Fachhochschule Nordwestschweiz
Hochschule für Technik

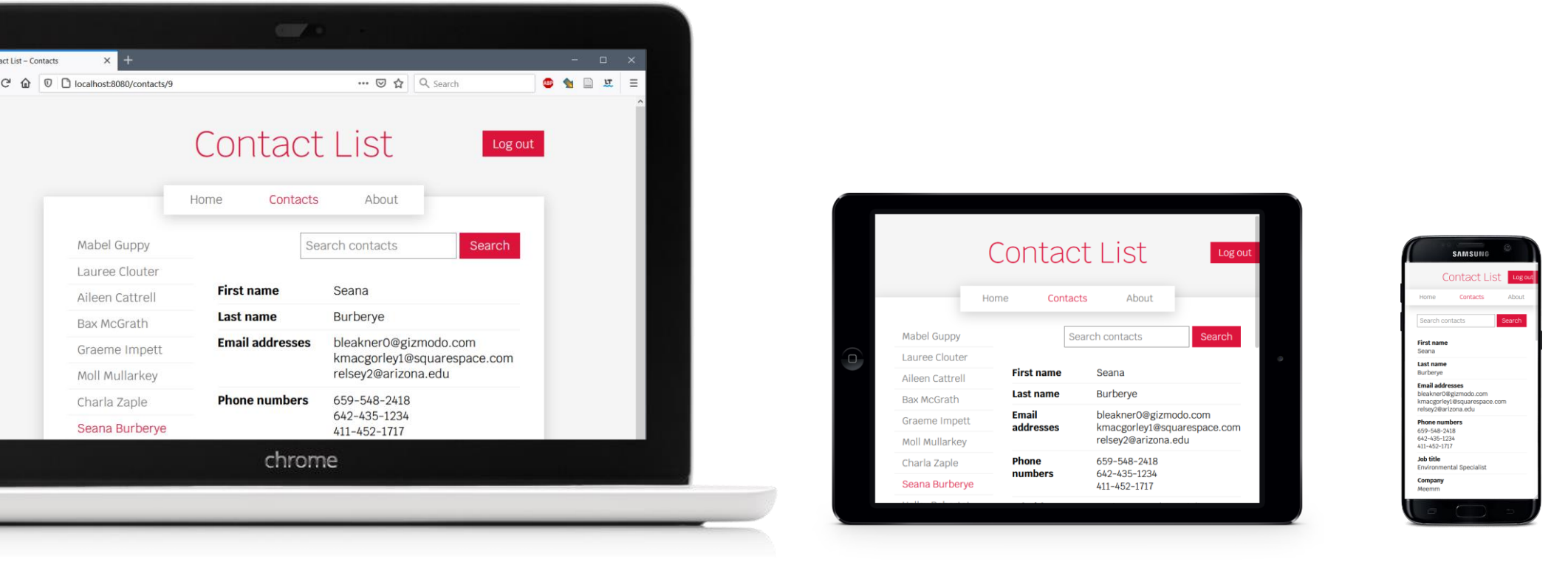
Web Engineering

Responsive Web Design

Adrian Herzog

(basierend auf der Arbeit von Michael Faes, Michael Heinrichs & Prof. Dierk König)

Responsive Design



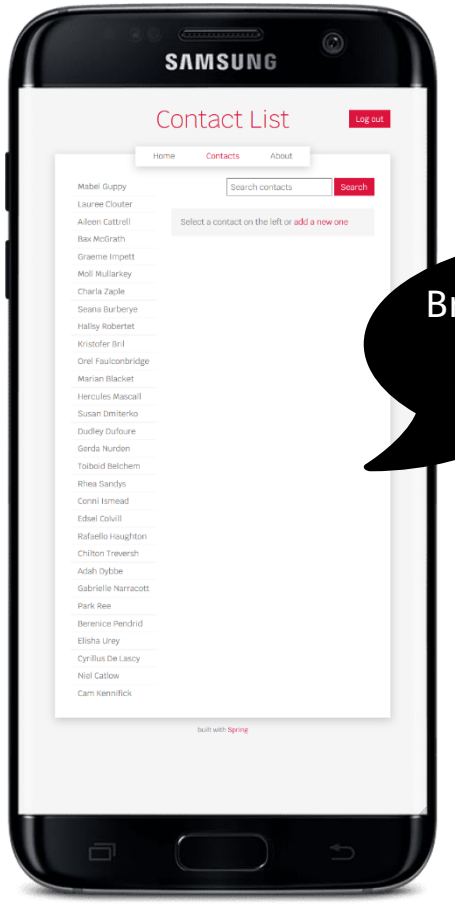
Ziel: Web-App soll auf grossen *und kleinen* Geräten gut aussehen und bedienbar sein. Seite «antwortet» auf Änderungen der Displaygrösse.

