POGLAVLJE 1

JavaScript je više nego što biste pomislili

Po završetku ovog poglavlja, moći ćete da

- Razumete istoriju Javascripta,
- Prepoznate delove programa JavaScript,
- Koristite *javascript* pseudo-protokol,
- Razumete gde se JavaScript smešta unutar veb strane,
- Razumete šta JavaScript može a šta ne može,
- Razumete kako se JavaScript koristi u Windowsu 8.

Kratka istorija JavaScripta

JavaScript nije Java. Toliko! Kada smo ovo razjasnili, možemo da pređemo na značajnije i važnije učenje, kao npr. kako da napravite dobre slajdere. Šalu na stranu, JavaScript je jedna implementacija specifikacije poznate kao ECMAScript. Detaljnije o ECMAScriptu ćete učiti kasnije u ovom poglavlju.

Kako je nastao JavaScrip? Možda ne znate bogatu i detaljno ispričanu istoriju JavaScripta – a možda vas i ne zanima. Ukoliko je to slučaj, možda ćete pokušati da pređete na naredno poglavlje i da počnete sa pisanjem koda u JavaScriptu. Ovo bi, naravno, bila greška – preskočili biste sve divne informacije koje slede u ovom poglavlju. A razumevanje istorije JavaScripta je važno za razumevanje načina na koji se jezik danas primenjuje u različitim okruženjima.

Jezik JavaScript je prvobitno razvio Brendan Ajh u kompaniji Netscape između 1995-1996. godine. U to vreme se jezik nazivao LiveScript. Bilo je to sjajno ime za novi jezik, i priča je mogla tu da se završi. Međutim, nesrećnom odlukom ljudi iz marketinga su imali svoje razloge i jezik su preimenovali u JavaScript. Uskoro je nastala konfuzija. Java je u to vreme bio uzbudljiv novi jezik, a neko je odlučio da zaradi na popularnosti jezika Java korišćenjem njegovog imena. Kao posledica toga, nastalo je povezivanje jezika JavaScript sa jezikom Java. Ovo je bio nedostatak JavaScripta, jer iako popularna u smislu česte upotrebe, Java je bila nepopularna zbog dosta loše reputacije – programeri su koristili jezik Java na veb lokacijama za predstavljanje podataka

ili dodavanje beskorisnih poboljšanja (poput dosadnog pomerajućeg teksta). Korišćenje Java programa bilo je otežano time što je bilo potrebno učitavanje dodatnih programa u veb pretraživač, pri čemu se usporavao proces pretrage, što je iritiralo posetioce i dovodilo do problema pristupačnosti. Tek nedavno počelo je odvajanje jezika JavaScript od nedostataka programa Java. Međutim, I dalje, bar jednom nedeljno, čujem ljude da mešaju Javu i JavaScript. Nadam se da vi to više nećete raditi!

JavaScript nije kompajlirani jezik, čime izgleda kao da mu nedostaje snaga. Međutim, programeri kojima je JavaScript novost uskoro su postali svesni njegove snage i korisnosti prilikom simulacije i kreiranja interaktvnosti u globalnoj računarskoj mreži. Dok nisu došli do realizacije ovoga, progameri su razvili brojne veb lokacije pomoću jednostavnog HTML (Hypertext Markup Language) jezika i grafike, kojoj često nedostaje vizualna privlačnost i sposobnost interakcije sa sadržajem lokacije. Pomoću Windowsa 8, JavaScript sada ima velike mogućnosti za kreiranje bogatih aplikacija koje se ne oslanjaju na pretraživač veba.

Prvobitno se jezik JavaScript fokusirao na proveru ispravnosti obrazaca (formi) na klijentskoj strani i rad sa slikama na veb stranama koje su obezbeđivale osnovnu, ali korisnu interaktivnost i odziv za posetioca. Kada bi posetilac veb lokacije ispunio formu, JavaScript bi odmah proveravao ispravnost sadržaja veb obrasca umesto da pravi kružni put do servera. Sprečavanje povratnog puta do servera je posebno bilo od pomoći pre nego što su širokopojasne mreže postale sveprisutne, jer je činilo da aplikacije izgledaju malo brže i da daju bolji odziv – a i dalje je tako.

Pojava programa Internet Explorer 3.0

Pojavom programa Internet Explorer 3.0, 1996. godine, Microsoft je uključio podršku za osnovu JavaScripta poznatu u Internet Exploreru kao Jscript, kao i podršku za još jedan programski jezik nazvan Microsoft Visual Basic, Scripting Edition, ili VBScript. Iako su JavaScript i Jscript bili slični, njihova primena nije bila ista. Iz tog razloga su razvijene metode pomoću kojih se otkriva koji pretraživač koristi posetilac veb lokacije, kako bi se odgovorilo odgovarajućim skriptom. Ovaj proces je poznat pod nazivom detekcija (otkrivanje) pretraživača i o njemu se govori u poglavlju 11 „Uvod u u biblioteku jQuery". Iako ovaj proces nije poželjan u većini aplikacija, videćete da se on i dalje koristi, posebno pojavom mobilnih uređaja koji imaju sopstveni izgled i osećaj.

A onda je došao ECMAScript

Kompanije Microsoft i Netscape su sredinom 1997. godine radile sa Evropskom asocijacijom proizvođača računara (ECMA – European Computer Manufacturers Association) na lansiranju prve verzije jezičke specifikacije poznate kao ECMAScript, formalnije znana kao ECMA-262. Od tada, svi pretraživači Microsofta su implementirali verzije standarda ECMAScript. Ostali popularni pretraživači, kao što su Firefox, Safari i Opera, takođe su implementirali ECMAScript standard.

