Podatkovni višemedijski prijenos i računalne mreže

Fakultet elktrotehnike i računarstva
Zagreb, Unska 3

Zavod za elektroničke sustave
i obradu informacija

ActionScript

Dario Plichta
0036375659

Uvod

Protok podataka je razlog zašto medijske mogućnosti prezentacije na Internetu nisu na nivou onih koje se skladište na CD-u. Dok CD može slobodno sadržavati kompleksne programe i nekomprijesirane video i audio zapise, slike i velike količine tekstualnih podataka, na web prijezentaciji , uz uvažavanje ograničenog protoka, je "dozvoljeno" mnogo manje – toliko da posjetilac može primiti informaciju koja mu se nudi u nekom, dovoljno kratkom, roku. Sa druge strane, raznovrsnost sadržaja koji se može naći na globalnoj mreži postavlja zahtjev autoru prezentacije da pružena informacija omogući korisniku da prati tok informacija koji mu se nudi putem prezentacije. Korisnik Takođerr ima svoje zahjteve: želi "što više – to bolje" u roku "što brže – to bolje"; a ukoliko mu se po tom kriteriju ne izađe u susret, jednostavno će krenuti zadovoljavati svoje potrebe na nekoj drugoj lokaciji, na nekoj drugoj prijezentaciji.

Problematika kreiranja HTML prezentacije

Autor mora imati na umu da sadržaj web stranice (osnovne cjeline u prijezentaciji) bude čitljiv na očekivani način od strane web-pretraživača koji posjetilac koristi. Međutim, ovo je osnovni kamen spoticanja pri kreiranju i dizajniranju stranice putem klasičnog HTML pristupa jer sam prikaz stranice zavisi od mnogo faktora: koji tip pretraživača se koristi i koja verzija na kom operativnom sistemu, da li su fontovi koji se specificiraju na stranici instalirani na računalu posjetioca, koju rezoluciju koristi posjetilac...

HTML – standardizacija i konkurencija

Jezik HTML prijedstavlja jezik za opis podataka za prikaz sadržaja stranica web prezentacije. Cilj jezika je da omogući prijedstavljanje različitih sadržaja posjetiocu prezentacije na način koji osmisli autor. Da bi se tome izašlo u susret, 1994. osnovan je web konzorcijum (*http://www.w3c.org/*), čiji zadatak je da se na svjetskom nivou brine o standardizaciji protokola, jezika i tehnologija od značaja za razvoj svjetske mreže. Tako je, na Primjer, jedan od njihovih dokumenata i specifikacija jezika HTML, zvanično u nazivu "prijeporuka" (umjesto specifikacija). Dokument sadrži opis elemenata jezika i načine upotrebe.
Upoznavanjem sa specifikacijom, autor bi trebalo biti u stanju da, prateći prijeporuku, sprovede svoje ideje u projekt prezentacije i omogući na taj način da posjetilac prezentacije može vidjeti stranice projekta koje izgledaju točno onako kako je autor osmislio.

U praksi, čitače HTML strana, koji su poznatiji pod nazivom web-pretraživači, proizvode različite kompanije čiji cilj je, prije svega, osvajanje tržišta i, kao posljedica toga, ostvarenje profita. Stoga kompanije nude tržištu verzije programa koje zadovoljavaju one aspekte jezika, koje sama kompanija smatra bitnim ili koje je uspela da realizira, istovrijemeno nudeći neka druga rješenja čime se stiče prijednost u odnosu na konkurentske kompanije i njihove programe.
Ovakva stvarnost autoru prezentacije stvara samo probleme. Autor mora poznavati točno šta koja verzija kog programa može pružiti, ukoliko i dalje želi osigurati posjetiocu svoje prezentacije prezentiranje sadržaja na željeni način. Iako suvrijemeni programi za kreiranje web prezentacija posjeduju određenu bazu znanja o ovoj problematici, krajnji rezultat je: ili povećanje količine podataka za opis sadržaja (zbog raznih ispitivanja o pretraživaču klijenta i kreiranja različitih rješenja za isti prikaz), ili optimizacija sadržaja za pojedine pretraživače (čime se oštećuju posjetioci koji koriste ostale), ili korištenje najjednostavnijih elemenata jezika, podržanih od svih pretraživača (gubi se atraktivnost naprijednih elemenata).
Dobra ilustracija autorske muke je mogućnost korištenja klijentskih skript jezika u okviru HTML stranica. Ovi jezici omogućavaju dodavanje različitih efektnih rješenja za proširenje funkcionalnosti stranice kao, na Primjer, poboljšanje navigacijskog sistema. I opet novi problemi: jezik VBScript prepoznaće samo Internet Explorer, a mogućnosti JavaScripta (u Internet Exploreru JScript) zavise od modela dokumenta koji poznaje promatrana verzija promatranog pretraživača. Znači, programira se u JavaScriptu a “univerzalnost” se osigurava sljedećom konstrukcijom (pseudokod):

```
ako je Netscape
    ako je verzija manja od 4
        //   radi ovo
    inače
        ako je verzija 4.x
            //   ovo za 4.x
        inače //   verzija je 6
            //   i tada radi ovo
inače //   nije Netscape, znači IE
    ako je verzija....
```

