*Mobile-first or Desktop-first, a study about the responsive approaches towards web development*

**1. Bakgrund**

Mobilutvecklingen har under de senaste åren gått fort framåt, vilket har lett till att mobiler numera utvecklas till att fungera likt en handburen minidator. Huvudsyftet med mobiler är inte längre att bara kunna kommunicera med andra, utan även att bland annat ha möjligheten att få ut information genom webben. Att en websida inte längre endast ses i en skrivbordsskärm har gjort att nya webblösningar krävs. En utvecklare måste ha i åtanke att webbdesignen bör fungera för en användare som sitter på kontoret och läser sidan från en skrivbordsskärm, likaväl för en användare som sitter på tåget och läser sidan från mobilen. *Responsive web design* är en lösning vilken beroende på skärmstorlek renderar samma sida på olika sätt. I **responsive** web design har det fokuserats på två olika metoder, *Mobile-first* och *Desktop-first*. Metoderna grundar sig på att utveckling sker för en typ av skärm först för att sedan utifrån det utveckla så att den sidan även passar för andra skärmar. Det finns mycket kunskap om metoderna var för sig, men inget underlag för valet av metod i olika situationer och inom olika områden. Mobilanvändarna ökar för varje dag medan skrivbordsanvändare minskar, detta leder till att det krävs kunskap om när metoderna appliceras bäst för att få effektivare lösningar inom webutveckling.

**1.2 Mobil användare**

Sedan 2007 då Apple visade upp den första versionen av iPhone har utvecklingen för **smartphones** eskalerat markant. Smartphones ger dig möjligheten att utföra datorliknande handlingar som att få ut information från webben och använda nätbaserade applikationer. Tidigare mobiler har delvis haft den funktionen men fokus har varit mestadels på musikspelare och kamera, då surfande gick trögt och var icke användarvänlig.

Med smartphones, vilket innebar större pekskärmar, gjorde åtkomsten till internet på mobilen mer användarvänlig och har under de senaste åren utvecklats till en oundviklig funktion i dagens mobiler.

Dagens användare av mobiler sträcker sig mellan 15-79 år gamla, av dessa är det cirka 40 % som använder sig av mobilt internet, i jämförelse med 2008 har det ökat med över 50 %. Telefonoperatörer har anpassat sig i den nya marknaden och erbjuder abonnemanger vilka innehåller en fast kostnad för surf på ett 3G nät. Då mobilsurf anses som en nödvändig funktion, sätts även krav hos mobiloperatörer, 3G nätet skall ha hög åtkomlighet och ha en hög hastighet. Effekten av att mobilanvändarnas åtkomst till internet ökar, leder till mobilt internet används mer och äldre tjänster ersätts. En tidning i tunnelbanan är inte alls lika vanligt nu som för fem år sen, tidtabeller vid busshållplatsen spelar är inte lika viktiga som förr och att checka-in inför en flygresa behöver inte nödvändigtvis ske via en check-in disk.

Konkurrensen som finns i dagens mobilmarknad har tvingat ledande företag att skapa mobiler med ny teknik till allt lägre priser. Pris, tillgänglighet och användbarhet har gjort att användning av internet via mobilen i världen närmar sig antalet användare av internet via en dator. Analytiker som förutspådde mobilen till att slå i marknaden har i dagsläget fått det bekräftat och förutspår att användare av mobilt internet i världen kommer att passera antalet användare från av **Desktop** internet [fotnot 2] i 2014 år. Detta kräver från webbplatser att följa målgruppen användare och anpassa webbsidor utifrån de enheter som webbsidor ses ifrån, vilket är till stor del är från mobilen.

**1.3 Desktop användare**

Att mobil användare ökar för varje dag har till viss del betydelse i minskningen av antalet **desktop** användare. Men datorer har i dagsläget funktioner som gör det osannolikt för mobiler att helt kunna ersätta datorer, funktioner som kräver att man sitter vid en dator och på så sätt sker informationssökandet i webben via **desktop**.

