Upotreba novih funkcionalnosti HTML5 pri razvoju web i mobilnih aplikacija

Igor Dujlović
Elektrotehnički fakultet Banja Luka
Banja Luka, BiH
dujlovic@gmail.com

Zoran Đurić
Elektrotehnički fakultet Banja Luka
Banja Luka, BiH
zoran.djuric@etfbl.net

Sadržaj – **U ovom radu dat je pregled novih funkcionalnosti koje nudi HTML5. Kako se HTML5 može koristiti za razvoj web i mobilnih aplikacija, u ovom radu su analizirani načini funkcionisanja i mogućnost upotrebe svih značajnih novih funkcionalnosti. Pored toga, u radu se ukazuje na potencijalna ograničenja koja se mogu pojaviti tokom razvoja i upotrebe ovih aplikacija.**

Ključne riječi: HTML5, web aplikacije, mobilne aplikacije

I. UVOD

HTML (*Hyper Text Markup Language*) je opisni jezik koji se koristi za izradu *web* stranica, pomoću koga se definiše izgled stranice, raspored komponenti i sadržaj koji se prezentuje. HTML5 je posljednja verzija ovog jezika koja treba da obezbijedi pojednostavljeno korištenje, bolju podršku multimedijalnom sadržaju, izbacivanje *plugin*-ova, upotrebu nezavisnu od platforme, bolju obradu grešaka, jednostavnije upravljanje kompleksnim *web* aplikacijama i upotrebu jednostavnih tagova umjesto komplikovanijih skripti. Kako bi se napravila kvalitetna *web* aplikacija poželjno je uz HTML koristiti i CSS, JavaScript, ali i neke dodatne biblioteke. HTML5 se može koristiti i za razvoj mobilnih aplikacija, u obliku aplikacija koje su prilagođene prikazu na mobilnim uređajima (mobilne web aplikacije) ili kao dio hibridnih mobilnih aplikacija. Trenutno nisu u potpunosti implementirane sve funkcionalnosti koje HTML5 nudi. U nastavku rada su analizirane nove HTML5 funkcionalnosti, pri čemu je naglasak na mogućnostima upotrebe i ograničenjima koji se pri tome mogu pojaviti.

II. PREGLED VAŽNIJIH FUNKCIONALNOSTI U HTML5

A. Audio element

Kako bi se izbjeglo korištenje dodatnih *plugin*-a za reprodukciju audio sadržaja, u HTML5 je uveden audio element. Audio element omogućava reprodukciju audio sadržaja. Ovaj element omogućava korišćenje dodatnih funkcija, kao što su kontrole za pokretanje i zaustavljanje reprodukcije audio sadržaja, promjenu jačine zvuka, reprodukciju zvuka pri učitavanju stranice, reprodukciju zvuka u petlji i mogućnost učitavanja sadržaja prije pokretanja [1]. Upotreba audio elementa vrši se na sljedeći način:

```
<audio controls="controls"
autoplay="autoplay" loop="loop">
 <source src="song.ogg" type="audio/ogg"
/>
 <source src="song.mp3" type="audio/mpeg"
/>
Vaš web čitač ne podržava audio element.
</audio>
```

Kao što se vidi iz prethodnog primjera, u jednom audio elementu moguće je navesti nekoliko istih audio sadržaja, koji su u različitim formatima, a za ovu namjenu koristi se *source* tag. To je zbog toga što *web* čitači podržavaju različite kodeke (eng. *codec*) i formate. Osim adrese audio fajla, u ovom tagu potrebno je navesti i vrstu audio sadržaja. Kada *web* čitač treba da reprodukuje audio sadržaj, iz liste navedenih izvora će automatski izabrati izvor za koji posjeduje odgovarajući kodek. Ukoliko *web* čitač ne podržava niti jedan navedeni izvor, ispisaće se poruka. Kako bi se omogućila maksimalna podrška za različite *web* čitače, audio sadržaj je neophodno enkodovati u različite formate (npr. mp3, wav i ogg), koji su trenutno podržani u većini *web* čitača. Osim razlika u vrsti audio fajla, postoje razlike i u podržanosti atributa u zavisnosti od *web* čitača koji se koristi, što predstavlja dodatni problem pri upotrebi ovog elementa. Pristup audio elementu se može ostvariti pomoću JavaScripta, pri čemu se za upravljanje sadržajem koristi audio API, koji omogućava programsku upotrebu događaja ili kontrola audio sadržaja. Na istoj stranici moguća je upotreba više audio elemenata.

