HTML, CSS, Editor, Kommentarer:

**HTML:**

HTML står for ”Hyper Text Markup Language”, og bruges til at opbygge skelettet af en hjemmeside. Dvs. at dets primære formal er at få tekst, billeder, og afsnit ind på siden. Man kan derfor ikke have en hjemmeside uden at mindst have *noget* HTML.

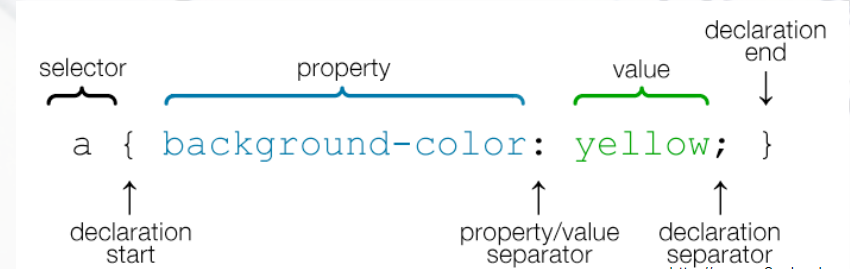
For at kunne gøre brug af HTML til opbygge en hjemmeside, så skal man bruge en ”editor”. En editor er et tekstredigeringsprogram hvori man skriver sin kode (eks. HTML, CSS, JavaScript, C#, C++ og mange flere). Hvilken editor man skal vælge, er helt til op en selv. I kodeverdenen er det en smagsmag, alt efter om man er til teksttype, farve, tema, layout, osv. Det kommer simpelthen an på, hvad man er mest til, og hvad der passer en bedst. Personligt så bruger jeg Visual Studio Code, fordi den har de funktioner og det layout jeg er mest vant til.

**CSS:**

CSS står for ”Cascading Style Sheets”, og er det computersprog der bruges til at style alt HTML. Hvis HTML er skelettet på hjemmesiden, så er CSS’en huden, øjnene, ørene, munden, hovedet, armene, benene osv. Herudover så bestemmer den også farverne, formerne og placeringen af disse ”kropsdele” på hjemmesiden. CSS er stylisten der har smag indenfor alt mode, makeup, tøjstil, og farver. Uden CSS så ville hjemmesiden se meget kedelig og død ud. Det ville bare være sort tekst på en hvid baggrund, uden noget struktur, og det ville køre i en lang række nedad, og slet ikke være ordentligt sat op. CSS er absolut nødvendigt, hvis ikke for farver, former, og design, så i det mindste for at kunne få det til at fungere på forskellige skærmstørrelser, eller sagt på en anden måde, ”mobile first.

”Mobile first” er den tilgang man tager til en hjemmeside med henblik på at opbygge den til mobilen først. Dette vil ikke kun gøre det nemmere at arbejde med tablet og desktopstørrelsen senere hen, men også eliminerer problemer som kunne have opstået, skulle man have startet med desktopversionen først. Mobile first er en tilgang med henblik på at gøre sin hjemmeside 100% mobil venlig først, før alt det andet.

Syntaks for CSS (regel for hvordan det skal skrives op, og i hvilken rækkefølge) ser sådan her ud:



**Editor og kommentarer:**

Editoren skriver i ren tekst og uden formatering (medmindre man har specielle extensions i en given editor, der gør således at man får formatering og andre funktionaliteter på sin tekst). Editoren hjælper også med at highlighte kode og er derfor en stor hjælp hvis man skulle skrive forkert. Den hjælper med at understrege koden således at man nemt kan fine hvilke vendinger og kodesætninger der har mangler, eller er forkerte.

Herunder også linjenummer, hvilke konflikter det har med andre kode, og med en rød markering på ”scrolleren” (det lille vindue der viser alt kode, som man kan scrolle op og ned i.) Derudover så er editoren også meget god til at ”write ahead”, hvilket vil sige at den vil give forslag på at afslutte en kodesætning, alt efter hvad man skriver. Dette er også med til at gøre kodeskrivning nemmere, ikke kun for professionelle, men også begyndere.

Nogle editors har også en funktion der tillader programmøren at se et ”preview” på hvordan siden vil se ud, med den kode man har indtil videre. Dette kan være meget dejligt at have integreret i sin editor, da man hurtigt kan se hvordan hjemmesiden ser ud, om der er mangler, og måske style ting på en anden måde en man lige tænkte over.

Når man skal kode, er det en god idé at træne sin kodeskik. Dvs. at skrive koden i små afsnit ad gangen, lave indrykninger med tabulatoren (”TAB”), linjeskift når det er nødvendigt, og ikke mindst **kommentarer!** Det er *vildt* vigtigt at skrive kommentarer til sin kode, da det ikke kun vil hjælpe en selv i fremtiden, skulle man rette eller tilføje noget til koden, men det vil også hjælpe med at gøre koden mere læsbar for andre, skulle de være nødt til at arbejde på den, eller hjælpe dig selv med den. Det gør det også meget nemmere at finde rundt i koden, og giver koden en meget bedre struktur. Så derfor husk altid at kommentér, kommentér, og atter kommentér!

Tags, Attributter, Elementer, Semantik, SEO:

**Elementer:**

For at kunne bygge en hjemmeside op på en ordentlig måde gennem HTML, så skal man gøre brug af ”tags”. Tags er med til at definere hvilket stykke kode er hvad, dvs. om det er f.eks. en paragraf, titel, undertitel, artikel, billeder mm. Vi refererer til et stykke kode som har et start, og et slut tag, som et ”element”. Man kan derfor også sige at en hjemmeside består af rigtig mange elementer

Et eksempel på et element kunne se således ud:

**<h1> Sikkerhed <h1/>**

**Tags:**

Tagget der er blevet brugt i dette eksempel, er et overskriftstag der hedder ”h1”. Tagget starter ved ”<h1>”, og har et slut tag ved <h1/>. Tags følger denne regel, lige meget hvilket slags tag der er tale om. I ovenstående eksempel betyder det at alt hvad der står imellem dette starttag, og sluttag, vil stå som en overskrift på hjemmesiden. I dette tilfælde vil vi få en overskrift hvor der står ”Sikkerhed”. Der findes flere overskriftstags og angives med ”<h1>, <h2>, <h3>, <h4>, <h5>, <h6>.

**Semantik og SEO:**

Grunden til at de har forskellige numre er fordi det er med til at indikere størrelsen af overskriften på hjemmesiden. Derudover så er det også med til at skabe semantik på hjemmesiden. At noget er semantisk, betyder egentlig bare at noget har mening/betydning. Hvis et tag er semantisk, så fortæller det noget om dets indhold.