Rezultat je različita realizacija istih zadataka u različitim uvjetnim granama. Umanjuje se prijeglednost koda, povećava se veličina datoteke, produžava se vrijeme učitavanja stranice, povećava se količina podataka koja klijentu nije od koristi...

Bitmap grafika

Grafičke datoteke – fotografije, crteži, navigacijske slike i drugo, na HTML stranici najšire su prisutni u dva formata - .jpg i .gif. To nije slučajno, jer najbolje odgovaraju zahtjevu da pruže što više informacija u što manjoj datoteci. Stranica je učitana tek onda kada su učitani svi njeni elementi. Znači da ukupna količina podataka, koja treba stići do klijenta da bi pretraživač mogao ispravno prikazati stranicu, nije manja od zbira veličina datoteka svih elemenata stranice. Također, svaki element putuje po posebnom zahtjevu, što uvjetuje određena kašnjenja u komunikaciji klijenta i servera i produžava vrijeme učitavanja. Keširanje elemenata koji se ponavljaju na različitim stranicama je od pomoći, ali ne i pri prvom pristupu prijezentaciji (prva stranica je najbitnija u odluci posjetioca hoće li ostati ili odustati od posjete).

Animacija

Klasični video formati neprimenljivi su za primjenu na Internetu zbog veličine zapisa datoteka. Iako je sve raširenija upotreba tipova zapisa koji koriste velike stupnjeve komprijesije podataka, ona ne osigurava prikaz u realnom vrijemenu preko globalne mreže. Pojava streaming formata zapisa dozvoljava prikaz prije skidanja kompletnog datoteke, mada je realno primenljiva samo pri korištenju zapisa manjih dimenzija (u pikselima) i na djelovima mreže koji mogu obezbjediti potreban protok za prikaz.

Odgovor: Flash

Flash je proizvod kompanije Macromedia čiji se datoteke (filmovi) sa ekstenzijom .swf prikazuju u web-pretraživaču putem plug-in dodatka. Flash plug-in Player je razvijen i razvija se za sve vodeće platforme (procesor + operativni sistem + pretraživač). Format .swf je u formi open-source što omogućava njegovo brže i raširenije prisustvo u svjetskim razmjerama. Na naslovnoj stranici prezentacije firme Macromedia najsvježiji podatak je da 98.3% korisnika Interneta posjeduje ovaj dodatak. Ovo praktično govori da Flash je postao standard u svjetu globalne mreže i, uz činjenicu da većina novijih pretraživača pri samoj instalaciji postavljaju ovaj plug-in u svoje okruženje, da onaj “ostatak” uglavnom predstavljaju korisnisnici zastarjelih verzija programa koji ionako ne mogu udovoljiti zahtjevima suvremenih prezentacija.

Nezavisnost od platforme

Znači, svaki posjetilac koji posjeduje Flash Player ima mogućnost da pogleda .swf film jer ne zavisi od platforme sa koje pristupa datoteci. Dalje, postoji mogućnost da se svi fontovi potrebni za pjezentaciju u Flashu uključe u dokument, tako da izgled ne zavisi ni od fontova instaliranih na računalu. Svi grafički elementi u Flashu, pa i fontovi, su vektorski opisani (ako se izuzme mogućnost uvoza bitmap slika) tako da je moguće skaliranje cjelog filma prema prozoru pretraživača odnosno – nema zavisnosti od ekranske rezolucije.
Kako Flash ne zavisi od platforme, konstrukcije ispitivanja okruženja zbog prilagođenja prikaza nisu potrebne – svaka realizacija prikaza radi se točno jednom.

