HTML5 och dess användning för mobila tjänster

Tobias Asplund

Obs, inte en färdig version

Referat

Med en växande marknad för mobila applikationer behövs kostnadseffektiva system för att utveckla applikationer som fungerar på så många mobila plattformar som möjligt. Dagens smarttelefoner kommer med förinstallerade moderna webbläsare som kan användas för att exekvera HTML5 oberoende av plattformens operativsystem. Man behöver alltså inte koda skilt i olika språk för varje plattform. Meningen med denna avhandling är att undersöka HTML5 i mobilapplikationer, både som en del av en nativ applikation och som möjligheten att föra över mobila tjänster till webben. Vilken av dessa man väljer beror på vad kraven på resurser är för applikationen. Om smarttelefonens egna resurser behövs, så som kamera eller accelerometer kan en nativ applikation vara att föredra. Ifall applikationen behöver utföra tunga beräkningar kan dock en webb-baserad lösning vara bättre. HTML5-standarden fastslogs av World Wide Web Consortium (W3C [1]) den 28 oktober 2014 och inkluderar lösningar för ljud, video och grafik.

Nyckelord:

Contents

INLEDNING

Efter en stor smarttelefonboom som pågått i ett par år [2] har nu nästan alla i västvärlden en modern smarttelefon i fickan. Med den sköter man allt från arbete, kalendrar, kontakter och ekonomi till underhållning. Detta ger ett stort behov för applikationer för smarttelefoner som snabbt, användarvänligt och säkert kan sköta dessa uppgifter. Att utveckla applikationer som fungerar på alla plattformar är dyrt och tidskrävande. Eftersom olika plattformar inte bara har olika hårdvara utan applikationerna behöver även kodas i olika språk. Googles tekniska chef (eng: VP of engineering) Vic Gundotra sade vid MobileBrat 2009 att inte ens Google har tillräckligt med resurser för att stöda alla smarttelefoner [3] [4]. Detta har lett till ett stigande behov för ett gemensamt språk för att utveckla mobilapplikationer. Både Google och Apple har därför satsat på stöd för och utveckling av HTML5 och CSS3. Cascading Style Cheets, CSS används för att grafiskt presentera den information som finns i HTML. HTML bygger upp och strukturerar innehåll och CSS styr hur innehållet presenteras grafiskt [4].

HTML5 och CSS3 är standarder utvecklade i och för webben, som under de senaste åren spridit sig till våra mobila apparater. Med mycket varierande hårdvara och mjukvara, med olika operativsystem och skärmstorlekar, krävs mycket flexibilitet i våra applikationer. Utvecklare måste nu avvänjas vid full kontroll på pixelnivå och vänja sig vid flexibla applikationer som anpassar sig till användarens apparat. Det är inte bara olika storlekar av datorskärmar och smarttelefoner som idag används för att köra applikationer i HTML5, även TV-apparater och spelkonsoler används idag för att ta webben till vardagsrummet. Antalet olika apparater som i framtiden behöver stödas i våra HTML5 applikationer begränsas bara av antalet nya och gamla vardagliga ting som industrin kan utveckla en ”smart-” version av.

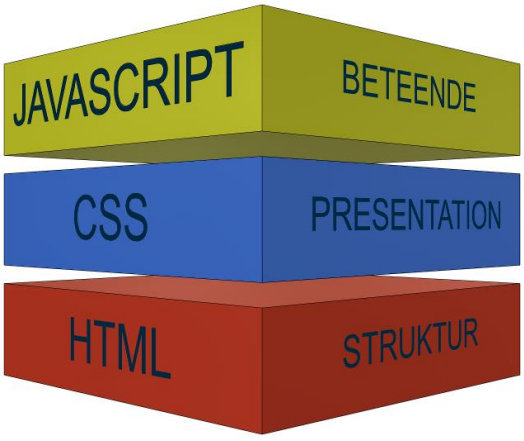


Bild 1 HTML, CSS och Javascripts funktionalitet representerat som lager

Hand i hand med HTML kommer CSS och Javascript. Denna trio fungerar tillsammans i tre lager (Bild 1). HTML bygger upp datastrukturen och innehållet, CSS står för stilen och den visuella representationen av informationen i HTML, Javascript lägger till användbarhet och ytterligare funktionalitet och interaktivitet utöver det som HTML och CSS erbjuder[9]. Denna avhandling kommer fokusera på HTML och det som är nytt i HTML5, men en liten del CSS och Javascript kommer ändå tas upp ibland på grund av deras nära relation till HTML.

