

Tallinna Reaalkool

Veebipraktikad Eesti koolide veebilehtedel

Uurimistöö

Erko Risthein

11b

Juhendaja: õp Inga Petuhhov

Tallinn, 2009

Sisukord

Sissejuhatus	4
1. Veebistandardid	6
1.1. HTML ehk hüperteksti märgistuskeel	6
1.2. CSS ehk kaskaadlaadistik	7
1.3. XHTML ehk laiendatav hüperteksti märgistuskeel	7
1.4. Dokumenditüübid	10
1.5. Uurderežiim (quirks mode).....	11
1.6. Valideerimine ja selle vajalikkus	12
1.7. Veebistandardid ja põhjused nende järgimiseks	12
1.8. Tähekodeeringud	13
1.9. JavaScript ja AJAX.....	15
1.9.1. JavaScript	15
1.9.2. AJAX.....	15
1.9.2. Probleemid JavaScripti ja AJAXiga.....	16
2. Uurimus	17
2.1. Varasemad uuringud	17
2.2. Uurimuse ülesehitus.....	17
2.3. Uurimuse läbiviimine	18
2.4. Uurimuse tulemused	19
2.4.1. Dokumenditüüp.....	19
2.4.2. Tähekodeering.....	20
2.4.3. (X)HTML vigade esinemissagedus.....	21
2.4.4. Kõige levinumad (X)HTMLi veateated	22
2.4.5. CSSi vigade esinemissagedus	24
2.4.6. Kõige levinumad CSSi veateated.....	25
2.4.7. JavaScripti ja AJAXi kasutatavus	26
Kokkuvõte	27
Kasutatud materjalid.....	29
Lisa 1 Neti.ee kataloogipuu parsimisskript	31

Lisa 2 Dokumenditüüp	32
Lisa 3 Tähekodeering	33
Lisa 4 (X)HTML vigade esinemissagedus	34
Lisa 5 Kõige levinumad (X)HTML veateated.....	35
Lisa 6 CSSi vigade esinemissagedus.....	36
Lisa 7 Kõige levinumad CSSi veateated	37

Sissejuhatus

Veebistandarditest on räägitud juba aastaid, kuid sellegipoolest on jäänud mulje, et plahvatuslikku arengut veebistandardite järgimise osas ei ole olnud, hoolimata faktidest, et veebistandardite mittejärgimine:

- takistab ja aeglustab veebilehtede arendusprotsessi
- ei garanteeri, et antud veebileht ka mõne aasta pärast uutes veebilehitsejates täpselt samamoodi töötab kui arenguhetkel turulolevates veebilehitsejates
- aeglustab veebilehe allalaadimist ja *renderdamist*
- muudab lehed ligipääsetamatuks puudega inimestele
- muudab lehe läbi otsingumootorite raskemini leitavateks
- takistab lehe kujundust ümber kohandada erinevatele seadmetele nagu mobiiltelefonid ja pihuarvutid

Käesoleva uurimistöö eesmärgiks on välja selgitada kui palju ja milliseid tehnoloogiaid Eesti koolide veebilehtedel kasutatakse ja kui suurel määral kasutatakse neid vastavuses veebistandarditega. Vaatluse all on Eesti koolide veebilehed, kuna uurimistöö autor on ise õpilane ning on varasemalt välja arendanud Tallinna Rahumäe Põhikooli veebilehe ning uurimistöö koostamise hetkel arendab ja haldab Tallinna Reaalkooli 125. lennu B klassi veebilehte. Autori andmetel ei ole varasemalt seda sihtgruppi eraldi uurimise alla võetud.

Uurimistöö hüpoteesid:

- Valdav osa lehti ei defineeri korrektselt dokumenditüüpi
- Valdav osa lehti ei kasuta standarditele vastavat (valideeruvat) (X)HTMLi ja CSSi
- Universaalset tähekodeeringut UTF-8 kasutatakse minimaalselt
- JavaScripti ja AJAXit kasutatakse ülemäära palju ehk leht pole täielikult funktsioneeriv JavaScripti mittetoetavatel veebilehitsejatel

Uurimistöö eesmärgi saavutamiseks viiakse läbi automatiseeritud uuring 884 Eesti kooli veebilehe seas ja kogutakse nende kohta statistilist informatsiooni.

Uurimistöö koosneb kahest osast. Töö esimene, teoreetiline osa annab ülevaate erinevatest veebitehnoloogiatest, veebistandardite olemusest ja seletab lahti erinevad töös kasutatavad terminid. Töö teine, praktiline osa räägib lühidalt varasemastest samalaadsetest uurimustest, käesoleva uurimuse ülesehitusest, uurimuse läbiviimisest ja toob välja uurimuse tulemused, koos nende analüüsiga.

Täna siinkohal Rene Saarsood, kes oma uurimistöö jaoks väljatöötatud tarkvara enda veebilehele tasuta üles riputas. Suurim tänu kuulub töö juhendajale Inga Petuhhovile, kes töö valmimise käigus toeks oli ja omapoolsete ideede ja nõuga aitas.

1. Veebistandardid

1.1. HTML ehk hüpertexti märgistuskeel

HTML (*HyperText Markup Language*) ehk hüpertexti märgistuskeel on mõeldud veebilehtede vormingu defineerimiseks. HTML koosneb siltidest (*tags*), millega määratakse lehel esinevate elementide tüüpe. Silte on kahte sorti – algussilt ja lõpusilt. Nende vahele jääv osa vormindatakse vastavalt sildi definitsioonile. Näiteks kui tahaksime defineerida pealkirja siis saaksime seda teha *h1* sildiga (*h1* ehk *heading 1* – kõrgeima astme pealkiri). Välja näeks see nii:

```
<h1>Pealkiri</h1>
```

`<h1>` on algussilt ja `</h1>` on lõpusilt.

HTMLis on defineeritud ka palju muid silte. Näiteks madalama astme pealkirjad on järjest *h2*, *h3*, *h4*, *h5*, *h6*. Lõigud defineeritakse *p* (*p* ehk *paragraph* – lõik) sildiga, hüperlingid *a* (*a* ehk *anchor* – ankur), pildid *img* (*img* ehk *image* – pilt) jne.

Lisaks lehe struktuuri väljendamiseks mõeldud siltidele on HTMLis olemas ka kujunduslikud sildid. Sinna alla kuuluvad näiteks *font*, *center*, *b* (*bold*), *i* (*italic*), *u* (*underline*) jne. Praktika aga näitas ajapikku, et nende siltide kasutamine ei ole mõistlik tegevus. Näiteks kui ühe lehe alamlehed on kõik samamoodi kujundatud kasutades HTML silte ning mingil hetkel tekib vajadus lehe kujundust muuta, siis tuleks kõik alamlehed ükshaaval käsitsi läbi käia ja vastavaid kujunduselemente muuta. Lahendus sellele ja ka mitmetele teistele probleemidele oli kõigi kujunduslike siltide eemaldamine ja uue tehnoloogia välja töötamine, milleks oli CSS. Tagasiühilduvuse säilitamiseks jaotati HTML kolmeks erinevaks dokumenditüübiks (*Strict*, *Transitional*, *Frameset*), millest saab lähemalt lugeda peatükis 1.4. Dokumenditüübid. (HTML Introduction)

1.2. CSS ehk kaskaadlaadistik

CSS (*Cascading Style Sheets*) ehk kaskaadlaadistik on mõeldud veebilehtedel esinevate elementide kujunduse defineerimiseks. Ehk teisisõnu HTMLiga määratakse ära mis osa dokumendist on navigatsioonimenüü, sisuosa, pilt, pealkiri, lõik jne, kuidas neid elemente visuaalselt lehele kuvatakse on aga juba CSSi pärusmaa. CSS-kood on tavaliselt kirjutatud eraldi faili, millega kõik üksikud HTML-dokumendid seotakse. Sellise meetodi eeliseid on mitmeid. Näiteks ei pea nüüd enam terve lehe kujunduse muutmiseks iga alamlehte üksikhaaval redigeerima hakkama, vaid saab teha vastavad muudatused tsentraalses CSS failis ning kõik HTML-dokumendid, mis selle failiga seotud on uuenevad automaatselt. Lisaks sellele on veel mitmeid eeliseid, mõned neist on välja toodud peatükis 1.7. Veebistandardid ja põhjused nende järgimiseks. (Introduction to CSS)

1.3. XHTML ehk laiendatav hüpertexti märgistuskeel

XHTML (*Extensible HyperText Markup Language*) ehk laiendatav hüpertexti märgistuskeel on HTMLi reformatsioon, mis paneb ta vastama XMLi (*Extensible Markup Language*) ehk laiendatava märgistuskeele palju rangematele süntaksireeglitele. (Common ideas...) Märkimisväärseimad muudatused:

Elemendid peavad korrektselt üksteise sees pesitsema

HTMLis võivad elemendid olla ebakorrektselt üksteise sees:

```
<b><i>See tekst on paks ja kaldkirjas (bold ja italic)</b></i>
```

XHTMLis peavad elemendid olema korrektselt üksteise sees:

```
<b><i>See tekst on paks ja kaldkirjas (bold ja italic)</i></b>
```