Vektorska grafika

Opis vektorske grafike je jednostavan i nema gubitka podataka jer nema ni komprijesije. Film je u jednoj datoteci tako da server obrađuje samo jedan zahtjev pri slanju filma klijentu. Ukupna veličina .swf datoteke je manja od analogno tome rađene HTML stranice. I sljedeće: Flash film može početi da se prikazuje odmah nakon učitavanja prvih kadrova filma, prije nego što se kompletno učita kod klijenta.

Animacija i interakcija

Flash datoteka naziva se film zbog mogućnosti animacije, osnovne prijednosti na Internetu u odnosu na sva druga ponuđena rješenja.
Primjer: animacija kvadrata koji se rotira po nekoj putanji i pritom mjenja svoju veličinu, pamti se kao objekat opisan pozicijom nekoliko točaka, početnom i krajnjom pozicijom objekta, faktorom slaliranja objekta i parametrom broja okretaja tokom animacije. Cjeli taj opis može biti smješten u desetak bajtova, bez obzira na veličinu kvadrata i broj slika koje učestvuju u cjeloj animaciji. Realizacija iste ove animacije u

bitmapiranom svjetu je niz potrebnog broja slika (dužina animacije), od kojih svaka slika kompletno opisuje sadržaj boje svojih točaka (broj točaka zavisi od dimenzija – širina puta visina) ili promjenu sadržaja u relaciji na prethodnu sliku. Znači, dimenzije animacije i njena dužina značajno utječu na veličinu zapisa animacije.
Jedan od elemenata Flasha su objekati koji mogu da reagiraju na određene događaje, među kojima i su korisnikove akcije unosa putem tipkovnice ili akcije miša, koji dalje mogu uvjetovati ponašanje u filmu. Kompletan sadržaj filma u Flashu može se mjenjati, kao odgovor na korisničke akcije, bez potrebe za dovlačenjem sadržaja sa servera, jer se cjelokupni sadržaj može se nalaziti u okviru samog filma. U okviru Flasha se za upravljanje kontrolom filma koristi jezik ActionScript.

ActionScript

ActionScript je skript jezik čiji je cilj kontrola objekata u Flash filmovima, kreiranje navigacionih i interaktivnih elementa i za kreiranje filmova visokog stepena interakcije i web aplikacija.

Skript jezik

Skript jezik je programski jezik koji se koristi za manipulaciju, prilagođenje i automatizaciju postojećeg sistema. U takvim sistemima, funkcionalnost je već osigurana putem korisničkog sučelja, a skript jezik prijedstavlja mehanizam za proširenje funkcionasti programskom kontrolom. Za postojeći sistem se kaže da osigurava domaćinsko okruženje za objekte i mogućnosti koje kontrolira skript jezik. Asocijacija europskih proizvođača računala (ECMA – the European Computers Manufacturers Association, *http://www.ecma.ch/*) sastavila je dokument pod nazivom ECMA-262 čiji je cilj standardizacija jezika JavaScript. ActionScript zasniva se na specifikaciji ovog dokumenta.
-:- *http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM*

1. Elementi jezika

Sintaksa ActionScripta podsjeća na Javu. Razlika je u tome što jezik upućuje na jednostavnost u korištenju. Na Primjer, varijabla ne mora imati deklariran tip niti se tipovi povezuju sa osobinama objekta, a definirane funkcije ne moraju biti deklarirane u tekstu prije njihovog poziva.
ActionScript se zasniva na objektima: osnovni jezik i okruženje Flasha osiguravaju se putem objekta, a sam program u ActionScriptu je put komunikaciji među objektima. Sam objekt je kolekcija osobina (properties) koje mogu imati atribute koji određuju način korištenja osobine (na Primjer – ReadOnly). Osobine su kontejneri koji sadržavaju druge objekte, primitivne vrednosti ili metode. Objekt u ovom smislu prijedstavlja referencu na objekt, primitivna vrednost je vrjednost nekog primitivnog tipa podataka, a metoda je funkcija koja se s objektom povezuje putem osobine.

