Šta je JavaScript i čemu služi?

U ovoj lekciji upoznaćemo se sa pojmom JavaScript tehnologije. Da bismo se upoznali sa programskim jezikom JavaScript na odgovarajući način, potrebno je da steknemo neka osnovna znanja o tome na koji način se naše internet stranice kreću kroz saobraćaj na internetu, kako se generišu i kako se prikazuju krajnjem korisniku.

Da bismo ovo razumeli, potrebno je prvo shvatiti pojmove **servera, klijenta i korisnika** u jednom procesu emitovanja internet stranice. U ovoj raspodeli, server je program koji će poslati stranicu klijentu putem interneta; klijent je program koji će tu stranicu prevesti u grafičku formu i prikazati korisniku, dok je korisnik, zapravo, osoba koja je pokrenula ovaj ceo proces, tako što je kliknula na link ili ukucala adresu u svom internet pretraživaču.

Kada se proces pokrene, na internet serveru se generiše strana. Parser (služi za dekodiranje, to jest za raščlanjivanje) čita oznake (Tags) na strani i, u odnosu na njih, rukuje sadržajem. Ukoliko naiđe na tag nekih od serverskih skripata (PHP, ASP, JSP), izvršava skripta koja se nalaze u tim tagovima i zatim, nakon izvršenog zadatka, prosleđuje stranu dalje, ka klijentu. HTML tagovi i tagovi klijentskih skripata se ne obrađuju na serveru, već se samo prosleđuju klijentu, nakon čega server više nema nikakav uticaj na stranu.

Kada klijent preuzme stranu, njegov zadatak je da raščlani dobijeni sadržaj (stranu) i da ga prezentuje korisniku. Ukoliko, prilikom tog raščlanjivanja, klijentska aplikacija naiđe na klijentska skripta (JavaScript), izvršava ih. To znači da je polje delovanja JavaScripta, od trenutka kada stranica dođe do klijenta do trenutka kada strana prestane da postoji, na klijentu (kada se pređe na drugu stranu ili se zatvori pretraživač...). Zbog toga, JavaScript može uraditi sve ono što serverska skripta ne mogu, jer su ona već izvršena u trenutku kada strana stigne do klijenta. Takođe, pruža i direktnu interakciju sa korisnikom. U praksi, JavaScript stoji iza animacija, padajućih menija, boksova sa porukama i ostalih dinamičkih grafičkih elemenata internet stranice.

Iako se često dešava permutacija Jave i JavaScripta, oni zapravo imaju veoma malo dodirnih tačaka. Moglo bi se reći da je ogromna sličnost u njihovom imenu analogna svim ostalim njihovim razlikama.

Slika 1.1. Razlika između programskih jezika Java i JavaScript

Kada je u pretraživač Netscape (najbolji pretraživač devedesetih godina prošlog veka) uvršćena i podrška za programski jezik Java, promenjeno je i dotadašnje ime klijentskog skriptnog jezika tog pretraživača LiveScript u JavaScript. Dakle, sličnost u imenima simbolizuje samo isti paket inovacija u pretraživaču Netscape, ali ne i sličnost u bilo kom aspektu jezika. Tada se, pod podrškom za Javu u pretraživaču, podrazumevala mogućnost aktiviranja Java Appleta, koji je već prevedena aplikacija, i koja, osim što je u inkapsuliranoj formi implementirana u HTML, nema nikakve veze sa njim (nešto poput Flash aplikacija).

Copyright © Link group

Sa druge strane, JavaScript je u direktnoj interakciji sa HTML-om i interpretira se u trenutku kada i HTML.

Konačno, kada je u pitanju imenovanje, JavaScript je zapravo sinonim za jezike koji poštuju jezički standard ECMAScript (deklarisan u instituciji za standardizaciju ECMA). Činjenica je da, iako funkcionišu isto, na različitim pretraživačima ovi jezici se mogu zvati različito (pa čak u nekim elementima i raditi različito). Na primer, na Microsoft Internet Exploreru ovaj jezik naziva se JScript. Različita funkcionalnost ovih dijalekata zadaje, ponekad, prilične muke developerima, jer je potrebno na različite načine obraditi neki blok koda, u zavisnosti od pretraživača.

