Responsiv webbdesign

Kandidatavhandling i datavetenskap

Anders Vikstrand 27307

Handledare: Dragos Truscan

Åbo Akademi

2014

Abstrakt

Responsiv webbdesign är en teknik för design av webbsidor som har som mål att beroende på användarens skärmstorlek anpassa sidans innehåll utan att ha en mobilversion av sidan. Detta för att kunna ge besökaren en smidig och optimal användarupplevelse med enkel navigering oberoende av vilken sorts enhet sidan besöks av, dvs. en mobil enhet, surfplatta eller bordsdator. Responsiv webbdesign baseras på tre olika tekniker, flytande layout, flexibel media och media förfrågningar.

Denna avhandling kommer att gå närmare in på vad exakt responsiv webbdesign är och varför det finns ett behov för denna metod. Huvudsakligen kommer fokus att vara på de tre specifika teknikerna som responsiv webbdesign baserar sig på.

I introduktionen kommer jag att gå igenom vad nyttan med responsiv webbdesign är och varför det är en viktig del i modern webbdesign och fördelar och nackdelar med tekniken. I därpå följande sektioner går jag då närmare igenom de tre teknikerna som utgör grunden för responsiv webbdesign, hur dessa tekniker används samt vad som är deras för- och nackdelar. Som avslutning ser jag lite på varianter av responsiv webbdesign och hjälpmedel.

Källorna kommer till största delen att bestå av artiklar från nätet och förhoppningsvis minst en bok.

Innehållsförteckning

1. Introduktion

1.1 Vad är responsiv webbdesign?

Responsiv webbdesign är en teknik inom webbdesign vars huvudmål är att anpassa en webbsida och dess layout enligt storleken på skärmupplösning de enheter som besöker sidan har, dvs. mobila enheter, surfplattor eller bordsdatorer och göra det utan att behöva ha en specifik mobilversion av sidan. Det är en kombination av tre tekniker som används för att nå målen med responsiv webbdesign. Dessa tre är: flytande layouter, flexibel media och media förfrågningar. Genom att kombinera och använda dessa tekniker tillsammans får man en sida att ändra layout vid behov. En sådan ändring av layout kan exempelvis vara att gå från en fyra-kolumns-layout vid större skärmupplösningar som 1024px och uppåt till två kolumner vid 800px och en kolumn vid 480px.

Fig.1: Exempel på en 4 kolumners layout som förvandlas till 2 kolumner och till sist en kolumn beroende på skärmupplösning

Första bilden i figur 1 motsvarar här en sidas layout på bredare skärmar, den andra på mellanstora och den sista på väldigt smala skärmar. Vad som kan anses vara bra brytningspunkter för att ändra på layouten diskuteras senare i avhandlingen.

Det som anses vara grunden till den här tekniken är en artikel som Ethan Marcote skrev 2010 med namnet ”Responsive Web Design” [1] och var det som öppnade mångas ögon för responsiv webbdesign och dess potential.

1.2 Varför responsiv webbdesign behövs

I sin artikel nämner Marcote statistik från Morgan Stanley som säger att användningen av internet genom mobila enheter kommer att vara större än genom bordsdator inom fem år, dvs. runt 2015, vilket gör responsiv webbdesign mycket aktuellt i dagens läge. Det förekommer väldigt stora variationer i skärmstorlekar bland de enheter som används på nätet och mobila enheter har ofta två visningslägen (porträtt eller landskap) och från detta har behovet av en mera flexibel variant av webbdesign uppkommit.

En liten fingervisning om hur den mobila användningen på nätet utvecklas ser man i statistiken som Forbes tar upp i en artikel om responsiv webbdesign med namnet ”Why Your Business Needs A Responsive Website Before 2014” [2].

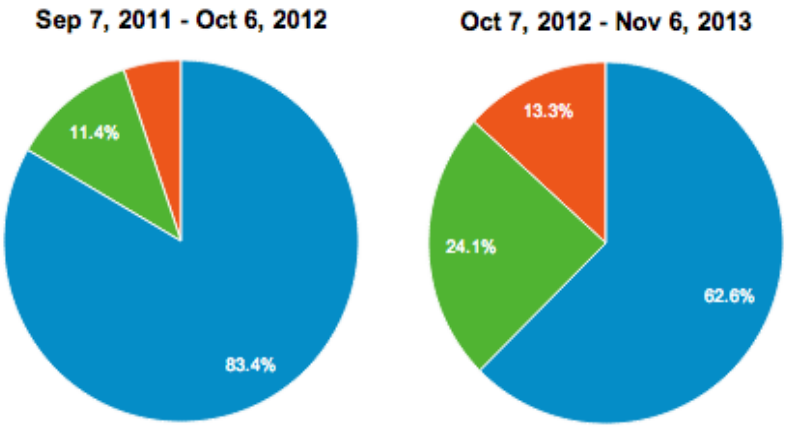

Fig.2: *Statistik över vilka enheter som besökt en sida och hur dessa ökat på ett år från Forbes artikel om responsiv webbdesign [2]. Grönt = mobila enheter, orange = surfplattor, blå = desktop. Mera information får man från Forbes artikel.*