B. Video element

Video element se koristi kako bi se izbjegla upotreba dodatnih *plugin*-a za reprodukciju video sadržaja kao što je Flash. Ovaj element omogućava i bolju integraciju sa ostalim elementima na stranici u odnosu na dodatne *plugin*-e što omogućava jednostavniju manipulaciju sadržajem. Upotreba video elementa vrši se na sljedeći način:

```
<video width="320" height="240" controls>
<source src="movie.mp4" type="video/mp4">
<source src="movie.ogg" type="video/ogg">
<source src="movie.webm"
```

```
type="video/webm">
<object data="movie.mp4" width="320"
height="240">
<embed src="movie.swf" width="320"
height="240">
</object>
</video>
```

Uz video element je moguće koristiti i kontrole za upravljanje sadržajem, pokretanje video sadržaja pri učitavanju stranice, kontinuiranu reprodukcija sadržaja, definisanje dimenzija video elementa, korištenje slike koja se prikazuje prije pokretanja sadržaja, kao i mogućnost učitavanja sadržaja prije pokretanja. Međutim, kao i kod audio elementa postoje razlike u podržanim formatima video sadržaja u zavisnosti od korištenog *web* čitača. Zbog toga je isti video sadržaj poželjno enkodovati u različitim formatima (npr. mp4, ogg/ogv i webm), kako bi se ostvarila potpuna podržanost za različite *web* čitače. Ukoliko nije moguće reprodukovati video sadržaj biće ispisana odgovarajuća poruka. Upravljanje video sadržajem moguće je pristupom video elementu iz DOM (*Document Object Model*) strukture, pri čemu se pomoću odgovarajućih metoda mijenjaju osobine video elementa ili se omogućava pokretanje/ zaustavljanje video sadržaja [2].

C. Canvas element

Canvas element omogućava iscrtavanje grafičkih elemenata pomoću JavaScript-a. Ovaj element definiše kontejner u kome se vrši iscrtavanje linija, geometrijskih figura, simbola, a na njega se mogu dodavati i slike. Canvas element nema vlastiti sadržaj, a od atributa posjeduje samo dimenzije. Podržava korištenje događaja i HTML5 globalne atribute. Prostor za prikaz definiše se na sljedeći način:

```
<canvas id="myCanvas" width="200"
height="100">
Poruka u slučaju da element nije podržan.
</canvas>
```

Popunjavanje canvas prostora se vrši na sljedeći način:

```
<script type="text/javascript">
var c =
      document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,150,75);
</script>
```

Canvas elementu se pristupa pomoću JavaScript-a, a sadržaj se dodjeljuje *drawing* kontekstu. Trenutno postoji podrška samo za dvodimenzionalni *drawing* kontekst [3].

D. Embed element

Embed element omogućava upotrebu ugrađenih sadržaja kao što su *plugin*-i. Podržava HTML5 globalne i *event* atribute.

E. Upotreba media elemenata

Osim osnovnih upotreba, media elementi se mogu kombinovati sa drugim elementima. Tako se, na primjer, kombinacijom video i canvas elemenata, iz video snimka mogu dobiti slike (*frame*-ovi), koje se mogu dalje upotrebiti. Uz media elemente moguće je koristiti i CSS kako bi se dobili različiti vizuelni efekti. Canvas elementi se mogu koristiti i za izadu grafičkih komponenata za igrice, na sličan način kako se realizuju Flash igrice.