Webbdesignen på en mobil är kompakt och uppfyller den nödvändigaste användbarheten.

På desktop är designen mer informationsrikt, vilket ger en större inblick till websidans innehåll och en mer simpel navigering i webbsidan. Det gör att en användare beroende på komplexitet och säkerhet av handlingen väljer att utföra den via en desktop, än via mobilen. En sådan handling kan t.ex. vara bankärenden eller web shopping. Om bankärendet är en simpel koll av saldo eller överföring mellan egna konton, är mobilen en smidig enhet att använda sig av. Däremot om handlingen innebär att betala räkningar eller överföra stora summor pengar väljer man att utföra det via desktoppen, då det ger en större säkerhet och en mer simpel navigering på websidan. Även web shop faller i samma kategori, då användare väljer att söka information om produkten via mobilen, men väljer sedan att utföra köpet via desktop.[fotnot 1] Detta tyder på att användare av mobilt- eller desktop internet inte är endera, utan är användare utav båda, beroende på situation, miljö och kontext.

Det finns ingen grundlig faktum som tyder på att mobilt internet kommer att ta över all desktop internet användande, endast att de kan bli fler. Därför krävs det att man förstorar vyerna kring webutveckling och lösningar som gynnar både mobil- och desktop användning av internet. För även om mobila användare är fler, så går inte att förbise desktop användare.

[Bild på kurva]

**1.4 Responsive Web design**

Responsive web design är ett koncept som innebär att gränssnittet på en websida ändras beroende på skärmstorlek, vilket således ger möjligheten att ha olika layout på en och samma websida anpassat efter en enhet. Konceptet definierades av Ethan Marcotte (2010) i en artikel kallat A List Apart, som sedan blev en del av boken ”responsive web design” där teorier och praktiska exempel används för att förklara begreppet.

Syftet med Responsive Web Design är att kunna rendera olika delar av sidan beroende på skärmstorlek för att ge en optimal vy för den enhet websidan ses genom. Om tre artiklar renderas bredvid varandra på en desktop site, så vore det optimala för en mobil vy, att kunna renderar artiklarna under varandra och även skala ner de till en rimlig storlek, för att på så sätt undvika att artiklarna försvinner, inte visas helt och hållet eller tar för stor plats på skärmen.

Då layouten på sidan ändras behövs inga särskilda versioner för varje enhet, vilket gör det möjligt för en webbutvecklare att på ett enkelt sätt utföra en ändring i en fil, istället för antalet versioner. Däremot krävs en flexibel grundlayout för att responsive web design skall fungera, vilket enligt Ethan görs genom tre grundtekniker, Fluid Grid, Fluid Images och Media Queries.

**1.4.1 Fluid Grid**

På en webbsida kan storleken på element definieras i olika sätt. Ett vanligt sätt definiera element är med bredd eller höjd i pixlar. När det definieras i pixlar betyder det att storleken på elementet är förutbestämd vare sig upplösning eller storlek på skärm. I tidigare skede informerade webbutvecklare i vilken upplösning som renderar sidan på bästa sätt och sedan var upp till användaren att ändra upplösning på skärmen som användes. I dagsläget finns många olika skärmar, och många olika alternativ till upplösning vilket gör en sida skapad för en fast upplösning inte är lika användbart. Fluid Grid är en teknik vilken använder sig procentsatser istället för pixlar. Det ger en mer flexible layout där elementen förstoras eller förminskas relativt till skärmens storlek, eller i själva verket, relativt till storleken på webbläsaren. Om man vill att ett element skall täcka hela bredden på skärmen, sätts bredden till 100 %, en fjärde del av skärmen, 25 % osv. det gör att storleken på elementet endast ändras relativt till skärmen. Ett sätt att räkna ut elementens storlek för att få en Fluid Grid, är att dividera pixlarna på elementet med pixlarna på sidans totala bredd, detta ger ett resultat i procent vilket gör att elementet anpassar sig beroende på sidans totala bredd. Den sidan man har skapat ser elementet relativt stor till skärmen som till andra skärmar.

