HTML5 sovellusalustana

Pyry Lehdonvirta & Jukka K. Korpela

RPS-yhtiöt • Helsinki

Mitä HTML5-sovellukset ovat?

”HTML5:n” kaksoismerkitys

Ilmauksella ”HTML5” on selvästi kaksi eri merkitystä. Sovellusten kehittäjien keskuudessa ja sovellusmarkkinoilla se tarkoittaa *sovellusten toteuttamisen tapaa*, joka perustuu web-tekniikoiden käyttöön.

Toinen merkitys on, että ”HTML5” tarkoittaa HTML-kielen ja sen määrittelyn (standardin) uutta kehitysvaihetta, jossa on uusia elementtejä ja muita piirteitä. Tämä merkitys on vallitseva virallisissa määrittelyissä, yleensä kirjoissa ja vaikkapa Wikipediassa. Muutoin käytetään ilmausta ”HTML5” eri medioissa eri merkityksissä.

Selvyyden vuoksi erotamme nämä ”HTML5:n” merkitykset toisistaan tässä kirjassa käyttämällä johdonmukaisesti nimityksiä **HTML5-sovellukset** ja **HTML5-kieli**. Niillä on yhtymäkohtia, mutta ne ovat eri asioita. Tämä kirja käsittelee HTML5-sovelluksia. HTML5-kielen uusille piirteille on kyllä käyttöä HTML5-sovelluksissa, etenkin sitten kun piirteet on laajemmin toteutettu selaimiin, mutta niitä ilmankin selvitään hyvin.

Web-tekniikoita käyttäviä sovelluksia

Tarkoitamme HTML5-sovelluksella

- sovellusta, joka
- on tarkoitettu toimimaan selainmoottorin perustalla ja
- käyttää webistä tuttuja avoimia tekniikoita, kuten HTML, CSS ja JavaScript sekä tarvittaessa HTTP.

Mainitut tekniikat toimivat **universaalina sovellusalustana,** jolloin sovellus toimii kaikissa laitteissa, joissa on selainmoottori. Usein käytetään tehokkaasti hyväksi näiden tekniikoiden uusia piirteitä (kuten CSS3 ja uudet JavaScript-APIt), mutta HTML5-sovelluksen käsite ei ole sidoksissa minkään tekniikan erityiseen versioon.

Ohjelmoinnin kannalta HTML5-sovellukset toteutetaan JavaScript-kielellä. Tämä johtuu siitä, että JavaScript on ainoa ohjelmointikieli, jota selaimet yleisesti osaavat tulkita ja suorittaa.

Tämä määrittely rajaa HTML5-sovellukset selkeästi. Määrittely ei kuitenkaan rajaa sitä, miten tekniikoita käytetään ja mitä suunnittelu- ja rakenneperiaatteita sovelletaan. Ne voivat olla asioita, jotka tekevät HTML5-sovelluksesta *hyvän* HTML5-

tään kallis ja hidas sovellusten kehittäminen eri laitteille erikseen.

Samaan aikaan olivat web-selaimet kehittyneet niin tehokkaiksi ja ominaisuuksiltaan monipuolisiksi, että ne tarjosivat hyvän sovellusalustan. Näin voidaan saavuttaa toimivuus eri laitteilla ilman laitekohtaista kehitystyötä.

Kuten luvussa *Mitä HTML5-sovellukset ovat?* kuvattiin, sovellusten toteutuksen perusvaihtoehdot ovat HTML5-sovellus, Flash ja natiivisovellus. Muut vaihtoehdot, kuten Java-sovelman eli appletin upottaminen web-sivuun, ovat jo jääneet varsin vähälle käytölle.

HTML5-sovellus eli sovelluksen toteuttaminen selainmoottorien pohjalle merkitsee avointen standardien ja määrittelyjen käyttämistä, vastakohtana eri valmistajien suljetuille ratkaisuille, kuten Flash-tekniikka. Tämä antaa vapauden valita, millaisilla tavoilla ja välineillä koodi tuotetaan. Maksullisia kehitysohjelmia ei välttämättä tarvita lainkaan.

