Vite és Tailwind

Node

- JS futtatókörnyezet - nodejs.org
- A Google V8 JavaScript Engine van mögötte
- Előnyei:
  - Frontenden és backenden azonos nyelv (JS) használható
  - Aszinkron, gyors I/O műveletek
- Csomagkezelők:
  - npm (alapértelmezett)
  - pnpm
  - Yarn (Meta / Facebbok)
- Build eszközök
  - Vite
  - esbuilld
  - Webpack
  - Rollup
- Alternatívák:
  - Deno - deno.com
  - Bun - bun.com

Dockerfile

```
FROM node:24-alpine

WORKDIR /app

RUN apk add --no-cache bash \
    && npm install -g pnpm

USER node

ENV PNPM_STORE_DIR=/home/node/.pnpm-store
RUN pnpm config set store-dir "$PNPM_STORE_DIR"

EXPOSE 5173

ENTRYPOINT ["/bin/bash"]
```

- A hivatalos `node` image az alap
- A projektünk a `/app` mappában lesz
- A `node`-ot globálisan telepítsük
- Használjuk a `root` helyett az image-ben lévő `node` felhasználót
- A `PNPM_STORE_DIR` adja meg hol tárolja a letöltött fájlokat
- `5173` jelölése a Vite-hez
- Fejesztéshez a`bash`-t indítjuk

Docker környezet indítása

Build

```
docker build -t rcs/node .
```

- A neve legyen `monogram/node`

Run

```
docker run -it -v "$(pwd):/app" rcs/node
```

- Interaktív módban futtatjuk: `-it`
- A aktuális mappát a container `/app` mappájra csatoljuk

Első próba

A containeren belül futtassuk:

```
node -v
```

- Megadja a `node` aktuális vezióját (pl: v24.3.0)

```
npm -v
```

- Megadja az alapértelmezett `npm` csomagezelő verzióját (pl: 11.4.2)

```
pnpm -v
```

- Megadja az alapértelmezett `pnpm` csomagezelő verzióját (pl: 10.26.2)

pnpm

- Csomagkezeelő Node.js-hez
- A csomagokat a `.pnpm-store` -ban tárolja
- A `node_modules` mappában csak hivatkozik rájuk
- Gyorsabb és helytakarékosabb

pnpm init

```
pnpm init
```

- Létrehozza a `package.json` fájlt

```
{
  "name": "folder_name",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "packageManager": "pnpm@10.26.2"
}
```

pnpm add

```
pnpm add chalk
```

- Az `add` hozzáadja a megadott csomagot a `dependencies`-hez
- A `-P` vagy a `--save-prod` is a `dependencies` részhez adja hozzá

```
pnpm add tailwindcss @tailwindcss/vite
```

- Egyszerre hozzáadja a `tailwindcss` és a `@tailwindcss/vite` csomagokat

```
pnpm add -g typescript
```

- A `-g` vagy `--global` hatására globálisan telepíti a csomagot

```
pnpm add -D vite
```

- A `-D` vagy `--save-dev` miatt a `devDependencies`-hez adja hozzá

https://pnpm.io/cli/add

Vite

- Frontend build tool
  - Fejlszetői servert biztosít
  - Teljes projektet publikálhatóvá teszi (optimalizálva)
  - HMR: Hot Module Replacement - Automatikus (részleges) újratöltődés
- Dokumentó: https://vite.dev/

Böngsző támogatás

- Fejelsztés közben:
  - **Legfrissebb** CSS és JS támogatás: `esnext` [1]
- Publikálásra előkészítve:
  - Csak a **széleskörűen elterjed** CSS és JS támogatása:
  - "Baseline - Widely available" [2]
  - Legalább 30 hónapos támogatottság

---

1. https://esbuild.github.io/api/#target ↩
2. https://web-platform-dx.github.io/web-features/ ↩

Vite projekt sablon alapján

Előfeltételek: Legyen `node` és `pnpm` feltelepítve!

```
pnpm create vite
```

```
◇  Project name:
│  hello-world
◇  Select a framework:
│  Vanilla
◇  Select a variant:
│  JavaScript
◇  Use rolldown-vite (Experimental)?:
│  No
◇  Install with pnpm and start now?
│  Yes
```

