CSS und Tailwind

Prof. Dr.-Ing. Thorsten Weiss

Version 1.0

Plain CSS

CLOUD COMPUTING

Heutiges CSS

- Styling mit CSS kann anstrengend sein. Im Normalfall müssen CSS-Klassen geschrieben und HTML Dateien zugeordnet werden.
- Die Eigenschaft style im html-Tag überschreibt (meistens) die Eigenschaften der Klasse. Dadurch kommt es häufig zu unaufgeräumten Projekten, wenn mehrere CSS Klassen importiert werden und zudem mit style einzelne Komponenten modifiziert werden. Wir erleben hier Webdesigns die nicht konsistent sind.
- Wordpress, Joomla und Typo-3 haben aufgrund ihrer Plugin-Architektur und dem Import verschiedener CSS-Dateien hier häufig Probleme. Für ein perfektes Design muss hier tief eingegriffen werden.
- Um dieses Problem zu lösen gibt es unzählige bereits fertige CSS-Dateien, die das Design von UI-Elementen definieren. Beispiele: Bootstrap (eines der ersten gut verwendbaren CSS), Material (für Material Design), u.v.m. Diese sind komponentenbasiert.
- Es wurde aber etwas sperrig, wenn man etwas ändern oder hinzufügen wollte. Wenn die Hierarchie der CSS-Struktur genau eingehalten wurde, konnten hiermit aber gute Ergebnisse erzielt werden.
- Ein eigenes CSS zu schreiben z.B. für eine CI (Corporate Identity) ist somit recht aufwendig und teuer.
- Hier ein Beispiel mit plain-CSS:

index.html

```html
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,
          initial-scale=1.0" />
  <title>CSS Demo</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <section class="hero">
    <div class="hero-text">
      <h1>CSS Demo</h1>
      <p style="padding-left: 10px;">
            CSS Klassen werden werden in HTML
            integriert und definieren das Design
      </p>
    </div>
    <div class="hero-image">
      <img src="cssmaster.png" alt="Hero Bild" />
    </div>
  </section>
</body>
</html>
```

styles.css

```css
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: Arial, sans-serif;
  background-color: #f5f5f5;
  color: #333;
}

.hero {
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: space-between;
  padding: 60px 80px;
  background: linear-gradient(to right, #e0eafc, #cfdef3);
  height: 100vh;
}

.hero-text {
  flex: 1;
  padding-right: 40px;
}

.hero-text h1 {
  font-size: 3rem;
  margin-bottom: 20px;
  color: #1a1a1a;
}

.hero-text p {
  font-size: 1.2rem;
  line-height: 1.6;
  color: #444;
}

.hero-image {
  flex: 1;
  display: flex;
  justify-content: center;
  align-items: center;
}

.hero-image img {
  max-width: 100%;
  height: auto;
  border-radius: 10px;
}

@media (max-width: 768px) {
  .hero {
    flex-direction: column;
    text-align: center;
    padding: 40px 20px;
  }

  .hero-text {
    padding-right: 0;
    margin-bottom: 30px;
  }

  .hero-image img {
    width: 100%;
    height: auto;
  }
}
```

Tailwind – Klassensystem in Tailwind

CLOUD COMPUTING

Anmerkung: rem vs. px

- rem ist eine relative Maßeinheit in CSS (relativ zur HTML Fontsize, wichtig für Barrierefreiheit und Zoom).
- px ist die absolute Maßeinheit in CSS:
  - 1 rem entspricht der Schriftgröße des html-Elements.
  - Standardmäßig ist das im Browser:
  - 1 rem = 16 px
    0.5rem = 8px
    2rem = 32px

Struktur der Tailwind-Klassen

- Tailwind verwendet vordefinierte, funktionale Klassen, die einem gewissen Schema entsprechen

  <category>-<property>-<value>

- Beispiele:

Klasse	Bedeutung
bg-blue-500	Hintergrund, blau, Stufe 500
text-sm	Test, Schriftgröße klein (small)
p-4	Padding auf allen Seiten 1 rem
px-4	Padding auf x (links rechts) mit 0.5rem
items-center	Elemente, mittig platzieren

