JAVASCRIPT: PONAŠANJE VEB STRANA

DEO IV

U OVOM DELU

UVOD U JAVASCRIPT

Autor: Mat Marquis

U OVOM POGLAVLJU

Ovo poglavlje sadrži uvod u JavaScript. Ukoliko zazirete od toga, razumljivo je. Prelazimo na „opasnu teritoriju“ pravog programskog jezika, a to može biti pomalo zastrašujuće. Ipak, nije tako strašno, obećavam!

Počećemo od toga šta JavaScript jeste – a šta nije – i razmotriti neke načine na koje se on koristi. U najvećem delu poglavlja upoznajete se sa sintaksom JavaScripta – promenljivama, funkcijama, operatorima, petljama i slično. Da li ćete pisati kôd kada završite čitanje ovog poglavlja? Verovatno nećete.

Ali, steći ćete dobru osnovu i bolje ćete razumeti šta se dešava u skriptu na koji naiđete. Poglavlje se završava razmatranjem nekih načina na koje možete da manipulišete prozorom čitača i da povezujete skriptove sa akcijama korisnika kao što su pritiskanje mišem ili slanje obrasca.

Šta je JavaScript?

Ako ste stigli do ovog dela knjige, verovatno već znate da je JavaScript programski jezik koji je zaslužan za interaktivnost i namenski definisana ponašanja veb lokacija. On je *jezik za skriptovanje na klijentskoj strani*, što znači da se izvršava na korisnikovom računaru a ne na serveru kao drugi programski jezici, npr. PHP i Ruby. To znači da se JavaScript (i načini na koje ga koristimo) oslanja na mogućnosti i parametre čitača veba. Moguće je čak i da ne bude dostupan, i to ili zato što ga je korisnik isključio ili zato što ga dati uređaj ne podržava – što je nešto čega su dobri programeri svesni i o čemu vode računa. JavaScript je poznat i kao *dinamički* i *labavo tipiziran* (engl. *loosely typed*) programski jezik. Ne razmišljajte mnogo o ovim opisima – kasnije ću objasniti šta oni znače.

Najpre želim da pokažem da je JavaScript donekle pogrešno shvaćen.

Šta JavaScript nije

Kao prvo, njegovo ime prilično zbunjuje jer JavaScript nema nikakve veze s Javom. Napravio ga je Brendan Ajh u kompaniji Netscape 1995. godine i prvobitno ga nazvao „LiveScript“. Ali, pošto je u to vreme Java bila glavni hit, „LiveScript“ je iz marketinških razloga postao „JavaScript“. Ili samo „JS“, ukoliko želite da zvučite najopuštenije moguće dok pričate o JavaScriptu.

JS ima i pomalo lošu reputaciju. Jedno vreme je bio sinonim za najrazličitije bezočne prevare na internetu – neželjene redirekcije, dosadne iskačuće prozore i brojna nebulozna „ugrožavanja sigurnosti“, da pomenemo samo neke. Nekada je JavaScript omogućavao nesavesnim programerima da rade sve to (pa i gore stvari), ali su savremeni čitači veba uglavnom uspeli da se izbore s tom tamnom stranom programiranja u JavaScriptu i onemoguće je. Ipak, ne bi trebalo da okrivljujemo sâm JavaScript za takvo stanje. Kao što ne tako stara izreka kaže: „Uz veliku moć ide i velika odgovornost“. JavaScript je oduvek pružao programerima ogromnu mogućnost kontrole nad načinom prikazivanja strana i ponašanjem čitača, pa je na nama da je koristimo odgovorno.

Šta JavaScript jeste

NAPOMENA

JavaScript je 1996. godine standardizovalo Evropsko udruženje proizvođača računara (European Computer Manufacturer's Association, ECMA), pa ga ponekad zovu i ECMAScript.

