Tallinna Ülikool
Informaatika Instituut

YouTube JavaScript playeri API kasutusjuhend

Seminaritöö

Autor: Einari Veldre
Juhendaja: Jaagup Kippar

Tallinn 2014

Autorideklaratsioon

Deklareerin, et käesolev seminaritöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.....	.....
(kuupäev)	(autor)

SISUKORD

SISSEJUHATUS

Käesoleva seminaritöö teemaks on YouTube JavaScript playeri API kasutusjuhend. YouTube API võimaldab mängida YouTube videosid meile sobival veebirakendusel, tarkvararakendusel või muul seadmel.

YouTube'i keskkonda külastab igakuiselt enam kui 1 miljard unikaalset kasutajat (YouTube, 2014). Kuna antud video keskkonna puhul on tegemist erakordselt populaarse teenusega, pean teema valikut piisavalt aktuaalseks, et teha selle kohta seminaritöö. Antud töö eesmärk on luua eestikeelne kasutusjuhend YouTube JavaScript playeri API'ile ja testida selle tõhusust informaatika instituudi õpilaste peal.

YouTube playeri API koosneb neljast erinevast osast: Android API, IFrame API, JavaScript API ja Flash API. Käesoleva API valiku tingis asjaolu, et töö autor puutub igapäevaselt kokku *JavaScript* programmeerimiskeelega, samuti on antud keel kasutusel üle 80% veebilehtedel (W3techs, 2014). Videode esitamiseks *JavaScript API*'s kasutatakse *Flash player*'it. Antud valik võib esialgu tunduda kummaline, kuna *YouTube* ise on viimased aastad lõpetanud oma *Flash player*'i arenduse. Samas selleks, et tagada ka videode esitlemise võimalus vanemates interneti brauserites, mis ei toeta *HTML5* funktsionaalsus ja enim levinud operatsioonisüsteemides ei jää muud üle.

Käesolev töö on jagatud kolmeks peatükiks. Esimeses peatükis räägitakse olemasolevatest sama teemalisest materjalidest. Teine peatükk, mis kujutab endas antud töö põhiosa, kasutusjuhendit, on jagatud omakorda seitsmeks alapeatükiks. Antud peatükis käiakse läbi ükshaaval API funktsionaalsus koos selgitava teksti ja koodinäidetega. Kolmandas peatükis võetakse kokku kasutajate tagasiside, kes lahendasid lisas olevat kolme ülesannet, kasutades antud seminaritööd.

1 OLEMASOLEVAD KASUTUSJUHEDID

Antud seminaritöö koostamise käigus sai otsitud internetist (*Google*'i otsingumootorist) olemasolevaid samateemalisi õpetusi. Tulemuseks on hulgaliselt inglisekeelseid lühiõpetusi ja koodinäiteid, mis kas hõlmavad endas ainult teatud osa *API*'st (nt üksikvideo või videonimekirja laadimist) ja teised, mis ei anna tervikliku ülevaadet antud rakendusest.

Järgnevates alapeatükides tuuakse välja ja antakse lühiülevaade kahest põhjalikumast *YouTube player*'i *JavaScript API*'it kirjeldavast materjalist, mida leiab internetist. Esimeses alapeatükis kirjeldatav õpetus keskendub *API* üksikvideole rakendatavatele võimalustele ning teises alapeatükis keskendutakse video nimekirja haldamisele.

1.1 *Õ*petus *Webmonkey* keskkonnas

Webmonkey näol on tegemist interneti keskkonnaga kuhu on üles pandud erinevaid artikleid ja õpetusi veebiarenduse valdkonnast.

Artikkel antud keskkonnas on Webmonkey Staff nimelise kasutaja poolt, kelle puhul on selge, et pole tegemist päris nimega, vaid kasutajaga, kes kuulub *Webmonkey* personali. (Webmonkey, 2010)

Õpetus koosneb neljast peatükist, millest esimene on pealkirjaga "Getting started" (alustuseks), milles mainitakse mõne lausega ära, et videode esitamiseks kasutatakse *Flash player*'it ning see laetakse dokumenti kasutades *SWFObject* nimelist *JavaScript* teeki. Järgmises peatükis kirjeldatakse seda kuidas *SWFObject*'i kasutades *YouTube* video dokumenti laetakse. Kolmandas peatükis kirjeldatakse video esitluse kolme põhi operatsiooni(mängi, peata ja lõpeta) ning kuidas laadida *player*'isse uus video. Viimaseks peatükiks on kokkuvõte, kus võetakse lühidalt kokku antud õpetus ning viidatakse ka *ActionScript API* olemasolule.

Õpetusega *Webmonkey* keskkonnas näol on tegemist pinnapealse materjaliga, kus piirdutakse üksikvideo mängimisele ning kirjeldatakse ainult kolme video esitluse operatsiooni, mida

tegelikult kokku on viis. Lisaks eelnevale pole antud õpetuses mainitud midagi *API* sündmustest ja puudub ka kirjeldus aadressi parameetritest, kust laetakse *YouTube player*.