Eksempel:

<div>: fortæller ingenting om sit indhold

<table>: fortæller noget om sit indhold idet den markerer alt inden i til at være en ”tabel”, hvilket er også hvad navnet på tagget selv indikerer.

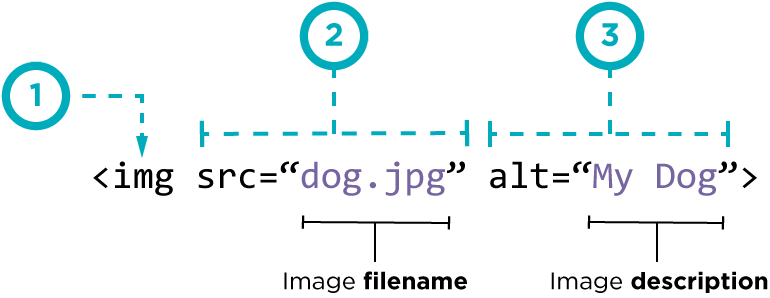
Semantik, eller semantiske tags, er med til at give hjemmesiden struktur, og gør det nemmere for søgemaskiner, at finde hjemmesiden, også kendt som ”SEO”. SEO står for ”Search Engine Optimization”, og betyder i bund og grund at det er med til at gøre det nemmere, og mere relevant for søgemaskiner at finde lige præcis DIN hjemmeside først, over andre hjemmesider. Dette er selvfølgelig relevant for at give din hjemmeside mest mulige besøgere, og kliks.

Følgende semantiske tags er gode at bruge, for at have sin hjemmeside SEO optimeret:

* **<section>:** Inddeler koden i sektioner, og giver mere overblik over de forskellige afsnit i HTML’en
* **<header>:** Bruges til at indikere ”hovedet” på hjemmesiden, og er det der ligger øverst på siden.
* **<footer>:** En footer bruges til at indikere det nederste på hjemmesiden eller ”fødderne” på den. Det står altid i bunden af en hjemmeside.
* **<form>:** Forms bliver brugt til at modtage bruger input, i form af små tekstbokse, hvori brugeren kan bl.a. skrive sit navn/adresse/alder, eller noget helt andet.
* **<table>:** Ligger lidt i navnet idet at det er en tabel som bliver inddelt med et yderligere tag der hedder <tr>, som står for ”table row”.
* **<article>:** Er med til at specificere selvstændigt indhold, og burde give mening på egen hånd uden noget andet yderligere.

Her er også nogle eksempler på andre tags som også bliver brugt, men som ikke er semantiske:

* **<p>:** Er en paragraf, og er stort set bare en linje tekst, der ignorerer linjeskift og mellemrum i koden.
* **<div>:** Bruges som regel til at indikere en sektion i koden som ikke nødvendigvis er nødt til at være semantisk, opfører sig mere eller mindre som <p> tagget.
* **<span>:** Ligner meget <div> tagget, men forskellen ligger i at den bruges til at lænke eksempelvis billede og tekst sammen, således at de står på samme linje. Derudover så er span tagget et inline element, mens div er et block element.
* **<img>:** Bruges til at indsætte et billede ind på hjemmesiden. Filen udpeges med en ”src” attribut, og man skriver stien til billedet mellem de to anførelsestegn.



[*https://cdo-curriculum.s3.amazonaws.com/media/uploads/img\_tag.png*](https://cdo-curriculum.s3.amazonaws.com/media/uploads/img_tag.png)

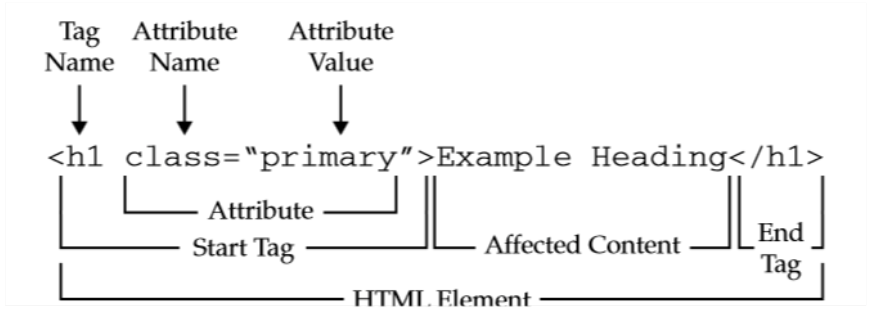
* **<li>** (herunder også <ol> og <ul>): Bruges til at lave lister på hjemmesiden. <ol> bliver brugt til at lave en ”ordered list” hvilket vil sige en liste med nummering på, den kunne f.eks. se sådan her ud:

1. **Item**
2. **Item**
3. **Item**
4. **Item**

En liste uden nummering bruger <ul> i stedet, som står for ”unordered list”, og kunne se sådan her ud:

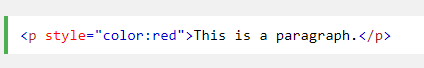
* **Item**
* **Item**
* **Item**
* **Item**

**Attributter:**



Attributter giver HTML elementer noget at relatere til. Dvs. at man nu kan nemmere, mere specifikt pege på hvilket element det er man snakker om. Man kan give et element en attribut som kunne eksempelvis være et ID, eller en klasse. En attribut består af to dele, et attributnavn, og en attributværdi. Navnet indikerer hvilken slags attribut der er tale om, mens værdien af attributten bruges til at pege på selve elementet med den specificerede værdi.

Eksempel:



*Her er attributnavnet ”****style****” og attributværdien ”****color:red****”.*

**Links:**

Links i HTML koden fungere på den måde at man skriver hvad der skal være klikbart imellem to <a> tags, det kunne f.eks. se sådan her ud:



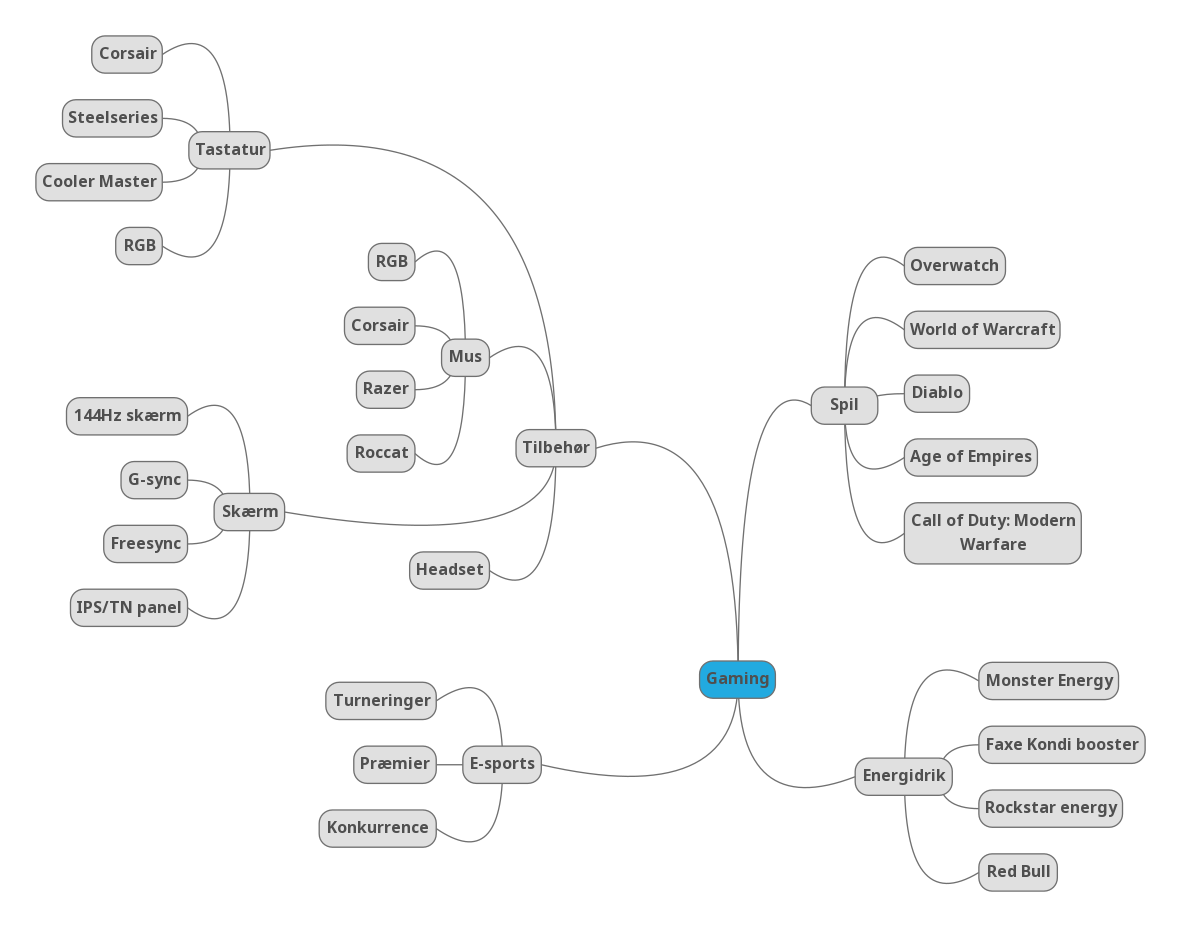
Her er “<a>” tagget starten og ”</a>” enden af elementet. Den røde tekst, er den hjemmeside som man bliver taget til når man klikker på linket, mens den grønne tekst er linkets navn (det man faktisk ser på hjemmesiden som blå highlighted tekst og som man kan klikke på).

**Mindmap:**

En mindmap minder meget om brainstorming. Det er en notationsform som holder fast på sammenhængen mellem de forskellige ord i mindmappet. Når man laver en mindmap så brainstormer man for at kunne finde på nye ord der relaterer sig til emnet, eller allerede nedskrevne ord. En mindmap kan med fordel også bruges med grafik/billeder og behøvest ikke kun at være tekst. Man kan bruge end mindmap til mange forskellige ting såsom:

* **Repitition**
* **Overblik**
* **Brainstorming**
* **Sammenhæng**
* **Struktur**
* **Associationer**

Et eksempel på en mindmap kunne se sådan her ud:



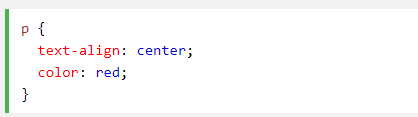
*Lavet af Ahmad Azzam.*

Selectors (Simple, Combinator, Pseudo):

**Simple selectors:**

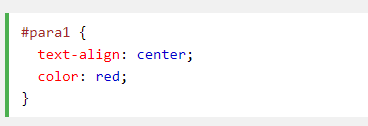
”**Selector**” er den første del i et stykke CSS-kode, og indikerer *hvor* i HTML’en der skal styles på. Herefter kommer ”**property**”, den indikerer *hvad* CSS’en skal style. Til sidst kommer ”**value”**, som egentlig bare er en værdi for property, og fortæller den *hvor meget* eller *i hvilken grad* HTML’en skal styles. Selve selectoren kan være flere forskellige ting. Den kan enten være et tag direkte fra HTML’en, eller et ID (ID’et vælger alle HTML elementer med samme ID, som i CSS’en, og angives med et ”#” før navnet på ID’et), eller en klasse (klassen vælger alle HTML elementer med samme klasse som i CSS’en og angives med et ”.” før klassenavnet på klassen).

**Element selector eksempel:**



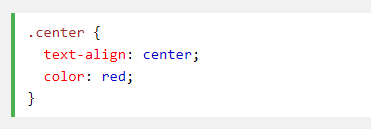
*Her er elementet ”p” valgt i CSS’en, og vil nu derfor style alt der har tagget <p> i HTML’en. Dette stykke CSS vil centrere teksten og give den en rød farve.*

**ID selector eksempel:**



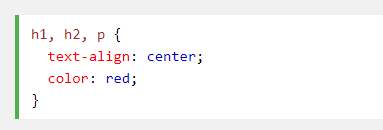
*I dette eksempel vælges alle HTML elementer med ID’et ”para1”. Her vil de blive stylet til a være centreret og have en rød farve.*

**Class selector eksempel:**



*Klasserne med navnet ”center” vil blive valgt i HTML’en, og vil blive centreret og få en rød tekst på hjemmesiden.*

**Grouping selector eksempel:**



*Udover ovenstående så kan man også gøre brug af grouping selectors. Her kan man vælger flere elementer samtidigt, og style dem alle sammen på samme måde, uden at skulle have til at skrive mange linjer kode i CSS.*

**Combinator selectors:**

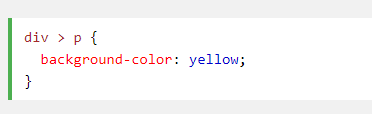
Herudover er der også ”combinator selectors” som vælger mere en bare én ting fra HTML’en, dette kunne være flere forskellige elementer, eller elementer der er ”child” af et andet element (hvilket vil sige at et element er et nedarvet element af et andet.) En ”combinator” er noget som beksriver forholdet mellem to selectors. Man kan derfor nemmere spotte i CSS’en hvilket element der er valgt, og i forhold til hvad.

