razvio znači da rezultujući sistem nije baš najsjajniji primer unutrašnje doslednosti. Neki njegovi delovi su dozlaboga zbunjujući i loše zamišljeni.

Mreže i internet

Računarske mreže postoje otprilike od pedesetih godina prošlog veka. Ako postavite kablove između dva ili više računara, i dozvolite im da jedni drugima šalju podatke preko tih kablova, možete raditi razne vrste divnih stvari.

A ako povezivanje dve mašine u istoj zgradi omogućava da radimo divne stvari, povezivanje mašina širom planete trebalo bi da bude još bolje.

Tehnologija za početak implementiranja ove vizije razvijena je osamdesetih godina prošlog veka, a dobijena mreža nazvana je *internet*. Ispunila je očekivanja.

Računar može da koristi tu mrežu da bi ispaljivao bitove ka drugom računaru. Da bi iz tog ispaljivanja bitova izronila ikakva efikasna komunikacija, računari na oba kraja moraju znati šta ti bitovi predstavljaju. Značenje bilo koje date sekvence bitova u potpunosti zavisi od vrste stvari koju ona pokušava da izrazi i od upotrebljenog mehanizma kodiranja.

Mrežni protokol (engl. *network protocol*) opisuje stil komunikacije na mreži. Postoje protokoli za slanje elektronske pošte, za pozivanje e-pošte, za razmenjivanje datoteka, čak i za upravljanje računarima koji su inficirani zloćudnim softverom.

Na primer, *Hypertext Transfer Protocol* (HTTP) protokol je za preuzimanje imenovanih resursa ili izvora (komada informacija, kao što su veb strane ili slike). On zadaje da strana koja šalje zahtev treba da počne ovakvim redom, imenujući izvor i verziju protokola koji pokušava da koristi:

```
GET /index.html HTTP/1.1
```

Postoji još mnogo pravila u vezi sa načinom na koji zahtevalac može da uključi još informacija u zahtev i načinom na koji druga strana, koja vraća izvor, pakuje svoj sadržaj. HTTP ćemo detaljnije razmatrati u poglavlju 18.

Većina protokola je izgrađena povrh drugih protokola. HTTP tretira mrežu kao uređaj nalik potoku u koji možete staviti bitove tako da oni stignu na ispravno odredište ispravnim redosledom. Kao što smo videli u poglavlju 11, i samo obezbeđivanje toga je krajnje težak problem.

Protokol za upravljanje prenosom (engl. *Transmission Control Protocol*, *TCP*) protokol je koji se bavi tim problemom. „Govore“ ga svi uređaji povezani u internet i većina komunikacije na internetu izgrađena je na njemu.

TCP konekcija radi na sledeći način: jedan računar mora da čeka, ili *osluškuje*, dok drugi računari ne počnu da govore s njim. Da bi na jednoj mašini mogao da osluškuje različite vrste komunikacije u isto vreme, svakom osluškivaču dodeljen je broj (koji se naziva *port*). Većina protokola unapred zadaje port koji treba da se koristi. Na primer, kada želimo da pošaljemo e-poruku koristeći protokol SMTP, mašina kroz koju je šaljemo treba da osluškuje na portu 25.

Drugi računar tada može da uspostavi konekciju sa ciljnom mašinom koristeći odgovarajući broj porta. Ukoliko se može doći do ciljne mašine i ona osluškuje taj port, konekcija je uspešno napravljena. Računar osluškivač naziva se *server*, a računar koji se povezuje sa njim naziva se *klijent* (engl. *client*).

Takva konekcija funkcioniše kao dvosmerna cev kroz koju mogu da teku bitovi – mašine na oba kraja mogu da ubacuju podatke u nju. Kada se bitovi uspešno prenesu, mašina na drugom kraju može da ih pročita. To je zgodan model. Moglo bi se reći da TCP omogućava apstrahovanje mreže.

Veb

Globalna računarska mreža, *World Wide Web* (koji ne treba mešati sa internetom kao celinom) skup je protokola i formata koji nam omogućavaju da posećujemo veb strane preko čitača veba (engl. *browser*). „Mreža“ (engl. *web*) u imenu odnosi se na činjenicu da se takve strane mogu lako povezati jedna s drugom i tako napraviti ogromnu mrežu kroz koju korisnici mogu da se kreću.

Da biste postali deo veba, potrebno je samo da povežete mašinu sa internetom i kažete joj da osluškuje na portu 80 sa HTTP protokolom, tako da drugi računari mogu od nje da traže dokumenta.