F. Drag and drop

Drag and drop funkcionalnost se odnosti na mogućnost da korisnik prebaci određeni element sa jednog na drugo mjesto na stranici [4]. Kako bi ovo bilo moguće neophodno je element kome treba biti dozvoljen prenos postaviti atribut *draggable* na *true*.

Za obavljanje prenosa, bitna su tri događaja koji se definišu na elementima. Na elementu koji je moguće prenijeti definiše se *ondragstart* metoda u kojoj se pomoću *dataTransfer.setData()* postavlja vrsta podatka i vrijednost koja se prenosi. Na elementu gdje se dozvoljava prenos potrebno je definisati *ondragover* događaj u kome se pomoću *event.preventDefault()* metode sprečava podrazumijevano ponašanje u slučaju *drop* akcije. Na istom elementu definiše se i *ondrop* događaj kojim se omogućava ubacivanje prenesenog elementa u dati element.

Problemi koji se sreću pri upotrebi *drag and drop* funkcionalnosti se odnose na definisanje većeg broja događaja i ograničenja u dimenzijama prozora, što otežava premještanje elemenata na željenu lokaciju. Prilikom upotrebe ove funkcionalnosti na mobilnim uređajima javlja se problem neprepoznavanja svih potrebnih događaja prilikom dodira ekrana, pa je veoma teško koristiti ovu funkcionalnost.

G. Keširanje aplikacija

Keširanje aplikacija omogućava da aplikacija bude dostupna i bez pristupa Internetu. To podrazumijeva da se sadržaj koji je označen za keširanje čuva na uređaju, i pri sljedećem učitavanju stranice prikazuje bez ponovnog skidanja sa Interneta [5]. Prednosti keširanja su:

- korisnici mogu gledati sadržaj kada su *offline*
- keširane stranice se brže učitavaju
- smanjeno opterećenje servera
- smanjen mrežni protok

Problem koji se javlja kod keširanja je taj da se jednom keširana stranica neće ponovo preuzimati sa Interneta čak i kada doživi promjenu, već će se koristiti lokalna verzija. Ovaj problem se može prevazići ručnim ili automatskim brisanjem keša u *web* čitaču, ili promjenom sadržaja *manifest* fajla. Moguće ograničenje je vezano za maksimalnu veličinu keša u

web čitačima, tako da neki *web* čitači podržavaju keš veličine do 5 MB.

Kako bi se omogućilo keširanje u HTML5 dokumentu neophodno je dodati sljedeći meta tag kojim se uključuje manifest fajl, sa preporučenom ekstenzijom . *appcache*.

```
<html manifest="test.appcache">
```

Manifest fajl je tekstualni dokument u kome se nalaze podaci o tome šta treba da bude keširano. Sastoji se iz tri dijela:

- *CACHE MANIFEST* – fajlovi navedeni u ovom dijelu se keširaju prilikom prve posjete stranici
- *NETWORK* – fajlovi navedeni u ovom dijelu se ne keširaju i zahtjevaju mrežnu konekciju
- *FALLBACK* – fajlovi u ovom dijelu određuju rezervne fajlove u slučaju da su nedostupni fajlovi sa mreže

Primjer manifest fajla:

```
CACHE MANIFEST
/theme.css
/logo.gif
/main.js

NETWORK:
login.asp

FALLBACK:
/html/ /offline.html
```

H. Forme

U HTML5 su uvedene dodatne vrste polja za unos podataka na formi, kao što su datum, vrijeme, telefonski broj, url adresa, email adresa, vrijednost iz definisanog opsega, brojevi, izbor boje i druge vrste podataka. Većina novih vrsta polja za unos još nisu podržana u svim *web* čitačima, ali se mogu koristiti kao obična polja za unos teksta.