Treće izdanje standarda ECMA-262 pojavilo se 1999. godine. Dobro je to što pretraživači kao što su Microsoft Internet Explorer 5.5 i Netscape 6 podržavaju treće izdanje standarda, kao i da od tada svaki važniji pretraživač podržava verziju JavaScripta formalizovanu u trećem izdanju

standarda ECMA-262. Loše je to da svaki pretraživač primenjuje ovaj standard na malo drugačiji način, tako da nekompatibilnost i dalje zadaje nevolje programerima koji koriste JavaScript.

Najnovija verzija ECMAScripta, koja je formalizovana u standardu nazvanom ECMA-262, pojavila se krajem 2009. godine i poznata je pod nazivom ECMA-262 peto izdanje. Iz mnogorazloga preskočena je verzija 4 specifikacije, kao i da bi se izbegla konfuzija između brojnih konkurentih predloga standarda. ECMA-262 izdanje 5.1 je postalo opšte prisutno u vreme pisanja ove knjige i verovatno će se, dok budete čitali ovu knjigu (nadam se), nalaziti u verzijama popularnih pretraživača kao što je Explorer, Chrome, Firefox, Opera i Safari.

Važno je da zapamtite da kao programer koji ugrađuje JavaScript u veb aplikacije, morate da uzmete u obzir razlike između verzija specifikacije ECMA-262, i brojnih implementacija JavaScripta. Uzimanjem u obzir ovih razlika možda će značiti implementaciju programskog jezika na malo drugačiji način, kao i testiranje, testiranje i ponovo testiranje u različitim pretraživačima i na različitim platformama. Danas na internetu korisnici imaju nizak prag tolerancije na loše dizajnirane aplikacije koje rade u samo jednom pretraživaču.

Iz dva osnovna razloga je uzimanje u obzir ovih razlika postalo je tokom poslednjih godina mnogo jednostavnije. Prvi razlog je da su se veb pretraživači konsolidovali oko specifikacija za HTML, CSS i JavaScrip, pa su prodavci radili na tome da interpretacije ovih specifikacija približe jedna drugoj. Drugi razlog jeste to što su JavaScrip biblioteke postale popularnije. Kroz ovu knjigu pokazaću kako da korišćenjem biblioteke jQuery učinite JavaScript jednostavnijim.

Važno Od presudne je važnosti da svoje veb lokacije testirate u više pretraživača – uključujući veb aplikacije koje ne biste koristili u nekom drugom pretraživaču osim u Internet Exploreru. Čak i kada ste sigurni da će se vaša aplikacija koristiti samo u Internet Exploreru ili da ako samo njega zvanično podržavate, i dalje morate da testirate aplikaciju u drugim pretraživačima. Ovo je bitno zbog bezbednosti, a time pokazujete da ste temeljan programer koji razume današnje internet tehnologije.

Toliko standarda...

Ako mislite da standardi JavaScript programiranja nisu strogo definisani, onda ste u pravu. Svaki pretraživač podržava JavaScript na malo drugačiji način, čime čini vaš i moj posao još težim. Pokušaj da opišete sve ove varijacije je još veći izazov od pisanja o jeziku koji implementira jedan određeni entitet, kao što je određena verzija Microsoft Visual Basica ili Perl. Vaš (i moj) posao je da pratite ove razlike i da ih po potrebi uzimate u obzir, i pokušate da između njih, što je više moguće, pronađete zajedničko.

DOM standard

Još jedan standard koji se razvija i važan je za programera JavaScripta je standard objektnog modela dokumenta (DOM – Document Object Model) razvijen od strane konzorcijuma W3C (World Wide Web Consortium). Konzorcijum W3C definiše standard DOM kao „interfejs

pratformski i jezički neutralan koji dozvoljava programima i skriptovima da dinamički pristupaju i ažuriraju sadržaj, strukturu i stil dokumenata." To znači da možete da radite sa specifikacijom koje se veb pretraživači drže prilikom razvijanja veb strane na dinamički način. DOM kreira strukturu stabla objekata za HTML i dokumenta proširivog jezika za označavanje (XML – Extensible Markup Language) i omogućava pisanje skriptova za te objekte. JavaScript je u mnogim bitnim funkcijama u tesnoj interakciji sa DOM-om. Poput JavaScripta i DOM standard se različito interperetira za svaki pretraživač, čineći život JavaScript programera interesantnijim. Internet Explorer 4.0 i prethodne verzije Netscapea uključivale su podršku za prethodni DOM standard, poznat kao Level 0. Ako koristite Level 0 DOM standard, budite sigurni da ćete pronaći podršku za DOM standard u tim pretraživačima, kao i svim pretraživačima koji su se pojavili kasnije.

Microsoft Internet Explorer 5.0 i Internet Explorer 5.5 uključivali su delimičnu podršku za Level 1 DOM standard, pri čemu su Windows Explorer 6.0 i kasnije verzije uključivale podršku za Level 2 DOM standard. Kasnije verzije Internet Explorera, Chromea, Firefoxa, Safarija i Opere podržavaju Level 3 DOM standard u nekoj formi. Safari pruža reprezentaciju WebKit mehanizma za renderovanje. WebKit mehanizam za renderovanje se takođe koristi kao osnova za pretraživač na uređajima kao što su iPhone i iPad, kao i na Android uređajima.