1.1. Tipovi podataka i varijable

Tip podataka prijedstavlja vrstu informacije koju varijabla ili element ActionScripta može sadržavti. Tipovi podataka djele se na primitivne i referentne tipove, a postoje i specijalni tipovi podataka.

- Primitivni tipovi su *string*, *number* i *boolean*
- Referentni tipovi su *object*, *function* i *movieClip*
- Specijalni tipovi podataka: *null* i *Undefined*

Referentni tipovi kao vrjednost imaju adresu svog sadržaja. Tip *movieClip* je grafička reprezentacija elemenata Flasha.

```
var myVar;
trace (“typeof (myVar): ” + typeof (myVar));   //    undefined
myVar = true;
trace (“typeof (myVar): ” + typeof (myVar));   //    boolean
myVar = 0;
trace (“typeof (myVar): ” + typeof (myVar));   //    number
myVar = “my String”;
trace (“typeof (myVar): ” + typeof (myVar));   //    string
myVar = new Array ();
trace (“typeof (myVar): ” + typeof (myVar));   //    object
myVar = trace;
trace (“typeof (myVar): ” + typeof (myVar));   //    function
myVar (“typeof (myVar): ” + typeof (myVar));   //    function
```

Varijabla se može deklarirati sa *var*. Deklaracija određuje njeno ime i lokalnost, odnosno varijabla tada postaje lokalna u bloku u kome je deklarirana. U navedenom primjeru korištene su funkcije *trace*, koja služi za ispis string vrjednosti izraza svog argumenta u posebni izlazni prozor (korisno pri testiranju programa), i *typeof*, koja vraća naziv tipa izraza svog argumenta. Dvostruka kosa crta (*//*) prijedstavlja početak linijskog komentara, pri interprijetaciji se ignoriraju svi karakteri u liniji nakon ove oznake (komentar u više linija postavlja se između oznaka */** i ***/**). Prvi poziv funkcije *trace* u izlazni prozor ispisuje “typeof (myVar): undefined” jer varijabli nije dodjeljena vrjednost, tako da joj nije određen ni tip. Dodjela vrjednosti određuje tip varijable. To znači da varijabla može mjenjti svoj tip. U primjeru, posljednjom dodjelom (*myVar = trace*) je varijabli dodjeljena funkcija *trace*, znači ona je tipa *function*, ona postaje funkcija koja se ponaša na isti način kao i *trace*, tako da posjlednja linija poziva varijablu *myVar* kao funkciju i dobija se također odziv u izlaznom prozoru.
Varijabla se ne mora deklarirati. U tom slučaju smatra se za globalnu.
ActionScript vrši automatsku konverziju tipova pri izvođenju operacija, tako da nedefinirane varijable mogu učestvovati u operacijama, gde se prevode kao neutralni element zbrajanja (logičko – *false*, numerički – *0*, string – *“”*).

1.2. Operatori

ActionScript definira skup ugrađenih operatora koji se mogu podjeliti u sljedeće grupe: numerički operatori, operatori usporedbe, String operatori, logički operatori, bit-orjentirani operatori, operatori jednakosti i operatori dodjele. Kao i u drugim programskim jezicima, operatori su hijerarhijski određeni po prioritetu izvršavanja operacija.
Zanimljivi operatori su, recimo, operatori dodjele, što pokazuje sledeći Primjer:

```
var myString = “my”;
myString += “ String”;
trace (“myString: “ + myString);   //    “myString: my String”
```

Prva linija deklarira varijablu *myString* i dodeljuje mu (“obična” dodjela) String vrjednost “my”. U drugoj linija koda također je dodjela, ali sa operatorom *+=*, što semantički prijedstavlja sljedeće: zbroji vrjednost operanda s ljeve strane operatora (varijablu *myString*) sa operandom s desne strane (“ String”) i rezultat zbrajanja dodjeli varijabli s ljeve strane.
Drugi primjer pokazuje operatore inkrementacije (*++*):

```
var myVar = 0;
trace (“myVar: “ + myVar);         //    “myVar: 0“
trace (“myVar++: “ + myVar++);     //    “myVar++: 0“
trace (“myVar: “ + myVar);         //    “myVar: 1“
trace (“++myVar: “ + ++myVar);     //    “++myVar: 2“
trace (“myVar: “ + myVar);         //    “myVar: 2“
```