**1.4.2 Fluid Images**

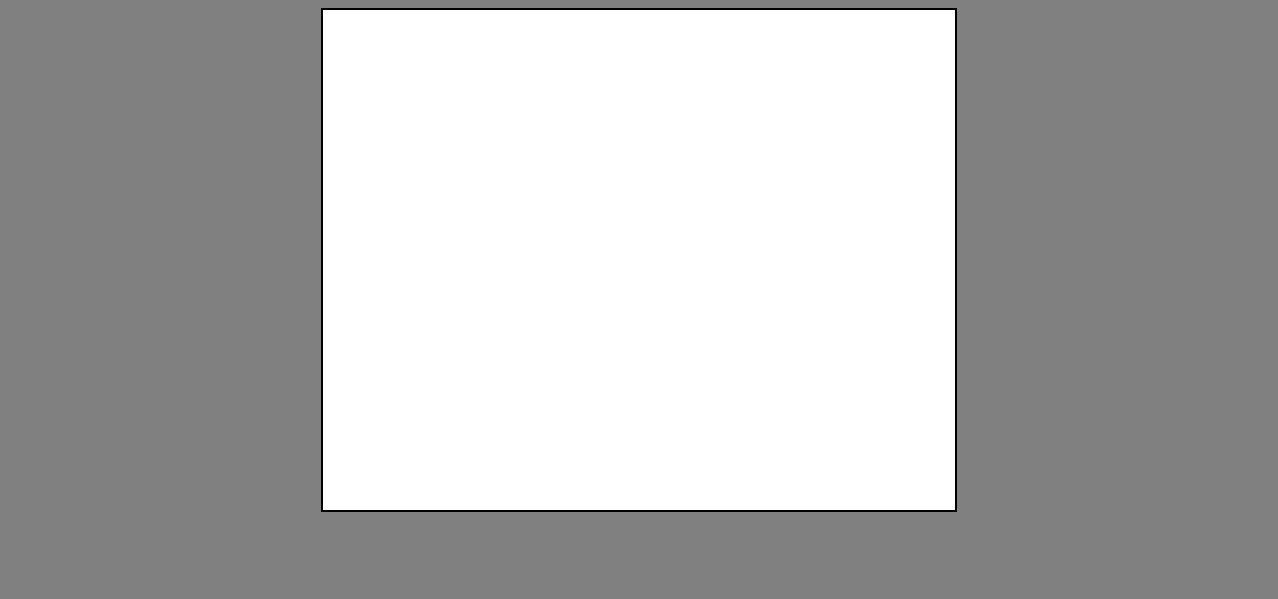
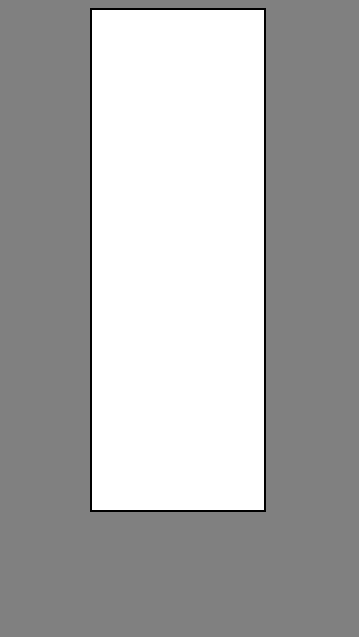
Fluid Images bygger på samma princip som Fluid Grid, däremot är lösningen mer komplext då innefattar skalning av bilder. Om en bild skalas på fel sätt leder detta till att bilden ser för utsträckt eller för intryckt ut, men om skalning inte görs så kan en bild i princip ta upp all plats på webbsidan och även försvinna ut i kanterna. Istället använder man sig av flexible hållare med en önskad storlek, därefter sätts bilden in med egenskapen ”max-width:100 % “. Genom att sätta egenskapen ”max-width:100 % “ på bilden, säger man att bilden bara får anpassa sig efter hållaren, samtidigt som den får en normal skalning när storleken på webbläsaren minskar och hållaren krymper, vilket är det är det width:100 % gör. Max i ”max-width” ser till bilden aldrig blir större än bildens verkliga storlek, vilket gör att bilden får ett in zoomat intryck där,om det inte är vektor grafiska bilder, gör pixelkanterna synliga. Däremot så kräver det oftast att man har en bild som är i en önskad storlek redan från början. Om man har en för liten bild kommer den inte att förstoras då den max kan vara 100 % av bildens ursprungliga storlek.

