Anteckningar

Agila utvecklingsmetoder och HTML.

# 2020-10-05 Måndag, Agila utvecklingsmetoder och HTML

# Agila utvecklingsmetoder

Se det ”Agila manifestet” ca 2001.

Mer ett ”mindset”.

Undvika ägarfokus.

Ser ej arbetare som kugghjul, utan lägger vikt på deras välmående.

Jobba efter förtroende, låta varje mindre team ta ansvar över arbetssätt och metodik själva.

Försöka jobba i kortare perioder och därefter ta kontakt med kund samt granska resultatet för att se om man kan gå vidare. Undvika stora ”inlämningar” (leverans till kund) då de kan ledda till stora förluster. Inte jobba efter långa avtal, värdera kundkontakt över avtal.

Att arbeta agilt innebär också att man är redo och villig att acceptera ändringar även nära arbetet/projektets tänkta slut.

Man måste även vara redo att använda olika strategier beroende på förutsättningar (pandemier till exempel).

Man kan summera att arbeta agilt är att vara lite flexibel och inte lägga så stor vikt på förutbestämda principer. Anpassa dig efter vad som krävs.

Det finns några olika etablerade agila arbetsmetoder. Två av dessa är SCRUM och Kanban.

## SCRUM

Se ”The SCRUM guide”.

SCRUM innebär att vi delar upp arbetet i kortare perioder så kallade ”sprint”.

Dessa perioder följer en viss ordning:

Först gör man en planering. Här bestämmer vi vad som ska kompleteras denna ”sprint”. Vi bestämmer detta efter företagets/gruppens/produktens ”backlog”. Det som vi bestämmer ska kompleteras formar periodens/”sprintens” backlog (Sprint backlog).

Vi jobbar sedan på våra uppgifter enigt agila principer (öppenhet, flexibilitet, kommunikation etc).

När ”sprinten” är över så gör vi en **granskning.** Här utvärderar vi arbetet. Dels funktionaliteten i vår produkt, men också hur arbetsmetoden har fungerat (god kommunikation, flexibilitet etc).

Varje dag inom vår ”sprint” räknas som en egen ”sprint”. Den ska likaså börjas med en planering om vad som ska göras över dagen och avslutas med en utvärdering.

## Kanban

Kommer från Japan. Kanban är japanska och betyder skylt. Metoden skapades i tillverkningsindustrin. Den värdera produktion över allt annat. ”inget” får stoppa att produkten går från tillverkning till kunden inom tidsramen.

Den har 3 grund principer:

### Fokusera på flödet.

”The Toyota way”

Gör arbetet enkelt. Bara det vi behöver.

Resurseffektivitet.

### Visualisera arbetet

Använd visualisering för att se över arbetsflödet.

### Begränsa hur mycket du arbetar på samtidigt

Som punkt 1, gör arbetet enkelt.

Gå ej vidare med nästa arbete till dess att du är klar med det du gör.

## SCRUM/Kanban

Scrum passar bättre när du har förutbestämda arbetstider. ”Du vet vad du behöver ha gjort om två veckor.” Hjälper dig att dela upp arbetet.

Kanban passar bättre för tex support när du inte riktigt vet vad som kommer hända under dagen. Arbetsuppgifter trillar in kontinuerligt.

# HTML

HTML är ett filformat som webbläsare använder för att skapa sidor. Websidor på internet består utav HTML-kod.

HTML byggs upp med ”taggar”. Dessa används för att specificera vad ett stycke ska vara. Se nedan.

<!doctype html>

Dett är en tagg som förklarar till webbläsaren att det är en HTML-fil. Websidan fungerar ej utan denna tagg och den ska skrivas längst upp.

<html>

</html>

Allt innehåll ska skrivas inom en sådan tagg. Notera sluttaggen (</>), de flesta taggar måste avslutas på ett sådant viss.

<body>

Denna ska skrivas inom </html> taggen. Här skriver man allt som ska vissas på hemsidan.

