LAPORAN RESMI

PRAKTIKUM DESAIN WEB

**PERTEMUAN 6**

**TAILWIND CSS**



**NIM : 21104410079**

**NAMA : Muhammad Yusuf Ibrahim Ramadhani**

**JURUSAN : Teknik Informatika**

**KELAS : TI 3-B 2022**

**TGL. PRAKTEK : 16 Januari 2023**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**UNIVERSITAS ISLAM BALITAR**

**2021**

LEMBAR PERSETUJUAN

**MODUL 6**

**TAILWIND CSS**

**NIM : 21104410079**

**NAMA : Muhammad Yusuf Ibrahim Ramadhani**

**JURUSAN : Teknik Informatika**

**KELAS : TI 3-B 2022**

**TGL. PRAKTEK : 16 Januari 2023**

**Disetujui,**

**Blitar, 23 Januari 2023**

**Dosen**

**Mohammad Faried Rahmat, S.ST., M.Tr.T**

**NIDN.**

# https://anf.al/assets/blog-images/2021-03-06-tailwindcss-review-by-a-backend-developer/tailwindcss.png

# **Tailwind Css**

Tailwind CSS merupakan framework CSS yang berbasis utility untuk membuat UI atau tampilan dari aplikasi web. Kebanyakan framework CSS yang lain menyediakan sesuatu secara berlebihan, penyediaan segala jenis komponen yang telah didesain sebelumnya, seperti: buttons, cards, alerts, dan lain sebagainya. Komponen-komponen tersebut mungkin membantu dengan cepat di awal, tetapi masalah baru muncul ketika kita ingin membangun situs kita dengan desain khusus yang sesuai keinginan kita. Melakukan perubahan terhadap komponen-komponen tersebut membutuhkan effort dan waktu yang tidak sedikit. Akan tetapi, Tailwind CSS memiliki pendekatan yang berbeda. Daripada menyediakan komponen-komponen yang sudah didesain sebelumnya, Tailwind menyediakan low-level utility classes yang dapat kita gunakan untuk membangun desain yang khusus secara keseluruhan, tanpa harus keluar dari file HTML kita.

1. Responsif

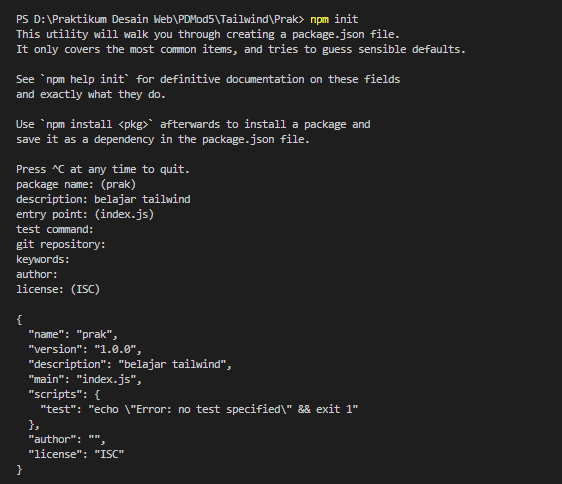
Setiap utilitas Tailwind memiliki jenis-jenis responsive yang mempermudah untuk membangun interface responsive tanpa perlu menggunakan CSS khusus. Tailwind menggunakan prefix {screen}: intuitif, sehingga mudah untuk mengetahui kelas-kelas responsif yang ada dalam markup kita sekaligus menjaga nama kelas original dapat dikenali dan utuh.

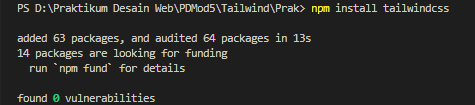
1. Component-friendly

Meskipun kita dapat melakukan banyak hal hanya dengan menggunakan kelas utilitas, seiring dengan berkembangnya sebuah proyek, akan berguna untuk mengkodifikasikan pola yang umum menjadi abstraksi tingkat tinggi. Tailwind menyediakan alat untuk mengekstraksi kelas komponen dari pola utilitas yang berulang, yang memudahkan untuk memperbarui beberapa instances dalam sebuah komponen dari satu tempat.

# **Install dan Setup Tailwind**

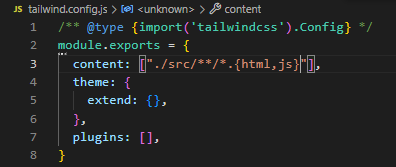
1. Pada proyek yang dibuat, masuk ke terminal ketikkan ‘npm init’. Perintah ini digunakan untuk membuat file package.json, dimana setelahnya kita diminta untuk memasukkan data dari proyek yang kita buat.