Elemendid peavad alati lõpetatud olema

Vale:

```
<p>See on lõik  
<p>See on veel üks lõik
```

Õige:

```
<p>See on lõik</p>  
<p>See on veel üks lõik</p>
```

Tühjad elemendid peavad samuti alati lõpetatud olema

Vale:

```

```

Õige:

```

```

Kõikide elementide ja atribuutide nimed peavad olema väikeste tähtedega

Vale:

```
<BODY>  
  <P>See on lõik</P>  
  <IMG SRC="mina.jpg" ALT="Pilt minust">  
</BODY>
```

Õige:

```
<body>  
  <p>See on lõik</p>  
    
</body>
```

Kõik atribuutide väärtused peavad olema ümbritsetud jutumärkidega

Vale:

```
<table width=100%>
```

Õige:

```
<table width="100%">
```

Atribuute ei tohi minimeerida

Vale:

```
<input checked>  
<input readonly>  
<input disabled>  
<option selected>
```

Õige:

```
<input checked="checked" />  
<input readonly="readonly" />  
<input disabled="disabled" />  
<option selected="selected" />
```

Kõikidel elementidel peab olema üksainus juurelement

Kõik XHTML elemendid peavad asuma <html> juurelemendi sees. Iga dokumendi põhistruktuur peab olema:

```
<!DOCTYPE Dokumenditüüp>  
<html>  
  <head>  
    <title>Pealkiri</title>  
  </head>  
  <body> ... </body>  
</html>
```

1.4. Dokumenditüübid

Dokumenditüübiga näidatakse missugust veebistandardit järgides on antud dokument koostatud. See on mõeldud selleks, et internetilehitsejad oskaksid lehte õige standardi järgi kuvada. Dokumenditüüp määratakse alati dokumendi kõige esimesel real ja sellega viidatakse vastava standardi dokumenditüübi definitsioonile (*Document Type Definition* ehk DTD), mis asub W3C veebiserveris.

W3C poolt on praegu soovitatud kolm erinevat dokumenditüüpi: HTML 4.01, XHTML 1.0 ja XHTML 1.1. HTML 4.01 ja XHTML 1.0 jagunevad omakorda veel kolmeks: *Strict*, *Transitional* ja *Frameset*. *Strict DTD* tähendab seda, et kõik kujunduslikud elemendid ja raamid on eemaldatud, *Transitional DTD* sisaldab endas ka kujunduslikke elemente, kuid siiski keelab raamid, *Frameset DTD* on täpselt sama, mis *Transitional*, aga ei keela raame.

HTML 4.01 Strict

Selles dokumenditüübis on defineeritud kõik HTML elemendid, välja arvatud need, mis on mõeldud lehe kujunduse väljendamiseks (nagu näiteks font). Raamid ei ole lubatud.

HTML 4.01 Transitional

Selles dokumenditüübis on defineeritud kõik HTML elemendid, kaasa arvatud need, mis on mõeldud lehe kujunduse väljendamiseks. Raamid ei ole lubatud.

HTML 4.01 Frameset

See dokumenditüüp on võrdväärne *HTML 4.01 Transitionaliga*, kuid siin on lubatud raamid.

XHTML 1.0 Strict

Selles dokumenditüübis on defineeritud kõik HTML elemendid, välja arvatud need, mis on mõeldud lehe kujunduse väljendamiseks (nagu font). Raamid ei ole lubatud. Kood peab vastama XMLi süntaksireeglitele.

XHTML 1.0 Transitional

Selles dokumenditüübis on defineeritud kõik HTML elemendid, kaasa arvatud need, mis on mõeldud lehe kujunduse väljendamiseks. Raamid ei ole lubatud. Kood peab vastama XMLi süntaksireeglitele.

XHTML 1.0 Frameset

See dokumenditüüp on võrdväärne *XHTML 1.0 Transitionaliga*, kuid siin on lubatud raamid.

XHTML 1.1

See dokumenditüüp on võrdväärne *XHTML 1.0 Strictiga* (minimaalsete erinevustega), kuid olles modulaarne võimaldab juurde lisada erinevaid mooduleid. (HTML doctype definition)

1.5. Uurderežiim (*quirks mode*)

Kui dokumenditüüp on määratlemata, ei vasta korrektsele süntaksile või viitab tundmatule standardile, siis lülitub internetilehitseja uurderežiimi (*quirks mode*) ehk taganeb vabamatele süntaksireeglitele. See režiim on loodud selleks, et oleks olemas tagasiühilduvus vanemate veebilehtedega, mille kirjutamishetkel ei olnud veel veebistandardeid olemas või ei peetud nende järgimist oluliseks. Uurderežiim on samuti palju leebem kõigi koodis esinevate vigade suhtes ning veebilehitseja püüab ära arvata, mida tarkvaraarendaja on oma vigase koodiga püüdnud väljendada.

Praktikas aga kasutavad paljud veebiarendajad veebilehitsejate uurderežiimi ebaetiliselt ära. Jäetakse meelega dokumenditüüp määramata ning ei vaevuta oma koodi valideerima. Tihti kontrollitakse oma lehekülge vaid mõne üldtuntud veebilehitsejaga, nagu näiteks Internet Explorer. Kuna aga uurderežiim ei ole mingi kindel standard, on ta realiseeritud igas veebilehitsejas erinevalt ning leht võib teistes lehitsejates hoopis teistsugune välja näha või sootuks kasutuskõlbmatu olla.

Tihti arvatakse, et uurderežiimis võib lehti arendada küll, kui ta niiviisi kirjutada, et kõik lehitsejad teda ühte moodi näitavad ja leht igal pool kasutatav on. See mõtteviis on paraku aga vale, sest nii ei ole garanteeritud, et leht ka tulevaste veebilehitsejatega samasugune välja näeb. Kui aga kirjutada leht kasutades veebistandardeid ja mõnda ülalmainitud W3C

soovitatud dokumentitüüpi, võib täiesti kindel olla, et ükski tulevikus välja tulev veebilehitseja ei näita lehte vigaselt ja leht on kasutuskõlblik ka mitmete aastate pärast. (Wikipedia 2009 s.v. Quirks mode)

1.6. Valideerimine ja selle vajalikkus

Valideerimine on automatiseeritud protsess, mille käigus tehakse kindlaks kas antud dokument on korrektse süntaksiga ja vastab seal kasutatud programmeerimiskeele standarditele või mitte. Seda võib võrrelda tavakeele õigekirjakontrolliga.

Valideerimine on väga tähtis osa veebiarendusest. See toob selgelt välja kõik vead, mida võibolla esmapilgul ei märka. Paraku aga paljud veebiarendajad ei valideeri oma lehti. See on suuresti veebilehitsejate süü, sest enamus lehitsejaid püüab ebastandardset koodi tõlgendada nii hästi kui nad oskavad (püüdes ära arvata, mida autor oma vigase koodiga võis tahta esitada). Seetõttu paljud arendajad ei teagi, et nende kood ei ole korrektne ega vasta standarditele. Seega peaksid veebilehitsejad praeguse viisi asemel hoopis kuvama veateateid, mis arendajat vigadest teavitaks ja laseks need parandada. (Johansson 2006: 3)

1.7. Veebistandardid ja põhjused nende järgimiseks

Veebistandardid on W3C (*WWW 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. Kõik varem nimetatud dokumentitüübid ja CSS on veebistandardid.

Palju veebiarendajad on täiesti teadlikud veebistandarditest, kuid sellegipoolest keelduvad neid järgimast. Tihti tuuakse välja põhjendusi nagu „see on liiga raske/ajakulukas“, „see töötab ju ka ilma standardeid järgimata“, „programmid, mida ma kasutan, genereerivad ebastandardset koodi“. Lihtne on millegi kasutamisest loobuda, kui selles nähakse ainult negatiivseid külgi ja positiivsete külgede juures pigistatakse silm kinni. Tegelikuses on aga veebistandardite järgimiseks mitmeid vägagi mõjuvaid põhjuseid:

- **Lihtsam arendus ja hilisem haldus:** Semantilist ja loogiliselt struktureeritud koodi on lihtsam ka järgmistel tarkvaraarendajatel mõista, hallata ja vajadusel laiendada. Hoitakse kokku aega ja raha.