**Decendant selector eksempel:**



*Vælger alle <p> tags, som ligger* ***inde i*** *<div> tags og giver dem en gul baggrund.*

**Child selector eksempel:**



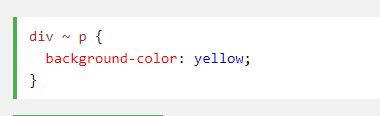
*Denne selector vælger alle <p> tags som er ”child” af <div> tags i HTML’en, og giver dem en gul baggrund. Dvs. at de <p> tags som bliver valgt,* ***skal*** *ligge inde i <div> tags i HTML’en.*

**Adjacent selector eksempel:**



*Denne type selector vælger det første <p> element i HTML’en som kommer* ***lige efter*** *<div> tagget. Alle andre <p> tags som kommer herefter blive ignoreret.*

**General sibling selector eksempel:**



*General sibling selectoren vælger i dette eksempel alle <p> tags som er søskende til <div> tags. Dvs. at her vil alle <p> tags som kommer* ***efter*** *<div> tags blive valgt.*

Kaskaderegler, Box model, Farver, Skrifttyper:

Når man snakker kaskaderegler, så snakker man om regler for hvordan CSS-koden skal opføre sig i visse situationer. Disse kaskaderegler skal definere hvordan to eller flere modstridende linjer i koden skal, opføre sig på hjemmesiden, og hvem af dem der faktisk får prioritet.

**Specificity:**

For at kunne forstå *hvorfor* et stykke CSS-kode, ikke gider at blive anvendt på hjemmesiden, så skal man forstå specificity. Specificity er det regelsæt i CSS-koden som angiver hvilken style bliver anvendt på hvad. Det er også med til at organisere koden, og give den struktur, således at den bliver mere læsbar. Sagt på en anden måde, så er der visse selectors der har forrang (højere ”vigtighed”) end andre selectors. Man inddeler som regel specificity i fire forskellige selector grupper (rangen på de forskellige selector er også i rækkefølge):

* **Inline Styles**
* **ID’er**
* **Klasser, atributter, pseudo-klasser**
* **Elements og pseudo-elements**

**Source order:**

Source order omhandler deklareringer i CSS-koden, og hvilken deklarering man har lavet senest. Hvis der nu skulle stå at man styler headeren ét sted i CSS’en, men samtidig længere nede i koden, stå et nyt stykke kode for samme header, så vil det seneste stykke kode tage prioritet. Det man har skrevet sidst, er også det CSS’en vil gå efter. Specificity har altid forrang over source order. Der er dog én undtagelse og det er brugen af ”!important”. Den går lige forbi disse slags regler, men er også samtidig med til at ødelægge ideen og funktionaliteten af kaskadereglerne, og dermed også ruinere pointen med kaskaderegler som en helhed.

**Box model:**

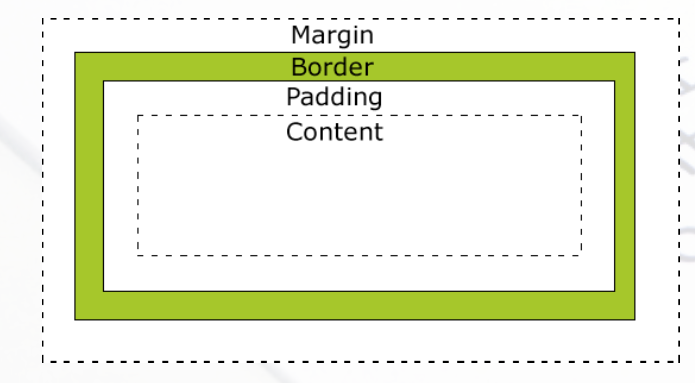
Når der snakkes om ”Box model” så snakkes der om hvordan forskellige HTML-elementer repræsenteres i forskellige slags kasser. Der findes to forskellige typer, ”inline” og ”block”. Inline er når boksene placerer sig horisontalt efter hinanden, mens block er når kasserne placeres ovenover hinanden, eller i en stak. Der findes også noget der hedder ”inline-block” hvilket er en kombination af de to fornævnte box models. Denne type boks, placere boksene horisontalt efter hinanden (inline), men beholder størrelsen af block boksen (block). For at kunne bruge dette i CSS’en, så kan man skrive eksempelvis:

**display: block/inline/inline-block**

**Box properties:**

I modsætning til ”Box model”, så er ”Box properties” de forskellige bokses egenskaber, idet de alle har følgende:

* **Content: (indhold)**
* **Padding: (afstanden mellem ramme og indholdet)**
* **Border: (kan slå synlighed på kant fra eller til)**
* **Margin: (afstanden mellem ramme og andre elementers margen)**



**Farver:**

Måden man styrer farver på i CSS’en er faktisk ret nemt og ligetil. Man kan enten vælge at skrive navnet på farven direkte (på engelsk), eller man kan vælge at gøre det med mere præcision med disse tre metoder:

* **RGB: rgb(255, 255, 255)**
* **HEX-kode: #000000**
* **HSL: (100%, 100%, 100%)**

**RGB:**

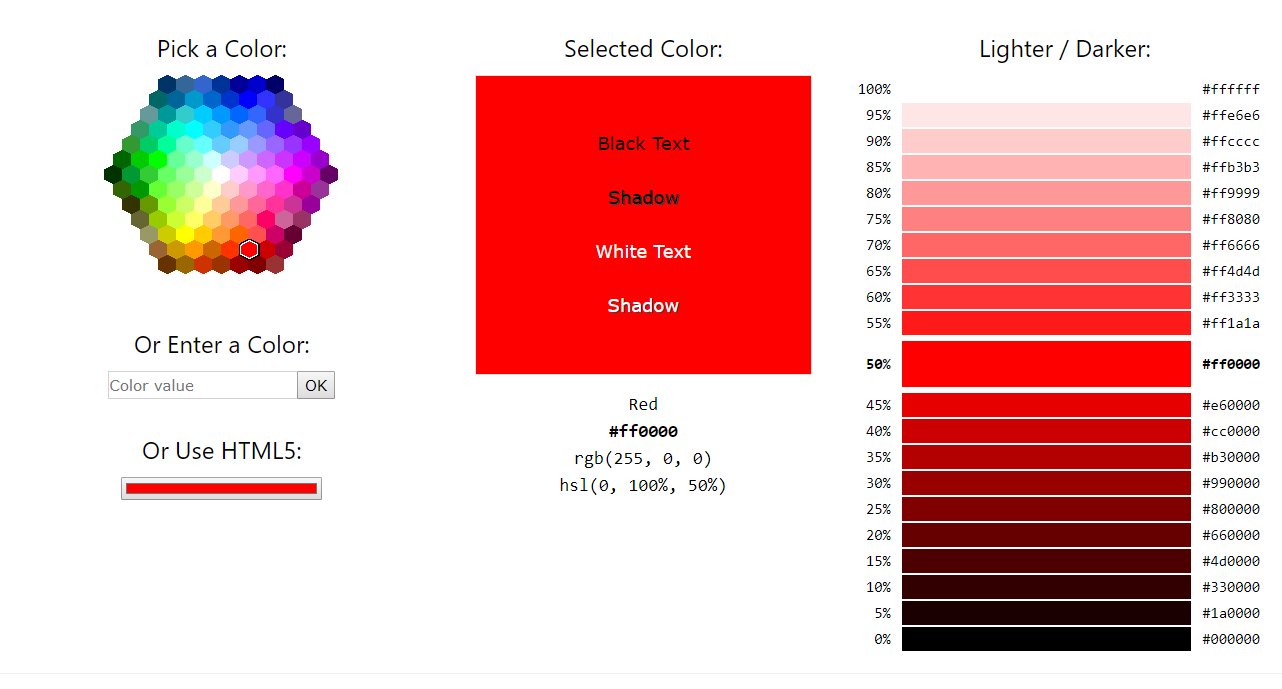
Den øverste måde at skrive farver på i CSS (RGB), fungere på samme måde som det lyder. RGB står får ”Red, Green, Blue”, og hver af de tre værdier kan gå fra 0-255. Jo højere et nummer, jo mere mæthed i farven. Den første værdi er rød, den anden grøn, og den tredje blå.



