Tailwind [v4.1]

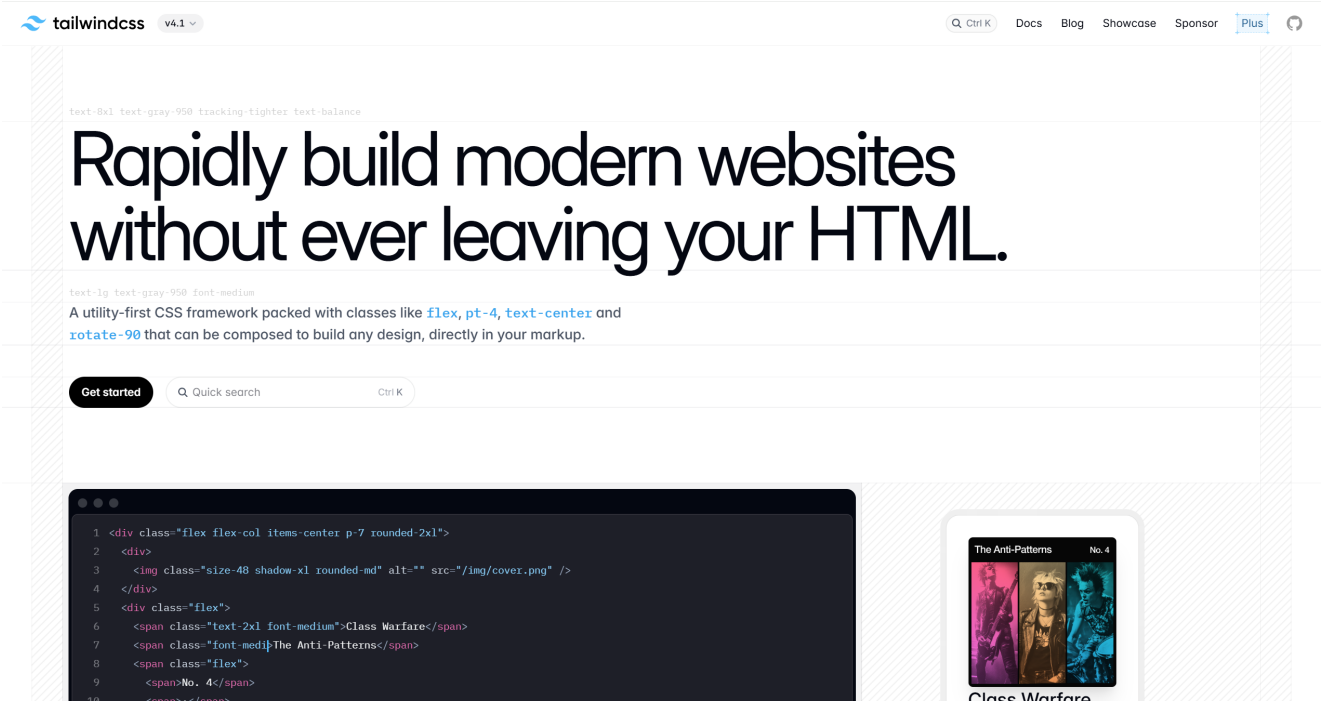
作者： -- 天气预报

日期： -- 2025-09-20

01. 官网

> https://tailwindcss.com/

- ui



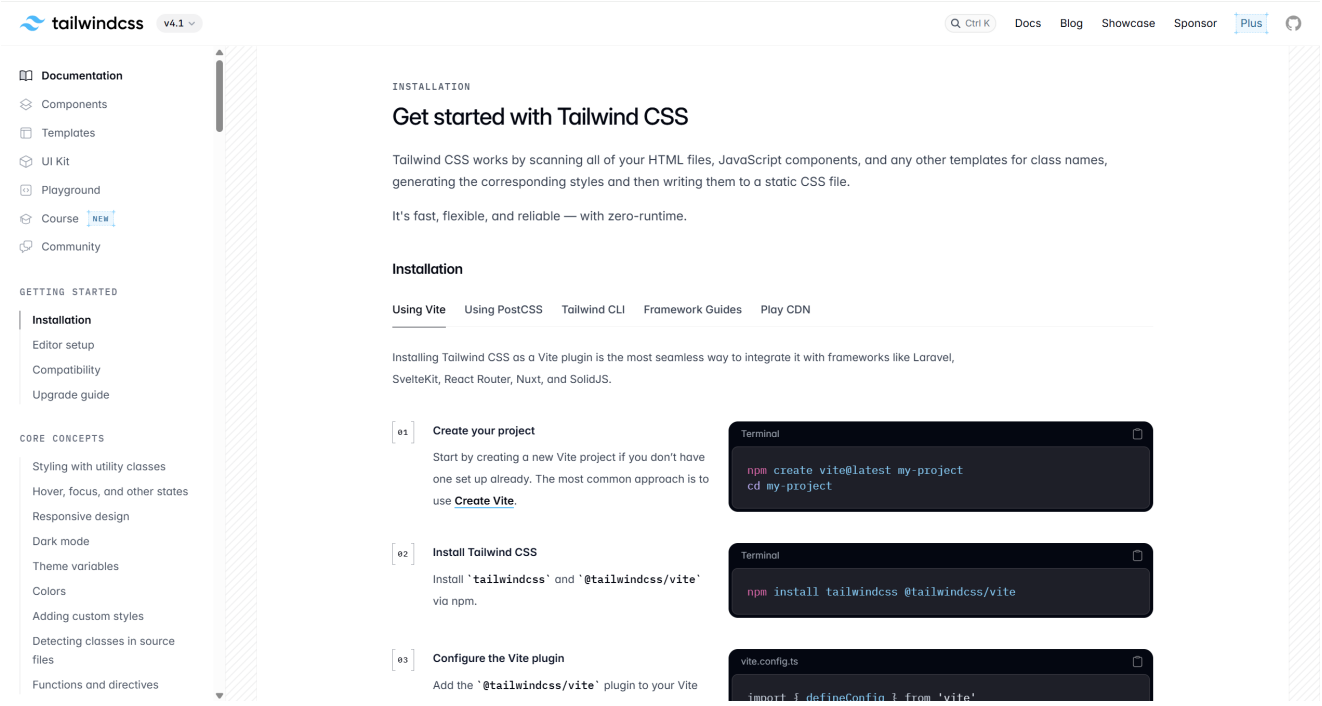
> **Note**

