LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

CSS-EDITORI

LAHDEN AMMATTIKORKEAKOULU
Tekniikan ala
Mediatekniikka
Tekninen visualisointi
Opinnäytetyö
31.5.2012
Erno Johansson

Lahden ammattikorkeakoulu
Mediatekniikka

JOHANSSON, ERNO:

CSS-Editori
Cascadin Style sheets editor

Teknisen visualisoinnin opinnäytetyö, 62 sivua, 0 liitesivua

Kevät 2013

TIIVISTELMÄ

Opinnäytetyössä käsitellään CSS 3-tyylimäärittelyn uusia ominaisuuksia, CSS-editorien toimivuutta ja käyttöliittymäsuunnittelua. Työn jälkimmäisessä vaiheessa suunnitellaan ja toteutetaan CSS-editori.

Teoriaosuudessa käsitellään yleisesti CSS-tyylimäärittelyjä ja tarkemmin CSS 3:n moduuleja ja niiden mukana tulleita uusia ominaisuuksia. Samalla tarkastellaan, mitä niillä voi tehdä ja millainen vaikutus niillä on nettisivujen koodaukseen. Työssä testataan myös tarjolla olevia CSS-editoreja ja sitä, mitä niillä pystyy tekemään. Lisäksi perehdytään tarkemmin käyttöliittymäsuunnitteluun käytettävyyden, suunnittelun ja testaamisen näkökulmista.

Opinnäytetyön case-osuudessa perehdytään oman CSS-editorin toteuttamiseen. Ohjelmalle asetetaan tavoitteet ja tehdään rakenne ja ulkoasusuunnitelma, joiden pohjalta tehdään käyttäjätestaus prototyypin avulla. Saadun informaation pohjalta ohjelmoitiin lopullinen työpöytäsovellus.

Asiasanat: CSS 3, CSS-editori, käyttöliittymäsuunnittelu, käytettävyys, käyttäjätestaus

Lahti University of Applied Sciences
Faculty of Technology

JOHANSSON, ERNO:

CSS-Editor
Cascadin Style sheets editor

Bachelor´s Thesis in Vizualition Engineering, 62 pages

Spring 2013

ABSTRACT

The thesis evaluates the new features of the CSS 3 styling specifications, the functionality of the CSS-editors and its user interface designing. The later part of the thesis includes designing and manufacturing an operational CSS-editor.

The theoretical part of the thesis comprises of the general CSS styling specifications, and the specific modules of CSS 3 and its features regarding their utililization for coding websites. It also studies the functionality of the available CSS-editors and takes a closer examination of the user interface designing from the viewpoint of accessability, designing and testing.

The case part of the thesis manufactures an operational CSS editor. The program is given specific goals, a structure and a layout. Based on these specifications a prototype was built for user testing. The final desktop application was programmed with the feedback from the test users.

Keywords: CSS 3, Cascadin Style sheets, User interface design, Usability, User testing

SISÄLLYS

1 JOHDANTO

Tämä työ kertoo CSS-editorin luomisesta ja sen toteuttamiseen tarvittavasta informaatiosta. Teoriaosuudessa perehdytään CSS tyylitiedoston toimintaan ja kehitykseen sekä tutustutaan tarkemmin CSS 3:n uusiin ominaisuuksiin. Tällä hetkellä markkinoilla olevia editoreita testataan ja selvitetään, millaisia toimintoja ne sisältävät. Lisäksi selvitetään hyvään käytettävyyteen vaikuttavia tekijöitä ja apuvälineitä.

CSS 3:ssa on monia uudistuksia verrattuna aikaisempiin versioihin, sen rakennekin on muuttunut ja se muodostuu moduuleista, joita voidaan kehittää toisistaan riippumatta, mikä nopeuttaa ja helpottaa uusien tyylimäärittelyjen kehitystä. CSS 3 toi ilmestyessään mukanaan myös paljon uusia ominaisuuksia nettisivujen ulkoasun muokkaamiseen ja osa niistä on todella uudistuksellisia. Suurimpia uusia ominaisuuksia ovat elementtien kiertäminen, koon muuttaminen, vääristäminen ja muu vastaavanlainen muokkaaminen, niin kaksi- kuin kolmiulotteisessakin ympäristössä. Toinen paljon mielenkiintoa aiheuttanut uudistus on mahdollisuus elementtien animoimiseen ja liikkeen tuominen muuten staattisille verkkosivuille.

Opinnäytetyössä tutustutaan moniin erilaisiin ilmaisiin ja maksullisiin CSS-editoreihin ja niiden ominaisuuksiin. Osa editoreista on todella monipuolisia ja laajoja ja toiset ovat taas suppeampia ja helppokäyttöisempiä. Editoreissa on kahta eri käyttöliittymätyyppiä, tekstieditorimaista ja visuaalista. Editoreita testataan ja selvitetään niiden hyvät ja huonot puolet. Lisäksi selvitetään minkälaiselle käyttäjäkunnalle ne soveltuvat parhaiten.

Käyttöliittymäosuudessa perehdytään käytettävyyden maailmaan ja selvitetään, mistä hyvä käyttöliittymä koostuu. Lisäksi tutkitaan tarkemmin, mitä asioita käyttöliittymää suunniteltaessa tulee huomioida. Lopuksi selvitetään vielä, miten eri testimenetelmillä voidaan parantaa ohjelman käytettävyyttä.

Case-osuudessa toteutetaan oma CSS-editori alusta loppuun asti hyödyntäen aikaisemmin tutkittuja asioita. Ohjelman tekoa seurataan vaihe vaiheelta ja selvitetään, mitä missäkin vaiheessa tapahtuu ja kuinka editorin kehittyy suunnitelmasta valmiiksi sovellukseksi.

2 CSS-TYYLIKIELI

Cascading Style Sheets (CSS) on yksinkertainen tyylikieli, jonka avulla määritellään HTML, XHTML ja XML -dokumenttien ulkoasu ja esitystapa. Sen tehtävänä on muotoilla HTML- dokumenteissa esiintyviä elementtejä tyylitiedostoon tai HTML:n sekaan kirjoitettujen ominaisuuksien mukaan. CSS on kehitetty W3C:n (World Wide Web Consortium) ohjauksessa ja siitä on julkaistu kaksi virallista suositusta CSS 1 ja CSS 2.1, mutta kehitys on ollut hidasta.

CSS on standardi, mutta on selainten oma tehtävä tulkita ja toteuttaa sitä ja vaikka standardi on selkeä ja tarkka eri selaimet tulkitsevat sen määrittelyjä hieman eri tavoilla. Tämä on johtanut epätasaiseen CSS:n toteutukseen ja sen takia toiset selaimet ovat selvästi toisia edellä.(Teague 2011, 6)

Tällä hetkellä on kehitteillä versio CSS 3, jonka ominaisuudet ovat aikaisemmista poiketen jaettu osiin, jotta niitä voitaisiin hallita paremmin. Siihen on myös lisätty paljon uusia ominaisuuksia, joista erikoisimpina elementtien animointia mahdollistavat ominaisuudet.

2.1 CSS:n historia

Tekstinkäsittelyohjelmien tyylimäärittelyjä voidaan pitää CSS:n edeltäjinä ja rinnakkaisilmiönä. Niiden tarkoituksena on, että käyttäjä voi valita joko valmiista tai itse luomistaan tyyleistä esimerkiksi eritasoisten otsikoiden fonttilajin, fonttikoon ja reunuksen. Jo HTML:n ensimmäisessä virallisessa määrittelyssä (HTML 2.0 vuonna 1995) mainittiin, että sivun ulkoasu riippuu muun muassa tyyliohjeista. Tyyliohjeiden järjestelmän määritteleminen ja toteuttaminen etenivät kuitenkin hitaasti, ja selainten valmistajat lisäsivät HTML:ään ulkoasun säätelyn piirteitä, joita sivujen tekijät opettelivat käyttämään. (Korpela 2008, 53–54.)

2.1.1 CSS 1

Työ ensimmäisen virallisen tyyliohjeen määrittelemiseksi aloitettiin vuonna 1995 W3C:n toimesta, ja jo 17.12.1996 julkistettiin CSS 1 - suositus, Cascadian Style Sheet, level 1. Ensimmäinen selain jossa oli jonkinlainen CSS-tuen tapainen, oli

elokuussa 1996 julkaistu Internet Explorer-selaimen versio 3. On sanottu, että sen CSS-tuen ehdottomasti paras puoli oli, että sen sai helposti pois käytöstä. Muilla sen ajan selaimilla oli myös suuria ongelmia CSS:n tukemisessa ja sen takia sivujen tekijät jatkoivatkin osittain HTML:n ulkoasupiirteiden käyttämistä, koska ajattelivat että näin sivut toimisivat paremmin eri selaimilla. (Korpela 2008, 54)

2.1.2 CSS 2

CSS 2 -suositus julkistettiin jo 12.5.1998, jolloin CSS 1:n toteutukset olivat vielä lähinnä kokeiluasteella. Selainten valmistajat eivät olleet keskittyneet kunnolla CSS 1 -tuen tekemiseen, eivätkä myöskään CSS 2:n järjestelmälliseen toteuttamiseen vaan tekivät vähän molempia. Ensimmäiset selaimet jotka varsinaisesti tukivat CSS 2:ta, olivat IE 6 ja Netscape 6. Näidenkin tuki oli vielä vajavaista, mutta pahojen virheiden määrä alkoi olla siedettävä. (Korpela 2008, 55)

CSS 2:sta päätettiin laatia eräänlainen realistinen versio CSS 2.1, johon oli tarkoitus kasata ne ominaisuudet, joita yleisimmät selaimet ovat toteuttaneet. CSS 2.1:tä määriteltäessä on jätetty pois ominaisuuksia, lisätty ja poistettu ominaisuuksien arvoja ja määrittelyjä on tarkennettu tai muutettu. Ensimmäinen luonnos siitä julkistettiin jo pian CSS 2:den jälkeen elokuussa 1998, viralliseksi suositukseksi CSS 2.1 pääsi vasta 7.6.2011 (W3C Recommendation 2011). CSS 2.1 vastaa melko hyvin sitä mitä nykyselaimilta voi odottaa.(Korpela 2008, 56)

2.1.3 CSS 3

CSS 3:n kehittely aloitettiin jo vuonna 1998, melko pian CSS 2 julkaisemisen jälkeen. W3C päätti kuitenkin keskeyttää uusien osien kehittämisen, koska selaimien tyyliluokkien tukeminen oli epäjohdonmukaista, ja keskittyä CSS 2.1:den standardisoimiseen. Vuonna 2005 kaikki CSS 3:n moduulit päätettiin palauttaa suunnittelutasolle ja aloittaa niiden arviointi alusta. Samaan aikaan CSS 3:lle vastahakoisen Internet Explorerin pitkä valtakausi alkoi tulla päätökseen ja alalle pyrki uusia kehitykselle avoimia selaimia.(Gasston, 2011, 2)

CSS 3 on jaettu moduuleihin, joka mahdollistaa joustavamman kehittämisen. Näin ollen yhden piirteen määrittelyä ja toteutusta voidaan tehdä, vaikka muilla alueilla ei päästäisikään eteenpäin.(Korpela 2008, 56)

3 CSS 3

3.1 Moduulit

Uusinta tyyliohjetta suunnitellessaan W3C tuli siihen tulokseen että CSS määrittelystä oli tullut jo niin massiivinen että sitä on vaikeaa ja kankeaa hallita yhtenä suurena kokonaisuutena, niinpä se päätettiin jakaa paremmin hallittaviin osiin. CSS 3:n suurin muutos edeltäjiin verrattuna onkin kaikkien vanhojen piirteiden jakaminen moduuleihin ja samalla uusien lisääminen.(Cederholm, 2010, 3)

Näin ollen eri moduuleja voidaan kehittää toisistaan riippumatta ja tärkeitä ja haluttuja ominaisuuksia saadaan julkaistua nopeammin, vaikka joidenkin vähäpätöisempien piirteiden toteuttaminen olisi kesken. Tämän ansiosta valmiiksi saatuja moduuleja voidaan myös asettaa saman tien suositukseksi, eikä tarvitse odottaa että koko uusi CSS 3-standardi saataisiin valmiiksi. Selainten valmistajat pystyvät muokkaamaan selaimensa tukemaan uusia suositeltuja moduuleita nopeammin ja sivujen kehittäjät voivat alkaa käyttämään uusia piirteitä sivuillaan. (Gasston, 2011, 2-3)

3.1.1 Media Queries

Ennen verkkosivuja tehdessä joutui miettimään miten sivut saisi sujuvasti toimimaan eri selaimilla, mutta viimevuosina lisääntyneet älypuhelimet ja tablet-laitteet tuovat tähän pulmaan aivan uudenlaisen kategorian. Pienemmillä näytöillä, hitaammilla yhteyksillä ja vajavaisemmilla selaimilla varustetut laitteet vaativat omanlaisensa verkkosivun ja tähän tarpeeseen uusi Media Queries moduuli vastaakin. Sen uusien ominaisuuksien ansiosta tyyliohjeen pystyy määrittämään suoraan laitteen kykyjä vastaavaksi, ikkunan ja kuvien kokoa, sekä toiminnallisuuksia muokkaamalla.(Gasston, 2011, 9-10)

Media Queries ominaisuuksia voi käyttää kolmella eri tavalla. Kahdessa ensimmäisessä esimerkissä kutsutaan erillistä tyylitiedostoa. Seuraavilla komennoilla:

> *<link href="file" rel="stylesheet" media="logic media and (expression)">*

@import url('file') logic media and (expression);

Kolmas vaihtoehto on upottaa niitä html-koodin sekaan style-elementtinä tai käyttää tyylitiedostossa erillisen @media määritteen avulla.

@media logic media and (expression) { rules }

Media Queries ominaisuuksilla voi muun muassa tarkistaa selaimen ikkunan maksimi/minimileveyden ja -korkeuden ja muokata sitä kautta elementtien ominaisuuksia.

```
@media screen and (min-width: 400px) {
    h1 {
        background: black url('landscape.jpg') no-repeat 50% 50%;
        height: 189px;
        margin-bottom: 0;
        padding: 20px;
    }
}
```

3.1.2 Background Images

CSS 3 mahdollistaa useiden kuvien käytön taustakuvana, niiden koon muokkaamisen lennosta, sekä paremman kontrollin kuvan sijainnista ja monistumisesta.(Gasston, 2011, 94)

Useiden taustakuvien näyttäminen määritetään background ominaisuudella, johon annetaan arvoiksi kuvan tiedostosijainti (url), monistuminen ja sijainti vasemmalta ja ylhäältä katsottuna (Kuva 1). Kuvista luodaan kerrokset käänteisessä järjestyksessä, niin että listan ensimmäinen kuva on ylin kerros.(Gasston, 2011, 94)

```
h2 {
background:
url(css3.png') no-repeat 95% 25%,
```

url(tausta.jpg') no-repeat;

}

Kuva 1. Esimerkkikuva kahdesta päällekkäisestä taustakuvasta (kuvakaappaus Chrome)

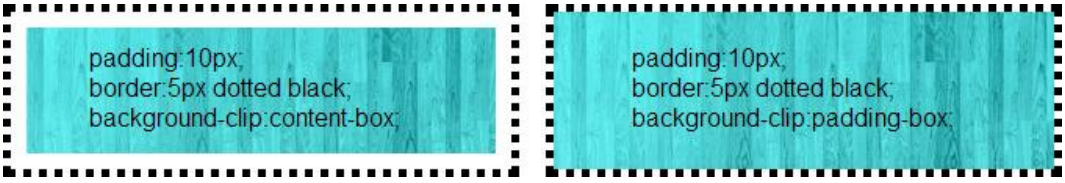
Taustakuvan kokoa voi muokata background-size ominaisuudella, jonka arvoksi voi laittaa mitan, prosenttiarvon, auto-, contain- tai cover-arvon. Contain-arvolla taustakuva mahdutetaan elementin sisälle ja cover-arvolla taustakuva täyttää koko elementin (Kuva 2). (Gasston, 2011, 96)

div { background-size: 100px 200px; }

Kuva 2. Esimerkkikuva background-size-ominaisuuden contain- ja cover-arvojen toiminnasta (kuvakaappaus Chrome)

Background Clip- ja Origin-ominaisuuksilla voi joko leikata taustakuvaa tai määrittää sen alkamiskohdan. Niillä on kolme mahdollista arvoa, jotka ovat border-box, content-box tai padding-box. Niillä määritetään alkaako tai leikkaantuuko taustakuva elementin ulko- vai sisäreunasta vai padding-arvosta (Kuva 3). (Gasston, 2011, 98)

h2{ background-clip: border-box; }



Kuva 3. Esimerkkikuva background-clip-ominaisuuden content-box- ja padding-box-arvojen toiminnasta (kuvakaappaus Chrome)

Backround-repeat on saanut kaksi uutta arvoa, jotka ovat space ja round (Kuva 4). Space-arvolla taustakuva toistuu alkuperäisessä koossa täyttäen koko alueen jättämällä tarvittavan määrän tyhjää tilaa kuvien väliin. Round-arvo toimii muuten samoin, mutta tyhjän tilan sijaan taustakuva täyttää tilan kasvattamalla kokoaan. (Gasston, 2011, 102)

.space { background-repeat: space; }

.round { background-repeat: round; }

Kuva 4. Esimerkkikuva background-repeat-ominaisuuden round- ja space-arvojen toiminnasta (kuvakaappaus Internet Explorer)

3.1.3 Text Effects

Tekstisisältö on ollut verkkosivujen perusta sen alusta alkaen, mutta sen muotoiluun ei ole ollut paljoa työkaluja. CSS 3 esittelee kuitenkin joukon uusia ominaisuuksia, joista tärkeimpinä ovat mahdollisuus lisätä tekstille varjo ja reunus. Nämä ovat käteviä ominaisuuksia varsinkin sivun otsikoita tehdessä.