Kada se JavaScript pojavio 1995. godine, njegova glavna svrha je bila da izvrši određenu validaciju ulaznih elemenata, koja je sve do tada bila prepuštena serverskoj strani. Netscape je video priliku da to promeni uvođenjem JavaScripta. Sposobnost da podrži izvršavanje nekih osnovnih mera validacije na klijentskoj strani predstavljala je veliki korak u napretku tehnologije u to vreme. Tada su internet konekcije bile znatno slabijih performansi, skoro neuporedive sa onima koje imamo danas. Slanje upita serverskoj strani za bilo koji klijentski unos uopšte nije bilo praktično zbog loših internet konekcija. Zbog toga je pojava JavaScripta donela nove mogućnosti obrade podataka na klijentskoj strani, što je znatno poboljšalo performanse tadašnjih internet sajtova.

Od tada, JavaScript se razvio u jednu važnu karakteristiku svakog većeg internet pretraživača na tržištu. Iako je u početku JavaScript podržavao isključivo rad sa validacijom elemenata, njegove karakteristike su se znatno poboljšale u kasnijim verzijama. Tako danas imamo programski jezik JavaScript, koji nam omogućava interakciju sa skoro svim aspektima koje pretraživač poseduje, počevši od samog sadržaja pa do celog prozora pretraživača (window). JavaScript je dobio priznanje da je programski jezik koji je sposoban da izvršava složena računanja i interakcije, čak i metaprogramiranje.

JavaScript je postao veoma važan deo interneta, tako da ga sada čak i alternativni pretraživači, uključujući i one na mobilnim telefonima, podržavaju. Brz napredak JavaScripta niko nije mogao da predvidi; od jednostavnog validatora ulaznih parametara, pa do čitavog programskog jezika.

Kratka istorija

Kako je internet postajao sve popularniji, razvijala se i potreba za skriptnim jezikom klijentske strane. U to vreme većina korisnika interneta je koristila modeme za uspostavljanje konekcije sa serverom, odnosno za pristupanje određenoj internet stranici. Korišćenje modema je za posledicu imalo veoma spor protok informacija, a internet stranice su postajale sve veće i složenije. Kao dodatni nedostatak, za svaku validaciju elementa koji unese korisnik uspostavljala se veza sa serverom, što je za rezultat imalo veliki gubitak vremena. Netscape je započeo razvoj skriptnog jezika poznatog pod nazivom Mocha, koji je kasnije postao LiveScript. Sa puštanjem verzije Netscape Navigator 2 počela je prva primena ovog skriptnog jezika. Netscape je započeo saradnju s kompanijom Sun Microsystems kako bi realizovao kompletnu implementaciju LiveScripta. Objavljena je verzija JavaScript 1.0, koja je doživela veliki uspeh, pa je Netscape ubrzo razvio noviju verziju 1.1, koja je bila implementirana u Netscape Navigator 3. Ubrzo nakon toga, Microsoft je odlučio da uloži više resursa u svoj pretraživač Internet Explorer, pa je nakon izlaska Netscape Navigatora 3, Microsoft 1996. godine lansirao Internet Explorer 3, koji je u sebi sadržao implementaciju JavaScripta, poznatu kao JScript.

Copyright © Link group

Uslovi za izvršavanje JavaScripta

Za izvršavanje koda JavaScripta zadužena je klijentska aplikacija. U većini slučajeva, to je Internet Explorer, FireFox, Chrome, Opera ili Safari, pri čemu korisnik ima slobodu konfiguracije izvršavanja koda JavaScripta, pa čak i eventualnog potpunog isključivanja. Zbog ovoga, treba obratiti pažnju pri upotrebi JavaScripta, jer korisnik možda neće hteti da prihvati njegovu aktivaciju, pa zato ne treba JavaScriptom ograničavati neke ključne elemente (na primer, da se neka kontrola uopšte ne iscrta ako nema JavaScripta i slično). U najgorem slučaju, nepostojanje JavaScripta (ma koliko redak slučaj to bio) treba uvek predvideti pri njegovoj upotrebi (na primer, ugradnjom noscript Taga). Danas je nezamislivo da koristite pretraživač koji ne poseduje podršku za JavaScript, no, sa druge strane, korisnik može namerno isključiti podršku za JavaScript na svom pretraživaču. Kao što vidite, korisnik ima svu moć nad kodom JavaScripta (čak ga može bez problema čitati i promeniti) i upravo zato u kodu JavaScripta nikada ne treba postavljati poverljive informacije, niti se valja osloniti na validaciju koju nam JavaScript obezbeđuje. Pored ove validacije, uvek treba izvršiti validaciju i na serverskoj strani.