HTML5

Inledning till kapitlet HTML5

Historia

HTML5 är ett steg i kedjan som är HTMLs historia och skall ta över efter HTML4.1 som i princip är HTML4 med en del felkorrigeringar. Html4 var inte dåligt utan gjorde exakt vad det var menat att göra, visa statiska dokument med linkar mellan dokumenten. När HTML4 nästan var färdigt valde World Wide Web Consortium (W3C[10]) att börja ta webben i en riktning mot XHTML, som motsvarar HTML4 men är formaterat som XML. Detta gav starkare syntaxregler, attribut måste ha citationstecken och element kan inte lämnas öppna utan måste avslutas. XHTML 2.0 motsvarade inte helt vad utvecklare gjorde på webben utan mer vad W3C då ville att man skulle göra, det var inte heller bakåtkompatibelt med tidigare HTML innehåll som redan fanns på webben. Detta ledde till att World Hypertext Application Technology Working Group (WHATWG [2]) bildades 2004 med representanter från bland annat Opera, Mozilla och senare även Apple. De ville utveckla en bättre och effektivare uppsättning av funktioner med bakåtkompatibilitet, vilket ledde till specifikationen Web Applications 1.0. Diskussioner med W3C resulterade i att de gick tillbaka ett steg i sitt arbete på HTML och skapade en ny och öppen arbetsgrupp. Baserat på Web Applications 1.0 utvecklade gruppen det som idag är HTML5, gruppen är öppen så att vem som helst kan delta i diskussionerna för utveckling av standarden. [3]

I dagens läge utvecklar W3C och WHATWG HTML5 specifikationen parallellt med varandra. WHATWGs specifikation utvecklas snabbt med ny funktionalitet som ofta är långt före officiella rekommendationer för vad som kan förväntas fungera i dagens webbläsare. Specifikationen är levande och saknar

versionsnummer. W3C specifikationen däremot innehåller den mest stabila och allmänt accepterade funktionaliteten och ger en verkligare bild av vad som idag förväntas fungera på webben. För att sprida hjälpa HTML5 att etablera sig skapades dessutom en HTML5 logo (Bild 2). [3]

Bild 2 HTML5 Logo

Delta i utvecklingen av W3C standarden på: www.w3.org/html/wg och WHATWG på www.whatwg.org . CSS och HTML

Nytt i HTML5

Med HTML 5 kommer en hel del nya märkord (eng: markup) som både kompletterar och ersätter funktionalitet från tidigare HTML-versioner. En stor del av de möjligheter att påverka det grafiska utseendet i webbapplikationer som tidigare funnits i HTML flyttas över till CSS3. Fenomenet allmänt känt som ”Divitis”, som är resultatet av många nästlade och möjligen onödiga *div* element, löses med en handfull nya element som skall göra det lättare att bygga upp en webbapplikation. Ny funktionalitet kommer i form av bland annat multimedia, geolokalisering och API för kommunikation mellan olika applikationer (Cross-Document Messaging). HTML5 fungerar till stor del i äldre webbläsare, men i vissa fall måste man fortfarande inkludera andra lösningar för att allt skall fungera. För äldre versioner än Internet Explorer 9 måste man till exempel definiera nya HTML5 element med Javascript för att webbläsaren skall förstå dem (koden nedan). Specifikationen ändras fortfarande och stödet för HTML5 varierar även mellan moderna webbläsare, exempel på detta och varför man ändå kan använda HTML5 tas upp tillsammans med formulären senare i detta kapitel.

```
<!--[if lt IE 9]>
<script type="text/Javascript" >
document.createElement("nav" );
document.createElement("header" );
document.createElement("footer" );
document.createElement("section" );
```

```
document.createElement("aside" );
document.createElement("article" );
</script>
<![endif]-->
```