Text-shadow ominaisuudelle pystyy antamaan arvot tekstin varjo-ominaisuuden sijainnille, sumeudelle ja värille (Kuva 5). Niitä pystyy myös määrittämään useita yhdelle tekstille.(Gasston, 2011, 68)

E { text-shadow: x y blur-radius color; }

h2 { text-shadow: 3px 3px 3px #BBB; }

Kuva 5. Esimerkkikuva text-shadow-ominaisuudella tekstille luodusta varjosta (kuvakaappaus Chrome)

Text-outline ominaisuudella pystytään määrittämään tekstille reunus, sen koko, sumeus ja väri.(Gasston, 2011, 72)

E { text-outline: width blur-radius color; }

h1 { text-outline: 2px 4px blue; }

3.1.4 Borders

Elementtien reunojen muotoilu on ollut aina todella vaivalloista ja aikaa vievää, mutta CSS 3:n tuomat uudistukset tekevät tästä asiasta huomattavasti helpompaa. Border-radius ominaisuus mahdollistaa vaivattoman elementin jokaisen kulman yksilöllisen pyöristyksen, niin x- kuin y-akselin suunnassa (Kuva 6). (Gasston, 2011, 108–109)

div {

border-top-left-radius: 10px 20px;

border-top-right-radius: 10px 20px;

border-bottom-right-radius: 10px 20px;

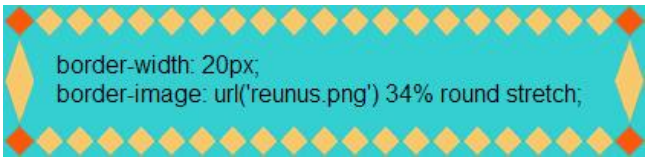
border-bottom-left-radius: 10px 20px;

}

Kuva 6. Esimerkkikuva border-radius-ominaisuudella luodusta elementtien kulmien pyöristämisestä (kuvakaappaus Chrome)

Border-image toiminnolla pystyy käyttämään yhtä ainutta kuvaa elementin reunojen skaalautuvaan koristeluun. Kuvaa voi määritellä kuinka se skaalautuu ja toistuu elementin eri reunoilla (Kuva 7). (Gasston, 2011, 113)

div { border-image: url('frame.png') 32 39 36 41 round stretch;}



Kuva 7. Esimerkkikuva border-image-ominaisuudella toteutetusta elementin reunojen koristelusta (kuvakaappaus Chrome)

Border-color ominaisuudella saa luotua haluamansalaisen monivärisen reunan (Kuva 8).

div {border-left-width: 3px; border-left-colors: green white red;}

Kuva 8. Esimerkkikuva border-color-ominaisuudella luodusta elementin reunuksesta (kuvakaappaus Firefox)

Box-shadow ominaisuudella pystyy luomaan halutun kokoisen ja näköisen varjon, joko elementin ulko- tai sisäpuolelle. Varjolle pystyy antamaan arvot sen sijainnista, sumeudesta, koosta ja väristä (Kuva 9). (Gasston, 2011, 115)

div { box-shadow: 4px 4px 3px #666; }

Kuva 9. Esimerkkikuva box-shadow-ominaisuudella luodusta elementin varjosta (kuvakaappaus Chrome)

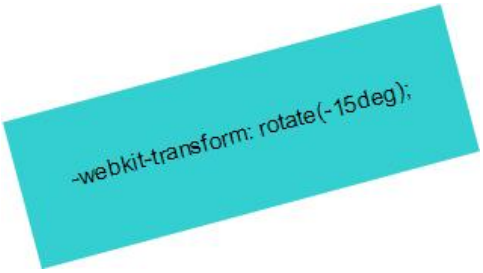
3.1.5 2D Transforms

Verkkosivut on normaalisti totuttu näkemään useimmiten laatikkomaisina, suorakulmaisina sivuina, jossa on enimmäkseen vaaka ja pystysuoria viivoja ja ainut keino saada jotakin muuta aikaiseksi on ollut käyttää kuvia. CSS 3:n esittelemä 2D transform -ominaisuus kuitenkin muuttaa kaiken ja mahdollistaa elementtien kiertämisen, koon muuttamisen, vääristämisen ja vastaavanlaisen muokkaamisen, niin kaksi- kuin kolmiulotteisessakin ympäristössä. 2D Transform -ominaisuus sisältää useita funktioita, joiden arvoja muuttamalla näitä muunnoksia saadaan aikaiseksi.(Gasston, 2011, 149)

Rotate-funktion arvoa muuttamalla saadaan kierrettyä elementtiä haluttu astemäärä (Kuva 10). Pistettä jonka ympäri elementti kiertyy, voidaan myös siirtää transform-origin-ominaisuuden arvoa muuttamalla. (Gasston, 2011, 147)

h2 { transform: rotate(-15deg); }

h2 { transform-origin: 200px 50px; }



Kuva 10. Esimerkkikuva transform-ominaisuuden rotate-funktiolla tehdystä elementin kiertämisestä (kuvakaappaus Chrome)

Translate-funktiolla voidaan siirtää elementtiä haluttu määrä haluttuun suuntaan (Kuva 11). (Gasston, 2011, 152)

h2 { transform: translate(20px,20px); }

Kuva 11. Esimerkkikuva transform-ominaisuuden translate-funktiolla tehdystä elementin siirtämisestä (kuvakaappaus Chrome)

Skew-funktiolla pystytään vääristämään elementtiä antamalla sille arvot x- ja y-akselin muunnoksille (Kuva 12).(Gasston, 2011, 153)

h2 { transform: skew(-45deg,5deg); }

Kuva 12. Esimerkkikuva transform-ominaisuuden skew-funktiolla tehdystä elementin vääristämisestä (kuvakaappaus Chrome)

Scale-funktiolla voidaan muuttaa elementin kokoa antamalla sille arvot vaaka- ja pystysuunnan kertoimiksi (Kuva 13).(Gasston, 2011, 154)

h2.transform-3 { transform: scale(1.5,2); }

Kuva 13. Esimerkkikuva transform-ominaisuuden scale-funktiolla tehdystä elementin koon muuttamisesta (kuvakaappaus Chrome)

3.1.6 3D Transforms

Aikaisemmin CSS:ssä on ollut vain kaksi ulottuvuutta ja elementeillä on ollut vain leveys ja korkeus arvot, mutta nyt CSS 3:n esitellessä kolmannen akselin mahdollistaa se elementtien muokkaamisen myös kolmannessakin ulottuvuudessa. 3D transform kuuluu 2D transformin kanssa samaan moduuliin, mutta siinä toteutetaan elementtien muokkaus eri akseleiden kautta.

Elementtejä pystyy kiertämään kaikkien kolmen akselin ympäri, liikuttamaan niitä pitkin ja skaalaamaan. Lisäksi pystytään määrittämään perspektiivi, etäisyys kappaleesta z-akselia pitkin (Kuva 14).

div { transform: rotateX(-60deg); }

div { transform: translateZ(30px); }

div { transform: scaleZ(3); }

div { transform: perspective(20); }

.trans-2 { transform: rotateX(-90deg) rotateY(-15deg) perspective(50) translateZ(10px); }

Kuva 14. Esimerkkikuva 3D transform -ominaisuuden funktioilla tehdystä elementin kiertämisestä (kuvakaappaus Chrome)

3.1.7 Transition

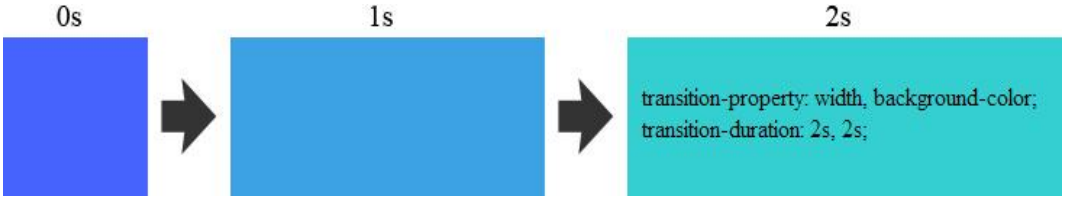
Verkkosivuilla on totuttu siihen että on niillä kolme eri kerrosta, sisältö (html), esitys (CSS) ja toiminnallisuus (JavaScript), mutta tähän tulee muutos CSS 3 esittelemien transition ja animations-moduulien myötä. Nämä moduulit mahdollistavat elementtien animoimisen ja liikkeen lisäämisen muuten staattiselle sivulle ilman JavaScriptiä.(Gasston, 2011, 163)

Suurin ero transition- ja animation-moduulien välillä on niiden liikkeen laukaisemisessa. Transitio alkaa toistaa liikettä elementin jonkin ominaisuuden arvon muuttuessa, kun taas animation toteutetaan silloin kun se liitetään elementtiin.(Gasston, 2011, 164)

Transition on animaatio ominaisuuden kahden arvon välillä. Toimiakseen se vaatii alkuarvon, loppuarvon, transition ja laukaisimen. Laukaisimena voi toimia esimerkiksi hover-ominaisuus tai muu vastaava, joka muuttaa elementin ominaisuuden arvoa.

Transitionin ominaisuuksia ovat muokattavan ominaisuuden valinta, transition kesto, transition tyyli ja transitionin aloituksen viivyttäminen (Kuva 15). Muokattavia ominaisuuksia ovat useat elementtien CSS ominaisuudet.(Gasston, 2011, 164-171)

```
h1 {
        font-size: 150%;
        transition-property: font-size;
        transition-duration: 2s;
        transition-timing-function: ease-out;
        transition-delay: 250ms;
}
h1 {
        font-size: 150%;
        transition: font-size 2s ease-out 250ms;
}
```



Kuva 15. Esimerkkikuva transition-ominaisuuksilla tehdystä elementin leveyden ja värin animoidusta muuttamisesta (kuvakaappaus Chrome)

3.1.8 Animations

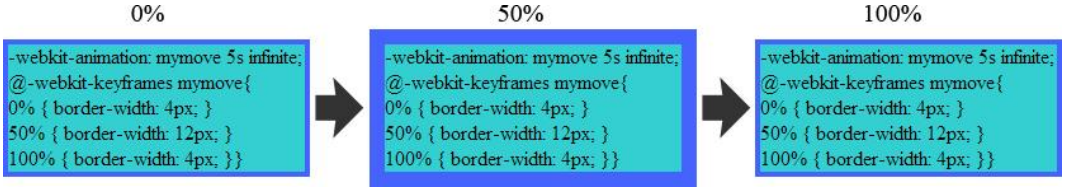
Animations on transitionin kaltainen, mutta toimintaperiaate animaatioiden luomisessa on varsin erilainen ja animations-moduulilla pystyy myös tekemään monimutkaisempia ja paremmin kontrolloituja animaatioita. Siinä määritellään ensin ominaisuudet ja ajoitus ja lisätään sitten animaation kontrollit elementtiin joka animoidaan.(Gasston, 2011, 172)

Animationsin ensimmäisiin vaiheisiin kuuluu key framejen määrittäminen. Niillä pystytään määrittämään kuinka monta vaihetta animaatiossa on, kun transitionissa niitä oli vain alku ja loppu. Key framejen sisään määritetään, mitkä elementin ominaisuudet animaatiossa muuttuvat milläkin hetkellä.(Gasston, 2011, 172)

```
@keyframes 'expand' {
        0% { border-width: 4px; }
        50% { border-width: 12px; }
        100% {border-width: 4px;}
}
```

Tyylitiedostossa määritellään animations moduulin ominaisuuksien arvot, joita ovat animaation nimi, kesto, tyyli, viivästyminen, toistojen määrä ja suunta (Kuva 16). Animationsilla on myös ominaisuus animation-play-state jolla voi määrittää onko animaatio toiminnassa vai pysäytetty. (Gasston, 2011, 176)

```
div {
        animation-name: border-changer;
        animation-duration: 6s;
        animation-timing-function: ease-in;
        animation-delay: 2s;
        animation-iteration-count: 10;
        animation-direction: alternate;
}
div { animation: 'border-changer' 6s ease-in 2s 10 alternate; }
```



Kuva 16. Esimerkkikuva animation-ominaisuuksilla tehdystä elementin reunan leveyden animoidusta muuttamisesta (kuvakaappaus Chrome)

3.1.9 Muita moduuleita

Multi-column Layout -moduulin ominaisuuksilla pystytään määrittelemään elementin teksti toistumaan halutussa määrässä sarakkeita, niiden leveyden tai määrän arvoa muuttamalla.(Gasston, 2011, 83)

User Interface -moduulin ominaisuuksilla pystytään muun muassa saamaan elementistä skaalautuva venyttämällä sitä suoraan selaimella.

Valitsimet ovat CSS:n yksi tärkeimmistä asioista, ja niillä määritetään mihin asioihin ominaisuudet vaikuttavat. CSS 1:ssä oli vain viisi tai kuusi valitsinta, CSS 2:ssa oli jo 12 ja CSS 3:n selectors moduuli tuplaa valitsimien määrän.(Gasston, 2011, 23)

3.2 Vaikutukset sivujen toteuttamiseen

CSS 3:n tuomat uudet ominaisuudet eivät pelkästään mahdollista hienompien ja joustavampien verkkosivujen suunnittelua, vaan tekevät niistä kevyempiä ja toimivampia ja myös nopeuttavat ja helpottavat niiden tekemistä. Uudet ominaisuudet myös vähentävät kuvankäsittelyohjelmien käytön tarvetta, ylimääräisten kuvien käyttöä sivujen ulkoasussa ja JavaScriptin ja flashin käyttöä.(Gillenwater, 2011, 16)

3.2.1 Sivun ulkoasun toteuttaminen

Verkkosivujen layoutin suunnittelu, muotoilu ja toteutus ovat vaatineet aina pikselintarkkaa työtä niin kuvankäsittelyohjelmassa kuin verkkosivujen koodausohjelmassakin. Otsikoiden, elementtien ja grafiikoiden tekeminen vaativat paljon tarkkaa työtä ja välillä jopa selainten huijaamista ja kikkailua, jotta sivut toimisivat hyvin eri selaimilla ja erikokoisilla näytöillä. CSS 3:n uudet ominaisuudet onneksi tuovat tähän ongelmaan kasan uusia ratkaisuja ja helpotuksia.

3.2.2 Reunojen koristelu

Verkkosivuilla suosittu elementtien kulmien pyöristäminen on ennen vaatinut huomattavan määrän töitä. Ensin täytyi tehdä neljän erillistä kuvaa elementin kulmista kuvankäsittelyohjelmassa, sekä neljä kuvaa sivuista mikäli niissäkin on jokin reunus, sitten sijoittaa kulmat elementin nurkkiin taulukon tai divien avulla ja näin mukautuva pyöristetty laatikko oli vihdoin valmis. Muita yleisiä tapoja elementtien muotoilussa ovat reunojen koristelu varjolla, väripalkilla tai muulla vastaavalla. Nämä muotoilut vaativat täsmälleen samanlaista esityötä kuin reunojen pyöristys toimiakseen. CSS 3:n uudet ominaisuudet kuitenkin helpottavat huomattavasti näiden kaikkien asioiden toteuttamisessa.

