Jukka Mäntylä

WWW-SIVUN SAAVUTETTAVUUDEN AUTOMAATTINEN ARVIOINTI DOM-RAJAPINTAA KÄYTTÄEN

Tietotekniikan pro gradu -tutkielma

Opettajankoulutuksen suuntautumisvaihtoehto

31.7.2006

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Jukka Mäntylä (synt. 5.10.1980).

Yhteystiedot: Sähköposti jmantyla@iki.fi ja puh. 044 043 2759

Työn nimi: WWW-sivun saavutettavuuden automaattinen arviointi DOM-rajapintaa käyttäen.

Title in English: DOM-based automatic web accessibility evaluation.

Työ: Pro gradu –tutkielma.

Sivumäärä: 138 + 27.

Suuntautumisvaihtoehto: Opettajankoulutus.

Teettäjä: Jyväskylän yliopisto, tietotekniikan laitos.

Avainsanat: DOM, XPath, WWW-sivut, saavutettavuus, esteettömyys, käytettävyys, vammaisuus, matkapuhelimet, automaattinen arviointi, mukauttaminen, osiin jakaminen, Mozilla Firefox -verkkoselain.

Keywords: DOM, XPath, WWW, web pages, accessibility, usability, automated evaluation, Design for All, disabilities, mobile phones, adaptation, segmentation, Mozilla Firefox web page browser.

Tiivistelmä: Tutkielmassa alussa tarkastellaan, miten erilaiset vammat ja tekniset rajoitteet vaikuttavat verkkosivujen selailuun. Seuraavaksi selvitetään kriteerit, jotka edistävät verkkosivujen yleistä saavutettavuutta. Kriteerien pohjalta kehitään automaattinen saavutettavuuden arviointijärjestelmä, joka hyödyntää selainympäristön tarjoamaa DOM-rajapintaa ja elementtien asemointitietoja. Näistä louhitut tiedot mahdollistavat dynaamisten sivustojen kattavan arvioinnin.

Abstract: In this thesis, we examine how different disabilities and technical limitations affect Web browsing. Then, we specify the criteria for universal accessibility. Based on the criteria, we create an automated accessibility evaluation system, which utilizes DOM-interface and element position properties provided by the browser environment. Extracted data makes the complete evaluation of dynamic pages possible.

Termiluettelo

CSS	Cascading Style Sheets. Standardi verkkosivun ulkoasun määräämiseksi.
DOM	Document Object Model. Rajapinta, joka mahdollistaa dokumenttien sisällön, rakenteen ja ulkoasutietojen lukemisen ja muokkaamisen.
Esteettömyys	Ominaisuus, joka kuvaa kuinka hyvin vammaiset käyttäjät pystyvät käyttämään verkkosivua.
Graafinen selain	Ohjelma, joka kykenee näyttämään visuaalisesti WWW-sivun sisällön isolla resoluutioilla.
Hahmonnus (engl. rendering)	Selaimen muistissa olevien tekstien ja muotoilu-tietojen muuttaminen näytölle kelpaavaan muotoon.
Heuristiikka	Asiantuntijatietoon perustuva tehokas nyrkkisääntö. Todistamaton ongelman ratkaisutekniikka.
HTML	HyperText Markup Language. WWW-sivujen kuvauskieli.
Kuvaileva äänipalvelu (engl. audio description)	Menetelmä, jossa ääniraidan tukena on visuaalisia yksityiskohtia kuvaileva kertoja.
Käytettävyys	Ominaisuus, joka kuvastaa kuinka intuitiivinen verkkosivun muodostama käyttöliittymä on.
Lähdekoodi	Verkkosivun muodostavat HTML- tai XHTML-merkinnät.
Mobiililaite	Kannettava ja helposti liikuteltava laite, jossa on internet-yhteys. Esim. älypuhelin tai PDA-laite.
PDA	Personal Digital Assistant. Kannettava kämmentietokone, jossa voi olla internet-selain.
PDF	Portable Document Format.

	Tarkkoihin tulostuksiin sopiva dokumenttiformaatti.
RTF	Rich Text Format. Hyvin tuettu dokumenttiformaatti, joka tallentaa myös tekstin muotoilut.
Saavutettavuus	Ominaisuus, joka kuvaa kuinka hyvin erilaiset käyttäjät kykenevät käyttämään verkkosivua.
SGML	Standard Generalized Markup Language. Metakieli, jolla kuvataan merkintäkieliä. Esimerkiksi HTML on SGML-sovellus.
Skaalautuvuus	Verkkosivun mukautumiskyky selainikkunan tai tekstin kokoon.
Spacer-kuva	Verkkosivulle sijoitettu ohut läpinäkyvä kuva, jonka avulla voidaan asetella muita sivun osia.
WYSIWYG	What You See Is What You Get. Ohjelma, jota käytettäessä nähdään tuotos lähes sellaisena kuin se on lopullisessa muodossaan.
XHTML	eXtensible HyperText Markup Language. HTML-kieli muokattuna XML-sovellukseksi.
XML	eXtensible Markup Language. Yleiskäyttöinen metakieli, jolla kuvataan rakenteisten dokumenttien kuvauskieliä.
XPath	Tekniikka, jolla voidaan osoittaa dokumentissa olevia solmuja yhdellä merkkijonolla.
XUL	XML User Interface Language. Graafisten käyttöliittymien kuvauskieli.
W3C-konsortio	World Wide Web Consortium. W3C on organisaatio, joka julkaisee verkkostandardeja.
WCAG	Web Content Accessibility Guidelines. W3C:n julkaisema saavutettavuussuositus.

Sisältö

1 JOHDANTO

Painotuotteita ja tulosteita suunniteltaessa tiedetään melko hyvin, millainen värivalikoima on käytettävissä, minkä kokoisia lopputuotteet ovat ja millaiset ihmiset lukevat niitä. Verkkosivuja suunniteltaessa ei voida tehdä tällaisia tarkkoja oletuksia. WWW-sivuja voi selata kuka tahansa erilaisilla laitteilla ja ohjelmilla. Sivuja voidaan katsella muun muassa matkapuhelimilla tai kannettavilla pelikonsoleilla. Sivustolla voi vierailla vaikkapa sokea verkon käyttäjä. Käsistään vammautunut henkilö ei voi välttämättä käyttää tietokoneen näppäimistöä tai hiirtä tavallisin keinoin verkkoa selatessaan.

On virheellistä olettaa, että verkkosivuja käytetään samanlaisin mahdollisuuksin ja sellaisessa ympäristössä, kuin missä ne on suunniteltu ja testattu. Suunnittelussa on otettava huomioon, että loppukäyttäjä päättää täysin, millä tavalla hän haluaa saavuttaa sivun sisällön. Valitettavasti huonoilla suunnittelutavoilla ja väärillä toteutustavan valinnoilla voidaan estää sivun toimivuus erityistilanteissa.

Saavutettavuuden arviointi on tärkeä osa verkkosivuston kehittämisprosessia. Täydellinen käytettävyystestaus, jossa huomioidaan myös vaihtelevat käyttötilanteet, on yleensä liian vaativa toimenpide omin voimin toteuttavaksi ja toisaalta liian kallista muualta hankittavaksi. Siksi on saatavilla erilaisia automaattisia arviointityökaluja verkkosivuston suunnittelijan oman arvioinnin tueksi. Tässä tutkielmassa käsittelen, miten näitä arviointityökaluja voidaan kehittää yhä paremmiksi.

Tutkielman rakenne on seuraava: Luvussa 2 esitetään saavutettavuuden ja tämän lähikäsitteiden määritelmät. Ongelmakenttää hahmotetaan saavutettavuuden historian, arvioinnin ja hyötynäkökohtien kautta. Luvussa 3 kuvaillaan ja ryhmitellään tavallisesta pöytäkoneympäristöstä poikkeavat verkon käyttötavat. Luvussa 4 selvitetään tekniset toteutustavat, joilla saavutettavuutta voidaan parantaa, ja menetelmät, joita sivuston tekijöiden olisi syytä välttää. Kriteerit on luokiteltu luvun 3 mukaisiin ryhmiin. Luvussa 5 tarkastellaan ensin, millaisin tavoin arviointijärjestelmiä voidaan toteuttaa. Sen jälkeen luvussa esitetään tekniikat ja arviointijärjestelmän runko, jolla saavutettavuuden arviointia

voidaan tehdä kattavasti suoraan verkkoselaimen avulla. Luvun loppuosassa esitetään menetelmiä, joilla voidaan tarkistaa yksittäisten saavutettavuuskriteerien toteutuminen.

Tutkielma on kohdistettu saavutettavien verkkosivujen toteuttamisesta ja saavutettavuuden automaattisesta arvioinnista kiinnostuneille. Tutkielman ymmärtämiseksi lukijan on hyvä tuntea HTML-, XHTML- ja CSS-kuvauskielien perusteet.

2 SAAVUTETTAVUUS

Saavutettavuus on verkkomaailmassa vielä melko uusi ja tuntematon käsite. Saavutettavuudella on kuitenkin pitkä historia muissa medioissa. Tiedon välityksen saavutettavuutta on kehitetty esimerkiksi kirjojen, elokuvien ja television suhteen. Elokuvissa ja televisiossa on mahdollisuus tekstitykseen (subtitling), jälkiäänitykseen (dubbing), kuvaselostukseen (captioning), äänitekstitykseen (audio subtitling) ja kuvailevaan äänipalveluun (audio description) [16, s. 39-42]. Suomessa ollaan digitelevision myötä siirtymässä tekniikkaan, jossa lisäpalvelut siirretään pois kuvavirrasta. Samalla mahdollisuudet tekstityksen erillislukemiseen, viittomakielisiin tulkkauksiin ja äänimuotoisiin lisäpalveluihin paranevat [48, s. 6-8].

Verkkosivujen saavutettavuudella on yhtymäkohtia muun viestinnän saavutettavuuden kanssa. Luvuissa 2.1-2.3 käsitellään saavutettavuuden määritelmää, siihen vaikuttavia tekijöitä ja sen merkitystä.

2.1 Saavutettavuuden ja lähikäsitteiden määritelmiä

Saavutettavuus tai esteettömyys (engl. accessibility) määritellään tuotteen, palvelun tai tilan käytettävyydeksi erikykyisillä ihmisillä [34] tai ominaisuudeksi joka ilmentää sitä, kuinka helposti henkilö voi saada laitteen tai palvelun käyttöönsä [73].

Esteettömyys viittaa erityisesti fyysisesti tai psyykkisesti toimintarajoitteisten huomioimiseen. Henkilö on toimintarajoitteinen, jos hänellä on määritettävissä oleva toiminnan rajoite suhteessa ympäristöön. Henkilön rajoitteet voivat olla joko pysyviä tai tilapäisiä. Rajoitteet voivat olla sekä fyysisiä, psyykkisiä että sosiaalisia. Toimintarajoitteisia henkilöitä voivat olla esimerkiksi tilapäisesti vammautuneet henkilöt, vammaiset ja ikäihmiset. Esteettömyysperiaatteesta puhuttaessa tarkoitaan sitä, että rakennetun ympäristön, tuotteiden ja palvelujen tulee tukea toimintarajoitteisen henkilön itsenäistä suoriutumista. [73]

Esteettömyys tarkoitti alun perin sen varmistamista, että julkisiin tiloihin ja rakennuksiin on helppo päästä pyörätuolilla [68], mutta on sittemmin laajentunut kattamaan erilaisia

toimia, jotka parantavat vammaisten mahdollisuuksia osallistua tasavertaisesti yhteiskuntaan. Esteettömyyden käsite on viime aikoina laajentunut kattamaan myös palvelujen, viestinnän ja ihmisten välisen vuorovaikutuksen esteettömyyden, esimerkiksi esteettömät WWW-sivut [73]. Esteettömyys havainnollistaa sitä, että kyse on erityisten esteiden poistamisesta, myös verkkosivuilla [39].

W3C:n[1] määritelmän mukaan verkkosivujen esteettömyys (engl. Web accessibility) tarkoittaa sitä, että vammaiset ihmiset voivat käyttää verkkoa. Esteettömyydestä on kuitenkin hyötyä muillekin, sillä perusperiaate esteettömien WWW-sivujen suunnittelussa on tehdä sivuista joustavia vastaamaan käyttäjien tarpeisiin, mieltymyksiin ja tilanteisiin. Esteettömyydestä hyötyvät muun muassa käyttäjät, joilla on hidas internet-yhteys tai jotka ovat väliaikaisesti fyysisesti loukkaantuneita tai joiden aistikyvyt ovat heikentyneet iän myötä. [70]

Saavutettavuus kuvaa kuinka helposti informaation, järjestelmän, laitteen, ohjelman tai palvelun voi saada käyttöönsä. Käsite kattaa esteettömyyttä laajemman alan [68]. Suppeasti tulkittuna esteettömyys koskee pelkästään vammaisia ihmisiä, mutta laajasti ymmärrettynä käsite ei rajoitu ainoastaan tähän ryhmään [34]. Saavutettavuudesta puhuttaessa otetaan huomioon esimerkiksi käyttäjien tekniset rajoitteet. Hyvä saavutettavuus tarkoittaa muun muassa matkapuhelimen tai PDA-laitteen pienen näytön huomioimista. Saavutettavalla suunnittelulla pyritään siihen, että yhä useammat ihmiset voivat käyttää sivua tehokkaasti erilaisissa tilanteissa [75, s. 13].

W3C:n Suomen toimiston käännöksen mukaan konsortion eräs keskeinen tavoite on tuoda verkon informaatiouniversumin hyödyt kaikkien ihmisten saataville ilman, että tämä saavutettavuus riippuu laitteistosta, ohjelmistoista, verkkopalvelusta, äidinkielestä, kulttuurista, maantieteellisestä sijainnista tai ihmisten fyysisistä tai psyykkisistä kyvyistä [62]. Sen mukaan saavutettavuudella tarkoitetaan erilaisten käyttötarpeiden huomioimista

[1] W3C on lyhenne sanoista World Wide Web Consortium. W3C on useiden informaatioteknologian alan yritysten, yhdistysten, yliopistojen ja organisaatioiden kansainvälinen yhteenliittymä. W3C julkaisee verkkostandardeja ja -suosituksia.

arkisia sovelluksia toteutettaessa. Saavutettavuus poistaa käytettävyyden esteitä ja lisää käyttömukavuutta [61].

Tässä tutkielmassa verkkosivuston *saavutettavuus* (engl. accessibility) määritellään ominaisuudeksi, joka kuvaa sitä, kuinka helposti ja yhdenvertaisesti sivu on käytettävissä erilaisten käyttäjien ja käyttötilanteiden kesken. Sivusto on *saavutettava* (engl. accessible), jos käyttäjä kykenee fyysisistä, psyykkisistä tai teknisistä rajoitteista huolimatta käyttämään sivua yhtä tehokkaasti kuin käyttäjä, jolla ei ole näitä rajoitteita.

Saavutettavuuden lähikäsitteitä ovat esteettömyys ja englanninkieliset termit Design for All, e-Inclusion, e-Accessibility, inclusive design, universal design, universal access ja barrier-free design. Englanninkielen sana accessible voidaan kääntää helppopääsyiseksi, helppokäyttöiseksi, käytettävissä olevaksi, saatavilla olevaksi, ymmärrettäväksi, selkeäksi ja helposti lähestyttäväksi. Inclusive tarkoittaa sisällyttämistä tai mukaan ottamista, universal yleismaailmallista ja barrier-free on esteistä vapaata. Kaikilla näillä viitataan samaan asiaan, yhdenvertaisuuteen verkkomaailmassa, vaikkakin pienillä vivahde-eroilla. Näiden tavoitteita voidaan pitää kuitenkin toisiaan vastaavina [32].

Esteettömyyttä ja saavutettavuutta käytetään joskus kuitenkin synonyymeinä [39]. Sanojen monimerkityksellisyyden takia on alettu käyttämään muita termejä, joiden on ajateltu kuvaavan paremmin yhdenvertaista osallistumista. Design for All (DfA) on EU:n piirissä käytettävä ilmaisu [39]. Sen tarkoitus on edistää tasa-arvoa. DfA parantaa ympäristöjen esteettömyyttä, tuotteiden helppokäyttöisyyttä ja palvelujen saavutettavuutta. DfA pyrkii tekemään näistä houkuttelevia mahdollisimman monimuotoiselle käyttäjäjoukolle. Monimuotoisuus tarkoittaa, että käyttäjien joukossa voi olla ikääntyvää väkeä, eri ikäisiä henkilöitä, joiden toimintaan vaikuttaa väliaikainen tai pysyvä vamma tai uuteen kulttuuri- ja kieliympäristöön vasta tutustumassa olevia maahanmuuttajia. DfA-käsitteen tulkitaan joskus tarkoittavan suunnittelua kaikille ja erityisesti suunnitteluammattilaiset ovat nähneet tällaisen orientaation hankalaksi [74].

Toinen EU:n lanseeraama termi on e-Accessibility, joka tarkoittaa kaikkien käyttäjien integroimista tietoyhteiskuntaan. Kaikilla käyttäjillä tarkoitetaan erityisesti niitä, jotka ovat vanhempia, vammaisia tai ihmisiä, jotka ovat toimintarajoitteisissa ympäristöissä. Tähän

tavoitteeseen päästään suunnittelemalla tuotteet ja palvelut siten, että ne saavuttavat mahdollisimman laajan käyttäjäkunnan. [32]

Universal Design on ennen kaikkea Yhdysvalloissa ja Japanissa käytössä oleva käsite, jonka taustalla on korostuneesti ihmisoikeuksiin liittyvä näkökulma. Englannissa käytetään yleisesti käsitettä Inclusive Design. [74]

Käytettävyys (engl. usability) kuvaa sitä, kuinka hyvin käyttäjät kykenevät käyttämään järjestelmän toiminnallisuutta. Hypertekstijärjestelmässä käytettävyys riippuu käyttöliittymästä, jonka muodostavat sekä käytettävä ohjelma että sivun sisältö ja rakenne. Käytettävyyden osa-alueita ovat opittavuus, käytön tehokkuus, muistettavuus, virhetilanteiden välttäminen ja käyttäjätyytyväisyys. [55]

Käytettävyys kuvaa missä määrin tietty käyttäjäjoukko pystyy käyttämään tuotetta tavoitteensa saavuttamiseksi tehokkaasti, tuloksellisesti ja tyytyväisesti [33]. Käytettävyyttä tarkastellaan yleensä rajattuna tiettyyn kohderyhmään, mikä ei välttämättä kata erityistilanteita tai kattaa vain tietyn erityiskäyttäjäjoukon, esimerkiksi matkapuhelinkäyttäjät. Käytettävyys korostaa usein hyvää toimivuutta tyypillisissä käyttötilanteissa, kun taas saavutettavuus tähtää toimivuuden varmistamiseen myös epätyypillisissä käyttötilanteissa [40]. Saavutettavuus voidaan myös määritellä käytettävyyden osa-joukkona, jossa käyttäjäjoukko on määritelty mahdollisimman laajaksi, käyttötilanteet vaihteleviksi ja käytön tehokkuus tärkeämmäksi ominaisuudeksi kuin käyttäjätyytyväisyys [75, s. 7-8]. Yleinen saavutettavuus verkkosivuilla parantaa useimmiten myös käytettävyyttä, mutta hyvä käytettävyys ei välttämättä takaa saavutettavuutta.

Saavutettavalla suunnittelulla pyritään kattamaan kaikenlaiset käyttötilanteet. Verkkosivujen saavutettavuus ei kuitenkaan tarkoita sitä, että kaikki käyttäjät näkevät tai kokevat sivut samalla tavalla. Saavutettavuus tarkoittaa sitä, että sivuston kaikki sisällöt (teksti, kuvat, multimedia) ovat tavoitettavissa [87, s. 130]. Saavutettavuuden ja käytettävyyden tutkimusalueilla on kuitenkin paljon yhteisiä piirteitä, koska molempien tavoitteena on pyrkiä verkkoympäristön mahdollisimman helppoon käyttöön. Taulukkoon

(Taulukko 1) on koottu esteettömyyden, saavutettavuuden ja käytettävyyden keskeisimmät toiminta-alueet ja huomion kohteet.

	Esteettömyys	**Saavutettavuus**	**Käytettävyys**
Käyttäjät	Pysyviä fyysisiä tai psyykkisiä toimintarajoitteita.	Pysyvät ja väliaikaiset toimintarajoitteet. Erilaiset käyttäjät.	Valtavirtaa tai määrättyä ryhmää edustavat käyttäjät.
Käyttötapa	Avustavat tekniikat ja ohjelmat.	Kaikki vaihtoehtoiset käyttötavat.	Yleisimmät käyttötavat.
Aistit	Ääni- ja tuntoaistit sekä puutteellinen näkökyky.	Ääni- ja tuntoaistit sekä heikentynyt tai väliaikaisesti rajoittunut näkökyky.	Näköaisti.
Käyttöä rajoittava tekijä	Käyttäjän kyvyt. Rajoitteiden huono huomiointi sivun toteutuksessa.	Käyttäjän kyvyt tai tekniset rajoitteet. Sivun toteutuksen keskittyminen tiettyyn ympäristöön.	Epäintuitiivinen käyttöliittymä.
Ympäristöjen erilaisuus ja muuttuvuus	Erilaisia ympäristöjä melko määrätyillä ominaisuuksilla.	Monenlaiset ympäristöt erilaisilla ominaisuuksilla.	Määrätty ympäristö hieman vaihtelevilla ominaisuuksilla.
Tavoite	Esteiden poistaminen.	Kohtuullinen toimivuus kaikkialla.	Käyttöliittymän parantaminen.

Taulukko 1. Käsitteiden keskeiset toiminta-alueet ja huomion kohdistuminen

2.2 Saavutettavuuden arviointi

Saavutettavuuden mittaaminen on haasteellista, sillä yhtä oikeaa arviointitapaa ei ole olemassa. Käytön tehokkuuden vertailuja perustilanteen ja useiden erityistilanteiden välillä voidaan kuitenkin tehdä erilaisilla menetelmillä. W3C:n mukaan saavutettavuuden arviointimenetelmiin kuuluvat [19]

- alustava arviointi,
- standardienmukaisuuden arviointi,
- tuotanto- ja ylläpitoprosesseihin vaikuttava arviointi,
- arviointi käyttäjien avulla,
- automaattinen arviointi ja
- arviointi kehittäjäryhmän tai asiantuntijan avulla.

Saavutettavuuden ymmärtäminen vaatii tietoa perinteisistä poikkeavista ohjelmista ja laitteista ja näitä hyödyttävistä suunnitteluteknikoista. Arvioinnin tueksi voidaan ottaa sopiva saavutettavuusohjeisto. Osa suosituksien arvioinneista voidaan tehdä automaattisilla tarkistuksilla. Asiantuntijan avulla voidaan tehdä kattava arvio lyhyellä aikavälillä. Käyttäjätestien avulla voidaan havaita todellisia ongelmia ja oppia erilaisista käyttötavoista. [19]

2.2.1 Alustava arviointi

Yksinkertaisimmillaan verkkosivun suunnittelija tai suunnittelijaryhmä arvioi sivuston toimivuutta erilaisilla selaimilla ja selainasetuksilla. Esteettömyyttä voidaan testata esimerkiksi avaamalla testattava sivusto Lynx-tekstiselaimeen ja jos sivusto näkyy ja toimii järkevästi, niin tällöin se on luultavasti riittävän saavutettava useimmille toimintarajoitteisille käyttäjille. [54, s. 311]

Tarkemmassa arvioinnissa voidaan käyttää useita erilaisia selaimia ja muuttaa selaimien asetuksia erilaisiksi. Sivuston suunnittelija voi testata esteettömyyttä kytkemällä kuvien latauksen pois päältä, sammuttamalla äänet, muuttamalla tekstin ja ikkunan kokoa, kokeilemalla sivustoa mustavalkoisena ja ohjaamalla sivua ainoastaan näppäimistöltä.

Edelleen sivuja voi kokeilla ääniselaimilla ja automaattisilla arviointityökaluilla paremman kattavuuden saavuttamiseksi. [19]

2.2.2 Käyttäjätestit

Paras tapa olisi testata sivustoa useilla henkilöillä, joilla on erilaisia toimintarajoitteita. Esimerkiksi värit ja kuvat pitäisi testauttaa värisokealla henkilöllä. Tämä ei ole kuitenkaan kovin käytännöllinen tapa, sillä toimintarajoitteita on hyvin monenlaisia [54, s. 311]. Kattavan kontrolloidun saavutettavuustestauksen järjestäminen erilaisilla käyttäjillä voi olla monimutkaista ja kallista. Käyttäjätestausta ei välttämättä voida myöskään järjestää sivuston suunnittelun tuotantovaiheessa.

2.2.3 Saavutettavuussuositukset

W3C on julkaissut suosituksia saavutettavuuden varmistamiseksi ja parantamiseksi eri osa-alueilla tarkistuslistojen muodossa. Saavutettavuussuosituksia on tuotantovälineille ja asiakasohjelmille, mutta sivuston suunnittelijaa koskee eniten verkkosisältöjen saavutettavuussuositus Web Content Accessibility Guidelines 1.0 [13] ja version 2.0 luonnos [7].

WCAG määrittää sen millaisia verkkosivuja käyttäjän pitäisi tehdä saavuttettavuuden varmistamiseksi, mutta toisaalta se määrittää myös sitä, mitä tuotantovälineiden pitäisi tuottaa. Perusperiaatteita ovat [13]

- standardienmukaisuus,
- riippumattomuus käyttäjän kyvyistä,
- riippumattomuus laitteista tai ohjelmista.

Saavutettavuussuosituksien mukaan tavoite on tehdä käyttöliittymistä käsiteltäviä, verkon sisällöistä ymmärrettäviä, havaittavissa olevia [7] ja navigoitavia sekä varmistaa verkkosivujen mukautuminen erilaisten käyttäjän tarpeisiin [13, luku 2].

Arviointimenetelmien avulla on ensisijaisesti tarkoitus parantaa saavutettavuutta edelliseen versioon nähden. Suosituksia voidaan pitää käytettävyysheuristiikkoja vastaavina.

Ohjeiden avulla voidaan myös jossain määrin mitata saavutettavuuden tilaa ja vertailla sitä sivustojen välillä. Jos tiettyjä ohjeita on noudatettu, niin sivustolle voidaan antaa luokitus. Näitä ovat esimerkiksi Web Content Accessibility Guidelines 1.0 -esteettömyyssuosituksen A-, AA- ja AAA-luokitukset [13, luku 5]. Luokituksen saamiseksi on toteutettava tietyt ohjeet täsmällisesti. Yksikin virhe pudottaa luokitusta. Ensimmäisen tason luokitus kattaa välttämättömien, toinen taso suositeltavien ja kolmas taso toivottavien ohjeiden noudattamisen. Näitä luokituksia ei kuitenkaan myönnä mikään erityinen taho. Luokituksen toteamiseen vaaditaan arviointityökalujen käyttötaitoa, merkintäkielien tuntemusta ja tietotaitoa verkkosivujen esteettömyydestä [19].

W3C:n ohjeet eivät ole kuitenkaan ainoita mahdollisia suosituksia, vaikkakin suurin osa muista suosituksista perustuu niihin. Muita saavutettavuussuosituksia ovat esimerkiksi Yhdysvaltojen kuntoutuslain pykälä 508, IBM- ja Novell-yritysten omat esteettömyyskriteerit, Näkövammaistahojen testausohjeet verkkosivuille ja -palveluille [36], Mobile Best Practices -suositus [66] ja Device Independence Principles [25]. On mielenkiintoista huomata, että vaikka nämä suositukset eroavatkin toisistaan hieman, niin niissä on paljon yhteisiä piirteitä. Erityisesti mobiiliympäristöjen käytettävyysohjeet ovat hyvin lähellä esteettömyyssuosituksia.

2.2.4 Automaattinen arviointi

Automaattinen arviointi perustuu yleensä myös saavutettavuussuosituksien tarkistuslistoihin, joista on valikoitu sopivat, koneellisesti tarkistettavissa olevat asiat. Diskreettien luokitusten rinnalle on myös kehitetty automaattisiin tarkistuksiin perustuva reaaliarvoinen mittari. Web Accessibility Barrier -pisteytys painottaa virheiden vakavuuksia, huomioi väärien hälytysten vaikutuksen ja sivujen määrän [88].

Automaattinen arviointi on käsin tehtävään arviointiin verrattuna kustannustehokasta, kuluttaa vähän aikaa ja tulokset ovat yhtenäisempiä. Lisäksi asiantuntijoiden tarve vähenee ja saavutettavuusarviointi voidaan liittää paremmin suunnitteluprosessiin [88]. Automaattisen arvioinnin avulla voidaan kasvattaa verkkosivujen tekijöiden tietoisuutta saavutettavuudesta.

2.3 Saavutettavuusongelmien historia

Tutkimukset osoittavat, että useissa sivustoissa ei ole otettu edes perustasolla huomioon saavutettavuussuosituksia. Esimerkiksi vain 19 % viidestäsadasta yleiseen terveystietoon keskittyvästä sivustosta saavutti eräässä tutkimuksessa W3C:n saavutettavuussuosituksien minimitason [18]. Vastaavan alan tutkimus osoitti, ettei yksikään 108 sivustosta selvinnyt puhtain paperein minimitasosta [89]. Eri sektoreiden verkkosivujen esteettömyyttä vertailevassa amerikkalaisessa tutkimuksessa esteettömyyden minimitason saavutti 26% liittovaltion, 11% voittoa tavoittelemattomien organisaatioiden ja 6% kaupallisten yritysten verkkosivustoista [51]. Pitkittäistutkimus vuosina 1997-2002 osoitti verkkosivustojen saavutettavuuden jatkuvaa heikkenemistä useissa amerikkalaisissa yliopistoissa [31]. Suomessa tehdyssä kuntien esteettömyyttä kartoittavassa selvityksessä 190 kunnasta vain 12 kuntaa ilmoitti huomioivansa strategiatasolla viestinnän esteettömyyden. Vain kymmenessä kunnassa oli tehty esteettömyyskartoitus, joka koski verkkoviestintää. Viestinnän esteettömyys oli kaikista vähiten huomioitu esteettömyyden osa-alue [50].

Vaikka standardointi on ollut voimakkaasti esillä, niin silti monet kaupalliset yritykset käyttävät vanhoja monimutkaisia tekniikoita (esim. esityskeskeiset taitot ja selainkohtaiset komentosarjat), joiden käytännöt ovat laajalle levinneitä [87, s. 49]. Saavutettavuuden huomioiminen vaatisi oikeastaan vain muutamia lisämääreitä ja sivuston uudelleenorganisointia. Ongelma voi johtua tietoisuudesta, motivaatiosta, kyvyistä, ajasta tai muista resursseista [76].

Tutkimuksien mukaan verkkosivujen kehittäjien pitäisi pystyä parantamaan saavutettavuutta tarkistuslistojen ja automaattisten tarkistustyökalujen avulla [18]. Saavutettavuus paranee huomattavasti sivustolla, jos kehittäjät yksinkertaisesti ajattelevat suunnitteluvaiheessa näitä näkökohtia. Useat verkkosivustojen kehittäjät ovat tietoisia saavutettavuussuosituksista ja tarkistusohjelmista. Useimmat suhtautuvat positiivisesti ajatukseen saavutettavuuden kehittämisestä, mutta eivät voi tehdä asialle tarpeeksi ajanpuutteen, asiakkaan tai johdon tuen puutteen tai puutteellisten työkalujen vuoksi [42].

Heikko saavutettavuuden nykytilanne saattaa johtua verkkosivujen tuotannon historiasta. Verkkosivujen suunnittelijat pitävät kiinni vanhoista suunnitteluperiaatteistaan ja tuottavat

sivuja WYSIWYG-tuotantovälineillä paperisivujen tapaan taittaen. Matala abstraktion taso tekee tuotantoprosessista helpompaa, koska graafisesti orientoituneet suunnittelijat voivat nähdä valmiin sivun esityksen suoraan ilman kattavia muunnoksia. Esitysmuotoon keskittyvästä suunnittelusta on kuitenkin se haitta, että sisältö ei ole käytettävissä uudelleen muissa konteksteissa. Näkövammaisten ohjelmilla tai eriresoluutioisilla näytöillä on vaikea saada esitysorientoituneita dokumentteja toimimaan. [46, s. 29]

Riippuvuus vanhoista tuotantoväleistä, jumiutuminen vanhoihin komentosarjatekniikoihin tai kiire uusien palvelintekniikoiden opettelussa on saattanut johtaa verkkosivustandardien unohtamiseen. Uusimmilla tuotantovälineillä voitaisiin saada standardin mukaista jälkeä, mutta käytettäessä vanhoja tapoja saadaan edelleenkin aikaan epästandardeja ja esteellisiä sivuja. [87, s. 76-78]

WYSIWYG-tuotantovälineiden ongelmat juontuvat selaimien kehityshistoriasta. Viime vuosikymmenellä selaimien välillä oli suuria epäyhteensopivuuksia, konflikteja standardien kanssa ja omia selainkohtaisia komentosarjakieliä. Epäyhteensopivuus johti Dreamweaver- ja GoLive-tuotantovälineiden suosioon, koska ne hoitivat ikävän yhteensovittamistyön. Näiden ohjelmien tuottama lähdekoodi ei valitettavasti ole ollut kovinkaan standardimukaista tai saavutettavaa.

Nykyään Dreamweaver ja GoLive ovat paljon yhteensopivampia standardien kanssa. Microsoft FrontPage –ohjelman tuotokset on tarkoituksella suunniteltu toimimaan ainoastaan Internet Explorer –selaimessa, samoin kuin Microsoft Wordillä tehdyt sivut. Tietoisuus standardeista jää usein markkinavoimien jalkoihin, sillä yritykset haluavat ja osaavat myydä tuotteitaan. [87, s. 90-93, 117-120]. Isojen kaupallisten yritysten sivustojen epästandardit ratkaisut eivät välttämättä johdu sivustojen ylläpitäjistä vaan siitä, että sisällönhallintajärjestelmät ja tietokantaratkaisut ovat vanhanaikaisia tai kolmannen osapuolen tuottama sisältö aiheuttaa ongelmia [87, s. 134-135].

2.4 Syitä saavutettavuuden parantamiseksi

Seuraavissa luvuissa kuvataan sitä, miksi saavutettavuuden huomioiminen on tärkeää.

2.4.1 Lait ja suositukset

Useiden maiden lakeihin on kirjattu erityisesti vammaisten oikeuksia koskevia säädöksiä. Nämä pykälät vaikuttavat epäsuorasti myös verkkosivujen esteettömyyteen. Suomen perustuslaki [23, 6 §] vaatii tasa-arvon noudattamista:

> *Ketään ei saa ilman hyväksyttävää perustetta asettaa eri asemaan sukupuolen, iän, alkuperän, kielen, uskonnon, vakaumuksen, mielipiteen, terveydentilan, vammaisuuden tai muun henkilöön liittyvän syyn perusteella.*

Vammaisten oikeuksia koskevissa lakipykälissä viranomaisia velvoitetaan ehkäisemään epäkohtia, jotka rajoittavat vammaisen henkilön toimintamahdollisuuksia. Samanlaisia syrjinnän kieltäviä lakipykäliä on lähes kaikissa länsimaisissa valtioissa [16, s. 17]. Vammaisten toimintamahdollisuuksien yhdenvertaistaminen on peräisin Yhdistyneiden kansakuntien kansainvälisestä toimintaohjelmasta vuodelta 1982, jossa on kiinnitetty myös erityistä huomiota uusien informaatio- ja viestintäteknologioiden saavutettavuuteen [75, s. 33-34]. Myöhemmin YK on ohjeistanut jäsenvaltioitaan informaation ja viestinnän saavutettavuudesta muun muassa seuraavasti [79]:

- Valtioiden tulisi kehottaa medioita, erityisesti televisiota, radiota ja uutislehtiä, tekemään palveluistaan saavutettavia.
- Valtioiden tulee varmistaa, että uusia tietokoneistettuja informaatio- ja palvelujärjestelmiä tarjotaan suurelle yleisölle ja ne tehdään joko alusta alkaen tai myöhemmin mukautettuna saavutettaviksi vammaisille.
- Valtioiden tulisi kehittää strategioita, joilla informaatiopalvelut ja dokumentit saadaan saavutettaviksi erilaisille vammaisille. Kohopistekirjoitusta, nauhapalveluita, isoja tulosteita ja muita soveltuvia tekniikoita pitäisi käyttää kirjoitetun tiedon saavutettavuuden parantamiseksi.

Tämän päivän tietoyhteiskunnassa verkkoselailu on arkipäiväinen asia ja sen pitäisi olla kaikille mahdollista. Esteelliset verkkosivut asettavat kuitenkin vammaiset eriarvoiseen asemaan. Australiassa sokea henkilö haastoi Sydneyn olympialaisten järjestäjäorganisaation oikeuteen kisasivujen esteellisyydestä syrjintälakien perusteella. Oikeuden mukaan tietyt sivut oli järjestetty selvästi näkökykyisille toimivaksi, mutta

sokealle saavuttamattomaksi. Sivujen mukauttaminen esteettömäksi ei olisi ollut liian vaativaa tai kallista. [75, s. 37-39]

Yhdysvaltojen kuntoutuslakiin on kirjattu erityinen pykälä (Section 508 of the Rehabilitation Act) koskien liittovaltion viranomaisten ja heidän yhteistyökumppaneiden elektronisen informaation saavutettavuutta. Pykälän tarkoitus on poistaa esteitä informaatioteknologiasta, antaa uusia mahdollisuuksia vammaisille henkilöille ja kannustaa avustavien teknologioiden kehittämiseen. Pykälässä määritellään yksityiskohtaisesti, miten verkkosivujen esteettömyyden vaadittu minimitaso saavutetaan. [75, s. 50-51]

Yleisempi tapa valtioilla on antaa erilaisia suosituksia siitä, mitä verkkosivujen toteuttamisessa pitäisi ottaa huomioon. Euroopan Unionissa saavutettavuus on huomioitu eEurope 2002 -toimintasuunnitelmassa. Toimintasuunnitelman yksi tärkeimpiä osa-alueita on taata kaikille mahdollisuus osallistua tietotalouteen. Komission tiedonannon mukaan tärkeimpiä toimia ovat vammaisten ja ikääntyvien digitaalisen syrjäytymisen ehkäisy, jatkuva sitoutuminen Design for All -aloitteen tukemiseen, W3C:n verkkosisällön saavutettavuussuosituksien hyväksyminen julkisiin internet-palveluihin ja kehityksen seuranta eAccessibility-asiantuntijaryhmän toimin [20]. Uusissa EU-poliittisissa ohjelmissa julkisten verkkosivujen esteettömyyden kehittämistä jatketaan, tuetaan W3C:n saavutettavuussuositusten toista versiota ja pyritään kehittämään yhteisiä arviointi- ja sertifiointimenetelmiä. Elinkeinoelämää rohkaistaan vapaaehtoisesti kehittämään esteettömiä verkkoratkaisuja, erityisesti esteettömyyttä tukevien sivunlaadintaohjelmien kehittämistä [21].

Suomessa ei ole myöskään suoraa lainsäädäntöä, joka määrittää verkkosivujen saavutettavuuden kriteerejä. Julkishallinnon verkkopalvelujen saavutettavuutettavuudesta on olemassa suositus ”JHS 129 Julkishallinnon verkkopalvelun suunnittelun ja toteuttamisen periaatteet”, jonka tarkoituksena on opastaa viranomaisia verkkopalveluiden suunnittelussa, toteutuksessa ja hankinnassa. Suositus perustuu muun muassa valtiovarainministeriön Julkisten verkkopalvelujen laatukriteerit -julkaisuun ja W3C:n saavutettavuussuosituksiin [35]. Vanhasen hallituksen laajakaistastrategian eräänä osana

on vammaisten ja vanhuksien tietoyhteiskuntavalmiuksien ja -mahdollisuuksien parantaminen, jota toteuttaa ”Kohti esteetöntä viestintää” -toimenpideohjelma [48]. Ohjeita esteettömyyden toteuttamiseksi verkkosivuilla julkaisevat myös erilaiset organisaatiot, kuten TIEKE Tietoyhteiskunnan kehittämiskeskus, Suomen Design for All -verkosto (Stakes) ja vammaisjärjestöt.

2.4.2 Palvelun laadun parantuminen

Yleinen virhekäsitys on, että sivuston visuaalinen ilme heikkenee, jos sivustosta tehdään saavutettava. Tutkimukset ovat osoittaneet päinvastaista. Kun 51 toimintarajoitteista käyttäjää testasi sataa erilaista verkkosivustoa, huomattiin, että kaikkein saavutettavimpia olivat ne sivustot, joissa oli myös hyvä visuaalinen ulkoasu ja vaativa sivun taitto. [65]

Toinen virhekäsitys on se, että saavutettavuudesta on hyötyä vain fyysisesti tai psyykkisesti toimintarajoitteisille ihmisille. Yleisesti ottaen samojen menetelmien käyttämisestä on hyötyä myös [87, s. 339-340]

- kämmentietokoneiden tai matkapuhelinselaimien käyttäjille,
- ikääntyville ihmisille,
- väliaikaisesti loukkaantuneille,
- erikoisselaimilla varustettujen julkisten verkkopisteiden käyttäjille ja
- hakukonetulosten optimoijille.

Saavutettavien sivujen tuottamisessa on kyse hyvin pitkälle siitä, että tuotetaan standardien mukaisia sivuja tarkoituksenmukaisesti. Saavutettavuussuositukset kehottavat käyttämään W3C:n tekniikoita määritysten mukaisesti. Standardit sisältävät itsessään saavutettavuusominaisuuksia, jotka on otettu huomioon avoimessa suunnitteluprosessissa. [13, luku 6.11]

Standardeja noudattamalla sivujen yhteensopivuus eri selaimien ja selainversioiden välillä paranee merkittävästi. Standardienmukaisuus mahdollistaa todellisuudessa yhden sivutoteutuksen lukemisen kaikkialla useiden selainversioiden sijaan [16, s. 22-23]. Ruudunlukuohjelmia, kämmentietokoneita, teksti- ja mobiiliselaimia käyttävät hyötyvät, sillä sivuista ei tarvitse tehdä erillisversiota jokaista laitetta varten [87, s. 60-68, 150-151].

Tulostusasettelu ja projektorilta näyttö onnistuvat sisältöön koskematta. Sivujen toimivuus ja hyödynnettävyys paranee nykyisten ja tulevien verkkotekniikoiden kanssa [81][87, s. 147-148].

2.4.3 Käyttäjäryhmän kasvu

Väestöennusteen mukaan vuonna 2030 Suomessa on 65 vuotta täyttäneitä yli 600 000 enemmän kuin nyt. Tämän ikäluokan prosentuaalinen kasvu on peräti 80 prosenttia. Tuolloin joka neljäs väestöstämme on täyttänyt 65 vuotta. Elinajanodotteen kasvun odotetaan jatkuvan entiseen tapaan tai kiihtyvän. [78]

Oletetusta passiivisuudesta huolimatta osa ikääntyvistä ihmisistä on kiinnostunut internetin käytöstä. Seniorit ovat euroopan tasolla nopeasti kasvava tieto- ja viestintäteknologian käyttäjäjoukko [74]. Väestön ikärakenteen muutos näkyy viestintäalalla todennäköisesti lisääntyneenä viestiliikenteenä ja sähköisten palvelujen kulutuksen nousuna [48].

Suomessa näkövammaisia on noin 80000, joista valtaosa on ikääntyneitä ihmisiä. Suurin osa näkövammaisista on heikkonäköisiä. Vuosittain näkökyvyn menettää osittain tai kokonaan useampi tuhat henkilöä. Kuuroja ihmisiä on noin 30000 ja jossain määrin heikentynyt kuulo on todella lähes joka kuudennella henkilöllä. Noin 20000 aikuisella ja 30000 lapsella on vamma, joka aiheuttaa ongelmia puhutun tai kirjoitetun kielen kanssa. Eri tavoilla liikkumis- ja toimimisesteisiin kuuluu joka kymmenes suomalainen. [48]

Pelkästään Yhdysvalloissa on yli 30 miljoonaa ihmistä, joilla on sellaisia toimintarajoitteita, etteivät he voi käyttää verkkopalveluita normaaleilla syöttö- ja vastemekanismeilla (ts. hiirellä, näppäimistöllä ja näytöllä) [54, s. 298]. Arviolta 7,3 miljoonalla amerikkalaisella on ongelmia näössä, 6,9 miljoonalla on häiriöitä kuulossa, 6,3 miljoonalla on vaikeuksia käsien käytössä ja 2,9 miljoonalla on oppimisvaikeuksia [16, s. 11-13].

Vuoden 2005 lopussa värinäytöllisten GRPS-, WAP-, MMS- ja Java-ominaisuuksilla varustettujen matkapuhelimien määrä kasvoi Suomessa 2,5 miljoonaan kappaleeseen ja internet-selailun mahdollistavien älypuhelinten määrä 0,3 miljoonaan. 3G-liittymiä

arvioidaan Suomessa olevan lähes 0,6 miljoonaa vuonna 2007 ja älypuhelinten määrän kasvavan 0,7 miljoonaan. 3G-puhelinten levinneisyys kasvaa merkittävästi lähivuosina. [72]

2.4.4 Taloudelliset syyt

Yrityksille on haaste ymmärtää erilaisten ikääntyvien ihmisten tarpeita. He eivät ole vain nopeasti kasvava markkinoinnin kohderyhmä, vaan myös vaurain ikääntyvien joukko koskaan. Ikäihmiset omistavat Yhdysvalloissa enemmistön kaikesta yksityisestä omaisuudesta. [74]

On arvioitu, että Yhdysvalloissa vammaisilla henkilöillä on käytössään harkinnanvaraista tuloa yli 175 miljardin dollarin edestä. Hyvä saavutettavuus yrityksen verkkosivustolla parantaa mahdollisuuksia tavoittaa tämä kuluttajajoukko. Useilla vammaisilla on lisäksi suurempi motivaatio hankkia palvelunsa internetistä esimerkiksi liikunta- tai näkövammojen vuoksi. [75, s. 16-17]

Saavutettavuus parantaa myös liikkuvien verkon käyttäjien mahdollisuuksia vierailla yrityksen verkkosivustoilla. Suomen matkapuhelinverkon pakettikytkentäisten, liikkuvaan internet-käyttöön kelpaavien, datapalvelujen markkinoiden arvon arvioidaan olevan noin 42 miljoonaa euroa (13 % matkapuhelinmarkkinoista) vuonna 2006, mikä kuvaa ainoastaan tiedonsiirrosta veloitettuja maksuja, eikä niiden kautta välitettyjen palveluiden markkina-arvoa. [72]

Yleinen väärinkäsitys on, että saavutettavan sivuston tekeminen maksaa paljon. Yhdysvaltojen oikeudessa esteettömyyden kustannuksia on perusteltu jopa sillä, että erillisen äänimateriaalin tuottaminen ja välittäminen maksavat paljon, mikä ei ole totta, sillä ruudunlukijat toimivat puhesyntetisaattorien avulla [75, s. 331]. Saavutettavuuden perustaso ei vaadi kuin muutamien lisämäärityksien tekemistä, jonka kustannukset ovat pieniä verrattuna esimerkiksi muun verkko-ohjelmointityön kustannuksiin [87, s. 336-337]. Saavutettavuutta voidaan myös pitää palvelun lisäarvona, jolla voidaan erottautua muista tekijöistä ja joka tuo uuden ryhmän asiakkaita. [16, s. 9-10, 21-22] Esteettömyyttä voidaan pitää esimerkkinä yrityksen yhteiskuntavastuusta [75, s. 25].

Standardeja noudattava sivusto kuluttaa verkkokaistaa huomattavasti vähemmän kuin vanhoilla tavoilla toteutettu. Verkkokaistan kulutuksen vähentyminen tarkoittaa palveluntarjoajalle merkittäviä kustannussääntöjä ja sivujen käyttäjälle nopeampia latausaikoja [87, s. 30-33, 212]. Saavutettavan sivuston yleinen käytettävyys kasvaa ja jatkossa ylläpito helpottuu [75, s. 24].