> ☑ Tailwind （v4.1）推荐新项目直接使用即可

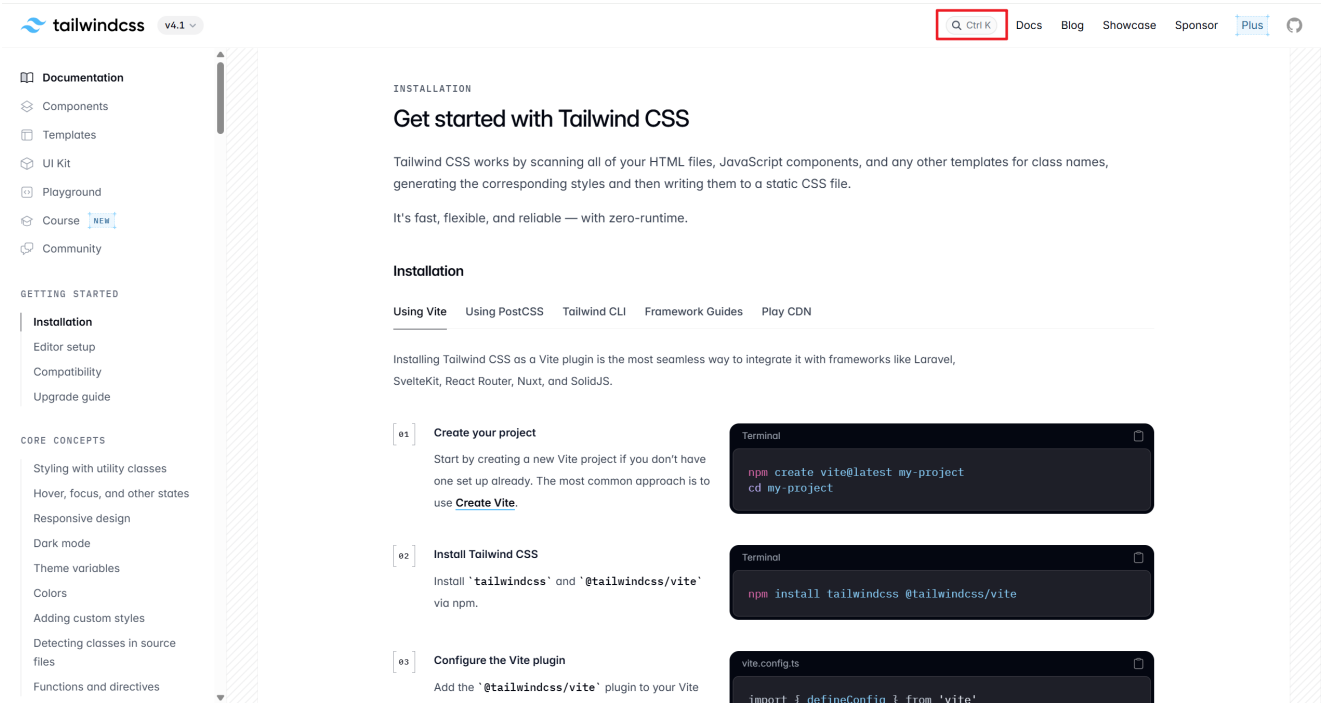
1.1 文档

> https://tailwindcss.com/docs/installation/using-vite

- ui



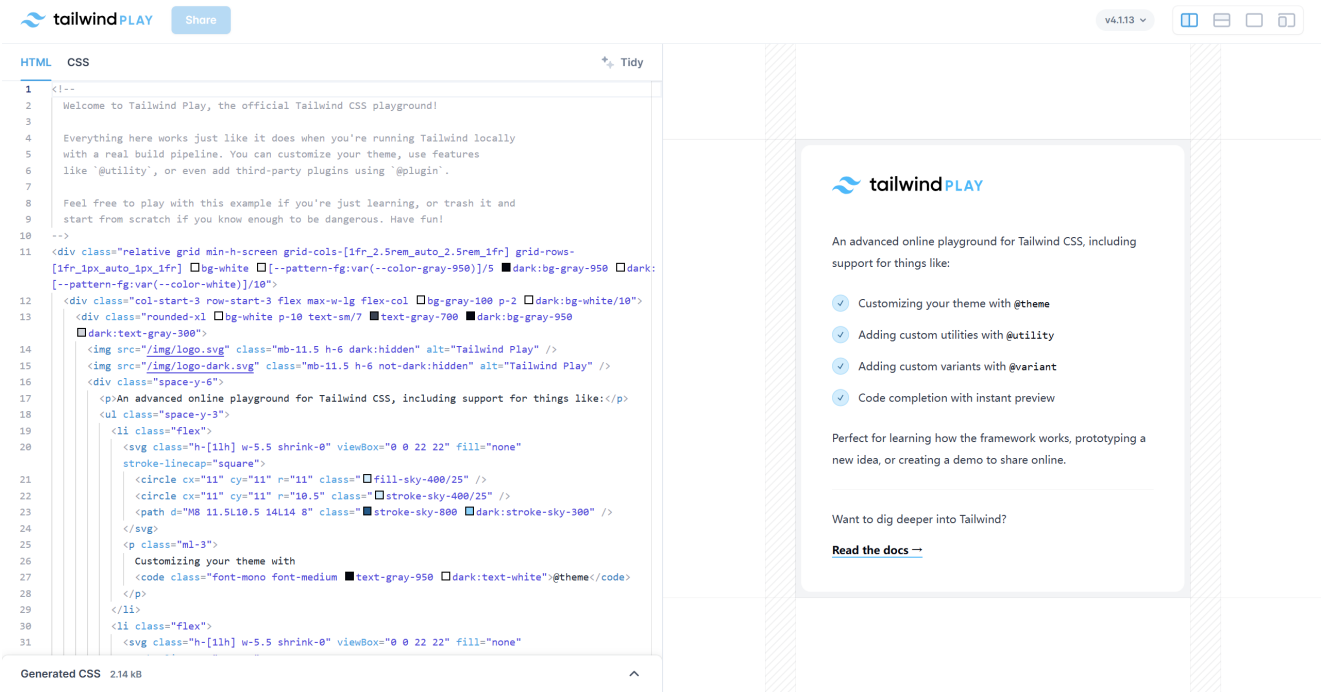
- ui



1.2 在线运行