Responsive Design: Mobile-First Ansatz

- Wenn nichts weiter angegeben wird, wird vom kleinsten Bildschirm (Smartphone) ausgegangen.
- Die Eigenschaften können für die verschiedenen Größen Prefixes festgelegt werden:
- Dies sind die Klassen:

Tailwind Prefix	min-width in px	Typische Geräte
sm:	640px	kleine Tablets
md:	768px	Tablets quer, kleine Laptops
lg:	1024px	große Laptops, Desktop
xl:	1280px	größere Desktops
2xl:	1536px	Ultra-Wide-Screens

Tailwind – Responsive

CLOUD COMPUTING

Responsive Design: Mobile-First Ansatz (2)

- Beispiel: Ein Text wird mit jeder Größen-Stufe größer.
- Dies ist so zu lesen: Bei kleinen Bildschirmen gilt text-sm – wenn keine weiteren Eigenschaften angegeben werden, bleibt die Schrift auf Schriftgröße text-sm:

```
<p class="text-sm">
  Der Text bleibt auf größeren Bildschirmen gleich groß!
</p>
```

- Wenn aber das md: und lg: Tag angegeben wird, dann wird die Schrift auf größeren Bildschirmen größer. Sprich, wenn die Bildschirmgröße md, also 768px übersteigt, dann gilt nicht mehr text-sm, sondern text-lg. Wenn die Bildschirmgröße md, also 1024px übersteigt, dann wird die Schriftgröße text-xl gewählt. Wenn die Bildschirmgröße auch die xl (>1280px) übersteigt, dann bleibt die Schriftgröße text-xl. Es wird also „vom kleinen zum großen Bildschirm gedacht“

```
<p class="text-sm md:text-lg lg:text-xl">
  Ich werde auf größeren Bildschirmen größer!
</p>
```

Zustände

- Bedienelemente können ihre Eigenschaften auch nach ihrem Zustand ändern. Wichtige Zustände sind:

Zustand	Beschreibung	Beispiel
hover:	Wenn die Maus über dem Element ist	hover:bg-blue-600
focus:	Wenn das Element fokussiert ist (z. B. per Tab)	focus:outline-none
active:	Während das Element geklickt wird	active:scale-95
visited:	Für bereits besuchte Links	visited:text-purple-600
focus-visible:	Nur sichtbarer Fokus (z. B. Tastatur-Navigation)	focus-visible:ring-2

- Für übergeordnete Group-Elemente. Dazu wird ein um jene Elemente eingebaut, die gruppiert werden sollen.

Zustand	Beschreibung	Beispiel
group-hover:	Wenn über das übergeordnete Element gehovert wird	.group:hover .child = group-hover:bg-blue-500
group-focus:	Wenn das übergeordnete Element fokussiert ist	group-focus:ring-2
group-aria-expanded:	Wenn aria-expanded="true" gesetzt ist	group-aria-expanded:bg-red-500

Tailwind – Zustände / Arias

CLOUD COMPUTING

Zustände bei Formularfeldern

- Viele SSR Seiten verwenden Formulare (hier werden Daten zusammengefasst und an den Server geschickt).
- Mit Tailwind kann hier eine gute Nutzerführung erreicht werden ohne JS zu laden:

Zustand	Beschreibung	Beispiel
disabled:	Wenn ein Formularfeld deaktiviert ist	disabled:opacity-50
checked:	Wenn ein Checkbox/Radio aktiviert ist	checked:bg-green-500
required:	Wenn ein Feld „required" ist	required:border-red-500
invalid:	Wenn ein Feld einen ungültigen Wert hat	invalid:border-red-600
placeholder-shown:	Wenn nur der Placeholder sichtbar ist	placeholder-shown:text-gray-400

Höhe/Breite

- Die Breite und die Höhe von Objekten kann mit h-x und w-x eingestellt werden.