1.2 Õpetus *IBM developerWorks* keskkonnas

IBM developerWorks on veebipõhine tehniliste vahendite allikas ja võrgustik eriala professionaalidele IT valdkonnas tegelevatele inimestele. (IBM developerWorks, 2014)

Õpetus käesolevas keskkonnas kirjeldab kuidas luua veebirakendus, mis esitaks *YouTube* video nimekirja. Artikli loomise ajendiks oli *API player*'i erinevate võimaluste puudumine *YouTube* keskkonna sisese *player*'i ees. *IBM developerWorks* keskkonnas oleva materjalis näidatakse kuidas *API player*'ile luua samad võimalused, mis on *YouTube* keskkonna sisesel *player*'il. (McCarthy, 2014)

Nendeks puudujääkideks on:

- video nimekiri on vaikimisi peidus
- puudub videode juhuesituse võimalus
- autori poolt lisatud märkused on eemaldatud

Artiklis loodud rakenduse tegemiseks kasutatakse *YouTube API*'it ning *jQuery*, *JsReader* ja *Boostrap* nimelisi teeke. Antud õpetuses ei kasutata *API* erinevaid operatsioone mängiva video esitluse haldamiseks (mängi, peata, lõpeta jne). Selleks, et videosid mängida suvalises järjestuses, kontrollib McCarthy poolt loodud rakendus nimekirjas olevate videode esitamist ise (tavaliselt teeb seda *YouTube player*), kasutades selleks *API* video laadimise meetodit ning *player*'i staatuse muutumise sündmust.

Antud õpetuse üheks põhiliseks puudujäägiks on *YouTube JavaScript player*'i *API* funktsionaalsuse vähene kasutamine. Näiteks videode laadimiseks oleks võinud kasutada *loadVideoByID* asemel *playVideoAt* meetodit, mis oleks vähem ressurssinõudlikum olnud. Samuti juhuesituse loomiseks ei oleks pidanud uut *JavaScript* loogikat looma, kuna selle jaoks on juba *API*'s *setShuffle* nimeline meetod olemas, mida saab rakendada video nimekirjades.

2 JUHEND YOUTUBE JAVASCRIPT PLAYERI API KASUTAMISEKS

Käesolev juhend on mõeldud ülevaate andmiseks ja põhiteadmiste omandamiseks *YouTube JavaScript player*'i *API*'st, mille sihtgrupiks on informaatika või mingi muu IT eriala tudengid. Juhendi sisus on seatud eesmärgiks läbi vaadata kõik *API* meetodid, operatsioonid ning sündmused üksikvideo ja video nimekirja osas. Eelduseks antud töö arusaadavaks läbi töötamiseks ja lisas olevate ülesannete lahendamiseks on vajalik, et juhendi kasutajal oleksid algteadmised *HTML* ja *JavaScript* keelest.

Vaja läheb serveriruumi, kuhu saame failid üleslaadida, veebilehitsejat, kus oleks *Flash player* (versioon 10.1 või uuem) ja *JavaScript*'i kasutusvõimalus ning tekstiredaktorit, kus saaks koodi kirjutada. Antud juhendi koostamisel valiti veebilehitsejaks *Google Chrome*'i ja tekstiredaktoriks *Notepad++* nimeline programm.

2.1 HTML lehe loomine

Järgnevas peatükis luuakse *HTML* dokument vajalike lisadega, mida hakatakse edaspidi kasutusjuhendi käigus koodiga täiendama.

Kui arvutisse on vajalikud rakendused paigaldatud, tuleb järgmisena luua *HTML* fail, kuhu hakatakse kirjutama koodi vastavalt YouTube *API* dokumentatsioonile. Eelnevalt loodud dokumenti lisame järgneva koodi (vt koodinäide 1).

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Youtube JavaScript player API test</title>
  <style>
    a {display:block;}
  </style>
</head>
<body>
</body>
</html>
```

Koodinäide 1. *HTML* leht

Järgmisena lisame *HTML* dokumenti *SWFObject* nimelise JavaScripti teegi, mis lihtsustab flash objektide kasutamist veebilehtedel (YouTube *API* soovitab selle kasutamist). Järgnev kood lisada <head> elemendi sisse. (vt koodinäide 2)

```
<script
src="//ajax.googleapis.com/ajax/libs/swfobject/2.2/swfobject.js"></scrip
t>
```

Koodinäide 2. SWFObject teegi laadimine

Järgmisena lisame <script> elemendi <head> elemendi sisse, kuhu hakkame edaspidi kirjutama JavaScript koodi. (vt koodinäide 3)

```
<script type="text/javascript"></script>
```

Koodinäide 3. Script elemendid

Lisame <body> elemendi sisse järgneva unikaalse ID-ga div elemendi (vt koodinäide 4). Antud elemendi sisse laeme YouTube'i mängija.