Post-inkrementacija povećava vrijednost varijable nakon njenog učestvovanja u izrazu, a prije-inkrementacija prvo povećava vrjednost varijable koja daje svoju novu vrjednost za izračunavanje izraza.
Treći Primjer, uvjetna dodjela:

```
var myBool = true;
var notBool = !myBool;
trace (“myBool: “ + ((myBool) ? myBool : notBool));
trace (“notBool: “ + ((notBool) ? myBool : notBool));
```

Uvjetna dodjela je ternarni operator. Prvi operand je logički izraz koji se izračunava. Ako je izračunao *true*, operator računa vrjednost drugog operanda (iza znaka pitanja) i to vraća kao rezultat, inače (izračunato *false*) se računa vrijednost trećeg operanda (iza dvotočke) i to postaje rezultat. U ovom primjeru prvi poziv trace ispisuje “myBool: true”, jer se ispituje uvjet *myBool* (*true*) i vraća vrijednost drugog operanda (*myBool* ima vrijednost *true*), dok drugi poziv ispisuje “notBool: false” jer je uvjet netočan (*notBool*, pri dodjeli je *!myBool*, negacija od *true*) pa se vraća vrijednost trećeg operanda (*notBool* je *false*).

1.3. Grananje i petlje

Naredba *if* u ActionScriptu ima sledeću sintaksu:

```
if (condition) {
    //    statements
} else {
    //    statements
}
```

Ovdje je *condition* logički izraz, obavezno naveden u zagradama, na osnovu kojeg se donosi odluka da li se izvršava kod bloka naredbi unutar vitričastih zagrada. Blok naredbi se uvjek navodi unutar vitričastih zagrada, a pojedine naredbe u bloku

razdvajaju se točka-zarezom (;). U slučaju samo jedne naredbe nije neophodno navođenje vitričastih zagrada. Druga grana (*else*) Također nije obavezna. U ActionScriptu kod Flasha 5 ne postoji *switch* naredba za višestruko grananje, a to je jedna od novina koje donosi nova verzija programa, Flash MX.
Ponavljanje djela koda osigurano je petljama *for*, *while*, *do-while* i *for-in*. U svim petljama mogu se koristiti komande *continue*, za skok na sledeću iteraciju, i *break*, za napuštanje petlje.

```
for (init; condition; next) {
    //  statements
}
```

Slično kao i u jeziku Java, *for* petlja zadaje se s tri parametra: inicijalna vrijednost kontrolne varijable, izlazni uvjet i izraz za izračunavanje prije naredne iteracije. Po definiciji je dozvoljeno izostavljanje bilo kog od ova tri djela (ostaje točka-zarez). Izraz *init* se izračunava samo jednom, prije početka prve iteracije, izraz *condition* mora biti logički i izračunava se prije svake iteracije i odlučuje da li se nastavlja sa iteracijom (izračunato *true*) ili ne (*false*), izraz *next* izračunava se na kraju svake iteracije. Ovakva konstrukcija dozvoljava mnogo veću slobodu pri baratanju petljom u odnosu na, recimo, pascalovu konstrukciju. Ipak, uobičajeno se koristi baš simulacija te osnovne konstrukcije, postavljanjem inicijalne vrijednosti kontrolne varijable u prvom djelu, usporedba kontrolne varijable s nekom izlaznom vrijednošću u drugom djelu i inkrementiranje ili dekrementiranje kontrolne varijable u trećem djelu.

```
while (condition) {
    //  statements
}
```

Dok god je uvjet *condition* ispunjen izvršava se blok naredbi petlje *while*. Ovo ispitivanje može biti i numeričko, u kom slučaju se ispituje različitost od vrijednosti *0*.

```
do {
    //  statements
} while (condition)
```

Slično je i kod petlje *do-while*, s razlikom što se prvo izvršava blok naredbi, pa tek onda proverava uvjet (što znači, kod bloka izvršava se bar jednom). Ovo podsjeća na paskalsko *repeat-until*, a razlika je kod uvjeta – *do-while* radi dok je ispunjen uvjet, a *repeat-until* prijekida po ispunjenju uvjeta.

```
for (variableiterant in object){
    //  statements
}
```

