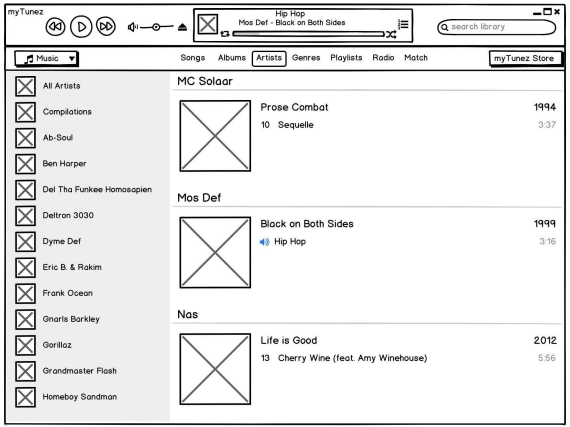
Oberflächendesign und UX

User Experience (UX)

- Beschreibt die Aspekte einer Software die zu der Anwendererfahrung beitragen
  - Visuelles Design - Wie sieht es aus
  - Informationsarchitektur - Wie komme ich an Inhalte
  - Interaktionsdesign - Wie ist die Nutzung für mich
  - Usability - Wie einfach ist die Nutzung (z.B. auch Barrierefreiheit)
- Das Web hat sehr breite Anwenderbasis
  - Daher: Optimiere gegen Standards und kenne deine Zielgruppe
  - Intuition vor Konvention vor Dokumentation
- Sehr breite und wichtige Disziplin(en)
- Webentwickler mindestens Barrierefreiheit beachten
  - Z.B. für Screenreader, Farbenblindheit, alternative Navigation

Wireframing

- Design Ansatz um sich über Aufbau und Struktur der Oberfläche klar zu werden
- Bietet Gesprächsgrundlage und kreativen Aufhängepunkt
- Sollte immer **vor** der Entwicklung der Oberfläche stehen
  - Fokus auf grober Struktur und Usability
  - Keine Details! (Farben, Schriftart, Bilder, …)
- Tools (z.B. Balsamiq) erlauben regelmäßige Pflege
- Helfen Features zu planen und zu erkennen
  - Z.B. Fehlende Funktionalität in der Oberfläche



Visuelles Design

- Gestaltung der visuell sichtbaren Eigenschaften der Oberfläche
- Sollte eine durchgängige Erkennbarkeit ermöglichen
  - Einheitliche Bildsprache
  - Durchgängige Farbgebung
- Muss oft CI/CD angepasst sein
- Faustregel: Weniger ist oft mehr
  - Web tendiert zu schlichten, flachen Designs
  - Wenige Farben (1-2 Hauptfarben, ein paar Akzente und Schluss)
- Sollte zum physikalischen Design passen
  - Z.B. Printmedien, Produktfarben, ...

Visuelles Design - Standards schaffen

- Durchgängigkeit eines Designs ist oberste Priorität
- CSS erlaubt grundlegende Definitionen an zentraler Stelle
  - Vererbung von Eigenschaften
  - Nutzung durch HTML Elemente
  - Beispiel: zentrale CSS Eigenschaften für
    - Farbelemente (Primärfarbe, Sekundärfarbe, Highlight 1, …)
    - Überschriften (*h1, h2, h3,* …)
    - Text, Links, …
    - Elemente, Borders, ...
    - Buttons, Infoboxes, ...
    - Abstände, Größenverhältnisse, …
- Einmal zentral geschaffen, nur sehr bewusst verändert

CSS

- Formatierungssprache des Webs
- Selektiert DOM Elemente über Selektor (-Liste)
- Verändert Elementeigenschaften über Anweisung(en)
  - Beinhalten neue Eigenschaften
- Bietet Vererbungs- und Überschreibungsmechanismen

```
Selektor1 [, Selektor2 [, …] ] {
    Eigenschaft-1: Wert-1;
    …
    Eigenschaft-n: Wert-n[;]
}
/* Kommentar */
/* In eckigen Klammern stehen optionale Angaben */
```

CSS - Beispiel

```css
p.info {
    font-family: arial, sans-serif;
    line-height: 150%;
    margin-left: 2em;
    padding: 1em;
    border: 3px solid red;
    background-color: #f89;
    display: inline-block;
}
p.info span {
    font-weight: bold;
}
p.info span::after {
    content: ": ";
}
```

```html
<p class="info">
    <span>Hinweis</span>
    Sie haben sich erfolgreich angemeldet.
</p>
```

Hinweis: Sie haben sich erfolgreich angemeldet.