> https://play.tailwindcss.com/

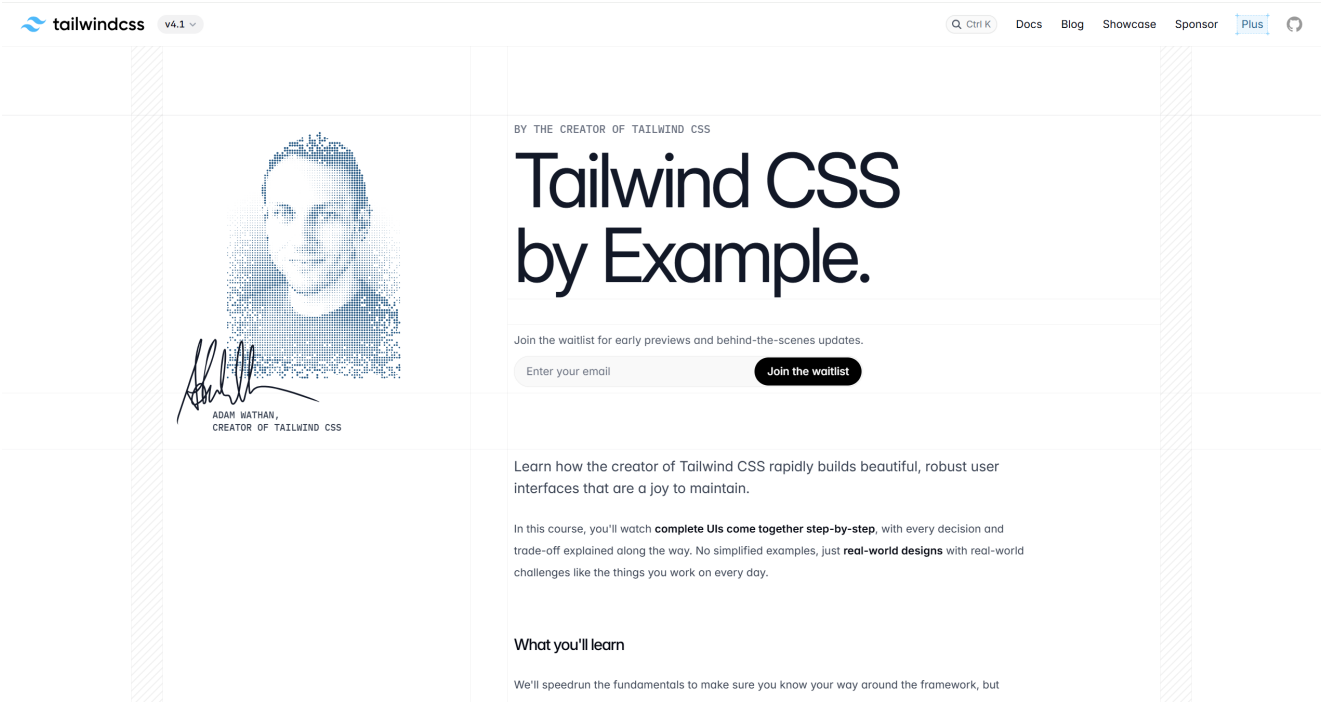
- ui



1.3 福利课程

> https://tailwindcss.com/course

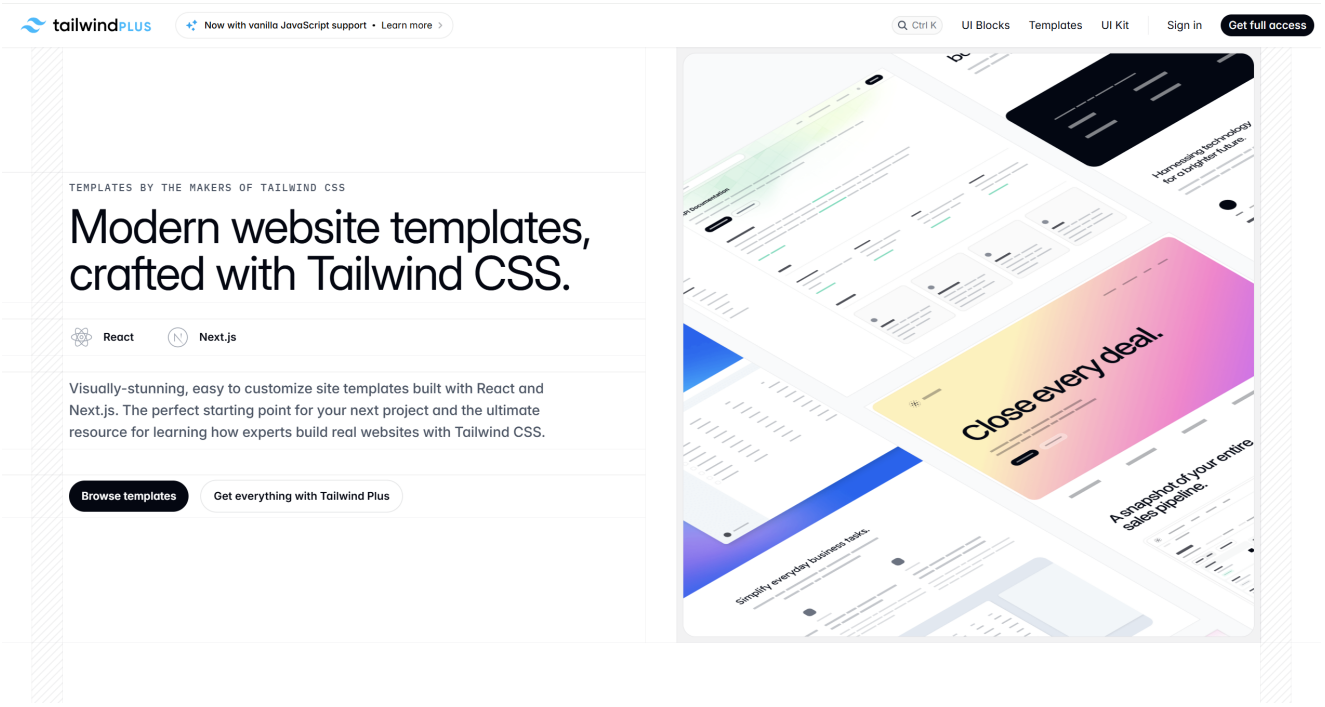
- [x] 邮件报名、邮件推送油管视频［科学上网］（* 小课程、分享特殊场景案例实现）