Att det behövs sidor som kan anpassa sin layout kommer från att webbsidor väldigt ofta inte har någon mobilversion alls. En smarttelefon hanterar detta genom att zooma ut fullt för att få hela sidan att visas i smarttelefonens skärm. Detta leder till att användaren får zooma in för att kunna läsa eller försöka navigera sig rätt på sidan vilket gör att användarvänligheten kan bli lidande. Orsaken till att detta kan hända är helt enkelt att sidan är för bred för att

rymmas in i den nuvarande skärmstorleken på enheten som man besöker sidan med.

1.3 Fördelar och nackdelar

Det som driver responsiv webbdesign är behovet av att kunna få en relativt enkel lösning på det problem som uppstått i och med att surfplattor och smarttelefoner fått så stort genomslag de senaste åren. Vad fördelarna är med responsiv webbdesign jämfört med att istället erbjuda en specifik mobilversion av en webbplats kan sammanfattas på följande vis:

- responsiv webbdesign är lättare att använda och att underhålla då man har endast en kodbas och en url, medan specifika mobilversioner har två kodbaser och två url:s att hantera.
- man täcker existerande och kommande modellers skärmstorlekar relativt enkelt genom responsiv webbdesign

Sedan finns det naturligtvis även nackdelar och specifikt är det ett stort problem existerar: en responsiv sida kan bli lidande av långa sidladdningstider på mobila enheter som ofta har låg bandbredd detta i jämförelse med en för ändamålet optimerad mobilversion där bilder och annan media är anpassade för mindre skärmstorlekar. Samma prestandaproblem gäller naturligtvis också för bordsdatorer men där kan man relativt säkert anta att bandbredden är tillräcklig.

Problemet med laddningstiderna beror på att man alltid tar ner all allt HTML-innehåll på en sida designad med responsiv webbdesign. Detta trots att vissa delar innehållet tas bort för att optimera det för mobila skärmstorlekar, exempelvis om man gömmer ett element via CSS deklarationer och detta element innehåller bilder så görs fortfarande requests till servern för att hämta dessa fastän de inte visas på den mobila enheten.

Orsaken till att bilder är problematiska i responsiv webbdesign är att de ofta levereras via <img src=”filnamn”> element i HTML-koden som inte kan modifieras via CSS. Dessa bilder kan vara av väldigt hög kvalitet (över 1mb i värsta fall) som är menade för bordsdator layouter med högre skärmupplösning och oftast bättre bandbredd och det är lätt att se hur detta kan bli en flaskhals om man är på en mobiltelefon med dålig 3G-uppkoppling.

2. Introduktion till viewports

En viewport har en viktig roll i responsiv webbdesign och förenklat är detta helt enkelt storleken i pixlar på webbläsaren man använder sig av och är den i slutändan den storlek som alla element på en webbsida ställer sina breddinställningar i relation till.

Denna enkla förklaring stämmer bra för bordsdator-webbläsare men på mobila enheter är detta koncept lite mera invecklat, här kan nämligen viewporten delas in i tre delar: visuell, layout och ideal viewport.[7] [8] [9] [10] Peter-Paul Koch definierar de två första på följande vis (egen översättning från engelska) i sina artiklar om viewportar:

Visuell viewport:

- *”Den visuella viewporten är den del av sidan som för närvarande visas på skärmen. Användaren kan scrolla för att ändra vilken del av sidan han ser, eller zooma för att ändra storleken på den visuella viewporten”*

Layout viewport:

- *”På bordsdator är layout viewporten lika bred som webbläsarfönstret. På mobila enheter har den ett grundvärde som är mellan 768 och 1024 pixlar och webbutvecklare kan ställa dess värde genom att använda meta viewport funktionen”*

Den tredje viewporten är den mest aktuella för responsiv webbdesign, den ideala viewporten som Koch kallar den. Den här är den som anger den ideala storleken för en sida på enheten och den varierar per enhet. Det vi vill veta är mängden så kallade CSS pixlar som ryms en i en enhet och det är det vi får via den ideala viewporten.