Multimedia

Nya element för att integrera multimedia i HTML är *audio* och *video* elementen. Med dessa kan man direkt integrera ljud och bild i en webbapplikation, utan att som tidigare behöva vara beroende av extra tilläggsmoduler. Det gör att till exempel Adobe flash och Microsoft Silverlight inte behövs i HTML5 applikationer med multimedia. Element för innehåll så som bilder, kod, olika typer av figurer och dylikt kan nu läggas i ett *figure* element. En beskrivning av inehållet i ett *figure* element sätts i ett *figcaption* element.

```
<figure>
        <img src=”bild.png” alt=”en bild”>
        <figcaption>Bild1- landskap</figcaption>
</figure>
```

Strukturella element

En del nya strukturella element hjälper till med att lätt strukturera sidan så att man själv och även så kallade skärmläsare (eng: screeners) kan analysera och läsa sidan. Skärmläsare används bland annat för att göra webbsidor tillgängliga för funktionshindrade personer. Tidigare har webbsidor byggs upp med *div* element nästlade i varandra, vilket gjort det svårt att tolka koden och se vilka delar som hör vart. I HTML5 finns nya element med samma funktion som *div* element, men med mer beskrivande namn. Nu när man bygger upp den visuella strukturen kan man samtidigt beskriva vilken typ av innehåll som finns var, bara genom att använda den nya elementen istället för *div* element. Det finns även en del nya strukturella element som för med sig ny funktionalitet till HTML. Till exempel *progress* elementet som kan användas för att visa hur långt någon uppgift fortskridit, *meter* elementet används för att visa till exempel använt minne eller diskutrymme. De nya elementen finns i bilaga 1.

Samma sida med *div* och nya element:

```
<!DOCTYPE HTML>
<HTML>
<head>
```

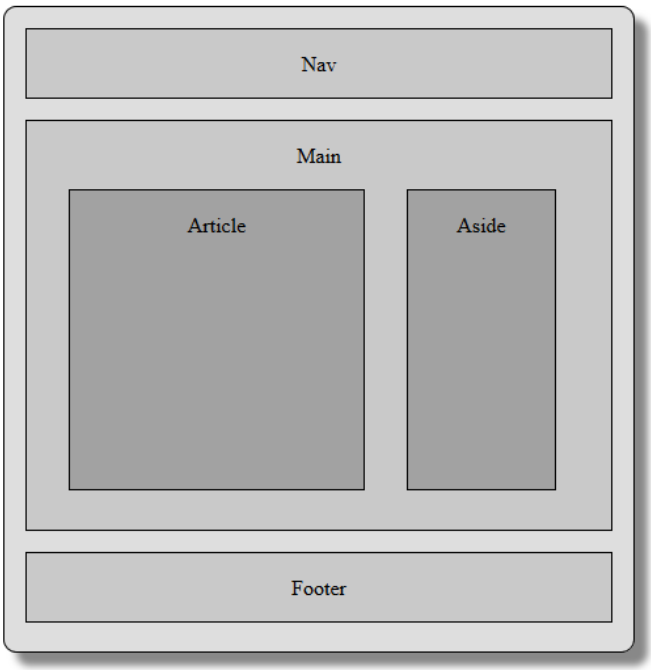
```
<!DOCTYPE HTML>
<HTML>
<head>
```

```
        <title>Divelement</title>
</head>
<body>
        <div>
                <div>
                        <ul>
                                <li>Hem</li>
                                <li>Info</li>

        <li>Kontakt</li>
        </ul>
                </div>
        </div>
        <div>
                <div>En artikel</div>
                <div>Reklam</div>
        </div>
        <div>
                Footer innehåll
   </div>
</body>
</HTML>
```

```
        <title>Semantiska element</title>
</head>
<body>
        <header>
                <nav>
                        <ul>
                                <li>Hem</li>
                                <li>Info</li>

        <li>Kontakt</li>
        </ul>
                </nav>
        </header>
        <main>
                <article>En artikel</article>
                <aside>Reklam</aside>
        </main>
        <footer>
                Footer innehåll
        </footer>
</body>
</HTML>
```



