Uvod u JavaScript

Dinamične i interaktivne web stranice

- Kombinacija PHP-a i MySQL-a najuobičajenija je za dinamični, bazom podataka obilježeni web dizajn temeljen na otvorenom kodu
- Druga su (teža) mogućnost integrirani okviri tipa Ruby on Rails
- Spajanje PHP-a i MySQL-a osnova je razvoja društvenih mreža i početak Web-a 2.0
- Dodavanjem JavaScripta i CSS-a omogućeno je stvaranje visoko dinamičnih i interaktivnih web stranica

Uvod u js

- Donosi dinamičku funkcionalnost mrežnim stranicama (popup prilikom mouse over, pojava novog teksta, boja ili slika, micanje objekta na stranici…) – efekti koji su mogući samo na taj način jer se izvodi unutar preglednika i ima izravan pristup svim elementima web dokumenta
- Prvi se puta javlja u Netscape Navigator pregledniku 1995. prilikom dodavanja potpore Java tehnologiji – no ime je tek marketinški trik
- Srce web 2.0 Ajax tehnologije koja (uz HTML5) osigurava fluidni web frontend
- Skriptni jezik klijentske strane koji se u potpunosti izvodi unutar web preglednika i ima izravan pristup svim elementima web dokumenta
- JavaScript i PHP kao jezici više razine podržavaju strukturiranu programsku sintaksu programskog jezika C pa su vrlo slični
- Varijabli se može jednostavno mijenjati tip njenom upotrebom u novom kontekstu

Značajke js

- OO skriptni jezik namijenjen izvođenju na klijentskoj strani (php je na strani poslužitelja)
- Omogućuje dodavanje interaktivnih sadržaja web stranicama, pa preglednici imaju za njega ugrađenu podršku
- Interpretira se (kao i php) pa ne zahtjeva posebna razvojna okruženja (dovoljan je uređivač teksta)
- Omogućuje odgovaranje na akcije korisnika-posjetitelja, provjeru korisničkih unosa podataka u HTML obrasce te komunikaciju stranice s poslužiteljem ili među dokumentima bez potrebe za ponovnim učitavanjem dokumenta (AJAX)
- JavaScript kod najčešće se ugrađuje izravno u HTML dokument, ali se pojedini dijelovi koda mogu odvojiti od HTML dokumenta u zasebnu datoteku
- 1997. je standardiziran (ECMAScript standard)
- Pojavom AJAX tehnologije i brojnih programskih okvira (framework) koji olakšavaju rad širi mu se primjena
- U suvremenom pristupu izrade web siteova preporuka je strogo odvajati strukturu stranica (HTML), dizajn (CSS) i interakciju (JavaScript) - čime se „slojevi" mogu nezavisno mijenjati

Sintaksna pravila JavaScripta

- Pod snažnim utjecajem programskog jezika C
- Nazivi i konvencije imenovanja iz Jave
- U HTML dokument se ugrađuje:
  - Unutar početne oznake HTML elemenata pomoću posebnih JavaScript akcijskih atributa
  - Veći dijelovi koda ugrađuju se u zaglavlje HTML dokumenta kao sadržaj elementa script
  - Moguće je odvojiti u zasebnu datoteku koja se s HTML datotekom povezuje elementom script
- Razlikuju se velika i mala slova, ključne (rezervirane) riječi i imena funkcija pišu se malim slovima
- Prijelaz u novi red je kao ; u drugim jezicima, no ako je više naredbi u retku odvajaju se sa ; pa naredbe završavaju sa ;
- Prazna se mjesta ignoriraju, ali se preporuča preglednost, komentiranje je kao u C

JavaScript ugrađen u HTML element

- Najjednostavniji način pisanja programskog koda – umetanje koda u početnu oznaku nekog HTML elementa. Postavlja se kao vrijednost jednog od specijalnih atributa koji označavaju neki događaj (klik, učitavanje i sl.)

