

# RESPONSIVE DESIGN

**SDBG Forår 2020**

# INTENDED LEARNING OUTCOMES

Efter kurset kan du:

- levere et overblik over, hvad user experience er, og hvordan det kan måles
- redegøre for designmetoder til udarbejdelse af brugergrænseflader
- designe en grænseflade systematisk ud fra en designmetode
- udarbejde prototyper af brugergrænsefladen og vurdere hensigtsmæssigheden af forskellige slags prototyper i givne situationer
- teste og analysere user experience med en prototype
- forklare hvordan en webbrowser er opbygget og fungerer
- **udarbejde en semantisk korrekt skalerbar brugergrænseflade ved brug af HTML og CSS**
- implementere logik der muliggør brugerrejsen gennem de forskellige dele af løsningen ved brug af JavaScript og API-kald

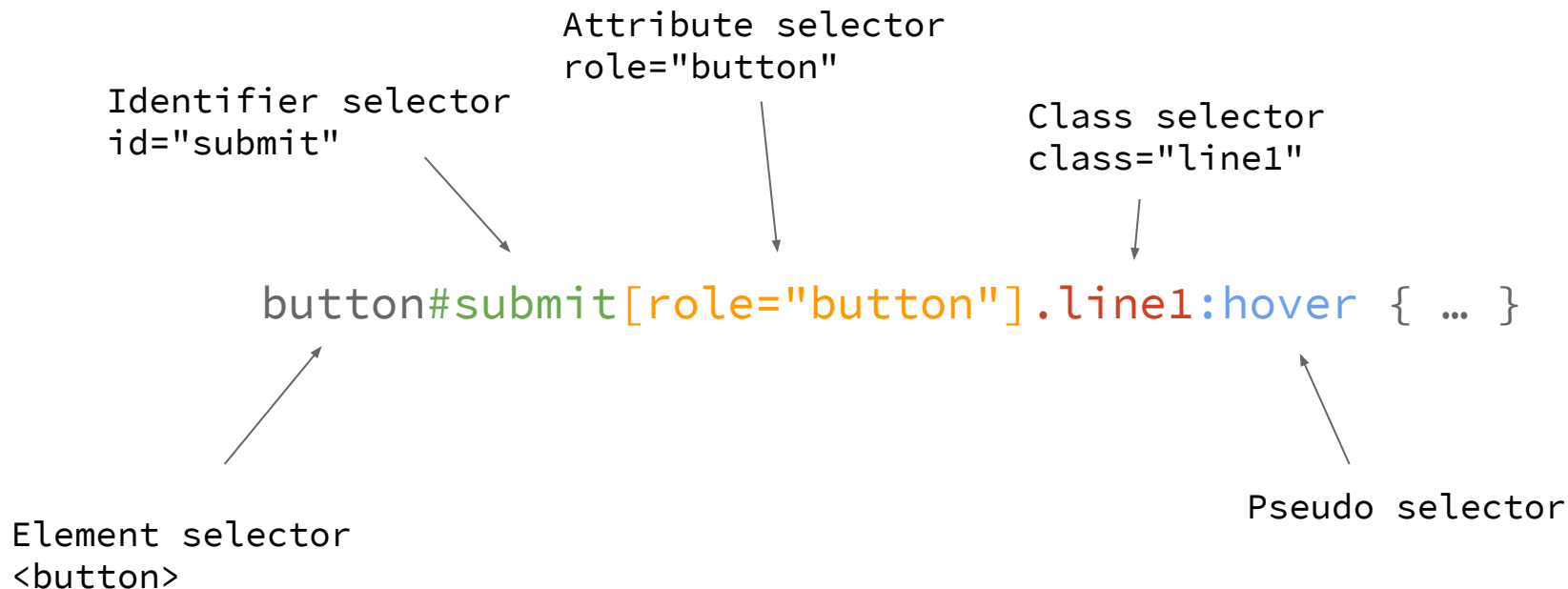
# INTENDED LEARNING OUTCOMES

Efter kurset kan du:

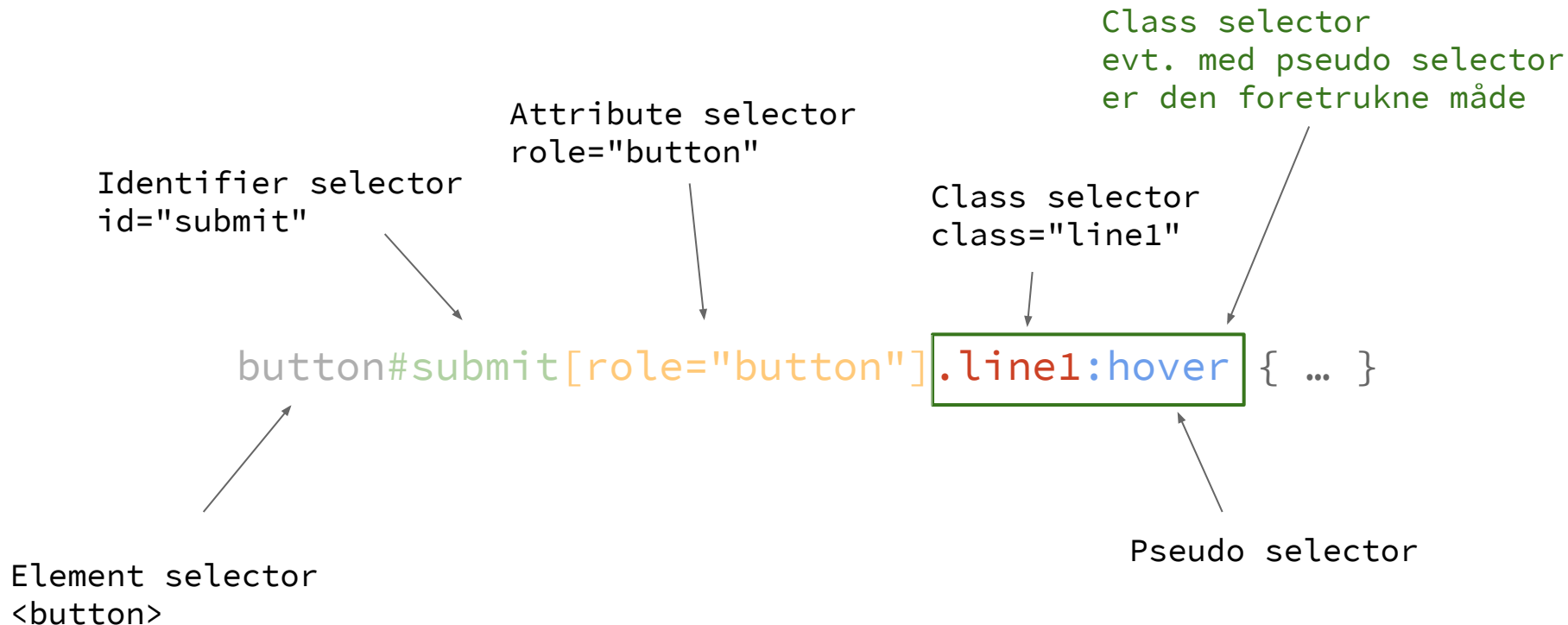
- levere et overblik over, hvad user experience er, og hvordan det kan måles
- redegøre for designmetoder til udarbejdelse af brugergrænseflader
- designe en grænseflade systematisk ud fra en designmetode
- udarbejde prototyper af brugergrænsefladen og vurdere hensigtsmæssigheden af forskellige slags prototyper i givne situationer
- teste og analysere user experience med en prototype
- forklare hvordan en webbrowser er opbygget og fungerer
- **udarbejde en semantisk korrekt skalerbar brugergrænseflade ved brug af HTML og CSS**
- implementere logik der muliggør brugerrejsen gennem de forskellige dele af løsningen ved brug af JavaScript og API-kald

# CSS RECAP

# SELECTOR TYPER



# SELECTOR TYPER



# UDSEENDE

## CSS

```
.class1 {  
  font: sans-serif;  
  font-size: 1em;  
  color: orange;  
  margin: 0.25em;  
}
```

## HTML

```
<div class="class1">Eksempel</div>
```

## OUTPUT

Eksempel

# ADDITIV SAMMENLÆGNING AF KLASSER

## CSS

```
.class1 {  
  font: sans-serif;  
  font-size: 1em;  
  color: orange;  
  margin: 0.25em;  
}
```

```
.class2 {  
  color: red;  
  font-size: 1.5em;  
}
```

## HTML

```
<div class="class1 class2">  
  Eksempel  
</div>
```

## OUTPUT

Eksempel



# RESET ELLER NORMALIZE

Inkluder altid et **CSS reset** eller **normalize** for at få et ensartet udgangspunkt for dokumentet, på tværs af enheder, OS og browsere.

## This is a motherfucking website.

And it's fucking perfect.