Sada znamo šta JavaScript nije: nije povezan s Javom i nije brkati zlikovac koji vas vreba iz čitača veba, kršeći ruke i čekajući da vas obavesti o „atraktivnim samcima u vašoj okolini“. Hajde da vidimo šta JavaScript jeste.

JavaScript je ne mnogo složen ali izuzetno moćan jezik za pisanje skriptova. Najčešće ga srećemo posredstvom čitača veba, ali se probio svuda – od ugrađenih aplikacija, preko PDF-ova do e-knjiga. Čak i veb servere može pokretati JavaScript.

Kao *dinamički programski jezik*, JavaScript ne mora da se izvršava uz posredovanje bilo kakvog kompajlera koji prevodi naš kôd razumljiv ljudima u nešto što razume čitač veba. Čitač efikasno čita kôd isto kao i mi, i interpretira ga u hodu.

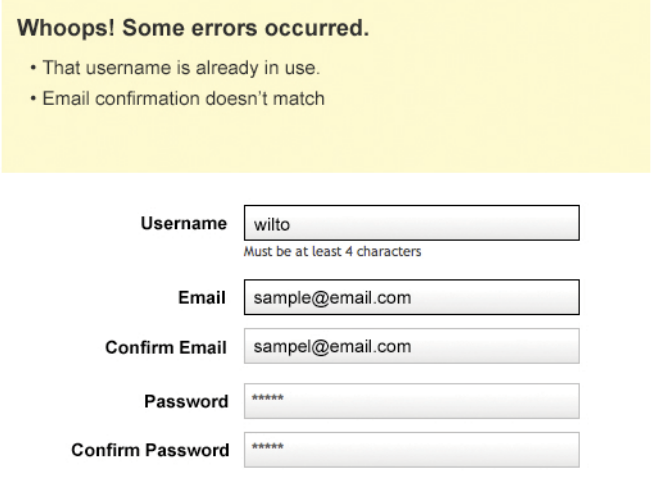
JavaScript je, takođe, *labavo tipiziran jezik*. To znači da ne moramo da saopštimo JavaScriptu tip promenljive. Ako promenljivoj dodeljujemo vrednost 5, ne moramo programski definisati tu promenljivu kao broj. Kao što ste možda i sami primetili, 5 je broj i JavaScript ga prepoznaje kao takvog.

Ipak, ne morate pamtiti pomenute termine da biste počeli da pišete JS kôd – iskreno, ja nisam. Navodim ih samo da bih vam predstavio nekoliko pojmova koje ćete često sretati dok budete učili JavaScript, i koji će s vremenom postajati sve smisleniji.

Šta JavaScript može

JavaScript se najčešće koristi za dodavanje interaktivnosti veb strani. Dok „strukturni“ sloj strane čine HTML oznake a „prezentacioni“ CSS stilovi, za treći sloj – „ponašanje“ – zaslužan je JavaScript. Skriptovi mogu pristupati svim elementima, atributima i tekstu na veb strani pomoću objektnog modela dokumenta (engl. *Document Object Model*, *DOM*), koji ćemo opisati u poglavlju 20, *Korišćenje JavaScripta*. Možemo pisati i skriptove koji reaguju na korisnikov unos, u hodu menjajući sadržaj strane, CSS stilove ili ponašanje čitača.

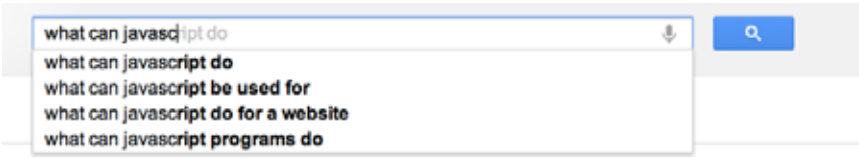
Verovatno ste videli kako to funkcioniše ako ste ikada pokušali da se registrujete na neku veb lokaciju, uneli korisničko ime i odmah dobili povratnu informaciju da to ime već koristi neko drugi (slika 19-1). Crno uokvireno polje za unos teksta i prikazivanje poruke tipa „sorry, this username is already in use“, primeri su toga kako JavaScript menja sadržaj strane i sprečava slanje obrasca, menjajući tako podrazumevano ponašanje čitača veba.