Flashilla ja HTML5-sovelluksella on se yhteinen piirre, että sovelluksen koodi on laiteriippumatonta. Laitekohtainen toteutus on ”alustan” asia; alustana on Flashin tapauksessa Flash Player, HTML5-sovelluksen tapauksessa selainmoottori.

Flashin merkitys on olennaisesti vähentynyt muun muassa laitevalmistajien asenteiden ja toimenpiteiden takia. Flashia on pidetty ongelmallisena suorituskyvyn, tehonkulutuksen (akunkeston) ja tietoturvan takia, ja Flash-tuki on poistunut tai poistumassa monista tärkeistä laitteista.

HTML5-sovellukset kilpailevatkin nyt lähinnä natiivien eli eri laitteille erikseen kehitettyjen sovellusten kanssa.

Natiivi vai HTML5-sovellus?

Natiivi sovellus on tehty tietylle laitetyypille, käyttäen sille tarkoitettuja välineitä ja menettelytapoja. Tähän sisältyy natiivien sovellusten voima ja heikkous. Ne voidaan tehdä tehokkaiksi ja laitteen ominaisuuksia hyödyntäväksi, mutta ne ovat sidoksissa tiettyyn tekniikkaan. Tästä seuraa ennen muuta se, että sovelluksen toteuttaminen toisenlaisille laitteille merkitsee uudelleenohjelmointia. Toki yleinen suunnittelutyö on jo tehty, mutta silti jokainen olennaisesti erilainen uusi laite vaatii samaa luokkaa olevan työmäärän kuin alkuperäinen toteutus. Tähän vaikuttaa sekin, että eri laitteiden natiivien sovellusten tekemiseen saatetaan käyttää eri ohjelmointikieliäkin. Esimerkiksi natiivit sovellukset iPhonelle kirjoitetaan Objective-C:llä, Androidille pääosin Javalla.

HTML5-sovellukset voidaan tehdä periaatteessa laiteriippumattomiksi. Käytännössäkin päästään yleensä ainakin siihen, että sovittaminen uudelle laitteelle on olennaisesti pienempi työ kuin uudelleenohjelmointi.

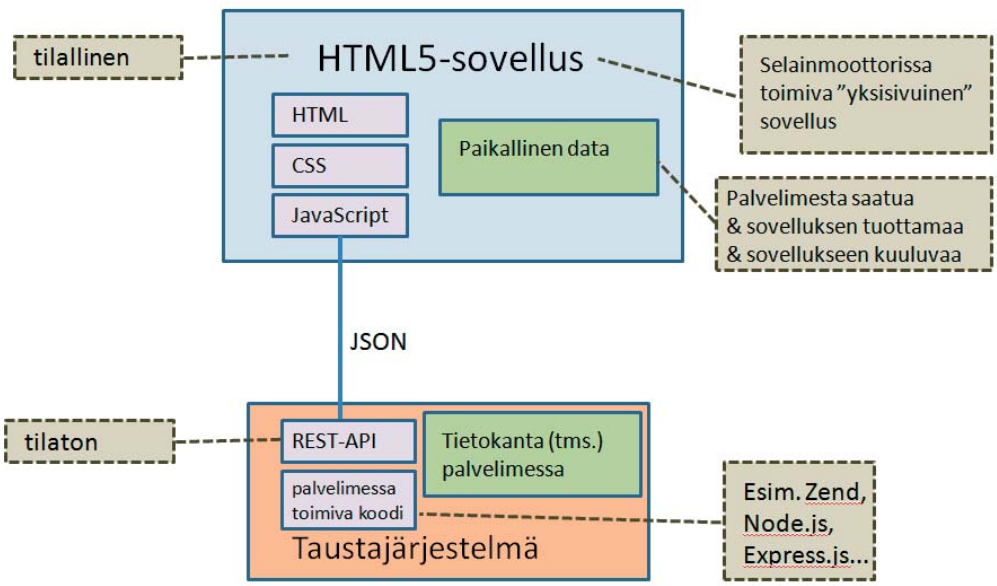
Tehokkuudessa HTML5-sovellus saattaa hävitä natiivisovelluksille muun muassa siksi, koska HTML:n, CSS:n ja JavaScriptin jäsentäminen ja tulkitseminen vie aikaa. Useimmiten erolla ei ole suurta merkitystä nykyaikaisten laitteiden ja selainmoottorien tehokkuuden takia.