Primjer: ispis informacije o uspješnom učitavanju stranice (u okviru upozorenja preglednika)

- Poruku ispisati nakon što se dokument u potpunosti učita u preglednik
  - Registrirati učitavanje stranice, odnosno elementa tijela web-stranice u pregledniku
  - Pozvati funkciju koja će ispisati poruku u okviru upozorenja

```
<body>
<p>Stranica sa ugrađenim JavaScript kodom</p>
</body>
```

- Budući da poruku o učitavanju stranice u preglednik trebamo ispisati nakon što se u pregledniku prikaže cijela stranica, odnosno nakon što se učitaju svi elementi unutar elementa body, js kod postavljamo unutar početne oznake elementa body
- Događaj koji će biti pokretač ispisa poruke je učitavanje elementa body pa upotrebljavamo poseban atribut HTML elementa (onload)

Atribut onload

- Standardni atribut HTML elementa body
- Nakon učitavanja elementa body (registrira se pomoću atributa onload) izvodi se vrijednost atributa – js programski kod-naredba alert:

```
<body onload=„alert(‘Stranica je učitana u preglednik’); ”>
<p>Stranica sa ugrađenim JavaScript kodom</p>
</body>
```

- Naredba poziva funkciju alert te joj prosljeđuje poruku koju želimo ispisati. Funkcija alert poziva okvir upozorenja i predaje okviru željenu poruku.
- Kod suvremenog pristupa izradi web stranica preporučuje se js kod držati odvojen od HTML dokumenta ili barem ugrađenog u zaglavlje dokumenta, a u elementima samo pozivati programske funkcije.

Pisanje JavaScript koda unutar elementa script

- Veći dijelovi JavaScript koda ugrađuju se kao sadržaj elementa script
- Unutar elementa script definira se korisnička funkcija i u njoj se smještaju naredbe js koda, a u elementu se samo poziva ta funkcija

```
<body onload=„izvrsi()”>
        <script>
        function izvrsi(){
                alert(‘Stranica je učitana u preglednik!’);
        }
        </script>
<p>Stranica sa ugrađenim JavaScript kodom!</p>
</body>
```

Primjer

- Poziva se smještajem između <script> i </script> HTML tagova

```
<html>
        <head> <title> hello world </title> </head>
        <body>
                <script type="text/javascript">
                document.write ('Hello World!')
                </script>
                <noscript>
                preglednik ne podržava javascript ili je onemogućen
                </noscript>
        </body>
        </html>
```

Smještaj

- Može se smještati u body dokumenta ili u <head> sekciju, što je idealno mjesto ako se skripta (kritični kod ili funkcije) želi izvesti kod učitavanja stranice
- Drugi razlog za smještaj skripte u head sekciju dokumenta je da se javascriptu omogući da piše stvari kao što su meta tagovi u <head> sekciju
- Ako se tu smjesti kritični kod i funkcije, može se osigurati da su spremne za korištenje neposredno u bilo kojoj skripti neke sekcije u dokumentu na koje se odnose
- Drugi razlog smještaja skripta u zaglavlje dokumenta je da se omogući js da piše stvari kao što su meta elementi u zaglavlje
- JavaScript kod može se uključiti s web sjedišta, odnosno s interneta sintaksom <script type=„text/javascript" src=„script.js"></script> ili <script type=„text/javascript" src=„http://nekiserver.com/script.js"></script>

Navođenje elementa script u zaglavlju dokumenta

- Element script može se navesti na bilo kojem mjestu unutar elementa body, no česta je praksa smjestiti ga u zaglavlje HTML dokumenta

```
<html>
<head>
        <script>
        function izvrsi(){
                alert('Stranica je učitana u preglednik!');
        }
        </script>
</head>
<body onload=„izvrsi()”>
        <p>Stranica sa ugrađenim JavaScript kodom!</p>
</body>
</html>
```

Unutar elementa head smješten je element script i u njemu definirana funkcija izvrsi, a u elementu body je samo poziv te funkcije.

Pisanje js koda na ovakav način omogućuje njegovo odvajanje od HTML koda stranice. Sav js kod se nalazi u elementu head, čime se ne mješa s HTML elementima u elementu body koji prikazuju sadržaj stranice.