CSS 3:n corner-radius, border-image, border-color ja box-shadow -ominaisuudet mahdollistavat elementtien vaivattoman muotoilun muutamalla rivillä CSS-koodia. Turhauttavasta kuvien ja reunojen tarkasta asettelusta eroon pääsemisen lisäksi uudet ominaisuudet myös helpottavat ja nopeuttavat elementtien päivittämistä ja muokkaamista, sekä tekevät niistä helpommin skaalautuvia erikokoisille näytöille.

3.2.3 Otsikot, taustat ja laatikot

Yleensä sivuilla on jonkunlainen yläpalkki jossa on sivun logo, nimi ja jonkinlainen taustakuva. Tämä joudutaan normaalisti tekemään kuvankäsittelyohjelmalla tehdyistä erillisistä kuvista, jotka asetetaan oikeille paikoilleen divejä tai taulukoita käyttämällä sivuja koodatessa. Tämäkin hoituu huomattavasti helpommin CSS 3 uusien tekstinmuotoilu- ja background-ominaisuuksien ansiosta. Pää- ja väliotsikoille saa helposti pienet koristelut varjoilla ja reunuksilla ja useita taustakuvia pystyy sijoittamaan helposti ja tarkasti elementtiin.

Sivujen muunkin sisällön tekemistä helpottavia uusia ominaisuuksia on lukuisia, kuten tekstin jakaminen palstoihin column-count-ominaisuudella, jolla jaat tekstin helposti useisiin palstoihin ilman tarvetta erillisille diveille tai taulukoille. Opacity- ja gradient-ominaisuuksilla saa helposti, ilman tarvetta turhille kuville, muokattua sivun elementtejä halutunlaisiksi.

3.2.4 JavaScript- ja flashkoodin korvaaminen

Transition- ja animation-moduulien ominaisuudet mahdollistavat elementin arvojen, ulkomuodon ja sijainnin muokkaamisen tietyllä aikavälillä. Tämä ennen JavaScriptillä tai flashillä tehty animointi onnistuu nyt helposti CSS 3 tuomien uusien ominaisuuksien avulla.(Teague 2011, 303)

Animationin-ominaisuuksilla ja yhdistämällä transitions ja transforms-moduulien ominaisuuksia saadaan sivulla toteutettua erilaisia toiminnallisuuksia, eikä Javascriptiin ja flashiin tarvitse turvautua heti, kun halutaan jonkinlaista toiminnallisuutta sivulle. Näin saadaan sivujen koodaamisesta hieman helpompaa ja sivusta hieman kevyemmät. Uusilla ominaisuuksilla voi muun muassa toteuttaa kuvagallerioita, joissa kuvat liikkuvat, kasvavat ja tausta pimenee ilman JavaScriptiä tai flashia (Kuva 17 ja 18).

Kuva 17. CSS-tyylimäärittelyillä toteutettu kuvagalleria (Splashnology)

Kuva 18. CSS-tyylimäärittelyillä toteutettu lightbox-ominaisuus (Carsonified)

3.2.5 Layoutin suunnittelu

Moni suunnitelee verkkosivun ulkoasun ja ilmeen yleensä ensimmäiseksi jollakin kuvankäsittelyohjelmalla ja alkaa sen jälkeen tehdä siitä täysin tarkkaa kopiota verkkosivueditorilla. Tällä tavalla saattaa suunnitella joitakin ominaisuuksia, jotka ovat vaikeita tai melkein mahdottomia toteuttaa selaimella. Vielä jos olet ehtinyt näyttää kyseisen suunnitelman asiakkaalle, eikä hän halua luopua niistä, olet melkoisessa pulassa.

CSS 3 on esitellyt ison määrän verkkosivujen koodaamista helpottavia ja nopeuttavia ominaisuuksia, joten sivujen suunnittelun voisi aloittaa suoraan verkkosivueditorilla. Luo ensin yksinkertainen pohja, johon alat pikkuhiljaa lisäämään CSS-koodia. Näin pystyt helposti muokkaamaan suunnitelmaasi ja näyttämään asiakkaallesi realistisia versioita verkkosivusta.(Gillenwater, 2011, 16, 43)

3.3 Yhteensopivuus

CSS 3:n jako moduuleihin tarkoittaa eri osien eriaikaista valmistumista, toiset valmistuvat nopeammin kuin toiset ja näin ollen niitä pystyy käyttämään aikaisemmin sivujen muotoilussa. Toinen ratkaiseva asia on selainten nopeus ja halu tukea uusia valmiita ja keskeneräisiä ominaisuuksia.

3.3.1 CSS 3:n valmiusaste

Moduulienvalmius astetta kuvataan statuksella, jonka W3C asettaa (Kuva 19A ja 19B). Statusarvo kuvaa moduulin matkaa julkaisusta valmiiksi suositukseksi.

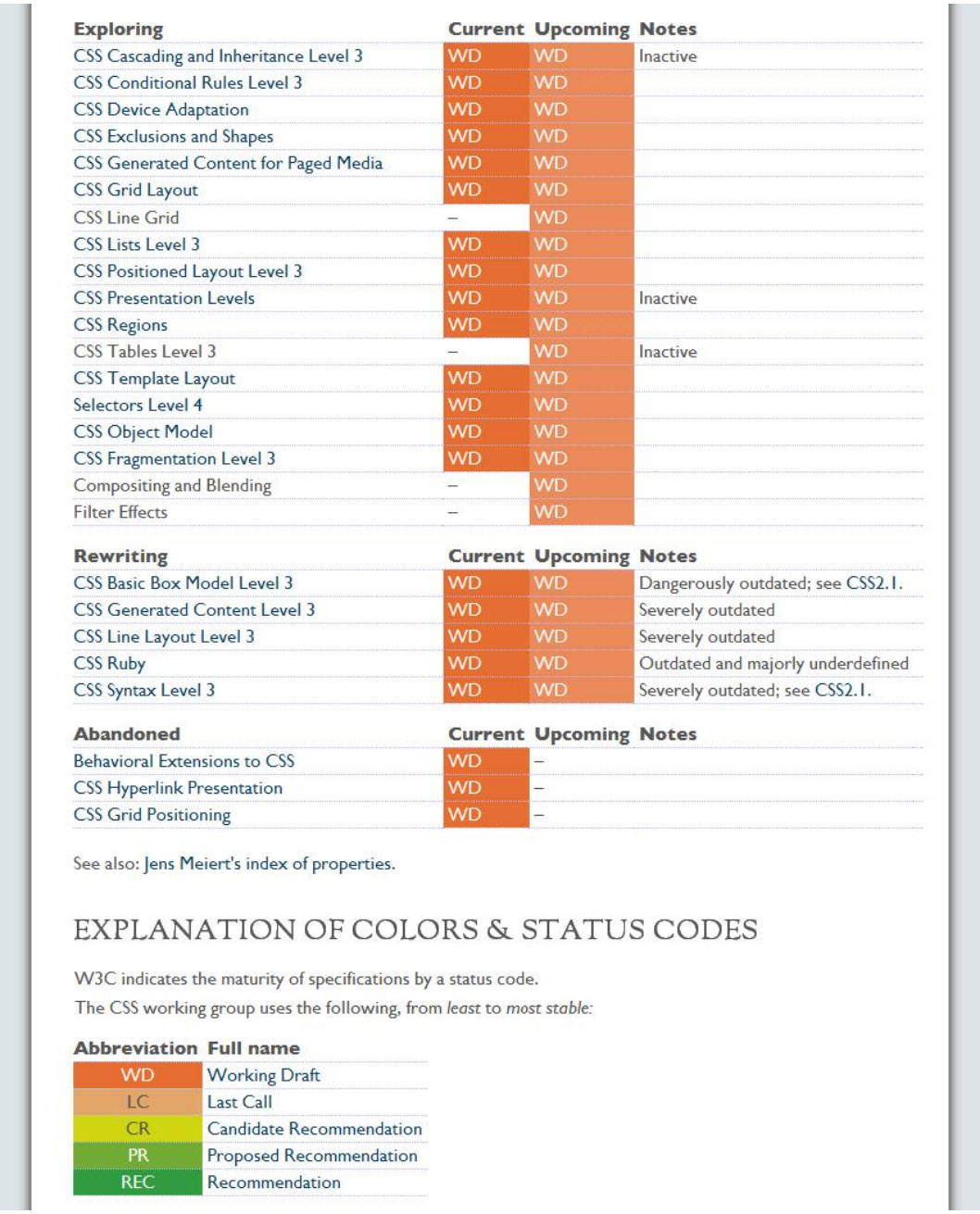
Kun uusi dokumentti hyväksytään osaksi CSS 3:a se saa ensiksi Working Draft -statuksen, joka tarkoittaa että se on julkaistu ja kehittäjät alkavat tutkia sitä. Seuraavaksi dokumentti saa Last Call -statuksen, joka tarkoittaa sitä että se on valmis etenemään seuraavalle tasolle. Seuraava taso on Candidate Recommendation, joka tarkoittaa W3C:n hyväksyneen, että dokumentti on järkevä ja siinä ei ole suuria virheitä. Tässä vaiheessa selaimet alkavat yleensä toteuttaa dokumentin ominaisuuksia. Kun kaksi tai useampi selain toteuttaa dokumentin ominaisuuksia samalla tavalla, eikä ongelmia ole ilmennyt saa dokumentti Proposed Recommendation statuksen. Tämä tarkoittaa sitä että dokumentti on valmis W3C:n neuvoa-antavan komitean hyväksyttäväksi, jonka jälkeen ehdotus saa Recommendation-statuksen.(Gasston, 2011, 3)

TABLE OF SPECIFICATIONS

Ordered from most to least stable:

Completed	Current	Upcoming	Notes
CSS Snapshot 2010	NOTE	–	Latest stable CSS
CSS Snapshot 2007	NOTE	–	
CSS Color Level 3	REC	REC	See Errata
CSS Namespaces	REC	REC	
Selectors Level 3	REC	REC	
CSS Level 2 Revision 1	REC	REC	See Errata
CSS Level 1	REC	–	Unmaintained, see Snapshot
Stable	**Current**	**Upcoming**	**Notes**
Media Queries	CR	PR	
CSS Style Attributes	CR	PR	
Testing	**Current**	**Upcoming**	**Notes**
CSS Backgrounds and Borders Level 3	LC	CR	
CSS Marquee	CR	PR	
CSS Multi-column Layout	CR	CR	
CSS Speech	CR	PR	
CSS Mobile Profile 2.0	CR	PR	Status unknown
CSS TV Profile 1.0	CR	?	Status unknown
Refining	**Current**	**Upcoming**	**Notes**
CSS Image Values and Replaced Content Level 3	LC	CR	
CSS Transforms	WD	WD	
CSS Transitions	WD	WD	
CSS Values and Units Level 3	LC	CR	
CSS Print Profile	LC	?	Status unknown
Revising	**Current**	**Upcoming**	**Notes**
CSS Animations	WD	WD	
CSS Flexible Box Layout	WD	WD	
CSS Fonts Level 3	WD	LC	
CSS Paged Media Level 3	LC	LC	Inactive
CSS Text Level 3	WD	WD	
CSS Basic User Interface Level 3	CR	LC	
CSS Writing Modes Level 3	WD	WD	
CSSOM View	WD	WD	

Kuva 19A. Moduulien valmiusaste status 11.4.2012 (W3C)



Exploring	Current	Upcoming	Notes
CSS Cascading and Inheritance Level 3	WD	WD	Inactive
CSS Conditional Rules Level 3	WD	WD	
CSS Device Adaptation	WD	WD	
CSS Exclusions and Shapes	WD	WD	
CSS Generated Content for Paged Media	WD	WD	
CSS Grid Layout	WD	WD	
CSS Line Grid	–	WD	
CSS Lists Level 3	WD	WD	
CSS Positioned Layout Level 3	WD	WD	
CSS Presentation Levels	WD	WD	Inactive
CSS Regions	WD	WD	
CSS Tables Level 3	–	WD	Inactive
CSS Template Layout	WD	WD	
Selectors Level 4	WD	WD	
CSS Object Model	WD	WD	
CSS Fragmentation Level 3	WD	WD	
Compositing and Blending	–	WD	
Filter Effects	–	WD	

Rewriting	Current	Upcoming	Notes
CSS Basic Box Model Level 3	WD	WD	Dangerously outdated; see CSS2.1.
CSS Generated Content Level 3	WD	WD	Severely outdated
CSS Line Layout Level 3	WD	WD	Severely outdated
CSS Ruby	WD	WD	Outdated and majorly underdefined
CSS Syntax Level 3	WD	WD	Severely outdated; see CSS2.1.

Abandoned	Current	Upcoming	Notes
Behavioral Extensions to CSS	WD	–	
CSS Hyperlink Presentation	WD	–	
CSS Grid Positioning	WD	–	

See also: Jens Meiert's index of properties.

EXPLANATION OF COLORS & STATUS CODES

W3C indicates the maturity of specifications by a status code.
The CSS working group uses the following, from *least* to *most stable*:

Abbreviation	Full name
WD	Working Draft
LC	Last Call
CR	Candidate Recommendation
PR	Proposed Recommendation
REC	Recommendation

Kuva 19B. Moduulien valmiusaste status 11.4.2012 (W3C)

3.3.2 Yhteensopivuus selainten kanssa

Uusien selainten horjutettua Internet Explorerin valta-asemaa ja kilpaillessa käyttäjistä ovat selainvalmistajat lähteneet innolla myös päivittämään selaimiaan. Tämän ansiosta esimerkiksi uudet CSS-ominaisuudet toimivat myös melko pian julkaisun jälkeen ainakin jollakin tavalla uusimmissa selaimissa, IE:n kuitenkin ollessa selvästi muita hitaampi (w3school, 2012).

Selaimen käyttäessä ominaisuutta joka ei ole vielä virallinen, se lisää ominaisuuden eteen prefix koodin sen omaa kääntäjää varten, jotta se osaisi toistaa ominaisuuden oikein (Kuva 20).

PREFIX	RENDERING ENGINE	POPULAR BROWSERS USING THIS RENDERING ENGINE
-khtml-	KHTML	Konqueror
-ms-	Trident	Internet Explorer
-moz-	Mozilla	Firefox, Camino, Flock
-o-*	Presto	Opera, Opera Mobile, Opera Mini, Nintendo Wii browser
-webkit-	Webkit	Safari, Safari on iOS, Chrome, Android browser

* In the Presto rendering engine, speech-related properties are prefixed with -xv- instead of -o-.

Kuva 20. Selainten käyttämät prefix koodit (Gillenwater)

IE:n CSS 3 -tuen puutteeseen kyllästyneet suunnittelijat ovat kehitelleet erilaisia projekteja uusien ominaisuuksien tukemiseksi IE:llä. Yksi esimerkeistä on CSS3 PIE joka mahdollistaa osan uusien ominaisuuksien käytöstä myös IE:llä lisäämällä pieni koodi tyylitiedostoon.(CSS3 PIE, 2012)

behavior: url(PIE.htc);

3.3.3 Milloin käyttää

CSS 3 on valmis käytettäväksi jo tänään, kunhan valitset mitä osia siitä käytät ja missä yhteydessä. Isoa osaa uusista ominaisuuksista tuetaan jo melkein kaikilla selaimilla ja kokeellisempiakin jo useilla. Uusia ominaisuuksia voi käyttää huoletta, kunhan ne eivät vaaranna sivun käytettävyyttä, saatavuutta ja rakennetta, ja jos jotkut uusista hienoista visuaalisista kikoista eivät näy oikein ei se haittaa, kunhan sivun yleisrakenne on toimiva. (Gasston, 2011, 4-5)

Sivun ulkoasua tehdessä on tarpeetonta pyrkiä siihen että se näyttäisi täsmälleen samalta jokaisessa eri selaimessa tai varsinkaan kaikissa eri laitteissa. Selaimet ja laitteet saattavat näyttää saman ominaisuuden hieman eri lailla, mutta tärkeintä olisi että sivu on joka tapauksessa toimiva ja hyvännäköinen.

4 CSS-EDITORIT

CSS-editori on ohjelma, jolla kirjoitetaan, muokataan ja luodaan CSS-tiedostoja. Millä tahansa tekstieditorilla pystyy myös kirjoittamaan CSS-koodia, mutta tehtävään varta vasten erikoistuneissa editoreissa on monia asiaa helpottavia ominaisuuksia. HTML-editoreissa on yleensä myös CSS-tyylimäärittelyjen tekemiseen tarkoitettuja toimintoja. CSS-ohjelmat ovat yleensä tiedostokooltaan hyvin pieniä ja kevyitä käyttää.

