Harri Lappalainen

PELIOHJELMOINTI JAVASCRIPT-KIRJASTOLLA

Opinnäytetyö
Tietojenkäsittely

Marraskuu 2012

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

KUVAILULEHTI

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Opinnäytetyön päivämäärä** 28.11.2012
Tekijä(t) Harri Lappalainen	**Koulutusohjelma ja suuntautuminen** Tietojenkäsittely
Nimeke Peliohjelmointi JavaScript-kirjastolla	

Tiivistelmä

Tämän opinnäytetyön tarkoituksena on tutkia, miten erilaiset JavaScript-kirjastot soveltuvat peliohjelmoinnin haasteisiin. Opinnäytetyön teoriaosuudessa käyn läpi erilaisia JavaScript-peliohjelmointiin liittyviä tekniikoita ja JavaScript-kirjastoja. Esittelen myös vaihtoehtoisia peliohjelmointiin soveltuvia ohjelmointikieliä.

Käytännön toteutuksena tein minun ja ystäväni kehittämään ja ideoimaan Viking Tale -peliin hahmoikkunan, jossa pelaaja voi vaihtaa hahmolleen varusteita hiirellä vetämällä. Käytännön toteutuksen osuudessa vertailen hahmoikkunan toteutusta kahdella erilaisella JavaScript-kirjastolla, yleiskäyttöisellä jQuery UI -kirjastolla ja peliohjelmointiin tarkoitetulla Crafty-pelikirjastolla. Kummassakin toteutuksessa tarvitsin myös jQuery-kirjastoa hahmoikkunan Ajax-käsittelyyn. Käsittelen kummankin toteutuksen yhteydessä vastaan tulleita ongelmia ja kehittämiäni ratkaisumenetelmiä.

Hahmoikkunan toteutus onnistui kummallakin kirjastolla. jQuery UI -kirjaston avulla sain lopputuloksesta juuri sellaisen kun suunnittelin, kun taas Crafty-pelikirjastolla jouduin tyytymään kompromisseihin. Opinnäytetyössäni päädyin sellaiseen lopputulokseen, että yleiskäyttöisetkin JavaScript-kirjastot, kuten jQuery, soveltuvat peliohjelmointiin. Kirjaston soveltuminen riippuu kuitenkin pelin tyypistä. Crafty-pelikirjasto soveltuisi luultavasti paremmin esimerkiksi 2d-räiskintäpelin toteutukseen.

Asiasanat (avainsanat) JavaScript, peliohjelmointi, jQuery		
Sivumäärä 58	**Kieli** Suomi	**URN**
Huomautus (huomautukset liitteistä)		
Ohjaavan opettajan nimi Janne Turunen	**Opinnäytetyön toimeksiantaja** Mikkelin ammattikorkeakoulu	

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

Date of the master's thesis

28th of November 2012

Author(s)
Harri Lappalainen

Degree programme and option
Business Information Technology

Name of the master's thesis

Game development with JavaScript library

Abstract

The purpose of this bachelor's thesis was to study how different JavaScript libraries suited for the challenges of game development. The theoretical part of this thesis presented different techniques involved in Javascript game development and JavaScript libraries. Theoretical part also introduced alternative programming languages suited for game development.

As the practical part of my thesis I made a character window for a role playing game called Viking Tale which I developed together with my friend. In Viking Tale the player could change his character's gear by dragging with the mouse. The practical part of this thesis compared the development of character windows with two different JavaScript libraries, universally practical jQuery UI and Crafty specializing in game development. I also needed the jQuery library for Ajax technique in both of these development cases. In both cases I introduced the problems faced and the solutions I came up with.

The development of character window was successful with both libraries. The outcome was exactly as planned with the jQuery UI library, but with the game library Crafty I had to settle for compromises. The conclusion was that even universally practical JavaScript libraries such as jQuery were suited for game development. The suitability of the library depends on the game's type though. The game library Crafty might probably be better, for example, for the development of 2d shooting games.

Subject headings, (keywords)

JavaScript, game development, jQuery

Pages	Language	URN
58	Finnish	

Remarks, notes on appendices

Tutor	Master's thesis assigned by
Janne Turunen	Mikkeli University of Applied Sciences

SISÄLTÖ

1 JOHDANTO

Tämän opinnäytetyön tarkoitus on tutkia, miten erilaiset JavaScript-kirjastot soveltuvat peliohjelmointiin. Käyn myös läpi JavaScript-peliohjelmointiin liittyviä tekniikoita ja hieman eri vaihtoehtojakin selainpelien ohjelmointiin. Käytännön osuuden käyttötapauksen toteutan kahdella erilaisella JavaScript-kirjastolla ja vertailen lopputuloksia toisiinsa.

Tämän opinnäytetyön toisessa luvussa käyn läpi JavaScript-peliohjelmointiin yleisimmin liittyviä tekniikoita. Käyn läpi HTML5- ja CSS3-kieliä ja kertaan hieman niiden historian vaiheita. Selitän molempien uusista ominaisuuksista, jotka ovat olennaisia JavaScript-peliohjelmoinnin kannalta. Esittelen myös pikaisesti pari vaihtoehtoista valintaa selainpelien ohjelmointiin; Flash-ohjelmoinnin omassa luvussaan ja WebGL:n HTML5-luvussa. Luonnollisesti käyn läpi JavaScript-ohjelmointikielen ja myös siihen liittyvän Ajax-tekniikan, jota käytän opinnäytetyöni käytännön käyttötapauksessa.

Opinnäytetyöni kolmannessa luvussa käyn läpi erilaisia JavaScript-kirjastoja. Ensin esittelen mielestäni käytetyimpiä yleiskäyttöisiä JavaScript-kirjastoja, kuten jQuery- ja MooTools-kirjastot. Nämä kirjastot ovat monipuolisia ohjelmointityökaluja, osa keskittyen joidenkin ominaisuuksien kehittämiseen enemmän kuin toiset. Kolmannessa luvussa esittelen myös peliohjelmointia varta vasten suunniteltuja JavaScript-kirjastoja, jotka käyttävät myös HTML5:n ja CSS3:n uusia ominaisuuksia hyödykseen. Kolmannen luvun lopuksi kerron muista JavaScript-kirjastoista, joista voi olla hyötyä peliohjelmoinnissa. Nämä kirjastot ovat suunniteltu vain tiettyjen ominaisuuksien kehittämistä varten, esimerkiksi pelkästään fysiikan mallinnukseen tarkoitettu Physijs-rajapinta.

Neljännessä luvussa kerron käytännön toteutuksen eri vaiheista. Opinnäytetyössäni käsiteltävä käyttötapaus on hahmoikkuna minun ja opiskelutoverini tekemään peliin. Pelille ei ole virallista asettajaa, vaan halusimme vain kokeilla pelin tekemistä. Ensin esittelen jQuery-toteutuksen käsittelemästäni käyttötapauksesta. Sen jälkeen kerron Crafty-pelikirjastolla tekemästäni toteutuksesta. Neljännessä luvussa käyn läpi kummankin toteutuksen ongelmakohtia ja kuinka ratkaisin ne.

Päätännössä kerron, millaiseen tulokseen opinnäytetyössäni päädyin. Pohdin käyttämieni toteutustapojen hyviä ja huonoja puolia. Lopuksi mietin mahdollisia jatkokehitystoimenpiteitä käsittelemälleni käyttötapaukselle.

2 TEKNIIKAT

Tässä luvussa käsittelen JavaScript-peliohjelmoinnin kannalta tärkeimpiä tekniikoita ja myös vaihtoehtoisia valintoja peliohjelmointiin, kuten Flash-ohjelmoinnin ja WebGL:n. HTML5:n ja CSS3:n yleistymisen myötä JavaScript-peliohjelmointi on nostanut päätään väkevämmin esiin ja vanha tuttu Flash on joutunut hiipumaan taka-alalle. HTML5-merkintäkieleen on sisäänrakennettuna helposti käyttöön otettavia multimediaelementtejä ja CSS3 mahdollistaa entisiä versioita hienovaraisemman ja paremman animoinnin ja elementtien käsittelyn.

2.1 HTML5

HyperText Markup Language (HTML) on standardiksi muodostunut merkintäkieli verkkosivujen sisällön esittämistä varten. HTML-rakenne koostuu elementeistä, joita laitetaan peräkkäin ja sisäkkäin. Esimerkiksi <head> aloittaa header-elementin, jonka sisään voi laittaa sivun otsikon <title>Esimerkki</title>. Header lopetetaan </head> elementillä. Esitettävä teksti sijoitetaan elementtien sisään. (Raggett 2005.)

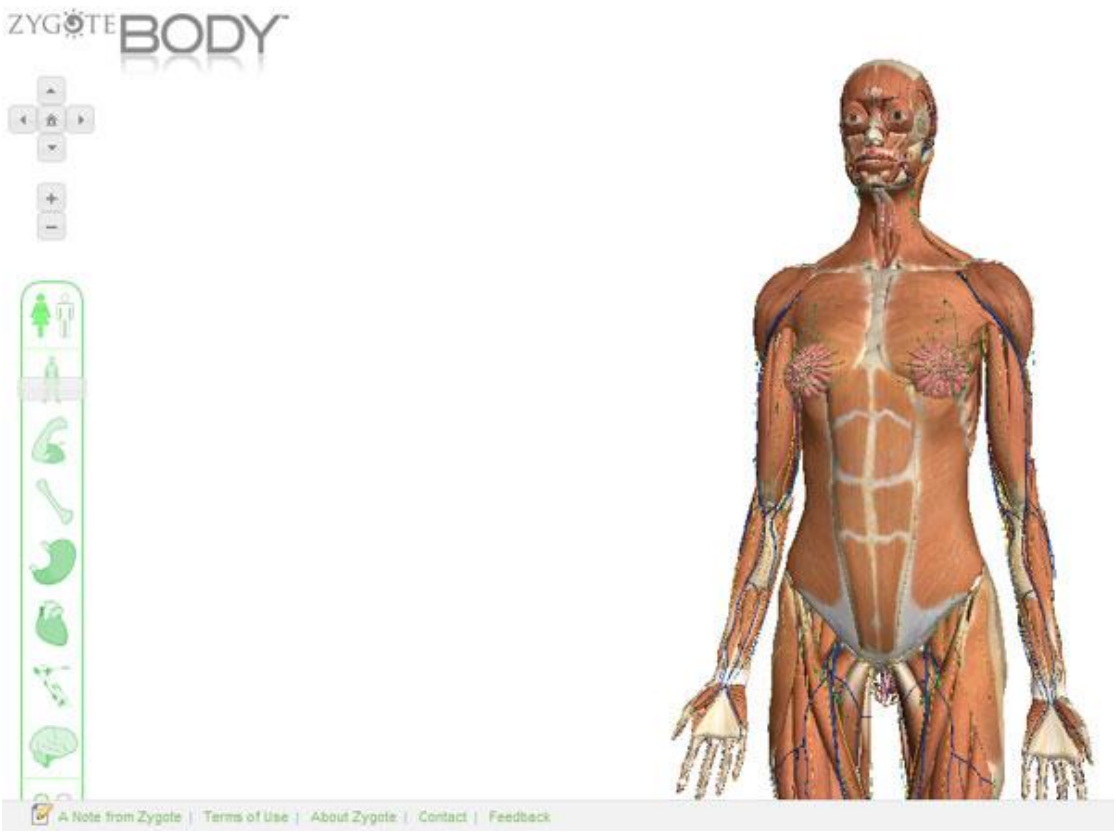
HTML kehitettiin 1990-luvun alussa, jotta käyttäjillä olisi mahdollisuus esittää Internet-dokumenttien sisältö erilaisilla tavoilla, esimerkiksi tietty teksti lihavoituna tai kursivoituna (Schafer 2010, 6). HTML-kieli muuttui jatkuvasti selainten tekijöiden mielihalujen mukaan. HTML ei ollut virallisesti määritelty ennen kuin vuonna 1995, jolloin määriteltiin HTML 2.0-versio. Vuonna 1997 tuli HTML 4.0, jonka jälkeen HTML-merkintäkielen kehitys taantui. HTML 4 -versiolle kehitettiin vielä päivitys 4.01, kunnes Applen, Mozillan ja Operan perustama WHATWG-yhteisö (Web Hypertext Application Technology Working Group) alkoi kehittää uutta versiota HTML-merkintäkielestä vuosina 2004 – 2006. Kehitys tiivistyi WHATWG:n ja W3C:n yhteistyön tuloksena vuodesta 2007 alkaen. HTML5 on ollut luonnosvaiheessa siitä hetkestä tähän hetkeen asti. (Korpela 2011, 24–25.)

Tässä luvussa käsittelen HTML5:n tärkeimpiä ominaisuuksia peliohjelmointia ajatellen. Yksi suurimmista HTML5:n uudistuksista on canvas-elementti, eli piirtoalue. Piirtoalueelle on mahdollista luoda kuvia dynaamisesti lennosta JavaScriptin avulla. Käytännössä tämä mahdollistaa muun muassa animaatioiden, kaavioiden ja diagrammien luomisen. (Cruse ym. 2011, 144.) Pilgrimin mukaan kaikki suurimmat selaimet tukevat tällä hetkellä canvas-elementtiä, kuten tarkemmin näkyy taulukosta 1. Internet Explorerin versiot 7 ja 8 toimivat Pilgrimin mukaan kolmannen osapuolen explorer-canvas-kirjaston avulla. Koska aivan kaikki selainten eri versiot eivät tue piirtoaluetta, ei sen varaan kannata turvautua liialti (Cruse ym. 2011, 144).

TAULUKKO 1. Canvas tuki (Dive into HTML5 2010)

Internet Explorer	Firefox	Safari	Chrome	Opera	iPhone	Android
v. 9.0+	v. 3.5+	v. 3.0+	v. 3.0+	v. 10.0+	v. 1.0+	v. 1.0+

Canvas-piirtoaluetta kutsutaan JavaScriptissä sille määritellyn id-attribuutin perusteella. JavaScriptissä tapahtuu varsinainen piirtely. JavaScriptissä piirtoalueelle määritellään konteksti, yleensä 2d. (HTML5 Tutorial 2012.) Tämä herättää luonnollisesti kysymyksen, onko 3d-konteksti mahdollinen myös, mutta tällä hetkellä siitä ei ole olemassa mitään standardisoitua versiota HTML5-kielelle. Yksityiset henkilöt ovat tehneet vain erilaisia kokeiluja. HTML5-määrittelyssä todetaan vain epämääräisesti, että 3d-konteksti on ehkä mahdollista joskus tulevaisuudessa. (Dive into HTML5 2010.) Piirtoalueelle voi määrittää myös WebGL-kontekstin. Piirtoalueella pelataan paljon x- ja y-koordinaateilla, jotta saataisiin aikaan haluttuja piirroksia (Korpela 2011, 200–201). WebGL tarjoaa selaimelle käyttöön 3d-grafiikka API-rajapinnan, joka on OpenGL ES 2.0 -standardin mukainen. WebGL:llä voi siis pyörittää 3d-pelejä HTML5:n avulla, mutta kaikki selaimet eivät tue sitä. (Makzan 2011, 9.) Kuvasta 1 näkyy WebGL:llä tehty 3d-ihmisvartalo. Mallia voi pyöritellä selaimessa ja määrittää, kuinka yksityiskohtaisesti ihmisvartaloa haluaa tarkastella. Vartaloa voi katsella pintapuolisesti kuin ketä tahansa muuta ihmistä, tai rajata näkymä luihin.



KUVA 1. WebGL:llä tehty ihmisvartalo (Techpack 2010)

Suuria HTML5:n mullistuksia ovat myös audio- ja videoelementtien käyttöönotto. W3Schoolsin (2012) mukaan video- ja audioelementeille voi määritellä myös muita erinäisiä ominaisuuksia:

- *controls*; kontrollit soittamiselle ja äänenvoimakkuuden säädölle
- *autobuffer*; Boolean arvo, joka kertoo selaimelle audion/videon lataamisen ennen kuin käyttäjä painaa play-nappia
- *autoplay*; selain alkaa soittaa tiedostoa automaattisesti
- *loop*; Boolean arvo, joka määrää selaimen pyörittämään tiedostoa uudestaan sen päätyttyä
- *preload*; tiedoston lataus käynnistyy ennen soittamista
- *poster*; staattinen kuva, joka on esillä videon latauksen ajan (käy luonnollisesti vain video elementille)
- *muted*; videon ääniraita on vaimennettu soiton aluksi.

Käytännössä kaikki suurimmat selaimet tukevat audio- ja videoelementtejä. Selainten videoelementtien tuki tällä hetkellä näkyy taulukosta 2. Taulukosta 2 ilmenee, että muut selaimet, paitsi Internet Explorer ja Opera, tukevat hyvinkin laajalti videoelementin käyttöönottoa.

TAULUKKO 2. Selainten videoelementin tuki (Dive into HTML5 2010)

Selain	**Internet Explorer**	**Firefox**	**Safari**	**Chrome**	**Opera**	**iPhone**	**Android**
Versio	9.0+	3.5+	3.0+	3.0+	10.5+	1.0+	2.0+

Taulukosta 3 tulee ilmi eri selainten tukemat videoformaatit. Koska kaikki suurimmat selaimet eivät tue samoja tiedostotyyppejä, formaattierot täytyy ottaa huomioon videoelementin käyttöönotossa lisäämällä eri tiedostotyyppejä samasta videosta. (Korpela 2011, 186.) Videoformaattituet ovat taulukon 3 mukaiset tällä hetkellä, mutta ne voivat hyvinkin vielä muuttua.

TAULUKKO 3. Selaimet ja videoformaatit (HTML5 Tutorial 2012)

Selain/Formaatti	**MP4**	**WebM**	**Ogg**
Internet Explorer	kyllä	ei	ei
Firefox	ei	kyllä	kyllä
Chrome	kyllä	kyllä	kyllä
Safari	kyllä	ei	ei
Opera	ei	kyllä	kyllä

Eri selaimilla on myös eri formaattituet audion suhteen. Audioelementille voi määritellä videon tapaan eri tiedostomuotoja lähteeksi ja myös tekstitiedotuksen (HTML5 Tutorial 2012). Taulukosta 4 näkyy selainten audioformaattituet. Kuten videoelementtien tuet, nämäkin voivat muuttua tulevaisuudessa.

TAULUKKO 4. Selaimet ja audioformaatit (W3Schools 2012)

Selain/Formaatti	**MP3**	**Wav**	**Ogg**
Internet Explorer	kyllä	ei	ei
Firefox	ei	kyllä	kyllä
Chrome	kyllä	kyllä	kyllä
Safari	kyllä	kyllä	ei
Opera	ei	kyllä	kyllä

Muita peliohjelmoinnin kannalta hyötyisiä elementtejä HTML5-merkintäkielessä ovat GeoLocation (paikannus), paikallinen tiedonsäilöntä ja mahdollisuus HTML5-

sovelluksille ilman Internet-yhteyttä. GeoLocation ominaisuuden ansiosta verkkosivu noutaa käyttäjän laitteen maantieteellisen sijainnin, leveys- ja pituusasteen. HTML5:n datavarasto säilöö avaimella ja arvolla sidottua tietoa, joka pysyy tallessa vaikka selain suljetaan. Tieto on saatavilla myös muille saman palvelimen selaimille. Paikallisen datavaraston ansiosta pystyy tallentamaan esimerkiksi pelitilanteen selaimeen. HTML5 mahdollistaa myös Web SQL-tietokannan, joka on asiakaspään relaatiotietokanta. Web SQL-tietokantaan voi tallentaa monimutkaisempaakin tietoa kuin pelkkään paikalliseen datavarastoon. Yhteydettömät HTML5-sovellukset ovat mahdollisia välimuistimanifestin ansiosta. Välimuistimanifestiin voi tallentaa lokaalisti peligrafiikat, JavaScript, CSS tyyli- ja HTML-tiedostot. Tämän avulla HTML5 pelit voi tallentaa vaikkapa tietokoneelle tai mobiililaitteelle. (Makzan 2011, 8–11.)

2.2 CSS 3

CSS (Cascading Styling Sheets) on kieli verkkosivujen ulkoasun määrittelemiseksi. Sillä voi määritellä muun muassa värejä, sivujen taittoa ja fontteja. CSS:ää voi käyttää minkä tahansa XML-pohjaisen merkkikielen kanssa. CSS-tyylejä voi määritellä suoraan HTML-dokumentin sisällä tai vaihtoehtoisesti erillisessä tiedostossa, jota HTML-tiedosto kutsuu. (HTML & CSS 2012.)

1990-luvun puolivälin paikkeilla Internetin suosio kasvoi räjähdysmäisesti. Tuolloin HTML oli ainoa tapa esitellä verkkosivua ja käyttäjät kaipasivat kipeästi parempaa kontrollia tyylien määrittelemiseen. Virallista standardia tyylitiedostoille ei kuitenkaan ollut luotu. Håkon Wium Lie kirjoitti CSS:n ensimmäisen luonnoksen vasta vuonna 1994. Bert Bos työskenteli samoihin aikoihin oman luonnoksensa parissa, mutta Håkonin luonnoksen ilmestyttyä hän yhdisti voimansa hänen kanssaan, ja tuloksena oli Cascading Styling Sheets. CSS sai kuitenkin aluksi ristiriitaisia vastaanottoja. (York 2005, 13.) Vuonna 1995 Web-konsortio W3C aloitti CSS:n määrittelyn ja seuraavana vuonna he julkaisivat CSS 1 -suosituksen. Toteutus kuitenkin eteni hitaasti ja selaimet siirtyvät sangen hitaasti tukemaan CSS:n ominaisuuksia. CSS 2 -suositus julkistettiin 1998 ja selainten CSS-tuki alkoi hiljakseen parantua, mutta ei ollut vielä kuitenkaan kelvollistakaan. W3C kehitti seuraavaksi eräänlaisen realistisen version, CSS 2.1, joka kattoi CSS 2:n selaimissa parhaiten tuetut ominaisuudet. CSS 3 versio on ollut kehitteillä jo pitkään, mutta työ on edistynyt hitaasti. (Korpela 2008, 54–56.)