```
<div id="youTubePlayer"></div>
```

Koodinäide 4. div element

2.2 Video laadimine HTML dokumenti

Selles peatükis kirjeldatakse kuidas YouTube *API*'iga laadida üksikvideo ning selgitatakse vajalikke parameetreid antud toimingu läbiviimiseks. Peatüki lõpus oskab kasutaja laadida dokumenti ette antud videot YouTube keskonnast ja vajadusel seadistada video esitlust.

Järgnevalt laeme esimese video dokumenti, kasutades eelnevalt loodud *HTML* lehte. Seejärel loome neli uut muutujat.

- Esimene on globaalne muutuja "player", mille väärtuseks saab *player*'i objekt, mis omistatakse talle koodinäites 6.
- Teine muutuja "params" on objekt, mis sisaldab muutujat "allowScriptAccess" väärtusega "always". Antud seadistus on vajalik selleks, et Youtube *player*'i *SWF* saaks kutsuda funkstioone antud *HTML* lehel.
- Kolmanda muutuja väärtuseks on laaditava video ID.

- Neljas muutuja on laaditava video aadress. Aadressiga on kaasa antud järgmised parameetrid.
    - *enablejsapi* - võimaldab kasutada YouTube *player*'i *JavaScript API*'t, väärtusteks 0 või 1, millest 0 on eitav ja 1 tõene
    - *playerapiid* - eesmärgiks kontrollida meie poolt laetud YouTube mängijat (vt koodinäide 6)
    - *version* - *JavaScript API* versioon, antud kasutusjuhendis kasutame kõige uuemat versioon kolme
    - *controls* - peidame video esitluse kontrollimise, väärtusteks 0 või 1, millest 0 on eitav ja 1 tõene
    - *autoplay* - keelame videol automaatselt mängima hakata, väärtusteks 0 või 1, millest 0 on eitav ja 1 tõene
- Viimasena kutsume välja *SWFObject* meetodi "embedSWF", mis laeb mängija YouTube-ist ja seob selle eelnevalt loodud div elemendiga (vt koodinäide 4)

Lisame kirjutatud *JavaScript* koodi (vt koodinäide 5) kolmandas koodinäites loodud <script> elemendi vahele.

```
var player,
    params = {allowScriptAccess: "always"},
    ytPlayerApiID = "ytplayer",
    videoID = "5TIL1PQ4Lj0",
    videoUrl = "http://www.youtube.com/v/" + videoID +
"?enablejsapi=1&playerapiid=" + ytPlayerApiID + "
&version=3&controls=0&autoplay=0";

swfobject.embedSWF(videoUrl, "youTubePlayer", "425", "356", "8", null,
null, params);
```

Koodinäide 5. Üksiku video laadmine Youtube-ist

Defineerime uue funktsiooni "onYouTubePlayerReady" (vt koodinäide 6), mis käivitatakse alati kui YouTube player on valmis laetud. Funktsioon kontrollib, kas koodinäites 5 muutuja "videoUrl"-iga kaasa antud muutuja "ytPlayerApiID" on võrdne funktsiooni parameetriga "playerApiID". Juhul kui võrdlus on tõene, omistatakse "player" muutujale YouTube *player*-i

objekt, mille kaudu saame edaspidi välja kutsuda erinevaid *API* meetodeid. Lisame selle <script> elemendi vahele (vt koodinäide 3).

```
function onYouTubePlayerReady(playerApiID) {
  if (playerApiID == ytPlayerApiID) {
    player = document.getElementById("youTubePlayer");
  }
}
```

Koodinäide 6. *JavaScript* funktsioon

Juhul kui on vaja laadida uus video *player*'isse, saab seda teha kutsudes välja *loadVideoById* või *cueVideoById* meetodit *player* nimelisest objektist. Kahe meetodi erinevus seisneb selles, et *loadVideoById* alustab video mängimist pärast video laadimist. Mõlemad meetodid kutsutakse ühe samasuguste parameetritega välja, milleks on järgmine objekt muutujatega:

- videoId - laaditava video ID
- startSeconds - video mängimise ajahetk. Vaikimisi väärtuseks on 0 ehk video algus.
- suggestedQuality - video kvaliteet, milleks võivad olla järgnevad väärtused: small, medium, large, hd720, hd1080, highres, default.

Näide kuidas kasutada *loadVideoById* meetodit. Koodinäites on lisatud muutujatele *startSeconds* ja *suggestedQuality* nende vaikimisiväärtused, antud muutjad võib ka määramata jätta, kuna *videoId* on ainukene kohustuslik muutuja. (vt koodinäide 7)