Petlja *for-in* koristi se osobinama objekta. Primjer:

```
myObject = {name:'Tara', age:27, city:'San Francisco'};
for (name in myObject) {
    trace ("myObject." + name + " = " + myObject [name]);
}
```

Primjer je iz rječnika ActionScripta 5. Izlaz izgleda ovako:

myObject.name = Tara
myObject.age = 27
myObject.city = San Francisco

Prva linija koda daje jedan način zadavanja objekta. Objekt *myObject* ima 3 osobine, svakoj je dodjeljena određena vrijednost. Varijabli name u *for-in* petlji pridružuju se imena osobina objekta *myObject*, redom kroz iteracije. Primjer Također prikazuje dva načina pristupa vrijednostima osobina objekta. U izlaznom prozoru prikazan je prikaz referenciranja "točkom", odnosno navodi se ime objekta, točka i sljedi ime osobine. To i je uobičajen način referenciranja, ali nije bilo moguće unutar *for-in* petlje pozvati *myObject.name*, jer bi se to odnosilo na osobinu name objekta, a ne varijablu petlje. Stoga je korištena referenca *myObject [name]*, koja izračunava izraz u uglastim zagradama (vrijednost varijable *name*) i potom traži vrijednost odgovarajuće osobine objekta.

1.4. Funkcije

Definicija funkcije:

```
function myFunc ([arg0, arg1,...argN]) {
    //  statements
}
//  ili
function ([arg0, arg1,...argN]) {
    //  statements
}
```

Funkcija je blok koda koji se može više puta koristiti pozivom funkcije. Primjer poziva bio bi *myFunc ([par1, par2,...parN])*. U zagradi ovde nalaze se parametri funkcije, koje ona pri izvršavanju ona prima kao argumente. Uglaste zagrade su u primjeru navedene u smislu BNF notacije, odnosno ukazuju na opcionu pojavu parametara i argumenata. Funkcija ove parametre vidi kao vrijednosne parametre, nema varijabilnih parametara, te se ne mogu mjenjati vrijednosti ovih parametara van lokalnosti funkcije, uz dužnu pažnju referentnim podacima, gde se može direktno pristupati osobinama objekta, na primjer.
Ulazne parametre funkcija vidi kao niz *arguments*, broj deklariranih imenovanih parametara funkcije ne mora se poklapati sa brojem argumenata koji se prosljeđuju funkciji. Pristup imenovanim parametrima u funkciji moguć je direktno navođenjem imena parametra, a pristup proizvoljnom parametru ostvaruje se kao pristup elementu niza *arguments* na određenom indeksu. Primjer:

```
function myFunc () {
    var myVar;
    for (var i = 0; i < arguments.length; i++) {
        myVar += arguments [i];
    }
    trace ("myVar: " + myVar);
}
```

```
myFunc (3, 4, “to”, 1, 1);   //    “myVar: 7to11”
```

U primjeru se lokalna varijabla *myVar* samo deklarira, inicijalno nema vrijednost, znači tipa je *undefined*. Prvo zbrajanje je sa argumentom tipa *number*, pa se varijabla konvertira u vrijednost *0*, zbraja sa *3* i dobija vrijednost *3*. Zbrajanje sa drugim argumentom je zbrajanje dva broja, pa *myVar* dobija vrijednost *7*. Treći argument je tipa *string*, što povlači konverziju *myVar* u *string*. Četvrti i peti argument su brojevi, ali se konvertuju u *string*. Tako *myVar* na kraju dobiva vrijednost “7to11”.
Funkcija može vratiti vrijednost korištenjem naredbe *return returnValue*, ovde *returnValue* prijedstavlja izraz čiju vrijednost funkcija vraća. Korištenjem same naredbe *return* (bez izraza) samo se napušta funkcija. Naredba *return* je izbor u funkciji, a može se pojavljivati na više mjesta u kodu funkcije (obično nakon ispunjenja nekog uvjeta), vraćajući različite vrijednosti, čak i različitog tipa. Primjer:

```
function returner (inputValue) {
     if (inputValue < 0) return;
     if (inputValue == 0) return false;
     if (inputValue == 1) return 0;
     if (inputValue == 2) return “”;
     if (inputValue == 3) return new Array ();
     if (inputValue == 4) return trace;
}
for (var i = -1; i < 6; i++) {
     trace (i + “: ” + typeof (returner (i)));
}
/*    output:
-1: undefined
0: boolean
1: number
2: string
3: object
4: function
5: undefined
*/
```

Ovdje je izbjegnuto duboko ugnježđivanje *if-else*, jer se iz funkcije izlazi po prvom ispunjenju nekog uvjeta.
U ActionScriptu postoje predefinirane funkcije, a mogu se definirati i nove. Često se funkcija pridružuje nekom objektu, tada ona postaje metoda objekta.