CSS-editorin käyttöliittymä voi olla tekstieditorimainen tai hieman graafisemmalta näyttävä ohjelma. Tekstieditorin kaltaisessa ohjelmassa tyylimäärittelyt yleensä kirjoitetaan tiedostoon ja ohjelma värjää ja muuttaa tekstin ulkonäköä koodin tehtävien mukaan. Graafisemmissa ohjelmissa voi olla objekteja, joita voidaan kääntää, venyttää ja muokata ja tuottaa näin halutunlaista CSS-koodia.

4.1 Käyttöliittymät

CSS-editorin tehtävä on helpottaa ja nopeuttaa koodin kirjoittamista jollakin tavalla, verrattuna tavalliseen tekstieditoriin. Yksinkertaisimmillaan se on tekstieditorin näköisessä ohjelmassa, jossa CSS-editori muokkaa tekstin ulkoasua värjäämällä, korostamalla ja jäsentelemällä sitä koodin sisällön mukaan, tehden koodista näin huomattavasti helpommin luettavaa ja hahmotettavaa. Yleensä tämän kaltaisissa ohjelmissa editori myös ehdottaa käyttäjälle vihjeitä ja vaihtoehtoja aloitetun koodipätkän lopputuloksesta, nopeuttaen tyylimuotoilun tekemistä. Tämän tyylisen editorin käyttö vaatii käyttäjältä jo jonkin asteista tietämystä CSS-koodikielestä ja sen sisältämistä syntakseista ja onkin sen takia usein ammattilaisten suosima, sillä he tietävät mitä tekevät ja näin he saavat sen nopeimmin tehtyä.

CSS-tyylimuotoilujen tekeminen ja aloituskynnys helpottuvat selvästi editorissa, jossa koodin kirjoittamisen sijaan elementtien muotoilu hoidetaan pudotusvalikoilla ja liukupalkeilla. Näin käyttäjän ei tarvitse tietää CSS-koodista oikeastaan mitään vaan hän voi valita haluamansa muotoilut valmiiden valikoiden tarjoamasta tarjonnasta. Myös kokeneempi käyttäjä saattaa löytää tällä tavalla uusia CSS-määrittelyjä joita ei ole aikaisemmin tuntenut. Editori kirjoittaa CSS-koodin tie-

dostoon valikoiden perusteella tehtyjen valintojen mukaan ja näin koodiin ei tule käyttäjän tekemiä virheitä.

Graafisen käyttöliittymän omaavassa CSS-editorissa käyttäjä näkee elementin, jota hän pystyy muokkaamaan hiiren avulla suoraan venyttämällä, kääntämällä ja raahaamalla sitä. Tällaisen editorin aloituskynnys on matala, sillä käyttäjä pystyy muotoilemaan elementtiä vaivattomasti ja näkee välittömästi hänen toimintojensa vaikutuksen. Graafinen editori sisältää usein myös valikoiden avulla tapahtuvan elementtien tarkemman muotoilun. Hyvässä editorissa tulisikin olla mahdollista muotoilla CSS-koodia kaikilla mahdollisilla tavoilla, riippuen käyttäjän kyvyistä, mieltymyksistä, tarpeista ja käyttötilanteista.

4.2 Ominaisuudet

Editorit auttavat huomattavasti CSS-koodin tekemisessä, mutta niissä on monia muitakin hyödyllisiä ominaisuuksia, jotka helpottavat tyylimäärittelyjen luomista. Yksi tärkeimmistä on useissa editoreissa oleva tyylimäärittelyjen esikatselu, jolloin koodia muokatessa näkee välittömästi millainen vaikutus sillä on elementteihin. Joissakin editoreissa on ohjelman oman esikatseluikkunan lisäksi mahdollista tarkastaa miten eri selaimet, kuten Chrome, Firefox ja IE ja niiden eri versiot näyttävät määrittelyt. Näin käyttäjä näkee vaivattomasti miltä määrittelyt näyttävät eri selaimilla ja kuinka ne tukevat mitäkin CSS-ominaisuuksia.

CSS-koodin tiivistäminen ja järjestäminen kuuluvat myös osan editoreista toimintoihin, tämä parantaa pitkien tyylitiedostojen luettavuutta ja hallittavuutta. Lisäksi editoreilla on mahdollista tarkistaa koodin oikeellisuus, jolloin se paljastaa virheet ja ehdottaa ratkaisuja niiden korjaamiseksi. Jotkin editorit ilmoittavat uusimpien CSS-ominaisuuksien kohdalla, jos jokin selain ei tue kyseistä ominaisuutta ja osassa editoreista on mahdollista valita minkä CSS version ominaisuuksia käyttää.

Tehokkaaseen CSS-editoriin on kerätty monipuolisesti kaikki CSS-tyylimuotoilun eri vaiheissa tarvittavat ominaisuudet yhteen ohjelmaan. Käyttäjä pystyy hahmottelemaan, kokeilemaan, tarkistamaan ja tekemään tyylitiedoston vaivattomasti ja tehokkaasti samalla ohjelmalla, eikä joudu hyppimään eri ohjelmien välillä kesken

kaiken. Monipuolinen ja helppokäyttöinen editori sopii erilaisille käyttäjille ja nopeuttaa ja helpottaa jokaisen koodin kirjoittamista taitotasosta riippumatta.

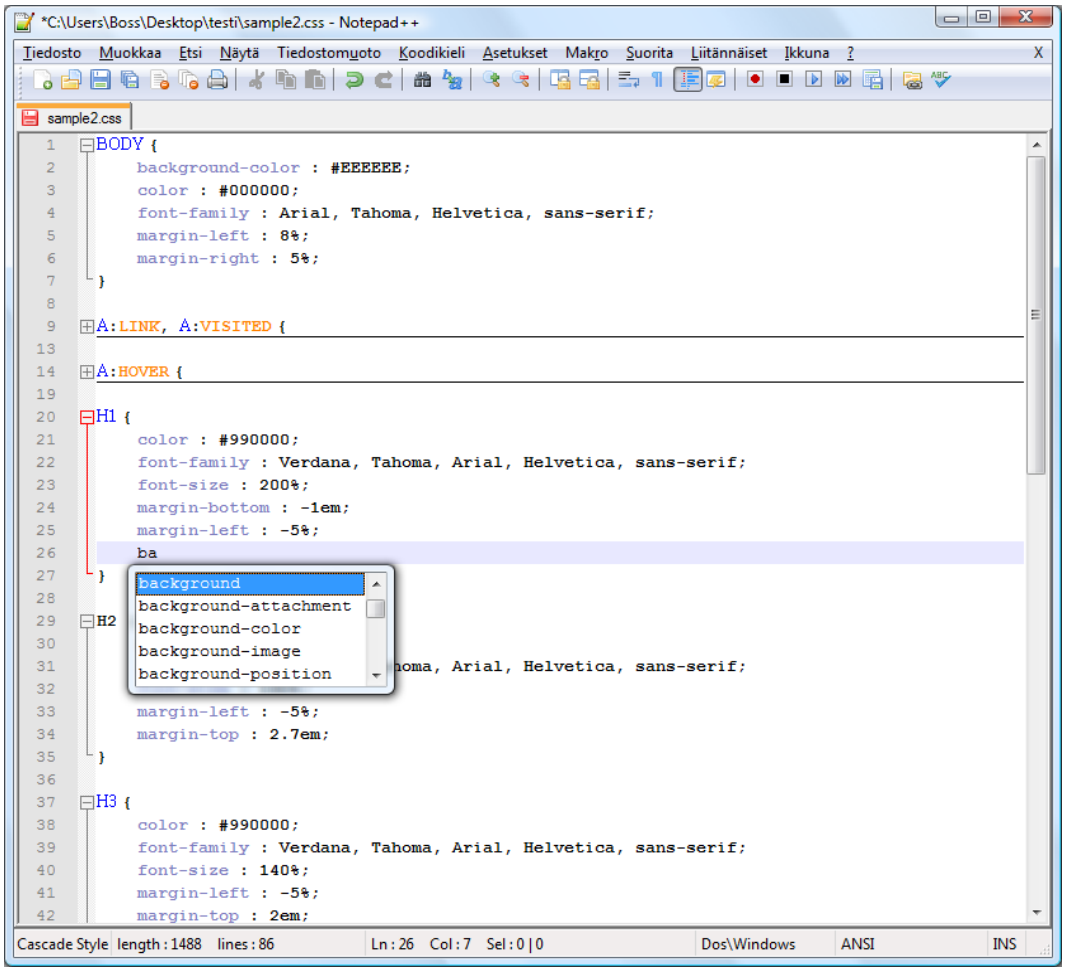
4.3 Ilmaiset ja kaupalliset editorit

Vaikka pelkällä tekstieditorilla CSS-tiedostojen luonti onnistuu ammattilaiselta melko vaivattomasti, voi tehokas CSS-editori helpottaa, nopeuttaa ja selkeyttää huomattavasti hänenkin työtään. Varsinkin hieman kokemattomampi käyttäjä hyötyy suuresti editorin tuomista eduista. Seuraavaksi esitellään erityylisiä editoreita, joista toiset ovat tekstieditorimaisia ja toiset hieman graafisempia. Osa editoreista on tietokoneeseen asennettavia sovelluksia ja osa on selaimella toimivia ohjelmia, jotka liitetään käyttäjän verkkosivuihin tai asennetaan selaimeen laajennukseksi.

4.3.1 Notepad++

Notepad++ on Windowssilla toimiva GPL-lisenssiin pohjautuva ilmainen tekstieditori. Se on todella pienikokoinen ja kevyt ohjelma, joka tukee useiden eri ohjelmointikielien syntaksin korostusta. (Notepad++ 2012)

Notepad++-ohjelman toimintoihin kuuluu käyttäjän kirjoittamien CSS syntaksien korostus, vihjelista valittavista tyyliominaisuuksista, sekä mahdollisuus tyyliluokan pienennyksen ja avaamisen. Notepad++ onkin lähinnä vain tekstieditori, jossa on muutama CSS-tyylitiedoston tekemistä helpottava asia (Kuva 21). Sen käyttäjäkunta muodostuu CSS syntaksit ulkoa muistavista ammattilaisista, jotka haluavat vain yksinkertaisen ohjelman, jolla koodin kirjoitus onnistuu ilman mitään ylimääräistä säätämistä.



Kuva 21. Notepadd++-editorin käyttöliittymä

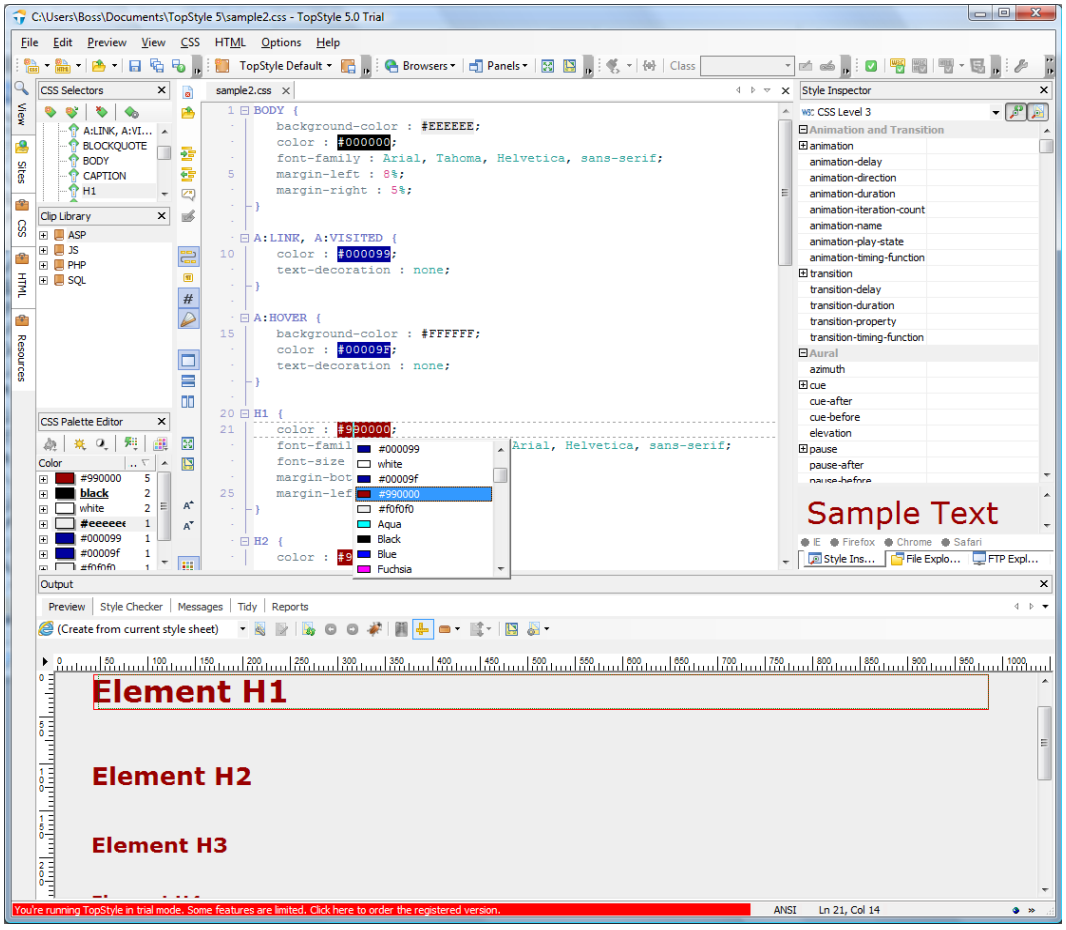
4.3.2 TopStyle 5.0

TopStyle 5 on Windows-käyttäjille suunnattu CSS- ja HTML-editori, josta on ladattavissa ilmainen hieman rajoitettu versio ja 79,95 dollaria maksava täysversio. Topstyle 5 on hyvin monipuolinen ohjelma, jossa on lukuisia CSS- ja HTML-tiedostojen luomista helpottavia ominaisuuksia.

Editorin käyttöliittymä on tekstieditorimainen ohjelma jonka ikkuna on jaettu useisiin osiin, joissa jokaisessa on monia eri valikoita (Kuva 22). Näkymä on tuttu vastaavanlaisista ohjelmointisovelluksista, mutta ensikertalaiselle se voi olla todella sekavan näköinen, silla kaikki mahdolliset ominaisuudet on heti asetettu näkyville. Hetken aikaa näkymään totuteltua alkaa se tuntua kuitenkin selkeältä ja

vaikka ohjelman ulkoasu on graafisesti melko karu, on se kuitenkin tehokas ja informatiivinen.

CSS-tyylimäärittelyjen tekeminen onnistuu perinteiseen tapaan kirjoittamalla tai oikeassa reunassa olevasta kattavasta valikosta josta pystyy vain valitsemaan haluamansa ominaisuuden ja sen arvon. Lisäksi osan määrittelyistä, kuten sijainti, reunukset ja valitsimet, saa tehtyä omissa erillisissä informatiivisissa ikkunoissa, joissa ominaisuuksien arvojen muutokset huomaa välittömästi. TopStyle 5:ssä on myös näkyvissä esikatseluikkuna, jossa käyttäjä huomaa suoraan tekemiensä muutosten vaikutukset. Esikatselunäkymässä voi lisäksi valita minkä selaimen tyyliin ikkuna toteuttaa CSS-määrittelyjä. Editorissa pystyy tarkistamaan koodin oikeellisuuden editorin omalla tarkistimella tai W3C CSS Validator-ominaisuudella. Ohjelmalla voi myös järjestellä, siistiä ja tiivistää koodia. Editorissa on lisäksi vielä lukuisia muita määrittelyjen tekemistä helpottavia ominaisuuksia.



Kuva 22. Topstyle 5.0-editorin käyttöliittymä

TopStyle 5 on todella kattava ohjelma, josta löytyy paljon hyödyllisiä ominaisuuksia niin aloittelijalle, kuin ammattilaisellekin. Ohjelma vaatii käyttäjältä hieman CSS-kielen tuntemusta ja aluksi pientä tutustumista käyttöliittymään, mutta sen jälkeen se on todella tehokas työkalu tyylimäärittelyjen tekemiseen alusta loppuun.