```
player.loadVideoById({
  videoId: 'VIDEO_ID',
  startSeconds: 0,
  suggestedQuality: 'default'
})
```

Koodinäide 7. Uue video laadimine

Nüüd peaks meil valmis olema järgnev kood (vt koodinäide 8):

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Youtube JavaScript player API test</title>
  <style>
    a {display:block;}
  </style>
  <script
src="//ajax.googleapis.com/ajax/libs/swfobject/2.2/swfobject.js"></scrip
t>
  <script type="text/JavaScript">
    var player,
        params = {allowScriptAccess: "always"},
        ytPlayerApiID = "ytplayer",
        videoID = "5TIL1PQ4Lj0",
        videoUrl = "http://www.youtube.com/v/" + videoID +
"?enablejsapi=1&playerapiid=" + ytPlayerApiID + "
&version=3&controls=0&autoplay=0";

    swfobject.embedSWF(videoUrl, "youTubePlayer", "425", "356", "8",
null, null, params);

     function onYouTubePlayerReady(playerApiID) {
       if (playerApiID == ytPlayerApiID) {
         player = document.getElementById("youTubePlayer");
       }
     }
  </script>
</head>
<body>
   <div id="youTubePlayer"></div>
</body>
</html>
```

Koodinäide 8. Valmis kood

Hetkel see kood laeb ainult video dokumenti.

2.3 Video esitluse operatsioonid

Selles peatükis selgitatakse tähtsamad video esitluse operatsioonid. Õppetüki lõpus oskab kasutaja dokumenti laetud video esitlust opereerida *API* kasutuses olevate meetoditega.

Järgnev on nimekiri meetoditest koos seletusega, mida kutsutakse välja *player* objektist (vt koodinäide 6).

- *playVideo* - alustab laetud video mängimist
- *pauseVideo* - peatab hetkel mängiva video
- *stopVideo* - lõpetab hetkel mängiva video
- *seekTo* - muudab mängiva või peatatud video vaatamise aja hetkel. Meetodi esimeseks parameetriks on ajahetk sekundites ning teiseks on *bool* tüüp. Teine parameeter määrab ära, kas video ajahetk võib olla rohkem hetkel video eellaetud ajast.
- *setPlaybackQuality* - määrab video kvaliteedi, milleks võivad olla järgnevad väärtused: small, medium, large, hd720, hd1080, highres, default.
- *setPlaybackRate* - määrab video esitluse kiiruse, milleks võivad olla näiteks järgnevad väärtused: 0.25, 0.5, 1, 1.5 ja 2. Kõik võimalikud kiirused antud video kohta saab kätte *getAvailablePlaybackRates* meetodit, mis väljastab väärtuste jada.
- *mute* - vaigista heli
- *unMute* - taastab heli
- *setVolume* - määrab video helitugevuse, väärtuseks võivad olla täisarvud vahemikus 0 kuni 100

Järgnev kood tuleb lisada <body> elemendi sisse, mis demonstreerib eelnevalt kirjeldatud video esitluse meetodeid. (vt koodinäide 9)

```
<a href="JavaScript: player.playVideo()">Mängi</a>
<a href="JavaScript: player.pauseVideo()">Peata</a>
<a href="JavaScript: player.stopVideo()">Lõpeta</a>
<a href="JavaScript: player.seekTo(60)">Alusta mängist alates esimest
minutist </a>
<a href="JavaScript: player.seekTo(0)">Alusta video mängimist
algusest</a>
<a href="JavaScript: player. setPlaybackRate(2)">Mängi video 2x
kiiremini</a>
<a href="JavaScript: player. setPlaybackRate(1)">Mängi video normaal
kiirusel</a>
<a href="JavaScript: player.mute()">Vaigista heli</a>
<a href="JavaScript: player.unMute()">Taasta heli</a>
<a href="JavaScript: player.setVolume(50)">Määra helitugevuseks 50%</a>
<a href="JavaScript: player.setPlaybackQuality('hd720')">Määra video
kvaliteediks 'hd720'</a>
<a href="JavaScript: player.setPlaybackQuality('default')">Määra video
kvaliteediks vaikimisi väärtus</a>
```

Koodinäide 9. Video esitluse operatsioonid

2.4 Video nimekirja laadimine HTML dokumenti

Paljudel juhtudel ei piisa ainult üksiku video mängimisest kodulehel, tuleb hoopis näidata videode nimekirja. Olgu selleks näiteks koolituskeskkond lynda.com, kus on vaja videod grupeerida peatükkide kaupa.

Antud õppetüki eesmärgiks kirjeldada video nimekirja laadimist YouTube'ist ja selgitada selle protseduuri seadistus parameetreid. Järgneva peatüki lõpus oskab lugeja mängida erinevat tüüpi video nimekirja, määrates ära video, selle kvaliteedi ja ajahetke, millest alustatakse mängimist.