CSS 3 on moduuleihin jaettu, eli ominaisuudet eivät ole riippuvaisia toisistaan. Nämä moduulit on tarkoitettu ottaa käyttöön itsenäisesti ja toisista erillään. Tämä tarkoittaa sitä, että toisia ominaisuuksia voi määritellä ja toteuttaa eteenpäin riippumatta siitä, etenevätkö muutkin ominaisuudet. Käytännössä CSS 3 on siis kokoelma eri vaiheessa olevia luonnoksia. (Cederholm 2010, 2–3.) Tässä kappaleessa käsittelen CSS 3:n tärkeimpiä ominaisuuksia JavaScript-peliohjelmointia ajatellen. CSS 3:n transform (muodonmuutos) komennon avulla voi käännellä ja muutella elementin kokoa tai asentoa. Muodonmuutos komennolla on mahdollista muutella 2d- ja 3d-muodoissa. (Gasston 2011, 163.) Isoimmat selaimet tukevat transform-komentoa, mutta jokainen vaatii oman etuliitteensä, esimerkiksi Firefox tarvitsee -moz- etuliitteen ja Opera -o- etuliitteen. 2d-muodonmuutosta tukee jokainen suurselain, mutta 3d-tuki on vain Chromella, Safarilla ja Firefoxilla. (CSS3 transform property 2012.) Transition (siirtymä) ja animation -komennoilla voi tuoda animaatiota elementeille. Kaikki isoimmat selaimet, paitsi Internet Explorer tukevat näitä komentoja. Transition- ja animation-komennot eroavat sillä, että transition alkaa vaikuttaa vasta, kun siihen liitetyn elementin tila (value) muuttuu, kun taas animation vaikuttaa heti, kun se liitetään johonkin elementtiin. Esimerkkinä tilanne, että halutaan muuttaa elementin leveyttä, kun kursori liikkuu sen päälle. CSS 2:lla leveyden muutos on välitön, mutta transition-siirtymän avulla leveys muuttuu sujuvasti ja tasaisesti. Siirtymälle voi asettaa muun muassa ajan, minkä verran siirtymän halutaan kestävän, viiveen siirtymän alkamiselle ja nopeuksien vaihteluita siirtymälle. (Gasston 2011, 164–168.) W3Schoolsin CSS3 animation propertyn mukaan animaatiolle voi määritellä samoja ominaisuuksia kuin siirtymälle, mutta sille voi määritellä myös animaation tyylin, animaation pyörimiskerrat ja pyöriikö animaatio väärinpäin joka toisella kerralla.

2.3 Flash

Flash on työkalu rikkaiden mediasovellusten luomiseen, joihin voi lisätä vuorovaikutteisuuksia, eli esimerkiksi pelejä ja animaatioita. Flash-ohjelmoinnin hyötyjä ovat pienet tiedostokoot, jotka ovat vektorigrafiikkojen ansiota. Vektorigrafiikat käyttävät huomattavasti vähemmän muistia kuin bittikarttagrafiikat, koska vektorigrafiikat esitetään matemaattisilla kaavoilla eikä isoina datakokoelmina. (Prayaga & Suri 2008, 1–2.) Thomasin mukaan (2010) Flash-kielen vahvuuksiin lukeutuu sen helppo saatavuus, 97 % tietokoneista selaimilla tukevat Flash-sovelluksia, sillä on helppoa tehdä pelejä ja se on tunnettu suurelle määrälle ohjelmoijia. Heikkouksiin Thomas (2010) laskee

Flashin toimimattomuuden useilla Applen mobiililaitteilla, lukuisat tietoturvaongelmat ja Flash-kehitystyökalujen maksullisuuden.

Flash-ohjelmoinnin juuret ovat 1993 perustetussa FutureWave Softwaressa, jonka Jonathan Gay perusti vuonna 1993 Charlie Jacksonin rahallisen tuen avustamana. FutureWave perustettiin dominoimaan grafiikkasovellusmarkkinoita elektronisille kynille, mikä oli iso juttu siihen aikaan ja siitä odotettiin jymymenestystä. Yritys nimeltä Go rakenteli käyttöjärjestelmää elektronisia kyniä varten ja FutureWave alkoi kehittää SmartSketch-sovellusta elektronisilla kynillä piirtelyyn tietokoneella. Go-yrityksellä ei kuitenkaan mennyt hyvin ja 1994 sen tarina loppui. (Gay 2001.) Vuonna 1995 FutureWave sai paljon palautetta, että heidän pitäisi kääntää SmartSketch animaatiosovellukseksi. Internetistä oli tulossa suosittu ilmiö samoihin aikoihin, joten FutureWave päätti alkaa kehittää animaatiosovellusta, jonka tuotoksia voisi esittää Internetissä, sekä verkkosoitinta animaatioille. SmartSketch Animatorin nimi muuttui FutureSplash Animatoriksi, koska sovelluksella ei ollut enää niinkään merkitystä piirtelyyn. FutureWave yritti myydä sovellustaan isommille yrityksille suosion saamiseksi, mutta muun muassa Adobe ja Fractal Design kieltäytyivät. Vuonna 1996 FutureSplash sai kuitenkin kaksi isoa asiakasta, Microsoftin ja Disney Onlinen. Samana vuonna Macromedia osti FutureWaven ja FutureSplash Animatorista tuli Macromedia Flash 1.0. (Waldron 2000.) Wikipedian (2012) mukaan vuonna 2005 Adobe osti Macromedian, josta lähtien Flashia on kehitetty luonnollisesti Adoben alaisuudessa. Vuonna 2011 Adobe ilmoitti, että Flash Playerin jatkokehitys mobiililaitteilla käytettävissä selaimissa lopetetaan ja kehitys keskittyy Adobe AIR:iin, jolla käyttäjät voivat luoda sovelluksia mobiililaitteille. Adobe aikoo keskittyä myös Flashin ja HTML5:n yhteistyöhön tietokoneselaimilla. (Winokur 2011.)

2.4 Javascript

JavaScript on monipuolinen ohjelmointikieli, joka toimii yhteistyössä selaimen kanssa. Koodirivit käännetään selaimessa sitä mukaa, kun ne tulevat vastaan (Moncur 2001, 4). JavaScript-kielellä voi muuttaa staattisen verkkosivun sisällön interaktiiviseksi ja mielenkiintoisemmaksi kokemukseksi. Esimerkiksi verkkosivu voi tervehtiä käyttäjää viestillä ”Oikein hyvää iltaa”, jos on ilta käyttäjän aikavyöhykkeellä siitäkin huolimatta, että sivuston palvelimen aikavyöhykkeellä olisi aamuaika. JavaScript

mahdollistaa myös yksinkertaisia, mutta tyylikkäitä perusanimaatioita, esimerkiksi itsestään kuvia vaihtelevan diaesityksen. (Goodman ym. 2010, 3.)

Goodmanin ym. mukaan (Goodman ym. 2010, 10) alkujaan JavaScript-ohjelmointikielen (alkuun käytettiin nimitystä LiveScript) tarkoitus oli lisätä pieniä parannuksia verkkosivuille, lähinnä lomakkeisiin, sekä antaa palvelimien ylläpitäjille mahdollisuus yhdistää verkkosivujaan toisiin palveluihin, kuten liittää hakukoneita palvelinpään tietokantoihin. Silloisen LiveScriptin kehittäjät Netscape ja Sun Microsystems olivat sitä mieltä, että LiveScript-kieltä ei pitäisi käyttää suoraan HTML-syntaksissa, vaan erillisenä skriptauskielenä. Ensimmäinen virallinen JavaScript-versio ilmestyi vuonna 1996 Netscape 2.0 -selaimen yhteydessä, jolloin nimi vaihtui LiveScriptistä JavaScriptiksi. (Horn 2009, 2.) JavaScript muistutti hieman Java-ohjelmointikieltä jo ennen uutta nimeäänkin. Moni JavaScript-kielen perussyntakseista muistutti Javan vanhaa tyyliä. JavaScriptin tarkoitus käyttäjäpuolella on kuitenkin hyvin erilainen. Se on integroitu HTML-dokumentteihin, toisin kuin Java, jolla kirjoitetaan sovelmia (applets), jotka vievät tietyn tilan sivulta eivätkä toimi muiden sivulla olevien elementtien kanssa. JavaScript on myös kevyempi ohjelmointikieli, sen ”sanavarasto” ei ole niin iso ja sen ohjelmointimalli on helpommin sulateltavissa. (Goodman ym. 2010, 10.)

JavaScriptin julkistamisen jälkeen Microsoft julkaisi Internet Explorer 3.0 -version, joka sisälsi vastaavanlaisen skriptauskielen, JScriptin. Seuraavien vuosien aikana Netscape ja Microsoft kehittivät kilpaa omia skriptauskieliään, yrittäen saada yliotetta toisesta. Vuonna 1997 Ecma, kansainvälinen standardilaitos, hyväksyi Netscapen lähettämän hakemuksen JavaScript-kielen standardoimiseksi. Tämä standardi tunnetaan nimellä ECMAScript (ECMA-262), ja sitä on kerrattu neljästi 1997 – 2009 välisenä aikana. Tavallaan siis ECMAScript on virallinen ohjelmointikieli, josta JavaScript ja JScript ovat eräänlaisia murteita. Vuonna 2007 ECMAScriptistä julkaistiin ehdotus ES4 (Edition 4) versiosta, joka jakoi valtaisasti mielipiteitä. Ehdotus sisälsi radikaalisia muutoksia ominaisuuksia, kuten luokkapohjaista perintää, nimiavaruuksia (namespaces) ja iteraattoreita. Tämä ehdotus ei juurikaan kerännyt kannatusta ja täten joukko Microsoftin ja Yahoo!:n työntekijöitä ryhtyi kehittämään vaatimattomampaa uudistusta, ES3.1, joka myöhemmin nimettiin ES5:ksi. Monien mielestä tämä uudistus on korjannut vain pikkuvikoja. ES4:n kannattajat ja vastustajat päättivät kuitenkin yhdistää voimansa asettaen vaatimattomammat tavoitteet uudelle versiolle, jotta tehokas yhteis-

työ olisi mahdollista. Vuonna 2009 he julkaisivat luonnoksen työnsä hedelmästä, ECMAScript Harmonysta, joka sisälsi hyvin vähän alkuperäisestä ES4 ehdotuksesta. Luultavimmin menee kuitenkin vuosia, ennen kuin selaimet tukevat version muutoksia kunnolla. (Horn 2009, 2–3.)

JavaScript-ohjelmointikielellä on omat rajoituksensa ja hyvä ohjelmoija tunnistaa ne ja käyttää JavaScriptiä ratkaisuna vain niissä tapauksissa, joihin se soveltuu parhaiten. JavaScript-ohjelmointiin kannattaa turvautua, jos halutaan verkkopalvelun reagoivan välittömästi käyttäjän valitsemiin elementteihin (kuten lomakkeisiin tai linkkeihin), tietokannan tarjoaman datan esittäminen käyttäjäystävällisessä muodossa, useiden navigaatioiden, lisäosien (plugin) sekä Java-sovelmien (Java applets) hallitseminen, lähtevän datan esikäsittely ennen palvelimelle lähettämistä, sisällön ja tyylien muokkaaminen dynaamisesti ja tiedostojen sekä luku- ja kirjoituspyyntöjen lähettäminen palvelimelle. (Goodman ym. 2010, 13.)

JavaScript-kielelle on tarkoituksella asetettu rajoituksia käyttäjien yksityisyyden suojaamiseksi, eikä seuraavia toimintoja voi suorittaa ellei käyttäjä nimenomaan salli pääsyä tietokoneensa suojattuihin osiin; selaimen ominaisuuksien asettaminen tai noutaminen, selainikkunan muutteleminen, tulostaminen, tietokoneelle asennetun sovelluksen käynnistäminen, tiedostojen tai kansioiden lukeminen tai kirjoittaminen (poikkeuksena evästeet), palvelimen tiedostoille kirjoittaminen, palvelimen livelähetyksien kaappaaminen tai sähköpostien lähettäminen salaa. (Goodman ym. 2010, 13.)

2.5 Ajax

Ajax tulee lyhenteestä Asynchronous JavaScript and XML, mikä tarkoittaa asynkronista, tahdistamatonta JavaScript- ja XML-käsittelyä. Käytännössä tämä tarkoittaa tekniikkaa, jolla voi tehdä verkkopalvelun taustalla pyyntöjä palvelimelle JavaScript-kielen avulla ilman koko verkkosivun latautumista. Ajax-tekniikalla voi siis esimerkiksi hakea uutta tietoa tai päivittää sitä päivittämättä koko verkkosivua. (Rutter 2010, 214.) . Ajax lähti liikkeelle käsitteenä vuonna 2005 Jesse James Garrettin julkaisemasta artikkelista nimeltään ”Ajax: A new Approach to Web Applications” (Ford 2008, 27.) Wikipedian (2012) mukaan, asynkronisia kutsuja oli ollut jo vuodesta 1996 lähtien, mutta Ajax käsitteenä vakiintui vasta paljon myöhemmin.

Ajax ei kuitenkaan ole ohjelmointikieli, vaan tekniikka, joka hyödyntää eri ohjelmointitekniikoita. Näihin tekniikkoihin kuuluu tietysti JavaScript ja XML, mutta myös HTML ja CSS. Ajax käyttää myös XMLHttpRequest-objektia, jonka avulla dynaaminen datan käsittely selaimen ja palvelimen välillä onnistuu. (Ford 2008, 27.) XMLHttpRequest-objektia tarvitaan tiedon välitykseen ja ymmärtämiseen selain- ja palvelinpään välillä. XMLHttpRequest-objektilla selainpään skripti voi lähettää palvelimelle datapareja, jotka koostuvat nimestä ja arvosta. Palvelin lähettää HTTP-tekniikan välityksellä vastauksen, joka on jäsennettävissä selainpään JavaScript-koodin avulla. Vastaus voi olla yksinkertaisessa tekstimuodossa, mutta on yleistä käyttää joko XML- tai JSON-formaatteja (JavaScript Object Notation). (Brinzarea-Iamandi ym. 2009, 17.)

Ajax on käytännöllinen tekniikka, jolla voi parantaa käyttäjäkokemusta verkkosivulla, mutta on tärkeää tietää, milloin sitä on viisasta käyttää. Ajax-tekniikan vahvuuksiin kuuluvat muun muassa interaktiivisten verkkopalvelujen mahdollistaminen ja ennestään tunnettujen ja laajasti tuettujen standardien hyödyntäminen. Ongelmiakin Ajax-tekniikan käyttämisestä löytyy. Jos käyttäjä on asettanut JavaScriptin pois päältä selaimessaan, Ajax ei toimi laisinkaan. Ajax-tekniikan hyödyntämisessä on myös se ongelma, että sen avulla haettua dataa ei oteta huomioon hakukoneissa, koska hakukoneet eivät aja mitään JavaScript-koodia. Yleensä Ajax-sovellukset ajetaan saman URL:n alla, mikä tekee kirjanmerkkien asettamisesta hankalaa. Esimerkiksi kirjanmerkin asettaminen Ajax-tekniikan avulla luodulle alasivulle tekee pelkästään palvelun pääsivusta kirjanmerkin. Jotta kirjanmerkit toimisivat oikein, JavaScript-koodiin täytyy lisätä sivuankkureita. Selaimen paluu edelliselle tai seuraavalle sivulle nappien käyttö ei myöskään toimi normaalisti, jollei Ajax-palvelua ole ohjelmoitu tukemaan lataus- ja tallennustiloja. Huolimaton Ajax-tekniikan käyttöönotto voi siis helposti heikentää verkkopalvelun tehokkuutta ja käytettävyyttä. (Brinzarea-Iamandi ym. 2009, 18.)

3 JAVASCRIPT-KIRJASTOT

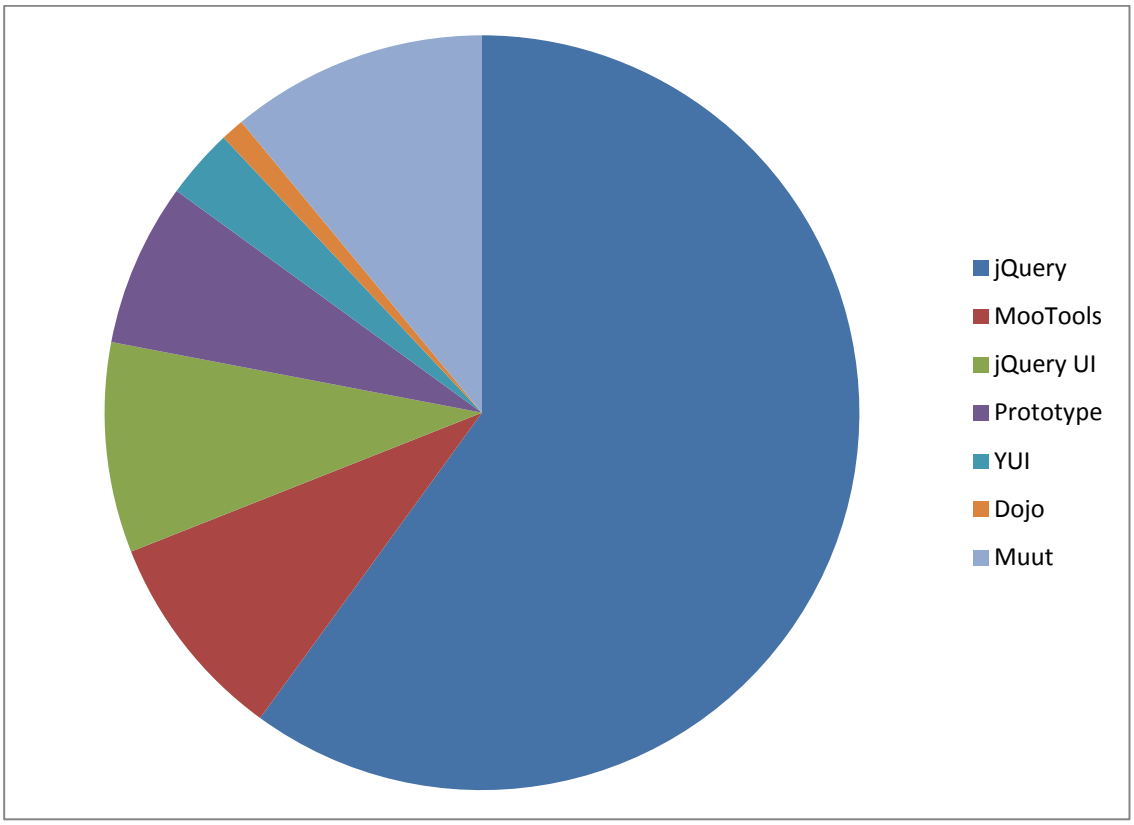
JavaScriptin kasvavan suosion myötä on kehitetty paljon erilaisia kirjastoja (frameworks), jotka helpottavat JavaScript-kehittäjän työtä. Esimerkiksi erilaiset funktiot ja toiminnallisuudet, kuten Ajax, ovat monesti JavaScript-kirjastoilla yksinkertaisempia ja nopeampia toteuttaa. Kirjastot helpottavat myös suuresti DOM-rakenteen (Docu-

ment Object Model) käsittelyä. Rutter kertoo (2010, 6), että DOM tarkoittaa verkkosivuston koko HTML-koodia, jonka rakenne muistuttaa puuta; jokainen oksa on sidottu toisiinsa hierarkkisesti.

Vaikka JavaScript-kirjastot perustuvat samalle ohjelmointikielelle, niillä voi olla hyvinkin erilaisia ratkaisumalleja samoihin kehitysongelmiin. Lisäksi jotkin kirjastot keskittyvät eri ominaisuuksiin enemmän kuin toiset. Tässä luvussa käyn läpi yleiskäyttöisiä kirjastoja, pelikirjastoja sekä tiettyjen pelillisten ominaisuuksien kehittämiseen erikoistuneita kirjastoja.

3.1 Yleiskäyttöiset JavaScript-kirjastot

Tällä hetkellä käytetyin JavaScript-kirjasto on jQuery, kuten kuva 2 osoittaa. Kuvan 2 data on kerätty Wappalyzer-dokumentaatiosta. Wappalyzer on ilmainen selaimen lisäosa, joka pystyy havainnoimaan, mitä teknologioita selaimella vierailluilla verkkosivustoilla on käytössä.



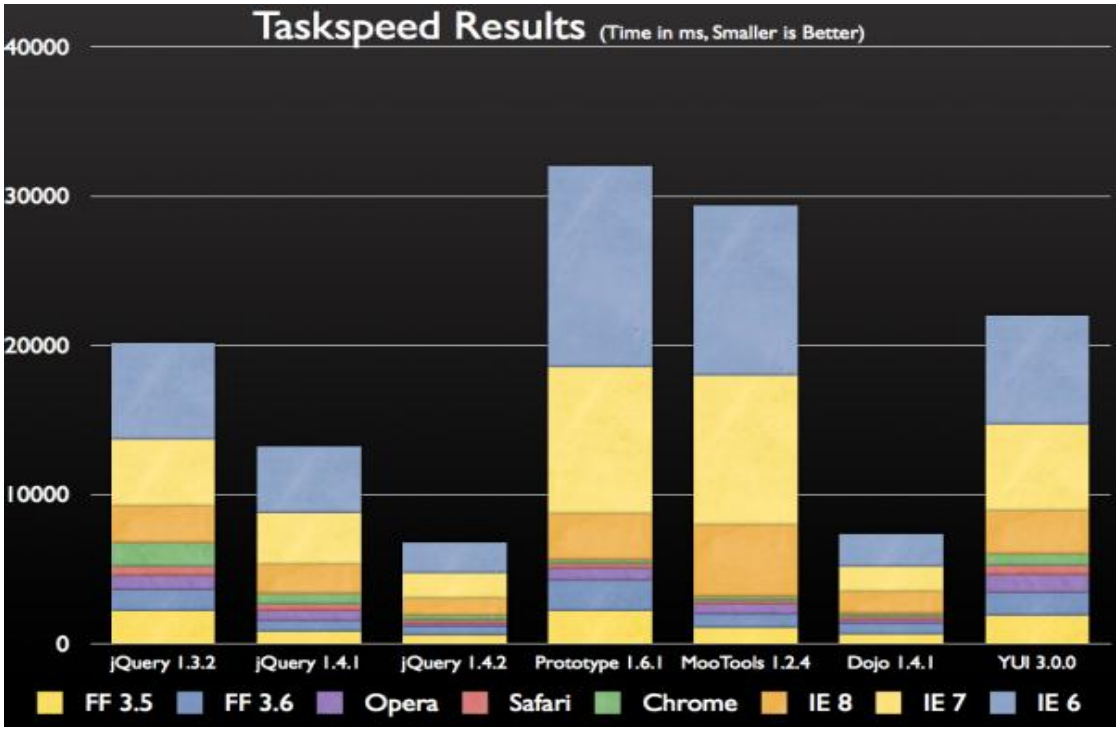
KUVA 2. JavaScript-kirjastojen markkinaosuudet Wappalyzeria mukaillen

Wappalyzer-dokumentaation (2012) tarjoaman tiedon mukaan peräti 61 % verkkosivuista käyttää jQuery-kirjastoa. Lisäksi 9 % käyttää jQuery-kirjaston pohjalta luotua jQuery UI-kirjastoa, joka on tarkoitettu käyttöliittymien kehitykseen. MooToolsia käyttää 9 %, Prototypeä 7 %, YUI:tä 3 % ja Dojoa 1 %. (Wappalyzer 2012.) Kuten taulukko 5 osoittaa, yli kaksi miljoonaa verkkosivua käyttää jQuery-kirjastoa. Seuraavaksi suurimpia kirjastoja MooToolsia löytyy hieman yli 300 000:lta ja Prototypeä noin 240 000:lta verkkosivulta. Verkkosivujen katselukerrat, niin kutsutut ”hitit”, vaihtelevat tosin suuresti kyseisiä kirjastoja käyttävillä sivuilla Wappalyzerin tarjoaman tiedon mukaan, kuten taulukko 5 osoittaa.

