JavaScript V8
ja Google Chrome

TURUN YLIOPISTO
Informaatioteknologian laitos
Tietojenkäsittelytiede
LuK-tutkielma
Joulukuu 2008

Jyri Lehtonen (72039)
jyri.lehtonen@utu.fi

TURUN YLIOPISTO
Informaatioteknologian laitos / Matemaattis-luonnontieteellinen tiedekunta

LEHTONEN, JYRI KRISTER OLAVI: JavaScript V8 ja Google Chrome

LuK-tutkielma, 32 s., 3 liites.
Tietojenkäsittelytiede
Joulukuu 2008

Edellisellä vuosikymmenellä WWW-kehittys keskittyi pääasiassa dokumenttien onnistuneeseen näyttämiseen asiakkaille. Internet-sovellukset ovat muuttunet tällä vuosikymmenellä. Enää ei ole tarkoituksena vain avata onnistuneesti käyttäjälle yksinkertaisia dokumentteja, vaan internet-sivustot sisältävät kokonaisia sovelluksia.

Tutkielma esittelee JavaScriptin, joka on yleisin ohjelmointikieli asiakaspuolen WWW-ohjelmoinnissa. Tutkielmassa käsitellään myös JavaScriptin turvallisuutta, keskittyen yhteen turvallisuustoimintaperiaatteeseen. Lopuksi aiheesta tarkastetaan muutamia vaaroja, jotka liittyvät JavaScriptin käyttöön internet-selaimessa.

Tutkielman pääaiheet ovat ajankohtaisia. Molemmat pääaiheet tulivat julkisuuteen syyskuussa 2008. Tästä johtuen tutkielma on mahdollisesti ensimmäinen aiheeseen perehtyvä suomenkielinen teos. Tutkielman ensimmäinen pääaihe käsittelee virtuaalikoneita. Virtuaalikone on sovellus, jossa on mahdollista ajaa ohjelmia kuten aidossa tietokoneessa. Virtuaalikoneita on kahdenlaisia: järjestelmävirtuaalikoneita sekä prosessivirtuaalikoneita. Tässä tutkielmassa keskitytään prosessivirtuaalikoneisiin ja tarkemmin JavaScript-virtuaalikoneisiin. JavaScript-virtuaalikoneet esitetään yleisesti ja tarkemmin määritellään uusin JavaScript-virtuaalikone. Tutkielma keskittyy virtuaalikoneiden osalta niihin syihin, miksi JavaScript V8 -virtuaalikone on tehokkain virtuaalikone JavaScriptille.

Ensimmäinen sovellus, joka käyttää JavaScript V8 -virtuaalikonetta, on Google Chrome internet-selain. Tämän tähden tutkielmassa käsitellään Google Chromea kokonaisvaltaisesti. Tutkielma lähestyy Chromea käyttäjän näkökulmasta sekä turvallisuudesta. Tutkielma käsittelee myös Chromen muistinkäytön eron muihin internet-selaimiin verrattuna sekä määrittelee sen, miksi Chrome on niin tehokas internet-selain JavaScriptin ajamisessa. Lopuksi tutkielmassa käsitellään Chromen ongelmia, vertaillaan yleisimmät internet-selaimet erilaisilla testeillä ja pohditaan sitä, mikä tulee olemaan Google Chromen kohtalo internet-selaimena.

Avainsanat: JavaScript, WWW-turvallisuus, virtuaalikone, V8, Google Chrome, WebKit, internet-selain

Sisällys

1 JOHDANTO 1

2 JAVASCRIPT 3

2.1 JavaScriptin turvallisuus 4

2.2 JavaScript-virtuaalikoneet 8

2.3 V8-virtuaalikoneen esittely 9

2.4 V8:n tehokkuuden perusta 10

3 GOOGLE CHROME 15

3.1 Käyttöliittymä 16

3.2 Muistin käyttö 17

3.3 Turvallisuuden edistäminen 18

3.4 Turvallisuusongelmat ja pettymykset 21

3.5 Tehokkuuden tausta 22

3.6 Vertailu ja testit 23

4 YHTEENVETO 27

Lähteet 31

LIITE 1: Käyttöliittymä 33

LIITE 2: Välilehden kaatuminen 34

LIITE 3: Sivun estäminen 35

1 JOHDANTO

JavaScript on pääasiassa asiakaspuolen ohjelmointikieli, jota käytetään dynamiikan toteuttamisessa internet-sivuille. JavaScript siis laajentaa internet-sivustojen toimintaa. Suurin osa internet-sivustoista sisältää pieniä ohjelmia, joilla esitetään esimerkiksi grafiikkaa tai lasketaan tiettyjä haluttuja tuloksia. Käyttäjät voivat usein tietämättään vastaanottaa internet-sivustolta JavaScript-lähdekoodia päivittäisessä toiminnassa internet-selaimillaan. Tämä johtuu siitä, että internet-selain on JavaScriptin yleisin toimintaympäristö. Tätä ohjelmointikieltä ei tarvitse kääntää (engl. compile), vaan internet-selain tulkkaa (engl. interpret) sen HTML:n (Hyper Text Transfer Protocol) joukkoon usein pyytämättä käyttäjältä mitään toimintaa. Tästä johtuen tässä tutkielmassa käsitellään myös JavaScriptin turvallisuutta käyttäjän kannalta. Parhaimmillaan JavaScript voi kuitenkin olla kokonainen internet-sovellus, jota käyttäjä käyttää internet-selaimien, kuten Internet Explorer tai Mozilla Firefox, kanssa.

Virtuaalikoneet ovat tässä tutkielmassa JavaScriptin toimintaympäristöjä. Ne määrittävät sen, miten JavaScript toimii sekä sen, kuinka tehokkaasti se toimii. Suurin merkitys JavaScriptin toiminnan tehokkuudessa ilmenee laajojen JavaScript internet-sovellusten parissa. Uusin JavaScript-virtuaalikone on nimeltään V8, ja ensimmäinen sitä käyttävä sovellus on Google Chrome -internet-selain. Monet Googlen tuotteista perustuvat JavaScriptiin, kuten Google Mail sekä Google Documents. Nämä internet-sovellukset ovat laajoja, ja tehokkaamman JavaScript toimintaympäristön avulla Google toivoo asiakkaidensa toiminnan olevan nopeampaa. JavaScriptin tehokkuus kasvaa huomattavasti käyttämällä Google Chromea minkä tahansa JavaScriptin ajamiseen. Tästä johtuen Google Chrome ei ole pelkästään Googlen omien sovellusten käyttämiseen tarkoitettu.

Internetin käyttäjät yleisesti web 2.0 -ajan mukaan eivät käytä enää pelkkiä internet-sivuja, vaan sovelluksia internetissä. Termi web 2.0 -aika tarkoittaa sitä, että internetin mahdollisuudet ovat laajentuneet. Käyttäjät katsovat esimerkiksi videoita, keskustelevat keskenään, pelaavat internet-selaimella moninpelejä tai käyttävät yhteisöllisiä sovelluksia. Uuden JavaScript-virtuaalikoneen ja Google Chromen olisi tarkoitus

nopeuttaa kaikkea tätä toimintaa. Koska kyseessä on juuri JavaScriptin toimintaympäristö, tutkielmassa käsitellään myös Google Chromen turvallisuutta käyttäjän kannalta.

Tämä tutkielma esittelee lyhyesti JavaScriptin, sekä käsittelee yhden sen turvallisuustoimintaperiaatteista. Tutkielma keskittyy tämän jälkeen JavaScriptin uusimpaan virtuaalikoneeseen, sekä sen tehokkuuden taustaan. Kun JavaScript sekä sen uusin virtuaalikone on esitelty, tutkielma määrittelee Google Chromen. Chrome on ensimmäisen sovellus, jossa on JavaScript V8 -virtuaalikone. Chrome käsitellään kokonaisvaltaisesti. Tutkielma keskittyy sovelluksen käyttöliittymään ja siihen, miten se eroaa muista internet-selaimista ja mitä hyötyä muutoksista on. Toiseksi käsitellään internet-selaimen muistinkäyttö. Kolmanneksi keskitytään sovelluksen turvallisuuteen ja ongelmiin. Neljänneksi esitellään lyhyesti sovelluksen tehokkuuden taustalla oleva WebKit-projekti. Viidenneksi tutkielmaan on tehty muutamia JavaScript-testejä erilaisilla internet-selaimilla. Testien avulla on mahdollista nähdä kuinka tehokas JavaScriptin käytössä uusin internet-selain on verrattuna muihin internet-selaimiin. Viimeiseksi tutkielmassa pohditaan Google Chromen kohtaloa internet-selaimena ja kerrataan tutkielman pääasiat yhteenvedossa.

Tutkielma käsittelee ajankohtaista asiaa, sillä Google Chrome ilmestyi julkisuuteen 2. syyskuuta 2008. Tutkielman kirjoittaminen alkoi samalla viikolla. Kummastakaan tutkielman pääaiheesta, JavaScript V8 -virtuaalikoneesta sekä Google Chromesta, ei ole olemassa vielä kattavia kirjallisia teoksia. Tutkielma on kirjoitettu laajasti perustuen internetissä oleviin tieteellisiin artikkeleihin, kehittäjien julkaisuihin, käyttäjien mielipiteisiin, ajankohtaisiin alan lehtiartikkeleihin sekä tutkielman kirjoittajan omiin kokemuksiin sovelluksen käytöstä. Lähteet ovat suurin osa englanniksi. Tämä tutkielma on ensimmäinen suomenkielinen teos aiheesta.

2 JAVASCRIPT

JavaScript on Netscapen kehittämä dynaaminen olio-ohjelmointi kieli. Kieltä käytetään miljoonissa internet-sivustoissa sekä palvelinsovelluksissa ympäri maailmaa. Netscapen JavaScript on periytynyt ECMA-262 Versio 3:sta (Peltomäki 2001, 8).

ECMAScript on suunniteltu sovellusriippumattomaksi komentosarjakieleksi, jolloin se sopii minkä tahansa sovelluksen ohjelmointiin. Sovellusriippumattomuus on saavutettu jakamalla komentosarjakieli kahteen osa-alueeseen: runkokieleen (engl. core language) ja sovellusriippuvaiseen oliomalliin. ECMAScript standardi keskittyy pelkästään kielen rungon määrittämiseen. Sovelluskohtaiset asiat jätetään sovelluskohtaiselle oliomallille. (Peltomäki 2001, 8.) Esimerkiksi DOM (Document Object Model) sekä ECMAScript muodostavat yhdessä nykyaikaisemman version JavaScriptistä. DOM on ohjelmointirajapinta, joka mahdollistaa komentosarjojen, kuten tutkielmassa käsiteltävä JavaScript, muuttaa dynaamisesti (X)HTML-dokumenetteja asiakaspuolella. JavaScriptin kanssa DOM:lla voidaan toteuttaa vuorovaikutteisia internet-sivuja, jotka eivät vaadi jatkuvaa palvelinyhteyttä. (W3C 2008.)