Arias

- Aria erlaubt es, Zustände von ganzen Seitenteilen ohne JS zu variieren
- Arias sind praktisch bei SSR-Seiten, die kein JS in den Browser laden sollen und trotzdem eine gewissen Reaktivität nutzen sollen.
- Eine intensive Beschäftigung mit Arias ist erforderlich.
- Beispiel: Akkordeon: siehe Projekt /pages/twaria.astro

Light/Darkmode

- Für diese beiden Modi können die Farben global definiert werden in der tailwind config oder mit daisy UI mit den fertigen Farbschemata. Für einen einheitlichen Look wäre das zu empfehlen.
- Zusätzlich kann aber explizit noch mit dem Prefix dark: und light: das Design modifiziert werden.

Tailwind - Spacing

CLOUD COMPUTING

Spacing: Padding, Margin, Gap

- Tailwind baut auf die Relative Größen in rem auf. Dadurch werden auch Abstände proportional größer zur Schriftgröße (Zoom im Browser).

Padding (Innenabstand):

- Hier kann unterschieden werden, wo das Padding angewandt wird. Im Beispiel immer 3 als Abstand:
  - **p-3:** Padding auf alle Seiten anwenden
  - **px-3:** Padding nur horizontal anwenden (left + right)
  - **py-3:** Padding nur vertikal anwenden (top + bottom)
  - **pt-3, pr-3, pb-3, pl-3:** Padding für die einzelne Seiten (top, right, bottom, left)

Margin (Außenabstand):

- Hier kann unterschieden werden, wo das Padding angewandt wird. Im Beispiel immer 3 als Abstand:
  - **m-0:** Kein Außenabstand
  - **mx-auto:** Horizontal zentriert
  - **mx-3, my-3, mt-3, mr-3, mb-3, ml-3:** wie bei Padding

Gap (Zwischenraum):

- Erzeugt einen Zwischenraum zwischen Childelemente (wichtig bei Grids, Flexbox etc.)
  - **gap-3:** Abstand von 3 in alle Richtungen.
  - **gap-x-3:** Horizontal Abstand von 3 (links-rechts)
  - **gap-y-3:** Vertikaler Abstand von 3 (oben-unten)
  - **mx, my, mt, mr, mb, ml:** wie bei Padding

```
<div class="grid grid-cols-2 gap-4">
  <div>1</div>
  <div>2</div>
</div>
```

Space (Gestapelte Elemente):

- Funktioniert ohne Grid/Flexbox, einfach Abstand zwischen Elementen untereinander
  - **space-y-3:** Abstand von 3 in horizontaler Richtungen.
  - **space-x-4:** Abstand von 2 zwischen 2 Elementen vertikal

```
<div class="flex space-x-4">
  <button class="bg-blue-500 text-white px-4 py-2 rounded">OK</button>
  <button class="bg-gray-300 px-4 py-2 rounded">Abbrechen</button>
</div>
```

Tailwind - Border

CLOUD COMPUTING

Borders:

- Soll ein Rahmen um ein Element gezeichnet werden, werden border-Klassen genutzt.

Kategorie	Beschreibung	Beispielklassen
Rahmen rendern	Rahmen auf allen oder einzelnen Seiten	border, border-0, border-t, border-b, border-l, border-r
Rahmenbreite	Dicke des Rahmens (in px)	border, border-2, border-4, border-8, border-t-4
Rahmenfarbe	Farbe des Rahmens (nach Farbsystem)	border-gray-300, border-red-500, hover:border-blue-500
Rahmenstil	Linienart des Rahmens	border-solid, border-dashed, border-dotted
Abgerundete Ecken	Rundung für alle oder gezielte Seiten	rounded, rounded-sm, rounded-md, rounded-lg, rounded-full, rounded-t-lg, rounded-b-none

- Durch die Kombination der Klassen kann das gewünschte Ergebnis erreicht werden.

```
<div class="border-2 border-blue-500 rounded-lg border-dashed p-4">
  Kasten Dicke 2 mit blauem, gestricheltem, gerundetem Rahmen
</div>
```

Custom Values:

- Tailwind erlaubt auch Custom Values in eckigen Klammern, falls die gewünschte Klasse nicht definiert ist: Beispiel class="gap-[10vw]"