TAULUKKO 5. Kirjastojen käyttöstatistiikkaa Wappalyzeria mukaillen

Kirjasto	**Verkkosivuilla**	**Katselukerrat**
jQuery	2 083 328	48 339 488
MooTools	301 057	3 880 638
jQuery UI	294 414	6 396 310
Prototype	242 666	4 872 387
YUI	99 274	3 740 861
Dojo	30 568	271 317

Taulukosta 5 näkyy, että jQuery-kirjastoa hyödyntävillä verkkosivuilla on reilusti eniten katseluja, 48 miljoonaa, mutta muiden kirjastojen vastaavia lukuja tarkasteltaessa huomaa, etteivät katselukertojen määrä ole suoraan verrannollinen kirjastoa käyttävien verkkosivujen määrään. Esimerkiksi MooToolsia käyttää noin 60 000:tta verkkosivua enemmän kuin Prototypeä, mutta katseluja on Prototype-kirjastoa hyödyntävillä verkkosivuilla noin 4,8 miljoonaa ja MooTools-kirjastoa hyödyntävillä 3,8 miljoonaa. Katselukerrat eivät kerro suoranaisesti kirjaston soveltuvuudesta JavaScript-ohjelmointiin tai paremmuudesta muihin vastaaviin kirjastoihin nähden, mutta on kuitenkin huomion arvoinen seikka. Esimerkiksi MooTools on käytössä noin kolmisen sadalla tuhannella verkkosivulla ja katseluja näillä sivuilla on melkein neljä miljoonaa. Näiden kolmen sadan tuhannen verkkosivujen joukossa voi kuitenkin olla esimerkiksi yksi erityisen suosittu sivu, joka kerää jo itsessään miljoonia katseluja. Tällaisen sivun avulla, jossa käy niin paljon käyttäjiä, MooTools voi kasvattaa mainettaan paremmin kuin usealla tuhannella sivulla, joilla on muutama sata katselua. Kirjastoa käyttävien verkkosivujen määrä kuitenkin kertoo siitä, kuinka arvostettu ja tunnettu JavaScript-kirjasto on verkkokehittäjien mielestä.



KUVA 3. JavaScript-kirjastojen nopeusvertailu (Koch 2010)

Kuvasta 3 näkyy erilaisia JavaScript-kirjastoja, muun muassa useampi versio jQuery-kirjastosta. Kuvassa verrataan eri kirjastoilla toteutetun tehtävän toteutuksen kestoa eri selaimilla, joten mitä pienempi pylväs on, sitä tehokkaampi kirjasto on kyseessä. Kuvassa 3 tehtävän kestoa on koeteltu selainten vanhemmillakin versioilla. Kuvasta 3 käykin ilmi, että usealla kirjastolla aikaa kuluu eniten vanhojen selainten versioiden kanssa, kuten Internet Explorer 7- ja Firefox 3.6 -versioilla. jQuery 1.4.2 -versiolla tehtävän kesto on pienin, ja sillä on myös otettu hyvin huomioon vanhemmatkin selaimet. Dojo 1.4.1 -versiolla tulokset ovat käytännössä samat. Erot ovat kuitenkin niin naurettavan pieniä, koska vertailuyksikkönä on käytetty millisekunteja.

3.1.1 jQuery

John Resig esitteli jQuery-kirjaston virallisesti BarCampissa NYC:ssä vuonna 2006. Sittemmin siitä on tullut käytetyin JavaScript-kirjasto ja tällä hetkellä 20 prosenttia 10 000 suurimmasta verkkosivustosta, mukaan lukien Google, käyttävät sitä. jQuery näki päivänvalon siitä syystä, että siihen aikaan oli pula JavaScript-kirjastosta, joka tarjoaisi käyttäjilleen yksinkertaisen ja helposti lähestyttävän syntaksin. jQuery keräsi nopeasti kiinnostusta ja uusia ominaisuuksia, ja esimerkiksi Nokia ja Microsoft ovat avoimesti tukeneet avoimen lähdekoodin jQuery-kirjastoa. Nopeasti kasvanut käyttä-

jäkunta on auttanut jQuerya parantamaan sen ominaisuuksien ja koodin laatua rivakkaan tahtiin, mikä taas houkuttelee lisää kehittäjiä käyttämään sitä. jQueryn koodi on avoin ja lisensoitu MIT:n ja GNU General Public:n lisensseillä. (Bai 2010, 8.)

jQuery on hyvin monipuolinen JavaScript-kirjasto. Se noudattaa tiettyjä strategioita pysyäkseen laajasta sisällöstään huolimatta kompaktina pakettina, alle 20 kilobittisenä. jQuery sisältää kattavan ymmärryksen CSS:stä ja sillä on yksinkertaistettu CSS-tyylien muokkausta. Kehittäjän ei tarvitse edes välittää verkkoselainten eriävistä standardeista, koska jQuery huolehtii siitä itse. Dokumentin rakenteen käsittely onnistuu helposti CSS-valitsimien (selectors) avulla. jQuery sisältää jopa erilaisia animointitekniikoita tuomaan visuaalista säväystä, kuten häivyttämisen ja liukumisen animaatioita.

jQuery-kirjasto mahdollistaa komentojen ketjuttamisen ja monien elementtien käsittelemisen yhdellä komennolla. jQuery-kirjasto myös tarjoaa kattavan valikoiman tapahtumakäsittelyjä. Ajax-käsittelyistä on myös tehty helpompia jQuery-kirjastolla, joka poistaa selainkohtaiset ongelmat koko Ajax-prosessista ja yksinkertaistaa Ajax-tekniikan käyttöä. jQuerylle on tarjolla lisäksi monenlaisia lisäosia, jotka on suunniteltu tiettyjen ominaisuuksien kehittämistä varten. Lisäosien hallinta on helppoa, niitä voi helposti lisätä ja poistaa. (Chaffer & Swedberg 2009, 8–9.)

3.1.2 Prototype

Prototype on Wikipedian mukaan (2012) Sam Stephensonin vuonna 2005 kehittämä JavaScript-kirjasto. Rutterin mukaan (2010, 8) Prototype on ensimmäinen Ruby on Rails -rajapinnan yhteyteen rakennettu JavaScript kirjasto. Se luotiin alun perin avittamaan selainten välisissä AJAX-käsittelyissä, mikä oli siihen aikaan kuin miinakentällä tanssahtelemista. Lisäksi Prototype tuo helpotusta selainten yhteensopivuusongelmiin tapauskäsittelyissä, AJAX-kutsuissa ja DOM:n käsittelyssä. (Orchard ym. 2010, 2.)

Prototype laajentaa niin JavaScript-kieltä kuin myös DOM-elementtejä. Näitä elementtejä tai JavaScript-objekteja voi käsitellä helposti esimerkiksi Prototype-kirjaston each()- ja map()-metodeilla, jotka on tarkoitettu useiden kohteiden läpikäymiseen. JavaScript-kielen sisään rakennettuja objekteja voi laajentaa sisältämään erilaisia me-

todeja, kuten datatyypin määrittelyjä. Lisäksi funktioihin pystyy sisällyttämään hyödyllisiä metodeja, kuten wrap()-metodin, joka mahdollistaa ominaisuuksien liittämisen metodiin. (Orchard ym. 2010, 2.)

Prototype on suunniteltu yksinkertaistamaan luokkien perintää. Objekteja voi kehittää eteenpäin ja luoda hierarkioita ilman vastaavia ongelmia, mitä staattisesti kirjoitetuissa ohjelmointikielissä esiintyy. Prototype tarjoaa lisäksi hyvän pohjan oman JavaScript-kirjaston luomiselle. (Orchard ym. 2010, 2.)

Yksi Prototype-kirjaston suurimmista tarjoamista ominaisuuksista, DOM-rakenteen laajentaminen, on ollut eräänlainen kirosana viime vuosina kehittäjäpiireissä. Tämän vuoksi Prototype-kirjaston kehittäjätiimi on luvannut poistaa DOM-laajennukset tulevasta Prototype 2.0 -julkaisusta. (Ajaxian 2010.) Yksi Prototype-kirjaston kehittäjistä, nimimerkki Kangax (2010), luettelee useita DOM-laajennukseen liittyviä ongelmia ja sanoo sen jopa olevan yksi suurimmista virhevalinnoista Prototype-kirjaston kehitystyössä. Lyhyesti sanottuna Kanax (2010) kertoo artikkelissaan, että DOM-elementtien laajennuksen tuomat ongelmat selainten välisessä kehityksessä ovat suuremmat kuin sen tuomat hyödyt.

3.1.3 YUI

YUI (Yahoo! User Interface) on ilmainen ja avoimen lähdekoodin JavaScript- ja CSS-kirjasto. Wikipedian mukaan (2012) Thomas Sha perusti YUI kirjasto –projektin vuonna 2005 ja Yahoo! alkoi sponsoroida sitä. Wikipedia (2012) tietää myös kertoa, että seuraavana vuonna YUI julkaistiin ja sitä kehitetään edelleen aktiivisesti. Purettuna se on jopa 50 megabittiä iso kansio. Kirjasto koostuu neljästä osiosta; YUI core (ydin), utilities (työkalut, hyödykkeet), controls/widgets (kontrollit/widgetit) ja CSS työkaluista. Jokainen kokonaisuus on jaettu edelleen pienempiin komponentteihin, jotka voi sisällyttää verkkosivulle yksittäin, jotta ei tarvitse ladata koko kirjastoa ja näin ollen hidastaa verkkosivun toimintaa. (Orchard ym. 2010, 90.)

Yahoon sivuston mukaan (2012) jokainen komponentti on riippuvainen YAHOO Global Objectista, joka luo tärkeän perustan tarjoamalla yhteisen nimiavaruuden komponenteille. YUI:n periaatteisiin kuuluu, että yleiset muuttujat ovat pahasta tehokkuuden kannalta ja koska ne voivat aiheuttaa konflikteja muiden muuttujien kanssa. Siksi

YUI käyttää vain YAHOO nimistä yleistä muuttujaa, jonka sisällä kaikki komponentit ovat organisoituina perittyjen objektien sisällä. Useimmat komponentit tarvitsevat lisäksi DOM Collectionin (kokoelman) ja Event Utilityn (tapaustyökalut). Komponenteista on saatavilla erilaisia versioita tarpeen mukaan, minimaalisessa koossa, standardissa ja testausversiona, jossa voi tarkastella tarkemmin komponenttien toimintaa. Komponentit löytyvät myös Yahoon palvelimelta, jota kautta ne voi myös sisällyttää verkkosivulleen. (Orchard ym. 2010, 90–91.)

Vuonna 2009 Yahoo! julkaisi YUI 3 version. Kirjastoa on kehitetty edellisestä versiosta purkamalla yhtenäisiä osia edelleen pienemmiksi. Täten moni moduuli koostuu useista pienemmistä alamoduuleista. Näin ollen kehittäjän haluama ominaisuus voi löytyä suoraan jostain alamoduulista, eikä hänen tarvitse ladata koko isäntämoduulia. Tämä lisää luonnollisesti kirjaston tehokkuutta ja nopeutta. YUI 3 sisältää muitakin uudistuksia, esimerkiksi se käyttää paljon CSS valitsimia tehokkaasti, tiedostokoko on pienempi ja moduulit ovat sidottuja niiden käyttötapauksiin, eli jos joku kehittäjätiimistä poistaa toisen käyttäjän käyttämän moduulin, se ei vaikuta itse koodiin. (Miraglia 2009.)

3.1.4 Ext JS

Ext JS on myös tunnettu JavaScript-kirjasto, jonka historia on vahvasti sitoutunut YUI-projektiin. Vuonna 2006 Jack Slocum kehitti YUI:hin pohjautuvan dataverkon (data grid), joka keräsi suosiota kehittyneiden ominaisuuksiensa, visuaalisuutensa ja käyttäjäystävällisyytensä ansiosta. Suunnilleen vuoden päästä Slocum päätti irtautua YUI kirjastosta ja julkaisi Ext JS 1.0 -hallintakirjaston. Slocum ei halunnut kirjastonsa olevan pelkästään riippuvainen YUI:sta, vaan halusi tarjota sitä muillekin JavaScript-kirjastoille. Ext JS -kirjasto toimi siis eräänlaisena adapterina, jonka pystyi liittämään muiden kirjastojen päälle. Oman yksityisen nimiavaruutensa ansiosta Ext JS ei aiheuttanut konflikteja kehittäjien käyttämien kirjastojen kanssa. Vain muutama kuukausi 1.0-version jälkeen ilmestyi versio 1.1, joka perustui nyt täysin Ext JS -koodiin ilman mitään riippuvuuksia muihin ohjelmiin tai kirjastoihin. Ext JS -kehittäjätiimi on säilyttänyt ”adapteri” ominaisuuden JavaScript-kirjastossaan, mutta lisännyt myös omia ominaisuuksiaan Ext JS -hallintakirjastoon. (Orchard ym. 2010, 336.)

Ext JS -kirjasto sisältää kaikki perusominaisuudet, joita JavaScript-kehittäjät ovat oppineet odottamaan JavaScript-kirjastoilta, kuten DOM:n käsittelyä, CSS:n ja HTML:n muuttelemista ja yksinkertaistettua Ajax-toimintaa. Ext JS -kirjaston kehittäjätiimi on kuitenkin pyrkinyt tekemään kirjastostaan hieman erilaisen peruskirjastoihin verrattuna. Kehittäjätiimi on tähdännyt tarjoamaan Ext JS -kirjastollaan kattavan ja teemakohtaisen hallintakirjaston, jota pystyy laajentamaan ja joka on visuaalisesti miellyttävän näköinen käyttää. Ext JS on modulaarinen kirjasto, eli kehittäjä voi itse valita mitä ominaisuuksia ottaa sovellukseensa käyttöön. Ext JS käyttää vahvasti hyödykseen JavaScriptin objekti orientoitumista, kuten monimuotoisuutta (polymorphism) ja kapselointia. (Orchard ym. 2010, 336.)

Vuonna 2011 ilmestyi Ext JS 4 -versio, joka sisälsi runsaasti uusia ominaisuuksia. Uusi versio vie entisistä Ext JS -versioista tutun luokkasysteemin vielä pidemmälle. Luokat määritellään kirjastossa uudella tavalla, joka vähentää mahdollisten virheiden määrää. Uuden version myötä luokkiin voi lisätä erikoisia käyttäytymisiä, joita on ennen ollut vaikea lisätä ja ketjuttaa. Lisäksi Ext JS 4 tuo mukanaan uuden ominaisuuden, dynaamisen latauksen. Tämä tarkoittaa sitä, että kehittäjä voi itse määritellä minkä luokan haluaa ladata ja dynaamisen latauksen ansiosta Ext JS lataa sen ja kaikki muut mahdolliset luokat, mistä ladattu luokka on riippuvainen. Dynaaminen lataus tapahtuu käyttäjäpäässä, eikä vaadi palvelinpään asennuksia. (Spencer 2011.)

3.1.5 Dojo

Dojo on avoimen lähdekoodin JavaScript-kirjasto, jonka päätavoitteena oli alun perin yhdistää kourallinen DHTM- työkaluja ja JavaScript-kirjastoja. Dojo-kirjaston tärkein tukija on voittoa tavoittelematon Dojo Foundation -järjestö. Kyseinen järjestö toimii sponsorina Dojo-projektia ja toimii myös lisensoijana jokaiselle Dojo-projektin osallistuvalle, joka edistää Dojo-kirjaston lähdekoodia.

Dojo sisältää muiden JavaScript-kirjastojen tapaan yksinkertaistettuja ratkaisuja JavaScriptin ohjelmointikieleen ja mahdollisuudet käsitellä DOM-rakennetta ja Ajax-kutsuja helposti. Dojon todellinen vahvuus löytyy käyttöliittymien ja kokonaisten sovellusten rakentamisesta HTML-merkintäkieltä soveltamalla. Dojon avulla ei tarvitse yhdistellä sovelluksia pitkillä pätkillä objektien alustuksia ja DOM-elementtien valmistelemisia, vaan Dojo-kirjastolla on mahdollista käydä läpi koko verkkosivu ja aset-

taa toimintojen käynnistyksiä Dojo-koodille. Kehittäjä voi ohjelmoida ilmankin tätä ominaisuutta, mutta se on kuitenkin tehokas tapa ohjelmoida. (Orchard ym. 2010, 452.) Näitä käynnistyskohtia kutsutaan Dijiteiksi, tai mahdollisesti Dojo hyödykkeiksi (widgets). Dijit-hyödykkeet ovat hyvin monipuolisia ja niistä on pyritty tekemään mahdollisimman itsenäisiä ja kevyitä. Jotkin Dijit-hyödykkeet ovat kuitenkin riippuvaisia toisistaan. Kehittäjät voivat kehittää myös omia hyödykkeitään tai muokata jo olemassa olevia omassa projektissaan. (Svensson 2008, 57.)

Dojo-kirjasto on koottu tehokkaasti ja organisoidusti, koostuen kattavasta valikoimasta komponentteja, hyödykkeitä ja lisäosista. Dojo on modulaarinen kirjasto ja kaikki moduulit on jaettu selkeisiin nimiavaruuksiin. Modulaarisuutensa ansiosta Dojo-kirjastosta voi sisällyttää verkkosivulleen vain tiettyjä ominaisuuksia. Dojo-kirjasto mahdollistaa moduulien dynaamisen latauksen. Dynaamisella latauksella Dojo lataa myös kaikki moduulin toiminnasta riippuvat ominaisuudet. Dojo-kirjaston hyvin suunnitellun pakkausmenetelmän ansiosta kehittäjä pystyy järjestelemään ja poimimaan hyödykkeisiin ja käyttäjän koodiin liittyviä tiedostoja, kuten CSS-tyylejä ja kuvia. (Orchard ym. 2010, 452.)

3.1.6 MooTools

MooTools (My Object-Oriented ”JavaScript” Tools) on Valerio Proiettin kehittämä kevyt, modulaarinen ja objekti orientoitunut JavaScript-kirjasto. MooTools on avoimen lähdekoodin JavaScript-kirjasto, joten kuka vain verkkokehittäjä voi ehdottaa parannuksia, uusia ominaisuuksia ja kehittää eteenpäin jo olemassa olevaa koodipohjaa. (Gube 2009, 8–11.) MooTools on enemmän natiivi JavaScript-työkalu kuin useimmat JavaScript-kirjastot. MooTools nojaa vahvasti standardoituun JavaScriptiin ja kehittää sitä eteenpäin. Toisin kuin esimerkiksi jQuery, joka käyttää periaatteessa omaa kieltään. MooTools on täynnä HTML-skriptejä ja on riippuvainen JavaScript-ohjelmointikieleen sisään rakennetuista nimiavaruuksista ja modulaarisuuksista. MooTools-kirjastoa verrataan usein Prototype-kirjastoon, koska MooTools lisää Prototype-kirjaston tavoin toiminnallisuuksia JavaScriptissä jo ennestään oleviin objekteihin. (Orchard ym. 2010, 690.) Vertaus ei ole ihan tuulesta temmattu, koska Guben (2009, 8) mukaan MooTools oli alun perin itse asiassa lisäosa Prototype-kirjastoon.

MooToolsin kehittäjät uskovat vahvasti objekti orientoituneen ohjelmoinnin periaatteisiin ja niiden hyödyntämiseen JavaScript-ohjelmoinnissa. MooTools mahdollistaakin helpomman ja näppärämmän tavan manipuloida JavaScriptin objekteja. (Gube 2009, 8.) Kuten useimmilla suosituilla JavaScript-kirjastoilla, MooTools-kirjastolla voi myös sisällyttää projektiinsa vain niitä ominaisuuksia, mitä kehittäjä mielestään tarvitsee. Lisäksi kehittäjä voi lisätä omia toiminnallisuuksiaan luomalla alaluokkia tai lisäämällä suoraan omia funktioita MooTools-projektiinsa. (Orchard ym. 2010, 690.) MooTools sisältää jo itsessään paljon hyödyllisiä funktioita ja metodeja, joita löytyy kaikenlaisiin erikoistilanteisiin, kuten erilaisten tapahtumien käsittelyihin. Kuten monen muunkin JavaScript-kirjaston kohdalla, kehittäjän ei tarvitse ohjelmoidessaan ottaa lainkaan huomioon eri verkkoselainten ominaisuuksia ja tapoja tulkita JavaScript-kieltä, vaan MooTools hoitaa sen kehittäjän puolesta. Lisäksi MooTools yksinkertaistaa DOM-rakenteen käsittelyä. MooTools on avoimen lähdekoodin JavaScript-kirjasto, joten kuka vain verkkokehittäjä voi ehdottaa parannuksia, uusia ominaisuuksia ja kehittää eteenpäin jo olemassa olevaa koodipohjaa. (Gube 2009, 9–11.)

3.1.7 Google Web Toolkit

Google Web Toolkit (GWT) on Java-ohjelmointikieleen pohjautuva työkalu JavaScript-verkkosovellusten rakentamiseen. GWT-kirjaston päälle voi myös lisätä muita kirjastoja. GWT-kirjaston tavoite on mahdollistaa tehokkaiden verkkosovellusten luominen ilman että kehittäjä tietää kaiken selainkohtaisista ongelmista, Ajax käsittelyistä ja JavaScriptistä. GWT on ilmainen, avoimen lähdekoodin kirjasto ja sen käyttäjäkunta kasvaa jatkuvasti. GWT sisältää SDK:n, joka sisältää Java API -kirjastoja, kääntäjän ja kehityspalvelimen. GWT-kirjastolle on tarjolla paljon lisäosia. Speed Tracer, joka on Chrome-selaimelle tarkoitettu lisäosa sovellusten tarkkailuun, oma lisäosa Eclipselle ja GWT Designer, joka on työkalu käyttöliittymien luomiseen. (Google Developers 2012.)

