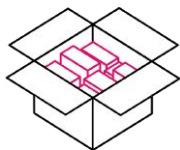




Workshop im Modul Web Technologien (WT)  
Medieninformatik Master  
Sommersemester 2022  
. 05 July .2022

Javad Alamdar & Sarfaroze Khakimov



Technology  
Arts Sciences  
TH Köln

# Agenda

- Einführung
- VueJS
  - Hintergrund
  - Vergleichen React & Angular & Vue
  - Install
  - Struktur von Vue
  - Aufgabe 1
- Tailwind CSS
- CSS Evolution
  - CSS
  - SASS
  - BEM
  - CSS-Moduls
  - Styled Components
- CSS Frameworks
  - Utility-First CSS Framework
  - Code Beispiel
  - Utility-Klassen
  - Vor- und Nachteile
  - Tailwind CSS verwenden
- Aufgabe 2-3
- Zusammenfassung

# Einführung

In diesem Workshop werden zunächst die Grundlagen des Vue.js-Frameworks gegeben und die Tailwind-Klassen in Vue-Komponenten verwendet. Der Schwerpunkt des Workshops liegt auf dem Utility-First CSS Konzept und TailwindCSS





vue.js

# Hintergrund

**Die Geschichte von Vue.js beginnt im Jahr 2013, als Evan You bei Google arbeitete und er veröffentlichte Vue.js offiziell im Jahr 2014.**

**Vue.js ist ein JavaScript-Framework für Frontendentwicklung.**

**Verwendung von Electron Framework für die Entwicklung von Desktop- und mobilen Anwendungen.**

**Vue.js Prinzipien des „Model View ViewModel“, kurz MVVM-Entwurfsmusters in den Fokus rückt.**

**License: MIT License**

**Geschrieben in: TypeScript**

**Aktuellste Version : 3.2 veröffentlichte im August 5, 2021**

# Vergleichen React & Angular & Vue

## Pro :

- Schnelles Laden von neuen Daten.
- Ermöglicht die Trennung von Daten und Präsentation.
- Der Start ist einfach und erfordert nicht viel Übung.
- Verwendet Jest als Test-Runner.



## Con :

- Es handelt sich nur um eine JavaScript-Bibliothek, nicht um ein Framework.
- Keine Möglichkeit zur Implementierung einer MVC-Architektur.

## Pro :

- Angular Templates sind HTML-Strukturen
- Ermöglicht MVC-Architektur.
- Einfache Unit- und End-to-End-Tests.



## Con :

- Langsame Ladezeit aufgrund der Ionic-App.
- Um alle Änderungen am DOM zu erfassen, erstellt Angular für jede Bindung einen Watcher.
- Um mit Angular arbeiten zu können, wird eine gewisse Einarbeitungszeit benötigt.

## Pro :

- Eine Liste von Werkzeugen und Bibliotheken.
- Flexibilität und Einfachheit in der Anwendung.
- Gründliche Dokumentation.



## Con :

- Begrenzte Community im Vergleich zu Angular und React
- Die Anzahl der verfügbaren Plugins

# Install

Besuchen Sie die Website <https://vuejs.org/>

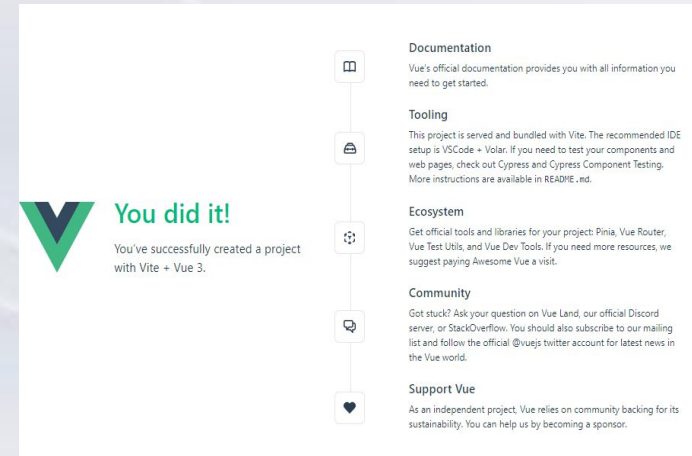
```
npm init vue@latest  
Oder  
npm init vue@3
```

```
Cd vue-project  
Npm install  
Npm run dev
```