För att ta reda på den ideala viewporten så används koden i figur 3, vilken är mycket viktig för att responsiv webbdesign ska fungera väl på enheter med mindre skärmar. Det som koden gör är att få enheten att ställa sin layout viewport att vara lika bred som enhetens ideala viewport. [4][7] [8] [10] [11] [12]

```
<meta name="viewport" content="width=device-width, initial-
                               scale=1.0">
```

Fig.3: *Kod som ställer viewportens bredd på mobila webbläsare*

Användning av denna kod innebär att en sida aldrig zoomas ut när en enhet kommer till sidan då layout viewporten är lika bred som enheten, istället för att ha ett fast värde på mellan 768 och1024 pixlar. På egen hand gör alltså denna kod en hel del och sätter en viktig grundsten för responsiv webbdesign.

I figur 4 ser man exempel på layout viewporten i sin grundinställning där sidan zoomats ut på enheten till vänster för att rymmas in i layout viewporten medan den på enheten till höger är 320px vilket är storleken på dess ideala viewport och kan ställas in genom koden i fig. 3.

Fig.4: *Exempel på hur olika viewport inställningar påverkar. Bilderna från Apple [12]*

2.1 Pixeltäthet

En intressant utmaning som responsiv webbdesign står för är framstegen som görs med högupplösta skärmar, både på desktop och mobila sidan, och hur man ska hantera dessa i en responsiv miljö.

Problemet som uppstår här är att det finns enheter som har samma fysiska mått och ideal viewport men pixeltätheten varierar så att vissa enheter kan ha dubbelt mera fysiska pixlar i sin skärm vilket gör att kvalitén på media (bilder igen) kan försämras i vissa situationer ifall detta inte hanteras på ett korrekt sätt.

Ett exempel på hur en bild kan försämras på en enhet med mera fysiska pixlar är en situation där två enheter matchar samma media förfrågan för att de har samma storleks viewport och denna förfrågan levererar en 320px bred bild åt enheterna. Då skalar den enhet som har dubbel pixeltäthet upp denna till 640px med resultatet att bildkvaliteten försämras och blir suddig. Orsaken är att i denna bild täcker en CSS pixel två fysiska pixlar i den pixeltätare enhetens skärm.

3. Flytande layout

En flytande layouts (även kallad flexibel) huvudsakliga uppgift är att få innehållet på en sida att "flyta” med variationer i viewportens storlek. Sidan ska med andra ord klara av att anpassa sig automatiskt till dessa variationer, som kan bero på att en surfplatta eller smarttelefon vrids från porträttläge till landskapsläge eller att en webbläsare på en bordsdator förstoras eller förminskas. I motsatts till en fast layout som använder sig av pixelvärden för bredd och marginaler så använder sig en flytande layout sig av procentuella inställningar för dessa mått.

För relationer mellan element på en sida kan man prata om förälder-barn relationer där ett förälder element bestämmer ett värde för sin bredd (i proportion till sin egen förälder) och barn till denna förälder ställer sin bredd (och eventuella marginaler) i proportion till förälderns bredd. Ett barn som ska fylla hela föräldra-elements bredd sätter detta värde till 100 %. Om vi istället har

fyra barn (fyra kolumner, se fig. 1 eller fig. 4) så sätts deras bredd att vara 25 % för att fylla hela förälderns bredd, inga marginaler i detta fall.

Om man letar ”uppåt” i detta släktträd så är det alltså viewporten som är roten och enväldigt styr hur alla dessa element ställer sin bredd beroende på proportionerna mellan alla elementen. På detta vis får en ändring av storlek på viewporten alla element som finns under den att automatiskt anpassa sig till den nya storleken. Detta fungerar alltid så länge det yttersta elementet på sidan (närmast till <html> blocket) har sin bredd att vara 100 %, och alla element under detta också använder sig av procentuella värden på sina breddinställningar.

3.1 Från fast till flytande layout

Att gå från en fast till en flytande layout är relativt enkelt och åstadkommas genom att använda den enkla formeln som finns i figur 5. Target är elementet man vill beräkna det procentuella värdet för, context är elementet som target ställer sin bredd i proportion till, med andra ord föräldraelementet.

target / context = result

Fig 5: *Formel för att beräkna proportioner mellan element*

Om man anpassar en tidigare färdigställd webplats som använder en helt fast layout till en responsiv kan det bli ett väldigt drygt projekt med en hel del användande av formeln.

För att visa detta skulle fungera i praktiken kan vi se på illustrationen i figur 6. Denna hypotetiska sida har en viewport som är 1366 pixlar bred med ett container element för sidans innehåll som är 1024 pixlar brett. Där finns även en sektion med fyra kolumner som vardera är 246 pixlar breda med en vänster/höger marginal på 8 pixlar. I den undre delen av figuren ser man hur relationerna mellan elementen ger oss de värden vi behöver för att bli flytande. Efter att pixelvärden ändrats till de procentuella i CSS deklarationerna har vi en flytande layout med samma skala som den fasta layouten.