Kako bi se poboljšala funkcionalnost formi uvedeni su sljedeći elementi: *datalist*, *keygen*, *output* [6]. *Datalist* element omogućava definisanje vrijednosti koje je moguće unijeti u polje za unos teksta, i na taj način se dobija efekat automatskog popunjavanja, a mora se povezati sa poljem za unos.

```
<input list="browsers" />
<datalist id="browsers">
        <option value="Firefox">
        <option value="Chrome">
</datalist>
```

Keygen element omogućava kreiranje privatnog i javnog ključa, i na taj način povećava sigurnost pri prenosu podataka. Privatni ključ se čuva lokalno, a javni se šalje serveru i može se iskoristiti za generisanje digitalnog sertifikata.

Output element se koristi za ispis vrijednosti dobijene obradom pomoću JavaScripta. Obrada se definiše na samoj formi kroz odgovarajući događaj, u kome se postavlja vrijednost *output* polja, a sam *output* element ima atribute koji definišu naziv i vezu sa elementima u formi, čije se vrijednosti obrađuju.

Osim novih elemenata i vrsta polja za rad sa formama uvedeni su i novi atributi, koji se koriste uz *form* i *input* elemente. Atributi, koji se mogu veoma često koristiti u *web*, ali i mobilnim HTML5 aplikacijama su vezani za unos sadržaja poput telefonskog broja, e-mail adrese, url adrese, datuma, vremena, polja za pretragu, broja, intervala brojeva, boje i mnogih drugih atributa [7]. Na ovaj način se olakšava unos podataka, a postoji i ugrađena validacija. Mogu se definisati obavezna polja, kao i polja koja nije potrebno validirati. Zbog činjenice da ni svi elementi i atributi formi nisu podržani u različitim *web* čitačima, moguća su određena ograničenja u primjeni.

I. Čuvanje podataka - Web storage

Čuvanje podataka omogućava da se kod klijenta čuvaju veće količine podataka koje ne moraju biti uključene u zahtjeve prema serveru u toku komunikacije. Ovi podaci se čuvaju nezavisno od kolačića (*cookies*) i ne utiču na performanse web aplikacije, a čuvaju se u obliku ključ-vrijednost.

Postoje dvije vrste objekata za čuvanje podataka [8]:

- *localStorage* – trajno čuvaju podatke
- *sessionStorage* – čuvaju podatke tokom jedne sesije

Prije korištenja ovih objekata neophodno je provjeriti da li su podržani od strane *web* čitača na sljedeći način:

```
if(typeof(Storage)!=="undefined") {
      localStorage.age="25" ;
      sessionStorage.age="25";
}
```

Čitanje podataka se vrši direktnim pristupom (za prethodni primjer *localStorage.age*). Bitno je naglasiti da se u ovim objektima čuvaju stringovi, tako da je prije upotrebe potrebno pretvoriti podatke u odgovarajući format. Za čuvanje nizova objekata poželjno je koristiti JavaScript metode za rad sa JSON objektima: *JSON.stringify* – metoda koja niz serijalizuje u string, i *JSON.parse* – metoda koja string deserijalizuje u niz.

Web čitači bi trebali ograničiti prostor za smještanje podataka (preporučeno 5 MB). Osim toga potrebno je spriječiti da maliciozni sajtovi zaobilaze prethodno ograničenje tako što će imati veći broj poddomena. Kada je prostor za određen sajt popunjen, korisnici ga mogu eksplicitno proširiti. Isto tako, korisnici treba da imaju uvid u trenutno zauzeće prostora za svaki od domena.

Prednosti ovog načina čuvanja podataka su podrška na skoro svim *web* čitačima i jednostavna upotreba. Nedostaci se ogledaju u lošim performansama pri čuvanju velikih i kompleksnih podataka zbog toga što se koristi sinhroni pristup. Zbog nedostatka indeksiranja, performanse su loše i pri

pretraživanju. Dodatni nedostatak je potreba da se osigura konzistentnost podataka, jer se podaci ne smještaju strukturno.