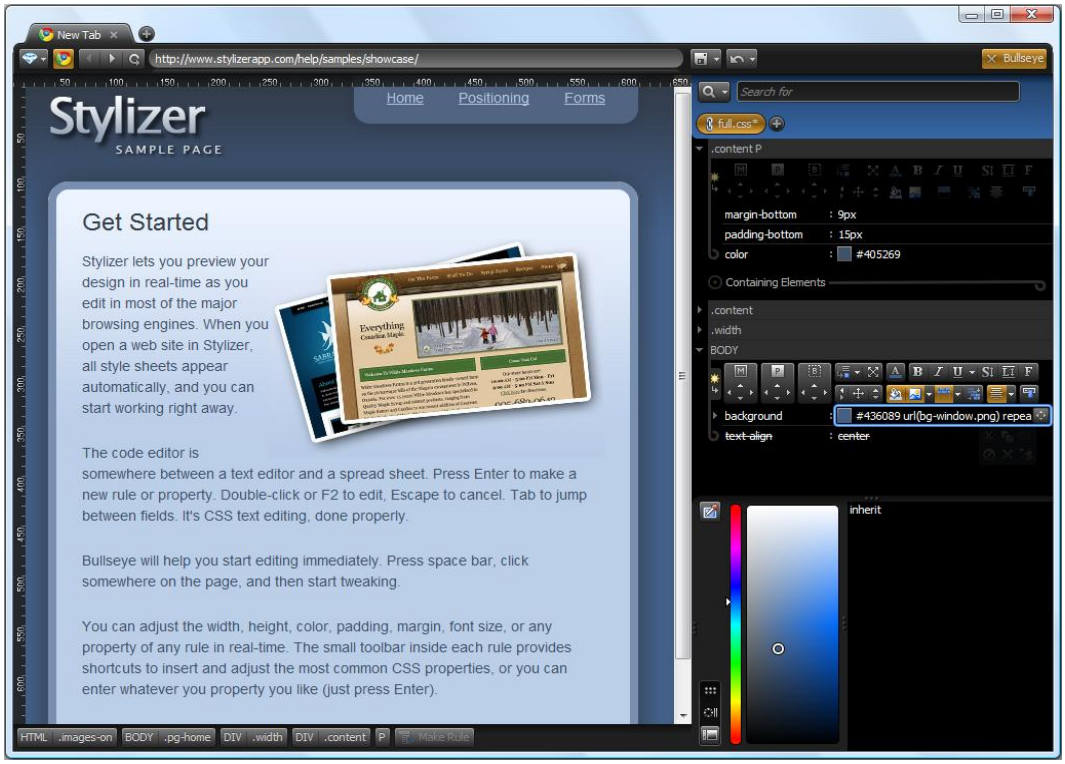
4.3.3 Stylizer 5

Stylizer 5 on Skybound Softwaren Windows ja Machintos käyttäjille kehittämä tyylikäs ja visuaalisen käyttöliittymän omaava CSS-editori. Ohjelman verkkosivuilla on ladattavissa ilmainen kokeiluversio, jossa on kahden viikon koeaika ja rajoitetut ominaisuudet, jonka jälkeen täytyy ostaa täysversio hintaan 103.99 euroa.

Ohjelman päänäkymä on jaettu kahteen osaan, joista vasemman puoleisessa näkyy suuri esikatseluikkuna ja oikeassa reunassa on valikoiden ja nappuloiden avulla toimiva tyylimuotoilujen muokkaus ja luominen (Kuva 23). CSS-muotoiluja ei tarvitse kirjoittaa, vaan kaikki ominaisuudet ja arvot löytyvät valikoiden tarjoamista vaihtoehdoista. Ohjelma useasti myös selittää, käyttäjälle näytettävien vihjeiden avulla, mitä milläkin nappulalla saa tehtyä. Tyylimuotoilun tapahtuessa valikoiden avulla jää käyttäjän kirjoittamat virheet kokonaan pois ja CSS-koodin pitäisikin olla toimivaa, kunhan selain vain vielä osaa näyttää määrittelyt oikein. Lisäksi käyttäjä näkee muutostensa vaikutuksen vieressä olevasta esikatselukuvasta reaaliajassa. Esikatselussa käytettävää selainta pystyy vaihtamaan Internet Explorerin, Chromen ja Firefoxin välillä ja lisäksi on valittavissa kyseisten ohjelmien eri versioita.

Stylizer 5 on tyylikkään näköinen ja kätevän oloinen editori, jolla myös vähemmän CSS:ään tutustunut henkilö pystyy melko vaivattomasti saamaan vaikuttavan oloisia tyylimäärittelyjä aikaiseksi. Joitakin kokeneempia käyttäjiä saattaa häiritä tekstieditoriominaisuuden puuttuminen, mutta tästä editorista onkin haluttu tehdä täysin erilainen. Myös kokeneempi käyttäjä voi hyötyä editorin helposta käytettä-

vyydestä ja voi nopeasti kokeilla erilaisia ja erikoisia määrittelyjä editorin esikatseluikkunan avulla.



Kuva 23. Stylizer 5-editorin käyttöliittymä

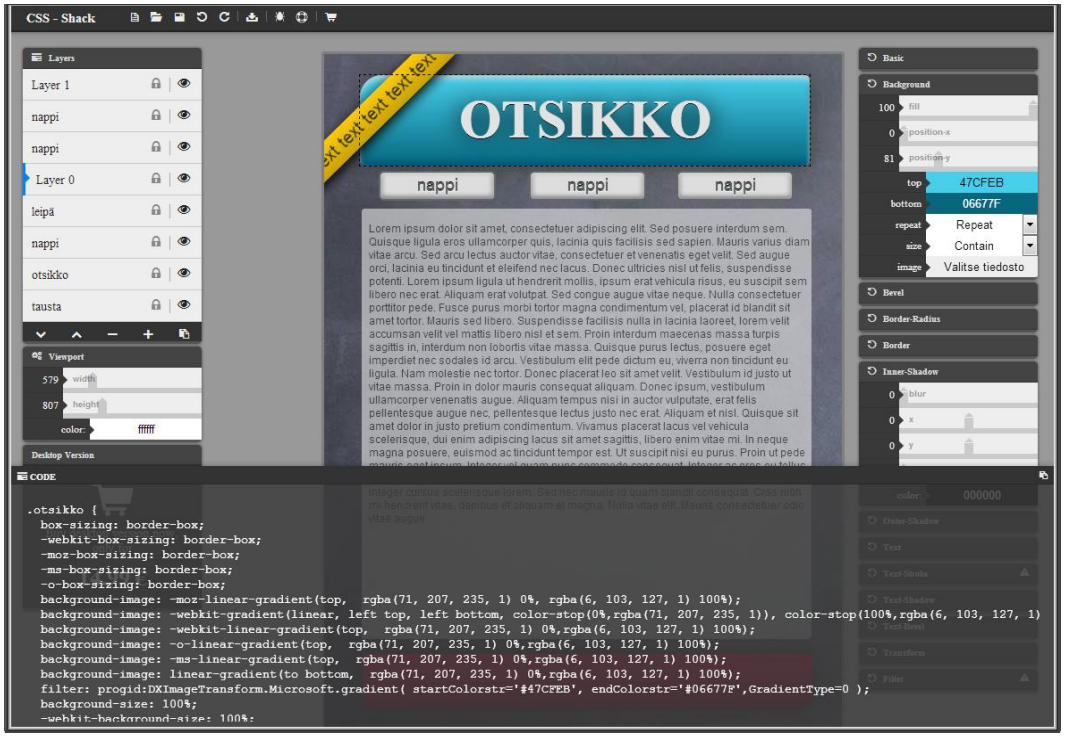
4.3.4 CSS-Shack

CSS-Shack on Googlen Chrome selaimeen asennettava sovellus, jolla pystyy luomaan css-tyylimäärittelyjä html-elementeille. Ohjelmasta löytyy myös linuxille, Windowsille ja machintosille asennettava maksullinen työpöytäsovellus, jonka hinta on vain 9,99 euroa.

Ohjelma on tyylikkään ja selkeän näköinen ja sen visuaalinen käyttöliittymä on todella helppokäyttöinen (Kuva 24). Ohjelmassa pystyy luomaan uusia elementtejä esikatseluikkunaan ja muuttamaan niiden sijaintia raahaamalla niitä hiirellä. Elementteihin pystyy myös kirjoittamaan tekstiä, jonka jälkeen oikeassa laidassa olevien valikoiden avulla saa muokattua elementtien ulkonäköä.

Valikoiden avulla tapahtuva tyylimäärittelyjen tekeminen on todella vaivatonta ja esikatseluikkunassa välittömästi näkyvät muutokset tekevät siitä myös todella

helppoa. CSS-Shack ohjelmalla pystyy nopeasti luomaan haluamansa tyylimäärittelyn elementille tai luonnoksen koko sivun ulkoasusta. Ohjelmasta voi kopioida yksittäisen elementin tyylimäärittelyn css-koodin tai tallentaa koko luonnoksen määrittelyt css-tiedostoksi.



Kuva 24. CSS-Shack-editorin käyttöliittymä

4.3.5 CSS 3.0 Maker

CSS 3.0 Maker on javascriptin avulla toteutettu internet-sivusto, jossa pystyy valikoiden avulla säätämään erilaisia CSS 3-tyylimäärittelyjä (Kuva 25). Sivusto on WebiBeris yrityksen toteuttama projekti ja sen käyttö on täysin ilmaista.

Sivustolla pystyy kokeilemaan kymmenen erilaisen uuden CSS 3-tyylimäärittelyn vaikutusta, pudotusvalikoiden, valintanappien ja liukusäätimien avulla. Vaihdettaessa ominaisuuksien arvoja, näkee välittömästi muutosten vaikutuksen esikatseluikkunassa olevaan elementtiin, lisäksi myös oikealla olevassa koodin esikatseluikkunassa muutokset tapahtuvat reaaliajassa. Saatuaan ominaisuudelle halutut

arvot pystyy valmiin CSS-koodin kopioimaan suoraan sen omasta esikatseluikkunasta tai tallentamaan sen html-tiedostona koneelle.

CSS 3.0 Maker-sivusto näppärä paikka, jossa kokenut ja kokemattomampi nettisivujen suunnittelija voi helposti luoda ja testata erilaisia tyylimäärittelyjä. Sivuston ulkoasu on todella selkeä ja sen käytön oppii nopeasti, kaikki tarpeellinen on näkyvissä samaan aikaan, eikä sivuston käyttö vaadi suurempaa tuntemusta CSS-koodista tai vastaavanlaisesta ohjelmasta. Suoraan selaimella toimivan ohjelman etuna on se, että kokeilemalla eri selaimilla CSS 3.0 Maker-sivustoa käyttäjä näkee miltä erilaiset ominaisuudet oikeasti näyttävät käytössä olevalla selaimella. Sivusto myös ilmoittaa suoraan minkä selaimen milläkin versiolla kyseisellä hetkellä muokattavan CSS-tyylimäärittelyn pitäisi toimia.

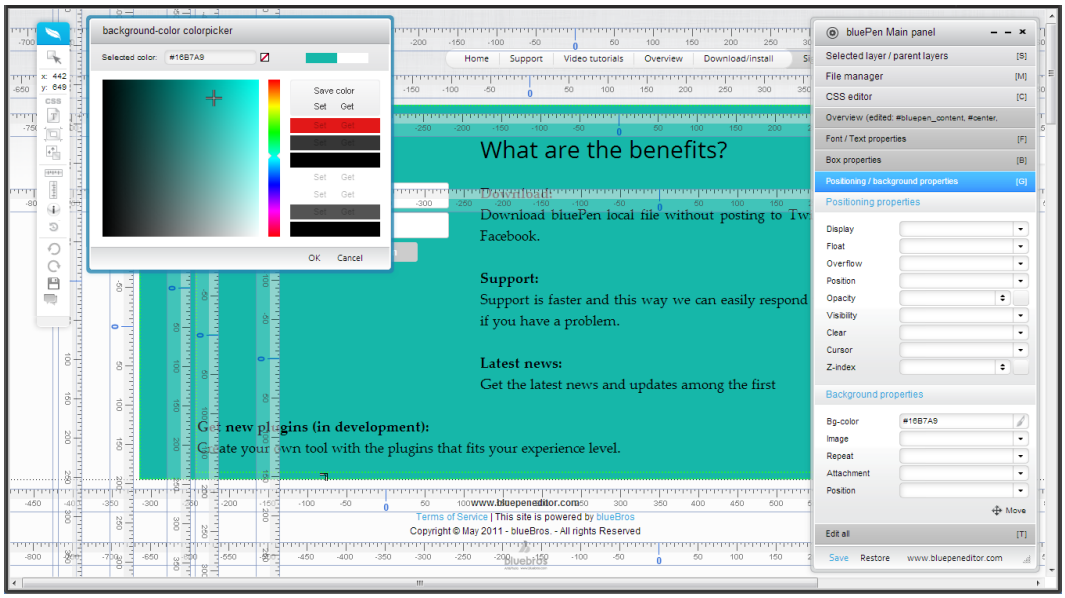
Kuva 25. CSS 3.0 Maker-editorin käyttöliittymä

4.3.6 bluePen

bluePen on Bluebros-yrityksen tekemä suoraan selaimessa toimiva CSS-tiedostojen muokkaamiseen tarkoitettu ohjelma. Ohjelman on maksuton ja sen lataaminen vaatii vain käyttäjältä yhden Twitter- tai Facebook-ilmoituksen asiasta. Saadakseen ohjelman toimimaan omilla verkkosivuillaan täytyy käyttäjän lisätä lataamansa php-tiedosto palvelimelleen ja yksi rivi javascript-tekstiä sivulleen. Lisättyään bluePen-ohjelman upotuskoodin sivulle ilmestyy sen oikeaan yläreunaan painike, josta klikkaamalla ohjelma käynnistyy.

Ohjelman avulla käyttäjä voi klikata sivultaan haluamaansa elementtiä ja alkaa sen jälkeen muokkaamaan sen tyylimäärittelyjä bluePenin valikoiden avulla (Kuva 26). Valikoista näkee millaisia ominaisuuksia ja arvoja elementille on annettu ja niitä pystyy myös vaihtamaan liuku- ja pudotusvalikoiden avulla. Arvojen muutoksen näkee välittömästi selaimen ikkunalla. Valikoiden lisäksi tyylimäärittelyjä pystyy muokkaamaan myös perinteisessä tekstimuodossa. Tehtyään elementille uudet tyylimäärittelyt pystyy käyttäjä tallentamaan ne CSS-tiedostoonsa tai palauttamaan alkuperäiset asettelut.

Pienen alkutotuttelun jälkeen ohjelman käyttö on melko selkeää ja sillä on helppoa muokata valmista tyylimäärittelyä tai tehdä kokonaan uusi. Valikot ovat selkeät ja antavat kattavat vaihtoehdot valitun ominaisuuden uudeksi arvoksi. Selaimella tapahtuva välitön muutos antaa lisäksi käyttäjälle selkeän kuvan siitä, miten uusi arvo vaikuttaa sivujen ulkoasuun ja kuinka selain näyttää sen. Ohjelma haittapuolia on sen ajoittainen hitaus, kun ohjelma lataa valittuja tyylimäärittelyjä ja se ettei se toimi kaikilla selaimilla, kuten esimerkiksi Internet Explorer-selaimella. Kaiken kaikkiaan ohjelma on kuitenkin todella näppärä ja erilainen verrattuna muihin CSS-editoriohjelmiin. bluePen-ohjelmalla on kätevä viimeistellä esimerkiksi tekstieditorilla aloitettu tyylimäärittely ja saada asettelut yhtenäisiksi ja lopulliseen muotoonsa, sillä suoraan selaimessa toimivana ohjelmana se paljastaa, miltä lopputulos todellisuudessa näyttää.



Kuva 26. bluePen-editorin käyttöliittymä

5 KÄYTTÖLIITTYMÄSUUNNITTELU

Käyttöliittymäsuunnittelu on tärkeässä osassa uutta ohjelmaa tehtäessä. Se vaikuttaa ratkaisevasti siihen, palaako ohjelmaa ensimmäistä kertaa kokeileva henkilö uudestaan sen pariin. Esteettisesti näyttävä ja paljon hienoja ja hyödyllisiäkin ominaisuuksia sisältävä ohjelma on melko hyödytön, ellei käyttöliittymäsuunnitteluun ole keskitytty yhtään. Käyttäjälle tarpeellinenkin ohjelma saattaa jäädä hyödyntämättä, mikäli hän tuntee ohjelman käytön hankalaksi ja tehottomaksi.

Käyttöliittymäsuunnitteluun panostamalla pyritään samaan aikaiseksi mahdollisimman käyttäjäystävällinen ohjelma. Ohjelmankehityksen kaikissa vaiheissa tehtävät ratkaisut voivat vaikuttaa käytettävyyteen ja tämä tuleekin pitää koko ajan mielessä, sillä hyvä käytettävyys ei ole sellainen ominaisuus jonka voi vain lisätä ohjelmaan projektin lopussa.(Wiio 2004, 9).