### Seriously, what the fuck else do you want?

You probably build websites and think your shit is special. You think your 13 megabyte parallax-ative home page is going to get you some fucking Awwward banner you can glue to the top corner of your site. You think your 40-pound jQuery file and 83 polyfills give IE7 a boner because it finally has box-shadow. Wrong, motherfucker. Let me describe your perfect-ass website:

- Shit's lightweight and loads fast
- Fits on all your shitty screens
- Looks the same in all your shitty browsers
- The motherfucker's accessible to every asshole that visits your site
- Shit's legible and gets your fucking point across (if you had one instead of just 5mb pics of hipsters drinking coffee)

### Well guess what, motherfucker:

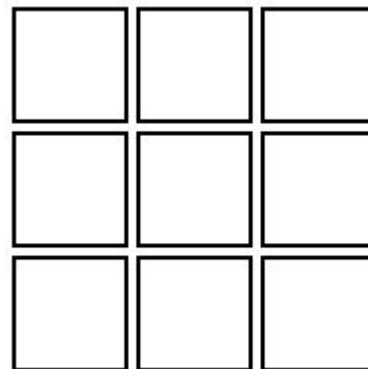
You. Are. Over-designing. Look at this shit. It's a motherfucking website. Why the fuck do you need to animate a fucking trendy-ass banner flag when I hover over that useless piece of shit? You spent hours on it and added 80 kilobytes to your fucking site, and some motherfucker jabbing at it on their iPad with fat sausage

LAYOUT RECAP

# LAYOUT RECAP

tables is ... just don't!

HTML tables er fantastiske til at vise tabel data, men må ikke bruges til layout.




# DIV LAYOUT

```
.green {  
  float: left;  
}  
.red {  
  float: none;  
}  
.blue {  
  float: right;  
}
```

Lorem ipsum dolor sit amet.

Ut rhoncus nibh.

Vestibulum id erat.

 Nullam eget nulla  
nibh.

Sed fermentum gravida..

Sed eget ultrices velit.



Fusce malesuada sem quis  
tortor.

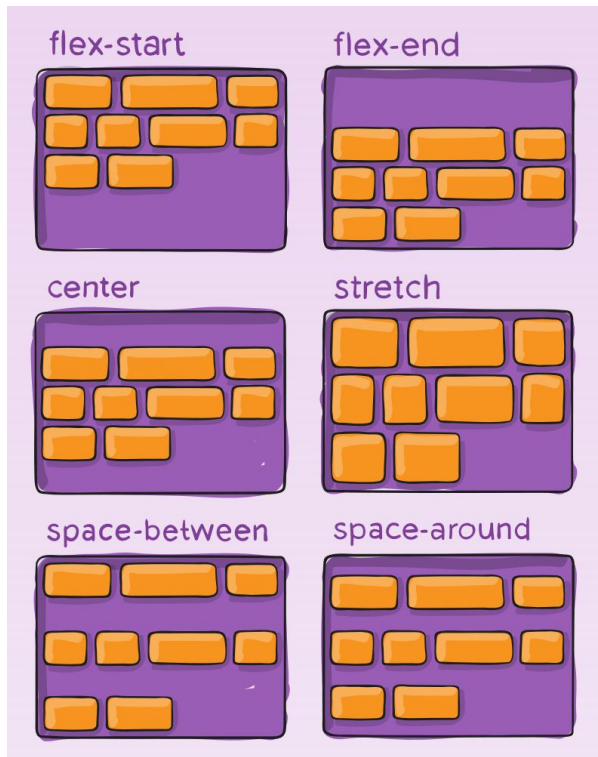
Sed interdum sodales  
finibus.



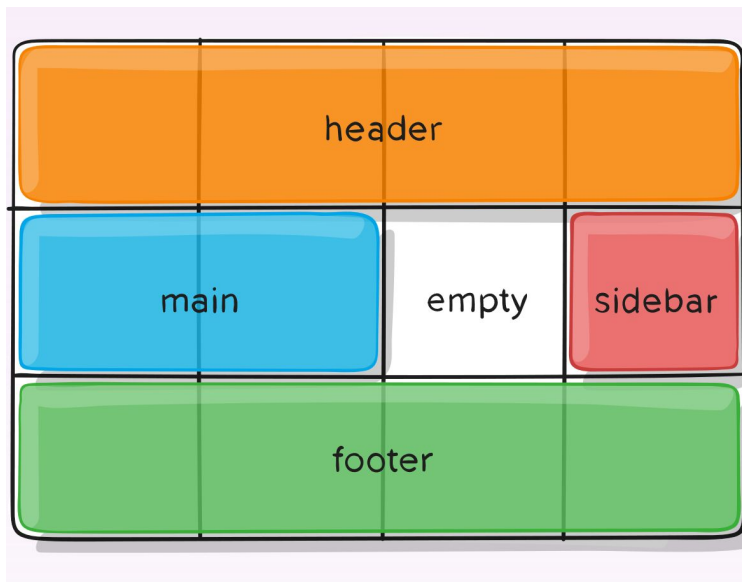
Pellentesque molestie.

Aliquam dictum dui.

# FLEXBOX OG GRID LAYOUT



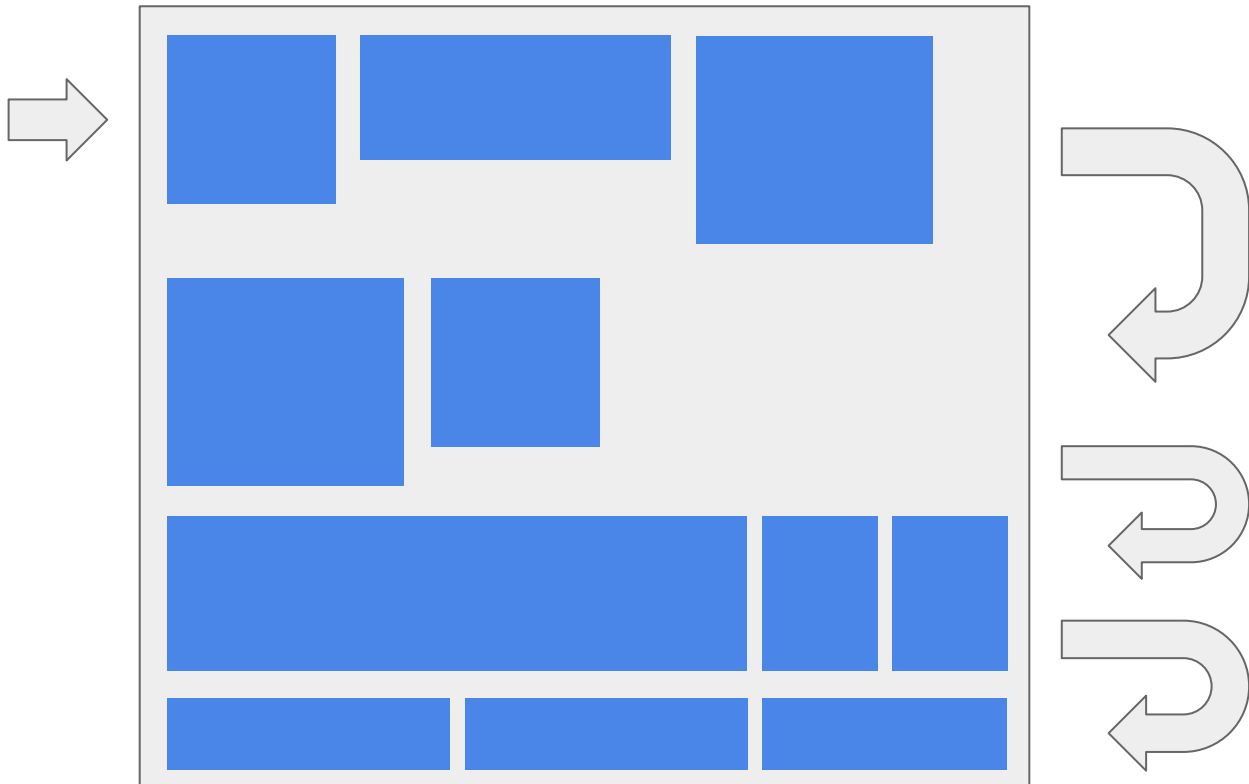
Flexbox og grid er moderne css layout teknikker der giver høj fleksibilitet og er gode til responsive layout.



# DOCUMENT FLOW

# DOCUMENT FLOW

Browserens  
rendering  
engine  
renderer  
dokumentet som  
en række  
kasser som  
blier lagt på  
række og  
bryder om på  
næste linje.