1.4 模版服务

> https://tailwindcss.com/plus/templates?ref=sidebar

- ui

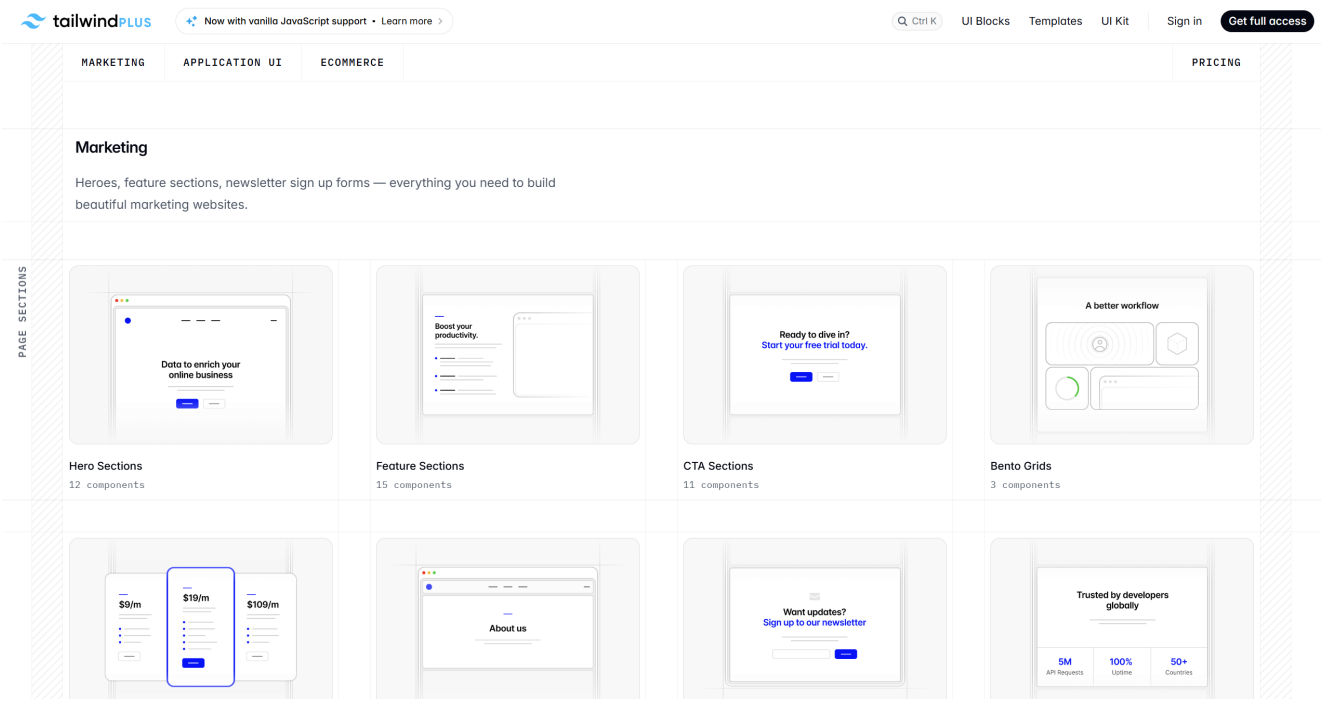


1.5 组件

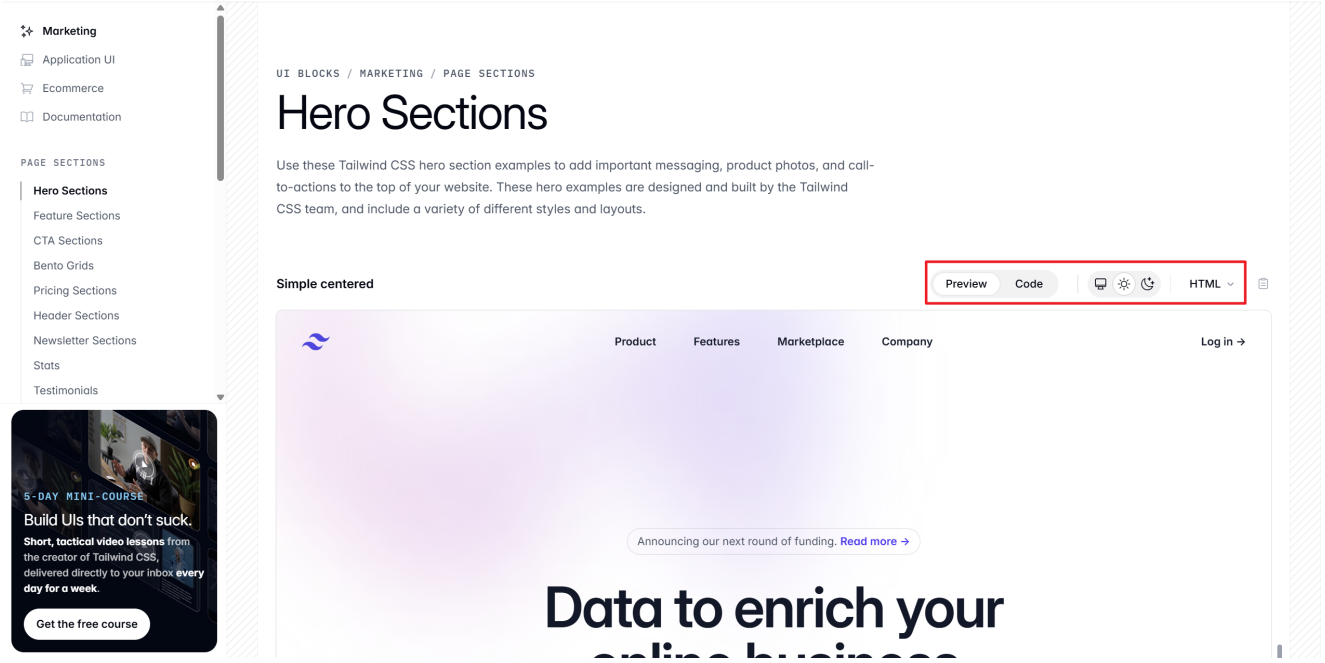
> https://tailwindcss.com/plus/ui-blocks?ref=sidebar