5.1 Käytettävyys

5.1.1 Ymmärrettävyys

Ohjelman pitää olla ymmärrettävä, jotta käyttäjän on helppo tajuta miten hän pääsee haluamaansa lopputulokseen. Käyttäjän tulee kyetä, ilman erillisiä ohjeita, päättelemään mitä ohjelmalla pystyy tekemään ja löytämään helposti kaikki toiminnot. Ohjelman tulee olla ymmärrettävä kaikille käyttäjille, eikä vain aiheeseen tarkemmin perehtyneille.(Wiio 2004, 29–30.)

Jotta ohjelma olisi helposti ymmärrettävä, täytyy sen keskustella käyttäjän kanssa hänen tarpeisiinsa ja näkökulmaansa liittyvillä käsitteillä. Ohjelman tekniseen toteutukseen liittyvät käsitteet eivät ole käyttäjälle läheskään yhtä selkeitä, kuin ohjelman luoneelle henkilölle. Ohjelman tekijöiden täytyy miettiä tilanteita käyttäjän näkökulmasta, jotta käyttäjä ei joudu miettimään ja kokeilemaan, mitä mikäkin termi ja ikoni tarkoittaa.(Wiio 2004, 74.)

5.1.2 Vaivattomuus

Käyttäjän tulee suoriutua tehtävistään mahdollisimman vaivattomasti. Ohjelma saattaa olla täysin ymmärrettävä, mutta silti vaivalloinen. Vaikka käyttäjä osaisikin käyttää ohjelmaa, saattaa hän kokea sen niin vaivalloiseksi ja aikaa vieväksi, ettei enää hyödynnä siinä tarjolla olevaa ominaisuutta. Käyttötilanteita voi olla useita ja erilaisia ja niitä tulee miettiä käyttäjän tarpeitten mukaisesta näkökulmasta(Wiio 2004, 84). Hänellä tulee olla kaikissa tilanteissa tarpeelliset tiedot ja ominaisuudet, jotta hän kykenee ratkaisemaan tehtävän vaivattomasti, eikä joudu rasittamaan itseään tarpeettomasti. (Wiio 2004, 30.)

Käyttäjällä on yleensä kokemusta tietokoneohjelmista ja hän etsiikin uudesta ohjelmasta tuttuja ja tunnistettavia asioita(Wiio 2004, 132). Tätä kannattaa hyödyntää ja laittaa painikkeet ja valikot niille paikoille mistä käyttäjä on niitä yleensä tottunut etsimään. Kaikki tarpeelliset ja hyödylliset toimintamahdollisuudet tulisi olla helposti esillä, jotta käyttäjä osaa hyödyntää niitä. Toiminnot ja valikot tulee olla johdonmukaisia, jotta käyttäjä tietää vaivattomasti kokemustensa pohjalta mitä mikäkin nappula tekee ja missä se sijaitsee(Wiio 2004, 153). Käyttäjän tulee saada myös välitöntä palautetta suoritetuista toimenpiteistä, jotta hän tietää toimenpiteiden tulokset ja vaikutukset(Wiio 2004, 145). Toimintojen peruutettavuus on tärkeää, koska se tekee ohjelmasta helposti tutkittavan ja käyttäjällä on turvallinen olo kokeillessaan erilaisia vaihtoehtoja.(Wiio 2004, 145).

5.1.3 Kattavuus

Kattavassa ohjelmassa on kaikki ne toiminnot ja tiedot, joita käyttäjä tarvitsee ohjelmalla halutun tehtävän suorittamiseen. Kattavuuden ja vaivattomuuden ongelmat esiintyvät usein yhdessä. Mikäli ohjelmassa ei ole kaikkia tarpeellisia ominaisuuksia ja tietoja halutun tehtävän suorittamiseen, joutuu käyttäjä vaivaamaan itseään tavallista enemmän.(Wiio 2004, 31.)

5.1.4 Esteettisyys

Ulkoisesti tyylikkään näköinen sovellus viestittää käyttäjälle laatua ja osaamista. Ruman näköinen ohjelma saa käyttäjän helposti epäilemään sen laatua ja hän saattaa hylätä muuten toimivan ohjelman pelkän ulkonäön takia. Esteettisesti hyvin suunniteltu ohjelma on selkeä, uskottava ja ohjaa käyttäjän huomiota haluttuihin asioihin.(Wiio 2004, 31.)

5.2 Suunnittelu

Uutta ohjelmaa suunniteltaessa keskeisessä osassa ovat käyttäjät ja heidän pyrkimyksensä, ja aluksi onkin selvitettävä käyttäjäystävällisyyttä edistävät suunnittelutavoitteet. Tässä apuna toimii systeemisuunnittelu, jonka tarkoituksena on tuottaa käyttöliittymän suunnittelussa tarvittava ymmärrys ohjelman tarkoituksesta ja tehtävistä sekä käyttäjien käsitteistä, tilanteista ja tarpeista. Näin käyttöliittymää suunniteltaessa tiedetään mitä ohjelmalta halutaan ja osataan etsiä ratkaisuja oikeisiin kysymyksiin.(Wiio 2004, 18–19.)

Ensin on kuitenkin tiedettävä ne yhdenmukaiset ja johdonmukaiset yleiset periaatteet, joiden mukaan käyttöliittymä toimii, ennen kuin aletaan suunnitella käyttöliittymän yksityiskohtia. Periaatteiden pohtimisessa käytetään apuna esimerkkitapauksia, prototyyppejä ja luonnoksia, joita sitten analysoidaan ja testataan. Näiden keinojen avulla periaatteita määritellään paremmiksi ja työvaiheita toistetaan kunnes on päästy haluttuun lopputulokseen.(Wiio 2004, 212.)

5.2.1 Käyttäjät

On tärkeä selvittää käyttäjän pyrkimykset ja tarpeet, sillä puutteet käyttäjän toiminnan ymmärtämisessä voivat aiheuttaa sellaisia käytettävyysongelmia, joita ei vain käyttöliittymän yksityiskohtia parantelemalla pysty korjaamaan. Pyrkimystä ja siihen kuuluvaa tekemistä nimitetään prosessiksi(Wiio 2004, 90). Joskus käyttäjän pyrkimykset ja prosessit on helppo tunnistaa, mutta monet ohjelmista ovat yleisluontoisia työkaluja monenlaisiin prosesseihin. Tällaisia ohjelmia voidaan

lähestyä luokittelemalla käyttäjiä, sekä tunnistamalla kaikille käyttäjille yhteisiä pyrkimyksiä. (Wiio 2004, 93.)

Käyttäjän pyrkimysten ja prosessien tunnistamisen jälkeen on selvitettävä ne tilanteet, joissa ohjelma ja käyttäjä ovat tekemisissä keskenään. Käyttötilanteet löytyvät parhaiten mallintamalla prosesseja ja usein riittääkin että tehdään jäsennys niistä tapahtumista ja toimenpiteistä, jotka vievät prosessia eteenpäin. Muodostetaan kuva tapahtumista ja vaiheista joita käyttäjä kohtaa, jäsennellään vaiheet ja mietitään minkälaisia toiminnallisuuksia näiden vaiheiden avuksi voitaisiin tarjota. (Wiio 2004, 94–95.)

Pyrkimysten, prosessien ja käyttötilanteiden tunnistamisen jälkeen seuraava askel kohti käyttäjäystävällisempää ohjelmaa on käyttäjän tarpeiden selvittäminen. Käyttäjillä voi olla samanlainen tapahtuma, mutta erilaiset tarpeet. Heidän prioriteettinsa voivat olla niin erilaiset, että niiden palveleminen vaatii erilaisia käyttöliittymiä. Kannattaakin miettiä sellaisia näkökulmia, jotka ovat hyödyllisiä monille erilaisille käyttäjille.

5.2.2 Keskustelukäyttö

Ohjelmissa olevien tehtävien tekemiseen on olemassa kaksi erilaista käyttämisen tapaa eli vuorovaikutusmallia, joita kutsutaan suorakäsittelyksi ja keskustelukäytöksi. Suorakäsittelyssä käyttäjällä on edessään olioita, joihin hän voi kohdistaa erilaisia toimenpiteitä, joiden vaikutukset hän näkee välittömästi. Keskustelukäyttö tapahtuu yleensä valikoiden ja lomakkeiden avulla. Piirtämisohjelmat ovat suorakäsittelyyn perustuvia ohjelmia, kun taas taloushallinnon ohjelmat ovat vastaavasti usein täysin keskustelukäyttöisiä. Osa ohjelmista sisältää kuitenkin osia molemmista vuorovaikutusmalleista. (Wiio 2004, 125.)

Keskustelukäyttöisen sovelluksen periaatesuunnittelussa käyttöliittymän jaottelu toiminnallisiin osiin on keskeisessä osassa. Tämän jaottelun pohjalta muodostuu ohjelman valikkorakenne, joka pitäisi rakentaa käyttäjän tavoitteiden ja käsitteiden mukaiseksi. Jaottelun lisäksi on paljon pieniä ja keskikokoisia asioita, jotka on toteutettava yhdenmukaisella ja johdonmukaisella tavalla. Näitä asioita ovat esimerkiksi yhtenäinen visuaalinen ilme, terminologia, tiedon esitystavat ja va-

kiomuotoiset dialogit. Näiden asioiden standardointi on tärkeää, jotta käyttäjän on helpompi oppia käyttämään ohjelmaa, kun samanlaisissa tilanteissa aukeaa tutulta näyttävä valikko, joka toimii odotetulla tavalla.(Wiio 2004, 216–217.)

5.3 Testaus

Käytettävyystestaus on tärkeä osa käyttäjäystävällisen ohjelman suunnittelua, sillä näin saadaan selville käyttäjien mielipiteitä ohjelman toimivuudesta. Suunnittelija tulee myös helposti sokeaksi omille virheilleen eikä huomaa niitä, mutta käyttäjätestissä ulkopuolinen koehenkilö katsoo ohjelmaa tuoreesta näkökulmasta. Testaamalla saadaan hyödyllistä ja konkreettista tietoa siitä, miten käyttäjät tulkitsevat käyttöliittymää ja löytävät tarvitsemiaan tietoja ja toimintoja (Wiio 2004, 66). Testaamalla voidaan myös selvittää, kuinka helppoa ohjelman käytön aloittaminen on ja kuinka tehokasta sillä työskentely on (Wiio 2004, 66). Testaaminen kannattaa aloittaa mahdollisimman aikaisin, jotta vältytään suurilta ja aikaa vieviltä ohjelman perusrakenteiden muutoksilta (Wiio 2004, 218).

5.3.1 Pikatestaus

Pikatestauksen ideana on, että se voidaan toteuttaa melko nopeasti, vaivattomasti ja niin useasti kuin tarpeen jo suunnittelun alkuvaiheessa. Testien tavoitteena on todeta, ovatko ohjelman perusratkaisut ymmärrettäviä (Wiio 2004, 227). Koehenkilöiden nopean ja helpon saatavuuden takia heidän ei tarvitse välttämättä edustaa varsinaista ohjelman käyttäjäpopulaatiota. Koekäyttäjiä voi myös kierrättää, kunhan heidän uusiokäyttönsä ei kohdistu samoihin toimintoihin, joita he ovat aikaisemmin testanneet. (Wiio 2004, 221–222.)

Pikatestit ovat usein vain muutaman toimenpiteen mittaisia ja niiden tulisikin testata vain yhtä asiaa. Toimivan prototyypin tekeminen vie usein paljon aikaa, joten siksi pikatesteissä käytetään suunnitellun käyttöliittymän kuvia. Käyttäjälle esitellään lähtötilanne ja tehtävä ja sitten kysytään miten hän lähtisi sitä toteuttamaan. Testin lopuksi käyttäjältä voidaan lisäksi kysyä, miten päättyneestä tilanteesta jatkettaisiin eteenpäin ja muita huomioita käyttöliittymästä. Testiä tehdessä tulee

olla huolellinen etteivät ohjeet ja kysymykset ole yhtään käyttäjän toimintaa ohjailevia. (Wiio 2004, 218–220.)

Projektin ulkopuolinen testaaja olisi paras vaihtoehto testien objektiivisen suorittamisen ja tulosten kirjaamisen kannalta. Pikatestejä tulisi tehdä kuitenkin melko pian uusien luonnostelujen jälkeen ja tämä on huomattavasti helpompaa jos testaajat on omasta takaa ja koehenkilötkin saadaan lähipiiristä. Ulkopuolisen asiantuntijan käyttöä kannattaa kuitenkin jossain vaiheessa harkita, sillä hän saattaa tuoda uusia näkökulmia testaamiseen. (Wiio 2004, 223–224.)

5.3.2 Heuristinen arviointi

Heuristinen arviointi on hyvä, nopea, yksinkertainen ja edullinen tapa etsiä käyttöliittymän ongelmakohtia. Siinä käydään läpi käyttöliittymän osat muistilistassa olevien erilaisten käytettävyysperiaatteiden avulla. Arvioinnissa voidaan käyttää ohjelmaa, sen ja käyttöliittymän suunnitelmia tai varhaisia prototyyppejä, ja näin arvioinnin tuloksia voidaan hyödyntää erivaiheissa kehitystyötä. Heuristisessa arvioinnissa ei oteta kantaa ohjelman hyödyllisyyteen, vaan käyttöliittymän toimivuuteen. (Riihiaho 2004, 2.)

Arvioinnin suorittavalla henkilöllä tulisi olla kokemusta käytettävyydestä, tutkittavasta ohjelmasta ja sen tarkoituksesta, mutta hyödyllisiä tuloksia saadaan myös kokemattomamman henkilön muistilistan avulla suorittamasta arvioinnista. Arvioijat käyvät ohjelman yksin huolella läpi ja kirjaavat ongelmakohdat ja niiden vakavuuden ylös. Ongelmat kasataan lopuksi yhdelle priorisoidulle listalle niiden vakavuuden mukaan ja sitten keskustellaan ryhmässä tuloksista ja parannusehdotuksista. (Riihiaho 2004, 2-4.)

Arvioinnissa käytetään yleensä apuna Nielsenin ja Molichin kymmenen säännön listaa:

- Käytä yksinkertaista ja luonnollista dialogia
- Käytä käyttäjien omaa kieltä
- Minimoi käyttäjän muistikuorma
- Tee käyttöliittymästä kauttaaltaan yhdenmukainen

- Anna käyttäjälle palautetta toiminnoista
- Anna selkeä poistumistapa eri tiloista ja toiminnoista
- Anna käyttäjälle mahdollisuus käyttää oikopolkuja
- Anna virhetilanteista selkeät virheilmoitukset
- Vältä virhetilanteita
- Anna riittävä ja selkeä apu ja dokumentaatio

Listan ohjeet ovat melko yleisluontoisia, joten niistä voi tehdä paremmin omaa ohjelmaa vastaavan version. Ohjelman aikaisemmissa versioissa havaitut ongelmat auttavat tarkentamaan listaa. (Riihiaho 2004, 5.)

6 CASE: CSS-EDITORIN LUOMINEN

Opinnäytetyön case-projektina on oman toimivan CSS-editorin suunnitteleminen ja ohjelmoiminen. Ohjelmointi tapahtuu käyttäen Microsoft Visual Studiota ja koodikieleksi on valittu C# (C sharp). Tavoitteena on saada aikaiseksi mahdollisimman toimiva, helppokäyttöinen ja monipuolinen editori, jonka käytöstä hyötyvät kaiken tasoiset käyttäjät.

Yhtenä suurena haasteena tässä projektissa on itse editorin ohjelmointi, sillä aikaisempi tietämys C#-kielestä on melko vähäinen, eikä aikaisempaa kokemusta tämän kokoluokan ohjelman tekemisestä ole. Editorin monipuolisuus riippuukin sille asetettujen tavoitteiden saavuttamisesta tämän projektin aikataulun sallimissa rajoissa. Ohjelman käytettävyyteen pyritään vaikuttamaan käyttöliittymäsuunnittelun ja käyttäjätestauksen avulla.

6.1 Tavoitteiden määrittäminen

Ohjelmalle pitää asettaa jonkinlaiset vaatimukset käyttäjäkunta huomioiden, jotka sen tulee saavuttaa, jotta siitä tulisi käyttäjän kannalta hyödyllinen ja järkevä ohjelma (Kuva 27). CSS-Editorin sisältämät ominaisuudet on mietitty kolmeen eri tavoitekategoriaan, jotka ovat: vähimmäisvaatimukset, halutut ominaisuudet ja

tulevat ominaisuudet.