# DOCUMENT FLOW

Element kan tages ud af flowet ved at bruge position:

- `position: absolute;`  
fastgør elementet på koordinater i dokumentet
- `position: fixed;`  
som `absolute`, men elementet bliver "hængende" ved scroll



# DOCUMENT FLOW

Fra Mozilla MDN:

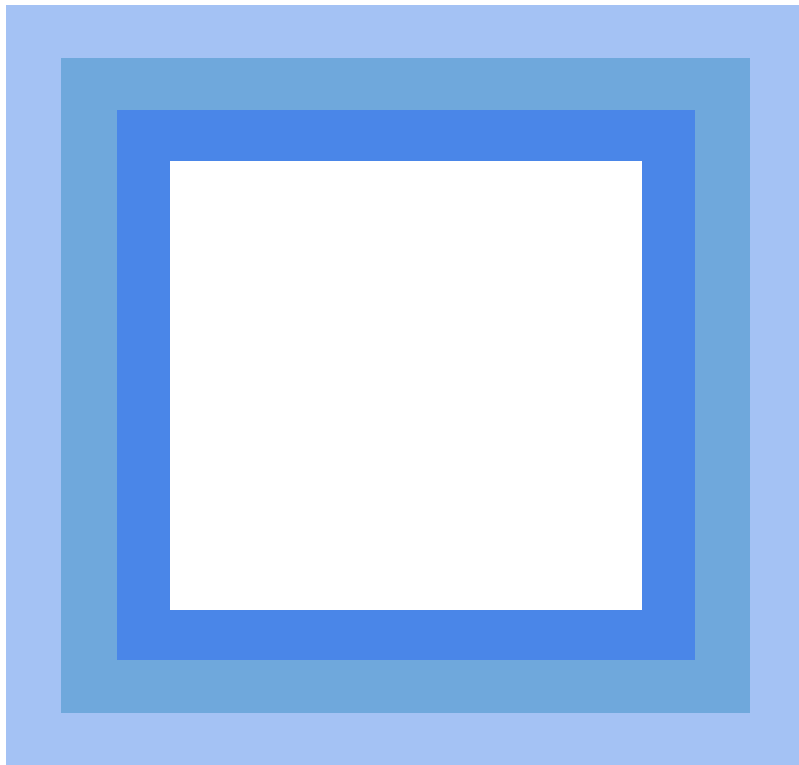
"When laying out a document, the browser's rendering engine represents each element as a rectangular box according to the standard CSS box model.

CSS determines the size, position, and properties (color, background, border size, etc.) of these boxes."

# BOX MODEL

Størrelser på elementer kan fastsættes ved brug af:

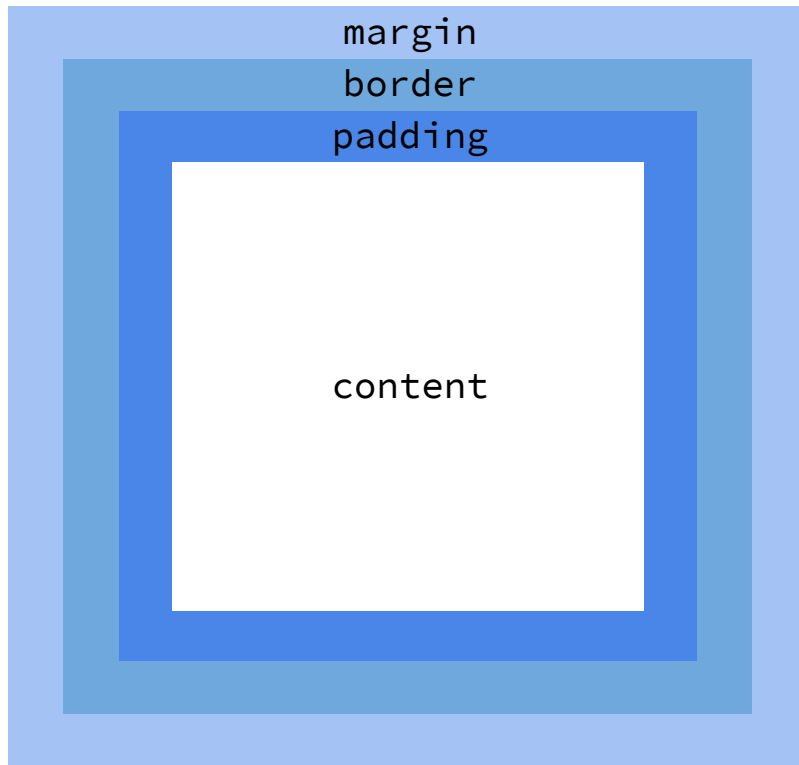
- `width: <value><unit>;`
- `height: <value><unit>;`



# BOX MODEL

Hver box defineret af et HTML element består af 2 dele.

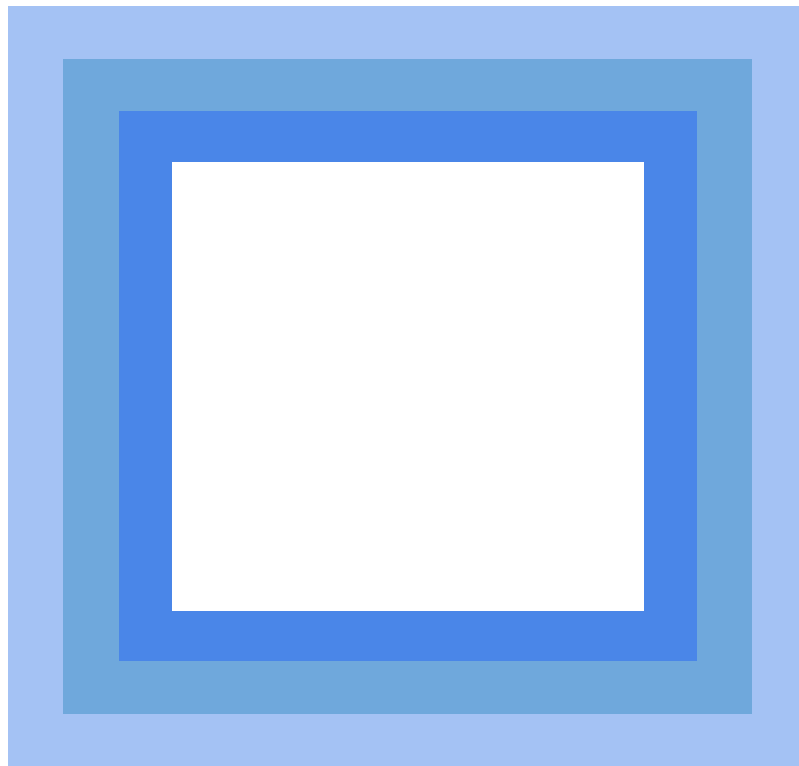
- content edge
- padding edge
- border edge
- margin edge



# BOX MODEL

```
width: 100px;  
height: 100px;
```

box-model definerer hvad  
der skal inkluderes når  
bredde og højde angives.

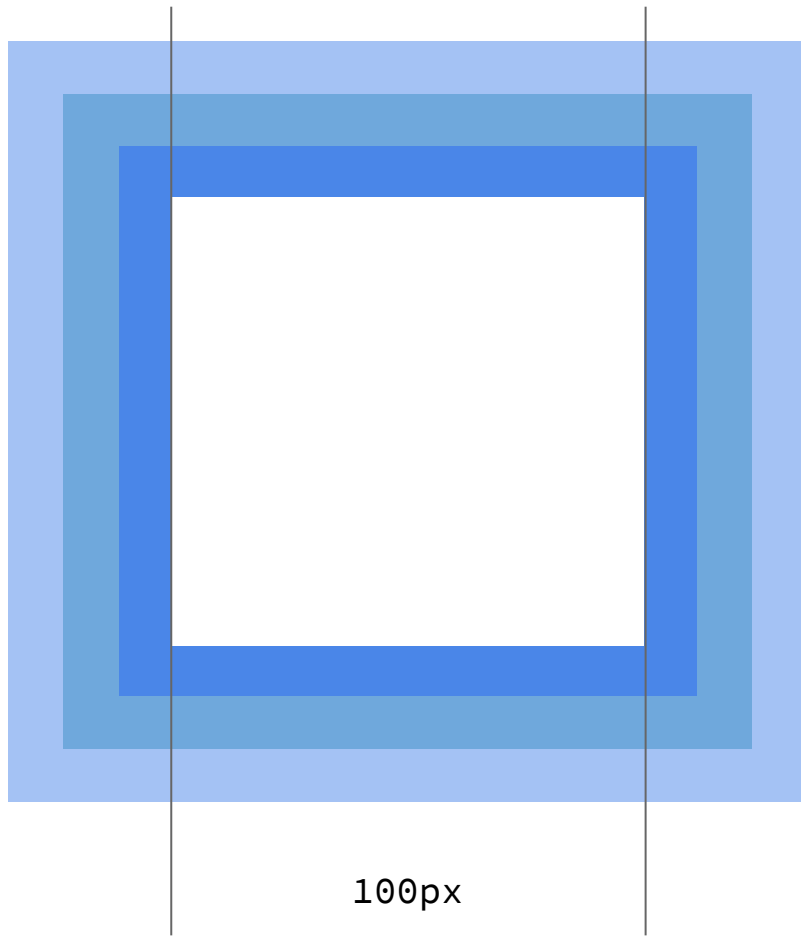


100px

# BOX MODEL

```
width: 100px;  
height: 100px;  
box-model: content-box;
```

Content-box er standard  
for block elementer.