*Man kan bruge en RGB calculator på w3school’s hjemmeside*

**HEX-kode:**

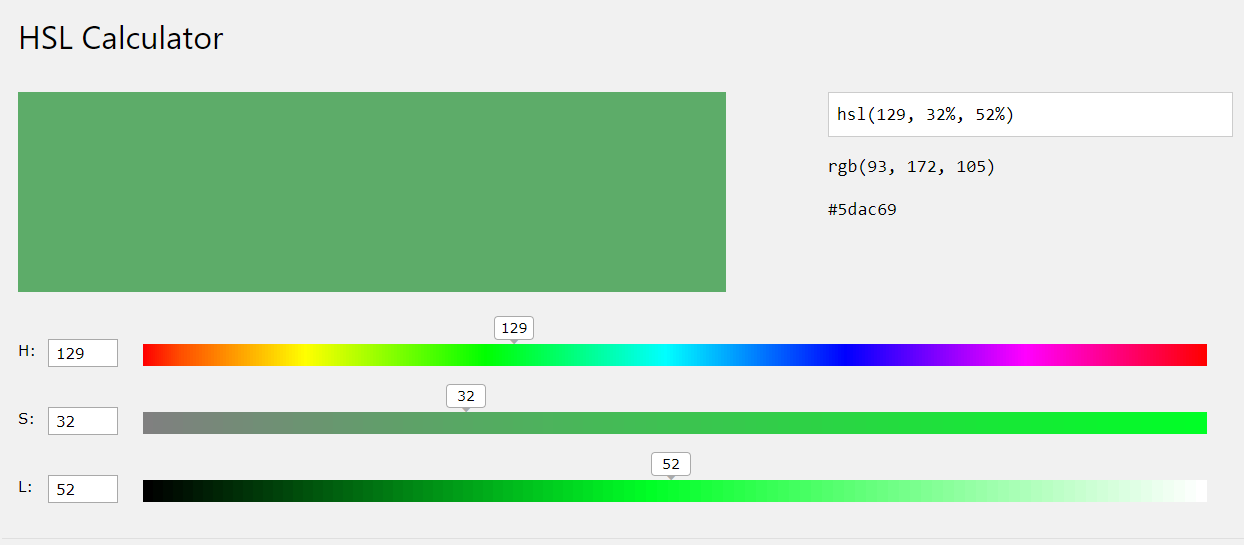
HEX-kode er en anden måde at skrive farver på. Her bestemmes værdien gennem et ”#” efterfulgt af en kombination af bogstaver og tal. Ingen kan huske alle værdierne i hovedet og hvilken farve de står får, men i eksemplet for oven, bliver farven **sort.**



*Der er en HEX-kode ”pick a color” som man kan lege med på w3schools hjemmeside.*

**HSL:**

I HSL arbejder man med procenter. Dvs. at man styrer hvor meget Hue (nuance), Saturation (mæthed), og Lightness (lyshed) Alle værdier går fra 0% - 100%.



*Der findes en ”HSL calculator” på w3school’s hjemmeside som man kan bruge*

**Skrifttyper:**

En god hjemmeside er en hjemmeside med relevante skrifttyper. Alle skrifttyper kan være gode, det er bare et spørgsmål om hvilke skrifttyper man skal bruge. Man bliver nød til at sætte sig ned og tænke over sit site, og hvad det egentlig handler om, og derudfra selv bestemme hvilken skrifttype man gerne vil have til overskriften, underoverskrifter, brødtekst, titler, undertitler osv. Kan man anvende den rigtige skrifttype til den rette hjemmeside, så vil man også få en glad læser, og det vil samtidigt være nemt og roligt for øjnene at læse siden.

Der vælges som regel højst, 2-3 skriftypper på en hjemmeside, dog er det også muligt at skabe en hjemmeside med blot 1 skrifttype. Det kan dog gå hen og blive lidt ensartet, og kedeligt at læse. Det er ikke alle computere der har installeret de samme slags fonte, så man skal derfor ud og lede efter nye fonte på nettet. Det kan muligvis være gennem ”Google Fonts”, eller ”Adobe Typekit”.

Google Fonts har en nem og gratis løsning til folk der gerne vil have et hav af valg, af skrifttyper. Skrifttypen selv bliver hostet af Google, og er meget let at implantere ind på sin hjemmeside. Adobe Typekit Har valgt at inddele de skrifttyper de har i ”packs” og ”family”, alt efter hvad man er ude efter. Herefter kan man vælge at tilføje dem til sin Adobe konto, hvor man derfra kan importere og implementere dem ind i sin kode.

**Versionsstyring:**

Versionsstyring eller ”version control” på engelsk, omhandler at holde styr på sine filer, og hvilken version af dem der er den seneste. Det kan være alt fra navngivning af mapper og filer, til at gemme noget i skyen gennem en service som Onedrive, Google Drive, eller lignende. Indenfor kodning bliver der typisk brugt hjemmesider som ”GitHub”, Gitlab” ”BitBucket”, og mange andre lignende. Disse hjemmesider har også typisk en software version, som er med til at gøre det endnu lettere, og hurtigere at opdaterer sine filer og mapper. Softwaren, er ligesom ”onlineversionen”, med til at holde styr på hvilken fil der er ændret, slettet, opdateret osv., og er altid i synkronisering med versionen i skyen.