Wo steht unsere Web-App?

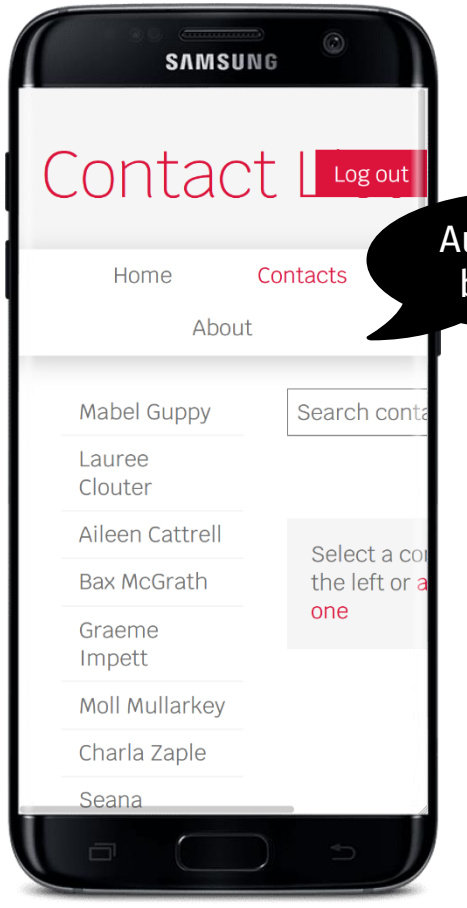
Komplett ohne Anpassung:

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

Grundlage für Responsive Design



Browser von Phones «lügen» über Bildschirmgrösse



Auch nicht besser...

Lösungs-Ansätze

webeC

Flexibles Layout (CSS)

Relative Schriftgrössen, relative Element-Breiten, Grids, …

Media-Queries (CSS)

Displaygrössen-spezifische Regeln

Libraries (CSS/JS)

z. B. *Bootstrap* getbootstrap.com

Dynamische Seitenlogik (JS)

z. B. `onresize="…"`

Geräte-spezifische Views (MVC)

z. B. github.com/blueconic/browscap-java

Flexibles Layout

Grundideen:

- Grösse von Elementen mit *relativen Einheiten* definieren
- Elemente *flexibel platzieren* (basierend auf Grösse & Platz)

Einheiten in CSS:

Einheiten	Beschreibung
`cm`, `mm`, `in`, …	*absolute* Einheiten (für Druckansicht)
`px`	*absolute* Einheit, aber nicht immer = 1 Pixel! (abhängig von Pixeldichte des Displays)
`em`, `ex`, `ch`	*relativ* zur **Schriftgrösse** des (äusseren) Elements (`<html>`-Element hat Defaultgrösse `16px`)
`rem`	*relativ* zur **Schriftgrösse** des `<html>`-Elements
`vw`, `vh`, `vmin`, `vmax`	*relativ* zur **Fenster-/Gerätegrösse**: 1% davon
`%`	*relativ* zur **Breite/Höhe/Schrift des äuss. Elem.**

z. B. Galaxy S20: 1px = 4 Pixel!

«Viewport»