☑ 免费

- ui



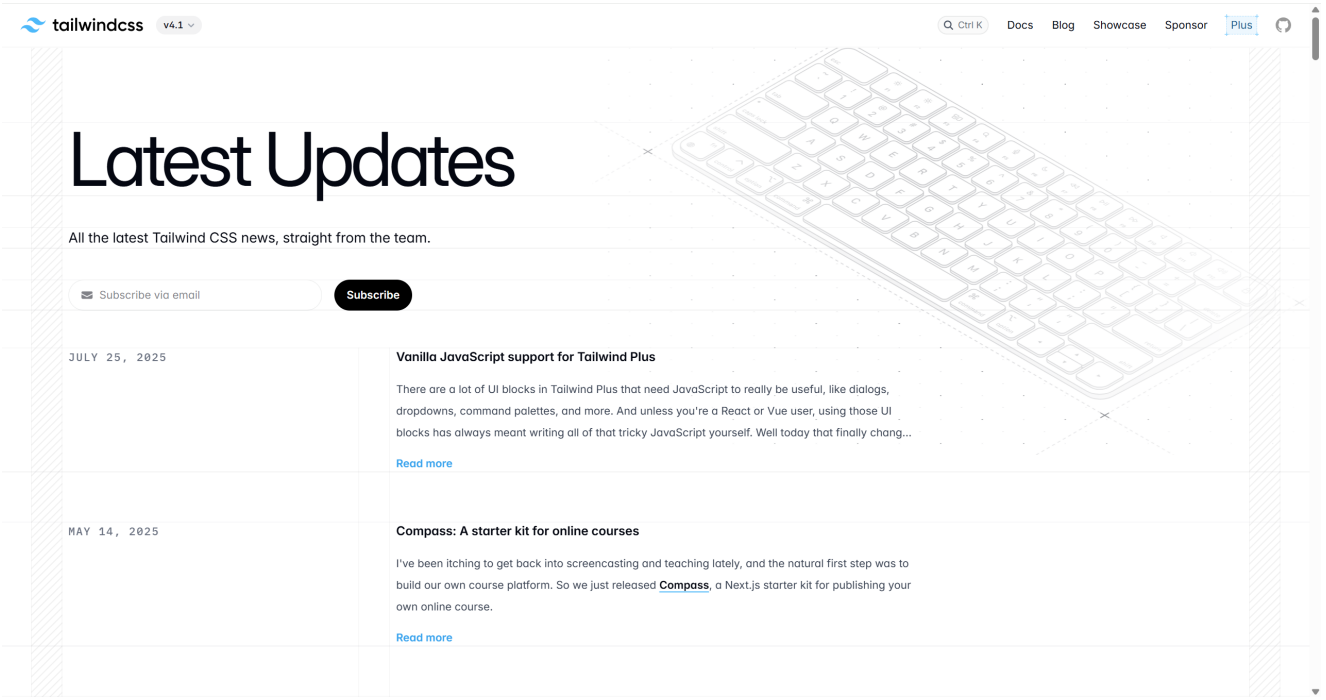
- ui



1.6 博客

> https://tailwindcss.com/blog

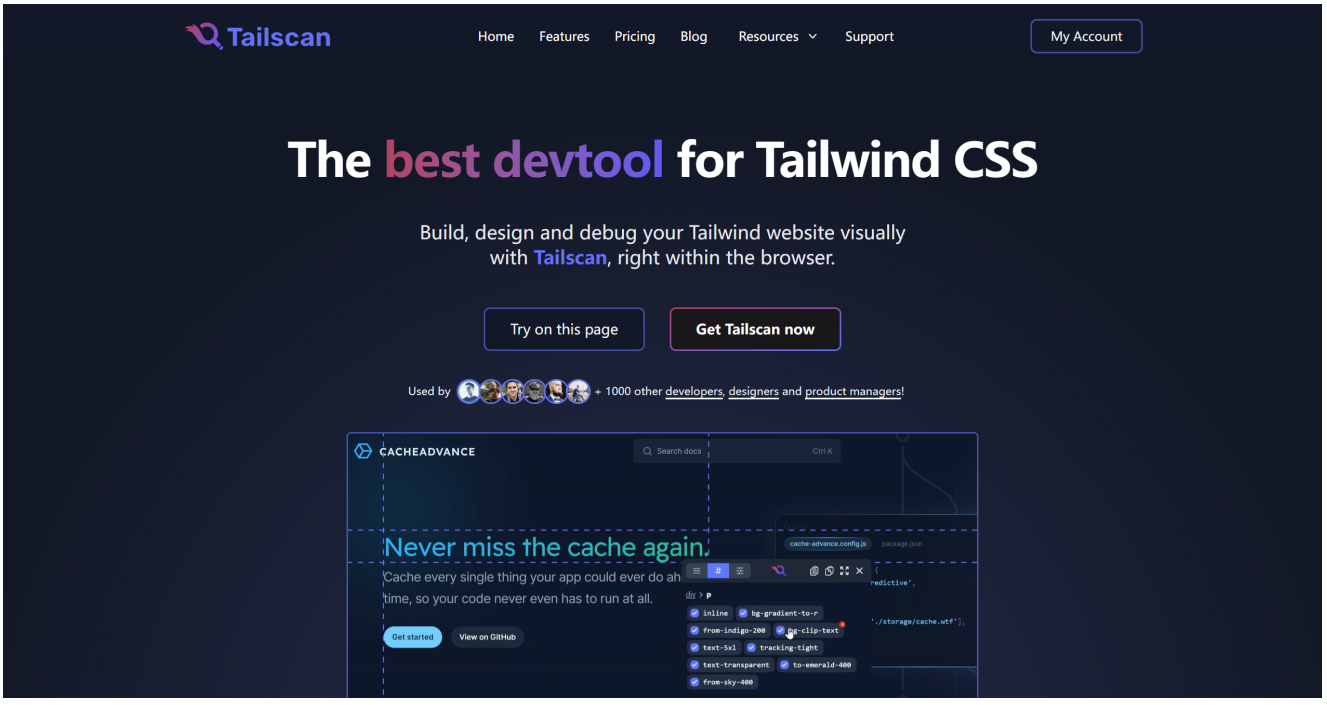
- ui



1.7 插件

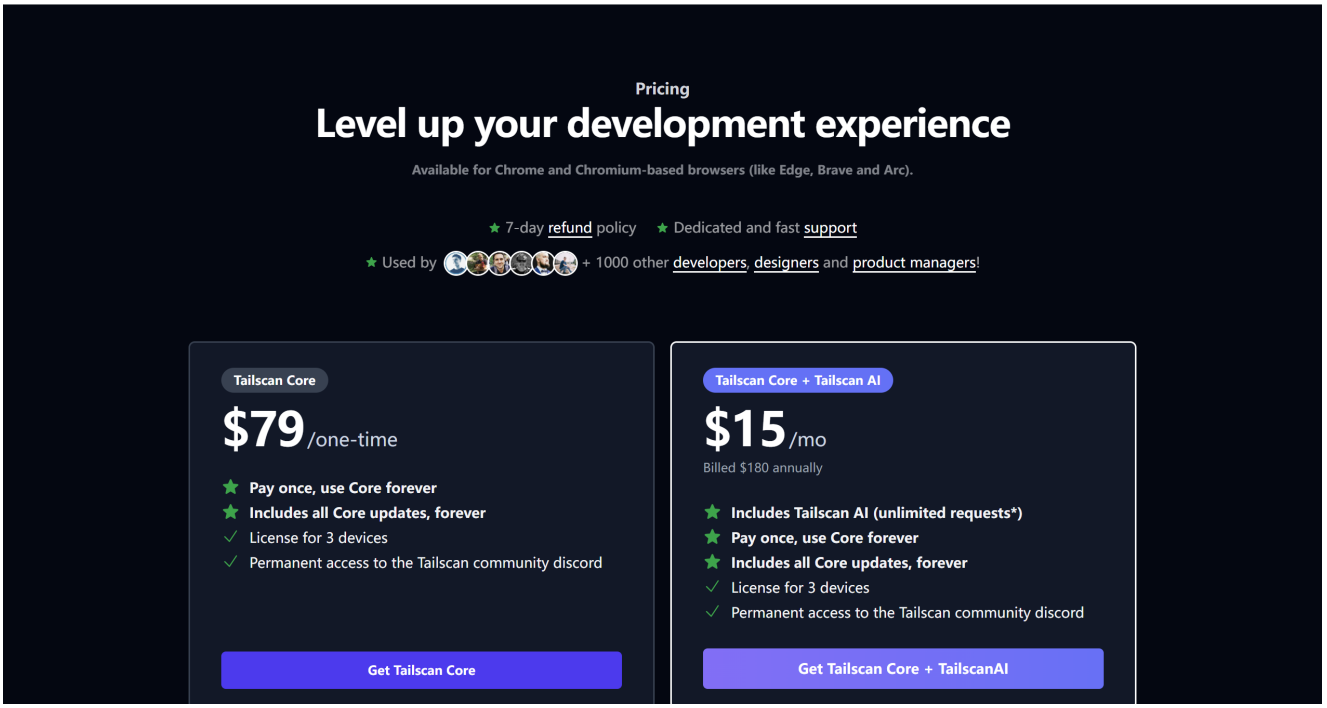
> https://tailscan.com/

- ui

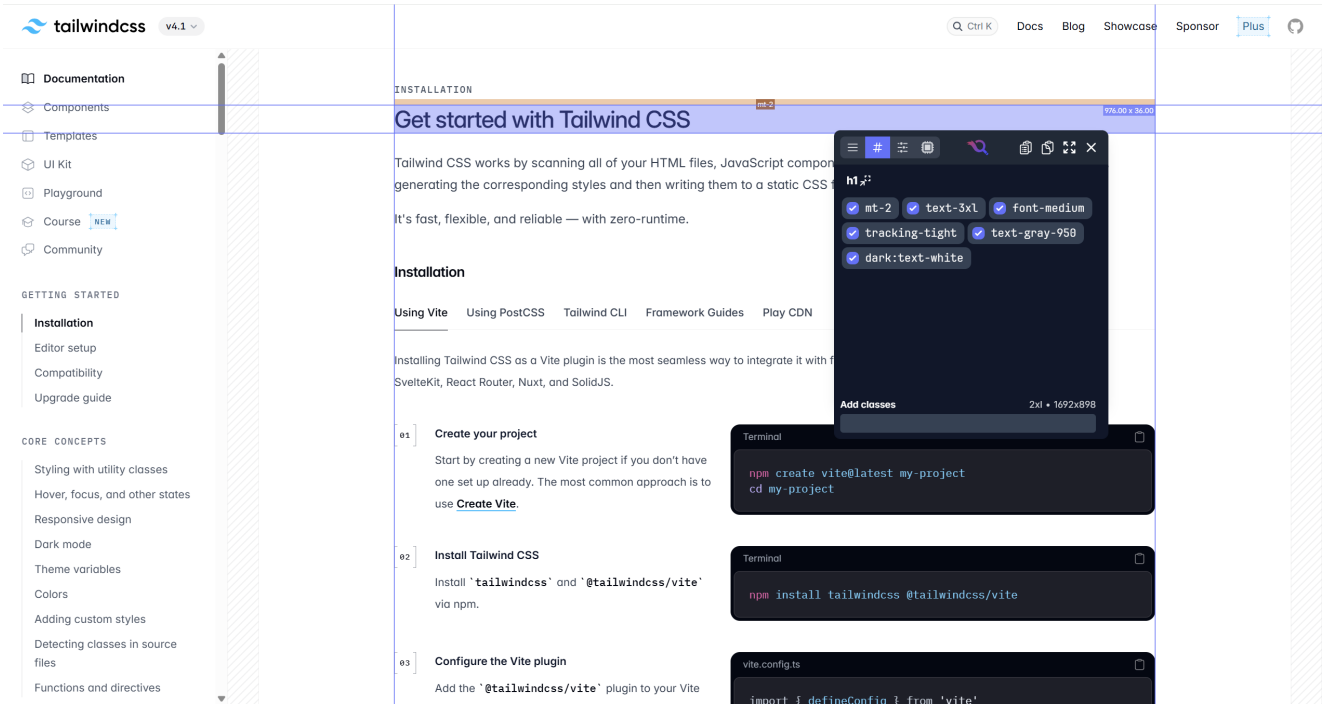


> Tailscan Core | Tailscan Core + AI

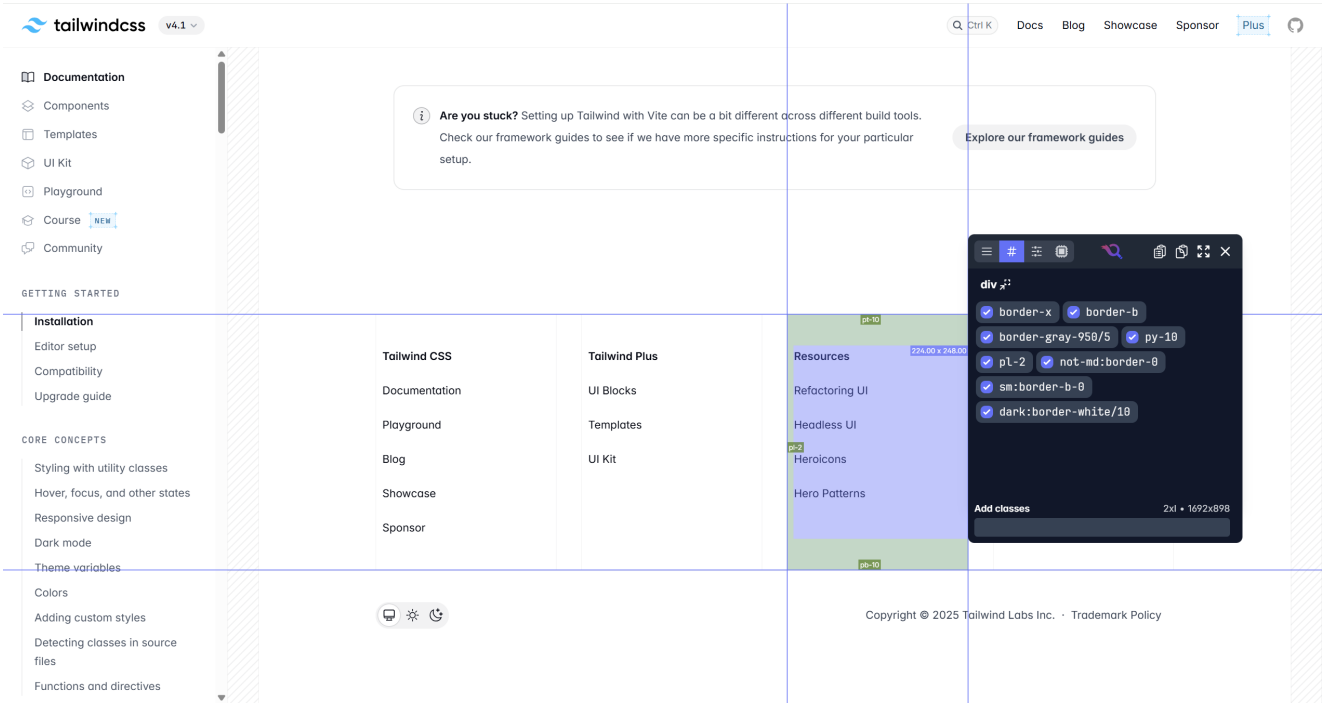
- ui



- ui



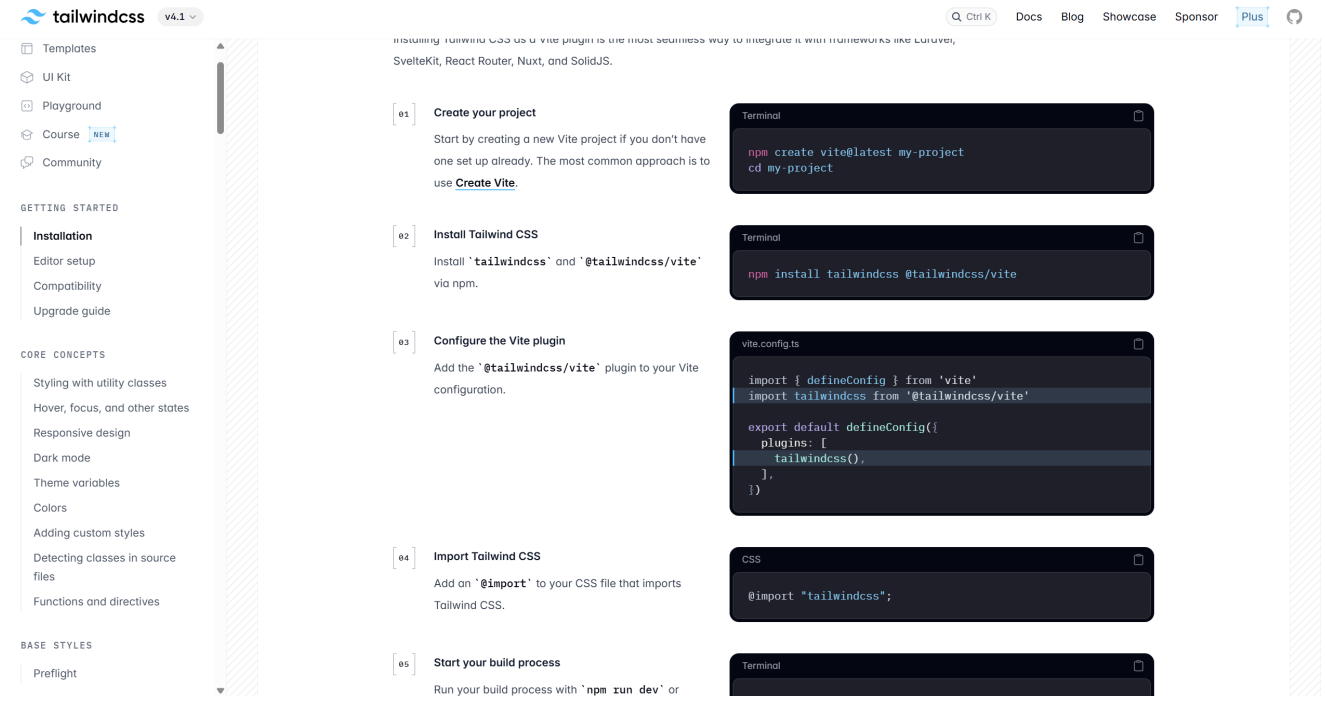
- ui



02. 环境

> https://tailwindcss.com/docs/installation/using-vite

- ui



2.1 CDN

> https://tailwindcss.com/docs/installation/play-cdn

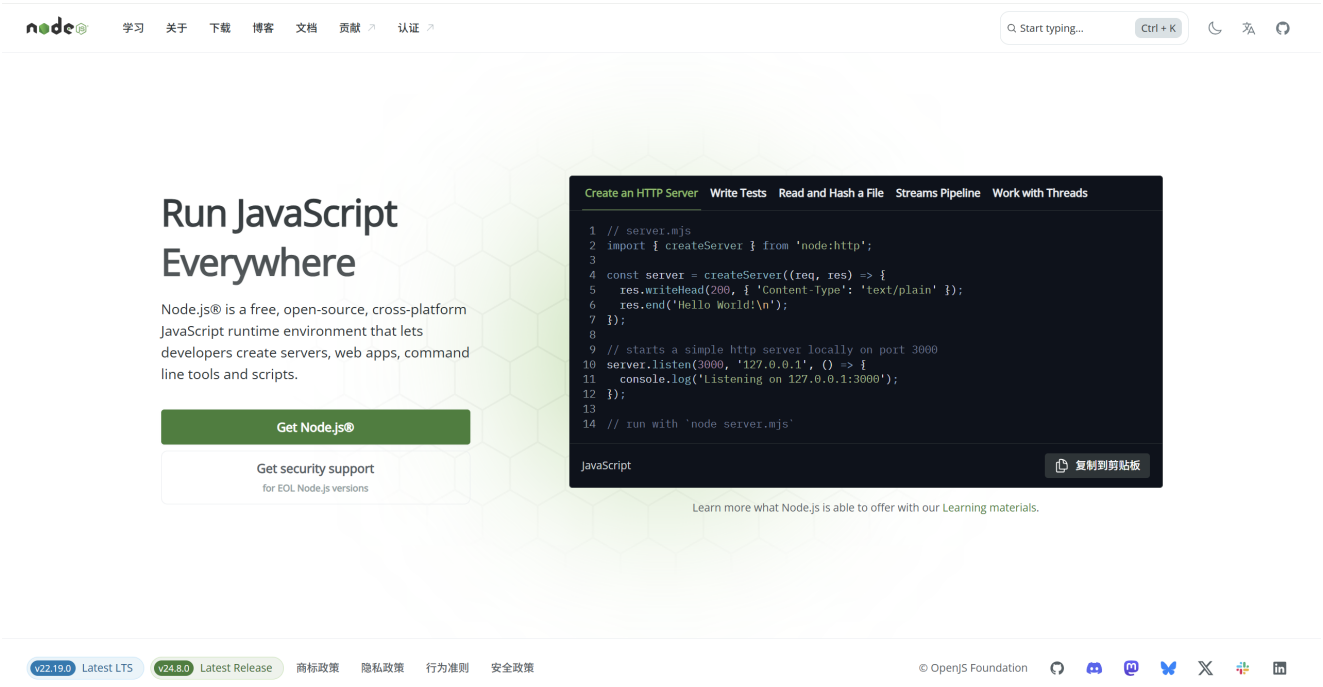
- **index.html**

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Tailwind</title>
        <script src="https://cdn.jsdelivr.net/npm/@tailwindcss/browser@4">
</script>
        <style type="text/tailwindcss">
            @theme {
                --color-primary: #000;
            }
        </style>
    </head>
    <body>
        <div class="p-10 bg-primary">
            <h1 class="text-2xl text-center text-white">tailwind</h1>
        </div>
    </body>
</html>
```

2.2 Tailwind CLI

> https://nodejs.org/zh-cn

- **ui**



> https://tailwindcss.com/docs/installation/tailwind-cli

- **npm**

```
npm init --yes

npm install tailwindcss @tailwindcss/cli