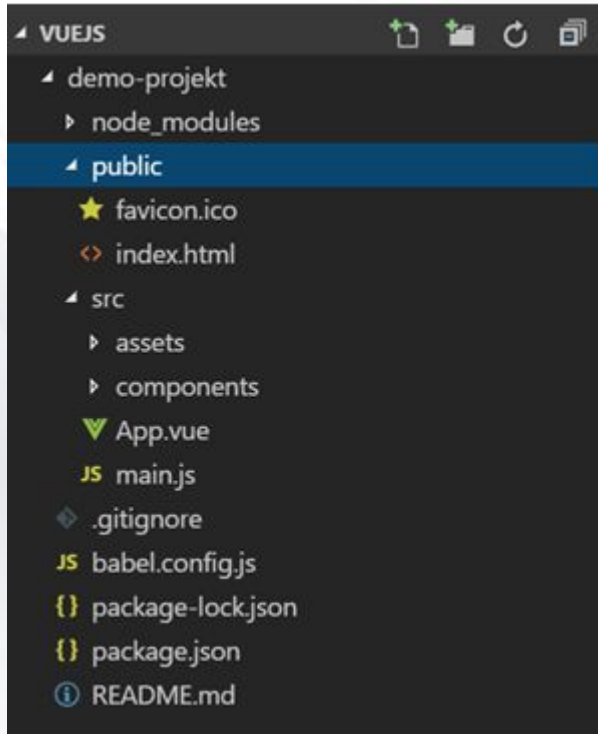
Server erreichbar sein unter  
<http://localhost:3000/>

```
✓ Project name: ... <your-project-name>  
✓ Add TypeScript? ... No / Yes  
✓ Add JSX Support? ... No / Yes  
✓ Add Vue Router for Single Page Application development? ... No / Yes  
✓ Add Pinia for state management? ... No / Yes  
✓ Add Vitest for Unit testing? ... No / Yes  
✓ Add Cypress for both Unit and End-to-End testing? ... No / Yes  
✓ Add ESLint for code quality? ... No / Yes  
✓ Add Prettier for code formatting? ... No / Yes  
  
Scaffolding project in ./<your-project-name>...  
Done.
```

```
> v3@0.0.0 dev  
> vite  
  
vite v2.9.13 dev server running at:  
  
> Local: http://localhost:3000/  
> Network: use `--host` to expose
```



# Struktur von Vue



```
<script setup>
import HelloWorld from './components/HelloWorld.vue'
</script>

<template>
  <header>
    

    <div class="wrapper">
      <HelloWorld msg="You did it!" />
    </div>
  </header>
</template>

<style>
@import './assets/base.css';
#app {
  max-width: 1280px;
  margin: 0 auto;
  padding: 2rem;
  font-weight: normal;}
</style>
```



## \ Multi Component

App.vue

```
<script>
import sidebar from "../components/sidebar.vue";
import anmelden from "../components/anmelden.vue";

export default {
  components: {
    sidebar,
    anmelden,
  }
};
</script>

<template>
  <sidebar />
</template>
```

sidebar.vue

```
<script>
import anmelden from "../anmelden.vue";
export default {
  components: {
    anmelden,
  }
}
</script>

<template>
  <anmelden />
</template>
```

anmelden.vue

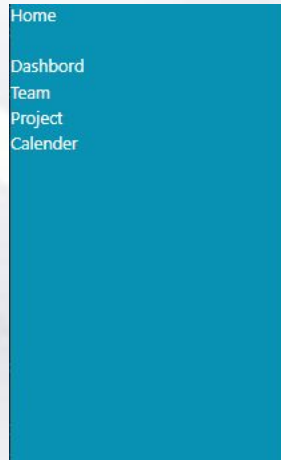
```
<script setup>
</script>

<template>
  <div class="flex-1 pt-8 bg-gray-200 text-gray-600">
    Content
  </div>
</template>
```