Video nimekirja laadimiseks on *API*-s olemas kaks meetodit: *cuePlaylist* ja *loadPlaylist*. Nende kahe meetodi erinevus seisneb selles, et *loadPlaylist* peale nimekirja laadimist alustab video mängimist. Mõlemad meetodid kasutavad ühesugust parameetrit välja kutsumisel. Antud parameetriks on *JavaScript* objekt, mis koosneb järgmistest muutujatest:

- *list* - nimekirja ID, ainukene kohustuslik muutuja.
- *listType* - nimekirja tüüp, milleks võib olla kolm järgnevat väärtust.
  - *list* - vaikimisi väärtus, eeldab, et *list* väärtuseks oleks nimekirja ID.
  - *search* - juhul kui *list* väärtuseks on ostingusõna.
  - *user_uploads* - juhul kui *list* väärtuseks on kasutaja, kelle üleslaetud videosid soovitakse saada.
- *index* - esimesena mängitava video järjekorra number (alustades nullist), vaikimisi väärtus on 0 ehk esimene video.
- *startSeconds* - esimese video mängimise ajahetk, sarnaneb *seekTo* meetodiga. Vaikimisi väärtuseks on 0 ehk video algus.
- *suggestedQuality* - videode soovitatav kvaliteet, sarnaneb *setPlaybackQuality* meetodiga. Vaikimisi väärtuseks on "default".

Järgev kood, mis laeb videod, kasutades otsinguks märksõna "tere maailm!", tuleks lisada <body> elemendi vahele kõige viimaseks.

```
<a href="JavaScript: player.cuePlaylist({list: 'tere maailm!', listType: 'search'})">Lae videode nimekiri</a>
```

Koodinäide 10. Videode nimekiri otsingu märksõna alusel.

2.5 Video nimekirja esitluse operatsioonid

Käesolevas peatükis vaatame üle video nimekirja esitluse operatsioonid. Lisaks alljärgnevatele operatsioonidele, kehtivad ka peatükis nimega "Video esitluse operatsioonid" video nimekirjas esitluses olevale videole (vt peatükk 2.3). Peatüki lõpus oskab kasutaja dokumenti laetud video nimekirja erinevaid videosid laadida, samuti määrata, kas videosid mängitakse tavapärases või suvalises järjestuses ning, kas esitluses olev video jääb korduma või mitte.

- *setLoop* - määrab, kas video nimekiri jääb lõpmatult korduma, väärtuse tüübiks on *bool* ("true" või "false"), vaikimisi lõpetab *player* mängimise, kui viimane video sai läbi. Valik jääb kehtima ka siis, kui laed uue video nimekirja.

- *setShuffle* - määrab, kas nimekirjas olevaid videosid mängitakse suvalises järjestuses, väärtuse tüübiks on *boolean* ("true" või "false"). Valik ei jää kehtima uue nimekirja laadimisel.
- *nextVideo* - alustab nimekirjas järgmise video mängimist. Juhul kui kutsutakse siis, kui viimane video mängib ning on määratud, et nimekiri jääb mängima lõpmatult, alustatakse nimekirjas esimese video mängimist. Kõigil teistel juhtudel lõpetab *player* mängimise.
- *previousVideo* - alustab nimekirjas eelmise video mängimist. Juhul kui kutsutakse siis, kui esimene video mängib ning on määratud, et nimekiri jääb mängima lõpmatult, alustatakse nimekirja viimase video mängimist. Juhul kui kutsutakse siis, kui esimene video mängib ning ei ole määratud, et nimekiri jääb mängima lõpmatult, alustatakse nimekirja esimese video uuesti mängimist.
- *playVideoAt* - alustab video mängimist nimekirja järjekorra alusel (alustades nullist).

Järgnev kood, mis kasutab ülalmainitud meetodeid tuleb lisada <body> elemendi kõige viimaseks. (vt koodinäide 11)

```
<a href="JavaScript: player.setLoop(true)">Jää nimekirja kordama</a>
<a href="JavaScript: player.setShuffle(true)">Mängi videosid suvalises
järjekorras</a>
<a href="JavaScript: player.nextVideo()">Mängi järgmist video</a>
<a href="JavaScript: player.previousVideo()">Mängi eelmist video</a>
<a href="JavaScript: player.playVideoAt(4)">Mängi viiendat video</a>
```

Koodinäide 11. Videode nimekirja esitluse operatsioonid

2.6 API sündmused

Selles peatükis uurime *API* sündmuseid, mille kasutamine on oluline osa igasuguste rakenduse arendamisel. Sündmuste jälgimine on eelkõige kasulik selleks, et loodud rakendusel oleks pidev ülevaade toimuvast ning suudaks vastavalt vajadusele reageerida. Õppetüki lõpus oskab kasutaja jälgida koodi tasandil kõiki *API* sündmusi ja nende toimumisel rakendada enda poolt määratletud funktsiooni.

Alljärgnevalt on kirjeldatud kõik *API* sündmused.