<p>

Paragraf-tagg. Skriver en paragraf

<h1>

Skriver en titel. Om du använder en mindre siffra så minskar storleken på texten.

<!—kommentar-->

Så här skriver du en kommentar. Det som skrivs inom denna tagg har ingen effekt på websidan och läses ej utav webbläsaren. De används för ditt eget syfte. Till exempel för att skriva påminnelser om en bit kods syfte. Väldigt användbart när det börjar bli mycket kod och för att hjälpa andra som kan komma att använda din kod att förstå. Använd flitigt.

Se video, förklarar bättre och mer än mig.

<https://www.youtube.com/watch?v=bWPMSSsVdPk>

# Dag 2 Mer HTML

**<!doctype html>**

Vår docktype.

**<html lang=”en”>**

Vår html tag. Notera ”lang”-attributet. Den förklarar för webbläsaren vilket språk som hemsidan är tänkt att skrivas på. Webbläsaren kan erbjuda översättning baserat på vad som står där.

**<head>**

Detta är en head-tagg. Det som skrivs här visas ej på hemsidan. Här kan du ange olika attribut som sidans titel, icon och länka separata filer/bibliotek som kan styra sidans utseende och funktion.

**<title>Här skriver du sidans titel.</tilte>**

Denna tagg används i ”head”-området för att ange sidans titel (vad som visas på fliken).

**<link rel=”icon” href=”ikon.img”>**

Så här anger jag vilken ikon jag vill använda. Rel=”icon” är vad som förklarar att länken ska användas som ikon.

**<link href=”style.css” rel=”stylesheet”>**

Med denna “link” tag så kan du länka olika externa filer. I detta fal har jag länkat en CSS-fil. Vi har inte gått igenom detta än men det är en fil som används för att styla och strukturera en websida. Man kan även länka externa bibliotek tex:

**<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">**

Här har jag länkat en extern hemsidas bibliotek med ikoner. Detta gör att jag inte behöver spara alla ikoner på min dator/server utan kan använda en länk till deras server för att ladda upp bilder på min hemsida. Sparar minne på min dator men gör så att sidan laddar något långsammare då webbläsaren behöver hämta ikonen över internet.

Href=”http://www.exempel.se” så här specifierar du en sökväg för en bild, länk, icon etc..

Rel=”stylesheet” denna specificerar förhållandet mellan ditt dokument och dokumentet du jobbar på.

Tex ”styleesheet” förklarar att det dokumentet du länkar ska användas för att styla det dokumentet du jobbar på. ”icon” förklarar att länken ska användas som ikon.

<link>-taggar är självstängande (ingen </link> behövs)

**</head>**

”head”-taggar behöver en sluttagg

**<body>**

### Listor

Det går att göra två olika sorters listor i html.

Taggarna för dessa är:

**<ul>** unordered list, en punkterad lista.

**<li>sak 1</li>** varje element du vill ha i din lista måst läggas i en ”li”-tagg (list item)

**<li>sak2</li>**

**</ul>**

Skulle se ut så här:

* Sak 1
* Sak 2

**<ol>** ”ordered list”, en numrerad lista.

**<li>sak 1</li>**

**<li>sak 2</li>**

**</ol>**

1. Sak 1
2. Sak2

### Tabeller

Du kan också göra tabeller.

**<table>** så här gör man en tabell.

**<caption>Tabell titel/namn<caption>** Så här skriver du ett namn/titel över tabellen.

**<thead>** ”table head”,så här gör du den översta raden i en tabell, tex namn/kön/ålder etc…

**<tr>** ”table row”, så här gör du en ny rad i tabellen.

**<th>Namn</th>** ”table head”, text i fet still för den översta raden.

**<th>Ålder</th>**

**</tr>**

**</thead>**

**<tbody>** ”tablebody”, här är alla övriga rader med data

**<tr>** måste göra en ny rad

**<td>Gustav</td>** ”table data”, vanlig text för tabellens data.

**<td>Göran</td>**

**</tr>**

**<tr>** ytterligare en rad

**<td>24</td>**

**<td>50</td>**

**</tr>**

**</tbody>**

**</table>**

Tabellen ovan skulle se ut något så här:



Viktigt att använda <thead> och <tbody> annars ser det väldigt konstigt ut.