Attribut

Attribut är modifierare för HTML element, beroende på element så kan de ändra bland annat beteende och utseende på elementet. Alla attribut fungerar inte på alla element, till exempel *selected* attributet fungerar bara på *option* element och markerar då detta *option* element som förvalt då applikationen laddas. Några nya attribut som ganska bra förklarar sig själva är *spellcheck*, *hidden* och

contenteditable. Förutom dessa tre och många andra attribut finns ett som står ut extra mycket, *data-* attributet som används för att inkludera egen data i sin HTML. Genom att inkludera data i sin HTML kan man göra Javascript enklare, genom att ha tillgång till metadata direkt i objektet och inte från en skild lista i Javascript. För att använda *data-* attributet skriver man in *data-attributnamn=värde*.

```
<span data-price="5" > potato </span>
```

Lagring och kommunikation

Cross Document Messaging, kommunikation mellan olika applikationer, gör det möjligt att skicka data mellan två olika applikationer som annars inte skulle kunna kommunicera med varandra. Kommunikationen görs via *post* metoden som stöds av HTTP protokollet. Applikationer kan då skicka data mellan varandra utan att använda *get* metoden och inkludera extra data i applikationens URL.

WebSocket är ett full-duplex protokoll, nytt för HTML5 som gör det möjligt att från servern skicka meddelanden till webbläsaren utan att det begärs av webbläsaren och då erbjuda kommunikation båda vägarna. Servern kan skicka en uppdateringsbegäran till webbläsaren till exempel när användaren fått ett nytt meddelande eller något annat skett på servern som användaren behöver informeras om.

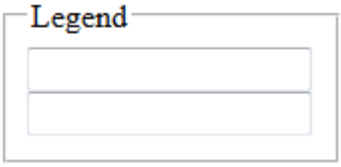
Web Storage är en specifikation som först inkluderades i specifikationen för HTML5, men senare blev en separat specifikation. Web Storage möjliggör lokal förvaring av data, upp till 10 MB per applikation och användare [4]. LocalStorage finns kvar mellan sessioner och förvaras som namn och värdepar utan utgångsdatum. SessionStorage förvaras på samma sätt men tas bort när webbläsaren stängs.

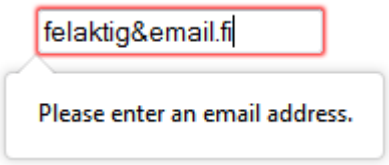
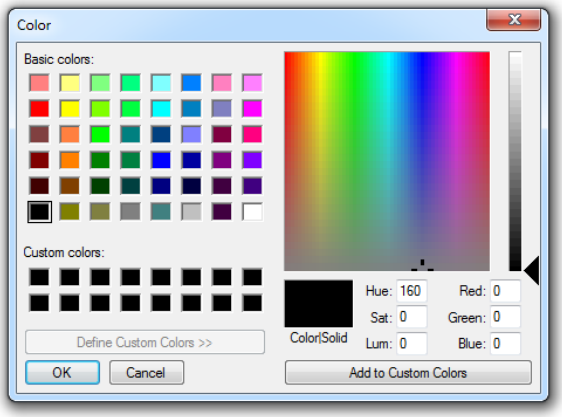
Formulär

Formulären i HTML, som står för en stor del interaktionen mellan användare och klient i HTML, får en del nya typer av *input* element och attribut. *Type* attributet i ett *input* element bestämmer vilken typ av data elementet tar emot och hur det beter sig och ser ut. De nya typerna listas i tabell 1 tillsammans med en bild på hur de ser ut i Mozilla Firefox. Element som saknar stöd i en webbläsare visas som