- **Garanteeritud ühilduvus tulevaste veebilehitsejatega:** Järgides ettekirjutatud standardeid on garanteeritud, et tulevased veebilehitsejad mõistavad varem kirjutatud koodi ja oskavad veebilehte korrektselt kuvada. See tagab, et külastajatel ei esine mingeid ebameeldivusi lehe kasutamisel.
- **Kiirem veebilehe allalaadimine ja kuvamine:** Standardite järgimine tähendab ka väiksemat koodi mahtu, mis omakorda tähendab väiksemaid failisuurusi ja kiiremat andmeedastust. Samuti *renderdavad* uudsed veebilehitsejad standardeid järgivaid veebilehti kiiremini.
- **Parem ligipääsetavus:** Kui on kasutatud semantiliselt korrektset (X)HTMLi, kus struktuur on eraldatud kujundusest, siis on lihtsam ekraanilugejatel lehe sisu tõlgendada.
- **Kõrgemad kohad otsingumootorite tulemuste seas:** Lehe sisu ja kujunduse eraldamine ja sisu semantiliselt korrektne esitamine muudab lehe otsingurobotite jaoks arusaadavamaks, mis tõstab teie lehe otsingutulemustes kõrgemale ja toob lehele rohkem külastajaid.
- **Lihtsam ümberkohandamine:** Semantiliselt korrektselt esitatud dokumendi saab lihtsalt ümber kohandada printimise ja alternatiivsete veebilehitsejate jaoks nagu näiteks mobiiltelefonid. Lehe sisu tuleb lihtsalt siduda uue CSS failiga. Samuti on kogu lehe kujundus ühtne: iga lehe puhul võetakse kõik kujunduselemendid ühest failist. (Johansson 2006: 3)

1.8. Tähekodeeringud

Iga täht arvutis on kodeeritud mingis tähekodeeringus. Kuna arvutimälus hoitakse andmeid vaid ühtede ja nullidena, siis tuleb iga täht mingi seaduspärasuse alusel ühtedeks ja nullideks teisendada, enne kui neid üldse kuskile talletada saab. Neid seaduspärasusi nimetataksegi tähekodeeringuteks. Tähekodeeringutel on alati olemas definitsioonitabelid, mis näitavad, mis numbrile mingi täht vastab. Üks esimesi laialdase kasutuse jaoks loodud standardeid on ASCII (*American Standard Code for Information Interchange*). Selles tähekodeeringus defineeritakse iga täht 8-bitina ehk ühe baidina. Ühe baidi sees saab ühtedest ja nullidest (binaarkoodis) moodustada 256 erinevat kombinatsiooni, seega on ASCII tabeli pikkuseks 256 märki. See oli piisav suurus aegadeks, kui arvutid levisid enamasti vaid Ameerikas ja kui polnud veel leiutatud internetti.

Välismaal kasutati tavaliselt sellisel puhul mõnda ASCII tabeli teisendit. Nende puhul olid tavaliselt esimesed 128 tähte (inglise alfabeet ja enamkasutatavad sümbolid) identsed ja järgmised 128 kasutati ära spetsiifiliste keelesümbolite jaoks. Erinevate keelte jaoks loodi 16 erinevat ISO (*International Organization for Standardization*) 8859 standardit. Näiteks eesti keeles kasutatavad tähed Š, š, Ž, ž on olemas ISO 8859-15 tähekodeeringus.

Personaalarvutite levikuga ja interneti leiutamisega tekkis aga vajadus kasutada palju rohkem erinevaid tähemärke, kui 256. Ainuüksi Hiina hieroglüüfe on juba mitu tuhat. Selle probleemi lahendamiseks loodi mitmeid uusi tähekodeeringuid, kui kuid ükski neist ei saanud laialt kasutatavaks standardiks, kuna kui tähti hakati rohkem kui ühes baidis talletama, suurenesid tekstide talletamiseks vajalikud andmemahud mitmekordselt. Seda aga ainult inglise alfabeeti kasutavad kasutajad ei soovinud ja jätsid ASCII tabeli kasutamist.

Selle probleemi lahenduseks loodi universaalne tähekodeering UTF-8, mis koosneb kuni kaheksast baidist. UTF-8 erineb eelmistest tähekodeeringutest selle poolest, et ta ei talleta tähti fikseeritud arvus baitides. Esimesed 128 tähte talletatakse ühes baidis, edasi aga baitide arv suureneb ükshaaval kuni kuue baidini. Sellega kaasneb huvitav kõrvalnähtus – nimelt on UTF-8 täielikult tagasiühilduv ASCII tähekodeeringuga, kuna nende esimesed 128 tähesümbolit on samad. Rohkem polegi vaja, sest kogu inglise alfabeet ja enamkasutatavad sümbolid asuvad esimeses 128 kombinatsioonis. Järgnevad sümbolid (128-256) varieeruvad erinevates ASCII tabeli teisendites. Sellega välditi tagasiühilduvuse probleeme ASCII-ga ning samuti andmemahu probleeme.

UTF-8 töötati välja nii, et see sisaldaks kõiki maailmas enimkasutatavaid alfabeete ja sümboleid ja lisati ka mõned vähemtuntumad, et saavutada tõeliselt universaalne tähekodeering, mis ei seaks mitte mingisuguseid piiranguid. Tabel on üles ehitatud sedasi, et kõik enamkasutatavad alfabeedid ja sümbolid on tabeli alguses, kuna alguses on sümbolite andmemahud väikesemad kui lõpus. Tabeli lõpus suurenevad sümbolite mahud kuni kuue baidini ning sinna on seetõttu paigutatud haruldasemad tähestikud ja sümbolid. (Spolsky: 2003)

UTF-8 on hetkel tuntuim ja soovitatavaim tähekodeering. Isegi kui kavatakse koostada leht mõnes spetsiifilises keeles (näiteks eesti keeles), siis on soovitatav ikkagi kasutada spetsiifilise tähekodeeringu asemel universaalset, sest kui tulevikus tekib näiteks vajadus teha

leht mitmekeelseks, on see täiesti valutu protsess. Samuti on raske ette näha, millal võib lehel olla tarvis kuvada näiteks mõnda venekeelset tsitaati vms.

1.9. JavaScript ja AJAX

1.9.1. JavaScript

JavaScript on maailma populaarseim veebi skriptimiskeel, mis on välja töötatud eesmärgiga lisada veebilehtedele interaktiivsust. JavaScripti abil saab veebilehtedele lisada dünaamilist sisu, valideerida vormide andmeid enne serverisse päringu saatmist, külastajate internetilehitseja, ekraaniresolutsiooni jms kohta infot saada, küpsiseid hallata, sündmustele (*events*) reageerida jpm. See kõik toimub külastaja veebilehitsejas, ilma et serverisse ühtegi päringut saadetakse ehk teisisõnu on tegemist kliendipoolse skriptikeelega.

JavaScripti aetakse tihti segamini Java keelega. Tegelikuses aga pole neil keelteil peale nime ja sarnase süntaksi mitte midagi ühist. Nad erinevad nii kontseptsioonilt kui ka disainilt. Java on palju laiahaardelisem, võimsam ja raskem keel kui JavaScript. Javat võib võrrelda keeltega nagu C ja C++. (Introduction to JavaScript)

1.9.2. AJAX

AJAX (*Asynchronous JavaScript and XML*) ei ole eraldi programmeerimiskeel, vaid kombinatsioon olemasolevatest tehnoloogiatest. AJAX lisab JavaScriptile võimaluse ka andmeid serverist pärida. Andmeid päritakse asünkroonselt ehk ilma lehte uuesti laadimata kasutades *XMLHttpRequest* objekti. Andmeid edastatakse XML (*Extensible Markup Language*) keeles. Hoolimata nimest ei ole tegelikuses vaja kasutada JavaScripti ega XMLi. Samuti ei pea andmete edastus olema asünkroonne. Seetõttu on akronüüm AJAX muudetud üldisemaks nimetuseks Ajax. Selle alla kuuluvad siis ka muud veebi skriptimiskeeled peale JavaScripti (näiteks ActionScript, JScript) ja muud andmete edastamiseks mõeldud süntaksid (näiteks JSON, YAML). (Wikipedia 2009 s.v. Ajax (programming))

Kuna tänu Ajaxile on võimalik lehe erinevaid osasid uuendada ilma tervet lehte uuesti laadimata, vähenevad üle interneti laetavad andmemahud ja seetõttu laevad ka lehed kiiremini. Kui veel juurde lisada JavaScriptiga näiteks laadimise animatsioone saame kokku palju kasutajasõbralikuma, mugavama ja kiiremini funktsioneeriva keskkonna.

AJAXi tegi kuulsaks Google aastal 2005, kui ta hakkas seda tehnoloogiate kombinatsiooni enda erinevates internetiteenustes kasutama. Mõni aeg hiljem ehitas Google kogu oma e-posti teenuse Gmaili veebiliidese AJAXi põhiselt üles.

1.9.2. Probleemid JavaScripti ja AJAXiga

Liigselt JavaScriptil põhineval veebilehel võib tekkida mõningaid probleeme. Näiteks osad veebilehitsejad ei toeta JavaScripti ja osad kasutajad keelavad oma veebibrauserites JavaScripti kasutamise. Kui aga lehekülg on täielikult üles ehitatud JavaScriptil (ja AJAXil) põhinedes, siis ei ole see lehekülg nendele kasutajatele kättesaadav. Seetõttu on soovitatav leht üles ehitada siiski traditsioonilisel staatilisel viisil ning alles seejärel sinna juurde lisada dünaamilisi JavaScripti ja AJAXi elemente. Sellega tagatakse, et veebileht on täielikult funktsioneeriv ka siis, kui JavaScripti tugi veebilehitsejas puudub.

2. Uurimus

2.1. Varasemad uuringud

Varasemaid uuringud on tehtud mitmeid, kuid neist on hetkel märkimisväärsamad kaks Rene Saarsoo koostatud uurimistööd ja üks Elyna Nevski uurimistöö.