Versionsstyring bliver brugt af alle i hele verden, til alverdens projekter, men bliver brugt især meget af it-folk. Det kan være alt lige fra:

* **Organisationer**
* **Hobby-kodere**
* **Programmører**
* **Firmaer/Iværksættere**

Og bliver brugt til udvikling af bl.a.:

* **IT-systemer**
* **Apps**
* **Hjemmesider**
* **Frameworks**
* **Hobby-udvikling**
* **Nye løsninger af problemer**

**Git:**

Git er en protokol, hvilket vil sige at det er en ”standard”. Git er ikke det samme som GitHub og skal ikke forveksles med det, men er i stedet et versionsstyringssystem. Man kan selv opsætte en Git-server, eller man kan vælge at gøre brug at tjenester som Github, Gitlab, Bitbucket, mm. Git blev oprettet og skrevet a Linus Torvald, og var oprindeligt lavet til Linux kernel, som er et open-source, unix lignende operativsystem.

**GitHub:**

GitHub er en webplatform, dog også med en pc-klient version der tillader at man kan oprette repositories (en form for bibliotek for filer og mapper). GitHub har en masse features, der hjælper med projektstyring, samarbejde, version kontrol, og meget andet. Det er et socialt medie idet at man kan dele kode med andre, og gøre brug af andres kode, helt frit og kvit. Det blev lavet med den intention at folk videregiver deres kode, eller hjælper hinanden med at arbejde på projekter, hvad end det er private eller offentlige opgaver. GitHub er blandt et af de mest populære kodedelingssite og ville være lidt mærkeligt hvis man ikke kendte til det som programmør. Det er også et site som man kan man bruge som en portfolio for den samling af kode, filer, og mapper man nu efterhånden får lavet, gennem sin karriere. Der er også mulighed for private repositories, til lukkede og egne projekter.

**Versionsstyringsbegreber:**

**Repository** (repo) – et bibliotek for alle filer, mapper. Bruges som en stor projektmappe.

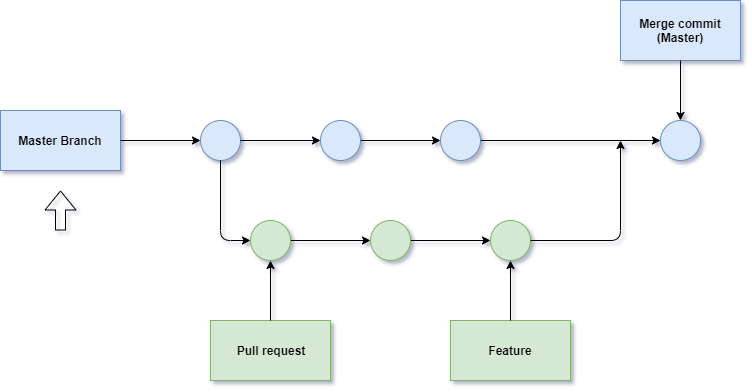
**Commit** (at comitte) – at gemme sine seneste ændringer i projektmappen (repository), med mulighed for at tjekke hvilke ændringer der er blevet lavet, og dermed også mulighed for at gå et skridt tilbage, skulle der være noget galt med den seneste version.

**Pull** – at nedhente de seneste ændringer fra repository, og merger dem med sine egne.

**Pull request** – en anmodning om at kunne pulle fra skyen.

**Branch** – en sidegren som arbejdes på, før man merger ind i *master branch* (hovedgrenen). Det er en kopi af den seneste version i skyen, og man berører ikke hovedgrenen imens man arbejder på denne forgrening

**Master branch** – hovedgrenen i versionstyringen, og er den endelige version af koden. Man skal kun merge, hvis man er sikker på at det ikke vil få konflikter, og man har fået testet alt hvad man vil teste af.



*Lavet af Ahmad Azzam.*

**Layout og Design:**

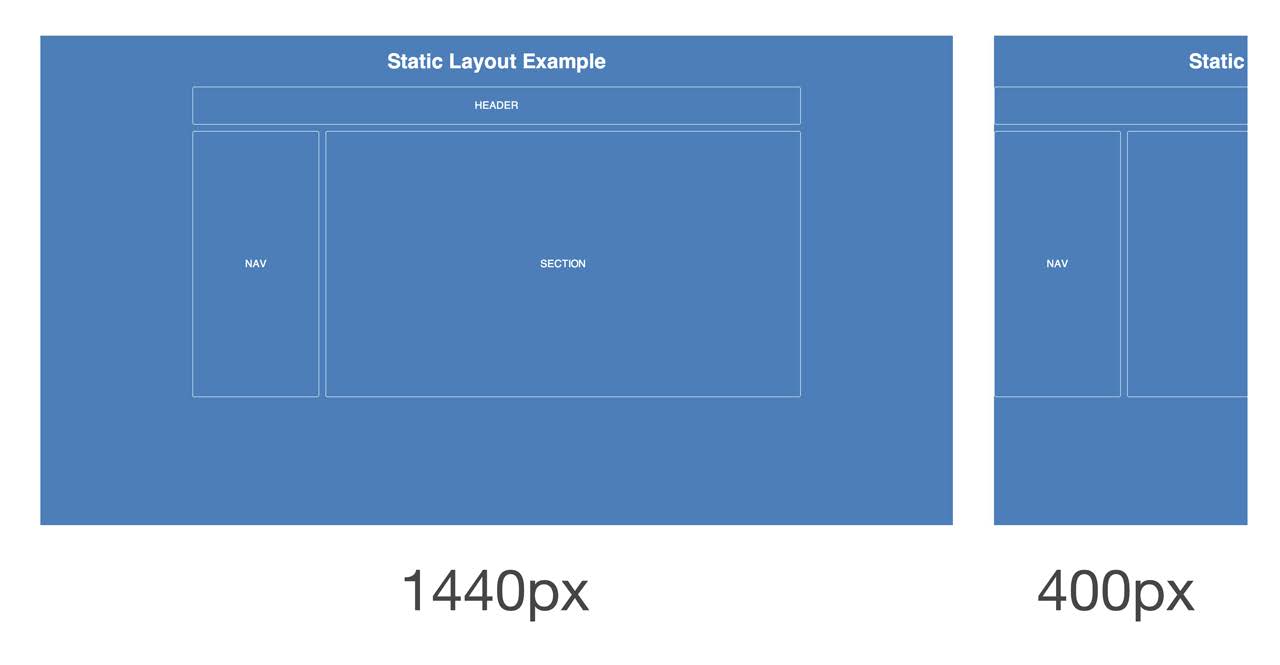
At kunne designe en hjemmeside er ikke bare noget man lige gør over en dag eller to, det handler om meget mere end det. Først og fremmest er der de forskellige skærmstørrelser på de forskellige enheder, og hver enhed har et sæt antal pixels i højden og i bredden. Som webdesigner så ønsker man at brugeren får så optimal en oplevelse som muligt uanset hvor henne i verden de er, og ligegyldigt hvilken enhed de er på. Samtidigt så ønsker man også som designer at skulle kode så lidt som muligt, men stadigvæk med det bedste resultat som kan opnås. Der skal kun være én version af hjemmesiden, men den skal samtidigt også kunne tilpasse sig de forskellige skærmstørrelser ”on the go”. Herudover så ønsker man også som designer at sit site har en ged SEO. For at kunne opnå denne tilfredsstillelse så skal man gøre brug af ”@mediaqueries”. @mediaqueries bruges som et breakpoint, således at når man når til en hvis skærmstørrelse (i bredden) så skifter siden layout, således at det bedre kan tilpasse sig hjemmesiden på den givne enhed.