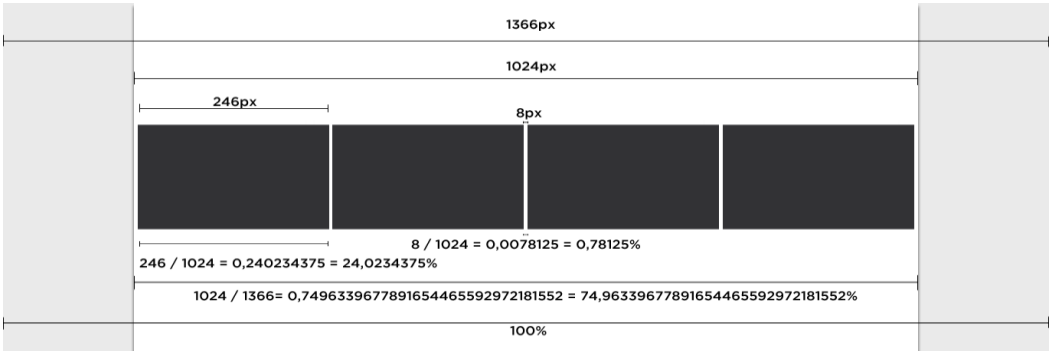

Fig.6: *Exempel på relationer mellan element och beräkningar av dessa*

En viktig fråga som dyker upp här är hur man ska hantera decimaler i en flytande layout. Hur många decimaler ska man ta man egentligen ta med från beräkningar som i exemplet och hur ska dessa avrundas eller ska man helt låta bli att avrunda.

3.2 Problem med flytande layouts

3.2.1 Stresspunkter

En flytande layout har ett problem som gör att den inte kan på egen hand hantera layouten för alla olika skärmstorlekar som existerar. Detta problem är att en flytande layout bara till en viss gräns klarar av att förstoras eller förminskas och när denna gräns passerats så påverkas layouten negativt på olika sätt beroende på om viewporten blivit bredare eller smalare. Denna gräns kallar Marcote för en stresspunkt. [1] [3]

Det som specifikt händer i stresspunkterna är att element på sidan blir för smala (fig. 7) eller för breda för att passa bra i den ursprungliga designen. Exempelvis bilder kan klippas av eller gå över sina behållarelements kanter, paragrafer ändras då deras radutrymme minskar vilket leder till att textrader börjar brytas av såpass mycket att det blir väldigt få ord per rad som i sin tur leder till en försämrad läsbarhet. Media förfrågningar bidrar med lösningar till en stor del av de problem som uppvisas i stresspunkterna.

Fig.7: *Element som blivit för smala (jfr. Fig 6)*

3.2.2 Pixelavrundning

Att pixelavrundning är ett problem beror på skillnader mellan hur olika webbläsare hanterar denna avrundning och dessa har uppstått på grund av dåliga standarder inom industrin.

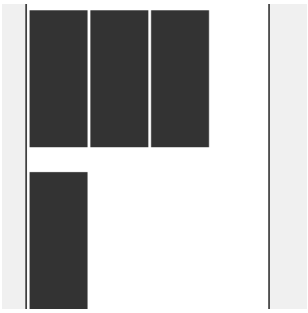

Fig.8: *Kolumnrad som brutits av pga. avrundningsproblem i IE 7 (jfr. Fig 7)*

De flesta webbläsare avrundar ned vid pixelavrundning men i vissa fall har det förekommit att webbläsare avrundat upp (Internet Explorer 6 och 7). Felavrundningar kan leda till att kolumnrader kan brytas av på ett felaktigt sätt (Fig. 7) pga. att pixelavrundning har gjort så att alla elementen inte ryms bredvid varandra.

4. Flexibel media

Flexibel media innebär att få bilder som levereras genom <img> element, strömmande video och dylika externa tjänster som levereras genom exempelvis <iframe> element att uppföra sig väl i en flytande miljö. Dessa media element lider alla i grunden av samma problem som en flytande layout överlag men med lite variation.