1.5. Objekti

Objekt je skup osobina koje mogu imati vrijednosti nekog tipa. Svaki objekt jedinstveno se identificira svojim imenom. Već spomenuti primjer:

```
myObject = {name:'Tara', age:27, city:'San Francisco'};
```

Objekt *myObject* ima tri osobine i svakoj osobini dodjeljena je vrijednost. Pristup osobini *age*, naprimjer, može se ostvariti sa *myObject.age* ili *myObject [“age”]*. Dodavanje

nove osobine moguće je: *myObject.state = 'California'*, kao i modifikacija postojeće osobine: *myObject.age = 28*.

```
function person (name, age, city) {
    this.name = name;
    this.age = age;
    this.city = city;
}
myObject = new person ('Tara', 27, 'San Francisco');
```

Ovaj kod kao krajnji rezultat daje također objekt *myObject* sa potpuno istim osobinama i vrijednostima osobina kao i prijethodni primjer. Razlika postoji. Sada je *myObject* instanca klase *person*, a funkcija *person* naziva se konstruktorom klase *person*. Objekt se instancira pozivanjem ključne reči *new* i konstruktora. Ključna riječ *this* u konstruktoru referencira na objekt koji se kreira. Sljedeći primjer dodaje metoda klasi person:

```
function showPerson () {
    var myString = this.name + ", " + this.age + ", " + this.city;
    trace (myString);
}
person.prototype.show = showPerson;
myObject.show (); //    "Tara, 27, San Francisco"
```

Sve funkcije posjeduju osobinu *prototype* koja se automatski kreira pri definiciji funkcije. Pri kreiranju novog objekta konstruktorom, sve osobine i metodae konstruktorske osobine *prototype* postaju osobine i metodae osobine *__proto__* novokreiranog objekta. Kada se referencira neka osobina ili metoda objekta, ActionScript traži postoji li takva osobina ili metodaa, ako ne postoji, traži se u okviru osobina i metodaa njegovog *__proto__* objekta (i dalje *__proto__.__proto__*, ...). Ovo je nasljeđivanje u ActionScriptu. Uobičajena praksa je dodeljivanje metoda osobini *prototype* konstruktora, kao što i stoji u navedenom primjeru.
ActionScript već dolazi sa dobrim skupom predefiniranih objekta.
Postoji skup ugrađenih Objekta prijeuzetih iz ECMA specifikacije:

- *Object* – najopča klasa objekta
- *Array* – niz, sa metodama manipulaciju nizom, dodavanje i uzimanje elementa niza sa početka ili kraja, sortiranje itd, osobina *length* je dužina niza...
- *String* – metode za manipulaciju stringovima, podniz, konkatenacija...
- *Boolean*
- *Number* – konstante, najmanja i najveća vrijednost, beskonačnost...
- *Math* – aritmetičke i trigonometrijske funkcije, matematičke konstante...
- *Date* – datum i vrijeme

Pored ovih, ugrađeni su i bitni objekti za Flash okruženje:

- *MovieClip* – osobine od značaja za vizualnu reprezentaciju objekta, širina, visina, horizontalna i vertikalna pozicija, vidljivost, providnost, rotacija...; metode za kontrolu osobina, za kretanje po vremenskoj osi, kreiranje i uništavanje instanci, prevlačenje objekta...
- *Selection* – kontrola tekstualnih polja
- *Mouse* – skrivanje i pokazivanje pokazivača miša

- *Key* – konstante specijalnih tipki tipkovnice, kodovi karaktera...
- *Color* – transformacije boje MovieClip objekta...
- *Sound* – kontrola zvuka i zvučnih efekata
- *XML* – učitavanje, slanje, parsiranje, izgradnja XML dokumenata...
- *XMLSocket* – stalna veza sa serverom, uspostavljanje i prijekid veze...