HTML5-sovelluksen suunnittelu

HTML5-sovellus järjestelmän osana

Seuraava kaavio havainnollistaa HTML5-sovelluksen tyypillistä kokonaisrakennetta ja toimintaympäristöä. **Edustajärjestelmä** (front-end) tarkoittaa selaimessa toimivaa osaa, joka on vuorovaikutuksessa käyttäjän kanssa ja suorittaa toimintoja paikallisesti käyttäjän laitteessa. Edustajärjestelmä on HTML5-sovellus, jos se on toteutettu HTML:llä, CSS:llä ja JavaScriptillä ja toimii selainmoottorissa. **Taustajärjestelmä** (back-end) on palvelimessa toimiva osa. Niiden välisessä tiedonsiirrossa käytetään yleisimmin jäljempänä kuvattavia REST-periaatteita ja datalle JSON-muotoa (JavaScript Object Notation), joka on luonteeltaan JavaScript-olioiden ja -arvojen esittämistä tekstimuodossa.

HTML5-sovelluksen kannalta taustajärjestelmät tarjoavat palveluita määritellyn liitännän (interface) kautta. Tässä kirjassa käsitellään taustajärjestelmän toteuttamista vain suppeasti, koska siinä käytettävät tekniikat ovat riippumattomia HTML5-sovelluksista.



HTML5-sovelluksen perusosat ja yhteys taustajärjestelmään.

tapa, joskin rajoittunut, sillä emulointi on puutteellista. Microsoftin sivulta *Internet Explorer Application Compatibility VPC Image* voi ladata virtuaalitietokoneita, joissa aiempia versioita voi ajaa aidosti. Toinen vaihtoehto on ladata ja asentaa *Windows Virtual PC*, jolloin voidaan käyttää virtuaalikoneessa Windows XP:tä ja siinä vanhaa IE:tä. Vielä varmempi tapa on käyttää sopivaa vanhaa Windowsin ja IE:n versiota jossain erillisessä fyysisessä koneessa.

IE:n osalta vanhempien versioiden testaaminen on paljon tärkeämpää kuin muiden selainten. Muut selaimet nimittäin yleensä päivittyvät automaattisemmin tai ainakin useammin kuin IE. Lisäksi IE:n päivittämiselle on esteitä niin, että päivitys saattaa vaatia Windowsin uudemman version ja tämä taas käytännössä usein koneen vaihtamisen uudempaan.

Älypuhelimien, tablettien ym. selainten testaamiseen on olemassa monia välineitä, ks. esim. *Mobile Emulators & Simulators: The Ultimate Guide*. Ne ovat kuitenkin yleensä rajoittuneita: niissä testataan olennaisesti vain HTML-dokumentin ulkoasua, ottamatta huomioon sen vuorovaikutteista käyttöä ja muuttumista. HTML5-sovelluksissa HTML-dokumentti on useinkin hyvin muuttuva, joten testausta varten pitää erikseen luoda näytteitä sen tilasta.

Niinpä tarvitaan lisäksi valikoima erilaisia fyysisiä laitteita testausta varten. Minimissään tarvitaan muutama laajasti käytetty, olennaisesti erilainen mobiililaite, mieluiten sekä uusin versio että vähän vanhempi. Käytännössä testataan yleensä vain ”virallisia”, laitteiden mukana tulevia selaimia.

Sivujen testaamista mobiililaitteilla helpottaa **Weinre** (WEb INspector REmote). Sen avulla voidaan kytkeä mobiililaitteen selain pöytäkoneen ohjaukseen. Samantapainen, uudempi ohjelmisto on **Adobe Edge Inspect**.