- Létrehozza a projektet a `hello-world` mappában

Alap mappaszerkezete

index.html

Projekt belépési pontja

public

A weboldalon elérhető publikus fájlok

- képek, videók, zenék

assets

Forrásfájlok

- JS, CSS, ikonok

viteconfig.js

Vite konfiguráiós fájl

index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>hello-world</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.js"></script>
  </body>
</html>
```

- Modulként tölti be a `main.js` fájlt
- Az elérést a (vite) szerver gyökerétől adja meg abszolút hivatkozással

vite.config.js

```
import { defineConfig } from "vite";

export default defineConfig({
    server: {
        host: true,
        port: 8080,
        strictPort : true,
        allowedHosts: ["vm1.test"]
    }
})
```

A fájlt nekünk kell létrehozni

- A `server` a dev szervert konfigurálja
- A `host` legyen `'0.0.0.0'` vagy `true`, így elérhető dockeren kívül is
- A megadott `port`-on fut a szerver
  - Alapértelmezett: `5173`
- Ha `strictPort` értéke `true`, akkor ragaszkodik a beállított porthoz, nem keres másikat
- Az `allowedHosts` egy string tömb, ami a használható host neveket

CSS

Az `src/style.css` fájl:

```css
:root {
    color-scheme: light dark;
    color: rgb(255 255 255 / 0.87);
    background-color: #242424;
}
```

Az `src/main.js` fájl:

```js
import './style.css'
```

- A CSS fájlunkat a Vite fogja betölteni!
- **Az útvonalat a `main.js` fájlhoz képest kell megadni!**

Alias

```
import { defineConfig } from "vite";
import { fileURLToPath } from "node:url";

export default defineConfig({
    server: {
        // ...
    },
    resolve: {
        alias: {
            "@" : fileURLToPath(new URL("./src", import.meta.url))
        }
    }
})
```

- A `fileURLToPath` függvény a (beépített) `url` modulban található
- A config `resolve.alias` blokkjában lehet álneveket definiálni

@

```
alias: {
    "@" : fileURLToPath(new URL("./src", import.meta.url))
}
```

- Az `import.meta`[1] a modul környezetfüggő metaadatait tartalmazza
  - Átalában az `url` tartalmazza a teljes elérési útvonalat
  - Node esetében az eleje: `file://`

```
import.meta.url                                 # file:///app/vite.config.js
fileURLToPath(import.meta.url)                  # app/vite.config.js
fileURLToPath(new URL("./src", import.meta.url))    # /app/src
```

Eredmény: A `/app/src` mappára hivatkozhatunk egyszerűen a `@` rövidítéssel.

---

1. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/import.meta ↩

VS Code compiler option

A projekt gyökerében lévő `jsconfig.json` fájlban megadható milyen álneveket használunk.

```
{
    "compilerOptions": {
        "baseUrl": ".",
        "paths": {
            "@/*": ["src/*"]
        }
    },
    "include": ["src"]
}
```

Alias használata

```
import '@/style.css'
```

- A `@` egy abszolút hivatkozás az `src` mappára pl.: `/app/src`

Ahhoz, hoqy a css fájlokat is felajánlja importálásra, létre kell hozni az `src/vite-env.d.ts` fájlt az alábbi tartalommal:

```
/// <reference types="vite/client" />
```

- Nélküle is működni fog, de így kényelmesebb

VS Code pluginok

- Tailwind CSS IntelliSense

Tailwind telepítés

```
pnpm create vite@latest hello-tailwind
```

```
◇  Select a framework:
│  Vanilla
◇  Select a variant:
│  JavaScript
◇  Use rolldown-vite (Experimental)?:
│  No
◇  Install with pnpm and start now?
│  Yes
```

- Létrehoz a Vite-en alapuló alap projektet a `hello-world` mappában

Csomagok telepítése