Lekcija o JavaScript standardima i povezanim DOM standardima, koju bi trebalo da što pre usvojite, jeste da obratite posebnu pažnju na kôd koji pišete (tu nema iznenađenja) i sintaksu koja se koristi prilikom implementacije tog koda. Ako to ne učinite, JavaScript može potpuno zakazati i sprečitida se vaša strana vizuelno prikaže u datom pretraživaču. Poglavlje 12 „Objektni model dokumenta" detaljnije se bavi DOM standardom.

Savet W3C ima aplikaciju koja može da testira module sa specifikacijom za razneDOM nivoe koje vaš veb pretraživač tvrdi da podržava. Ova aplikacija može da se nađe na *http://www.w3.org./2003/02/06-dom-support.html*.

Šta se nalazi u JavaScript programu?

JavaScript program se sastoji iz iskaza i izraza koji su formirani od tokena različitih kategorija, uključujući ključne reči, literale, separatore, operatore i identifikatore zajedno postavljene na način koji ima smisla za JavaScript interpreter, koji se nalazi u većini veb pretraživača. Ova rečenica zvuči komplikovano, ali nije komplikovana za one koji su programirali u bilo kom drugom jeziku. Izraz bi mogao biti:

```
varsmallNumber=4;
```

U ovom izrazu, iza tokena (rezervisane reči) –*var*-slede drugi tokeni, kao što je identifikator (*smallNumber*), operator (=) i literal (*4*). (Detaljnije o ovim elementima učićete u nastavku knjige.) Svrha ovog izvora je da podesi promenljivu pod nazivom *smallNumber* da bude jednaka celom broju *4*.

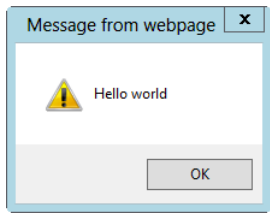
Kao u bilo kom programskom jeziku, iskazi se postavljaju zajedno, po redosledu koji čini da program izvršava više od jedne fukcije. JavaScript definiše funkcije na sopstveni način, o kome ćete više čitati u poglavlju 7 „Rad sa funkcijama". JavaScript definiše nekoliko ugrađenih funkcija koje možete da koristite u vašim programima.

Korišćenje *javascript* pseudoprotokola i funkcije

1. Otvorite veb pretraživač.
2. U traci sa adresom upišite sledeći kôd i pritisnite Enter:

   ```
   javascript:alert(„HelloWorld");
   ```

   Pošto pritisnete taster Enter, videćete okvir za dijalog sličan ovome:



Čestitamo! Upravo ste programirali svoj prvi (doduše ne mnogo koristan) komad JavaScript koda. Međutim, u ovom malom komadu koda, nalaze se dve važne stavke koje ćete verovatno koristiti prilikom programiranja u JavaScriptu: identifikator *javascript* pseudo-protokola u pretraživaču i još važnije, funkciju *alert*. Ove stavke ćete detaljnije ispitati u kasnijim poglavljima; za sada je dovoljno da ste naučili nešto o nečemu što ćete koristiti u budućnosti!

Napomena Internet Explorer 10 u Windowsu 8 nekada ne prikazuje ili ne koristi *javascript* pseudo protokol na pravi način. U tom slučaju, pokušajte sa drugim pretraživačima, kao što su Firefox ili Chrome.

JavaScript je takođe *vođen događajima,* što znači da može da odgovori na određene događaje ili „stvari koje se dešavaju", kao što je kliktanje mišem ili izmena teksta unutar polja obrasca. Povezivanje JavaScripta sa nekim događajem je ključno za većinu situacija u kojima se koristi ovaj jezik. U poglavlju 11 videćete kako da odgovorite na događaje pomoću JavaScripta.

Postavka JavaScripta na vašoj veb strani

Ako vam je HTML novina, sve što je potrebno da znate za sada je da HTML obeležava (markira) elemente na veb strani pomoću para odgovarajućih oznaka (tagova) oivičenih zagradama. Završna oznaka počinje kosom crtom (/). Elementi mogu da se ugnezde jedan u drugom. JavaScript se smešta unutar oznaka <*SCRIPT*> koje se nalaze u <*HEAD*></*HEAD*> i/ili <*BODY*></*BODY*>oznakama veb strane, kao u sledećem primeru:

```
<!doctypehtml>
<html>
<head>
<title>AWebPageTitle</title>
<scripttype=”text/javascript”>
//JavaScriptide ovde
</script>
</head>
<body>
<scripttype=”text/javascript”>
//JavaScriptmože i ovde da ide
</script>
</body>
</html>
```

JavaScript postavljen unutar oznaka <*BODY*> izvršava se kao da se susreo sa pretraživačem, što je od koristi kada treba da napišete dokument pomoću JavaScript funkcije kao što sledi (pozivi funkcija su prikazani podebljano):

```
<!doctypehtml>
<html>
<head>
<title>AWebPageTitle</title>
<scripttype=”text/javascript”>
//JavaScriptide ovde
</script>
</head>
<body>
<scripttype=”text/javascript”>
document.write(„hello”);
document.write(„world”);
</script>
</body>
</html>
```