## Aufgabe 1

Entwickeln Sie 2 Komponenten mit den Namen **Navbar.vue** und **Login.vue**. Die **Navbar.vue** muss als die Hauptkomponente betrachtet werden.

in **Navbar.vue** eine Navigationsliste wie im Beispiel erstellen

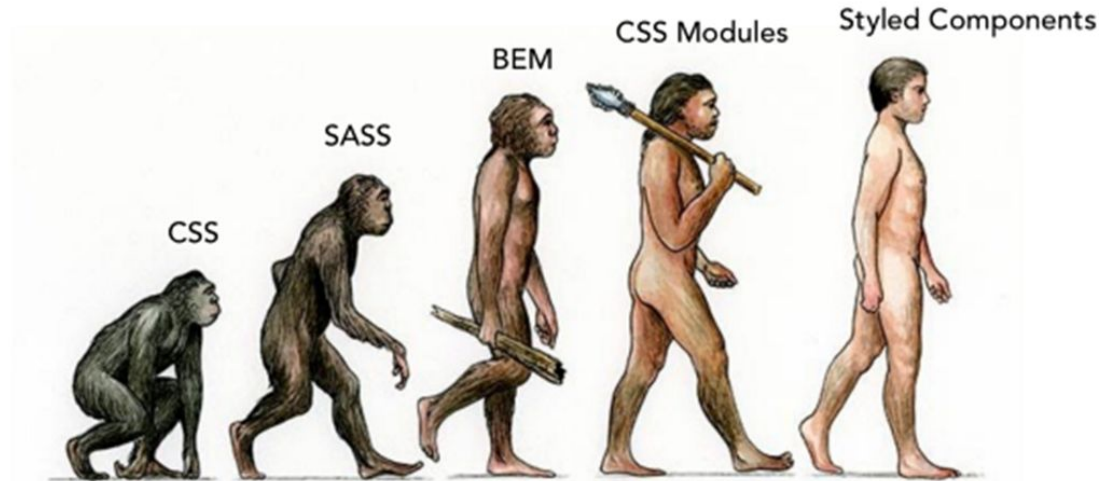


Erstellen sie in der **Anmelden.vue** einen Contentbereich, der in Aufgabe 3 weiter entwickelt werden muss.



# Tailwind CSS

# CSS Evolution



Quelle: <https://miro.medium.com>

# CSS

- jeder Entwickler hatte seine eigene Art, CSS zu machen
- „!important DOES NOT FIX YOUR BAD CSS “
- Probleme mit zunehmender Komplexität und Teamzugehörigkeit

# SASS

- SASS zur Rettung
- neue Möglichkeiten
- neue Problemen

# BEM

- Wiederverwendbarkeit/ reusability
- uniqueness of components

```
<body class="scenery">
  <section class="scenery__sky">
    <div class="sky [sky--dusk / sky--daytime] [sky--foggy]">
      <div class="sky__clouds"></div>
      <div class="sky__sun"></div>
    </div>
  </section>
  <section class="scenery__ground"></section>
  <section class="scenery__people"></section>
</body>
```

BEM Codebeispiel

## Entstehung neuer Probleme

- Wahl eines Klassennamens
- Markup ist mit all diesen langen Klassennamen aufgebläht.
- UI-Komponentenerweiterung
- Überflüssig semantisches Markup

```
// Block
.scenery {
  //Elements
  &__sky {
    fill: screen;
  }
  &__ground {
    float: bottom;
  }
  &__people {
    float: center;
  }
}

//Block
.sky {
  background: dusk;

  // Elements
  &__clouds {
    type: distant;
  }
  &__sun {
    strength: .025;
  }
}

// Modifiers
&--dusk {
  background: dusk;
  .sky__clouds {
    type: distant;
  }
  .sky__sun {
    strength: .025;
  }
}

&--daytime {
  background: daylight;
  .sky__clouds {
    type: fluffy;
    float: center;
  }
  .sky__sun {
    strength: .7;
    align: center;
    float: top;
  }
}
```