- *onStateChange* - kutsutakse välja igakord kui *player*'i staatus muutub ning edastatakse uus staatus. Erinevaid staatuseid võib kokku olla kuus.
    - -1 - video mängimist pole alustatud, käivitatakse alati peale esimese video laadimist.
    - 0 - video lõpetas mängimise
    - 1 - video alustas mängimist
    - 2 - video on pausi peal
    - 3 - videot puhverdatakse
    - 5 - juhul kui uus video on laetud
- *onPlaybackQualityChange* - toimub juhul, kui video mängimise kvaliteet muutus. Väljundiks on sama sisend, mida kasutatakse *setPlaybackQuality* meetodis.
- *onPlaybackRateChange* - esineb siis, kui muudetakse video mängimise kiirust. Väljundiks on sama sisend, mida kasutatakse *setPlaybackRate* meetodis.
- *onError* - kutsutakse välja igakord, kui *player*'is esineb viga. Väljastatakse konkreetne veakood, mis võib olla üks järgmistest.
    - 2 - päring sisaldab vigast parameetrit (näiteks video ID ei sisalda 11 tähemärki või sisaldab keelatud tähemärke)
    - 100 - juhul kui päritud videot ei leitud. Väljastatakse tavaliselt siis, kui video on eemaldatud või privaatne.
    - 101 - video omanik ei luba selle kasutamist YouTube keskkonna väliselt
    - 150 - sama, mis veakood 101

Iga sündmuse jälgimiseks tuleb lisada jälgija, mis käivitab igakord vastavalt defineeritud funktsiooni sündmuse toimumisel. Selleks, et jälgida sündmuseid meie koodis, tuleb läbi viia koodis mõned muudatused.

Alustuseks loome uue funkstiooni "addYouTubeListeners", mis lisab erinevatele sündmustele kuularid ning lisame selle koodi <script> elementide vahele kõige lõppu. (vt koodinäide 12)

```
function addYouTubePlayerListeners(player) {
  player.addEventListener("onStateChange", "onPlayerStateChange");
  player.addEventListener("onPlaybackQualityChange",
"onPlayerQualityChange");
  player.addEventListener("onPlaybackRateChange", "onPlayerRateChange");
  player.addEventListener("onError", "onPlayerError");
}
```

Koodinäide 12. "onYouTubePlayerReady" funktsioon

Teisena loome uue div elemendi ID'ga "eventLog", kuhu läheb sündmuste logi ning lisame uue elemendi <body> elemendi vahele kõige viimaseks elemendiks. (vt koodinäide 13)

```
<div id="eventLog"></div>
```

Koodinäide 13. Sündmuste logi element

Järgmisena defineerime koodinäites 11 lisatud kuularitele pöördumise funkstioonid ning lisame selle peale "addYouTubePlayerListeners" funkstiooni. (vt koodinäide 14)

```
function onPlayerStateChange(state) {
  document.getElementById("eventLog").innerHTML =
document.getElementById("eventLog").innerHTML + "<div>Staatuse muutus: "
+ state + "</div>";
}
function onPlayerQualityChange(quality) {
  document.getElementById("eventLog").innerHTML =
document.getElementById("eventLog").innerHTML + "<div>Video kvaliteedi
muutus: " + quality + "</div>";
}
function onPlayerRateChange(rate) {
  document.getElementById("eventLog").innerHTML =
document.getElementById("eventLog").innerHTML + "<div>Video kiiruse
muutus: " + rate + "</div>";
}
function onPlayerError(error) {
  document.getElementById("eventLog").innerHTML =
document.getElementById("eventLog").innerHTML + "<div>Viga: #" + error +
"</div>";
}
```

Koodinäide 14. Kuularite pöördumisfunktsioonid

Kõige viimasena on vaja eelnevalt koodinäites 11 defineeritud funktsioon "addYouTubeListeners" kutsuda välja koodinäites 6 olevas funktsioonis nii, et tulemus oleks järgmine. (vt koodinäide 15)

```
function onYouTubePlayerReady(playerApiID) {
  if (playerApiID == ytPlayerApiID) {
    player = document.getElementById("youTubePlayer");
    addYouTubePlayerListeners(player);
  }
}
```

Koodinäide 15. Muudatused funktsioonis "onYouTubePlayerReady"

2.7 Kokkuvõte

YouTube *player*'i JavaScript API on oma olemuselt väike ning loogika poolest lihtne, samas on selles olemas kogu vajalik funktsionaalsus, et arendada korralik videokeskkond. *API* võimaldab laadida üksikvideosid ja videode nimekirjasid ning olemasolevate meetoditega saab erinevaid funktsionaalsusi rakendada videode esitlemisel.

Üheks puudujäägiks peab töö autor seda, et sündmuste lisamisel meetodiga *addEventListener* ei saa pöördumisfunktsiooni ennast lisada meetodi parameetrina.

Kasutades antud kasutusjuhendis õpitud teadmisi, on kasutajal piisavalt ülevaatlik sissejuhatus olemas, et arendada väiksemat laadi videode esitamise rakendus. Samuti piisab praegustest baasteadmistest, et edasi õppida keerukamaid *API* funktsioone.

3 KASUTAJATE TAGASISIDE

Kasutajate tagasiside on koostatud Tallinna Ülikooli bakalaureuse õppeastme informaatika eriala tudengite peal tunnis Veebiraamistikud, kes lahendasin antud töö lisas olevaid ülesandeid. Ülesannete lahendamine ja tagasiside andmine toimus 07.10.2014 Tallinna Ülikoolis, ruumis A-406 8:15 hommikul, kus oli kohal 18 õppurit.

