Tallinna Ülikool
Informaatika Instituut

# Veebilehe loomine HTML5 abil

Seminaritöö

Autor: Vladimir Vološin
Juhendaja: Andrus Rinde

Autor: ..... ..... „ ..... „ 2011
Juhendaja: ..... „ ..... „ 2011

Tallinn 2011

## *Sisukord*

# Sissejuhatus

Veeb on oma olemasolu jooksul teinud läbi pika arengu – algselt tekstipõhine, nüüd erinevaid interaktiivseid tehnoloogiaid kasutav, erinevaid meediaid (tekst, graafika, heli, video) hõlmav keskkond. Senine veebilehtede loomise keel HTML *(HyperText Markup Language)* on oma võimalustelt aja nõuetele jalgu jäänud. HTMLi pole edasi arendatud juba aastast 2000 ning seepärast ongi aja jooksul peale tulnud uusi erinevaid tehnoloogiaid, mis reeglina nõuavad endale lisaks pistikprogramme ning muudavad veebilehed andmemahukateks.

Kuna veebilehed muutuvad üha rohkem dünaamilisemaks oli vaja, et ka HTML ajaga kaasa läheks. Oli aeg luua uus standard, mis võtaks kokku senised tööd ning muudaks koodi kirjutamise lihtsamaks ning semantilisemaks. Alates 2004. aastast on hakati looma HTML uut versiooni, mis ühendab endas XHTML, DOM ning eelnevat versiooni HTML 4.01. HTML5 saab standardi staatuse küll alles 2012. aastal, kuid mitmed uued võimalused on kasutatavad juba täna ning väärivad tutvustamist.

Antud töö eesmärgiks on tutvustada HTML5 uusi olulisi võimalusi. Eesmärgi saavutamiseks annab autor kirjanduse põhjal ülevaate tekkeloost, HTML arengust, selle erinevatest versioonidest. Näitab, kuidas töötavad uued elemendid, vormid, sisestusvälja tüübid ning annab ülevaate sellest, mida on võimalik saavutada uute programmeerimise liidestega (API). Uusi võimalusi katsetades, annab soovitusi veebi loojatele kuidas oma lehekülg nii otsingumootoritele kui kasutajatele lihtsamaks muuta.

Töö käigus koostas autor veebilehe, kasutades uut markeerimiskeelt, mis vastas täielikult tulevasele standardile. Leheküljega saab tutvuda aadressil http://www.tlu.ee/~vova/web/html5/

# 1. HTML ajalugu

Vajadus luua standartseid dokumente, kus sisu ja vorm oleks üksteisest eraldatud, on märksa vanem kui veeb - samu põhimõtteid rakendati ka paberdokumentatsiooni ettevalmistamisel. Ajal kui hakati looma arvutitevahelisi ühendusi, pakkus Tim Berners-Lee välja uue infovahetuse protokolli - hypertext, kus mingi võtmesõna kaudu saab pöörduda sellega seotud dokumendi juurde. Nii loodigi 70. aastatel SGML (Standardized Generalized Markup Language), millega loodeti saada üldkasutatav standard igasuguse dokumentatsiooni koostamiseks. Ning juba 1991. aastal võis leida internetist HTMLi tutvustav dokument mis kandis nime HTML Tags.

Veebistandardid on W3C *(World Wide Web Consortium)* poolt koostatud tehnoloogiad, mis garanteerivad, et kõik veebilehed on ja jäävad internetilehitsejate jaoks üheselt mõistetavateks ning näevad tulevikus uutel veebilehitsejatel samasugused välja nagu praegu. See rahvusvaheline organisatsioon loodi 1994. aasta oktoobris ning selle asutaja oli Tim Berners-Lee. Algselt loodi veeb kommunikatsiooni vahendiks, et igaüks määramata asukohast saaks vahetada informatsiooni kuid nüüdseks on see juba midagi enamat. Sellepärast nende põhieesmärgiks ongi luua standardeid, et olenemata riistvara ning tarkvara võimsusest, keelest, kultuurist või asukohast oleks veeb kõigile ühtemoodi kättesaadav.

## 1.1 Esialgne versioon

Esimese põlvkonna veebilehtedeks loetakse aastatel 1991 – 1994 loodud lehti. Selle aja veebilehed olid peamiselt staatilised ning kujunduselt väga lihtsad, sisaldades vaid pealkirju ning tekstilõike, vahel ka viiteid teistele veebilehtedele. Väheseid tehnoloogilisi võimalusi "kompenseeriti" aga kirjude taustapiltide jms abil.
Esialgne HTML (*HyperText Markup Language)* võimaldas kasutada lihtsaid tekstivorminguid (lõik, reavahetus, nimistud), linke ja lihtsat graafikat. HTMLi versiooni 1.0 ametlikult ei eksisteeri, kuna alguses kasutasid erinevad grupid erinevaid tõlgendusi.
( http://www.eneta.ee/SiteCollectionDocuments/vs/Veebistuudium_disain.pdf, 2010 )

## 1.2 HTML versioon 2.0

Aastast 1994 oli esimene laialdaselt heakskiidetud standard, mis võimaldas lisaks eelnevale kasutada ka sisestusvorme (eeldasid töötlustarkvara olemasolu serveris). Veebilehed hakkasid enam keskenduma sisule ja külastajatele – lisandusid külalisteraamatud ning alustati külastajate arvu järgimist. Lihtsad pealkirjad ja tekstilõikud asendasid värvilised ja vilkuvad tekstid/pidid, andmetabelid, reklaambännerid. Lisandusid elemendid nagu *headings, anchors, lists.* Tekkisid vormid (*forms*), lisaks sai nüüd kasutada tabeleid, rea vahetust ning loodud dokumenti kommenteerida nii, et teised seda ei näe. Kommentaar pidi jääma <! ------ > nurksulgude vahele.
( http://www.eneta.ee/SiteCollectionDocuments/vs/Veebistuudium_disain.pdf, 2010 )

## 1.3 HTML versioon 3.2

Aina olulisemaks muutus navigatsioon - veebilehtedel oleva info struktureerimine ning selle kiiresti ülesleidmine. Võimaldas juba luua tabeleid ja teksti ümber pildi mähkida. Samuti võeti kasutusele raamida *frames.* Veebi muutis elavamaks ning vahetumaks uus tehnoloogia Macromedia Flash, populaarseks said tervituslehed *intro*. Lehe lihtsustamiseks alustati kujunduse ning sisu eraldamisega – kasutusele võeti CSS *(Cascading Style Sheet)* stiililehed. Lisandusid sellised teegid nagu *address* –

kontakt andmete ja allkirja andmiseks, *applet* – Java programmi kasutamiseks dokumendis, *div* – dokumendi jaotamiseks, *body* – veebilehe kehaosa, samuti erineva suurusega pealkirjad, *center* – teksti viimiseks dokumendi keskele. Algselt arendas seda Netscape HTML 3.0 nime all, kuid hiljem kuulutati see aegunuks. W3C hakkas ise esimest korda arendama uut versiooni 3.2 ning see oli esimene veebistandard mille nad väljastasid.