Tailwind - Schrift

CLOUD COMPUTING

Schriftart, größe und stärke

- Für die Einstellung von Schriftarten muss diese in **tailwind.config.js** eingebunden werden:

```
module.exports = {
    content: ['./src/**/*.{astro,html,js,jsx,md,mdx,svelte,ts,tsx,vue}'],
    theme: {
        // zusätzliche Schriften
        extend: {
            sans: ['Open Sans', 'sans-serif'],
            serif: ['Merriweather', 'serif'],
            mono: ['Fira Code', 'monospace']
        }
    },
...
```

Name in den Klassen

Schriften einbinden

- Beispiel: Astro fontdemo.astro

```
<html>
  <head>
    <title>Font Demo</title>
  </head>
  <body class="container mx-auto p-4">
    <h1 class="font-sans text-4xl mb-4">Open Sans</h1>
    <p class="font-sans text-xl">This is Open Sans font.</p>

    <h1 class="font-serif text-4xl mb-4 mt-8">Merriweather</h1>
    <p class="font-serif text-xl">This is Merriweather font.</p>

    <h1 class="font-mono text-4xl mb-4 mt-8">Fira Code</h1>
    <p class="font-mono text-xl">This is Fira Code font.</p>
  </body>
</html>
```

- Zuvor müssen Schriften geladen werden. Aufgrund von DSGVO Vorgaben dürfen Schriften nicht mehr in productive Systemen von Google geladen werden. Die folgende Einbindung ist also nicht mehr zulässig:

```
<link href="https://fonts.googleapis.com/css2?family=Open+Sans&family=Merriweather&family=Fira+Code&display=swap" rel="stylesheet">
```

- Alternativ: Fonts herunterladen und in Ornder /public/fonts ablegen.
- Die Schrift kann in Styles styles/global.css eingebunden werden. Dann wird die Schrift nicht mehr von Google geladen.

```
@font-face {
  font-family: 'Open Sans';
  src: url('/fonts/OpenSans-Regular.woff2') format('woff2');
  font-weight: 400;
  font-style: normal;
}
```

Tailwind – Schriften

CLOUD COMPUTING

Wichtige Schriftklassen

Kategorie	Beispiele
Größe	text-sm, text-base, text-lg, text-xl, text-2xl bis text-9xl text-base ist die Standard-Größe (16px bei 100% Zoom) Individuell mit Parameter: Beispiel: text-[22px]
Familie	font-sans, font-serif
Gewicht	font-thin, font-extralight, font-light, font-normal, font-medium, font-semibold font-bold, font-extrabold, font-black
Stil & Format	italic, not-italic, uppercase, lowercase, capitalize, normal-case
Zeilenhöhe	leading-none, leading-tight, leading-snug, leading-normal, leading-relaxed, leading-loose
Buchstabenabstand	tracking-tighter, tracking-tight, tracking-normal, tracking-wide, tracking-wider, tracking-widest

Farbsystem

- Das Tailwind folgt folgender Struktur:

  <property>-<color>-<shade>

- Es kann auf verschiedene Elemente angewandt werden. Beispiele:

```
bg-blue-500, Hintergrundfarbe: mittelblau
text-gray-700, Textfarbe: dunkles Grau
border-red-300, Rahmenfarbe: hellrot
```

- Folgende Farben sind vordefiniert:

  gray, red, orange, yellow, green, teal, blue, indigo, purple, pink, slate, zinc, neutral, stone, black, white, transparent, current

- Folgende **Helligkeiten** sind definiert (50-950)

Farbklasse	Beschreibung
*-50	Sehr hell (fast weiß)
*-100 bis *-300	Hell
*-400 bis *-600	Mittel (häufig benutzt)
*-700 bis *-900	Dunkel
*-950	Sehr dunkel (fast schwarz)

- Dark-Mode mit dark:

```
<div class="bg-white text-black dark:bg-gray-900 dark:text-white">
  Inhalt
</div>
```

Tailwind – Container

CLOUD COMPUTING

Container

- Container erzeugt einen Bereich, der eine maximale Breite hat und automatisch zentriert ist.

```
<div class="container mx-auto px-4">
  Inhalt mit Rand und automatischer Zentrierung
</div>
```

- Es wird häufig ein Container um die ganze Seite herum gebaut, damit die Inhalte nie breiter als der Container wird (Beispiel: Artikel soll zur besseren Lesbarkeit nicht zu breit werden).
- Der Container wächst, wenn der Screen wächst.
- Wichtige Container Klassen.