typen *text*, till exempel typerna relaterade till tid (*date, month, week, time, datetime, datetime-local*) har i skrivandets stund inte stöd i Firefox. Fastän en typ saknar stöd i många webbläsare är det ändå bra att använda den istället för ett textfält, dels för att stödet kan finnas där i framtiden och dels för att det gör det lättare för skärmläsare och funktionshindrade användare att känna igen vilken indata det frågas efter. De nya typerna kan, beroende på stöd i olika webbläsare, validera användarindata före formuläret skickas. Till exempel epostvalidering som tidigare behövt en del extra arbete av utvecklaren kan nu valideras direkt av webbläsaren. Om datan i fältet inte överensstämmer med fälttypen markeras rutan och en liten varning visas när formuläret skall skickas (exempel finns i tabell 1). Det nya elementet *fieldset* används för att markera ett set av indataelement, setet markeras visuellt med en ruta runt elementen. Elementet *legend* används för att sätta en rubrik i *fieldset* rutan (bild i tabell 1). I den mobila världen kan man beroende på plattform få anpassade virtuella tangentbord för olika typer av indataelement, en del smarttelefoner lägger till ”.com” i tangentbordet då ett fält av typen *url* är i fokus. Typerna *color* och *date*, beroende på webbläsare, kan ge en färg eller datumväljare vid fokus (bild på Mozilla Firefox färgväljare i tabell 1).

Tabell 1 Nya element och attribut

<fieldset> <legend>Legend</legend> <input type=”text”> <input type=”text”> </fieldset>	
<input type=”range”>	
<input type=”slider”>	Saknar stöd i Firefox (textfält)
<input type=”number”>	
<input type=”date”> (även month, week, time, datetime, datetime-local)	Saknar stöd i Firefox (textfält)
<input type=”email”>	
<input type=”url”>	
<input type=”color”>	

Exempel på felaktig input	
Färgväljare som visas vid klick på input av typen color	
<input type=”text” placeholder=”Adress”>	

Formulären får en del nya attribut som fungerar på alla indataelement. *Autofocus* attributet gör att elementet kommer i fokus då applikationen laddas, vilket innebär att användaren kan börja skriva direkt utan att själv markera fältet. *Placeholder* används för att visa en beskrivande text i indatafältet tills fältet kommer i fokus, exempel finns i tabell 1. Med attributet *autocomplete* kan man stänga av förslag från tidigare inmatningar i fälten. Ett indatafält med attributet *required* visar ett felmedelande i stil med exemplet för felaktig input i tabell 1 om användaren försöker skicka in formuläret utan att fylla i fältet.

En del element och attribut tas bort, en del funktionalitet flyttas över till CSS och vissa element försvinner helt och hållet medan några ersätts av nya.

Element vars funktionalitet flyttas till CSS:

- *basefont, big, center, font, strike, tt*

Element som tas bort:

- *frame, frameset, noframes*

Element som ersätts:

- *acronym* ersätts av *abbs*

- *applet* ersätts av *object*
- *dir* ersätts av *ul*

Attribut som tas bort och/eller flyttas till CSS:

- *align, link, vlink, alink, text* i *body* elementet
- *bgcolor, height, width, scrolling* i *iframe*
- *calign, hspace, vpace, cellpadding, cellspacing* i *table*
- *target _blank* i *a* element (länk)
- *profile* i *head*
- *longdesc* i *img* och *iframe*

HTML 5 i mobila applikationer

Antalet mobiltelefonanvändare ökar kraftigt för varje år som går. År 2014 beräknades 63,5% av jordens befolkning ha tillgång till en mobiltelefon varav 38,5% är smarttelefoner [2]. Marknadsföringen för smarttelefoner har gått från att vara fokuserad på hårdvara till utbudet av applikationer för telefonen. Det har gett upphov till applikationsbutiksmodellen (eng: Application store model) som gör det enkelt för användaren att hitta och installera applikationer på sin smarttelefon. Exempel på detta är Apples App Store [3] och Googles Google Play [4]. Dessa gör det även enklare för mindre applikationsutvecklare att få ut sina produkter till smarttelefonanvändarna vilket I sin tur ger ett bredare utbud utöver applikationerna från större utvecklare [8].

I dagens läge fungerar applikationsbutiker genom att man laddar ner applikationer som hämtas från en appstore och körs lokalt på mobiltelefonen. Nativa applikationer kan fullt ut, med användarens godkännande använda resurser så som kamera, accelerometer och information som finns lagrad på telefonen. Den största nackdelen med dessa nativa applikationer är fragmentering av versioner. Det är inte bara de olika plattformarna som behöver olika versioner, utan även samma OS kan ha betydande skillnader mellan sina versioner, som kräver extra arbete att underhålla [8]. Ett annat problem som direkt ekonomiskt drabbar utvecklarna är att de är bundna av Application stores avtal för fördelning av inkomster, vanligtvis cirka 70% åt utvecklaren [8].