Svaki dokument na vebu imenovan je *jedinstvenim identifikatorom resursa* (engl. *Uniform Resource Locator, URL*), koji izgleda otprilike ovako:

```
http://eloquentjavascript.net/13_browser.html
|      |                      |              |
protokol       server             putanja
```

Prvi deo nam govori da ovaj URL koristi HTTP protokol (a ne, na primer, šifrovani, bezbedni HTTP, koji bi bio *https://*). Sledi deo koji označava od kog servera tražimo dokument. Poslednji je niz sa putanjom koji označava određeni dokument (ili *resurs*) za koji smo zainteresovani.

Mašine povezane na internet dobijaju *IP adresu*, a to je broj koji se može koristiti za slanje poruka toj mašini, i ona izgleda ovako: `149.210.142.219`, ili ovako: `2001:4860:4860::8888`. Međutim, liste manje-više nasumičnih brojeva teško se pamte i nezgodne su za upisivanje, pa umesto njih možete registrovati ime domena za određenu adresu ili skup adresa. Ja sam registrovao ime domena *eloquentjavascript.net* koja pokazuje IP adresu mašine kojom ja upravljam i to ime mogu da koristim da bih isporučivao veb strane.

Ako taj URL upišete u adresno polje svog čitača veba, čitač će pokušati da pozove i prikaže dokument na tom URL-u. Vaš čitač prvo mora da otkrije na koju se adresu odnosi *eloquentjavascript.net*. Potom, koristeći HTTP protokol, on se povezuje sa serverom na toj adresi i traži resurs */13_browser.html*. Ako sve prođe kako treba, server šalje dokument koji vaš čitač prikazuje na ekranu.

HTML

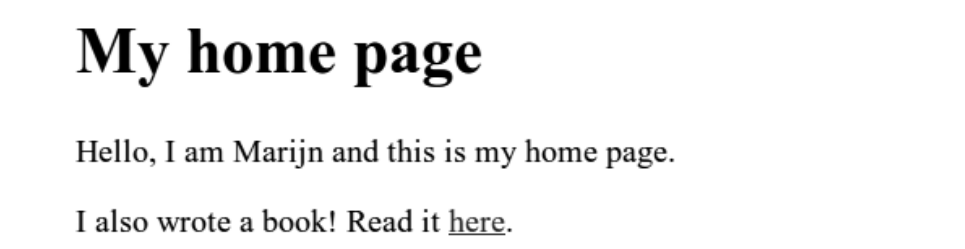
HTML, što je skraćenica za *Hypertext Markup Language* – jezik za označavanje hiperteksta, format je dokumenata koji se koristi za veb strane. HTML dokument sadrži tekst, kao i *oznake* (engl. *tags*) koje daju strukturu tekstu opisujući stvari kao što su hiperveze, pasusi i naslovi.

Kratak HTML dokument mogao bi izgledati ovako:

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My home page</title>
  </head>
```

```
  <body>
    <h1>My home page</h1>
    <p>Hello, I am Marijn and this is my home page.</p>
    <p>I also wrote a book! Read it
      <a href="http://eloquentjavascript.net">here</a>.</p>
  </body>
</html>
```

Takav dokument bi u čitaču izgledao ovako:



Oznake, postavljene u uglaste zagrade (< i >, znake za *manje od* i *veće od*), pružaju informacije o strukturi dokumenta. Preostali tekst je običan tekst.

Dokument počinje sa `<!doctype html>`, što čitaču govori da stranu tumači kao *savremeni* HTML, za razliku od raznih dijalekata koji su bili u upotrebi u prošlosti.

HTML dokumenti imaju zaglavlje i telo. Zaglavlje sadrži informacije *o* dokumentu, a telo sadrži sâm dokument. U ovom slučaju, zaglavlje deklariše da je naslov ovog dokumenta „My home page“ i da on koristi kodiranje UTF-8, što je način da se Unicode tekst kodira kao binarni podaci. Telo dokumenta sadrži naslov (`<h1>`, od „heading 1“, tj. „naslov 1“; `<h2>` do `<h6>` daju podnaslove) i dva pasusa (`<p>`).

Oznake mogu imati nekoliko oblika. Element (kao što je telo, pasus ili hiperveza) počinje *početnom oznakom* (engl. *opening tag*) kao što je `<p>`, a završava se *završnom oznakom* (engl. *closing tag*) kao što je `</p>`. Neke početne oznake, kao što je oznaka za hipervezu (`<a>`), sadrže dodatne informacije u vidu parova `ime="vrednost"`. One se nazivaju *atributi*. U ovom slučaju, odredište hiperveze je naznačeno sa `href="http://eloquentjavascript.net"`, gde `href` predstavlja „hypertext reference“ – referencu hiperteksta.

Neke vrste oznaka ne obuhvataju ništa drugo, pa ne moraju da imaju završni deo. Oznaka za metapodatke, `<meta charset="utf-8">` primer je takve oznake.

Pošto uglaste zagrade u u HTML-u imaju posebno značenje, da bi mogle da se koriste u tekstu, mora se uvesti još jedan oblik specijalne notacije. Obična otvorena uglasta zagrada („manje od“) piše se kao `&lt;`, a zatvorena uglasta zagrada („veće od“) piše se kao `&gt;`. U HTML-u, znak ampersend (&) nakon kog sledi ime ili kôd znaka i tačka i zarez (;) naziva se *entitet* i biće zamenjen znakom koji kodira.

To je analogno načinu na koji se obrnute kose crte koriste u JavaScriptovim znakovnim nizovima. Pošto ovaj mehanizam i znaku ampersend daje specijalno značenje, taj znak u tekstu morate pisati kao `&amp;`. U vrednostima atributa, koje su postavljene u navodnike, `&quot;` se može koristiti da bi se ubacio sâm navodnik.

Raščlanjivanje HTML-a ima veliku toleranciju za greške. Kada nedostaju oznake koje bi trebalo da postoje, čitač veba ih rekonstruiše. Način na koji se to radi je standardizovan, i možete se uzdati da će svi savremeni čitači to činiti na isti način.

Naredni dokument će biti tretiran potpuno isto kao i onaj prethodni:

```
<!doctype html>
<meta charset=utf-8>

<title>My home page</title>
<h1>My home page</h1>

<p>Hello, I am Marijn and this is my home page.
<p>I also wrote a book! Read it
  <a href=http://eloquentjavascript.net>here</a>.
```

Oznake `<html>`, `<head>` i `<body>` potpuno su nestale. Čitač zna da `<meta>` i `<title>` pripadaju zaglavlju, a da `<h1>` znači da je počelo telo. Štaviše, nisam izričito zatvorio pasuse, pošto će ih otvaranje novog pasusa ili završavanje dokumenta implicitno zatvoriti. Navodnici oko vrednosti atributa takođe su nestali.

U ovoj knjizi će iz primera obično biti izostavljene oznake `<html>`, `<head>` i `<body>` da bi bili kraći i manje zakrčeni. Ali zatvaraću oznake i postavljati navodnike oko atributa.

Pored toga, obično ću izostavljati deklaraciju `doctype` i `charset`. Time vas ne ohrabrujem da ih izostavljate iz HTML dokumenata. Čitači često rade smešne stvari kada ih zaboravite. Smatrajte da `doctype` i `charset` metapodaci uvek postoje u primerima, čak i kada se ne vide u tekstu.

HTML i JavaScript

U kontekstu ove knjige, najbitnija HTML oznaka je `<script>`. Ona nam omogućava da u dokument uključimo JavaScript.

```
<h1>Testing alert</h1>
<script>alert("hello!");</script>
```

Takav skript će biti izvršen čim čitač, pri čitanju HTML-a, naiđe na njegovu oznaku `<script>`. Kada se ova strana otvori, na njoj će iskočiti okvir za dijalog – funkcija `alert` podseća na `prompt` po tome što otvara mali prozor, ali ona samo prikazuje poruku, ne tražeći unos.

Postavljanje velikih programa direktno u HTML dokument često je nepraktično. Oznaka `<script>` može da dobije atribut `src` da bi uzela datoteku skripta (tekstualnu datoteku koja sadrži JavaScript program) sa nekog URL-a.

```
<h1>Testing alert</h1>
<script src="code/hello.js"></script>
```

Datoteka *code/hello.js* koja je ovde dodata sadrži isti program – `alert("hello!")`. Kada HTML strana referencira druge URL-ove kao deo te strane – na primer, datoteku slike ili skript – čitači veba će ih odmah pozvati i uključiti ih u stranu.

Oznaka za skript uvek mora biti završena sa `</script>`, čak i ako se odnosi na datoteku skripta i ne sadrži nikakav kôd. Ako to zaboravite, ostatak strane će biti protumačen kao deo skripta.

U čitač možete učitati ES module (pročitajte „ECMAScript moduli“ na strani 167) tako što ćete oznaci za skript dati atribut `type="module"`. Takvi moduli mogu zavisiti od drugih modula tako što će u deklaracijama `import`, kao imena modula koristiti URL-ove koji su relativni u donosu na njih.

Neki atributi mogu da sadrže i JavaScript program. Oznaka `<button>`, prikazana u nastavku (koja prikazuje dugme) ima atribut `onclick`. Vrednost tog atributa biće pokrenuta svaki put kada se dugme pritisne.