4.1 Bilder

Problemet med bilder är att de kan klippas av opassligt av sin behållares kanter när viewporten blir smalare eller växa över kanterna på sin behållare när viewporten blir bredare. Som tur är behövs bara en simpel rad CSS kod för att åtgärda dessa

```
img {max-width:100%;}
```

Fig.9: *En simpel CSS deklaration för att få bilder att bli flexibla*

problem och efter att man implementerat kodsnutten i figur 9 [3] uppför sig bilder som ett vanligt element i en flytande layout. Max-width har bra stöd i alla webbläsare. [17]

Koden i figur 9 gör så att en bild som laddas in via ett <img> element kan skalas upp och ner i en flytande layout utan att växa över sin behållares kanter eller klippas av. När behållaren runt bilden blir för smal skalas bilden ner för att rymmas innanför dess marginaler och när den skalas upp och dess behållare blir bredare än bildens pixel storlek så skalas den inte mera upp.

Även om koden i figur 9 löser problemen med bilder i en flytande layout så löser den inte prestandaproblemen bilder bidrar till på enheter med låg bandbredd i responsiv webbdesign. För att försöka lösa dessa behöver man använda sig av lite mera sofistikerade lösningar som exempelvis adaptive images eller RESS (se kapitel 6). Det som är lovande är att en ny standard är på kommande med namnet <picture> och denna ska förhoppningsvis lösa problemen som det oflexibla <img> elementet medför. [20]

En annan teknik som kan nämnas här är möjligheten att leverera bilder via CSS deklarationer som i figur 10. Denna metod gör det möjligt att byta ut bilder genom media förfrågningar då filkällan kan editeras direkt i CSS koden.

```
background-image: url(images/header_full.jpg);
```

Fig.10: CSS selektor för att ställa in ett elements bakgrund

Detta är dock inte alltid praktiskt och ibland finns det helt enkelt inga alternativ till <img> element, exempelvis i ett webbpubliceringssystem där en användare kan ladda upp bilder själva och bilderna levereras direkt genom <img> element i HTML koden utan att användaren kan påverka detta.

4.2 Inbäddad video och externa tjänster

Koden från figur 9 i föregående avsnitt kan också anpassas till andra CSS selektorer som kan ses i figur 11. [2]

```
img, embed, object, video { max-width:100%; }
```

Fig.11: CSS selektor som hanterar flera olika media typer

Att få inbäddade strömmande video-element att följa en flytande layout har ett problem som beror på att leverantörerna av dessa tjänster (t.ex. youtube) ofta skickar med en fast höjd och bredd inställning genom <iframe> element. Detta görs helt enkelt för att hålla bildformatet på videon intakt (4:3, 16:19 osv.). Om man försöker manipulera dessa värden direkt i <iframe> elementet blir resultatet vanligtvis en svart ruta.

Även här finns det en relativt smidig lösning som Chris Coyier förevisar i sin artikel ”Fluid Width Video” [6] och kan ses i figur 12 och 13. Genom att omlinda <iframe> elementet med ett annat element och manipulera den genom detta nya element så är det möjligt att göra iframen flytande och få videon att behålla sitt bildformat.

```
.videoWrapper {
  position: relative;
  padding-bottom: 56.25%; /* 16:9 */
  padding-top: 25px;
  height: 0;
}
```

Fig.12: *Wrapper elementet för en iframe [6]*

```
.videoWrapper iframe {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
}
```

Fig.13: *Iframens kod [6]*

Detta lyckas genom att <iframe> elementet positioneras absolut innaför det omslutande elementet och ”spänns över” detta genom att höjd och bredd sätts till 100 %, vilket gör att det flyter med i detta omlindande elements ändringar i storlek.

Denna metod kan också användas för andra tjänster som använder sig av <iframe> som exempelvis Google och Microsoft i sina karttjänster och även för andra element som exempelvis <object> eller <embed>. [6]

5. Media förfrågan

Media förfrågan är den tredje och kanske viktigaste delen i responsiv webbdesign och är den teknik som gör det möjligt att anpassa CSS kod efter specifika storlekar på viewportar. Detta gör den genom att fungera som en conditional statement inne i ett stylesheet och består av två delar, en media type kontroll och en (eller flera) media feature kontroller. [13] [20]

5.1 Media type

En media type används för att känna av vilken typ av enhet det är frågan om, vanligtvis är det screen eller print som används men aural, braille, handheld, projection, tty, tv är de andra alternativen som finns [w3.org, 9]. Stödet för media typer varierar bland webbläsare, exempelvis Mozilla Firefox stöder enbart screen och print media typer [10].

Ett enkelt exempel på hur en dylik media type förfrågan kommer till användning på egen hand är för att anpassa en sida för utskrift genom print media typen som i figur 14. Vanligt är också att använda helt skilda print-stylesheets som laddas in via en media type kontroll i <head>.

```
@media print {utskriftsanpassad CSS}
```

Fig.14: *Exempel på användning av en media type i CSS*

5.2 Media feature

Med en media feature menas exempelvis bredd, höjd eller bildformat på en enhets viewport. Features kan användas tillsammans med min- eller max- prefix som då motsvarar större än eller mindre än uttryck [12]. Media feature förfrågningar kan kombineras genom de logiska operationerna and, not och only till att täcka många olika features i en och samma media förfrågan.