Klasse	Zweck
mx-auto	Zentriert den Container horizontal, wenn dies nicht gesetzt wird, ist der Container linksbündig.
px-4	Innenabstand links/rechts (sicherer Rand)
pt-8, pb-12	vertikaler Abstand oben/unten
max-w-*	Begrenzung der Breite manuell überschreiben
w-full	Optional: volle Breite innerhalb Eltern

- Vermutlich selten gebraucht, daher nur zur Info:
- Die Standard-Breakpoints (die im Abschnitt Responsive gezeigt wurden), können für einen Container überschrieben werden in tailwind.config.js. Beispiel:

```
module.exports = {
  theme: {
    container: {
      center: true,
      padding: '1rem',
      screens: {
        sm: '600px',
        md: '728px',
        lg: '984px',
        xl: '1240px',
        '2xl': '1496px',
      },
    },
  },
}
```

Tailwind – Flexbox, Grid

CLOUD COMPUTING

Gridbox vs. FlexBox

- Um Elemente (auch Responsive) anordnen zu können, werden in Tailwind Grid und Flexbox genutzt.
- In Tutorials und Code-Generierung werden diese beiden häufig durcheinander gebracht oder ungeschickt eingesetzt. Daher hier eine Gegenüberstellung:

Kriterium	Flexbox	Grid
Layout-Ausrichtung	**1-dimensionale Anordnung (entweder horizontal oder vertikal)**	**2-dimensionale Anordnung (Spalten und Zeilen gleichzeitig)**
Beispiele	Navigationsleisten, Buttons in einer Reihe, Cards nebeneinander	Gitterartige Anordnung wie Dashboards oder Card-Gruppen.
Inhalt steuert Layout	Ja, sprich die Inhalte bestimmen Breite/Höhe	Nein, denn die Rasterstruktur ist fix.
Reihen/Spalten kombinieren	Nicht direkt	Ja – ideal für komplexe Designs
Reihenfolge steuerbar	order-* Klassen	auch, aber weniger intuitiv (Eher wie eine Tabelle denken)
Ausrichtung von Elementen	Sehr einfach (justify-*, items-*)	Möglich, aber etwas Einarbeitung nötig.
Nutzung	• Elemente linear angeorndet werden sollen • Eine horizontale oder vertikale Liste / Leiste erstellt werden soll. • Inhalte zentriert werden sollen willst • Für einfache, responsive Layouts. • Gleichmäßige Abstände eingeführt werden sollen (space-x-*, gap-*)	• Inhalte in einem Raster angeordnet werden sollen. • mehrere Zeilen und Spalten kontrollieren werden sollen • Gleichmäßige Spalten und/oder Zeilenanordnungen gebraucht werden. • Layout unabhängig vom Inhalt strukturieren willst
Kombination	Grid und Flexbox können kombiniert und ineinander verschachtelt werden.	

FlexBox

- Aktivieren der FlexBos mit ...
- Ausrichtung entlang der Hauptachse (horizontal):

Klasse	Wirkung
justify-start	Elemente am Anfang
justify-center	Zentriert
justify-end	Am Ende
justify-between	Platz zwischen den Elementen
justify-around	Gleichmäßig verteilt mit Außenabstand
justify-evenly	Gleichmäßig ohne Außenabstand

Tailwind – Flexbox, Container

CLOUD COMPUTING

FlexBox

- Ausrichtung entlang der Querachse (vertikal):

Klasse	Wirkung
items-start	Oben ausrichten
items-center	Vertikal zentrieren
items-end	Unten ausrichten
items-stretch	Auf volle Höhe dehnen (Standard)

```
<div class="flex items-center justify-between">
  <span>Links</span>
  <span>Rechts</span>
</div>
```

Container

- Container erzeugt einen Bereich, der eine maximale Breite hat und automatisch zentriert ist.

```
<div class="container mx-auto px-4">
  Inhalt mit Rand und automatischer Zentrierung
</div>
```

- Es wird häufig ein Container um die ganze Seite herum gebaut, damit die Inhalte nie breiter als der Container wird (Beispiel: Artikel soll zur besseren Lesbarkeit nicht zu breit werden).
- Der Container wächst, wenn der Screen wächst.
- Wichtige Container Klassen.