**</body>**

**</html>**

# Dag 3, Grundläggande CSS och bildelement

## Bildelement

Kodas i html med taggen:

**<img src=”bild.png” alt=”en bild på en blomma” title=”en bild”>**

**Img**

image (bild). Unik tagg för bilder.

**Src=””,**

här anger du vilken länks som bilden har. Om du har en fil på din dator så anger du bara dess poition och namn ex:

bild.png

exemplet ovan funkar bara om du har bilden i samma mapp.

När du gör en hemsida så kan det dock bli väldigt mycket bilder och andra filer och man kan vilja sortera in allt i olika mappar.

När du lägger in en bild i en annan mapp måste du ändra sökvägen.

Du gör då så här:

/images/bild.jpg

Du kan också länka bilder som finns på internet genom att länka dess url.

**Alt=””**

Här kan du lägga in ett alternativt om bilden inte kan nås, text internet är nere men du har hemsidan lokalt.

Då visas vad du skrivit. Bra praxis är att göra en beskrivning utav bilden. Du kan också ange en annan sökväg.

**Title=””**

Här kan du ange en text som kommer upp som en ”tooltip”, en liten text vid muspekaren som kommer upp när du håller pekaren över bilden.

Img taggen är självstängande i de nyare varianterna utav html. Du behöver alltså inte skriva </img>.

## CSS

Css är ett ange olika stilar.

Man kan även ange stilar direkt i html genom att lägga till attributet **style=””.**

**Ex:**

**<p style=”border:solid; color:red;”>**

Här har jag givit en paragraf en gräns och angett färgen på texten som röd. Stilar som du angett direkt i html taggen har alltid högre prioritet över externa filer. Så om du har en specifik paragraf som du vill ha på just ett sätt så kan du göra så här.

Det är dock inte rekommenderat då det kan bli väldigt rörigt. Det vore bättre att ge den paragrafen ett id eller en klass och sedan mål sätta den specifikt i CSS.

För att använda css så skappar man först en CSS-fil. Den css-filen som styr utseendet på en hemsida kallas vanligtvis för ”style.css”.

Jag gick igenom detta tidigare men för att css-filen ska kunna styla en html-fil så använder man länken.

**<link href=”style.css” rel=”stylesheet”>**

**<link>** taggen som du använder för att länka externa filer.

**Href=””** här anger du sökvägen till den filen som ska användas.

**Rel=””** här förklarar du vad den filen du länkar har för syfte/funktion.

Inne i css filen så skriver du först vilket element du vill styla. För att styla vanliga taggar som finns i html så skriver du bara dem mellan två {}. Sedan skriver du vilka attribut du vill tillge det/dem elementen.

Ex

**H1{**

**Color: red;**

**Background-color:blue;**

**Text- align:center;**

**}**

Ovan har vi gjort så att all text skriven mellan två h1-taggar ska vara skriven i röd färg med blå bakgrund och vara centrerad.

Det finns hur många attribut som helst som man kan ange och det bästa sättet att bli bekant med den är att googla och testa. Säg att du vill ändra storleken på texten. Googla då: ”text size css” så kommer du garanterat att hitta hur du gör.

Jag nämnde tidigare att man kan vilja ändra ett specifikt element. Text en specifik paragraf.

Jag sa då att det är bättre att ge elementet ett eget id eller class och sedan mål sätta det i css.

För att ge ett element ett id eller en klass så gör du det i html taggen på det elementet.