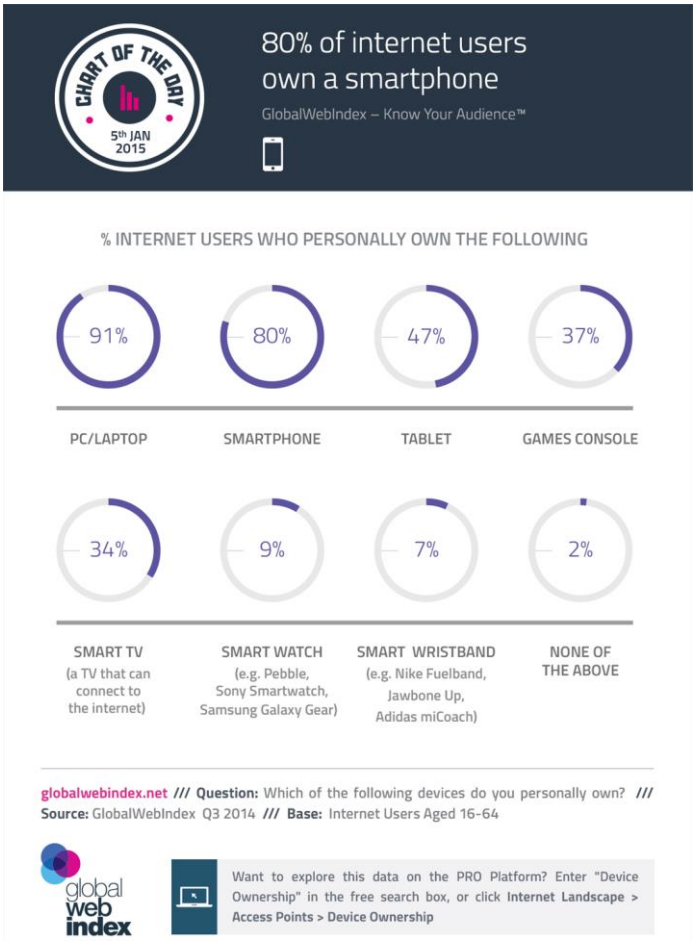


Bild 3 http://www.globalwebindex.net/blog/80-of-internet-users-own-a-smartphone

Responsiv webdesign

Responsiv webbdesign, RWD innebär att innehållet anpassar sig efter användaren och inte användaren efter innehållet. Det innebär att när miljön varierar så anpassas layout och delvis även innehåll för att fungera bra just var det visas. Till exempel storlekar definierade i pixlar hör inte hemma i en responsiv design. Mått i pixlar ersätts av relativa enheter så som procent av föräldraelementet eller *em* som är textens storlek. Om en bild har *width:50%* definierat i sin CSS, betyder det att bilden tar upp hälften av föräldraelementets storlek. Om vi definierar en bild att vara *2em* hög så innebär det att den har höjden av två bokstäver. I en responsiv

design behöver bilder och annan media anpassa sig efter skärmens och det övriga innehållets storlek med hjälp av relativa mått, men man måste komma ihåg att filstorleken på bilden är densamma oberoende av måtten den tilldelats i CSS.

Med CSS3 mediaförfrågningar (eng: media queries) går det att variera CSS mellan olika typer av skärmar. I exemplet nedan anpassas CSS reglerna endast på skärmar som har en bredd som är minst 480 pixlar. Ordet *screen* i detta fall betyder att CSS-reglerna anpassas på media visat på en skärm. För att reglerna skall gälla på en hemsida som skrivs ut på papper kan ordet *print* användas istället.

```
@media screen and (min-width:480px) { /* CSS Rules */}
```

Genom att följa tankesättet för responsiv design kan man använda samma HTML och CSS på alla plattformar och webbläsare, dock så lönar det sig ibland att ha en helt skild mobilversion av hemsidor, om den vanliga versionen är alldeles för komplex för att smidigt anpassas till mobilvärden. [2] ?