Klasse	Zweck
mx-auto	Zentriert den Container horizontal, wenn dies nicht gesetzt wird, ist der Container linksbündig.
px-4	Innenabstand links/rechts (sicherer Rand)
pt-8, pb-12	vertikaler Abstand oben/unten
max-w-*	Begrenzung der Breite manuell überschreiben
w-full	Optional: volle Breite innerhalb Eltern

Vermutlich selten gebraucht, daher nur zur Info: Die Standard-Breakpoints (die im Abschnitt Responsive gezeigt wurden), können für einen Container überschrieben werden in tailwind.config.js. Beispiel:

```
module.exports = {
  theme: {
    container: {
      center: true,
      padding: '1rem',
      screens: {
        sm: '600px',
        md: '728px',
        lg: '984px',
        xl: '1240px',
        '2xl': '1496px',
      },
    },
  },
}
```

Tailwind – Flexbox

CLOUD COMPUTING

FlexBox

- Ausrichtung entlang der Querachse (vertikal):

Klasse	Wirkung
items-start	Oben ausrichten
items-center	Vertikal zentrieren
items-end	Unten ausrichten
items-stretch	Auf volle Höhe dehnen (Standard)

```
<div class="flex items-center justify-between">
  <span>Links</span>
  <span>Rechts</span>
</div>
```

Tailwind – Grid

CLOUD COMPUTING

Grids

- Grids erlauben die responsive und flexible Darstellung von 2D-Layouts.
- Folgende Tailwind Klasse aktiviert das Grid

```
<div class="grid">...</div>
```

- Grids sind in Rows und Colums organisiert. Jede Row kann Columns enthalten.
- Die Anzahl der Columns wird folgendermaßen festgelegt.

Klasse	Bedeutung
grid-cols-1	1 Spalte
grid-cols-2	2 Spalten
...	...
grid-cols-12	12-Spalten-Raster
grad-cols-none	Keine definierte Spaltenstruktur
col-span-x	In einer Row werden mehrere x Columns zu einer Columns zusammengefasst.

- Bei den Rows kann für einfache Layouts einfach `<div class="row">...</div>` angegeben werden. Damit werden die Rows responsive einfach untereinander dargestellt und können auch deren Größe ändern:

- Es gibt jedoch 3 Fälle, in denen die Anzahl der Rows festgelegt ist.
  - Wenn die Größe fest sein soll, hier im Beispiel h-60:

```
<div class="grid grid-rows-3 h-60">
  <div class="bg-red-200">1</div>
  <div class="bg-green-200">2</div>
  <div class="bg-blue-200">3</div>
</div>
```

  - Wenn Elemente gezielt platziert werden sollen, hier im Beispiel mit row-span – hier werden zwei Zeilen zusammengefasst und eine darunter dargestellt.

```
<div class="grid grid-rows-3">
  <div class="row-span-2 bg-blue-200">nimmt 2 Zeilen</div>
  <div class="bg-yellow-200">normales Element</div>
</div>
```

  - Wenn Layouts mit fester Struktur umgesetzt werden solle, hier im Beispiel Header, Main und Footer-Teil.

```
<div class="grid grid-rows-3 min-h-screen">
  <header class="bg-gray-200">Header</header>
  <main class="bg-white">Inhalt</main>
  <footer class="bg-gray-300">Footer</footer>
</div>
```

Tailwind – Grid

CLOUD COMPUTING

- Klassen für die Festlegung der Reihenanzahl:

Klasse	Bedeutung
grid-rows-1	1 Zeile
...	...
grid-rows-6	bis zu 6 Zeilen
grid-rows-none	Keine Zeilenstruktur
row-span-x	In einer Row werden mehrere x Columns zu einer Columns zusammengefasst.