BEM Codebeispiel

# CSS Modules and local scope

- Erstellung dynamischer Klassennamen
- BEM-Automatisierung

- `.app-components-button-__root - 3vvFf {}`

```
@import '~tools/theme';

:local(.root) {
  border: 1px solid;
  font-family: inherit;
  font-size: 12px;
  color: inherit;
  background: none;
  cursor: pointer;
  display: inline-block;
  text-transform: uppercase;
  letter-spacing: 0;
  font-weight: 700;
  outline: none;
  position: relative;
  transition: all 0.3s;
  text-transform: uppercase;
  padding: 10px 20px;
  margin: 0;
  border-radius: 3px;
  text-align: center;
}
```

```
@mixin button($bg-color, $font-color) {
  background: $bg-color;
  color: $font-color;
  border-color: $font-color;

  &:focus {
    border-color: $font-color;
    background: $bg-color;
    color: $font-color;
  }

  &:hover {
    color: $font-color;
    background: lighten($bg-color, 20%);
  }

  &:active {
    background: lighten($bg-color, 30%);
    top: 2px;
  }
}

:local(.primary) {
  @include button($color-primary, $color-white)
}

:local(.secondary) {
  @include button($color-white, $color-primary)
}
```

lokale CSS



# Vollständige CSS-Infusion in JavaScript mit Styled Components

```
import React from "react"
import styled from "styled-components"
// Simple form component

const Input = styled.input`
  background: green
`

const FormWrapper = () => <Input placeholder="hola" />

// What this compiles to:
<input placeholder="hola" class="dxLjPX">Send</input>
```

Codebeispiel Styled Components

```
import styled from "styled-components"

const Sky = styled.section`
  ${props => props.dusk && 'background-color: dusk' }
  ${props => props.day && 'background-color: white' }
  ${props => props.night && 'background-color: black' }
`;

// You can use it like so:
<Sky dusk />
<Sky day />
<Sky night />
```

Codebeispiel Styled Components

# CSS Frameworks



# What is the Utility-first CSS framework?

```
<button class = "bg-red-500 text-white font-bold py-2 px-4  
rounded">  
  Sign in  
</button>
```

*Codebeispiel Tailwind CSS*

```
hover:bg-blue:400
```

*Codebeispiel Tailwind CSS*

# Code Beispiel

## Vanilla CSS

```
<button class="btn">Click me</button>
```

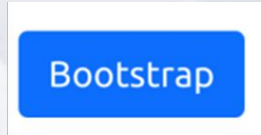
```
.btn {  
  background-color: #000;  
  color: #fff;  
  padding: 8px;  
  border: none;  
  border-radius: 4px;  
}
```

Codebeispiel 1 : CSS mit externen Stylesheet

## Bootstrap

```
1 <button type="button" class="btn btn-primary">  
2   Bootstrap  
3 </button>
```

Codebeispiel 2 : Button Komponente von Bootstrap



UI-Komponente : Bootstrap button

## Tailwind CSS

```
1 <button type="button" class="rounded bg-blue-500 hover:bg-blue-700 py-2 px-4 text-white">  
2   Tailwind  
3 </button>
```

Codebeispiel 3: Button-Element mit Tailwind CSS Utility-Klassen



UI-Komponente: Tailwind CSS Button

# Utility-Klassen

Utility-Klasse	CSS-Style
rounded	<code>border-radius: 0.25rem;</code>
bg-blue-500	<code>--tw-bg-opacity: 1;</code> <code>background-color: (59,130,246,var(--tw-bg-opacity));</code>
py-2	<code>padding-top: 0.5rem;</code> <code>padding-bottom: 0.5rem;</code>
px-4	<code>padding-left: 1rem;</code> <code>padding-right: 1rem;</code>
text-white	<code>--tw-text-opacity: 1;</code> <code>color: (255,255,255,var(--tw-text-opacity));</code>