Saavutettavuuden arvellaan parantavan tuloksia hakukoneissa, sillä sivuja indeksoivat hakurobotit toimivat ruudunlukijan tai tekstiselaimen tapaan. Hakurobottien toimintatapa vastaa oleellisesti sokean ihmisen selailutapaa. Sivun indeksoitavuus hakukoneessa paranee, kun saavutettavuus otetaan huomioon. [54, s. 303]

Siitä on muutamia selkeitä viitteitä, että jotkut hakukoneet arvostavat saavutettavuutta ja antavat saavutettavalle sivulle parempia hakutuloksia. Yahoo! ohjeistaa sivuston ylläpitäjää seuraavilla vinkeillä [86]:

- Käytä sivua kuvaavaa sivun otsikkoa.
- Käytä `meta`-elementtejä kuvailemaan kompaktisti sivun sisältöä.
- Pidä tekstit ja linkit HTML-lähdekoodissa kuvien sijaan.
- Kirjoita kuville `alt`-tekstit.

Hakukoneiden sivujen arvottamisperusteista ja hakujen täsmäämisestä ei anneta useinkaan selviä algoritmeja, joten varmaa tietoa ei ole, parantaako saavutettavuus myös hakutuloksia. Google-hakukoneen pisteytystä ja sivustojen saavutettavuuden suhdetta arvioinut tutkimus ei löytänyt näiden välille yhteyttä [89]. Google on äskettäin julkaissut hakukoneestaan kokeiluversion, joka on mukautettu sokeiden ja heikkonäköisten tarpeisiin. Google Labs Accessible Search järjestää hakutulokset sivun esteettömyyden perusteella. Painotettavia asioita ovat sivuston helppolukuisuus, epäolennaisen visuaalisen materiaalin vähäisyys ja näppäimistöohjauksen hyvä tuki. Normaali Google-haku etsii työtehtävän kannalta relevantteja dokumentteja [26]. Julkaisu voidaan tulkita siten, että normaali hakukone ei luultavastikaan painota saavutettavuutta hakutuloksissa.

Toisaalta hakukoneyritykset ohjeistavat omilla suosituksillaan sivustojen ylläpitäjiä hyvän ja onnistuneen indeksoinnin saavuttamisessa. Esimerkiksi Google kehottaa testaamaan

sivua Lynx-selaimella [27]. MSN Search opastaa pitämään sivun koon pienenä [52]. Näistä ohjeista löytyvät ylläolevan listan vinkit. Nämä ohjeet ovat hyvin yhteneväisiä standardien ja saavutettavuussuositusten kanssa [13, luvut 6.1, 6.13].

On ainakin selvää, että hakuroboteilta jää jotain olennaista tietoa saamatta, ellei kuvilla, kuvakartoilla, äänillä tai videoilla ole tekstivastineita. Hakurobotit ovat ohjelmia, jotka pystyvät vain hyvin rajoitetusti tunnistamaan visuaalista informaatiota. Kielen merkinnän ja selvän kielen käyttämisen teksteissä voidaan olettaa myös parantavan hakutuloksia, koska tällöin haut kohdistuvat paremmin sivuilta löytyviin sanoihin [13, luvut 6.4, 6.14].

3 ERILAISIA TAPOJA KÄYTTÄÄ VERKKOA

Saavutettavan verkkosivun toteuttamisessa on otettava huomioon useita tekijöitä, sillä tavoitteena on kelvollinen toimivuus kaikissa ympäristöissä. Verkkosivuston toimivuuden takaaminen normaaliolosuhteissa ei ole kovinkaan haasteellista. On huomattavasti kiinnostavampaa pohtia sitä, mitä suunnittelussa on otettava huomioon, että sivusto olisi käytettävissä myös normaalista poikkeavissa tilanteissa.

Ennen kuin voimme siirtyä käsittelemään tarkemmin teknisiä ratkaisuja, joilla saavutettavuutta voidaan parantaa (luvussa 4), on ymmärrettävä, millaisia erilaisia huomioitavia käyttötapoja ja -tilanteita on olemassa. Näihin kuuluvat muun muassa ne tavat, joilla vammaiset käyttävät verkkoa. Samankaltaisia menetelmiä tarvitaan, jos jokin aisti on heikentynyt, esimerkiksi iän myötä. Joskus tilanne saattaa estää tietyn aistin käytön yhtä aikaa verkon käytön kanssa. Käytössä voi olla myös jokin sellainen laite, jonka ominaisuudet eivät vastaa pöytäkoneisiin tottuneita.

Seuraavissa luvuissa jäsennetään niitä asioita, joilla on vaikutusta verkkosivun saavutettavuuteen.

3.1 Käyttäjien fyysiset ja psyykkiset rajoitteet

Fyysisesti ja psyykkisesti vammaiset tai yksilöt, joilla jonkin aistin toiminta on rajoittunut, pystyvät toimimaan yhteiskunnassa, kun nämä rajoitteet otetaan palveluiden suunnittelussa huomioon. Fyysisiä tai psyykkisiä rajoitteita ovat

- sokeus, näkövammaisuus tai heikkonäköisyys,
- värisokeudet,
- kuurous tai heikkokuuloisuus,
- motoriset häiriöt ja puhehäiriöt,
- neurologiset vammat,
- oppimisvaikeudet ja
- muut kognitiiviset vaikeudet.

Näkökyvyn rajoitteet

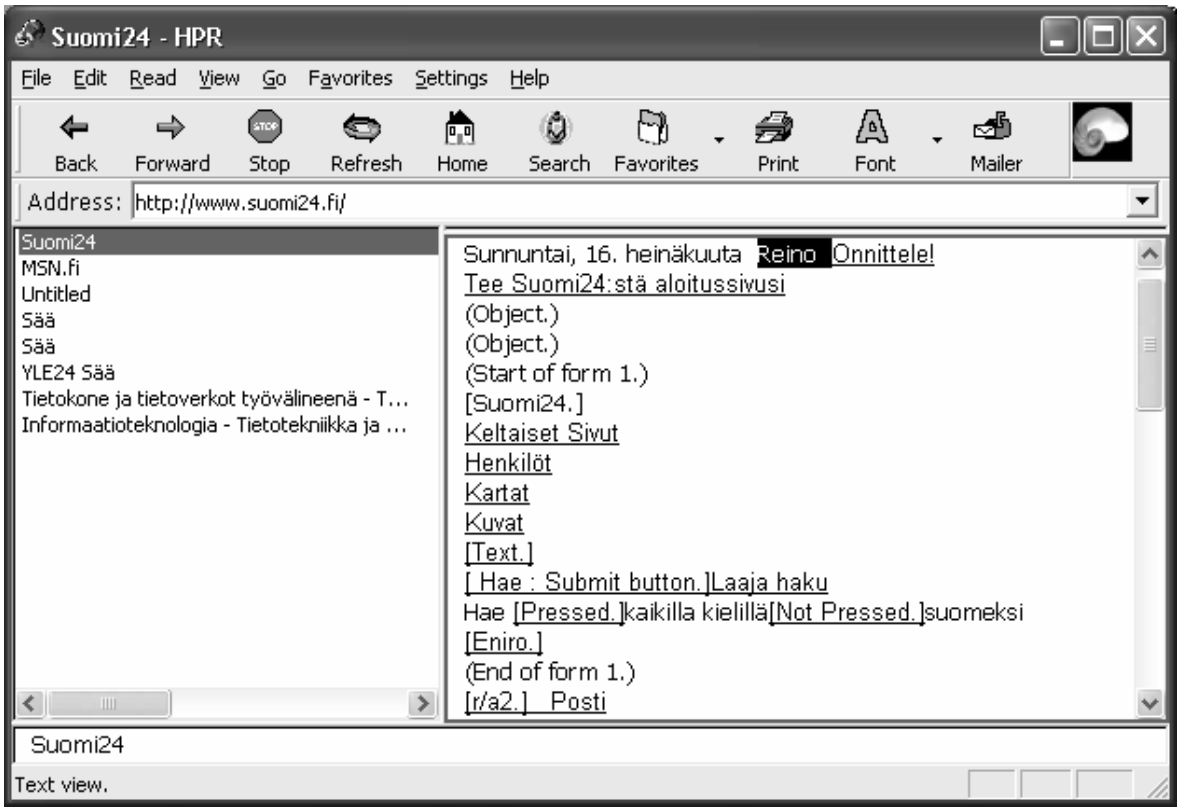
Näön ongelmia on erilaisia. Yleisimpiä ongelmia ovat taittovirheet, jotka ovat korjattavissa. Henkilöllä voi olla kuitenkin muita vaikeuksia hahmottaa pieniä kohteita. Näkökentässä voi olla muita puutteita, kuten esimerkiksi näön puuttuminen keskeltä näkökenttää, putkinäkö tai muu epäsäännöllinen näkökentän puutos. Putkinäköinen pystyy yleensä lukemaan hyvin, mutta esimerkiksi keskeisnäön puute aiheuttaa lukuongelmia. Karsastuksessa silmien kohdistuspiste on eri paikassa, mikä aiheuttaa kaksoiskuvia ja silmien nopeaa rasittumista. Silmien värveliikkeessä silmä ei löydä heti sopivaa tarkennuspistettä. [1]

Mykiön taittokyky vähenee iän myötä. Seurauksena on lähimmän selvästi näkyvän pisteen etääntyminen. Likipisteen etääntymisen nopein siirtymävaihe on 40-50 vuoden iässä. Noin 50-vuotiaalla likipiste on yli puolen metrin päässä. Lukulasit tarvitaan yleensä noin 45 vuoden iässä. [57]

Kuva 1. Verkkosivun selailu Windows XP -käyttöjärjestelmän ruudunsuurennosohjelman avulla.

Heikkonäköinen pystyy näkemään tai toimimaan normaalisti, mutta tarvitsee esimerkiksi verkosta lukiessaan tekstin suurennosta tai koko ruudun suurennosta. Jos näön häiriö on vähäinen, niin käyttäjälle voi riittää pelkkä tekstin suurennos, joka onnistuu suoraan selainohjelmasta [76]. Iso näyttö matalalla resoluutiolla voi myös riittää ratkaisuksi

ongelmaan. Koko ruudun suurennosta varten on olemassa erillisiä avustavia laitteita ja ohjelmia. Ruudunsuurentaja kasvattaa osan näyttöalueesta koko ruudun kokoiseksi, jolloin vain osa kokonaisuudesta näkyy. Tarkennuksen tasoa voidaan vaihdella yleensä tarpeen mukaan, sillä vähäisempi tarkennus voi olla välillä tarpeen kokonaiskuvan hahmottamisessa. Joissain ohjelmissa voi olla suurennetun kuvan lisäksi erillinen kokonaiskuva kokonaisuuden hahmottamista varten [75, s. 60-61]. Kuvassa (Kuva 1) on käytetty verkkosivun selailuun Windows XP -käyttöjärjestelmän Magnifier-ruudunsuurennosohjelmaa nelinkertaisella suurennoksella ja käänteisillä väreillä. Useimmat heikkonäköiset pitävät vaaleasta tekstistä tummalla pohjalla [76].



Kuva 2. IBM Home Page Reader -selain lukemassa ääneen `suomi24.fi`-portaalin etusivua.

Näkövammaisella on sen sijaan jokin vakava häiriö näkökyvyssä, joka haittaa merkittävästi arkipäiväistä elämää. Sokea ei kykene näkemään ollenkaan. Sokeista tai näkövammaisista suurin osa käyttää ruudunlukijoita tai ääniselaimia [16, s. 26-28]. Ruudunlukija on avustava ohjelma, joka hyödyntää käyttöjärjestelmän tai ohjelmien

tarjoamia saavutettavuusrajapintoja ja lukee näiden avulla käyttöliittymästä tai verkkodokumentista löytyvät tekstit ääneen. Ääniselain on ohjelma, jossa on itsessään ominaisuutena ääniohjaus, dokumentin ja käyttöliittymän osien ääneen lukeminen tai molemmat yhdessä [75, s. 54-59]. Kuvassa (Kuva 2) on IBM Home Page Reader -ääniselain, jonka graafista näkymää voidaan käyttää esimerkiksi saavutettavuuden testaamisessa [19].

Huomattavaa on, että ruudunlukijoiden puheen nopeutta voidaan säädellä ja useimmat sokeat selaajat kuuntelevat syntetisoitua puhetta jopa kaksinkertaisella nopeudella verrattuna tavalliseen keskustelunopeuteen. Käyttäjä navigoi sivulla näppäimistön ja avustajatekniikoiden navigointiapujen avulla. [16, s. 32-33]

Sokeista pieni osa käyttää kohopistekirjoitusta, jota kaikki kuurosokeat tarvitsevat verkosta saadun tekstin tulkitsemiseen. Tällöin muutama rivi kerrallaan teksti tulkitaan laitteelle, joka muodostaa pystyyn nousevista metallipinneistä Braille-järjestelmän mukaisia merkkejä. [16, s. 26-33]

Värinäössä voi olla myös puutteita. Täysin puuttuva värinäkö on erittäin harvinainen. Yleisin värinäön puute on puna-viher-värisokeus. Miehistä 7-8% on puna-vihervärisokeita [40]. Tämä tarkoittaa, ettei henkilö kykene erottamaan punaista ja vihreää. Tämä tapahtuu yleensä pienissä yksityiskohdissa, värit voivat erottua isoissa alueissa hyvin [1]. Puna-viher-värisokeus saattaa vaikeuttaa välivärien havaitsemista. Välivärejä ovat esimerkiksi beige, keltainen ja oranssi. Värien kirkkauksien erottelukyky on myös yksilöllistä [16, s. 202-205]. Värisokeat saattavat määrätä selaimen käyttämään sellaisia värejä, joilla he erottavat varmasti tietyt sivun osat toisistaan, mutta muuta avustavaa tekniikkaa ei tarvita [5].

Kuulon rajoitteet

Kuurot eivät kuule mitään ja käyttävät yleensä viittomakieltä kommunikointiin. Heikkokuuloiset voivat yleensä hyödyntää kuuloaan ja pystyvät puhumaan ymmärrettävästi. Useat heikkokuuloiset tai kuurot ovat joskus kuulleet puhuttua kieltä [16, s. 26-27]. Kuulo heikkenee iän myötä kuuloradan neuronien rappeutumisen ja

tuhoutumisen takia. Korkeiden äänien kuuleminen häiriytyy eniten, matalia ääniä kuullaan aivan niin kuin ennenkin [57]. Kuulo vaikuttaa verkon käyttöön vain vähäisesti, sillä verkko on pääosin visuaalinen media. Erillisiä avustavia tekniikoita ei tarvita [16, s. 31].

Motoriset häiriöt

Motorisista häiriöistä verkon käyttöä rajoittavat lähinnä häiriöt käsien tai käsivarsien toiminnassa. Näillä henkilöillä ei todennäköisesti ole kuitenkaan ongelmia näytöltä saadun informaation ymmärtämisessä. Motorisia häiriöitä saadaan kompensoitua sopivilla lisälaitteilla, joita ovat esimerkiksi näppäimistön päälle asetettavat sormien ohjastimet, painalluksien määrän tasaajat, kosketusnäytöt, jalkapedaalit ja ohjauspallot. [16, s. 27-31]

Ohjauslaitteina voidaan käyttää myös vaihtoehtoisia osoitusvälineitä, kuten esimerkiksi päällä tai suulla ohjattavaa sauvaa. Silmien liikkeen tai puheen tunnistuksen avulla voidaan toimia vuorovaikutuksessa verkkosivujen kanssa [5]. Puheen tunnistuksessa ohjaus tapahtuu rajoitetulla komentomäärällä, linkkien valinta linkkitekstin tai linkkinumeron perusteella ja tekstin syöttö kirjaimittain lausumalla [14]. Hiiren osoitinta voidaan myös ohjata joissain toteutuksissa sanallisesti lausumalla osoittimen liikkeen suunta ja haluttu matka [6].

Vaikeassa motorisessa häiriössä yksilö saattaa kyetä vain harvoihin tarkkoihin liikkeisiin, jolloin tarvitaan soveltuvaa kytkintä. Kytkin perustuu esimerkiksi painonappiin, pään nyökäytykseen tai liikkeentunnistuslaitteeseen, joiden avulla voidaan muodostaa yksittäisiä valintaoperaatioita. Tällaisen kytkinohjauksen (engl. switch access) kanssa voidaan käyttää esimerkiksi virtuaalista näppäimistöä, joka näyttää näytöllä näppäinvaihtoehdot ja josta voidaan valita haluttu merkki riittävän monella kytkimen painalluksella. Selainta voidaan ohjata myös toimintovalikolla, joka mahdollistaa lisäksi hiiritoimintojen läpikäynnin ja valitsemisen. Linkin tai lomake-elementin valinta verkkosivulla tapahtuu siis näppäimistöohjauksen tapaan sarkainnäppäintä käyttäen, mutta paljon hitaammin. [16, s. 30-31, 149-150, 279]

Kognitiiviset vaikeudet

Kognitiiviset vaikeudet ja oppimisvaikeudet liittyvät tarkkaavaisuuteen ja tiedon käsittelyprosesseihin. Näistä voi johtua esimerkiksi heikko kuullunymmärtäminen, visuaalinen hahmottaminen tai lukihäiriö. Näille käyttäjille voi olla hyötyä useiden käyttötapojen hyödyntämisestä, esimerkiksi tekstin kuulemisesta ääneen mielessä lukemisen lisäksi. Puhetta tai ääntä kuunnellessa voi olla hyötyä, jos on mahdollisuus seurata tekstitystä [5]. Lukihäiriöisten on huomattu hyötyvän hitaasta ääniselailusta ja ruudunsuurennoksista. Vähäistä näyttöä on myös siitä, että kuvailevasta äänipalvelusta olisi hyötyä tarkkaavaisuushäiriöihin [16, s. 35].

Neurologiset vammat voivat johtaa oppimisvaikeuksiin, kognitiivisiin vaikeuksiin ja motorisiin häiriöihin. Kehityshäiriöt voivat johtaa tekstin ymmärtämisvaikeuksiin. Henkilölle, jolla on muistihäiriöitä, voi olla välttämätöntä, että sivustolla on yhtenäinen ja johdonmukainen navigointirakenne. Kohtauksia aiheuttavien häiriöiden, kuten epilepsian, huomioimiseksi on sivustolta löydyttävä mahdollisuus kytkeä välkkyvät tekstit ja animaatiot pois päältä. [5]

3.2 Laitteiden tekniset rajoitteet

Viime vuosina verkon käyttö on tullut mahdolliseksi muillakin laitteilla kuin pöytätietokoneilla. Käyttäjät olettavat saavansa nykyään tärkeän informaation verkosta muillakin kuin perinteisillä mekanismeilla [25, luku 1.3]. Näitä vaihtoehtoisia laitteita ovat esimerkiksi

- kannettavat laitteet, joihin kuuluvat
  - matkapuhelimet (mobile phone),
  - multimedia- ja älypuhelimet (smartphone),
  - kämmentietokoneet (personal digital assistant),
  - paneelitietokoneet (web tablet) ja
  - kannettavat pelikonsolit (handheld game console),
- kannettavat tai kiinteät tietokoneet, joissa on erikokoisia näyttöjä,
- tulostimet ja projektorit,

- tekstipäätelaitteet,
- televisiot ja televisioon kytkettävät pelikonsolit.

Laitteiden tekniset ominaisuudet vaihtelevat kuitenkin sekä syöttö- ja ohjausmekanismien että näyttöjen toteutuksen, koon ja värien suhteen. Verkkosisällön tuottajat eivät voi enää luottaa siihen, että käyttäjät selailevat verkkoa yhdellä tavalla [25, luku 1.3]. Verkkosivu on toteutettava laitteistoriippumattomasti.

Seuraavissa kappaleissa kuvaillaan erilaisia ominaisuuksia, joita verkkoon kytkeytyvillä päätelaitteilla voi olla.

Tietokoneet

Tyypillisessä tietokoneessa on näyttö, jonka halkaisijan koko on tällä hetkellä noin 15-21”. Erityistapauksissa näytön koko voi olla pienempi tai suurempi. Näytön resoluutio voi vaihdella näyttötilasta riippuen 640×480 pisteestä jopa 2048×1536 pisteeseen. Yleisimmät tarkkuudet ovat kuitenkin 1024×768, 1280×1024 ja 1600×1200. Kuvasuhde on tällöin 4:3, mutta erityisesti kannettavissa laitteissa ovat yleistyneet laajakuvasuhteessa 16:9 olevat näyttöresoluutiot. Näyttöjen tarkkuuksissa on huomattavaa vaihtelua, eikä yhteen tilaan voi luottaa, sillä esimerkiksi vuonna 1999 suosituinta pistetarkkuutta (800×600) käytti vain noin puolet verkon käyttäjistä [54, s. 28].

Verkon selailussa voidaan käyttää monia erilaisia selaimia. Selainta ohjataan tyypillisesti normaalilla näppäimistöllä ja pöytähiirellä. Kannettavissa tietokoneissa hiiren tilalla voidaan käyttää sormen liikkeitä tunnistavaa tappi- tai tasohiirtä, joiden avulla saadaan lähes normaalihiirtä vastaava toimintatarkkuus. Pöytätietokoneissa on yleensä myös värimustesuihkutulostin tai mustavalkolasertulostin, jolla verkkosivut saadaan tarvittaessa paperille. Tietokone voidaan myös kytkeä videoprojektoriin, jolloin kuvaa voidaan katsella huomattavasti kauempaa.

Matka- ja älypuhelimet

Vanhoissa matkapuhelimissa internetin käyttö rajoittuu WML-sivujen selailemiseen puhelimen omalla WAP-selaimella. Hieman uudemmissa Java Micro Edition -sovelluksia tukevissa matkapuhelimissa on mahdollisuus käyttää Opera Mini -selainta, joka muuntaa

ulkoisen palvelimen avulla HTML- ja XHTML-verkkosivuja puhelimeen sopivaksi. Perusmatkapuhelimissa on yleensä maksimissaan megatavu muistia, alle 150×150 pikselin kokoinen näyttö maksimissaan 4096 värillä (12 bittiä) ja navigointiin käytetään numero- ja pikanäppäimiä (softkeys). Puhelimissa, joissa on digitaalinen kamera, saattaa olla hieman isompi näyttö, 65536 väriä (16 bittiä) ja joitain megatavuja muistia. Lisäksi verkkoselailua rajoittaa tietoliikenneyhteyden hitaus. Normaalilla GRPS-yhteydellä päästään noin 40kbit/s nopeuteen, mikä on pöytäkoneiden modeeminopeusluokassa.

Kuva 3. Matkapuhelimien verkkoselaimia simuloivia ohjelmia.

Älypuhelimet ovat matkapuhelimia, joissa on kämmentietokoneiden ominaisuuksia. Näissä puhelimissa on yleensä jokin puhelimeen sopiva käyttöjärjestelmä ja selainohjelma (esim. Opera tai Nokia S60 mobile browser), jolla voi suoraan lukea internetistä selaimelle sopivia sivuja. Älypuhelimien selaimet tukevat yleensä vähintään XHTML Mobile Profile -standardia, mutta usein ne mahdollistavat myös HTML- ja XHTML-sivujen selaamisen.

Älypuhelimissa on tyypillisesti 176 pikseliä leveä, yli 200 pikseliä korkea ja 65536 värinen näyttö [59]. Hieman uudemmissa älypuhelimissa näyttö voi olla 240×320 pikselin kokoinen. Muistia puhelimissa on 32 megatavua tai enemmän. Älypuhelimissa voi olla kosketusnäyttö, näytöltä käytettävä virtuaalinäppäimistö tai piirrinkirjoituksen tunnistus. Muutoin joudutaan käyttämään puhelimen näppäimiä. Älypuhelimiin saatavissa

selainohjelmissa voi olla mahdollisuus ääniselailuun ja ääniohjaukseen. Joskus verkkosivujen käyttäjä voi olla sellaisessa tilanteessa, että hän ei voi käyttää kaikkia aistejaan selailemiseen vaan joutuu käyttämään ääntä esimerkiksi autossa ajaessaan [54, s. 303]. Kuvassa (Kuva 3) on kolme erilaista älypuhelimen toimintaa simuloivaa ohjelmaa: Openwave Phone Simulator, Series 60 Content Authoring SDK v.2.0 ja Opera Mini Simulator. Näiden avulla voidaan testata sivun saavutettavuutta mobiiliympäristössä ilman varsinaista matkapuhelinta. Kuvassa on testattu `www.opera.com`-sivustoa, jossa teksti ja kuvat mukautuvat hyvin pienen näytön kokoon.

Kalliimmissa yrityspuhelimissa on yleensä kaikki normaalit näppäimet kattava pieni näppäimistö ja mahdollisesti tavallista puhelinta suurempi ja värikylläisempi näyttö. Älypuhelimien tiedonsiirrossa voidaan yleensä käyttää uusia nopean tiedonsiirron standardeja, kuten EDGE, WCDMA, 3G ja WLAN, joilla päästään laajakaistayhteyksien nopeusluokkiin.

Kämmen- ja paneelitietokoneet

Kämmen- ja paneelitietokoneet ovat älypuhelimia vastaavia laitteita, joiden pääasiallinen tarkoitus on toimia liikkuvana tietojen varastona ja organisoijana. Modernit PDA-laitteet kykenevät lisäksi muun muassa verkon selailuun Bluetooth- ja WLAN-yhteyksien avulla. Perinteisesti näissä laitteissa ei ole itsessään ollut puhelinominaisuuksia, mutta raja älypuhelimien välillä on häilyvä.

Näiden laitteiden näyttöjen koot voivat olla isompia kuin älypuhelimissa. Resoluutiot vaihtelevat 320×240 pisteestä jopa 800×480 pisteeseen 16-bittisillä väreillä. Muistia on yleensä 64 megatavua tai enemmän. Käytössä on älypuhelimia vastaavia selaimia, jotka tukevat yleisimpiä verkkostandardeja. Laitetta ohjataan kosketusnäyttöön osoitettavalla piirtimellä ja kirjoitus tapahtuu joko virtuaalinäppäimistöllä tai piirrettyjen kirjaimien tunnistuksella. Joissain malleissa voi olla näppäimistö tai mahdollisuus ääniohjaukseen.

Kannettavissa pelikonsoleissa, kuten Playstation Portable ja Nintendo DS, on myös täysi mahdollisuus verkon selailuun.

Televisiot ja pelikonsolit

Kodin viihdekeskuksiin tarkoitetut medialaitteet ja media-PC:t mahdollistavat nykyään internetin käytön televisiossa. Näistä konsepteista saatetaan käyttää nimitystä interaktiivinen televisio tai Web TV. Digiboksit ja kaapelimodeemit voivat sisältää myös mahdollisuuden selaimen käyttöön televisiossa.

Näiden laitteiden ominaisuudet vaihtelevat merkittävästi, mutta useimmissa on riittävästi tehoa täysipainoiseen verkon selailuun. Esimerkiksi MSN TV -medialaitteessa on Intel Celeron -prosessori, 128 megatavua muistia ja selaimena toimii Internet Explorer 6. Laitteet kytkeytyvät WLAN- tai laajakaistayhteydellä verkkoon. Ohjaukseen käytetään langatonta näppäimistöä ja kaukosäädintä. Laite kytketään televisioon antenni- tai S-Video-kaapelilla. Perustelevisiossa erottelukyky on noin 520×576.

Uusissa pelikonsoleissa, kuten Nintendo Wii ja Playstation 3, tulee olemaan mahdollisuus internet-selailuun. Pelikonsolit kytketään televisioihin, jolloin käytettävän näyttötilan voidaan arvella vaihtelevan normaalin television resoluutiosta HDTV:n parhaimpaan tarkkuuteen (1920×1080). Nintendon pelikonsolia ohjataan erityisellä kaukosäätimellä ja Playstation 3 -konsoliin voidaan kytkeä näppäimistö ja hiiri. Laajakaistainen tietoliikenne on mahdollista Ethernet, USB2, Bluetooth tai WLAN-liitäntöjen kautta.

Tekstipäätteet

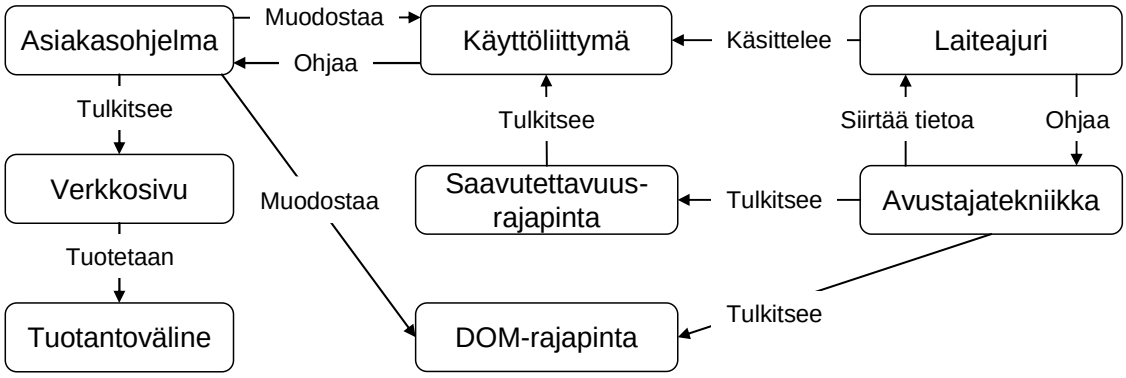
Tekstipäätteet ovat laitteita, joissa ei ole käytössä graafista käyttöliittymää ja ohjelmien vasteet näytetään määrätyn kokoisessa kirjainruudukossa. Tekstipäätteisiin voidaan laskea esimerkiksi kaukokirjoittimet, kuurojen tekstipuhelimet, pistematriisikirjoittimet ja ohjelmat tai laitteet, joilla voidaan ottaa tekstimuotoinen pääteyhteys toiseen koneeseen. Kuurot eivät käytä tekstipuhelimia tai vastaavia laitteita verkon selailuun, joten ainoastaan pääteyhteyksien tekstiselaimilla on käytännön merkitystä. [16, s. 264-268]

Tekstiselaimet ovat komentorivikäyttöliittymässä toimivia selaimia, jotka näyttävät sivut pelkkänä tekstinä linearisoidussa muodossa. Linkkitekstit saatetaan vahventaa ja tämän hetkinen paikka merkitä erilaisella värillä. Tekstiselaimia käytetään pelkästään

näppäimistöltä. Tekstiselaimet eivät tue grafiikkaa, CSS:ää, JavaScriptiä tai muita HTML:stä poikkeavia erillistekniikoita [75, s. 59-60].

3.3 Yhteenveto käyttötavoista

Verkkosivun toteutuksen näkökulmasta saavutettavuus muodostuu ketjusta, joka alkaa verkkosivun suunnittelijasta ja päättyy käyttäjään. Tässä välissä on monia erilaisia tekniikoita, ohjelmia ja laitteita, jotka pyrkivät vastaamaan erilaisten käyttäjien tarpeisiin. Verkkosivun ja ohjelmien tekijöiden on oltava tietoisia näistä vaiheista.



Kuva 4. Dokumentin selailuun vaikuttavat osat avustajatekniikoita käytettäessä.

Tuotetun verkkosivun käsittely riippuu hieman selainohjelmasta, mutta sitäkin enemmän tuotetun sivun rakenteesta ja rakennetta laitteille tulkitsevista rajapinnoista ja avustajatekniikoista. Kuvassa (Kuva 4) on havainnollistettu näiden osien välisiä suhteita.

Sokea selaaja saattaa esimerkiksi käyttää ruudunlukijaa, joka kykenee Microsoft Active Accessibility -saavutettavuusrajapinnan kautta tulkitsemaan ja käsittelemään Internet Explorer -selainohjelman muodostamaa käyttöliittymää. Ruudunlukija pystyy myös lukemaan verkkosivun rakennetta ja sisältöjä täysipainoisesti selaimen tarjoaman DOM-rajapinnan kautta. [75, s. 54-56]

Hyvään saavutettavuuteen tähdättäessä on siis huomioitava erilaiset verkon käyttötavat. Luvussa 3.1 esitettyjen fyysisten ja psyykkisten rajoituksien takia voidaan joutua käyttämään avustavia laitteita tai ohjelmia. Toisaalta taas joitakin rajoitteita varten riittää

tehdä sivusta sellainen, että se mukautuu käyttäjän tarpeisiin perusselaimilla ja -laitteilla. Luvussa 3.2 esitetyt tekniset rajoitteet johtuvat käyttötilanteesta ja sen hetkisistä heikommista resursseista.

Erilaiset käyttötavat voidaan luokitella karkeasti neljään pääluokkaan. Ensimmäinen ryhmä muodostuu selailutavoista, joissa *ei ole mahdollista käyttää visuaalista informaatiota*. Ryhmään kuuluu selailu ruudunlukijalla, ääniselaimella, kohopistekirjoittimella tai tekstiselaimella. Näissä tavoissa ei voida hyödyntää pikselipohjaista näyttöä ja sen tarjoamia navigointimahdollisuuksia, kuten esimerkiksi sivun silmäilyä ja hiiren kursorilla valitsemista. Luokkaan voidaan laskea myös matkapuhelimet, jotka ovat yksivärisiä ja joissa on pieni resoluutio.

Toinen merkittävä ryhmä muodostuu näkökykyä normaalisti hyödyntävästä selailutavasta, jota rajoittaa kuitenkin *poikkeava näytön koko tai tila*. Poikkeavuus voi olla esimerkiksi suuri tai pieni näytön koko, heikko värienerottelukyky tai käyttäjän tarpeisiin mukautettu näkymä. Tähän ryhmään kuuluvat käyttötilanteet, joissa käyttäjät tarvitsevat verkkosivun selailuun matkapuhelimia, älypuhelimia, kämmenlaitteita, televisioita, tulostimia, projektoreja, ruudunsuurentajia, tekstinsuurennosta tai isoja näyttöjä.

Kolmas luokka liittyy *erilaisiin ohjaustapoihin*, jossa erityisesti tiedonsyöttö tai sivun osien valinta ovat hitaita. Erilaisia ohjaustapoja ovat ääniohjaus, ohjaus kytkinlaitteella, erilaiset osoitinsauvat, kosketusnäytöt, näytöllä esiintyvät näppäimistöt ja tekstinsyötön ohjastimet.

Neljänteen luokkaan lasketaan kaikki muut käyttötavat, joissa ohjelmilla tai laitteilla ei ole rajoitteita. Tällöin on kuitenkin huomioitava *muut toimintarajoitteet*, joita ei voida helpottaa avustavilla tekniikoilla. Ryhmään kuuluvat käyttäjät, joilla on esimerkiksi lievä värisokeus, heikentynyt kuulo tai kognitiivisia vaikeuksia.

Luokittelu ei ole täysin erotteleva, sillä esimerkiksi käyttäjä, jolla on oppimisvaikeuksia, voi hyötyä erilaisista ohjaustatavoista, värisokea selaimen mukauttamisesta ja matkapuhelinkäyttäjä tekstuaalisesta selailusta. Alla olevaan taulukkoon (Taulukko 2) on kerätty erilaiset käyttötavat, syyt käyttötapaan ja käyttötavan aiheuttamat rajoitteet.

Käyttötapa	Luokka	Syy käyttötapaan	Käyttötavan rajoitteet
Ruudunlukija, ääniselailu, puhesynteesi	Ei-visuaaliset	o Sokeus o Heikkonäköisyys o Oppimisvaikeudet	o Sisältö saadaan rakenteesta lineaarisesti. o Navigoitavuus riippuu sivun rakenteesta. o Ulkoasumuotoilut katoavat. o Kuvat, videot ja animaatiot tarvitsevat tekstivastineet.
Kohopiste-kirjoitus	Ei-visuaaliset	o Sokeus o Kuurous	Kuten yllä, mutta o tekstiä luetaan hitaammin muutama rivi kerrallaan ja o vain osaa merkistöstä tuetaan.
Tekstipäätteet	Ei-visuaaliset	o Käyttöliittymän ja verkkoyhteyden rajoitteet	Kuten yllä, mutta o tekstiä saadaan useita rivejä kerralla ja o navigoinnissa voidaan hyödyntää rajallisesti visuaalisia vihjeitä.
Ruudun-suurentajat	Poikkeava näyttötila	o Näkövamma o Heikentynyt näkökyky	o Yleiskuva sivusta katoaa. o Lähiympäristön hahmottaminen verkkosivuilla heikkenee. o Sivun osien (esim. navigointipalkit) löytäminen hankaloituu. [75, s. 60-61]

Matkapuhelimet, älypuhelimet, kämmenlaitteet, televisiot	Poikkeava näyttötila, erilaiset ohjaus-tavat	o Verkkoselailu matkalla tai liikkeessä o Ei tietokonetta	Kuten yllä, ellei sivu ole mukautuva. Tällöinkin [66] o tekstin syöttö voi olla hidasta ja o halutun tiedon löytäminen sivustolta voi olla työlästä.
Selaimen mukauttaminen, isot näytöt, projektorit	Poikkeava näyttötila	o Värisokeus o Heikentynyt näkökyky o Normaalin näkökyvyn rajat	o Sivun osien visuaaliset vihjeet voivat kadota. o Kokonaiskuvan hahmottaminen voi hankaloitua. o Sivun taitto voi hajota, jolloin tekstin luettavuus heikkenee.
Kytkinohjaus, virtuaaliset näppäimistöt	Erilaiset ohjaus-tavat	o Motoriset häiriöt	o Tekstin syöttö hidasta. o Kohteen valinta on tehtävä sarkainjärjestyksessä. [16, s. 30-31, 150-151]
Osoitinsauvat, avustavat näppäimistöt	Erilaiset ohjaus-tavat	o Motoriset häiriöt	o Tekstin syöttö on hidasta. o Navigointi tapahtuu useimmiten näppäimistöltä.
Ääniohjaus	Erilaiset ohjaus-tavat	o Motoriset häiriöt o Äly- tai kämmenlaitteen ohjaus	o Tekstin syöttö tapahtuu kirjaimittain tai muilla tekniikoilla. [6] o Komentojen määrä on rajoitettu virhetunnistuksien takia. o Valintaoperaatiot ovat hitaampia kuin hiirellä. [14]

Kosketusnäytöt, piirrinsauvat, pallo- ja tappihiiret, tekstintunnistus	Erilaiset ohjaus-tavat	o Motoriset häiriöt o Äly- tai kämmenlaitteen ohjaus	o Tekstin syöttö voi olla hidasta riippuen käyttäjästä ja syöttömekanismista. o Kohteen osoittaminen ei välttämättä ole tarkkaa.
Graafiset selaimet tietokoneilla	Muut toiminta-rajoitteet	o Värisokeus o Heikentynyt kuulo o Kognitiiviset vaikeudet o Epilepsia	o Osa verkkosivun informaatiosta voi jäädä saavuttamatta. o Verkkosivut voivat aiheuttaa sairaskohtauksia.

Taulukko 2. Erilaiset käyttötavat ja näiden rajoitteet

4 SAAVUTETTAVIEN WWW-SIVUJEN TUOTTAMINEN

W3C:n julkaisemia verkkostandardeja ovat HTML, XHTML ja XML rakenteisille dokumenteille, CSS ja XSL tyylimäärityksille, XSLT rakenteisten dokumenttien muunnoksille, RDF metatiedon määrittämiseen, MathML matemaattisille kaavoille, SMIL multimediaesityksille ja PNG grafiikalle [11, luku 12]. Muita laajasti hyväksyttäjä tekniikoita ovat XML-pohjainen vektorigrafiikkakieli SVG, sivustojen sisältökuvauksiin käytettävä XML- ja RDF-pohjainen RSS, valokuvissa käytetty JPEG-formaatti ja standardoitu komentosarjakieli ECMAScript 262, jonka yleisimmästä toteutuksesta käytetään nimitystä JavaScript.

Muita WWW-ympäristöön integroituvia erillistekniikoita ovat PDF-dokumenttiformaatti, interaktiiviseen vektorigrafiikkaan keskittyvä Flash (Adobe Systems), Java-ohjelmointikielellä toteutettavat sovelmat (Sun Microsystems) ja erilaiset videoformaatit, kuten Quicktime (Apple), Windows Media (Microsoft) ja RealMedia (RealNetworks). Näiden tekniikoiden saavutettavuutta tarkastellaan lyhyesti luvussa 4.8.

WWW-sivustojen tuottamiseen vaikuttavat myös kehitysympäristöt ja palvelinpohjaiset tekniikat, kuten PHP (Zend), JSP (Sun Microsystems), ColdFusion (Adobe Systems), ASP ja ASP.NET (Microsoft). Nämä tekniikat ovat tarpeellisia, kun halutaan tehdä dynaamisesti tuotettuja, käyttäjän syötteisiin vastaavia tietokantapohjaisia sivustoja. Sivuston saatavuus, saavutettavuus ja selainyhteensopivuus riippuu kuitenkin lopputuotoksesta (eli siitä miten selainpohjaisia tekniikkoja käytetään). Verkkosivun pitäisi olla semanttisesti ja rakenteellisesti järkevää. Palvelinpohjaisia tekniikoita voidaan käyttää siten, että ne tuottavat standardien mukaista jälkeä. [87, s. 141-144]

Sivustosta saadaan normaalia saavutettavampi, kun käytetään muutamia erityisesti saavutettavuutta parantavia merkintöjä ja lisätietoja. Toivottavaa olisi, että tulevaisuudessa tehtäisiin työkalu, joka varastoisi näitä tietoja ja lisäisi ne automaattisesti tuotetuille sivuille. Tämä on jo nykyään mahdollista, mutta nämä sisällönhallintajärjestelmät ovat kalliita. [16, s. 354-355]

Ratkaisevaa saavutettavuuden kannalta on, mitä selaintekniikoita käytetään. Erilaisten standarditekniikoiden oikeaoppinen käyttö on jo sinänsä yksi saavutettavuuskriteeri [13, luku 6.11]. Muut tekniikat tarvitsevat yleensä erillisen laajennoksen. Näitä formaatteja ei pystytä käsittelemään normaaleilla ohjelmilla ja käyttöä helpottavilla apuohjelmilla. Tässä luvussa käsitellään pääasiassa HTML-, XHTML- ja CSS-tekniikoita ja niihin liittyviä saavutettavuusattribuutteja, koska ne ovat merkittävimmässä asemassa verkkosisällön saavutettavuuden kannalta. Kunkin aliluvun loppuun on koottu yhteenveto saavutettavuuteen vaikuttavista suunnitteluperiaatteista ja näistä hyötyvistä käyttötavoista.

4.1 Standardien ja syntaksin noudattaminen

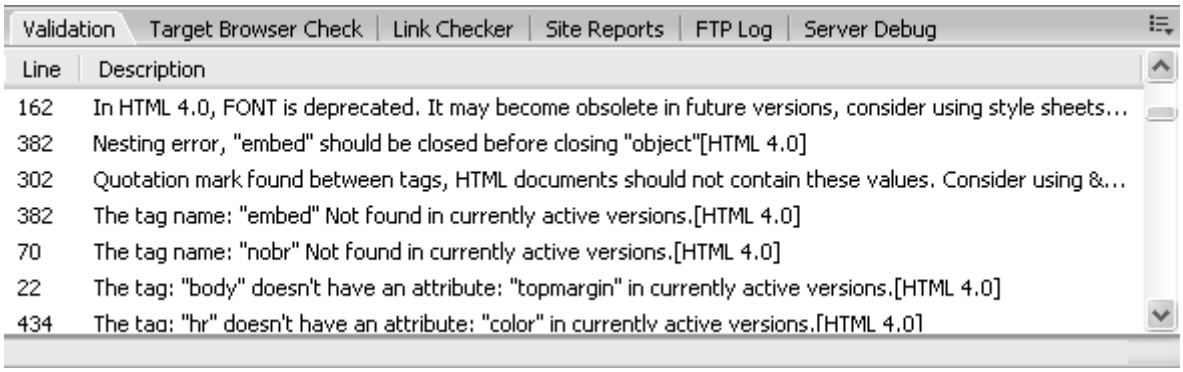
WWW-sivua tuotettaessa on valittava dokumenttityyppi, joka kuvaa dokumentissa käytettävän kieliopin. Dokumenttityyppi kuvataan esimerkiksi DTD, XML Schema, RELAX NG tai Schematron-kielellä. Tyypinmääritys kertoo selainohjelmalle, millä tavalla dokumenttia pitää tulkita. Sivun tekijälle dokumenttityyppi määrää, mitä elementtejä ja attribuutteja on käytettävissä ja miten niitä voi käyttää. Dokumenttityyppi kerrotaan sivun lähdekoodin alussa viittauksella sopivaan määritykseen. Esimerkiksi HTML 4.0-dokumenttityypin mukaisuus ilmoitettaisiin seuraavasti

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Dokumentin sanotaan olevan *validi*, kun se noudattaa määrätyssä dokumenttityypissä esitettyjä rajoitteita. Validoinnissa voidaan tarkistaa dokumenttityypissä määritettyjen sääntöjen pohjalta merkintöjen oikeellisuus, järkevä rakenne, sisällön tietotyyppien kelpaavuus, kytkennät toisiin dokumentteihin sekä muita erikoissääntöjä. Epävalidi dokumentti näkyy selaimissa ennustamattomasti ja mahdollisesti puutteellisesti [66, luku 5.4.7].

Dokumentin validiteetin voi tarkistaa automaattisesti verkosta löytyvällä W3C:n validaattori-palvelulla (http://validator.w3.org/). WWW-sivuja tuottavissa ohjelmissa voi olla myös sisäänrakennettuna tuotetun dokumentin validointi. Esimerkiksi Dreamweaver MX -ohjelmassa voi yhdellä valikkokomennolla käynnistää sisäänrakennetun HTML/XHTML-validaattorin, joka antaa raportin virheistä ja korostaa virheelliset kohdat

lähdekoodissa [75, s. 201]. Kuvassa (Kuva 5) on Dreamweaver 8 -ohjelman antaman validointiraportin osa `suomi24.fi`-portaalin etusivusta. Validointitoiminto huomauttaa myös vanhentuneista merkintätekniikoista.



Kuva 5. Dreamweaver 8 -ohjelman validointiraportti `suomi24.fi`-portaalin etusivusta.

XHTML-kielessä, kuten kaikissa muissakin XML-sovelluksissa, on lisäksi ilmoitettava ennen dokumenttityypin määritystä *XML-deklaraatio*, joka kertoo dokumentissa käytettävän merkistön ja että dokumentti on XML-määrittelyn mukainen.

```
<?xml version=”1.0” encoding=”ISO-8859-1”?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/2000/REC-xhtml1-20000126/DTD/xhtml1-
strict.dtd">
```

Dokumentti on *hyvin muodostettu* (engl. well-formed), jos se noudattaa XML-määrityksiä. Hyvin muodostettu XHTML-dokumentti eroaa perinteisestä SGML-pohjaisesta HTML-dokumentista erityisesti siten, että jokaisella elementillä on alku- ja loppumerkintä. Yksiosaiset elementit merkitään erikoismerkinnällä. Lisäksi sisäkkäisten elementtien alku- ja loppuosat eivät saa limittyä vaan niiden täytyy sijoittua oikealla tavalla sulkumerkkien tapaan. [81]

CSS-tyylimäärityksien oikeellisuus tarkoittaa sitä, että käytetyt merkinnät noudattavat standardia. Oikeaoppisten määrityksien on noudatettava annettua kielioppia ja notaatiota, joka riippuu tuetusta CSS-spesifikaation versiosta. [3]

4.1.1 Yhteensopivuus vanhojen selaimien kanssa

Dokumenttityypin merkitsemisellä on lisäksi toinen erityinen merkitys. Virallisesti dokumenttityypin valinnan ei pitäisi vaikuttaa kuin sivun oikeellisuuden tarkistamiseen, mutta useat selaimet päättelevät dokumenttityyppimäärityksen perusteella käytetäänkö *standardi*- vai *yhteensopivuustilaa* (engl. quirks mode). XHTML ja HTML 4 -dokumenttityypin tarkalla määrityksellä kytketään standarditila päälle, muutoin käytetään yhteensopivuustilaa [71].

Tila määrittää sen, millä tapaa sivu graafisesti hahmonnetaan (engl. render). Standarditilassa selain noudattaa tarkasti CSS2-spesifikaation laatikkomallia, kun taas yhteensopivuustilassa sivun taitossa ja ulkoasumääritysten käsittelyssä käytetään epästandardia tapaa, joka vastaa Internet Explorer 5 (Windows) ja Netscape Navigator 4 -selaimien toimintaa [2]. Tilan vaihtuminen dokumenttityypin mukaan mahdollistaa uusien selaimien yhteensopivuuden vanhojen sivujen kanssa [16, s. 47-48][87, s. 260-265].