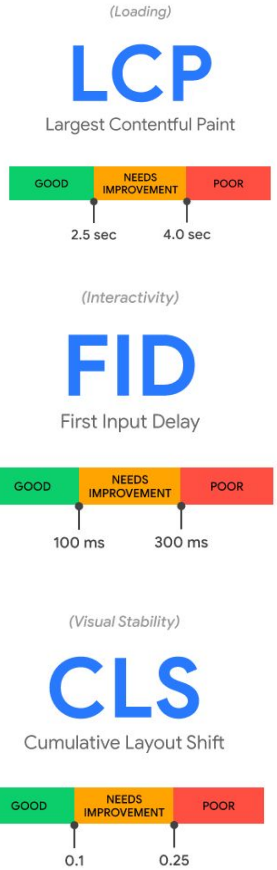
Ranking Auswirkungen

- Suchmaschinen (und Nutzer) strafen schlechte Entscheidungen und falsche Ansätze
- Viele verschiedene Metriken für verschiedene Anwendungsfälle
  - Performancemetriken
    - Ladezeiten, Responsiveness (Reaktionszeiten)
  - Compliance/Compatibility
    - Responsiveness (Darstellung)
    - Vergleiche caniuse.com
  - UX Metriken
    - Google Web Vitals
- Verschiedene Tools zur Optimierung von Webseiten frei verfügbar
- Ziel: Verbesserung des Webs
  - Aber auch Mitsprache

Ranking Auswirkungen - Core Web Vitals

- Largest Contentful Paint
  - Wie lange dauert es den größten (sinnstiftenden) Block einer Webseite zu laden?
- First Input Delay
  - Zeit bis eine erste Interaktion mit der Seite möglich ist
    - Z.B. JS Funktionen verfügbar sind
- Cumulative Layout Shift
  - Wie stabil verhält sich die Darstellung einer Seite?
  - https://web.dev/cls/

- https://developers.google.com/speed/pagespeed/insights/?url=th-rosenheim.de



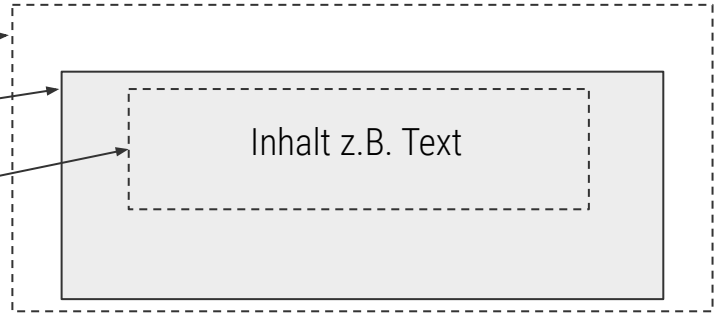
CSS Boxmodell

- Modell einer Gedachten Box die **jedes** HTML Element umgibt
- Kann, muss aber nicht für styling genutzt werden
- Kann sehr frei gestaltet werden