GWT-kirjastolla kehitettyjen sovellusten virheiden tarkistaminen tapahtuu samoin kuin Java-sovelluksilla. Samalla sovellusta voi tarkastella selaimella aivan kuin se olisi luotu JavaScript-ohjelmointikielellä. Kehitys siis tapahtuu Java-kielellä, mutta sovellus muutetaan valmistuttuaan JavaScriptiksi. Valmis GWT-kirjastolla luotu sovellus koostuu tarkemmin sanottuna JavaScriptistä, HTML:stä ja XML.stä. Käännetty JavaScript on tarkkaan optimoitua ja auttaa suojautumaan Cross site scripting (XSS,

sivulle hyökkäämistä asiakaspään HTML:n tai JavaScriptin avulla) -hyökkäyksiä vastaan. Tämä kuitenkin toimii vain, jos käyttäjä ei lisää omaa JavaScript-koodia tai käytä JSON API -rajapintaa analysoimaan epäluotettavia tekstinpätkiä. (Guermeur & Unruh 2010, 12.) Tästä löytyy kuitenkin GWT-kirjaston heikkous, koska kehittäjän täytyy ohjelmoida pelkästään GWT-kirjastoon sisäänrakennetuilla ominaisuuksilla eikä voi kehittää sitä eteenpäin suoraan omassa sovelluksessaan ilman tietoturvaongelmia.

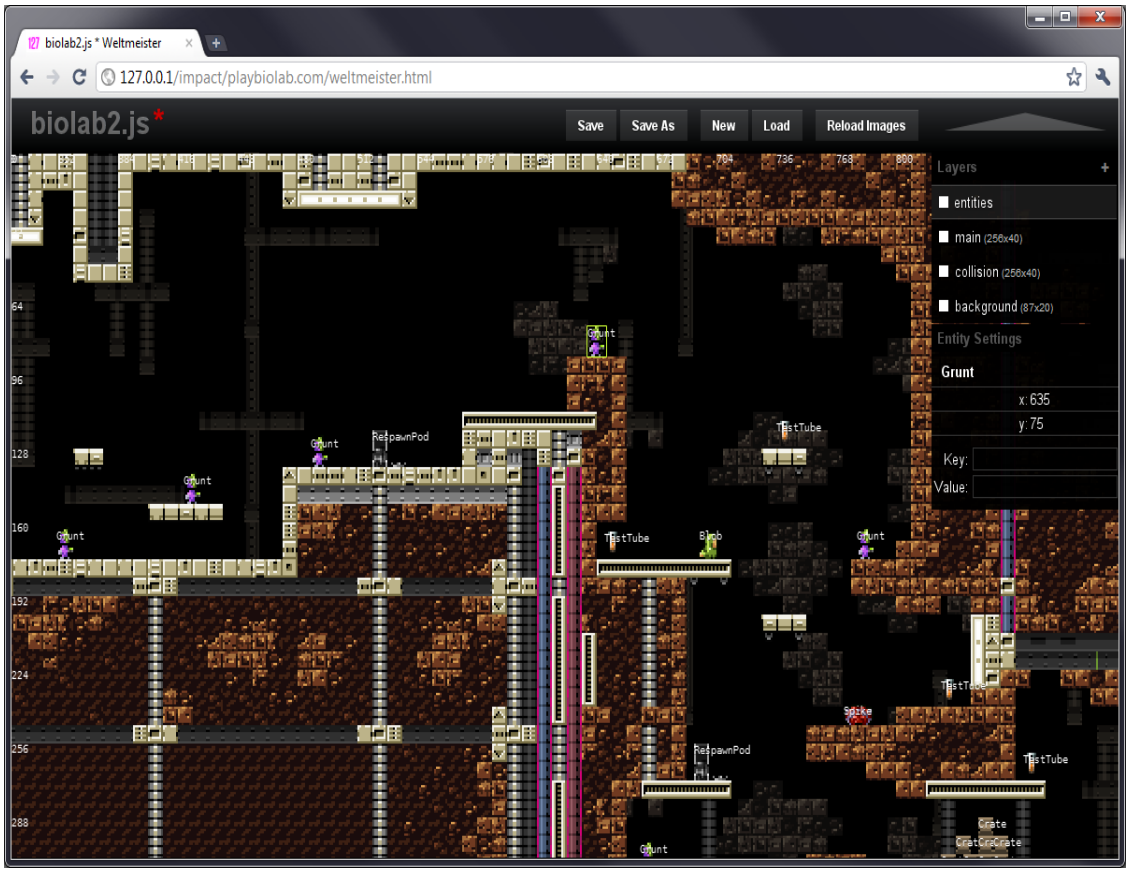
Kuten sanottu, GWT-kirjastolla kehittäessä ei tarvitse ottaa huomioon eri selainten ongelmia. GWT luo erilaisen version jokaiselle selaimelle, kun sovellus muunnetaan JavaScriptiksi. Kun käyttäjä saapuu GWT-kirjastolla luotuun palveluun, palvelu lataa käyttäjän selaimelle tarkoitetun version. GWT-kirjastolla on myös mahdollista jakaa koodin latauksen eri aikajaksoihin, jos koodipohjan latausaika on merkittävän suuri. Kehittäjä voi asettaa eräänlaisia lataustaukoja koodiinsa, joissa sovellus pysähtyy lataamaan. GWT-kirjastolle on tarjolla monia valmiita paneeleita ja graafisia elementtejä (widgets) käyttöliittymäsuunnitteluun. Käyttäjä voi myös kehittää omiaan. (Guermeur & Unruh 2010, 13.)

3.2 JavaScript-pelikirjastot

HTML5:n ja CSS3:n tarjoamien ominaisuuksien myötä pelien kehitys kyseisillä tekniikoilla on lisääntynyt merkittävästi. JavaScript on olennainen osa tätä pelikehitystä sen tarjoaman dynaamisuuden ja interaktiivisuuden vuoksi. Perinteiset JavaScript-kirjastojen tarjoamat ominaisuudet ja toiminnallisuudet eivät kuitenkaan välttämättä täysin kohtaa JavaScript-pelikehittäjän vaatimuksia ja ne saattavat tarjota sellaisiakin ominaisuuksia, joille pelikehittäjällä ei ole käyttöä. Tähän tarpeeseen on syntynyt useita JavaScript-pelikirjastoja ja tässä luvussa käsittelen mielestäni niistä käytetyimpiä ja potentiaalisimpia.

Impact JavaScript-pelikirjasto on Dominic Szablewski kehittämä ja se on kenties paras valinta JavaScript- ja HTML5-pelikehitykseen tällä hetkellä. Impact hyödyntää canvas-piirtoalustaa tehokkaiden 2d-pelien tekemiseen verkkoon ja mobiililaitteille. Useat pelikehittäjät kokevat varmasti Impactin 99 $ hinnan esteenä, kun on olemassa runsaasti ilmaisia, avoimen koodipohjan JavaScript-pelikirjastoja. Freemanin mielestä Impact on kuitenkin niin pitkälle kehitetty ja hiottu, ettei hän ole nähnyt vastaavaa muiden ilmaisten JavaScript-pelikirjastojen kohdalla. Impactin suurimpia etuja on sen

helppokäyttöisyys, havainnolliset koodiesimerkit, aktiivinen kehittäjäkunta ja kehittynyt kenttäeditori Weltmeister. (Freeman 2012, 1.) Weltmeisterillä onnistuu kaikenlaiset pelimaailmat sivusta kuvatuista aina ylhäältä alas kuvattuihin 2d-peleihin. Kaikkia pelin elementtejä, kuten vihollisia ja eri tilanteiden laukaisijoita, voi välittömästi käyttää Weltmeisterissä ja niitä voi yhdistää toisiinsa logiikkaketjujen muodostamiseksi. Esimerkiksi kun pelihahmo painaa nappia, ovi avautuu toisaalla. (Szablewski 2012.) Kuvassa 4 näkyy kuvakaappaus Weltmeisteristä. Kuvan peli on Impact-verkkosivuston mukaan (Szablewski 2012) Biolab disaster, joka on tehty Impact pelikirjastolla selaimessa, iPhonessa ja Android puhelimilla pelattavaksi.



KUVA 4. Impact pelikirjaston Weltmeister kenttäeditori (Szablewski 2012)

Impactilla tehdystä pelistä voi myös luoda natiivin oloisen iOS-pelin, jota voi myydä Apple Storessa. Tämä onnistuu Ejecta-kirjastolla, joka ajaa JavaScript-koodin JavaScriptCorella (JavaScript moottori), muuntaa HTML5:n canvas-piirtoalustan toiminnot OpenGL:lle ja audion toimimaan OpenAL:lla. Monet muutkin rajapinnat, kuten kosketus ja paikallinen datavarasto, toimivat aivan kuin oikeassa selaimessa. Ejecta-kirjasto oli alun perin osa Impactia, mutta kasvoi ulos siitä omaksi kirjastokseen. Vanhemmat Ejecta-versiot (aiemmin nimi oli iOSImpact) käyttivät paljon eräänlaisia

purkkaviritelmiä eivätkä olleet kovin vakaita, mutta nyt Ejecta tavallaan matkii HTML5-rajapintaa ja toimii useiden canvas-kirjastojen kanssa, kuten CAAT:n ja ThreeJS:n. Ejecta ei ole täysin valmis kirjasto, se ei tue vielä kaikkia canvas-piirtoalustan metodeja, kuten liukuvärejä, varjojen muodostusta ja tekstipiirtelyn funktioita. (Szablewski 2012.)

enchant.js on HTML5-merkkikieltä ja JavaScriptiä yhdistävä pelikirjasto, jolla on mahdollista kehittää pelejä eri alustoille; tietokoneelle, Macille, iPhonelle ja iPadille sekä Androidille. enchant.js kirjaston lähdekoodi löytyy hyvin kompaktissa 30 kilobitin paketissa. Ubiquitous Entertainment Inc laitoksen Akihabara Research Center on kehittänyt enchant.js-kirjastoa ja vuoden 2011 huhtikuussa julkaisi sen. Kahden kuukauden sisällä julkaisusta sillä oli kehitetty jo lähes 200 peliä. enchant.js on lisensoitu MIT:n ja GPL:n alle. (enchant.js 2011.)

Pelikirjasto enchant.js ohjelmoinnissa käytetään paljon objekteja. Itse asiassa jokainen elementti on objekti. Peliobjekti ohjaa koko peliä ja tarjoaa perustan pelin toiminnalle. Peliobjekti on koko pelin sydän. Kohtausobjektit ovat kokoelmia visuaalisia elementtejä. Esimerkiksi pelissä pisteiden esitys on kohtausobjekti. Noodiobjektit ovat visuaalisia objekteja, joista peli koostuu. Noodit voivat olla spritejä, joilla kuvataan hahmoja, tekstielementtejä tai pelikarttoja. Pelisuunnittelu enchant.js:llä on kohtaussidonnaista, eli pelissä edetään tapahtumien (events) kautta asynkronisesti. enchant.js kirjastolle on myös mahdollista asentaa erilaisia lisäosia ja sille on myös saatavilla ilmaista pelimateriaalia suoraan pelikirjaston sivuilta. Tällä hetkellä enchant.js tukee vain yksittäistä kosketusta lukuisilla Android-laitteilla, mutta kehittäjillä on suunnitteilla tuki multikosketukselle, kehittyneemmät piirtotyökalut, ilmaiset grafiikkamateriaalit kaikille käyttäjille ja erikoistunut karttaeditori. (enchant.js 2011.)

jQuery pelikirjastolle on kehitetty varta vasten peliohjelmointia ajatellen gameQuery niminen lisäosa. gameQuery helpottaa JavaScript-pelikehitystyötä jQuerylla lisäämällä yksinkertaisia peleihin liittyviä luokkia. gameQueryn syntaksi ei juuri eroa jQueryn syntaksista. Tietokoneinsinööri Selim Arsever kehitti gameQueryn ja jatkaa edelleen sen kehittämistä. Arsever päätti perustaa gameQueryn ohjelmoinnin DOM:n käsittelyyn canvas-piirtoalustan sijasta useista syistä. Hän halusi oppia lisää DOM:n käsittelystä, kokeilla JavaScriptin rajoja ja tehdä gameQuerysta aloittelijaystävällisen virheentarkistuksia varten, sillä kaikki objektit ovat selkeästi näkyvissä yksinkertaisessa

DOM rakenteessa. Arsever on listannut gameQueryn pääperiaatteiksi helppokäyttöisyyden, pysymisen lähellä jQueryn filosofiaa, nopean toiminnan ja mahdollisuuden aloittelijoillekin tehdä rikkaita 2d-pelejä. gameQuery-pelikirjaston ominaisuuksiin kuuluu monen tason sprite-animaatiot, spritehierarkiat, suorakulmaiset laattakartat, törmäyksentunnistus, erinäisiä audiotukia, jaksottaisia kutsuja ja näppäimistön tilan tarkastelu. jQueryn henkeen myös gameQuery on ilmainen ja lähdekoodi on vapaa kaikille. (gameQuery 2012.)

LimeJS on Digital Fruit -yhtiön kehittämä JavaScript-pelikirjasto, joka pyrkii tarjoamaan nopean ja helpon tavan tehdä pelejä moderneille selaimille ja mobiililaitteille, joihin kuuluu iOS-laitteet (iPad, iPhone ja iPod Touch) sekä Android-laitteet. LimeJS-kirjastolla kehitetty peli nimeltään Voodoo Friends näkyy kuvasta 5. LimeJS tukee kaikkia suurimpia selaimia, paitsi Internet Exploreria, ainakin tällä hetkellä. LimeJS on kehitetty Closure JavaScript-kirjastolla, joka taas on Googlen kehittämä. Closurella on kehitetty esimerkiksi Gmail ja Google Docs. LimeJS tukee spritekarttoja ja WebGL-tukikin on kehitteillä. LimeJS-pelikirjaston haittapuolia ovat sen riippuvaisuudet. Kehitystyö LimeJS:llä vaatii Python 2.6 tai uudemman version, Git-palvelun ja Subversion- tai GitSVN-versionhallintapalvelut. Lisäksi suorituskyky huolestuttaa, koska alkuasetuksen määritteleminen jo lataa useita ohjelmia, kuten Closure-kirjaston sekä Box2D-fysiikkakirjaston, Closure kääntäjän ja valmiita malleja Closure-kirjastolle. (LimeJS 2012.)

KUVA 5. Voodoo Friends (LimeJS 2012)

LimeJS ei ole ehkä vielä ihan viimeiseen asti hiottu, mutta sillä on potentiaalia nousta varteenotettavaksi vaihtoehdoksi HTML5/JavaScript pelikehitystyökaluksi. LimeJS:n kehitystyö on vain kysymysmerkki. LimeJS:n (2012) sivuilta käy ilmi, että kaksi edellistä päivitystä ovat maaliskuulta 2012 ja kesäkuulta 2011, ja molemmat päivitykset käsittelevät LimeJS:llä tehtyjä sovelluksia. Näiden tietojen pohjalta kehitys ei vaikuta siis liikkuvan juuri eteenpäin, mikä on harmi. LimeJS:n (2012) sivuilla lukee, että se julkaistiin vuoden 2011 helmikuussa ja sitä kehitettiin lyhyessä ajassa rivakasti eteenpäin.

Crafty on JavaScript-pelikirjasto, jolla voi tehdä pelejä järjestelmällisesti. Crafty:n avainominaisuuksia ovat oliot ja komponentit. Oliot ovat perusrakennuspalikoita Crafty-kehityksessä. Jokainen interaktiivinen osa on olio. Komponentit ovat toiminnallisuuksia, joita voi liittää olioihin. Crafty-pelikirjastolla on paljon hyviä ominaisuuksia, jotka tekevät siitä oivan valinnan JavaScript-pelikehittäjälle. Se tukee kaikkia suurselaimia, jopa Internet Explorer 9:ää. Se on suhteellisen pienikokoinen, 88,4 KB minimalistisessa versiossa. Pelikehittäjälle on jätetty valinnanvaraa pelielementtien luomisessa ruudulle, sillä Crafty:llä voi käyttää joko HTML5:N canvas-piirtoalustaa tai perinteistä DOM-rakennetta. Crafty tukee myös spritekarttoja ja sisältää törmäystunnistusjärjestelmän sekä tapahtumasidonnaisuuden, kehittäjä voi luoda omia tapahtumiaan myös. Crafty-pelikirjaston tulevaisuus vaikuttaa lupaavalta, sillä sen ilmaisen ja avoimen koodipohjan ansiosta sillä on tukena innokas kehittäjäkunta, joka kehittää jatkuvasti omia moduuleita, joita kehittäjä voi oman makunsa mukaan ottaa käyttöön omaan peliprojektiinsa. (Crafty 2012.) Craftyn vahvuus on sen monipuolisuus, koska sillä voi ohjelmoida eri tavoilla ja siihen voi myös sisällyttää omia kirjastoja täydentämään kehitystyötä.

EntityJS on JavaScript-pelikirjasto, jonka pääperiaatteisiin kuuluu olla joustava, uudelleen käytettävä ja laaja. Kaiken tämän saavuttamiseksi se käyttää olio- ja komponenttisuunnittelua, jossa kaikki logiikka on komponenteissa ja oliot kootaan näistä komponenteista. Komponentin logiikka kirjoitetaan vain kerran ja sitä voi käyttää missä vain. Näin metodit eivät ole sidonnaisia luokkahierarkiasta eikä kehittäjän tarvitse kirjoittaa samoja asioita uudestaan ja uudestaan, tehden pelin koodista entistä suurempaa ja sekavampaa. EntityJS tukee kaikkia suurimpia selaimia ja tuki mobiili-

laitteille on tulossa. EntityJS:llä on tuki myös Tiled Map Editor -kenttäeditorille, mutta pelikirjasto ei tällä hetkellä sisällä valmista komponenttia karttojen lataamista varten, joten kehittäjän täytyy kirjoittaa oma. EntityJS:n kotisivu tarjoaa tosin ohjeet tähän. EntityJS on kirjoitettu Ruby-ohjelmointikielellä, ja EntityJS onkin tarjolla Rubylle lisäosana (gem), jonka voi ladata ja liittää osaksi projektiaan. Rubyn ansiosta tiedostojen liittäminen projektiin on helppoa. Ne tarvitsee vain laittaa oikeaan kansioon projektissa. Datatiedostot, kuten json- tai xml-tyyppiset, muunnetaan automaattisesti komponenteiksi ja ovat saatavilla pelikehityksessä. (EntityJS 2012.)

KUVA 6. melonJS tutoriaalin demopeli (melonJS 2012)

Yksi lupaavista pelikirjastoista on myös melonJS, jolla on tehty useita hyvän näköisiä pelejä, kuten kuvasta 6 näkyy. Kuvassa 6 on yksinkertainen tasohyppelydemo, jonka tekemiseen löytyy yksityiskohtaiset ohjeet pelikirjaston kotisivuilta. melonJS on ilmainen, avoimen lähdekoodin JavaScript-pelikirjasto. Kirjasto on vielä kehittäjien mukaan kesken, mutta sillä on jo tehty monia hienoja pelejä. melonJS on kevyt pelikirjasto spritepohjaisten 2d-pelien tekemiseen, mikä toimii kaikilla suurimmilla selaimilla. Pelikirjastoon sisältyy alkeelliset törmäyksen tunnistuksen ja fysiikan mallinnuksen mekaniikat, jotta pelit toimisivat vanhemmillakin koneilla. Kirjasto pitää tällä hetkellä sisällään perusvalikoiman objektiolioita, joita tulee myöhemmin lisää kehittäjien mukaan. Pelikirjastossa on sisäänrakennettu pelitilan hallintajärjestelmä esimerkiksi latausruudun ja päävalikon hallitsemiseen. melonJS tukee myös Tiled map editor –kenttäeditoria ja kosketuslaitteita. (melonJS 2012.) melonJS pelikirjaston kehitys on ollut nopeaa ja sillä on paljon hyviä ominaisuuksia, mutta vielä riittää kuitenkin kehitettävää. Tulevaisuudessa se on varmasti hyvä vaihtoehto JavaScript-pelikehitykseen, eikä ole huono valinta tälläkään hetkellä.

3.3 Muut JavaScript-pelikirjastot

Yleiskäyttöiset JavaScript-pelikirjastot eivät ole ainoa tapa käyttää JavaScriptiä peliohjelmointiin, sillä on olemassa myös tiettyjen peliominaisuuksien kehittämiseen luotuja JavaScript-kirjastoja. Esimerkiksi jos kehittäjä ei ole tyytyväinen jonkin JavaScript-pelikirjaston näppäinten käsittelyyn, hän voi ladata tätä tarkoitusta varten luodun erillisen kirjaston, kuten THREEx.KeyBoardstate.js-kirjaston, joka yksinkertaistaa näppäinten käsittelyä JavaScriptillä. Tässä luvussa käyn läpi näitä pelien eri ominaisuuksia varten kehitettyjä kirjastoja. Moni näistä kirjastoista liittyy 3D-pelien tekemiseen, joihin aiemmin luetellut JavaScript-pelikirjastot eivät juuri keskity.

three.js on kevyt ja avoimen lähdekoodin 3D-kirjasto, josta on pyritty tekemään yksinkertainen aloittelijoitakin varten. Kirjasto tarjoaa renderöintiä varten joko canvas-piirtoalustan, SVG:n tai WebGL:n. (three.js 2012.) Lewisin (2012) mukaan three.js-kirjaston avulla voi luoda kameroita, objekteja, valoja, materiaaleja ja paljon muuta. Internetistä löytyy lukematon määrä vaikuttavia demoja, jotka on tehty three.js-kirjaston avulla ja tällä hetkellä three.js on kerännyt paljon suosiota kehittäjäpiireissä. Kirjasto three.js:n alphaversio julkaistiin vuonna 2010 ja siitä lähtien sitä on kehitetty aktiivisesti ja sen sisältämä ominaisuustarjonta on kasvanut kattavaksi.

Bullet on ammattilaiskäyttöön tarkoitettu ilmainen pelifysiikkamoottori. Bullet-kirjastolla ohjelmointi tapahtuu C++-ohjelmointikielellä, mutta se täytyy esitellä, koska siitä on tehty JavaScript-versio. Bullet on ollut käytössä useissa suuren budjetin elokuvissa ja peleissä kaikilla suurilla pelikonsoleilla. (Bullet 2012.) Bullet tarjoaa työvälineet tehdä törmäyksentunnistuksia, hallinnoida törmäyksiä ja rajoitteita sekä fysiikanmallinnus jokaiselle objektille. Bullet on suunniteltu modulaariseksi ja muokattavaksi. Kehittäjä voi tehdä siitä oman näköisensä ja vastaamaan projektinsa tarpeita esimerkiksi ottamalla joitain ominaisuuksia pois tai muokkaamalla olemassa olevia. Pelikehittäjä voi vaikkapa käyttää projektissaan pelkästään Bulletin tarjoamaa törmäyksentunnistusta tai käyttää omaa muistin käytön priorisointia.