Dokumenttityypin lisäksi XML-deklaraatiolla saattaa olla vaikutusta sivun hahmontamiseen. Esimerkiksi Opera 7.0 ja Internet Explorer -selaimissa määritys kytkee yhteensopivuustilan päälle [63]. Selainongelmien takia määritys kehotetaan jättämään pois kokonaan ja ilmoittamaan merkistö `meta`-elementillä [87, s. 157-159].

4.1.2 Dokumenttityyppi kannettaville laitteille

Kannettavia laitteita varten tehdyt verkkosivut on esitetty perinteisesti WAP-protokollamäärityksessä olevalla WML-kielellä. Tämä kieli ei ole suoraan yhteensopiva verkossa laajasti käytettyjen HTML- ja XHTML-kielien kanssa, mikä on rajoittanut sen käyttöä muissa kuin matkapuhelinympäristöissä. Nykyinen trendi kannettavien laitteiden standardeissa on kohti yhteistä internetiä, jossa samoja modulaarisia tekniikoita ja standardeja voidaan hyödyntää erilaisissa ympäristöissä [60, s. 8].

XHTML Mobile Profile on dokumenttityyppi, joka soveltuu resurssirajoitteisille verkkolaitteille, kuten matkapuhelimille, kämmentietokoneille, hakulaitteille ja televisioon kytkeytyville medialaitteille. XHTML-dokumenttityypin osajoukkona se on yhteensopiva

laitteiden kanssa, jotka tukevat täyttä standardia. XHTML Mobile Profile -standardia tukevien laitteiden on tyylisivuja ulkoasumuotoiluissa käytettäessä tuettava ainakin WAP CSS:ää, joka on CSS2-standardin osa-joukko muutamilla omilla lisäyksillä. [85]

4.1.3 Epästandardit esitystapaan viittaavat elementit

Muutamissa selaimissa (esim. Netscape 4 ja uudemmat) on mahdollisuus käyttää standardin ulkopuolelta löytyviä elementtejä. Näistä on erityisesti vältettävä `blink`- ja `marquee`-elementtejä, koska näiden toimivuus eri selaimissa on epävarmaa ja näiden aiheuttamaa välkkymistä tai liikettä ei voida kontrolloida. Puhesyntetisaattoreille nämä elementit ovat merkityksettömiä [46, s. 29]. Välkkyvät ja liikkuvat tekstit ovat myös usein ärsyttäviä. Ruudunsuurentajia käyttäville selaajille ne ovat erityisen ärsyttäviä [75, s. 108]. Esteettömyyssuositukset kehottavat välttämään epilepsian takia välkkyviä elementtejä [75, s. 107][13, luku 6.7].

4.1.4 Vältettävät elementit

Tekstin esitystapaan tai asemointiin viittaavat elementit, kuten `font`, `fontbase`, `dir`, `menu`, `center`, `strike` ja `u`, on merkitty standardeissa välttäväksi elementeiksi (engl. deprecated) [67, luku 15] ja uusissa XHTML Strict ja XHTML 1.1 -suosituksissa vältettäväksi asetetut merkintätavat ovat kokonaan kiellettyjä [87, s. 145]. Saavuttavuusohjeissa kehotetaan myös erityisesti välttämään näitä vanhentuvia tekniikoita [13, luku 6.11].

4.1.5 Vaihtoehtoiset dokumenttiformaatit

HTML- tai XHTML-dokumenttien sijaan tekstiä voidaan jakaa verkossa myös muissa formaateissa, kuten PDF-, RTF- tai Word-dokumentteina. Muut formaatit eivät kuitenkaan toimi suoraan perusselaimilla tai avustajatekniikoilla, vaan vaativat erillisen ohjelman tai laajennoksen [13, luku 6.11]. Yhdysvaltojen opetus- ja oikeuslaitokset ovat ilmaisseet, että PDF-dokumentit voivat olla hankalasti luettavissa ruudunlukijoilla ja niiden rinnalla olisi syytä käyttää HTML-vastineita [75, s. 36, 348].

Toisaalta nykyään erilaisten tekstiformaattien saavutettavuusominaisuudet ovat parempia kuin ennen. PDF-dokumenteista voidaan tehdä saavutettavia, kun käytetään rakennetta merkitseviä tageja. Näiden avulla voidaan myös merkitä kuville vaihtoehtotekstejä samaan tapaan kuin HTML-kielessä. Saavutettavuusominaisuuksia varten tarvitaan kohtuullisen uusia ja usein kaupallisia työkaluja. [17]

Normaaleja PDF-dokumenttejakin voidaan jossain määrin lukea avustavilla tekniikoilla, kunhan ne eivät ole suoria skannauksia paperilta. Adoben PDF-lukijasta on olemassa myös ilmainen kevytversio useita älypuhelimia ja PDA-laitteita varten. Word- ja RTF-dokumentit ovat myös suhteellisen hyvin luettavissa Windows-pohjaisilla ruudunlukijoilla, jos näissä on käytetty tyylejä rakenteen merkitsemiseksi ja kuville on annettu vaihtoehtotekstit.

Voidaan kuitenkin olettaa, että XHTML-sivujen *saatavuus* on muita dokumentteja parempi. XHTML on tekstimuotoinen toisin kuin binäärimuotoiset Word-, RTF- ja PDF-formaatit. XHTML-dokumentteja voidaan lukea useilla erilaisilla selainohjelmilla, joita on valmiina tarjolla lähes kaikille erilaisille käyttöjärjestelmille ja laitteille. Hyvin toteutettu sivusto, joka käyttää oikealla tavalla XHTML- ja CSS-tekniikoita, on myös *saavutettava* erilaisissa käyttötilanteissa.

XHTML-dokumentteja voidaan käyttää yhdessä muiden XML-pohjaisten kielien kanssa. Esimerkiksi matemaattisia tekstejä voidaan kirjoittaa MathML-kielellä verkkosivujen yhteyteen, vaikka ainoastaan Mozilla tukee toistaiseksi tätä mahdollisuutta [16, s. 100]. MathML-merkintöjä voidaan muuntaa ääneksi ja matemaattiseksi kohopistekirjoitukseksi sopivilla adaptiivisilla tekniikoilla toisin kuin pelkkiä kaavakuvia.

4.1.6 Standardeilla varmistetaan yhteensopivuus

Yhteenveto

- ✓ XHTML mahdollistaa saavutettavat dokumentit, jotka tukevat samalla kerralla *kannettavia laitteita*, *avustajatekniikoita* ja tavallisia selaimia, joissa voidaan käyttää *mukautettuja asetuksia*.
- ✓ *Erilaiset selaimet* jäsentävät validin ja hyvin muodostetun sivun samalla tavalla.
- ✓ Selaajat, jotka *eivät voi käyttää visuaalista informaatiota*, hyötyvät, kun vältetään esitystapaan viittavia elementtejä.
- ✓ Standardien avulla yhteensopivuus tulevien W3C-tekniikoiden kanssa parantuu.

4.2 Rakenteellinen ja semanttinen merkintätapa

Sivun validiteetti on tärkeä osa sivuston saavutettavuutta, mutta se ei pelkästään takaa sivun saavutettavuutta. Tärkeä osa saavutettavuutta on standardien oikea käyttötapa. HTML- ja XHTML-dokumentteja tuotettaessa on käytettävä standardin määrittämää elementtivalikoimaa kokonaisuudessaan ja tarkoituksenmukaisesti. Elementtejä on käytettävä niiden merkityksen mukaisesti.

Rakenteellisella merkintätavalla tarkoitetaan sitä, että sisältö jäsennetään loogisiin osiin (X)HTML-elementtien avulla sen mukaan, mikä on osien merkitys. Graafinen tekstin osien jäsennys sen sijaan tarjoaa merkityksiä ainoastaan niille, joilla on kyky erotella osat visuaalisen hahmotuskyvyn perusteella. Rakenteellisilla merkintätavoilla ei sen sijaan kiinnitetä millään tavalla dokumentin ulkoasua, vaan ainoastaan osien merkityksiä. Järkevästi rakennettu dokumentti toimii ruudunlukijoilla, mobiililaitteilla sekä vanhoilla että uusilla graafisilla selaimilla. CSS-määrityksien avulla voidaan ehdottaa, miltä erilaisten rakenneosien pitäisi näyttää. Sivun on toimittava myös ilman ulkoasumäärityksiä. [87, s. 167-169][16, s. 126-128]

Alla olevassa esimerkki on `zeldman.com`-sivun alusta. Lähdekoodia on muokattu vähäisesti poistamalla linkeistä pitkiä `title`-kuvauksia ja `maincontent`-osasta leipätekstiä.

```
<h1><a href="/">Zeldman: Web design news</a></h1>
<p><a href="#maincontent">Skip navigation</a></p>
<ul id="menu">
  <li id="dailymenu"><a href="/">daily report</a></li>
  <li id="dwwsmenu"><a href="/dwws/">designing with web
standards</a></li>
  <li id="glammenu"><a href="/glamorous/">my glamorous life</a></li>
  <li id="classicsmenu"><a href="/classics/">classics</a></li>
  <li id="aboutmenu"><a href="/about/">about</a></li>
</ul>
<div id=”maincontent”>
  <img src="/rot/rotate.php" width="436" height="200" alt="" />
  <h3><a href="http://www.zeldman.com/2006/07/14/post-aea-nyc/">
      14 July 2006</a> 10 am  eastern</h3>
  <h2>And boy are my arms tired</h2>
  <p>…</p>
  <h3>We couldn&#8217;t have done it without you</h3>
```

Dokumentti on rakenteellisesti järkevä, sillä ulkoasumuotoiluja ei ole upotettu HTML-lähdekoodiin ja tekstin osat on merkitty otsikko-, kappale- ja listaelementeillä. Tällä tavalla jäsennettyä tekstiä on helppo tulkita erilaisissa ympäristöissä.

Kuva 6. Järkevästi jäsennettyä dokumenttia voidaan tulkita useilla tavoilla.

Kuvassa (Kuva 6) ylläoleva dokumentti on nähtävissä erilaisissa selaimissa. Ensimmäinen esimerkki on Mozilla Firefox-selaimesta CSS-tyylitiedoston ja taustakuvien avulla muotoiltuna, kun taas toisessa tyylisivut on otettu pois käytöstä. Sisältö on tällöinkin hyvin saavutettavissa. Kolmas näkymä on Opera-selaimen mobiiliympäristöä mukailevasta Small Screen Rendering -tilasta. Sivustolla on käytettävissä tyylitiedosto pienlaitteita varten, jolloin esimerkiksi navigointilinkkilistasta tulee pystysuuntainen. Viimeinen näkymä on IBM Home Page Readerin tekstinäkymästä. Ääniselain osaa hyvän rakenteen vuoksi pitää

listakohtien välillä taukoa, antaa äänimerkkejä otsikkojen kohdalla ja käyttää linkeissä ja normaalissa tekstissä erilaisia puheääniä.

4.2.1 Kappalerakenteet

Kaikki lohkotason rakenteelliset osaset on merkittävä sopivilla elementeillä. Näitä ovat esimerkiksi otsikot, listat, taulukot ja kappalelainaukset.

Otsikkotasot

Useilla sivuilla otsikkoteksti merkitään esimerkiksi `p`- tai `td`-elementillä ja kappaleen kirjasin muutetaan isoksi käytettäen upotettua CSS:ää, `big`-elementtiä tai `font`-elementtiä. Näkökykyinen kykenee tunnistamaan otsikon, mutta sokea ei voi tietää, mikä merkitys tekstin koolla on. WWW-ympäristössä ei ole myöskään millään tavalla kiinnitetty sitä, miltä otsikoiden pitäisi visuaalisesti näyttää. [13, luku 6.3][16, s. 126-128, 140][87, s. 168-169][75, s. 19]

Tekstissä olevat otsikot kannattaa merkitä elementeillä `h1`-`h6`. Tällöin esimerkiksi ruudunlukijaa käyttävällä sokealla selaajalla on mahdollisuus ymmärtää, että kyseessä on otsikko. Otsikot mahdollistavat navigoinnin sivun eri osiin ja tarvittaessa myös sivun pilkkomiseen pienempiin osiin [66, luku 5.4.3].

Web Content Accessibility Guidelines 1.0 ja Mobile Best Practices suosittelevat, että otsikkotasot merkitään loogisessa järjestyksessä siten, että pääotsikkoa `h1` seuraa alaotsikko `h2`, alaotsikon jälkeen voi olla toinen `h2`-elementillä merkitty alaotsikko tai alemman tason väliotsikko `h3`-elementillä merkittynä, kolmannen tason jälkeen voidaan käyttää `h2`-, `h3`-, `h4`-elementtiä ja niin edelleen [12, luku 1.2.1][66, luku 5.4.3]. Clark esittää, että otsikkotasojen järjestyksellä ei ole kuitenkaan käytännön merkitystä [16, s. 127]. Loogisesta otsikkorakenteesta voidaan kuitenkin olettaa olevan hyötyä esimerkiksi dokumentin yleiskuvan hahmottamisessa [54, s. 302]. Otsikkorakenteesta on erityisesti hyötyä niille käyttäjille, joilla on oppimisvaikeuksia [39, luku 3.5].

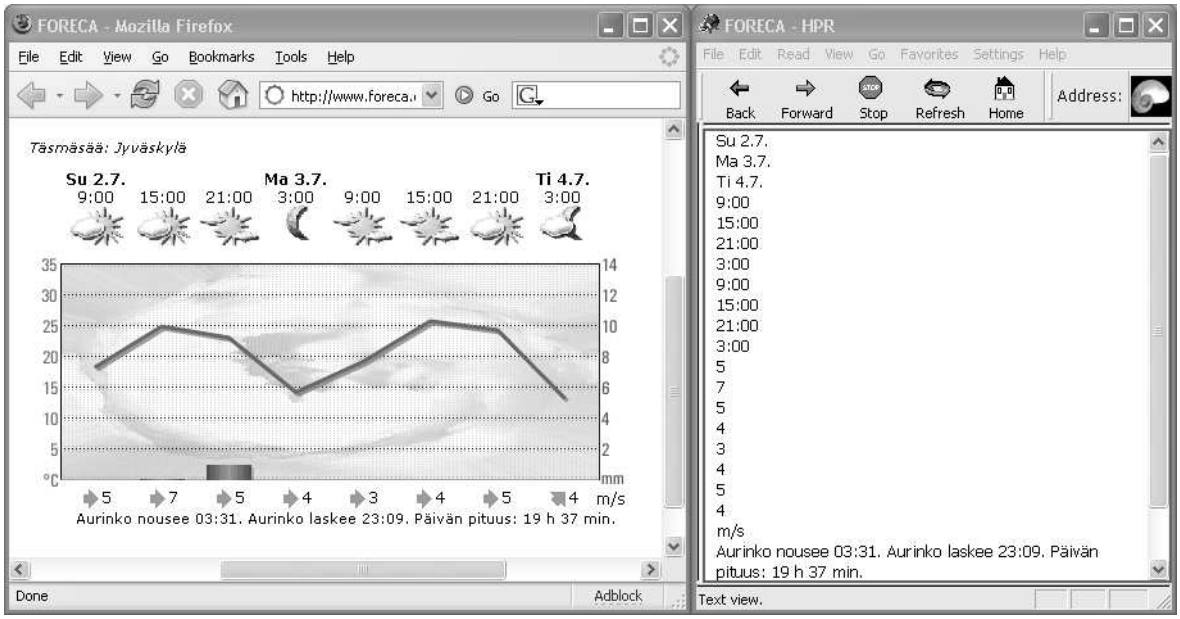
Kappaleet

Yhtenäinen tekstikappale on merkittävä ainoastaan yhdellä `p`-elementillä. Ruudunlukijat saattavat pysähtyä jokaisen rivin lopussa, jos kukin rivi merkitään erikseen `p`-elementillä. Tällainen epäsemanttinen kappalemerkintä aiheuttaa sivuston saavutettavuudelle merkittäviä ongelmia, sillä tekstin ymmärrettävyys huononee. [87, s. 48-49]

Kappaleiden kieli on merkittävä `xml:lang` tai `lang`-ominaisuudella, jolloin esimerkiksi puhesyntetisaattorit pystyvät lausumaan tekstit oikeaoppisesti. Jokaiseen kappaleeseen ei tarvitse erikseen kirjoittaa kielimäärityksiä, jos sivulla käytettävän kielen määrittää `html`-elementissä. Jos kieli kuitenkin vaihtuu jossain kappaleessa toiseksi, niin muutos on merkittävä kyseiseen kappaleeseen.

Taulukot ruudunlukijoilla

Taulukkomuotoinen tieto on jäsennettävä `table`-elementillä ja sen yhteydessä käytettävillä elementeillä ja ominaisuuksilla. Taulukossa `tr`-elementeillä (engl. *table row*) merkityille riveille on merkittävä tietoa sisältävät solut `td`-elementillä (engl. *table data*) ja taulukon otsikot `th`-elementillä (engl. *table header*). Taulukon otsikot antavat riveillä tai sarakkeilla oleville tietosoluille rakenteellista informaatiota. Otsikkosolut voidaan sijoittaa periaatteessa mihin kohtaan taulukkoa tahansa, mutta käytännössä ne ovat useimmiten ensimmäisillä riveillä tai sarakkeilla. Joissain tapauksissa ne voivat toimia esimerkiksi väliotsikkoina. [16, s. 231-232]



Kuva 7. Esteellinen säätilataulukko ja -kaavio graafisessa selaimessa ja ääniselaimen tekstinäkymässä.

Kuvassa (Kuva 7) on `foreca.com`-sivuston täsmäsää-palvelu Mozilla Firefox -selaimessa ja IBM Home Page Reader -ääniselaimen tekstinäkymässä. Sivun alussa esiintyvässä säätilataulukossa päivämäärät on merkitty `b`-elementillä tavallisiin `td`-datasoluihin. Otsikoiden välissä on tyhjiä soluja, joissa on ainoastaan rivinvaihtoa kuvaava `br`-elementti. Kellonajat on merkitty toisella rivillä tavallisiin datasoluihin tekstinä. Säätilaa esittävillä kuvilla ei ole `alt`-tekstejä, joten varsinaista säätilatietoa ei voida saavuttaa ääniselaimella. Lämpötilakaaviolla tai ilmansuunnilla ei ole myöskään vaihtoehtotekstejä. Ilmannopeuksiin liittyvät yksikkö- ja aikatiedot ovat helposti hahmotettavissa sivun visuaalisesta asettelusta, mutta ääniselaimen lineaarisessa lukujärjestyksessä otsikkotietoja kuullaan ennen ja jälkeen numerosarjan.

Taulukot aiheuttavat ongelmia käyttäjille, jotka eivät voi hahmottaa niitä kaksiulotteisesti. Taulukon ymmärtäminen vaatii informaatiota siitä, mitkä ovat kunkin tietosolun otsikot. Avustavat tekniikat eivät voi tietää, onko tietty otsikkosolu tarkoitettu rivi-, sarake- vai väliotsikoksi. Tätä varten otsikkosoluun voidaan antaa `scope`-ominaisuuden arvoksi `row` tai `col`, joka kertoo kumpaan suuntaan otsikko viittaa. Jos käytössä on useita rivejä tai sarakkeita koskevia otsikoita tai väliotsikoita, niin rivi- ja sarakeotsikoiden sijaan kukin tietosolu voidaan liittää `headers`-ominaisuudella otsikkosoluihin, joilla on `id`-arvo.

Tällöin ruudunlukija tai tekstiselain voi ilmoittaa kunkin tiedon jälkeen siihen liittyvät otsikot. Kuvan (Kuva 7) esittämällä sivulla ei ole tehty otsikoiden ja tietosolujen välistä kytkentää. Vaikka kuvilla olisikin vaihtoehtoteksti, niin käyttäjä ei välttämättä pysty yhdistämään säätilatietoa päivään ja kellonaikaan.

Taulukkoa voidaan tarvittaessa vielä jäsentää `tbody`-, `thead`- ja `tfoot`-elementeillä sisältöosaan ja otsikko-osiin. Otsikko-osat voivat olla sisältöosan ylä- tai alapuolella. Useita sarakkeita voidaan ryhmitellä `colgroup`- ja `col`-elementeillä. Nämä ryhmittelyt eivät ole kuitenkaan saavutettavuuden kannalta välttämättömiä.

Taulukkoon on sen sijaan syytä liittää ruudunlukijoita varten tärkeää metatietoa, joka antaa kokonaiskuvan taulukon sisällöstä. `caption`-elementin sisälle voidaan kirjoittaa koko taulukon otsikko, mikä näkyy myös graafisissa selaimissa. `table`-elementin `summary`-attribuuttiin voidaan kirjoittaa tiivistelmäteksti, joka kuvaa taulukon sisällön lyhyesti tekstimuodossa. Tämä ominaisuus on erityisesti tarkoitettu niille, jotka eivät voi hyödyntää visuaalista informaatiota, eivätkä halua lukea koko taulukon sisältöä. [16, s. 238-248]

Taulukot kannettavissa laitteissa

Matkapuhelimia varten on syytä tarkistaa, että taulukot ovat luettavissa pienilläkin näytöillä. Pienimmille näytöille mahtuu noin kolme saraketta, jos tietosolut eivät ole liian isoja. Koot ja ulkoasumuotoilut kannattaa tehdä tyylisivujen avulla. Taulukon solujen kokojen määrittelyssä on oltava tarkkana, erityisesti jos käytetään sisäkkäisiä taulukoita. Suhteellisia ja kiinteitä kokoyksiköitä ei kannata sekoittaa keskenään, sillä pienen ruudun takia taulukon eri osissa voi tulla helposti ylivuotoja. Monimutkaisia sisäkkäisiä taulukkoja on vältettävä, koska niiden latautuminen voi viedä aikaa. [59, luvut 2.2, 2.8, 6.2]

Taulukoiden sijaan on syytä harkita muita esitysvaihtoehtoja [66]. Toisaalta mukautumiskykyiset selaimet pystyvät automaattisesti linearisoimaan taulukon tietosoluille annettujen otsikkokytkentöjen perusteella samaan tapaan kuin ruudunlukijat [12, luku 5.3]. Taulukot voitaisiin mukauttaa myös mobiilijärjestelmiin sopivilla standardoiduilla komentosarjakielillä, kunhan niiden tuki yleistyy. Jos

matkapuhelinselaimet eivät suoraan tue taulukoiden linearisointia, niin tämä voidaan toteuttaa välityspalvelimessa sopivalla muunnoksella.

Listat

Listoja ei saa toteuttaa siten, että merkitsee listamerkit `img`-elementillä ja jakaa listakohdat omille riveilleen `br`-elementillä. Tämä merkintä ei kerro, että kyseessä on lista [87, s. 146, 170]. Lista on aloitettava `ul`-elementillä (engl. unordered list) ja kukin listakohta merkittävä `li`-elementillä, jolloin erilaiset selaimet kykenevät erottamaan listan. Tällöin ne voivat tulkita sen käyttäjälle sopivalla tavalla. Järjestetyn listan numeroita ei myöskään tarvitse merkitä itse, vaan käytetään `ul`-elementin sijaan `ol`-elementtiä (engl. ordered list). Määritelmälistat on merkittävä `dl`-, `dd`- ja `dt`-elementeillä.

Jos halutaan järjestämätön lista ilman listamerkkejä, niin edelleenkin on käytettävä samaa rakennetta, eikä pakotettuja rivinvaihtoja. CSS-määrityksien avulla voidaan tarvittaessa poistaa listamerkit. Jos järjestetyn listan normaali numerointitapa ei kelpaa, niin tapaa voi muuttaa CSS-määrityksien avulla. Olennaista on, että listat merkitään rakenteellisesti oikealla tavalla.

Kappalelainaukset

Kokonaiset lainaukset toisista teksteistä on merkittävä `blockquote`-elementillä. Jos on mahdollista, niin `cite`-ominaisuuden arvoksi täytyy laittaa verkko-osoite, mistä lainaus on otettu. Vain harvat selaimet tosin tukevat tätä ominaisuutta täysin, mutta ne merkitsevät lainatun tekstin kuitenkin eri tavalla. Joskus `blockquote`-elementtiä käytetään visuaalisiin tarkoituksiin sisentämään tekstiä, mutta tämä ei ole suositeltavaa. [16, s. 136-137]

Jos kappale on esimerkiksi runo tai ohjelmakoodi, jossa välilyönnillä ja rivinvaihdoilla on merkitystä, voidaan se merkitä `pre`-elementillä. Näillä väleillä ei ole kuitenkaan merkitystä ruudunlukijoita käyttäville. Elementtiä ei saa myöskään käyttää normaalien kappaleiden sisentämiseen tai rivittämiseen. [67]

4.2.2 Tekstin osia jäsentävät elementit

Tekstikappaleen tai lohkon sisällä olevia tekstin osia voidaan edelleen jäsentää erilaisilla elementeillä, jotka parantavat sivun saavutettavuutta.

Lyhenteet ja määritelmät

Lyhenteet ja määritelmät kannattaa merkitä `abbr`-, `acronym`- tai `dfn`-elementillä. `abbr` on tarkoitettu yhtenä sanana lausuttaviin lyhenteisiin, `acronym` kirjain kerrallaan esitettäviin lyhenteisiin ja `dfn` yleisiin määritelmiin. Lyhenteen tai määritelmän selite ilmoitetaan `title`-attribuutin arvossa. On riittävää, että lyhenteen selittää ainoastaan ensimmäisen esiintymiskerran yhteydessä. Jos määritelmä on eri kielellä kuin muu dokumentti, niin kieli voidaan määrätä `lang` ja `xml:lang`-ominaisuuden avulla. Nimien etukirjaimia tai mittayksiköitä ei ole kuitenkaan tarpeen merkitä näillä elementeillä.

Lyhenteet ja määritelmät ovat tärkeitä saavutettavuuden kannalta, sillä ne auttavat esimerkiksi ruudunlukijaa lausumaan oikealla tavalla kirjainryhmän. Näkökykyisille nämä selitteet näkyvät ponnahdustekstinä. [16, s. 129-133]

Lainaukset ja viittaukset

Muutamien sanojen lainaukset on syytä merkitä `q`-elementillä. Elementti toimii samaan tapaan kuin `blockquote`, mutta kappaleen sisällä. `cite`-elementillä voidaan merkitä viittaus erilaisiin nimikkeisiin, kuten esimerkiksi kirjoihin, elokuviin, ohjelmiin, oikeustapauksiin jne. ja luonnollisesti normaaleihin sitaatteihin. [16, s. 128, 137]

Esimerkit

Tietokoneen käytön ohjeistuksiin on muutama erityinen elementti. Kappaleen sisällä ohjelmakoodi merkitään `code`-elementillä, muuttujat `var`-elementillä, käyttäjän syöte näppäimistöltä `kbd`-elementillä ja `samp`-elementillä tietokoneen antamat vasteet. [67]

Korostukset

Esitystapaan liittyvien elementtien, kuten `b`, `i`, `tt`, `big` ja `small`, sijaan on syytä käyttää CSS-tyylimäärityksiä ja rakenteellisia HTML-merkintöjä, kuten `strong` ja `em` [16, s. 47,

123, 127-129][87, s. 171][75, s. 19]. `em`-elementillä tarkoitetaan korostusta ja `strong`-elementillä tarkoitetaan vielä hieman vahvempaa korostusta. [16, s. 128]

Rakenteellisilla elementeillä ei oteta kantaa siihen, millä tavalla merkitty teksti esitetään. Graafisissa selaimissa esimerkiksi korostus ilmaistaan tavallisesti muutoksena kirjasimessa, mutta ruudunlukijassa se voitaisiin ilmaista puheäänen muutoksella [12, luku 3]. Kirjasimen muutoskin voi olla hyvin erilainen, kun otetaan huomioon eri kielien merkintätavat.

4.2.3 Rakenteilla ilmaistaan tekstin osien merkitys

Yhteenveto

- ✓ Oikeilla rakenteilla jäsennetty teksti toimii sellaisenaan ilman ulkoasumuotoiluja *erilaisissa selaimissa*.
- ✓ *Ruudunlukijoilla* ja muutamilla *mobiiliselaimilla* voidaan navigoida sivua rakenteellisesti merkittyjen otsikkojen avulla.
- ✓ *Ruudunlukijoille* ja *tekstiselaimille* on merkittävää hyötyä datataulukossa käytettävistä otsikoista ja datasolujen kytkennöistä otsikoihin.
- ✓ Kielenmääritykset, listarakenteet ja lyhenteiden selitykset auttavat *ruudunlukijaa* syntetisoimaan puheen oikein.

4.3 Luettava ja ymmärrettävä teksti

Teksti on verkkosivujen tärkein elementti, sillä se muodostaa suurimman osan verkon varsinaisesta sisällöstä. Saavutettavuuden kannalta on erittäin tärkeä varmistaa, että kaikki pystyvät lukemaan tekstiä.

4.3.1 Yksinkertainen kieli

Oppimisvaikeudet vaikuttavat siihen, kuinka hyvin käyttäjä kykenee ymmärtämään sivulla esitettyä tekstiä. Sivun lukeminen voi viedä huomattavan kauan. Kuvista voi olla hyötyä asian hahmottamisessa tälle käyttäjäryhmälle. Tekstin on oltava mahdollisimman selkeää.

Myös kuurot, jotka käyttävät pääasiassa viittomakieltä, voivat hyötyä, jos teksti ei ole liian monimutkaista. [5]

Yksinkertaisessa ja ymmärrettävässä kielessä on oltava lyhyitä virkkeitä. Pitkiä sanoja on syytä välttää. Lukihäiriöiselle vieraskieliset sanat ovat ongelmallisia. Pilkut, pisteet ja kaksoispisteet ovat tärkeitä tekstin jaksottajia niin normaalissa tekstin lukemisessa kuin myös ääniselailussa. Osmo A. Wiio on kehittänyt suomen kieltä varten mittarin, joka arvioi karkeasti tekstin luettavuutta. Englannin kieltä varten on olemassa myös useita luettavuutta mittaavia indeksejä, esimerkksi *Gunning fox index*. [39, luku 5.3]

Ei ole kuitenkaan selvää, mitä tarkkaanottaen pitäisi tehdä, että tekstistä tulisi paremmin saavutettava niille, joilla on oppimisvaikeuksia. On melko selvää, ettei ole mitään käytännöllistä tapaa tehdä tekstimuotoisista verkkosivuista täysin esteettömiä niille, joilla on ongelmia lukemisen kanssa. Avustavista tekniikoista ja erilaisten kanavien käytöstä saattaa olla hyötyä, mutta varmaa tutkimustietoa asiasta ei ole. [16, s. 34-35]

Esimerkiksi lukihäiriössä pitkien sanojen ja rivien hahmottaminen on hankalaa, jolloin esimerkiksi ääneen lukemisesta voi olla hyötyä. Selkeä otsikkorakenne voi parantaa myös kokonaisuuden hahmottamista. [39, luku 3.5]

4.3.2 Kirjasimet

Kirjasinta merkittäessä on ilmoitettava mahdollisimman monta kirjasinvaihtoehtoa siten, että erilaisten käyttöjärjestelmien vaihtelevista kirjasinvalikoimista jokin tulee valituksi. Aina ei voi olla kuitenkaan varma siitä mitä kirjasimia erilaisissa laitteissa on, joten on merkittävä yleinen kirjasintyyppi viimeiseksi vaihtoehdoksi. CSS mahdollistaa tämän oikein käytetyllä `font-family`-määrityksellä. [87, s. 217-218]

Ruudulta lukemista varten suositellaan usein päätteetöntä kirjasinta. Päätteettömissä kirjasimissa ei käytetä kirjainten pääteviivoja. Käytettävyystutkimuksien mukaan sekä heikkonäköiset että myös täysin näkökykyiset pitävät päätteetöntä kirjasinta paremmin luettavana [76]. Tulostusnäkymässä suositellaan käytettävän päätteellisiä kirjasimia [39,

luku 5]. Kirjasintyypin valinnalla ei ole käytännössä kuitenkaan suurta merkitystä saavutettavuuden kannalta, kunhan jokin yleinen kirjasintyyppi on valittu [16, s. 225-226].

Kirjasimen koko ei varsinaisesti vaikuta sivun esteettömyyteen, jos teksti on ylipäätään saatavilla. Sokealle käyttäjälle tekstin koolla ei ole merkitystä [16, s. 223]. Hyvän saavutettavuuden kannalta on kuitenkin tärkeää, että sivua voi mukauttaa omalle näkökyvylle sopivaksi. Jos sivun käyttäjän näkökyky on heikentynyt tai hän on heikkonäköinen, niin normaalia isompi kirjasimen koko voi olla tarpeen. Monissa tämän hetken sivuissa on aivan liian pienikokoinen kirjasin esimerkiksi 50-vuotiaille käyttäjille [39, luku 5]. Tekstin koon kasvattaminen voi olla tarpeen esimerkiksi isoilla näytöillä, joissa oletuskirjasimen koko ei ole riittävän suuri [54, s. 29]. Heikkonäköiset käyttäjät voivat muuttaa kirjasimen kokoa jatkuvastikin ison ja pienen välillä pystyäkseen lukemaan tekstiä ja saadakseen sivusta kokonaiskuvan [76]. Lähes kaikissa uusissa selaimissa on mahdollisuus muuttaa ehdotettua kirjasimen kokoa.

Nielsen kehottaa ehdottomasti välttämään kiinteitä kirjasinkokoja, jotka merkitään esimerkiksi `pt-` tai `px`-yksiköillä, ja niiden tilalla olisi syytä käyttää suhteellisia kokoja, jotka merkitään prosenteilla tai `em`-yksiköillä [54, s. 302-303]. Swierenga suosittelee CSS-määritysten käyttämistä aina koon määrityksissä ja ehdottaa `em`, `px` tai `%`-yksiköiden käyttöä. Hän suosittelee suhteellisia yksiköitä, koska ne soveltuvat paremmin erilaisten oletusasetuksien kanssa, vaikka pikselit näkyisivätkin tasaisemmin eri selaimien välillä [75, s. 238-239]. Mobile Best Practices suosittelee ainoastaan suhteellisten yksiköiden käyttöä, jolloin sisältö voi mukautua pienen näytön kokoon [66, luku 5.4.8].

Korpelan mukaan kirjasimen koon asettamista pitäisi kokonaan välttää [39, luku 5]. Zeldmanin mukaan graafisten selaimien käyttäjät voivat toisaalta turhautua, jos kokoa ei määritetä ollenkaan ja selaimen oletuskoko on liian iso. Suhteellisesti määritetyt, normaalia pienemmät koot saattavat myös olla ongelmallisia, jos käyttäjän oletuskirjasimen koko on jo valmiiksi pieni. Pikselikoot saattavat olla varmempi tapa määrittää vaihdettavissa oleva

peruskoko[2] [87, s. 244-245]. Clarkin mukaan koon määrityksellä ei ole merkitystä merkittävästi heikkonäköisille, koska he käyttävät ruudunsuurentajia. Ruudunsuurentajien näkökulmasta yhteensopivin tapa on määrittää kirjasimen koko pikseleinä [16, s. 223]. Käyttäjät voivat myös ohittaa kirjasimen koon määritykset, mutta tällöin sivun taitto saattaa rikkoutua (kts. luku 4.6.2).

4.3.3 Riittävät värikontrastit

Tekstin ja taustan värin on erotuttava riittävästi jokaisessa sivun lohkossa niin, että teksti on erotettavissa. Lisäksi sellaisia värikombinaatioita on vältettävä, jotka estävät värisokeita käyttäjiä lukemasta tekstiä.

Aina kun elementille määrää tekstin värin, on syytä määrätä myös taustan väri, sillä ei ole varmuutta siitä, millaiset oletusväritykset kuhunkin selaimeen on käyttäjän toimesta asetettu [75, s. 245]. Samalla on syytä myös määrittää normaalin, aktiivisen ja vieraillun linkin väri, koska muutoin käyttäjän oletusvärit voivat sekoittua määritetyn normaalin tekstin värin kanssa [16, s. 217].

Riittävistä värikontrasteista on hyötyä myös matkapuhelinselaajille, koska näitä käytetään vaihtelevissa olosuhteissa ja näyttöjen värienerottelukyky on rajoitettu. Taustakuvan ja tekstin välinen kontrasti on oltava myös riittävä näiden rajoitteiden takia. [66, luvut 5.3.6, 5.3.7]

---

[2] Internet Explorer 6 -selaimessa tekstin koon muuttaminen ei onnistu, jos kirjasimen koko on määrätty pikseleissä. Selaimen muotoiluasetuksista voi kuitenkin halutessaan laittaa päälle asetuksen *Ohita Web-sivulla määritetyt fonttikoot*, jolloin voidaan valita haluttu fonttikoko. Toiminto ei kuitenkaan vastaa tekstin koon muuttamista, sillä se kadottaa tekstien kokoerot. [87, s. 319-320]

4.3.4 Tekstin näkyvyys on tärkeää

Yhteenveto

- ✓ Selkeä ja yksinkertainen teksti hyödyttää käyttäjää, jolla on *oppimisvaikeuksia*.
- ✓ *Värisokeat, heikkonäköiset* ja *kannettavien laitteiden* käyttäjät hyötyvät riittävästä värikontrastista.
- ✓ Jos näkökyky on hieman *heikentynyt*, niin on oltava mahdollisuus suurentaa tekstin kokoa.

4.4 Navigointi ja linkitys

WWW on navigointiin perustuva järjestelmä, jonka yksinkertaisin vuorovaikutus muodostuu linkkien valitsemisesta [54, s. 188]. Normaaliolosuhteissa käytettävät navigointimenetelmät eivät tee sivusta välttämättä saavutettavaa. Saavutettavan navigoinnin tekeminen vaatii yleensä lisätyötä ja tekee sivusta usein hieman monimutkaisemman [16, s. 141-142].

4.4.1 Navigointiin liittyviä käytettävyysnäkökohtia

Pelkät linkit eivät yleensä helpota liikkumista informaatioavaruudessa, vaan sivuston tekijän on lisättävä sivulle varsinaisesta sisällöstä erillisiä navigointiapuja. Näiden tehtävä on vastata, missä käyttäjä on ollut, missä hän on nyt ja mihin hän voi mennä [54, s. 188]. Navigoinnin yleisestä käytettävyydestä on hyötyä myös käyttäjille, joilla on kognitiivisia vaikeuksia [54, s. 309].

Sivun otsikot

Sivun otsikko, joka merkitään `title`-elementillä, on erittäin tärkeä osa sivua. Sitä käytetään usein tekstinä muualla, kun halutaan viitata tiettyyn sivuun. Tämän takia otsikkotekstin pitää olla myös sivukontekstista erottuva. Sivun otsikoita käytetään myös useissa navigointivalikoissa, kuten kirjanmerkeissä tai historialistauksessa. Käyttäjät hyödyntävät otsikoita navigointiapuna luettavan sivun valinnassa. [54, s. 123]

Sivuston rakenne

Riippumatta siitä millä tavalla navigointi rakennetaan, on sivustolla oltava järkevä rakenne. Jos sivuston informaatioavaruutta ei ole jäsennetty järkevästi, niin navigointi voi olla hankalaa. Useimmilla sivustoilla on hierarkkinen rakenne, jossa informaatio löytyy asteittain tarkentuvan luokittelun avulla. Taulukkomuotoinen luokittelu on myös eräs vaihtoehto, jolloin sivuja voidaan järjestää useiden parametrien perusteella. Lineaarinen rakenne ei ole useinkaan hyvä, sillä verkko ei ole luonteeltaan lineaarinen. [54, s. 198-199]

Navigointipalkit

Käytettävältä sivustolta voidaan löytää yhtenäiset ja samalla tavalla toimivat sivustohierarkian navigointimekanismit. Esimerkiksi tämän hetkiseltä sivulta on järkevää tehdä linkit ylemmille ja alemmille sivutasoille sekä muihin samantasoisiin sivuihin [54, s. 195]. Yksinkertaistetusta vakionavigointipalkista on hyötyä kaikille, mutta erityisesti se on välttämätön niille, joilla on hahmottamisvaikeuksia [54, s. 310].

Kuva 8. Navigointipalkki sivuston pääalueisiin.

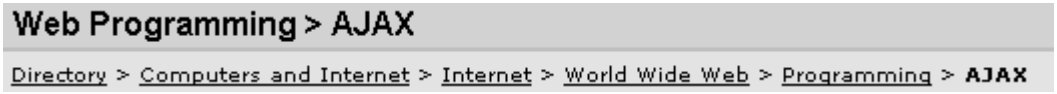
Useimmiten navigointipalkista löytyy sivuston pääkategoriat. Nämä esitetään listana linkkejä, jotka vievät sivuston ylätasoille. Kuvassa (Kuva 8) on välilehdiksi muotoiltu `bbc.co.uk`-uutispalvelun sivujen ylälaidasta löytyvä perusnavigointipalkki. Monesti sivuilla käytetään myös toissijaisia navigointialueita, joiden avulla annetaan käyttäjälle mahdollisuus valita saman tai alemman tason sivuja. Näitä ovat esimerkiksi valittuun aiheeseen liittyvät sivut, samankaltaiset artikkelit tai vastaavat tuotteet. Hakutoimintoa tarjotaan myös yleisesti navigointiapuna. Navigointialueet väritetään yleensä eri tavalla kuin sisältöosa. [54, s. 203-205]

Navigointilinkistö toteutetaan sivustoilla usein siten, että kukin linkki on oman taulukon `td`-solun sisällä tai omassa yleisessä `div`-lohkossa. Joskus linkit laitetaan samaan lohkoon ja erotetaan visuaalisesti esimerkiksi pystysuuntaisella viivalla tai rivinvaihdolla `br`-elementtiä käyttäen [87, s. 146-147]. Navigointilinkistö on kuitenkin semanttisesti *lista*,

jolloin se pitäisi merkitä esimerkiksi `ul` ja `li`-elementeillä. Listan oletusulkoasun voi vaihtaa mieleisekseen käyttäen CSS-määrityksiä.

Navigointipolut

Sivukartta on sivu, jossa sivuston osa-alueet ovat aiheittain ryhmiteltynä, järjestettynä diagrammiksi tai koko sivuhierarkia on esitetty puumaisena listana. Sivukartta voi auttaa käyttäjiä, joilla on kognitiivisia vaikeuksia hahmottaa sivuston rakennetta. Sivukarttaan voidaan myös merkitä, mitä polkuja pitkin käyttäjä on kulkenut ja millä sivulla tällä hetkellä ollaan. [54, s. 221, 310]



Kuva 9. Murupolkunavigointi Yahoo! Directory -palvelussa kertoo käyttäjän navigointihistorian.

Murupolkunavigointi (Kuva 9) auttaa hyvin hahmottamaan nykyisen sivun kontekstin ja sen avulla pääsee helposti palaamaan takaisin, jos sivu ei ollut oikea. Tällainen navigointi toimii ainoastaan sivustoilla, joissa informaatio on jaettu hierarkkisesti [54, s. 206]. Tällainen navigointi ei myöskään näytä muita samantasoisia vaihtoehtoja, mutta sen avulla on helppo vilkaista nykyisen sivun paikka sivustohierarkian syvyyssuunnassa [54, s. 213]. Murupolkunavigointi on suunniteltu silmäilyä varten. Jos käyttäjä on sokea, niin navigointipolun toistuva kuunteleminen ääniselaimella voi olla epämiellyttävää [16, s. 193].

Navigoinnin yhtenäisyys

Käyttäjät, joilla on kognitiivisia vaikeuksia, voivat hämääntyä, jos sivulta toiselle siirryttäessä tapahtuu merkittäviä muutoksia. Näiden käyttäjien on helpompi hahmottaa sivustoja, jotka ovat johdonmukaisesti muotoiltuja ja joissa on yhtenäiset navigointimahdollisuudet. Mahdollisuus tekstihakuun voi olla myös hyödyllistä. Ikääntyvät käyttäjät, joilla on muistihäiriötä, voivat olla myös riippuvaisia koko sivuston kattavasta yhtenäisestä navigaatiorakenteesta. [5]

Yhtenäinen sivun taitto on tärkeää niille, jotka ovat riippuvaisia ruudunsuurentajista. Sijoittamalla navigointilinkit samaan paikkaan joka sivulla helpotetaan huomattavasti niiden käyttäjien selailua, jotka näkevät vain osan ruudusta. Jos navigointien paikka on epäyhtenäinen, niin heikkonäköinen tai sokea selaaja joutuu käymään koko sivun läpi löytääkseen navigointipalkin uudesta paikasta. Sama pätee älypuhelimiin ja kämmentietokoneisiin, koska niissäkin sivusta on vain rajoitettu osa näkyvissä. [75, s. 13, 61]

Metatiedot navigointiapuna

Metatiedoilla voidaan luokitella ja kuvata dokumentin sisältöjä. Näillä voidaan helpottaa tiedon hakua ja navigointia [13, luku 13]. Toistaiseksi metatietoja voidaan hyödyntää selainsovelluksissa vain `link`-elementtien osalta. Näiden avulla voidaan kuvailla dokumentin suhde toisiin samalla sivustolla oleviin dokumentteihin, mikä mahdollistaa sivuston sisäisen navigoinnin. Viittaukset toisiin dokumentteihin merkitään `head`-osassa esimerkiksi seuraavasti [67, luku 12.3]

```
<link rel="Index" href="../index.html" />
<link rel="Next"  href="Chapter3.html" />
<link rel="Prev"  href="Chapter1.html" />
```

Lisäksi on olemassa monia muitakin dokumenttien suhteiden tyyppejä, kuten esimerkiksi Start, Contents, Parent, First, Last, Search, Alternate ym. [67, luku 6.12]. Tekstiselaimet Lynx ja Links sekä graafiset selaimet Mozilla, Opera ja iCab kykenevät käyttämään näitä tietoja ja tarjoamaan niihin pohjautuvia vaihtoehtoisia navigointimenetelmiä. Valitettavasti Internet Exploreriin pohjautuvat ruudunlukijat eivät pysty hyödyntämään näitä mahdollisuuksia. Kuvassa (Kuva 10) on Opera 9 -selaimen *navigointipalkki*, jonka avulla voidaan selata helposti yhteenkuuluvaa dokumenttijoukkoa.

Kuva 10. Suomenkielisen Opera 9 -selaimen työkalurivi sivustonavigointia varten.

Tulevaisuudessa metatietojen avulla voitaisiin esimerkiksi järjestää tai poistaa sivuilla olevia osia. Nykyään tämä on vaikeaa, koska sivun osille ei ole standardoitua merkintätapaa. Sivuja pystyy kuitenkin jo nyt luokittelemaan HTML:n ja Dublin Coren

metatietojen avulla, joten yhteinen standardoitu tapa osien merkitsemiseksi ei ole kaukana. W3C on julkaissut standardeja sivujen mukauttamista ja laitteistoriippumatonta toteutusta varten, mutta käytännön sovelluksia vielä odotellaan. [16, s. 352-354]

4.4.2 Navigointi ruudunlukijoilla

Perusongelma ruudunlukijoiden, kohopistekirjoittimien tai kytkinohjauksen kanssa on sivun rakenteen lineaarinen käsittely. Erityisen ongelman muodostavat ennen sisältöosaa tulevat navigointilinkit, joita ei voida avustavilla tekniikoilla ohittaa, ellei sivun tekijä ole huomioinut vaihtoehtoisia navigointimenetelmiä. [16, s. 147-149]

Otsikkonavigointi

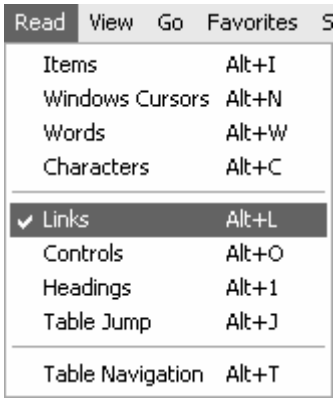
Otsikot (`h1-h6`) toimivat ääniselaimille ja ruudunlukijoille navigointiapuna. IBM Home Page Reader mahdollistaa lukutilan, jossa luetaan vain otsikot [75, s. 132-133]. JAWS- [24] ja Window-Eyes-ruudunlukijat [29] mahdollistavat siirtymisen otsikosta toiseen. Tällöin ruudunlukijan ei aina tarvitse aloittaa sivun alusta vaan selaaja voi hakea tietoa suoraan dokumentin loppupuoleltakin. Otsikkotasot mahdollistavat sokealle käyttäjälle visuaalista silmäilyä vastaavan tavan sivun yleiskuvan ja mielenkiintoisten osien hahmottamiseksi [54, s. 302]. Otsikkonavigoinnilla sokea selaaja voi suoraan hypätä myös pääotsikkoon eli sisältötekstin alkuun. Otsikkotasoista on mahdollista saada ruudunlukijoilla automaattinen sisällysluettelo, jonka avulla on helppo hahmottaa sivun rakennetta.