```
pnpm add tailwindcss @tailwindcss/vite
```

- Szükséges a Tailwind keretrendszer (`tailwindcss`) telepítése
- Szükséges a Tailwind Vite integrációjának (`@tailwindcss/vite`) a telepítése

Konfigurálás

A `vite.config.js` fájlban:

```
import { defineConfig } from "vite";
import tailwindcss from '@tailwindcss/vite';

export default defineConfig({
    server: {
        host: true,
        port: 8080,
        strictPort : true,
        allowedHosts: ["vm1.test"]
    },
    plugins: [
        tailwindcss()
    ]
})
```

- Kell a `tailwindcss` függvény a `@tailwindcss/vite` modulból
- A config `plugins` blokkjában kell meghívni.

Konfigurálás II.

A `style.css` tartalma:

```
@import "tailwindcss";

body {
  background-color: lightskyblue;
}
```

- Betölti a szükséges fájlokat a `@tailwindcss/vite`-nek köszönhetően

> 📌 **Fonts!**
>
> Az import fájl elején legyen!

> 🟡 **Hiba!!**
>
> [vite:css][postcss] @import must precede all other statements

Mi az a tailwind?

- Utility-first CSS framework
- Alapvetően utility osztályokat definiál
- Léteznek komponensek is (fizetős)

```
<button class="px-4 py-2
    bg-blue-600 hover:bg-blue-700 text-white
    transition-colors duration-400 cursor-pointer
    rounded-md text-sm md:text-lg lg:text-xl">
    Mentés
</button>
```

- Töréspontok egyszerű alkalmazása: `md:text-lg`
- Pszeudo osztályok: `hover:bg-blue-700`

Preflight[1]

- A böngészők különböző alap CSS beállításokat használnak
- A preflight.css[2] egységes alapot ad
- A külső és belső margókat lenullázza
- A szegély 0 méretű folytonos vonallá állítja
- A címsorok betűméretét és vastagságát alapra állítja
- Képek és videók megkapják a
  - `display:block`, `max-width: 100%`, `height auto` tulajdonságokat

---

1. Preflight -tailwindcss.com ↩
2. preflight.css ↩

oklch

- Az oklch színkeverési módot használ https://oklch.com/
  - Az RGB-nél nagyobb színtartományt fed le

oklch(L C H / a)

- `L`: lightness
- `C`: chroma
- `H`: hue
- `a`: alpha (opacity)

Színárnyalatok

- A paletta 100 és 900 között 100-as lépésközzel változik
- +1 legvilágosabb az 50
- +1 legsötétebb a 950 lesz
- Így összesen 11 különböző árnyalatot használhatunk

```
<div class="bg-green-500"></div>
<span class="text-green-500"></span>
<p class="border-2 border-amber-400"></p>
```

Speciális színek

- A `bg-current` a hátteret `currentColor` -t állítja be
- A `border-transparent` a szegély színét átlátszóra állítja
- A `text-black` feketére, míg a `text-white` fehérre állítja a szöveg színét

Átlátszóság

```
<div class="bg-pink-500/40">
```

- Az átlátszóságot `/` jellel lehet meghatározni
- Az `alpha` értéket a megadott értékre állítja (itt 40-re)
- A háttér átlátszósága így `60%` lesz

Pszeudo osztályok

- https://tailwindcss.com/docs/hover-focus-and-other-states#pseudo-class-reference

Tailwind

```
<button class="bg-green-500 hover:bg-green-700">
    Save
</button>
```

- A `hover:` prefix megadásával megadható, hogy hogyan nézzen ki, ha az egeret az elem fölé visszük
- A CSS Specificity miatt a sorrend itt nem számít

Generált CSS

```
.bg-green-500 {
    background-color: #008236;
}

.hover\:bg-green-700:hover {
    background-color: #00c951;
}
```

- Külön osztályt hoz létre
- Ami csak `:hover` állapotban érvényesül

Néhány pszeudo osztály

Felhasználói interakció és linkek

- `hover:` → `:hover`
- `focus:` → `:focus`
- `active:` → `:active`
- `visited:` → `:visited`
- `target:` → `:target`

Nyomtatás

- `print:` → `@media print`

Elheylezkedés szerint

- `first:` → `:first-child`
- `last:` → `:last-child`
- `odd:` → `:nth-child(odd)`
- `even:` → `:nth-child(even)`
- `first-of-type:` → `:first-of-type`
- `last-of-type:` → `:last-of-type`
- `empty:` → `:empty`

Gomb példa