Zbog načina na koji pretraživači učitavaju JavaScript, trenutno najbolja praksa za postavljanje JavaScripta u vaš HTML jeste pozicioniranje <*SCRIPT*> oznaka na kraj <*BODY*> elementa umesto u <*HEAD*> element. Ukoliko pretraživač blokira unošenje podataka dok se datoteke JavaScripta učitavaju, ovim se obezbeđuje da se sadržaj strane prikaže

Kada koristite JavaScript na Extensible Hypertext Markup Language (XHTML) strani, znak manje (<) i ampersend znak (&) se tumače kao XML, što može dovesti do problema za JavaScript. Da biste izbegli ovo, koristite sledeću sintaksu na XHTML strani:

```
<scripttype=”text/javascript”>
<![CDATA[
    //JavaScriptide ovde
]]>
</script>
```

Pretraživači koji ne odgovaraju XHTML-u ne tumače ispravno CDATA odeljak. Ovaj problem možete izbeći postavljanjem CDATA odeljka unutar JavaScript komentara – red ili skup redova ispred koji su dve kose crte (//), na način prikazan ovde:

```
<scripttype=”text/javascript”>
```

```
//<![CDATA[
    //JavaScriptide ovde
//]]>
</script>
```

Tačno je, kôd je stvarno ružan. Međutim, to se lako može srediti: upotrebite spoljne JavaScript datoteke. U poglavlju 2, „Razvoj u JavaScript-u", učićete kako tačno da postignete ovaj jednostavan zadatak.

Tipovi dokumenata

Ako ste neko vreme proveli programirajući za veb, sigurno ste ste upoznati sa dekleracijama tipa dokumenta ili DOCTYPE dekleracijama, kako su se nekad zvale. Jedan od najvažnijih zadataka koje možete da uradite prilikom dizajniranja veb strana jeste da uključite precizan i sintaksno ispravan odeljak DOCTYPE deklaracije na vrh strane. DOCTYPE deklaracija, često skraćeno kao DTD, dozvoljava pretraživaču (ili drugom programu za analiziranje sintakse) da zna pravila koja treba slediti prilikom analize sintakse elemenata dokumenta.

Jedan primer DOCTYPE deklaracije za HTML 4.01 izgleda ovako:

```
<!DOCTYPEhtmlPUBLIC"-//W3C//DTDHTML4.01//EN"
„http://www.w3.org/TR/html4/strict.dtd">
```

Ako za kreiranje veb projekta koristite verziju Microsoft Visual Studija pre one iz 2012. godine, svakoj stranije automatski dodeljena DOCTYPE deklaracija za XHTML 1.0 standard, kao ovde:

```
<!DOCTYPEhtmlPUBLIC"-//W3C//DTDXHTML1.0Transitional//EN""http://www.w3.org/TR
/xhtml1/DTD/xhtml1-transitional.dtd">
```

HTML verzija 5 koristi mnogo jednostavniju deklaraciju DOCTYPE:

```
<!DOCTYPEhtml>
```

Ukoliko ne uspete da deklarišete DOCTYPE, pretraživač će interpretirati stranu korišćenjem režima rada poznatog kao *Quirks Mode*. Prelazak na režim rada Quirks Mode znači da će dokument možda prikazivati drugačije od onoga kako vi želite, posebno kada se gleda u višepretraživača.

Ako deklarišete DOCTYPE, važna je provera da li dobijeni HTML, kaskadna lista stilova (takođe poznata kao CSS) i JavaScript takođe prate veb standarde, kako bi dokument mogao da se prikazuje najširoj mogućoj publici, bez obzira na to koji se interfejs ili pretraživač koristi. W3C stavlja na raspolaganje onlajn validator koji možete koristiti za proveru valjanosti bilo koje javno dostupne veb strane na adresi *http://validator.w3.org/*.

Savet Koristite redovno Markup Validator, ve dok vam ne postane blisko programiranje u skladu sa standardima, i uvek proveravajte ispravnost pre puštanja u javnost vašeg veb projekta.

Šta JavaScript može?

JavaScript je uglavnom komplementaran jezik, što znači da nije neobičajeno da se cela aplikacija napiše samo u JavaScriptu, bez pomoći drugih jezika kao što je HTML i bez prezentacije u veb pretraživaču. Neki Adobe proizvodi podržavaju JavaScript, a Windows 8 počinje da menja ovo, međutim, osnovna upotreba JavaScripta je u pretraživaču.

JavaScript predstavljena je slovom J u akronimu AJAX (asinhroni JavaScript i XML), ljubimac veb 2.0 fenomena. Međutim, izvan toga, JavaScript je svakodnevni jezik koji obezbeđuje očekivanu interaktivnost, možda čak i zahtevanu od strane današnjih posetilaca veba.

JavaScript može da izvršava brojne zadatke na klijentskoj strani aplikacije. Na primer, može da dodaje potrebnu interaktivnost na veb lokaciji kreiranjem padajućih menija, transformisanjem teksta na strani, dodavanjem dinamičkih elemenata na stranu i pružanjem pomoći pri unosu obrasca.

Pre nego što naučite šta sve JavaScript može – što je fokus ove knjige – potrebno je da razumete i šta JavaScript ne može da radi, ali imajte na umu da nijedna od ovih diskusija nije sveobuhvatna.

Šta JavaScript ne može?

Većina operacija koje JavaScript ne može da izvrši potiče od toga da je upotreba JavaScripta donekle ograničena na okruženje veb pretraživača. Ovaj odeljak ispituje neke zadatke koje JavaScript ne može da izvrši i neke koje JavaScript ne bi trebalo da izvrši.