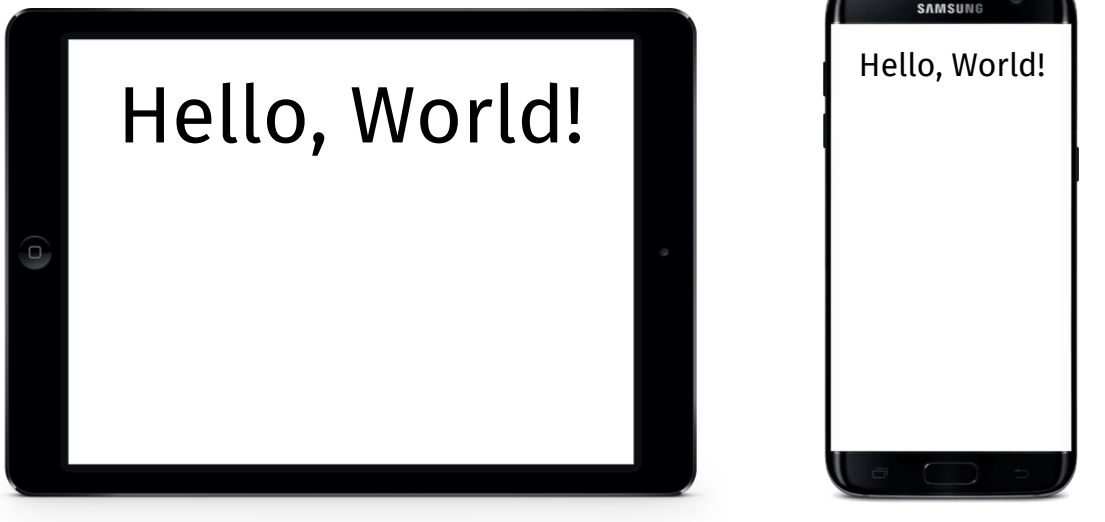
Relative Einheiten: Beispiele

Relative Breite:

```
#cat-img {
    width: 100%;
}
```

Relative Schriftgrösse:

```
h1 {
    font-size: 8vw;
}
```



Übung 1: Erste Schritte mit Responsive

Lade die neue Vorlage in deine IDE, starte die App und betrachte sie im *Responsive Design Mode* in deinem Browser. Wähle als Gerät ein Smartphone mit 360 – 400 px Breite.

1. Füge das in den Folien gezeigte `<meta>`-Element innerhalb von `<head>` zum App-Layout hinzu und beobachte die Änderung im Browser.
2. Mache einen ersten Schritt Richtung Responsive Design, indem du die Titelgrösse für `<h1>` anpasst. Definiere die Grösse relativ zum Viewport, aber nur bis zu einem bestimmten Maximum. Überprüfe im Browser, dass die Titelgrösse jetzt responsive ist.

Flex und Grid Layout

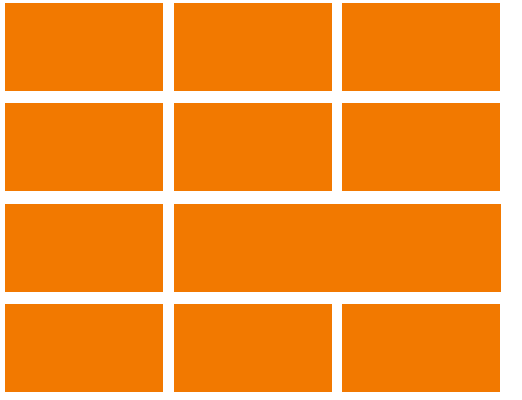
- Beide können für Responsive Layout verwendet werden
- Beide haben gewisse Stärken und Schwächen, siehe auch: Quick! What's the Difference Between Flexbox and Grid?

Flex row

Siehe auch Woche 2 (CSS)

Flex column

Grid



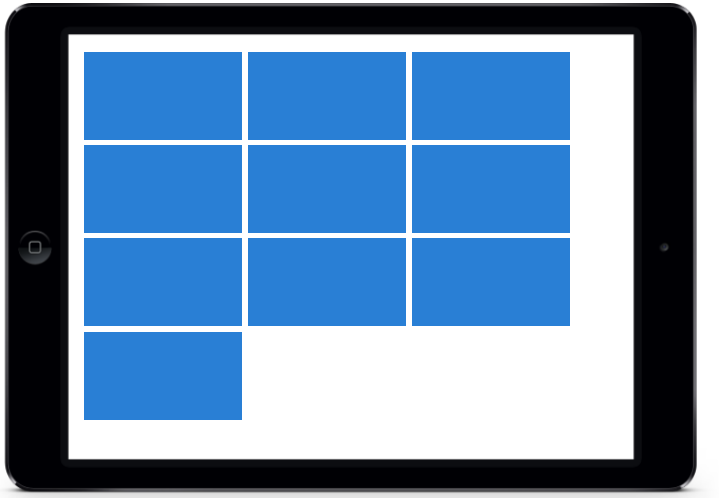
Elemente flexibel platzieren: Grid Layout

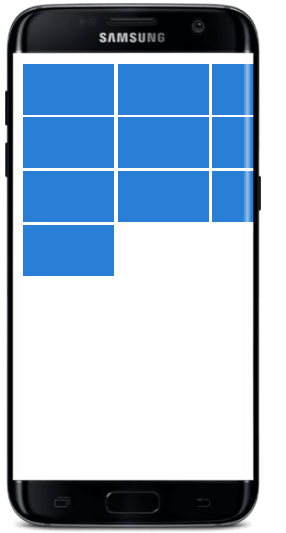
Mächtiger Layout-Mechanismus in CSS: *Grid Layout*