J. Lokalne baze podataka

HTML5 je trebao da omogući korištenje web SQL baza podataka (SQLite). Web SQL baze podataka omogućavaju upotrebu SQL-a. Ova vrsta baza podataka nije implementirana u potpunosti i u budućnosti neće biti dio HTML specifikacije [9]. Upotreba sličnog načina čuvanja podataka dostupna je u PhoneGap programskom okruženju, koji se koristi za razvoj hibridnih mobilnih aplikacija.

Indeksirane baze podataka su mješavina SQL baza podataka i objekata za čuvanje podataka zbog toga što se podaci čuvaju u obliku ključ – vrijednost ali je pretraga mnogo brža zbog toga što se podaci indeksiraju [10].

Prednosti ovog načina čuvanja podataka su dobre performanse zbog asinhronog pristupa, dobre performanse prilikom pretraživanja zbog indeksiranja prema ključu koji se koristi za pretragu i podrška izvršavanju transakcija. Nedostaci se ogledaju u komplikovanijem API-ju, a potrebno je osigurati konzistentnost i integritet podataka. Trenutno veoma mali broj mobilnih *web* čitača podržava indeksirane baze podataka.

K. Fajl sistem

Za smještanje kompleksnijih i većih podataka nije pogodno koristiti prehodno opisane načine smještanja podataka. Umjesto toga, moguće je pristupiti fajl sistemu klijenta i izvršiti pohranjivanje podataka.

Prednosti ovog načina čuvanja podataka su dobre performanse zbog asinhronog pristupa i mogućnost čuvanja većih količina podataka i velikih fajlova. Nedostaci su vezani za ograničenu dostupnost (samo Chrome), nedostatak podrške za izvršavanje transakcija, nedostatak ugrađene podrške za indeksiranje i pretraživanje i potencijalne sigurnosne probleme.

L. Web workers

Web workers (radnici) predstavljaju mehanizam koji omogućava izvršavanje JavaScript koda u pozadini aplikacije, bez uticaja na performanse [11]. Na ovaj način sprečava se zaglavljivanje aplikacije, koje je nastajalo u slučaju da JavaScript funkcija postane neaktivna. Dok se izvršava funkcija na web *worker*-u na ostatku stranice se mogu istovremeno izvršavati ostale funkcije.

Da bi se pokrenuo *web workers* mehanizam neophodno je ispitati da li *web* čitač podržava ovu funkcionalnost. Funkcionalnost koju *web worker* treba da izvrši potrebno je izdvojiti u poseban .js fajl, u kome se pozivom metode *postMessage()* vraća vrijednost koja treba da se prikaže u HTML fajlu.

Kreiranje objekta vrši se na način da se konstruktoru prosljeđuje naziv fajla u kome su definisane funkcionalnosti koje se izvode.

```
if(typeof(w)=="undefined"){
        w=new Worker("demo_workers.js");
}
```

Preuzimanje vrijednosti koja se dobija od *web worker*-a vrši se na sljedeći način:

```
w.onmessage=function(event){
        document.getElementById("result").
innerHTML=event.data;
};
```

Web workers mehanizam je sličan nitima (*thread*) koje se koriste u drugim programskim jezicima. Koordinacija i komunikacija se obavlja pomoću poruka. Zbog karaktera niti, *web workers* mehanizam može koristiti *navigator* i *location* objekte, keš aplikacija, *XMLHttpRequest* objekat, JavaScript funkcije za rad sa intervalima i tajmerima (*timeout*), učitavanje eksternih skripti i komunikaciju i saradnju sa drugim *web worker*-ima. Upotreba *web worker*-a posjeduje i određena ograničenja, tako da ih nije moguće koristiti za promjenu DOM strukture i upotrebu *window*, *document* i *parent* objekata.

Web worker-i se mogu koristiti za sljedeće namjene: upravljanjem keširanim podacima, formatiranje teksta u realnom vremenu i korekciju napisanog sadržaja, analizu video i audio sadržaja, pozadinski rad sa *web* servisima, procesiranje velikih nizova ili JSON podataka, iscrtavanje *canvas* sadržaja i rad sa lokalnim bazama podataka. Trenutno postoji podrška u većini *desktop web* čitača, dok na određenim mobilnim *web* čitačima ovaj mehanizam još uvijek nije podržan.