Så här:

**<p id=”paragraf”></p>** Här gav jag just denna paragraf-tagg ett id med namnet ”paragraf”

**<p class=”pragraf”></p>**

**<p class=”paragraf””></p>** Här gav jag två element klassen ”paragraf”.

Skillnaden mellan id och klass är att ett id är unikt för ett element medans klass kan anges till flera olika element. Så om du har flera element som du vill ska se likadana ut så använd klass.

För att mål sätta klasser i css-filen så skriver du en punkt följt utav klassnamnet

Ex:

**.paragraf{**

**Color:white;**

**Font-size:50px;**

**Font-family: arial;**

**}**

Här gjorde jag så att all text i alla element med klassen ”paragraf” får vitt text, i text storleken 50px och i typsnitet arial.

För att mål sätta och styla id:n så skriver du en # följt utav id-namnet.

Ex

**#paragraf{**

**Color:red;**

**Background-color:green;**

**Font-size:100px;**

**Padding:20px;**

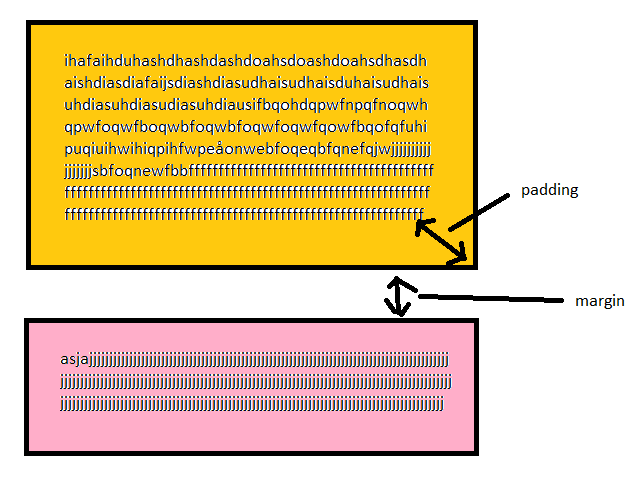
**Margin:20%;**

**}**

Här har jag ändrat på all text inom elementet med id:t ”pragraf”. Jag har även givit den lite ”padding” och ”margin”.

Padding är ett extra tomt område mellan elementets innehåll och elementets kant.

Margin är ett tomt område mellan elementet och nästa element. Man kan ange deras mängd i pixlar (px) eller procent (%)



Man kan även kommentera i CSS med /\*kommentar\*/

När man anger olika storlekar på margin/padding så kan man specifiera cilken sida som man vill ändra.

Antagligen så skriver du:

Margin-top:100px;

Mragin-left:0px;

Margin-right:20%;

Margin-bottom:2%;

Man kan också skriva allt i en rad text:

Margin:100px 20% 0px 2%; (topp, höger, vänster, bot)

Margin:100px 20%; (topp och bot, höger och vänster)

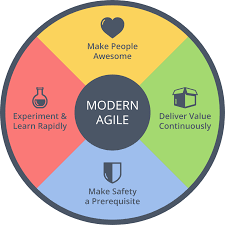
## Webbläsar-utvecklingsverktyg.

Webbläsare har i regel sina egna verktyg där du kan se och redigera i real tid.

# Dag 4, Moderna Agile och semantik.

## Modern Agile

Ett nyare utskott utav agila arbetsprinciper. Formades när agile började sprida sig och bli populärt.



Ovan är de fyra delarna som modern agile består utav.

### Få alla att må bra.

Alla som är med i arbetsprocessen ska må bra. Från arbetarna som jobbar på en produkt till kunden som tar emot den.

Både arbetsprocessen och produkten måste byggas på ett intuitivt sätt. Det ska inte vara något krångel.

Arbetet måste gå någorlunda snabbt och man ska kunna få omedelbar feedback. Man fokuserar på att göra små leveranser.

### Skapa trygga förutsättningar

Det måste vara ok att göra fel ibland och det måste finnas ett test nät som ser till att så många fel som möjligt inte kommer ut till kund. Kan återkoppla till första punkten. För att alla ska må bra så måste det finnas skydd mot olika händelser. Både för personal och kund.