```

- **src/input.css**

```
@import "tailwindcss";
```

- **src/index.html**

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Tailwind</title>
        <link rel="stylesheet" href="./output.css">
    </head>
    <body>
        <div class="p-10 bg-rose-500">
            <h1 class="text-2xl text-center text-white">tailwind</h1>
        </div>
    </body>
</html>
```

- **npx**

```
npx @tailwindcss/cli -i ./src/input.css -o ./src/output.css --watch
```

2.3 Next

> https://nextjs.org/

☑ 内置安装引导支持、仅需创建项目时选中即可

- **npm**

```
npx create-next-app@latest
```

- **console**

```
# 项目名称
1. What is your project named? [name]

# 是否安装 TypeScript
2. Would you like to use TypeScript? [No / Yes]

# 是否安装 ESLint 代码检查工具
3. Which linter would you like to use? [ESLint / Biome / None]

# 是否安装 Tailwind
4. Would you like to use Tailwind CSS? No / Yes

# 是否需要 src 作为项目源代码存放目录
5. Would you like your code inside a `src/` directory? [No / Yes]

# 是否安装 文件路由系统
6. Would you like to use App Router? (recommended) [No / Yes]

# 是否安装 Turbopack 打包构建工具
7. Would you like to use Turbopack? (recommended) [No / Yes]

# 是否配置 @/* 别名 模块导入方式
8. Would you like to customize the import alias (`@/*` by default)? [No / Yes]

# 配置别名 是否使用 @/*
9. What import alias would you like configured? @/*