Postoje i određena sigurnosna ograničenja prilikom upotrebe ovog mehanizma. *Web worker* mora biti definisan u eksternoj datoteci koja se dobija sa iste vrste izvora.

M. Geolokacija

Geolokacija se koristi za dobijanje informacija o trenutnoj lokaciji korisnika [12]. Da bi se ovi podaci dobili neophodna je dozvola kako se ne bi povrijedila korisnikova privatnost. Primjena geolokacije je posebno značajna kod mobilnih klijenata jer se na taj način mogu realizovati opcije navigacije pomoću GPS-a (*Global Positioning System*), utvrđivanja trenutne lokacije, dobijanja lokacija objekata u neposrednoj blizini. Dobijene koordinate koje predstavljaju trenutnu lokaciju korisnika mogu se prikazati na mapi.

Informacije o lokaciji određuju se na osnovu GPS, IP adresa, RFID, WiFi, *BlueTooth* Mac adresa, ID – eva GSM/CDMA ćelija ili korisničkog unosa. Ne postoji garancija da će ovi podaci dati stvarnu lokaciju uređaja.

Potencijalni problemi koji se mogu javiti zbog upotrebe geolokacije su vezani za privatnost korisnika. Web sajt ili mobilna aplikacija mogu zahtijevati informacije o lokaciji korisnika jedino u slučaju kada su te informacije neophodne, pri čemu korisnik mora da odobri davanje infomacija. Informacije o lokaciji mogu se koristiti jedino u svrhu za koju se traže, a nakon upotrebe se ne smiju čuvati bez eksplicitne dozvole korisnika. Ukoliko se informacije čuvaju, onda ih je neophodno zaštititi od neovlaštenog korištenja, a korisniku se moraju dati informacije o razlogu čuvanja, vremenskom trajanju, načinu osiguranja kao i mogućnosti izmjene ili korištenja sačuvanih informacija. Kada se za dobijanje lokacije

koriste WiFi podaci, tada se šalju informacije o svim dostupnim AP - ima (*Access Point*), što može dodatno naruštiti privatnost korisnika za koga se utvrđuje lokacija, a posebno korisnika-vlasnika AP-ova koji se koriste za utvrđivanje lokacije.

Korištenje geolokacije se obično obavlja na sljedeći način:

```
var x=document.getElementById("demo");

function getLocation() {
 if (navigator.geolocation) {
            navigator.geolocation.
getCurrentPosition(showPosition);
       }
      else{
x.innerHTML="Geolocation nije podrzana.";
}
}

function showPosition(position) {
       x.innerHTML= "Latitude: " +
position.coords.latitude +
        "<br />Longitude: " +
 position.coords.longitude;
}
```

Na prikazanom primjeru se vidi da funkcija *getLocation()* provjerava da li *web* čitač podržava geolokaciju, a zatim, ako je geolokacija podržana, prikuplja podatke o trenutnoj lokaciji pomoću metode *getCurrentPosition()* kojoj se prosljeđuje funkcija za prikaz podataka, i opcionalno funkcija za obradu greške. Greške koje se mogu pojaviti su: *PERMISSION_DENIED, POSITION_UNAVAILABLE, UNKNOWN_ERROR i TIMEOUT* i one se dobijaju iz *error.code* objekta koji se prima kao parametar funkcije za obradu greške.

Osim navedenih osobina bitno je spomenuti i metode *watchPosition()* koja konstantno daje informacije o položaju korisnika koji je u pokretu, a može poslužiti za navigaciju, i *clearWatch()* metodu koja zaustavlja *watchPosition()* metodu.

Osim W3C implementacije, postoji i *geo.js* implementacija, koja omogućava korištenje geolokacijskih usluga na uređajima gdje nije podržana W3C implementacija.