Komentosarjakieliä käytetään erityisesti erilaisten sovellusten toimintojen laajentamiseen. Komentosarjojen ei kuitenkaan tarvitse olla lyhyitä. JavaScriptillä on mahdollista kehittää joko yhden rivin vaativa toiminto internet-sivustolle, tai täydellinen sovellus. Kuitenkin kaikissa tapauksissa JavaScript-ohjelma on aina suoritettava internet-selaimessa tai muussa sovelluksessa, joka sisältää JavaScript-tulkin. JavaScript on siis tulkattava kieli, kuten kaikki muutkin komentosarjakielet. Komentosarjakielillä on alkujaan automatisoitu tehtäviä ilman, että tarvitaan varsinaisia ohjelmointikieliä. Kuitenkin komentosarjakieli on yleistetty tarkoittamaan kaikkia kieliä, joilla on helppo tehdä toimintoja ilman ohjelman kääntämistä. Komentosarjakielet voidaan jakaa ensimmäisen ja toisen sukupolven kieliin. Vanhemmat ensimmäisen sukupolven kielet ovat esimerkiksi Unix- tai Linux-tietokoneiden komentokielet sh ja bash tai MS-DOS ajan BAT-tiedostot. Uusimpien toisen sukupolven kielien ei välttämättä tarvitse suorittaa ainuttakaan käyttöjärjestelmäkomentoa. Uusiin komentosarjakieliin luetaan esimerkiksi seuraavia: JavaScript, PHP, Python ja Ruby.

JavaScript-tulkki on mahdollista upottaa mihin tahansa isäntäsovellukseen, kuten internet-selaimeen tai internet-palvelimeen. Tulkattavuuden etuna on se, että JavaScript-sovellusta on mahdollista muokata yhtä vaivattomasti kuin tavallista HTML-dokumenttia. Muutokset sovellukseen tulevat voimaan heti, kun dokumentti ladataan uudestaan internet-selaimeen. Tulkattavuuden haittana voidaan pitää sitä, että tulkattava ohjelma on hitaampi kuin jos se olisi käännetty. (Peltomäki 2001, 8-11.)

Syntaksiltaan JavaScript on samankaltaista kuin Java tai C++. Syntaksin taustalla on se, että ohjelmoijat kykenevät siirtymään vaivattomasti myös JavaScriptin pariin. Ohjelmointikielillä on muuten suhteellisen vähän yhteistä. (Mozilla Developer 2008.)

2.1 JavaScriptin turvallisuus

JavaScriptillä on mahdollista toteuttaa haitallista toimintaa käyttäjän tietokoneella. Tämän mahdollistaa yleisimmin tietoturva-aukko komentosarjakielessä. Käyttäjän toimintaympäristöön on myös mahdollista vaikuttaa ilman tietoturvan rikkomista. Kyseisiin ongelmiin palataan luvun lopussa.

Perinteiset haittaohjelmat asentuvat käyttäjien tietokoneelle yleensä heidän omasta toimesta. Käyttäjän on hyväksyttävä internet-sivulla tarjolla olevan ohjelman asentumisen tietokoneeseensa. Toinen tapa on se, että käyttäjän turvallisuusmalli on heikolla tasolla, jolloin haittaohjelman ei tarvitse kysyä käyttäjältä lupaa asentaakseen itsensä. JavaScriptin käytössä on otettava huomioon se, että toiminta vastaa tietyllä tasolla edellä mainittuja esimerkkejä. Käyttäjän internet-selain vastaanottaa lähdekoodia, ja yleensä suorittaa sen automaattisesti. Lähdekoodin on kirjoittanut tietty kolmasosapuoli, kenestä käyttäjä ei tiedä välttämättä mitään.

Internet-selaimet käyttävät turvallisuustoimintaperiaatteita suojatakseen käyttäjää. Turvallisuustoimintaperiaate on joukko sääntöjä, jotka määräävät mitä komentosarjat voivat tehdä ja missä olosuhteissa (Powell 2001). Esimerkiksi internet-selaimet estävät JavaScriptiä saamasta käsiinsä mitään tiedostoja käyttäjän tietokoneelta. Jos tämä turvallisuustoimintaperiaate puuttuisi, jokin internet-sivusto pystyisi tuhoamaan kaikki käyttäjän tietokoneen tiedostot.

Nykyaikaiset JavaScript-turvallisuustoimintaperiaatteet ovat lähtöisin Javasta. Internetistä ladatut komentosarjat ajetaan hiekkalaatikko (engl. Sandbox) ympäristössä, joka on eristetty muusta käyttäjän järjestelmästä. Komentosarjoille annetaan oikeudet käynnissä olevaan dokumenttiin tai dokumentteihin, jotka ovat samalla internet-sivustolla. Oikeutta ei anneta paikallisen käyttäjän järjestelmän tiedostoihin, käyttäjän käynnissä olevien ohjelmien muistialueeseen, tai käyttöjärjestelmän verkkotyöskentelytasoon. (Powell 2001.)

Ensisijainen JavaScriptin turvallisuustoimintaperiaate on sama-alkuperä (engl. same-origin). Muut turvallisuustoimintaperiaatteet ovat rajattu tutkielman ulkopuolelle. Kyseinen periaate estää komentosarjoja vaikuttamasta eri internet-sivustojen välillä (Powell 2001). Tämä tarkoittaa sitä, että periaate estää vihamielistä koodia muuttamasta dokumentin tietoja toisella sivustolla. Ilman tätä turvallisuustoimintaperiaatetta JavaScript pystyisi vaikka tarkastamaan sen, mitä käyttäjä kirjoittaa sisään kirjautuessaan toiselle sivustolle. Esimerkiksi JavaScript-lähdekoodi voisi odottaa käyttäjän kirjautumista hänen verkkopankkiinsa ja välittömästi siirtää käyttäjän tililtä rahaa toiselle tilille. Kun JavaScript yrittää päästä käsiksi ominaisuuksiin tai metodeihin toisessa ikkunassa, esimerkiksi window.open()-palautuksella, internet-selain toteuttaa same-origin -testauksen avattavien dokumenttien osoitteille (Powell 2001). Alla olevassa taulukossa 1 ovat mahdolliset tapahtumat same-origin -testauksessa.

Esimerkki (www.testi.fi)	**Same-origin -testi**	**Syy**
http://www.testi.fi/ index.html	Hyväksytty	Sama kohdealue ja protokolla
http://www.testi.fi/ kansio1/kansio2/index.html	Hyväksytty	Sama kohdealue ja protokolla
http://www.testi.fi:8080/ kansio/sivu.html	Estetty	Eri portti
http://www2.testi.fi/ kansio/sivu.html	Estetty	Eri palvelin
http://testaus.fi	Estetty	Eri kohdealue
ftp://www.testi.fi	Estetty	Eri protokolla

Taulukko 1: Same-origin -testauksen vaihtoehdot

Yleinen JavaScript-turvallisuusongelma on kuitenkin yllämainittu sivustolta A sivustoon B vaikuttaminen. Tätä toimintaa kutsutaan Cross-Site Scripting eli XSS. XSS hyökkäyksiä on erilaisia, mutta kaikista yleisimmät ovat JavaScriptin DOM-malliin liittyviä. Nämä hyökkäykset häpäisevät same-origin -periaatetta. Tämä voi tapahtua siten, että hyökkääjä onnistuu saamaan luotettavan internet-sivuston lähettämään vihamielistä lähdekoodia internet-dokumenttiin, joka näytetään käyttäjälle. Toinen tapa on se, että internet-selaimien tekijät ovat toteuttaneet same-origin -periaatteen virheellisesti. XSS hyökkäys on onnistunut silloin, jos sen tekijä saa käsiinsä käyttäjän kirjautumis-, eväste-, sivuhistoriatiedot, tai IP-osoitteen. (Monthie, 2008.) Lukuisat internet-sivustot eivät ole suojautuneet XSS:ää vastaan. Tämä johtuu joko siitä, että internet-sivuston ylläpitäjät eivät kiinnitä huomiota XSS:n vaaroihin, tai he eivät tunne XSS:n mahdollista vaaraa. XSS on todellinen vaara, koska web 2.0 -ajan käyttäjät vierailevat esimerkiksi massiivisilla yhteisöllisillä internet-sivustoilla kuten YouTube, MySpace, tai eBay. Kyseisissä internet-sivustoissa on laaja käyttäjäkanta ja käyttäjillä on mahdollisuus lisätä sisältöä omiin sivuihinsa.

Turvallisuusongelmat eivät ole pelkästään turvallisuusperiaatteiden puutetta tai virheellisyyttä. JavaScriptillä on mahdollista toteuttaa lähdekoodia, joka vaikuttaa käyttäjän toimintaympäristöön, vaikka mitään turvallisuustoimintamallia ei ole edes yritetty ohittaa. Tutkielmassa käsitellään seuraavaksi kolme esimerkkityyppiä, jolla on mahdollista, tai on ollut mahdollista vaikuttaa käyttäjän järjestelmään JavaScriptin avulla.

Loputon silmukka

Yksinkertaisella tavalla on mahdollista saada JavaScript loputtomaan silmukkaan. Uusimmat internet-selaimet pystyvät jonkin verran estämään tätä toimintaa. Mitä taitavammin haitallinen JavaScript-koodi on toteutettu, sitä pienemmällä todennäköisyydellä internet-selain havaitsee hyökkäyksen. Alla on esimerkki loputtomasta silmukasta.

```
function pyöri() {
ympäri();
}
pöyri();
```

```
function ympäri() {
pyöri();
}
```

Muistin täyttö rekursiolla

Toinen esimerkki on järjestelmän rekursiivisten käskyjen seurantaan liittyvän muistin täyttäminen. Seuraavan funktion toteuttaminen ajaa yllämainitun muistinosan täyteen.

```
function rekursiivinen(){
   var x=1;
   rekursiivinen();
}
```

Toinen esimerkki samasta aihepiiristä liittyy String-tyyppisten alkioiden rekursiiviseen toistamiseen. Alla oleva esimerkki ei ole funktio, vaan yksinkertainen komentosarja.

```
var kaatuu = "Internetselaimen kaatuminen";
   while (true) {
      kaatuu += kaatuu;
}
```

Huomio-ikkunat

Viimeinen esimerkki on varsinkin vanhempien internet-selaimien ongelma. Vanhemmiksi internet-selaimiksi voidaan luokitella edelliseltä vuosikymmeneltä internet-selaimet kuten IE versiot 1 - 5 tai nykyiseltä vuosikymmeneltä internet-selaimet kuten Firefox versiot 0.x – 1.x. Uusimmat internet-selaimet pystyvät estämään kyseisen toiminnan varoittamalla käyttäjää suojausvaroituksella tai huomauttaa käyttäjää pitkään jatkuneesta komentosarjasta, mutta vanhat internet-selaimet kaatuvat tähän varmasti. Esimerkki avaa loputtomasti käyttäjän internet-selaimeen huomio-ikkunan viestillä "Apua". Ikkunoita aukeaa joko nopeammin kuin käyttäjä ehtii niitä sulkemaan, tai sama varoitus ikkuna loputtomiin. Käyttäjä menettää joko internet-selaimensa tai käyttöjärjestelmänsä hallinnan. Alla on esimerkki huomio-ikkunoiden väärinkäytöstä.

```
function kysyminua() {
   alert("Apua!");
   kysyminua();
}
```

2.2 JavaScript-virtuaalikoneet

Virtuaalikone on ohjelmallisesti toteutettu tietokone, jossa on mahdollista ajaa ohjelmia kuten aidossa tietokoneessa. Virtuaalikoneet jaetaan kahteen ryhmään. Jako perustuu siihen, mihin virtuaalikonetta käytetään. Järjestelmävirtuaalikone mahdollistaa kokonaisen käyttöjärjestelmän ajamisen. Prosessivirtuaalikone mahdollistaa yhden ohjelman eli yhden prosessin ajamisen. Ajo tapahtuu käyttäjän käyttöjärjestelmän sisällä. Prosessivirtuaalikone luodaan silloin, kun tietty prosessi käynnistetään. Se poistetaan silloin, kun tietty prosessi päättyy. Tarkoituksena on mahdollistaa käyttöjärjestelmästä riippumaton ohjelmointiympäristö. (Smith & Nair 2005.) Tässä tutkielmassa keskitytään prosessivirtuaalikoneeseen ja tarkemmin määritellen JavaScript V8 -virtuaalikoneeseen.