Otsikkonavigointia käytetään paljon ja se koetaan hyödylliseksi, koska se mahdollistaa sivuston oleellisten osien nopean läpikäynnin. Sokeat selaajat toimivat tässä suhteessa aivan samaan tapaan kuin näkökykyiset ihmiset silmäillessään otsikoita ja listoja. [77]

Otsikoiden avulla voidaan navigoida myös normaaleissa graafisissa selaimissa. Opera-selaimessa on suoraan mahdollisuus navigointiin otsikkotasojen avulla ja Mozilla Firefox -selaimeen otsikkonavigointi on saatavilla laajennoksien avulla.

Linkkitekstit

Käyttäjät navigoivat myös linkkilistojen avulla. Uudet ruudunlukijatyökalut kykenevät keräämään sivuston linkit yhteen listaan, josta käyttäjä voi valita haluamansa ja siirtyä uudelle sivulle. Tämä mahdollistaa sivuston nopean linkkien läpikäynnin samaan tapaan kuin näkökykyiset silmäilevät sivustoa. [77]



Kuva 11. Pelkkien linkkien lukeminen ääniselaimella.

Kuvassa (Kuva 11) on IBM Home Page Reader -ääniselaimeen valittu lukutila, jossa sivulta luetaan ainoastaan linkit. Lukutilaa voi muuttaa tarpeen tullen. Ruudunlukijoiden keräämissä linkkilistoissa valittavana tekstinä käytetään linkkitekstiä. Linkkitekstin on oltava tekstikontekstista erottuva, jotta sitä voitaisiin käyttää järkevästi navigointilistoissa. Huonoja linkkitekstejä ovat esimerkiksi ”napauta tästä” tai ”lue lisää”, koska ne eivät sellaisenaan kerro, mihin linkki vie.

Linkkitekstiä voidaan myös täsmentää tarkemmalla kuvauksella, joka ilmoitetaan `a`-elementin `title`-ominaisuudessa. Ruudunlukijat eivät välttämättä normaalitilassa lue näitä lisäkuvauksia, mutta esimerkiksi JAWS-ruudunlukijassa saa sopivalla näppäinyhdistelmällä luettua `title`-tekstit linkkitekstien sijaan [24]. Nämä lisäkuvaukset parantavat myös yleisesti käytettävyyttä, sillä graafisisissa selaimissa `title`-kuvaus näkyy ponnahdustekstinä ja helpottaa käyttäjää ymmärtämään, mitä linkin takaa löytyy [54, s. 60-62].

Navigointipalkin ohittaminen

Vasemmassa laidassa oleva navigointialue on lähes normi kaupallisilla sivuilla. Ruudunlukijoita ja kohopistekirjoittimia käyttävät joutuvat jokaisen sivun alussa kuuntelemaan tai lukemaan tämän palkin linkkitekstit. Kytkinohjausta käyttävät joutuvat useaan kertaan valitsemaan sarkainnäppäimen painalluksen, jos haluavat valita jonkin kohdan sisältöosasta tai muusta sen jälkeisestä osasta. [16, 148-151]

Kuva 12. Pääuutisen saavuttaminen `HS.fi`-palvelussa on helpompaa visuaalisesti kuin auditiivisesti.

Kuvassa (Kuva 12) `HS.fi`-palvelun etusivun osa esitettynä Mozilla Firefox ja IBM Home Page Reader -selaimilla. Sivun vasemmassa laidassa on pitkä pystysuuntainen vakionavigointipalkki, joka ei kuitenkaan graafisessa näkymässä vie tilaa tai näkyvyyttä vieressä olevalta sisällöltä. Pääuutisen saavuttaminen ääniselaimella on sen sijaan työlästä, koska käyttäjä joutuu kuuntelemaan ensin kaikki vasemman palkin sisällöt. `HS.fi`-palvelun uutisista löytyy tosin erillinen tekstiversio, mutta tämä ei ole kuitenkaan tasavertainen tapa saavutettavuuden näkökulmasta.

Useista linkeistä koostuvasta navigoinnista voidaan tehdä saavutettava

- sijoittamalla sisältö lähdekoodissa ennen navigointia tai
- lisäämällä sivun sisäinen linkki, jolla navigointiosa voidaan ohittaa.

Jos navigointi halutaan saada visuaalisesti vasempaan laitaan, niin tämä voidaan toteuttaa CSS:n avulla yksinkertaisesti absoluuttisesti asemoimalla navigointilohko vasempaan laitaan. Taulukkotaittoa käyttävillä sivuilla voidaan hyödyntää ns. *table hack* -menetelmää. Tässä ensimmäisen rivin ensimmäinen solu on tyhjä ja toisessa on sivun sisältö, joka venytetään kahden rivin korkeudelle. Toisen rivin ensimmäiseen soluun laitetaan

navigointilinkit ja siirretään tämä `valign`-ominaisuudella ensimmäisen rivin tasolle. Tällöin ruudunlukijat ja kohopistekirjoittimet käsittelevät sisältöosan ensin. [16, s. 150-152]

Toinen yleisemmin käytetty tapa on lisätä sivun alkuun hyppylinkki, joka ohittaa sisällön edellä olevat materiaalit. Yksinkertaisimmillaan linkin voi tehdä seuraavasti [esim. 75, s. 116]

```
<a href=”#content” title=”Siirry pääsisältöön”>Siirry
pääsisältöön</a>

<!-- navigointiosat -->

<a id=”content” name=”content” />

<!-- sisällöt -->
```

Sisäisen linkin kohde voi olla suoraan myös esimerkiksi `div`-lohko, jonka `id`-attribuutin arvo on `content`, mutta vanhanmuotoinen paikallinen ankkuri on toistaiseksi yhteensopivin ratkaisu.

Hyppylinkit ovat graafisten selainten käyttäjille melko tarpeettomia, joten useimmiten ne halutaan piilottaa. Piilottamista ei kannata tehdä CSS:n `display`- ja `visibility`-ominaisuuksilla, koska tällöin hyppylinkit saattavat kokonaan kadota ruudunlukijoiden käytettävistä. Linkin piilottaminen onnistuu kuitenkin

- pienentämällä hyppylinkin tekstin kokoa minimaaliseksi [16, s. 155],
- käyttämällä 1×1 pikselin kokoista läpinäkyvää kuvaa, jossa linkkiteksti on sijoitettu kuvaelementin `alt`-attribuuttiin [16, s. 153-154],
- siirtämällä CSS:n avulla linkkiteksti pois kuvaruudusta tai muuttamalla linkkitekstin sisältävän lohkon korkeus ja leveys nollan pikselin kokoiseksi tai
- muuttamalla hyppylinkin tekstin väri taustan väriseksi. Hyppylinkki voidaan palauttaa takaisin näkyväksi, kun se tulee valituksi näppäimistöltä. [75, s. 120-121]

Jos sivulla on useampia kuin yksi navigointialue, niin on suositeltavaa tehdä hyppylinkki jokaisen osion alkuun ja muodostaa näistä ketju, joka vie sisältöosaan asti. Oikealla puolella olevaan navigointiosaan on myös hyvä päästä ennen sisältöosaa. Tällöin

ruudunlukijaa, kohopistekirjoitusta tai kytkinohjausta käyttävä voi nopeasti valita haluamansa osion ja jatkaa selailua siitä eteenpäin. [16, s. 157-163]

Ruudunlukijoissa ja ääniselaimissa on myös sisäänrakennettuna toimintoja, joiden avulla voidaan ohittaa lohkoja, joissa on paljon linkkejä. JAWS, Window-Eyes ja IBM Home Page Reader -ohjelmissa hyppytoiminto siirtää seuraavaan tekstilohkoon, jossa on riittävä määrä sanoja tai rivejä. Hyppy saattaa kuitenkin pysähtyä esimerkiksi tekstiin, joka on navigointipalkin keskellä. [75, s. 122-123]

Sarkainjärjestys

Sivun osien läpikäyntijärjestystä näppäimistöltä voidaan muokata käyttämällä `tabindex`-ominaisuutta. Ominaisuus voidaan liittää hyperlinkkeihin, kuvakarttoihin ja lomake-elementteihin. Tärkeät linkit ja sivun hakulaatikot voidaan merkitä halutussa järjestyksessä kasvavalla numerosarjalla. Näin määrätään järjestys, jossa kyseiset elementit tulevat valituiksi, kun käyttäjä napauttaa näppäimistöltä sarkainnäppäintä. [16, s. 175-176]

Normaalisti linkkien ja muiden valittavien elementtien läpikäyntijärjestys on lineaarinen lähdekoodin määräämässä järjestyksessä [75, s. 57]. Käyttäjä voi hämääntyä, jos tätä järjestystä muutetaan merkittävästi. Sivun sisäisessä navigoinnissa `tabindex`-ominaisuutta pitää käyttää harkiten [16, s. 180]. Järjestettyjen valintavaihtoehtojen jälkeen sivun kohteiden läpikäynti alkaa taas normaalissa järjestyksessä. Ominaisuutta ei voida siten käyttää korvaavana vaihtoehtona hyppylinkeille sivun osien välisessä navigoinnissa, ellei kaikille valittaville elementeille anneta järjestystä.

Pikanäppäimet

Näppäimistöltä navigointia voidaan vielä helpottaa `accesskey`-ominaisuudella. Ominaisuus voidaan kytkeä linkkeihin ja lomake-elementteihin. Tämän avulla kyetään muodostamaan sivukohtaisia pikanäppäimiä, jotka valitsevat ja toteuttavat määrätyn toiminnon. Useimmissa selaimissa pikanäppäimen kanssa on käytettävä jotain toista näppäintä, esimerkiksi `ALT`-painiketta. Pikanäppäin voidaan kytkeä esimerkiksi sivun sisäiseen hyppylinkkiin, jolloin näppäinyhdistelmä siirtää tiettyyn kohtaan sivua.

Pikanäppäin voidaan kytkeä myös normaaleihin navigointilinkkeihin, jolloin selain siirtyy suoraan linkissä määrätylle sivulle. [16, s. 164-165]

Pikanäppäimiä on syytä käyttää vain usein käytetyissä linkeissä ja aina samalla tavalla sivuston eri osissa. Lisäksi on varottava näppäinyhdistelmiä, jotka ovat jo varattuja selaimissa [16, s. 165-166]. Määrätty pikanäppäinyhdistelmä menee tällöin valikkokomentojen ohi ja käyttäjän täytyy huomata napauttaa erikseen ALT-näppäintä (tai muuta vastaavaa) päästäkseen valikkoon [75, s. 157]. Selaimissa on valitettavasti myös todella harvoin mahdollisuus näyttää lähdekoodiin merkittyjä pikanäppäimiä. Ainoastaan iCab Macintosh näyttää suoraan pikanäppäimet linkin vieressä [87, s. 203], JAWS-ruudunlukija ilmoittaa lukiessaan myös pikanäppäimet [24] ja Mozilla Firefox -selaimeen on saatavilla laajennoksena toiminto, joka mahdollistaa pikanäppäimien listauksen. Tyypillisesti sivun tekijän on kuitenkin itse merkittävä pikanäppäimet osaksi tekstiä.

4.4.3 Navigointi mobiililaitteilla

Matkapuhelimilla selattaessa käyttäjät ovat hyvin päämäärätavoitteisia, hakevat tiettyä informaatiota ja haluavat sen nopeasti ilman selailua [60, s. 16].

Sivun rakenne ja navigointi

Navigointi ei ole itseisarvo ja sen koko on minimoitava [54, s. 18]. Mobiiliselaimia varten on syytä tehdä ainoastaan yksinkertainen navigointi, jonka on oltava sivun ylälaidassa. Sivun lopussa voi olla toissijaisia navigointilinkkejä. On tärkeää, että käyttäjä näkee heti sivun latauduttua osan sivun sisällöstä ilman vieritystä [66, luku 5.2.2]. Sivun alussa on oltava tärkeimmät navigointilinkit ja hakutoiminnot. Käyttäjälle on annettava mahdollisuus palata takaisin palvelun etusivulle [60, s. 17, 19]. Etusivulle vievä linkki suositellaan lisättäväksi sivun loppuun [37].

Navigointimallin on oltava yhtenäinen kaikilla sivuilla opittavuuden takia [60, s. 16, 18]. Navigoinnin on tähdättävä siihen, että käyttäjä saa mahdollisimman vähällä odottelulla ja linkkien valinnoilla haluamansa informaation. Nyrkkisääntönä voidaan pitää alle viiden sivun lataamista, muutoin käyttäjä voi turhautua. Vähemmän käytetyt sivut voivat olla useampienkin linkkien päässä [66, luku 5.2.3].

Linkkitekstit ovat tärkeitä kannettavissa laitteissa, sillä yhteydet ovat hitaita ja navigointi väärälle sivulle voi maksaa ajassa ja rahassa. Linkkiteksteinä ei saa käyttää ”klikkaa tästä” -tyyppisiä tekstejä, sillä ne eivät kuvaile kohdetta. Jos kohde ei ole HTML-muotoinen ja tiedoston koko on suuri, on tämä syytä merkitä jollain tavalla linkin viereen. [66, luku 5.2.6]

Sisäiset linkit

Pidemmissä tekstisivuissa on järkevä tehdä sisäinen linkitys pääotsikoihin. Tämä mahdollistaa nopean siirtymisen sisällön osasta toiseen. Jokainen luvun jälkeen on syytä lisätä linkki, joka vie dokumentin alkuun. [66, luku 5.2.4]

Pikavalinnat

Jos mobiilikäytössä ei ole sopivaa osoitinlaitetta, niin pikavalinnat voivat olla hyödyksi linkkien valitsemisessa. Tällä voidaan välttää navigointinäppäimien toistuvia painalluksia. Pikavalintojen tulisi toimia koko sivustolla samalla tavalla ja niitä on syytä käyttää vain usein käytetyissä toiminnoissa. [66, luku 5.2.5]

Pikavalinnat ovat hyödyllisiä, sillä näiden avulla voidaan välttää tarpeetonta linkkilistojen läpikäyntiä. Pikavalinnat toimivat myös sisäisissä linkeissä. Näitä käytettäessä on järkevää merkitä näppäimet kuvioilla, jotka muistuttavat matkapuhelimesta löytyviä näppäimiä. Merkityt valintavaihtoehdot saattavat näkyä myös muuallakin kuin selainikkunassa. Esimerkiksi Nokian omissa XHTML-selaimissa sivun pikavalintalinkit löytyvät myös `Options`-valikosta. [60, s. 18, 34-36]

Toisaalta käyttäjät eivät välttämättä huomaa, että sivustolla on mahdollisuus pikanäppäinten käyttöön. Tämä on tavallista myös muissakin selainympäristöissä. Näppäinoikoteiden käyttöä ei välttämättä lisää edes pikavalintanumeron lisämerkintä kuvalla. [37]

Sivuston rakenne

Liian moneen tasoon jaettu hakemistohierarkia ei ole hyvä mobiiliselailun näkökulmasta. Neljässä tai viidessä kerroksessa olevan sivuston rakennetta on vaikea hahmottaa. Aloitteleville käyttäjille on tärkeää, että sivusto on hyvin organisoitu. [60, s. 17-19]

Sivun otsikot

Mobiiliympäristössä näyttäisi olevan erittäin tärkeää, että käyttäjä tietää, missä sijaitsee suhteessa sivustohierarkiaan. Usein paikka ilmoitetaan sivun otsikossa tai väliotsikoissa (kts. luvut 4.2.1 ja 4.4.1). Tutkimuksissa on huomattu, että lähes kaikki käyttäjät huomaavat matkapuhelimen selaimen ylälaidassa olevan sivun otsikon ja hyödyntävät sitä pyrkiessään ymmärtämään sivuston rakennetta. Useat käyttäjät hyödyntävät myös navigointihistoriaa, mikäli sellainen on tarjolla. Tällöin sivun otsikkotekstit ovat myös olennaisessa asemassa. [37]

Otsikkotekstin on oltava lyhyt, maksimissaan 14 merkkiä. Jos teksti ei mahdu ruutuun, niin se katkaistaan. Ei ole kuitenkaan suositeltavaa käyttää lyhennelmiä otsikoissa, sillä käyttäjät eivät välttämättä ymmärrä niitä. [60, s. 23, 26]

4.4.4 Navigointi ruudunsuurentajilla

Ruudunsuurennoksessa sivusta näkyy vain pieni osa, jolloin tilanne on samankaltainen kuin mobiiliselaimissa.

Sisäiset linkit

Ruudunsuurentajaa käyttävät eivät välttämättä näe tai ymmärrä käyttää oikeassa laidassa olevaa vierityspalkkia. Useimmat eivät huomaa myöskään sivun oikeassa osassa olevia linkkejä. Tällöin olisi hyödyllistä, jos ylhäällä vasemmassa laidassa olisi sisäisiä linkkejä sivuston eri sisältöihin. [76]

Navigointipalkit

Ruudunsuurentajia käyttävät saattavat luulla vasemmassa laidassa olevaa navigointipalkkia osaksi sisältötekstiä, kun käytössä on sekä mukautetut väriasetukset että ruudun suurennos

[76]. Navigointipalkin käyttäminen voisi olla helpompaa heikkonäköisille käyttäjille, jos sen tärkeimmät osat siirrettäisiin sivun ylälaitaan muutamaksi riviksi ja vähemmän tärkeät linkit sivun alalaitaan. Tällöin käyttäjä näkee linkit varmasti, eikä sekoita niitä varsinaiseen sisältöön [15].

4.4.5 Erilaiset ohjausmenetelmät on huomioitava navigoinnissa

Yhteenveto

- ✓ Samalla tavalla toimiva ja samasta paikasta löytyvä navigointi hyödyttää käyttäjiä, joilla on *kognitiivisia vaikeuksia* tai *puutteita näkökyvyssä*. Ennakoitava navigointi lisää myös palvelujen käytettävyyttä *mobiililaitteilla*.
- ✓ Mahdollisuus ohittaa navigointiosa helpottaa käyttäjiä, jotka käyttävät *kytkinohjausta* tai *ruudunlukijoita*.
- ✓ Sisäiset linkit sisältöön sivun alussa parantavat saavutettavuutta pienillä näyttötiloilla, siis esimerkiksi *mobiililaitteilla* ja *ruudunsuurentajilla*.
- ✓ Sivun otsikon ja linkkien tekstien on oltava kuvaavia, koska *ruudunlukijat* ja *mobiiliselaimet* käyttävät näitä navigointitoiminnoissaan.
- ✓ Pikavalinnat ja sarkainjärjestys mahdollistavat joustavan navigoinnin ilman osoitinlaitetta, mistä on hyötyä *kytkinohjausta* käyttäville sekä useille *mobiililaitteille* ja *ruudunlukijoille*.

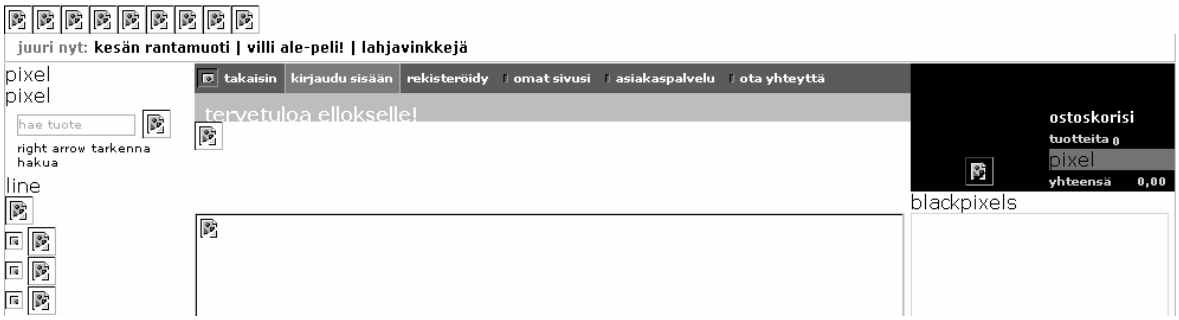
4.5 Kuvat

Kuvat ovat merkittävä osa nykyisiä verkkosivuja. Normaaliin sisältöön liittyvien kuvien lisäksi niitä käytetään kaavioihin, logoihin, täytekuviin, ikoneihin, navigointipainikkeisiin, lomakkeisiin ja jopa tekstin korvikkeena. Jos käyttäjä ei voi saavuttaa näiden tarjoamia merkityksiä, niin hänellä on vastassa suuria ongelmia sivun käytössä [16, s. 61]. Tekstivastineiden tekeminen kuville ja objekteille on ehkä yksi saavutettavuuden tärkeimpiä piirteitä [75, s. 69].

4.5.1 Vaihtoehtotekstit kuvainformaation vastineena

Jokaista kuvaa kohden on merkittävä vaihtoehtoinen tekstiesitys kuvaelementin `alt`-ominaisuuteen. `alt` on pakollinen attribuutti HTML- ja XHTML-dokumenttityypeissä. `alt`-tekstin on oltava lyhyt ja kuvaileva. Pituudeksi suositellaan enintään 8-10 sanaa. Tekstissä on kerrottava tiiviisti, mitä kuva esittää ja mitä sillä yritetään viestittää. Jos kuva on sijoitettu `a`-elementin sisälle, niin `alt`-tekstin kuvaus voi myös kertoa mitä tapahtuu, jos linkkiä napautetaan [54, s. 305-306]. Ei ole kuitenkaan tarpeen kertoa, että kyseessä on linkki johonkin paikkaan, sillä tämä on havaittavissa rakenteesta [16, s. 94]. On vältettävä tekstejä, jotka eivät anna mielekästä informaatiota sokealle käyttäjälle [16, s. 63-64]. Tällaisia tekstejä voisivat olla esimerkiksi ”kuva”, ”logo”, ”klikkaa tästä” tai ”sponsorin sivulle”.

Ääniselain tai kohopistekirjoitin pyrkii tarjoamaan käyttäjälleen sen informaation, jonka se saa verkkosivulta. Jos vaihtoehtotekstiä ei ole käytössä, niin nämä selaimet yrittävät saada tiedon esimerkiksi tiedostonimestä. Tämä voi johtaa epämiellyttävään kirjainsekamelskan luettelemiseen [75, s. 73].



Kuva 13. Tarpeettomia vaihtoehtotekstejä sivun taittoon käytettävissä kuvissa.

Kuvassa (Kuva 13) on verkkoselain, josta on kytketty kuvien lataus pois päältä. Tällöin kuvien tilalla nähdään vaihtoehtotekstit. `Ellos.fi`-verkkokaupan etusivun merkityksellisiltä kuvilta puuttuvat vaihtoehtotekstit, mutta niitä löytyy kuitenkin tarpeettomasti sivun taittoon liittyvistä kuvista. Jos kuvalla ei ole sisällöllistä merkitystä, on tällöin `alt`-attribuutti merkittävä tyhjänä. Näin esimerkiksi ruudunlukija ei lue kuvan tiedostonimeä vaan ohittaa kuvan kokonaisuudessaan. Tällaisia kuvia ovat esimerkiksi ns.

spacer-kuvat, jotka liittyvät sivun taitossa välien tekemiseen (kts. luku 4.6), ja erilaiset koristekuvat, joiden tarkoitus on ainoastaan miellyttää katsojaa [39, luku 5].

Tekstiä ei kannata laajassa mittakaavassa toteuttaa pelkästään kuvina, vaikka `alt`-tekstillä onkin mahdollista korvata näkymättömissä olevat sisällöt. Pienissä määrin tekstikuvat eivät ole haitallisia ja sopivat esimerkiksi painikkeisiin ja ikoneihin [16, s. 102]. Ongelma on siinä, että tekstiä korvaavien kuvien takia mahdolliset tekstiä jäsentävät rakenteelliset merkinnät vähenevät, latausajat pitenevät, tekstin kokoa ei voida muuttaa ja se on hankalammin käännettävissä muille kielille [54, s. 29].

`alt`-teksti on tarkoitettu pelkästään sellaiseen tilanteeseen, jossa ei ole mahdollisuutta näyttää varsinaista kuvaa. Jotkin vanhat selaimet näyttävät väärällä tavalla `alt`-tekstin ponnahdustekstinä, kun hiiren kursori viedään kuvan päälle. Tällaista toiminnallisuutta varten on olemassa kuvan `title`-ominaisuus. Ominaisuuden arvoksi voidaan kirjoittaa hieman `alt`-tekstiä pidempi kuvan sisällön kuvailu. `title`-tekstin tarkoitus on vastata selittävää kuvaotsikkoa. [16, s. 66-67]

Kaikkein pisin kuvan kuvaus voidaan tehdä `longdesc`-attribuutilla. Tätä varten kirjoitetaan kokonaan uusi HTML-sivu ja siihen kirjoitetaan kompaktisti normaaleja tekstin rakenteita käyttäen kuvan kuvaus. Dokumentin lopussa voi olla käytettävyyden parantamiseksi linkki, joka vie takaisin alkuperäisen dokumentin ankkurikohtaan, joka on juuri kuvan jälkeen [16, s. 68-70]. `longdesc` sopii esimerkiksi kaavioista löytyvän informaation kuvailua varten [75, s. 71, 86].

4.5.2 Kuvat kannettavissa laitteissa

Kuvia suositellaan käytettävän harkiten myös kannettaville laitteille suunnitelluissa sivuissa, sillä ne ovat hyvä havainnollistusväline ja piristävät monotonista tekstiä. Kuvien koko voi vaikuttaa sivuston käytettävyyteen ja käyttäjäkokemukseen matkapuhelimella tai älylaitteella, sillä latausajat saattavat olla hyvin pitkiä. Isojen kuvien käyttöä ei suositella, ellei niillä ole erityistä tarkoitusta. Isot kuvat haittaavat joustavaa selailua [60, s. 18-20]. Jotkin käyttäjät kytkevät kuvat pois päältä säästääkseen tietoliikennekustannuksissa ja

saadakseen sivut latautumaan nopeammin [59, s. 8]. Tällöin myös kuvien vaihtoehtoteksteistä on hyötyä (kts. 4.5.1).

Bittikarttakuvien koko kannattaa merkitä erikseen HTML- tai CSS-määrityksiin `width`- ja `height`-ominaisuuksilla. Tämä kertoo kannettavalle laitteelle, minkä kokoisena kuva pitää esittää, eikä sivua tarvitse uudelleenjärjestellä kuvan lataamisen jälkeen [66, luku 5.4.6].

Kuvien määrä kannattaa pitää myös pienenä sivun uudelleenjärjestelyn aiheuttamien hidastelujen vuoksi. Useiden kuvatiedostojen lataamisesta aiheutuneiden vasteaikojen vuoksi pienemmän kuvamäärän lataaminen on myös nopeampaa kuin isomman, vaikka kuvien koko olisikin sama. Samojen pienten kuvien käytöstä sivuston eri sivuilla voi olla hyötyä, sillä matkapuhelimet osaavat tallettaa nämä välimuistiin. [59, s. 9-10]

4.5.3 Kuvakartat

Kuvakartoilla tarkoitetaan kuvia, joiden osat ovat valittavissa. Kuvan osan valinta, joka yleensä tarkoittaa hiirellä napautusta, siirtää selaimen johonkin toiseen osoitteeseen. Vanhoilla sivuistoilla kuvakartta saattoi olla toteutettuna siten, että kuvan napautus siirsi kohdistimen koordinaatit CGI-ohjelmalle, joka taas ohjasi palvelimelta saatujen tietojen pohjalta selaimen määrättyyn osoitteeseen [87, s. 189-190]. Saavutettavuuden kannalta tällaiset kuvakartat ovat kuitenkin ongelmallisia, koska hiirellä osoittaminen ei ole kaikille mahdollista. Kuvakartat eivät ole myöskään kovin käytettäviä kannettavilla laitteilla, joissa on pieni näyttö ja ainoastaan numeronäppäimistö [66, luku 5.2.7].

Onneksi useimmiten kuvakartat käsitellään selaimissa. Kuvakartta merkitään dokumenttiin `map`- ja `area`-elementeillä. `area`-elementtiin voidaan merkitä `alt`-attribuutilla teksti, joka selittää kuvan osan niille, jotka eivät voi sitä nähdä. Tällöin teksti on myös valittavissa ilman hiiren napautusta. [87, s. 189-191]

Kaikkien kuvakarttojen tulisi toimia aina asiakaskoneen puolella. Tästä on hyötyä myös graafisen selaimen käyttäjälle, sillä hän näkee sivun latautumisvaiheessa, mihin kohteet

vievät. [54, s. 305] Nykyään latausajat ovat kuitenkin sen verran nopeita, ettei `alt`-tekstien näkymistä ehdi yleensä huomata.

`area`-elementtiin voi myös lisätä `title`-attribuutilla tekstin kuvaamaan tarkempaa tietoa siitä, mihin valinta vie. Tämä teksti näkyy myös useimmissa graafisissa selaimissa ponnahdustekstinä, kun hiiren vie alueen päälle. Myös `map`-elementtiin on syytä lisätä `title`-teksti kuvaamaan valinta-alueen tarkoitusta. Jos jokin alue on merkittävä ja pysyy samana useilla sivuilla, niin siihen voi kytkeä pikanäppäimen `accesskey`-ominaisuudella. Tärkeät alueet voi laittaa myös etusijalle `tabindex`-attribuutilla. [16, s. 186-192]

Kuvakarttoja voidaan käyttää kuitenkin väärällä tavalla esimerkiksi siten, että koko sivun sisältö teksteineen sijoitetaan kuvaan. Tällöin sivulta puuttuu rakenteellinen jäsennys kokonaan. Pelkästään kuvakarttoihin pohjautuvia sivustoja on enää vähän, sillä siirtyminen sivulta toiselle aiheutti aina koko kuvan uudelleenlatautumisen. Useat kehittäjät siirtyivätkin kuvakartoista kuvan pilkkomiseen ja taulukkotaittoon. [87, s. 189-191] Taulukkotaitosta lisää luvussa 4.6.5.

Kuvakartoilla on kuitenkin oma järkevä tarkoituksensa ja se on yksittäisissä kuvissa, jotka jäsentyvät karttojen tapaan.

4.5.4 Kuville on tarjottava vaihtoehtotekstit

Yhteenveto

- ✓ Kuvien vaihtoehtotekstit ja kuvaukset ovat välttämättömiä *ruudunlukijoita*, *kohopistekirjoittimia* ja *tekstipäätteitä* käyttäville.
- ✓ Kuvien pieni ja määrätty koko hyödyttää *mobiililaitteita* käyttäviä.
- ✓ Kuvakarttojen osille määrätty sarkainjärjestys helpottaa navigointia *ruudunlukijoilla* ja *kytkinohjauksella*.

4.6 Sivun taitto

Hyvin suunniteltu sivu voi olla visuaalisesti näyttävä ja samalla saavutettava. Saavutettavuuden kannalta sivun taitossa olennaista on se, että sivun sisältö tulee järkevässä järjestyksessä, jos sitä käsitellään lineaarisesti ilman asemointimäärityksiä. Tämän toteuttamiseksi vaihtoehtoina on käyttää joko CSS-asemointia tai yksinkertaista taulukkotaittoa.

4.6.1 Värien merkitys sivun osissa

Jos jonkin osan värillä on merkitys sivun toiminnan tai hahmottamisen kannalta, niin tällöin myös on oltava jokin toinen menetelmä asian ilmaisemiseksi. Esimerkiksi jos lomakkeella on pakolliset kentät merkitty punaisella värillä, niin esimerkiksi tähtimerkki selitteellä paikkaa tilanteen, jossa värejä ei voida käyttää. Värit aiheuttavat ongelmia, jos käyttäjä on esimerkiksi värisokea juuri valituille väreille tai jos käytössä on matkapuhelin, jossa ei ole värinäyttöä. [13, luku 6.2][16, s. 206-209]

Väreillä ei saa välittää sellaista informaatiota, mitä ei voi muuten saada, koska kaikissa laitteissa ei ole värinäyttöä. Selaimien asetuksia mukauttavat tai ruudunsuurentajia käyttävät saattavat muuttaa tekstin ja taustan väriasetuksia, jolloin alkuperäinen merkityksellinen väritys ei enää toimi.

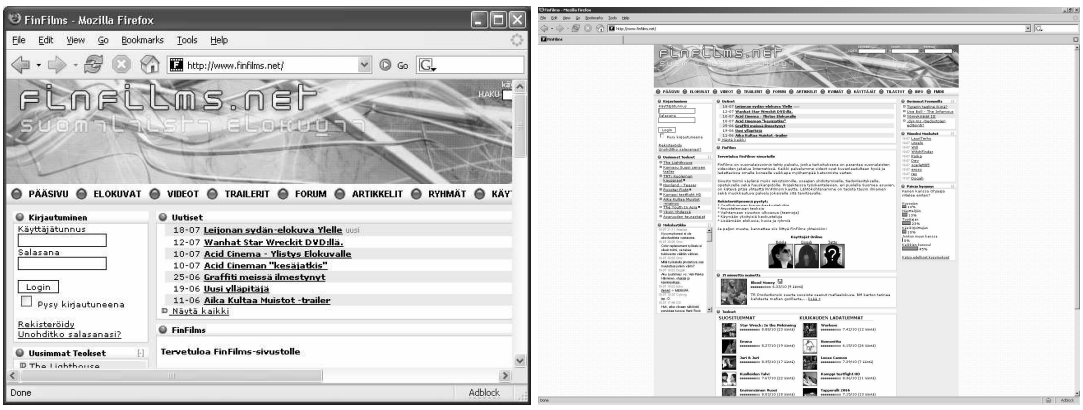
4.6.2 Visuaalinen skaalautuvuus

WWW-ympäristössä emme voi tietää, minkä kokoinen ruutu käyttäjällä on [16, s. 44-45]. Näyttöjen koot ovat muuttuneet myös ajan myötä. Nykyhetkellä verkkoa voidaan selata hyvin erikokoisilla näytöillä aivan pienimmistä matkapuhelinnäytöstä suuriin plasma- tai nestekidenäyttöihin asti.

Sivun taitto erikokoisilla näytöillä

Kiinteät korkeuden ja leveyden määritykset pikseleillä ovat ongelmallisia isoilla monitoreilla, koska tilaa jää käyttämättä. Pienillä monitoreilla sisältö ei mahdu ikkunaan ja on hankalasti käytettävissä, koska vaakasuuntainen vierityspalkki tulee näkyviin. Kiinteät taitot aiheuttavat ongelmia myös tulostuksessa, jolloin sivun oikea osa saattaa jäädä pois paperilta. [56]

Perusperiaate resoluutioriippumattomassa ulkoasussa on olla käyttämättä kiinteitä pikseleitä elementtien koon määrityksissä. Kiinteiden määritysten sijaan koot on määriteltävä sommittelussa käyttäen prosenttiyksikköjä. [54, s. 29]



Kuva 14. Kiinteään pikselikokoon taulukkotaitettu sivusto.

Kuvassa (Kuva 14) on `finfilms.net`-palvelu 640×480 ja 1600×1200 pikselin näytön tarkkuudella tarkasteltuna. Matalalla resoluutiolla lähes puolet sivun sisällöstä menee oikean laidan yli ja on saavutettavissa ainoastaan vaakavierityksen avulla. Yläosan koristeellinen kuvabanneri täyttää lähes puolet ensimmäisestä ruudullisesta. Korkealla resoluutiolla oletuskirjasimen koko on liian pientä luettavaksi ja suurennettu teksti valuu

tiukasti asetelluista laatikoista ulos. Lähes puolet selaimen tarjoamasta tilasta jää myös käyttämättä.

Taiton skaalautuvuus tekstin kokoon

Saavutettavuuden kannalta on tärkeää, että käyttäjä voi määrittää itse tekstin koon (kts. luku 4.3.2). Heikkonäköiset käyttäjät voivat tarvita tekstin suurennosta, mutta suuri teksti voi rivittyä oudosti tai vuotaa yli kiinteään kokoon määrätyistä lohkoista [75, s. 196]. Tekstin kokoa pitäisi pystyä muuttamaan rikkomatta sivun taittoa. Ongelmat johtuvat siitä, että sivu suunnitellaan luottaen määrätyn koon käyttämiseen [39, luku 5]. Jos sivun taitto on kiinteästi määrätty tiettyyn pikselikokoon, niin kirjasimen koon kasvattaminen tekee tekstistä heikosti luettavaa [56].

Sivun taitosta saadaan helpoiten skaalautuva määrittelemällä lohkon koko ja asemointi prosenteilla pikselien sijaan [75, s. 196]. Sivun taiton skaalautuvuus useisiin kirjasimen kokoihin ei ole kuitenkaan aina helposti tehtävissä, joten on hyväksyttävää, jos sivusto näyttää hieman kehnommalta isolla kirjasimen koolla [54, s. 302-303]. Lohkoille voi määrittää soveltuvan minimikoon pikseleissä niin, ettei teksti valu alueen yli. Muutoin lohkon koko pitäisi määrittää suhteellissa yksiköissä, jotka ovat joko verrannollisia tekstin tai ikkunan kokoon. Tämä koskee myös ns. *taulukkotaittoa*, jossa sivun sisällön visuaalinen asemointi tapahtuu `table`-elementin avulla.

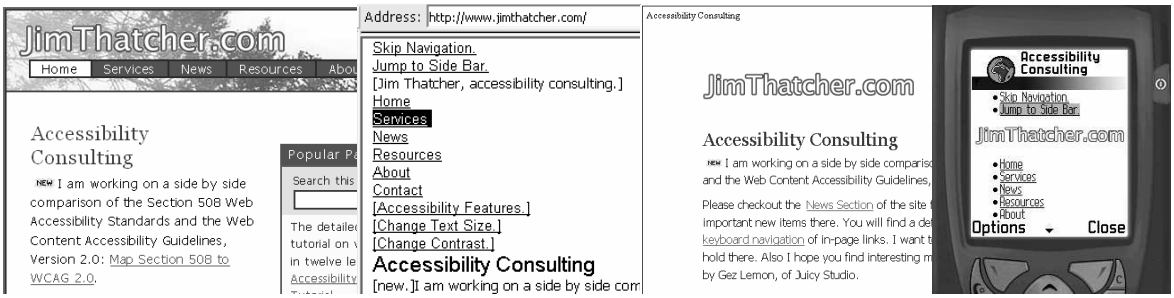
4.6.3 Sivun ulkoasun mukauttaminen

Käyttäjillä on erilaisia toiveita ja tarpeita sivun ulkoasun suhteen pystyäkseen saavuttamaan sivun sisällön mahdollisimman hyvin. Sivun yleinen skaalautuvuus tekstin ja ikkunan kokoon parantaa saavutettavuutta erikokoisilla graafisilla selaimilla, mutta tämä ei välttämättä riitä kaikille laitteille. Vaativammissa tilanteissa ulkoasua on mukautettava erikseen.

Mediatyypit mahdollistavat sivukomponenttien asemoinnin ja ulkoasun muokkaamisen uudella tavalla perustuen käyttäjän selainohjelman tai laitteen ominaisuuksiin. Sivun tekijä voi määrätä sopivia CSS-ominaisuuksia, jotka ohittavat normaalit perusmääritykset tietyissä tiloissa. CSS2-spesifikaatiossa on määritelty mediatyypit

- äänisyntetisaattoreille,
- kohopistekirjoittimille,
- päätelaitteille,
- tulostimille,
- projektoreille,
- kämmenlaitteille,
- televisioille ja
- kuvaruuduille.

Selainohjelma tunnistaa, mitä tyyppiä laite on ja osaa sen perusteella valita sopivat CSS-määritykset. Kolmea ensimmäistä mediatyyppiä tuetaan erittäin huonosti. Tämä voi osaksi johtua siitä, että avustavilla tekniikoilla ei ole toistaiseksi suoraa mahdollisuutta käsitellä selaimien sisäisiä tyylisivuja. Toisaalta näiden mediatyyppien tarjoamat mukauttamisominaisuudet ovat melko tarpeettomia. Harva ääniselaaja haluaa esimerkiksi antaa sivun tekijän muuttaa tarkasti valikoituja oletusääniasetuksia. [16, s. 264-271]



Kuva 15. Sama sivu graafisella selaimella, ääniselaimella, tulostimella ja matkapuhelimella.

Kuvassa (Kuva 15) on avattu `jimthatcher.com` neljässä erilaisessa tilassa. Käyttäjän ei ole tarvinnut itse tehdä mitään valintoja, vaan selain on automaattisesti valinnut sopivat asetukset. Ensimmäisessä kuvassa käytetään ulkoasumäärityksiä, jotka soveltuvat normaaleille monitorinäytöille. Toisessa kuvassa ääniselain on ohittanut kaikki CSS-määritykset ja käsittelee pelkästään XHTML-lähdekoodia. Nyt myös hyppylinkit ovat nähtävissä. Kolmas tila on tavallisen selaimen tulostustilasta. Tässä tilassa CSS-määritykset piilottavat taustakuvat ja navigointipalkit. Neljännelle tilalle ei löytynyt erillisiä CSS-määrityksiä, joten sivu näytetään lineaarisessa järjestyksessä. Tällä sivulla

muotoilematonkin ulkoasu toimii ja kuvat näkyvät hyvin. Kannettavia pienlaitteita varten olisi voinut tehdä myös omat ulkoasumääritykset.

CSS3-spesifikaation luonnoksessa on esitetty *mediakyselyt* (engl. media queries) uutena tapana selvittää ja asettaa esitystapaan liittyviä CSS-määrityksiä tietyille laitteille. Mediakyselyiden avulla sivun tekijä voi määrätä CSS-määrityksiä, joita käytetään vain tietyn parametrijoukon täyttävissä laitteissa tai ohjelmissa. [47]

CSS mahdollistaa myös *vaihtoehtoiset tyylimääritykset*. Näiden avulla voidaan toteuttaa sopiva ulkoasu esimerkiksi heikkonäköisille, joille ei vielä ole määritelty erillistä mediatyyppiä. Tyylin vaihto voidaan tehdä joko selaimen valikosta tai sivulta JavaScript-kieltä käyttäen.

Vaihtoehtoiset tyylimääritykset ovat parempi vaihtoehto kuin erilliset toimintarajoitteisille tehdyt sivut, koska kaikkien käytössä on tällöin yhteinen sisältö. Erilliset sivut voivat olla tarpeettomasti pelkästään tekstimuotoisia, niissä voi olla vähemmän sisältöä ja niitä voidaan päivittää hitaammin. Heikkonäköiset tai sokeat käyttäjät eivät halua erillisiä sivuja. Erilliset sivut lajittelevat toimintarajoitteiset omaksi erikoisryhmäkseen, mikä voi johtaa syrjintään. [16, s. 139-140] [76]

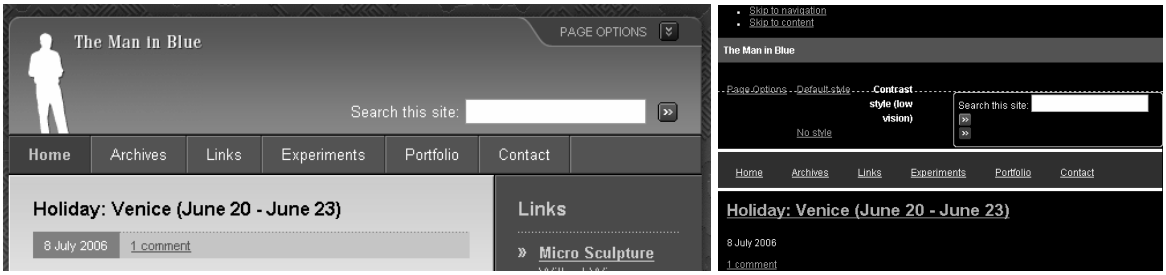
Sivun esitystavan mukauttaminen ruudunsuurentajille

Jos näkökyky on hyvin heikko, on eräs mahdollisuus käyttää ruudunsuurentajia. Suurentajat kasvattavat myös käyttöjärjestelmän ikoneiden ja tekstin kokoa, mikä on usein tarpeen. Tällöin ei myöskään ole merkitystä, minkä kokoiseksi teksti on sivustolla määrätty. Suurennos kasvattaa myös kuvien kokoa, jolloin näiden tarkkuus kärsii. Tekstin suurennos saatetaan myös joskus toteuttaa pisteittäin, jolloin palikkaisuus tulee näkyviin koko sivulla. [16, s. 224-225]

Uusissa selaimissa on nykyään myös toimintoja, joiden avulla sivun voi suurentaa ilman erillisiä ohjelmia tai helppokäyttötoimintoja. Opera ja IE7 Beta tukevat *page zoom* -toimintoa, joka kasvattaa tekstien ja kuvien kokoa, mutta säilyttää silti sivun taiton. Teksti ei menetä tarkkuuttaan, mutta bittikarttagrafiikan tarkkuus kärsii. Toiminto voi myös

tuoda vaakasuuntaisen vierityspalkin. Heikkonäköinen voi arvostaa myös pelkän tekstin koon kasvattamista, koska tällöin navigointi on helpompaa [76].

Useimmat heikkonäköiset, jotka joutuvat käyttämään ruudunsuurentajia, pitävät enemmän tekstistä, joka on vaaleaa tummalla tai mustalla pohjalla. Kaikille käyttäjille tämä ei välttämättä ole visuaalisesti miellyttävä vaihtoehto. Paras tapa tällaisen toteuttamiseksi onkin käyttää *vaihtoehtoisia tyylimäärityksiä*, joiden avulla käyttäjä voi valita hänelle parhaaksi sopivan ulkoasuvaihtoehdon sivustolla.



Kuva 16. Vaihtoehtoisilla tyylimäärityksillä voidaan mukauttaa sivusto heikkonäköisille.

Kuvassa (Kuva 16) on vasemmalla `themaninblue.com`-etusivu oletusulkoasulla ja oikealla heikkonäköisille mukautettuna. Sivulla käytettävä tyylitiedosto on vaihdettu oikean yläkulman `PAGE OPTIONS` -valikon kautta. Mukautetussa sivussa on vain yksi palsta ja teksti erottuu kirkkaana tummasta pohjasta.

Palstoitus on ongelmallinen heikkonäköisille, jotka käyttävät ruudunsuurentajia. Heikkonäköiset ovat hyvin hiiriorientoituneita ja liikkuvat vikkelästi paikasta toiseen kadottaen joskus sijaintinsa sivustolla. Erityisesti oikeassa laidassa olevat pystysuuntaiset palstat jäävät näiltä käyttäjiltä huomaamatta. Vaakasuunnassa toisistaan melko pienelläkin välillä erotetut elementit saattavat jäädä huomaamatta suurennoksen takia. Heikkonäköiset vaihtavat myös sivuston väritystä, jolloin palstojen rajat saattavat kadota. Vasemmassa laidassa oleva navigointipalkki saatetaan tulkita tällöin sivun sisällöksi. [76]

Vaihtoehtoisilla tyylimäärityksillä voidaan tehdä perusulkoasua saavutettavampi sivun taitto [15]

- poistamalla palstoitus,

- siirtämällä navigoinnit ylälaitaan,
- vähentämällä navigointilinkkien määrää,
- muuttamalla taustan väri tummaksi,
- muuttamalla kirjasimen koko isoksi ja väri vaaleaksi.

Sivun ulkoasun mukauttaminen kannettaville laitteille

Kannettavat laitteet poikkeavat esimerkiksi pöytäkoneista merkittävästi [37]:

- Näytön koko on pieni ja vaihtelee laitteittain.
- Jotkin mobiililaitteet kykenevät ainoastaan sisällön pystysuuntaiseen vieritykseen.
- Vierityspalkkeja ei ole aina näkyvillä.

Joustava ja skaalautuva sivun ulkoasu ei ole riittävä hyvin kapeissa näytöissä, koska tekstille jäävä tila ei ole riittävä erilaisten marginaalien, täytteiden ja asemointien takia [84]. Esimerkiksi monipalstainen sivu on mahdollista saada mukautumaan selaimen tai ikkunan kokoon, mutta kannettavien laitteiden pienille ja useimmiten kapeille ruuduille ei voida mitenkään mahduttaa useita palstoja. Sivusta on tehtävä yksipalstainen ja supistettu navigointiosa on siirrettävä sivun ylälaitaan.