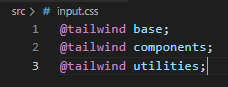


1. Setelah itu, kita memlakukan pengistalan dengan memasukkan perintah ‘npm install tailwindcss’. Ini akan menambahka tailwind css kedalam package.json dan membuat folder node\_module dialam proyek yang dibuat.
2. 
3. Setelah penginstalan selesai seperti diatas, kita perlu mengkonfigurasi tailwind css dengan memasukkan perintah ‘npx tailwind init’. Perintah ini akan membuat file baru bernama tailwind.config.js, yang berisi konfigurasi yang dibutuhkan tailwind.

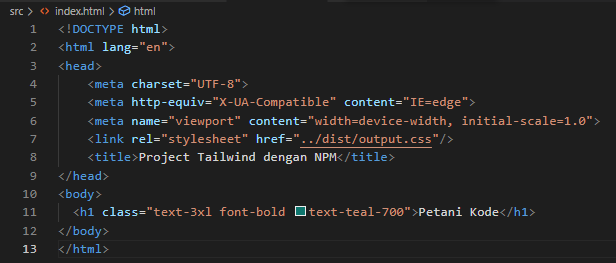


1. Lakukan konfigurasi path konten pada tailwind.config.js

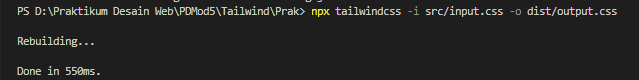


1. Setelah itu, buat folder baru dengan nama src di dalam project. Kemudian buat file CSS baru di dalam src dengan nama input.css dan isi seperti ini:
2. 
3. Sebagai contoh kita membuat file index.html di dalam folder src untuk menggunakan class-class dari Tailwind.

Isi file index.html seperti ini:



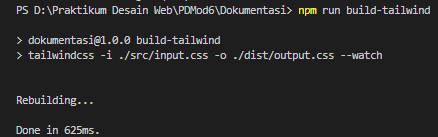
1. Pada gambar, terdapat kode ../dist/output.css. File ini tidak terdapat dalam project, maka kita perlu membuatnya dengan perintah ‘npx tailwindcss -i src/input.css -o dist/output.css’.



1. Lakukan konfigurasi pada file package.json dengan menambahkan script berikut.



1. Jalankan tailwind dengan memberikan perintah ‘npm run build-tailwind’.



# **Homepage Sekolah Koding**

1. **Bagian 1**

Menggunakan framework TailwindCSS sangat memudahkan developer dalam pembuatan sebuah website. Dengan memanfaatkan utilities class yang disediakan developer mampu membuat web dengan lebih praktis tanpa perlu menuliskan sintaks CSS secara menual.

Di tailwind sering dijumpai istilah sumbu X dan Y seperti pada contoh class my-2 atau px-2. Dimana X merujuk ke sumbu horizontal (kanan-kiri) dan Y merujuk ke sumbu vertical (atas-bawah).

Dan berikut ini merupakan beberapa class utilities yang digunakan dalam membangun halaman homepage.

|  |  |
| --- | --- |
| Class Utilities | Fungsi |
| p | Mengatur Padding (p, pt, pb, pl, pr, px, py) |
| M | Mengatur Margin (m, mt, mb, ml, mr, mx, my) |
| Flex | Display flex |
| Justify-center/between/around/evenly | Mengatur letak elemen kea rah horizontal |
| Items-center/start/end | Mengatur letak elemen ke arah vertikal |
| Rounded | Membuat elemen menjadi bulat |
| bg | Mengatur background |
| Bg-gray-500 | 500 di sini merupakan ketebalan background |
| Text-xs | Mengatur ukuran text (text-xs, text-sm, text-md, text-xl) |
| Text-white | Mengatur warna text (text-(color)) |
| w | Mengatur width |
| h | Mengatur height |

1. **Bagian 2**

Konsep **responsive** pada tailwind menggunakan Breakpoint prefix. Konsep ini akan sering digunakan ketika ingin membuat tampilan website yang berbeda pada saat ukuran tertentu.