Virtuaalikoneen käytöstä on erilaisia hyötyjä. Niistä tärkeimmät ovat lueteltu seuraavassa:

- **Yhdistäminen:** Monta käyttöjärjestelmää voidaan ajaa samalla palvelimella.
- **Vakaus ja turvallisuus:** Virtuaalikoneet ovat eristetty muusta toimintaympäristöstä siten, että yhden turvallisuusongelma ei vaikuta muihin.
- **Kehitys:** Kehittäjät pystyvät testauttamaan sovelluksensa erilaisissa käyttöjärjestelmissä samalla tietokoneella. Yhden sovelluksen kaatuminen ei kaada koko järjestelmää.
- **Siirtäminen:** Virtuaalikoneen pystyy helposti siirtämään palvelimelta toiselle tasatakseen palvelimiin kohdistuvan työmäärän.
- **Laitteistoriippumattomuus:** Virtuaalikone ei ole riippuvainen käyttöjärjestelmästä. (PCMag 2008.)

ECMAScript virtuaalikoneita on kehitetty erilaisiin käyttötarkoituksiin. Ensimmäisen sukupolven virtuaalikoneita on 17, ja niistä käsitellään tässä tutkielmassa vain muutama esimerkiksi. SpiderMonkey on ensimmäinen JavaScript-virtuaalikone. Se on toteutettu C-kielellä, ja sen kehitti Bredan Eich Netscapesta. Nykyään virtuaalikonetta hallinnoi Mozilla Corporation. SpiderMonkeyn käyttötarkoituksena on upottaa virtuaalikone muihin sovelluksiin, jolloin se luo ympäristön JavaScriptin toiminnalle. Rhino on avoimen lähdekoodin JavaScript-virtuaalikone. Se on toteutettu Javalla, ja sitä hallinnoi

nykyään Mozilla Corporation. Rhino-projekti alkoi vuonna 1997 ja sen on alun perin kehittänyt Norris Boyd Netscapesta. Rhinossa JavaScript muutetaan Javan luokiksi. Rhino toimii käännettynä sekä tulkattuna. Sen käyttötarkoituksena on pääasiassa palvelinpuolen ohjelmointi. (Mozilla Corporation 2008.) JavaScriptCore on Applen kehittämä JavaScript-virtuaalikone Mac OS X -käyttöjärjestelmälle. Yhdistettynä WebCoren kanssa nämä kaksi ovat osa WebKit-projektia, jota käsitellään tutkielmassa myöhemmin. (WebKit 2008.)

Toisen sukupolven JavaScript-virtuaalikoneita on toistaiseksi kehitetty neljä. Nämä on luotu internet-selaimia varten. Ensimmäinen käsiteltävä virtuaalikone on Tamarin. Se on ilmainen virtuaalikone, jossa on JIT (Just-In-Time) eli ajonaikainen kääntäminen, kuten kaikissa muissakin toisen sukupolven virtuaalikoneissa. Tamariniin on tarkoituksena sisällyttää ECMAScriptin neljännen version standardit. Virtuaalikoneen kehitti alun perin Adobe Systems, mutta Adobe lahjoitti projektin Mozilla Corporationille vuoden 2006 lopussa. Toinen virtuaalikone on TraceMonkey. Virtuaalikone on kehittyneempi versio aikaisemmin mainitusta ensimmäisen sukupolven SpiderMonkeystä. Päämääräinen kehityskohde on ollut Trace Tree -hakupuun lisääminen SpiderMonkeyyn. (Mozilla Corporation 2008.) Trace-johtoisten kääntäjien toiminta perustuu dynaamiseen silmukoiden päiden (engl. header) löytämiseen ja tämän jälkeen kaikkien silmukan reittien (engl. path) tallettamiseen ja kääntämiseen, jotka toteutuvat tarpeeksi usein (Gal 2006). TraceMonkey on vielä kehityksen kohteena, ja sen arvioitu julkaisuaika on vuosien 2008 ja 2009 vaihteessa. Kolmas virtuaalikone Squirrel Fish on WebKit-projektin kehittämä ensimmäisen sukupolven JavaScriptCore-virtuaalikoneen uusi versio. Virtuaalikone kehittyi Squirrel Fish Extremeksi syyskuussa 2008. Tässä uudistuksessa virtuaalikone kääntää JavaScriptin konekielelle, mikä nopeuttaa JavaScriptin toteutusta, koska ei ole tarvetta tulkille. (Mozilla Corporation 2008.) Uusin toisen sukupolven virtuaalikoneista on V8. Tämä virtuaalikone on oleellisin tämän tutkielman kannalta, ja sitä käsitellään tarkemmin seuraavassa luvussa.

2.3 V8-virtuaalikoneen esittely

Internet-sivustojen luonne on muuttunut. Internet-sivustot eivät enää lisää pientä

dynamiikkaa JavaScriptillä, vaan ne sisältävät kokonaisia sovelluksia. Edeltävät JavaScript-virtuaalikoneet ovat suunniteltu pienten sovellusten ajamiseksi. Tarkoituksena on ollut yksinkertaisten asioiden onnistunut ajo internet-sivulla. Nykyään on olemassa hyvin raskaita JavaScript-sovelluksia, joita internet-selaimet ajavat. Googlen mukaan edeltävien virtuaalikoneiden yksinkertainen lähestymistapa ei enää riitä. (McCloud 2008, 1-2.)

Googlen tanskalainen ryhmä on kehittänyt uuden JavaScript-virtuaalikoneen ilman esikuvia (McCloud 2008, 13). Googlen johtoporras haluaa, että heidän asiakkaat aloittavat internet-palveluiden käytön Googlella ja lopettavat sen sillä (Graham 2008). Uuden virtuaalikoneen kehittäminen Chrome-projektinsa pohjaksi oli täten Googlelle välttämätöntä. Google Chrome on internet-selain, joka käyttää JavaScript V8 -virtuaalikonetta. Google Chrome käsitellään tutkielmassa myöhemmin.

2.4 V8:n tehokkuuden perusta

Virtuaalikone on kehitetty laajojen JavaScript-sovellusten nopeaan ajamiseen. Sen tehokkuus perustuu kolmeen toimintoon: nopeaan tilan hakuun, dynaamiseen konekielisen koodin tuottamiseen sekä tehokkaaseen roskien keräämiseen.

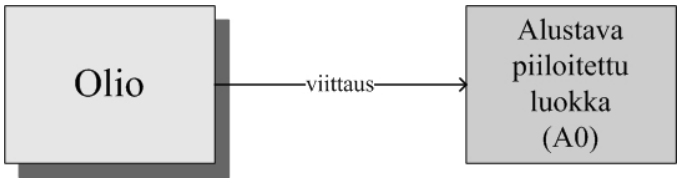
JavaScript on dynaaminen ohjelmointikieli. Olioiden ominaisuuksia on mahdollista sekä lisätä että poistaa ajon aikana. Tämä tarkoittaa sitä, että varsin todennäköisesti olion ominaisuudet muuttuvat. Edeltävät JavaScript-virtuaalikoneet käyttävät sanakirjan (engl. dictionary) kaltaista tietorakennetta tallentaessaan olion ominaisuuksia. Tällöin jokainen haku vaatii dynaamista hakua löytääkseen ominaisuuden sijainnin muistista. Kyseinen toiminta on paljon hitaampaa kuin ilmentymämuuttujien (engl. instance variable) haku ohjelmointikielissä kuten Javassa. V8 ei käytä kyseisen kaltaista tiedonhakua, vaan sen nopeus perustuu piilotettuihin luokkiin (engl. hidden class). Kaikki olioiden ominaisuuksien muutokset perustuvat piilotettuihin luokkiin. (V8 JavaScript 2008.)

JavaScript on luokaton ohjelmointikieli. Kieli mahdollistaa uuden olion luonnin ja dynaamisen ominaisuuksien lisäyksen. V8:ssa ajon aikana oliot, jotka sisältävät samat

ominaisuudet, jakavat saman piilotetun luokan. Dynaaminen optimointi on mahdollista aloittaa tästä eteenpäin. (McCloud 2008, 14.) Dynaamisella optimoinnilla tarkoitetaan sitä, että tehtävä ratkaistaan vaiheittain ja vaiheet yhdistetään rekursiivisesti. Piilotettujen luokkien käytöstä on kaksi hyötyä: tilan haku ei vaadi sanakirjamaista hakua ja se mahdollistaa V8:n käyttää klassista luokkapohjaista optimisointia eli avointa välimuistia.

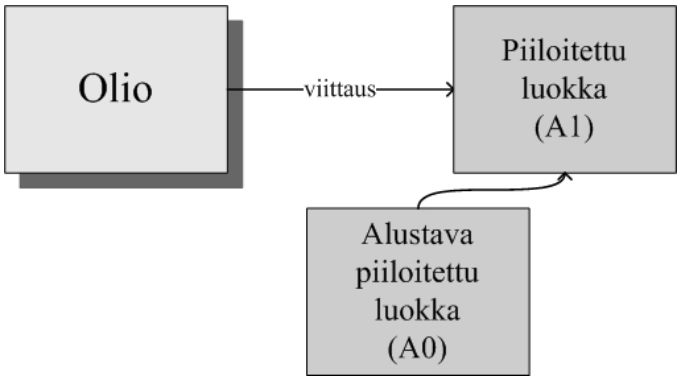
```
function Testi(x, y) {
   this.x = x;
   this.y = y;
}
```

Kun yllä olevan esimerkkilähdekoodin mukaan uusi viittaus luodaan JavaScriptillä (`new Testi(x, y)`), luodaan uusi Testi-olio. Tehdessään tämän ensimmäisen kerran V8 luo alustavan piilotetun luokan A0. Alkuperäinen luokka ei ole määrännyt oliolle vielä mitään ominaisuuksia. Esimerkki on hahmotettu kuvassa 1.



Kuva 1: Piilotetut luokat (1/2)

Ensimmäinen olion ominaisuus asetetaan esimerkki lähdekoodissa seuraavasti: `Testi(this.x = x);` . Tässä tapauksessa V8 luo uuden piilotetun luokan A1 A0:n pohjalta. Tämän jälkeen V8 lisää A1:n tietoihin sen, että oliolla on yksi ominaisuus x, jonka arvo on tallennettu poikkeamaan (engl. prefix) 0 (nolla) Testi-oliossa. Seuraavaksi V8 päivittää A0:n luokkasiirtymisen siten, että jos ominaisuus x lisätään olioon, joka on A0:n kuvaama, pitäisi käyttää piilotettua luokkaa A1 eikä luokkaa A0. Tässä vaiheessa Testi-olion piilotettu luokka on A1. Esimerkki on hahmotettu kuvassa 2.



Kuva 2: Piilotetut luokat (2/2)

Toinen ominaisuus oliolle Testi asetetaan esimerkki lähdekoodissa seuraavasti: `Testi(this.y = y); .` Tämän johdosta V8 luo uuden piilotetun luokan A2, joka pohjautuu A1:een. V8 lisää A2:n tietoihin sen, että sillä on myös ominaisuus Y, joka on tallennettuna poikkeamaan 1 (yksi) Testi-oliossa. Seuraavaksi V8 päivittää A1:n luokkasiirtymisen siten, että jos ominaisuus Y lisätään olioon, joka on A1:n kuvaama, pitäisi käyttää piilotettua luokkaa A2 eikä luokkaa A1. Tässä vaiheessa Testi-olion piilotettu luokka on A2.