JavaScript ne može biti nametnut klijentu

JavaScript se zbog svoje funkcionalnosti zasniva na drugačijem interfejsu ili matičnom programu. Ovaj matični program je obično veb pretraživač klijenta, takođe poznat kao *korisnički agent*. S obirom da je JavaScript jezik klijentske strane, može da radi samo ono što mu klijent dozvoli.

Neki ljudi i dalje koriste starije verzije pretraživača koje uopšte ne podržavaju JavaScript. Neki neće biti u mogućnosti da iskoriste prednosti brojnih modernih karakteristika JavaScripta zbog pomoćnih programa, čitača teksta i drugih dopunskih softvera koji pomažu iskustvo u pretraživanju, dok će neki odabrati da isključe JavaScript zato što to mogu, zato što brinu za bezbednost (bilo da je ona zamišljena ili stvarna), ili zbog loše reputacije JavaScripta koja potiče od određenih dosadnih stvari kao što su iskačući oglasi.

Bez obzira na razloge, potrebno je da izvršite dodatan posao kako biste obezbedili da veb lokacija koju dizajnirate bude raspoloživa onim pojedincima koji nemaju JavaScript. Već čujem negodovanje: „Ali ova karatkeristika je stvarno [umetnite sopstveni superlativ ovde: sjajna, divna, neophodna, lepa, fantastična]."

Bez obzira na to koliko ova karakteristika može biti dobra, imate šansu da izvučete korist od bolje međuoperativnosti i više posetilaca lokacije. U nastavku odeljka „Saveti za upotrebu JavaScripta"

ponudio sam smernice koje možete da pratite kako biste pravilno koristili JavaScript na vašoj veb lokaciji.

Možda će vam pomoći razmišljanje na drugačiji način. Prilikom izgradnje veb aplikacije koja potiče iz Microsoft Internet Information Services (IIS) 6.0, možete da pretpostavite da će aplikacija obično raditi kada se izvršava na iz IIS6.0 servera bilo gde. Slično, kada gradite aplikaciju za Apache 2, budite sigurni da će raditi na drugim Apache 2 instalacijama. Međutim, ista pretpostavka ne važi za JavaScript. Kada pišete aplikaciju koja dobro radi na računaru, ne možete da garantujete da će raditi na nečijoj drugoj. Ne možete da kontrolišete kako će aplikacija raditi nakon što se pošalje klijentu.

JavaScript ne može da garantuje sigurnost podataka

S obzirom da se JavaScript u potpunosti izvršava na klijentu, programer mora da nauči da prepusti kontrolu. Kao što možete očekivati, preprepuštanje kontrole programu ima ozbiljne posledice. Nakon što je program na računaru klijenta, klijent može da uradi brojne neželjene stvari na podacima pre nego što ih pošalje nazad serveru. Kao u slučaju bilo kog drugog veb programiranja, nikada ne treba da verujete podacima koji dolaze od klijenta. Čak i kada ste upotrebili funkcije JavaScripta za proveru ispravnosti sadržaja na obrascima, i dalje morate, kada ponovo dođe do servera, da potvrdite ovaj input. Klijent sa isključenim Javascriptom može da pošalje nazad đubre podataka preko veb obrazaca. Ukoliko naivno verujete da je funkcija klijentske strane JavaScripta već proverila podatke kako bi se utvrdila njihova valjanost, možda ćete otkriti da se neispravni podaci vraćaju serveru uzrokujući neočekivane i potencijalno opasne posledice.

VAŽNO Setite se da na računaru postioca JavaScript može biti onemogućen. Ne možete se oslanjati na to da će simpatični trikovi biti uspešni, kao što je upotreba JavaScripta za onemogućavanje pritiska na desni taster miša ili sprečavanje posetilaca da pregledaju izvor strane,a i ne bi trebalo da ih koristite kao sigurnosne mere.

JavaScript ne može da ukršta domene

JavaScript programer treba da bude svestan *politike istog porekla (Same-Origin Policy)*, koja nalaže da skriptovi koji se izvršavaju unutar jednog domena ne mogu da imaju pristup izvorima iz nekog drugog internet domena, niti mogu da utiču na skriptove i podatke iz drugog domena. Na primer, JavaScript može da se koristi za otvaranje novog prozora pretraživača, ali sadržaji tog prozora su nekako ograničeni na pozvani skript. Kada strana sa moje veb lokacije (*braingia.org*) sadrži JavaScript, ta strana ne može da pristupi bilo kom JavaScriptu koji se izvršava iz drugog domena, kao što je *microsoft.com*. Ovo je suština politike istog porekla: JavaScript treba da se izvršava ili da potiče sa iste lokacije.

Politika istog porekla je često ograničenje sa kojim se suočavamo u kontekstu okvira i AJAX-ovog objekta *XMLHttpRequest*, gde više JavaScript zahteva može da se pošalje različitim veb serverima. Predstavljanjem program Windows Internet Explorer 8, Microsoft je predstavio podršku za objekat *XdomainRequest*, koja dozvoljava ograničeni pristup podacima sa drugih domena.