Utility-Klassen und ihre CSS-Eigenschaften

# Vorteile

- Der Wechsel zwischen HTML-Files und CSS-Files entfällt
- Ermöglicht individuellere Lösungen für wichtige Elemente auf einer Website
- Responsive Framework und funktioniert auch auf mobilen Devices
- Schneller Entwicklungsprozess
- Komprimieren CSS durch vordefinierten Klassen
- Implementierung Modal-Komponenten
- Umfangreiche und leicht verständliche Dokumentation

# Nachteile

- Erste Schritte mit dem Framework ist schwierig
- Gegen das Prinzip der „Separations of concerns“ (Trennung der Verantwortlichkeiten)
- Bei der Installation von Tailwind CSS werden alle standardmäßigen CSS-Stile gelöscht
- Gegen den Grundsatz „Don't repeat yourself!“

# Flexbox-Beispiel

```
<div class="flex flex-row">
  <button> Button 1 </button>
  <button> Button 2 </button>
  <button> Button 3 </button>
</div>
```



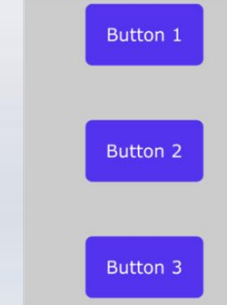
Codebeispiel Flex-row

```
<div class="grid grid-cols-3">
  <button> Button 1 </button>
  <button> Button 2 </button>
  <button> Button 3 </button>
  <button> Button 4 </button>
  <button> Button 5 </button>
  <button> Button 6 </button>
</div>
```



Codebeispiel für ein Raster

```
<div class="flex flex-col">
  <button> Button 1 </button>
  <button> Button 2 </button>
  <button> Button 3 </button>
</div>
```














































Codebeispiel Flex-col



# Farben

```
<button class="bg-red-50">Click me</button>
```

```
<p class="text-yellow-600">Hello World</p>
```

Blue										
	50	100	200	300	400	500	600	700	800	900
	#e6fffa	#d1f2e9	#c8e6c9	#b2dfdb	#a1d99b	#81c784	#66bb6a	#4fc3f7	#39a69d	#185e76
Indigo										
	50	100	200	300	400	500	600	700	800	900
	#e6e6ff	#d1c4e9	#c8a2c4	#b297bb	#a1887f	#81c784	#66bb6a	#4fc3f7	#39a69d	#185e76
Violet										
	50	100	200	300	400	500	600	700	800	900
	#f5f3ff	#ede9fe	#ddd6fe	#c4b5fd	#a78bfa	#8b5cf6	#7c3aed	#6d28d9	#5b21b6	#4c1d95
Purple										
	50	100	200	300	400	500	600	700	800	900
	#faf5ff	#f3e8ff	#e9d5ff	#d8b4fe	#c084fc	#a855f7	#9333ea	#7e22ce	#6b21a8	#581c87
Fuchsia										
	50	100	200	300	400	500	600	700	800	900
	#fdf4ff	#fae8ff	#f5d0fe	#f0abfc	#e879f9	#d946ef	#c026d3	#a21caf	#86198f	#701a75

Farben aus der Standardpalette von Tailwind anpassen

# Padding

- `p` bezeichnet das Padding über das gesamte Element.
- `py` bezeichnet padding padding-top und padding-bottom.
- `px` steht für padding-left und padding-right.
- `pt` bezeichnet padding-top.
- `pr` kennzeichnet padding-right.
- `pb` bezeichnet padding-bottom.
- `pl` kennzeichnet padding-left

```
<button class="p-0">Click me</button>
<button class="pt-1">Click me</button>
<button class="pr-2">Click me</button>
<button class="pb-3">Click me</button>
<button class="pl-4">Click me</button>
```