Aluksi voi vaikuttaa tehottomalta lisätä aina uusi piilotettu luokka, kun uusi ominaisuus lisätään olioon. Kuitenkin luokkiin siirtymisiä piilotetussa luokissa voidaan käyttää uudelleen. Esimerkiksi jos luodaan uusi Testi-olio, ei luoda yhtään uutta piilotettua luokkaa. Uusi Testi-olio jakaa piilotetut luokat ensimmäisen Testi-olion kanssa seuraavasti:

- Alustavasti uudella Testi-oliolla ei ole mitään ominaisuuksia, vaan uusi olio viittaa alustavaan piilotettuun luokkaan A0.
- Kun lisätään ominaisuus x, V8 seuraa piilotettujen luokkien siirtymää A0:sta A1:een ja kirjaa x:n arvon poikkeamaan, joka on määritetty A1:ssä.
- Kun lisätään ominaisuus y, V8 seuraa piilotettujen luokkien siirtymää A1:stä A2:een ja kirjaa y:n arvon poikkeamaan, joka on määritetty A2:ssa.
  (V8 JavaScript, 2008.)

Toiseksi V8:n tehokkuus perustuu dynaamisen konekielisen koodin tuottamiseen. Aikaisemmin ilmestyneet JavaScriptin virtuaalikoneet kävivät lähdekoodin läpi ja toteuttivat sisäisen esityksen, jonka ne pystyivät tulkkaamaan. Tulkattaessa

ohjelmointikoodia on käytävä läpi jatkuvasti koko sisäisen esityksen rakenne. (McCloud 2008, 15.) Kun sovellus käynnistetään V8:lla, se kääntää lähdekoodin suoraan konekielelle, tulkkia ei ole (V8 JavaScript 2008). Konekieli voidaan ajaa suoraan suorittimelle, joka pitää internet-selainta käynnissä. Kun tulkkaa ja kääntää konekielelle, konekielinen lähdekoodi on itsessään sisäinen esitys JavaScriptin lähdekoodista, eikä sitä ole tarvetta tulkata enempää. (McCloud 2008, 15.)

Esimerkkinä dynaamisesta konekielestä voidaan mainita seuraava: JavaScriptissä olion ominaisuuteen pääsee käsiksi piste-notaatiolla. Lauseessa `testi.x` käsitellään Testi-olion ominaisuutta X. V8:ssa konekielinen tuotos X:n saamiseksi on seuraava: (V8 JavaScript 2008).

```
# ebx = the point object
cmp [ebx, <hidden class offset>],<cached hidden class>
jne <inline cache miss>
mov eax,[ebxm <cached x offset>]
```

Avoimen välimuistin kautta toteutettua tilanhakua on mahdollista päivittää konekielellä V8:n toteutuksen aikana. Olion ominaisuuksiin viittaavan lähdekoodin ensimmäisen toteutuksen aikana V8 määrittää olion senhetkisen piilotetun luokan. V8 optimoi tilanhaun ennakoimalla myös sen, että muut oliot tästä osasta lähdekoodia tulevat käyttämään tätä luokkaa. Tämä tapahtuu siten, että avoimessa välimuistissa oleva lähdekoodi päivitetään käyttämään tiettyä piilotettua luokkaa. Jos tämä ennustus on osunut oikeaan, ominaisuuden arvo asetetaan tai haetaan yhdellä operaatiolla. Ennustuksen ollessa virheellinen, V8 päivittää koodin poistamaan optimoinnin. (V8 JavaScript Engine 2008.)

Kolmannes tehokkuuden peruste V8:ssa on sen tehokas roskienkeruu. Nykyisissä JavaScriptin virtuaalikoneissa roskienkeruu on jäänyt vähemmälle huomiolle. JavaScript sekä muut aikamme olio-ohjelmointikielet käyttävät automaattista muistinhallintaa. Tämä tarkoittaa sitä, että jos viittausta tiettyyn olioon ei enää ole jäljellä, järjestelmä pystyy vapauttamaan sen vaatiman muistin. V8:n käyttää täsmällistä roskienkeruuta. Tämä tarkoittaa sitä, että kaikki viittaukset ovat tiedossa.

(McCloud 2008, 16.) Tämän lisäksi mahdollistaakseen nopean olioiden osoituksen, lyhyen roskienkeruuajan sekä muistin pirstaloitumisen (engl. fragmentation) estämisen V8 on ottanut käyttöön ns. maailman pysäyttäjän (engl. stop-the-world) roskienkerääjäksi. Tämä tarkoittaa sitä, että V8 pysäyttää sovelluksen suorituksen, kun se toteuttaa roskienkeruukierrosta. Pienentääkseen sovelluksen pysäytysaikaa V8 käsittelee vain osan oliokeosta. Oliokeko on jaettu kahteen osaan. Ensimmäinen osa on alue, jossa oliot luodaan. Toinen osa sisältää sen alueen, jossa ovat roskienkeruusta selviytyneet oliot. Säilyttääkseen viittausten oikeellisuuden V8 päivittää kaikki viittaukset olioon, jos se on siirtynyt roskienkeruunaikana. (V8 JavaScript Engine 2008.)

3 GOOGLE CHROME

Google Chrome on Googlen kehittämä avoimella lähdekoodilla toteutettu WWW-selain eli internet-selain (CDD 2008). Internet-selain on ollut kehityksen alla jo kaksi vuotta (Green 2008). Julkinen beta-versio julkaistiin 2.9.2008 yli sadassa maassa (Lehto 2008) ja sitä on mahdollista käyttää 43 kielellä (Google 2008). Chromen taustalla on avoimen lähdekoodin projekti nimeltään Chromium. Projekti on toteutettu C++ kielellä. Chromen lähdekoodi sekä Chromiumin suoritettava tiedosto ovat lisenssoitu BSD:llä (Berkeley Software Distribution) (CDD 2008). Google palkkasi sarjakuvakirjailijan nimeltä Scott McCloud kehittämään heidän kokouksistaan sarjakuvan, joka kuvastaisi millainen projekti on Google Chrome (Gustines 2008).

Internetissä oli liikkunut vuosia sitten huhuja Googlen yrityksestä kilpailla Microsoftin kanssa internet-selaimella. Kaksi vuotta sitten Googlen johtoporras päätti kehittää internet-selaimen. Johtoporras päätti myös sen, että on parempi ohjelmoida oma selain, kuin sanella vaatimuksia Googlen rahoittamalle Firefox-selaimelle eli toiselle yritykselle. Chromen on tarkoitus kilpailla Internet Explorerin, Mozilla Firefoxin sekä Applen Safarin kanssa internet-selainten markkinoista. (Graham 2008.)

Googlen johtoportaan mukaan tavoitteena ei ole pelkästään voittaa osa internet-selain markkinoista, vaan vaikuttaa internet-selailuun kokonaisuudessaan. Chromen on tarkoitus siirtää tietojenkäsittelyä Microsoftin työpöytä alueelta Googlen etä-työpöytä alueelle. Googlen mukaan Google Chrome tulee nopeuttamaan ja tekemään turvallisemmaksi sovelluksia, jotka ovat käyttäjän käyttöjärjestelmän ulkopuolisia. Microsoft kuitenkin uskoo siihen, että käyttäjät vaativat tehokkaita paikallisia sovelluksia sekä tietokoneita työpöydäkseen myös jatkossa. Kuitenkin Microsoft on kehittänyt uusimpaan internet-selaimeensa Internet Explorer 8:aan toiminnon, joka käyttäjän valitessa estää niiden tietojen lähettämisen, mitkä auttavat Googlea löytämään käyttäjäkohtaisia mainoksia internet-sivustoihin. Internet Explorerin pääsuunnittelija Dean Hachamovich pelkää sitä, että Google lähtee internet-selaimellaan erottamaan sen, ja sen käyttäjät, muusta internetin liikenteestä. (Green 2008.)

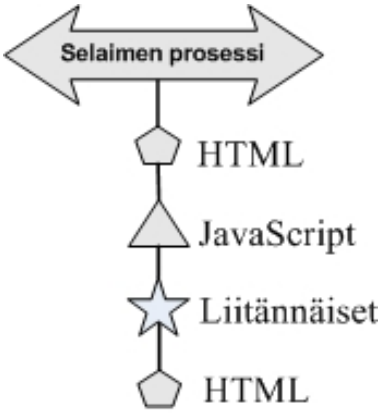
3.1 Käyttöliittymä

Chrome on ensivaikutelmaltaan muiden ajan internet-selaimien kaltainen (kts. liite 1). Uudistukset ilmenevät kuitenkin nopeasti käytön aikana. Muut internet-selaimet avaavat joko käyttäjän määrittelemän kotisivun tai edellisellä käyttökerralla suljetut välilehdet. Chromen käynnistyessä on mahdollista toteuttaa jokin neljästä vaihtoehdosta: avata tietyt sivut käyttäjälle välilehtiin, palauttaa edellisen käyttökerran sivut takaisin välilehtiin, siirtää käyttäjä omalle kotisivulle tai siirtää käyttäjä omalle etusivulle. Oma etusivu ei tarkoita tässä yhteydessä käyttäjän valitsemaa kotisivua. Omalla etusivulla ovat kuvakkeet yhdeksästä sivustosta, joissa käyttäjä on eniten vieraillut. Tämän lisäksi omalta etusivulta on mahdollista hakea omasta selaushistoriasta, nähdä koko selaushistorian ja nähdä uusimmat suosikit.

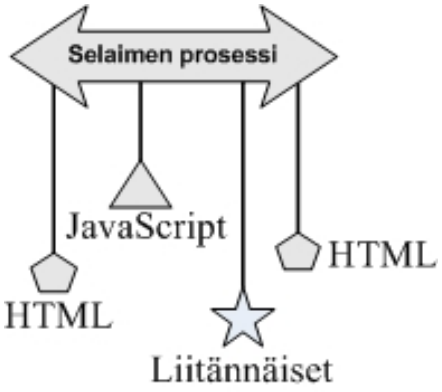
Välilehdet ovat oleellisia Chromen toiminnassa. Tästä johtuen ne ovat siirretty graafisesti kaikista ylimmäksi kun kaikissa muissa internet-selaimissa välilehdet ovat internet-selaimen kontrollien alapuolella (kts. liite 1). Jokainen välilehti voidaan käsittää omana vanhanaikaisena edellisen vuosikymmenen internet-selaimena, joissa on oma osoitteen syöttörivi sekä välilehteen kuuluvat kontrollit. Toiseksi syöttörivin toimintaa on muutettu. Chromesta puuttuu kokonaan hakemiseen tarvittava syöttörivi. Tämä johtuu siitä, että nämä kaksi riviä on yhdistetty yhdeksi. Osoitteen syöttörivi toimii osoitteiden ja hakusanojen vastaanottajana sekä ehdotusten antajana. Tätä yhdistelmää Googlen kehitystiimi kutsuu nimellä Omnibox. Kolmanneksi ponnahdusikkunat toimivat eritavalla. Ponnahdusikkunat ovat liitetty siihen välilehteen, missä ne ovat alun perin käynnistyneet. Käyttäjä voi halutessaan siirtää ponnahdusikkunan pois tietystä välilehdestä. Tämä tarkoittaa sitä, että ponnahdusikkunat eivät toimi omassa prosessissa. Chromen prosessit käsitellään tutkielman seuraavassa luvussa. Neljänneksi tilarivin toimintaa on muutettu. Muissa internet-selaimissa tilarivin saa näkyviin ja pois näkyvistä halutessaan näytä-valikon kautta. Chromessa tilarivi on näkyvissä silloin, kun on näytettävää tietoa kuten internet-linkki. Muulloin tilarivi häviää näkyvistä. Viidentenä internet-selaimella voi luoda käyttäjän valitsemasta internet-sivustosta ohjelmapikapainikkeen. Painikkeen voi käynnistää esimerkiksi työpöydältä, jolloin internet-selain avaa internet-sivuston ilman internet-selaimen kontrollipainikkeita.