Margin - Außenabstand zu anderen Elementen

Border - Begrenzende Außenkante

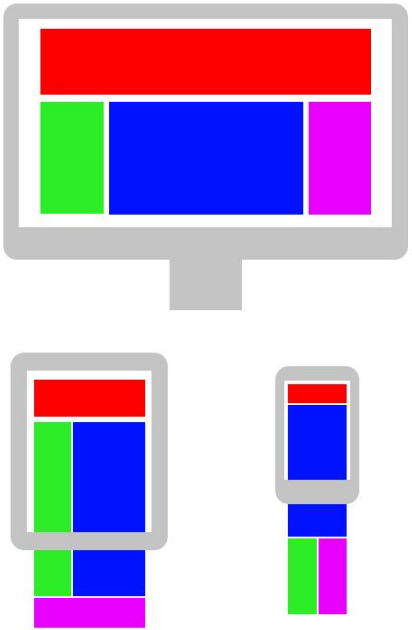
Padding - Abstand den Inhalts zur Border




```
div {
    width: 300px;
    border: 2px solid black;
    padding: 50px 20px 5px 20px;
    margin: 7px 35px 60px 35px;
}
```

Responsive Web Design

- Soll Webanwendungen für alle Geräte zugänglich machen
- Vielzahl an Geräten nötigen zu adaptivem Design
- Bezieht sich vor allem auf:
  - Die Ausdehnung der Elemente auf einer Seite
  - Die Zusammenstellung der Elemente
  - Manche Inhalte z.B.
    - Unterschiedliche Bilder je Gerät ausliefern
- **Ist ein Must-Have**
  - Entwickler muss Geräte seiner Zielgruppe kennen
  - Entsprechendes Testing ist wichtig (auch Regression)



Viewport

- Allgemein: Für den Nutzer sichtbarer Bereich der Anwendung
  - Stark abhängig vom Endgerät
  - Vergleiche auch *Above the Fold*
- Technisch: Meta Tag zur Inhalt-Skalierung
  - Beschreibt Optionen für Browser
- Zwingend notwendig im heutigen Web!

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- ... -->
```

Media Queries

- Erlaubt Formatierung abhängig von
  - Größe des Viewports (*Breakpoints*)
  - Genutzten Client Geräten (Mediatype)
  - Darstellungsformat (quer/hoch)
  - Sog. Media-Features
    - Auflösung des Geräts
    - Seitenverhältnis
    - ...
- Z.B. spezielle Formatierung für Drucker
- Nutzt die *Viewport* Meta Information

```
<!-- ... -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
    body {
        background-color: yellow;
    }