**Layout**:

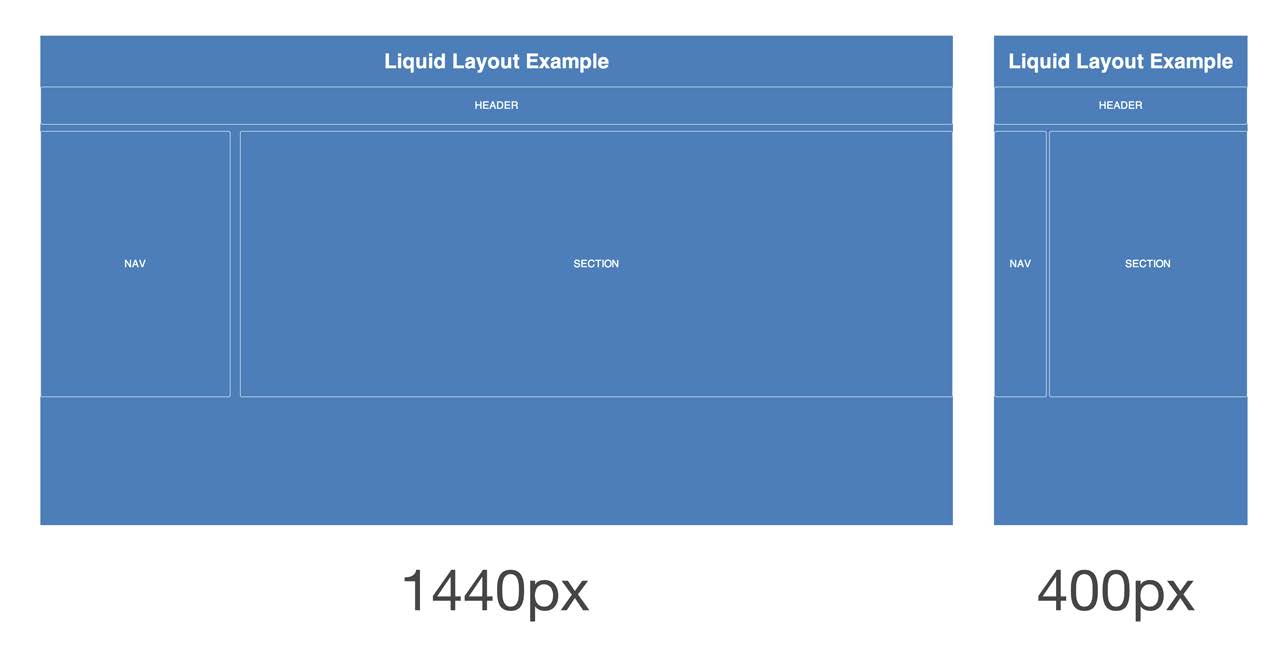
Layoutet defineres som regel i fire overordnede former:

* **Statisk (fixed width)**
* **Flydende (”liquid layout”, flydende og elastisk)**
* **Responsiv (tilpasser sig skærmstørrelsen ”on the go”)**
* **Adaptiv (blanding mellem statisk og responsiv)**

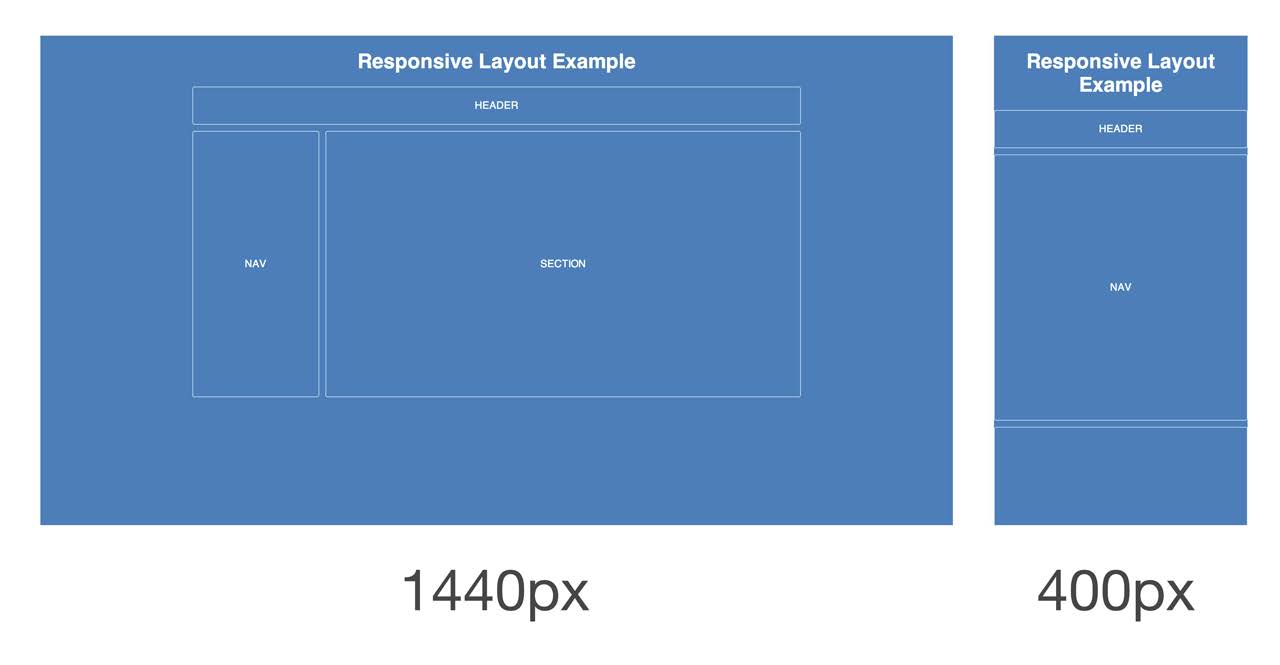
Et statisk webdesign omhandler brugen af en fast bredde på hjemmesiden. Dvs. at denne type design ikke vil ændre størrelse ved forskellige skærmstørrelser. Alt ekstra plads på skærmen, vil blive til tomt plads, med noget mellemrum. På mobile enheder skaleres dette ved at zoome ind. Det var engang det dominerende design, og designeren havde også samtidigt mere kontrol over resultatet. Dog fungerer dette meget dårligt på alle andre skærmstørrelser end den tiltænkte.