3.2 Muistin käyttö

Google Chormen kehittäjäryhmän mielestä nykyisten internet-selaimien ongelmana on se, että ne ovat luonnostaan yksisyklisiä. Tämä tarkoittaa sitä, että esimerkiksi jos internet-selain lähtee suorittamaan JavaScript-sovellusta, internet-selain ei voi tehdä mitään ennen kuin JavaScript palauttaa hallinnan internet-selaimelle. Kuvassa 3 on esimerkki yksisyklisestä internet-selaimesta. Jos JavaScript ei palauta hallintaa, internet-selain voi lopettaa toimimasta. Chrome ei ole toteutettu yksisyklisenä. Kehitystiimi pohti ratkaisuksi monisyklistä vaihtoehtoa, joka on esitetty kuvassa 4, mutta päätyi hylkäämän myös sen. Chrome on moniprosessinen. Tämä tarkoittaa sitä, että prosessit ovat eriteltynä samalla periaatteella kuin moderneissa käyttöjärjestelmissä. Havainnollistus moniprosessisesta internet-selaimesta on kuvassa 5. Tämä mahdollistaa sen, että yksi internet-selaimen välilehti voi olla kiireinen, kun taas toisia voi käyttää sillä aikaa vapaasti. Jos tämä kiireinen välilehti esimerkiksi kaatuu, se ei kaada koko internet-selainta, vaan kyseisen välilehden (kts liite 2). Tekniikka käyttää enemmän muistia internet-selaimen käynnistettäessä kuin muut internet-selaimet, mutta vähän ajan kuluttua käytön aikana se tulee kuluttamaan vähemmän muistia kuin muut internet-selaimet. (McCloud 2008, 3-5.)



Kuva 3: Yksisyklinen



Kuva 4: Monisyklinen

Tavanomaisissa internet-selaimissa on vain yksi prosessi, sekä yksi osoiteavaruus johon käyttäjä lataa dokumenttinsa. Muistia voi vapauttaa sulkemalla ylimääräiset välilehdet. Uuden välilehden avaamiseen voidaan käyttää muistia, joka aikaisemmin vapautettiin. Ajan kanssa tapahtuu kuitenkin muistin sirpaloitumista (engl. fragmentation). Tämä

tarkoittaa sitä, että osa muistista jää silti käyttöön, vaikka välilehti suljettaisiin. Tämä voi johtaa siihen, että kun käyttäjä avaa uuden välilehden, se ei mahdu osoiteavaruuteen. Täten käyttöjärjestelmän on laajennettava internet-selaimen tarvitsemaa osoiteavaruutta. Mitä kauemmin käyttäjä käyttää internet-selainta, sitä enemmän tätä prosessia tapahtuu. Chromessa välilehden sulkeminen sulkee koko prosessin, ja sen prosessin koko muisti vapautuu. Esimerkiksi jos käyttäjän avaama internet-sivu vuotaa muistia, se ei vaikuta käyttäjän käyttöjärjestelmään pitkään, koska sivun sulkeminen vapauttaa koko prosessin vaatiman muistin. Huomioitava asia internet-selaimen siirtymisessä sivustosta toiseen on vielä se, että kahdella toimialueella ei tarvitse olla minkäänlaista yhteyttä toisiinsa. Chrome pystyy pyyhkimään vanhan toimialueen kääntäjän, vanhat tietorakenteet sekä vanhat prosessit pois uuden toimialueen tieltä tehokkaalla roskienkeruulla. (McCloud 2008, 4-7.)

Kuva 5: Moniprosessinen internet-selain

Internet-selaimessa on modernien käyttöjärjestelmien tavoin tehtävienhallinta. Tämän avulla on mahdollista nähdä mitkä sivut tai liitännäiset kuluttavat keskusyksikön muistia ja kuinka paljon. Tehtävienhallinnan avulla on mahdollista poistaa prosessit, jotka hidastavat internet-selaimen tai tietokoneen toimintaa. (McCloud 2008, 7-8.)

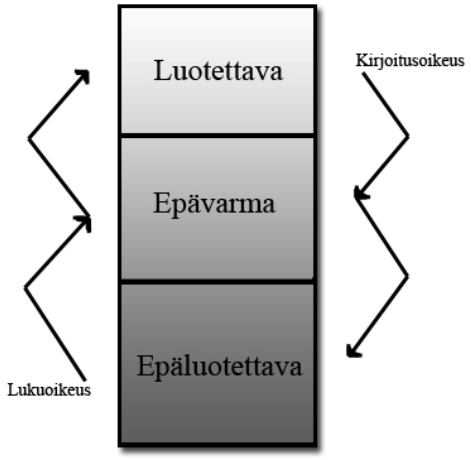
3.3 Turvallisuuden edistäminen

Edellisen vuosikymmenen internet-selaimet painottivat toimintaansa onnistuneeseen internet-sivun näyttämiseen käyttäjän tietokoneella. Ei ollut aloitteita asentaa

haittaohjelmia käyttäjien tietokoneille. Tällä vuosikymmenellä tilanne on muuttunut. Haittaohjelmat keskittyvät salasanojen varastamiseen ja rahan siirtämiseen. (McCloud 2008, 25.)

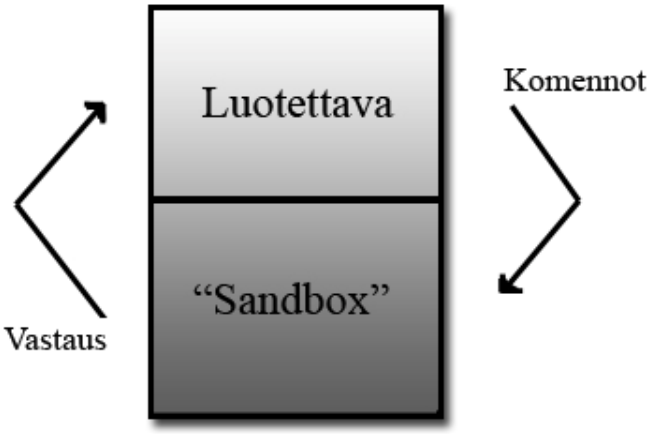
Chromen lähtökohtana on se, että internet-selaimesi tulee olemaan haittaohjelmien hyökkäysten kohteena. Chromen toiminta perustuu hiekkalaatikko-periaatteeseen (engl. Sandbox). Tarkoituksena on estää haittaohjelmien asentuminen käyttäjän tietokoneelle tai estää yhden internet-selaimen välilehden vaikuttamisen muihin välilehtiin. Jokainen Chromen välilehti on oma prosessinsa, jonka oikeudet ovat rajoitettu. Nämä prosessit voivat ajaa, mutta eivät voi kirjoittaa tiedostoja käyttäjän tietokoneen kovalevylle. Lukuoikeus tietokoneen luottamuksellisilta alueilta, kuten omat dokumentit tai työpöytä, ovat myös poistettu. Tämä tarkoittaa sitä, että kolmannet osapuolet eivät voi seurata käyttäjän kirjoittamia tietoja, muuttaa hiiren toimintoja, eivätkä voi määrätä Windowsia ajamaan suorittava tiedosto käynnistyksen aikana. Jos tietty välilehti toteuttaa haitallista toimintaa, sen sulkeminen hävittää koko toiminnan. Järjestelmä sekä muut internet-selaimen prosessit ovat turvassa. (McCloud 2008, 26-27.)

Hiekkalaatikko-periaatteen rajat määritetään käyttöoikeuksilla. Esimerkiksi Microsoft Vista käyttää Kenneth J. Biba -turvallisuusmallia, jossa on kolme tasoa. Kyseiset kolme tasoa ovat havainnollistettu kuvassa 6. Ylin taso on luotettava, keskimmäinen taso on epävarma ja alin taso ei ole luotettava lainkaan. Ylin taso on tarkoitettu varmuuskopiointiin sekä sovelluksiin, jotka päivittävät itseään. Keskimmäinen taso on tarkoitettu käyttäjän normaaleihin suorituksiin. Lukeminen on sallittua alimmalta tasolta ylös, mutta kirjoittaminen vain ylhäältä alas. Ongelmana on keskimmäinen taso. Tässä tasossa on myös hyvin paljon luottamuksellista käyttäjänkohtaista tietoa, jota alimman tason toiminnot eivät saisi lukea. (McCloud 2008, 27-28.)



Kuva 6: Microsoft Vista, Biba.

Chromessa on vastaava turvallisuusmalli toteutettu seuraavasti: ylimmällä tasolla on käyttäjä, ja alimmalla tasolla hiekkalaatikko-mallin sisällä tapahtuvat toiminnot. Kyseinen malli on kuvattuna kuvassa 7. Alimmalta tasolta voidaan vain vastata pyyntöihin ylemmälle tasolle, mutta se ei saada tietoonsa mitään, mitä käyttäjä ei ole antanut. Tämä on mahdollista Chromessa, koska sen koodi on uutta ja tähän tarkoitukseen toteutettu. Kuitenkin on olemassa yksi poikkeus, internet-selaimien liitännäiset. Järjestelmän käyttöoikeuksissa Chrome voi toimia matalilla oikeuksilla, mutta liitännäiset voivat olla samalla tasolla tai jopa korkeammalla kuin internet-selain. Liitännäisillä on toimintoja, jotka eivät ole yleisiä käytäntöjä. Chromessa kuitenkin liitännäiset ovat oma prosessinsa, ja tämän tähden koko muu internet-sivusto on mahdollista asettaa hiekkalaatikon sisälle, vaikka liitännäinen ei laitettaisikaan. Liitännäiset keskustelevat suoraan tulkin kanssa, mikä toimii matalilla oikeuksilla. (McCloud 2008, 29-31.)



Kuva 7: Chromen turvallisuusmalli

Chromessa on tuki NPAPI:lle (Netscape Plugin Application Programming Interface), mutta ei ole tukea upottaa ActiveX hallintoja. NPAPI:iin kuuluvat seuraavat: Flash, QuickTime, Java, Real, Acrobat, Windows Media ja Silverlight. Chromessa ei ole Mozillan kaltaista laajennosta XPInstall-arkkitehtuuriin. Javan Appletit ovat tuettu tulevan Javan version 6 päivitys 10 kanssa. (Wikipedia 2008.)

Hiekkalaatikko-malli auttaa käyttäjää torjumaan haittaohjelmia, mutta phishing-tekniikoita ei pystytä tarpeeksi tehokkaasti estämään tämän mallin avulla. Phishingissä

hyökkääjät lähettävät todentuntuisia sähköposteja käyttäjille, joissa pyydetään luottamuksellisia tietoja. Estääkseen tätä toimintaa Chrome lataa jatkuvasti kahta Googlen päivittämää listaa vaarallisista internet-sivustoista. Ensimmäinen lista käsittää internet-sivustot, joissa on haittaohjelmia. Toinen lista sisältää internet-sivustot, joissa on phising-toteutusta. Käyttäjän siirtyessä haitalliselle internet-sivustolle Chrome antaa käyttäjälle varoituksen ja mahdollisuuden joko poistua internet-sivustolta tai ladata se varoituksesta huolimatta (kts. liite 3). (McCloud 2008, 32-33.)