## 1.4 HTML versioon 4.01

Neljanda põlvkonna veebilehtede hulka kuuluvad Web 2.0 rakendused. Olulise sisuna on veebi liikunud videod. Veebilehed pole enam vaid staatilised ressursid. Veebilehe külastaja on üks paljudest, kes loob selle sisu. Wikid ja ajaveebid annavad igale inimesele võimaluse avaldada oma arvamust ning jagada oma mõtteid. 1998. aastal loodi HTML 4.0 kuid 24. detsembriks 1999 tehti nii palju muudatusi, et standard nimetati ümber 4.01'ks. Hakati eraldama dokumente stiilidest. Uued elemendid *id, class, style* lubasid omistada kindlale elemendile väärtuseid. Samuti uuendusi tabelitele ja listidele atribuutide näol. Vormide täitmisel uus nupp *button* nuppude loomiseks. Skriptide lisamine sai võimalikuks tänu *script* elemendile. Said võimalikuks eraldi aknad *iframe*, nüüdsest lingile vajutades ei kuvatud tervet lehekülge uuesti, vaid uus link koos informatsiooniga teatud aknas, sellega hoiti ressursse kokku. HTML 4.01 sai juurde atribuute skriptimise jaoks *onclick, onmousedown, onmousemove, onkeypress* ning samuti mitmeid muid elemente.

## 1.5 XML & XHTML

XHTML (*Extensible HyperText Markup Language*) on veebilehtede loomiseks kasutatav keel. XML on aastast 2000. Antud standard on samuti SGML tuletis, mille eesmärk on luua hea struktuuriga dokumente. XML on mõeldud andmete kirjeldamiseks HTML aga nende näitamiseks. HTMLi ja XMLi "kokkusulatamisel" 2001. aastal saadi XHTML. XHTML 1.0 sai W3C soovituseks 26. jaanuaril 2000. XHTML on HTMLi puhtam variant. Kaotati ära terve hulk väga probleemseid elemente. XHTMLi eripäraks varasemate versioonidega võrreldes on mõnevõrra suurem rangus - kõik märgendid on paaris, nõutav on väiketähtede kasutamine märgendites sest võrreldes HTMLiga on XML tõstutundlik. Põhimõtteks on saavutada

ühilduvus XMLiga, samal ajal täites ka HTML 4.01 nõudeid. Brauser suudab avada XHTML veebilehte kiiremini sest ei pea hakkama läbi vaatama mitte standardset koodi.

## *1.6 HTML tulevik*

HTML4.01 *(HyperText Markup Language)* on tänaseks päevaks juba vananenud tehnoloogia, sest seal ei ole lehe struktuur ja kujundus omavahel piisavalt lahutatud. Hetkel on tuntumateks veebitehnoloogiateks XHTML *(Extensible HyperText Markup Language)* ja CSS *(Cascading Style Sheets)*. Esimene on mõeldud lehe struktuuri ülesehitamiseks, teine lehe disainimiseks.

Osa veebiarendajaid ei soovinud XHTML'i omaks võtta sest see ei pakkunud piisavalt võimalusi ning seepärast 2004. alustas grupp WHATWG (Web Hypertext Application Technology Working Group) tööd uue standardi kallal. 2007. liitus selle grupiga W3C. Kui veel mõned aastad tagasi kinnitas W3C, et HTML 4.01 jääb viimaseks HTML-seeria keeleks, siis nüüd on olukord muutunud. Kolm suurt lehitsejate tarnijat - Apple, Opera ja Mozilla Foundation - said kokku WhatWG raames, eesmärgiks: arendada välja täiendatud ja uuendatud versioon klassikalisest HTMList. Hiljuti puhuti ka HTMLile taas elu sisse, nimelt 2007. aastal avalikustati HTML5 esimene mustand.

Uus HTML oleks koheselt äratuntav ka sellise www-disaineri poolt, kes „külmutati" 1999. aastal ning sulatati täna. HTML5 on sihipäraselt loodud sellisena, et ta oleks loetav ka lehitsejatega, mis teda ei tunnista. Isegi vanade veebilehitsejatega nagu näiteks Netscape 4 ja Internet Explorer 5. seeriaga saab uue tehnoloogiaga loodud lehte külastada. Tõsi, lehitseja ei tunneks uusi elemente ega teeks nendega midagi, kuid leht oleks siiski loetav. Kuigi öeldakse, et HTML5 tööversioon saab valmis aastaks 2012, on lõplikuks tähtajaks määratud 2022, mil antakse ametlik soovitus kasutada uut standardit. ( http://www.interneti.info/artiklid/html5/**,** 2010 )

# 2. HTML 5

HTML5 tutvustab uusi elemente HTML keelde mida pole juhtunud eelmise aastatuhande lõpust saadik. Mitmed uued väljapakutud elemendid omavad lihtsalt suuremat semantilist väärtust ning ei tee suurt muud midagi, kui on lihtsalt tähendusrikkamad alternatiivid vanale *div* elemendile. Sellegipoolest muudavad *header*, *footer* ja *nav* elemendid leheküljed otsingumootoritele kiiremini leitavamaks.
Uute struktuurielementide seas on *aside, figure, article* ja *section,* mis muudavad lehekülje struktueeritumaks ning kergelt loetavamaks. Uued reasisesed elemendid on näiteks *time, meter* ja *progress*. Uued lisamis-elemendid oleksid *video* ja *audio*. Uued interaktiivsed elemendid *details, datagrid, keygen, output* ja *command*. Erineva kirjastiili jaoks olemas on ka *em, code, dfn, cite, samp* ja *strong*. Ainuke muutus on *strong* stiilis, mis enne tähendas rõhutatud nüüd aga tähtsat teksti.
Elemendid ei pea olema suletud kuna lehitsejad „andestavad" vigu. Paljud neist on sellised vahendid, mida disainer vajanuks juba 1999. aastal. Kõik need uued elemendid on kergesti õpitavad tänu võimalikele paralleelidega juba olemasolevate elementidega. ( http://www.interneti.info/artiklid/html5/, 2010 )