Slika 19-1. *JavaScript prepoznaje da korisničko ime nije na raspolaganju, i zatim umeće poruku i menja stilove kako bi problem postao očigledan.*

Ukratko, JavaScript vam omogućava da napravite interfejse veoma brzog odziva, koji poboljšavaju ukupno korisničko iskustvo i obezbeđuju dinamičko funkcionisanje bez čekanja da server učita novu stranu. Na primer, pomoću JavaScripta možemo uraditi bilo šta od sledećeg:

- Predložiti kompletan pojam koji je korisnik počeo da unosi u polje za pretragu – dok ga unosi. To možete videti na Google.com (slika 19-2).



Slika 19-2. *Google.com koristi JavaScript za automatsko dopunjavanje traženog pojma dok se on unosi.*

Slika 19-3. *JavaScript se može koristiti za prikazivanje i skrivanje delova sadržaja.*

- Tražiti sadržaj i informacije od servera i ubaciti ih u tekući dokument prema potrebi, ne učitavajući ponovo celu stranu – što je poznato i kao tehnologija „Ajax".
- Prikazati i sakriti sadržaj kada korisnik pritisne vezu ili naslov, i tako dobiti „stapajuću" oblast za sadržaj (slika 19-3).
- Ispitati osobine i mogućnosti pojedinačnih čitača. Na primer, možete ispitati postoje li „događaji dodira", koji ukazuju na interakciju korisnika sa stranom preko čitača veba na mobilnom uređaju, pa dodati pogodnije stilove i metode interakcije.
- Popuniti praznine tamo gde ugrađene mogućnosti čitača ne zadovoljavaju potrebe, ili starijim čitačima dodati neke mogućnosti iz novijih. Takve vrste skriptova obično se nazivaju *shims* ili *polyfills*.
- Učitati sliku ili sadržaj u namenski stilizovan, modalni pano za prikaz (engl. *lightbox*) – izolovan na strani pomoću CSS-a – nakon što korisnik pritisne umanjenu verziju date slike (slika 19-4).

Ova lista nije ni izbliza detaljna!

Slika 19-4. *JavaScript se može koristiti za učitavanje slika u galerije s panoima za prikaz.*

Dodavanje JavaScript koda veb strani

Kao i u slučaju CSS-a, skript možete ugraditi direktno u dokument ili ga držati u spoljnoj datoteci i povezati sa stranom. U obe metode koristi se element **script**.

Ugrađen skript

Da biste skript ugradili u stranu, samo dodajte odgovarajući kôd kao sadržaj elementa **script**:

```
<script>
… Ovde ide JavaScript kôd
</script>
```

Spoljni skriptovi

U drugoj metodi koristi se atribut **src** za pokazivanje na datoteku sa skriptom (sa nastavkom *.js*) pomoću njenog URL-a. U ovom slučaju, element **script** je prazan.

```
<script src="my_script.js"></script>
```

Prednost spoljnih skriptova je to što isti skript možete primeniti na više strana (ista prednost koju pružaju i spoljni opisi stilova). Naravno, nedostatak je to što se za svaki skript upućuje zaseban HTTP zahtev serveru, što usporava rad.

Položaj skripta

Element **script** može da se nalazi bilo gde u dokumentu, ali se skriptovi najčešće smeštaju unutar eelementa **head** dokumenta i na sam kraj elementa **body**. Preporučljivo je da ih ne razbacijute svuda po dokumentu jer bi se onda teško pronalazili i održavali.