JavaScript se ne bavi serverima

Prilikom razvijanja koda serverske strane kao što je Visual Basic.NET ili PHP (rekurzivni akronim koji označava *PHP: Hypertext Preprocessor*), možete biti prilično sigurni da će server implementirati određene funkcije, kao što su razgovor sa bazom podataka ili odobravanje pristupa modulima neophodnim za veb aplikaciju. JavaScript nema pristup promenljivama serverske strane. Na primer, JavaScript ne može da pristupi bazama podataka koje se nalaze na serveru. Kôd JavaScripta je ograničen na ono što može da se uradi unutar platforme na kojoj se skript izvršava, što je obično pretraživač.

Još jedna promena u načinu vašeg razmišlja koju treba da uradite, ukoliko ste upoznati sa programiranjem serverske strane, jeste ta da sa JavaScriptom morate da testirate kôd na mnogo razlilčitih klijenata kako biste znali na šta je određeni klijent sposoban. Ukoliko, prilikom programiranja serverske strane, server ne implementira zadatu funkciju, vi to odmah znate, jer je skript serverske strane otkazao prilikom testiranja. Ako zanemarimo neposlušne administratore, implementacija koda krajnjeg servera ne može olako da se menja, pa ćete stoga lakše znati šta možete, a šta ne možete da kodirate. Međutim, ne možete da predvidite JavaScript kôd koji treba da se izvrši na klijentima, s obzirom da su ovi klijenti totalno izvan vaše kontrole.

Napomena Postoji implementacija JavaScripta za serversku stranu, ali je ona van okvira ove knjige.

Saveti za korišćenje JavaScripta

Više faktora čini dobar veb dizajn, i stvarno, ko je taj ko određuje da li je nešto dobro ili nije? Za jednog posetioca će lokacija možda biti ružna mešavina boja i teksta,kreiranog kao da su ovi elementi ubačeni u džak, izmešani i razbacani po stranici; sledećem posetiocu se može svideti dizajn i šema boja.

Pošto čitate ovu knjigu, pretpostavljam da vam je potrebna pomoć za korišćenje JavaScripta kako biste poboljšali svoju veb lokaciju. Takođe, pretpostavljam da želite da koristite ovaj programski jezik kako biste pomogli ljudima u korišćenju vaše lokacije i kako biste učinili da lokacija bolje izgleda, da je posetioci bolje dožive i da radi bolje.

Dizajn veb lokacije nije i nikada neće biti potpuno objektivan proces. Cilj jedne veb lokacije može biti informacioni, što nalaže i određeni pristup u dizajnu, dok cilj neke druge veb lokacije može biti povezivanje sa aplikacijom, pa se stoga zahteva specijalan dizajn i funkcionalnost. Prema tome, mnoge popularne i naizgled dobro dizajnirane lokacije imaju zajedničke određene aspekte. Ovde ću pokušati da razložim ove aspekte, ali ću vas zamoliti da zapamtite da nisam kreirao sveobuhvatnu listu i da stavke reflektuju samo lične stavove.

Dobro dizajnirana veb lokacija čini sledeće:

- **Stavlja funkciju iznad forme** Kada korisnik poseti veb lokaciju, on obično želi da dođe do određenih informacija ili da izvrši zadatak. Što je vaša lokacija komplikovanija za pretrživanje, to će korisnik pre otići na neku drugu lokaciju koja se lakše pretražuje.

  Animacije i trepćući delovi dolaze i odlaze, ali ono što ostaje su lokacije koje sadrže osnovne informacije prezentovane na profesionalan, jednostavan i pristupačan način. Korišćenje najnovijih modernih softvera za animaciju ili veb tehnologije podsećaju me na vreme HTML <*BLINK*>oznake. Za obe koji je nikad nisu videli na delu, <*BLINK*> oznaka-uzrokuje da tekst koji se nalazi unutar nje nestane i ponovo se pojavi na ekranu. Izgleda da skoro svi veb programeri mrze <*BLINK*> oznaku i ono što ona radi na veb strani. Istim ovim programerima bi bilo pametno da imaju na umu da će današnja uzbudljiva svojstva ili specijalni efekti veb strane biti <*BLINK*> oznaka sutrašnjice. Uspešne veb lokacije se drže osnova i koriste ove vrste elemenata samo kada sadržaj od njih to zahteva.

  Koristite elemente kao što su mapa lokacije, alt oznake i jednostavne alatke za navigaciju i ne zahtevajte specijalan softver ili dodatne programe za prikazivanje glavnog sadržaja lokacije. Suviše često sam posećivao veb lokacije i bivao zaustavljen, jer mi je bio potreban dodatni program ili najnovija verzija ovog ili onog plejera (koji nemam) da bih pretražio lokaciju.

  Iako mape lokacije, alt oznake i jednostavna navigacija deluju možda zastarelo, to su nezamenljive stavke za pristupačnost. Čitači teksta i druge slične tehnologije koje omogućavaju glasno čitanje lokacije ili koje pretražuju pojedinci sa posebnim potrebama koriste ove pomoćne karakteristike i često imaju probleme sa složenim JavaScriptom.

- **Prate standarde** Veb standardi postoje da bi se pratili, tako da ako ih ignorišete, činite to na sopstveni rizik. Korišćenje ispravne DOCTYPE deklaracije i dobro formiranog HTML-a pomaže da se vaša lokacija ispravno prikazuje posetiocima. Provera ispravnosti koja koristi alatku konzorcijuma W3C Markup Validator se toplo preporučuje. Ako je vaša lokacija neispravna, popravite je!