Unflexibel:

```
<div class="gallery">
  <div>…</div>
  <div>…</div>
  <div>…</div>
  ...
</div>
```

```
.gallery {
  display: grid;
  grid-template-columns:
    150px 150px 150px;
}
```



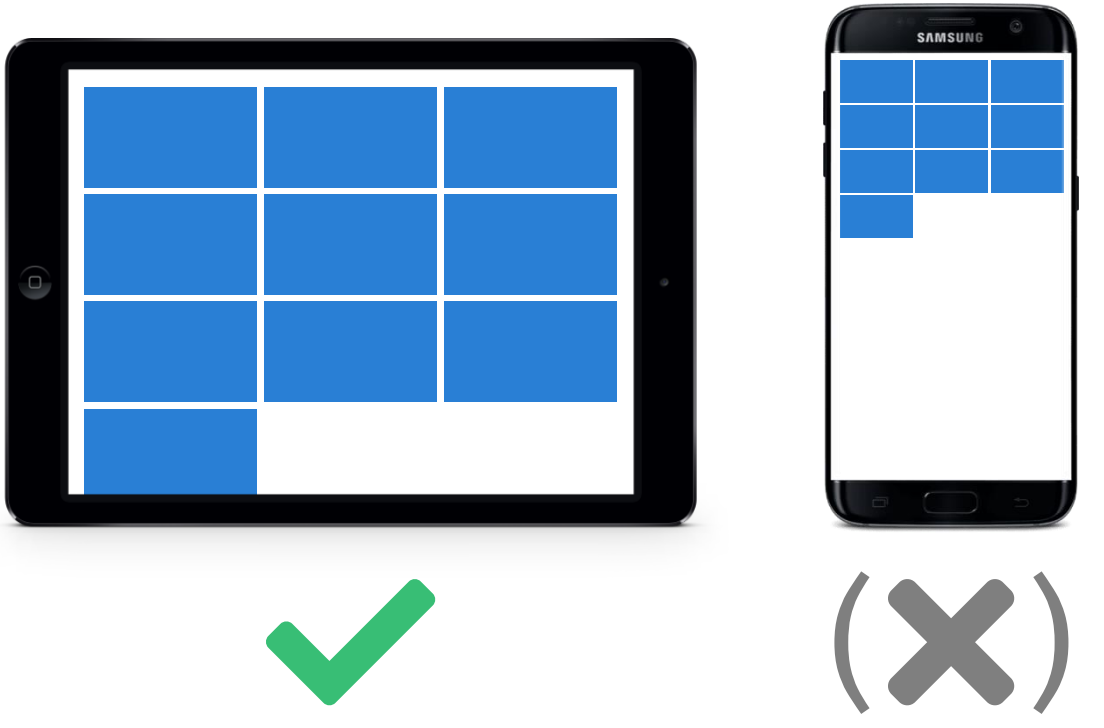


Unflexibel, aber mit *relativer Grösse*: Spezielle Einheit `fr` (*fraction*)

```
.gallery {
  display: grid;
  grid-template-columns:
    1fr 1fr 1fr;
}
```

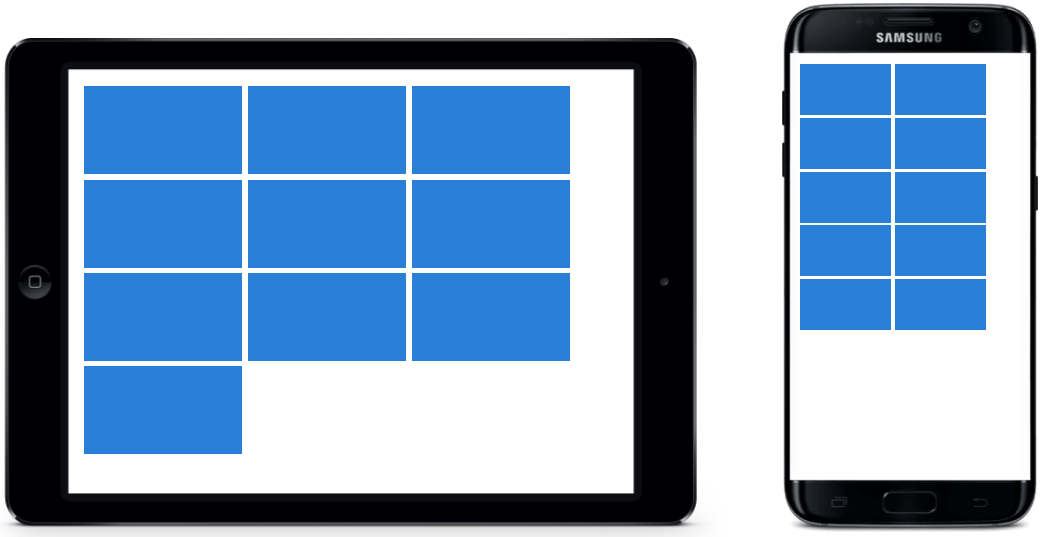
oder (gleichwertig):

```
.gallery {
  display: grid;
  grid-template-columns:
    repeat(3, 1fr);
}
```