Kuigi HTML5 lisas palju uusi, loobuti ka mõningatest elementidest. Kaotatud on *basefont, big, center, font, s, strike, tt, u* sest need elemendid on puhtalt presenteerimise jaoks ning on paremini käsitletavad CSSis (*Cascading Style Sheets*).
Elemendid *frame, frameset* ja *noframes* kustutati, kuna arvatakse et nende kasutamine kahjustab juurdepääsetavust. *Acronym* tekitas segadust ning soovitatakse kasutada lühendite jaoks nüüd hoopis *abbr* elementi. Element *applet* mis tuli sisse versioonis 3.2 on nüüd tarvituselt kadunud ning selle asemel pakutakse kasutada *embed* või *object* elementi.
Samas kadusid ära ka teatud elementide atribuudid. Eemaldatud atribuut *align* mõjutas järgmisi elemente: *caption, col, colgroup, div, iframe, h1..h6, hr, img, input, legend* jm. Kadusid ka *alink, link, text, background, border, framborder, frameset, marginheight, marginwidth* mõjutades suuresti *body* ning *table* elemente. Atribuutide kaotamine ei tähenda seda, et see või teine parameeter nüüd puudub. See tähendab, et HTML5 on puhtalt loodud dokumendi struktuuri loomiseks ja kõik kujundused tuleb teha CSS3 stiililehtedega.

# 3. Veebilehe loomine kasutades HTML5

HTML5 pakub arvukalt uusi elemente, märgendeid ja atribuute, mis iseloomustavad tänapäeva veebikülgede tüüpilist ülesehitust. Mõni uus on olemuselt sarnane <*div*> ja <*span*> märgenditele, kuid mõni nagu näiteks <*nav*> (veebilehe navigeerimisala) ja <*footer*> (jalus) omavad semantilist tähendust veebilehel. Selliste märgendite eesmärk on hõlbustada otsimootori indekseerimist ja veebilehtede näitamist pisikeste ekraanidega seadmete poolt.

Lehe loomisel polnud autor ise varasemalt HTML5ga kokku puutunud ning lähtus sellest, et tulevasel saidil peaksid olema demonstreeritud peamised uued elemendid: *header, article, footer, section, aside, figure* ning loomulikult *video* ja *audio*. Autor toob näiteid uuest vormi ja atribuutidest ning loomulikult integreeritud programeerimise liidestest (API). Kuidas mingi element välja näeb graafiliselt seda on võimalik näha lisades.

## *3.1 Veebilehe päis*

Kuna autor soovis proovida uut markeerimiskeelt, otsustas luua lihtsa lehekülje, kasutades HTML5 ja kujundus CSS 3.0is. Põhifunktsioonidest parema ülevaate ja puhtama koodi saavutamiseks viisin kujunduse täielikult HTMLst välja - CSS faili.

Esimeseks suuremaks ja lihtsamaks muudatuseks võib pidada *DOCTYPE* deklaratsiooni kirjeldust. Standardi kohaselt peab dokumendi kirjutamist alustama just sellest. Kui varasemalt tähendas *DOCTYPE* midagi keerukat, mida keegi ise üles kirjutada ei osanud ning seega tuli korrektse koodi nimel see kuskilt mujalt kopeerida, siis nüüd on asi palju lihtsam.

1. Kui varem võis DOCTYPE deklaratsioon välja näha selline

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTDHTML4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

2. Siis HTML 5 korral saab kasutada vormi

```
<!DOCTYPE html>
```

Edasi liigub päisesesse ehk teistesõnadega <*head*>. Elementi <*head*> sisse peavad jääma kõik metaandmed dokumendi kohta. Muudetud on ka viisi kuidas browserile tutvustatakse kasutusel olevat keelt ja koodilehti

3. Meta kirjeldus HTML 4.01

```
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<meta name="keywords" content="HTML5,CSS" />
```

4. Siis HTML 5 puhul kasutas

```
<meta charset="ISO-8859-1">
<meta name="keywords" content="HTML5,CSS" />
```

Eelpool mainitud <*meta*> peab jääma <*head*> elemendi sisse.

## *3.2 Veebilehe kehaosa*

Selles osas on kõik dokumendi sisu mis on mõeldud külastajale. Kehaosas <*body*> on uued elemendid, mis asendavad vana tuttavat <*div*>'i ning aitavad luua lehele paremat struktuuri. Kui vanasti tuli olla ettevaatlik, et mitte luua liiga palju *div* elemente, siis nüüd on võimalus kasutada erinevaid semantilisi märgendeid, mis jagavad lehekülje loogilisteks osadeks nagu *header, nav, article, aside, section.* Need loovad struktuuri ja hoiavad kirjutajaid eemale hulgalisest <*div*> kasutamisest. Kuigi kui vajaminevat elementi ei ole võib selle alati abiks võtta.

Edasi <*header*> poole, autor kasutas uut elementi <*hgroup*>. Seda uut elementi kasutatakse kui on vaja grupeerida mingi hulk pealkirju ning sealhulgas ka alampealkirjad (kirjatüübid *h1-h6*). Üks element mis arvatavasti ei leia palju kasutust.

1. *Header* koos *hgroup*'ga autori veebilehel.

```
<header>
 <hgroup>
   <h1>HTML5</h1>
   <h2>Creating webpage using html5</h2>
 </hgroup>
```

Element *<header>* esindab sissejuhatust ja juhendite gruppi, mille alla peaksid kuuluma: logod, ülemine navigatsioon(menüü), otsingu riba jms. Samuti võib *header*'it kasutada ka iga postituse, blogi või artikli alguses.