Nativa applikationer och webben

När Apple släppte iPhone var planen att applikationerna skulle implementeras med Open Web Platform (OWP) som täcker bland annat HTML och CSS, Apple släppte även utvecklingsverktygen som skulle användas för utvecklingen av applikationerna [9]. Tre år senare kunde vi se att nativa applikationer användes fullt ut, till stor del på grund av den tidens prestandaproblem i OWP. Med dagens utbud av mobiltelefoner och surfplattor är det dyrt och tidskrävande att utveckla och underhålla en applikation i nativ kod. I bild 2 finns exempel på de kunskaper som krävs för att underhålla en applikation för nio olika mobiloperativsystem. Det är inte bara programmeringsspråket utan även utvecklingsmiljöerna och möjligheterna de erbjuder som skiljer sig. Lösningen för samma problem i olika miljöer kan variera mycket och leda till kodbaser för samma applikation med mycket varierande innehåll. Det enda som de olika plattformarna har gemensamt, i alla fall i moderna smarttelefoner och surfplattor, är att de kommer med en modern webbläsare.

Idag är alternativen till nativa applikationer att utveckla applikationerna i HTML5 med CSS3 och Javascript. Endera genom att lägga upp applikationen på en server som en webbsida eller genom att använda ett ramverk (eng: framework) som endera använder webbläsaren för att exekvera HTML i ett läge som ser ut och beter sig som en nativ applikation eller själv hanterar HTML koden, till exempel PhoneGap[link]. HTML5 applikationer existerar i en sandlåda (eng: sandbox) och har vanligtvis inte direkt tillgång till systemets resurser. Ramverk som PhoneGap ger tillgång till de nativa funktioner och sensorer som en applikation i en webbläsare vanligtvis inte har tillgång till så som lagringsmedia, kamera och accelerometer. Möjligheterna att komma åt dessa tjänster utvecklas dock hela tiden, nu går det till exempel att komma åt geolokalisering direkt från en HTML5 applikation. W3C har till och med en arbetsgrupp, DAP (Device API Working Group [11]) som arbetar för att utveckla åtkomsten för webbapplikationer till de

Required skill sets for nine mobile OSs.

Mobile OS Type	Skill Set Required
Apple iOS	C, Objective C
Google Android	Java (Harmony flavored, Dalvik VM)
RIM BlackBerry	Java (J2ME flavored)
Symbian	C, C++, Python, HTML/CSS/JS
Windows Mobile	.NET
Window 7 Phone	.NET
HP Palm webOS	HTML/CSS/JS
MeeGo	C, C++, HTML/CSS/JS
Samsung bada	C++

Web vs Native arikln

tjänster som de olika plattformarna erbjuder.

Flaskhalsen för mobilapplikationer är prestandan, där de begränsande faktorerna är nätförbindelse och hårdvara. Ett argument för nativa applikationer är att HTML5, CSS och Javascript är tolkade språk och språken i nativa applikationer är färdigt kompilerade och då självklart snabbare. I dagens läge så gäller detta argument bara när det kommer till mer krävande applikationer så som spel och

bildbearbetning. Skillnaden i prestanda för övriga applikationer i en bra byggd HTML5 applikation och en nativ applikation är minimal. Den kompilerade nativa koden ritar direkt pixlar på skärmen, HTML och CSS tolkas först av webbläsaren och ritas sedan på skärmen. En del variationer kan uppstå mellan olika webbläsare men oftast handlar det om såpass små saker att det inte påverkar användarupplevelsen mellan olika plattformar.

Det som gör en HTML5 applikation krävande för systemet är inte egentligen HTML5 utan CSS och Javascript. Javascript sköter om beräkningar och interaktion med användaren och CSS sköter om det visuella i applikationen. I jämförelse med nativa applikationer är dock inte Javascript så krävande som det kan låta. I konkurrensen mellan webbläsare är Javascripts virtuella maskin i fronten för kampen mellan webbläsare. Microsoft, Google, Opera och Mozilla jobbar alla hårt för att erbjuda snabbare och bättre Javascriptimplementationer i sina webbläsare i jämförelse med de andra [4].