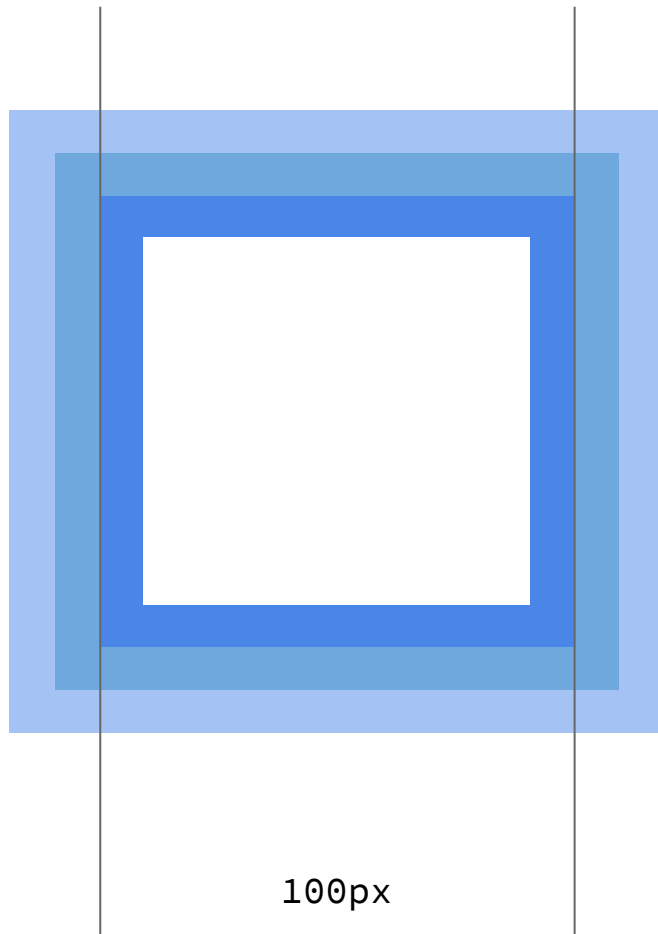


# BOX MODEL

`width: 100px;`

`height: 100px;`

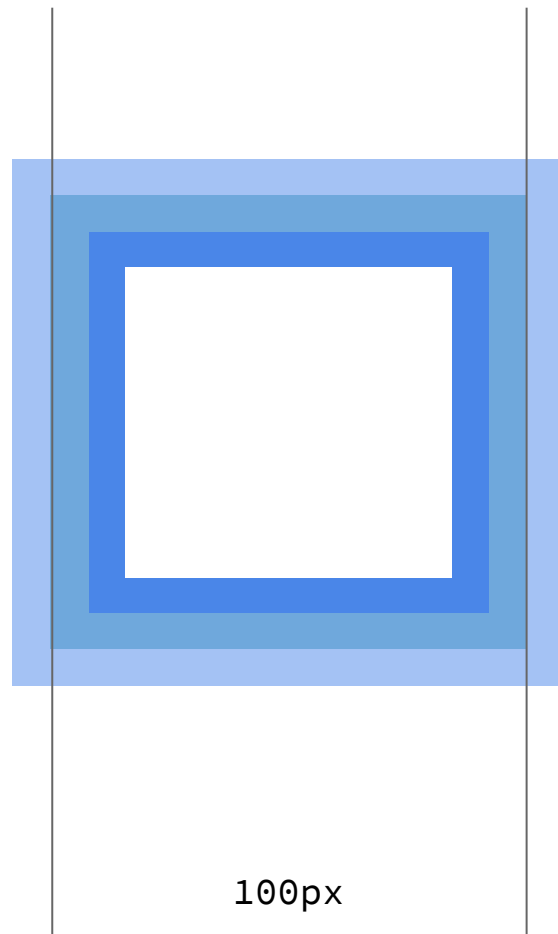
`box-model: padding-box;`



# BOX MODEL

```
width: 100px;  
height: 100px;  
box-model: border-box;
```

Den her er den mest  
brugbare.

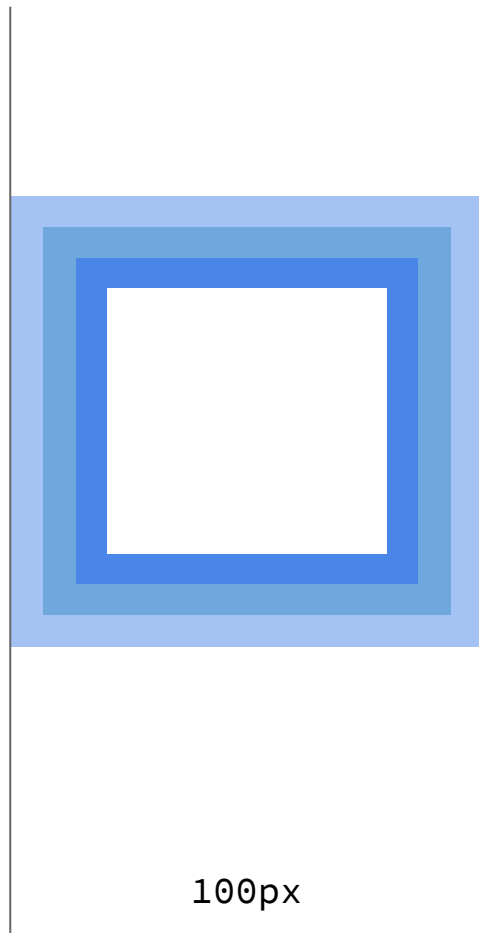


# BOX MODEL

`width: 100px;`

`height: 100px;`

`box-model: margin-box;`





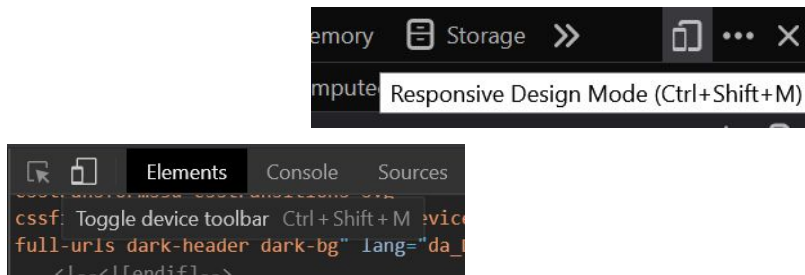
RESPONSIVE SETUP

# RESPONSIVE OPSÆTNING

```
<!DOCTYPE html>
<html>
  <head>
    <title>titel</title>
  </head>
  <body>
    <p>indhold</p>
  </body>
</html>
```

Opret responsive.html med dette markup.

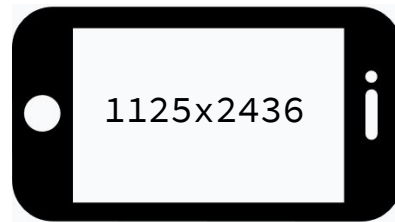
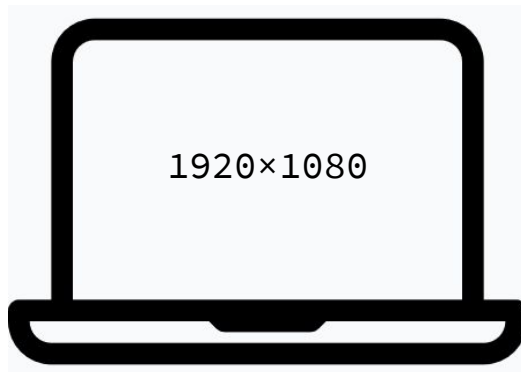
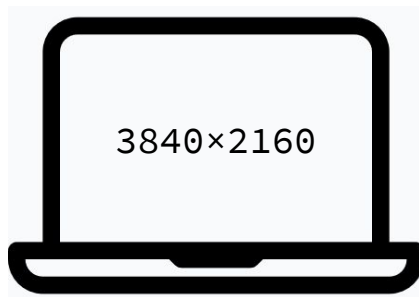
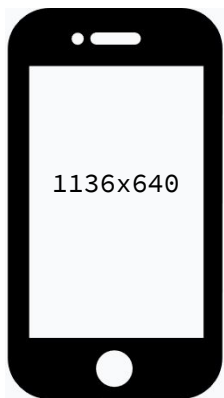
Åbn Responsive design via Developer tools i Chrome eller Firefox.



Indstil Responsive design til at være en smartphone android eller ios

# DEVICE INDEPENDENT-PIXELS

Behov for abstrakt pixel for at kunne tilpasse størrelser til at mange forskellige skærmstørrelser og opløsninger.



# DEVICE INDEPENDENT-PIXELS

Device independent-pixels (dip og dp)  
er defineret således af w3c:

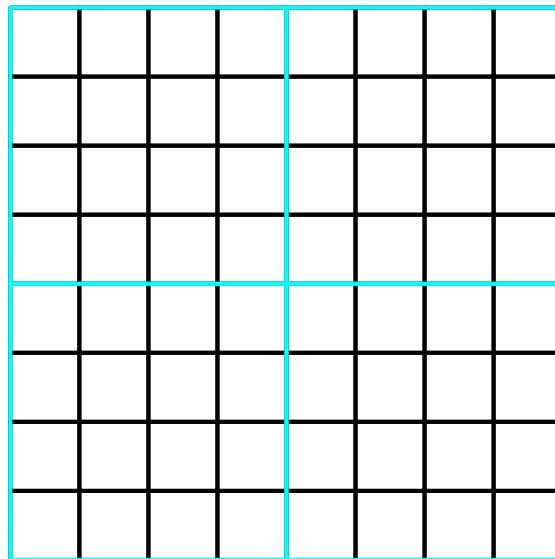
"The reference pixel is the visual angle of one pixel on a device with a pixel density of 96dpi and a distance from the reader of an arm's length. For a nominal arm's length of 28 inches, the visual angle is therefore about 0.0213 degrees. For reading at arm's length, 1px thus corresponds to about 0.26 mm (1/96 inch)."