Kuva 27. Suunnitelma CSS-editoriin tulevista toiminnoista

Tärkein tavoite editoria tehtäessä on sen helppo käytettävyys. Käyttöliittymän tulisi olla selkeä ja selittävä, niin että se tukee sellaisia käyttäjiä, jotka eivät välttämättä ymmärrä mitään itse CSS-koodista. Editorin avulla tällaiset käyttäjät voisivat muokata ja päivittää heidän tekemiään tai heille tehtyjen sivujen ulkoasua muun muassa taustakuvien ja värien osalta. Editorin olisi myös tarkoitus olla käyttökelpoinen työkalu kokeneemmallekin nettisivujen tekijälle. Tämä käyttäjäryhmä

voisi editorin avulla luoda nopeasti yksinkertaisemmat CSS-tiedostot tai isomman tiedoston pohjan.

Editorin saavutettua vähimmäisvaatimukset ja toimintavalmiuden, tullaan siihen mahdollisesti lisäämään verkkosivujen tekemistä helpottavia toimintoja, kattavampien CSS-ominaisuuksien, mahdollisen esinäkymäikkunan, jonkin asteisen html-sivupohjan ja käyttäjätutkimuksessa ilmenneiden ehdotusten muodossa.

6.1.1 Vähimmäisvaatimukset

Vähimmäisvaatimukset on mietitty niin, että kun editorin ohjelmoinnissa on saavutettu kaikki siinä listatut asiat tulisi sen olla jo käyttökelpoinen, hyödyllinen ja järkevä ohjelma CSS-tiedostojen tekemiseen. Nämä ominaisuudet ovat ohjelman selkäranka, joita myöhemmin täydennetään ja monipuolistetaan ja joiden seuraksi lisätään uusia CSS-koodin ja verkkosivujen tekemistä helpottavia ominaisuuksia.

Ensimmäinen ohjelmalle asetettu vaatimus on se että sillä täytyy pystyä luomaan ja tallentamaan uusia CSS-tiedostoja, lisäksi sillä tulisi pystyä avaamaan, tulkitsemaan ja muokkaamaan jo olemassa olevia tiedostoja. Editorissa tulisi olla mahdollisuus kaikkein yleisimpien CSS-ominaisuuksien ja -arvojen määrittämiseen elementeille.

Ohjelman käyttöliittymän pitää olla niin selkeä ja informatiivinen, että käyttäjä tietää koko ajan, (vaikka ei tuntisikaan CSS-koodia) mitä vaihtoehtoja hänellä on kyseisen elementin muokkaamiseen ja millainen vaikutus niillä on. Lisäksi editorin tulisi varoittaa ja estää käyttäjää tekemästä epäkelpoa koodia.

6.1.2 Halutut ominaisuudet

Halutuilla ominaisuuksilla tarkoitetaan niitä asioita, joita editoriin pyritään lisäämään sen saavutettua kaikki perustoiminnan kannalta pakolliset vaatimukset. Halutuilla ominaisuuksilla on tarkoitus lisätä ohjelman monipuolisuutta ja käyttömukavuutta. Näitä ominaisuuksia pyritään lisäämään ohjelmaan projektin aikataulun sallimissa rajoissa.

Lisättäviä ominaisuuksia olisivat CSS 3-tyylimäärittelyt, jotka olisi mahdollista valita editorissa käytössä oleviksi tai pois päältä. Ohjelma voisi myös kertoa mitkä selaimet tukevat mitäkin CSS 3-määrittelyjä. Editorissa voisi myös olla erilaisia valmiita muokattavia tyylipohjia, joita hyödyntämällä käyttäjä voi nopeasti luoda tyylitiedoston. Ohjelmalla voisi myös halutessaan kommentoida ja järjestellä tyylitiedoston valitsimia samankaltaisiin ryhmiin paremmin hallittaviksi. Editorissa voisi myös olla perinteinen tekstieditori-näkymä, jossa tyylitiedostoa pystyy muokkaamaan.

6.1.3 Tulevat ominaisuudet

Tulevilla ominaisuuksilla tarkoitetaan asioita, joita editoriin tullaan luultavasti lisäämään tämän case-projektin jo päätyttyä. Nämä ominaisuudet on arvioitu sen verran aikaa ja työtä vaativiksi tai vastaavasti alkuperäistä tavoitetta sivuaviksi, että niiden toteuttaminen näin alkuvaiheessa on suosiolla jätetty myöhemmäksi.

Editoriin myöhemmin lisättävistä ominaisuuksista tärkeimpänä on esikatseluikkuna. Esikatseluikkunassa käyttäjä näkee suoraan erilaisten tyylimäärittelyjen vaikutuksen elementtiin ja lisäksi ikkunassa olisi mahdollisuus vaihtaa näkymää eri selainten esitystavan mukaan. Esikatseluikkuna nopeuttaisi ja helpottaisi tyylimäärittelyjen tekemistä, sekä lisäisi käyttömukavuutta. Muita lisättäviä ominaisuuksia olisi tyylitiedoston lataaminen verkkoon ja useiden tiedostojen avaaminen samanaikaisesti. Nämä kaksi ominaisuutta helpottaisivat ja nopeuttaisivat tyylimäärittelyjen tekemistä, sillä käyttäjä pystyisi hallinnoimaan paremmin tiedostoja, eikä hänen tarvitsisi vaihtaa ohjelmaa tiedoston verkkoon lataamista varten.

6.2 Toteutustavan valitseminen

6.2.1 Selain- vai työpöytäsovellus

Toteutustavassa mietin työpöytä- ja selainsovelluksen välillä ja pohdin molempien hyviä puolia. Suoraan selaimessa toimivassa ohjelmassa olisi etuna, että siihen pääsisi kirjautumaan mistä tahansa paikasta, jossa on tietokone ja Internet-yhteys ja pystisi tekemään nopeasti pieniä päivityksiä. Selainsovelluksessa näkisi myös

välittömästi miltä tyylimäärittelyt oikeasti näyttäisivät kyseessä olevalla selaimella. Haittapuolena on selainten suuri määrä, lukuisat erilaiset versiot, tiuha päivitystahti, sekä erilaiset lisäosat ja laajennukset. Näin monen muuttuvan asian takia olisi luultavasti vaikea saada aikaiseksi ohjelma, joka toimisi moitteettomasti kaikilla mahdollisilla selaimilla ja niiden eri versioilla.

Työpöytäsovelluksissa on myös useampia eri alustavaihtoehtoja, kuten esimerkiksi Windows, Macintosh ja Linux. Windows on näistä kuitenkin selvästi käytetyin ja minulle tutuin ympäristö, joten muita alustoja en suuremmin edes harkinnut, vaan päätin tehdä CSS-editorista Windows sovelluksen. Yksi mielenkiintoinen vaihtoehto olisi myös ollut kasvavat älypuhelinmarkkinat, mutta koska minulla ei ole siitä alueesta minkäänlaista aikaisempaa kokemusta voisi se olla liian haastavaa ja laajemman ohjelman tekeminen puhelimelle voisi olla epäkäytännöllinen.

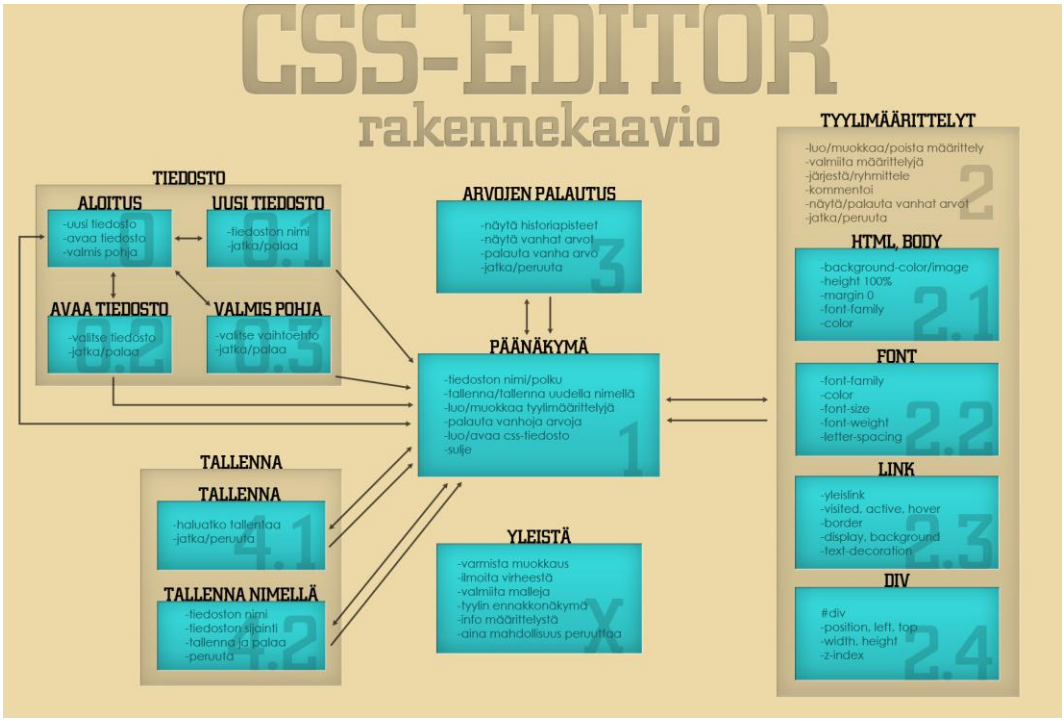
6.2.2 Ohjelmointikieli

Käytettävällä ohjelmointikielellä on vaikutus ohjelman laatuun ja se tulisikin valita niin että se sopii hyvin olemassa olevien ongelmien ratkaisuun. Tutustuttuani verkossa erilaisiin ohjelmointikieliin ja niillä toteutettuihin ohjelmiin, sekä kysyttyäni ammattilaisen mielipidettä, päädyin käyttämään tässä projektissa C#:a (C sharp) ja käytettävänä ohjelmana toimisi Microsoft Visual Studio 2010. Minulla on myös aikaisempaa kokemusta kyseisestä ohjelmointikielestä, joten se tuntui luontevalta valinnalta.

6.3 Rakenteen ja ulkoasun suunnittelu

Ennen ohjelman varsinaista ohjelmoimista tulee miettiä sen rakennetta, tehtävien suoritusjärjestystä ja ulkoasua. Halutuista ominaisuuksista muodostetaan graafinen algoritmi, eli vuokaavio, jonka avulla selvitetään tehtävien suoritusjärjestystä ja logiikkaa (Kuva 28). Vuokaavion avulla voidaan selvittää mihin mikäkin ominaisuus kannattaa sijoitta, jotta halutut tehtävät saadaan suoritettua järkevästi ja yhdenmukaisella tavalla. Ohjelman rakenne tulee miettiä tarkkaan ennen visuaalisen ilmeen suunnittelua, sillä se vaikuttaa suuresti ohjelman käytettävyyteen ja

hyvän suunnitelman pohjalta on huomattavasti helpompaa ja nopeampaa alkaa miettiä ohjelman visuaalista ulkonäköä.



Kuva 28. Suunnitelma CSS-editorin ranteesta

Editorin visuaalisesta ilmeestä tulee selkeä ja yksinkertainen, jotta käyttäjä havaitsee helposti kaikki erilaiset toiminnot. Sommittelussa käytetään perinteistä lähestymistapaa ja otetaan huomioon ihmisten jo erilaisissa ohjelmissa tulleet käyttöliittymätottumukset. Kaikki toiminnot pyritään tuomaan suoraan esille päänäkymään, ilman että niitä laitetaan erilaisten valikoiden taakse piiloon. Värimaailmassa käytetään hillitysti korostusvärejä, joilla tuodaan johdonmukaisesti esiin erilaiset toiminnot. Muutamille toiminnoille tulee yksinkertaiset symbolit, jotka säästävät tilaa pääikkunassa ja selkeyttävät ohjelman käyttöliittymää.

6.4 Käyttäjätutkimus ja parannusehdotukset

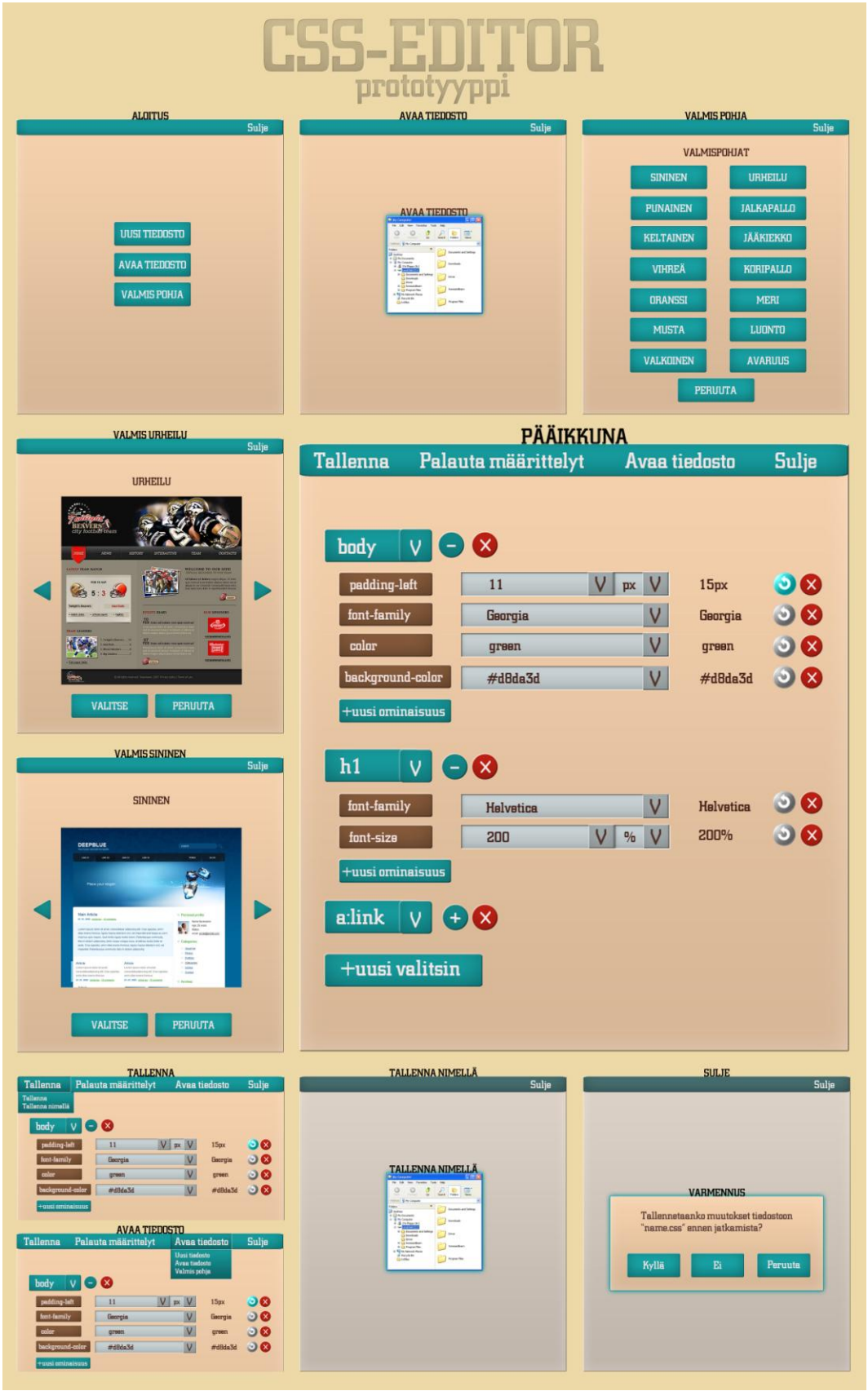
Käytettävyystestaus toteutettiin käyttämällä Photoshopissa tehtyjä suunnitelmia ohjelman eri näkymistä ja niiden toiminnoista (Kuva 29A ja 29B). Näiden kuvien avulla tehtiin yksinkertaiset verkkosivut, joihin lisättiin hieman toiminnallisuutta vastaamaan ohjelman toimintoja ja valikkoja. Näin saatiin tehtyä nopeasti proto-

tyyppi CSS-editorista, jonka avulla käyttäjät pääsevät testaamaan ohjelman käytettävyyttä.