Edasi menüü juurde, selleks on uus element *<nav>*. *Nav* on mõeldud tähistama navigeerimise blokki. Selles menüüs peaks olema kõik navigeerimiseks vajalik. Uue elemendiga saab kasutada ka uusi standard atribuute nagu näiteks *contenteditable, draggable, hidden* jm. Mis lubavad kasutajal muuta sisu, lohistada teksti ning peita menüüd. *Nav*'i on võimalik paigutada ka *footer*, milleks on navigatsiooni jalus, seda kasutatakse üldjuhul ajaveebides.

2. Navigatsiooni blokk, vasakult paremale

```
 <nav>
   <ul>
     <li><a href="index.html">Home</a></li>
     <li><a href="elements.html" target="main">Elements</a></li>
     <li><a href="contact.html" target="main">Contact</a></li>
     <li><a href="figure.html" target="main">Figure</a></li>
   </ul>
 </nav>
```

3. Varem võis luua tabeli ning sellel anda väärtused

```
<table width="670"  height="700" title="header" border="1" align="center"
cellpadding="0" bgcolor="#FFFFFF">
   <ul>
     <li><a href="index.html">Home</a></li>
     <li><a href="elements.html" target="main">Elements</a></li>
     <li><a href="contact.html" target="main">Contact</a></li>
     <li><a href="figure.html" target="main">Figure</a></li>
   </ul>
</table>
```

Siis oli oht, et neid tabeleid koguneb palju ja leht muutub tänu omaduste andmisele kirjuks. Enam ei lasta sellist valideerimisets läbi.

Uus element <*section*> on mõeldud jagama lehekülje loogilisteks tükkideks, mille sisse saab paigutada <*article*> elementi. Artikkel on kui eraldiseisev osa lehe põhiaknas ehk *section*'is. See mõeldud sisaldama uudist, artikklit, ajaveebi postitust, kommentaare ja muud sellist.

4. *Article sectioni* sees

```
<section>
        <article>
        HTML5 on ülemaailmse veebi tuumikkeele HTML viies põhiredaktsioon...
        </article>
</section>
```

Vasakpoolse menüü puhul ei kasutanud autor mitte *nav* silti vaid *section'*i, kuna loob uue kategooria „application programming interface" jaoks, kuna see pole otseselt seotud elementidega vaid liidestega.

5. Eraldiseisev *section* uue kategooria jaoks

```
<section>
 <h1>Categories</h1>
   <ul>
      <li><a href="audio.html" target="main">Audio </a></li>
      <li><a href="video.html" target="main">Video </a></li>
      <li><a href="#audio" target="main">Drag n Drop </a></li>
      <li><a href="address" target="main">Elements </a></li>
      <li><a href="address.html" target="main">Address </a></li>
   </ul>
</section>
```

Palju kõneainet on tekitanud ka silt <*aside*>. On levinud tõlkest lähtuv arusaam, et seda märgendit peaks kasutama külgriba loomisel, kuid see on väär, kuna seda tööd teeb juba *nav* element ise. Seda tuleb kasutada selleks, et tuua artikklist või ajaveebist välja täpsustavaid andmeid, märkmete tegemiseala ja muud.

Autor proovis sellist lahendust, et võtis lõigu ühest artiklist ja tõi *aside* abil esile loo tähtsaima osa. Et tuua ta nähtavalt esile, määras stiili, suurema kirja, rohelise värvuse ning positsiooni artikli paremas nurgas.

5. Element *aside* artikklis, näite võib leida lisades

```
<article>
While companies from Google to Apple to Microsoft voice their ardent support for HTML5
  <aside style="font-size:larger;
        font-style:italic;
        color:green;
        float:right;
        width:120px;">
   W3C Says HTML5 Isn't Ready for the Web!
  </aside>
.. and developers rush to show off the fun tricks it can do, those who actually oversee HTML5 are telling the world to cool their britches ....
</article>
```

Element <*address*> esindab lähima autori või postitaja kontaktandmeid. Informatsiooni kaasautorite kohta või muud sellist. Seda ei tohi kasutada edastamaks näiteks posti või elukoha aadressit.

6. Näiteks, selline *addressi* kasutamine on väär:

```
<address>
   Karl Kaalikas
  Tallinn, Võidu 40-1
  Tel. 111999
</address>
```

7. Õige oleks selline

```
See dokument on kirjutatud järgmiste üliõpilaste poolt:
<address>
   <a href="mailto:us@example.org">Karl Kaalikas</a><br/>
  <a href="mailto:us@example.org">Maali Maasikas</a><br/>
</address>
```

Soovitatavalt võiks see paikneda koos muu informatsiooniga postituse jaluses. Viimase, 2010 eelnõu järgi, ei tohi <*footer*> ega <*address*> elementides kasutada elementi <*header*>.

## *3.3 Veebilehe jalus*

Element <*footer*> nagu nimest võib järeldada, peaks asuma lehekülje allservas, kuid see ei ole kohustuslik, sest *footerit* võib kasutada ka iga artikli või posituse lõpus. Elementi *footer* võib kasutada näiteks autoriõiguse teavet vms. Lisamiseks veebilehele.

1. *Footeri* kasutamine artikklis

```
<article>
<header>
  <h1>Juku lugu..</h1>
</header>
        Ja selleks ajaks kui Juku kooli oli ...
<footer> autori nimi ....</footer>
</article>
```

Enda töös kasutas autor <*footer*>'it siiski lehekülje jaluses. Autor lisas jalusesse lingi, millel klõpsates kontrollitakse kas leht on valideeruv või mitte ning väljastatakse vastav teade. *Footer*'i kasutamine *header*'is on keelatud.

2. *Footer* lehekülje lõpus

```
This is a valid
<aref="http://validator.w3.org/check?uri=http://www.tlu.ee/~vova/web/html5">
HTML5</a>
```

# 4. Vorm ja selle elemendid

Kui varem sai vorme kasutada enamasti JavaScripti või PhP abiga, siis nüüd on HTML5 selleks loonud omad võimalused. Kuigi JavaScriptil ei ole probleemiks, on sellel palju erinevaid moodiseid, kuidas kinnitada ja saata edasi trükitut või valitut, võis see kaasa tuua suuri probleeme. Näiteks, mis saab siis kui kasutajal ei ole JavaScript lubatud? Üheks suurimaks riskiks JavaScripti kasutamisel on tema turvalisus, kuna script ise käivitatakse kasuaja arvutis (client-side), on mõnel puhul võimalik kasutada seda kuritegelike eesmärkide saavutamiseks. See on ka üks põhjustest miks inimesed on otsustanud blokeerida skripte.