### Exprimentera och lära sig snabbt

Ibland så måste man prova sig fram och försöka med nya arbetsmetoder. Se till att återkoppla och undersök hur varje ändring i arbetsprocessen eller produkten har spelat ut.

### Lita på din personal

Detta är inte en del utav ovan men det knyter in i det. För att din personal ska kunna uppnå allt ovan så gäller det att kunna lita på att om man ger dem ett ansvar så löser de det. Man måste ge personalen lite frihet och metoder som att begränsa en arbetares tillgång till internet är inte särskilt agilt.

## Semantiska element

Semantik=hur man presenterar något. Hur man delar upp en hemsida.

Exempel på semantiska element = h1, h3, p, article etc. Taggen är semantisk för den förklarar vad den har för syfte. Värt att notera är att de har ingen specifik funktion i sig själva utan det är tänkt att du ska använda dem för att ordna upp din hemsida (i semantiskt syfte alltså). Det finns dock bra och dålig praxis även om valet hur du vill gör är upp till dig.

Icke-semantiska element = div, span. De har inga förbestämda syften utan du får använda dem i eget syfte.

div.

Div:ar är väldigt bra att använda. Man kan likna dem med lådor att stoppa olika element i. Säg att du vill ha en bild eller en grupp med bilder som ska se ut på ett vist sätt och vara placerade ihop. Du kan då stoppa dina bilder i en div och sedan ge div:en ett id eller en klass och sedan använda css för få gruppen att placeras och se ut som du vill.

Span

Span är ett inline element som man kan placera inom andra element (ofta text) liknande <b> och <strong>. I sig själva så gör de inget utan du måste tilldela attribut på dem i css

Article: article är en förutbestämd tagg som ska innehålla huvudinnehåll. Tex ett bloggflöde.

Section: en under rubrik för article.

Header: sidhuvud där du kan placera titel etc

Nav: ett navigationsfällt, placeras vanligtvis i ”header”

Aside: ett sidofällt

Footer: En fotnot.

Det finns ganska många mer att använda. Kolla ww.w3schools.com.

 Hur semantiska element används för att strukturera upp en sida.

### Gästföreläsare, Hur är det att vara scrummaster i ett högfungerande team.

Scrum? Kan vi vara agila utan?

* Make people awsome
* Make safety a prerequisite
* Experiment & learn rapidly
* Deliver value continuously

Scrum, kanban etc är bra verktyg för att uppnå ett agilt arbete.

#### Ny som scrummaster

Var ödmjuk, andra kan mer än du

Lyssna in teamet, identifiera behov

Inse att teamet vill programmera i första hand. Se till att det ni gör är relaterat till produkten.

#### Vi lever i en introvert värd

Inse att de flesta personer inom IT/programmering gillar att göra sitt och inte gillar sociala påhopp.

Du måste bygga trygghet, dela på information, respektera alla för vilka de är och ha roligt tillsammans.

#### Agila i aristocrats, egna erfarenhet

Förändring är förväntat, planera för det

Inkludera alla i teamet, ingen kör eget race

Focus på kunden

Par programmera kan vara nyttigt.

Kommunicera, kommunicera, KOMMUNICERA!!!!!!!!

#### Rutiner

Retroperspektiv – viktigt att få känn på känslan i teamet.

Daily – särskilt viktigt för distansarbete

Sprintplanering-planera och inkludera alla.

Sprint review-leverera till kunden.

#### Annat som kan bidra till agilt arbete.

Fika-gör något som inte har med jobbet att göra.

par programmering-att jobba ihop gör så att man inte behöver granska koden senare.

#### Vad gör teamet inte

ingen whiteboard. Blir onödigt om alla är på samma sida

#### Lärdomar

Ha roligt (jobba för att skapa en situation där du kan ta det lugnt)

Var flexibel, acceptera andras behov

Utgå från att alla tar sitt ansvar. Undvik onödig Kontroll

Se till att teamet får göra vad de brinner för.