En viktig detalj att notera är att media förfrågningar måste komma efter den kod som den är ämnad att modifiera i ett stylesheet och på samma sätt måste även media förfrågningarna själva komma i rätt ordning, beroende på vilka brytningspunkter dessa är ämnade för.

Med andra ord så måste en query för max-width: 1024px komma före en som är ämnad för max-width: 800px. Om dessa vore omsvängda och viewporten vore exempelvis 750px bred så skulle i detta fall media förfrågan för max-width: 1024px vara den som aktiveras fastän båda är sanna för en 750 pixel bred viewport. Den som är sist och sann vinner så ordningen är viktig i CSS.

Exemplen i figur 15 och 16 [19] förevisar hur en media förfrågan är upplagd som en conditional statement och hur man kombinerar flera olika media features inne i en förfrågan.

```
@media screen
and (min-device-width: 768px)
and (max-device-width:
1024px)
and (orientation : portrait)
{
}
```

Fig.15: *iPad anpassad query, landskapsläge*

```
@media screen
and (min-device-width: 768px)
and (max-device-width: 1024px)
and (orientation : landscape) {
}
```

Fig.16: *iPad anpassad query, porträttläge*

Här innebär min-device-width att om enheten är lika med eller större än 768px så är denna feature förfrågan sann och max-device-width säger att om enheten är mindre än eller lika med 1024px så är denna feature också sann. Med andra ord enheten ska vara minst 768px bred eller max 1024px bred och i båda fallen kontrolleras vilket läge enheten är i (porträtt eller landskap).

W3 listar följande media features på sin sida om CSS3 media förfrågningar, utöver de som redan nämndes [9]: aspect-ratio, device-aspect-ratio, color, color-index, monochrome, resolution, scan grid. En feature som kan vara till nytta är resolution som används för att kontrollera en enhets pixeltäthet.

5.1 Media förfrågningar och flytande layouts

För att se hur media förfrågningar och en flytande layout arbetar tillsammans är ett bra ställe att titta på stresspunkterna i en flytande layout. Det är vid dessa punkter det finns ett behov av att modifiera element för att de blivit för små eller för stora.

I praktiken kan detta exempelvis innebära att man i stresspunkten går från en fyra kolumns layout till två kolumner, eller arrangerar om layouten så förälder elementet kan ta mera utrymme på sidan och därmed också elementen under den. I figur 17 och 18 visas hur en fyra kolumns layout byter till två kolumner (se figur 1 för exempel) genom en media förfrågan som modifierar en kolumn till att bli ~48,5% bred istället för det tidigare värdet på ~23,7%.

```
.box {
  width: 23.721881390593047034764826175869%;
  margin: 0 0 0 1.022494887525562372188139059305%;
  background: #333;
  height: 110px;
  float:left;
}
```

Fig.17: *Baskoden i CSS, fyra kolumner*

Att endast width värdet modifieras i figur 18 betyder inte att de andra variablerna försvinner utan de ärvs från originalkoden som ses i figur 17, dvs. marginalerna, bakgrunden, höjden och float värdena är de samma i figur 17 och 18.

```
@media screen and (max-width: 1024px) {
  .box {
    width: 48.46626%;
  }
}
```

Fig.18: *Koden modifierad genom en media förfrågan till att bli 2 kolumner*

Figur 19 och 20 visar ett exempel på hur man går över från en fast till en flytande layout med en brytningspunkt vid 1024px. Huvudmålet här att ändra bredden på ett behållarelement från att vara 1024px brett till att bli flytande genom att

bredden sätts till 100 % genom media förfrågan i fig. 18 . Även padding värdet sätts att vara 0 i detta fall.

```
.main_content_wrapper {
   margin: 0 auto;
   width: 1024px;
   background: #fff;
   border: 1px solid #333;
   padding: 10px;
}
```

Fig.19: *Baskoden i CSS*

```
@media screen and (max-
width:1024px){
   .main_content_wrapper {
      width: 100%;
      padding: 0;
    }
}
```

Fig.20: *Koden modifierad genom en media förfrågan att bli flytande när viewporten är 1024px eller smalare*

Vad som kan vara en bra brytpunkt för en fast layout är upp till varje designer själv att avgöra, exempelvis genom att se på statistik över vilka de mest använda skärmupplösningarna är på nätet i nuvarande läge. Enligt statistik från statcounter (fig. 21) verkar den vanligaste resolutionen vara 1366x768, på andra plats 1024x768 och på tredje plats 1280x800.