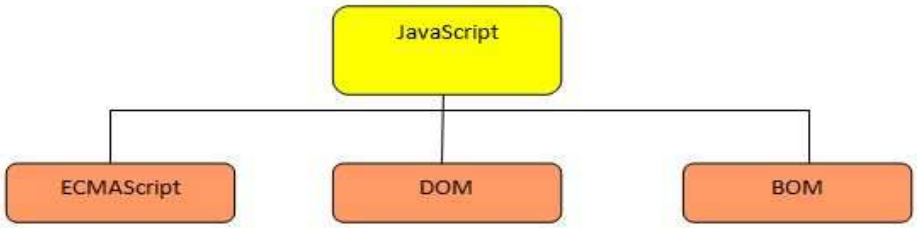
Uslovi za proizvodnju aplikacija

Da bismo stvorili neki kôd u JavaScriptu nisu nam potrebni posebni preduslovi. Samo kodiranje može se izvršiti i u najobičnijem tekst editoru. A najčešće se za svrhe kodiranja koriste isti alati kao i za HTML (DreamWeaver, Visual Studio, Notepad++...).

Pojam programiranja

JavaScript, kao i većina modernih programskih jezika, za osnovu ima programski jezik C. I iako se život JavaScripta odvija na samo korak od samog HTML-a, njihovi načini funkcionisanja su potpuno drugačiji.

Iako se JavaScript često koristi kao sinonim za ECMAScript, JavaScript je mnogo više nego ono što je definisano ECMAScriptom. Kompletna implementacija JavaScripta se sastoji od tri odvojena dela: Jezgro (ECMAScript), Document Object Model (DOM) i Browser Object Model (BOM), što možemo videti na slici:



Slika 1.2. Implementacija JavaScripta kroz ECMAScript, DOM i BOM

ECMAScript je jezik definisan standardom ECMA – 262. ECMA – 262 utvrđuje sintaksu, tipove, naredbe, ključne reči, operatore i objekte koje će jezik ECMAScript koristiti. JavaScript predstavlja implementaciju jezika ECMAScript.

Copyright © Link group

Pre svega, HTML zapravo i nije programski jezik, već samo sistem obeležavanja različitih elemenata dokumenta na različite načine putem tagova. Na primer, ukoliko imamo element: Moj sadrzaj , to nije ništa drugo do obaveštenje nekom čitaču da na strani postoji određeni element čiji je sadržaj »Moj sadrzaj«. To je jednostavan sistem tagova koji čini da HTML ne omogućava bilo kakve uslovne radnje tokom iscrtavanja tog sadržaja. Na primer, ako bismo želeli da, uz pomoć samo HTML-a, kažemo pretraživaču da pod nekim uslovom prikaže jedan, a pod drugim – drugi sadržaj, ne bismo uspeli.

JavaScript, za razliku od HTML-a, radi baš to – kontroliše tok nekog procesa i formira neku dinamičku strukturu, u zavisnosti od rezultata. Da bismo videli kako to radi, potrebno je da se upoznamo sa osnovnim načinom funkcionisanja jednog programskog jezika.

Ono što je odlika svakog programskog jezika jeste rukovanje nekim podacima kroz izvođenje različitih operacija nad njima. Ono što je u tom procesu bitno to je da ti podaci ne budu uvek isti, kao i da operacije ne budu uvek iste, jer jedino tako bilo kakav program može imati smisao. Zato se podaci obično i ne fiksiraju u programu, već se, isto kao i u matematičkim formulama, reprezentuju kroz promenljive. Na primer, ako naš program treba da sabira dva broja, on će glasiti ovako:

a=b+c

Na kraju nam ostaje samo da, na početku programa, dodelimo vrednosti promenljivima **b** i **c**, te da na kraju programa prikažemo vrednost promenljive **a**.

Pored promenljivih (varijabli), u programiranju važnu ulogu igraju i uslovi.