Chromessa on Incognito-toiminto, joka estää internet-selainta tallettamasta mitään selaushistoriaa tai evästeitä käyttäjän vierailemilta internet-sivustoilta. Se ei estä internet-sivuja vaihtamasta sitä tietoa, missä käyttäjä oli aikaisemmin liikkunut. Google muistuttaa käyttäjää varomaan seuraavia Incognito-tilassa:

- Verkkosivustoja, jotka keräävät tai jakavat tietoa käyttäjästä.
- Internet-palveluntarjoajia, jotka seuraavat sivuja, joilla käyttäjä vierailee.
- Haittaohjelmia, jotka seuraavat näppäinpainalluksia ja antavat vastineeksi ilmaisia hymiöitä
- Turvallisuuspalveluiden työntekijöiden tekemää seurantaa.
- Internet-selaimen käyttäjän selän takana olevia ihmisiä.

Vastaava toiminto on Apple Safari -internet-selaimessa sekä uusimmassa Internet Explorer versio 8:ssa. (Baig 2008.)

3.4 Turvallisuusongelmat ja pettymykset

Tähän mennessä tutkielmassa on käsitelty lähestulkoon pelkästään Chromen hyviä tai uusia puolia. Kuitenkin Chromessa on myös huonoja puolia, jotka on käsiteltävä. Merkittävimmät huonot puolet ovat laajennusten puuttuminen, yksityistiedon kerääminen sekä Chromen EULA (End User License Agreement) julkaisussa tapahtunut virhe.

Firefoxista voi helposti saada laajennusten avulla todella monipuolisen internet-selaimen. Chrome ei ainakaan vielä tue millään tasolla internet-yhteisön tekemiä

laajennuksia. Toinen huono puoli on se, että Chrome tilastoi kaiken, mitä käyttäjä tekee yllämainitun Incognito-tilan ulkopuolella. Chrome voi lähettää kaiken Googlen palvelimille, mitä käyttäjä kirjoittaa Chromen Omniboxiin tai internet-sivujen tekstikenttiin, ja mukaan vielä käyttäjän IP-osoitteen. Kolmanneksi Chrome voi ladata tiedostot internetistä automaattisesti ilman käyttäjän vahvistusta sivustoilta. Google väittää tähän syyksi sen, että tiedostojen lataus ei ole vaarallista, vaan automaattinen ladattujen tiedostojen ajaminen on. (Berschewsky 2008.) Neljäntenä Chromesta puuttuu tällä hetkellä myös RSS-syötteiden lukija.

Edellä mainitut huonot puolet ovat olleet suuri pahennuksen herättäjä internet-yhteisöissä. Viimeisenä mainittakoon Chromen julkaisussa ilmentyneen EULA:n mukaan Google saa oikeuden näyttää ja levittää kaiken mikä kulkee Chromen kautta. Tämä herätti suurta kohua, ja Google on korjannut EULA:n sisällön tästä johtuen. (Anderson 2008.)

3.5 Tehokkuuden tausta

Google Chromen pohjana toimii WebKit, joka on avoimen lähdekoodin runko. Tämän rungon päälle on mahdollista kehittää internet-selain. WebKitin on alun perin kehittänyt Apple Inc. Rungon päälle on kehitetty lukuisia sovelluksia, ja niistä uusin on tällä hetkellä Googlen Chrome internet-selain. WebKit on siis virtuaalikone internet-selaimien rakentamiseen. WebKit-projekti on itse määritellyt heidän tärkeimmiksi tavoitteiksi vakauden, tehokkuuden, turvallisuuden, käytettävyyden, koodin selkeyden sekä siirrettävyyden. WebKit-projekti ei aio itse kehittää internet-selainta, vaan jättää sen muiden tehtäväksi. Kuka vain voi ladata uusimman WebKit-kokoonpanon tietokoneellensa ja kääntää sen.

WebKitin runko on kehitetty neljästä osasta, jotka käsitellään seuraavaksi. Ensimmäinen osa on WebCore, joka on pohjapiirros, DOM-kirjasto HTML:lle ja SVG:lle (Scalable Vector Graphics), ja sen on kehittänyt WebKit-projekti. SVG on kieli, joka kuvaa kaksiulotteista grafiikkaa sekä graafisia sovelluksia XML:ssä (eXtensible Markup Language) (W3C 2008). Toinen osa on JavaScriptCore, joka on aikaisemmin tutkielmassa käsitelty virtuaalikone, KJS-kirjasto (KDE’s JavaScript

Engine) sekä PCRE (Perl Compatible Regular Expressions) säännöllisten ilmausten kirjasto. PCRE-kirjasto on kokoelma funktioita, jotka toteuttavat säännöllisten ilmausten toimintamallia Perl 5:den syntaksilla ja semantiikalla. Säännöllinen ilmaus (Regular Expression) on taas tavallaan ohjelmointikieli ohjelmointikielen sisällä. Niiden avulla on mahdollista kuvata sääntöjoukko, jota sovitetaan johonkin merkkijonoon. (PCRE 2008.) Syyskuun 18. päivänä 2008 JavaScriptCore korvattiin SquirrelFish virtuaalikoneella. WebKitin kolmas osa oli Drosera, joka on JavaScript-virheidenjäljittäjä. Droseraa on mahdollista käyttää minkä tahansa sovelluksen kanssa, joka on WebKitn pohjalta toteutettu. Drosera on lähes kokonaan (90 %) toteutettu HTML:llä, CSS:llä (Cascending Style Sheets) ja JavaScriptillä. Drosera on kuitenkin korvattu tehokkaammalla virheidenjäljittelijällä, Web Inspectorilla, vuonna 2006. (WebKit 2008.) Viimeinen osa WebKitiä on SunSpider, jonka tehtävänä on arvioida JavaScriptin tehokkuutta. SunSpider käsitellään tarkemmin seuraavassa luvussa.

WebKitissä on kuitenkin myös huonoja puolia, ja nämä huonot puolet tietenkin siirtyvät myös Chromeen. Chrome käyttää WebKitin versiota, jossa on toteutusvirhe, joka on nimetty Carpet-Bombingiksi. Tässä toteutuksessa huijataan käyttäjää ajamaan suorittava tiedosto suoraan internet-selaimesta. Google Chromen ensimmäistä julkista versiota pystyy siis huijaamaan siten, että se lataa ja käynnistää JAR-tiedoston (Java Archive) automaattisesti. (Andy 2008.)

3.6 Vertailu ja testit

Testaus on kriittistä jokaiselle uudelle sovellukselle. Googlella on mahdollista testauttaa internet-selaimensa uusin versio puolessa tunnissa kymmenillä tuhansilla internet-sivustoilla. Joka viikko Chrome-Bot testaa miljoonia internet-sivuja ja antaa kehittäjille tärkeätä tietoa, jota jouduttaisiin muuten odottamaan ulkoisen beta-version tuloksista. Lähtökohtana on löytää ongelmat mahdollisimman aikaisin, jolloin niiden korjaaminen mahdollisimman helppoa. Googlen teknologioiden avulla on mahdollista luokitella internet-sivustot joissa peruskäyttäjät yleensä vierailevat. (McCloud 2008, 9-11.)

Yleisimmät internet-selaimet tällä hetkellä ovat seuraavat: (W3schools 2008)

Selain	Valmistaja	Ilmestyi	Päättyi	Saatavuus
Navigator	Netscape	1994	2008	browser.netscape.com
Internet Explorer	Microsoft	1995	Jatkuu	Windows
Opera	Opera Software	1996	Jatkuu	www.opera.com
Firefox	Mozilla Corporation	2002	Jatkuu	www.mozilla.fi
Safari	Apple	2003	Jatkuu	www.apple.com/safari
Sea Monkey	Sea Monkey Council	2005	Jatkuu	www.seamonkey-project.org
Chrome	Google	2008	Jatkuu	www.google.com/chrome

Taulukko 2: Yleiset internet-selaimet

Tutkielmaan on otettu mukaan pieni testaus-osuus, joka on toteutettu seuraavilla sivulla mainittujen internet-selainten versioilla:

Internet-selain	Versio
Navigator	9.0.0.6
Internet Explorer	7.0.5730.13
Opera	9.60
Firefox	3.03
Safari	3.1.2
Sea Monkey	1.1.12
Chrome	0.2.149.30

Taulukko 3: Internet-selaimien versiotiedot

Ensimmäisenä testinä on käytetty SunSpider JavaScript Benchmark -testiä, joka on avoimessa käytössä, kuten kaikki muutkin tutkielmassa käytetyt testit. Ensimmäisessä testissä verrataan selainten JavaScriptn nopeutta. Mitä pienempi on tulos, sitä parempi internet-selaimen kannalta. Testi on pyritty toteuttamaan kahden periaatteen mukaan: testi keskittyy oikeisiin ongelmiin, joita JavaScript-kehittäjät kohtaavat, sekä testi on tasapainoinen eli testejä on erilaisissa kategorioissa. Testin tulokset ovat seuraavalla sivulla. (SunSpider 2008.)

Tehtävä	Navigator	Internet explorer	Opera	Firefox	Safari	Sea Monkey	Chrome
3D	4037	2856	884	746	806	3722	204
Access	2459	3465	1220	906	1009	2309	136
Bitops	6950	3109	1059	728	712	6581	115
Controlflow	191	1012	106	68	172	190	5
Crypto	1097	1994	428	386	491	1375	10
Date	6900	1768	665	595	821	7100	1478
Math	2152	2350	699	694	909	2424	211
Regexp	1378	600	725	313	403	1462	577
String	5285	53934	2181	1565	1618	5682	973
Tulos (ms)	**30451**	**71091**	**7970**	**6022**	**6943**	**30847**	**3808**

Taulukko 4: Tulokset SunSpider-testistä

Toisena testinä on käytetty Web Standards -projektin Acid-tuoteryhmän testejä. Testejä on yhteensä kolme. Ensimmäinen testi kehitettiin vuonna 1998. Sillä oli tärkeä rooli varhaisten internet-selaimien CSS1:n toiminnan tarkastuksessa. Toinen testi julkaistiin vuonna 2005. Testin tarkoituksena oli edelleen tarkastella kehittyneempää CSS:n toimintaa. Kolmas testi kehitettiin maaliskuussa 2008. Se määrittää päämääräisesti internet-selaimen DOM:in sekä JavaScriptin toimintaa. Testin julkaisuhetkellä yksikään selain ei läpäissyt testiä täysin pistein. Testin tulokset ovat alla: (Acid Tests 2008.)

	Acid 1	Acid 2	Acid 3
Navigator	Suoriutui	Epäonnistui	52 / 100
Internet Explorer	Suoriutui	Epäonnistui	Epäonnistui
Opera	Suoriutui	Suoriutui	78 / 100
Firefox	Suoriutui	Suoriutui	70 / 100
Safari	Suoriutui	Suoriutui	75 / 100
Sea Monkey	Suoriutui	Epäonnistui	53 / 100
Chrome	Suoriutui	Suoriutui	**79 / 100**

Taulukko 5: Tulokset Acid-testistä

Kolmas testi sisältää Googlen käyttämiä JavaScript-testejä, joilla he ovat kehittäneet V8-virtuaalikoneen. Testi jakautuu viiteen eri aihealueeseen. nämä aihealueet ovat lueteltuna alla:

- Richards: Käyttöjärjestelmän simulaatio -testi (Martin Richards)
- DeltaBlue: Yhdensuuntaisten muuttujien ratkaisija (John Maloney and Mario Wolczko)
- Crypto: Salaus ja salauksenpurku -testi (Tom Wu)
- RayTrace: RayTrace -testi (Adam Brumister)
- EarleyBoyer: Perinteisiä testejä (Florian Loitsch's Scheme2Js complier)

(V8 Benchmark 2008.)