    @media only screen and (max-width: 600px) {
        body {
            background-color: lightblue;
        }
    }
</style>
<!-- ... -->
```

Positionierung

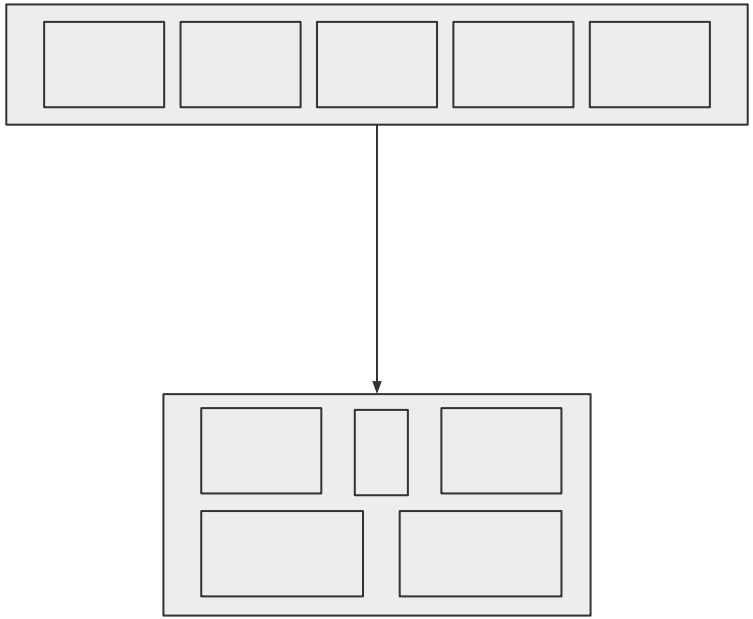
- DAS Kernthema bei Oberflächendesigns
- CSS bietet viele Positionierungsarten
  - Block - Für ganze Bereiche einer Webseite
  - Inline - Für Textformatierung
  - Table - Für tabellarische Daten
  - Position - Zur expliziten Positionierung einzelner Elemente
  - Flexbox - Zur dynamischen Positionierung
  - Grid - Für Rasterlayouts
- Wahl der richtigen Technik ist entscheidend für Responsive Design
  - Kann mituntern schwer zu treffen sein

Positionierung - Flexbox

- Erlaubt dynamische Positionierung von Elementen in Containern
- **Ist eindimensional** (Zeilen- oder Spalten-Layout)
- Fokus auf Responsiveness und Element-Fluss
- Sehr leichtgewichtiger Ansatz
- Besteht aus:
  - Richtung in die Elemente fließen
  - Verhalten bei Umbruch
  - Reihenfolge im Container
  - Wachstums- und Schrumpfverhalten bei Viewport-Änderungen
- Vorteil: erlaubt viel Freiheit bei variabler Viewport Größe

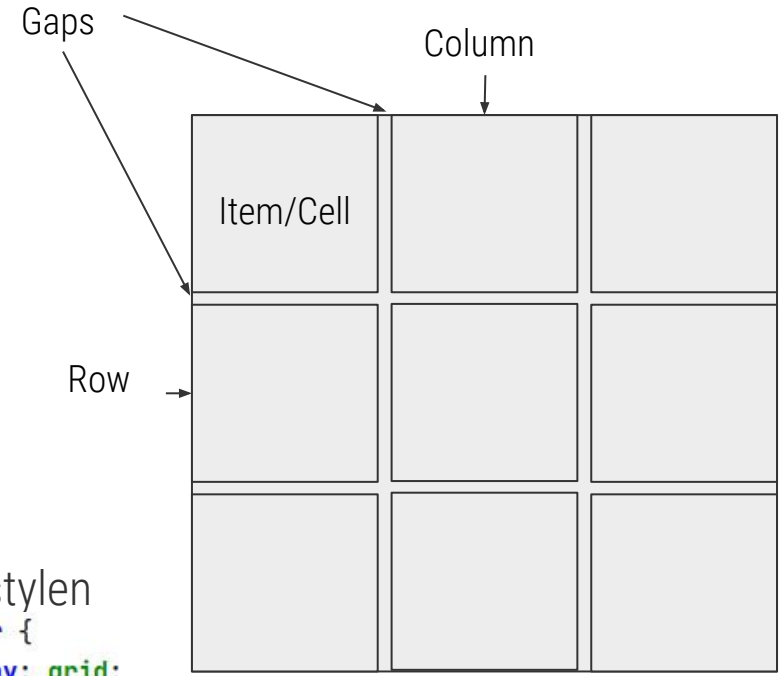
```css
.flex-container {
    display: flex;
    flex-wrap: wrap;
    flex-direction: row;
    align-items: center;
}

.flex-container > div {
    width: 100px;
}
```

Positionierung - Flexbox



Positionierung - Grid

- Unterteilung in ein Gitter-Raster
  - **Ist zweidimensional**
- Braucht DOM Elemente für:
  - Container (enthält das Raster)
  - Items (Unterelemente des Containers)
- Beschreibt Raster aus Spalten und Zeilen
- Initial homogenes Raster
- Skaliert sehr gut mit Viewport
- Einzelne Bestandteile lassen sich individuell stylen
  - Z.B. Breite der Gaps
  - Ausrichtung des Items in der Cell

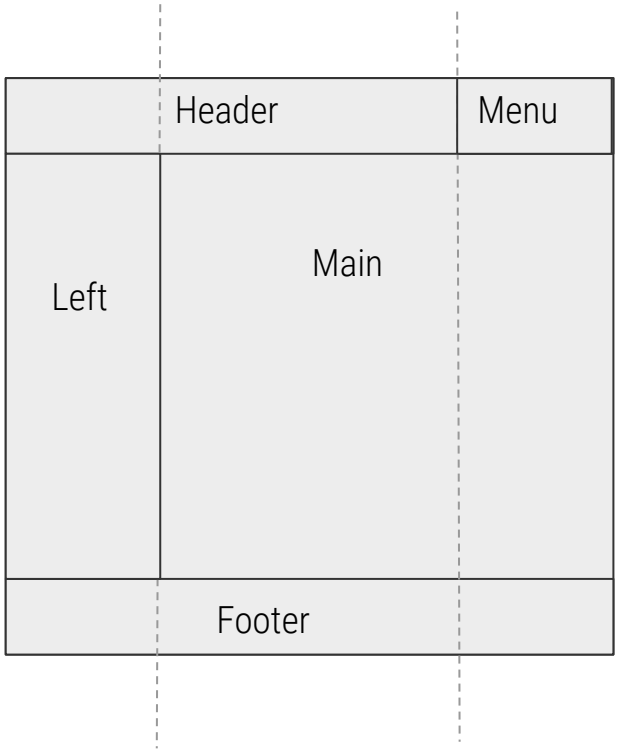