Svaki program teško da će biti funkcionalan bez bar jednog uslova. Dakle, kada bismo, u prethodnom primeru, želeli da prikažemo rezultat – samo pod uslovom da je **a** veće od nula, morali bismo negde da uporedimo vrednost promenljive **a** sa nulom i na osnovu toga odlučimo da li ćemo emitovati rezultat ili ga nećemo prikazati.

Sledeća bitna osobina jednog programa je njegov tok. Svaki program ima svoj početak i kraj (i, naravno, neki sadržaj). Bitno je da u svakom trenutku izvršavanja programa znamo u kom se njegovom delu nalazimo, kao i da možemo da lociramo neki njegov deo. Tok možemo kontrolisati (osim pomenutim poređenjima – uslovima) i različitim vrstama petlji, kada, u zavisnosti od nekog uslova, određene delove koda ponavljamo određeni broj puta, dok se neki drugi uslov ne ispuni. Na primer, ako bismo želeli da deset puta prikažemo rečenicu „moj program”, rekli bismo:

***a** je 0*

*Sve dok je **a** manje od 10 radi sledeće:*

prikaži rečenicu »moj program«

*povećaj **a** za jedan*

Često u programu postoji potreba za ponavljanjem jednog njegovog dela više puta, ali ne u okviru jednog intervala (što je slučaj sa malopređašnjim primerom), već samo na pojedinim mestima u programu. Na primer, želimo da pratimo trajanje programa i u više navrata želimo da nam program ispiše vreme proteklo od aktivacije programa. Na primer,

Copyright © Link group

početak programa
uzmi početno vreme
uradi operaciju jedan
uradi operaciju dva
...
uzmi trenutno vreme
oduzmi trenutno vreme od početnog vremena
emituj rezultat (vremensku razliku) korisniku

uradi operaciju tri
uradi operaciju četiri
...
uzmi trenutno vreme
oduzmi trenutno vreme od početnog vremena
emituj rezultat (vremensku razliku) korisniku

U ovom primeru dva puta se ponavlja blok sa preuzimanjem trenutnog vremena, oduzimanjem trenutnog vremena od početnog (dobijanje vremenske razlike) i emitovanja rezultata korisniku.

S obzirom na to da je u programiranju pravilo da se ništa ne ponavlja, ovakve blokove treba smestiti u posebne odeljke i pozivati ih u svakom trenutku kada su nam potrebni. Ti odeljci zovu se obično funkcije (u nekim jezicima i – procedure).

U slučaju funkcije, isti program izgledao bi ovako:

početak programa

funkcija vremenska razlika

uzmi početno vreme
uradi operaciju jedan
uradi operaciju dva
...
startuj funkciju vremenska razlika
uradi operaciju tri
uradi operaciju četiri
...
startuj funkciju vremenska razlika

Ovaj program, osim što je već posle nekoliko poziva iste funkcije kraći, omogućava izmenu svakog izračunavanja vremenske razlike, na samo jednom mestu.

Nemojte se truditi da u ovom trenutku zapamtite redosled pisanja izjava, jer ćemo se sintaksom baviti u narednim lekcijama. Sada je bitno da razumemo logiku kojom bi trebalo da se služimo prilikom pisanja programa.

Copyright © Link group

Pitanje

Koja je bila prvobitna svrha JavaScript jezika?

- **Validacija ulaznih elemenata na klijentskoj strani**
- Uvođenje animacije na stranu
- Manipulacija strukturom HTML dokumenta

Objašnjenje:

Kada se JavaScript pojavio 1995. godine, njegova glavna svrha je bila da izvrši određenu validaciju ulaznih elemenata, koja je sve do tada bila prepuštena serverskoj strani.

Zadaci za samostalni rad

1. Na osnovu prikazanih primjera iz lekcije, pokušajte da kreirate redosled koraka za program koji bi ispisivao ime, prezime i trenutni datum, sve dok je broj a manji od 5.

Rešenja svih zadataka možete pronaći unutar repozitorijuma kursa.

Rezime

- Za izvršavanje JavaScript koda zadužena je klijentska aplikacija.
- JavaScript nije isto što i Java.
- JavaScript je moguće proizvoditi u bilo kom editoru teksta.
- JavaScript živi od trenutka kada pretraživač raščlani kôd do trenutka zatvaranja strane u pretraživaču.
- JavaScript, kao i većina modernih programskih jezika, za osnovu ima programski jezik C.

Copyright © Link group