Interaktiivisen sivun on oltava lyhyt, mutta tekstisivu voi viedä useampikin ruudullisia. Sivujen pilkkominen selainikkunaan sopivaksi ei ole hyvä strategia. Käyttäjät pitävät miellyttävämpänä, jos sivulla on hieman pituutta pystysuuntaisesta vierityksestä huolimatta. Tämä johtuu pääasiassa hitaista latausajoista. [37]

4.6.4 Kehykset

Kehyksien käyttöä on syytä välttää, koska ne eivät yleisesti ottaen ole käytettäviä ja parempia tekniikoita on tarjolla. Kehyksellisistä sivuista on hankala tehdä kirjanmerkkejä, ne ovat hitaita ladata ja ne pakottavat tiettyyn ikkunan minimikokoon. Kehyksien navigointi on ruudunlukijoilla hankalaa, koska ne tekevät sivun rakenteesta kompleksisen. [16, s. 255-256]

Mikäli kehyksiä tarvitaan, niin tällöin kullekin `frame`-elementille on annettava kehyksen sisältöä kuvaava `title`- ja `name`-attribuutin arvo. Jälkimmäistä ominaisuutta käytetään

linkkien kytkemisessä tiettyihin kehyksiin, mutta myös useiden tekstiselaimien ja ruudunlukijoiden kehyksen valintalistassa [75, s. 124-127]. Kehyssivun `noframes`-osaan pitää sijoittaa linkitykset sivuston eri osiin, jos navigointi kehyksien avulla ei onnistu. Virheilmoitus puuttuvasta kehystuesta ei ole hyväksyttävä toimintatapa [16, s. 257-258].

Kehyksien käyttöä kannattaa välttää myös kannettavien laitteiden takia, sillä rajoitetun tilan takia nämä aiheuttavat käytettävyysongelmia [59, s. 43]. Kaikki kannettavat laitteet eivät myöskään tue kehyksiä [66, luku 5.4.2].

4.6.5 Taulukkotaitto

Taulukkotaitolla tarkoitetaan sivun osien asemointia `table`-elementin avulla. Taulukon solujen reunukset piilotetaan usein, jolloin solut toimivat sisältöä asemoivina lohkoina. Taulukko voidaan sijoittaa toisen taulukon soluun, jolloin sanotaan käytettävän sisäkkäisiä taulukoita. Sisäkkäisillä taulukoilla voidaan muodostaa monipuolisia ja monimutkaisia visuaalisia sivun taittoja kohtuullisen helposti.

Taulukkotaitto on peräisin ajoilta, jolloin sivut suunniteltiin kuvankäsittelyohjelmalla ja navigointi tehtiin kuvakartoilla. Kuvakarttoihin perustuvat sivustot olivat ongelmallisia saavuttavuuden kannalta, koska bittikarttagrafiikka oletti tiettyä näytön kokoa ja graafista selainta. Sivuston päivitettävyys muodostui kuitenkin ongelmaksi, kun uuden asian tai linkin lisääminen vaati sivun uudelleensuunnittelua. Lisäksi pelkkään kuvaan pohjautuva sivu kulutti huomattavasti verkkokaistaa. Tällainen sivusto oli myös täysin ruudunlukijoiden, tekstiselaimien, PDA-laitteiden ja mobiiliselaimien saavuttamattomissa. [87, s. 50-51]

Selaimien välimuistien hyödyntämiseksi useat kehittäjät siirtyivät tekniikkaan, jossa kuva ensin pilkottiin osiin, sijoiteltiin sopivalla taulukkotaitolla yhteen ja toiminnallisuus toteutettiin JavaScript-kielellä. Myöhemmin verkkokaistan säästön ja parempien hakutuloksien toivossa kuvina esitetyt tekstit korvattiin oikeilla teksteillä, jotka muotoiltiin esitystapaan liittyvillä elementeillä tai dokumenttiin upotetuilla CSS-merkinnöillä. Luonnollisesti kaistan kulutus ei tästä merkittävästi pienentynyt, eikä sivuista tullut saavutettavia, koska näillä ei ollut semanttista rakennetta. [87, s. 189-193]

Tämän hetken sivustoista suurin osa perustuu jäykkään taulukkotaittoon, joka on suunniteltu pöytäkoneen tarkkuuksille [84]. Tekstiä ei toteuteta kuvina, mutta se muotoillaan vältettäviksi luokitelluilla elementeillä, joilla ei ole rakenteellista merkitystä (kts. luku 4.1). Usein tyylimääritykset upotetaan suoraan dokumenttiin. Sivuston taittoa ei kuitenkaan enää suunnitella pelkästään grafiikkaan perustuen, vaan nykyään pieniä erillisiä kuvia käytetään koristeina sivun visuaalisen ilmeen toteutuksessa. Sivustot ovat usein monipalstaisia ja pääsivun sisältö on jaettu useihin lohkoihin. Sivun osien asemointia tehdään sisäkkäisillä taulukoilla ja osien välit toteutetaan *spacer*-kuvia käyttäen [87, s. 238, 351]. Monipalstaisuus ja kiinteä taulukkotaitto aiheuttavat kuitenkin ongelmia sivun skaalautuvuudelle ja mukautumiselle erilaisten käyttäjien tarpeisiin.

Taulukkotaiton esteet ruudunlukijoille

W3C:n suosituksissa kehotetaan käyttämään taulukkoja ainoastaan taulukkomuotoisen tiedon esittämiseen ja välttämään taulukkojen käyttöä taitoissa, koska ne aiheuttavat ongelmia ruudunlukijoille [13, luvut 6.3, 6.5]. Useimmille ruudunlukijoille yksinkertainen taulukkotaitto ei ole kuitenkaan ongelmallista. Ruudunlukijoissa on toimintoja taulukon sisäiseen navigointiin [16, s. 232].

Taulukkotaittoa voidaan käyttää, jos taulukon sisältö on linearisoituna ymmärrettävä [13, luku 5.3]. Linearisoinnilla tarkoitetaan ruudunlukijoiden lukutapaa, jossa taulukon solujen sisällöt luetaan rivi kerrallaan vasemmalta oikealle. Jos taulukon solun sisällä on toinen taulukko, niin tämä käsitellään normaalissa lukujärjestyksessä kokonaisuudessa ensin ja vasta sitten jatketaan seuraavaan soluun [75, s. 94]. Riittävän uutta ruudunlukijaa käyttävä voi kuitenkin päättää sisemmän taulukon ohittamisesta ja siirtyä ulomman taulukon seuraavaan soluun [16, s. 233].

Monimutkaiset sisäkkäiset taulukot muodostavat ongelman ruudunlukijoille. Sivun taulukkotaitto ei näy mitenkään näkökykyiselle käyttäjälle, mutta sokea joutuu läpikäymään alla olevan rakenteen. Ruudunlukija antaa jokaisesta solusiirtymästä tai uudesta taulukosta ilmoituksen äänellä, mikä voi tehdä taulukkosokkelossa navigoinnista sekavaa. Pelkän äänen avulla selattaessa on hankala arvioida, mikä merkitys sisemmillä taulukoilla on ja mihin suuntaan pitäisi navigoida. Käyttäjä voi kuunnella sisällöt

järjestyksessä ilman rakenneinformaatiota, mutta tällöin linearisoidun järjestyksen on oltava järkevä. [16, s. 233-234]

Hyvässä saavutettavassa rakenteessa ei ole sisäkkäisiä taulukoita. Sivulla voi olla kuitenkin useampia taulukoita, esimerkiksi navigointia ja sisältöä varten. Tällöin taulukoiden `summary`-ominaisuudessa voidaan kertoa näiden osien merkitys. Ruudunlukija voi lukea koosteet, jolloin käyttäjä tietää mitä taulukot sisältävät. [87, s. 198-200]

Muutoin taulukkotaitossa on vältettävä kaikkia mahdollisia ominaisuuksia, jotka parantavat normaalien tietotaulukoiden saavutettavuutta. On käytettävä pelkästään `table`-, `tr`- ja `td`-elementtejä. Taulukon `summary`-ominaisuuskin voidaan jättää tyhjäksi. Tällöin ruudunlukijoita käyttävien ei tarvitse kuunnella tarpeettomasti otsikkotekstejä. [16, s. 232, 237]

Taulukkotaitto mobiiliselaimessa

Taulukot eivät sovellu sivun taittoon hyvin mobiiliselaimien pienillä näytöillä. Sisäkkäisiä taulukoita on vältettävä ja on suositeltavaa käyttää vaihtoehtoisia tapoja esittää taulukkomuotoon aseteltua informaatiota. Taulukoiden selaaminen voi vaatia pysty- ja vaakasuuntaista vieritystä, mikä ei ole toivottavaa. [66, luku 5.4.4]

Vaakasuuntainen vieritys aiheuttaa turhautumista ja joissain laitteissa se ei ole edes mahdollista. Operan selaimet kykenevät mukauttamaan normaaleja sivuja pienelle ruudulle, mutta paras tulos saavutetaan, jos jo sivuston suunnittelussa huomioidaan näytön koko. [84]

4.6.6 Asemointi CSS-ominaisuuksilla

Taulukkotaitto aiheuttaa ongelmia vanhemmille ruudunlukijoille ja muille selaimille, jotka eivät pysty navigoimaan yksittäisten solujen välillä. Sisäkkäiset taulukot ja läpinäkyvät *spacer*-kuvat muodostuvat ongelmaksi niille selaajille, jotka eivät voi hahmottaa sivua visuaalisesti. CSS2 mahdollistaa täytetilojen, reunaviivojen, marginaalien, sisennyksien ja tasauksien määräämisen. Tyylisivuilla sisällön osat voidaan täsmällisesti asemoida mihin tahansa kohtaa ruutua tai liu’uttaa ruudun laitoihin. Sisältö voidaan asetella visuaalisesti eri

järjestykseen kuin lähdekoodissa. Tyylisivujen käyttö ohjaa rakenteelliseen ajatteluun ja HTML-merkintöjen oikeaoppiseen käyttöön, koska CSS-määritykset voidaan kokonaan erottaa omaan tyylitiedostoon ja kytkeä haluttuihin elementteihin. Absoluuttinen asemointi mahdollistaa lähdekoodin organisoinnin siten, että tärkeimmät osiot ovat alussa. Tästä on hyötyä esimerkiksi ruudunlukijoille. Rakenteisesti suunnitellun sivun kanssa voidaan käyttää myös useita tyylisivuja, joka mahdollistaa erilaisen osien asemoinnin erilaisilla laitteilla. [75, s. 232-233, 240-241]

Esteellisiä CSS-muotoiluja

CSS-määrityksissä on syytä käyttää suhteellisia yksiköitä tai hyvin pieniä pikselimääriä. CSS:n avulla voidaan tehdä taittoa pikselin tarkkuudella, mikä aiheuttaa osittain samankaltaisia ongelmia kuin mitä kiinteään kokoon määrätyssä taulukkotaitossa on. Kiinteisiin kokoihin määrätty sivu ei mukaudu hyvin esimerkiksi heikkonäköisen tarpeisiin. [75, s. 196]

Sivujen pitää myös toimia ilman tyylitiedostoja, jos käyttäjä tai käyttäjän selain ei halua hyödyntää niitä. Ilman tyylitiedostoa toimivista sivuista on hyötyä sokeille ja heikkonäköisille käyttäjille, selaimen asetuksia mukauttaville käyttäjille ja ihmisille, jotka käyttävät vanhoja selaimia. Mahdollisia ongelmatilanteita ovat esimerkiksi tekstien asemoinnit, joilla saadaan aikaan jokin visuaalinen efekti, mutta joissa ei ole järkeä, kun sisältö linearisoidaan [54, s. 81]. Sivun osien vapaa absoluuttinen asemointi voi myös johtaa siihen, että alla olevan HTML-koodin järjestyksestä ei välitetä. Tällöin ruudunlukija voi saada sisällöt sekavassa järjestyksessä [75, s. 196].

CSS-asemoinnilla ei saa antaa rakenteellisille osille sellaisia merkityksiä, jotka ovat ymmärrettävissä vain kun sivua tarkastellaan graafisella selaimella. Tällainen tapaus voisi olla esimerkiksi datataulukon muotoon asettuvat lohkot, joita ei ole merkitty `table`-elementillä. Sivun ja käyttäjän tyylimääritykset eivät saa mennä ristiin. Paras tapa tämän toteuttamiseksi on käyttää erillisiä tyylitiedostoja [75, s. 196-197, 376].

Tyylimääritysten oikeaoppinen käyttö vaatii verkkosivuston tekijältä totuttautumista kokonaan uuteen suunnittelumalliin. CSS-muotoilujen ja -asemointien sotkeminen

vanhojen taittotapojen kanssa ei tuota yleensä hyvää lopputulosta. Huonon lähestymistavan voi tunnistaa seuraavista HTML-koodin piirteistä:

- Lähes jokaiselle elementille on annettu oma `class`-attribuutti, vaikka elementit olisivatkin sisäkkäin.
- Sivulla on useita `style`-attribuuteilla upotettuja tyylimäärityksiä.
- Sivu on täynnä `div`- ja `span`-elementtejä, mutta rakenteelliset elementit puuttuvat (esim. otsikot `h1` - `h6`).

Merkintäsotku voi olla myös jonkin suunnittelutyökalun aikaansaannos, sillä visuaaliset editorit eivät useinkaan osaa yleistää samankaltaisia sivun osia omiksi luokikseen samalla tavalla kuin ihmiset. Upotetut tai näennäisesti luokitellut tyylimääritykset eivät säästä verkkokaistaa ja tekevät ulkoasun mukauttamisesta hankalaa. Yleisten lohkojen liiallinen käyttö kertoo siitä, että tekijä ei ole ymmärtänyt rakenteellisten elementtien merkitystä. Valitettavan usein uutisotsikot tai sivun linkkilistat merkitään CSS-tyylitellyillä `div`- ja `span`-elementeillä, vaikka rakenteellinenkin vaihtoehto olisi olemassa. [87, s. 182-186, 228-229]

4.6.7 Sivun taiton on mukauduttava käyttötilanteeseen

Yhteenveto

- ✓ *Värisokeat, ruudunsuurentajia* käyttävät ja värinäytöttömien *kannettavien laitteiden* käyttäjät hyötyvät, kun väreillä ei anneta sivun osille merkityksiä.
- ✓ *Erikokoisia näyttötiloja* ja *selaimien asetuksia mukauttavat* käyttäjät hyötyvät yleisestä skaalautuvuudesta ikkunan ja tekstin kokoon.
- ✓ *Ruudunsuurennosta* ja *pieniä näyttöjä* käyttävät hyötyvät yksipalstaisesta taitosta, joka voidaan toteuttaa vaihtoehtoisilla tyylisivuilla.
- ✓ *Ruudunlukijoita* käyttäville yksinkertainen taulukkotaitto tai asemointi CSS:llä mahdollistaa sisällön helpon saamisen. CSS on välttämätön *mobiililaitteille*.

4.7 Vuorovaikutus

Verkkosivuilla tapahtuvaa vuorovaikutusta on kahdenlaista. Lomakkeita täyttämällä ja lähettämällä käyttäjä voi kommunikoida palvelimella olevan ohjelman kanssa. Saavutettava lomake on helposti käytettävissä erilaisilla ohjauslaitteilla, kuten näppäimistöllä, kytkinohjauksella ja matkapuhelimen numeronäppäimistöllä [16, s. 279].

Lomakkeiden käsittely tapahtuu palvelimella, joten tällä ei ole suoraa vaikutusta saavutettavuuteen. Yleensä käsittely päättyy siihen, että palvelin lähettää uuden sivun selaimelle. Saatavassa vastineessa pitäisi olla ensimmäisenä olennaisin informaatio [16, s. 278]. Tämä on usein käsittelyn tulos. Muutoin lomakkeiden saavutettavuus riippuu staattisen lomakkeen rakenteesta.

Verkkosivulle voidaan myös liittää selaimessa käsiteltäviä ohjelmia, joiden toiminta riippuu käyttäjän tekemisistä verkkosivustolla. Tapahtumapohjainen verkkosivusto on saavutettava, jos sen toimivuus on varmistettu avustavilla tekniikoilla ja se toimii myös ilman komentosarjoja [13, luku 6].

4.7.1 Lomakkeiden käyttö ruudunlukijoilla

Saavutettavuus ruudunlukijoilla, kohopistekirjoittimilla ja kytkinohjaimilla on kiinni siitä, kuinka hyvin lomake on käytettävissä näppäimistöltä. Navigointia ja lomakkeen ymmärtämistä voidaan helpottaa [12, luku 11]

- antamalla jokaiselle lomake-elementille otsikkoteksti (engl. label) ja kytkemällä ne eksplisiittisesti toisiinsa,
- ryhmittelemällä lomake-elementtejä `fieldset`- ja `legend`-elementillä,
- antamalla lomake-elementeille looginen läpikäyntijärjestys `tabindex`-ominaisuudella, ellei lähdekoodin mukainen lineaarinen järjestys ole järkevä, tai
- määräämällä tärkeimmille lomake-elementeille pikanäppäin `accesskey`-ominaisuudella.

Otsikkotekstin ja lomake-elementin kytkentä onnistuu sijoittamalla lomake-elementti `label`-elementin sisälle tai liittämällä ne toisiinsa `for`- ja `id`-ominaisuusparilla. Tämä on ainut tapa varmistaa, että ruudunlukijaa käyttävä täyttää varmasti oikean kentän. Pelkkään otsikon asemointiin tai paikkaan lähdekoodissa ei voi luottaa. Jos saman otsikon pitäisi toimia useammalle lomake-elementille otsikkona tai otsikkoa ei ole järkevä toistaa visuaalisista syistä (esim. kysely Likert-asteikolla), niin voidaan käyttää samanlaisia tekstin piilotustekniikoita kuin hyppylinkkien tapauksessa (kts. luku 4.4.2) tai sijoittaa otsikkoteksti lomake-elementin `title`-ominaisuuteen. [75, s. 150-154]

Lomaketta voidaan selventää merkitsemällä lomakkeen osakokonaisuudet omiksi ryhmikseen. Ryhmä erotetetaan useiden selaimien oletusulkoasuissa reunaviivalla [75, s. 155]. Ruudunlukijoista Window-Eyes mahdollistaa nopean siirtymisen lomakeryhmien välillä ja ilmoittaa kunkin ryhmän alussa ryhmän nimen [29]. Jaws ilmoittaa ryhmän nimen jokaisen lomake-elementin kohdalla [24]. Jos valintanappeja tai -ruutuja on useita peräkkäin, nämä on hyvä sijoittaa omaan ryhmään. Tällöin `legend`-elementissä voi kertoa, mikä on valintaryhmän tarkoitus.

Pikanäppäinkomentoja on syytä antaa vain tärkeille kohteille, eikä kaikille lomake-elementeille, sillä näppäimistökäyttäjä pystyy siirtymään sarkainnäppäimellä helposti

seuraavaan kohtaan. Pikanäppäimiä kannattaa käyttää lomakkeiden `submit`-painikkeissa ja tärkeiden lomakkeiden (esim. haku) alussa. Pitkissä lomakkeissa kunkin osan ensimmäinen lomake-elementti kannattaa merkitä `tabindex`- ja `accesskey`-ominaisuuksilla [16, s. 280-281]. Pikanäppäin voidaan myös kytkeä lomakeryhmän `legend`-elementtiin, jolloin ryhmien välillä siirtyminen helpottuu [75, s. 156].

Lomake-elementin `title`-ominaisuuteen voi myös laittaa lomakkeen täyttöä opastavaa informaatiota. Lisäksi `reset`-painikkeen käyttöä lomakkeella tulisi välttää, koska painike tulee helposti vahingossa valituksi ja sille on harvoin todellista käyttöä. [16, s. 281-282, 288-289]

4.7.2 Lomakkeiden käyttö mobiililaitteilla

Lomakkeella navigointi ja tiedon syöttäminen matkapuhelimien numeronäppäimistöllä on hidasta. Hiirtä vastaavaa kohdistinta ei ole välttämättä käytössä. Linkin aktivointitapa riippuu mobiiliselaimen valmistajasta. Vaihtuvatoimintoisten pikanäppäimien (engl. softkeys) määrä vaihtelee laitteittain. [37]

Mobiililaitteelle sopivalla lomakkeella [60, s. 16, 37-39]

- on mahdollisuus lomakesivuilta poistumiseen,
- on mahdollisuus virhesyöttöjen korjaamiseen,
- on pituutta enintään muutaman ruudullisen verran,
- on enemmän valittavia kuin tekstin syöttöä vaativia elementtejä ja
- elementtien välit ovat pieniä ja ne on sijoitettu yhdelle sivulle.

Vapaita tekstilaatikoita on vältettävä aina, kun on mahdollista. Hitaan tekstin syötön takia valintalistat, -ryhmät ja -napit ovat suositeltavia. Näissä on käytettävä oletusarvoja [66, luku 5.5.1]. Kaikkien elementtien kanssa on suositeltavaa käyttää `label`-elementtiä ja pyrkiä mahduttamaan otsikko ja lomake-elementti yhdelle riville. Samalla rivillä ei saa olla kahta valintaelementtiä, sillä on hankala arvioida mihin otsikkotekstiin valinta kuuluu [60, s. 39-42]. Jos lomakkeen normaali dokumentin mukainen läpikäyntijärjestys ei ole järkevä,

on käytettävä kaikissa lomake-elementeissä `tabindex`-ominaisuutta loogisen järjestyksen tekemiseksi [66, luku 5.5.2].

Joskus lomakkeita voidaan ohjata äänen avulla, esimerkiksi käyttäen VoiceXML-kieltä ja äänentunnistusta. Pelkkään ääniohjaukseen rajoittuminen voi estää sivun käytön niiltä, joilla on puhehäiriöitä tai ongelmia kuulossa [5]. Tällainen tilanne on kuitenkin erittäin harvinainen.

4.7.3 Muita lomakkeiden käyttötapoja

Mobiililaitteita vastaavassa tilanteessa ovat virtuaalinäppäimistöä kytkinohjauksella käyttävät. Tiedon syöttö on todella hidasta. Pitkien lomakkeiden läpikäynti voi olla myös työlästä ja hankalia ovat myös pitkät listat, joista oikean kohdan löytäminen vaatii useita näppäinvalintoja. [16, s. 279, 284]

Ruudunsuurentajia käyttävät voivat helposti kadottaa tärkeitä painikkeita lomakkeella, jos näiden välissä on liikaa tyhjää tilaa tai lomake-elementit ovat liian leveitä [76]. Lomake-elementit pitäisi asemoida mieluiten allekkain kuin vierekkäin.

Lyhyistä lomakkeista on hyötyä käyttäjille, joilla on kognitiivisia vaikeuksia. Visuaalista hahmottamista helpottaa pitkien `select`-elementillä tehtyjen valintalistojen vaihtoehtojen ryhmittely `optgroup`-elementillä. Pitkiä listoja pitäisi toisaalta välttää kokonaan, koska näiden käsittely ruudunlukijoilla ja erilaisilla ohjaustekniikoilla on hidasta [16, s. 284, 286-287]. Valitettavasti avustavat tekniikat eivät vielä hyödynnä tällaisen ryhmittelyn tarjoamia mahdollisuuksia.

4.7.4 Vältettäviä navigointitapahtumia

Tiettyihin elementteihin liittyvillä sopivilla attribuuttien arvoilla saadaan aikaan toiminnallisuutta, joka ei sovellu hyvin ruudunlukijoille tai mobiililaitteille.

JavaScript-linkit

On vältettävä linkkejä, jotka alkavat `javascript`-viitteellä, sillä komentosarjojen tuen puute estää näiden käytön [12, luku 12]. Sivun on myös toimittava ilman komentosarjoja.

Jos tämä ei ole mahdollista, niin `noscript`-elementissä on annettava täysi tekstivastine tai on tehtävä vaihtoehtoiset saavutettavat sivut [13, luku 6].

JavaScript-kieltä ei voida käyttää matkapuhelimiin suunnatuissa palveluissa, koska XHTML Mobile Profile ei tue komentosarjakieliä [60, s. 48]. Tulevaisuudessa ECMAScript-standardista on tulossa versio mobiililaitteille. Toistaiseksi komentosarjojen tuki kannettavissa laitteissa on rajoitettua eikä siihen voida luottaa [66, luku 5.4.5]. Ruudunlukijoilla `javascript`-linkit toimivat sen sijaan ongelmitta, jos linkkiteksti on annettu normaalisti [75, s. 384].

Ponnahdusikkunat

Ponnahdusikkunoiden käyttöä on vältettävä, koska nämä hankaloittavat selailua ja pöytäkoneista poikkeavissa ympäristöissä nämä toimivat eri tavalla kuin sivun tekijä olettaa. Useat mobiiliselaimet eivät pysty esittämään kuin yhden ikkunan ja uuden avautuminen voi aiheuttaa hämmennystä [66, s. 5.2.8]. Series 60 -ympäristössä ponnahdusikkuna näytetään uutena sivuna [60, s. 48]. Sokeat käyttäjät eivät välttämättä heti ymmärrä, että uusi ikkuna on avautunut. Käyttäjät, joilla on oppimis- tai ymmärtämisvaikeuksia, voivat hämääntyä paljon ponnahdusikkunoista [16, s. 80-81].

Ajastetut toiminnot

Tyypillinen ajastettu toiminto on sivun edelleenohjaus tai toistuva päivitys. Näiden käyttöä on syytä välttää, koska käyttäjä hämääntyy ja vuorovaikutus hidastuu. Mobiiliympäristössä toistuva päivitys voi aiheuttaa myös yllättäviä kuluja [66, luku 5.2.8]. Ruudunlukijoilla sivun päivittymisen jälkeen lukeminen aloitetaan uudelleen sivun alusta, jolloin käyttäjä ei välttämättä pääse koskaan haluamaansa osaan sivua [75, s. 199].

Joillain erityissivuilla voi olla myös muunlaisia komentosarjojen avulla ajastettuja toimintoja. Tällöin on ilmoitettava käytettävissä olevasta ajasta ja tarjottava mahdollisuus muokata toiminnon päättymisajankohtaa. [75, s. 161-162]

4.7.5 Tapahtumapohjainen vuorovaikutus ruudunlukijoilla

Yksinkertaiset komentosarjat toimivat useimmissa ruudunlukijoissa, koska ruudunlukijat toimivat normaalien selaimien päällä. Nykyisissä verkkosovelluksissa JavaScriptiä käytetään toteuttamaan graafista käyttöliittymää vastaavia komponentteja ja näiden toiminnallisuutta. HTML ei kuitenkaan kykene tarjoamaan avustaville tekniikoille riittävästi semanttista informaatiota verkkosivulla olevien komponenttien tilasta ja roolista. [69]

Ruudunlukijoille ongelmia aiheuttavat

- hiiren toiminnoista riippuvat herättimet,
- erilaiset tilanvaihdostapahtumat ja
- sivun sisältöä muuttavat komentosarjat.

Näppäimistöllä on mahdollista laukaista kertapainallusta vaativat `onclick`- ja `onkeypress`-tapahtumat. Ohjauslaitteista riippumattomat `onload`- ja `onunload`-tapahtumat eivät muodosta ongelmaa. Tuplanapautusta tai hiiren liikkeitä vaativat tapahtumat ovat sen sijaan ongelmallisia. Yleisesti käytetty `onchange`-tapahtuma on ongelmallinen, koska ruudunlukijaa käyttävä voi tahattomasti käynnistää sen [75, s. 385-386]. Mobiiliselaimilla, jotka tukevat komentosarjoja, suositellaan käytettävän `onclick`-herätintä `onkey`- ja `onmouse`-herättimien sijaan [66, luku 5.4.5]. On suositeltavaa käyttää mieluummin loogisia kuin laitteista riippuvia tapahtumakäsittelijöitä [13, luku 9].

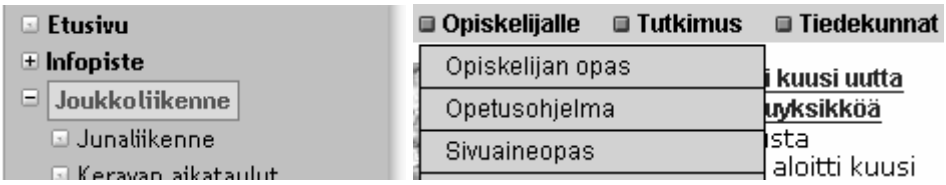
DOM-rajapinnan avulla voidaan muokata dokumentin sisältöjä ja ulkoasua. Näin voidaan toteuttaa erilaisia dynaamisia valikoita ja toimintoja. Graafisessa selaimessa sisällön muutos on ilmeinen ja välitön, mutta ruudunlukija ei välttämättä ymmärrä ilmoittaa tapahtuneesta muutoksesta [75, s. 198]. Vaikka muutos huomioitaisiinkin, niin käyttäjä ei välttämättä pääse heti lukemaan tapahtunutta muutosta tai ei saa selville, missä muutos tapahtui. Hyvin toteutettu komentosarja voi pyrkiä asettamaan kohdistuksen lähelle uutta sisältöä, jolloin ruudunlukija pääsee jatkamaan lukemista uudesta sisällöstä alkaen. Toistaiseksi XHTML-spesifikaation mukaan fokus voidaan kuitenkin antaa vain linkeille, objekteille ja lomake-elementeille, mutta ei esimerkiksi tekstikappaleille [44]. Ongelma

voidaan kuitenkin kiertää uusimmissa Internet Explorer 6 ja Mozilla Firefox 1.5 -selaimissa määräämällä kohdistettavan tekstilohkon `tabindex`-ominaisuuden arvoksi luku -1, jolloin fokus on mahdollista siirtää kappaleeseen komentosarjan avulla [69].

Tämä kiertotie ei ole paheksuttava, sillä XHTML 2 -standardi mahdollistaa minkä tahansa elementin kohdistamisen. Ongelmatonta dynaamisen sisällön saavutettava käsittely ei kuitenkaan ole.

Valikot

Tyypillinen esteellinen komentosarjoilla toteutettu komponentti on alasvetovalikko, joka tuo esiin uusia linkkivaihtoehtoja. Näiden ongelmana on riippuvuus hiirestä, jota sokeat tai liikuntavammaiset eivät voi käyttää. Tällaisista valikoista saadaan melko saavutettavia, kun yläotsikoista tehdään linkit sivuille, jotka sisältävät alivalikossa olevat linkkivaihtoehdot [16, s. 196].



Kuva 17. Dynaamiset valikot voivat olla esteellisiä.

Kuvassa (Kuva 17) on vasemmalla `kerava.fi`-sivustolta löytyvä hierarkkinen vakionavigointikomponentti ja oikealla `jyu.fi`-sivuston etusivulta löytyvä ponnahdusvalikko. Keravan kaupungin sivuston vakionavigointipalkkia pystyi ennen uudistusta (30.3.2006) käyttämään ainoastaan hiirellä JavaScript-tuki päälle kytkettynä. Nykyään palkkia voidaan käyttää myös näppäimistöltä nuolinäppäimiä käyttäen. Ruudunlukijaa käyttävä voi silti hämääntyä, koska osa valinnoista muuttaa pelkästään navigointipalkin sisältöä. Uloimman tason linkkejä voidaan valita, vaikka käytössä ei olisikaan komentosarjatukea. Jyväskylän yliopiston etusivulla ei voi näppäimistöllä valita ponnahdusvalikoista löytyviä kohtia ilman hiirtä. Yläkohdat toimivat kuitenkin linkkeinä alasivuille, joista valikkolinkit ovat löydettävissä kohtuullisella vaivalla. Ratkaisu kuitenkin asettaa käyttäjät eriarvoiseen asemaan.

Useilta sivuilta löytyy valintalistalla tehty valikko, josta valintavaihtoehdon valitsemalla pääsee suoraan seuraavalle sivulle. Tällainen hyppyvalikko on kuitenkin ongelmallinen ruudunlukijoille, jotka eivät välttämättä pääse muutamaa vaihtoehtoa pidemmälle, kun selain siirtyy jo uuteen osoitteeseen [75, s. 199]. Valintalistalla toteutettu navigointi ei ole myöskään hyvä normaalin käytettävyyden kannalta, koska virhevalinnan jälkeen ei ole mahdollisuutta peruuttaa komentoa. Tällaisen valikon käytettävyys- ja saavutettavuusongelmat voidaan kuitenkin korjata yksinkertaisesti luopumalla komentosarjoista ja laittamalla valikon viereen tavallinen `submit`-painike [16, s. 293].

Komentosarjojen avulla toteutetut valikot eivät toimi myöskään useissa matkapuhelinlaitteissa. Standardikeinoin tuotetut valikot kuluttavat myös vähemmän älypuhelimen akkua. [66, luku 5.4.5]

Verkkosovellukset

Uusimmat graafiset selaimet mahdollistavat sisällön hakemisen palvelimelta `XMLHttpRequest`-objektin avulla ilman, että sivua tarvitsee päivittää. Haettu data voidaan siirtää suoraan dokumenttiin DOM-rajapinnan avulla tai sen perusteella voidaan dynaamisesti muokata sivun CSS-ulkoasumäärityksiä. Tästä tekniikkakokoelmasta käytetään nimitystä AJAX. Sen avulla voidaan toteuttaa lähes reaaliaikaisesti toimivia verkkosovelluksia. Tunnetuin AJAX-esimerkki on Googlen GMail-sähköpostipalvelu.

Saavutettavien AJAX-palveluiden tekemisessä on huomioitava luvun alussa esitettyjen kohdistusongelmien lisäksi tilan muutostapahtumiin liittyvät ongelmat. Verkkosivun tekstiosia ja taulukoita luettaessa ruudunlukijat joutuvat käyttämään ns. *virtuaalipuskuritilaa*. Tässä tilassa ruudunlukija käsittelee omassa muistissaan olevaa kopiota selaimen dokumenttipuusta. Normaalissa tapahtumakäsittelijässä `XMLHttpRequest`-objektilla pyydetyn datan saapuminen selainohjelmalle huomioidaan toisella `onreadystatechange`-tapahtumalla, joka mahdollistaa dokumentin sisällön muokkaamisen asynkronisesti. Tilan muutostapahtuma päivittää virtuaalipuskurin sisällön kuitenkin vain harvoissa selain- ja ruudunlukijayhdistelmissä. Ruudunlukijoilla voidaan kuitenkin siirtyä pois puskuritilasta, jolloin sisällön muutokset tapahtuvat oikein. Ongelmana on kuitenkin se, että perustilassa käyttäjä voi lukea vain käyttöjärjestelmän

saavutettavuusrajapinnan tarjoamia kohdistettavia elementtejä, kuten sivulta löytyviä linkkejä tai lomakkeen osia. Muuttuneen tekstisisällön lukemiseksi ruudunlukija on vaihdettava takaisin puskuritilaan. [44]

Tämän hetken selaimia ja ruudunlukijoita tukevissa verkkosovelluksissa on siis toimittava seuraavasti [44]:

1. Ilmoita käyttäjälle, että ruudunlukijan virtuaalipuskuritila on kytkettävä pois päältä.
2. Suorita komentosarja mieluiten `onclick`-tapahtumakäsittelijässä, sillä `onchange`- tai `onblur`-tapahtumat voivat hämmentää käyttäjää.
3. `XMLHttpRequest`-objektin `onreadystatechange`-tapahtumankäsittelijässä on annettava kohdistus muokatulle sivun osalle.
4. Päivitetyn lohkon `tabindex`-ominaisuuden arvoksi on annettava -1, ellei se ole normaalisti näppäimistöltä valittava elementti.
5. Lisätyn kohdan alussa on ilmoitettava esimerkiksi `title`-ominaisuudessa, että sisällön lukemisen ajaksi virtuaalipuskuritila on kytkettävä takaisin päälle.

4.7.6 Kaikki eivät käytä hiirtä verkkolomakkeilla

Yhteenveto

- ✓ Lomakkeen osien ryhmittelyt helpottavat käyttäjiä, joilla on *kognitiivisia vaikeuksia, motorisia häiriöitä* tai jotka käyttävät *ruudunlukijoita*.
- ✓ Pikanäppäimillä, sarkainjärjestyksellä ja otsikkotekstien kytkennöillä helpotetaan lomakkeen käyttöä *ruudunlukijoilla, kytkinohjauksella* ja *mobiililaitteilla*.
- ✓ Lyhyistä ja kompakteista lomakkeista on hyötyä käyttäjille, joilla on *ymmärrysvaikeuksia* tai jotka käyttävät *ruudunsuurentajia* tai *mobiililaitteita*.
- ✓ Ponnahdusikkunat ja ajastetut toiminnot häiritsevät lähes *kaikkia käyttäjiä*.
- ✓ Sivustosta on tarjottava *mobiililaitteille* versio, joka ei käytä komentosarjoja.
- ✓ *Ruudunlukijoita* käyttävät hyötyvät dynaamisten valikoiden linkki-vaihtoehdoista. Sisältöä muokkaavissa komentosarjoissa on muistettava opasteet ja kursorin kohdistaminen.

4.8 Multimedia

Multimedian saavutettavuus riippuu erillistekniikoiden tarjoamista mahdollisuuksista, koska multimediaominaisuuksia ei tueta vielä laajasti suoraan selaimissa. Laajennoksien ja erillisohjelmien kautta tarkasteltavaan verkkosisältöön eivät päde HTML-, XHTML- ja CSS-tekniikkojen saavutettavuusohjeistukset. Toisaalta saavutettavan multimedian tuottamisessa verkkoon on huomioitava samanlaisia asioita kuin perinteisillä sivuilla, sillä verkkosisällön käyttötavat pysyvät samanlaisina.

Seuraavasta W3C:n saavutettavuussuosituksesta onkin tulossa teknologianeutraali. Tekniikasta riippumatta saavutettavuus muodostuu seuraavista pääpiirteistä [7]:

- Kaikille ei-tekstuaalisille sisällöille on annettava tekstivaihtoehdot.
- Esitystapa ei saa estää sisällön saavutettavuutta.

- Sisältöjä pitää pystyä navigoimaan helposti myös näppäimistöltä.
- Käyttäjän on kyettävä kontrolloimaan kaikkea toimintaa.
- Sivuston on toimittava nykyisillä ja tulevilla ohjelmilla.

4.8.1 Multimedian upottaminen verkkosivuille

Multimediaa liitettäessä verkkosivulle on useissa selaimissa mahdollisuus käyttää epästandardia `embed`-elementtiä. Vaikka tämä ei olekaan kelvollinen elementti, niin valitettavasti se on toistaiseksi tuetuin tapa upottaa objekteja sivulle. Toinen tapa on käyttää `applet`-elementtiä, joka on HTML-standardissa, mutta sen käyttöä ei suositella, koska sen käyttö on rajattu ainoastaan Java-sovelmille. Standardinmukainen tapa on käyttää `object`- ja `param`-elementtejä. Objekteja voidaan sijoittaa sisäkkäin. Jos uloin vaihtoehto ei toimi suoraan selaimissa, niin silloin voidaan käyttää sisempiä vaihtoehtoja. Viimeisenä vaihtoehtona on oltava tavallinen teksti, mikä on tarpeellista niiden käyttäjien kannalta, jotka eivät voi katsoa video- tai multimediaesitystä. [16, s. 300-302]

4.8.2 Interaktiivinen vektorigrafiikka

Interaktiivisella vektorigrafiikalla tarkoitetaan matemaattisilla lausekkeilla tallennettuja kuvioita, joita voidaan animoida ohjelmallisesti ja joiden toimintaa voidaan ohjata käyttäjän syötteestä.

Animaatioita tehdessä on erityisesti huomioitava käyttäjät, joilla on taipumusta saada kohtauksia liikkuvista ja välkkyvistä kuvista. Liike voi aiheuttaa ongelmia käyttäjille, joilla on keskittymisvaikeuksia. Näitä käyttäjiä varten on oltava mekanismi, jolla animaation suoritus voidaan keskeyttää. [5]

Flash

Sopivasti parametrisoiduilla sisäkkäisillä `object`-elementeillä on mahdollista saada Flash-animaatiot toimimaan standardinmukaisesti, mutta tällöin suoratoisto ei ole mahdollista. Suoratoisto voidaan toteuttaa käyttämällä sivulla pientä Flash-objektia, joka lataa toisen sisällön. JAWS-ruudunlukijalla ja Internet Explorerilla tämä menetelmä kuitenkin aiheuttaa ongelmia. Eräs mahdollisuus on käyttää JavaScript-kieltä Flash-

sisältöjen upottamiseen, jolloin sivusta tulee ainakin näennäisesti validi ja toimiva niille, jotka eivät voi toistaa esitystä. Tulevaisuudessa voidaan kuitenkin olettaa, että `object`-elementin tuki paranee selaimissa, jolloin kiertoteitä ei tarvita. [87, s. 291-296]

Flash-ohjelman avulla on tuotettu paljon huonosti käytettävää ja esteellistä verkkosisältöä [87, s. 94]. Toisto-ohjelman versiosta 6 lähtien Flash-objekteista on voinut tehdä saavutettavia [16, s. 327]. Flash MX ja sitä uudemmat tuotantoympäristöt mahdollistavat mm. tekstivaihtoehtojen antamisen kaikille komponenteille, animaation ja lomake-elementtien kontrolloinnin ja navigointijärjestyksen määräämisen näppäimistöltä. On silti suositeltavaa tarjota vaihtoehtoinen formaatti niille, jotka eivät voi saavutettavasta toteutuksesta huolimatta hyödyntää kyseistä erillistekniikkaa. Flash-animaatiot voivat myös olla hyvistä mahdollisuuksista huolimatta esteellisiä samoin kuin HTML-sivut. [75, s. 199, 268-280]

SVG

SVG on modulaarinen (osista koostuva) XML-pohjainen kieli, jolla kuvataan kaksiulotteisia vektori- tai rasterigrafiikasta koostuvia kuvia [22]. SVG:tä voidaan tuottaa monenlaisilla ohjelmilla, kuten Inkscape ja Adobe Illustrator -kuvankäsittelyohjelmilla. SVG:tä voi kirjoittaa myös pelkällä XML- tai tekstieditorilla.

SVG:n hyviin puoliin kuuluu skaalautuvuus. Sama kuva voidaan sovittaa visuaalisesti mihin tahansa kokoon, sillä se on vektorigrafiikkaa. SVG:tä voidaan käsitellä ECMAScript-kielen ja DOM-rajapinnan avulla, jolloin vuorovaikutus käyttäjän kanssa on mahdollista. SVG:n tekstisisältö on helposti saavutettavissa, sillä se ei ole bittikarttagrafiikkaa. [87, s. 112-113]

SVG-standardista on kolme versiota, joista Tiny on tarkoitettu mobiiliselaimiin, Basic PDA-laitteita varten ja SVG 1.1 (Full) muille laitteille, joissa on riittävästi muistia ja suorituskykyä. [8]

SVG ei ole vielä kovin laajalti tuettu. Firefox ja Opera-selaimissa on osittainen tuki SVG-standardille, joka on tulevien versioiden myötä kehittymässä. Internet Explorerille on tarjolla Adoben tarjoama melko kattava SVG-laajennos ja KHTML-pohjaisille selaimille,

kuten Konquerorille ja Safarille, on saatavilla laajennos, joka tukee osittain SVG:tä. Uusissa älypuhelimissa on kuitenkin kohtuullisen hyvä tuki SVG-standardin suppeammille profiileille. Suosituksissa kehotetaankin käyttämään SVG-Tiny-profiilin mukaisia kuvia bittikarttakuvien sijaan [60, s. 20].

4.8.3 Sovelmat

Verkkosivuille voidaan lisätä sovelmien avulla pieniä ohjelmia, kehittyneitä navigointitoimintoja tai dynaamisesti käsiteltäviä sisältöjä [54, s. 257-258]. Internet Explorer -selaimessa voidaan käyttää ActiveX-komponentteja ja useissa selaimissa voidaan suorittaa Java-sovelmia erillisellä laajennoksella. Sivulle on suositeltavaa laittaa linkki, joka vie laajennoksen lataussivulle [75, s. 172].

Verkkosivuilla on syytä käyttää ainoastaan sellaisia tekniikoita, joilla on mahdollista tuottaa avustavilla tekniikoilla saavutettavia sovelmia [75, s. 348]. Sivulle integroidun vuorovaikutteisen objektin pitäisi toimia ilman hiirtä. Java ja ActiveX mahdollistavat näppäimistöohjauksen. Saavutettavien sovelmien tekeminen on mahdollista noudattamalla tekniikan valmistajan antamia ohjeita [54, s. 309]. Java-kielessä on esimerkiksi oma saavutettavuusrajapinta, jonka avulla sovelmista voidaan tehdä esteettömiä [75, s. 309].

4.8.4 Ääni ja video

Videon upottaminen verkkosivuille on tehtävissä `object`-elementillä Flash-animaatioiden tapaan. On suositeltavaa tarjota useita vaihtoehtoisia tapoja siltä varalta, että video ei heti lähde pyörimään. Toisena vaihtoehtona voi olla esimerkiksi kuva ja kolmantena pelkkä teksti. Kaupallisia videoformaatteja ovat Quicktime, RealVideo ja Windows Media, jotka tarjoavat vaihtelevasti saavutettavuusominaisuuksia. [16, s. 300-304]

Kuurot ja heikkokuuloiset tarvitsevat videolle kuvaselostuksen. Kuvaselostuksessa ilmaistaan kaikki sanat, puhujien nimet, puheen sävyt sekä tärkeät äänet ja tapahtumat. Helpoin tapa parantaa heikkokuuloisten saavutettavuutta on tarjota videolle tekstitys, mutta tämä ei tarjoa samanarvoista kokemusta. Kuvaselostusta vastaava tekniikka sokeille on

kuvaileva äänipalvelu, jossa ääniraidan tukena on kertoja, joka kuvailee kaikki tärkeät visuaaliset yksityiskohdat. Tekstitys ei paranna videon saavutettavuutta sokeille, sillä äänen syntesointi muun ääniraidan yhteyteen oikealla ajoituksella ei ole teknisesti mahdollista. [16, s. 38-41, 318-321]

Pelkän äänimateriaalin rinnalle tarvitaan tekstityksiä heikkokuuloisia ja kuuroja varten [5]. Puheäänestä voidaan tehdä saavutettava tekstityksellä. Tekstitys voidaan tarjota erillisellä sivulla, johon on linkki äänitiedoston vieressä. Erillisen tekstityksen voi tarjota myös videon ääniraidalle hakumahdollisuuksien parantamista varten. [16, s. 318, 331].

SMIL

Standarditapa yhdistää tekstiä, ääntä ja videota on käyttää SMIL-kuvauskieltä, joka on XML-sovellus. Kuvauksessa kerrotaan eri medioiden väliset ajoitukset. SMIL mahdollistaa tekstitysten, kuvaselostuksien ja kuvailevan ääniraidan synkronoimisen liikkuvaan kuvaan. Myös kuvia ja tekstiä voidaan yhdistää, jolloin multimediaesitysten tekeminen on mahdollista [75, s. 312-313]. SMIL mahdollistaa myös äänen ja tekstityksen synkronoinnin, mikä parantaa puheen saavutettavuutta erilliseen tekstitykseen verrattuna. SMIL on kohtuullisen hyvin tuettu, sillä QuickTime, RealPlayer ja Internet Explorer tukevat standardia [16, s. 305-306, 331].

4.8.5 Kuvailtu ja kontrolloitava multimedia on saavutettavissa

Yhteenveto

- ✓ Yhteensopivuus *erilaisten selaimien* välillä paranee, kun multimedian upottamisessa käytetään standardeja tapoja.
- ✓ Tarjoa vaihtoehtoinen formaatti niille, jotka *eivät voi käyttää erillistekniikkaa*.
- ✓ Käytä erillistekniikoiden saavutettavuusominaisuuksia näppäimistönavigoinnin toteuttamiseksi. Tästä on hyötyä *ruudunlukijoille* ja *kytkinohjausta* käyttäville.
- ✓ Kontrolloitavasta multimediasta on hyötyä käyttäjille, joilla on *keskittymisvaikeuksia* tai herkkyyttä *epileptisiin kohtauksiin*.
- ✓ SVG skaalautuu hyvin *erilaisten näyttöjen* kokoon ja sopii *mobiililaitteille*.
- ✓ Tekstityksestä ja kuvaselostuksista on hyötyä *kuuroille* ja kuvailevasta ääniraidasta *sokeille*. SMIL mahdollistaa näiden synkronoinnin kuvan kanssa.