```
<button
  class="
    px-4 py-2 rounded-md
    bg-sky-500 text-white font-medium

    hover:bg-sky-600
    hover:shadow-md

    focus:outline-none
    focus:ring-2 focus:ring-sky-400
    focus:ring-offset-2

    transition duration-400

    cursor-pointer
  "
>
  Kattints rám
</button>
```

- Kapjon közepes lekerekítést és némi belső kitöltést
- Ha az egeret a gomb fölé viszi a felhasználó váltson háttérszínt és kapjon közepes árnyékot
- Ha a gomb kerül fókuszba, akkor `outline` helyett árnyékkal oldja meg a gomb kiemelését. Animálható és nem mozdul el
- Az áttünések 400 ms alatt teljesüljenek
- A gombon lévő kurzor legyen a mutató ujj

Pszeudo elemek

```
<input class="placeholder:text-gray-500 placeholder:italic" ...>
```

- A placeholder legyen szürke, és dőlt

```
<p class="selection:bg-yellow-300 selection:text-yellow-900" >loerm ipsum</p>
```

- A kijelölés színe sárga lesz

---

https://tailwindcss.com/docs/hover-focus-and-other-states#pseudo-elements

Gyakrabban használt pszeudo elemek

- `before:` → `::before`
- `after:` → `::after`
- `first-letter:` → `::first-letter`
- `first-line:` → `::first-line`
- `marker:` → `::marker`
- `selection:` → `::selection`
- `file:` → `::file-selector-button`
- `placeholder:` → `::placeholder`
- `backdrop:` → `::backdrop`

Tetszőleges értékek

```
<p class="text-[red]">Lorem ipsum</p>
<p class="bg-[#660000]">Lorem ipsum</p>

<p class="mt-[1rem]">Lorem ipsum</p>
<div class="right-[68px]"></div>

<p class="bg-[green]/[28%]">Lorem ipsum</p>
```

- Egyedi értékeket adhatunk meg létező Tailwind tulajdonságoknak, szögletes zárójelek `[]` közé téve

```
<p class="text-[rgb(255_0_0)]">Lorem ipsum</p>
```

- Szóköz helyett aláhúzást kell alkalmazni

- https://tailwindcss.com/docs/adding-custom-styles#using-arbitrary-values
- https://tailwindcss.com/docs/adding-custom-styles#handling-whitespace

Tetszőleges értékek - változókkal

```
<b class="bg-[var(--strong-bg-color)] text-white">Lorem ipsum</p>
```

- Változókat is lehet használni

```
<b class="bg-(--strong-bg-color) text-white">Lorem ipsum</b>
```

- A `[]` helyett `()` használva automatikusan `var()`-ba teszi a változót

```
<p class="bg-(--primary-bg-color)/[28%]">Lorem ipsum</p>
```

- A háttérszín `--primary-bg-color` színű lesz, `72%`-ban áttetsző.

Tetszőleges tulajdonságok

```
<p class="[color:red]">Lorem ipsum</p>
<p class="[margin-top:1rem]">Lorem ipsum</p>
```

- Az osztályban szögletes zárójelet használva tetszőleges értéket lehet használni
- Használható Tailwind-ben még nem implementált tulajdonsággal is
- A IntelliSense javaslatot tesz, ha a megadott tulajdonságra már létezik saját megoldása
  - Az `[margin-top:1rem]` helyett `mt-4`-et javasolja,
  - míg a `[color:red]` helyett a `text-[red]`-at javasolja.

---

https://tailwindcss.com/docs/adding-custom-styles#arbitrary-properties

nth