Bullet-kirjastosta löytyy työkalut tehdä niin sanottuja jäykkiä (rigid) tai pehmeitä (soft) kehodynamiikkoja. Pehmeä kehodynamiikka tarkoittaa tietyllä tapaa objektin ulkopuolista dynamiikkaa, kuten vaatteiden fysiikkamallinnusta. Jäykkä dynamiikka

tarkoittaa tavallaan objektin sisäistä kehodynamiikkaa, se lisää voimaa, massaa, inertiaa, nopeutta ja rajoitteita. Jäykkä kehodynamiikka on rakennettu törmäyksentunnistuksen päälle. (Coumans 2012, 11–32.) Bullet-verkkosivuston (2012) mukaan Bullet on integroitu Cinema 4D, Lightwave, Blender ja Carrara-3D-mallinnusohjelmiin, ja Bullet-lisäosia on saatavilla myös Maya, Houdini ja 3ds Max-ohjelmille.

Bulletista käännetty JavaScript-versio on nimeltään ammo.js. Käännös on tehty Emscriptenillä. Bullet-kirjaston lähdekoodin käännös JavaScriptiin on tapahtunut täysin ohjelmallisesti ilman ihmiskosketusta. Tästä syystä kehityksessä voi ilmetä ongelmia, minkä ammo.js:n kehittäjäkin myöntää. JavaScript-version nimi ammo tulee lauseesta ”Avoided Making My Own js physics engine by compiling bullet from C++”, mikä tarkoittaa, että kehittäjä vältti oman JavaScript-fysiikkamoottorin luomista vain kääntämällä Bullet-kirjaston C++ -lähdekoodin. Nimi itsessään siis jo kertoo, mistä ammo.js:ssä on kyse. ammo.js:n GitHub-sivuilla on ohjeet, miten sillä voi ohjelmoida joko C++:lla kääntämällä se Emscriptenin avulla JavaScriptiksi, tai ohjelmoida suoraan JavaScriptillä.

JavaScript-kirjasto three.js:lle on olemassa helppokäyttöinen rajapinta Physijs, joka on tarkoitettu yksinomaan fysiikanmallinnusta varten. Physijs on rakennettu ammo.js-kirjaston päälle. Fysiikkasimulaatiot ajetaan erillisellä haaralla, jotta se ei hidasta kehittäjän sovelluksen suorituskykyä tai 3D-renderöintiä. Physijs-lisäosan ohjelmointisyntaksi on pyritty pitämään three.js-kirjastoa vastaavana, jotta three.js:n käyttäjille ei tulisi uutta opittavaa. Physijs-lisäosan käyttö three.js-kirjaston päällä helpottaa huomattavasti fysiikan mallinnuksen kehitystyötä. Physijs-objektien käytöllä kehittäjä saa automaattisesti dynaamisen ympäristön eikä hänen tarvitse käyttää aikaa esimerkiksi muotojen määrittelyyn, objektien pitämiseen oikeilla paikoilla tai törmäyksen tunnistukseen. Physijs-lisäosan ominaisuustarjontaan kuuluu muun muassa tuki useammalle objektimuodolle, yksinkertainen kitkan hallinta ja sisäänrakennettu törmäyksentunnistus ja tapahtumakäsittelyt. Tulevaisuudessa olisi luvassa muun muassa fysiikan mallinnusjärjestelmä ajoneuvoille ja lisää tarkempaa optimointia. (Physijs 2012.)

Box2D-sivuilta (2012) käy ilmi, että 2d-peleille on tarjolla fysiikkamoottori nimeltä Box2D, jolla ohjelmointi tapahtuu C++:lla. JavaScript-kehittäjien iloksi siitä on tehty useita JavaScript-käännöksiä, joita ovat muun muassa Box2DJS ja box2dweb. box2dweb- (2012) ja Box2DJS-verkkosivuilta (2008) selviää, että itse asiassa kumpi-

kaan näistä ei ole suora käännös Box2D:stä, vaan molemmat ovat käännöksiä Flash-käännös Box2DFlash:sta. Box2D-moottorin tarjoamiin ominaisuuksiin kuuluu muun muassa törmäyksentunnistus ja niin sanottu jäykkä kehon dynamiikka. Objekteille voi antaa monenlaisia ominaisuuksia, kuten esimerkiksi kitkaa, painovoimaa ja määritellä erilaisia sidekohtia. (Box2D 2012.) Mainitut JavaScript-käännökset eroavat toisistaan vain sillä, että Box2DJS on riippuvainen Prototype-kirjastosta (Box2DJS 2008). Pienemmän tiedostokoonsa ansiosta box2dweb saattaisi siis olla tehokkaampi valinta.

Paljon huomiota kerännyt 3D-pelimoottori HTML5-peleille on PlayCanvas, jota on ollut kehittämässä kovan luokan pelikehittäjiä. PlayCanvasin perustaja on Will Eastcott, veteraanipeliohjelmoija, joka on ollut töissä Activisionilla ja ollut mukana tekemässä Call of Duty, Grand Theft Auto ja Max Payne -pelejä. Dave Evans oli perustamisessa mukana, ja hän on ollut perustajajäseniä myös PlayStation Homelle Sonyn Lontoon toimipisteessä. PlayCanvasin pelikehitys tapahtuu näppärästi selaimessa. PlayCanvas käyttää WebGL:ää, joten se tukee WebGL:ää käyttäviä selaimia, joihin Internet Explorer ei kuulu. (PlayCanvas 2012.) PlayCanvas-pelimoottorilla tehty demopeli ”D.E.M.O. 3rd Person Shooter” näkyy kuvasta 7. Kuvan 7 demopeli on suhteellisen vaikuttava, ottaen huomioon, että sitä pystyy pelaamaan suoraan selaimessa.

KUVA 7. D.E.M.O. 3rd Person Shooter (PlayCanvas 2012)

PlayCanvasilla ohjelmointi tapahtuu JavaScriptillä, joka ajetaan suoraan selaimessa. Ohjelmointi tapahtuu kanssakehittäjien kanssa reaaliajassa, joten he näkevät toisten tekemät muutokset heti. Tätä varten PlayCanvasissa on tallennuskansioita versionhallintaa varten, jotta kehittäjien tekemät muutokset eivät mene ristiin ja aiheuta konflikteja. PlayCanvasin sisäänrakennettujen ominaisuuksien tarjonta on laaja, johon kuuluu esimerkiksi dynaamiset valaistukset sekä olio- ja komponenttijärjestelmä. PlayCanvas tukee suurimpia 3D-mallinnusohjelmia, kuten Mayaa, 3ds Maxia ja Blenderiä. (PlayCanvas 2012.)

Yksinomaan älypuhelimien ja tablettien kosketusnäyttöjä varten kehitetty jQuery Mobile on JavaScript-kirjasto, joka pohjautuu jQuery- ja jQuery UI-kirjastoihin (User Interface). jQuery Mobile on suunniteltu helposti käytettäväksi työvälineeksi rakentaa mobiililaitteilla selattavia verkkosivuja. Kirjasto tukee laajaa valikoimaa käyttöjärjestelmiä, joihin lukeutuu muun muassa iOS, Windows Phone, Android, BlackBerry, Symbian ja Meego. Tätä varten jQuery Mobile on rakennettu standardin HTML-merkintäkielen pohjalta. jQuery Mobile -kirjastoon kuuluu Ajax-navigaatiojärjestelmä, joka tarjoaa animoidut sivulta siirtymiset ja valmiit käyttöliittymän rakennuspalikat, kuten lomake-elementit, ikonit ja työkalurivit. Tehokkaan suorituskyvyn takaamiseksi jQuery Mobile on pienikokoinen ja modulaarinen, jotta kehittäjä voi sisällyttää projektiinsa vain tarvitsemansa kokonaisuuden. (jQuery Mobile 2012.)

On olemassa muitakin varteenotettavia vaihtoehtoja JavaScript-pohjaiseen käyttöliittymäsuunnitteluun mobiililaitteille. Lisäosana jQuerylle on jQTouch, joka on tarkoitettu mobiilipalveluiden kehittämiseen iPhonelle ja Android-laitteille. jQTouch on riippuvainen jQuery-kirjastosta, joten tiedostokoot ovat suuria vähentäen suorituskykyä. jQuery sisältää myös turhaa painoa mobiililaitteita ajatellen, koska jQuery-kirjastosta löytyy tukea vanhoille selainten versioille, joita ei uusissa älypuhelimissa ja tableteissa näy. jQTouchin kehittäjä, David Kaneda, on julkaissut myös toisen JavaScript-kirjaston mobiilisovellusten kehittämiseen, Sencha Touch-kirjaston. Sencha Touch ei ole riippuvainen jQuerysta, joten sen koko on pienempi, mutta ominaisuustarjonta on kuitenkin laaja. Kehitys Sencha Touch:lla tapahtuu samoille laitteille, kuin jQTouch-kirjastolla. Yksinomaan iPhone-kehitystä varten on olemassa iWebKit, joka on tehty niin yksinkertaiseksi, että täysin vailla ohjelmointitaustaa olevat henkilöt voivat kehittää ammattimaisen näköisiä verkkosivuja. (Bai 2011, 9–14.) Näihin edellä

mainittuihin JavaScript-kirjastoihin verrattuna jQuery Mobile on siis huomattavasti yleiskäyttöisempi jo senkin takia, että sen laitetukivalikoima on paljon laajempi.

4 HAHMOIKKUNAN TOTEUTUS ERI JAVASCRIPT-KIRJASTOILLA

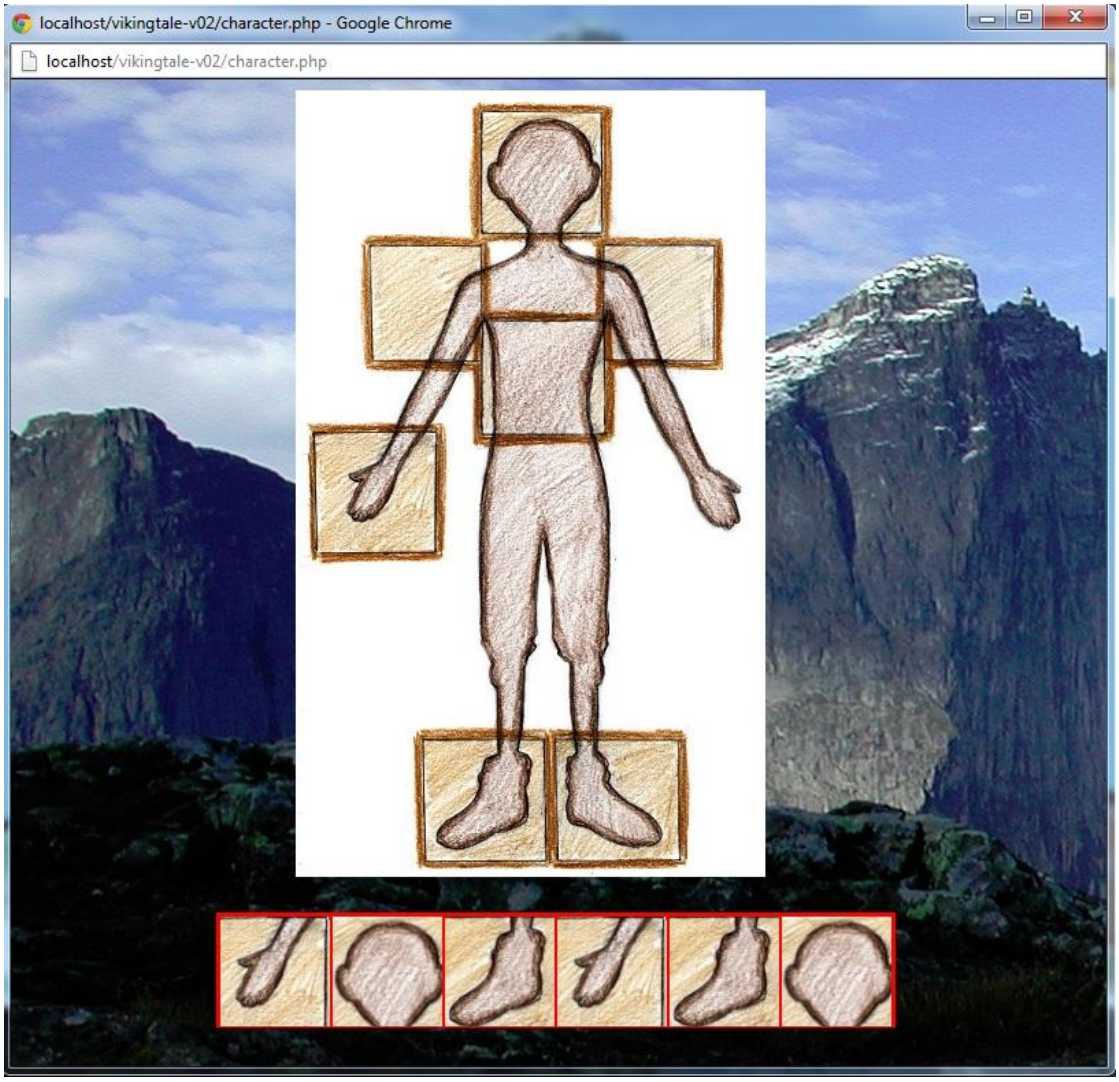
Tein opiskelutoverini Jere Karppisen kanssa selaimella pelattavaa roolipeliä nimeltään Viking Tale, jossa tarkoituksena on taistella vihollisten kanssa ja kerätä parempia varusteita. Keksimme pelin idean ja maailman kokonaan itse, eikä Viking Tale -pelillemme ole asettajaa. Ryhdyimme tekemään yhteistä peliä, koska kumpaakin kiinnosti kokemus pelin toteuttamisesta.

Viking Tale -pelissä pelaaja ohjaa yhtä hahmoa pelkästään hiirtä käyttäen. Peli-ikkunassa on painettavia kohtia, joista syntyy tapahtumia. Viking Tale -pelin pääjuonena on käydä noutamassa pelin hahmon isän haamulle suuri riimukivi. Matka riimukivelle on kuitenkin pitkä ja hahmon täytyy suorittaa lukuisia pienempiä tehtäviä. Peliin sisältyy vihollisten kanssa taistelemista, tavaroiden keräämistä ja myymistä. Suunnittelimme lisäävämme peliin aivopähkinöitäkin, mutta nykyisessä versiossa niitä ei vielä ole. Viking Tale on tällä hetkellä demoasteella.

Päätimme käyttää Viking Tale -pelin toteutukseen meille tuttuja tekniikoita; PHP:tä, MySQL:ää, HTML:ää, CSS:ää ja JavaScript-kirjasto jQuerya. Meitä kiinnosti nähdä, miten osaamamme tekniikat soveltuisivat selainpelin toteutukseen. Minua kiinnosti pelillisten toiminnallisuuksien toteutus jQuery-kirjastolla, koska osasin oikeastaan vain sen perusteet ja halusin syventää osaamistani. Hahmoikkunan rakenteelliseen toteutukseen valitsin DOM-rakenteen HTML5:n canvas-piirtoalustan sijasta. DOM:n käsittely on minulle tuttua ja HTML5:n piirtoalustaa emme halunneet käyttää, koska pelin tekohetkellä kaikki selainversiot eivät tukeneet sitä ja halusimme pelimme olevan saatavilla kaikille.

Opinnäytetyöni käyttötapaukseksi valitsin pelistämme yhden kokonaisuuden, joka on toteutettu perusteellisesti jQuery-kirjastolla. Pelissämme on käytetty paljon PHP-ohjelmointikieltä toiminnallisuuksiin ja selkeästi jQuery-kirjastolla toteutettuja ominaisuuksia on vain pari. Valitsemani ominaisuus on hahmoikkuna, jossa pelihahmolle voi asettaa varusteita tai ottaa niitä pois. Hahmoikkuna näkyy kuvassa 8. Varusteiden asettaminen tapahtuu vetämällä hiirellä hahmon kantamien tavaroiden joukosta varus-

te sitä vastaavalle kohdalle hahmon päällä. Esimerkiksi jos pelaaja haluaa vaihtaa kypärää, hän vetää varustevalikoimastaan kypärän kuvan hahmonsa kypärän päälle, jolloin varusteet vaihtavat paikkoja.



KUVA 8. Hahmoikkuna

Tässä luvussa kerron tämän hahmoikkunan toteutuksesta jQuery-kirjastolla ja vertaan sitä Crafty-pelikirjastolla toteutettuun versioon. Onnistuin saamaan kummankin kirjaston hahmoikkunoista kuvan 8 mukaiset. Valitsin Crafty-kirjaston siitä syystä, että se vaikutti niin hyvin dokumentoidulta. Dokumentti on hyvin organisoitu ja sisältää selkeät ohjeet sekä havainnolliset esimerkit. Lisäksi Crafty-kirjasto on itsenäinen, se ei ole riippuvainen muista sovelluksista tai palveluista, ja se mahdollistaa monipuolisen ohjelmoinnin.

4.1 jQuery-toteutus

Hahmoikkunaan toteutukseen tarvitsin kahta kirjastoa, jQuerya ja jQuery UI:tä. jQuery UI -kirjasto helpotti varusteiden vedä ja pudota -toteutusta sen valmiiksi tarjoamilla draggable- ja droppable-metodeilla. Varusteiden liikutteluun riittää pelkkä jQuery UI, mutta tarvitsin jQuery-kirjastoa Ajax-käsittelyyn. Ajax käsittelystä tulee paljon yksinkertaisempaa jQuery-kirjaston avulla kuin pelkällä JavaScript-ohjelmoinnilla. Elementtien vetämiseen ja pudottamiseen löysin hyvät ohjeet jQuery UI:n (2012) dokumentaatiosta. jQuery UI-verkkosivujen keskustelupalstat auttoivat myös paljon vedä ja pudota -toiminnallisuuksien kehittämisessä

Ominaisuuden toteutuksessa kesti hyvän aikaa. JavaScript- tai jQuery-ohjelmoinnista minulla ei kuitenkaan ollut juurikaan kokemusta ennen opinnäytetyön aloittamista. Olin tietoinen HTML5:n tarjoamasta vedä ja pudota -ominaisuudesta, mutta halusin välttämättä tehdä sen jQuery UI:lla, koska olin niin kiinnostunut jQuery-kirjastoilla ohjelmoinnista.

4.1.1 Varusteiden vaihto

Aluksi määrittelen vetämiselle ja pudottamiselle asetukset, jotka löytyvät kuvasta 9 options-muuttujasta. Asetukset ovat hyvin yksinkertaiset. Määritän src-muuttujan vedettävän kohteen isäntäelementiksi ja mikäli elementin vetäminen lopetetaan, se palaa takaisin paikalleen revert:true -käskyllä. Liitän vetämisen ja sille määritellyt asetukset haluamiini elementteihin yksinkertaisesti valitsemalla elementit luokan perusteella jQueryn $-valitsimella, ja lisäämällä niihin draggable(options)-komennon.

```
src = null;
options = {
    revert:true,
    start: function() {
            src = $(this).parent();
    }
};
//Itemeiden liikuttelu
$(".gearitem").draggable(options);
$(".invitem").draggable(options);
```

KUVA 9. Alkumäärittelyt

Hahmoikkuna saa sisältönsä PHP-tiedoston luokan metodista, joka tuottaa sisällön HTML-elementeiksi. Raahattavat elementit ovat div-elementtejä. jQuery UI-dokumentaatiota selaillessani sain selville, että olisi helpointa sijoittaa raahattavat elementit pudotettavien elementtien sisään. Tein siis varusteista raahattavia (draggable) elementtejä ja sijainneista pudotettavia (droppable) kohtia.

Jotta pelaaja voi asettaa oikean varusteen oikeaan paikkaan, esimerkiksi saappaat menevät vain jalkoihin eikä niitä voi asettaa muualle, minun täytyi tehdä varusteelementeistä yksilöllisiä. Toteutin tämän sellaisella rakenteella, missä jokaisen varusteen div-elementin arvo (value) on sen varusteluokka. Varusteluokkia pelissämme on 5; 1 on kypärä, 2 on hartiasuojukset, 3 on ase, 4 on housut ja 5 on jalkineet. Kun esimerkiksi pelaaja raahaa varusteen valikoimastaan hahmon päällä olevaan sijaintielementtiin, vertailu tapahtuu raahattavan esineen ja pudotettavan elementin sisällä olevan varusteen arvojen välillä. Haen varusteiden arvot kuvassa 8 näkyvien dragVal- ja dropVal-muuttujiin. jQuery UI -kirjastolla pystyy näppärästi käsittelemään vedettävää elementtiä ui.draggable-komennolla, jota käytän valitsimena (selector), jotta voin hakea vedettävän elementin arvon. Kuvan 8 näkyvän muuttujan dropVal-arvo haetaan pudotettavan sijaintielementin ns. lapsielementistä, eli elementistä, joka on täsmälleen yhden pykälän pudotettavan sijaintielementin sisällä.

```
//Otetaan raahatun value ja dropattavan value
var dragVal = $(ui.draggable).attr("value");
var dropVal = $(this).children().attr("value");

var dragclass = $(ui.draggable).attr("class");

//Jos item käy gearslottiin, niin antaa mennä
//eli saapasta ei voi laittaa päähän, vain päähine käy jne
if (dragVal == dropVal) {
```

KUVA 10. Varusteluokkien vertailu

Noukittuani varusteiden luokka-arvot elementeistä vertaan niitä keskenään kuvassa 10 näkyvässä if-lauseessa. Jos varusteluokat eivät täsmää, raahattu esine palaa yksinkertaisesti paikalleen. Tämä onnistuu varsin helposti asettamalla raahatulle esineelle ominaisuuden draggable({ revert:true }).