4.9 Yhteenveto

Liitteestä 1 löytyvään taulukkoon on koottu luvussa 4 esitetyt saavutettavuuteen vaikuttavat elementit ja ominaisuudet. Taulukossa on merkitty myös ohjeen noudattamisesta hyötyvät ryhmät, jotka on luokiteltu luvun 3.3 mukaisesti ei-visuaalisiin selailutapoihin, poikkeaviin näyttötiloihin, erilaisiin ohjaustapoihin ja muihin rajoitteisiin.

5 SAAVUTETTAVUUDEN AUTOMAATTISEN ARVIOINTIJÄRJESTELMÄN KEHITTÄMINEN

Saavutettavuuden arviointia voidaan tehdä useilla eri tavoilla. Paras tulos saavutetaan yhdistämällä erilaisia tapoja. Näitä tapoja ovat esimerkiksi [19]

- alustava arviointi graafisia selaimia mukauttamalla,
- sivujen arviointi erityisselaimilla, kuten ääni- ja tekstiselaimilla,
- automaattinen testaus validaattoreilla ja arviointityökaluilla,
- yhteisöllinen asiantuntija-arviointi ja
- käyttäjätestaus.

Verkkosivujen tuotantoprosessi on usein kuitenkin nopeaa ja resurssit vähäisiä, jolloin laatunäkökohdat jäävät vähemmälle huomiolle. Tämän takia saavutettavuuden automaattiset arviointityökalut ovat välttämätön osa suunnitteluprosessia ja ylläpitoa. On tärkeää, että ainakin jotain laatuarviointeja pystytään tekemään automaattisesti. [4]

5.1 Automaattinen arviointi

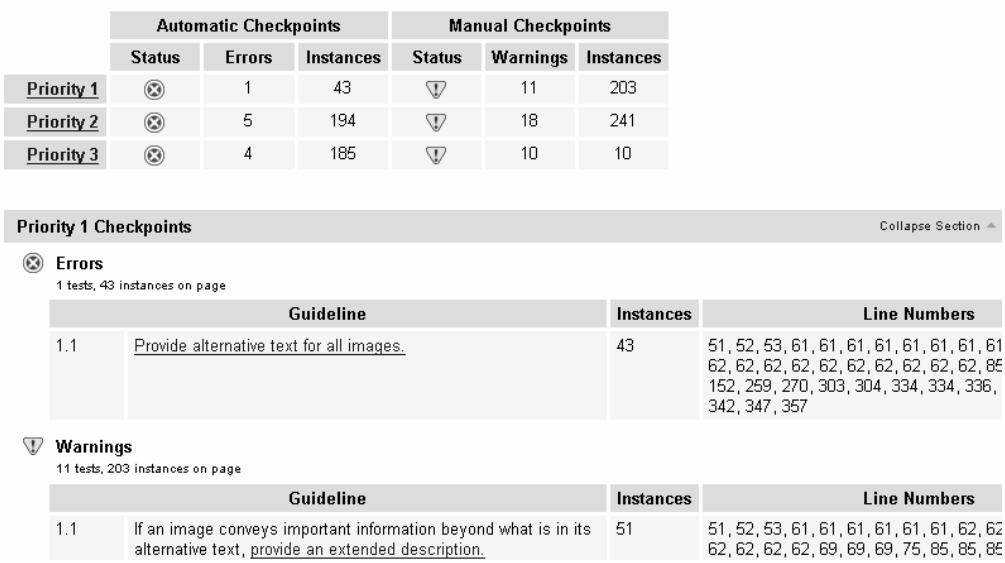
Työkalujen avulla voidaan merkittävästi vähentää aikaa, jota saavutettavuuden arviointi normaalisti vaatisi. Käyttämällä automaattisia arviointityökaluja osana verkkosivujen kehittämisprosessia voidaan välttää esteitä ja parantaa sivuston yleistä laatua. Automaattisten tarkistustyökalujen avulla pystytään suoraan varmistamaan muutamia saavutettavuussuosituksista löytyviä tarkistuskohtia. Arviointityökalujen avulla voidaan myös tehokkaasti opastaa käyttäjää joidenkin kohtien manuaalisessa tarkistuksessa. On kuitenkin huomattava, että näiden lisäksi on vielä useita saavutettavuuskriteerejä, joiden tarkistaminen vaatii asiantuntija-arviota. [19]

5.1.1 Automaattisia arviointityökaluja

Useimmat arviointityökalut ovat verkkopohjaisia. Näille syötetään URL-osoite, josta automaatti hakee sivun ja tarkistaa sen. Myös tiedoston siirto lomakkeiden avulla on usein mahdollista. Tunnetuimpia tarkistustyökaluja ovat WebXACT, Wave ja LIFT.

WebXACT (ent. Bobby)

WebXACT on verkossa toimiva arviointityökalu, joka tarkistaa WCAG 1.0 ja Section 508 -suosituksien mukaisia tarkistuskohtia. Tarkistus perustuu ohjelman lataaman HTML-koodin analysointiin. Tarkistuskohdista tarkistetaan automaattisesti yksiselitteisesti tarkistettavat kohdat. Muista mahdollisista kohdista ohjelma antaa ilmoituksen, jossa kehotetaan käyttäjää tarkistamaan asia itse. Osa näistä ilmoituksista esitetään vain silloin, kun tietty elementti löytyy lähdekoodista. [83]



	Automatic Checkpoints			Manual Checkpoints		
	Status	Errors	Instances	Status	Warnings	Instances
Priority 1	⊗	1	43	⚠	11	203
Priority 2	⊗	5	194	⚠	18	241
Priority 3	⊗	4	185	⚠	10	10

Priority 1 Checkpoints Collapse Section

Errors
1 tests, 43 instances on page

Guideline		Instances	Line Numbers
1.1	Provide alternative text for all images.	43	51, 52, 53, 61, 61, 61, 61, 61, 61, 61, 61 62, 62, 62, 62, 62, 62, 62, 62, 62, 62, 62, 85 152, 259, 270, 303, 304, 334, 334, 336, 342, 347, 357

Warnings
11 tests, 203 instances on page

Guideline		Instances	Line Numbers
1.1	If an image conveys important information beyond what is in its alternative text, provide an extended description.	51	51, 52, 53, 61, 61, 61, 61, 61, 61, 62, 62 62, 62, 62, 62, 69, 69, 69, 75, 85, 85, 85

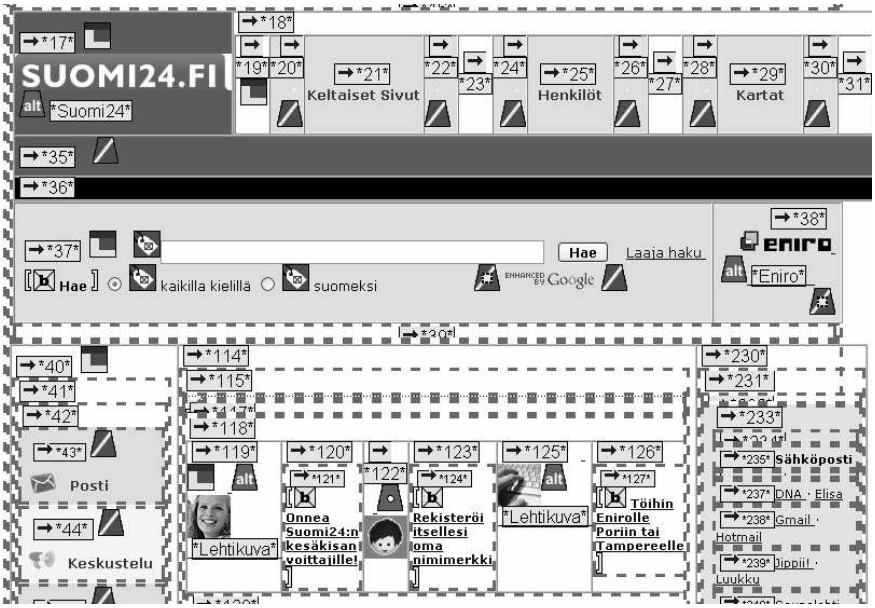
Kuva 18. WebXACT-arviointityökalun raportti `suomi24.fi`-palvelun etusivusta.

Kuvassa (Kuva 18) on WebXACT-työkalun tuottama raportti. Raportin ylälaidassa on koostetaulukko sivun virheistä ja varoituksista. Tämän jälkeen tulee tarkempi lista tarkistuskohdista ja rivinumeroista, joista virhe on löytynyt. Manuaalisesti tarkistettavat kohdat on merkitty varoituksina. Muita huomioitavia erityispiirteitä ovat:

- Työkalu tarkistaa myös linkkien toimivuuden.
- Virhekohtien elementtien nimet ja ominaisuudet saa tarvittaessa näkyviin.
- Ohjelma ei huomioi CSS-ominaisuuksia.

Wave

Wave 3.0 (Web Accessibility Visual Evaluator) on verkkopohjainen arviointityökalu, joka automaattisten tarkistuksien lisäksi helpottaa käyttäjää manuaalisten tarkistuksien tekemisessä. Wave näyttää raportissa informaatiota, jota sokeat selaajat käyttävät hyödykseen, ja merkitsee objektit, jotka vaativat erityistä huomiota esimerkiksi kuurojen tai liikuntavammaisten suhteen. [38]



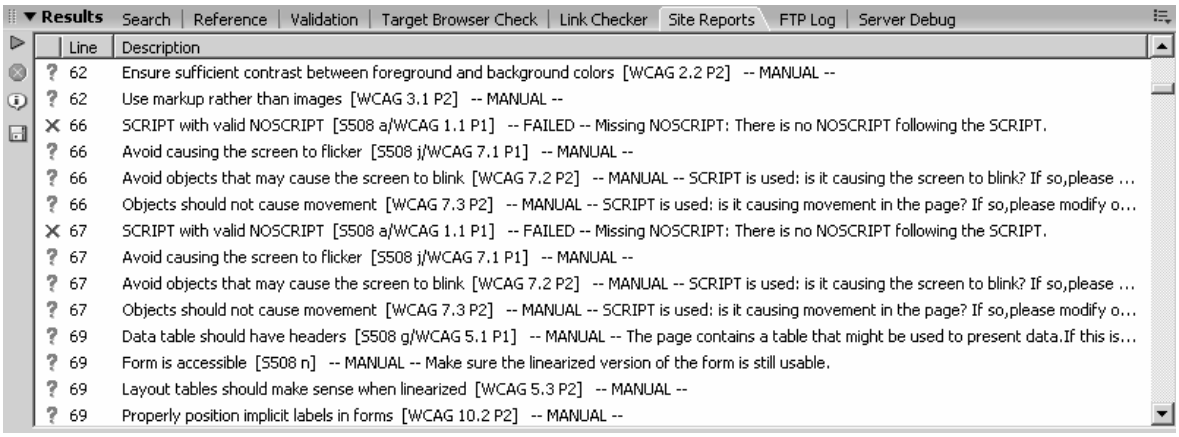
Kuva 19. Wave-tarkistustyökalun raportti `suomi24.fi`-portaalin etusivusta.

Kuvassa (Kuva 19) on Wave-työkalun antama raportti. Lähdekoodista löydetyt lohkot on merkitty katkoviivalla ja erilaiset huomiot sivun saavutettavuudesta on esitetty ikoneilla. Ikonien selitykset löytyvät erillisestä dokumentista. Wave-työkalun erityisominaisuuksia ovat [38]:

- Vaihtoehtotekstit ja valittavat alueet tuodaan raportissa näkyviin.
- Epäilyttävät kuvat (esim. *spacer*-kuvat) merkitään kysymysmerkillä.
- Raportissa esitetään visuaalisesti ruudunlukijan lukujärjestys.
- Rakenteelliset ja ulkoasulliset tekstin muotoiluelementit merkitään ikoneilla.
- Mediaelementtien käyttö merkitään varoituksella. Tällöin on tarkistettava saavutettavan vastineen olemassaolo.

LIFT

LIFT on verkossa toimiva arviointityökalu, joka on myös saatavilla tuotantoväleisiin integroituvana osana. Adoben Dreamweaver-ohjelmaan LIFT on sisäänrakennettuna ja Microsoft Frontpage -ohjelmaan se on saatavilla erillisenä komponenttina.



Kuva 20. LIFT-komponentin antama raportti `suomi24.fi`-etusivusta Dreamweaver-ohjelmassa.

Kuvassa (Kuva 20) on Dreamweaver-ohjelmasta löytyvän *Check Accessibility* -toiminnon antaman raportin osa. LIFT-komponentti osaa tarkistaa osan saavuttavuuden tarkistuskohdista automaattisesti, kun taas osa vaatii manuaalista tarkistamista. Raportin virhekohdasta pääsee muutamalla napautuksella oikeaan kohtaan lähdekoodia. LIFT-työkalun erityisiä ominaisuuksia ovat [80]:

- Ohjelma tarkistaa muutamia CSS:n saavutettavuusominaisuuksia, esimerkiksi absoluuttisten yksiköiden ja liian pienen kirjasimen käyttämisen.
- Tunnistaa erityyppisiä kuvia ja niiden vaatimien `alt`-tekstien muotoja.
- Arvioi käytetäänkö taulukkoa datan esittämiseen, sivun taittoon, lomake-elementtien asetteluun tai navigointilistan muodostamiseen.
- Huomioi JavaScript-linkit ja uusien ikkunoiden avaamisen.

5.2 Arviointijärjestelmän kehittäminen

Automaattiset arviointityökalut perustuvat perinteisesti HTML-koodin analysointiin. Verkkosivujen saavutettavuutta voitaisiin kuitenkin arvioida yksiselitteisiä tarkistuksia

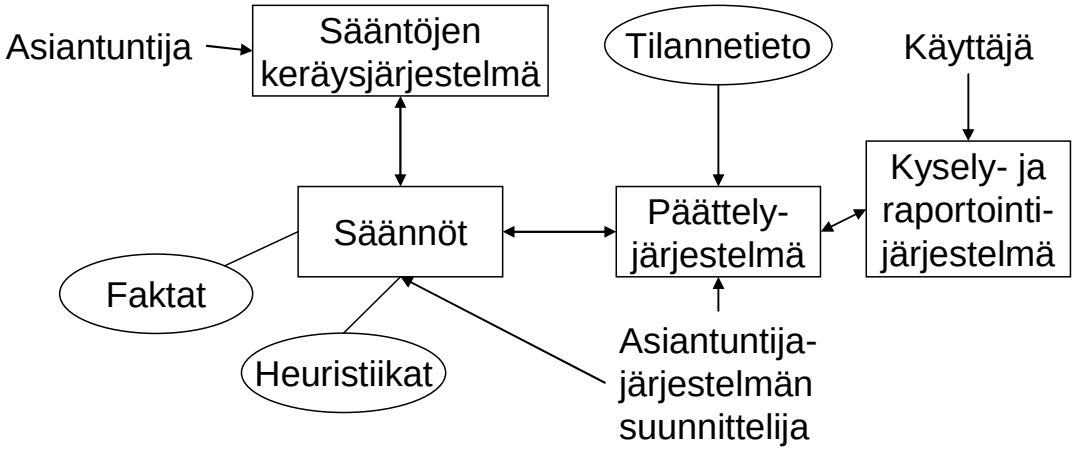
kattavammin kehittämällä järjestelmä, joka kykenee erottelemaan sivun osien merkityksiä. Arviointijärjestelmän pitäisi olla myös helposti laajennettavissa uusilla saavuttavuuskriteereillä.

5.2.1 Asiantuntijajärjestelmät

Asiantuntijajärjestelmät mahdollistavat sen, ettei saavutettavuuden arvioijan tarvitse itse olla alan ekspertti. Monipuolisimmat saavutettavuuden arviointityökalut voidaan rinnastaa asiantuntijajärjestelmiin, joissa tietokoneohjelma kykenee asiantuntijatasoiseen päättelyyn. Asiantuntijajärjestelmiä on hyödynnetty onnistuneesti lääketieteen, tekniikan ja liiketalouden tietyissä kapeissa, mutta vaativissa tehtävissä. [58, s. 280-281]

Asiantuntijajärjestelmä koostuu tyypillisesti päättelyjärjestelmästä ja säännöistä, joita asiantuntija syöttää järjestelmälle. Asiantuntijan tietämys ilmaistaan yleensä predikaattilogiikan keinoin, kun tavoitteena on saavuttaa annetuihin tietoihin perustuva päättelyketju. Järjestelmän ydin sisältää prosessit, joiden avulla tietämyskannasta voidaan johtaa vastaus käyttäjän pyytämään kysymykseen. Asiantuntijajärjestelmän tietämys perustuu isoon määrään sääntöjä, jotka on muokattu tietokoneen ymmärtämään muotoon. Säännöt voivat olla esimerkiksi loogisia lausekkeita, joista saataviin tuloksiin on lisätty todennäköisyysarvioita. [58, s. 281-285]

Saavutettavuuden automaattisessa arvioinnissa voidaan hyödyntää samankaltaista rakennetta. Arviointijärjestelmän tehtävä on vastata inhimillisistä päättelyä ja tuottaa yhtä luotettavia tuloksia kuin mihin ihmisarvioija kykenee. Saavutettavuuden arviointia voidaan tehdä käyttäjätestauksen sijaan myös arvioimalla sivun osia heuristisilla säännöillä muutamien tarkkojen sääntöjen lisäksi. Tätä tietämystä voidaan siirtää tietokoneohjelman käsiteltäväksi.

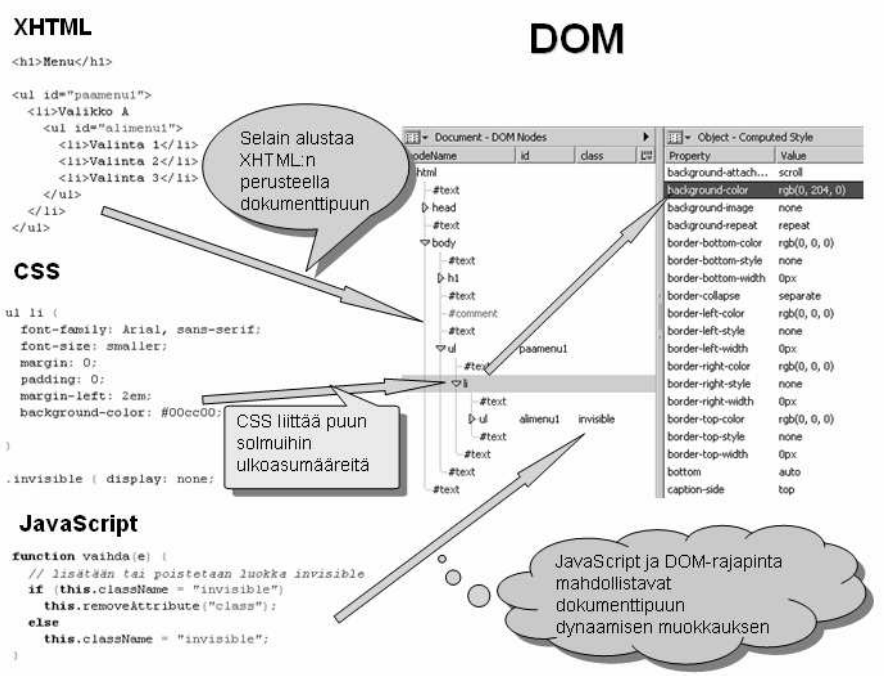


Kuva 21. Asiantuntijajärjestelmän perusrakenne.

Kuvassa (Kuva 21) on esitetty asiantuntijajärjestelmän rakenne. Saavutettavuuden arvioinnin näkökulmasta sääntöjä ovat erilaiset saavutettavuusohjeistukset ja hyväksi koetut toimintatavat. Esimerkiksi vaatimus kuvien vaihtoehtoteksteistä on selvä fakta. Asiantuntijan kuvailema heuristinen sääntö voi liittyä esimerkiksi sivun osien asemointiin. Käyttäjän antama syöte on verkkosivu, josta halutaan tietää, onko sivu saavutettava ja miten sivun saavutettavuutta voidaan parantaa. Päättelyjärjestelmä on ohjelma, joka esiprosessoi käyttäjän antamaa tietoa ja vertailee sitä suhteessa kerättyihin sääntöihin.

5.2.2 Dynaamisten verkkosivujen ongelma

Verkkosivun elementteihin voi liittyä erilaisia CSS-ominaisuuksia ja näitä ominaisuuksia voidaan muokata dynaamisesti DOM-rajapinnan ja JavaScript-komentosarjojen avulla. Tämä asettaa uudenlaisia vaatimuksia arviointijärjestelmille.

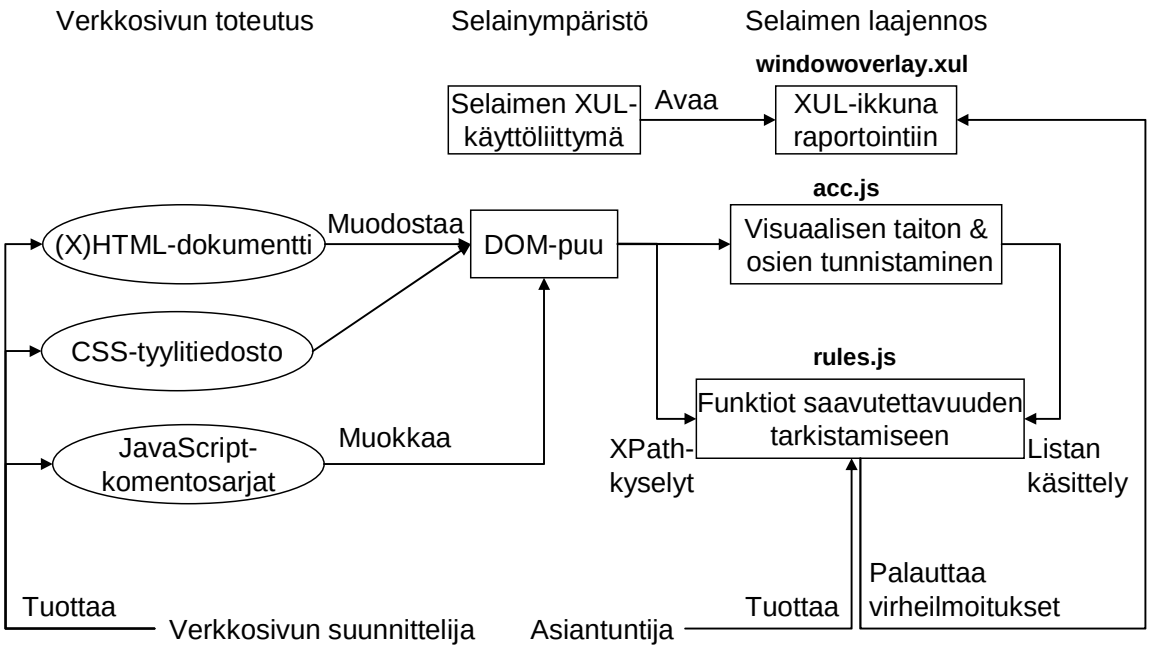


Kuva 22. Dynaamisen verkkosivun toiminta selaimessa.

Kuvassa (Kuva 22) on kuvio, joka havainnollistaa miten selain prosessoi dynaamisen verkkosivun osia. Kuvan oikeassa laidassa on näkymä Mozilla Firefox -selaimen DOM Inspector -työkalusta. DOM on laitteisto- ja kieliriippumaton rajapinta, jonka avulla ohjelmat voivat lukea ja muokata sivun sisältöä, rakennetta ja tyyliominaisuuksia [45]. Erilaiset tapahtumat, kuten painikkeen ja linkin valitseminen, voivat laukaista komentosarjan, joka muokkaa sivun rakennetta. Tällöin sivun saavutettavuus on arvioitava uudelleen. Pelkkää lähdekoodia analysoimalla on vaikeaa tai jopa mahdotonta arvioida dynaamisen sivun saavutettavuutta.

5.2.3 Selaimeen integroituva arviointijärjestelmä

Ratkaisu erilaisiin ongelmiin on toteuttaa arviointijärjestelmä selaimen laajennoksena. Tällöin DOM-rajapinnan kautta on saatavilla elementtien ulkoasutiedot, tapahtuneet dynaamiset muutokset ja sivun taittoon liittyvä semanttinen informaatio implisiittisessä muodossa.



Kuva 23. Selaimen laajennoksena toteuttavan arviointijärjestelmän rakenne.

Kaaviossa (Kuva 23) on esitetty Mozilla Firefox -selaimeen laajennoksen integroituvan arviointijärjestelmän rakenne. Verkkosivun tekijä tuottaa HTML- tai XHTML-dokumentteja ja näihin liittyviä CSS- ja JavaScript-tiedostoja. Nämä muodostava selaimen muistiin puuhierarkian, jonka solmujen ominaisuuksiin päästään käsiksi DOM-rajapinnan metodien avulla.

Arviointijärjestelmä koostuu kolmesta pääosasta. Ensimmäisen tarkoitus on esiprosessoida DOM-puusta tarvittava informaatio. Toisen tehtävä on arvioida erilaisista lähteistä saatavaa tietoa saavutettavuuden kannalta. Kolmannen osan tavoite on esittää selkeällä tavalla virheet graafisessa käyttöliittymässä.

Tämän tutkielman osana on toteutettu *Acc - an Accessibility Evaluator* -laajennos Mozilla Firefox -selaimeen [53]. Ohjelma toteuttaa kuvatun kaltaisen arviointijärjestelmän rungon ja useita luvussa 5.3 esitettyjä tarkistuksia.

5.2.4 Sivun osien tunnistaminen

Elementtien koko- ja paikkatiedot ovat saatavissa JavaScript-objekteista tai lasketuista CSS-ominaisuuksista. Tätä informaatiota käytetään visuaalisen taiton selvittämisessä. Sivun osien asemointitiedot ovat tärkeitä, sillä monet saavutettavuusongelmat liittyvät juuri erilaisten alueiden, kuten navigointien, esteelliseen asetteluun. Kustakin osasta pyritään selvittämään heuristisilla arvioilla sen tyyppi (esim. sisältö ja navigointi) ja näistä tiedoista muodostetaan lista, jota voidaan käydä tarvittaessa useaan kertaan läpi.

Sivun osien tunnistamisessa esiprosessoidaan DOM-puusta saatavia ulkoasutietoja ja muodostetaan muistiin toinen puuhierarkia, joka kuvastaa DOM-puuta suoremmin niitä merkityksiä, joita verkkosivun tekijä on halunnut sivun taitolla implisiittisesti esittää. DOM-puuhierarkiasta saadut lohkot eivät suoraan riitä osien tunnistamisessa, sillä elementit voivat olla toisensa sisällä. Lisäksi yhtenäinen sivun osa voi koostua useasta DOM-puun lohkosta.

Osien erottelu tehdään seuraavissa vaiheissa:

1. Käydään läpi koko DOM-puu lineaarisessa järjestyksessä eli kunkin elementin tarkastelun jälkeen tutkitaan ensin, onko elementillä lapsielementtejä ja vasta sitten onko elementillä sisarelementtejä. Jos elementillä on lapsielementtejä, niin käydään nämä ensin läpi.
   - Kustakin tekstiä, kuvia, lomakkeita tai multimediaa sisältävistä DOM-puun osasta tutkitaan elementin rajaaman laatikon koko ja paikka, ja lisätään tieto listaan.
   - Lohkoista, jotka sisältävät yleensä muita elementtejä, tallennetaan reunaviivat merkiksi mahdollisista osien erottimesta. Tällaisia lohkoja ovat esimerkiksi `td`- tai `div`-elementit.
2. Luodaan uusi X-Y-leikkauspuu, jonka juurielementiksi laitetaan koko dokumentin sisältävä elementti. Pilkotaan puun lehtisolmuissa olevat alueet vuorotellen vaaka- ja pystysuuntaan. Aluksi lehtisolmuna on vain juurisolmu, mutta jokainen leikkaus saa aikaan uusia lehtisolmuja.

- Ennen leikkauksia selvitetään rajalaatikkolistan perusteella, onko jossain välissä eksplisiittisten erottimien lisäksi riittävän suuri tyhjä väli. Tällöin väli lisätään leikkauskohdaksi.
- Lopetetaan pilkkominen, kun tarvittava leikkaustarkkuus on saavutettu. Tarkkuus on riittävä, kun isot alueet on saatu näkyviin. Hyväksyttävälle alueelle annetaan minimikoko, jonka lähelle päästään, kun leikkauksia tehdään tarpeeksi.

3. Tutkitaan näiden isojen alueiden sisällä olevia elementtejä ja luokitellaan ne heuristisilla arvioilla joko sisältöosaksi, navigaatio-osaksi, yleiseksi alueeksi tai tyhjäksi lohkoksi. Arvio perustuu tekstin, linkkien ja lomake-elementtien määrän suhteisiin.

Eksplisiittisten erottimien selvittämistä sivun osien jakamisessa on hyödynnetty verkkosivujen mukauttamisessa pieninäyttöisille laitteille [10]. Erottimien selvittämisen pääperiaate esitettiin kohdassa 1 ja toteutus löytyy liitteen 2 funktioista `acc_getBoundingBoxes` ja `acc_getCuts`.

Rekursiivinen X-Y-leikkausalgoritmi on peräisin dokumenttien optisen tunnistuksen menetelmistä. Leikkausalgoritmin eräässä variaatiossa hyödynnetään rajalaatikoita pikselinpohjaisen leikkauksen sijaan [30]. Toisessa tutkimuksessa on esitetty Mozilla-selaimen DOM-rajapinnasta saatujen rajalaatikoiden ja X-Y-leikkausalgoritmin hyödyntämistä datataulukoiden erottamisessa [41]. X-Y-leikkauksien pääperiaate esitettiin kohdassa 2 ja toteutus löytyy liitteen 2 funktiosta `acc_divideXYNode`.

Kohdan 3 mukainen osien tunnistus on toteutettu liitteen 2 funktiossa `acc_findXYNodeTypes`. Osien erottelumenetelmä on toteutettu kokonaisuudessaan liitteen 2 funktiossa `acc_extractLayoutMetadata`.

Kuva 24. Osien automaattinen tunnistus toimii `suomi24.fi`- ja `bbc.co.uk`-sivustojen etusivuilla.

Kuvassa (Kuva 24) on esitetty menetelmän tekemät osiojaot viihdeportaalissa ja uutissivustolla. Osiojako on mahdollista tuoda näkyviin laajennoksen *View Layout* -painiketta napauttamalla. Tämä lisää dokumenttiin väliaikaisesti läpinäkyviä lohkoja, joissa on reunaviiva ja osan tyyppi merkittynä vasempaan yläkulmaan.

Kuvista voidaan huomata, että tärkeimpien aluiden tunnistus onnistuu monimutkaisillakin sivuilla. Muutamissa osissa tarkkaa tyyppiä ei löydetä, mutta tämä on toistaiseksi heuristisen arvioinnin ongelma. Kapeat alueet saattavat joskus mennä halki, mutta ilman tätä mahdollisuutta X-Y-leikkausmenetelmä ei toimi kaikkein tiiviimmissä asetteluissa.

5.2.5 Tarkistukset ja raportointi

Sivun taittoon liittyvää metadataa, DOM-puun ominaisuuksia ja puuhun kohdistuvia XPath-kyselyjä hyödyntämällä voidaan toteuttaa JavaScript-komentosarjoja, joiden avulla kyetään monipuolisesti selvittämään, onko sivu saavutettava. Sääntöjen kirjoittaminen vaatii hieman ohjelmointitaitoa. Sääntöjä varten voidaan tarvittaessa toteuttaa oma kuvauskieli, mutta tämä on ehkä tarpeetonta, sillä useat tarkistukset vaativat monimutkaisia ehdollisia sääntöjä. Tarkistus voidaan toteuttaa suoraan JavaScript-funktioilla, joita pääohjelma kutsuu vuorotellen. Funktioita kutsuva pääohjelma on toteutettu liitteen 2 funktioon `acc_run_tests`. Kutsuttavat arviointifunktiot löytyvät liitteestä 3. Pääohjelma suoritetaan liitteessä 4 kuvaillun raportointi-ikkunan *Run Check* -painikkeen `oncommand`-tapahtumasta.

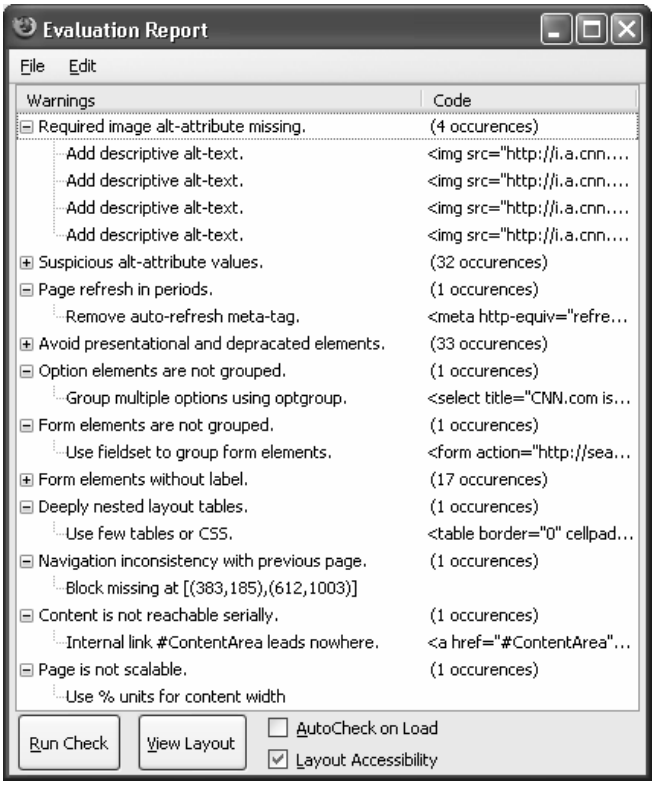
JavaScriptin avulla voidaan myös kontrolloida selainikkunan tilaa. Tämä mahdollistaa olosuhteiden muutokset ja uudelleenarvioinnin vaihtuneessa ympäristössä. Esimerkiksi selaimen käyttämää tyylitiedostoa voidaan vaihtaa ja selainikkunan kokoa pienentää.

Mozilla Firefox -selaimessa on myös tuki DOM Level 3 XPath -standardille. XPath mahdollistaa JavaScript-komentosarjoissa useiden DOM-solmujen helpon osoittamisen yksittäisellä merkkijonolla [28]. Esimerkiksi kaikki kuvasolmut, joissa ei ole `alt`-tekstiä saadaan komennolla:

```
var i = document.evaluate("//img[not(@alt)]", opener.content.document,
null, XPathResult.ORDERED_NODE_ITERATOR_TYPE, null);
```

Paluutyypin voi määrätä `evaluate`-funktion parametrina. Se voi olla myös luku tai totuusarvo. Tässä tapauksessa saamme olion, jonka avulla tulosjoukko voidaan käydä järjestyksessä läpi [28]. XPath mahdollistaa avatun sivun DOM-puun solmujen helpon tarkastelun. Virheellisistä solmuista on helppo ottaa pieni pala lähdekoodia raporttiin `innerHTML`-funktiolla, vaikka tämä ei olekaan DOM-standardin mukaista.

Arviointifunktiot palauttavat virheilmoituksen ja lähdekoodin palasen funktiolle, joka on kutsunut näitä. Funktiossa lisätään tiedot XUL-käyttöliittymän tarjoamaan puunäkymään. Laajennoksen käyttöliittymän kuvaileva dokumentti löytyy liitteestä 4.



Kuva 25. Arviointiohjelman antama raportti `cnn.com`-etusivusta.

Kuvassa (Kuva 25) on esitetty, miltä laajennoksen käyttöliittymä näyttää. Samaan ryhmään kuuluvat virheet niputaan pääkohtien alle. Pääkohdan alla olevat korjausehdotukset saa näkyviin napauttamalla virhetekstin edessä olevaa plus-painiketta. Valitsemalla haluttu alakohta ja valikosta toiminto *Find Code*, päästään tarkastelemaan DOM-puusta luotua lähdekoodia. Virheellinen osa näytetään lähdekoodista korostuksella. Kaikista virheistä voidaan myös luoda XHTML-muotoinen raportti.

5.3 Arviointimenetelmiä

Seuraavissa luvuissa kuvaillaan, miten erilaisia saavutettavuustarkistuksia voidaan toteuttaa esitetyssä arviointijärjestelmässä.

5.3.1 Suoraan XPath-kyselyillä tehtäviä tarkistuksia

Useissa tutkimuksissa on esitetty WCAG-suosituksessa kuvailtuja tarkistuskohtia formaaleissa muodoissa. Centeno *et al.* on useissa artikkeleissa esittänyt, miten WCAG-

suosituksen objektiivisesti tarkistettavat kohdat voidaan selvittää XPath-lausekkeiden avulla [9]. Näitä formalisointeja voidaan hyödyntää suoraan saavutettavuuden arvioinnissa.

Alt- ja title-ominaisuuksien arvot

XPath-lausekkeella voidaan vuorotellen hakea

- `img`-elementit, joissa ei ole `alt`-attribuuttia,
- kuvapainikkeet, joissa ei ole `alt`-tekstiä,
- `object`-elementit, joiden sisällä ei ole tekstivaihtoehtoa, ja
- `area`-elementit, joilla ei ole vaihtoehtotekstiä.

Tarvittaessa voidaan myös selvittää `title`-ominaisuuden olemassaolo. Vastaavasti voidaan hakea myös elementit, joilla nämä ominaisuudet ovat ja tutkia niistä DOM-rajapinnan tarjoamilla metodeilla,

- onko `alt`-teksti liian pitkä (jolloin ehdotetaan `longdesc`-ominaisuuden käyttöä) tai
- onko kyseessä kapea *spacer*-kuva, jolla pitäisi olla ainoastaan tyhjä `alt`-teksti.

`alt`-tekstien järkevyyden arviointi ei näillä menetelmillä onnistu kattavasti. Muutamista yleisimmistä virheellisistä teksteistä voidaan kuitenkin kerätä lista. Näin tekee esimerkiksi Wave-arviointityökalu. Näkyviä `alt`-tekstejä voidaan selainympäristössä visualisoida vaihtamalla ne näkyvien kuvien tilalle. Tämä onnistuu helposti muokkaamalla dokumenttipuuta.

Elementtien ominaisuuksien arvoja tarkastelevia tarkistusfunktiota ovat esimerkiksi liitteen 3 funktiot `acc_imgWithoutAlt`, `acc_imgWithAltText` ja `acc_inputImageWithoutAlt`.

Sivun ominaisuuksia

Vastaavalla tavalla sivuun ja sivun taittoon liittyvistä ominaisuuksista voidaan selvittää

- päivittyykö sivu säännöllisesti `meta`-elementin avulla,
- onko `frameset`- ja `frame`-elementtejä käytetty ja onko niillä `title`-tekstit,

- onko dokumentin pääkieli merkitty ja
- onko sivun `head`-osan `title`-elementissä tekstiä.

Tällaisia tarkistuksia on toteutettu liitteen 3 funktioihin `acc_metaRefresh`, `acc_htmlLang` ja `acc_pageTitle`.

JavaScript-linkit, tapahtumat ja uudet avautuvat ikkunat

Linkkeihin kytketyistä ominaisuuksista voidaan helposti löytyy

- sellaiset linkit, jotka alkavat `javascript`-määreellä,
- linkit, joiden tapahtuman herättimenä toimii ainoastaan hiiri ja
- linkit, joiden `target`-ominaisuuden arvo on `_blank`.

Viimeinen kohta on toteutettu liitteen 3 funktiossa `acc_openNewWindow`.

Vältettävistä elementeistä varoittaminen

Vältettävistä, epästandardeista ja epäsemanttisista elementeistä voidaan kerätä lista ja suorittaa kutakin elementtiä kohden XPath-kysely. Jos elementtejä löytyy, niin käydään ne läpi ja annetaan kustakin kohdasta virheilmoitus yhden pääkohdan alle. Vältettävien elementtien haku on toteutettu liitteen 3 funktiossa `acc_depracatedElements`.

5.3.2 DOM-rajapintaa hyödyntävät tarkistukset

Seuraavissa tarkistuksissa käytetään DOM-rajapinnan tarjoamia elementtihierarkian läpikäyntimetodeja ja rajapinnan kautta saatavia ulkoasuatietoja.

Kuvan longdesc-ominaisuuden järkevyys

`longdesc`-ominaisuudesta voidaan tarkistaa, että kyseessä on todellakin viittaus olemassaolevaan HTML-tiedostoon kokeilemalla osoitetta joko `XMLHttpRequest`-objektilla tai avaamalla dokumentti selaimessa uuteen välilehteen. Jos vaste saadaan, niin dokumentista voidaan erottaa DOM-rajapinnan avulla tekstisisältö ja tutkia sen pituutta. Jos teksti on riittävän pitkä, niin tällöin kyseessä on järkevästi määritelty ominaisuus.

Tekstin saavutettavuus

Kunkin tekstielementin CSS-arvoista tai suoraan tyylisivuobjektin määrityksistä voidaan tarkistaa, että yleinen kirjasinperhe on määritelty `font-family`-ominaisuuteen. Samalla voidaan tarkistaa, onko käytössä suhteellinen vai absoluuttinen fontin koko. Tekstin ymmärrettävyyttä voitaisiin arvioida laskemalla sanoista sopiva luettavuusindeksi [39]. Jos luettavuusindeksi ylittää tietyn varman ylärajan, niin voidaan huomauttaa vaikeasti luettavasta tekstistä.

Värikontrastien arviointi

Tekstin ja taustan värin riittävää kontrastia voidaan arvioida algoritmilla, jota W3C:n Accessibility Evaluation and Repair Tools -dokumentti ehdottaa. Tällainen arviointityökalu on toteutettu Mozilla Firefox-selaimen laajennoksena. Colour Contrast Analyzer -menetelmä toimii seuraavasti [43]:

1. Haetaan kaikki lohkot, joissa on tekstiä.
2. Selvitetään taustan ja tekstin RGB-väriarvot DOM-rajapinnan `getComputedStyle`-metodin avulla.
3. Vertaillaan kunkin lohkon taustan ja tekstin kirkkauseroa ja värien erottuvuutta annetuilla algoritmeilla.

Menetelmä on mahdollista toteuttaa myös esitetyssä arviointiympäristössä, koska rajalaatikko-listasta saadaan suoraan DOM-puun solmut, joissa on tekstiä. Virheestä voidaan ilmoittaa myös rajaamalla ongelmallinen kohta visuaalisesti selaimessa.

Jos tekstin taustalla on kuva, niin taustakuvan voidaan siirtää palvelimelle ja arvioida siellä taustan keskimääräiset kirkkaus- ja väriarvot. Jos lohkon on iso, niin arviointia voidaan tehdä myös paloittain.

Lomake-elementtien saavutettavuus

Lomake-elementtien kytkennän tarkistus `label`-elementteihin onnistuu DOM-rajapintaa käyttäen. Jos lomake-elementillä ei ole vanhempielementtinä tai edeltäjänä `label`-

elementtiä, niin tutkitaan löytyykö elementin `id`-ominaisuuden arvoa `label`-elementtien `for`-ominaisuudesta. Tällaiset `label`-elementit saa helpoiten yhdellä XPath-kyselyllä.

`select`-elementillä toteutettujen valintalistojen valintavaihtoehtojen ryhmitellyn tarkistus onnistuu niin ikään XPath-lausekkeen avulla. Valintalistoista voidaan ottaa tarkasteluun myös pelkästään sellaiset elementit, joissa on liikaa irrallisia `option`-elementtejä. Samaan tapaan voidaan tarkistaa, että `form`-elementin sisällä ei ole liikaa irrallisia lomake-elementtejä vaan, että lomake-elementit on ryhmitelty `fieldset`-elementeillä. Lomakeryhmän `legend`-otsikon olemassaolo voidaan myös tarkistaa.

Kytkentöjen ja ryhmittelyjen tarkistukset on toteutettu liitteen 3 funktioissa `acc_connectedLabels`, `acc_optionGroups`, `acc_fieldsetGroups` ja `acc_fieldsetLegend`.

Esteellisen taulukkotaiton todentaminen

Liian useat sisäkkäiset taulukot tuottavat ongelmia ruudunlukijoille. Taulukkoja voidaan käyttää sekä datataulukkoina että sivun taitossa, jolloin saavutettavuuden arviointi ei ole triviaali asia. Seuraavalla heuristiikalla voi kuitenkin tunnistaa aidon datataulukon [64]:

- taulukko on lehtisolmu,
- taulukossa on useita rivejä ja sarakkeita,
- solujen tekstisisältö on lyhyt,
- soluista löytyy enintään yksi tekstin muotoiluun liittymätön elementti ja se ei sisällä mitään rakenteellisia elementtejä.

Seuraavalla menetelmällä voidaan todeta, jos sisäkkäisiä taittotaulukoita on liikaa:

1. Otetaan DOM-puusta kaikki `table`-elementit. Karsitaan näistä tarkasteluun ainoastaan kolmannen tason taulukot.

2. Arvioidaan onko kyseessä datataulukko vai taittoon liittyvä taulukko.

    - Jos taulukon sisällä on neljännen tason taulukko, niin silloin taulukkotaitto on kyseistä kohdasta liian syvä.

- Jos kolmannen tason taulukossa on `th`-elementtejä, niin kyseessä on datataulukko.
- Jos soluja on vähemmän kuin neljä (ja ne ovat samalla rivillä tai sarakkeella), niin kyseessä on taittoon liittyvä taulukko.
- Jos solujen tekstin pituus on liian suuri ja niistä löytyy riittävästi rakenteellisia elementtejä, niin kyseessä on taittoon liittyvä taulukko.

Menetelmä on toteutettu liitteen 3 funktiossa `acc_nestedTables`.

Taulukon esteettömyysominaisuuksien hyödyntäminen

Kukin datataulukon `td`-elementti pitää kytkeä ainakin yhteen `th`-soluun `scope`- tai `headers`-ominaisuudella. Tämän voi tarkistaa DOM-rajapinnan tarjoamilla metodeilla seuraavasti:

1. Haetaan kaikki `table`-elementit, joiden sisällä ei ole toisia taulukoita.
2. Tutkitaan ylläolevilla heuristiikoilla, onko kyseessä datataulukko.
3. Jos datataulukosta puuttuvat otsikkosolut, niin taulukossa on virhe.
4. Jos `tr`-elementin sisällä on `th`-solu, jonka `scope`-attribuutin arvo on `row`, niin tämän rivin voi jättää pois tarkasteluista.
5. Tarkastellaan kutakin `td`-solua.
   - Jos `td`-elementillä on `headers`-ominaisuudessa arvo, joka viittaa `th`-elementin `id`-ominaisuuteen, niin kytkentä on kunnossa. Tarkistuksen voi tehdä esimerkiksi `getElementById`-metodilla.
   - Jos suoraa kytkentää ei ole, niin on tarkasteltava löytyykö taulukon muilta riveiltä samasta indeksin arvosta `th`-elementti, jonka `scope`-attribuutin arvo on `col`. Muutoin taulukko on esteellinen.

Datataulukosta voidaan vielä tarkistaa, että se sisältää tarvittavan `caption`-elementin ja että `table`-elementissä on `summary`-attribuutti järkevän pituisella tekstillä.

Sivun skaalautuvuus pienille näyttöresoluutioille

1. Pienennetään selainikkunan kokoa.

2. Tarkastetaan, onko sisältöä mennyt oikeasta laidasta yli.

Mozilla Firefox tarjoaa kohdan 2 tietoon suoran ominaisuuden. Tarkistuksen toteutus löytyy liitteen 3 funktiosta `acc_commonLayoutScalability`. Sivuston skaalautuvuutta voitaisiin tarkastella vielä tarkemmin suhteessa sisältöosien koon muutoksiin, mutta tällöin joudutaan selvittämään uudelleen sivun osien koot ja paikat.

5.3.3 Sivun osajakoa hyödyntävät tarkistusfunktiot

Seuraavat tarkistukset hyödyntävät tarkistuksen alussa selvitettyä X-Y-leikkauspuuta, joka tarjoaa tietoa sivun visuaalisesta asettelusta.

Navigoinnin yhtenäisyys

1. Kun siirrytään sivulta toiselle, niin pidetään edellinen leikkauspuu tallessa.

2. Jos edellinen sivu oli samalla palvelimella, niin tarkistetaan löytyykö isoja navigointilohkoja samasta paikkaa kuin edellisellä sivulla.