Låt inte seniorerna göra all planering, juniorerna blir inte lärda på rätt sätt då.

Kommunicera med kunden

Satsa alltid på att leverera en fungerande produkt.

# Dag 5, Fördjupad CSS

## Selektorer.

Selektorer kan användas för att ange och styla ett specifikt element i css. Säg att du vill ha en länk i navigationen som ska ha lite större text.

Om du gör så här:’

A{

Font-size:large;

}

Då blir alla länkar i hela dockumenet påverkade.

Gör istället så här:

Nav a{

Font-size:large;

}

På så viss så anger du att det är bara alla länkar i navigationen som ska påverkas.

Man kan även använda bägge. Säg att du vill att alla länkar ska ha stor text men de i navigationen ska vara röda så gör du så här:

A{

Font-size:large;

}

Nav a{

Color:red;

}

De attributen som är målsatta närmst taggen är de som applicerar först. Alltså om du specificerar ett visst element eller rent utav skriver det direkt i html så kommer de ha prioritet över alla övergripande attribut som kommer användas efter om de inte strider mot något utav de närmre.

### Klasser och ID

Klasser och ID har tidigare nämnts. De har samma funktion men ska användas i olika syften.

De är selektorer som man kan ange till vilket element som helst genoms att skriva in dem i taggen.

**<p class=”paragraf”></p>**

**<h1 id=”titel”></h1>**

Vad de används till är att kunna ge olika element ett ”namn” man kan ange för att sedan referera till i CSS så att just det elementet blir till delade de atribut som man anger.

Tex

Vi har en artikel som är ett utav visa element på en sida som är extra viktiga. Allt som är viktigt ska ha röd text.

**<article class=”important”></article>**

Vi ger denna artikel klassen ”important”

**<p class=”important”></P>**

**<a href=”google.se” class=”important”>**

Vi kan nu använda CSS för att ange de attribut som vi vill ha.

**.important{** (OBS!! När du vill målsätta en klass som du själv har skapat så skriver du en punkt följt utav klassnamn).

**Color: red;**

**}**

Värt att veta är att ett element kan ha flera olika klasser.

Nu skulle alla element som vi har givit klassen ”important” ha text i röd färg.

Vi angav en titel (h1) ett ID tidigare. Just denna titel vill vi ha grön men vi vill inte att några andra element påverkas.

Det är varför vi använt just ID och inte klass och det är det som är skillnaden mellan dessa. Ett ID ska användas på ett unikt element som ska stylas unikt, medan klasser ska användas på flera olika element som ska ha samma attribut.

Vi gör så här:

**#titel{** (OBS!! När du vill styla ett ID som du själv har skapat så gör du ett # följt utav ID:namn).

**Color: green;**

**}**

**Em**

Em är ett attribut i css som anger att något ska vara x gånger större än den vanliga texten. Säg att text du markerat i ett <span> eller med <strong> också ska vara 2 gånger större än vad de vanligtvis är. Då skriver du (i CSS):

Strong{

Font-size: 2em;

}

Väldigt användbart att använda när en sida ska kunna publiceras på olika storlekar utav skärmar. Då det inte har några absolut belopp utan de skalar sig efter vad standarden är per skärm.

Man kan även göra så att vissa attribut bara gäller när man håller muspekaren över dem. Detta med regeln ”:hover”.

Tex

**.important :hover{**

**Color: pink;**

**}**

Så här så skulle alla element med klassen ”important” få en rosa text-färg när man håller muspekaren över dem.

Om man vill ha ett par element i en klass ska ha olika attribut så kan man också använda sig utav **.imoprtant :nth-child(0, 2, 3) {**

**Background-color:black;**

**}**

Här anger man x ”barn utav klassen”. Alltså x element som ingår i klassen. Här har jag angett att element 1, 3 och 4 (i programmering så börjar man räkna från 0) i klassen ”important” ska ha en svart bakgrundsfärg.