- Saarsoo, R. (2006) Standardid ja Eesti veebimaastiku olukord nende osas
- Saarsoo, R. (2006) Veebilehtede kodeerimispraktikad
- Nevski, E. (2008) Veebistandardid ja nende järgimine Eesti lasteportaalides

Rene Saarsoo uurimistöodes on rõhku pandud rohkem erinevate tehnoloogiate kasutatavusele, koodi korrektsusele, standardite järgimisele ja erinevate lehtede kodeerimispraktikate kasutamise uurimisele. Elyna Nevski töös on aga eraldi välja toodud ka ligipääsetavuse (*accessibility*) probleemid.

2.2. Uurimuse ülesehitus

Käesolevas uurimistöös on vaatluse all Eesti koolide veebilehed, kuna uurimistöö autor on ise õpilane ning on varasemalt välja arendanud Tallinna Rahumäe Põhikooli veebilehe ning uurimistöö koostamise hetkel arendab ja haldab Tallinna Reaalkooli 125. lennu B klassi veebilehte.

Veebilehtedel uuritavateks objektideks on:

- Dokumenditüüp
- Tähekodeering
- (X)HTMLi vigade esinemissagedus ja kõige levinumad veateated
- CSSi vigade esinemissagedus ja kõige levinumad veateated
- JavaScripti ja AJAXi kasutatavus

2.3. Uurimuse läbiviimine

Kõigi koolide veebilehtede valideerimiseks oli tarvis nimekirja koolide veebiaadressidest. Nimekirja moodustamiseks kasutati PHP keeles kirjutatud skripti (vt lisa 1). Skripti abil parsiti neti.ee kataloogipuud, mis sisaldab üsna täiuslikku 884 kooli veebiaadresside loetelu.

Veebilehtedel esinevate vigade statistika tegemiseks osutus sobilikuks Rene Saarsoo koostatud programm, sest see vastas kõikidele uurimuse läbiviimiseks vajalikele nõudmistele ning samuti on tänu identsele uurimismeetodile kogutavad andmed Rene Saarsoo uurimistööga võrreldavad. Programmi ülesehitusega saab lähemalt tutvuda Rene Saarsoo uurimistöös „Veebilehtede kodeerimispraktikad“ (Saarsoo 2006: 33).

Programmile anti ette varemkoostatud veebiaadressidega tekstifail, mida ta rida realt läbima hakkas. Iga rea peal laeti veebilehe HTML alla, analüüsiti ja valideeriti seda. Kui lehel oli kasutatud CSSi, siis analüüsiti ja valideeriti seda. Kui esines JavaScripti, siis analüüsiti ja otsiti kas seal esineb ka AJAXit (*XMLHttpRequest*).

Programmi jooksutamisel tõsisemaid probleeme ei tekkinud, vaid mõningad üksikud aadressid tulid sisendandmete seast eemaldada, kuna skript ei saanud teadmata põhjusel nende analüüsimisega hakkama.

Kõigi sisendandmete läbivaatamiseks kulus programmil mõni tund. Peale seda tuli kogutud andmete hulgast eemaldada ebasobivad lehed. Nende alla kuulusid lehed:

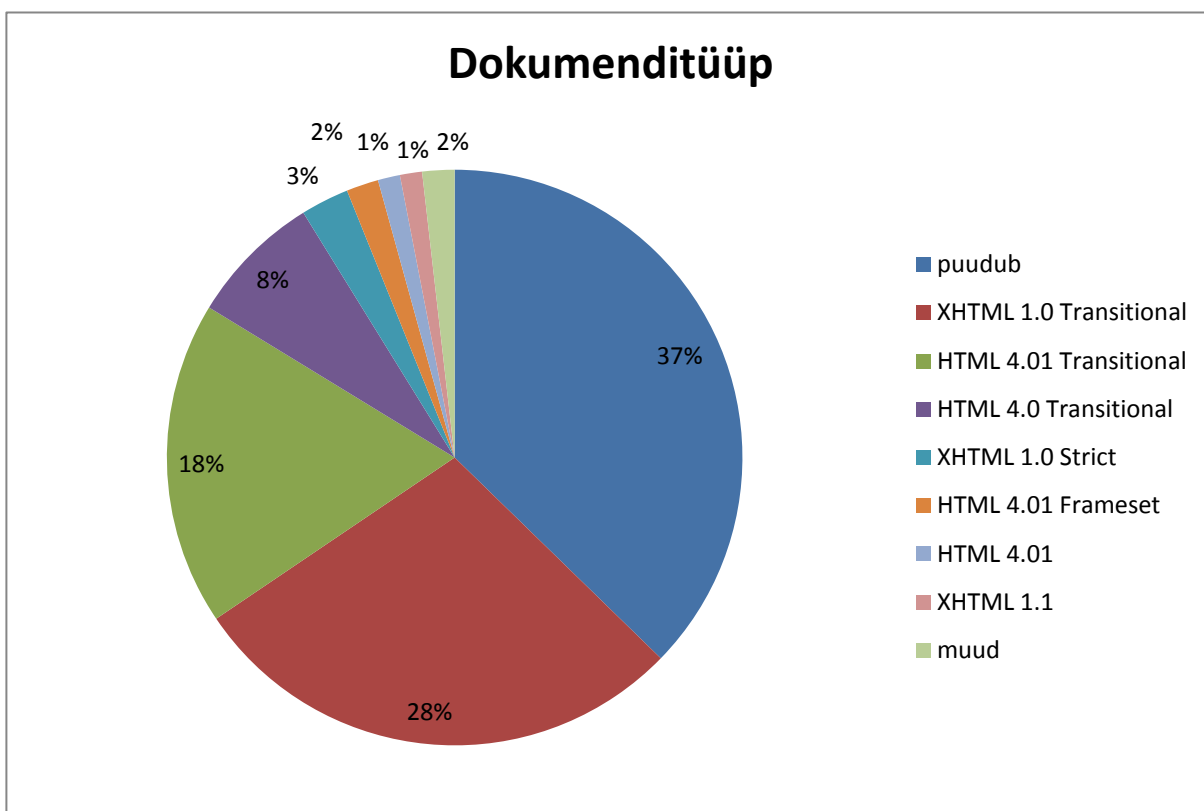
- mida ei suudetud valideerida (tundmatu dokumenditüüp)
- mida ei eksisteerinud (HTTP päise *status code* oli erinev *200 OK'st*, näiteks *404 Not Found*)
- mille failisuurus oli 0 baiti
- mis ei sisaldanud endas mitte ühtegi (X)HTML elementi

Kokku eemaldati uurimusest 14 kooli veebilehtede andmed, seega järele jäid 870 kooli andmed.

Andmete analüüsimiseks koostas autor MySQL andmebaasipäringud. Andmebaasist päritud andmed eksporditi Microsoft Excelisse ja kogutud andmete põhjal genereeriti järgnevalt esitatud diagrammid.

2.4. Uurimuse tulemused

2.4.1. Dokumenditüüp



Joonis 1. Dokumenditüüp

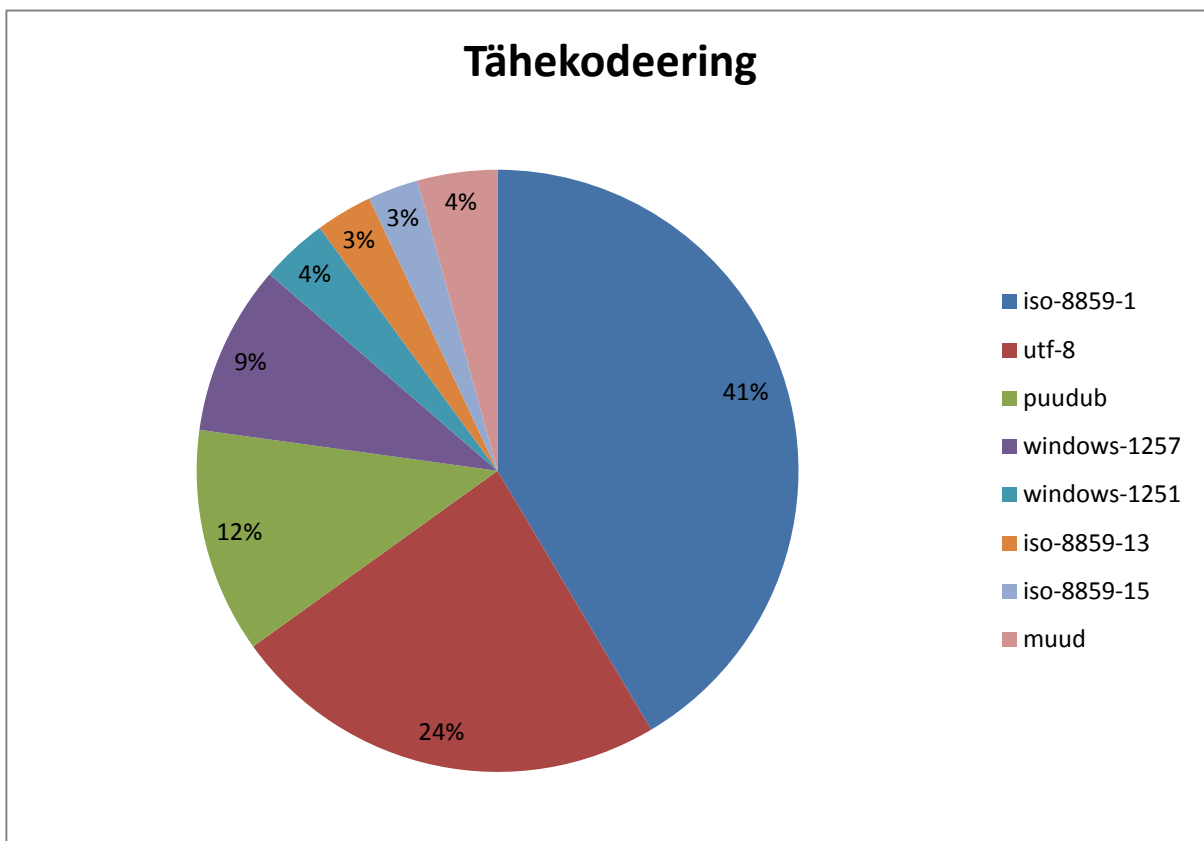
Dokumenditüübi määrangute osakaal näitab kuivõrd suurel määral kasutatakse XHTML tehnoloogiat vananenud HTMLi asemel. Samuti näeme ära kui paljudel lehtedel üldse on dokumenditüüp määratud. Selle määramata jätmine näitab, et veebiarendaja pole teadlik veebistandardite olemasolust.