Tulos on viiden erilaisen testin keskiarvo. Mitä korkeampi pisteytys, sitä paremmin internet-selain menestyi JavaScriptin ajamisessa. Tulos ei ole yllätys Chromen kannalta, sillä juuri tätä testiä on kerran käytetty V8:n toiminnan tehostamiseen. Kuitenkin erot muihin internet-selaimiin ovat yllättävän suuria tässä testissä. Testin tulokset seuraavassa taulukossa:

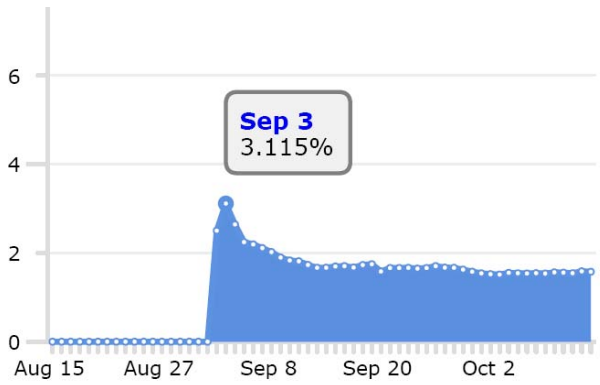
Tehtävä	**Navigator**	**Internet explorer**	**Opera**	**Firefox**	**Safari**	**Sea Monkey**	**Chrome**
Richards	57	14	86	106	58	59	1465
DeltaBlue	45	12	95	120	65	51	1036
Crypto	45	33	73	69	77	46	835
RayTrace	59	29	227	97	119	56	558
EarleyBoyer	53	44	409	151	163	51	1456
Tulos	**51**	**23**	**141**	**105**	**89**	**23**	**1006**

Taulukko 6: Tulokset V8 Benchmark –testistä

4 YHTEENVETO

Muutamia kuukausia on kulunut Google Chromen julkaisusta. Mielenkiintoisinta on nähdä kuinka pitkälle Chrome on päässyt internet-selainten välisessä kamppailussa. Googlella oli suuret odotukset internet-selaimen menestymisestä, mutta muutamat internet-selaimeen liittyvät kritiikit ovat olleet voimakkaita. Internetin käyttäjät pitävät tärkeänä anonyymisyyden säilymistä. Käyttäjien keskuudessa voidaan pitää suurena ongelmana sitä, että Chromen Omnibox voi lähettää jokaisen painalluksen Googlen palvelimille. Google on kyllä ilmoittanut lyhentävänsä palvelimillensa talletettujen IP osoitteiden säilytysaikaa 18 kuukaudesta 9:ään (Fleischer 2008.) Tämä tarkoittaa sitä, että osoitteet tehdään nimettömiksi ajan jälkeen. Chromessa on kuitenkin mahdollista valita pois päältä Googlen tarjoamat ehdotukset Omniboxissa tai vaihtaa hakukonetta kokonaisuudessaan. Ensimmäisessä Googlen julkaisemassa käyttäjäsopimuksessa oli myös virheellisesti kohta, jossa kaikki tuotos mikä kulkee Chromen kautta, on Googlen omistamaa.

Mitä hyötyä Googlelle on Chromesta? Nopea internet-selailu tarkoittaa myös nopeampia hakuja sekä enemmän internet-mainontaa. Näihin kahteen Googlen toiminta pääosin perustuu.

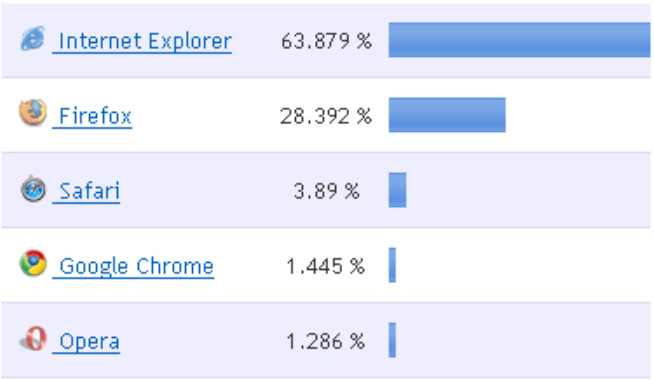


Kuva 8: Chromen markkinaosuus (GMS 2008.)

Yllä olevassa kuvassa on Chromen markkinaosuus internet-selaimista sen julkaisupäivästä lähtien. Suurin käyttäjä määrä oli ensimmäisinä päivinä. Käyttäjien

määrän lasku tämän jälkeen johtuu siitä, että kokeilukäyttäjät siirtyvät takaisin alkuperäisen internet-selaimensa pariin. Chrome on vasta alkutiellään, mutta kuvasta päätellen ei Chrome ainakaan lähiaikana tule viemään suurta markkinaosuutta internet-selaimissa. Osa syytä tähän voi olla se, että sovellus on beta-tilassa. Vaikka sen on ilmoitettu olevan turvallinen, se on kuitenkin vielä viimeistelemätön tuote. Internet-selaimen pitkäaikaisen käytön jälkeen tulee näkyviin se, miten hyvin internet-selaimeen on toteutettu turvallisuusperiaatteet ja on otettu XSS huomioon. Käyttöön liittyviä ongelmia tulee varmasti vastaan vielä enemmän, kuin niitä nyt on. Suurin osa Chromen kannattajista voivat siirtyä internet-selaimen pariin vasta vähän myöhemmin, kun sitä on kehitetty eteenpäin.

Suurin osuus internet-selaimista on edelleen Microsoftin Internet Explorerilla (IE). Mozillan Firefox seuraa toisena ja Googlen Chrome neljäntenä, alle 2 % markkinaosuudella. Alla olevassa kuvassa on viisi suurinta internet-selainta ja niiden markkinaosuudet lokakuussa 2008.



Kuva 9: Internet-selainten markkinaosuus (GMS 2008.)

Vaikuttaisi siltä, että Chrome ei välttämättä tule saavuttamaan Googlen haluamaa paikkaansa. Kuitenkin projektin varrella kehitetty JavaScriptin V8 -virtuaalikone edistää JavaScriptin toimintaa huomattavasti. V8:n tehokkuus perustuu kolmeen toimintoon: nopeaan tilan hakuun, dynaamiseen konekielisen koodin tuottamiseen sekä tehokkaaseen roskien keräämiseen. Tästä johtuen Chrome-projekti ei missään tapauksessa ollut turha internet-selain. Projekti on avointa lähdekoodia, eli kuka tahansa voi hyödyntää näitä Googlen kehittämiä työkaluja. Chrome on ylivoimaisesti kaikista

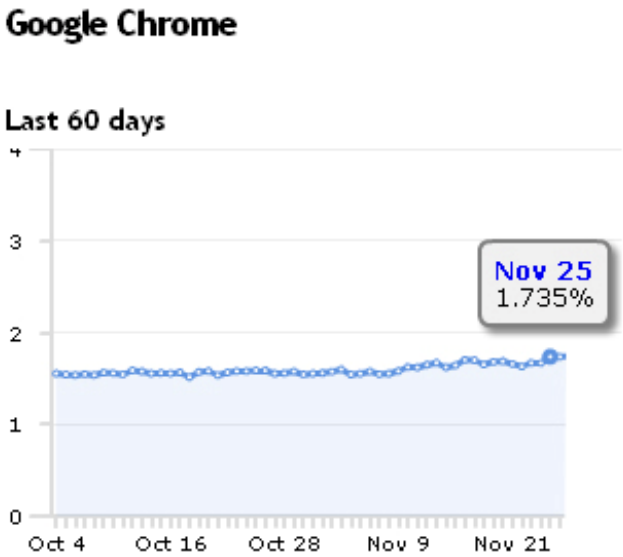
internet-selaimista nopein JavaScriptin suorittaja tutkielman tekohetkellä, mutta kuinka monta tuntia lisää suunnittelua aikovat WWW-kehittäjät käyttää näin pienelle osalle käyttäjiä? Heille tämä on kuitenkin vain yksi internet-selain lisää tarkistettavaksi.

Chrome on tarkoitettu tulevaisuuden internetin käyttöjärjestelmäksi. Chromen ilmestyminen vaikutti suuresti, mutta näyttäisi siltä, ettei niin suuresti kuin Google olisi toivonut. Tuskin heidän kehittäjänsä istuivat viimeisessä kokouksessa toivoen saavuttavansa lokakuuhun mennessä 1 % markkinaosuuden. Näyttää siltä, että Chromesta tulee aluksi kehittäjien alusta, työkalu testata JavaScriptin nopeutta menestyneitä internet-selaimia vastaan. Kuitenkin Googlen mukaan Chromelle ei ollut asetettu mitään käyttäjätavoitteita: ”Google tarjoaa palveluita, ja ihmiset käyttävät niitä, jos haluavat” kertoo Peter Fleisher. Muita tavoitteita Chromella oli ja niiden katsotaan toteutuneen. Muihin tavoitteisiin kuuluivat: valinnanvapauden lisääminen, avoimen lähdekoodin käyttö, nopean internet-selaimen tarjoaminen, sekä innovaatioiden tuomista internet-selainten maailmaan. (Fleisher 2008.)

Julkisesti Googlen johto on tarkoittanut Chromen kamppailemaan Microsoftin Internet Explorerin kanssa. Kuitenkin Chromen julkaisu tulee vaikuttamaan Firefoxiin. Käyttäjät tulevat määräämään tämän kamppailun lopputuloksen. Ensivaikutelma kuitenkin tuntuu siltä, että Firefox kärsii enemmän kuin Internet Explorer Google Chromen julkaisusta. Ne käyttäjät, jotka ovat tyytyväisiä siihen mitä Windows tuo mukanaan, eivät lataa ylimääräistä internet-selainta. Käyttäjät jotka hakevat kolmannen osapuolen internet-selaimia, voivat siis helpommin vaihtaa Firefoxista Chromeen, kuin Internet Explorerin käyttäjät. Chromessa on innovatiivinen käyttöliittymä, erilainen kuin muissa internet-selaimissa. Internet-selaimen moniprosessinen muistinhallinta lähenee käyttöjärjestelmien muistinhallintaa, jossa on mahdollista katsoa mikä prosessi kuluttaa järjestelmän muistia ja pysäyttää se. Google on myös mainostanut sen olevan turvallinen. Ilman esikuvia tuotettu lähdekoodi mahdollistaa tuotteen räätälöinnin yrityksen tarkoitukseen. Chromesta yritetään tehdä turvallista seuraavilla tavoilla: asettamalla prosesseja hiekkalaatikko-mallin sisään, lataamalla kahta haitallisia sivustoja sisältävää listaa ja mahdollistamalla käyttäjän ajaa Incognito-tila. Chromessa on siis kaikki elementit kohdallaan, jotka voivat houkutella uusia käyttäjiä.