```
.container {
    display: grid;
}
```

Positionierung - Grid

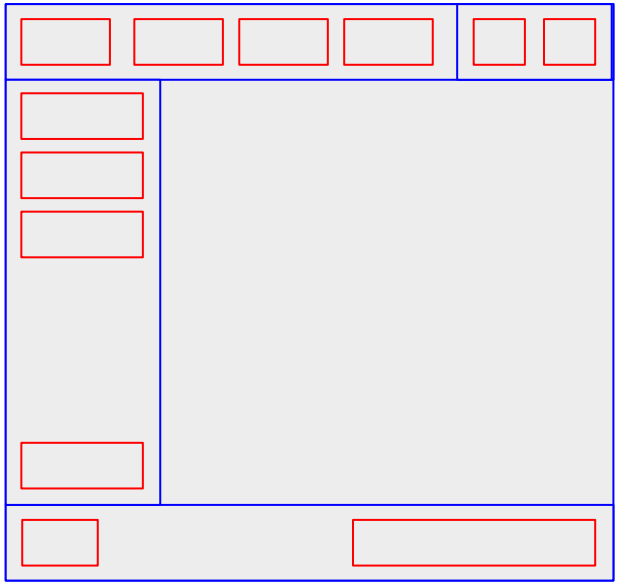
- Ausmaße der Columns und Rows können variieren
- Items können über mehrere Cells gelegt werden
  - Erlaubt sehr freie Layout Verteilung
  - Können zu benannten Areas zusammengefasst werden