Za većinu skriptova, najpogodnije mesto je kraj dokumenta, neposredno ispred oznake **</body>**, zato što će čitač veba tada već završiti s raščlanjavanjem (analiziranjem) dokumenta i njegove DOM strukture. Znači, te informacije će biti spremne i dostupne u vreme kada čitač dođe do skriptova, pa se oni mogu brže izvršiti. Osim toga, učitavanje i izvršavanje skriptova blokira iscrtavanje strane na ekranu, pa performanse strane posetiocu deluju bolje kada se skriptovi izvršavaju na kraju. Ipak, možda ćete ponekad želeti da skript obavi nešto pre nego što se telo dokumenta potpuno učita, pa ćete smeštanjem skripta u element **head** dobiti bolje performanse.

NAPOMENA

U verziji HTML 4.01, oznaka **script** *mora da sadrži atribut* **type** *da bi bila ispravna:*

```
<script type="text/
javascript">…</script>
```

U XHTML dokumentima, sadržaj elementa **script** *morate identifikovati kao CDATA kôd u sledećem omotaču:*

```
<script type="text/javascript">
  // <![CDATA[
  …Ovde ide JavaScript kôd
  // ]]>
</script>
```

Anatomija skripta

Postoji razlog zbog kojeg knjiga Dejvida Flanagana *JavaScript: sveobuhvatni vodič* (O'Reilly, Mikro knjiga) ima preko 1000 strana. Ima mnogo toga da se kaže o JavaScriptu! U ovom odeljku, na raspolaganju nam je samo nekoliko strana da vas upoznamo sa osnovnim gradivnim elementima JavaScripta

kako biste razumeli skriptove kada naiđete na njih. Mnogi veb programeri su sami naučili da koriste JavaScript tako što su pronalazili postojeće skriptove i prilagođavali ih svojim potrebama. Nakon malo prakse, bili su spremni da počnu da pišu sopstvene skriptove od nule. Možda i vi želite da naučite da sami pišete JavaScript kôd kako biste zaokružili svoja umeća veb dizajnera. Prvi korak ka tome je prepoznavanje delova skripta, pa ćemo odatle i početi.

```
alert("Hi there");
```


```
confirm("I'm gonna do something, okay?");
```


```
prompt("What should I do?");
```

Slika 19-5. *Ugrađene JavaScript funkcije:* **`alert()`** *(gore),* **`confirm()`** *(u sredini) i* **`prompt()`** *(dole).*

Funkcionalnost JavaScripta prvobitno je bila uglavnom ograničena na sirove metode interakcije s korisnikom. Mogli smo koristiti nekoliko ugrađenih funkcija JavaScripta (slika 19-5) da bismo omogućili dobijanje povratnih informacija od korisnika; takve su bile funkcija **`alert()`** za slanje obaveštenja korisniku i **`confirm()`** pomoću koje smo tražili da korisnik odobri ili odbije akciju. Kada je trebalo zahtevati da korisnik unese nešto, najčešće smo bili ograničeni na ugrađenu funkciju **`prompt()`**. Mada ove metode i danas imaju svoje mesto, one predstavljaju neprijatan, nametljiv i – bar po uobičajenom shvatanju – prilično grozan način interakcije s korisnikom. S razvojem JavaScripta, dobili smo mnogo suptilnije načine za definisanje ponašanja veb strana, što je doprinelo i boljem iskustvu korisnika u radu s njima.

Da bismo iskoristili prednosti tih metoda interakcije, prvo moramo razumeti logiku na kojoj se zasniva skriptovanje. To su logički obrasci (engl. *logic patterns*) zajednički svim programskim jezicima, bez obzira na njihovu različitu sintaksu (slično kao što se reči u govornim jezicima najčešće razlikuju, ali većina njih deli mnoge gramatičke obrasce).

Do kraja ovog odeljka, upoznaćete promenljive, nizove, operatore poređenja, naredbe if/else, petlje, funkcije i još mnogo toga. Spremni?

Osnove

Nekoliko uobičajenih sintaksnih pravila važi za ceo JavaScript.