Untuk mengatur lebar pada point tertentu biasanya digunakan sintaks **breakpoint:size** atau sebagai contoh **md:w-4/6**.

|  |  |
| --- | --- |
| Breakpoint Prefix | Minimum width |
| sm | 640px |
| md | 768px |
| lg | 1024px |
| xl | 1280px |
| 2xl | 1536px |

Konsep **hover** adalah aksi dimana cursor terletak di atas elemen. Dengan tailwind penggunaan metode hover sangat mudah, yaitu dengan menggunakan sintaks **hover:aksi** atau sebagai contoh **hover:bg-red-500**.

Konsep **fokus** adalah aksi dimana elemen tertentu sedang digunakan. Dengan tailwind penggunaan metode fokus hampir sama denganhover, yaitu dengan menggunakan sintaks **focus:aksi** atau sebagai contoh **focus:outline-none**.

1. **Bagian 3**

Tailwind mendukung developer untuk melakukan kustomisasi pada class ataupun membuat class sendiri. Untuk membuat kustomisasi pada tailwind dapat dilakukan pada file **tailwind.config.js**. Sebagai contoh untuk kustomisasi warna tambahan pada tailwind dapat dilakukan pada baris **theme.**

/\*\* @type {import('tailwindcss').Config} \*/

module.exports = {

  content: ["./src/\*\*/\*.{html,js}"],

  theme: {

    extend: {

      colors: {

        trycol:{

          light: '#67e8f9',

        DEFAULT: '#06b6d4',

        dark: '#0e7490',

        }

      }

    },

  },

Pada baris kode diatas memasukkan warna kustom pada baris kode extend theme. Di letakkan pada baris extend karena disini kita hanya ingin menambahkan keyword warna baru pada projek website kita. Jika sudah rebuild ulang tailwind untuk melakukan perbaruan pada tailwind.

Untuk penggunaannya sama dengan warna pada tailwind biasa. Sebagai contoh perhatikan baris kode dengan keyword mycolor seperti yang telah dibuat di atas

<button class="mt-2 w-full text-center bg-trycol hover:bg-trycol-dark text-white py-2 text-l rounded-md">Daftar</button>

1. **Bagian 4**

Dalam menggunakan tailwind mungkin saja developer menemui kesulitan dimana harus menuliskan banyak kode untuk setiap elemen. Sebagai contoh untuk membuat **button** harus menambahkan beberapa utilities class tailwind. Dengan menggunakan konsep fungsi **apply** pada tailwind akan memudahkan developer dalam menulis kode agar lebih optimal.

Penggunaan konsep **apply** pada tailwind saya kira hampir mirip seperi pada pembuatan class dan css pada umumnya. Penulisan **apply** tailwind terjadi pada file main css pada project atau satu file pada import variable pada tailwindnya. Sebagai contoh perhatikan penjelasan berikut.

Jika ingin membuat button tanpa menggunakan konsep apply maka akan menggunakan beberapa class pada satu buah element.

<a class="text-sm text-white py-1 px-6 rounded-md bg-red-600 hover:bg-red-800">Google</a>

                    <a class="text-sm text-white py-1 px-6 rounded-md bg-gray-700 hover:bg-gray-900">Github</a>

                    <a class="text-sm text-white py-1 px-6 rounded-md bg-blue-400 hover:bg-blue-600">Twitter</a>

Dengan menggunakan **apply** baris kode tersebut bisa terlihat lebih rapi. Cara penggunaannya sebagai berikut.

Pada file utama css buat sebuah selector yang akan digunakan sebagai nama keyword class lalu masukkan keyword **@apply** dan diikuti nama class tailwind yang digunakan sebelumnya pada bagian deklarator. Dalam penggunaan **apply** sebisa mungkin hindari penggunaan nama class yang terlalu spesifik. Sebagai contoh perhatikan baris kode berikut.

.button{

    @apply text-sm text-white py-1 px-6 rounded-md;

}

Jika sudah cara penggunaannya sama seperti halnya penggunaan class tailwind pada umumnya. Sebagai contoh pada baris class sebelumnya baris kode terlihat sangat Panjang, dan setelah penggunaan metode **apply** baris kode akan terlihat lebih rapi.

<a class="button bg-red-600 hover:bg-red-800">Google</a>

                    <a class="button bg-gray-700 hover:bg-gray-900">Github</a>

                    <a class="button bg-blue-400 hover:bg-blue-600">Twitter</a>

1. **Bagian 5**

Penggunaan **group hover ke child** sama seperti konsep css position relative dan absolute. Dimana kedua metode tersebut saling berhubungan. Sama halnya konsep **group hover to child** pada parent element harus menerapkan class **group** untuk menandai bahwa class tersebut berada pada parent dari elemen HTML. Dan pada child class nya untuk menggunakan hover yaitu dengan menggunakan class **group-hover:aksi**. Sebagai contoh **group-hover:text-gray-900**. Karena property pada tailwind masih terbatas, maka jika ingin menambahkan sebuah property baru bisa ditambahkan pada file **tailwind.config.js** pada bagian variants lalu extend.