Selainten käyttöosuuksista löytyy runsaasti tilastoja, joita kuitenkin kannattaa pitää enintään suuntaa-antavina. Tavallisimmin käytettyjä sivustoja ovat StatCounter.com ja W3Counter.com. Yleisesti on kuitenkin melko epäolennaista, mitkä selainten yleisyydet ovat. Sen sijaan voi olla merkittävää, jos jokin uusi tai aiemmin harvinainen selain alkaa yleistyä. Silloinkin pitää muistaa, että maailmanlaajuiset, suomalaiset ja tietyn sivuston käyttäjien selainjakaumat voivat olla keskenään hyvin erilaisia.

Käyttöliittymäkehityksen tekniikoita

HTML5 Boilerplate

HTML5 Boilerplate on dokumenttipohja, josta voi aloittaa uuden HTML-dokumentin kehittämisen. Siinä ei ole mitään erityistä sisällön rakennetta. Se keskittyy HTML:n ”pakollisten kuvioiden” tekemiseen harkitulla, eri selaimissa toimivalla ja monia selainten ongelmia huomioon ottavalla tavalla.

CSS-asetuksilla on merkittävä osa HTML5 Boilerplatessa, mutta ne ovat luonteeltaan hyvin yleisluonteista siistimistä pikemminkin kuin erityisen tyylin luo-

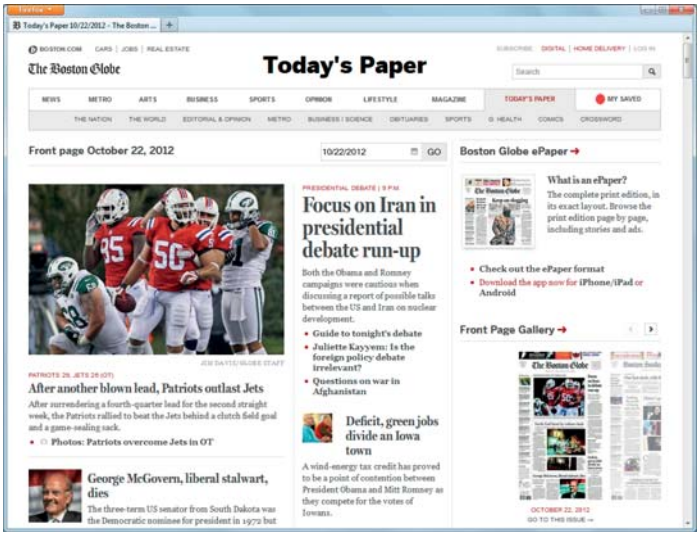
mista. Ne sisältyvät tiedostoon css/style.css, joka paljolti perustuu normalize.css-koodiin. Se on nykyaikainen vaihtoehto CSS Reset -asetuksille, joilla on aiemmin pyritty poistamaan jopa kaikki selainten oletusarvoisesti tekemät muotoilut. Normalize.css pyrkii vain huolehtimaan selainten tarpeettomista eroista ja virheistä.

Teknisesti HTML5 Boilerplate koostuu HTML-dokumentista ja useista siihen liittyvistä tiedostoista, jotka ovat CSS-, JavaScript- tai tekstimuotoisia. Se on saatavissa perusversiona, jossa on kommentit mukana, kommenteista riisuttuna versiona ja räätälöityinä versioina, joihin otetaan mukaan halutut piirteet. Räätälöinnin sijasta voi lähteä perusversiosta ja poistaa siitä tarpeettomat osat. Seuraavassa on kuitenkin esimerkkinä räätälöidyistä versioista kaikkein riisutuin:

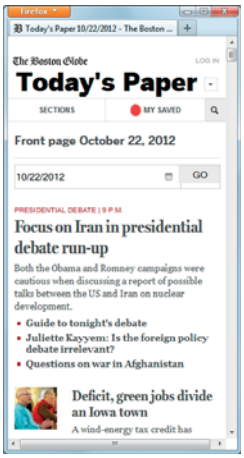
```
<!doctype html>
<html class="no-js" lang="en">
<head>
 <meta charset="utf-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
 <title></title>
 <meta name="description" content="">
 <meta name="author" content="">
 <meta name="viewport" content="width=device-width">
 <link rel="stylesheet" href="css/style.css">
 <script src="js/libs/modernizr-2.5.3.min.js"></script>
</head>
<body>
<header>
</header>
<div role="main">
</div>
<footer>
</footer>
</body>
</html>
```

Valinnaisia lisäyksiä tähän ovat:

- jQueryn mukaan ottaminen (mukana perusversiossa, tiivistettynä)
- palvelimen konfigurointi (perusversiossa mukana näistä .htaccess Apachelle)
- luokkamäärittelyt, jotka vastaavat IE:n eri versioita (perusversiossa mukana)
- script.js ja plugin.js (perusversiossa mukana; tarkoitus on, että kehitystyön aikana niihin kirjoitetaan sivuston tai sovelluksen yleinen koodi sekä erilaisten lisäosien koodi)



The Boston Globe -lehden verkkosivut ovat isolla näytöllä sanomalehtimäisesti taitettuja.



Kapeassa näytössä The Boston Globe muuttuu yksipalstaiseksi.

Responsiivisuuden testaus

Kun responsiivisuus toteutetaan edellä kuvatulla tavalla, sitä voidaan testata tavallisella pöytäkoneella, säätämällä selainikkunan leveyttä. Vielä helpommin voidaan perustestaus tehdä esimerkiksi käyttämällä Chromessa tai Firefoxissa Web Developer Extension -lisäosaa. Sen komentovalikon Resize-kohdassa on valinta ”View

Responsive Layouts", joka näyttää sivun erikokoisissa alueissa. Vaihtoehtoja voidaan pitää yhtenä perusteltuna näkemyksenä siitä, millaisia kokoja voidaan pitää tyypillisinä ja tärkeinä. Tätä kirjoitettaessa vaihtoehdot ovat:

- 320 × 480 ("mobile portrait")
- 480 × 320 ("mobile landscape")
- 600 × 800 ("small tablet portrait")
- 800 × 600 ("small tablet landscape")
- 768 × 1024 ("tablet portrait")
- 1024 × 768 ("tablet landscape").

Asettelun vaihtoehtoja

Edellä on kuvattu, miten sovellus saadaan käyttämään eri tyyliohjeita (CSS-tiedostoja) näytön leveyden mukaan. Näiden tyyliohjeiden sisältö suunnitellaan niin, että asettelu (layout) sopii tietynlevyisille näytöille. Tällöin kannattaa yleensä noudattaa seuraavia periaatteita:

Asettelu voi olla joko kiinteä, pikselimääräinen asettelu (mediakyselyn mukaisen alarajan mukaan) tai joustava asettelu, joka käyttää koko käytettävissä olevan leveyden. Edellinen on helpompi toteuttaa sekä suunnittelun että tekniikan tasolla. Sitä käytettäessä on yleensä parasta asettaa vasen ja oikea marginaali niin, että aseteltu sisältö näkyy keskitettynä, jos näyttö on alarajaa suurempi. Joustava asettelu sopii paremmin responsiivisen suunnittelun ideaan ja tavoitteisiin, mutta vaatii enemmän työtä. Keskeinen kysymys on: Onko saavutettavissa selvää etua sillä, että käytetään koko leveys hyväksi?

Sisällön määrän vaihtelu

Usein ajatellaan, että mobiilikäytössä sisällön määrää on supistettava. Tämä on teknisesti melko helppoa: kapeaa näyttöä varten tehdyssä tyyliohjeessa voidaan kokonaan poistaa joitakin elementtejä esityksestä (display: none).