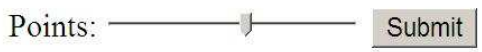
Vorm (*form, web form*) on HTML element, mis võimaldab kasutajal saata erinevaid andmeid veebiserverile. See võimaldab näiteks kasutada otsingukasti, logida või registreerida end veebilehel, anda tagasisidet jne. Veebilehe administraatorid kasutavad näiteks vorme uudiste lisamiseks või muutmiseks. Uued sisend tüübid loodi samuti selleks, et anda neile suuremat väärtust. Autorid ei pea enam kasutama ebavajalike *class'e* ning *id*'sid.

Need uued funktsoonid võimaldavad paremat kontrolli ja valideerimist. Lisandunud on *email, autofocus, autocomplete, url, number, range, search, color* ja muud. Need on mõeldud kasutaja tegevuse lihtsustamiseks, näiteks määratakse ette mille vahel valida saab *list* või siis on lubatud *autocomplete* millega lõpetatakse kirjutatu automaatselt või siis määratakse miinimum ja maksimum numbrite jada.
Kuid kahjuks pakub hetkel ainult veebibrauser Opera ja Google Chrome parimat toetust uutele sisenditele. Neid uusi elemente saab kasutada ka teistes brauserites, kuid kui neid ei toetata, kuvatakse need kui tavalised tühjad teksti väljad.

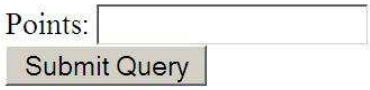
1. Toon näite <*range*>'st

```
<form>
  Points:  <input type="range" name="points" min="1" max="10" />
        <input type="submit" />
</form>
```

2. Ja Google Chromiga saame sellise tulemuse



3. Firefox 4.0 ei toeta sellist sisendit ning kuvab sellise




Ning näide numbritest. Saame samuti määrata *min* ja *max* väärtuse nagu *range's* ning *step*'i ehk sammu, mis näitab mitme numbri kaupa saab üles või alla poole liikuda.

4. Min/Max näide

*Points: <input type="number" name="points" min="1" max="70" step="7" />*

5. Chrome näitab


Ning järekordselt Explorer ning Firefox näitavad vaid tühja lahtrit.

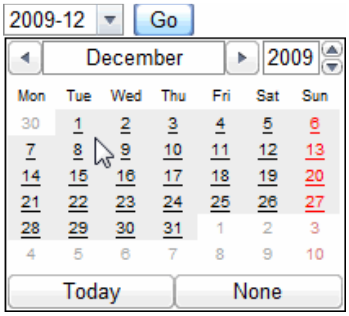
Lõpuks defineeritakse ka HTML5 keelde kuupäeva valimise kontroll. Uusi elementi selles vallas on tegelikult isegi 6: *date, month, week, time, date + time* ning *date + time.* Kuid hetkel toetab seda ainult Opera brauser.

6. Avades Google Chromiga


```
<form>
  <input name="m" type="month">
  <input type="submit" value="Go">
</form>
```

Uueks sisestus meetodiks siis *<input type="month">*, mis tähendab et kuvatakse aasta+kuu. Seda tüüpi saaks kasutada näiteks määramaks krediitkaardi kehtivuse lõppu. Joonisel on näha, et esimene number näitab aastat, teine aga kuud.

7. Märksa ilusam vaade on aga Operaga.



Uuteks atribuutideks *form* elemendile on näiteks eelpool mainitud auto-fookus lehele sisenendes fokusseeritakse kursor automaatselt tühjale väljale, enam ei pea hiirt taga otsima ega toetuma JavaScriptile.

8. Seda saab teha lisades vaid omaduse <*autofocus*>.

```
User name: <input type="text" name="user_name"  autofocus="autofocus" />
```

Atribuut *autocomplete* määrab kas vormi sisestusväljal on automaatteksti funktsioon või mitte. See võimaldab kasutajal eelsalvestatud andmeid kohe väljadele lisada. Funktsioon töötab <*form*>'i ja järgmiste sisend-tüüpidega: *text, search, url, telephone, email, password, datepickers*

Kohatäite tekst (*placeholder*) kuvatakse sisestusvälja sees senikaua, kuni väli on tühi või ei ole kasutatud *autofocus*'t.

9. *Placeholder*'i lisamine

```
<input type="text" name="email"  placeholder="example@msn.com" />
```

10. Saame sellise tulemuse

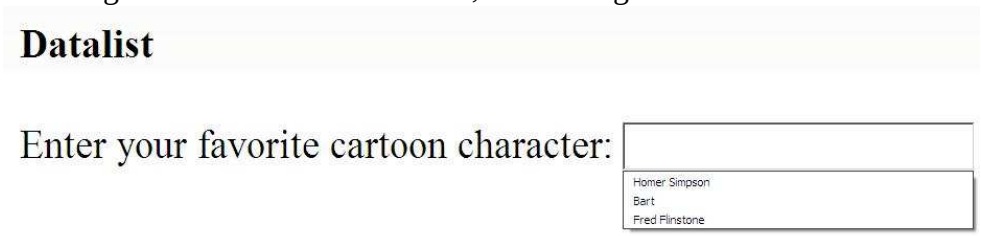

Datalist pakub automaatteksti funktsiooni <*form*> elementidele, see tähendab, et kui kasutaja hakkab sisestama andmeid, siis see võimaldab eelnevalt määratud loetelu kuvada rippmenüüs.

11. Näide *datalist*'st

```
<input type="text" list="characters">
 <datalist id="characters">
  <option value="Homer Simpson">
  <option value="Bart">
  <option value="Fred Flinstone">
</datalist>
```

Mis tähendab, et kui kasutaja lahtrile vajutab või trükkima hakkab, kuvatakse ette võimalikud variandid.