Elementtien vetäminen onnistui nopeasti, mutta oikeaan paikkaan pudottamisessa kesti. Toimivan logiikan keksimiseen varusteluokkien tarkistusta varten meni hyvän aikaa. Lisäksi oli suhteellisen hankalaa pitää se toimivana. Hahmoikkunan DOM-rakenne muodostetaan PHP-kielellä tehdyssä Web Servicessä luokan sisällä, missä se muuttui useaan otteeseen ja oli hankalasti luettavissa. Perusrakenne pysyi kuitenkin samana, eli yksilöidyt isäntäelementit, joiden sisällä on lapsielementtinä itse varuste, jota voidaan vedellä ympäri hahmoikkunaa.

```
var k=1;
for(k;k <= 6;k++)
{
//Inventory slottiin pudottaminen
$(".item_container"+k).droppable({
    drop: function(event, ui) {


        //Otetaan raahatun parentti
        var dragParent = $(ui.draggable).parent().attr("class");
        var dropParent = $(this).attr("class");

        //Jos raahattavan parentin class ja pudotettavan parentin
        //class on sama, niin mennään pois koko funktiosta
        if (dragParent == dropParent) {
            return false;
        }
```

KUVA 11. Varusteen pudottaminen ja virheen tarkistus

Kuvassa 11 näkyy, miten käyn jokaisen item_container-luokkaisen div-elementin läpi for-lauseella ja asetan niille pudotusmahdollisuuden droppable-määrittelyllä. Nämä item_container-elementit ovat siis sijaintielementtejä, joihin voi pudottaa varusteen säilöttäväksi. Varustevalikoimassa on kaikkiaan 6 item_container-luokkaista elementtiä. Olen tehnyt jokaisesta item_container-elementistä yksilöllisen numeroilla, koska ilman sitä tulisi virhe jos pelaaja vetäisi varustetta, mutta pudottaisikin sen takaisin omalle paikalleen. Tämä virheen tarkistus näkyy if-lauseena kuvassa 11. Otan raahatun elementin alkuperäisen isäntäelementin luokka-arvon (class) muuttujaan dragParent ja vertaan sitä pudotettavan sijaintielementin luokkaan, jonka määritän dropParent-muuttujaan. Jos ne täsmäävät, eli raahattu elementti on pudotettu samalle paikalle, mistä sitä lähettiin raahaamaan, niin funktion ajo keskeytyy return false -komennolla. Tämä tarkistus on pakollinen, koska ilman sitä varusteita tulisi kaksi sen isäntäelementin sisään, mistä varustetta lähettiin raahaamaan. Tämä johtuu siitä, että raahatun esineen pudottaminen ja vaihto paikalla olleen esineen kanssa tapahtuu poistamalla raahattu esine alkuperäisestä sijainnistaan ja sijoittamalla se uudelle paikalleen. Samalla varuste, jonka tilalle raahattu esine tuotiin, siirretään raahatun esineen alkuperäiselle paikalle. Tähän käytetty koodi näkyy kuvasta12. Tämä ongelma tuli esille vasta usean tunnin testauksen jälkeen. Virheen huomattuani itse virheen tarkistuksen luomiseen ei mennyt kauaa aikaa.

Kun varusteiden vaihto menee sääntöjen mukaan, selain ajaa kuvassa 12 näkyvän koodin. Kyseessä olevassa vaihdossa hahmon päältä on raahattu varuste valikoimaan. Kuvan 12 näkyvän src.append-metodin (lähteeseen lisäys) sisällä otetaan paikalla oleva varuste, poistetaan se remove() -komennolla, kloonataan, poistetaan sen luokka ja asetetaan uudeksi luokaksi gearitem, joka on sama luokka kuin jokaisella hahmon päällä olevalla varusteella. Luokkia vaihdetaan keskenään CSS-tyylien vuoksi. Lopuksi ”uudelle” varusteelle annetaan vielä raahausominaisuus draggable-komennolla. Tämän jälkeen raahattu esine poistetaan edelliseltä paikaltaan ja lisätään uuteen paikkaansa uudella luokalla, invitem, joka vastaa varustevalikoimassa olevien luokkien nimeä. Tähän koodin pätkään löysin apuja jQuery UI:n keskustelupalstoilta.

```
//Poistetaan invitem ja vaihdetaan paikka raahatun
//itemin tilalle
src.append(
    $('.invitem', this).remove().clone()
    .removeClass().addClass("gearitem")
    .css({"left": '', "opacity": '',"top":''})
    .draggable(options)
);
//Poistetaan raahattu gearitem entiseltä paikaltaan
//ja laitetaan invitemin tilalle
$(this).append(
    ui.draggable.remove().clone()
    .removeClass().addClass("invitem")
    .css({"left": '', "opacity": '',"top":''})
    .draggable(options)
);
```

KUVA 12. Varusteiden vaihto

Minun täytyi tehdä sekä varustevalikoiman että hahmon varusteiden isäntäelementeille omat drop-funktionsa. Funktiot ovat lähestulkoon samanlaiset, luokkien nimet vain vaihtuvat eivätkä DOM-rakenteet ole täysin samanlaisia. Minun täytyi tehdä kummallekin omat funktionsa, koska pelaaja voi haluta vain järjestellä valikoimansa varusteita haluamaansa järjestykseen. Halusimme antaa tämän mahdollisuuden pelaajille. Hahmon päällä olevia varusteita ei kuitenkaan voi järjestellä, koska tiettyyn sijaintiin käy vain tietty varusteluokka. Hahmon varustevalikoiman järjestelyn koodi löytyy kuvasta 13. Idea on sama kuin kuvan 10 koodissa, mutta molempien elementtien luokka pysyy samana.

```
//Jos käyttäjä haluaa järjestellä inventorynsa itemeitä
else if (dragclass == "invitem ui-draggable ui-draggable-dragging") {

    src.append(
    $('.invitem', this).remove().clone()
    .removeClass().addClass("invitem")
    .css({"left": '', "opacity": '',"top":''})
    .draggable(options)
    );
    //Poistetaan raahattu gearitem entiseltä paikaltaan
    //ja laitetaan invitemin tilalle
    $(this).append(
    ui.draggable.remove().clone()
    .removeClass().addClass("invitem")
    .css({"left": '', "opacity": '',"top":''})
    .draggable(options)
    );
```

KUVA 13. Varusteiden järjestely

Kaiken kaikkiaan pelkkään elementtien vetämisen ja pudottamisen toteutukseen ei mennyt niin kauaa aikaa, mutta ominaisuuden valmiiksi saamiseen meni monta päivää, jopa viikkoja. Määritin toiminnallisuuden olevan valmis, kun elementtejä pystyi vetämään pitkin ikkunaa ja pudottamaan oikealle sijainnille ilman virheitä. Onneksi vastaan ei tullut muita virhetoiminnallisuuksia, kuin esineen kaksinkertaistuminen, kun esineen pudotti samalle paikalle, mistä sitä lähti raahaamaan.

4.1.2 Ajax-toteutus

Vaikka jQuery-kirjaston käyttöönotto yksinkertaistaa ja nopeuttaa Ajax-toimintojen toteuttamista, minulla oli suuria hankaluuksia saada tietojen tallentaminen SQL-tietokantaan toimimaan hahmoikkunan sulkeutuessa. Tavoite oli, että kun pelaaja sulkee hahmoikkunan, jQuery käy erikseen läpi hahmon päällä olevat varusteet ja valikoimassa olevat, lähettää ne PHP-kielellä ohjelmoituun Web Serviceen, missä data tallennetaan tietokantaan. Hahmon päällä olevat ja valikoimassa olevat tallentuvat eri tauluihin tietokannassa. Varusteista tallentuu vain esineen yksilöllinen id. Hahmon varusteisiin pelaajan yhdistää pelaajan id.

Ajatus tietojen tallentamisesta ikkunan sulkeutuessa tuntui haastavalta ja halusin ehdottomasti toteuttaa sen. En halunnut missään vaiheessa toteuttaa tietojen tallennusta napin painalluksesta, koska halusin käyttöliittymän olevan yksinkertainen ja mutka-

ton. Halusin tehdä hahmoikkunan niin, että käyttäjän ei tarvitse tehdä muuta kuin vaihdella varusteita mielensä mukaan, sulkea hahmoikkuna ja tiedot tallentuvat.

Päädyin käyttämään jQueryn bind()-metodiin (sidonta) beforeunload-funktiota (ennen sulkeutumista) sitoakseni toiminnallisuuden hahmoikkunan sulkemiseen. Kyseinen koodirivi jQuery-kirjastolla luotuna näyttää tältä: $(window).bind('beforeunload', function(){)}. Testasin funktion toimivuutta lisäämällä sen sisään return-komennon, joka tulostaa pienen varmistusikkunan ruutuun, kun ikkunaa yrittää sulkea. Ikkuna kysyy, haluaako käyttäjä varmasti poistua sivulta. Funktio osoittautui toimivaksi. Suunta oli siis oikea, mutta tie määränpäähän osoittautui pitkäksi ja tuskalliseksi.

Yksi ongelma Ajax-ajon kanssa oli päättää, millaisessa muodossa tieto lähetetään Web Serviceen. Dataan kuului viisi hahmon päällä olevaa varustetta ja kuusi valikoimassa olevaa. Lisäksi datasta täytyi tulla ilmi, mihin varusteluokkaan jokainen päällä oleva varuste kuuluu. Tällainen määrä dataa olisi fiksua lähettää taulukossa, mutta taulun lähettämisestä Ajax-tekniikalla minulla ei ollut kokemusta. Lisäksi jQuery-kirjastolla luodun datarakenteen tarkistaminen tuntui tässä tapauksessa olevan lähes-tulkoon mahdotonta. Pystyin tarkastamaan, mitä arvoja taulu sisälsi, mutta eniten minua kiinnosti tietää, millaisessa muodossa tieto tarkalleen menee PHP Web Service -tiedostoon. Kun tiedän millainen rakenne on, tiedän miten sitä voi käsitellä. Tämän ongelman kanssa taistelin todella pitkään. Jo siihen meni hyvän aikaa, ennen kuin sain tietopaketin lähetettyä Web Serviceen. Lopulta kysyin Mikkelin ammattikorkeakoulun lehtori Janne Turuselta apua ja ratkaisu oli yksinkertaisempi kuin ajattelin. Jos lähettää Ajax-tekniikalla JSON-tyyppisen taulun toiseen palveluun, joka on muodostettu PHP-kielellä, voi käyttää suoraan PHP:stä löytyvää json_decode-metodia JSON-taulun muuttamiseen PHP-muotoon. Lehtori Turusen antamasta tiedon murusesta viisastuneena tiedon tallentaminen lähti kulkemaan nopeasti eteenpäin.

Ennen Ajax-ajoa määritän muuttujiin URL:n Web Serviceen ja taulukot hahmon päällä oleville varusteille ja varustevalikoimassa oleville. Tämän jälkeen käyn for-lauseella päällä olevat varusteet läpi ja tallennan tauluun varusteen id:n ja varusteluokan. Sama toimenpide seuraa myös varustevalikoimalle. Varusteen yksilöivän id:n olen tallentanut raahattavan varusteen div-elementin id-arvoon ja varusteluokan taas saman elementin value-arvoon. Kuvasta 12 näkyy kirjoittamani koodi for-lauseille.

```
//Otetaan item_id ja item_class_id ylös
for (var piece in gearPieces) {
    item_id = $('.'+gearPieces[piece]).children(".gearitem").attr("id")
    + ","+$('.'+gearPieces[piece]).children(".gearitem").attr("value");
    gear_stats.push(item_id);
}
for (var item in inventorySlots) {
    item_id = $('.'+inventorySlots[item]).children(".invitem").attr("id")
    + ","+$('.'+inventorySlots[item]).children(".invitem").attr("value");
    inventory.push(item_id);
}
```

KUVA 14. For-lause varustetaulukoille

Kuvasa 14 näkyvässä gearPieces-taulukossa ovat hahmon päällä olevien div-elementtien luokkien nimet ja inventorySlots-taulukossa ovat valikoimassa olevien div-elementtien luokkien nimet. For-lause käy jokaisen taulun arvon läpi. Eli yhden kierroksen aikana koodi hakee DOM-rakenteesta vuorossa olevan luokan nimisen elementin, ottaa talteen kyseessä olevan elementin id:n ja arvon, sijoittaa sen item_id-muuttujaan ja laittaa väliin pilkun. Päätin käyttää pilkkua erottimena, koska sen voi helposti purkaa Web Servicessä käyttämällä esimerkiksi PHP:n explode-komentoa, jonka avulla voi jakaa merkkijonon taulukoksi käyttämällä jotain tiettyä merkkiä erottimena. Kun data on poimittu, se asetetaan push-komennolla taulukkoon, joka myöhemmin lähetetään Ajax-tekniikalla toiseen paikkaan tallennettavaksi.

```
gear_stats = JSON.stringify(gear_stats);
inventory = JSON.stringify(inventory);
$.ajax({
    url: urli,
    async: false,
    type: 'POST',
    traditional: true,
    dataType: 'json',
    data: {'gear_stats' : gear_stats, 'inventory' : inventory}
});
```

KUVA 15. Ajax-lähetys

Kuvassa 15 muutan molemmat taulukot tekstijonoksi (string) JSON-formaattiin (JavaScript Object Notation) JSON.stringify-metodilla. Näin Web Servicessä voin

PHP:llä muuntaa JSON-datan taas taulukkomuotoon, jossa dataa on minulle helpompi käsitellä. Stringify-metodi tekee taulukosta JSON-tekstiä, joka kulkee helposti URL:ssa parametreina WebServiceen. Tässä tapauksessa gear_stats taulukko näyttäisi JSON-tekstinä esimerkiksi tältä: { ”1,1”, ”2,2”, ”9,3”, ”7,4”, 10,5” }. Ensimmäinen luku lainausmerkkien sisässä on varusteen yksilöllinen id ja pilkun jälkeinen on varusteen luokka. Lainausmerkkien jälkeinen pilkku tarkoittaa JSON-formaatissa, että on seuraavan datan vuoro.

Kuvasta 15 näkyy, miten yksinkertaista ja helposti ymmärrettäviä Ajax-käsittelyt jQuery-kirjastolla ovat. JavaScriptillä samainen koodi olisi monenkertaisesti pitempi. Ensin jQuery-kirjastolla selvennetään, että Ajax-käsittely on tulossa $.ajax{()}-komennolla, jonka sisään voidaan määritellä ominaisuuksia. Parametrivalikoima Ajax-käsittelyille on hyvin laaja, mutta tähän Ajax-käsittelyyn olen lisännyt vain tarpeellisiksi katsomani parametrit. URL-määritelmään tulee luonnollisesti polku, mihin dataa lähetetään. Kuvassa 15 näkyvä ominaisuus async: false tarkoittaa asynkronisuuden laittamista pois päältä. Tein tämän varmistukseksi, että Ajax-ajo menee varmasti läpi ennen kuin mitään muuta tapahtuu, eli tässä tapauksessa ikkuna sulkeutuu ja tietokanta-ajo jäisi mahdollisesti tekemättä. Ajax-kutsulle täytyy määritellä sen tyyppi, joka on tässä tapauksessa tietojen tallennus eli post. Post-tyyppiä käytetään usein myös tietojen päivitykseen tai poistamiseen. Näitä varten voi käyttää niille tarkoitettuja http-tyyppejä, put ja delete, mutta kaikki selaimet eivät tue niitä. Tyypiksi voi määritellä myös tietojen haun, get. Parametri dataTypellä määritellään datan tyyppi ja data-parametriin tulee lähetettävä data. Olen kirjoittanut lähetettävän datan JSON-muotoon, kuten kuvasta 15 näkyy. JSON-formaatissa data näyttäisi lopuksi esimerkiksi tältä: {”gear_stats” : { ”1,1”, ”2,2”, ”9,3”, ”7,4”, 10,5” }, ”inventory” : { ”11,1”, ”21,2”, ”14,3”, ”5,4”, 14,5” }. Kun selain on käsitellyt jQuery-skriptin, esimerkin mukainen data lähetetään URL:ssa Web Serviceen, jossa JSON-data muunnetaan taas taulukoksi ja tallennetaan tietokantaan.

4.2 Crafty toteutus

Crafty-pelikirjaston saa näppärästi käyttöönsä lataamalla sen suoraan Craftyn palvelimelta: <script type="text/javascript" src="http://cdn.craftycomponents.com/crafty-release.js"></script>. Tässä on sekä hyviä että huonoja puolia. Hyvä puoli on, ettei kehittäjän tarvitse ladata kirjastoa konkreettiseksi kopioksi omaan projektiinsa. Huono

puoli taas on, että jos Craftyn palvelin kaatuu, ei kehittäjän peli toimi. Koska käytän Crafty-kirjastoa vain testaamiseen enkä julkaise lopputulosta kaiken kansan nähtäväksi, päätin käyttää Craftyn palvelimella olevaa kirjastoa.

Crafty-pelikirjastolla on jQuery UI-kirjaston tapaan valmis komponentti elementtien vetämiselle, mutta tällä hetkellä se on vasta testausvaiheessa eikä ole loppuun asti valmis. Päätin käyttää Crafty-kirjastoon sisään rakennettua törmäyksen tunnistusta varusteiden vaihtoon. Elementtien vetämisen ja pudottamisen mahdollisuus ei ole minulle niin tärkeä, että vaatisin ehdottomasti varusteiden vaihdon tapahtuvan aina sillä tavoin. Minulle käy ihan yhtä hyvin se, että varusteet vaihtuvat niiden törmätessä. Päätin kuitenkin jättää jQuery-versiossa olleen varustevalikoiman järjestelemisen sikseen, koska hahmoikkunan käytettävyydestä olisi tullut sekava. Heti kun varustevalikoimasta vedetty varuste hipaiseekin toista varustevalikoiman esinettä, ne vaihtuisivat ja tästä voisi seurata hektistä varusteiden paikkojen vaihtoa.

4.2.1 Olioiden luonti

Ensiksi peliympäristö luodaan Crafty.init() -komennolla. Lyhenne init tarkoittaa englanninkielistä sanaa initialization, joka on suomeksi käynnistys. Sulkujen sisälle sijoitetaan pelitilan leveys ja korkeus. Tämä pelitilan käynnistys luo div-elementtikehyksen, jonka sisällä olevassa div-elementissä peli pyörii.

Crafty-pelikirjastolla ohjelmoinnissa avainsanat ovat oliot ja komponentit. Kaikki pelin osaset ovat olioita, joille voi määrittää toimintoja komponenteilla. Ymmärsin, että loogisinta on tehdä raahattavista varusteista olioita. Tästä johtuen en voinut käyttää vanhaa toteutustapaa, jossa PHP-luokasta välitetään iso möhkäle HTML-merkintäkieltä. Tarvitsemani data, varusteiden id:t ja varusteluokat, on HTML:ssä olevien elementtien sisällä ja olisi turhan hankalaa kaivaa tarvitsemani tieto läjästä HTML-kieltä. Päätin toteuttaa datan välitysmuodon paremmin Crafty-pelikirjastolle sopivaksi ja kuljettaa datan pelkässä yksinkertaisessa tekstimuodossa. Suunnittelin käsitteleväni datan hahmoikkunassa JavaScriptillä ja sijoittaa dataa elementtien sisälle. PHP-luokasta tulevilla HTML-elementeillä on kirjoitetut tyylit CSS-tiedostossa, mutta voin kirjoittaa tyylit uudestaan elementeille Crafty-kirjastolla.

```
//Luodaan hahmon kuva peliin
var gearKuva = Crafty.e("2D, DOM, Image")
.attr({x: 220, y: 10})
.image("testikuvat/character.jpg");
```

KUVA 16. Hahmon kuva oliona

Kuvassa 16 näkyy Crafty-toteutukseni ensimmäinen olio, kuva hahmosta, jonka päälle varusteita asetetaan. Crafty-kirjastolla olioiden luonti tapahtuu Crafty.e()-metodilla. Ensiksi määritän, mitä komponentteja olioon haluan sisällyttää. Tässä tapauksessa niitä on vain kolme; 2D, DOM ja Image, eli kuva. Nämä komponentit ovat Crafty-kirjastossa sisään rakennettuja ja niihin liittyy omia asetuksiaan. Seuraavaksi määritän attr({})-määritelmän sisälle x- ja y-arvot, ja image()-asetukseen tiedostopolun kuvaan.

Crafty-pelikirjastossa Image-komponentille on määritelty oletukseksi, että jos kuvalle ei ole asetettu leveyttä ja korkeutta, ja kuvalle on asetettu no-repeat -käsky (ei toistoa), kuva näkyy alkuperäiskoossa. Tämä sopii kätevästi minulle, koska käyttämäni kuvan alkuperäiskoko on jo sopiva. Huomasin kuitenkin kokeillessani kuvan asetuksia, että jos kuvalle ei erikseen määritä toistoa (repeat), ei voi määritellä kuvalle haluamaansa leveyttä tai korkeuttakaan. Tämä on outoa ja selkeä haittapuoli. Jos käyttäjä haluaa näyttää kuvan vain kerran (no-repeat) ja määritellä itse kuvan mittasuhteet, jos kuva vaikka on liian iso, niin pitäisihän hänen siihen ilman muuta pystyä.

```
//Luokasta saatava data
var data = "<?php echo $_SESSION["GameEngine"]->CharacterWindow();?>";

var taulujako = data.split("|");

//Muodostetaan gear ja inventory taulut
var gear = taulujako[0].split(";");
var inv = taulujako[1].split(";");
```

KUVA 17. Datan jako tauluihin

Kuvassa 17 sekoitan hieman PHP-kieltä JavaScriptin kanssa, missä pitää olla välillä tarkkana. JavaScript kun on dynaaminen ohjelmointikieli, kun taas PHP-ajetaan vain

sen yhden kerran, kun sivulle tullaan. Kuvan 17 PHP-koodinpätkä ajaa sessiomuuttujasta pelin luokan funktion läpi, mikä palauttaa hahmon varusteet. Koska ne tarvitsee hakea vain sen kerran, kun hahmoikkuna avataan, voin turvallisesti käyttää siihen PHP-kieltä. Haettuani datan, joka on yksinkertaisessa tekstimuodossa, jaan sen kahteen arvoon taulussa. Datan jakamiseen käytän JavaScript-metodia split(), jonka sisälle määritellään, mistä merkistä jako tapahtuu. Web Servicestä tulevassa datassa olen sijoittanut |-merkin hahmon päällä olevien ja valikoimassa olevien varusteiden väliin, joten ensimmäinen taulukkojako tapahtuu kyseisen merkin kohdalta. Muuttujassa taulujako on siis kaksi arvoa, joista toinen sisältää varusteet, jotka ovat hahmon päällä, ja toinen sisältää ne, jotka ovat valikoimassa. Tämän jälkeen käytän split-metodia taulujako-muuttujan molempiin arvoihin, käyttäen erottimena puolipistettä. Puolipisteen olen sijoittanut jokaisen tietokantarivin loppuun. Toisin sanottuna puolipiste jakaa jokaisen varusteen tiedot keskenään. Näin minulla on gear- ja inv-nimisissä taulukoissa arvoja, joista jokainen on oma varusteensa. Varusteen tiedot (id, varusteluokka id, nimi ja voima) olen erottanut pilkulla, johon käytän myös split-metodia myöhemmin. Varusteiden datan käsittelyn ideoimiseen minulta ei mennyt kauaa aikaa. Taulukoiden käsittely on minulle tuttua ja ymmärsin sen olevan kätevä ratkaisu tässä tapauksessa.