37% lehtedest on dokumenditüüp määramata jäetud või on määranguks pandud mõni tundmatu dokumenditüüp, mis ei ole W3C standard. Kui dokumenditüüp on korrektelt määratud, siis on see enamikel juhtudel *XHTML 1.0 Transitional* (28%). 18% lehtedest esines dokumenditüüp *HTML 4.01 Transitional*. Üllataval kombel kasutab koguni 8% lehtedest HTMLi versiooninumbriga 4.0. See versioon anti välja aastal 1997, kaks aastat enne seda, kui *HTML 4.01* W3C standardiks kuulutati (Common Ideas...).

Struktuurset dokumenditüüpi (*XHTML 1.0 Strict* ja *XHTML 1.1*) kasutatakse vaid 4% lehtedest, ülejäänud 96% lehtedest kasutavad vananenud ja jätkusuutmatuid tehnoloogiaid.

Kokkuvõttes on XHTMLi osakaal 33%, HTMLi osakaal 30% ja ülejäänud 37% lehtedest pole korrektset dokumenditüüpi määranud.

2.4.2. Tähekodeering



Joonis 2. Tähekodeering

Tähekodeeringute osakaalust saame välja lugeda kui suurel määral on veebiarendajad kasutama hakanud universaalset UTF-8 tähekodeeringut, milles on defineeritud kõik maailmas enamlevinud keeltes kasutatavad tähemärgid ja sümbolid. Selle tähekodeeringuga ei teki kunagi probleeme täpitähtedega vms erisümbolitega.

Üsnagi märkimisväärne hulk (24%) lehtedest kasutab universaalset UTF-8 tähekodeeringut, kuid suurem enamus on siiski püsima jäänud erinevatele ASCII tabeli teisenditele ISO-8859-1, Windows-1257, Windows-1251, ISO-8859-13 ja ISO-8859-15.

ISO-8859-1 on ladina tähestikku toetav tähekodeering, kuid sealt on puudu mõned üksikud sümbolid nagu š, Š, ž, Ž, € (Wikipedia 2009 s.v.). Hoolimata faktist, et sealt on puudu eelmainitud hädavajalikud tähed eesti keele kirjutamiseks, kasutatakse teda sellegipoolest 41% lehtedest.

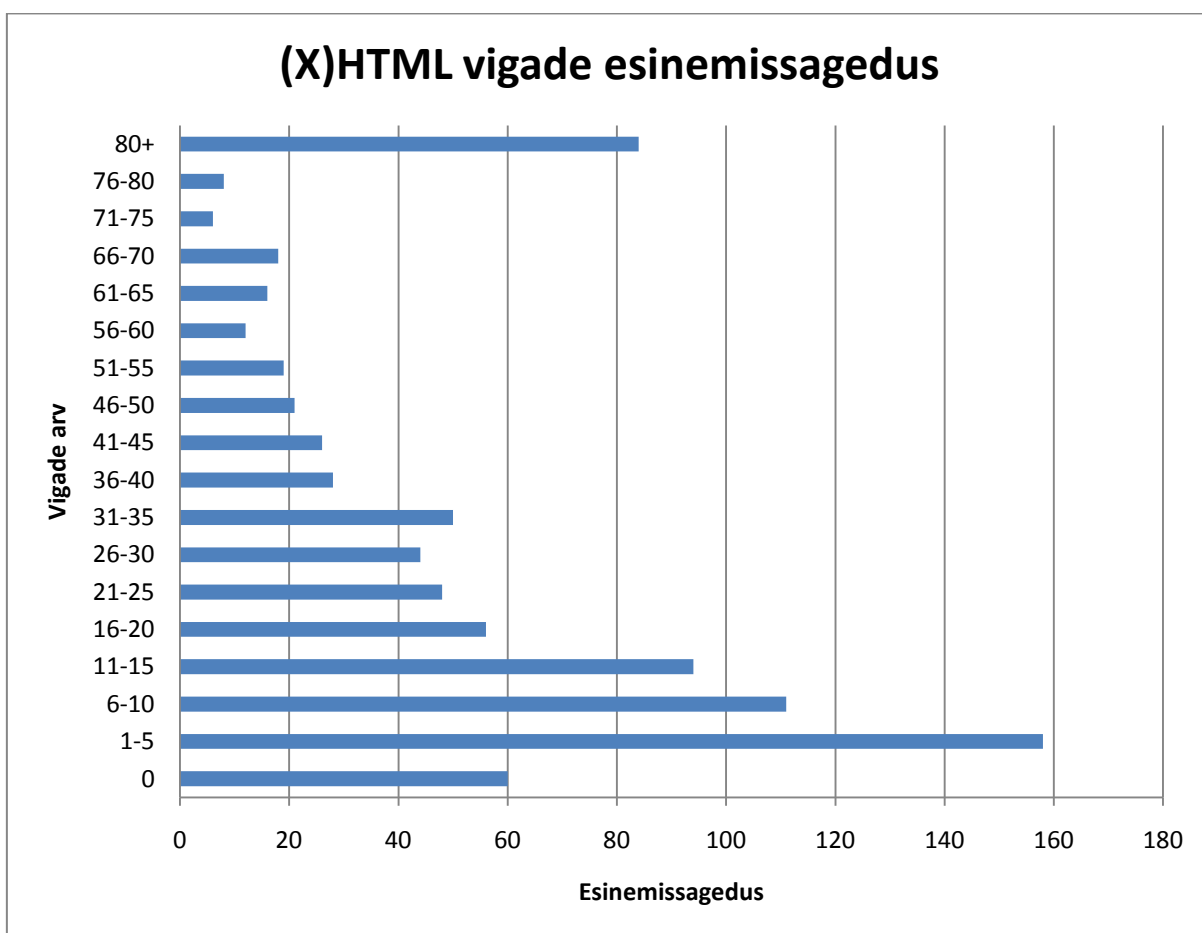
Eelmainitud sümbolite toe jaoks loodi uus tähekodeering ISO-8859-15, mille aluseks võeti ISO-8859-1, kuid asendati mõned vähem vajalikud sümbolid eelmainitutega. Seda tähekodeeringut kasutatakse vaid 3% lehtedest.

ISO-8859-13, mis on spetsiaalselt loodud Baltimaade keelte jaoks, kasutab samuti 3% veebilehtedest.

Windows-1257 on peaaegu sama, mis ISO-8859-13 ning seda kasutab 9% veebilehtedest.

Windows-1251, mis on loodud spetsiaalselt slaavi tähestiku jaoks, kasutab 4% lehtedest. Suure tõenäosusega on need Vene koolide veebilehed.