Js kod u zasebnoj datoteci koja se s HTML datotekom povezuje elementom script

- Js kod može se pisati izravno u HTML dokumente ili se može uključiti datoteke js koda s web stranice ili s interneta
- <script type=„text/javascript” src=„script.js”></script>
- <script type=„text/javascript” src=„http://someserver.com/script.js”></script>
- Bez <script> i </script> elementa jer preglednik već zna da se poziva js datoteka, njihovo pisanje u js datoteke uzrokuje pogrešku
- Preferirani način uključivanja js datoteka (drugih autora) na web stranicu

Izdvajanje js koda u zasebnu .js datoteku

- U elementu head navodi se element script koji u svom atributu src sadržava putanju do .js datoteke
- Prilikom učitavanja HTML dokumenta preglednik automatski učitava i js kod svih na ovaj način povezanih datoteka

```
<html>
<head>
•           <script type=„text/javascript” src=„vanjska.js”></script>
</head>
<body onload=„izvrsi()”>
            <p>Stranica sa ugrađenim JavaScript kodom!</p>
</body>
</html>
```

- Element script u svom atributu src sadrži relativnu putanju do .js datoteke (koja je u istoj mapi u kojoj se nalazi i HTML dokument). Sadržaj atributa type određuje o kakvom je tipu datoteke riječ. Vijrednost text/javascript predstavlja datoteku sa js kodom. Kod datoteke vanjska.js sadrži definiciju funkcije izvrsi.

Upotreba komentara i ;

- //
- /* i */

- Ako je više naredbi u retku treba staviti ;
- U slučaju nesigurnosti bolje je staviti

Varijable, operatori i kontrola toka programa

Varijable

- Imaju ime i tip. U js nije nužno navoditi tip varijable, js nakon inicijalizacije automatski dodjeljuje tip varijabli.
- Prije upotrebe varijable je potrebno deklarirati i dodijeliti im inicijalnu vrijednost. Varijable se deklariraju ključnom riječi var koja se može i izostaviti: var a; ili samo a=4; Deklaracijom se određuje ime, a inicijalizacijom (upotrebom operatora =) dodjeljuje neka vrijednost.
- Slova, brojke, $ i _, prvi znak ne može biti brojka
- Imena su case-sensitive
- String varijable ograničavaju se navodnicima (‘ ili „)
- Polja: karte=[‘pik’, ‘tref’,’srce’,’karo’], slično za dvodimenzionalna po redovima
- Pristupa se preko indeksa: document.write(polje[1][2])

Tipovi varijabli

- Određuju se kad se varijabli pridruži vrijednost i mogu se mijenjati kod pojavljivanja varijable u različitim kontekstima
- typeof operator se može koristiti za ispis tipa
- Primjenjena operacija može konvertirati varijablu u drugi tip

Globalne i lokalne varijable

- Globalne varijable - definirane izvan bilo koje funkcije (ili unutar funkcije, ali definirane bez ključne riječi var) – može im se pristupiti iz bilo kojeg dijela scripte
- Parametri koji se predaju funkciji automatski imaju lokalni doseg – može im se pristupiti samo iz te funkcije; iznimka su polja koja se funkciji predaju preko reference, tako da promjena na elementima polja ima za posljedicu i promjenu elemenata izvornog polja
- Da bi se definiralo lokalnu varijablu koja ima doseg samo unutar neke funkcije, a nije predana kao parametar, koristi se ključna riječ var
- Primjer: if (typeof a!=‘undefined’) …

Operatori

- Aritmetički, nizovni, logički, pridruživanja, usporedbe, za polja, bitove i drugo
- Mogu biti unarni, binarni i ternarni
- Operatori imaju prioritete
- Smjer procesiranja operatora naziva se asocijativnost (važna je kad se ne oslanjamo eksplicitno na prioritete)
- Operatori mogu imati lijevu ili desnu asocijativnost (++ i – nemaju asocijativnost)

Operatori

- Matematički (aritmetički: +, -, *, /, %, ++, --)
- Operatori pridruživanja i složenog pridruživanja (=, +=, -=, *=, /=, %=)
- Relacijski operatori usporedbe (<, >, <=, >=, ==, === znači jednako i istog tipa, !== različito), upotrebljavaju se u logičkim izrazima (prilikom kontrole tijeka programa), ispituju dva operanda i vraćaju true ili false
- Logički operatori (&&, ||, !, nema xor) kao rezultat daju logičku vrijednost koja može imati dva stanja: logičku istinu ili laž
- Ako su operandi prilikom ispitivanja jednakosti različitih tipova, JavaScript ih konvertira u smisleni tip; kod korištenja operatora identičnosti automatska konverzija tipova se ne događa
- Spajanje stringova izvodi se sa +. U slučaju zbrajanja podataka tipa string s brojem, rezultat je uvijek podatak tipa string. String se piše pomoću znakova navoda (‘ ili „).
- U stringovima se mogu koristiti specijalni esc znakovi (npr. \n)