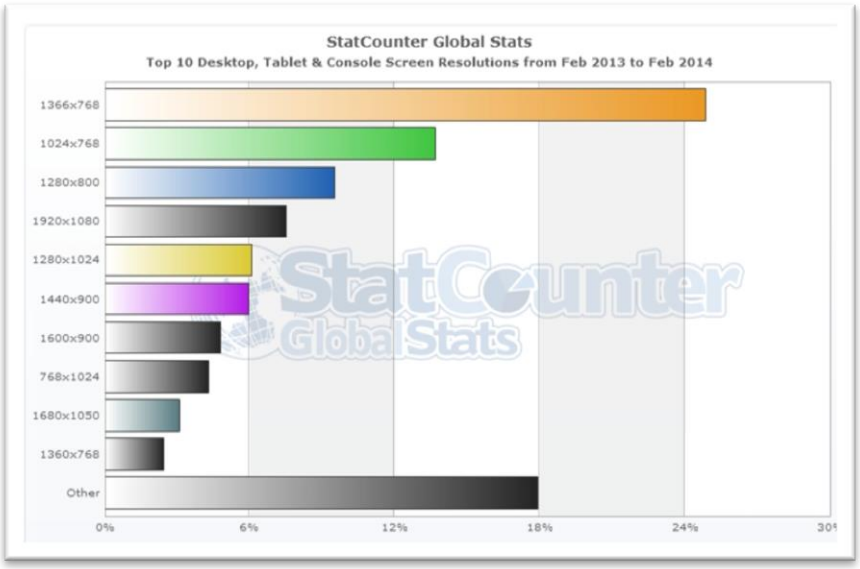

Fig.21: *Mest använda skärmupplösningarna från februari 2013 till februari 2014.*

Skärmdump från statcounter.com

6. Variationer på responsiv webbdesign

De diskussioner som förekommer på nätet om responsiv webbdesign handlar ofta om hur problemet med bilder och onödig överföring av data på mobila

enheter ska lösas. Dessa diskussioner involverar ofta så kallade hybrida metoder som genom att använda servern till att känna av vilken typ av enhet och viewport en användare har och baserat på denna information leverera optimerad data och inställningar för denna enhet och dess viewport storlek.

6.1 RESS - Responsive design + server side components

RESS är en så kallad hybrid metod som börjat få genomslag inom industrin och kan bli den teknik som är nästa stora inom webbdesign. Som titeln säger så är det en kombination av responsiv design och användning av server för att leverera optimerade komponenter till en sida baserad på enheten som detekteras. Dessa komponenter kunde t.ex. vara bilder vilket skulle ge en bra lösning på problemen med dessa i responsiv webbdesign. [15]

I exemplet i figur 22 ser man hur strukturen på en sida som använder sig av RESS kan vara uppbyggd. I detta fall är det header och footer som är det komponenter som kan bytas ut via servern. Beroende på typ an enhet som servern känner av så levereras motsvarande header och footer komponent. Sidans vanliga innehåll använder sig av responsiv webbdesign för att anpassa sig.

```
<body>
{{>header}}

[...document content...]

{{>footer}}
</body>
```

Fig.22: *RESS struktur [15]*

RESS skiljer sig alltså från responsiv webbdesign på det viset att den kan leverera olika innehåll genom komponentstruktur baserat på vilken typ av enhet som besöker sidan, medan responsiv webbdesign endast arbetar med att ändra på presentationen av ett och samma innehåll hela tiden.

6.2 Adaptive images

Adaptive images mål är att leverera optimerade och korrekt skalade bilder till den enhet som besöker sidan, baserat på vad enheten har för skärmstorlek och de media förfråge inställningar sidan använder sig av. [17]

Detta gör adaptive images genom att med hjälp av en simpel rad javascript skapa en cookie (figur 23) som innehåller information om enhetens viewport (höjd,bredd) och skicka den till servern samtidigt som en <img> request skickas dit.

```
<script>document.cookie='resolution='+Math.max(screen.width,s
creen.height)+'; path=/';</script>
```