N. Server-Sent Events

Server-Sent Events (SSE) predstavljaju mogućnost dobijanja sadržaja sa servera tokom rada aplikacije. Ova mogućnost se razlikuje od prethodnih po tome što podaci mogu biti dobijeni sa servera nakon što je stranica učitana, odnosno aplikacija može slušati i obrađivati poruke sa servera u bilo koje vrijeme, što se koristi za komunikaciju u realnom vremenu ili dugotrajni prenos manjih količina vremenski zavisnih podataka. SSE razmjenjuje podatke pomoću HTTP protokola, a vrsta podataka koja se šalje sa servera je *text/event-stream* [13].

Kako bi se implementirala ova mogućnost neophodno je provjeriti da li *web* čitač podržava SSE. Nakon toga kreira se izvor, i njemu se dodjeljuje *handler*. Podacima, dobijenim sa servera pristupa se pomoću *event.data*.

```
var source=new EventSource("demo.php");
source.onmessage=function(event)
 {

document.getElementById("result").
innerHTML+=event.data + "<br />";
 };
```

Događaji koji su podržani sa SSE su: *onopen* - kada je veza prema serveru otvorena, *onmessage* - kada je poruka primljena i *onerror* - kada dođe do greške. Serverski dio, koji predstavlja izvor podataka i čija adresa se prosljeđuje prilikom kreiranja objekta se može realizovati u proizvoljnom programskom jeziku poput PHP-a ili ASP-a.

O. Web sockets

Web Socket omogućava razmjenu podataka između klijenta i servera u oba pravca, u bilo kom trenutku. Podržan je *cross origin communication*, što znači da se komunikacija može uspostaviti između hostova koji su na različitim domenima. Pomoću *web socket*-a moguće je razmjenjivati tekstualni i binarni sadržaj [14].

Web Socket funkcioniše na način da se između dva hosta uspostavi veza kreiranjem *WebSocket* objekta. Konstruktor *WebSocket* objekta prihvata dva parametra. Prvi parametar koji se prosljeđuje je URL *Web Socketa* sa prefiksom ws za običnu komunikaciju, i wss za sigurnu komunikaciju. Drugi parametar predstavlja skup podprotokola koji se koriste u komunikaciji. Komunikacija između hostova se vrši upotrebom funkcija *send* za slanje sadržaja, i *onmessage* događaja kojim se definiše funkcija za obradu primljene poruke. U nastavku je prikazan primjer upotrebe *web sockets* mehanizma.

```
<script type='text/javascript'>
var ws = new WebSocket
('ws://localhost:8876/Channel',
'mySubprotocol.example.org');
ws.onmessage = function (message) {
      var messages =
      document.getElementById('messages');
      messages.innerHTML += "<br>[in] " +
      message.data;
      };

sendmsg = function() {
      var message = document.
      getElementById('message_to_send')
      .value;
      document.getElementById
('message_to_send').value = '';
```

```
      ws.send(message);
      var messages = document.
getElementById('messages');
      messages.innerHTML += "<br>[out] " +
message;
};
     </script>
```

Primjena *web socket*-a je u aplikacijama koje treba da osiguraju komunikaciju u realnom vremenu između različitih izvora, kao što su *chat* aplikacije, razmjena podataka na društvenim mrežama, praćenje podataka koji se često mijenjaju, igre i druge slične aplikacije.

P. Razlike između SSE i Web Socket-a

Osnovna razlika između SSE i *Web Socket*-a je u tome što SSE omogućava jednosmjernu komunikaciju od strane servera prema klijentu a *Web Socket* omogućava dvosmjernu komunikaciju, što omogućava češću primjenu. *Web Socket* koristi iste portove kao i HTTP konekcije. Za razliku od SSE, *Web Socket* ne omogućava sigurnu isporuku poruka, i ne omogućava ponovno uspostavljanje konekcija, pa je manje pouzdan. SSE omogućava mehanizam rekonekcije i sinhronizacije poruka.