Ispisivanje – document.write

- Funkcija document.write ispisuje na lokaciju trenutnog preglednika – dobro za brzi ispis rezultata, izlaz se ispisuje u preglednik odmah uz web sadržaj i kod
- Neki ovu funkciju smatraju nesigurnom jer kad se poziva nakon što je web stranica u potpunosti učitana, ona će prepisati tekući dokument (to je moguće, no neće se dogoditi u slučaju kad se document.write koristi kao dio procesa kreiranja stranice - tako da se poziva samo prije nego je stranica u potpunosti učitana i ispisana) – moguće za jednostavne primjere
- Ne koristi se u produkcijskom kodu (osim iznimno, kad je zapravo nužna), nego se često piše izravno na posebno pripremljen element (npr. korištenjem svojstva innerHTML izlaznih elemenata)

Ispisivanje – console.log, alert, HTML element

- Funkcija console.log ispisuje rezultat vrijednosti ili izraza koji se daju kao argument na konzolu trenutnog preglednika
- Specijalan način sa prozorom odvojenim od prozora preglednika, a u kojem se mogu ispisivati poruke o greškama i druge poruke
- Način pozivanja i rada konzole razlikuje se kod preglednika
- Funkcija alert ispisuje u pop-up prozoru sa gumbom za zatvaranje (nespretno), a ispisuje samo trenutnu poruku, prethodne su obrisane: alert ('1\n2\n3');
- Moguće je ispisivati i izravno u tekst HTML dokumenta (najbolje za produkcijske sajtove) – traži kreiranje dokumenta i JavaScript kod da mu se pristupi

Izrazi

```
<script>
    document.write(„a: „+(8==2)+”<br>”);
</script>
```

False su: „false”, 0, prazan string, null, undefined, NaN (ilegalna operacija)

Najjednostavniji oblik izraza je literal, vrijednost

Izraz može biti i varijabla

Kontrola toka izvođenja programa – uvjetno grananje – naredba if()

- Omogućuje nam izvođenje različitih blokova naredbi s obzirom na ispunjavanje (istinitost) postavljenog uvjeta
- If-elseif-else

Naredba switch()

- Korisna kad neka varijabla ili rezultat izračunavanja izraza mogu imati više različitih vrijednosti koje zahtjevaju izvođenje različitih funkcionalnosti
- Npr. predaja stringa kodu glavnog izbornika ovisno o zahtjevu korisnika: <script> switch(page){case „Home”: document.write(…);break; case „About”:…}</script>
- default: //ako nije zadovoljen niti jedan uvjet

Operator ?

- Ternarni operator
- size=a<=5?”short”:”long”;

Ponavljanje izvođenja programskog koda i petlje: petlje for(), while() i do-while()

- Kad se određeni blok instrukcija treba izvršiti više puta za ponavljanje koriste se programske petlje
- Petlja for koristi se kad unaprijed znamo točan broj ponavljanja izvođenja bloka instrukcija
- Petlju while() upotrebljavamo kad ne znamo točan broj ponavljanja
- Za raniji izlazak ili preskakanje dijela koda koristi se break, odnosno continue

Naredba with

- Omogućuje pojednostavljenje nekih vrsta naredbi reduciranjem sa većeg broja referenci na objekt na samo jednu referencu – podrazumijeva se da se sve reference na svojstva i metode unutar with bloka odnose na taj objekt

```
<script>
        string=„neki tekst”
        with(string)
        {
           document.write(„Duzina stringa je: ”+length+” znakova.<br>”)
           document.write(„Velikim slovima: „+toUpperCase())
        }
</script>
```

Interpreter prepoznaje na koji se objekt odnosi svojstvo i primijenjuje metoda.