12. Ning tulemus kasutades Firefoxi, kuna Google Chrome seda ei toeta




*Multiple* element lubab kasutajal valida mitu elementi *datalisti'*st. Näiteks kui sul on vorm, mis saadab uudiskirja saidilt, siis kasutades *multiple* elementi vormis on kasutajal lubatud valida mitu saajat korraga. Selleks tuleb lisada vaid üksainus käsk.

13. Elemendi *multiple* lisamine vormi

```
<input name="form_url" id="form_url" type="url" list="url_list" multiple>
```

*Required* ehk nõutud, element tähendab seda, et kasutaja peab kindlasti ära täitma vajaliku lahtri enne saatmis nupule vajutamist.

14. Sisend *required* näide


```
<form>
  <input name="q" required>
  <input type="submit" value="Go">
</form>
```

*Url, email, telephone* ja *color* – kaks esimest märgendit omavad isegi automaatset kontrolli, näiteks kas sisestatud e-mail ning aadress vastavad nõutele.

# 5. Application Programming Interfaces (APIs)

Tegemist on API rakendusliidestega - *Application Programming Interface*. Need on juba programmeeritud koodid mis on valmis suhtlemiseks. See tähendab, et programmeerija ei pea koodi täielikult ise kirjutama vaid ainult mingit osa. Uuteks integreeritud liidesteks on *audio* ja *video, offline web applications, inline editing, storage, drag* ja *drop, web sockets* lisaks veel seotud liidesed nagu *geolocation* ning *messaging*, mis osalevad HTML5s kuid omavad oma spetsifikatsiooni. Siinkohal võtab autor vaatluse alla tema arvates tähtsaimad.

## *5.1 Andmete talletamine*

HTML5 pakub kahte uut meetodit andmete salvestamiseks: local storage ja session storage. Varasemalt oli see saavutatud nn. küpsistega (cookies), kuid need ei sobi suurte andmehulkade jaoks sellepärast, et need antakse edasi serverile iga uue käsuga, mistõttu on see väga aeglane ja ei anna suurt tulemust. Küpsised on piiratud kuni 4 kilobaidini, mis on tegelikult üpriski vähe. Uue rakenduse puhul võib see suurus aga ulatuda megabaitidesse olenevalt brauseri sätetest. Uue liidesega ei edastata andmeid igal uuel nõudmisel, vaid kasutatakse neid siis kui  nõutakse. Iga saidi andmed on salvestatud erinevasse piirkonda ning veeb saab ligi ainult enda salvestatud informatsioonile. Seepärast ongi võimalik salvestada suuri andmehulki mõjutamata veebisaidi tõhusust.

*Session storage* – salvestatakse andmed ühe sessiooni jaoks. Kui aken suletakse kustutatakse kõik väljad. Meetod ise näeb JavaScriptis välja selline:

Võtme ning väärtuse andmine:

*sessionStorage.setItem('võti,'väärtus');*

Väärtusele juurdepääsemine:

*sessionStorage.getItem(võti);*

Väärtuse või võtme eemaldamine:

*sessionStorage.removeItem('võti');*

Kõikide andmete puhastamine:

*sessionStorage.clear();*

1. Sessiooni võtmeks sai 1111

Content loaded from previous sessions:

- sessionStorage: 1111
- localStorage: 666

Local storage – sulgedes brauseri ja pärast seda uuesti avades mäletab see kõiki varem täidetud välju. Andmed salvestatakse iga veebilehitseja kohta eraldi, näiteks kui loendur tõuseb igal uuel korral kui veebilehte külastada Chromiga siis vahetades Firefoxi peale hakkab nendevahel uus salvestamine. Suuresi tuleb see kasuks kasutades nutitelefone ja muid väikseid seadmeid, mis ei jõua andmeid piisavalt kiiresti talletada ning kuvada. Autor kasutas sama koodi kuid *sessionStorage* asemel tuleb lihtsalt *localStorage.* Nagu joonis 1. pealt on näha sai endae võtmeks 666 ning peale veebilehe taaskäivitamiseks sessioon muutub tühjaks kuid lokaalne jääb.

2. Veel üks näide - *localStorage.* Peale igat taaskäivitamist number suureneb

Local storage Visits: 5 time(s).

Refresh the page to see the counter increase.

Close the browser window, and try again, and the counter will continue.

## *5.2 Video ja Audio*

Tänapäeval näidatakse enamus videosid läbi lisandmoodulite nagu näiteks „flash“ või „silverlight“ kuid mitte kõikidel lehitsejatel pole samad moodulid. Lisamoodulite kasutamine veebilehel sidus selle alati kommertstehnoloogiaga. Siinkohal pakub HTML tulevikus välja uue standardi, kus lisasid pole vaja kuna kõik on sisse ehitatud. Suurimad veebilehitsejad, mis on seotud meedia mängimisega, on üle läinud HTML5'le, sest enam ei sõltu nad Adobe tarkvarast. Kui veebilehitseja on HTML5 toega siis nüüd on võimalik videod mängima saada pistikprogrammita, lähemalt http://www.youtube.com/html5

HTML 5 sisaldab endas *<audio>* ning *<video>* elemente ning elemendid on täielikult skriptitavad. See tähendab, et saab puhtalt JavaScript vahenditega valmistada täiesti korraliku muusikapleieri. Samas on uues keeles kõik juhtvahendid (stopp, edasi, tagasi jm.) sisse ehitatud.

Kui varem tahtsid arendajad lisada video oma veebilehele, pidid nad kasutama *object* elementi.

1. Koodirida võis välja näha selline

```
<object width="425" height="344">
 <param name="movie" src="movie.ogg" />
 <param name="allowFullScreen" value="true" />
 <param name="allowscriptaccess" value="always" />
</object>
```

Hullem oli veel see, et lehitseja pidi selle edasi andma kolmandale osapoolele vajaliku pistikprogrammi (plugin) jaoks ning vaatajal pidi olema õige versioon selle nägemiseks.

2. Uus ja märksa lihtsam viis

```
<h1>Video in html5</h1>
<video src="movie.ogg" width="320" height="240" controls="controls">
```

Audio lisamiseks tuleb samuti lihtsalt määrata audiofaili asukoht ja soovitavalt kohe ka *controls* parameeter, mille abil saad heli kontrollida - käivitus, kerimisriba, aeg ja helitugevuse regulaator.