    - Tutkinta voidaan tehdä siten, että kokeillaan sopivatko edellisen sivun isot navigointilohkot jonkin nykyisen sivun navigointilaatikon sisälle. Nykyisen sivun navigointilohkojen kokoa voidaan kasvattaa hieman.
    - On huomioitava myös se, että sivun alalaidan pikselikoordinaatit voivat erota edellisestä sivusta, jos kyseessä on esimerkiksi edellistä pidempi sivu. Tällöin tarkistukset pitää tehdä myös käänteisillä `y`-koordinaateilla.

Toteutus löytyy liitteen 3 funktiosta `acc_navigationConsistency`.

Navigoinnin ohittaminen

1. Otetaan muutamia ensimmäisiä elementtejä DOM-puusta lineaarisessa järjestyksessä. Jos näiden sisältämät laatikot ovat ison sisältöosan sisällä, niin tällöin sisältö on suoraan saavutettavissa, eikä hyppylinkkejä tarvita.

2. Muutoin tutkitaan alussa olevia linkkejä. Jos näistä löytyy sisäinen linkki, niin tutkitaan, mihin elementtiin se viittaa.

3. Tarkastellaan, onko tämän elementin rajalaatikko ison sisältöalueen sisällä. Jos on, niin hyppylinkki toimii oikein.

Toteutus löytyy liitteen 3 funktiosta `acc_skipToContentLink`.

Sivun taiton mukautuminen mobiililaitteille ja ruudunsuurentajille

Pientä näyttöä tai isoa suurennosta varten tarvitaan yksipalstaista taittoa. Tämän voisi todentaa mobiilitilassa seuraavasti:

1. Muutetaan selaimen dokumenttinäkymän koko älypuhelimen resoluutiota vastaavaksi.
2. Kytketään päälle ainoastaan `handheld`- ja `all`-medioille sopivat tyylitiedostot.
3. Tehdään sivun osien paikannus uudelleen.
4. Jos nyt jokin iso osa vie vähemmän kuin puolet ikkunan leveydestä, niin sivun taitto ei mukaudu hyvin mobiiliselaimia varten.

5.3.4 Toteutetut tarkistukset

Yllä olevista arviointimenetelmistä on toteutettu työkaluun kaikki paitsi

- kuvan `longdesc`-ominaisuuden järkevyyden tarkistus,
- värikontrastien arviointi,
- taulukon esteettömyysominaisuuksien hyödyntäminen ja
- sivun taiton mukautumisen arviointi mobiiliympäristössä.

Näiden tarkistuksien kokeiluimplementaatioita ei ehditty tarpeeksi testaamaan, joten ne jätettiin tutkielmasta pois. Menetelmät ovat kuitenkin täysin toteuttamiskelpoisia. Lisäksi luvussa 4 esitetyistä saavutettavuuskriteereistä voidaan helposti tarkistaa useita kohtia, kuten esimerkiksi otsikkotasojen ja sarkainjärjestyksen järkevyys.

6 POHDINTA

6.1 Arviointityökalujen tärkeimmät ominaisuudet

Arviointityökalun tärkeimpiä ominaisuuksia ovat aukottomuus, virheettömyys ja tarkkuus. Aukottomuudella tarkoitetaan sitä, kuinka hyvin työkalu löytää saavutettavuussuosituksien vastaiset virheet. Tärkeää on se, kuinka automatisoidusti virheitä pystytään havaitsemaan. Virheettömyys merkitsee sitä, ettei työkalu raportoi väärin virheistä eikä tarpeettomasti mahdollisista virheistä. Tarkkuus kuvastaa sitä, kuinka ymmärrettävästi ohjelma raportoi virheistä. [4]

Useissa tutkimuksissa on osoitettu, että saatavilla olevat automaattisen arvioinnin menetelmät ovat puutteellisia, ohjaavat väärällä tavalla verkkosivujen suunnittelijoita arvioimaan sivun saavutettavuutta eivätkä vastaa käyttäjien todellisiin ongelmiin. Automaattitestien, kuten Bobby tai LIFT, hyväksymissä verkkosivuissa voi olla vielä hyvinkin suuria saavutettavuusongelmia esimerkiksi ruudunlukijoita käyttäville sokeille verkon selaajille. [77]

Erilaisista tarkistuskohdista arviointityökalut pystyvät tyypillisesti arvioimaan automaattisesti vain noin 30-50% [4]. Kaupallisesta toteutuksesta huolimatta esimerkiksi LIFT-työkalu kehottaa käyttäjää tekemään manuaalisesti seuraavat kohdat:

- Taustan ja tekstin kontrastin arviointi.
- Datataulukoiden tietosolujen kytkennät otsikkosoluihin.
- Taulukkotaitetun sisällön järkevyys linearisoituna.

Luvussa 5.3 on esitetty menetelmät, miten kaksi ensimmäistä kohtaa voitaisiin toteuttaa luvussa 5.2 kuvatulla arviointijärjestelmällä.

6.2 Tulokset

Alustavat, suppeat tarkastelut osoittavat hyviä tuloksia tutkielmassa toteutetulle työkalulle. Esitettyjen menetelmien avulla voidaan tarkistaa asioita, joihin nykyiset saavutettavuuden arviointityökalut eivät kykene. Näitä tarkistuskohtia ovat esimerkiksi:

- Navigoinnin yhtenäisyyden tarkistukset.
- Navigointiosien ohittamismahdollisuuksien tarkistus.
- Sivun skaalautuvuus.
- Liian monen sisäkkäisen taulukon käyttö.

Lisäksi tutkielmassa esitettiin menetelmät, jotka mahdollistavat tulevaisuudessa seuraavien normaalia monimutkaisempien saavutettavuuskriteerien tarkistamisen:

- Värikontrastien automaattinen arviointi.
- Sivun taiton mukautuminen mobiililaitteille.
- Datataulukon esteettömyysominaisuuksien varmistaminen.

Lisäksi työkalu on kohtuullisen helposti laajennettavissa. Uuden tarkistuksen tekemiseksi riittää kirjoittaa `rules.js`-tiedostoon uusi funktio, joka lukee X-Y-leikkaushierarkiaa ja selainikkunan DOM-puuta sekä palauttaa mahdollisen virhetekstin ja korjausehdotustaulukon. Uusi tarkistus alkaa toimimaan, kun se lisätään tiedoston alussa olevaan listaan.

6.3 Jatkokehitysideat

Toteutettu työkalu tarvitsee vielä lisää sääntöjä, niin että arvioinnista tulee tarpeeksi kattava. Lisäksi työkaluun voisi toteuttaa sivun validointimahdollisuuden käyttämällä ulkoista validaattoripalvelua. Ohjelman voisi integroida myös muihin verkossa toimiviin saavutettavuuden arviointityökaluihin.

Raportointiosa voisi palauttaa arviointiraportin myös EARL-muodossa, joka on standardi esteettömyysraporteille. Raportointia voisi kehittää myös siihen suuntaan, että ongelmakohdat merkittäisiin suoraan tarkastetulle sivulle. Tämä onnistuu muokkaamalla selainikkunassa olevaa dokumenttia väliaikaisesti.

Tarkistus on jossain määrin hidas. Koodin osia voisi optimoida vielä tehokkaammaksi. Mozilla Firefox mahdollistaa XPCOM-komponenttien käytön osana selainta. Komponenttien avulla voisi mahdollisesti tehostaa laajennoksen toimintaa.

Sivuston kokonaisvaltaisessa arvioinnissa voisi hyödyntää palvelimelle kerättävää dataa. Näin esimerkiksi navigointien yhtenäisyydestä saataisiin kattavampi kuva. Monissa testeissä on jouduttu käyttämään heuristista päättelyä ja raja-arvoja. Tällaisissa kohdissa voisi hyödyntää tilastollista päättelyä ja datasta kerättyjä tietämyskantoja. Päättelyjärjestelmää voisi kehittää selviämään paremmin epävarman informaation kanssa esimerkiksi todennäköisyyksien avulla.

Työkalussa toteutettua menetelmää, jolla sivu jaetaan tunnistettaviin osiin, voitaisiin hyödyntää myös sivun mukauttamisessa selailun helpottamiseksi. Esimerkiksi ruudunlukijaa tai ruudunsuurentajaa käyttävä voisi hyötyä, jos sivulta voisi valita suoraan sisältölohkot luettavaksi. Sivun osista voisi tehdä myös koosteen, josta voisi hypätä haluttuun osioon.

7 YHTEENVETO

Liikenne- ja viestintäministeriön esteetöntä viestintää seuraavan työryhmän mukaan verkkosivujen esteettömyyttä koskevia suosituksia on riittävästi, mutta niiden soveltamiseen ja seurantaan tulisi kiinnittää enemmän huomiota [49]. Tämän tutkimuksen tarkoituksena oli kehittää saavutettavuuden automaattisen arvioinnin menetelmiä. Automaattisen arvioinnin tärkeä aspekti on saada kehittäjät tietoisiksi saavutettavuuteen vaikuttavista asioista.

Saavutettavuus ja sen lähikäsitteet määriteltiin tutkielman alussa. Tämän jälkeen tarkasteltiin aiheeseen liittyviä ongelmia, joita ovat arvioinnin haasteellisuus, saavutettavuuden vähäinen huomiointi ja tarve ulkoiselle motivoinnille. Tarkastelussa huomattiin, että hyvällä saavutettavuudella on monia positiivisia vaikutuksia pitkällä aikavälillä.

Seuraavissa luvuissa selvitettiin, millaisin eri tavoin verkkosivuja voidaan selata, mitkä ovat erilaisten tapojen oleellisimmat rajoitteet selailun kannalta ja mitä verkkosivuston ylläpitäjä voi tehdä tukeakseen erilaisia käyttäjäryhmiä. Samalla selvitettiin verkkosivun saavutettavuuteen vaikuttavat tekniset kriteerit. Tutkielmassa selvitetty kriteerilista ei pohjaudu pelkästään yhteen saavutettavuusohjeistoon, vaan se on kooste useiden erilaisten asiantuntijalähteiden suosituksista. Kriteerit on myös luokiteltu niiden noudattamisesta hyötyviin ryhmiin.

Kriteerien pohjalta lähdettiin luomaan helppokäyttöistä ja laajennettavissa olevaa arviointijärjestelmää, joka kykenee nykyisiä työkaluja paremmin huomioimaan ulkoasuun ja sivun taittoon liittyviä virheitä. Työkalu toteutettiin Mozilla Firefox -selaimen laajennoksena, mikä mahdollisti perinteisten yksitulkintaisten virhetarkistuksien lisäksi elementtien asettelun ja dynaamisten ominaisuuksien arvioinnin.

Nykyiset saavutettavuuden arviointityökalut, kuten WebXACT ja LIFT, pohjautuvat HTML-lähdekoodista saataviin tietoihin. Tällöin monia tärkeitä saavutettavuusnäkökohtia jää huomaamatta. Implisiittisten merkityksien analyysi jää näiltä työkaluilta kokonaan

tekemättä. Tässä työssä esitetty arviointijärjestelmä kykenee sen sijaan dekonstruoimaan piilossa olevien osien semanttisen informaation.

Alustavat testit isoissa uutissivustoissa ja portaaleissa antavat uudenlaiselle arviointityökalulle lupaavia tuloksia. Sivun osat tunnistetaan melko hyvin. DOM-rajapinnan, visuaalisen osioinnin ja osien tunnistuksen perusteella ohjelma kykenee tarkistamaan, onko sisältöosa helposti saavutettavissa ruudunlukijoilla, ovatko navigointiosat samoissa paikoissa eri sivuilla tai onko sivustolla esimerkiksi käytetty taulukkotaittoa esteellisesti. Tällaisia mahdollisuuksia ei muissa tarkastelemissani arviointityökaluissa ollut.

LÄHTEET

[1] Arla-instituutti, ”Toiminnallisen näön ongelmat”, saatavilla HTML-muodossa <URL: http://www.arlainst.fi/nv-peda/naonkaytto/ongelmat.htm >, viitattu 18.6.2006.

[2] Baron David, “Mozilla’s Quirks Mode”, saatavilla HTML-muodossa <URL: http://developer.mozilla.org/en/docs/Mozilla’s_Quirks_Mode >, 12.12.2004.

[3] Bos Bert, Çelik Tantek, Hickson Ian, Lie Håkon Wium, “Cascading Style Sheets, level 2 revision 1”, saatavilla HTML- ja PDF-muodoissa <URL: http://www.w3.org/TR/CSS21/ >, 11.4.2006.

[4] Brajnik Giorgio, “Comparing accessibility evaluation tools: a method for tool effectiveness”, Universal Access in the Information Society, numero 3, sivut 252-263, lokakuu 2004.

[5] Brewer Judy, “How People with Disabilities Use the Web”, saatavilla HTML-muodossa <URL: http://www.w3.org/WAI/EO/Drafts/PWD-Use-Web/ >, 5.5.2005.

[6] Brøndsted Tom, Aaskoven Erik: "Voice-controlled internet browsing for motor-handicapped users. design and implementation issues", INTERSPEECH-2005, sivut 185-188, 2005.

[7] Caldwell Ben, Chisholm Wendy, Slatin John, Vanderheiden Gregg, “Web Content Accessibility Guidelines 2.0”, saatavilla HTML-muodossa <URL: http://www.w3.org/TR/WCAG20/ >, 27.4.2006.

[8] Capin Tolga, “Mobile SVG Profiles: SVG Tiny and SVG Basic”, saatavilla HTML-muodossa <URL: http://www.w3.org/TR/SVGMobile/ >, 14.1.2003.

[9] Centeno Vicente, Kloos Carlos, Fisteus Jesús, Álvarez Luis, “Web Accessibility Evaluation Tools: A Survey and Some Improvements”, Electronic Notes in Theoretical Computer Science, numero 157, sivut 87-100, 2006.

[10] Chen Yu, Xie Xing, Ma Wei-Ying, Zhang Hong-Jiang, “Adapting Web Pages for Small-Screen Devices”, IEEE Internet Computing, numero 9, sivut 50-56, helmikuu 2005.

[11] Chisholm Wendy, Vanderheiden Gregg, Jacobs Ian, “Core Techniques for Web Content Accessibility Guidelines 1.0”, saatavilla HTML-muodossa <URL: http://www.w3.org/TR/WCAG10-CORE-TECHS/ >, 6.11.2000.

[12] Chisholm Wendy, Vanderheiden Gregg, Jacobs Ian, “HTML Techniques for Web Content Accessibility Guidelines 1.0”, saatavilla HTML-muodossa <URL: http://www.w3.org/TR/WCAG10-HTML-TECHS/ >, 6.11.2000.

[13] Chisholm Wendy, Vanderheiden Gregg, Ian Jacobs, “Web Content Accessibility Guidelines 1.0”, saatavilla teksti-, HTML- ja PDF-muodoissa <URL: http://www.w3.org/TR/WCAG10/ >, 5.5.1999.

[14] Christian Kevin, Kules Bill, Shneiderman Ben, Youssef Adel, ” A comparison of voice controlled and mouse controlled web browsing”, ACM SIGACCESS Conference on Assistive Technologies, Proceedings of the fourth international ACM conference on Assistive technologies, sivut 72 - 79, 2000.

[15] Clark Joe, “Big, Stark & Chunky”, saatavilla HTML-muodossa <URL: http://www.alistapart.com/articles/lowvision >, 11.1.2005.

[16] Clark Joe, ”Building Accessible Websites”, New Riders Press, Indianapolis, 2002.

[17] Clark Joe, “Facts and Opinions About PDF Accessibility”, saatavilla HTML-muodossa <URL: http://www.alistapart.com/articles/pdf_accessibility >, 22.8.2005.

[18] Davis Joel, “Disenfranchising the Disabled: The Inaccessibility of Internet-Based Health Information”, Journal of Health Communication, numero 7, sivut 355-368, 2002.

[19] Education and Outreach Working Group (EOWG), “Evaluating Web Sites for Accessibility”, saatavilla HTML-muodossa <URL: http://www.w3.org/WAI/eval/ >, viitattu 28.6.2006.

[20] Euroopan yhteisöjen komissio, “eEurope 2002: Julkisen sektorin verkkosivujen ja niiden sisällön saavutettavuus”, saatavilla PDF-muodossa <URL: http://europa.eu.int/eur-lex/fi/com/cnc/2001/com2001_0529fi01.pdf >, 25.9.2001.

[21] Euroopan yhteisöjen komissio, ”Komission tiedonanto neuvostolle, Euroopan parlamentille, Euroopan talous- ja sosiaalikomitealle sekä aluieiden komitealle - Esteetön tietoyhteiskunta (eAccessibility)”, saatavilla PDF-muodossa <URL: http://eur-lex.europa.eu/LexUriServ/site/fi/com/2005/com2005_0425fi01.pdf >, 13.9.2005.

[22] Ferraiolo Jon, Jun Fuijisawa, Jackson Dean, “Scalable Vector Graphics (SVG) 1.1 Specification”, saatavilla HTML-muodossa <URL: http://www.w3.org/TR/SVG/ >, 14.1.2003.

[23] Finlex, “Suomen perustuslaki 11.6.1999/731”, saatavilla HTML-muodossa http://www.finlex.fi/fi/laki/ajantasa/1999/19990731, viitattu 11.6.2006.

[24] Freedom Scientific BLV Group, “Jaws 7.0 Help”, saatavilla CHM-muodossa <URL: http://www.freedomscientific.com/fs_support/doc_screenreaders.asp >.

[25] Gimson Roger ym. ” Device Independence Principles”, saatavilla HTML-muodossa <URL: http://www.w3.org/TR/di-princ/ >, 1.9.2003.

[26] Google Inc., “Accessible Search FAQ”, saatavilla HTML-muodossa <URL: http://labs.google.com/accessible/faq.html >, 19.7.2006.

[27] Google Inc., “Webmaster Guidelines”, saatavilla HTML-muodossa <URL: http://www.google.com/support/webmasters/bin/answer.py?answer=35769 >, viitattu 15.6.2006.

[28] Graham James, Thompson James, ”Introduction to using XPath in JavaScript”, saatavilla HTML-muodossa <URL: http://developer.mozilla.org/en/docs/Introduction_to_using_XPath_in_JavaScript >, 25.3.2006.

[29] GW Micro, “Window-Eyes 5.5 Manual”, saatavilla CHM-, HTML- ja PDF-muodossa <URL: http://www.gwmicro.com/Window-Eyes/Manual/ >, 2005.

[30] Ha Jaekyu, Haralick Robert, Phillips Ihsin, ”Recursive X-Y cut using bounding boxes of connected components”, Third International Conference on Document Analysis and Recognition (ICDAR'95), numero 2, sivut 952-955, 1995.

[31] Hackett Stephanie, Parmanto Bambang, “A longitudinal evaluation of accessibility: higher education web sites”, Internet Research, numero 15, sivut 281-294, 2005.

[32] Information Society and Media Directorate-General, “Design for All (DfA)”, saatavilla HTML-muodossa <URL: http://europa.eu.int/information_society/policy/accessibility/deploy/dfa/index_en.htm >, viitattu 11.6.2006.

[33] International Standards Organization (ISO), “ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 11: Guidance on usability.”, 1998.

[34] International Standards Organization (ISO) “ISO 16071:2002 Accessibility of Human-Computer Interfaces.”, ISO copyright office, Geneva, 2002.

[35] Julkisen hallinnon tietohallinnon neuvottelukunta JUHTA, ”JHS 129 Julkishallinnon verkkopalvelun suunnittelun ja toteuttamisen periaatteet”, saatavilla HTML- ja PDF-muodoissa <URL: http://www.jhs-suositukset.fi/suomi/jhs129 >, 15.6.2005.

[36] Juntunen Reijo, Jylhä Virpi, Laatunen Petri, Söderholm Maria, ”Näkövammaistahojen testausohjeet verkkosivuille ja -palveluille”, saatavilla HTML-muodossa <URL: http://www.nkl.fi/tietoa/esteettomyys/testohje.htm >, viitattu 28.6.2006.

[37] Kaikkonen Anne ja Roto Virpi, *Navigating in a Mobile XHTML Application*, “Proceedings of the SIGCHI conference on Human factors in computing systems”, ACM Press, sivut 329 - 336, 2003.

[38] Kasday Leonard, “A tool to evaluate universal Web accessibility”, ACM Conference on Universal Usability, Proceedings on the 2000 conference on Universal Usability, sivut 161-162, 2000.

[39] Korpela Jukka, ”Esteettömyysopas”, TIEKE, saatavilla HTML-muodossa <URL: http://arkisto.tieke.fi/esteettomyysopas/ >, 2002.

[40] Korpela Jukka, ”Esteettömyysopas”, TIEKE – Tietoyhteiskunnan kehittämiskeskus ry., saatavilla PDF-muodossa <URL: http://www.tieke.fi/tuotteet_ja_palvelut/esteettomyys/esteettomyysopas/ >, 2003.

[41] Krüpl Bernhard, Herzog Marcus, Gatterbaur, “Using visual cues for extraction of tabular data from arbitrary HTML documents”, Special interest tracks and posters of the 14th international conference on World Wide Web (WWW '05) , sivut 1000-1001, toukokuu 2005.

[42] Lazar Jonathan, Dudley-Sponaugle Alfreda, Greenidge Kisha-Dawn, “Improving web accessibility: a study of webmaster perceptions”, Computers in Human Behavior, numero 20, sivut 269–288, 2004.

[43] Lemon Gez, “Colour Contrast Analyser Firefox Extension”, saatavilla HTML-muodossa <URL: http://juicystudio.com/article/colour-contrast-analyser-firefox-extension.php >, 12.2.2006.

[44] Lemon Gez, Faulkner Steve, “Making Ajax Work with Screen Readers”, saatavilla HTML-muodossa <URL: http://juicystudio.com/article/making-ajax-work-with-screen-readers.php >, 25.5.2006.

[45] Le Hégaret Philippe, Wood Lauren, Whitmer Ray, “W3C Document Object Model”, saatavilla HTML-muodossa <URL: http://www.w3.org/DOM/ >, 12.6.2006.

[46] Lie Håkon Wium, “Cascading Style Sheets”, Series of dissertations submitted to the Faculty of Mathematics and Natural Sciences, University of Oslo, numero 498 (2005).

[47] Lie Håkon Wium, Çelik Tantek, Glazman Daniel, “Media Queries”, saatavilla HTML-muodossa <URL: http://www.w3.org/TR/css3-mediaqueries/ >, 8.7.2002.

[48] Liikenne- ja viestintäministeriö, ”Kohti esteetöntä viestintää. Toimenpideohjelma.”, saatavilla RTF-muodossa <URL: http://www.mintc.fi/oliver/upl406-1_2005%20rtf.rtf >, 26.1.2005.

[49] Liikenne- ja viestintäministeriö, ”Kohti esteetöntä viestintää. Toimenpideohjelman seurantaraportti.”, Liikenne- ja viestintäministeriön julkaisuja, numero 10, 10.1.2006.

[50] Liikenne- ja viestintäministeriö, Stakes Design for All -verkosto, Helsingin ammattikorkeakoulu Stadia, ”Selvitys esteettömyyden huomioimisesta Suomen kunnissa”, saatavilla PDF-muodossa <URL: http://dfasuomi.stakes.fi/NR/rdonlyres/B10FBE70-D748-4339-AE0A-995F7EDB8D31/0/Kuntaselvitys.pdf >, 8.7.2005.

[51] Loiacono Eleanor, McCoy Scott, “Website accessibility: a cross-sector comparison”, Universal Access in Information Society, numero 4, sivut 393-399, 10.12.2005.

[52] Microsoft, “Guidelines for successful indexing”, saatavilla HTML-muodossa <URL: http://search.msn.com/docs/siteowner.aspx?t=SEARCH_WEBMASTER_REF_GuidelinesforOptimizingSite.htm >, viitattu 8.6.2006.

[53] Mäntylä Jukka, “Acc - an Accessibility Evaluator”, saatavilla HTML-muodossa <URL: http://appro.mit.jyu.fi/tools/acc/ >, viitattu 31.7.2006.

[54] Nielsen Jakob, ”Designing Web Usability: The Practice of Simplicity”, New Riders Publishing, Indianapolis, 2000.

[55] Nielsen Jakob, “Hypertext and hypermedia”, Academic Press, San Diego, 1990.

[56] Nielsen Jakob, “Top Ten Web Design Mistakes of 2005”, saatavilla HTML-muodossa <URL: http://www.useit.com/alertbox/designmistakes.html >, 3.10.2005.

[57] Niensted Walter, Hänninen Osmo, Arstila Antti, Björkqvist Stig-Eyrik, ”Ihmisen fysiologia ja anatomia”, WS Bookwell Oy, Porvoo, 2000.

[58] Nilsson Nils, “Artificial Intelligence: a new synthesis.”, Morgan Kaufmann Publishers, San Francisco, 1998.

[59] Nokia Corporation, “S60 Platform: Designing XHTML Mobile Profile Content”, saatavilla PDF-muodossa <URL: http://sw.nokia.com/id/4f7b6805-47d7-4914-885c-6ef2b487adf6/Series_60_Platform_Designing_XHTML_MP_Content_v1_4_en.pdf >, 24.5.2005.

[60] Nokia Corporation, ”XHTML Guidelines for Creating Web Content”, saatavilla PDF-muodossa <URL: http://sw.nokia.com/id/7f3f1424-b51e-4067-a3ef-acaab08e484f/XHTML_Guidelines_For_Creating_Web_Content_v1_3_en.pdf >, 14.10.2005.

[61] Nykänen Ossi, W3C Suomen toimisto, ”Web-saavutettavuuden 1-2-3”, saatavilla HTML-muodossa <URL: http://www.w3c.tut.fi/reports/2003/0508wai-intro/index.html >, 7.10.2003.

[62] Nykänen Ossi, W3C Suomen toimisto, ”World Wide Web Consortium …7 kohdan tiivistelmä”, saatavilla HTML-muodossa <URL: http://www.w3c.tut.fi/translations/consortium/w3c-in-7-points/index.html >, 28.8.2003.

[63] Opera Software, “The Opera 7 and 8 DOCTYPE Switches”, saatavilla HTML-muodossa <URL: http://www.opera.com/docs/specs/doctype/ >, viitattu 9.6.2006.

[64] Penn Gerald, Hu Jianying, Luo Hengbin, McDonald Ryan, “Flexible Web Document Analysis for Delivery to Narrow-Bandwidth Devices”, Sixth International Conference on Document Analysis and Recognition (ICDAR'01), sivut 1074-1078, syyskuu 2001.

[65] Petrie Helen, Hamilton Fraser, King Neil, ”Tension, what tension? Website accessibility and visual design”, Proceedings of the 2004 international cross-disciplinary workshop on Web accessibility (W4A), ACM Press, sivut 13 - 18, 2004.

[66] Rabin Jo, McCathieNevile Charles, “Mobile Web Best Practices 1.0”, saatavilla HTML-muodossa <URL: http://www.w3.org/TR/mobile-bp/ >, 18.5.2006.

[67] Raggett Dave, Le Hors Arnaud, Jacobs Ian, “HTML 4.01 Specification”, saatavilla HTML- ja PDF-muodoissa <URL: http://www.w3.org/TR/REC-html40/ >, 24.12.1999.

[68] Raike Antti, ”Löytäjät elokuvantajua rakentamassa: [yhteisöllinen www-palvelun tuotanto]”, Taideteollinen korkeakoulu, Helsinki, 2005.

[69] Schwerdtfeger Richard, Gibson Becky, “DHTML Accessibility - Solving the JavaScript Accessibility Problem”, ACM SIGACCESS Conference on Assistive Technologies, Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility, sivut 202 - 203, 2005.

[70] Shawn Lawton Henry ym., “Introduction to Accessibility”, saatavilla HTML-muodossa <URL: http://www.w3.org/WAI/gettingstarted/ >, syyskuu 2005.

[71] Silver Lance, “CSS Enhancements in Internet Explorer 6”, saatavilla HTML-muodossa <URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnie60/html/cssenhancements.asp >, maaliskuu 2001.

[72] Snellman Kalle, ”Mobiilipalvelumarkkinat Suomessa 2005”, Liikenne- ja viestintäministeriön julkaisuja, numero 22, 26.4.2006.

[73] Stakes Viestintä, “Sosiaali- ja terveyshuollon palveluketjusanasto”, saatavilla PDF-muodossa <URL: http://sty.stakes.fi/FI/sanastot/index.htm >, 2002.

[74] Tahkokallio Päivi, Essi-työryhmä, ”Tosi maailma käytettäväksi ja saavutettavaksi”, saatavilla HTML-muodossa <URL: http://appro.mit.jyu.fi/essikurssi/dfa/t2/ >, 7.1.2004.

[75] Thatcher Jim ym. ”Constructing Accessible Web Sites”, Springer-Verlag, New York, 2002.

[76] Theofanos Mary, Redish Janice, “Helping low-vision and other users with websites that meet their needs: Is one site for all feasible?”, Technical communication, numero 52, sivut 9-20, helmikuu 2005.

[77] Theofanos Mary, Redish Janice, “Guidelines for Accessible - and Usable - Web Sites: Observing Users Who Work With Screenreaders”, interactions, numero 10, 2003, sivut 6-36.

[78] Tilastokeskus, ”Väestöennuste 2004”, saatavilla HTML-muodossa <URL: http://www.stat.fi/til/vaenn/2004/ >, 20.9.2004.

[79] United Nations, ”48/96. Standard Rules on the Equalization of Opportunities for Persons with Disabilities”, saatavilla HTML-muodossa <URL: http://www.un.org/documents/ga/res/48/a48r096.htm >, 20.12.1993.

[80] UsableNet, “UsableNet Technology”, saatavilla HTML-muodossa <URL: http://www.usablenet.com/usablenet_technology/usablenet_technology.html >, 8.10.2003.

[81] W3C, “XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)”, saatavilla XHTML ja PDF-muodoissa <URL: http://www.w3.org/TR/xhtml1/ >, 1.8.2002.

[82] Valtioneuvoston kanslia, ”Hyvä yhteiskunta kaikenikäisille – Valtioneuvoston tulevaisuusselonteko väestökehityksestä, väestöpolitiikasta ja ikärakenteen muutokseen varautumisesta”, Valtioneuvoston kanslian julkaisusarja, numero 27, 2004.

[83] Watchfire, “How a Scan for Accessibility Works”, saatavilla HTML-muodossa <URL: http://webxact3.watchfire.com/themes/standard-en-us/help/AXM_overview5.html >, viitattu 31.7.2006.

[84] Wilton-Jones Tarquin, “The Opera Browser: Extensible Rendering Architecture”, saatavilla HTML-muodossa <URL: http://www.opera.com/docs/whitepapers/ >, viitattu 20.7.2006.

[85] Wireless Application Protocol Forum, “XHTML Mobile Profile”, saatavilla PDF-muodossa <URL: http://www.openmobilealliance.org/tech/affiliates/wap/wap-277-xhtmlmp-20011029-a.pdf >, 29.10.2001.

[86] Yahoo!, ”How do I improve the ranking of my web site in the search results?”, saatavilla HTML-muodossa <URL: http://help.yahoo.com/help/us/ysearch/ranking/ranking-02.html >

[87] Zeldman Jeffrey, ”Designing with web standards”, New Riders Publishing, Indianapolis, 2003.

[88] Zeng Xiaoming, Parmanto Bambang, “Metric for Web Accessibility Evaluation”, Journal of the American Society for Information Science and Technology, numero 56, sivut 1394-1404, 2005.

[89] Zeng Xiaoming, Parmanto Bambang, “Web content accessibility of consumer health information web sites for people with disabilities: A cross sectional evaluation”, Journal of medical Internet research, numero 6, 2004.

LIITTEET

Liite 1. Saavutettavuuteen vaikuttavat attribuutit ja niistä hyötyvät ryhmät

Ohje	Ei-visuaaliset	Poikkeavat näyttötilat	Erilaiset ohjaustavat	Muut rajoitteet
Merkitse dokumenttityyppi, mahdollinen XML-deklaraatio ja tee valideja ja hyvin muodostettuja dokumentteja.	x	x		
Älä käytä epästandardeja elementtejä, kuten `blink`, `marquee` tai `embed`.	x			
Vältä vanhentuvia elementtejä, kuten `b`, `i`, `u`, `font`, `fontbase`, `dir`, `menu`, `center` ja `strike`.	x	x		
Kirjoita sivua kuvaava otsikko `title`-elementillä.	x	x	x	
Merkitse otsikot `h1-h6` -elementeillä loogisessa järjestyksessä, jossa tason muutos on edeltävään enintään yksi.	x	x	x	
Merkitse kukin yhtenäinen tekstikappale `p`-elementillä. Yksittäinen rivi ei välttämättä ole kappale. `br`-elementillä ei saa rivittää tarpeettomasti.	x			x
Listat on merkittävä `ul`-, `ol`- tai `dl`-elementillä. Listoja ei saa tehdä `div`-, `p`-, `span`- ja `br`-elementtien avulla. Listamerkkiä ei saa tehdä img-elementillä, vaan se on määriteltävä CSS:ssä.	x			x
Uudet erikoiset lyhenteet ja määritelmät on merkittävä `abbr`-, `acronym`- tai `dfn`-elementeillä.	x			x
Merkitse taulukon otsikkosolut `th`-elementillä. Älä käytä `b`-elementtejä merkitsemään otsikkoa.	x	x		
Kirjoita taulukon otsikko `caption`-elementtiin ja tiivistelmäteksti `summary`-ominaisuuteen.	x			
Liitä kukin datataulukon tietosolu otsikkosoluihin `scope`-ominaisuudella tai `headers`- ja `id`-parilla.	x	x		

Merkitse luonnollinen kieli ja sen muutokset `xml:lang` tai `lang`-ominaisuudella.	x			
Käytä yksinkertaista ja selkeää kieltä.				x
Määrittele tekstin ja taustanvärin tai taustakuvan kontrasti riittävän suureksi.				x
Käytä ulkoasun muotoiluun validoituja CSS-tyylejä.	x	x		
Sivun on toimittava myös ilman CSS-määrityksiä.	x	x		
Älä merkitse pelkästään väreillä mitään tärkeää informaatiota, joka on sivun ymmärrettävyyden tai vuorovaikutuksen kannalta olennaista.		x		x
Määritä kirjasin CSS:n avulla. Määritä aina myös yleinen kirjasinperhe aina `font-family`-määrityksissä.		x		
Tekstin kokoa täytyy pystyä muuttamaan. Käytä siis CSS:ää ja suhteellisia yksiköitä, kuten `em` ja `%`.		x		
Älä käytä kehyksiä vaan asemoi CSS-ominaisuuksien avulla. Jos käytät kehyksiä, niin merkitse kehykset kuvaavilla `name`- ja `title`-ominaisuuksilla.	x	x		
Käytä sivun taitossa vain yksinkertaisia taulukoita (ei sisäkkäisiä).	x			
Älä käytä taulukkotaittoa. Asemoi lohkot CSS-ominaisuuksien avulla.		x		
Tarjoa yksipalstainen taitto vaihtoehtoisten tyylisivujen avulla ruudunsuurennosta ja `handheld`-mediaa varten.		x		
Sivun sisältöosan mukauduttava tekstin ja ikkunan kokoon.		x	x	x
Sivun navigointiosan on mukauduttava pieniin näyttöihin ja se on sijoitettava sivun ylälaitaan.		x		
Navigointipalkkien rakenteen ja paikan on oltava yhtenäinen eri sivuilla.	x	x		x
Tee sisäinen linkki, jolla voi ohittaa navigointipalkin, tai sijoita asiasisällöt sivun alkuun.	x	x	x	

Kytke useita lohkoja sisältävässä sivussa merkitykselliset osat toisiinsa sisäisillä linkeillä.	x		x	
Kirjoita kohdetta kuvaavia linkkitekstejä. Täsmennä kuvausta `a`-elementin `title`-ominaisuudella.	x	x	x	
Ilmaise `link`-elementeillä, miten sivu liittyy toisiin dokumentteihin.	x		x	
Merkitse useasti käytetyt linkit ja lomakkeen pääkohdat `accesskey`-ominaisuudella. Käytä ominaisuutta johdonmukaisesti kaikilla sivuilla.	x	x	x	
Määritä hyperlinkeille ja lomake-elementeille looginen läpikäyntijärjestys `tabindex`-ominaisuudella, ellei lineaarinen järjestys ole järkevä.	x		x	
Anna lomake-elementeille otsikko `label`-elementillä ja kytke ne toisiinsa `for`-ominaisuudella tai sijoittamalla lomake-elementti otsikon sisään.	x		x	
Ryhmittele lomake-elementit `fieldset`- ja `legend`-elementeillä.	x			x
Valintalista, jossa on monta `option`-elementillä tehtyä vaihtoehtoa, on ryhmiteltävä `optgroup`-elementillä.	x		x	x
Tee lomakkeista lyhyitä ja tiiviitä.		x	x	
Kirjoita kuvan merkitystä kuvaava lyhyt vaihtoehtoteksti `alt`-ominaisuuteen.	x	x		
Välistys- tai täytekuville täytyy antaa tyhjä `alt`-teksti.	x			
Kirjoita pidempi kuvan kuvaus `title`-ominaisuuteen. Jos selitys on pitkä, niin kirjoita se erilliseen dokumenttiin ja linkitä se kuvaan `longdesc`-ominaisuudella.	x			
Kirjoita kuvakartan jokaiselle alueelle `alt`-teksti ja tarkenna `title`-elementtiin kohteen kuvaus.	x	x	x	
Pidä sivun koko kohtuullisena ja vältä isoja kuvia. Suosi kuvaformaateista SVG:tä.		x		

Tarkista, että kuvalle on määrätty `width`- ja `height`-ominaisuuksien arvot.		×		
Vältä ajastettuja sivun uudelleenlatauksia.	×	×		×
Älä avaa uusia ikkunoita.	×	×		×
Tee linkit `a`-elementeillä JavaScript-komentojen sijaan.	×			
Anna käyttäjälle mahdollisuus kontrolloida liikkuvia ja välkkyviä elementtejä sekä multimediaa sivulla.		×		×
Vältä hiireen tilaan ja sivun tilan muutoksiin pohjautuvia tapahtumankäsittelijöitä.	×		×	
Dynaamisen sivun muokkauksen on päivitettävä avustavien tekniikoiden muistit ja asetettava kursorin kohdistus.	×			
Kirjoita vaihtoehtoiset tekstit graafisille objekteille ja sovelmille.	×	×		
Tarjoa multimedialle varmasti saavutettava vaihtoehtomuoto.	×	×		
Lisää multimediasisällöille tekstivaihtoehdot.	×			
Toteuta multimedian ja sovelmien navigointi myös pelkästään näppäimistöllä toimivaksi.	×		×	
Synkronoi kuvaileva tekstitys videon ja äänen kanssa.				×
Mahdollista kuvailevan ääniraidan käyttö videon kanssa.	×			