# DEVICE INDEPENDENT-PIXELS

Enheder med højere dpi end 96 vil så bruge flere fysiske pixels til at repræsenterer en enkelt dp-pixel.

Oftte referreret som CSS *pixels* og *Device pixels*.

Nye smartphone har en dpi på omkring 500.



2x2 px (CSS) bliver repræsenteret af 8x8 px (Device)

# RESPONSIVE OPSÆTNING

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>titel</title>
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
  </head>
```

```
  <body>
```

```
    <p>indhold</p>
```

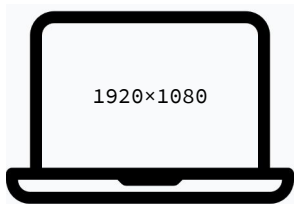
```
  </body>
```

```
</html>
```

Opret responsive.html med dette markup.

viewport meta-element fortæller browseren hvordan den skal udregne størrelser i forhold til enheden.

# MEDIA QUERIES

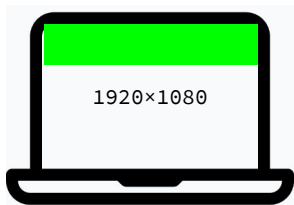


Device pixels: 640x1136

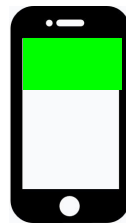


CSS pixels: 320x568

# MEDIA QUERIES



Header er for høj  
på små skærme

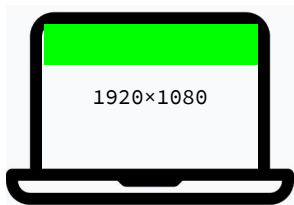


CSS pixels: 320x568

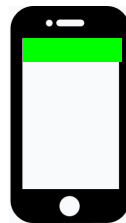
```
.header {  
  display: block;  
  width: 100%;  
  height: 200px;  
  background: green;  
}
```



# MEDIA QUERIES



```
.header {  
  display: block;  
  width: 100%;  
  height: 200px;  
  background: green;  
}
```



CSS pixels: 320x568

```
@media(max-width:480px) {  
  .header {  
    height: 50px;  
  }  
}
```

# RESPONSIVE OPSÆTNING

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>titel</title>
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
    <link href="styles.css" rel="stylesheet">
```

```
  </head>
```

```
  <body>
```

```
    <p>indhold</p>
```

```
  </body>
```

```
</html>
```

Opret opret en tom styles.css fil.

Link til styles.css

Dette vil være vores mobile-first styles.

# RESPONSIVE OPSÆTNING

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>titel</title>
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
    <link href="styles.css" rel="stylesheet">
```

```
    <link href="enhanced.css" rel="stylesheet">
```

```
  </head>
```

```
  <body>
```

```
    <p>indhold</p>
```

```
  </body>
```

```
</html>
```

Opret opret en tom styles.css fil.

Link til enhanced.css  
Styles til større skærme skrives  
i denne fil.

# MEDIA QUERIES OG BREAK POINTS

Breakpoints er de steder hvor man ændrer i layoutet, typisk ud fra browser vinduets bredde.

```
.product-img {  
  width: 50%;  
  float: left;  
}  
  
@media screen and (max-width: 480px) {  
  .product-img {  
    width: auto;  
    float: none;  
  }  
}
```

## **DONT!**

Dette responsive CSS gør at enheder med små skærme skal merge CSS regler (kræver CPU), og enheder med store skærme slipper.

# MEDIA QUERIES OG BREAK POINTS

Breakpoints er de steder hvor man ændrer i layoutet, typisk ud fra browser vinduets bredde.

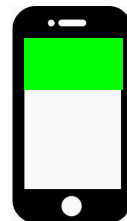
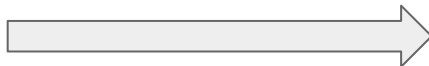
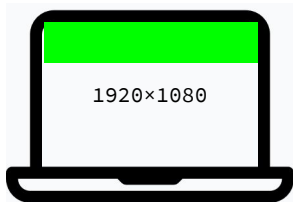
```
.product-img {  
  width: auto;  
  float: none;  
}  
  
@media screen and (min-width: 480px) {  
  .product-img {  
    width: 50%;  
    float: left;  
  }  
}
```

## **DO!**

Dette responsive CSS gør det omvendt, altså mobile-first. Enheder med små skærme slipper billigere (mindre CPU)

# MOBILE TWEAKING

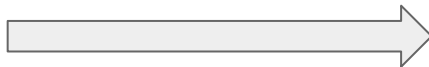
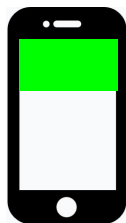
Undgå at udvikle UI på desktop,  
hvis det skal kunne vises på mobil



CSS pixels: 320x568

# MOBILE FIRST

Start på mobile og "enhance" jeres CSS til at skalere layout op til store skærme.



# RESPONSIVE OPSÆTNING

## HTML

```
<body>
  <div class="header"></div>
  <p>indhold</p>
</body>
```

Tilføj div med class header i jeres responsive.html

## styles.css

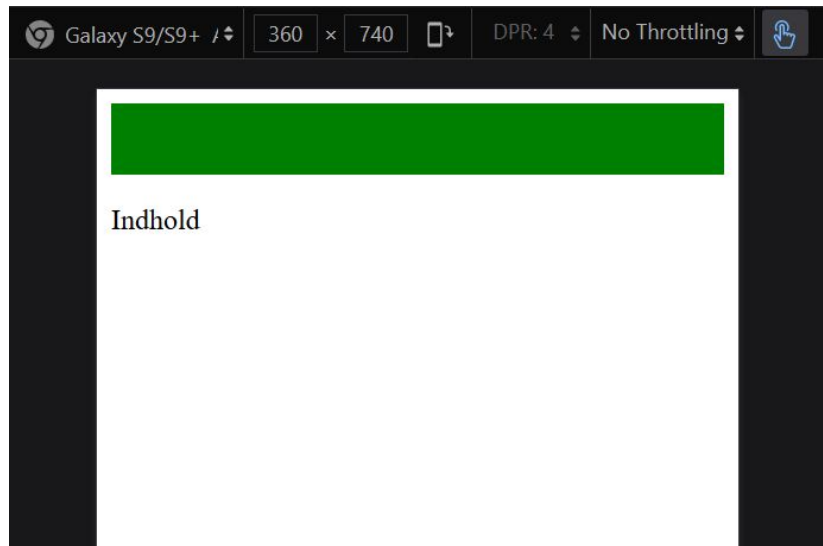
```
.header {
  background: green;
  height: 40px;
}
```

Tilføj regel med selector .header



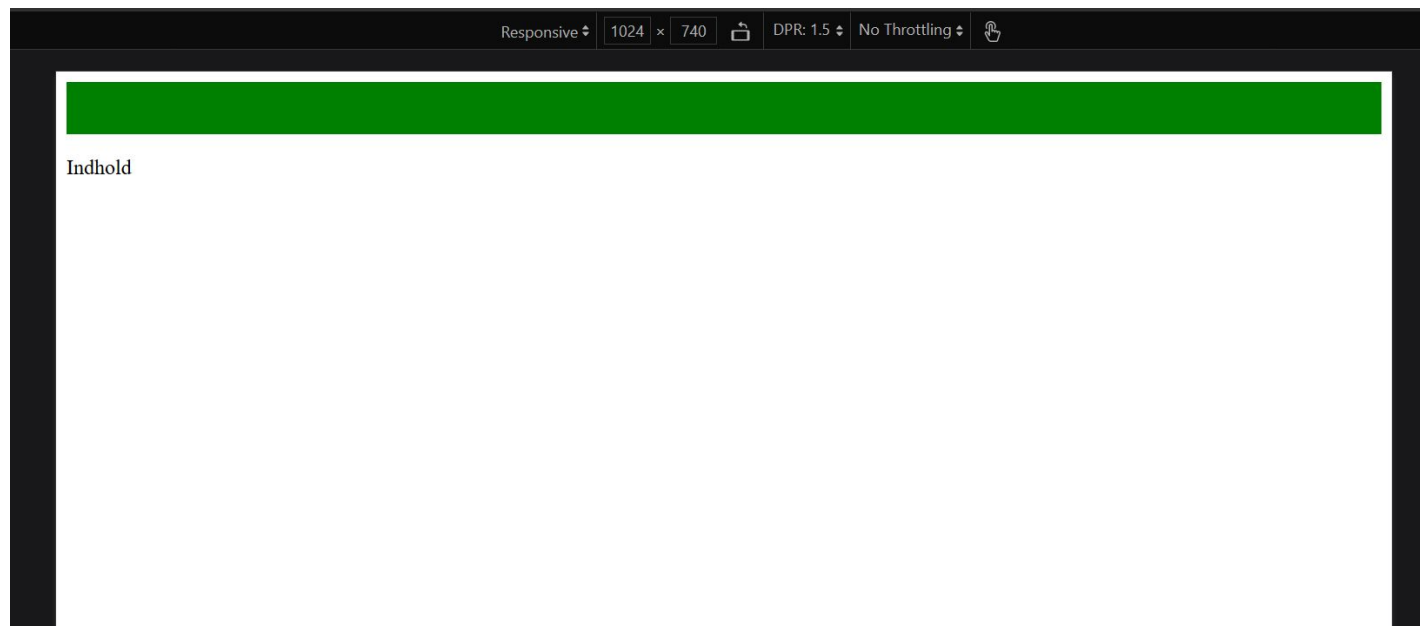
# RESPONSIVE BILLEDER

Visning af vores header som den vil se ud på mobil.



# RESPONSIVE BILLEDER

Visning af vores header som den vil se ud på desktop.



# RESPONSIVE OPSÆTNING

## **enhanced.css**