- **Prikazuje se ispravno u više pretražvača** Iako je udeo Internet Explorera na tržištu 90 procenata, nikada nije dobro da programeri ignorišu druge pretraživače. To bi značilo i da se pristupačnost ignoriše, pa ljudi sa čitačima teksta ili drugim dodacima ne mogu da koriste lokaciju. Oni koji koriste druge operativne sisteme, a ne Microsoft Windows, takođe neće moći da posećuju ove lokacije.

  Iako je Internet Explorer i dalje lider među pretraživačima koje koriste veb posetioci, velika je verovatnoća da će bar 30 do 40 procenata posetilaca koristiti druge pretraživače. Što su posetioci tehnički potkovaniji, to morate da obratite više pažnje prilagođavanju sajta za pregled sa drugim pretraživačima. Prema tome, ako se vaša lokacija obraća tehnički potokovanoj publici, vaša lokacija bi trebalo da radi u Firefoxu, Safariju ili čak i Lynxu.

  Nikada nećete želeti da zbog izbora pretraživača odbijete posetioce. Zamislite prodavca u radnji koji od deset potencijalnih mušterija odbije troje zbog njihovih cipela. Takva radnja ne bi dugo opstala u poslu – ili u krajnjem slučaju, ne bi bila uspešna..

Ako težite da sledite veb standarde, verovatno već radite većinu onoga što je potrebno da omogućite podršku za više pretraživača. Izbegavanje upotrebe vlasničkih programskih dodataka na vašoj veb lokaciji je još jedan način da obezbedite da se vaša lokacija ispravno prikazuje. Potrebno je samo da pogledate iPad i vidite uređaj koji je popularan, ali čija je upotreba ograničena jer izvorno ne podržava Flash. Iz tog razloga, kreiranje lokacija koje poštuju standarde i izbegavaju vlasničke dodatne programe, obezbeđuje da vaša lokacija bude vidljiva najširoj mogućoj publici.

- **Koriste adekvatne tehnologije u pravo vreme** Kada govorimo o dodatnim programima, dobro dizajnirana veb lokacija ne koristi tehnologiju ni previše ni premalo. Na video lokaciji prikladno je reprodukovanje video zapisa. Slično tome, na lokaciji sa muzikom reprodukovanje muzičke pozadine je prikladno. Na drugim lokacijama ova svojstva možda nisu prikladna. Ako mislite da je vašoj lokaciji potrebna muzička pozadina, vratite se na početak i ispitajte zašto želite veb lokaciju! I dalje se naježim kada se setim veb lokacije jednog advokata koju sam jednom posetio. Bez da sam išta uradio, lokacija je počela da reprodukuje u pozadini džingl kompanije. Prijatelji ne dozvoljavaju prijateljima da koriste muzičku pozadinu na njihovim lokacijama, osim ako ti prijatelji nisu iz benda Rush i ako ste radili na njihovoj veb lokaciji.

Gde se uklapa JavaScript?

Današnji veb se još uvek razvija. Jedan od popularnijih pokreta je poznat kao *nenametljivo pisanje skriptova*. Paradigma nenametljivog pisanja skriptova sadrži dve komponente progresivno poboljšanje i bihejvioralno odvajanje. *Bihejvioralna odvajanje* poziva da se struktura odvojiod stila i da se oba ova elementa razdvoje od ponašanja. U ovom modelu HTML ili XHTML pružaju strukturu, dok CSS pruža stil a JavaScript ponašanje. Progresivno poboljšanje znači dodavanje još karakteristika na stranu prilikom testiranja mogućnosti pretraživača; poboljšanje iskustva korisnika kada je moguće, a da se pri tome ne očekuje da će JavaScript ili određena JavaScript funkcija uvek biti na raspolaganju. Na ovaj način je JavaScript nenametljiv; ne stoji na putu korisničkog iskustva. Ako JavaScript nije raspoloživ u pretraživaču, veb lokacija će i dalje raditi, jer posetilac može da koristi veb lokaciju na drugačiji način.

Kada se ispravno primeni, nenametljivo pisanje skripta znači da se ne pretpostavlja da će JavaScript biti raspoloživ i da će JavaScript otkazati na dostojanstven način. Dostojanstvena degradacija pomaže funkciji stranice bez JavaScripta ili koristi odgovarajuće pristupe da učini JavaScript raspoloživim kada to lokacija zahteva.

Zagovornik sam nenametljivog pisanja skripta, jer to znači da su se ispoštovali standardi i da je dobijena lokacija u skladu sa četiri pretpostavke koje smo naveli u prethodnom odeljku. Na žalost, ovo nije uvek slučaj. Možete da izdvojite HTML, CSS i JavaSrcipt i da opet preterate sa upotrebom vlasničkih oznaka, ali kada programirate na nenametljiv način, obratićete više pažnje na detalje i više ćete brinuti da krajnji rezultrat bude u skladu sa standardima.

Cilj mi je da u ovoj knjizi prikažem ne samo osnove JavaScripta, već i najbolji način za njegovu efektivnu upotrebu, što je moguće više nenametljivu.

Napomena o Jscriptu i JavaScriptu i ovoj knjizi

Ova knjiga obrađuje JavaScript definisan prema ECMA standardu u verzijama sve do poslednjeg izdanja 5. Ovo je razlika u odnosu na Microsoftovu implementaciju Jscripta, koji nije obrađen u ovoj knjizi. Za dodatne reference o Jscriptu, preporučujem da posetite sledeću lokaciju: Jscript (Windows Script Technologies) na adresi *http://msdn.microsoft.com/en-us/library/hbxc2t98.aspx*.