Liite 2. Arviointijärjestelmän ohjelmakoodi

```
/* acc.js - Jukka Mäntylä 31.7.2006 */

const ACC_XYNODE_COMMON = 0;
const ACC_XYNODE_CONTENT = 1;
const ACC_XYNODE_NAVIGATION = 2;
const ACC_XYNODE_EMPTY = 3;
const ACC_XYNODE_CONTAINER = 4;

const ACC_BOX_COMMON = 0;
const ACC_BOX_TEXT_ELEM = 1;
const ACC_BOX_TEXT_NODE = 2;
const ACC_BOX_REPLACED = 3;
const ACC_BOX_FORM = 4;
const ACC_BOX_VERTICAL_SEPARATOR = 5;
const ACC_BOX_HORIZONTAL_SEPARATOR = 6;
const ACC_BOX_ANCHOR = 7;
const ACC_BOX_CONTAINER = 8;

const ACC_BOXEXTRACT_RECURSION_DEPTH = 5;
const ACC_BOXEXTRACT_X_SPLIT_THRESHOLD = 0.75;
const ACC_BOXEXTRACT_Y_SPLIT_THRESHOLD = 0.8;
const ACC_BOXEXTRACT_MAX_COLUMNS = 9;
const ACC_BOXEXTRACT_MAX_ROWS = 50;
const ACC_BOXEXTRACT_TEXTLENGTH_RATIO_THRESHOLD = 40.0;
const ACC_BOXEXTRACT_ANCHOR_RATIO_THRESHOLD = 0.4;
const ACC_BOXEXTRACT_X_SEPARATOR_THRESHOLD = 0.4;
const ACC_BOXEXTRACT_Y_SEPARATOR_THRESHOLD = 0.1;

var currentResultsLocation = "";
var currentXYTree = null;
var currentXYTreeLocation = "";
var currentPageWidth = 0;
var currentPageHeight = 0;
var previousXYTree = null;
var previousXYTreeLocation = "";
var previousPageWidth = 0;
var previousPageHeight = 0;
var islayoutvisible = false;
var boundingboxlist = null; // Lista sisältöelementtien koosta ja asemoinnista

window.onclose = function() {
  if (window.getElementById("acc_autocheck").checked == true)
    opener.removeEventListener("DOMContentLoaded", acc_run_tests, false);
}

// painikkeiden käsittelijöitä
function toggleShowLayout(event, checkbox) {
  if (islayoutvisible && currentXYTreeLocation ==
opener.content.document.location.toString()) {
    acc_hideLayout();
  } else {
    acc_showLayout();
  }
}

function toggleCompareWithPrevious(event, checkbox) {
  if (checkbox.checked == true) {
    acc_extractLayoutMetadata();
  }
}

function toggleChangeListener(event, checkbox) {
  if (checkbox.checked == true)
    opener.window.addEventListener("DOMContentLoaded", acc_run_tests, false);
  else
    opener.window.removeEventListener("DOMContentLoaded", acc_run_tests, false);
}
```

```
// Puuhierarkian datamalli raportointia varten
var visibleData = new Array();
var childData = new Array();

var accTreeView = {

  treeBox: null,
  selection: null,

  get rowCount()                     { return visibleData.length; },
  setTree: function(treeBox)         { this.treeBox = treeBox; },
  getCellText: function(idx, column) { var col_id = typeof(column) == "object" ?
column.id : column;
                                       if (col_id == "acc_warnings")
                                         return visibleData[idx][0];
                                       else
                                         return visibleData[idx][1]; },
  isContainer: function(idx)         { return visibleData[idx][2]; },
  isContainerOpen: function(idx)     { return visibleData[idx][3]; },
  isContainerEmpty: function(idx)    { return false; },
  isSeparator: function(idx)         { return false; },
  isSorted: function()               { return false; },
  isEditable: function(idx, column)  { return false; },

  getParentIndex: function(idx) {
    for (var t = idx - 1; t >= 0 ; t--) {
      if (this.isContainer(t)) return t;
    }
    return -1;
  },
  getLevel: function(idx) {
    if (this.isContainer(idx)) return 0;
    return 1;
  },
  hasNextSibling: function(idx, after) {
    var thisLevel = this.getLevel(idx);
    for (var t = idx + 1; t < visibleData.length; t++) {
      var nextLevel = this.getLevel(t);
      if (nextLevel == thisLevel)
        return true;
      if (nextLevel < thisLevel)
        return false;
    }
    return false;
  },
  toggleOpenState: function(idx) {
    var item = visibleData[idx];
    if (!item[2]) return;

    if (item[3]) {
      item[3] = false;

      var thisLevel = this.getLevel(idx);
      var deletecount = 0;
      for (var t = idx + 1; t < visibleData.length; t++) {
        if (this.getLevel(t) > thisLevel) deletecount++;
        else break;
      }
      if (deletecount) {
        visibleData.splice(idx + 1, deletecount);
        this.treeBox.rowCountChanged(idx + 1, -deletecount);
      }
    }
    else {
      item[3] = true;

      var label = visibleData[idx][0];
      var toinsert = childData[label];
      for (var i = 0; i < toinsert.length; i++) {
        visibleData.splice(idx + i + 1, 0, toinsert[i]);
```

```
      }
      this.treeBox.rowCountChanged(idx + 1, toinsert.length);
    }
  },

  getImageSrc: function(idx, column) {},
  getProgressMode : function(idx,column) {},
  getCellValue: function(idx, column) {},
  cycleHeader: function(col, elem) {},
  selectionChanged: function() {},
  cycleCell: function(idx, column) {},
  performAction: function(action) {},
  performActionOnCell: function(action, index, column) {},
  getRowProperties: function(idx, column, prop) {},
  getCellProperties: function(idx, column, prop) {},
  getColumnProperties: function(column, element, prop) {}
}

// Bounding Box -luokka (lohkon koko- ja paikkatiedot)
function Box(x1,y1,x2,y2,type) {
  this.x1 = x1;
  this.y1 = y1;
  this.x2 = x2;
  this.y2 = y2;
  this.type = type;
}

Box.prototype.getXMin = function() { return this.x1; }
Box.prototype.getXMax = function() { return this.x2; }
Box.prototype.getMin = function(isVertical) { if (isVertical) return (this.x1); return
(this.y1); }
Box.prototype.getMax = function(isVertical) { if (isVertical) return (this.x2); return
(this.y2); }
Box.prototype.getYMin = function() { return this.y1; }
Box.prototype.getYMax = function() { return this.y2; }
Box.prototype.getBox = function() { return [this.x1,this.y1,this.x2,this.y2]; }
Box.prototype.getWidth = function() { return (this.x2 - this.x1 + 1); }
Box.prototype.getHeight = function() { return (this.y2 - this.y1 + 1); }
Box.prototype.getType = function() { return this.type; }
Box.prototype.isInsideBox = function(abox) {
  return (abox.getXMin()-2 <= this.x1 && this.x2 <= abox.getXMax()+2 &&
          abox.getYMin()-2 <= this.y1 && this.y2 <= abox.getYMax()+2)
}
Box.prototype.splitsBox = function(abox, isvertical) {
   if (isvertical) {
     if (abox.getXMin() <= this.x1 && this.x2 <= abox.getXMax()) {
       // sisälle jäävän osan koko riittävän iso
       if ( (Math.min(this.y2, abox.getYMax()) - Math.max(this.y1, abox.getYMin())) /
abox.getHeight() > ACC_BOXEXTRACT_X_SPLIT_THRESHOLD )
         return true;
     }
  } else {
     if (abox.getYMin() <= this.y1 && this.y2 <= abox.getYMax()) {
       if ( (Math.min(this.x2, abox.getXMax()) - Math.max(this.x1, abox.getXMin())) /
abox.getWidth() > ACC_BOXEXTRACT_Y_SPLIT_THRESHOLD )
         return true;
     }
  }
  return false;
}
Box.prototype.isInsideInterval = function(coord, isVertical) {
  if (isVertical)
    return ((this.x1 <= coord && coord <= this.x2) ? 1 : 0);
  else
    return ((this.y1 <= coord && coord <= this.y2) ? 1 : 0);
}

// ContentBox-luokka (sisältää lisäksi tekstin pituuden)
ContentBox.prototype = new Box;
ContentBox.prototype.constructor = ContentBox;
function ContentBox(x1,y1,x2,y2,type,textlength)
```

```
{
  Box.call(this,x1,y1,x2,y2,type);
  this.contentlength = textlength;
}
ContentBox.prototype.getContentLength = function() { return this.contentlength; }

// RefBox-luokka (sisältää referenssin DOM-puun solmuun)
RefBox.prototype = new ContentBox;
RefBox.prototype.constructor = RefBox;
function RefBox(x1,y1,x2,y2,type,textlength,ref)
{
  ContentBox.call(this,x1,y1,x2,y2,type,textlength);
  if (ref)
    this.ref = ref;
  else
    this.ref = null;
}
RefBox.prototype.getNode = function() { return this.ref; }

// XYNode-luokka (X-Y-leikkauspuuta varten)
function XYNode(x1,y1,x2,y2,boxtype) {
  this.box = new Box(x1,y1,x2,y2,boxtype);
  this.children = null;
  this.type = ACC_XYNODE_COMMON;
}

XYNode.prototype.setType = function(newtype) { this.type = newtype; }
XYNode.prototype.getType = function() { return this.type; }
XYNode.prototype.getChildren = function() { return this.children; }
XYNode.prototype.addChild = function(newnode) {
  if (this.children == null)
    this.children = new Array();
  if (newnode != null)
    this.children.push(newnode);
}
XYNode.prototype.getLeafChildrenOfType = function(type) {
  function acc_addLeafToArray(nod, array) {
    var children = nod.getChildren();
    if (children != null) {
      for (var i=0;i<children.length;i++) {
        acc_addLeafToArray(children[i], array);
      }
      return;
    }
    if (type == nod.getType())
      array.push(nod);
  }

  var leafchildren = new Array();
  acc_addLeafToArray(this, leafchildren);
  return leafchildren;
}

// puhdistaa raportin tulokset
function acc_clear_results() {
  visibleData = new Array();
  childData = new Array();
  document.getElementById("acc_tree").view = accTreeView;
}

/* avaa lähdekoodi-ikkunan
   huomioi dynaamiset muutokset */
function acc_find_source() {

opener.content.getSelection().selectAllChildren(opener.content.document.documentElemen
t);
        var sel = opener.content.getSelection();
        var dlg = opener.openDialog("chrome://global/content/viewPartialSource.xul",
                            "_blank", "scrollbars,resizable,chrome,dialog=no",
                            currentResultsLocation,
opener.content.document.characterSet, sel, "selection");
```

```
        opener.setTimeout(acc_clear_selections, 1500, sel, dlg);
}

// etsii koodipalasen lähdekoodista
function acc_clear_selections(sel, dlg) {

  var tree = document.getElementById("acc_tree");
  var col = tree.columns ? tree.columns['acc_snippets'] : 'acc_snippets';
  var src = tree.view.getCellText(tree.currentIndex, col);

  sel.removeAllRanges();
  dlg.getSelection().removeAllRanges();
  dlg.find(src);
}

// HTML-raportti-ikkuna
function acc_html_report() {
  var report_win_id =
window.open("chrome://acc/content/report.xhtml","_blank","centerscreen,location=no,scr
ollbars=yes,menubar=yes,resizable=yes");
  window.setTimeout(acc_html_report_createcontent, 1000, opener, report_win_id);
}

// kopioi ikkunasta saadut tiedot tulostettavaksi dokumentiksi
function acc_html_report_createcontent(op, rp) {
  var body = rp.content.document.getElementById("acc_report_body");
  var tree = document.getElementById("acc_tree");
  var elem = rp.content.document.createElement("h1");
  elem.textContent = "Accessibility Report";
  body.appendChild(elem);

  for (var i=0;i<visibleData.length;i++) {
    if (tree.view.getLevel(i) == 0) {

        var h2 = document.createElementNS("http://www.w3.org/1999/xhtml","h2");
        var src = visibleData[i][0];
        h2.textContent = src;
        body.appendChild(h2)
        var ul = document.createElementNS("http://www.w3.org/1999/xhtml","ul");
        body.appendChild(ul);

        for (var j=0;j<childData[src].length;j++) {
          var li = document.createElementNS("http://www.w3.org/1999/xhtml","li");
          li.textContent = "[ " + childData[src][j][0] + " ] " + childData[src][j][1];
          ul.appendChild(li);
        }
    }
  }
}

// Pääfunktio testien ajamista varten
function acc_run_tests() {

  if (opener.content.document.location.toString().substr(0,12) == "view-source:")
    return;

  currentResultsLocation = opener.content.document.location.toString();

  acc_clear_results();
  acc_hideLayout();

  var selectedFunctions = basicTestFunctions.slice(0);

  // Haetaan visuaalisesta asemoinnista tiedot
  if (document.getElementById("acc_comparewithprevious").checked == true) {
    acc_extractLayoutMetadata();
    for (var j=0; j<layoutTestFunctions.length; j++)
      selectedFunctions.push(layoutTestFunctions[j]);
  }
```

```
  // Suoritetaan testifunktiot
   for (var i = 0; i < selectedFunctions.length; i++) {

    var func = selectedFunctions[i];
    try {
      var reportObject = eval(func + "()");
    } catch (e) {
      alert( 'Error in analysis: ' + e );
    }

    // lisätään virheet rakenteeseen
    addErrorArrayToTree(reportObject);

  }

  if (visibleData.length < 1)
    visibleData[0] = ["No errors found.", "", false, false];


  // Lisätään virheet näkyväksi
  updateTree();
}

function addErrorArrayToTree(reportObject) {
    if (reportObject != null && typeof(reportObject[0]) == "string") {
      if ((reportObject[1] instanceof Array) && reportObject[1].length > 0) {
        visibleData[visibleData.length] = [reportObject[0], " (" +
reportObject[1].length + " occurences)" , true, false];
        childData[reportObject[0]] = reportObject[1];
      } else {
        visibleData[visibleData.length] = [reportObject[0], "", false, false];
      }
    }
}

function updateTree() {
  document.getElementById("acc_tree").view = accTreeView;
}

// Antaa DOM-solmusta lähdekoodin palanen
function acc_getCode(node) {
  var elem = opener.window.content.document.createElement("p");
  var clonenode = opener.window.content.document.importNode(node, true);
  elem.appendChild(clonenode);
  return elem.innerHTML.substr(0,100);
}

// Visuaalisesta asemoinnista saatavien tietojen hakeminen
function acc_extractLayoutMetadata() {
  var doc = opener.content.document;
  var list = new Array();

  // tallennetaan edellisen sivun tiedot
  if (currentXYTree != null) {
    previousXYTreeLocation = currentXYTreeLocation;
    previousXYTree = currentXYTree;
    previousPageWidth = currentPageWidth;
    previousPageHeight = currentPageHeight;
  }

  currentXYTreeLocation =  doc.location.toString();
  currentPageWidth = doc.documentElement.scrollWidth;
  currentPageHeight = doc.documentElement.scrollHeight;

  // haetaan elementtien sijainnit ja koot
  acc_getBoundingBoxes(doc.documentElement, list);

  /* käydään rekursiivisesti läpi sivu osien asetteluissa saatuja tietoja ja jaetaan
     nämä osiin vuorottaisin X-Y-leikkauksin */
```

```
  currentXYTree = new
XYNode(0,0,currentPageWidth,currentPageHeight,ACC_BOX_CONTAINER);
  if (!(acc_divideXYNode(currentXYTree, list, false, 0,
ACC_BOXEXTRACT_RECURSION_DEPTH)))
    acc_divideXYNode(currentXYTree, list, true, 0, ACC_BOXEXTRACT_RECURSION_DEPTH);

  // haetaan leikkauksista saatujen alueiden tyyppi
  acc_findXYNodeTypes(currentXYTree, list);
  boundingboxlist = list;

}

// osa-jaon piirtäminen selainikkunaan
function acc_showLayout() {

  if (currentXYTree == null || currentXYTreeLocation !=
opener.content.document.location.toString())
    acc_extractLayoutMetadata();

  acc_drawXYNodes(currentXYTree);
  islayoutvisible = true;
}

function acc_drawXYNodes(nod) {

  if (nod.getType() != ACC_XYNODE_CONTAINER) {
    if (nod.getType() != ACC_XYNODE_EMPTY) {
      var div = opener.content.document.createElement("div");
      div.className = "acc_drawing";
      var sisadiv = opener.content.document.createElement("div");
      if (nod.getType() == ACC_XYNODE_CONTENT) {
        div.setAttribute("style","z-index:100;font-size:1em;font-weight:900;font-
family:Courier,monospace;color:blue;background-
color:transparent;position:absolute;left:" + nod.box.getXMin() + "px;width:" +
nod.box.getWidth() + "px;top:" + nod.box.getYMin() + "px;height:" +
nod.box.getHeight() + 'px;border: 1px dashed blue;' );
        sisadiv.appendChild(opener.content.document.createTextNode("CONTENT"));
        sisadiv.setAttribute("style", "width: 6em;background-color: white; color:
blue;");
        div.appendChild(sisadiv);
      }
      if (nod.getType() == ACC_XYNODE_NAVIGATION) {
        div.setAttribute("style","z-index:100;font-size:1em;font-weight:900;font-
family:Courier,monospace;color:red;background-
color:transparent;position:absolute;left:" + nod.box.getXMin() + "px;width:" +
nod.box.getWidth() + "px;top:" + nod.box.getYMin() + "px;height:" +
nod.box.getHeight() + 'px;border: 1px dashed red;' );
        sisadiv.appendChild(opener.content.document.createTextNode("NAVIGATION"));
        sisadiv.setAttribute("style", "width: 8em;background-color: white; color:
red;");
        div.appendChild(sisadiv);
      }
      if (nod.getType() == ACC_XYNODE_COMMON) {
        div.setAttribute("style","z-index:100;font-size:1em;font-weight:900;font-
family:Courier,monospace;color:red;background-
color:transparent;position:absolute;left:" + nod.box.getXMin() + "px;width:" +
nod.box.getWidth() + "px;top:" + nod.box.getYMin() + "px;height:" +
nod.box.getHeight() + 'px;border: 1px dashed black;' );
        sisadiv.appendChild(opener.content.document.createTextNode("BLOCK"));
        sisadiv.setAttribute("style", "width: 4em;background-color: white; color:
black;");
        div.appendChild(sisadiv);
      }

      opener.content.document.body.appendChild(div);
    }

  } else {
      var children = nod.getChildren();
      // piirretään osat rekursiivisesti X-Y-puusta
      for (var i=0;i<children.length;i++) {
```

```
        acc_drawXYNodes(children[i]);
      }
  }

}

function acc_hideLayout() {
  var body = opener.content.document.body;
  var bodychildren = body.childNodes;

  for (var i=0;i<bodychildren.length;i++) {
    if (bodychildren[i].className) {
      if (bodychildren[i].className == "acc_drawing") {
        body.removeChild(bodychildren[i]);
        i--;
      }
    }
  }

  islayoutvisible = false;
}

function acc_findXYNodeTypes(nod, list) {
  // käydään X-Y-leikkauspuu läpi
  var children = nod.getChildren();
  if (children != null) {
    for (var i=0;i<children.length;i++) {
      acc_findXYNodeTypes(children[i], list);
    }
    nod.setType(ACC_XYNODE_CONTAINER);
    return;
  }

  var textcount = 0;
  var bigtextnodecount = 0;
  var textlengthcount = 0;
  var anchorcount = 0;
  var formcount = 0;
  var replacedcount = 0;
  // tutkitaan minkä tyyppisiä lohkoja on ison alueen sisällä
  for (var i=0;i<list.length;i++) {
    if (list[i].isInsideBox(nod.box)) {
      var type = list[i].getType();
      if (type == ACC_BOX_TEXT_ELEM || type == ACC_BOX_TEXT_NODE) {
          textcount++;

        var contentlength = list[i].getContentLength();
        if (contentlength > 80)
          bigtextnodecount++;

        textlengthcount = textlengthcount + contentlength;
      } else if (type == ACC_BOX_REPLACED) {
          replacedcount++;
          var contentlength = list[i].getContentLength();
          textlengthcount = textlengthcount + 2*contentlength;
      } else if (type == ACC_BOX_ANCHOR)
        anchorcount++;
      else if (type == ACC_BOX_FORM)
        formcount++;
    }
  }
  // tehdään arvioita saatujen tietojen perusteella
  if (textcount < 1 && replacedcount < 1 && anchorcount < 1 && formcount < 1) {
    nod.setType(ACC_XYNODE_EMPTY);
  } else if ( ((anchorcount + formcount) / textcount) >=
ACC_BOXEXTRACT_ANCHOR_RATIO_THRESHOLD && bigtextnodecount == 0 ) {
    nod.setType(ACC_XYNODE_NAVIGATION);
  } else if ( (textcount > 1  && (textlengthcount / textcount) >
ACC_BOXEXTRACT_TEXTLENGTH_RATIO_THRESHOLD) || bigtextnodecount > 0) {
      nod.setType(ACC_XYNODE_CONTENT);
  }
```

```
}

// X-Y-leikkaukset
function acc_divideXYNode(nod, list, isvertical,currentlevel, maxlevel) {

   var boundingbox = nod.box;
  if (isvertical)
    var boxthreshold = currentPageWidth / ACC_BOXEXTRACT_MAX_COLUMNS;
  else
    var boxthreshold = currentPageHeight / ACC_BOXEXTRACT_MAX_ROWS;

  // Laatikkoa ei voida jakaa enää kahtia
  if (nod.box.getMax(isvertical) - nod.box.getMin(isvertical) < 2*boxthreshold)
    return false;

  var separatorlist = new Array();
  var cutlist = new Array();
  var widegapthreshold = acc_getCuts(cutlist, separatorlist, boundingbox, list,
isvertical);

  // Lisätään riittävän isot implisiittiset erottimet (eli tyhjä tila) erotinlistaan
  for (var i=0;i<cutlist.length;i++) {
    if (cutlist[i][1] - cutlist[i][0] >= widegapthreshold) {
      separatorlist.push(Math.round( (cutlist[i][0] + cutlist[i][1]) / 2 ));
    }
  }

  // Luodaan leikatut laatikot ja lisätään ne X-Y-puuhun
  if (separatorlist.length == 0)
    return false;

  separatorlist.sort(acc_compareValues);

  var start = boundingbox.getMin(isvertical);
  for (var k=0;k<separatorlist.length;k++) {
    if (separatorlist[k] - start > boxthreshold) {
      acc_addXYNode(nod, start, separatorlist[k], isvertical);
      start = separatorlist[k]+1;
    }
  }

  var children = nod.getChildren();

  if (children == null)
    return false;

  // viimeinen osa
  if (nod.box.getMax(isvertical) - start > boxthreshold) {
    acc_addXYNode(nod, start, nod.box.getMax(isvertical), isvertical);
  } else {
    // yhdistetään edelliseen, koska oli liian pieni
    var start = children[children.length-1].box.getMin(isvertical);
    children.splice(children.length-1,1);
    acc_addXYNode(nod, start, nod.box.getMax(isvertical), isvertical);
  }

  // jaetaan jaetut osat toiseen suuntaan
  for (var i=0;i<children.length;i++) {
    if (currentlevel < maxlevel) {
      if (!(acc_divideXYNode(children[i], list, !isvertical, currentlevel+1,
maxlevel)))
        acc_divideXYNode(children[i], list, isvertical, currentlevel+1, maxlevel);
    }
  }

  return true;

}

function acc_addXYNode(parentnode, start, end, isvertical) {
```

```
  var newnode = null;
  if (isvertical)
    var newnode = new XYNode(start, parentnode.box.getYMin(), end,
parentnode.box.getYMax(), ACC_BOX_CONTAINER);
  else
    var newnode = new XYNode(parentnode.box.getXMin(), start,
parentnode.box.getXMax(), end, ACC_BOX_CONTAINER);

  if (newnode != null)
    parentnode.addChild(newnode);
}


function acc_getCuts(cutlist, separatorlist, boundingbox, list, isvertical) {

  var boundarycoords = new Array();
  var count = 0;
  var avg = 0;

  // otetaan vain reuna-alueen sisällä olevat laatikot
  var boundingboxmin = boundingbox.getMin(isvertical);
  var boundingboxmax = boundingbox.getMax(isvertical);
  var boxmin,boxmax;

  for (var i=0;i<list.length;i++) {
    /* linkkejä ei oteta huomioon erotuksessa, mutta näiden sisällä olevat
       tekstit kylläkin */
    if (list[i].getType() == ACC_BOX_ANCHOR)
      continue;
    // kerätään alueelle osuvat eksplisiittiset erottimet omaan listaan
    if (list[i].getType() == ACC_BOX_VERTICAL_SEPARATOR) {
        if (isvertical && list[i].splitsBox(boundingbox, isvertical)) {
            var separatorpoint = list[i].getXMin();
            for (var k=0;k<separatorlist.length;k++) {
              if ((separatorlist[k]-1) <= separatorpoint && separatorpoint <=
(separatorlist[k]+1))
                break;
            }
            if (k==separatorlist.length)
              separatorlist[k] = separatorpoint;
        }

    } else if (list[i].getType() == ACC_BOX_HORIZONTAL_SEPARATOR) {
        if (!isvertical && list[i].splitsBox(boundingbox, isvertical)) {
            var separatorpoint = list[i].getYMin();
            for (var k=0;k<separatorlist.length;k++) {
              if ((separatorlist[k]-1) <= separatorpoint && separatorpoint <=
(separatorlist[k]+1))
                break;
            }
            if (k==separatorlist.length)
              separatorlist[k] = separatorpoint;
        }

    } else if (list[i].isInsideBox(boundingbox)) {

          // muutoin kerätään muut alueella olevat laatikot
          boundarycoords.push([list[i].getMin(isvertical),
list[i].getMax(isvertical)]);

          // lasketaan keskimääräinen fonttikoko, tällä estetään tarpeettomat
pystysuuntaiset erotukset
          if (list[i].getType() == ACC_BOX_TEXT_ELEM) {
            avg += acc_getCSSPx(list[i].getNode(),'font-size');
            count++;
          }
    }
  }

  if (count > 0)
    var boxfontsize = Math.round(avg/count);
```

```
  var gapstart = boundingboxmin, gapend = boundingboxmax;
  var threshold = 4;
  var cend = boundingboxmin;
  boundarycoords.sort(acc_compareFirstValues);
  var gapsize = 0;

  // haetaan tyhjät välit
  if (boundarycoords.length > 1) {

    gapsize = boundarycoords[0][0] - boundingboxmin;
    if (gapsize > 0) {
      cutlist.push( [boundingboxmin, boundarycoords[0][0]] );
    }
    // leikkaukset
    cend = boundarycoords[0][1];
    for (var j=1;j<boundarycoords.length;j++) {
      if (boundarycoords[j][0] > cend) {
        var gapstart = cend + 1;
        var gapend = boundarycoords[j][0]-1;
        gapsize = gapend - gapstart;
        if (gapsize > 0) {
          cutlist.push( [gapstart, gapend] );
          if (threshold < gapsize) threshold = gapsize;
        }
        cend = boundarycoords[j][1];
      } else {
        if (boundarycoords[j][1] > cend)
          cend = boundarycoords[j][1];
      }
    }

    gapsize = boundingboxmax - cend;
    if (gapsize > 0) {
      cutlist.push( [cend+1, boundingboxmax] );
    }

  }

  // pystysuunnassa ei saa erottaa tekstirivien väleistä
  if (!isvertical && threshold < boxfontsize)
    threshold = boxfontsize;

  return threshold;

}

/* haetaan DOM-elementtien asemointi-informaatio suhteessa ikkunaan ja
   merkitään ylös erotinlohkojen reunat */
function acc_getBoundingBoxes(elem, list) {

if (elem.nodeType != Node.COMMENT_NODE) {

  if (elem.nodeType == Node.TEXT_NODE && is_not_all_ws(elem)) {
    parent = elem.parentNode;
    if (!(parent.tagName == "SCRIPT" || parent.tagName == "NOSCRIPT" || parent.tagName
== "TITLE" ||
        parent.offsetWidth == 0 || parent.offsetHeight == 0))
    {
      if
(parent.tagName.match("P|A|H1|H2|H3|H4|H5|H6|FONT|B|BIG|SMALL|STRONG|EM|SPAN")) {
        var xy = acc_getXY(parent);

        list[list.length] = new RefBox(xy[0], xy[1], xy[0] + parent.offsetWidth, xy[1]
+ parent.offsetHeight, ACC_BOX_TEXT_ELEM, parent.textContent.length, parent);

      } else {
        /* jos kyseessä on pelkkä tekstisolmu, niin se on suljettava jonkin
           elementin sisälle kokoinformaation saamiseksi */
        var span = opener.content.document.createElement("span");
        span.style.position = "relative";
```

```
        span.appendChild(opener.content.document.createTextNode(elem.nodeValue));
        parent.replaceChild(span, elem);

        var xy = acc_getXY(span);
        if (span.offsetWidth && span.offsetHeight) {
          if (span.offsetWidth > 0 && span.offsetHeight > 0) {

            list[list.length] = new ContentBox(xy[0], xy[1], xy[0] + span.offsetWidth,
xy[1] + span.offsetHeight, ACC_BOX_TEXT_NODE, span.textContent.length);
          }
        }

        parent.replaceChild(elem, span);
      }
    }
  }

  if (elem.nodeType == Node.ELEMENT_NODE) {
    var nname = elem.nodeName;

    if (nname == "A") {
      var xy = acc_getXY(elem);
      list[list.length] = new RefBox(xy[0], xy[1], xy[0] + elem.offsetWidth, xy[1] +
elem.offsetHeight, ACC_BOX_ANCHOR, elem.textContent.length, elem);
    }

    // luodaan riittävän isoja reunoja joiden kohdalta voi halkaista laatikkoja kahtia
    if (nname == "HR") {
      var suhde = elem.offsetWidth / currentPageWidth;
      if (ACC_BOXEXTRACT_X_SEPARATOR_THRESHOLD < suhde && suhde < 1.0) {
        xy = acc_getXY(elem);
        list[list.length] = new Box(xy[0],xy[1], xy[0]+elem.offsetWidth, xy[1],
ACC_BOX_HORIZONTAL_SEPARATOR);
      }
    }
    if (nname == "IMG") {
      // pitkä kapea pystysuuntainen erotinkaistale
      if (elem.offsetWidth < 3) {
        var suhde = elem.offsetHeight / currentPageHeight;
        if (ACC_BOXEXTRACT_Y_SEPARATOR_THRESHOLD < suhde && suhde < 1.0) {
          xy = acc_getXY(elem);
          list[list.length] = new Box(xy[0],xy[1], xy[0], xy[1]+elem.offsetHeight,
ACC_BOX_VERTICAL_SEPARATOR);
        }
      }
      // pitkä vaakasuuntainen erotin
      if (elem.offsetHeight < 3) {
        var suhde = elem.offsetWidth / currentPageWidth;
        if (ACC_BOXEXTRACT_X_SEPARATOR_THRESHOLD < suhde && suhde < 1.0) {
          xy = acc_getXY(elem);
          list[list.length] = new Box(xy[0],xy[1], xy[0]+elem.offsetWidth, xy[1],
ACC_BOX_HORIZONTAL_SEPARATOR);
        }
      }
    }
    // näiden lohkojen reunat ovat eksplisiittisiä erottimia
    if (nname.match("TABLE|TD|TR|DIV|TBODY|THEAD|TFOOT|COL|COLGROUP")) {

      var suhde = elem.offsetWidth / currentPageWidth;
      if (ACC_BOXEXTRACT_X_SEPARATOR_THRESHOLD < suhde && suhde < 1.0) {
        // ei oteta kuitenkaan sisältöosia erottavia lohkoja
        if (suhde > 0.6 || elem.parentNode.textContent.length < 20) {

          xy = acc_getXY(elem);
          list[list.length] = new Box(xy[0], xy[1], xy[0]+elem.offsetWidth, xy[1],
ACC_BOX_HORIZONTAL_SEPARATOR);
          list[list.length] = new Box(xy[0], xy[1]+elem.offsetHeight,
xy[0]+elem.offsetWidth, xy[1]+elem.offsetHeight, ACC_BOX_HORIZONTAL_SEPARATOR);
        }
      }
      var suhde = elem.offsetHeight / currentPageHeight;
```

```
      if (ACC_BOXEXTRACT_Y_SEPARATOR_THRESHOLD < suhde && suhde < 1.0) {
        xy = acc_getXY(elem);
        list[list.length] = new Box(xy[0],xy[1], xy[0], xy[1]+elem.offsetHeight,
ACC_BOX_VERTICAL_SEPARATOR);
        list[list.length] = new Box(xy[0]+elem.offsetWidth, xy[1],
xy[0]+elem.offsetWidth, xy[1]+elem.offsetHeight, ACC_BOX_VERTICAL_SEPARATOR);
      }

    }

    if (nname.match("IMG|OBJECT|INPUT|TEXTAREA|SELECT")) {
      xy = acc_getXY(elem);
      if (elem.offsetWidth && elem.offsetHeight) {
        // poistetaan spacer-kuvat
        if (elem.offsetWidth > 1 && elem.offsetHeight > 1) {

            if (nname == "IMG") {
              var type = ACC_BOX_REPLACED;

              if (elem.hasAttribute("alt")) {
                list[list.length] = new ContentBox(xy[0], xy[1], xy[0] +
elem.offsetWidth, xy[1] + elem.offsetHeight, type, elem.getAttribute("alt").length);
              } else
                list[list.length] = new ContentBox(xy[0], xy[1], xy[0] +
elem.offsetWidth, xy[1] + elem.offsetHeight, type, 0);
            } else if (nname == "OBJECT") {
              var type = ACC_BOX_REPLACED;
              list[list.length] = new ContentBox(xy[0], xy[1], xy[0] +
elem.offsetWidth, xy[1] + elem.offsetHeight, type, elem.textContent.length);
            } else {
              var type = ACC_BOX_FORM;
              list[list.length] = new Box(xy[0], xy[1], xy[0] + elem.offsetWidth,
xy[1] + elem.offsetHeight, type);
            }
        }
      }
    }
  }
}

// käydään rekursiivisesti läpi kaikki solmut
if (elem.hasChildNodes())
  acc_getBoundingBoxes(elem.firstChild, list);
if (elem.nextSibling)
  acc_getBoundingBoxes(elem.nextSibling, list);

}


// Apufunktioita

function acc_getCSSPx(node, cssproperty) {
  // 5 = CSS_PX
  var val =
opener.content.document.defaultView.getComputedStyle(node,null).getPropertyCSSValue(cs
sproperty).getFloatValue(5);
  return val;
}

function is_not_all_ws( nod )
{
  return (/[^\t\n\r ]/.test(nod.data));
}

function acc_getXY(obj)
{
  var posleft = 0;
  var postop = 0;
  if (obj.offsetParent)
  {
    while (obj.offsetParent)
```

```
    {
      posleft += obj.offsetLeft
      postop += obj.offsetTop;
      obj = obj.offsetParent;
    }
  }
  return [posleft,postop];
}

function acc_compareFirstValues(a,b) {
  return (a[0] - b[0]);
}

function acc_compareValues(a,b) {
  return (a - b);
}
```

Liite 3. Saavutettavuuden arvioinnissa käytettävät funktiot

```
/* rules.js - Jukka Mäntylä 31.7.2006 */
//Testifunktioryhmät
var basicTestFunctions = ["acc_imgWithoutAlt", "acc_imgWithAltText",
"acc_inputImageWithoutAlt","acc_openNewWindow","acc_metaRefresh","acc_htmlLang","acc_p
ageTitle",
"acc_depracatedElements","acc_optionGroups", "acc_fieldsetGroups",
"acc_fieldsetLegend", "acc_connectedLabels"];
var layoutTestFunctions = ["acc_navigationConsistency", "acc_skipToContentLink",
"acc_commonLayoutScalability",
"acc_nestedTables"];

// Funktioita saavutettavuuden testaamiseen

function acc_XPathMatch(xpathexpression, errormessage, suggestmessage) {
  var iterator = opener.content.document.evaluate(xpathexpression,
opener.content.document, null, XPathResult.ORDERED_NODE_ITERATOR_TYPE, null );
  var errors = new Array();
  var node = iterator.iterateNext();

  while(node) {
    errors[errors.length] = [suggestmessage, acc_getCode(node)];
    node = iterator.iterateNext();
  }

  if (errors.length > 0) {
    return [errormessage, errors];
  } else
    return null;
}

// Esimerkkejä XPath-lausekkeiden avulla tehtävistä tarkistuksista
function acc_imgWithoutAlt() {
  return acc_XPathMatch("//img[not(@alt)]","Required image alt-attribute
missing.","Add descriptive alt-text.");
}
function acc_inputImageWithoutAlt() {
  return acc_XPathMatch('//input[@type="image"][not(@alt)]','Required image-button
alt-attribute missing.','Add descriptive alt-text.');
}
function acc_metaRefresh() {
  return acc_XPathMatch('//meta[@http-equiv="refresh"]','Page refresh in
periods.','Remove auto-refresh meta-tag.');
}
function acc_openNewWindow() {
  return acc_XPathMatch('//*[@target="_blank"]','Link opens a new window.','Avoid
using attribute target.');
}


function acc_depracatedElements() {
  var depracatedlist = ["b", "i", "blink", "marquee", "embed", "u", "font",
"fontbase", "dir", "menu", "center", "strike"];

  var errors = new Array();
  for (var j=0;j<depracatedlist.length;j++) {
    var iterator = opener.content.document.evaluate("//"+depracatedlist[j],
opener.content.document, null, XPathResult.ORDERED_NODE_ITERATOR_TYPE, null );
    var node = iterator.iterateNext();

    while(node) {
      if (j < 2)
        errors[errors.length] = ["Use strong and em.", acc_getCode(node)];
      else if (j < 5)
        errors[errors.length] = ["Remove non-standard elements.", acc_getCode(node)];
      else
```

```
        errors[errors.length] = ["Use structural elements and CSS.",
acc_getCode(node)];
      node = iterator.iterateNext();
    }
  }

  if (errors.length > 0) {
    return ["Avoid presentational and depracated elements.", errors];
  } else
    return null;
}


/* DOM-rajapinnasta saatavien attribuuttien ja XPath-prosessoinnin avulla tehtäviä
tarkasteluja */

function acc_htmlLang() {
  var iterator = opener.content.document.evaluate("//html", opener.content.document,
null, XPathResult.ORDERED_NODE_ITERATOR_TYPE, null );
  var node = iterator.iterateNext();
  var errors = new Array();

  if (node == null) {
    errors[errors.length] = ["Document element is missing.", "<html>"];
  } else if (! (node.hasAttribute("xml:lang") || node.hasAttribute("lang")) ) {
    errors[errors.length] = ["Add attribute xml:lang or lang.", acc_getCode(node)];
  } else
    return null;

  return ["Primary language is not identified.", errors];
}

function acc_pageTitle() {
  var iterator = opener.content.document.evaluate("//title", opener.content.document,
null, XPathResult.ORDERED_NODE_ITERATOR_TYPE, null );
  var node = iterator.iterateNext();
  var errors = new Array();

  if (node == null) {
    errors[errors.length] = ["Add title.", "<head>"];
  } else if (is_not_all_ws(node.firstChild))
    return null;
  else
    errors[errors.length] = ["Add descriptive title.", "<head>"];
  return ["Page title is missing.", errors];
}

function acc_imgWithAltText() {

  var nodesSnapshot = opener.content.document.evaluate('//img[@alt]',
opener.content.document, null, XPathResult.ORDERED_NODE_SNAPSHOT_TYPE, null );
  var errors = new Array();

  for ( var i=0; i < nodesSnapshot.snapshotLength; i++ ) {
    var node = nodesSnapshot.snapshotItem(i);
    var alt = node.getAttribute("alt");
    var src = node.getAttribute("src");

    if (alt.length > 150) {
      errors[errors.length] = ['Alt-text too long. Use title.', acc_getCode(node)];
    }

    if (alt.length > 0 && ( node.offsetWidth < 2 || node.offsetHeight < 2) ) {
      errors[errors.length] = ['Use alt="" for spacer images.', acc_getCode(node)];
    }
  }

  if (errors.length > 0)
    return ["Suspicious alt-attribute values.", errors];
  else
    return null;
```

```
}

// Lomakkeiden saavutettavuustarkistuksia

function acc_optionGroups() {
  /* etsitään select-elementit, joissa on yli 5 option-elementtiä, mutta ei
ryhmittelyä */
  var iterator = opener.content.document.evaluate("//select[not(child::optgroup) and
count(child::option) > 5]", opener.content.document, null,
XPathResult.ORDERED_NODE_ITERATOR_TYPE, null );
  var node = iterator.iterateNext();
  var errors = new Array();
  while (node) {
     errors[errors.length] = ["Group multiple options using optgroup.",
acc_getCode(node)];
     node = iterator.iterateNext();
  }
  if (errors.length > 0)
    return ["Option elements are not grouped.", errors];
  else
    return null;
}

function acc_fieldsetGroups() {
  // kerätään lomake-elementit, jotka eivät ole fieldset-lohkon sisällä
  var iterator =
opener.content.document.evaluate("//form[(count(descendant::input[not(ancestor::fields
et)]) + count(descendant::select[not(ancestor::fieldset)]) +
count(descendant::textarea[not(ancestor::fieldset)])) > 5]", opener.content.document,
null, XPathResult.ORDERED_NODE_ITERATOR_TYPE, null );
  var node = iterator.iterateNext();
  var errors = new Array();
  while (node) {
     errors[errors.length] = ["Use fieldset to group form elements.",
acc_getCode(node)];
     node = iterator.iterateNext();
  }
  if (errors.length > 0)
    return ["Form elements are not grouped.", errors];
  else
    return null;

}

function acc_fieldsetLegend() {
  // haetaan fieldset-elementit, joiden sisällä ei ole legend-elementtiä
  var iterator = opener.content.document.evaluate("//fieldset[not(child::legend)]",
opener.content.document, null, XPathResult.ORDERED_NODE_ITERATOR_TYPE, null );
  var node = iterator.iterateNext();
  var errors = new Array();
  while (node) {
     errors[errors.length] = ["Add legend to name fieldset.", acc_getCode(node)];
     node = iterator.iterateNext();
  }
  if (errors.length > 0)
    return ["Fieldset has no legend.", errors];
  else
    return null;
}

function acc_connectedLabels() {
  // haetaan label-elementit, joilla on for-attribuutti
  var nodesSnapshot = opener.content.document.evaluate('//label[@for]',
opener.content.document, null, XPathResult.UNORDERED_NODE_SNAPSHOT_TYPE, null );

  var errors = new Array();
  var formelements = ["input", "textarea", "select"];

  // haetaan lomake-elementit, jotka eivät ole label-elementin sisällä
  for (var j=0;j<formelements.length;j++) {
```

```
    var iterator = opener.content.document.evaluate('//' + formelements[j] +
'[not(ancestor::label)][not(@type="hidden")]', opener.content.document, null,
XPathResult.ORDERED_NODE_ITERATOR_TYPE, null );
    var node = iterator.iterateNext();
    while(node) {
      var found = false;
      /* kullakin tällaisella elementillä on oltava id, johon
         jonkin label-elementin for-attribuutti viittaa */
      if (node.hasAttribute("id")) {
        var search_id = node.getAttribute("id");
        for ( var i=0; i < nodesSnapshot.snapshotLength; i++ ) {
          var labelnode = nodesSnapshot.snapshotItem(i);
          var for_id = labelnode.getAttribute("for");
          if (for_id == search_id) {
            found = true;
            break;
          }
        }
      }
      if (!found) {
        errors[errors.length] = ['Add and connect label with ' + formelements[j] +
'.', acc_getCode(node)];
      }
      node = iterator.iterateNext();
    }
  }

  if (errors.length > 0)
    return ["Form elements without label.", errors];
  else
    return null;
}

// Sivun asetteluun liittyviä saavutettavuustarkistuksia

function acc_skipToContentLink() {
  // Etsitään pääsisältöalueet
   var contentnodes = currentXYTree.getLeafChildrenOfType(ACC_XYNODE_CONTENT);

   var startindex = 0;
   if (contentnodes.length == 0)
     return null; // sisältöalueita ei löytynyt
   else if (contentnodes.length == 1)
     startindex = 0;
   else if (contentnodes.length == 2)
     startindex = 1;
   else
     startindex = Math.round(contentnodes.length / 2);

   // otetaan sisältöalueista isoimmat
   var sizeorderedcontentnodes = new Array();

   for (var i=0;i<contentnodes.length;i++) {
     sizeorderedcontentnodes[sizeorderedcontentnodes.length] = [
contentnodes[i].box.getWidth() * contentnodes[i].box.getHeight(), contentnodes[i] ];
   }

   sizeorderedcontentnodes.sort(acc_compareFirstValues);

  /* jos DOM-puun alkupuolella olevat tekstisolmut pääsisältöalueiden sisällä,
     niin hyppylinkkejä ei tarvita */
  var textnodecount = 0;
  var i = 0;
  var found = false;

  while (textnodecount < 20 && i < boundingboxlist.length && found == false) {
     if (boundingboxlist[i].getType() == ACC_BOX_TEXT_NODE ||
boundingboxlist[i].getType() == ACC_BOX_TEXT_ELEM) {
       // tarkistetaan onko tekstilohko pääsisältöalueen sisällä
       for (var j=startindex;j<sizeorderedcontentnodes.length;j++) {
         if ( boundingboxlist[i].isInsideBox( (sizeorderedcontentnodes[j][1]).box )) {
```

```
            found = true;
            break;
          }
        }
        textnodecount++;
      }
      i++;
  }

  if (found)
    return null;


  var errors = new Array();
  found = false;
  var i=0;
  var count = 0;
  /* otetaan muutama ensimmäinen linkki DOM-puusta saadussa järjestyksessä
     jos linkki vie pääsisältöalueelle, niin navigoinnin ohittava hyppylinkki on */
löydetty
  for (var i=0;i<boundingboxlist.length;i++) {
    if (found || count > 5)
      break;

    if (boundingboxlist[i].getType() == ACC_BOX_ANCHOR) {
      count++;
      if (boundingboxlist[i].getNode().hasAttribute("href")) {
        var href = boundingboxlist[i].getNode().getAttribute("href");
        // sisäinen linkki
        var risuindex = href.indexOf("#");
        if (risuindex != -1) {
          href = href.substr(risuindex+1);
          // löytyykö kohde id- tai name-ominaisuuden avulla
          var referredelem = opener.content.document.getElementById(href);
          if (referredelem == null) {
            var nodelist = opener.content.document.getElementsByName(href);
            if (nodelist.length > 0) {
              referredelem = nodelist[0];
            } else {
              errors[errors.length] = ["Internal link #" + href + " leads nowhere.",
acc_getCode(boundingboxlist[i].getNode()) ];
              continue;
            }
          }
          // haetaan kohteen määräämä alue
          var xy = acc_getXY(referredelem);
          if (referredelem.offsetWidth && referredelem.offsetHeight) {
            var testbox = new Box(xy[0],xy[1], xy[0]+referredelem.offsetWidth,
xy[1]+referredelem.offsetHeight, ACC_BOX_COMMON);
          } else {
            var testbox = new Box(xy[0],xy[1], xy[0], xy[1], ACC_BOX_COMMON);
          }
          // onko alue pääsisältöosan sisällä
          for (var j=startindex;j<sizeorderedcontentnodes.length;j++) {
            if ( testbox.isInsideBox( (sizeorderedcontentnodes[j][1]).box ) ) {
              found = true;
              continue;
            }
          }
        }
      }
    }
  }

  if (found)
    return null;
  else {
    if (errors.length == 0)
      if (opener.content.document.body.firstChild)
        errors[errors.length] = ['Add "skip to main content" link.', ""];
      else
```

```
        errors[errors.length] = ['Add "skip to main content" link.',
acc_getCode(opener.content.document.body.firstChild)];
    return ["Content is not reachable serially.", errors];
  }

}

function acc_commonLayoutScalability() {

  // tapahtumankäsittelijä ikkunan koon muutokselle
  function acc_scalabilityTestOnResize() {
    if (opener.outerWidth < 650 && !lock) {
      // tarkistetaan tuleeko vaakasuuntainen vierityspalkki
      if (opener.content.document.defaultView.scrollMaxX > 20) {
        var errors = new Array();
        errors[0] = ["Use % units for content width",""]
        var treeobj = ["Page is not scalable.", errors];
        // lisätään virheilmoitus erikseen ja päivitetään virhelista
        addErrorArrayToTree(treeobj);
        updateTree();
      }
      lock = true;
    }
  }


  function acc_backToOriginal() {
    opener.removeEventListener("resize", acc_scalabilityTestOnResize, false);
    opener.resizeTo(currentwindowwidth, currentwindowheight);
    opener.moveTo(currentX, currentY);

  }

  var currentwindowwidth = opener.outerWidth;
  var currentwindowheight = opener.outerHeight;
  var currentX = opener.screen.left;
  var currentY = opener.screen.top;
  var errors = new Array();
  var lock = false;

  opener.addEventListener("resize", acc_scalabilityTestOnResize, false);
  opener.resizeTo(640, currentwindowheight);
  window.setTimeout(acc_backToOriginal, 500);
}

function acc_nestedTables() {
  /* funktiolla tutkitaan onko taulukon sisällä rakenteellisia elementtejä =>
     ei ole datataulukko */
  function acc_getStructuralElementCount(elem) {
    var omacount = 0;
    if (elem.nodeType == Node.ELEMENT_NODE) {
      if
(elem.tagName.match("UL|OL|LI|H1|H2|H3|H4|H5|H6|FORM|INPUT|TEXTAREA|SELECT|IMG|OBJECT|
HR"))
        omacount++;
    }
    if (elem.hasChildNodes())
      omacount += acc_getStructuralElementCount(elem.firstChild);
    if (elem.nextSibling)
      omacount += acc_getStructuralElementCount(elem.nextSibling);
    return omacount;
  }

  // funktiolla haetaan elementin edeltäjien table-elementtien määrä
  function acc_getAncestorTableCount(elem) {
    var tablecount = 0;
    var tempnode = elem;
    var docnode = opener.content.document.documentElement;
    while (tempnode.parentNode && tempnode != docnode) {
      tempnode = tempnode.parentNode;
      if (tempnode.tagName)
```

```
        if (tempnode.tagName == "TABLE")
          tablecount++;
    }
    return tablecount;
  }

  var errors = new Array();
  var tablearray =
opener.content.document.documentElement.getElementsByTagName("TABLE");

  for (var i=0;i<tablearray.length;i++) {
    var node = tablearray[i];

    /* tutkitaan vain kolmannen tason taulukoita, koska jokaisesta
       sisäkkäisestä taulukosta ei kannata valittaa */
    if (acc_getAncestorTableCount(node) != 2)
      continue;

    // löytyykö tämän table-elementin sisältä neljännen tason taulukoita
    var tables = node.getElementsByTagName("TABLE");
    if (tables != null && tables.length > 0) {
      errors[errors.length] = ["Use few tables or CSS.", acc_getCode(node)];
      continue;
    }

    // jos on otsikoita, niin on luultavasti datataulukko
    var ths = node.getElementsByTagName("TH");
    if (ths != null && ths.length > 0)
      continue;

     var isdatatable = true;
     var count = 0;
     /* jos sisältää div-lohkoja, jotka eivät ole pelkästään erottimia,
        niin ei ole datataulukko */
     var divs = node.getElementsByTagName("DIV");
     for (var j=0;j<divs.length;j++) {
        if (divs[j].offsetWidth && divs[j].offsetHeight)
          if (divs[j].offsetWidth > 10 && divs[j].offsetHeight > 10)
            count++;
     }
     if (count >= 5)
       isdatatable = false;
     else {

       var tds = node.getElementsByTagName("TD");
       // datataulukossa pitäisi olla ainakin muutama solu
       if (tds.length < 4) {
         isdatatable = false;
       } else {
         var failcount = 0;
         for (var k=0;k<tds.length;k++) {
           // jos solu sisältää paljon tekstiä, niin ei ole datataulukko
           if (tds[k].textContent.length > 150)
             failcount += 5;
           if (tds[k].hasChildNodes())
             failcount += acc_getStructuralElementCount(tds[k].firstChild);
         }
         if (failcount / tds.length > 10.0)
           isdatatable = false;
       }
     }

     if (!isdatatable)
       errors[errors.length] = ["Use few tables or CSS.", acc_getCode(node)];
  }

  if (errors.length > 0)
    return ["Deeply nested layout tables.", errors];
  else
    return null;
}
```

```
/* Vertaillaan edellisen sivun taittoa nykyiseen ja tutkitaan ovatko isot
   navigointilohkot samassa paikassa. Funktion pitäisi vielä tarkistaa, että
   ollaan samalla domain-alueella kuin edellisessä sivussa. */
function acc_navigationConsistency() {

   function isInside(innerbox, outerbox, xthreshold, ythreshold) {
      return (outerbox.getXMin()-xthreshold <= innerbox.getXMin() &&
innerbox.getXMax() <= outerbox.getXMax()+ythreshold &&
              outerbox.getYMin()-ythreshold <= innerbox.getYMin() &&
innerbox.getYMax() <= outerbox.getYMax()+ythreshold);
   }

   function isInsideReverseYCoord(innerbox, outerbox, xthreshold, ythreshold) {
      return (outerbox.getXMin()-xthreshold <= innerbox.getXMin() &&
innerbox.getXMax() <= outerbox.getXMax()+ythreshold &&
              (currentPageHeight-outerbox.getYMax()) -ythreshold <=
(previousPageHeight - innerbox.getYMax()) && (previousPageHeight - innerbox.getYMin())
<= (currentPageHeight - outerbox.getYMin()) +ythreshold);
   }

   function acc_findSimilar(previousnavnode) {
     for (var j=0;j<currentnavnodes.length;j++) {
       /* sallitaan pienet x:n muutokset ja isommat y:n muutokset
          (esim. sivun ylälaidan mainosten takia)
          jos on sivun alalaidassa niin tällöin on tarkasteltava y-suunnassa
          käänteisiä koordinaatteja, sillä sivujen pituudet voivat olla erilaisia */
       if (isInside(previousnavnode.box,currentnavnodes[j].box,10,50) ||
           isInsideReverseYCoord(previousnavnode.box,currentnavnodes[j].box,10,50)) {
         return true;
       }
     }
     errors.push( ["Block missing at [(" + previousnavnode.box.getXMin() + "," +
previousnavnode.box.getYMin() + "),(" + previousnavnode.box.getXMax() + "," +
previousnavnode.box.getYMax() + ")]", ""] )
     return false;
   }

   var errors = new Array();
   // ei edellistä, testattua sivua
   if (currentXYTree == null || previousXYTree == null || currentXYTreeLocation ==
previousXYTreeLocation)
     return null;

   // poistetaan pienet navigoinnit
   var previousnavnodes = previousXYTree.getLeafChildrenOfType(ACC_XYNODE_NAVIGATION);
   var currentnavnodes = currentXYTree.getLeafChildrenOfType(ACC_XYNODE_NAVIGATION);

   var bignavnodes = new Array();
   for (var i=0;i<previousnavnodes.length;i++) {

     if ( ((previousnavnodes[i].box.getWidth() / previousPageWidth) > 0.5 ||
          (previousnavnodes[i].box.getHeight() / previousPageHeight) > 0.3) &&
           previousnavnodes[i].box.getYMax() < currentPageHeight) {
       bignavnodes.push(previousnavnodes[i]);

     }
   }

   var ok = 0;
   for (var i=0;i<bignavnodes.length;i++) {
     if (acc_findSimilar(bignavnodes[i]))
       ok++;
   }
   // valitetaan vain jos virheitä on enemmän kuin oikeasta paikkaa löytyneitä
   if (ok < errors.length) {
     return ["Navigation inconsistency with previous page.", errors];
   } else
     return null;
}
```

Liite 4. Laajennoksen käyttöliittymän kuvaileva XUL-dokumentti

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css" ?>
<!DOCTYPE window>
<window id="accWindow" title="Evaluation Report" width="400" height="600"
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

  <script type="application/x-javascript" src="rules.js"/>
  <script type="application/x-javascript" src="acc.js"/>

  <toolbox flex="0">
  <menubar id="acc_menubar">
    <menu id="acc_menu_file" label="File" accesskey="F">
      <menupopup id="acc_menu_file_popup" >
        <menuitem label="Export to HTML" accesskey="H"
oncommand="acc_html_report();"/>
        <menuseparator/>
        <menuitem label="Exit" accesskey="X" oncommand="window.close();"/>
      </menupopup>
    </menu>
    <menu id="acc_menu_edit" label="Edit"  accesskey="E">
      <menupopup id="acc_menu_edit_popup">
        <menuitem label="Find Code" accesskey="F" oncommand="acc_find_source();"  />
        <menuitem label="Clear Results" accesskey="C"
oncommand="acc_clear_results();"/>
      </menupopup>

    </menu>
  </menubar>
  </toolbox>

  <tree flex="1" id="acc_tree" hidecolumnpicker="true" seltype="single">
    <treecols>
    <treecol id="acc_warnings" label="Warnings" flex="2" primary="true"/>
    <splitter class="tree-splitter"/>
    <treecol id="acc_snippets" label="Code" flex="1"/>
    </treecols>
    <treechildren />
  </tree>

 <hbox>
  <button label="Run Check" oncommand="acc_run_tests();" accesskey="R"/>
  <button id="acc_showlayout" label="View Layout" accesskey="V"
oncommand="toggleShowLayout(event);" />

  <vbox>
  <checkbox id="acc_autocheck" label="AutoCheck on Load" checked="false" accesskey="A"
oncommand="toggleChangeListener(event, this);" />
  <checkbox id="acc_comparewithprevious" label="Layout Accessibility" accesskey="L"
checked="true" oncommand="toggleCompareWithPrevious(event,this)"/>
  </vbox>
  </hbox>
</window>
```