Tund oma ülesehituselt koosnes neljast osast. Esimeses osas tutvustas töö autor ennast ja põhjust, miks ta antud tunnis viibib. Teises osas käidi läbi antud seminaritöö peatükk-haaval. Kolmandas osas lahendasid õpilased töö lisas olevat kolme ülesannet, mille tegemiseks oli neil kasutada käesolev seminaritöö. Viimaseks osaks oli kasutajate tagasisside, kus alustuseks esitas töö autor auditooriumile küsimuse kasutusjuhendi selguse kohta ja kes kui palju ülesandeid valmis sai. Ülesannete lahendamiseks oli õppuritel aega pool tundi. Seejärel toimus õpilaste suuline tagasiside YouTube *player*'i *API* kohta.

Tunnis antud kolmest ülesandest jõudsid kõik valmis ainult üksikud üliõpilased (kahe esimesega said kõik hakkama). Hiljem selgus, et probleemiks oli ajapuudus. Õpilaste tagasiside käigus selgus, et kasutusjuhendi sõnastus oli arusaadav ning antud juhendi põhjal sai probleemideta etteantud ülesanded lahendada.

Üheks esimeseks puuduseks *YouTube*'i *API*'s, tõid õpilased esile, et sündmuste lisamisel ei saa sündmuse pöördumisfunktsiooni ennast lisada sündmuse lisamise meetodile. Sama puudus on ka mainitud kasutusjuhendi kokkuvõttes (vt peatükk 2.7).

KOKKUVÕTE

Antud seminaritöö eesmärk oli luua eestikeelne kasutusjuhend ja proovida see läbi õpilastega, et tutvustada *YouTube JavaScript player*'i *API* funktsionaalsusi. Töö ehitati üles suurel osal koodinäidetest koos selgitava tekstiga. Kasutusjuhendi loomisel kasutati inglisekeelset materjali, mis on kättesaadav *YouTube JavaScript Player API Reference* lehel.

Otsides olemasolevaid sama teemalisi materjale internetist selgus, et õpetusi, mis kirjeldaks *API* üksikvideo võimalusi leidub erinevaid kuid mõned üksikud on piisavalt ülevaatlikud, ning selgitavad põhjalikumalt koodinäiteid. Video nimekirjale rakendatavaid *API* võimalusi selgitavaid õpetusi, mis oleksid vähegi sisukad leidus üksikuid ning needki kasutasid vähesel määral *API* funktsionaalsusi.

Tagasisidest üliõpilastega, kes antud kasutusjuhendi põhjal prooviülesandeid lahendasid, selgus, et loodud kasutusjuhend osutus efektiivseks. Samas ilmnesid ka mõned kitsaskohad – õpilastel tekkis kasutusjuhendiga töötamisel ajapuudus ja fakt, et *API*'i sündmuste tagasiside meetodile ei saa koheselt parameetriks kirjutada funktsiooni, vaid ainult funktsiooni nime.

Antud seminaritöö tegemise käigus jõudis autor järeldusele, et *Flash* tehnoloogia on vaikselt oma aja ära elanud ja pole enam aktuaalne, millele ka viitas asjalike olemasolevate õpetuste puudumine. *YouTube* videode esitamiseks oleks mõistlikum kasutada *Flash player*'i asemel *HTML5* tehnoloogiat. Selleks, et seda teha tuleks *JavaScript API* asemel kasutada *IFrame API*'t. *IFrame API* on oma funktsionaalsuse poolsest võrdne *JavaScript API*'ga, erinevus seisneb selles, et *YouTube* rakendus asub *iframe HTML* elemendi sees.

KASUTATUD ALLIKAD

YouTube (2014). Statistics. Kasutamise kuupäev 24.09.2014, allikas: https://www.youtube.com/yt/press/statistics.html

YouTube (2014). YouTube JavaScript Player API Reference. Kasutamise kuupäev 24.09.2014, allikas: https://developers.google.com/youtube/js_api_reference

W3Techs (2014). Usage of JavaScript for websites. Kasutamise kuupäev 24.09.2014, allikas: http://w3techs.com/technologies/details/cp-javascript/all/all

Webmonkey (2010). YouTube Tutorial Lesson 1 – The Player API. Kasutamise kuupäev 02.12.2014, allikas: http://www.webmonkey.com/2010/02/youtube_tutorial_lesson_1_the_player_api

Joseph P. McCarthy (2014). Build a custom YouTube playlist player. Kasutamise kuupäev 02.12.2014, allikas: http://www.ibm.com/developerworks/library/wa-bluemix-youtube/index.html

IBM developerWorks (2015). Welcome to developerWorks. Kasutamise kuupäev 02.12.2014, allikas: http://www.ibm.com/developerworks/aboutdw

LISAD

Ülesanded

Alljärgnevad kolm ülesannet on iseseisvaks lahendamiseks.