2.4.3. (X)HTML vigade esinemissagedus

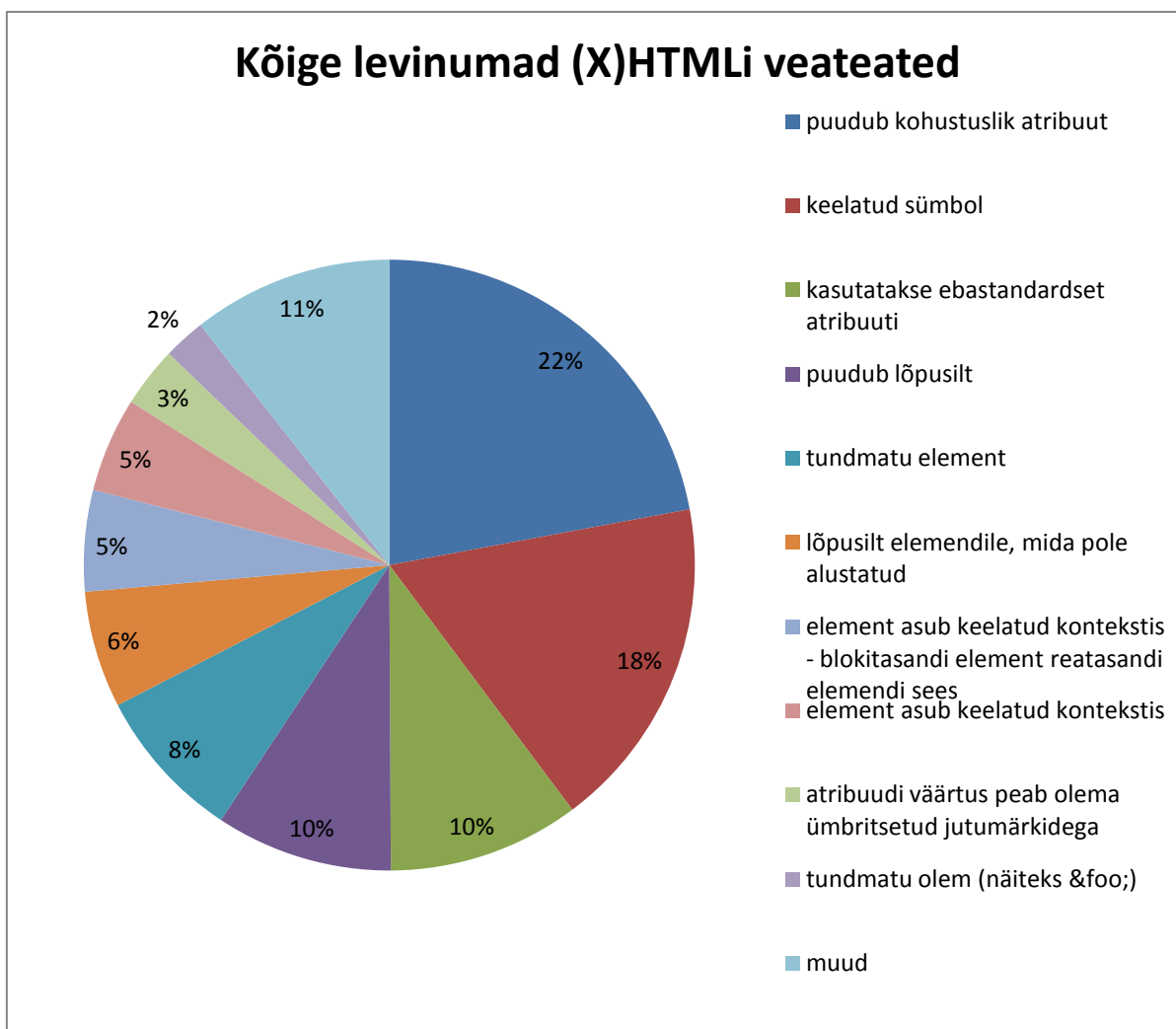


Joonis 3. (X)HTML vigade esinemissagedus

Vaid 60 lehte (7%) kõigist 870 lehest valideerub ilma ühegi veata. Suuremal osal lehtedest esinevad vaid mõned üksikud vead. Rohkemate vigadega lehti on järjest vähem, kuid sellegipoolest on märkimisväärne hulk (10%) lehti, kus vigade arv küündib üle kaheksakümne.

Kuna 37% lehtedest on korrektne dokumentitüüp määramata jäetud, siis need lehed valideeriti kui *HTML 4.01 Transitional* dokumentidena. See pole võib-olla päris korrektne ning kallutas uurimise tulemusi paremuse poole, kuna eelmainitud dokumentitüüp on üks leebematest (ei ole keelatud kujunduslikud sildid). Kõige õigem oleks olnud need lehed edasisest uurimisest välja jätta, kuid seda oleks tehniliselt tülikas teha olnud.

2.4.4. Kõige levinumad (X)HTMLi veateated



Joonis 4. Kõige levinumad (X)HTMLi veateated

Kõige levinumaks veaks on kohustusliku atribuudi puudumine (22%). Suure tõenäosusega moodustab valdava osa sellest piltide jaoks kasutatava img-sildi alt (*alternative text*) atribuudi ärajätmine. See atribuut on mõeldud selleks, et kui internetilehitsejal on mingil põhjusel piltide kuvamine ja allalaadimine keelatud, siis näidatakse pildi asemel teda kirjeldavat teksti.

Teiseks kõige levinumaks veaks on keelatud sümbolite kasutamine (18%). Suure tõenäosusega moodustab valdava osa sellest probleemist linkides esinevad &-märgid. Nimelt on &-märk HTML-keeles eritähendusega märgend (olemi algusmärgend) ning teda tuleb kasutada olemiga *&* (*amp* ehk *ampersand*).

Kolmandaks kõige rohkem esinevaks veaks on ebastandardsete atribuutide kasutamine (10%). Suure tõenäosusega kasutatakse täiesti standardset atribuuti, mis on aga mõnes muus dokumentitüübi standardis lubatud. Seega tuleks nende atribuutide kasutamiseks ära muuta dokumentitüübi määrang vastavalt selleks, kus see atribuut lubatud on (näiteks *Strict* dokumentitüüp *Transitionaliks*).

Neljandaks kõige rohkem esinevaks veaks on lõpusildi puudumine (10%). See on ilmselge lohakusviga, kus kirjutatakse elemendi algusesse algussilt, kuid unustatakse hiljem elemendi lõppu lõpusilt kirjutada.

Viiendaks kõige tihedamini esinevaks veaks on tundmatu elemendi kasutamine (8%). Selle vea kõige tõenäolisem põhjus on sama, mis ebastandardsete atribuutide kasutamise puhulgi.

Viga, kus on lõpusilt lisatud elemendile, mida pole alustatud, esineb 6% juhtudest. See on samasugune lohakusviga nagu lõpusildi puudumine.

„Element asub keelatud kontekstis“ viga esineb näiteks siis, kui dokumendi sisu ei asu *body* sildi sees või kui blokitasandi element on reatasandi elemendi sees.

Atribuudi väärtuse jutumärkidega ümbritsemata jätmine (3%) on levinud viga, kuna ei teata, miks see vajalik on. Jutumärgid näitavad kus algab ja lõpeb atribuudi väärtus. See on vajalik selleks, et atribuutidele saaks mitmesõnalisi väärtusi anda. Kui jutumärke poleks, siis veebilehitseja arvaks, et peale tühikut algab uus atribuut:

Vale:

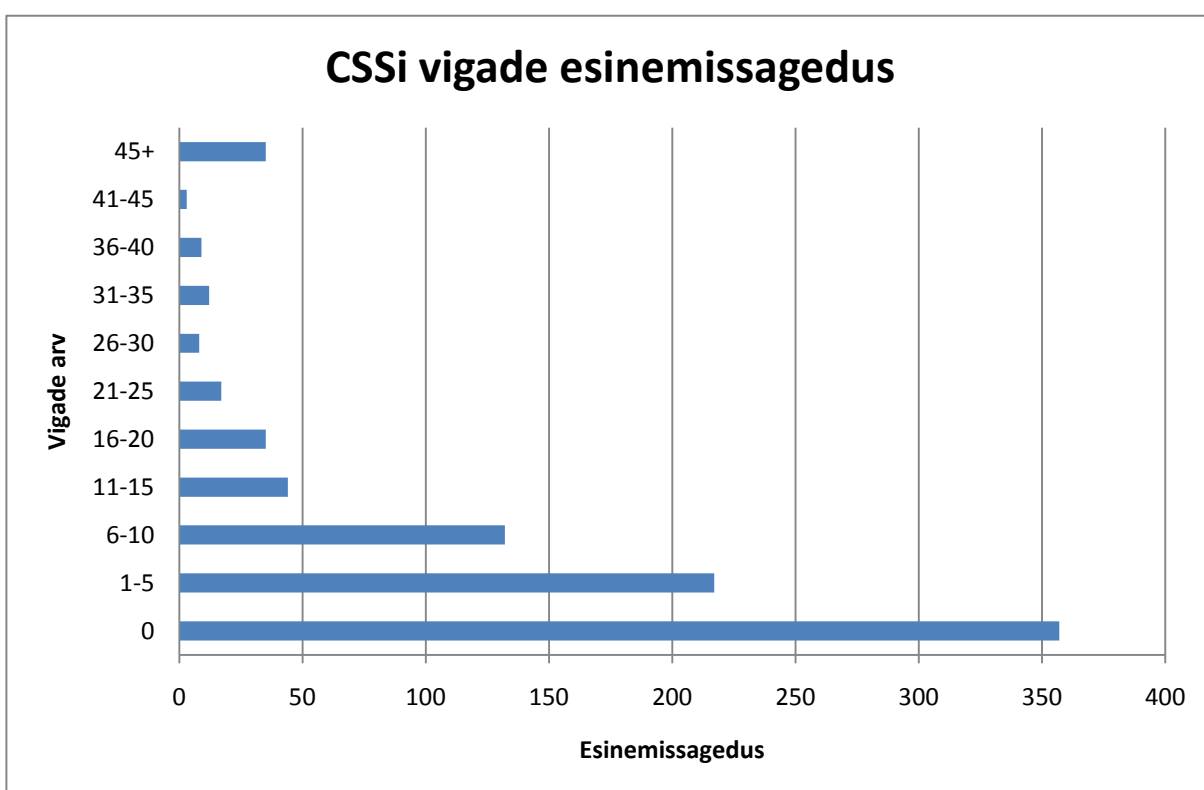
```
<silt atribuut=foo bar>baz</silt>
```

Õige:

```
<silt atribuut="foo bar">baz</silt>
```

Tundmatute olemite esinemine (2%) on ilmselt põhjustatud &-märgi kui keelatud sümboli kasutamisest.