3. Audio lisamine

```
<audio controls>
  <source src="audio.mp3">
  <source src="audio.ogg">
</audio>
```

4. Audio player

## 5.3 Joonistuslõuend

HTML5 spetsifikatsiooniga lisati loendisse <*canvas*> element, mis lubab veebilehele lisada jooniseid, fotosid, animatsioone jm. Joonistuslõuend lubab tuua veebilehtedele rohkem interaktiivsusus diagrammide, uhkete kasutajaliideste ning graafikute näol. Kasutada saab ristkülikuid, kaare, ruutfunktsioonide kõveraid. Effekte annavad varjud, läbipaistvus, kompositsioon ja muu. Elemendiga töötamiseks peab mõistma ja oskama kasutada Javascript'i. Javascripti abil võib veebilehe kasutaja isegi vaba käega lõuendil joonistada.

Järgnevalt demonstreerib autor lühidalt *canvase* funktsiooni, et Javascript teaks, kuhu joonistus lisada, tuleb <*canvas*> elemendile lisada unikaalne *id.*

Ristküliku joonistamiseks on kolm moodust:
*strokeRect*(x, y, laius, kõrgus) - joonistab ristküliku äärise
*fillRect* - joonistab täidetud ristküliku
*clearRect* - joonistab tühja ristküliku, mis nö kustutab alumised kujundid.
Ristküliku värvimiseks kasutame kahte funktsiooni:
*fillStyle* - sisu värv ehk kujundi täide
*strokeStyle* - joone värv ehk ääris.

1. *Canvase* näide

```
<canvas id="myCanvas" width="200" height="200"></canvas>
        <script type="text/javascript">
        var canvas=document.getElementById('myCanvas');
        var ctx=canvas.getContext('2d');
  ctx.fillStyle   = '#00e';
  ctx.strokeStyle = '#f00';
  ctx.strokeRect(30,  60, 150, 50);
  ctx.fillRect(10,  15, 150, 75);
  ctx.clearRect(45,  45, 15, 75);
</script>
```

2. Näide pildina

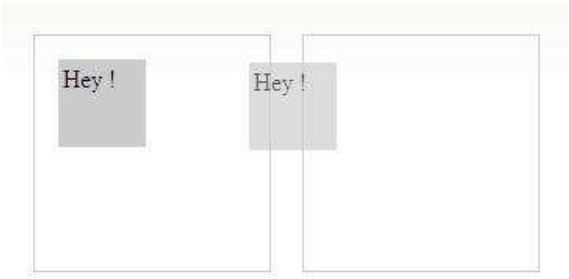
## 5.4 Drag & Drop

*HTML5 Drag-and-Drop API* võimaldab kasutajal elemente hiirega lohistada ja soovitud kohta tõsta. See on võimalik tänu uutele funktsioonidele

- *dragstart* – see läheb käima kui kasutaja hakkab lohistama objekti
- *drop* – sel hetkel kui vabastatakse lohistatav.
- *dragenter* tuleb kasutusse kui tõmmatav liigub üle elemendi, kuid midagi ei toimu kui elemendil pole kuulajat, seda ei ole vaikimisi lubatud.

Loomulikult on funktsioone veel: *dragleave, dragover, dragend*. Drag ja drop on nii mahukas teema, et eraldi kirjutades saaks sellest seminaritöö. Näiteks Google Mail kasutab seda pikemat aega võimaldades kasutajal kirju tõsta erinevatesse kaustadesse.

1. Liigutame elementi ühest „kaustast“ teise.



# Kokkuvõte

Käesoleva töö eesmärgiks oli uurida võimalusi mida pakub uus HTMLi viies redaktsioon. Tuua näiteid uutest elementides, vormidest, sisestusvälja tüüpidest ning vaadata mida suudavad teha uued integreetitud programmeerimise liidesed ning luua demonstratiivne veebi leht, mis katsetab uusi vahendeid ja võimalusi.

Uus standard toob kaasa uue põlvkonna tööriistad nagu selgus on kood tehtud lihtsamaks, disaini elemendid on täielikult jäetud CSSi (Cascading Style Sheets) kanda. Uued struktuurielemendid *aside, footer nav* ning *article* toovad rohkem semantilist väärtust ning tegid autori arvates lehe ülesehituse loogilisemaks ja koodi kergemini loetavamaks. Uued vormi elemendid ja atribuudid annavad rohkem võimalusi kasutajale, teevad vormi täitmise lihtsamaks ning suhtlemise saidiga kiiremaks. Lõpuks astub HTML sammu eemale JavaScriptist ning mis enne oli saavutatud pika koodiga, tehakse ära nüüd oma vahenditega.

Antud tööd saab täiendada nutikate CSS 3.0 ning JavaScripti lahendustega. Kuna lehekülje disainipool polnud autori prioriteediks, jäi see küllaltki lihtsaks. Ühildades CSS ning Javaskript saaks kokku juba Adobe tarkvarale sarnaseid lahendusi. Internet pole enam arvutis vaid ka mobiiltelefonides, kus uus markeerimiskeel saab pakkuda tugevamat alust, kuna ei toetu videote vaatamisel ja andmete salvestamisel lisamoodulitele. Üle saja näite, mida võib luua nende komponentide koos kasutamisel, võib leida aadressil http://www.chromeexperiments.com/