## Positionering

Hur man får olika element placerade på den plats som man vill ha dem. Detta görs i CSS med olika attribut.

### Display

Med display kan du ange hur delar i ett, eller elementet själv element ska visas. Tex hur listelementen (<li>) i en ordnadlista (<ol>) ska visas.

**Display: inline;**

Elementet blir som ett inlines element (som <span>). Storleks attribut får ingen effekt.

**Display: block;**

Elementet visas som ett block. Det börjar på en ny rad och tar upp hela radens bredd.

### Relativ positionering

Med attributet **position:;** kan man ange hur ett element ska vara positionerat.

Med **position: relativ;** så kan man få element att bli positionerade relativt till något. Här så börjar olika element sluta att själv organisera sig efter varandra och det är lite mer i dina händer att se till att olika element inte lägger sig över varandra.

### Clear fix

Clear fix används i för att se till att float element inte råkar glida ned bland element under det flytande elementet. Det ser till att nästa element börjar på en ny rad.

### z-index

bestämmer vad som ska ligga överst. Om du har flera element som kan hamna över varandra och du vet att ett element ska vara synligt och över de andra så kan du ge den ett högre z-värde.

# Dag 6

## Lägga till videor

Använd <video>-tagg

**<video src=”youtube.com” controls autoplay>Gick ej att spela<video>**

**Controls** = skapar spelknapar (play, pause etc).

Texten mellan taggarna kommer fram om videon inte går att spela.

**Autoplay =** videon spelar automatiskt.

Ett enkelt sätt att linka en video från Youtube är att klicka ”dela” på videon och använda ”bädda in”. Då får man en HTML kod att klistra in.

### Iframe

Med <iframe> så kan du skapa en sida i en sida. På så vis så behöver du inte all HTML kod.

Använd:

**<iframe src=”aftonbladet.se”></iframe>**

## Klient/server

Klienten är du när du skriver in google.se. Vad som då händer är att din dator letar upp vilken IP-adress som motsvarar den DNS (DNS=Domän Namn Systemet) du sökte efter. Sedan så skickar din dator (klient) en efterfrågan till den server som håller hemsidan du vill åt. Servern skickar då den data som du har frågat efter i en HTML-fil.

http:

TCP: Transmission Control Protocol. Ett dataöverförings protokoll. Används för att säkra och konfirmera en dataström. Om en del utav den data som skickas över är felkorrigerad så kan det leda till stora fördröjningar.

UDP: User Datagram Protocol. En förbindelselös protocokoll som är mer lämpligt när du ska skicka stora mängder data. Tex streaming eller dataspel.

IP:

## WCAG (tillgänglighet) och validering

### WCAG

Beskriv allt innehåll som inte är text med en text.

Erbjud alternativ om en inspelning utav ljud eller video för synnedsatta.

Etc. Det finns en lång lista med förutsättningar man kan försöka uppnå för olika nivåer.

Validering

Det finns också ”validatorer” (vet ej hur man säger validation i plural)

HTML: https//:validator.w3.org

CSS: <https://jigsaw.w3.org/css-validator/>

Tillgänglighet: https://achecker.ca/checker/index.php

Använd valitatorerna och rätta till alla fel som de pekar ut.

Lighthouse. Ett tillägg som betygsätter din hemsida enligt prestanda tillgänglighet och liknande.

## Gästföreläsare Johan från veoneer

Levererar till bilindustrin.

Gör säkerhetslösningar. Bilbälten och krockkuddar tidigare.

Nu aktiva säkerhetsmedel, kameror radar etc. Auto styrning.

Enbart focus på aktiv säkerhetsteknologi.

### Traditionell projektmodell, vattenfall

**1: Krav**

**2: Design**

**3: kodning**

**4: Test**

**5: Utrullning**

**6: Underhåll**

* **Utveckling i faser**
* **Förväntade resultat**
* **Design spikad från start**

På Veoneer har de vattenfall i botten. Projektet går igenom faser.