Fig.23: *Adaptive images Cookie kod [17]*

Tekniken är baserad på php och .htaccess (web server konfiguration som apache använder sig av) och fungerar även på nginx genom att editera virtuella-host inställningar för att hantera jpg, gif och png requests. Dessa requests går sedan genom adaptive-images.php och som skalar ner och optimerar bilderna vid behov förrän de levereras. [17]

```
/html/blog/wp-content$ ls -l /usr/share/nginx/html/ai-cache/1920/cg/v1/images/

114492 Mar 21 17:57 1.jpg
 77007 Mar 21 17:57 2.jpg
161623 Mar 21 17:57 3.jpg
137357 Mar 21 17:57 4.jpg
 98799 Mar 21 17:57 5.jpg
161466 Mar 21 17:57 7.jpg
140626 Mar 21 17:57 8.jpg
/html/blog/wp-content$ ls -l /usr/share/nginx/html/ai-cache/800/cg/v1/images/

34040 Mar 21 16:56 1.jpg
20534 Mar 21 16:56 2.jpg
35275 Mar 21 16:56 3.jpg
33402 Mar 21 22:06 4.jpg
23279 Mar 21 16:56 5.jpg
23511 Mar 21 22:06 6.jpg
44289 Mar 21 16:56 7.jpg
40688 Mar 21 22:06 8.jpg
```

Fig.24: *Resultatet av adaptive images på en testsida*

Som man ser i figur 24 är det en förminskning av filstorlek på ungefär 70 % som åstadkommits i detta fall, vilket gör en hel del skillnad för en enhet med låg bandbredd. Detta test var inte så omfattande men borde ge liten idé om vad adaptive images kan göra. Nackdelar finns, exempelvis att man måste ha tillgång till servern och för att implementera detta vilket inte alltid är möjligt. [17]

Slutsatser [ska skrivas om]

Flytande layout, flexibel media och media förfrågningar utgör grunden för responsiv webbdesign och genom en smart kombinering av dessa tre tekniker kan man designa webbsidor som anpassar sig automatiskt till en besökande enhets viewport.

Fördelarna med responsiv webbdesign överväger stort nackdelarna om man kan lösa prestandaproblemet som bilder medför, genom exempelvis någon bättre för ändamålet fungerande standard än <img>. En sådan är som tur är på kommande genom <picture> elementet men som det verkar fungera med standarder i webbvärlden kommer det att ta en bra tid förrän det finns stöd för denna i de flesta webbläsare. Tills det finns bättre standarder är det att förlita sig på tekniker som RESS och adaptive images för att försöka lösa detta problem.

Källor [ska fixas/rensas/uppdateras]

[1] E. Marcotte, daterat 25 maj 2010. “Responsive Web Design.” http://www.alistapart.com/articles/responsive-web-design/

[2] J. Steimle, Forbes, daterat 8 november 2013. Why Your Business Needs A Responsive Website Before 2014 http://www.forbes.com/sites/joshsteimle/2013/11/08/why-your-business-needs-a-responsive-website-before-2014/

[3] E. Marcotte, daterat 7 juni 2011. “Fluid Images. “ http://alistapart.com/article/fluid-images

[4] B. Frain, *Responsive web design with HTML5 and CSS3*

[5] E. Marcotte, daterat 3 mars 2009. “Fluid Grids.” http://alistapart.com/article/fluidgrids

[6] C. Coyier, “Fluid Width Video” http://css-tricks.com/NetMag/FluidWidthVideo/Article-FluidWidthVideo.php

[7] P-P. Koch, ”A pixel is not a pixel is not a pixel” http://www.quirksmode.org/blog/archives/2010/04/a_pixel_is_not.html

[8] P-P. Koch, ”Meta Viewport” http://www.quirksmode.org/mobile/metaviewport/

[9] P-P. Koch, ” A tale of two viewports - part one” http://www.quirksmode.org/mobile/viewports.html

[10] P-P. Koch, ” A tale of two viewports - part two” http://www.quirksmode.org/mobile/viewports2.html

[11] ”Using the viewport meta tag to control layout on mobile browsers.” https://developer.mozilla.org/en/docs/Mozilla/Mobile/Viewport_meta_tag

[12] “Configuring the Viewport” https://developer.apple.com/library/ios/DOCUMENTATION/AppleApplications/Reference/SafariWebContent/UsingtheViewport/UsingtheViewport.html

[13] ”Media Queries.” http://www.w3.org/TR/css3-mediaqueries/

[14] J-A. Wilkins, daterat 16 Juli 2012 “Responsive Design’s Dirty Little Secret.” http://www.palantir.net/blog/responsive-design-s-dirty-little-secret

[15] L. Wroblewski, daterat 12 september 2011. “RESS: Responsive Design + Server Side Components.” http://www.lukew.com/ff/entry.asp?1392

[16] StatCounter GlobalStats. http://gs.statcounter.com/#resolution-ww-monthly-201302-201402-bar

[17] ”Adaptive Images” http://adaptive-images.com/details.htm

[18] ”Media Queries for Standard Devices” http://css-tricks.com/snippets/css/media-queries-for-standard-devices/

[19] ” CSS media queries” https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries

[20] ”The picture element - An HTML extension for adaptive images” http://www.w3.org/TR/html-picture-element/