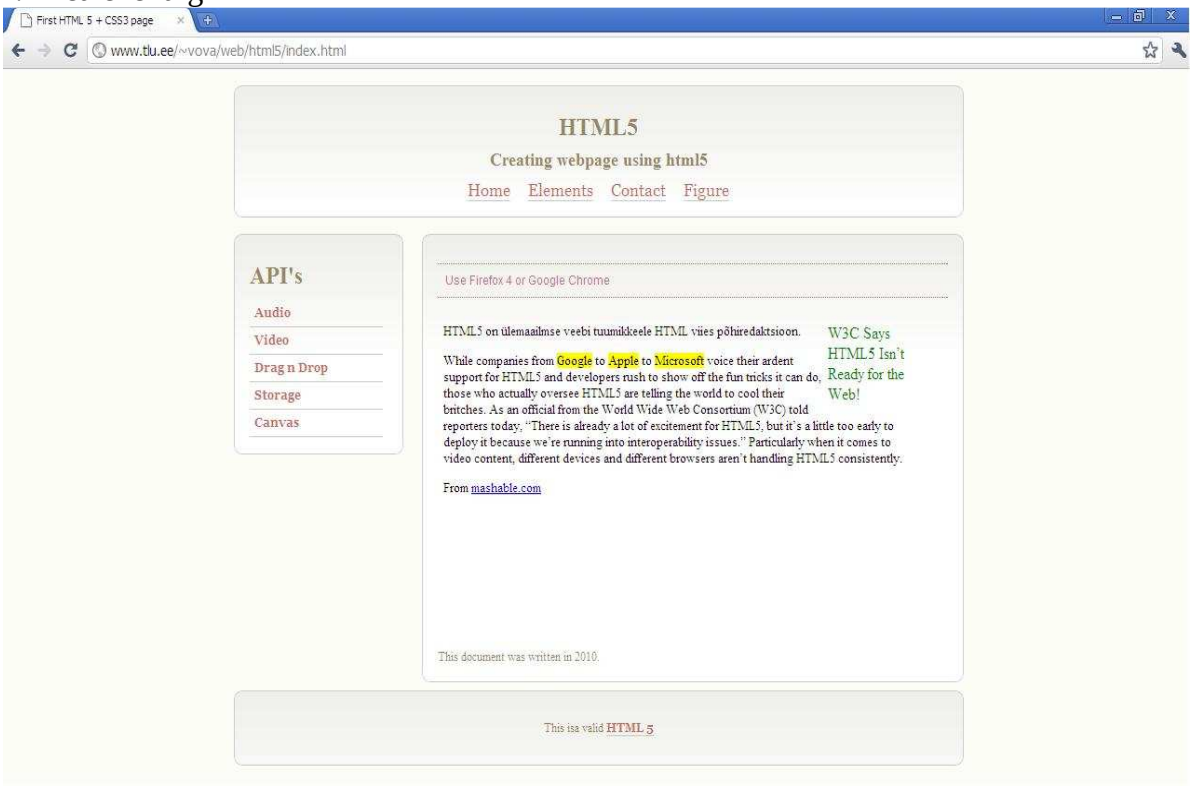
# Kasutatud kirjandus


1. Lawson, B. & Sharp, R. (2010). Introducing HTML5. Berkeley: New Riders.
2. Munner, M. (2010). HTML 5, mis see on? URL http://www.interneti.info/artiklid/html5/ (10.10.2010)
3. Pilgrim, M. (2010). Dive into HTML5. URL http://diveintohtml5.org/ (1211.2010)
4. W3Schools (2010). URL http://w3schools.com/html5/default.asp (22.11.2010)
5. Hickson, I. (2010). A vocabulary and associated APIs for HTML. URL http://www.w3.org/TR/html5/ (3.12.2010)
6. Kesteren A.(2010). HTML5 differences from HTML4. URL http://www.w3.org/TR/html5-diff/ (8.12.2010)
7. Google (2010). URL http://www.html5rocks.com/ (1.10.2010)

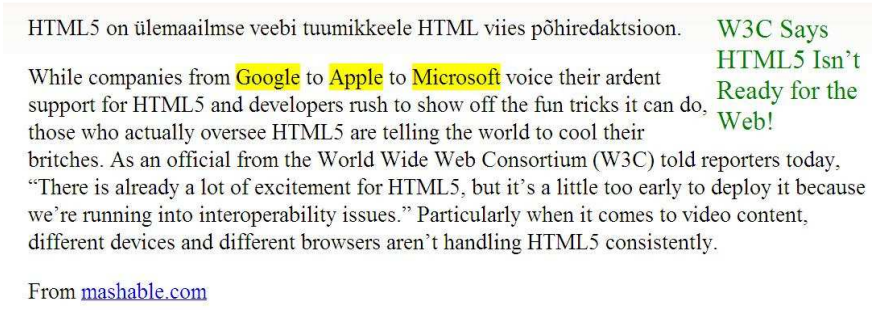
# Lisad

Autor toob välja tööst välja jäänud pildid ning kasutab selleks Google Chrome veebilehitsejat. Lehekülg asub aadressil http://www.tlu.ee/~vova/web/html5/

1. Pealehekülg

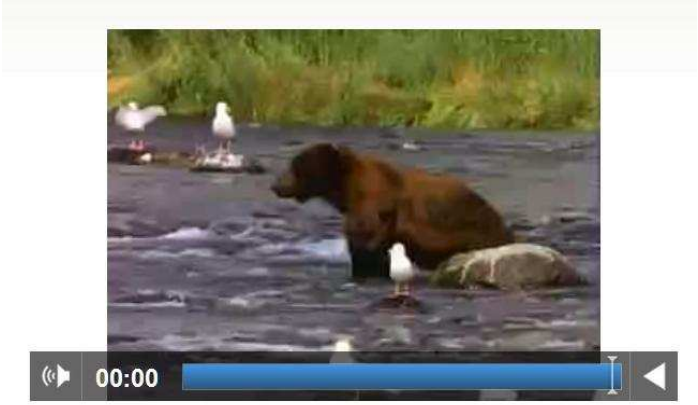



2. Roheliselt, element *aside*

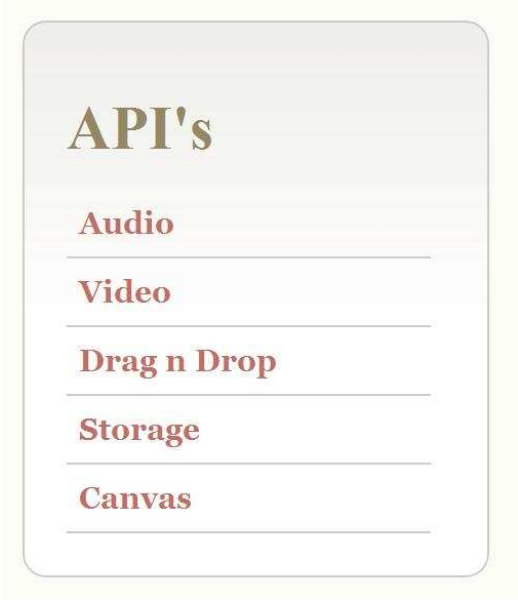



3. Elemendi *nav* (navigatsiooni menüü) kasutus

4. Video




5. Section




6. Selline näeb välja IE8ga, enne skripti lisamist