Ohjelman loppukäyttäjäryhmä on hyvin vaihteleva, se kattaa verkkosivujen suunnittelun hyvin hallitsevat henkilöt, kuten myös sellaiset henkilöt jotka eivät välttämättä tiedä asiasta mitään, joten käyttäjätestin koehenkilöidenkin tulee vastata hyvin tätä ryhmää. Koehenkilöiksi valittiin yhteensä kahdeksan testaajaa erilaisen iän, tietokoneen yleisen käyttötaidon ja verkkosivujen toteutuksen osaamisen perusteella.

ikä:	*40–60 vuotta*	*25 %*
	20–30 vuotta	*75 %*
tietokoneen käyttötaito:		
	hyvä	*37,5 %*
	keskiverto	*50 %*
	heikko	*12,5 %*
verkkosivujen toteutuksen osaaminen:		
	hyvä	*25 %*
	keskiverto	*12,5 %*
	heikko	*62,5 %*

Koehenkilöitä ohjeistettiin tekemään kaksi erilaista tehtäväsarjaa ja kaksi lyhyempää toimintoa prototyypin avulla ja lopuksi tutustumaan yleisesti ohjelmaan. Valitut tehtäväsarjat olivat ohjelmalla useimmiten käytettäviä toimintoja, jotta niiden sujuvuudesta ja opittavuudesta saataisiin kerättyä mahdollisimman paljon tietoa. Testattavien toimintojen avulla selvitettiin kuinka nopeasti käyttäjä oppii käyttämään täysin uutta ohjelmaa ja kuinka nopeasti ja vaivattomasti hän siitä suoriutuu. Testien avulla kerättiin myös tietoa ohjelman rakenteesta ja loogisuudesta, lisäksi haluttiin selvittää osaako koehenkilö hyödyntää kaikkia ohjelman ominaisuuksia ja mitä lisäominaisuuksia hän ohjelmaan toivoisi. Käyttäjätestin pääpaino oli enemmän ohjelman käyttöliittymän rakenteen ja toiminnallisuuden selvittämisessä, kuin ohjelman visuaalisessa ilmeessä.



Kuva 29A. Käyttäjätestauksessa käytetyt prototyyppinäkymät CSS-editorista

Kuva 29B. Käyttäjätestauksessa käytetyt prototyyppinäkymät CSS-editorista

Käyttäjätestissä selvisi että ohjelman rakenne oli mietitty tarkkaan ja hyvin, sillä jokainen käyttäjä osasi käyttää ohjelmaa ja piti sitä selkeänä ja loogisena, siitäkin huolimatta ettei osa testaajista puutteellisen CSS-tietämyksen takia ymmärtänyt mitä ohjelma oikeasti tekee. Ohjelman selkeän rakenteen ja ulkoasun ansiosta koehenkilöt pystyivät suorittamaan annetut tehtävät, vaikkeivät kaikki aina tienneet mitä jokaisen toiminnon lopullinen vaikutus oli. Koehenkilöiltä saatiin myös arvokkaita huomioita ja ideoita CSS-editorin ominaisuuksien lisäämiseen ja muokkaamiseen ja ulkoasun parantamiseksi. Paremmin verkkosivujen tekemistä tuntevilta testaajilta saatiin enemmän tarkempaa tietoa tyylimäärittelyjen toimivuudesta ja muut käyttäjät keskittyivät enemmän ohjelman yleiseen toimivuuteen ja rakenteeseen.

Palaute:

- valitsimien järjesteleminen ja ryhmitteleminen
- mahdollisuus vaihtaa näkymää tekstieditori puolelle
- hakukenttä
- valikoissa totuttujen termien käyttäminen
- mahdollisuus palata alku valikkoon
- symbolien toiminnan selittäminen
- arvojen toiminnan selittäminen
- mahdollisuus tarkastella määritysten vaikutusta
- ominaisuuksien vaikutusten selittäminen.

6.5 CSS-editorin ohjelmoiminen

Varsinaisen editorin ohjelmointi tapahtui käyttäen Microsoftin Visual Studio ohjelmaa, jolla tehtiin Windows Forms sovellus ja koodikielenä toimi C#. Ensin tutustuttiin koodikielen yleiseen toimintaan, rakenteeseen, tietotyyppeihin ja luokkiin.

CSS-editoriin halutut ominaisuudet pilkottiin mahdollisimman pieniin ja yksinkertaisiin toimintoihin, jolloin oli kaikkein helpoin selvittää kuinka se ohjelmoi-

taisiin. Jokaisesta toiminnasta tehtiin oma metodinsa, eli toiminnan suorittamiseksi tarvittavat lauseet koottiin yhdeksi kokonaisuudeksi, jotta niistä voidaan myöhemmin muodostaa editoriin haluttuja lopullisia ominaisuuksia.

Tärkein ja haastavin ohjelmoitava ominaisuus oli CSS-parseri, jonka tehtävä on lukea CSS-tiedosto ja tarkastaa sen oikeakielisyys. Parseri myös tallentaa luetut ominaisuudet ja arvot muuttujiin. Haasteena oli tunnistaa erityyliset ja rakenteiset CSS-tiedostot, jotka eivät ole CSS-editorilla tehtyjä. Lisäksi tuli ratkaista, kuinka parseri reagoisi virheellisiin määrittelyihin tiedostoissa ja kuinka ne esitettäisiin ohjelmassa. Tiedostojen lukemisen lisäksi piti myös päättää millä tyylillä editori kirjoittaisi määrittelyt takaisin CSS-tiedostoon.

Tiedostojen lukemisen ja tulkitsemisen ohella toinen suuritöinen ohjelmoitava osuus oli saatujen määrittelyjen esitteleminen käyttäjälle. Kaikki saadut tiedot tuli näyttää ohjelman käyttäjälle selkeässä ja muokattavassa muodossa. Jokaisesta tyylimäärittely piti luoda ohjelman käydessä näkyvä komponentti ja ne tuli nimetä ja hallita niin että niitä pystyi muokkaamaan, poistamaan ja tallentamaan editoria käytettäessä.

6.6 Casen arviointi

Minulla oli alusta alkaen melko selkeä kuva, siitä minkälaista CSS-editoria lähdin suunnittelemaan ja toteuttamaan. Luettuani eri kirjoista käytettävyydestä ja ohjelmistosuunnittelusta tunsin että minulla oli riittävästi tietoa ohjelman suunnitteluun. Saatuani listattua ohjelmaan halutut ominaisuudet tein ohjelman rakenteesta alustavan kaavion, jonka pohjalta sain luotua lopullisen vuokaavion melko nopeasti. Ohjelman ulkoasu muodostui myös helposti vuokaavion pohjalta. Käyttäjätestit paljastivat että rakenteen ja ulkoasun suunnittelut onnistuivat hyvin, sillä niitä ei tarvinnut paljoa muutella. Tämä vaihe työstä sujui melko vaivattomasti ja oli mukavaa ja olin tyytyväinen lopputulokseen.

Tiesin että editorin ohjelmoiminen voisi olla melko haastavaa, sillä en aikaisemmin ollut tehnyt ihan vastaavanlaista ohjelmaa ja tässä mittakaavassa. Viimeaikaisesta C#:in käytöstä oli myös kulunut jonkin aikaa, joten jouduin aluksi taas hieman tutustumaan kielen saloihin. Aluksi jouduin tuskastumaan useampaan kertaan

ohjelmoidessani ohjelman ominaisuuksia, mutta pikkuhiljaa kieli alkoi tuntua kokoajan helpommalta. Microsoftin kirjastoista löytyy kattavat esimerkit melkein kaikista tilanteista, joten sain usein apua pulmakohtiin. Päästyäni tutuksi C#:in kanssa tuli minulle hieman pitempi tauko editorin ohjelmoinnista ja palattuani ohjelman pariin tuntui se taas aluksi hieman vieraalta. Sain lopulta lisättyä ohjelmaan kaikki haluamani ominaisuudet, vaikka ulkoasuun en ehkä pystynyt panostamaan niin paljoa kuin olisin halunnut. Olen kuitenkin tyytyväinen ohjelman lopputulokseen, sillä haaste oli melko suuri, mutta suoriuduin siitä kuitenkin kunnialla.

CSS-editorin tekeminen sujui kokonaisuudessa melko hyvin, vaikka ohjelmointi osuus olikin hieman hidasta. Huomasin myös suunnitteluvaiheessa tekeväni välillä turhan tarkkaa ja hienoa työtä, kun olisi vain pitänyt tehdä yksinkertaisia ja nopeita protyyppejä testausta varten. Pidin tätä projektia kuitenkin kokoajan mielenkiintoisena ja haastavana ja opin sitä tehdessä paljon, sillä jouduin tekemään jokaisen työvaiheen itse alusta loppuun ja pääsin tekemään monia erilaisia osa-alueita suunnittelusta, käyttäjätestaukseen ja ohjelmointiin. Tarkoituksenani on vielä tulevaisuudessa parannella ja jatkaa editorin kehittelyä monipuolisemmaksi sovellukseksi.

7 YHTEENVETO

Opinnäytetyön tarkoituksena oli oman CSS-editorin ohjelmoiminen. Samalla tuli selvittää, kuinka laaja-alaista tietotaitoa sen menestyksekäs toteuttaminen vaatii. Tarkempaan selvitykseen otettiin CSS 3 ja sen mukanaan tuomat uudet ominaisuudet sekä niiden vaikutus verkkosivujen ulkoasuun ja toteutukseen.

CSS 3 on selvästikin suuri ja mielenkiintoinen uudistus verkkosivujen tekijöille. Se mahdollistaa monien asioiden toteutuksen suoraan tyylimäärittelyjen avulla, ilman erillisen ohjelmointikielen opiskelua ja käyttöä. CSS 3 tuo verkkosivuille kaivattua eloa ja toiminnallisuutta, sivut eivät näytä enää vain Internetiin skannatuilta kuvilta joissa on muutama nappula, vaan ne ovat oikeasti dynaamisia. Sivujen suunnittelijoille tämä tuo taas paljon uusia mahdollisuuksia toteuttaa ideoitaan ja muutamia oikein mielenkiintoisia ja raikkaita tapauksia onkin jo nähty.

Haasteena CSS 3:lla on kuitenkin taas sama ongelma kuin aikaisemmillakin versioilla. Sen kehitys on ollut hyvin sirpaleista ja uusia ominaisuuksia tulee käyttöön pikkuhiljaa, vaikkeivät ne vielä edes olisi virallisia. Selainten määrä on myös lisääntynyt eikä millään selaimella ole määräävää valta-asemaa, vaan kaikki taistelevat tasaisesti käyttäjistä. Selaimet ottavatkin uusia tyyliominaisuuksia mahdollisimman nopeasti käyttöön ja pyrkivät näin tekemään omasta selaimestaan mahdollisimman kehityksellisen ja houkuttelevan. Tämä tekee verkkosivujen tekijöiden työstä entistä haasteellisemman, sillä on vaikea tietää mitä ominaisuuksia voi jo käyttää. Eri selaimilla on lisäksi erilaiset prefix-koodit, joita ne käyttävät uusimpien ominaisuuksien tukemiseen. Kun uudet tyylimäärittelyt lopulta sitten saavuttavat suositus-statuksen tulevat ne helpottamaan ja monipuolistamaan verkkosivujen tekemistä.

Case-projektissa toteutettiin oma CSS-editori ja siinä päästiinkin hyödyntämään laajasti kaikkea aikaisemmin opittua teoriaa käyttöliittymäsuunnittelusta, tyylimäärittelyistä ja valmiiden editoreiden toiminnasta. Työtä tehdessä huomasi, kuinka kattavaa tietämystä toimivan ohjelman tekemiseen vaaditaan. Kirjoissa näytetyt uudet lähestymistavat, ideat ja tekniikat toivat paljon uutta näkemystä editorin suunnitteluun. Käyttäjätestauksessa paljastui myös monia asioita käyttömukavuuden parantamiseksi ja olikin mielenkiintoinen huomata, minkälaisiin

asioihin erilaisella taustatietämyksellä ja käyttökokemuksella varustautuneet testaajat kiinnittivät huomiota. CSS-editorin lopputulos on toimiva, mutta parannettavaa vielä jäi niin visuaalisuuden kuin toimintojenkin puolesta. Olisi mielenkiintoista testata kuinka hyvälaatuisen, monipuolisen ja selkeän ohjelman saisi kehitettyä uhraamalla sille vielä aikaa ja tekemällä käyttäjätestauksia.

Tarjolla on jo laaja valikoima erilaisia ilmaisia ja maksullisia CSS-editoreita, erilaisiin käyttötarpeisiin ja erilaisille käyttäjille. Osa editoreista on todella monipuolisia ja helppokäyttöisiä ja niiden käytöstä hyötyvät myös tekstieditorien nimeen vannovat kovan luokan ammattilaiset. Graafiset editorit selkeyttävät ja helpottavat tyylimäärittelyjen tekemistä ja voivat parhaassa tapauksessa opettaa asiaa tuntemattomalle yhteyden koodin muutoksen ja elementin ulkoasun muuttumisen välillä. Editorit mahdollistavat myös sen, että täysin CSS-tyylimäärittelyjä tuntematon käyttäjä pystyy nyt muokkaamaan verkkosivujen ulkonäköä, joka näin ollen vähentää verkkosivujen muokkaamiseen perehtyneen ammattilaisen arvoa. Samanlaista ilmiötä on tapahtunut viimeaikoina myös kuvankäsittelypuolella, kun tavalliset kuvien ottajat ovat alkaneet voida muokata kuviaan erilaisilla pikaohjelmilla, kuten instragam, joita löytyy suoraan kameroista ja puhelimista, ilman että heidän tarvitsisi opiskella raskaamman kuvankäsittelyohjelman käyttöä. Tällaiset ohjelmat helpottavat asioihin vähemmin perehtyneitä saamaan jonkinlaista jälkeä aikaiseksi. Ne tuskin kuitenkaan tulevat syrjäyttämään ammattilaisia, silloin kun halutaan saada huippuluokkaista ja yksilöllistä jälkeä aikaiseksi, sillä näillä ohjelmilla tehty jälki on aina melko samannäköistä.

LÄHTEET

Painetut lähteet

- Korpela, J. 2008. CSS verkkosivujen muotoilussa. Porvoo: WS Bookwell.
- Teague, J. 2011. CSS3 visual QuickStart Guide. USA
- Wiio, A. 2004. Käyttäjäystävällisen sovelluksen suunnittelu. Helsinki: Edita Prima Oy.

Elektroniset lähteet

- Cederholm, D. 2010. CSS3 FOR WEB DESIGNERS. New York. A Book Apart. Sähkökirja
- CSS3 PIE. 2012. About PIE [viitattu 10.4.2012]. Saatavissa: http://css3pie.com/about/
- Gasston, P. 2011. THE BOOK OF CSS3 a developers guide to the future of web design. Canada. Sähkökirja
- Gillenwater, Z. 2011. Stunning CSS3: A project-based guide to the latest in CSS. USA. Ner Riders. Sähkökirja
- Notepad++ 2012. [viitattu 20.11.2012]. Saatavissa: http://notepad-plus-plus.org/
- Riihiaho S. 2004. Käytettävyyden arviointi ilman käyttäjiä [viitattu 10.1.2013]. Saatavissa: http://www.soberit.hut.fi/T-121/T-121.600/asiantuntija-arviot.pdf
- W3C Recommendation 2011. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification [viitattu 19.3.2012]. Saatavissa: http://www.w3.org/TR/CSS2/
- w3school. 2012. CSS3 Browser Support Reference [viitattu 10.4.2012]. Saatavissa: http://www.w3schools.com/cssref/css3_browsersupport.asp

Kuvalähteet

- Kuva 17. Splashnology. 2012. Making an Image Gallery with CSS3 [viitattu: 22.3.2012]. Saatavissa:

http://www.splashnology.com/article/making-an-image-gallery-with-css3/3356/

- Kuva 18. Carsonified. 2012. Valid Non-Javascript Lightbox [viitattu: 22.3.2012]. Saatavissa: http://carsonified.com/blog/design/css/how-to-create-a-valid-non-javascript-lightbox/
- Kuva 19. W3C. 2012. CSS SPECIFICATIONS [viitattu: 11.4.2012]. Saatavissa: http://www.w3.org/Style/CSS/current-work
- Kuva 20. Gillenwater, Z. 2011. Stunning CSS3: A project-based guide to the latest in CSS. USA. Ner Riders. Sähkökirja, s. 25
- Kuva 21. Johansson E. 2013. Kuvakaappaus Notepadd++-editorista
- Kuva 22. Johansson E. 2013. Kuvakaappaus Topstyle 5.0-editorista
- Kuva 23. Johansson E. 2013. Kuvakaappaus Stylizer 5-editorista
- Kuva 24. Johansson E. 2013. Kuvakaappaus CSS-Shack-editorista
- Kuva 25. Johansson E. 2013. Kuvakaappaus CSS 3.0 Maker-editorista
- Kuva 26. Johansson E. 2013. Kuvakaappaus bluePen-editorista