# Ränder

- `m`
- `my`
- `mx`
- `mt`
- `mr`
- `mb`
- `ml`

# Wie man ein Tailwind CSS Plugin erstellt

```
const plugin = require("tailwindcss/plugin");

module.exports = {
  content: ["/src/**/*.html", "/public/**/*.html"],
  theme: {
    extend: {},
  },
  plugins: [
    plugin(function ({ addUtilities }) {
      const myUtilities = {
        ".bg-aqua": { background: "aqua" },
        ".rotate-150deg": {
          transform: "rotateX(150deg)",
        },
      };
      addUtilities(myUtilities);
    }),
  ],
};
```

Schritt 1

```
const plugin = require("tailwindcss/plugin");
```

Schritt 2













```
plugins: [
  plugin(function ({ addUtilities }) {
    const newUtilities = {
      ".bg-aqua": { background: "aqua" },
      ".rotate-150deg": {
        transform: "rotateX(150deg)",
      },
    };
    addUtilities(newUtilities);
  }),
],
```

Schritt 3

```
<button class="bg-aqua rotate-150deg">Click me</button>
```

Schritt 4

# Instalieren

 <b>Next.js</b> > Full-featured React framework with great developer experience.	 <b>Laravel</b> PHP web application framework with expressive, elegant syntax.	 <b>Vite</b> Fast and modern development server and build tool.
 <b>Nuxt.js</b> Intuitive Vue framework for building universal applications.	 <b>Gatsby</b> Framework for building static sites with React and GraphQL.	 <b>Create React App</b> CLI tool for scaffolding a new single-page React application.
 <b>SvelteKit</b> The fastest way to build apps of all sizes with Svelte.js.	 <b>Angular</b> Platform for building mobile and desktop web applications.	 <b>Ruby on Rails</b> Full-stack framework with all the tools needed to build amazing web apps.
 <b>Remix</b> Full stack framework focused on web fundamentals and modern UX.	 <b>Phoenix</b> A framework to build rich, interactive applications with Elixir.	 <b>Parcel</b> The zero-configuration build tool for the web.

## Aufgabe 2

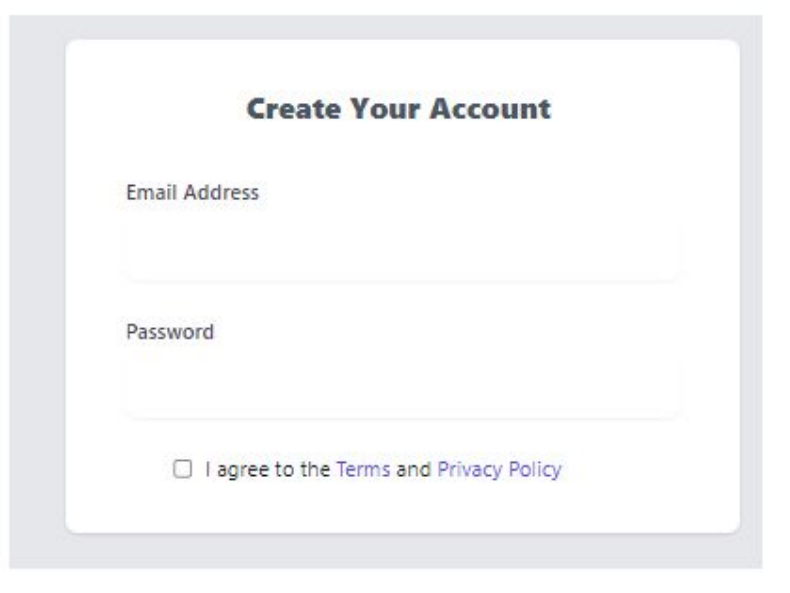


- Die Navigationsleiste mit den Tailwind CSS Klassen erweitern.

Für die Entwicklung der Navigationsleiste benötigen Sie die Icons, die wir vorher im Icon-Ordner als SVG-Datei vorbereitet habe.

### Aufgabe 3

- In der Component `anmelden.vue` haben Sie einen schönen `<div> Content </div>` erstellt. Entwickeln Sie nun diesen Bereich als Beispiel für ein Anmeldeformular.



**Create Your Account**

Email Address

Password

☐ I agree to the [Terms](#) and [Privacy Policy](#)

Für diese Aufgabe versuchen sie auch mit `index.css` arbeiten, wie beispiel.