III. ZAKLJUČAK

Nove funkcionalnosti koje su uvedene u HTML5 omogućavaju razvoj web i mobilnih aplikacija sa velikim mogućnostima u pogledu kvaliteta korisničkog interfejsa. Osim toga, HTML5 nudi brojne korisne funkcionalnosti koje zamjenjuju dodatne *plugin*-e i biblioteke. Korištenjem funkcionalnosti za čuvanje podataka ili *offline* korištenje aplikacije, mogu se djelimično ili u potpunosti zadovoljiti potrebe za smještanjem različitih podataka. Takva aplikacija se može približiti *native* aplikacijama, kada su u pitanju mobilne aplikacije. Kada su u pitanju web aplikacije, zbog činjenice da se određeni podaci čuvaju kod klijenta, dobijamo aplikaciju koja podsjeća na *desktop* aplikaciju. Trenutno postoje rješenja i za izradu HTML5 *desktop* aplikacija. Korištenjem komunikacionih mehanizama može se smanjiti potreba za upotrebom drugog programskog jezika, jer je razmjenu podataka između aplikacije i drugog hosta ili servera moguće obaviti pomoću ugrađenih mehanizama. Kada su u pitanju performanse HTML5 aplikacija, značajno poboljšanje se može postići upotrebom mehanizma koji podsjeća na niti u klasičnim aplikacijama.

Ipak, najvažniji nedostak HTML5 se ogleda u činjenici da određene funkcionalnosti nisu podržane ili se ne mogu koristiti na isti način u svim *web* čitačima. Svaka funkcionalnost ima određene nedostatke o kojima je bilo riječi u okviru rada.

Na kraju, bitno je naglasiti da je HTML5 još uvijek u fazi razvoja i da bi konačna specifikacija trebala biti definisana krajem 2014. godine pa bi do tada svi uočeni nedostaci trebali biti otklonjeni. HTML5 se sada veoma mnogo koristi, i sve više aplikacija, koje su realizovane u prethodnim verzijama HTML-a se unaprijeđuju na poslijednju verziju. Ukoliko se HTML bude dodatno unaprijeđivao, u budućnosti bi mogao postati dominantna tehnologija za izradu klijentskih aplikacija.

LITERATURA

[1] http://html5doctor.com/html5-audio-the-state-of-play/, posjećivano decembra 2012.

[2] http://www.html5rocks.com/en/tutorials/video/basics/, posjećivano decembra 2012.

[3] http://diveintohtml5.info/canvas.html#divingin, posjećivano decembra 2012.

[4] http://www.html5rocks.com/en/tutorials/dnd/basics/, posjećivano decembra 2012.

[5] http://diveintohtml5.info/offline.html, posjećivano decembra 2012.

[6] http://www.w3schools.com/html/html5_form_elements.asp, posjećivano decembra 2012.

[7] http://diveintohtml5.info/forms.html, posjećivano decembra 2012.

[8] http://diveintohtml5.info/storage.html, posjećivano decembra 2012.

[9] http://www.w3.org/TR/webdatabase/, posjećivano decembra 2012.

[10] http://www.w3.org/TR/IndexedDB/, posjećivano decembra 2012.

[11] http://blogs.msdn.com/b/davrous/archive/2011/07/15/introduction-to-the-html5-web-workers-the-javascript-multithreading-approach.aspx, posjećivano decembra 2012.

[12] http://diveintohtml5.info/geolocation.html, posjećivano decembra 2012.

[13] http://html5doctor.com/server-sent-events/, posjećivano decembra 2012.

[14] http://today.java.net/article/2010/04/26/html5-server-push-technologies-part-2, posjećivano decembra 2012.

ABSTRACT

This paper presents an overview of new HTML5 features. As HTML5 can be used for development of web and mobile applications, we analyzed the most important HTML5 features. Also, some potential usage limits of these features were discussed.

USAGE OF NEW HTML5 FEATURES FOR WEB AND MOBILE APPLICATIONS DEVELOPMENT

Igor Dujlović, Zoran Đurić