2. Programiranje u ActionScriptu

Programski elementi, akcije, u Flash filmu mogu se nalaziti na više mjesta. Kako se film odvija po vremenskoj osi podjeljenoj na sličice, djelovi koda mogu biti dodeljeni sličicama. Taj kod izvršava se kada se film nađe na određenom sličici. Pored toga, instance simbola tipa tipka također se mogu programirati za izvođenje određenih akcija pri pojavi nekog događaja, poput pritiska na tipku. Instance simbola tipa filmski klip također mogu reagirati na neke događaje ako im se pridruže potrebne akcije. Uz to, simbol tipa filmski klip je također film po svojoj definiciji, posjeduje svoju vremensku os, a može sadržavati i druge instance simbola tipa tipka ili filmski klip, pa se cjela priča može ponoviti u kontekstu definicije simbola. Ovo daje mogućnost velike zbrke i raštrkavanja koda na sve moguće strane, što s druge strane onemogućava korektno praćenje toka filma, nalaženje grešaka, i slično. Stoga je preporuka držanje koda na manje mjesta, pogotovo definicije varijabli, funkcija i objekta, recimo u prvoj sličici filma ili definicije simbola, dok se za obradu događaja može samo navesti poziv već definirane funkcije ili metodaa nekog objekta.

Događaji instanci simbola tipa tipka

Kod koji se pridružuje instanci simbola tipa tipka nalazi se u hendleru *on*:

```
on (mouseEvent) {
    //   statements
}
```

Za jednu instancu može se postaviti više hendlera, zavisno od događaja mouseEvent. Jedan hendler može opsluživati više događaja razdvojenih zarezom u zaglavlju hendlera. Kod pridružen hendleru izvršava se kada se desi neki od događaja navedenih u zaglavlju hendlera. Sljedi opis događaja koje prepoznaje objekt tipa button:

- *press* – tipka miša je pritisnuta dok je pokazivač iznad tipkata
- *release* – tipka miša je otpuštena dok je pokazivač iznad tipkata
- *releaseOutside* – tipka miša je otpuštena dok je pokazivač van tipkata
- *rollOver* – pokazivač miša kreće se iznad tipkata
- *rollOut* – pokazivač miša je napustio površinu tipkata
- *dragOver* – dok je pokazivač iznad tipkata, tipka miša je pritisnuta, pokazivač pomeren van površine tipkata i potom vraćen iznad
- *dragOut* – dok je pokazivač iznad tipkata, tipka miša je pritisnuto i potom pokazivač pomaknut van površine tipkata
- *keyPrijess ("key")* – tipka *key* je pritisnut

Događaji instance simbola tipa filmski klip

Instanca simbola tipa filmski klip Također reaguje na događaje.

```
onClipEvent (movieEvent) {
    //  statements
}
```

Ovdje je *movieEvent* neki od sledećih događaja:

- *load* – akcija se inicira instanciranjem objekta njegovom pojavom u vremenskoj osi
- *unload* – akcija se inicira u prvom sličici nakon uklanjanja objekta sa vremenske osi, a izvodi se prije bilo koje akcije vezane za promatranu sličicu
- *enterFrame* – akcija se inicira pri svakoj sličici, a izvodi se nakon svih akcija vezanih za posmatranu sličicu
- *mouseMove* – akcija se inicira pri svakom pomicanju miša
- *mouseDown* – akcija se inicira pritiskom na ljevo tipkalo miša
- *mouseUp* – akcija se inicira otpuštanjem ljevog tipkla miša
- *keyDown* – akcija se inicira pritiskom na neku tipku
- *keyUp* – akcija se inicira otpuštanjem tipki
- *data* – akcija se inicira prijemom podataka pri učitavanju

Zaključak

Vektorska grafika i animacija Flasha i jezik ActionScript svojom raznovrsnošću omogućavaju rješavanje širokog spektra web problematike. Od kreiranja navigacijskih rješenja u okviru HTML stranica, animiranih reklamnih sličica, do izrade kompletnih web-prezentacija u Flashu i direktnu razmenu podataka sa web serverom i njihovu dalju obradu.

Literatura

[1] Dejan Katašić “Flash i ActionScript” diplomski rad