Viiden vuoden kuluttua tilanne voi olla internet-selaimen kannalta erilainen, tai se voi olla täysin samanlainen. Seuraavat kysymykset nousivat esille tutkielman kirjoittamisen aikana, mutta ne saavat vastauksensa vain ajan kanssa: tuleeko Chrome vastaamaan Firefoxin suosiota? Liittyvätkö kyseiset internet-selaimet yhteen ottaen huomioon Googlen rahoituksen Firefox-projektille? Jos Chrome saa hurjan suosion ensivuoden puolella, käykö Firefoxille kuten Netscapelle aikanaan IE:n kanssa vai hylkäävätkö käyttäjät ja kehittäjät Chromen kokonaan? Tutkielman valmistumishetkellä Chromen suosio alkaa kuitenkin hitaasti kasvaa (GMS 2008).

Kuva 10: Chrome lokakuussa



Kuva 11: Chrome marraskuussa

Lähteet

Anderson, N. (2008): *"Google on Chrome EULA controversy: our bad, we'll change it"*. (http://arstechnica.com/news.ars/post/20080903-google-on-chrome-eula-controversy-our-bad-well-change-it.html). Luettu 21.10.2008

Andy, J. (2008): *"A deeper look behind Google's shiny new Chrome"*. PalmAddict 7.9.2008. (http://palmaddict.typepad.com/palmaddicts/2008/09/a-deeper-look-b.html) Luettu 14.9.2008

Baig, E. (2008): *"Chrome kicks off with a good start"*. USA Today 3.9.2008

Berschewsky, T. (2008): *"Selain 2010 -luvulle > Google Chrome Beta"*. Mikrobitti 10/2008

CDD (Chronium Developer Documentation (2008)). (http://dev.chromium.org/). Luettu 18.9.2008

Fleischer, P. (2008): *"Another step to protect user privacy"*.. (http://googleblog.blogspot.com/2008/09/another-step-to-protect-user-privacy.html). Luettu 24.10.2008

Gal, A. Bebenita, M. Chang, M. Franz, M. (2006): *"Making the Compilation "Pipeline" Explicit: Dynamic Compilation Using Trace Tree Serialization"*. Computer Science Department, University of California

GMS (Global Marketshare Statistics (2008). (http://getclicky.com/global-marketshare-statistics). Luettu 14.10.2008

Graham, J. (2008): *"Chrome enters browser wars"*. USA Today 3.9.2008

Green, H. (2008): *"Google's Chrome Ups the Ante"*. Business Week Online 3.9.2008

Google (2.9.2008): *"Google Chrome"*. (http://www.google.com/chrome). Luettu 6.10.2008

Gustines, G (2008): *"Abstract from publisher"*. New York Times 8.9.2008

Lehto, T. (2.9.2008): *"Google julkaisee viimein web-selaimen"*. (http://www.tietokone.fi/uutta/uutinen.asp?news_id=34739). Luettu 3.9.2008

McCloud, S. (2008): *"Google Chrome Comic Book"*. (http://www.google.com/googlebooks/chrome/). Luettu 19.9.2008

Milstein, S. Biersdorfer, J. D. MacDonald M. (2006): *"Google the missing manual"*. California: Pogue Press/O'Reilly.

Monthie, B. (2008): *"What, who, when, where, why, how of XSS"*. Network World 7/21/2008, Vol. 25 Issue 28.

Mozilla Corporation. Luettu 15.10.2008
Rhino: (http://www.mozilla.org/rhino/)
Spidermonkey: (http://www.mozilla.org/js/spidermonkey/)
Tamarin: (http://www.mozilla.org/projects/tamarin/)

Mozilla Developer (2008). (http://developer.mozilla.org/En/About_JavaScript). Luettu 15.10.2008

PCMag (PCMag Encyclopedia 2008) (http://www.pcmag.com/encyclopedia_term/0,2542,t=virtual+machine&i=53927,00.asp). Luettu 29.10.2008

PCRE (2008): *"Perl Compatible Regular Expressions"*. (http://www.pcre.org/). Luettu 24.10.2008

Peltomäki, J. (2001): *"JavaScript"*. Docendo Finland, Sanoma WSOY konserni.

Powell, T. (2001): *JavaScript: The Complete Reference, Second Edition*. (http://www.devarticles.com/c/a/JavaScript-Security). Luettu 18.10.2008

Smith, J. & Nair, R. (2005): *"The Architecture of Virtual Machines"*. Computer 38 (5): 32–38. IEEE Computer Society

V8 JavaScript Engine (Google Code (2008)). (http://code.google.com/p/v8/). Luettu 17.9.2008

W3C (2008): *"World Wide Web Consortium"*. (http://www.w3.org/Graphics/SVG/). Luettu 24.10.2008

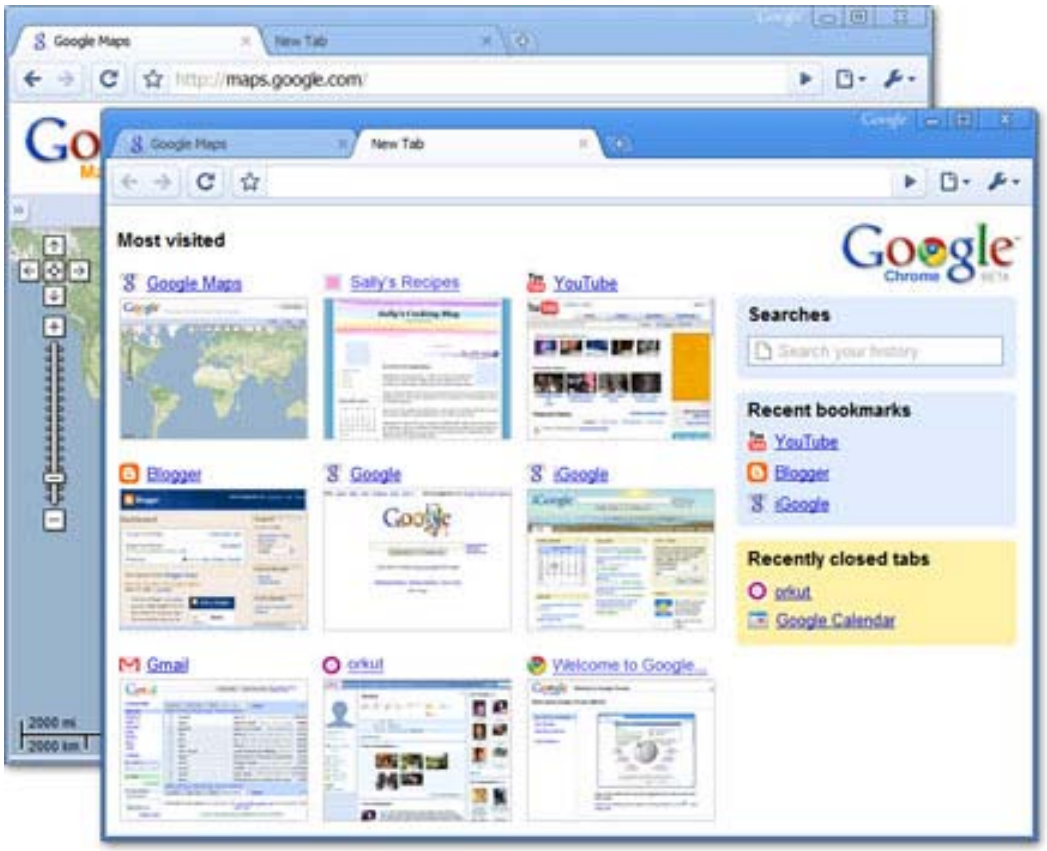
W3schools (2008). (http://www.w3schools.com/browsers/). Luettu 14.10.2008

WebKit (2008). (http://webkit.org/). Luettu 18.10.2008

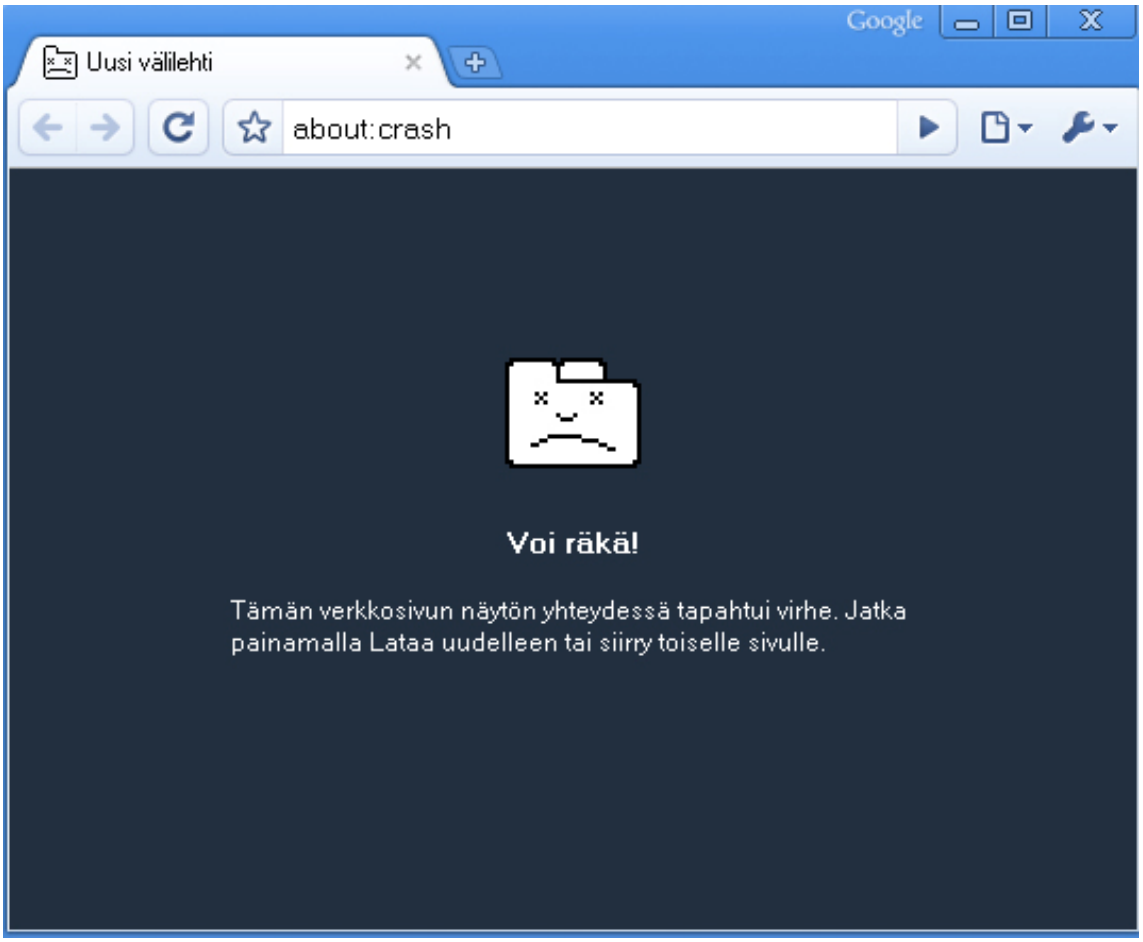
Wikipedia (18.9.2008): *"Google Chrome"*. (http://en.wikipedia.org/wiki/Google_Chrome). Luettu 18.9.2008

Tutkielmassa käytetyt testit:
Sun Spider: http://www2.webkit.org/perf/sunspider-0.9/sunspider.html
Acid Tests: http://www.acidtests.org/
V8 Benchmark: http://code.google.com/apis/v8/run.html

LIITE 1: Käyttöliittymä



LIITE 2: Välilehden kaatuminen



LIITE 3: Sivun estäminen