```
@media screen and (min-width: 480px) {  
  
}
```

Tilføj et media query koblet på screen mediet, og som skal gælde når skærmen er mindste 480px bred.

# RESPONSIVE OPSÆTNING

## **enhanced.css**

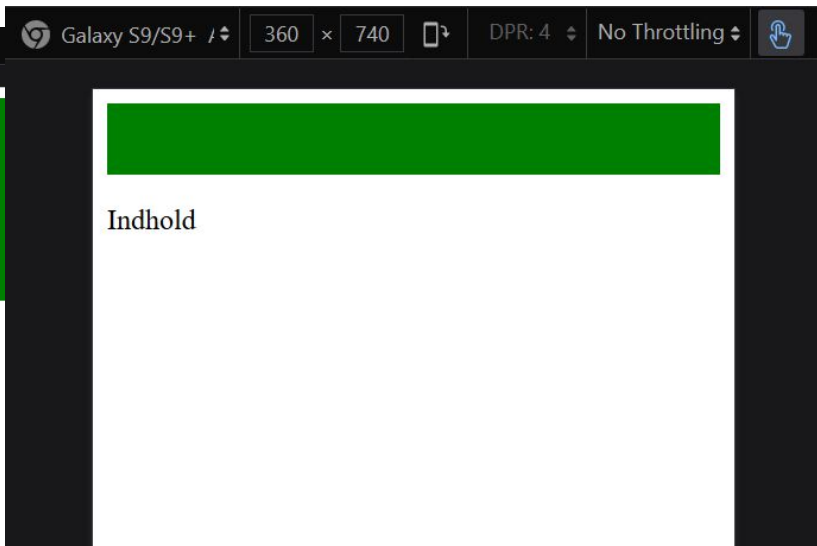
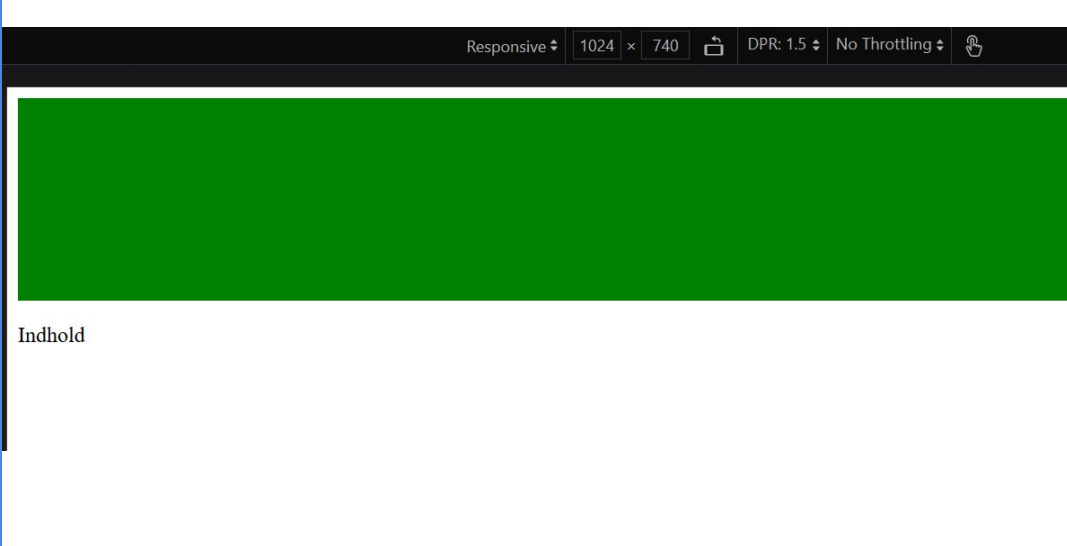
```
@media screen and (min-width: 480px) {  
  .header {  
    height: 150px;  
  }  
}
```

Gentag regel med selector  
.header og just højden til  
150px.

# RESPONSIVE OPSÆTNING

Desktop

Smartphone



# RESPONSIVE OPSÆTNING

## **enhanced.css**

```
@media screen and (min-width: 480px) {  
  .header {  
    height: 150px;  
  }  
}
```

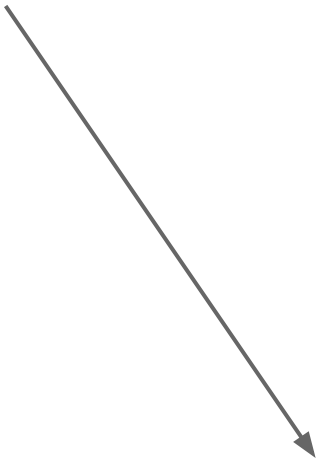
Ved at linke til styles.css og enhanced.css vil begge filer blive downloadet, men på små skærme vil indholdet af enhanced.css ikke blive brugt.

Det kan vi gøre noget ved!

# RESPONSIVE OPSÆTNING

## enhanced.css

```
@media screen and (min-width: 480px) {  
  .header {  
    height: 150px;  
  }  
}
```



Ved at indsætte media query i media attributten kan vi undgå at alle enheder henter css filen, selvom de ikke skal bruge den.

Herefter kan vores media query i enhanced.css slettes.

## responsive.html

```
<link href="enhanced.css" media="screen and (min-width: 480px)" rel="stylesheet">
```

# MEDIA QUERIES

## File splitting



## Nested queries

```
.product-img {
  width: auto;
  float: none;
}

@media screen and (min-width: 480px) {
  .product-img {
    width: 50%;
    float: left;
  }
}
```



# RESPONSIVE IMAGES

# RESPONSIVE IMAGES

Billeder på nettet kan deles i 2.

Raster billeder (PNG og JPG)



Vector billeder (SVG)



# RASTER BILLEDER

Raster billeder er baseret på farve information for hver pixel i billedet. Det bliver hurtigt meget data.

Fordi farver i billedet er baseret på pixels skalerer de ikke særlig pænt. Slet ikke hvor billedet bruges større end dets dimensioner er.

Brug JPG til fotos med mange farver.

Brug PNG til ikoner og symboler med få farver.



# VEKTOR BILLEDER

Vektor billeder bliver tegnet i browseren ud fra tekst baseret instruktioner.

## Cirkel:

```
<svg xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle cx="40" cy="40" r="24" style="stroke:#006600; fill:#00cc00"/>
</svg>
```

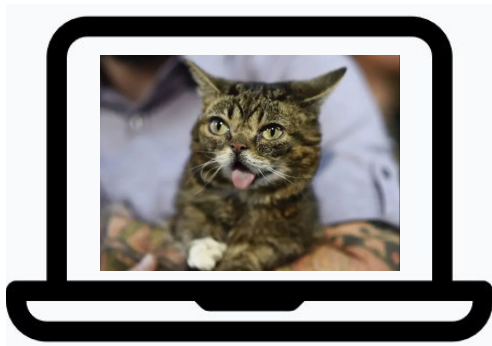
Derfor kan vektor billeder skaleres uden fald i kvalitet. Det er dog ikke ment til billeder med mange detaljer eller farver.

Brug SVG til ikoner og symboler.

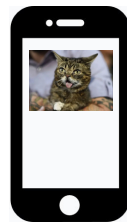


# RESPONSIVE BILLEDER

Mobile enheder har ikke kun mindre plads på skærmen, de har typisk også en ringe trådløs forbindelse til internettet.



500kb billeder er ok



500kb billeder er  
tungt på mobilnettet

# RESPONSIVE BILLEDER

Et almindelig `img`-element er ikke responsive i forhold til hvilket billede det viser.

```

```

# RESPONSIVE BILLEDER

Source sets og sizes attributter kan hjælpe browseren til at hente et passende billede.

```

```

# RESPONSIVE BILLEDER

Source sets og sizes attributter kan hjælpe browseren til at hente et passende billede.

```

```

Default billede der skal vises hvis ingen billeder i srcset passer, eller hvis browser ikke understøtter srcset attributten.



# RESPONSIVE BILLEDER

Source set (srcset) attributten indeholder listen af billeder browseren kan hente. Efter hvert billede kan vi angive bredden på billedet eller dpi (feks 2x).