Dynamiska skuggor, färggradienter och fina animerade övergångar i CSS3 kan belasta en telefon till den grad att hårdvaran inte hänger med och användarupplevelsen försämras. Speciellt CSS-gradienter är kända för att inte prestera bra i mobila applikationer. Allt blir dock hela tiden bättre. En del tillverkare erbjuder hårdvarustöd för vissa funktioner, till exempel iOS har hårdvaruacceleration för CSS-transformationer som ger mjuka rörelser och övergångar mellan visningslägen i applikationen. Det går självklart att använda dessa funktioner, men det lönar sig att tänka igenom ifall det förbättrar användarupplevelsen eller distraherar från applikationens egentliga uppgift.

Avslutning

Litteratur

1. **W3C.** [Online] [Viitattu: 31. April 2015.] http://www.w3.org/.

2. **Emarketer.** Smartphone Users Worldwide Will Total 1.75 Billion in 2014 - See more at: http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536#sthash.fTQcKumD.dpuf. *Emarketer.* [Online] http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536.

3. **Boutin, Paul.** Google VP: "We're not rich enough" to support individual smartphones. *Venturebeat.* [Online] Juli 2009. http://venturebeat.com/2009/07/16/google-vp-were-not-rich-enough-to-support-smartphones-individually/.

4. *Mobile application development: web vs. native.* **Andre Charland, Brian Leroux.** 5, s.l. : Communications of the ACM, 2011, Osa/vuosik. 54. 0001-0782.

5. What is CSS? *W3.* [Online] http://www.w3.org/standards/webdesign/htmlcss#whatcss.

6. **Lyza Danger Gardner, Jason Grigsby.** *Head First Mobile Web.* s.l. : O'Reilly Media, 2011. 978-1-4493-0266-5.

7. **Christopher Murphy, Richard Clark, Oli Studholme, Divya Manian.** *Beginning HTML5 and CSS3.* s.l. : Apress, 2012. 978-1-4302-2874-5.

8. **Roberto Brunetti, Vanni Boncinelli.** *Advanced Windows Store App Development Usinh HTML5 and JavaScript.* s.l. : Microsoft Press, 2013. 978-0-7356-7680-0.

9. **Apple.** *Official Apple Store.* [Online] http://store.apple.com.

10. **Google.** *Google Play.* [Online] https://play.google.com.

11. *HTML 5 in Mobile Devices -- Drivers and Restraints .* **Juntunen A., Jalonen E. , Luukkainen S.** s.l. : System Sciences (HICSS), 2013 46th Hawaii International Conference on System Sciences, 2013. 978-0-7695-4892-0 .

12. **W-M, Lee.** Build Web Apps for iPhone using Dashcode. *Mobiforge.* [Online] http://mobiforge.com/developing/story/buildweb-apps-iphone-using-dashcode.

13. Device APIs Working Group. *W3.* [Online] http://www.w3.org/2009/dap/.

14. **Hogan, Brian P.** *HTML5 and CSS3.* s.l. : The Pragmatic Programmers, 2011. 978-1-93435-668-5.

15. A vocabulary and associated APIs for HTML and XHTML W3C Recommendation 28 October 2014. *W3.* [Online] http://www.w3.org/TR/2014/REC-html5-20141028/forms.html.

Bilaga 1

http://www.w3schools.com/HTML/HTML5_new_elements.asp

Tag	Description
<article>	Defines an article in the document
<aside>	Defines content aside from the page content
<bdi>	Defines a part of text that might be formatted in a different direction from othe text
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<figcaption>	Defines a caption for a <figure> element
<figure>	Defines self-contained content, like illustrations, diagrams, photos, code listings etc.
<footer>	Defines a footer for the document or a section
<header>	Defines a header for the document or a section
<main>	Defines the main content of a document
<mark>	Defines marked or highlighted text
<menuitem>	Defines a command/menu item that the user can invoke from a popup menu
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links in the document
<progress>	Defines the progress of a task
<rp>	Defines what to show in browsers that do not support ruby annotations
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>	Defines a ruby annotation (for East Asian typography)
<section>	Defines a section in the document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time
<wbr>	Defines a possible line-break