Kaikki hahmoikkunan varusteet ovat olioita. Tein niiden luomiselle omat käsittelynsä for-lauseisiin, joissa käyn läpi gear- ja inv-taulujen arvot. Olioiden luominen onnistui helposti, se on tehty yksinkertaiseksi Craftyssa. Olioiden vetäminenkin onnistuu Crafty-kirjastossa jo valmiina olevan Draggable-komponentin ansiosta. Pelkästään liittämällä Draggable-komponentti olioon, sitä voi vedellä ympäri Crafty-pelitilaa. Kuvasta 18 näkyy for-lause inv-taulukon käsittelylle. For-lause käy läpi jokaisen arvon taulukossa. Heti ensimmäiseksi käytän jälleen split-metodia jakamaan taulukon arvo uuteen tauluun, jonka arvoina ovat varusteen id, luokan id, nimi ja vahvuus. Oikeastaan välttämättömimmät ovat vain id ja luokan id. Varusteluokan id:tä tarvitsen varusteiden vertailuun, ja molempia id:tä tarvitsen tietokantaan tallennusta varten.

```
var muuttujaX = 180;
for (var varuste in inv) {
    var datapala = inv[varuste].split(",");

    var items = Crafty.e("2D, DOM, Text, Image, Draggable, Collision, Item, Vaihto")
    .image("testikuvat/items/class_id_"+datapala[1]+".jpg")
    .attr({w: 80, h: 80, x:muuttujaX, y: 500}).collision().text(datapala[2]);
    //Kasvatetaan x-arvoa jotta elementit tulevat vierekkäin
    muuttujaX += 82;
}
```

KUVA 18. Varustevalikoiman olioiden luonti

Kuvasta 18 käy ilmi, että asetan olioille paljonkin komponentteja. Collision-komponentti (törmäys) on varusteiden vaihtoa varten, josta kerron myöhemmin. Törmäykseen liittyy oma Vaihto-komponentti, joka liittyy varusteiden vaihtoon. Päätin hyödyntää Crafty-pelikirjaston olio- ja komponenttijärjestelmää ja suorittaa törmäyksen tunnistuksen ja varusteiden vertailun komponentin sisällä. Tämä on siitä syystä kätevää, että tämän komponentin nimen voi vain julistaa olion komponenttiluettelossa ja sitten ne toimivat yhdessä. Olioon liittämäni Text-komponentin avulla voin esittää oliossa tekstiä. Kuvassa 18 näkyvä Item-komponentti on myös oma komponenttini. Item-komponentille en määrittele mitään asetuksia, käytän sitä vain varusteiden vaihtoa varten.

Kuvassa 18 luodun olion kuvan tiedostopolusta näkyy, miten käytän ensimmäistä kertaa varusteen taulukon arvoa hyödykseni. Jokaiselle varusteelle on oma ikoninsa ja ne kuvat on nimetty niiden luokan mukaisesti; kypäröiden luokka id (class_id) on 1, joten class_id_1.jpg -tiedostonimellä saa kypärän ikonin. Elementin lopussa julistan törmäyksen olion käyttöön ja asetan oliolle tekstiksi varusteen nimen.

Olioiden x- ja y-arvojen määrittämisessä koin vaikeuksia. Varustevalikoiman koordinaatistoarvot muuttuvat vain x-arvon osalta, varusteet tulevat siis vierekkäin samalle tasolle, mutta hahmon päällä olevien varusteiden sijainti vaihtelee suuresti. Suunnittelin jälleen käyttäväni taulukkoa tämänkin ongelman ratkaisemiseen. Sijoitin taulukkoon jokaiselle varusteelle omat x- ja y-arvonsa ja kutsuin niitä taulukosta suoraan olion attr()-määrittelyssä x- ja y-arvoille. Hahmoikkunan lataus kuitenkin hidastui valtavasti eikä sovellus toiminut ollenkaan latauksen jälkeenkään. Oliot kuitenkin menivät määrittelemiini sijainteihin, joten Crafty kyllä sai arvot taulukosta noukittua.

Ihmettelin tätä ongelmaa ja kokeilin kaikenlaista, mutta päätin lopulta luovuttaa taulukon käyttämisen suhteen ja keksiä jotain muuta. Ainakin varustevalikoimalle sijaintien määrittely onnistui helpommin kuin hahmon päällä oleville varusteille. Varustevalikoimalle päätin käyttää tasaisesti kasvavaa muuttujaa. Kuvassa 18 näkyy, miten määritän ennen for-lausetta muuttujaX-arvon. Ensimmäinen arvo on 180, joka annetaan ensimmäiselle varustevalikoiman oliolle. For-lauseen lopussa lisään edelliseen muuttujaX-arvoon 82, eli ensimmäisen kierroksen jälkeen muuttujaX-arvo on 262. Tämän x-koordinaatiston arvon annan toiselle muuttujalle, ja näin muuttujaX aina vain kasvaa. Näin varustevalikoiman oliot tulevat kaikki vierekkäin parin pikselin päähän toisistaan.

```
var gearx;
var geary;
var i=0;

//Luodaan elementit
for (var varuste in gear) {
    var datapala = gear[varuste].split(",");

    //Käsitellään koordinaatistoarvot
    switch(i) {
        case 0:
        gearx = 345;
        geary = 20;
        break;
        case 1:
        gearx = 345;
        geary = 130;
        break;
        case 2:
        gearx = 230;
        geary = 40;
        break;
        case 3:
        gearx = 345;
        geary = 270;
        break;
        case 4:
        gearx = 340;
        geary = 400;
        break;
    }
```

KUVA 19. Switch-lause

Hahmon päällä olevien varusteiden sijainnin päätin määritellä perinteisesti ehtolauseilla, kuten kuva 19 todistaa. Pelkäsin tämänkin kenties hidastavan hahmoikkunan latautumista, mutta onneksi näin ei käynyt. Ennen for-lausetta alustan muuttujat varusteen x- ja y-arvoille sekä kasvavalle numerolle, i-muuttujalle. Numero kasvaa joka kierroksella yhdellä. Koska päällä olevat varusteet tulevat aina tietyssä järjestyksessä, voin käyttää i-muuttujaa varusteen sijainnin määrittelyssä. Esimerkiksi ensimmäinen varuste on aina kypärä, joten kun i on 0 (ensimmäisellä kierroksella), gearx-muuttuja saa arvon 345 ja geary arvon 20. Tämän jälkeen switch-lauseesta poistutaan break-käskyllä. Kuvasta 19 näkyy myös, että suoritan taulun jokaiselle varusteelle split-metodin, kuten kuvan 18 varustevalikoiman varusteille.

```
var gears = Crafty.e("2D, DOM, Text, Image, Draggable, Collision, Gear")
.image("testikuvat/items/class_id_"+datapala[1]+".jpg")
.attr({w: 80, h: 80, x: gearx, y: geary})
.collision().text(datapala[2]);
```

KUVA 20. Päällä olevan varusteen luonti

Kuvassa 20 näkyy päällä olevan varusteen olion luonti. Se on hyvin samanlainen kuvan 18 olion luonnin kanssa, mutta pieniä eroja löytyy. Komponentit ovat lähestulkoon samoja, mutta items-oliossa ollut Vaihto-komponentti puuttuu. Kuvasta 20 näkyy myös oma Gear-komponenttini, joka on komponenttiluettelossa vain tiedoksi, että kyseessä on päällä oleva varuste. Käytän Gear- ja Item-komponentteja varusteiden vaihdon yhteydessä. Olion asetuksista näkyy, miten sijoitan switch-lauseessa määrittelemäni gearx- ja geary-muuttujat.

4.2.2 Törmäyksen tunnistus

Crafty-pelikirjaston sisään rakennettu törmäyksen tunnistus on tehty yksinkertaiseksi ja helposti ymmärrettäväksi. Törmäyksen tunnistuksen yhteydessä tapahtuvan funktion voi liittää suoraan olion luontiin, mutta tässä on tiettyjä haittapuolia. Olion luomiseen käytetystä koodista tulee tällöin pitkä ja hieman sekavakin, eikä se ratkaisu yksinkertaisesti toimisi minun pelisovelluksessani tehokkaasti. Jos liitän funktion törmäyksen tunnistuksesta olioihin jo luontivaiheessa, niin kun varusteita vaihdetaan valikoimasta päällä oleviin, ne funktiot pitäisi poistaa ja tehdä uudet, tai oliot pitäisi jQuery-toteutuksen tapaan kloonata, poistaa ja tehdä uusiksi. Tällainen toteutus vaatisi mo-

nen monituista riviä koodia ja huolellista suunnittelua. Lopullisesta toteutuksestani tuli kuitenkin varsin yksinkertainen.

Kuten kuvia 20 ja 18 vertailemalla huomaa, päätin liittää varusteiden vaihtoa varten luomani Vaihto-komponentin vain varustevalikoimassa oleviin varusteisiin. Aluksi liitin sen kaikkiin varusteolioihin ja luulin molempien olioluokkien tarvitsevan omat vaihtokomponentit. Tällä ratkaisulla törmäykset kuitenkin toimivat oudosti, ja tutkittuani ja kokeiltuani asiaa huomasin pärjääväni vain yhdellä komponentilla, jonka liitän vain yhteen olioluokkaan. Vähemmällä määrällä koodia saan luonnollisesti tehokkaamman sovelluksen. Kun liitän törmäyksen tunnistuksen vain toiseen olioluokkaan, niin törmäyksen sisältämä funktio ajetaan joka kerta, kun olioon törmää jotakin.

Kuvassa 21 näkyy kirjoittamani komponentti törmäyksen tunnistukselle. Crafty-pelikirjastossa uutta komponenttia aletaan määrittää Crafty.c()-metodilla. Ensin syötetään komponentin nimi, tässä tapauksessa Vaihto. Sitten aaltosulkeisiin määritellään komponentin toiminta. Komponenttiin sijoittamani init-komento tarkoittaa, että sen sisältämä funktio liitetään välittömästi siihen olioon, johon komponentti liitetään. Funktion sisältä löytyy toinen funktio, joka ajetaan vain silloin, kun komponentin olio törmää Gear-nimisen komponentin kanssa.

```
Crafty.c("Vaihto", {
    init: function() {

        //Aina kun gear kohtaa itemin, ajetaan funktio
        this.onHit('Gear', function () {
```

KUVA 21. Komponentti törmäykselle

Päädyin liittämään Vaihto-komponentin varustevalikoiman varusteisiin testauksen ansiosta. Aluksi nimittäin asetin Vaihto-komponentin hahmon päällä oleviin varusteisiin ja varusteiden asettaminen toimi mainiosti. Myöhemmin kuitenkin huomasin, että kun valikoimassa on useampi kuin yksi saman varusteluokan varusteita ja käyttäjä vaihtelee näitä pelihahmonsa päällä, vaihdossa tapahtuu virheitä. Varusteet eivät aina vaihtuneet ja varusteiden tallennuksen yhteydessä tapahtui virheitä. Lisäksi valikoimassa olevia varusteita pystyi vaihtamaan keskenään, jollaiseksi en Crafty-pelikirjaston hahmoikkunaversiota suunnitellut. Uskon virheiden johtuneen siitä, että

kun törmäyksen tunnistus ennen ajoi vaihtofunktion olion kohdatessa Item-komponentin sisältävän olion, vaihto tapahtui tietysti valikoimassa olevienkin kesken. Funktiota ei ole kuitenkaan suunniteltu sitä varten, joten olion asetuksiin tuli virheitä. Mutta kun Vaihto-komponentti liitetään vain valikoimassa oleviin varusteisiin ja tarkastelemaan törmäystä hahmon päällä olevien varusteiden kanssa, ei virheitä tapahdu. Päällä olevia varusteita kun ei voi vaihtaa keskenään, koska yhdelläkään hahmon päällä olevalla varusteella ei ole samaa luokka id:tä. Esimerkiksi hanskoja ei voi laittaa jalkoihin.

Törmäystä varten tarvitsin törmänneiden varusteiden luokka id:t, jotta voin tarkastella, voiko niitä vaihtaa keskenään. Crafty-kirjastolla osoittautui kuitenkin ongelmalliseksi kuljettaa tietoa olioiden mukana. En halunnut kuljettaa id-numeroita olion tekstissäkään. Crafty-kirjaston dokumentaatiosta ei löytynyt apua ongelmaani, mutta sain sieltä ovelan ajatuksen. Crafty-dokumentaatiossa (Crafty 2012) lukee, että olion id on luotu olion taulukkoon. Esimerkiksi luomani gearKuva-olion id:n saa hahmoKuva[0]-komennolla. Tästä juolahti mieleeni, että voisin säilöä dataa olion taulukkoon. Tarkastin, että olioiden taulukoissa on tilaa ja tallensin jokaisen for-lauseen kierroksella kyseisen olion taulukkoon sen kyseessä olevan varusteen id:n ja luokka id:n, kuten kuvasta 22 näkyy. Lisäksi tallensin olion x- ja y-koordinaatistoarvot varusteiden vaihtoa varten. Hahmon päällä oleville varusteille tein aivan samoin, mutta gears-olion taulukon kohtiin kaksi ja kolme sijoitin määrittelemäni gearx- ja geary-arvot.

```
var items = Crafty.e("2D, DOM, Text, Image, Draggable, Collision, Item, Vaihto")
.image("testikuvat/items/class_id_"+datapala[1]+".jpg")
.attr({w: 80, h: 80, x:muuttujaX, y: 500}).collision().text(datapala[2]);

//Sijoitetaan olion taulukkoon class_id arvo! Sekä x ja y
items[1] = datapala[1]; //class_id
items[2] = muuttujaX;
items[3] = 500;
items[4] = datapala[0]; //id
```

KUVA 22. Arvojen sijoitus olion taulukkoon

Vaikka sainkin arvot kätevästi talteen, ei vertailu sujunut kuin rasvattuna. Kohtasin ongelmia saada haluamani varusteen luokka id toisesta törmänneestä oliosta, tässä tapauksessa Gear-komponentin sisältävästä oliosta. Crafty-kirjaston dokumentaation

(Crafty 2012) mukaan hit()-metodilla saa palautettua törmänneen olion tietoja. Tarkemmin sanottuna metodi palauttaa jokaisesta törmänneestä oliosta taulukollisen objekteja. Kuten kuvasta 23 näkyy, hit()-metodi osoittautui hyvin hankalaksi hyödyntää. Lopulta onnistuin löytämään haluamani varusteen luokka id:n kokeilun ja huolellisen pohdinnan tuloksena. Sain selville, että hit()-metodi palauttaa taulukon, jonka sisällä on käytetty taulukoita eri arvoille. Kuvassa 23 näkyvä komento this.hit('Gear')[0] palauttaa osuneen objektin taulukon, jonka sisällä ovat avaimet overlap, normal, obj ja type. Jokaisen avaimen sisällä on arvo. Obj-avaimen dataparissa, eli objektissa, on haluamani varusteen luokka_id. Obj-avaimen datapari siis sisältää törmänneen olion. Objektin arvo on kuitenkin taulukkomuodossa. Tämän vuoksi törmänneeseen olioon pääsee käsiksi kirjoittamalla arvoa vastaavan avaimen, tässä tapauksessa ['obj']. Tämä palauttaa olion taulukon, josta saan sijoittamani varusteen luokan id:n hakemalla avainta yksi. Vaihto-komponentin sisältävän olion luokka id on paljon helpompi noutaa, kuten kuvasta 23 näkyy item_class-muuttujasta. Kun molempien olioiden luokka id:t on noudettu, vertaan niitä if-lauseella toisiinsa. Jos arvot täsmäävät, funktio siirtyy varusteiden vaihtoon.

```
//Aina kun gear kohtaa itemin, ajetaan funktio
this.onHit('Gear', function () {

//Poimitaan törmänneiden olioiden class_id:t
var gear_class = this.hit('Gear')[0]['obj'][1];
var item_class = this[1];

//Vertaillaan
if(gear_class == item_class) {
```

KUVA 23. Arvojen haku ja vertailu

Ennen varsinaista herkkua, eli varusteiden vaihtoa, haen vaihdettavien varusteiden oliot muuttujiin kuvassa 24. Crafty-kirjaston dokumentaatiosta (Crafty 2012) sain selville, että Crafty-pelitilassa voi hakea oliota id:n perusteella, esimerkiksi Crafty(1)-komennolla. Olion id on sijoitettu olion taulukkoon kohtaan 0, joten haen ensiksi Vaihto-komponenttiin törmänneen olion id:n kuvassa 24 näkyvään muuttujaan olio_id. Tämän jälkeen haen olio_id-muuttujan perusteella kyseisen olion gearitem-muuttujaan ja invitem-muuttujaan Vaihto-komponentin sisältävän olion.

```
//Haetaan olion id
var olio_id = this.hit('Gear')[0]['obj'][0];

//Käsiteltävät oliot
var gearitem = Crafty(olio_id);
var invitem = this;
```

KUVA 24. Olioiden haku

Kun oikeat oliot ovat määritelty muuttujiin, on aika suorittaa itse vaihto, joka tapahtuu kuvassa 25 näkyvällä koodilla. Käytännössä vain vaihdan olioiden x- ja y-arvoja sekä komponentteja keskenään. Crafty-kirjastolla voi poistaa vain yhden komponentin kerrallaan removeComponent()-metodilla, mutta useampia komponentteja voi lisätä addComponent()-metodilla. Tämä tuntuu minusta hieman ristiriitaiselta, mikseivät poisto ja lisäys toimi samalla tavalla? Kuvassa 25 näkyy, että poistan valikoimassa olleelta oliolta Vaihto-komponentin, koska se olio on menossa hahmon päälle. Vastaavasti sijoitan Vaihto-komponentin valikoimaan menevälle oliolle.

```
//Asetetaan uudet ominaisuudet
invitem.attr({x: gearitem[2], y: gearitem[3]})
.removeComponent("Item").
removeComponent("Vaihto").addComponent("Gear");

gearitem.attr({x: invitem[2], y: invitem[3]})
.removeComponent("Gear")
.addComponent("Item", "Vaihto");

//Sijoitetaan uudet x ja y arvot
invitem[2] = invitem.x;
invitem[3] = invitem.y
gearitem[2] = gearitem.x;
gearitem[3] = gearitem.y;
```

KUVA 25. Varusteiden vaihto

Olioiden koordinaatistoarvot sijoitin olioiden taulukoihin luontivaiheessa, joten olioiden uudet sijainnit saan kätevästi suoraan taulukoista. Kuten kuvassa 25 näkyy, asetan invitem-oliolle gearitem-olion taulukosta x- ja y-arvot ja toisinpäin. Tämän jälkeen

olioiden taulukossa olevat koordinaatistoarvot ovat vanhoja. Kuvassa 25 päivitän kummankin olion taulukon koordinaatistoarvot uusiin. Crafty-kirjastolla olion x- ja y-sijainnit saa kätevästi invitem.x-komennolla.

4.2.3 Ajax-toteutus

Crafty-pelikirjastoon ei ole integroitu Ajax-toimintoja, joten päätin käyttää jQuery-kirjastoa Ajax-käsittelyyn. Ajax-tallennus onnistuisi pelkällä JavaScriptillä, mutta tehokkuuden ja ajan säästämisen nimissä päätin käyttää jQuery-kirjastoa. jQuery-koodi Ajax-käsittelylle minulla on valmiina jQuery-versiostani, mutta minun täytyi keksiä uusi tapa kuljettaa varusteiden id:t ja luokka id:t jQuery-skriptiin. Kun huomasin olevan mahdollista sijoittaa arvoja olioiden taulukoihin, käytin sitä Ajax-käsittelyäkin varten.

Crafty-versiossani hahmoikkunasta käytän siis kahta eri kirjastoa. Toisin sanottuna minulla on kaksi erillistä skriptiä. Jotta voi kuljettaa tietoa skriptilohkosta toiseen, täytyy käyttää globaaleja muuttujia. Globaali muuttuja tarkoittaa, että se julistetaan kaikenlaisten funktioiden ulkopuolella. Tässäkin datan kuljetustapauksessa taulukot tuntuivat luontevalta ratkaisulta, joten julistan Crafty-skriptissä kaksi globaalia taulukkomuuttujaa, saveGear ja saveInv. Suunnittelin sijoittaa saveGear-taulukkomuuttujaan hahmon päällä olevat varusteet, ja saveInv-taulukkomuuttujaan valikoimassa olevat. jQuery-toteutuksen tapaan aioin sijoittaa taulukkoon varusteen id:n ja luokka id:n samaan arvoon. Esimerkkiarvo taulukossa: ”1,1”.

```
saveGear.push(datapala[0]+","+datapala[1]);
```

KUVA 26. Taulukkoon arvon sijoittaminen

Sijoitan arvoja saveGear- ja saveInv-taulukoihin heti varusteolioiden luonnissa, eli käytännössä heti kun hahmoikkuna ladataan. Kuvassa 26 näkyvä taulukkoon arvon syöttäminen tapahtuu for-lauseessa, joka luo kaikki hahmon päällä olevat varusteet. Sijoitan taulukkoon ensin varusteen id:n ja sitten luokan id:n, ja laitan vielä pilkun väliin erottimeksi. Aivan kuten jQuery-versiossani. Datapala-muuttuja on taulukko, jonka teen jokaiselle varusteelle, koska jokaisen varusteen tiedot on myös erotettu

pilkuilla. Jokaisella for-lauseen kierroksella erotan arvot toisistaan split-metodilla, joka luo varusteen arvoista taulukon. Valikoimassa oleville varusteille teen saman kuin kuvassa 26, mutta sijoitan arvot vain saveInv-taulukkomuuttujaan.

```
//Tallennettaviin tauluihin oikeat tiedot
saveGear.splice(saveGear.indexOf(gearitem[4]+","+gearitem[1]), 1);
var uusi_arvo = invitem[4]+","+invitem[1];

saveGear.push(uusi_arvo);

saveInv.splice(saveInv.indexOf(invitem[4]+","+invitem[1]), 1);
var uusi_arvo = gearitem[4]+","+gearitem[1];
saveInv.push(uusi_arvo);
```

KUVA 27. Tallennettavien taulukoiden ylläpito

Luomiani globaaleja taulukoita täytyy luonnollisesti päivittää, kun käyttäjä vaihtaa hahmonsa varusteita. Periaatteessa vaihdan arvoja taulukosta toiseen. Kuten kuvasta 27 näkyy, käytin arvon poimimiseen taulukosta indexOf()-metodia, joka palauttaa taulukkoon käytettynä etsityn arvon sijainnin. Haluttu arvo haetaan taulukosta sijoittamalla hakuarvo indexOf()-metodin sulkeiden sisään. Sijainnin avulla voin poistaa kyseisen arvon taulukosta splice()-metodilla. Kyseiselle metodille täytyy määrittää, kuinka paljon tavaraa taulukosta poistetaan. Kuvasta 27 näkyy, että olen sijoittanut numeron yksi splice()-metodin sulkeiden loppuun. Tämä tarkoittaa, että vain yksi arvo poistetaan. Jos numeroa nostaa, splice()-metodi poistaa syötetyn numeron verran arvoja alkaen annetusta sijainnista. Kun varuste on poistettu taulukosta, määrittelen uusi_arvo-muuttujaan tilalle tulevan varusteen id:n ja luokan id:n. Sen jälkeen käytän vain push()-metodia uuden arvon sijoittamiseen taulukkoon.

```
<script type="text/javascript" src="includes/jquery.js"></script>
<script>
    //Kun sulkee character.php:n, niin ajaa tietokantaan uudet tiedot
    $(window).bind('beforeunload', function(){

        var urli = "ajax.php";
        var gear_stats = JSON.stringify(saveGear);
        var inventory = JSON.stringify(saveInv);
        $.ajax({
            url: urli,
            async: false,
            type: 'POST',
            dataType: 'json',
            data: {'gear_stats' : gear_stats, 'inventory' : inventory}
        });
    });
</script>
```

KUVA 28. Ajax-käsittely Crafty-versiossa

Tallennettava data pysyy koko ajan taulukoissa tallessa ja kuvassa 28 näkyvä jQuery-skripti on valmiina lähettämään datan. Kuvan 28 skripti on muuten sama kuin jQuery-versiossa käyttämäni, mutta käytän JSON.stringify()-metodia Crafty-versioni taulukkomuuttujiin, saveGear- ja saveInv-taulukoihin. Koska taulukoiden data on koostettu samalla lailla kuin jQuery-versiossani, minun ei tarvitse tehdä muutoksia PHP Web Serviceen.