```

```

Størrelser i srcset angives med enheden w og referer til bredden på billedet.

# RESPONSIVE BILLEDER



460w



1024w

# RESPONSIVE BILLEDER

Sizes hjælper browseren med at vælge et billede før rendering er gået i gang. Det kan derfor få browseren til at hente billedet tidligere og det føles hurtigere.

```

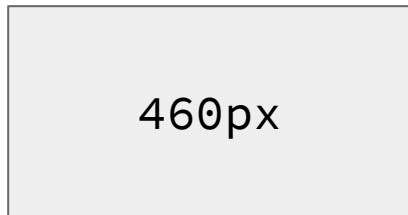
```

Størrelser i size angives med media queries og en angivelse af hvor stort området billedet vises i vil være.

Det bruger browseren til at hente billedet med bredte *460w* når skærmen er mindre end *460px*

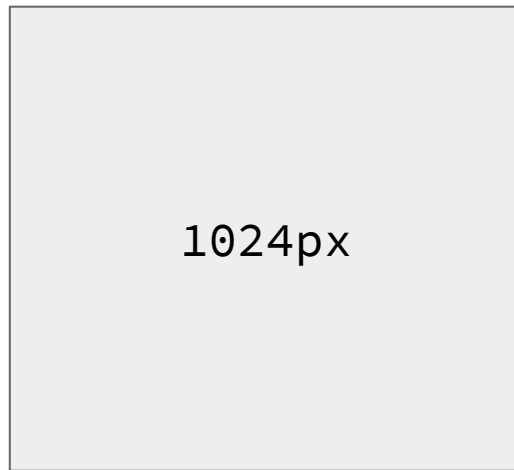
# RESPONSIVE BILLEDER

`sizes="(max-width: 460px) 460px, 1024px"`



I viewports med en bredden på 460px eller mindre, find det bedste billede at indsætte i mit billedslot der er 460px, uagtet af elementets fysiske størrelse.

`sizes="(max-width: 460px) 460px, 1024px"`



I viewports med en bredden på mere end 460px, finder det bedste billede fra srcsets at indsætte i et slot der er 1024px.

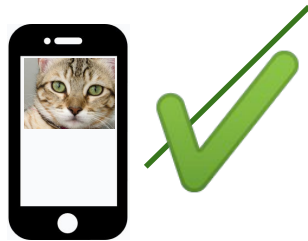
# ART DIRECTION PROBLEM

Forskellige skærme og layout til dem, skabe et problem der hedder  
"The Art Direction Problem"



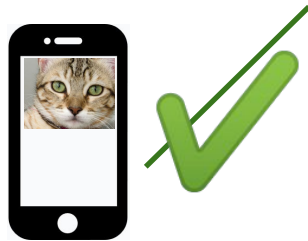
# ART DIRECTION PROBLEM

Ved at bruge forskellige billeder i vores `srcset` og hinte dem med `sizes`, kan vi vise forskellige billedstørrelser, men det kan være svært at blande med forskellig beskæring.



# ART DIRECTION PROBLEM

Ved at bruge forskellige billeder i vores `srcset` og hinte dem med `sizes`, kan vi vise forskellige billeder størrelser af billeder, men det kan være svært at blande med forskellig beskæring.



# PICTURE ELEMENT

For at løse problemet med beskæring kan vi benytte os af `<picture>`

```
<picture>
  <source media="(min-width:460px)"
          srcset="img-50px.png 100w">
  <source srcset="img-500px.png 500w">
  
</picture>
```

Ovenstående gør det samme som vores forrige eksempel med `srcset` og `sizes`.



# PICTURE ELEMENT

Få fuld kontrol over den bedste billedstørrelse at bruge sammen med den rette beskæring ved at blande *picture* med *srcset* og *sizes*.

Her bruger vi 2 forskellige source baseret på enhedens rotation. Derefter vælger vi hvilket billede vil passe bedst i det tilgængelig område.

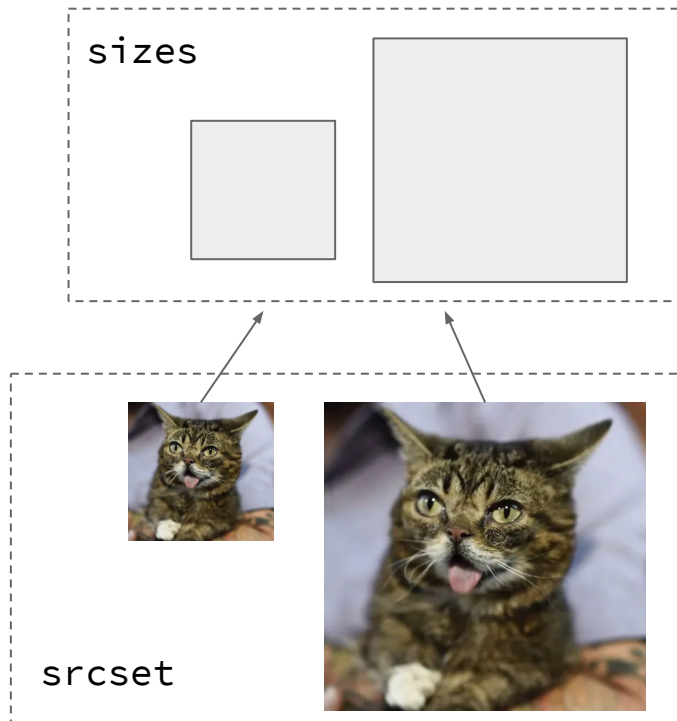
```
<picture>
  <source media="(orientation: landscape)"
    srcset="image-small.png 320w,
            image-large.png 1200w"
    sizes="(min-width: 60rem) 80vw,
           100vw">

  <source media="(orientation: portrait)"
    srcset="image-small-landscape.png 160w,
            image-large-landscape.png 600w"
    sizes="(min-width: 60rem) 80vw,
           100vw">

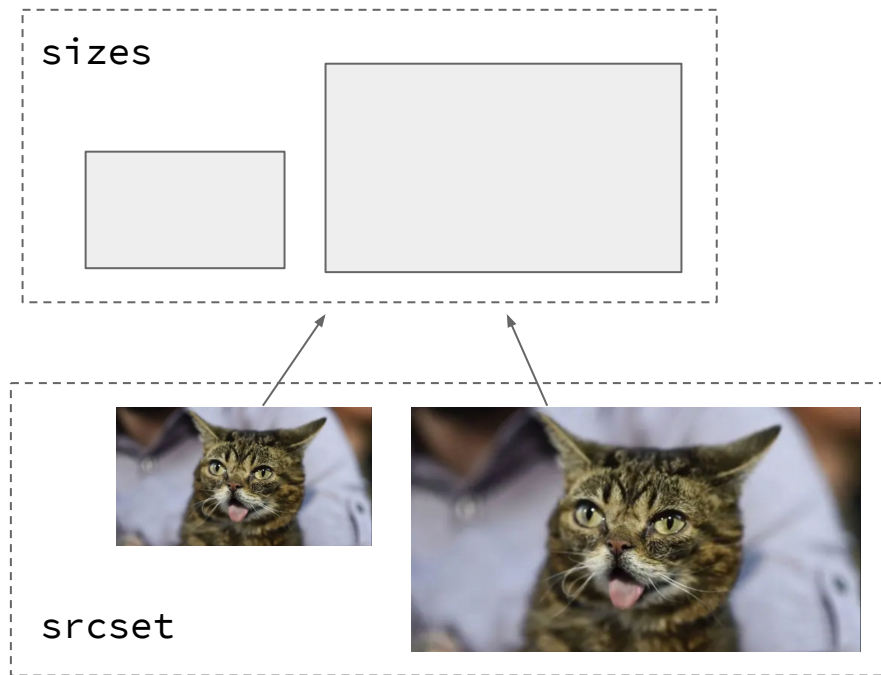
  
</picture>
```

# RESPONSIVE BILLEDER

orientation: portrait



orientation: landscape



# FLEXBOX

## HTML

```
<div class="container">
  <nav>Navigation</nav>
  <main>Indhold</main>
  <aside>Relateret links</aside>
</div>
```

## CSS

```
.container {
  display: flex;
  flex-direction: row;
}
nav {
  flex: 0 1 auto;
}
main {
  flex: 0 1 auto;
}
aside {
  flex: 0 1 auto;
}
```

# FLEXBOX

`flex-direction` bruge til at styre retningen på flex items.

Default er `row`, alternativt kan `column` bruges for at få items til at ligge under hinanden.

## CSS