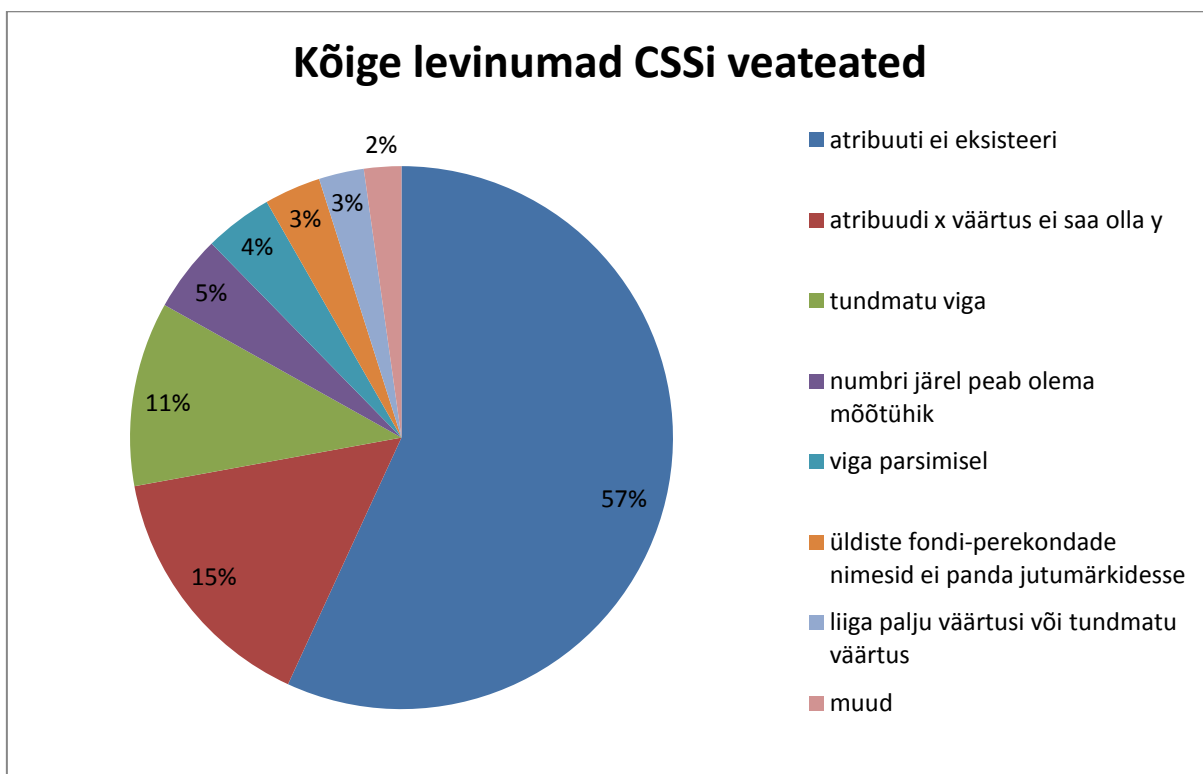
2.4.5. CSSi vigade esinemissagedus



Joonis 5. CSSi vigade esinemissagedus

CSSi on kasutatud 86% lehtedest (749 lehte). Enamus lehti, mis CSSi kasutasid, tegid seda täiesti standarditele vastavalt. 357 lehte valideerus ilma ühegi veata (48%), 217 lehel esines 1-5 viga (29%) ja 132 lehel esines 6-10 viga (17%). Rohkem kui 10 veaga lehti oli juba tunduvalt vähem (163 lehte ehk 22%).

2.4.6. Kõige levinumad CSSi veateated



Joonis 6. Kõige levinumad CSSi veateated

Kõige levinumaks veaks (57%) on atribuutide kasutamine, mida tegelikkuses ei eksisteeri. Selle vea põhjuseks võivad olla trükivead atribuutide nimedes jms lohakusvead.

Teiseks enamlevinumaks veaks (15%) on atribuutidele ebakorrektsed väärtused andmine. Selle vea põhjuseks võib samuti trükiviga olla.

11% juhtudest ei suudetud kindlaks teha mis veaga täpselt tegemist on.

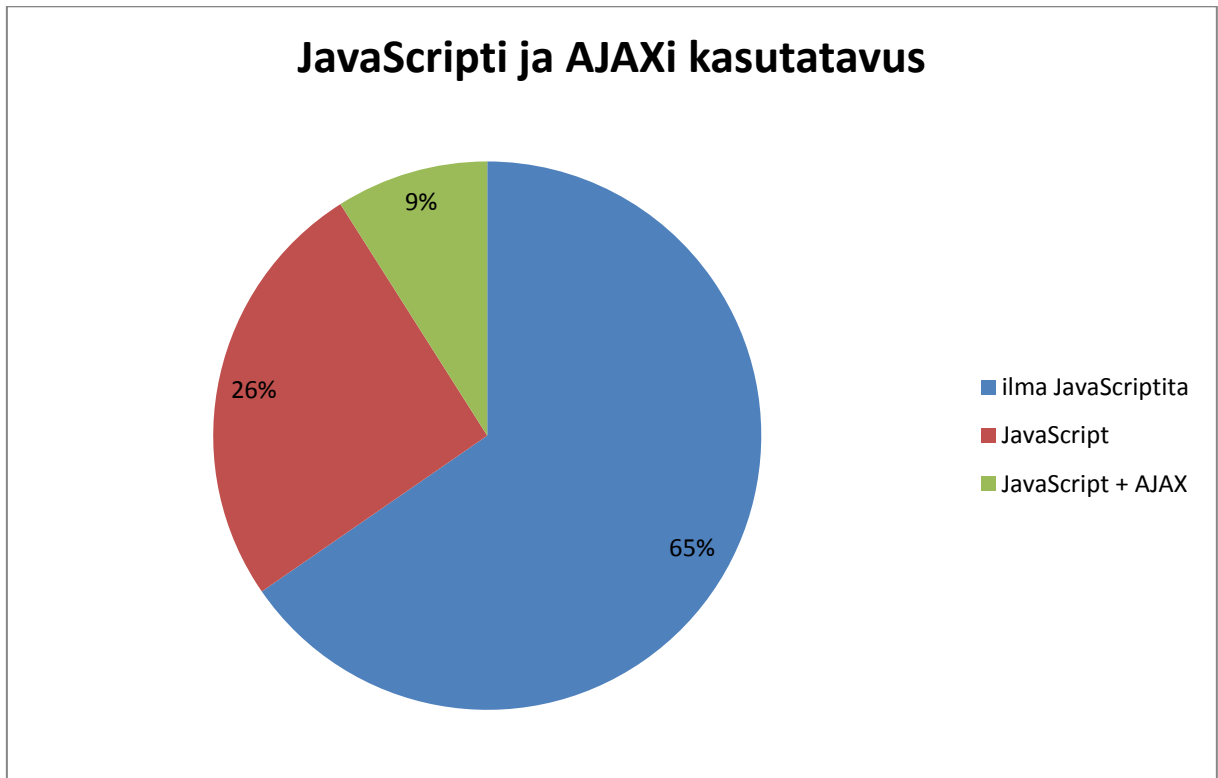
5% juhtudest unustati atribuudi numbrilise väärtuse korral sinna järele mõõtühik lisada.

4% juhtudest ei suudetud viga parsida.

3% juhtudest pandi üldiste fondi-perekondade nimed jutumärkidesse (näiteks serif ja sans-serif).

3% juhtudest anti atribuudile liiga palju väärtusi või sootuks tundmatu väärtus.

2.4.7. JavaScripti ja AJAXi kasutatavus



Joonis 7. JavaScripti ja AJAXi kasutatavus

JavaScripti kasutatavus näitab kliendipoolsete dünaamiliste lehtede osakaalu. AJAX, olles üsnagi uus tehnoloogia, omakorda näitab aga mõningal määral veebilehtede kaasaegsust.

JavaScripti esines 331 lehel ehk 38% kõigist lehtedest, AJAXit esines 86 lehel ehk 9% kõigist lehtedest. Pelgalt nende andmete põhjal on raske öelda, kas neid tehnoloogiaid kasutatakse ülemäära palju või mitte. Selleks tuleb juba iga leht eraldi vaatluse alla võtta, kuna selle automatiseeritult kindlakstegemine on üsna keeruline, kui mitte võimatu.