Vorher

```
<form class="mb-0 mt-8 space-y-6" action="#" method="POST"></form>
```

`index.css`

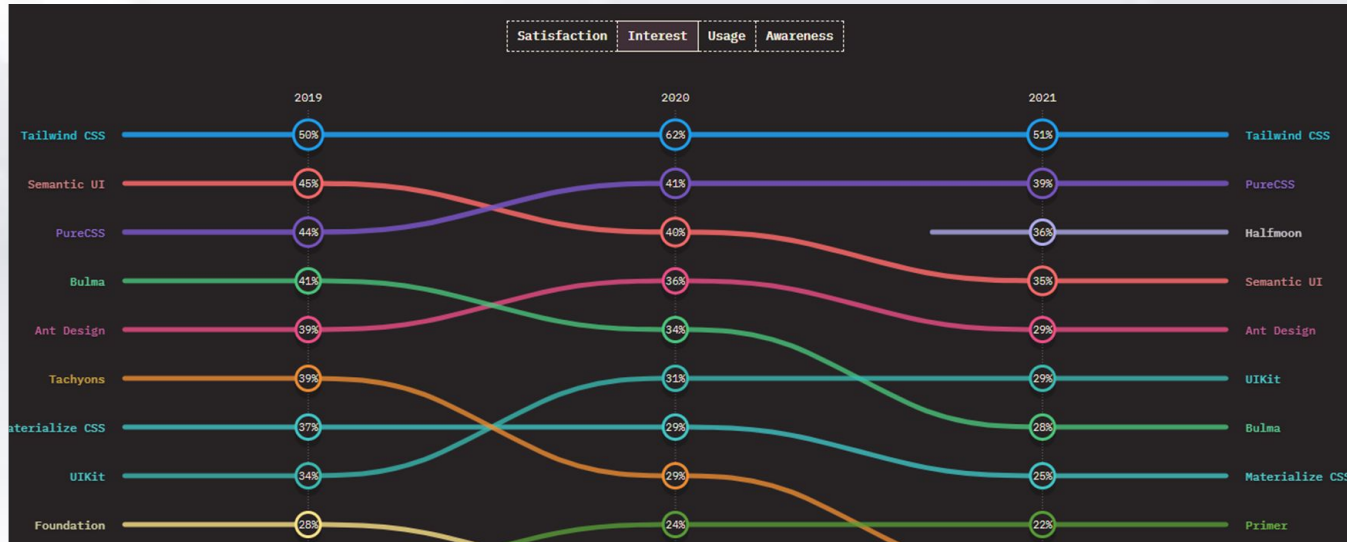
```
@layer components {  
  .form {@apply mb-0 mt-8 space-y-6;}  
}
```

Nachher

```
<form class="form" action="#" method="POST"></form>
```

# Zusammenfassung

- Wie ist eure Erfahrung mit diesem Framework ?
- Welche Probleme noch gelöst werden müssen ?
- Hat jemand eine Vorstellung davon, was die nächste Entwicklung von CSS (oder Web-Styling im Allgemeinen) sein wird?





**Vielen Dank für Ihre  
Aufmerksamkeit**



# Fragen? Anmerkungen?

## \ Install mit vue/cli

Besuchen Sie die Website <https://cli.vuejs.org/>

```
npm install -g @vue/cli  
Oder  
yarn global add @vue/cli
```

```
Cd vue-project  
  
Npm run serve
```

Server erreichbar sein unter  
<http://localhost:8080/>

```
Vue CLI v3.4.0  
? Please pick a preset: Manually select features  
? Check the features needed for your project:  
> Babel  
  TypeScript  
  Progressive Web App (PWA) Support  
  Router  
  Vuex  
  CSS Pre-processors  
  Linter / Formatter  
  Unit Testing  
  E2E Testing
```

```
DONE Compiled successfully in 2039ms  
  
App running at:  
- Local: http://localhost:8080/  
- Network: http://10.0.12.15:8080/
```