```
.item-a {
    grid-area: header;
}
/* ... */

.container {
    display: grid;
    grid-template-columns: 25% 50% 25%;
    grid-template-rows: auto;
    grid-template-areas:
    "header header menu"
    "left main main"
    "footer footer footer";
}
```



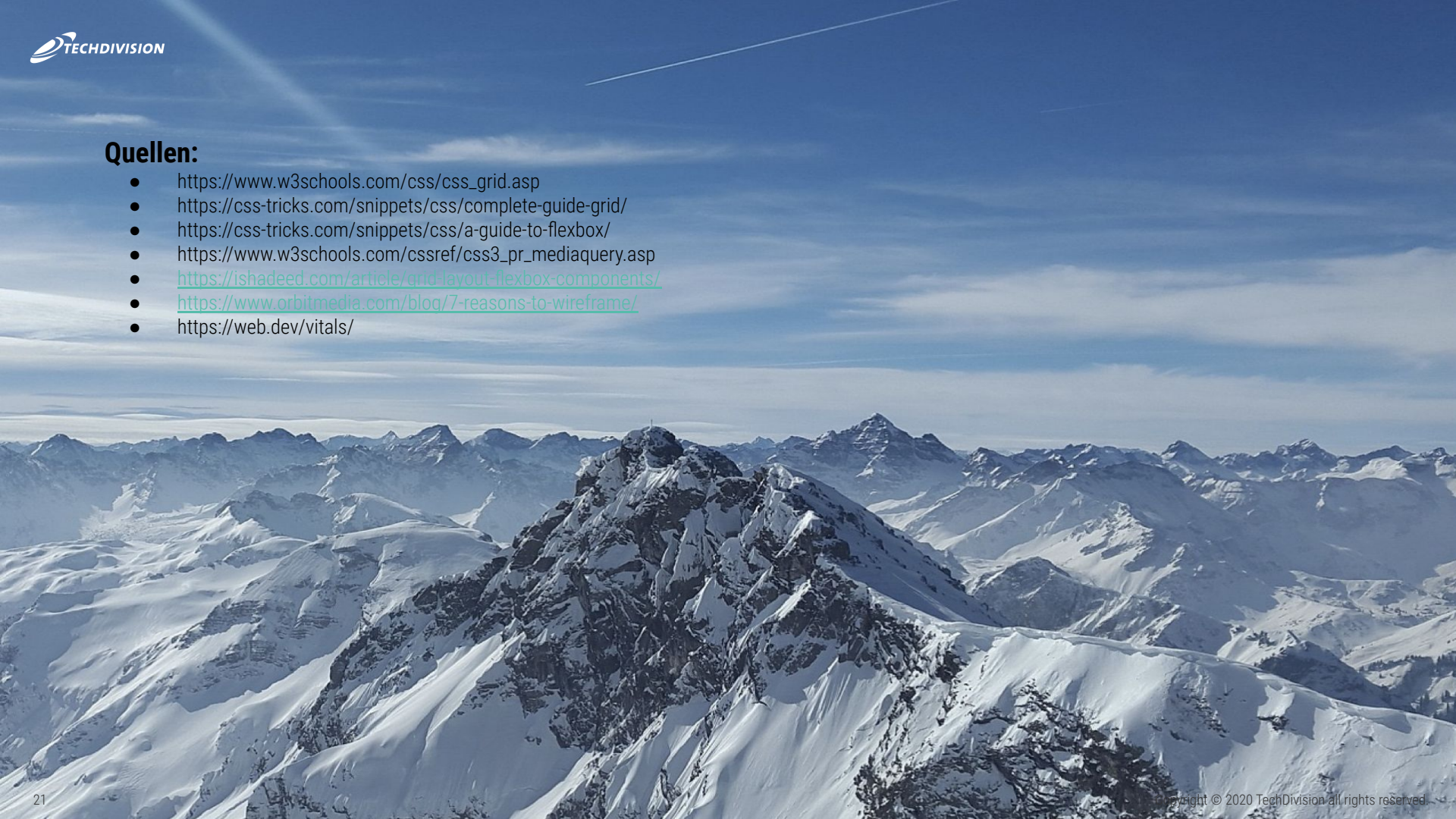
Positionierung - Flexbox vs. Grid

- Nicht “was”, sondern “was wo”!
- Beide Techniken lassen sich gemeinsam nutzen
- Fragestellung:
  - Wieviel Dimensionen hat mein Layout?
  - Geht es um globale oder inline Positionierung?
- Faustregel:
  - Grid für Groblayout
  - Flexbox für einzelne Elemente
- **Die Kombination macht die Musik**



Flexbox

Grid



Quellen:

- https://www.w3schools.com/css/css_grid.asp
- https://css-tricks.com/snippets/css/complete-guide-grid/
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- https://www.w3schools.com/cssref/css3_pr_mediaquery.asp
- https://ishadeed.com/article/grid-layout-flexbox-components/
- https://www.orbitmedia.com/blog/7-reasons-to-wireframe/
- https://web.dev/vitals/

Bildquellen:

- myTunez by Balsamiq Team - https://balsamiq.com/wireframes/
- Responsive Web Design By Tomáš Procházka - Autor, CC0, https://commons.wikimedia.org/w/index.php?curid=19294633
- Viewport Example by w3schools - https://www.w3schools.com/css/css_rwd_viewport.asp