Yleensä on kuitenkin parempi lähteä siitä, että erityyppisillä laitteilla tarjotaan sama sisältö, mutta eri muodossa. Jos sisällön muotoilu yksipalstaiseksi ei ole riittävä keino, voidaan osa sisällöstä asettaa alkutilanteessa näkymättömäksi. Verkkosivustoihin verrattaessa tämä vastaa sisällön siirtämistä linkin taakse. HTML5-sovelluksessa käytetään linkin sijasta painiketta, joka tuo sisällön näkyviin (siirtymättä uudelle sivulle).

Pienessä näytössä voidaan osa sisällöstä ja toiminnoista jättää aluksi piiloon.

Vaikka *sisällön* kokonaan pois jättäminen ei yleensä ole hyvä ajatus, *koristeiden* vähentäminen voi olla aiheellista. Esimerkiksi kuvia, joiden tarkoitus on lähinnä luoda tunnelmaa, voi pienissä laitteissa pudottaa pois tai korvata pienemmillä.

HTML5-sovellusten yleistyminen ei ole ollut eikä tule olemaan suoraviivaista. Tunnettu esimerkki on, että Facebook siirtyi HTML5-sovellusten tapaisesta lähestymistavasta vuonna 2012 takaisin natiiveihin sovelluksiin. Tämä johtui kuitenkin lähinnä siitä, että Facebook arvioi siirtyneensä HTML5-sovelluksiin liian aikaisin.

Verkkopalvelut muutetaan sovelluksiksi

Vuorovaikutteisuutta sisältävät palvelut verkossa tullaan toteuttamaan HTML5-sovelluksina. Tämäkään muutos ei tapahdu suoraviivaisesti, mutta kuitenkin suhteellisen nopeasti. Verkkopalveluja uusitaan eri syistä muutaman vuoden välein, ja yhä useammin huomataan HTML5-sovellus parhaaksi vaihtoehdoksi.

Tämä voi aiheuttaa myös pyrkimyksiä muuttaa verkkosivustoja sovelluksiksi vain koska se on muodikasta. Kun kyse on sivustosta, jossa ei ole merkittävää vuorovaikutteisuutta edes suunnitteilla, on yleensä järkevää säilyttää se sivustona. Tähänkin voi tulla muutoksia, jos havaitaan, että sovellusmaisesta käyttöliittymästä ja käyttäjäkokemuksesta on merkittävää hyötyä. Sovelluksiin tottuminen voi myös johtaa siihen, että käyttäjät alkavat odottaa ja vaatia sovellusmaisuutta.

Vastauksia haasteisiin

HTML5-sovellusten keskeiset edut painavat paljon, mutta edessä on myös haasteita. Seuraavassa tarkastellaan eräitä asioita, jotka voidaan nähdä isoiksi ongelmiksi, mutta jotka ovat enemmänkin HTML5-sovellusten vahvuuksia.

Löydettävyys

Ei auta, että sovellus on maailman hienoin, jos potentiaaliset käyttäjät eivät löydä sitä. Ihmiset ovat tottuneet etsimään tietoa hakukoneilla, eivätkä hakukoneet monissa tapauksessa näe sovelluksen tekstisisältöä. Kuten tässä kirjassa on kuvattu, tähän on kuitenkin hyviä ratkaisuja. Ihmiset ovat myös oppineet hakemaan sovelluksia sovelluskaupoista. HTML5-sovellusten saaminen sovelluskauppajakeluun on voi vaatia lisätyötä ja järjestelyjä, mutta niiden yleistyessä sovelluskaupat joutuvat todennäköisesti laajemmin hyväksymään myös puhtaita HTML5-sovelluksia.

HTML5-sovellusten etuna siis on se, että ne voidaan saada jakeluun sekä sovelluskauppoihin että webiin, jolloin löydettävyys on paras mahdollinen.

Seuranta

Kaupallisessa käytössä on usein tärkeää seurata käyttäjien liikkumista sivustossa ja heidän toimintaansa, kuten ostokäyttäytymistä. Verkkosivustoille kehitetyt seurannan välineet eivät useinkaan sellaisinaan sovi HTML5-sovelluksiin. Google tar-

Hakemisto