Ülesanne 1

Kasutades koodinäidet number 7, lae suvaline endapoolt valitud video *HTML* dokumenti. Järgmisena lisa mängi (*play*) ja peata (*pause*) nuppudega.

Ülesanne 2

Täienda ülesanne 1 tehtud valmis lahendust. Lisa nupp, mis laeb enda poolt valitud video nimekirja *player*'isse ning täienda lahendust eelmine (*previous*) ja järgmine (*next*) nuppudega.

Ülesanne 3

Kasutades koodinäidet number 7 lisa nupp, mis laeb enda poolt valitud video *HTML* dokumenti. Juhul kui video lõpetab mängimise, lae automaatselt enda poolt valitud otsingu sõnaga video nimekiri *player*'isse, mis alustaks mängimist teisest videost. Selleks, et paremini oma lahendust testida saab video laadimise aadressil parameetri *controls* väärtuse 1 muuta.

Lahendused

Alljärgnevad koodinäited sisaldavad ülesannete lahendusi.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Youtube javascript player API test</title>
  <style>
    a {display:block;}
  </style>
  <script
src="//ajax.googleapis.com/ajax/libs/swfobject/2.2/swfobject.js"></scrip
t>
  <script type="text/javascript">
    var player,
        params = {allowScriptAccess: "always"},
        ytPlayerApiID = "ytplayer",
        videoID = "5TIL1PQ4Lj0",
```

```
        videoUrl = "http://www.youtube.com/v/" + videoID +
"?enablejsapi=1&playerapiid=" + ytPlayerApiID + "
&version=3&controls=0&autoplay=0";

    swfobject.embedSWF(videoUrl, "youTubePlayer", "425", "356", "8",
null, null, params);

     function onYouTubePlayerReady(playerApiID) {
       if (playerApiID == ytPlayerApiID) {
         player = document.getElementById("youTubePlayer");
       }
     }
  </script>
</head>
<body>
      <div id="youTubePlayer"></div>
      <a href="JavaScript: player.playVideo()">Mängi</a>
      <a href="JavaScript: player.pauseVideo()">Peata</a>
</body>
</html>
```

Koodinäide 16. Ülesanne 1 lahendus

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Youtube javascript player API test</title>
  <style>
    a {display:block;}
  </style>
  <script
src="//ajax.googleapis.com/ajax/libs/swfobject/2.2/swfobject.js"></scrip
t>
  <script type="text/javascript">
    var player,
        params = {allowScriptAccess: "always"},
        ytPlayerApiID = "ytplayer",
```

```
        videoID = "5TIL1PQ4Lj0",
        videoUrl = "http://www.youtube.com/v/" + videoID +
"?enablejsapi=1&playerapiid=" + ytPlayerApiID + "
&version=3&controls=0&autoplay=0";

    swfobject.embedSWF(videoUrl, "youTubePlayer", "425", "356", "8",
null, null, params);

     function onYouTubePlayerReady(playerApiID) {
       if (playerApiID == ytPlayerApiID) {
         player = document.getElementById("youTubePlayer");
       }
     }
  </script>
</head>
<body>
      <div id="youTubePlayer"></div>
      <a href="JavaScript: player.playVideo()">Mängi</a>
      <a href="JavaScript: player.pauseVideo()">Peata</a>

      <a href="JavaScript: player.cuePlaylist({list:
'PL36D6D555F9CA0A28'})">Lae videode nimekiri</a>
      <a href="JavaScript: player.nextVideo()">Mängi järgmist video</a>
      <a href="JavaScript: player.previousVideo()">Mängi eelmist
video</a>
</body>
</html>
```

Koodinäide 17. Ülesanne 2 lahendus

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Youtube javascript player API test</title>
  <style>
    a {display:block;}
  </style>
  <script
```

```
src="//ajax.googleapis.com/ajax/libs/swfobject/2.2/swfobject.js"></scrip
t>
  <script type="text/javascript">
    var player,
        params = {allowScriptAccess: "always"},
        ytPlayerApiID = "ytplayer",
        videoID = "5TIL1PQ4Lj0",
        videoUrl = "http://www.youtube.com/v/" + videoID +
"?enablejsapi=1&playerapiid=" + ytPlayerApiID + "
&version=3&controls=1&autoplay=0";

    swfobject.embedSWF(videoUrl, "youTubePlayer", "425", "356", "8",
null, null, params);

     function onYouTubePlayerReady(playerApiID) {
       if (playerApiID == ytPlayerApiID) {
         player = document.getElementById("youTubePlayer");
             player.addEventListener("onStateChange",
"onPlayerStateChange");
       }
     }
       function onPlayerStateChange(state) {
            if (state == 0) {
              player.cuePlaylist({list: 'hangover 3', listType:
'search', index: 1})
            }
       }
  </script>
</head>
<body>
      <div id="youTubePlayer"></div>
</body>
</html>
```

Koodinäide 18. Ülesanne 3 lahendus