Koje pretraživače lokacija treba da podržava?

Kompatibilnost sa starijim verzijama je bilo pitanje za veb programere već duže vreme. Izbor koju verziju pretraživača podržavati svodi se na kompromis između upotrebe najnovije funkcionalnosti raspoložive u najnovijim pretraživačima i kompatibilne funkcionalnosti koju zahtevaju stariji pretraživači. Ne postoji teško i brzo pravilo o tome koje pretraživače bi trebalo da podržavate na vašoj veb lokaciji, tako da je odgovor: to zavisi.

Internet Explorer je nekada bio kralj među veb pretraživačima, bar po upotrebi, sigurno nije bio vrednovan tako od strane programera. Danas se, Internet Explorer, u različitim verzijama i dalje najviše koristi na lokacijama od opšteg interesa. Međutim, specijalizovane lokacije ili lokacije koje se odnose na tehnologiju, manje koriste Interne Explorer. Na njima se mnogo više koriste Firefox, Chrome, Safari, Opera i drugi pretraživači. Pretraživači poput Chromea i Firefoxa su takođe popularni čak i na lokacijama od opšteg interesa. Sve ovo znači da je potrebno razviti i testrati u različitim pretraživačima pre puštanja veb lokacije u rad.

Napomena Iako su slike ekrana u ovoj knjizi prikazane u Internet Exploreru, kôd za ovu knjigu je pisan i testiran prvo u Firefoxu, kao i u nekoliko drugih pretraživača. Slike ekrana su prikazane u Internet Exploreru isključivo iz pravnih razloga – tačnije, usled poteškoća u dobijanju dovoljno pravnih razjašnjenja za prikazivanje brojnih slika ekrana od drugih pretraživača.

Pribavljanje MSDN naloga od Microsofta omogućiće vam pristup do novih i starijih verzija proizvoda, uključujući i Internet Explorer. Dodatni izvori su Application Compatibility Virtual PC Images, besplatno raspoloživi od strane Microsofta. Ovim se omogućava upotreba vremenski ograničenih verzija Microsoft Windowsa koji sadrži starije verzije Internet Explorera. Za više informacija posetite *http://www.microsoft.com/downloads/details.aspx?FamilyId=21EABB90-958F-4B64-B5F1-73D0A413C8EF&displaylang=en*.

Mnogi veb dizajni i JavaScript funkcije ne zahtevaju novije verzije veb pretraživača. Međutim, kao što sam već objasnio, provera da li se vaša lokacija ispravno prikazuje u različitim pretraživačima je uvek dobra ideja. Potražite na adresi *http://browsers.evolt.org/* linkove kako biste arhivirali mnoge istorijske verzije veb pretraživača. Čak i kada izvršavate iscrpna testiranja više pretraživača, možete da dizajnirate lokaciju tako da otkaže na dostojanstven način. Potrebna vam je lokacija koja će se pravilno prikazivati nevezano od pretraživača koji se koristi.

A onda je došao Windows 8

Microsoft Windows 8 predstavlja paradigmu promene za JavaScript programere. U windowsu 8, Microsoft je podigao JavaScript na isti nivo na kome se nalaze i ostali jezici klijentske strane, kao što su Visual basic i C#, za razvijanje Windows 8 aplikacija. Pre pojave Windowsa 8, ako ste želeli da kreirate aplikaciju koja se izvršava na radnoj površini, bilo je potrebno koristiti Visual Basic, C# ili neki sličan jezik. Sa pojavom Windowsa 8, za kreiranje potpune aplikacije u stilu Windowsa 8, dovoljna je samo upotreba HTML-a i JavaScripta.

Windows 8 izlaže interfejs za programiranje aplikacija (API – Application Programming Interface), pružajući skup funkcija koje omogućavaju javscript programerima da izvorno pristupe oblastima izvan scene operativnog sistema. To znači da se programiranje za Windows 8 malo razlikuje od programiranja JavaScripta za veb pretraživače.

Naravno, vaše veb aplikacije će i dalje raditi u Internet Exploreru, koji se isporučuje sa Windowsom 8. Ove veb aplikacije su odvojene i drugačije od osnovnih Windows 8 aplikacija.

Ova knjiga će vam pokazati kako da razvijete Windows 8 koristeći JavaScript. Pre nego što dođete dotle, videćete kako možete da kreirate JavaScript programe koji se izvršavaju u veb pretražvačima.

Vežbanje

1. Tačno ili netačno: JavaScript je definisan od strane tela za standarde i podržan je na svim veb pretraživačima.
2. Tačno ili netačno: Kada posetilac na čijem je uređaju onemogućen JavaScript dođe na vašu veb lokaciju, trebalo bi da mu blokirate pristup lokaciji, jer ne postoji validan razlog za onemogućavanje JavaScripta.
3. Kreirajte blok definicije JavaScripta koji će se uobičajeno pojaviti na HTML stranici unutar bloka <*HEAD*> ili bloka <*BODY*>.
4. Tačno ili netačno: važno je da deklarišete koja verzija JavaScripta se koristi unutar DOCTYPE bloka definicije.
5. Tačno ili netačno: JavaScript može da se pojavi u <*HEAD*> bloku i unutar <*BODY*> teksta HTML stranice.