Första fasen är en ”tull” där de kollar om de har allt de behöver för att börja projektet.

# Dag 7 Responsiv design

Att göra en hemsida som fungerar på olika storlekar utav skärmar genom att ändra olika elements storlek eller ändra deras funktion och/eller utseende.

## Viewport

En grundläggande tagg som vi måste använda iHTML för att kunna göra en responsiv design. Läggs i **<head>**.

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

Använd kommandot följsamt designläge i webutvecklings verktygen för att se hur din sida ser ut på olika skärmar.



## Brytpunkter med media-queries

Queries = fråga.

För att undvika att din sida blir svår att använda så kan man använda sig utav ”media queries”. Vad de gör är att de applicerar visa egenskaper och attribut bara när de tillstånd du anger uppfylls. I exemplet nedan så vill vi göra vissa saker när skärmen är mindre än 640 pixlar.

Det blir lite som att skapa en CSS i din CSS. För att ange vilka attribut som ska gälla i en ”media querie” så gör du precis som vanligt fast inuti måsvingarna. Se nedan.

När skärmen är mindre än 640 pixlar så vill vi att ett element med klassen ”logo” ska vara 200 pixlar brett och ett element med klassen ”title” ska ha sin text centrerad.

I CSS

**@media only screen and (min-width: 640px){**

**.logo{**

**Width: 200px;**

**}**

**.title{**

**Text-align: center;**

**}**

**}**

## Dölja element

Säg att du vill ta bort ett element när skärmen är mindre än 500 pixlar. Lika som tidigare så använder vi ”media queries”.

**@media only screen and (min-width :500px){**

**.logo{**

**Display: none;**

**}**

Ovan har vi tagit bort ett element med klassen ”logo” när skärmen är mindre än 500px.

## Rem

En storleksenhet man kan ge element. Storleken blir relativ mot grundelementet. Om du har tidigare givit ett element så blir rem storleken inte relativ mot den storleken du satt (så funkar ”em”) utan det blir relativt till vad grundstorleken är (standarden i html).

## Mobile first.

Designa din sida för mobiltelefon i första hand. Det är lättare att lägga till element allt eftersom skärmen blir större istället för att behöva ta bort då den blir mindre.

## Flexbox

Flex är ett attribut du kan ge en container som gör så att elementen i containern sorterar upp sig automatisk på det område som finns tillgängligt. Om skärmen blir mindre så kommer elementen automatiskt att försöka bibehålla sin position enligt samma proportion som tidigare.

Om man ger dem attributet **flex-wrapp: wrap;** så kan elementen också flexa sin position genom att skapa fler rader om området blir för litet för att få plats med alla element på en rad.

Med **flex-direction: ;** så kan man bestämma om man vill att elementen ska lägga sig på en rad (: **row;**) eller i en kolumn (**: column;**).

Man kan även ange om man vill att ett utav elementen i flex-containern ska ta upp mer plats än de andra proportionerligt mot containern. Detta med **flex-grow:;**

Om alla element har **flex-grow: 1;** så tar de alla upp lika mycket plats i containern men om du ger ett utav elementen **flex-grow: 2;** så kommer det att ta 2 gånger så mycket plats som de andra i containern.

Man kan också göra så att ett element kan krympa om det blir ont om plats med **flex-shrink: ;**

Det funkar likadant som flex-grow i med att du anger ett värde bara att i detta fal så krymper det elementet med högst flex-shrink värde mest.

Med flex så lägger sig vanligtvis elementen så nära varandra som möjligt. Om du istället säg vill sprida ut dem så kan man använda **justify-content: space-evenly;**

Med det kommandot så skulle elementen sprida ut sig gämmt med ett lika stort avstånd mellan varje element och containerväggarna.

Man kan även centrera och göra massor annat med **justify-content: ;** använd google som hjälp eller:

https://css-tricks.com/snippets/css/a-guide-to-flexbox/