```
<ul>
  <li class="nth-2:underline nth-[2n+1]:text-red-500 nth-[2n]:font-bold">Alma</li>
  <li class="nth-2:underline nth-[2n+1]:text-red-500 nth-[2n]:font-bold">Barack</li>
  <li class="nth-2:underline nth-[2n+1]:text-red-500 nth-[2n]:font-bold">Eper</li>
  <li class="nth-2:underline nth-[2n+1]:text-red-500 nth-[2n]:font-bold">Szilva</li>
</ul>
```

- A második elem (Barack) aláhúzást kap
- A páratlan elemek (Alma, Eper) piros színt kapnak
- A páros elemek (Barack, Szilva) félkövéren jelennek meg

```
<ul>
  <li class="nth-2:underline odd:text-red-500 even:font-bold">Alma</li>
  <li class="nth-2:underline odd:text-red-500 even:font-bold">Barack</li>
  <li class="nth-2:underline odd:text-red-500 even:font-bold">Eper</li>
  <li class="nth-2:underline odd:text-red-500 even:font-bold">Szilva</li>
</ul>
```

- A páros (`even`) és páratlan (`odd`) elemekre egyszerűbben is hivatkozhatunk

nth-last

```
<ul>
  <li class="border-b nth-last-[1]:border-b-0">Alma</li>
  <li class="border-b nth-last-[1]:border-b-0">Barack</li>
  <li class="border-b nth-last-[1]:border-b-0">Eper</li>
  <li class="border-b nth-last-[1]:border-b-0">Szilva</li>
</ul>
```

- Minden elem kap szegélyt, kivéve az utolsó

```
<ul>
  <li class="border-b last:border-b-0">Alma</li>
  <li class="border-b last:border-b-0">Barack</li>
  <li class="border-b last:border-b-0">Eper</li>
  <li class="border-b last:border-b-0">Szilva</li>
</ul>
```

- Az `nth-last-[1]` helyett használható a `last`
- Elhagyhatnánk `border-b`-t, de generáláskor külön odafigyelést igényelne

Dark mode

```
<div class="bg-white dark:bg-black text-black dark:text-white">
  Dark/Light
</div>
```

- A `dark:` prefix adja meg, hogy sötét módban hogy viseklkedjen az oldal
- Oda kel figyelni, hogy **mindig** legyen megfelelő kontraszt
- Tesztelni kell
- Alapértelmezetten a `prefers-color-scheme` media query alapján határozza meg, azaz OS / böngésző beállításai szerint

Dark mode kézzel

```
@custom-variant dark (&:where(.dark, .dark *));
```

- Alapvetően a `prefers-color-scheme` media query határozza meg, hogy az oldal dark mode-ban van vagy sem
- A stíluslapra helyezve már a `dark` osztályt keresi

```
<html class="dark"></html>
```

- A `html` elemre kiadva az egész oldalra érvényesül
- A `dark` osztály hiányában light mode-ban mutatja az oldalt
- Könnyen tesztelhető

@theme

```
@theme{
    --color-primary: #2B668C;
}
```

- A Tailwind specifikus `@theme` direktíva több utility osztályt is legenerál

```
<div class="bg-primary text-white">Lorem Ipsum</div>
<div class="bg-white text-primary">Lorem Ipsum</div>
<div class="border border-primary m-2 p-2">Hello</div>
```

- A `--color*` névtérben lévő változók létrehozhatnak háttérszínt, szövegszínt, szegély színt, stb…
- https://tailwindcss.com/docs/theme
- https://tailwindcss.com/docs/theme#theme-variable-namespaces
- https://tailwindcss.com/docs/theme#default-theme-variable-reference

@theme vs :root

Tailwind

```
@theme{
    --color-primary: #2B668C;
}
```

```
<div class="bg-[var(--color-primary)]"></div>
```

- Létrehozz a CSS változót **is** a megadott néven

CSS

```
:root{
    --color-primary: #2B668C;
}
```

```
<div class="bg-[var(--color-primary)]"></div>
```

- A `@theme` helyett a `:root`-ban megadva csak a változót hozza létre, a további osztályokat nem