- Der Abstand zwischen den Zellen kann mit gap eingestellt werden:

Klasse	Beschreibung
gap-4	1rem (16px) Abstand zwischen Zeilen & Spalten
gap-x-4	Horizontaler Abstand (Spalten)
gap-y-4	Vertikaler Abstand (Zeilen)

- Die Ausrichtung erfolgt mit folgenden Klassen

Klasse	Funktion
justify-items-start	Inhalt innerhalb der Zelle linksbündig
justify-items-center	horizontal zentriert
items-start	vertikale Ausrichtung oben
items-center	vertikale Zentrierung
place-items-center	zentriert in beide Richtungen

- Automatische Ausrichtung

Klasse	Beschreibung
auto-cols-auto	Spalten passen sich Inhalt an
auto-rows-fr	Zeilen mit gleichem Anteil
auto-cols-fr	Spalten im gleichen Verhältnis

Tailwind – Grid

CLOUD COMPUTING

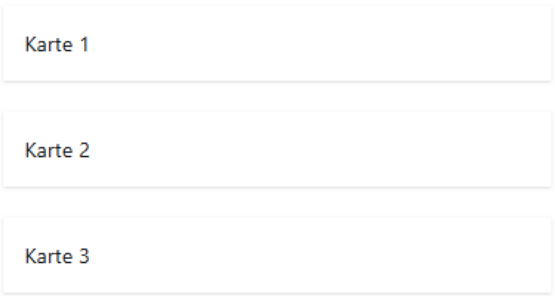
- Responsive Grids: Durch die Selektoren sm, md, ... kann das Grid flexibel an die Bildschirmgröße angepasst werden. Im Beispiel werden 2 Colums bei kleinen Screens und 4 bei großen gewählt.

```
<div class="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4">
  ...
</div>
```

- Beispiel

```
<div class="grid grid-cols-1 md:grid-cols-3 gap-6 p-4">
  <div class="bg-white shadow p-4">Karte 1</div>
  <div class="bg-white shadow p-4">Karte 2</div>
  <div class="bg-white shadow p-4">Karte 3</div>
</div>
```

md:grid-cols-3



grid-cols-1 bis md, also für sm

Tailwind – Navbar

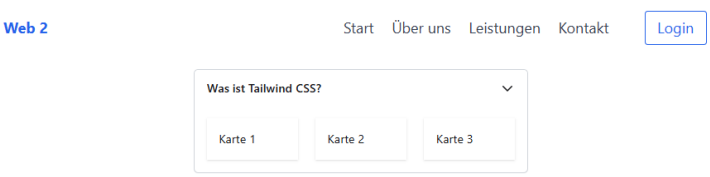
CLOUD COMPUTING

- Navbars können ebenfalls mit Tailwind gestaltet werden.
- Navbars bestehen meist aus folgenden Teilen:
  - einem Container (div, nav, header)
  - einem Logo / Titel
  - einer Navigationsliste (ul > li > a)
  - evtl. Buttons (zum Login, Sprachumschaltung, etc.)
  - Empfohlen: Burger-Menü für Mobilwebseiten (Selektor sm)