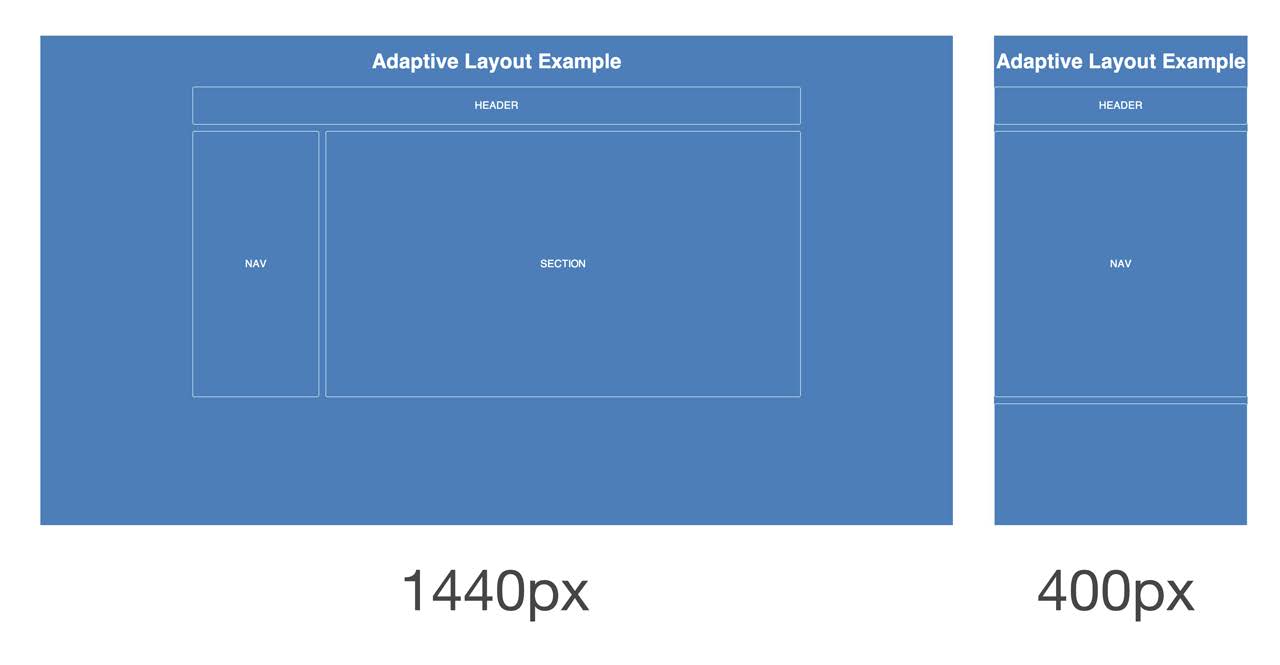
Flydende webdesign er meget mere tilpassende idet at det selv tilpasser sig til bredden, og man angiver samtidigt bredder hyppigt i procenter. Her kan elementer blive meget små og smalle og der er samtidigt grænser for hvor småt disse elementer kan gå. Et fluid webdesign bliver normalt valgt når indholdet og funktionaliteten på sitet, vægtes højere end den grafiske del. Det flydende webdesign udnytter pladsen godt, og giver også brugeren en bedre oplevelse.



Det responsive layout består meget typisk af et fluid layout, herunder samtidigt under en række regler som ændrer på skærmbredden afhængigt af hvilken enhed man som bruger er på. Her bliver der gjort brug af ”media quieries” (@media) i CSS, og det er altid samme HTML og CSS-filer uanset hvilken enhed der bliver brugt. Det er den foretrukne måde i dag at lave hjemmesider på, og er også blevet en standard. Det er ikke særligt svært eller vanskeligt, men kræver dog en masse planlægning og omtanke. Herunder skal man også gå ind med mindsettet ”**mobile first**” da det ellers vil få store konsekvenser senere hen skulle man have sprunget over, eller glemt dette trin. Det er altid nemmest at designe til noget der har begrænset plads **FØRST**, før man overhovedet går i gang med en skærmstørrelse hvor der kan være meget mere plads.



Det adaptive layout er noget der ligger imellem en statisk og en responsiv tilgang. Består selv af en række statiske layouts og styres af @mediaqueries. Dog ville der være for lidt eller for meget plads på skærme der ikke rammer lige præcis de breakpoints der er i koden. Samtidigt er det noget man vælger ret sjældent hvis der ellers er mulighed for at gøre det responsivt. Bruges dog ofte til at konvertere et statisk layout om til et responsivt layout, uden at skulle skrive det hele om igen. Her er tanken, i modsætning til det responsive, ”**Mobile last**”.



**Kilder:**

* <https://da.wikipedia.org/wiki/HTML>
* <https://da.wikipedia.org/wiki/CSS>
* <https://www.w3schools.com/>
* <https://www.netkablet.dk/internettet-skrfter-mobile-first-design/>
* Intro til HTML.pdf
* Intro til CSS.pdf
* Versionsstyring.pdf
* <https://en.wikipedia.org/wiki/Version_control>
* Responsiv i praksis.pdf

**Billeder:**

* CSS-syntaks - Intro til CSS.pdf (side 9)
* <Img> element - <https://cdo-curriculum.s3.amazonaws.com/media/uploads/img_tag.png>
* Atributter - Intro til HTML.pdf (side 12)
* Links - Intro til HTML.pdf (side 23)
* Box model - Intro til CSS.pdf (side 23)
* RGB calculator - <https://www.w3schools.com/colors/colors_rgb.asp>
* HEX-kode - <https://www.w3schools.com/colors/colors_hexadecimal.asp>
* HSL-calculator <https://www.w3schools.com/colors/colors_hsl.asp>
* Statisk webdesign – Responsiv i praksis.pdf (side 6)