Sebagai contoh:

 varriants: {

    extend:{

      display:['group-hover']

    },

  },

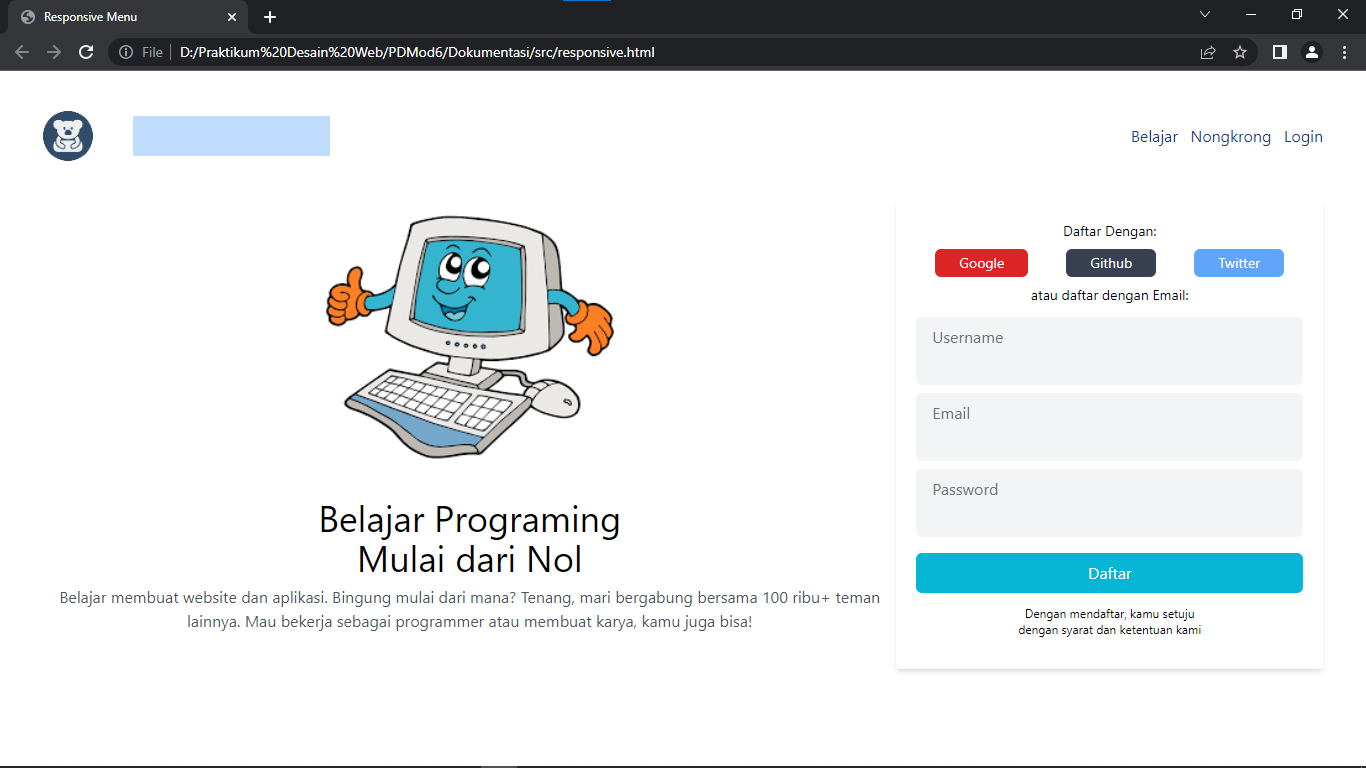
Setiap melakukan penambahan atau kustomisasi pada tailwind lakukan re build untuk melihat perubahan yang terjadi.

1. **Bagian 6**

Responsive menu mempunyai konsep yang sama pada responsive pada tailwind yang sudah dijelaskan. Namun pada responsive bagian menu terdapat beberapa trik untuk mengakali tampilan website yang berbeda pada setiap device. Dengan penggunaan sintaks yang sama dan diikuti dengan pengaturan display pada elemen HTML hal itu dapat digunakan untuk mengatur tampilan website antara device satu dengan lainnya.

**Output:**

**Desktop**

****