Važno je znati da JavaScript *razlikuje mala i velika slova*. Promenljive po imenu „mojaPromenljiva“, „mojapromenljiva“ i „MOJAPromenljiva“ tretiraće se kao tri različita objekta. Pored toga, beline kao što su tabulatori i razmaci se ignorišu, osim ako su deo znakovnog niza (teksta) i obuhvaćeni navodnicima.

JavaScript razlikuje mala i velika slova.

Naredbe

Skript se sastoji od niza *naredaba* (engl. *statements*). Naredba je komanda kojom se nalaže čitaču šta da radi. Evo jednostavne naredbe koja čini da čitač prikaže poruku „Thank you.“

```
alert("Thank you.");
```

Tačka i zarez na kraju naredbe kazuju JavaScriptu da je komanda završena, isto kao što tačka označava kraj rečenice. Prema JavaScript standardu, znak za prelom reda takođe će označiti kraj komande, ali se završavanje svake naredbe tačkom i zarezom smatra dobrom praksom.

Komentari

JavaScript omogućava da ostavljate komentare koji će se zanemarivati u vreme izvršavanja skripta, tako da možete ubacivati podsetnike i objašnjenja kroz ceo kôd. To je posebno korisno ukoliko će vaš kôd u budućnosti menjati neki drugi programer.

Dva su načina korišćenja komentara. Za jednoredne komentare koristite dve kose crte (**//**) na početku reda. Jednoredni komentar možete pisati u istom redu kao naredbu, pod uslovom da je iza nje. Komentar ne morate zatvarati jer ga zatvara znak za prelom reda.

```
// Ovo je jednoredni komentar.
```

Za višeredne komentare koristi se ista sintaksa kao u CSS-u. Čitač ignoriše sve što se nalazi između znakova **/* */**. Možete ih koristiti da biste „isključili“ napomene, pa i delove skripta, kada ispitujete ima li grešaka.

```
/* Ovo je višeredni komentar.
Sve što se nalazi između ovih znakova potpuno
će se zanemariti pri izvršavanju skripta.
Ovaj oblik komentara se mora zatvoriti. */
```

Koristiću jednoredne komentare da bih dodao kratka objašnjenja u primere koda, a ranije opisanu funkciju **alert()** (slika 19-5) upotrebiću da bismo na brzinu pogledali rezultate našeg rada.

Promenljive

Promenljivu zamislite kao kontejner za podatke.

Ako ste slični meni, i sâm termin „promenljive“ pokreće neprijatna sećanja na srednjoškolske časove matematike. Suština je uglavnom ista, mada sada nema vašeg smešno začešljanog profesora.

Promenljiva je kontejner za podatke. Date joj ime a zatim dodelite vrednost, koja može biti broj, znakovni niz, element DOM-a ili funkcija – u stvari, bilo šta. Tako dobijamo zgodan način da je kasnije referenciramo po imenu. Sama vrednost se može menjati i ponovo dodeljivati kako god nalaže logika skripta.

Sledećom deklaracijom pravi se promenljiva po imenu „foo“ i dodeljuje joj se vrednost 5:

```
var foo = 5;
```

Počinjemo tako što deklarišemo promenljivu pomoću rezervisane reči **var**. Jedan znak jednakosti (=) ukazuje na to da joj dodeljujemo vrednost. Pošto je to kraj naše naredbe, red završavamo tačkom i zarezom. Promenljive se mogu deklarisati i bez rezervisane reči **var**, što utiče na to koji će deo skripta imati pristup njihovom sadržaju. To ćemo detaljnije razmotriti u odeljku „Oblast važenja promenljivih i rezervisana reč var“, kasnije u ovom poglavlju.

Kao ime promenljive možete koristiti bilo šta, ali obavezno izaberite neko koje će vam i kasnije imati smisla. Svakako ne biste želeli da promenljivoj date ime „podaci“; ono bi trebalo da opisuje podatke koje promenljiva sadrži. U našem