5 PÄÄTÄNTÖ

Opinnäytetyöni tavoitteena oli tutkia, miten erilaiset JavaScript-kirjastot soveltuvat peliohjelmointiin. Toteutin käytännön osuuden käyttötapauksen kahden tyyppisillä JavaScript-kirjastoilla; yleiskäyttöisillä jQuery-kirjastoilla ja varta vasten peliohjelmoinnin tarpeisiin kehitetyllä Crafty-pelikirjastolla. Ennen käsittelemäni käyttötapauksen aloittamista minulla ei ollut paljoa kokemusta JavaScript-kirjastoilla työskentelemisestä, mutta taidot karttuivat nopeasti tehdessä.

Opinnäytetyöni käsittelemän käyttötapauksen toteutus jQuery-kirjastoilla sujui vaihtelevaan sävyyn. Välillä kehitys eteni sujuvasti, mutta monesti vastaan tuli ongelmia, joiden ylipääsemiseen meni joskus paljonkin aikaa. Käsittelemäni käyttötapauksen avain toiminnallisuuden, elementtien vetämisen ja pudottamisen, toteutus oli kuitenkin nopeasti valmis. Erilaisiin virhetarkistuksiin meni vain paljon aikaa.

Käsittelemäni käyttötapauksen Ajax-käsittelyyn meni todella kauan, vaikka siitä on tehty yksinkertaisempaa jQuery-kirjastolla. Ajax-käsittelyssä suurin ongelma oli lähetettävässä datamuodossa. Kesti pitkään, ennen kuin sain selville millaisessa muodossa data lähtee ja saapuu datan tallentavaan Web Serviceen. Koska Crafty-pelikirjastoon ei ole integroitu Ajax-käsittelyjä, käytin jQuery-kirjastoa Crafty-toteutuksessa. Sama tosin päti jQuery-toteutuksessa; tarvitsin jQuery UI -kirjastoa elementtien vetämistä varten ja jQuery-kirjastoa oikeastaan vain Ajax-ajoon. Crafty-kirjaston Ajax-kehityksen puuttumisen voi laskea selkeäksi puutteeksi, ovathan Ajax-käsittelyt hyvinkin arkipäivää ja käyttökelpoisia peleissä. Luulen Ajax-käsittelyiden puuttuvan Crafty-pelikirjastosta siitä syystä, että HTML5 mahdollistaa datan tallennuksen lokaalisti. JavaScript-pelikirjastot kuitenkin tukevat niin paljon HTML5-ominaisuuksia, esimerkiksi canvas-piirtoalustan tuki löytyy suunnilleen jokaisesta JavaScript-pelikirjastosta.

Opinnäytetyöni käsittelemän käyttötapauksen jQuery-toteutukseen sain paljon apua jQuery-dokumentaatiosta ja keskustelupalstoilta. Vastaavaa apua ei löytynyt samanlaisessa mittakaavassa Crafty-pelikirjaston toteutuksen yhteydessä. Crafty-kirjaston käyttäjäkanta on kuitenkin todella kaukana tällä hetkellä suositun jQuery-kirjastosta. Lisäksi Crafty-kirjaston dokumentaatio on varsin suppea. Dokumentaatio vaikuttaa aluksi laajalta, mutta kun aloin käymään dokumentaatiossa esiteltäviä kohteita läpi, huomasin esittelyjen ja opastusten olevan hyvin rajattuja. Monesta aiheesta ei ollut kunnollisia esimerkkejäkään. Vaillinainen dokumentaatio hidasti hieman kehitystyötäni Crafty-pelikirjastolla.

Käsittelemästäni käyttötapauksesta onnistuin kuitenkin saamaan samannäköiset versiot molemmilla toteutustavoilla. Toiminnallisuudet eroavat vain hieman. Crafty-toteutuksessa ei ole lainkaan elementin pudottamismahdollisuutta, vaan varuste vaihtuu saman tien kun se koskettaa käypää varustetta. Lisäksi Crafty-kirjastolla luodussa versiossa ei voi järjestellä varustevalikoimaa, mutta en kokenut sitä välttämättömäksi ominaisuudeksi.

Crafty-toteutuksen koodista tuli mielestäni tehokkaampi kuin jQuery-kirjastojen lopputuotoksesta. jQuery-toteutuksessa käytin toistoa esimerkiksi sijaintielementtien pudottamisen asetuksessa. Tein hahmon päällä oleville ja varustevalikoimassa oleville

sijaintielementeille omat pudotusfunktionsa. Crafty-toteutuksessa onnistuin saamaan varusteiden vaihdon yhteen funktioon. Siinä suhteessa kehitystyö olisi tarpeellista jQuery-toteutuksessani.

Testaus ja virheiden tarkistus onnistui jQuery-toteutuksen parissa paljon paremmin kuin Crafty-kirjastolla. jQuery-kirjastoilla luodun koodin toiminnallisuutta pystyi tarkastelemaan selaimen virhekonsolissa, esimerkiksi lähettääkö sovellus dataa eteenpäin ja onko koodissa virheitä. Crafty-pelikirjastolla tällainen ei ollut lainkaan mahdollista. Sovellus tapahtui Craftyn omassa pelitilassa, jonka toimintaa ei voinut tarkastella selaimella lainkaan. Edes pelin elementtejä ei voinut tarkastella. Jonkinlainen virhekonsoli olisi hyvä olla saatavilla.

Crafty-kirjasto osoittautui siis hyvinkin keskeneräiseksi. Dokumentaatio on suppea ja koodi toimii paikoitellen oudosti, kuten käsittelemässäni käyttötapauksessa koordinaatistoarvojen sijoittaminen taulukosta pisti koko sovelluksen aivan sekaisin. Osa Craftyn tarjoamista ominaisuuksista ei ole vielä tällä hetkellä loppuun asti kehitettyjä, kuten elementtien vetäminen ja pudottaminen. Crafty-pelikirjaston käyttöön ottama olio- ja komponenttijärjestelmä on kuitenkin ohjelmointitapana hyvin käytännöllinen, ja tekee ohjelmoinnista tehokasta ja aikaa säästävää. Törmäyksentunnistus osoittautui myös hyvin toteutetuksi.

jQuery-kirjastot olivat hyvin käyttökelpoisia käsittelemässäni käyttötapauksessa. Dokumentaatio on kattava ja tiedon löytäminen onnistui helposti. JavaScript-kirjastoina jQuery-kirjastot ovat paljon eheämpiä kuin Crafty, mutta jQuery-kirjastoilla onkin monen vuoden aktiivinen kehitystyö ja sankka kehittäjäkunta takana. Crafty-kirjaston julkaisu tapahtui kuitenkin vasta helmikuussa 2012.

Kaikki edellä mainitut seikat huomioon ottaen, mielestäni yleiskäyttöiset jQuery-kirjastot soveltuivat opinnäytetyöni käsittelemään käyttötapaukseen paremmin. jQuery-toteutuksella sain tehtyä hahmoikkunasta juuri sellaisen kun halusin, mutta Crafty-pelikirjastolla jouduin tekemään kompromisseja, esimerkiksi varusteiden pudottamisessa. Apua ohjelmointiin jQuery-kirjastoilla löytyi helposti, kirjastojen tarjoamat ominaisuudet ja toiminnallisuudet toimivat varmasti ja virheen tarkistus oli mahdollista. Kaikilla näillä saroilla Crafty-kirjaston toteutustyö jäi vajaaksi. Crafty-pelikirjasto

olisi kuitenkin varmasti käytännöllisempi vaihtoehto toisenlaisessa käyttötapauksessa, jossa olisi esimerkiksi paljon törmäyksen tunnistuksia ja spritekarttoja.

Opinnäytetyöni käsitellyn käyttötapauksen jQuery-toteutus olisi jatkokehityksen tarpeessa. Koodista pitäisi saada tehokkaampaa, mutta muuten olen tyytyväinen tuotokseeni. Hahmoikkunaa voisi parannella niin, että varusteiden vaikutukset näkyisivät käyttäjälle. Mutta mikäli Crafty-pelikirjaston kehitys jatkuu aktiivisena, en pitäisi ollenkaan mahdottomana kokonaan siihen siirtymistä. Ohjelmointitapa sillä on kuitenkin tehokasta, kunhan sitä vain parannellaan hieman lisää.

JavaScript-pelikirjastoilla on tällä hetkellä paljon potentiaalia, mutta sieltä täytyisi nousta edes yhden, eheän ja toimivan kirjaston, että ne tulisivat todella tunnetuiksi. JavaScript-pelikirjastoja on ollut olemassa jo muutaman vuoden, mutta ne eivät ole tähän mennessä oikein lyöneet itseään läpi kehittäjien keskuudessa. Impact-pelikirjasto on tällä hetkellä varmaan toimivin valinta JavaScript-pelikirjastoksi, mutta sen 99 $ hinta on suuri este todelliselle läpimurrolle. Yleiskäyttöisistä JavaScript-kirjastoista käytetyimmiksi ovat kuitenkin nousseet ilmaiset kirjastot, jotka ovat sitä myötä keränneet laajan käyttäjä- ja kehittäjäkannan.

JavaScript-kirjastojen tulevaisuus vaikuttaa hyvin valoisalta. Monet HTML5-merkintäkielen uudet ominaisuudet tukevat ja hyödyntävät JavaScript-ohjelmointia. HTML5- ja JavaScript-ohjelmointi yleistyy koko ajan yhä enemmän ja enemmän, mikä avaa ovia niin uusille, kuin vanhoillekin kirjastoille sekä peliohjelmointiin keskittyville kirjastoille. Voisi ehkä jopa sanoa, että JavaScript-ohjelmointi elää kukoistuskauttaan. JavaScript-ohjelmointi on tosin ollut suosittua jo monen vuoden ajan. Tietysti paljon riippuu siitä, jatkuuko HTML5-kehitys aktiivisena ja miten suurimmat selaimet tukevat sen uusia ominaisuuksia. Tähän mennessä kehitys on kuitenkin ollut tasaista ja verkkokehittäjät ovat ottaneet HTML5:n uusia ominaisuuksia innolla vastaan. Uusien ominaisuuksien pohjalta on kehitetty jo monen monituista kirjastoa ja palvelua, kuten lupaava 3d-pelimoottori PlayCanvas. HTML5- ja JavScript-ohjelmoinnilla on varmasti tulevaisuutta.

LÄHTEET

Bai, Giulio 2011. jQuery Mobile First Look. Birmingham: Pact Publishing Ltd.

Bai, Giulio 2010. jQuery Plugin Development Beginner's Guide. Birmingham: Packt Publishing Ltd.

Box2D 2012. Box2D. WWW-dokumentti. box2d.org. Päivitetty 31.31.8.2012. Luettu 8.10.2012.

Box2DJS 2008. Box2DJS. WWW-dokumentti. http://box2d-js.sourceforge.net/. Päivityksestä ei tietoa. Luettu 8.10.2012.

box2dweb 2012. Google. WWW-dokumentti. http://code.google.com/p/box2dweb/. Päivityksestä ei tietoa. Luettu 8.10.2012.

Brinzarea-Iamandi, Bogdan, Darie, Cristian, Hendrix, Audra 2009. AJAX and PHP: Building Modern Web Applications. Birmingham: Packt Publishing Ltd.

Bullet 2012. Bullet. Google. WWW-dokumentti. http://code.google.com/p/bullet/. Päivitetty 5.10.2012. Luettu 5.10.2012.

Cederholm, Dan 2010. CSS3 for Web Designers. New York: A Book Apart.

Chaffer, Jonathan, Swedberg, Karl 2009. Learning jQuery 1.3. Birmingham: Packt Publishing Ltd.

Crafty 2012. Crafty. WWW-dokumentti. http://craftyjs.com/. Päivitetty 3.4.2012. Luettu 27.9.2012.

Cruse, Dale, Jordan, Lee 2011. HTML5 Multimedia Development Cookbook. Birmingham: Packt Publishing Ltd.

CSS3 transform propery 2012. W3Schools. WWW-dokumentti. http://www.w3schools.com/cssref/css3_pr_transform.asp. Päivityksestä ei tietoa. Luettu 7.9.2012.

Coumans, Erwin 2012. Bullet 2.80 Physics SDK Manual. Coumans, Erwin. PDF-dokumentti. http://bulletphysics.com/ftp/pub/test/physics/Bullet_User_Manual.pdf. Päivityksestä ei tietoa. Luettu 5.10.2012.

D´angelo, Ben 2012. EntityJS. D´angelo, Ben. WWW-dokumentti. http://entityjs.com/. Päivityksestä ei tietoa. Luettu 28.9.2012.

enchant.js 2011. Ubiquitous Entertainment Inc. WWW-dokumentti. http://enchantjs.com. Päivityksestä ei tietoa. Luettu 26.9.2012.

Freeman, Jesse 2012. Introducing HTML5 Game Development. California: O'Reilly Media.

gameQuery 2012. gameQuery. WWW-dokumentti. http://gamequeryjs.com/. Päivitetty 25.7.2012. Luettu 26.9.2012.

Gasston, Peter 2011. The Book of CSS3. Canada: William Pollock.

Gay, Jonathan 2001. The History of Flash. Adobe. WWW-dokumentti. http://www.adobe.com/macromedia/events/john_gay/. Päivitetty 13.8.2003. Luettu 10.9.2012.

Goodman, Danny, Morrison, Michael, Novitski, Paul 2010. JavaScript Bible. New Jersey: Wiley.

Google Developers 2012. Google. WWW-dokumentti. https://developers.google.com/web-toolkit/overview. Päivitetty 27.6.2012. Luettu 24.9.2012.

Gube, Jacob 2009. MooTools 1.2 Beginner's Guide: Learn How to Create Dynamic, Interactive, and Responsive Cross-browser Web Applications Using this Popular JavaScript Framework. Birmingham: Packt Publishing Ltd.

Guermeur, Daniel, Unruh, Amy 2010. Google App Engine Java and GWT Application Development: Build Powerful, Scalable, and Interactive Web Apps in the Cloud. Birmingham: Packt Publishing Ltd.

Horn, Shannon 2009. JavaScript Programmer's Reference. New Jersey: Wrox.

HTML5 Tutorial 2012. W3Schools. WWW-dokumentti. http://www.w3schools.com/html5/default.asp. Päivityksestä ei tietoa. Luettu 28.8.2012.

HTML & CSS 2012. W3Schools. WWW-dokumentti. http://www.w3.org/standards/webdesign/htmlcss#whatcss. Päivityksestä ei tietoa. Luettu 6.9.2012.

Kangax 2010. Perfection Kills. Kangax. WWW-dokumentti. http://perfectionkills.com/whats-wrong-with-extending-the-dom/. Päivitetty 5.4.2012. Luettu 27.10.2012.

Koch, Andreas 2010. Migratin from the Prototype JavaScript Framework to jQuery. Koch, Andreas. WWW-dokumentti. http://andykdocs.de/andykdocs/document/Migrating-from-the-Prototype-JavaScript-Framework-to-jQuery. Päivitetty 29.3.2010. Luettu 21.9.2012.

jQuery Mobile 2012. The jQuery Foundation. WWW-dokumentti. http://jquerymobile.com/. Päivityksestä ei tietoa. Luettu 8.10.2012.

jQuery UI 2012. Droppable. WWW-dokumentti. http://jqueryui.com/droppable/. Päivityksestä ei tietoa. Luettu 9.10.2012.

Korpela, Jukka 2008. CSS verkkosivujen muotoilussa.. Porvoo: Bookwell Oy.

Korpela, Jukka 2011. HTML5 uudet ominaisuudet. Porvoo: Bookwell Oy.

Lewis, Paul 2012. Getting Started with three.js. Lewis, Paul. WWW-dokumentti. http://aerotwist.com/tutorials/getting-started-with-three-js/. Päivitetty 4.10.2012. Luettu 4.10.2012.

LimeJS 2012. DigitalFruit. WWW-dokumentti. http://www.limejs.com/. Päivitetty 22.3.2012. Luettu 27.9.2012.

Makzan 2011. HTML5 Games Development by Example: Beginner's Guide. Birmingham: Packt Publishing Ltd.

melonJS 2012. melonJS. WWW-dokumentti. http://www.melonjs.org/. Päivitetty 14.9.2012. Luettu 28.9.2012.

Miraglia, Eric 2009. YUI 3.0.0: First GA Release of YUI's Next-Generation Codeline. YUI Blog. WWW-dokumentti. http://www.yuiblog.com/blog/2009/09/29/yui-3-0-0/. Päivitetty 29.10.2009. Luettu 20.9.2012.

Moncur, Michael 2001. JavaScript Trainer Kit. Helsinki: Edita Oyj.

Orchard, Leslie M., Pehlivanian, Ara, Snook, Jonathan 2010. Professional JavaScript Frameworks: Prototype, YUI, ExtJS, Dojo and MooTools. New Jersey: Wrox.

Prall, Chandler 2012. Physijs. WWW-dokumentti. http://chandlerprall.github.com/Physijs/. Päivitetty 5.10.2012. Luettu 5.10.2012.

Pilgrim, Mark 2010. Dive into HTML5. Pilgrim, Mark. WWW-dokumentti. http://diveinto.html5doctor.com/. Päivityksestä ei tietoa. Luettu 27.8.2012.

PlayCanvas 2012. PlayCanvas. WWW-dokumentti. http://playcanvas.com/. Päivityksestä ei tietoa. Luettu 5.10.2012.

Prayaga, Lakshmi, Suri, Hamsa 2007. Beginning Game Programming with Flash. Boston: Course Technology.

Prototype 2.0 will not extend the DOM 2010. Ajaxian. WWW-dokumentti. http://ajaxian.com/archives/prototype-2-0-will-not-extend-the-dom. Päivitetty 6.4.2010. Luettu 27.10.2012.

Raggett, Dave 2005. Getting started with HTML. W3C. WWW-dokumentti. http://www.w3.org/MarkUp/Guide/. Päivitetty 24.5.2005. Luettu 25.8.2012.

Rutter, Jake 2010. Smashing jQuery: Professional Techniques with Ajax and jQuery. New Jersey: Wiley.

Schafer, Steven 2010. HTML, XHTML, and CSS Bible. New Jersey: Wiley.

Spencer, Ed 2011. Countdown to Ext JS 4: Dynamic Loading and New Class System. Sencha. WWW-dokumentti. http://www.sencha.com/blog/countdown-to-ext-js-4-dynamic-loading-and-new-class-system. Päivitetty 19.1.2011. Luettu 20.9.2012.

Svensson, Peter 2008. Learning Dojo. Birmingham: Packt Publishing Ltd.

Szablewski, Dominic 2012. ImpactJS. WWW-dokumentti. http://impactjs.com/. Päivityksestä ei tietoa. Luettu 26.9.2012.

Techpack 2010. Google Body Browser Can Map Human Body in 3D. Techpack. WWW-dokumentti. http://techpack.org/2010/12/googles-new-browser-can-map-human-body.html. Päivitetty 17.12.2010. Luettu 10.9.2012.

Thomas, Heather 2010. Strengths and Weaknesses of HTML5 and Flash. Lectora. WWW-dokumentti. http://lectora.com/strengths-and-weaknesses-html5-and-flash. Päivitetty 17.11.2010. Luettu 10.9.2012.

three.js 2012. GitHub.WWW-dokumentti. https://github.com/mrdoob/three.js/. Päivitetty 1.10.2012. Luettu 4.10.2012.

Waldron, Rick 2000. The Flash History. Flash Magazine. WWW-dokumentti. http://www.flashmagazine.com/news/detail/the_flash_history/. Päivitetty 20.11.2000. Luettu 10.9.2012.

Wappalyzer 2012. JavaScript Frameworks. Wappalyzer. WWW-dokumentti. http://wappalyzer.com/categories/javascript-frameworks. Päivityksestä ei tietoa. Luettu 21.9.2012.

Wikipedia 2012. Ajax (programming). Wikipedia. WWW-dokumentti. http://en.wikipedia.org/wiki/Ajax_(programming). Päivitetty28.9.2012. Luettu 4.10.2012.

Wikipedia 2012. Adobe Flash. Wikipedia. WWW-dokumentti. http://en.wikipedia.org/wiki/Adobe_Flash. Päivitetty 9.9.2012. Luettu 10.9.2012.

Wikipedia 2012. Prototype JavaScript Framework. Wikipedia. WWW-dokumentti. http://en.wikipedia.org/wiki/Prototype_JavaScript_Framework. Päivitetty 30.8.2012. Luettu 18.9.2012.

Wikipedia 2012. YUI Library. Wikipedia. WWW-dokumentti. http://en.wikipedia.org/wiki/YUI_Library. Päivitetty 29.8.2012. Luettu 20.9.2012.

Winokur, Danny 2011. Flash to focus on PC browsing and mobile apps; Adobe to aggressively contribute to HTML5. Adobe. WWW-dokumentti. http://blogs.adobe.com/conversations/2011/11/flash-focus.html. Päivitetty 9.11.2011. Luettu 10.9.2012.

Yahoo 2012. YUI 2: YAHOO Global Object. Yahoo. WWW-dokumentti. http://developer.yahoo.com/yui/yahoo/. Päivitetty 20.9.2012. Luettu 20.9.2012.

York, Richard 2005. Beginning CSS: Cascading Style Sheets for Web Design.Indianapolis: Wiley Publishing Inc.