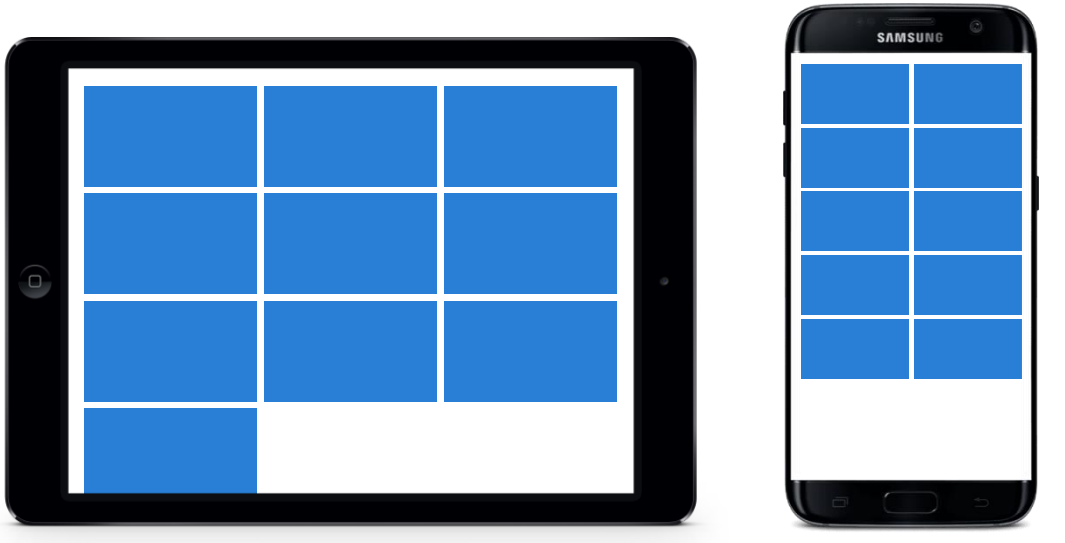
Flexibel: auto-fit

```
display: grid;
grid-template-columns:
  repeat(auto-fit, 150px);
```



Flexibel & ***relative Grösse***:

```
display: grid;
grid-template-columns:
  repeat(auto-fit,
         minmax(150px, 1fr));
```



Media Queries

Noch mächtiger: Media Queries. Erlauben es, beliebige CSS-Regeln *unter bestimmten Bedingungen* anzuwenden

Medientyp *Medien-Feature*

```
@media screen and (min-width: 480px) {
    body {
        margin: 30px;
    }
}
```