Kokkuvõte

Uurimise käigus võeti vaatluse alla 870 kooli veebilehed, 14 kooli veebilehed eemaldati uurimisest. Tulemuste kokkuvõte:

- Valdav osa (38%) lehtedest ei defineeri korrektselt dokumenditüüpi
- Kõige levinumad dokumenditüübid on *XHTML 1.0 Transitional* (28%) ja *HTML 4.01 Transitional* (18%) ehk teisisõnu vananenud dokumenditüübid, kus pole lehe struktuur ja kujundus üksteisest eraldatud
- Struktuurseid dokumenditüüpe (*HTML 4.01 Strict*, *XHTML 1.0 Strict* ja *XHTML 1.1*) kasutab vaid 4% lehtedest
- Kõige laialdasemalt kasutatav (41%) tähekodeering on ISO-8859-1, mis ei ole reaalselt sobilik eesti keele jaoks, kuna seal puuduvad tähed š, Š, ž, Ž
- Universaalset tähekodeeringut UTF-8 kasutatakse koguni 24% lehtedest
- Vaid 7% Eesti koolide veebilehtedest kasutab standardile vastavat (valideeruvat) (X)HTMLi
- 86% lehtedest kasutab CSSi ja 48% nendest kasutab seda standardile vastavalt
- 38% lehtedest kasutab JavaScripti ja 9% lehtedest kasutab AJAXit

Tulemustest järeldub, et Eesti koolide veebilehtedel on veebistandardite järgimise osas veel tublisti arenguruumi. Struktuurseid dokumenditüüpe kasutatakse minimaalselt ja enamusel lehtedest on kasutusel eesti keelele mittesobiv tähekodeering.

Valideeruvate lehtede osakaal on võrreldes varasemate uurimustega veidi suurem, kuid arvuliselt väljendatuna on 7% siiski häbiväärselt väike.

Uurimistöö hüpoteesidest said kinnitust kaks esimest:

- Valdav osa lehti ei defineeri korrektselt dokumenditüüpi
- Valdav osa lehti ei kasuta standarditele vastavat (valideeruvat) (X)HTMLi ja CSSi

Paika ei pidanud autori poolt püstitatud kolmas hüpotees - „Universaalset tähekodeeringut UTF-8 kasutatakse minimaalselt“. Uurimus tõestas, et UTF-8 tähekodeeringut kasutati koguni 24% lehtedest.

Neljandat püstitatud hüpoteesi - „JavaScripti ja AJAXit kasutatakse ülemäära palju ehk leht pole täielikult funktsioneeriv JavaScripti mittetoetavatel veebilehitsejatel“ ei olnud võimalik antud uurimusega kinnitada ega ümber lükata. Uurimistöös kasutatud automatiseeritud meetodi tõttu ei olnud võimalik kogutud andmete põhjal hüpoteesi osas järeldusi teha.

Kasutatud materjalid

1. AJAX Introduction. Kättesaadav: w3schools.com/Ajax/ajax_intro.asp, 15.03.2009
2. Common Ideas Between HTML and XHTML. Kättesaadav: webstandards.org/learn/tutorials/common_ideas/, 22.02.2009
3. Differences Between XHTML And HTML. Kättesaadav: w3schools.com/XHTML/xhtml_html.asp, 22.02.2009
4. HARIDUS JA KULTUUR / Haridus. Kättesaadav: neti.ee/cgi-bin/teema/HARIDUS_JA_KULTUUR/Haridus/, 23.01.2009
5. HTML doctype definition. Kättesaadav: w3schools.com/tags/tag_DOCTYPE.asp, 27.01.2009
6. HTML Introduction. Kättesaadav: w3schools.com/html/html_intro.asp, 02.03.2009
7. HTML Versus XHTML. Kättesaadav: webstandards.org/learn/articles/askw3c/oct2003/, 07.03.2009
8. Introduction to CSS. Kättesaadav: w3schools.com/css/css_intro.asp, 13.03.2009
9. Introduction to JavaScript. Kättesaadav: w3schools.com/JS/js_intro.asp, 15.03.2009
10. Introduction to XHTML. Kättesaadav: w3schools.com/xhtml/xhtml_intro.asp, 07.03.2009
11. Johansson, R. (2006) Developing With Web Standards. Kättesaadav: 456bereastreet.com/lab/developing_with_web_standards/, 24.04.2008
12. Saarsoo, R. (2006) Veebilehtede kodeerimispraktikad
13. Spolsky, J. (2003) The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!). Kättesaadav: joelonsoftware.com/articles/Unicode.html, 17.02.2009
14. Wikipedia s.v. Ajax (programming). Kättesaadav: [en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming)), 15.03.2009
15. Wikipedia s.v. HTML. Kättesaadav: en.wikipedia.org/wiki/HTML, 02.03.2009
16. Wikipedia s.v. ISO/IEC_8859-1. Kättesaadav: en.wikipedia.org/wiki/ISO/IEC_8859-1, 02.03.2009
17. Wikipedia s.v. ISO/IEC_8859-13. Kättesaadav: en.wikipedia.org/wiki/ISO/IEC_8859-13, 02.03.2009
18. Wikipedia s.v. ISO/IEC_8859-15. Kättesaadav: en.wikipedia.org/wiki/ISO/IEC_8859-15, 02.03.2009
19. Wikipedia s.v. JavaScript. Kättesaadav: en.wikipedia.org/wiki/JavaScript, 15.03.2009

20. Wikipedia s.v. Quirks mode. Kättesaadav: en.wikipedia.org/wiki/Quirks_mode, 13.01.2009
21. Wikipedia s.v. UTF-8. Kättesaadav: en.wikipedia.org/wiki/UTF-8, 02.03.2009
22. Wikipedia s.v. Windows-1252. Kättesaadav: en.wikipedia.org/wiki/Windows-1252, 02.03.2009
23. Wikipedia s.v. Windows-1257. Kättesaadav: en.wikipedia.org/wiki/Windows-1257, 02.03.2009

Lisa 1 Neti.ee kataloogipuu parsimisskript

PHP kood:

```
<?php
$url = 'http://www.neti.ee/cgi-
bin/teema/HARIDUS_JA_KULTUUR/Haridus/Keskkoolid/';
$content = file_get_contents($url);
$matches = array();
preg_match_all('#<li><a href="(.*)"
target="_top">(.*</a>(.*</li>#', $content, $matches);
foreach ($matches[1] as $key => $val) {
    echo $val . "\r\n";
}
?>
```

Lisa 2 Dokumenditüüp

Tabel 1. Dokumenditüüp

Dokumenditüüp	Esinemissagedus
puudub	330
XHTML 1.0 Transitional	250
HTML 4.01 Transitional	161
HTML 4.0 Transitional	66
XHTML 1.0 Strict	24
HTML 4.01 Frameset	16
HTML 4.01	11
XHTML 1.1	11
muud	16

Lisa 3 Tähekodeering

Tabel 2. Tähekodeering

Tähekodeering	Esinemissagedus
iso-8859-1	367
puudub	309
utf-8	209
windows-1257	81
windows-1251	32
iso-8859-13	27
iso-8859-15	24
iso-8859-4	13
windows-1252	12
latin-1	8
muud	5

Lisa 4 (X)HTML vigade esinemissagedus

Tabel 3. (X)HTML vigade esinemissagedus

Vigade arv	Esinemissagedus
0	60
1-5	171
6-10	111
11-15	94
16-20	56
21-25	48
26-30	44
31-35	50
36-40	28
41-45	26
46-50	21
51-55	19
56-60	12
61-65	16
66-70	18
71-75	6
76-80	8
80+	84

Lisa 5 Kõige levinumad (X)HTML veateated

Tabel 4. Kõige levinumad (X)HTML veateated

Veateade	Esinemissagedus
puudub kohustuslik atribuut	7559
keelatud sümbol	6061
kasutatakse ebastandardset atribuuti	3473
puudub lõpusilt	3210
tundmatu element	2783
lõpusilt elemendile, mida pole alustatud	2118
element asub keelatud kontekstis - blokitasandi element reatasandi elemendi sees	1835
element asub keelatud kontekstis	1717
atribuudi väärtus pole ümbritsetud jutumärkidega	1093
tundmatu olem (näiteks &foo;)	760
muud	3633

Lisa 6 CSSi vigade esinemissagedus

Tabel 5. CSSi vigade esinemissagedus

Vigade arv	Esinemissagedus
0	357
1-5	217
6-10	132
11-15	44
16-20	35
21-25	17
26-30	8
31-35	12
36-40	9
41-45	3
45+	35

Lisa 7 Kõige levinumad CSSi veateated

Tabel 6. Kõige levinumad CSSi veateated

Veateade	Esinemissagedus
atribuuti ei eksisteeri	5585
atribuudi x väärtus ei saa olla y	1500
tundmatu viga	1079
numbri järel peab olema mõõtühik	447
viga parsimisel	399
üldiste fondi-perekondade nimesid ei panda jutumärkidesse	330
liiga palju väärtusi või tundmatu väärtus	264
muud	218