**1.4.2 Media Queries**

Fluid Grid och Fluid Images skapar tillsammans en del av en responsiv sida då storleken på elementen i sidan renderas utifrån storleken på skärmen. Minskningen eller förstoringen av elementen fungerar däremot endast till en viss gräns. Är storleken på webläsaren för liten, finns risk att element kommer för nära varandra och skapar konflikt i layouten, vilket gör att element hamnar på fel ställen och ger en vy som designmässigt ser förstörd ut. Blir storleken på webläsaren för stor, finns risk att *max-width* på bilderna uppnås och stannar i storlek medan andra element följer förstoringen hos webbläsaren, blir för stora och skapar en osymmetri i layouten. Media Queries är en lösning som tillåter element att ha olika värden beroende på skärmstorlek.

Inom responsive web design är bredden och höjden viktiga egenskaper då de avgör hur mycket av en sida skall synas på webläsarfönstret. Media queries är anrop som görs i CSS filen, där man kollar antingen höjden och eller bredden för att utföra nödvändiga ändringar i layouten för att upprätthålla en bra design. Tillsammans med Fluid Grid och Fluid Image skapar Media Queries en responsiv web design.

Vid användning av fluid grid och fluid image sätts egenskaper hos element i procentform, t.ex 50%. Centreras rutan, blir resultatet en ruta i mitten av skärmen som täcker 50% och har 25% utrymme på var sida. Detta ger en bra design för skrivbordskärmar, men är inte nödvändigtvis en bra design för mobilskärmar. 50% av en mobilskärm som går till spillo åt kanter för den centrerade rutan är mycket med tanke på hur kompakt information på en mobilsida är. Därav en försämring på design och användbarhet när den ses ifrån mobilen.

Denna design har endast en ruta med egenskaperna:

*div {*

*margin: 0 auto;*

*width:50%;*

*height:500px;*

*border: 2px solid black;*

*}*

Designen fungerar bra för skrivbordskärmar(till vänster), då rutan är centrerad och symmetrisk, 25% utrymme på var sida av rutan ger en design som blir läsvänlig och användbart. Däremot är den inte lika optimal när skärmen blir mindre(till höger). Rutan fortfarande 50% av skärmen, men får designen att se kompakt och tillåter inte mycket information i rutan.

Med media anrop i CSSen tillåter vi att värden hos ”div” elementet skrivs över när bredden på webläsaren underskrider 380px.

*@media all* ***and*** *(****max-width: 380px****){*

*div {*

*margin:0;*

*width:100%;*

*}*

*}*

Med media anropet kommer ”width” och ”margin” skrivas över när skärmstorleken är 380px eller mindre. Detta leder till att sidan får andra design värden när den ses från en mobil och andra enheter där skärmen har max 380px i bredd. Det ger en mer optimal lösning i både användbarhets perspektiv och design, då man kan få plats med mer information och det är lättare för en användare att navigera.



På det sättet kan man med samma CSS-fil låta element ha olika värden på egenskaper utifrån storleken på webläsaren.

Fördelen med RWD varför inte native???

1.5 Tekniker, Vektryg

1.5.1 HTML

1.5.2 CSS

1.5.3 javascript