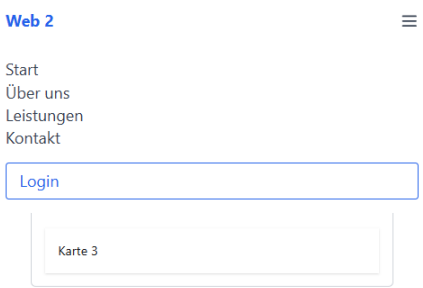
Relative z-10 wird für Burger Menu gebraucht

```
<nav class="bg-white w-full relative z-10">
  <div class="max-w-6xl mx-auto px-4 py-4 flex items-center justify-between">

    <!-- Logo links -->
    <div class="text-xl font-bold text-blue-600">Web 2</div>

    <!-- Burger-Menü für Mobil -->
    <input id="nav-toggle" type="checkbox" class="peer hidden" />
    <label for="nav-toggle" class="cursor-pointer md:hidden">
      <svg class="w-6 h-6 text-gray-700" fill="none" stroke="currentColor" stroke-width="2" viewBox="0 0 24 24">
        <path stroke-linecap="round" stroke-linejoin="round" d="M4 6h16M4 12h16M4 18h16"/>
      </svg>
    </label>

    <!-- Navigation + Login rechts -->
    <div class="hidden peer-checked:flex absolute top-full left-0 w-full flex-col gap-4 bg-white px-4 py-4
                md:flex md:static md:flex-row md:items-center md:gap-6 md:ml-auto md:w-auto">

      <!-- Navigation rechts -->
      <ul class="flex flex-col md:flex-row md:gap-6 md:items-center md:ml-auto">
        <li><a href="#" class="text-xl text-gray-700 hover:text-blue-600">Start</a></li>
        <li><a href="#" class="text-xl text-gray-700 hover:text-blue-600">Über uns</a></li>
        <li><a href="#" class="text-xl text-gray-700 hover:text-blue-600">Leistungen</a></li>
        <li><a href="#" class="text-xl text-gray-700 hover:text-blue-600">Kontakt</a></li>
      </ul>

      <!-- Login-Button ganz rechts -->
      <a href="#"
         class="text-xl border border-blue-600 text-blue-600 rounded px-4 py-2 hover:bg-blue-600
                hover:text-white transition md:ml-6">
        Login
      </a>
    </div>

  </div>
</nav>
```

Navbar rechtsbündung



ab md



für sm

Tailwind – Animations

CLOUD COMPUTING

Animations

- Tailwind beinhaltet einige (kleine) Animationseffekte.
- CSS bietet eine @keyframe an, mit denen Animationseffekte wie Fades oder ähnliches umgesetzt werden können.
- Beispiel für einen Fade:

```
@keyframes fadeIn {
  0%   { opacity: 0; }
  100% { opacity: 1; }
}
```

- Dies kann dann in CSS genutzt werden:

```
.element {
  animation: fadeIn 1s ease-in-out;
}
```

- Eigene @keyframes können in der tailwind.config.js definiert werden und dann global im ganzen Projekt genutzt werden.

```
module.exports = {
  theme: {
    extend: {
      keyframes: {
        fadeIn: {
          '0%': { opacity: '0' },
          '100%': { opacity: '1' }
        }
      },
      animation: {
        'fade-in': 'fadeIn 0.5s ease-out'
      }
    }
  }
}
```

- In Tailwind wurden bereits einige Animationsklassen vordefiniert:

Klasse	Wirkung
animate-none	Keine Animation
animate-spin	Dreht das Element kontinuierlich im Uhrzeigersinn
animate-ping	Einmaliger, wachsender "Ping"-Effekt (wie ein Radarsignal)
animate-pulse	Sanftes Ein-/Ausblenden (ideal für Ladezustände)
animate-bounce	Auf-/Ab-Bewegung (z. B. für Aufmerksamkeit)

- Beispiel: Endlos drehender Ladespinner

```
<div class="animate-spin w-8 h-8 border-4 border-blue-500 border-t-transparent rounded-full"></div>
```

Tailwind – Animations

CLOUD COMPUTING

Animationsparameter

- Animationen können durch Angabe der folgenden Klassen angepasst werden.

Thema	Beispiele
Standardklassen	animate-spin, animate-bounce, animate-pulse
Dauer	duration-75 bis duration-1000 (Stufen 75,100,150,200,300,500,700,1000)
Verzögerung	delay-75-1000
ease-linear	Gleichmäßige Geschwindigkeit
ease-in	Langsamer Start
ease-out	Langsames Ende
ease-in-out	Langsamer Start und langsames Ende

Nur zur Info:

- Tailwind Animationen werden immer dauerhaft angezeigt (animieren also so lange die Seite angezeigt wird, wenn nichts anderes angegeben ist).
- Bei eigenen CSS-Klassen zur Animation muss @appli animate-bounce angegeben werden, wenn die Animation nicht stoppen soll. Beispiel:

```
.my-bounce {
  @apply animate-bounce;
  animation-iteration-count: infinite;
}
```