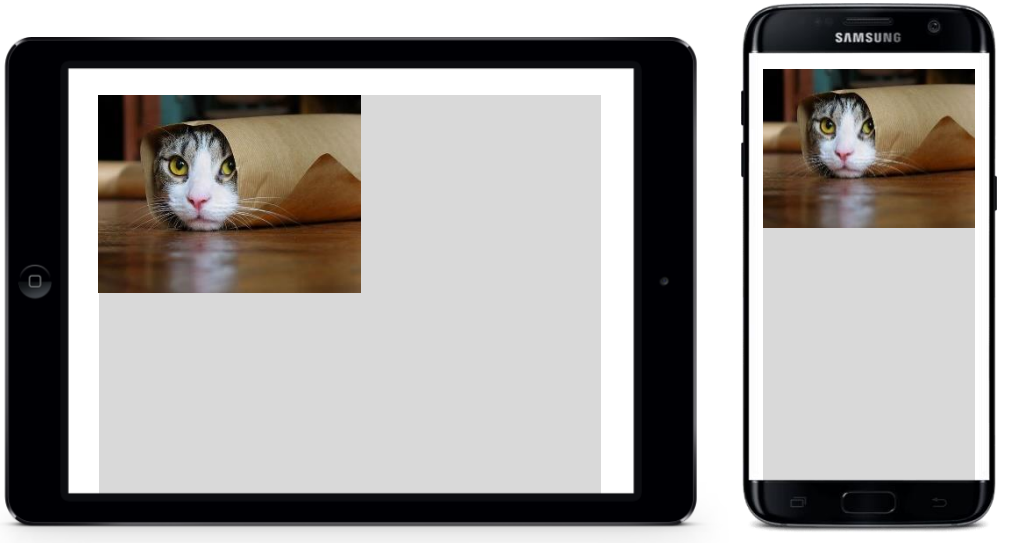
Medien-Feature-Ausdruck

Media Queries: Beispiel

Unterschiedliche Regeln, je nach Viewport-Grösse:

```
#cat-img {
    width: 100%;
}

@media (min-width: 640px) {
    #cat-img {
        width: 50%;
    }
}
```



Media-Queries & Priorität/Spezifizität

Rückblick:

Priorität	Merkmal	Beschreibung
1	Wichtigkeit	Mit **`!important`** kann alles überschrieben werden *Sollte aber möglichst nie bzw. höchstens als Notlösung verwendet werden!*
2	Inline	Inline-Regeln überschreiben andere Regeln
3	Selektoren-Spezifität	Verschiedene Arten von Selektoren haben unterschiedliche Priorität
4	Reihenfolge	Spätere Sheets / Regeln überschreiben frühere
5	Vererbung	*Gewisse* undefinierte Eigenschaften werden von Eltern-Elementen geerbt

Media Queries haben keinen Einfluss auf Priorität!

Medientypen und -Features

Medientyp	Beschreibung
`screen`	Anzeige auf einem Bildschirm
`print`	gedruckte Medien (und Druckvorschau)
`speech`	Sprach-Synthesizer
`all`	passend für alle Geräte

Medien-Feature	Beschreibung
`width`	Breite des Viewports (Fenster/Gerät), z.B. in `px`
`height`	Höhe des Viewports
`resolution`	Pixeldichte des Anzeigegeräts, in `dpi`
`aspect-ratio`	Seitenverhältnis des Viewports, Zahl oder Verhältnis
`...`	

alle mit `min-`, `max-` oder ohne

`min-width` oder `max-width`?

`min-width`: Default-Regeln für kleine Geräte, Spezialregeln für grosse

`max-width`: Default-Regeln für grosse Geräte, Spezialregeln für kleine

Theoretisch gleichwertig, aber…

Zuerst für Desktop-Geräte designen und für mobile Geräte anpassen, oder umgekehrt?

Kommt auf Web-App an, aber häufig:

Mobile First

Print Layout

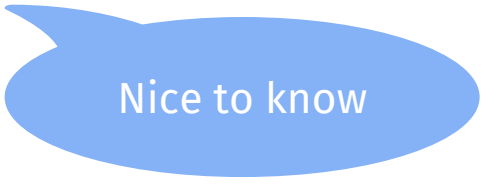
Beim Druck-Layout hilft es oft, gewisse Elemente ausblenden (z.B. Menus und Gestaltungselemente).

Dafür kann z.B. eine Hilfsklasse definiert werden, die dann auf den Elementen gesetzt wird, welche nicht gedruckt werden sollen.

```
@media print {
    .hide-print {
        display: none;
    }
}
```

```
<div class="hide-print">
    <img src="frame-top.png" alt="decorative">
</div>
```

Container Queries



- Relativ neues Feature von CSS
- Ähnlich wie Media Queries, jedoch relative zu einem anderen Element (statt zum Bildschirm / Viewport / Gerät)
- https://www.joshwcomeau.com/css/container-queries-introduction/
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_containment/Container_queries

Übung 2: Einfache Media Query

Der Text der App ist allgemein zu gross für kleine Geräte.
Implementiere mittels einer Media Query folgende Regeln:

- Für «kleine» Geräte wird als Grundschriftgrösse 100% verwendet.
- Für «grosse» Geräte wird hingegen 120% verwendet (wie bisher).

Entscheide, ab welcher Breite ein Gerät als «gross» gilt und wähle zwischen einer `min-width`- und einer `max-width`-Regel.

Fragen?