```
<button onclick="alert('Boom!');">NE PRITISKAJ</button>
```

Primetićete da sam upotrebio polunavodnike za znakovni niz u atributu `onclick` jer se navodnici već koriste za navođenje celog atributa. Mogao sam da upotrebim i `&quot;`.

U izolovanom okruženju

Pokretanje programa preuzetih sa interneta može biti opasno. Ne znate mnogo o ljudima koji stoje iza većine lokacija koje posećujete i ne mora značiti da vam svi žele dobro. Pokretanje programa ljudi koji vam ne žele dobro način je da svoj računar zarazite virusima, da vam podaci budu ukradeni i nalozi hakovani.

Pa ipak, privlačnost veba je u tome što možete da ga pregledate ne morajući da verujete svim stranama koje posetite. Zbog toga čitači oštro ograničavaju stvari koje JavaScript program može da uradi: on ne može da pregleda datoteke na vašem računaru niti da menja išta što nije povezano sa veb stranom u koju je ugrađen.

Ovakvo izolovanje programskog okruženja je poput ostavljanja programa da se „igra“ u kutiji s peskom (engl. *sandbox*). Međutim, ovu konkretnu vrstu kutije s peskom treba da zamislite kao da je pokrivena kavezom od debelih gvozdenih šipki, tako da programi koji se u njoj igraju ne mogu da izađu.

Teži deo postavljanja u izolaciju jeste obezbeđivanje programima dovoljno prostora da budu korisni, a u isto vreme sprečiti ih da urade išta opasno. Mnogo korisnih funkcionalnosti, kao što je komunikacija s drugim serverima ili čitanje sadržaja sa klipborda, takođe se može upotrebiti za problematične stvari koje narušavaju privatnost.

Svako malo, neko smisli nov način da zaobiđe ograničenja čitača i uradi nešto štetno, od otkrivanja nebitne lične informacije, do preuzimanja cele mašine na kojoj čitač radi. Programeri čitača reaguju popravljajući rupu i sve je opet kako treba – sve dok se ne otkrije sledeći problem i, nadamo se, objavi javnosti, umesto da ga u tajnosti eksploatiše neka vladina agencija ili mafija.

Kompatibilnost i ratovi čitača

U ranim fazama veba, čitač Mosaic dominirao je tržištem. Nakon nekoliko godina, prevagu je odneo Netscape, kojeg je potom, zauzvrat, uveliko premašio Microsoftov Internet Explorer. Kad god je samo jedan čitač bio dominantan, njegov proizvođač uzimao je za pravo da jednostrano izmišlja nove mogućnosti za veb. Pošto je većina korisnika koristila najpopularniji čitač, veb lokacije bi prosto počele da koriste te mogućnosti – ko mari za ostale čitače.

To je bilo mračno doba kompatibilnosti, često nazivano *ratovi čitača veba*. Veb programeri nisu dobili jedan ujedinjeni veb, već dve ili tri nekompatibilne platforme. Da bi stvari bile još gore, čitači koji su se koristili oko 2003. bili su puni programskih grešaka, a, naravno, greške su se razlikovale za svaki čitač. Život nije bio lak za ljude koji su čekali veb strane.

Mozilla Firefox, neprofitni izdanak Netscapea, ugrozio je položaj Internet Explorera krajem prve decenije ovog veka. Pošto Microsoft u to vreme nije bio naročito zainteresovan da ostane konkurentan, Firefox je mu je oduzeo veliki deo tržišta. Otprilike u isto vreme, Google je predstavio svoj čitač Chrome, a Appleov čitač Safari stekao je popularnost, dovevši do situacije u kojoj su postojala četiri velika igrača, umesto jednog.

Novi igrači su imali ozbiljnije stavove o standardima i bolje prakse u projektovanju, dajući nam manje nekompatibilnosti i manje programskih grešaka. Microsoft, videvši da se njegov udeo u tržištu smanjuje, dozvao se pameti i prihvatio te stavove u svom čitaču Edge, koji zamenjuje Internet Explorer. Ako danas počinjete da učite veb programiranje, smatrajte se srećnim. Najnovije verzije glavnih čitača ponašaju se prilično ujednačeno i imaju relativno malo grešaka.