```
.container {  
  display: flex;  
  flex-direction: row;  
}  
nav {  
  flex: 0 1 auto;  
}  
main {  
  flex: 0 1 auto;  
}  
aside {  
  flex: 0 1 auto;  
}
```

# FLEXBOX

Flex-property på flexbox children angiver hvordan elementerne skal distribuere tilgængelig plads, relativt til de andre elementer.

Flex er shorthand for:

flex-grow, flex-shrink, flex-basis

*flex-grow*: styrer hvordan et element vokser.

*flex-shrink*: styrer hvordan elementet skal blive mindre.

*flex-basis*: styrer hvilket udgangspunkt der skal skaleres fra.

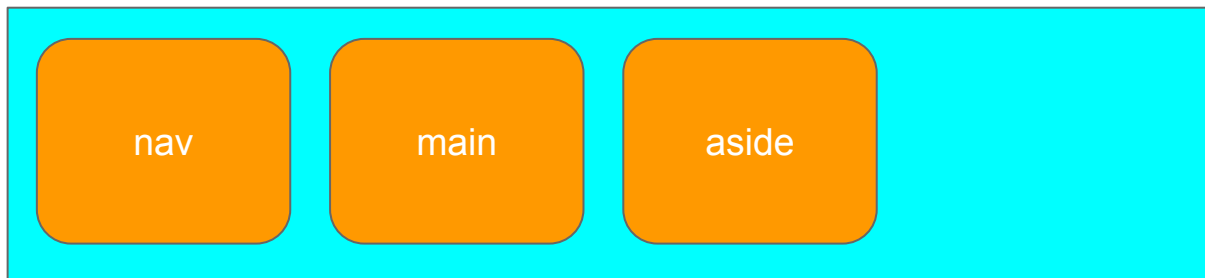
## CSS

```
.container {  
  display: flex;  
  flex-direction: row;  
}  
nav {  
  flex: 0 1 auto;  
}  
main {  
  flex: 0 1 auto;  
}  
aside {  
  flex: 0 1 auto;  
}
```

# FLEXBOX

`flex: 0 1 auto`

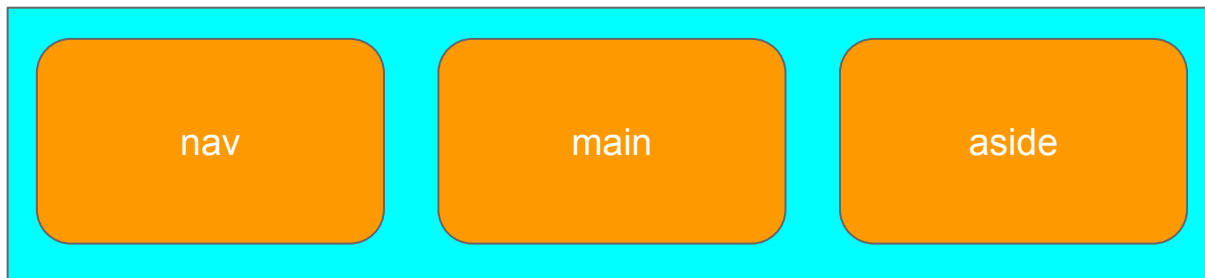
For child elementerne til at fylde det samme som deres indhold og ellers lægge sig på række.



# FLEXBOX

`flex: 1 0 auto`

For child elementerne til at fylde det samme som deres indhold og fordele tilgængelig plads ligeligt imellem sig.



# FLEXBOX

`flex: 2 0 auto`

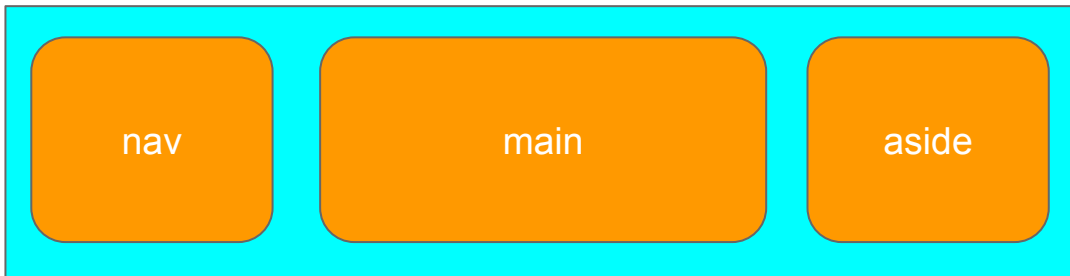
Når plads skal distribueres bliver det delt ud fra summen af flex.

I dette tilfælde  $1+1+2=4$ .

`nav = 1/4`

`main = 2/4`

`aside = 1/4`



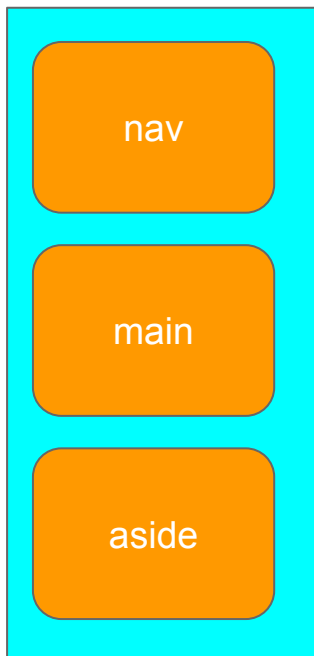
## CSS

```
.container {  
  display: flex;  
  flex-direction: row;  
}  
nav {  
  flex: 1 0 auto;  
}  
main {  
  flex: 2 0 auto;  
}  
aside {  
  flex: 1 0 auto;  
}
```



# FLEXBOX

Med flex-direction kan vi ændre retning for vores child elementer.

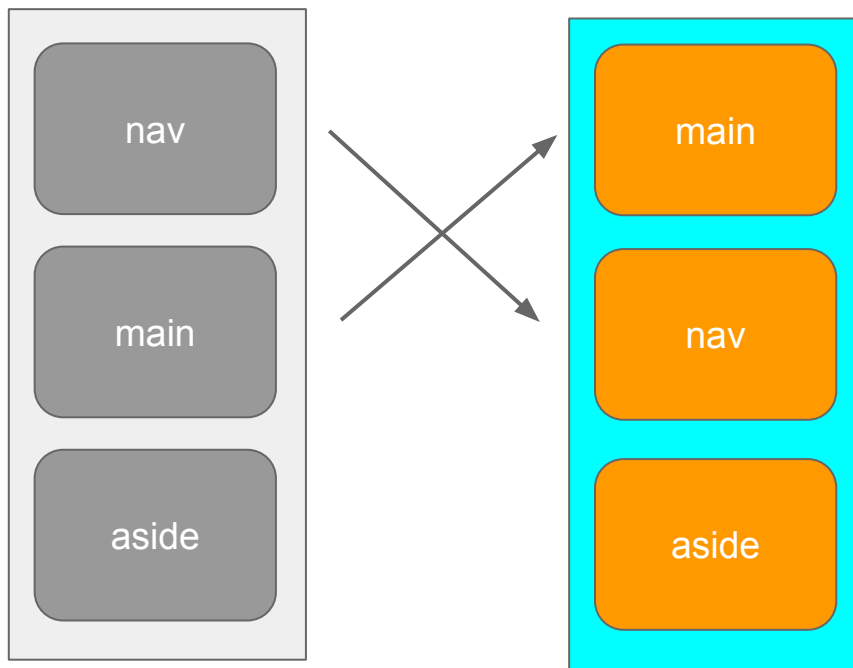


## CSS

```
.container {  
  display: flex;  
  flex-direction: column;  
}  
nav {  
  flex: 1 0 auto;  
}  
main {  
  flex: 2 0 auto;  
}  
aside {  
  flex: 1 0 auto;  
}
```

# FLEXBOX

med `order`-property på vores child elementer kan vi ændre rækkefølge på vores elementer. Jo højere tal, jo senere i rækken.

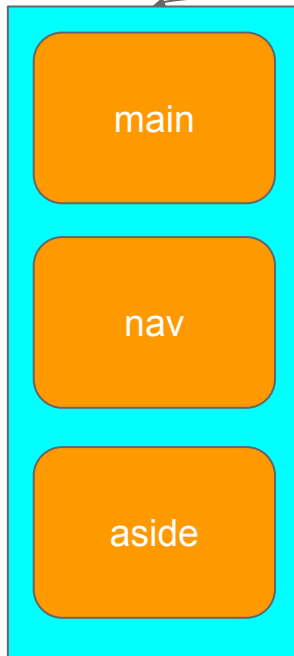


## CSS

```
.container {  
  display: flex;  
  flex-direction: column;  
}  
nav {  
  flex: 1 0 auto;  
}  
main {  
  flex: 2 0 auto;  
  order: -1;  
}  
aside {  
  flex: 1 0 auto;  
}
```

# FLEXBOX

flex-direction og order bliver rigtig stærke når de bruges til responsive design med media queries.



## CSS

```
.container {  
  display: flex;  
  flex-direction: column;  
}  
nav {  
  flex: 1 0 auto;  
}  
main {  
  flex: 2 0 auto;  
  order: -1;  
}  
aside {  
  flex: 1 0 auto;  
}
```

## CSS

```
@media screen and (min-width: 600px) {  
  .container {  
    flex-direction: row;  
  }  
  main {  
    order: 0;  
  }  
}
```

# ØVELSER

Kl 12:15 starter vi Kahoot quiz i Responsive Design fra i dag.

Vi ses efter Kahoot til øvelsestid i disse lokaler:

- 3A52: Kasper & Simon & Christian
- 3A54: Peter & Jakob
- 4A14: Reserve.