```

- **page.tsx**

```
export default function Page() {
    return (
        <div className="p-10 bg-rose-500">
            <h1 className="text-2xl text-white text-center">tailwind</h1>
        </div>
    )
}
```

2.4 Vite React

> 请认真学习本节示例过程、后续示例基于此环境、本环境无须掌握 React

2.4.1 init

> npm init --yes

- **package.json**

```
{
    "name": "tailwind",
    "version": "1.0.0",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "description": "",
    "type": "module"
}
```

2.4.2 react

> npm install react react-dom

- **package.json**

```
{
    "name": "tailwind",
    "version": "1.0.0",
    "main": "index.js",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "description": "",
    "type": "module",
    "dependencies": {
        "react": "^19.1.1",
        "react-dom": "^19.1.1"
    }
}
```

2.4.3 vite

```
npm install vite -D
```

- **package.json**

```
{
    "name": "tailwind",
    "version": "1.0.0",
    "main": "index.js",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "description": "",
    "type": "module",
    "dependencies": {
        "react": "^19.1.1",
        "react-dom": "^19.1.1"
    },
    "devDependencies": {
        "vite": "^7.1.6"
```

```
    }
}
```

2.4.4 plugin-react

> https://cn.vite.dev/plugins/

- ✅ npm install @vitejs/plugin-react -D
- ✅ npm install @vitejs/plugin-react-swc -D [https://swc.rs/]

- **package.json**

```
{
    "name": "tailwind",
    "version": "1.0.0",
    "main": "index.js",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "description": "",
    "type": "module",
    "dependencies": {
        "react": "^19.1.1",
        "react-dom": "^19.1.1"
    },
    "devDependencies": {
        "@vitejs/plugin-react-swc": "^4.1.0",
        "vite": "^7.1.6"
    }
}
```

2.4.5 vite.config.js

> 项目根目录创建 vite.config.js 文件

- **vite.config.js**

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react-swc";

export default defineConfig({
    plugins: [
        react(),
    ]
})
```

2.4.6 index.html

> 项目根目录创建文件 index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>React</title>
    </head>
    <body>
        <div id="root"></div>
        <script type="module" src="./src/main.jsx"></script>
    </body>
</html>
```

2.4.7 src/main.jsx

> 项目根目录创建目录及文件 src/app.jsx | src/main.jsx

- **src/app.jsx**

```jsx
export function App() {
    return (
        <div>react</div>
    )
}
```

- **src/main.jsx**

```jsx
import { createRoot } from "react-dom/client";
import { App } from "./app.jsx";

const root = document.getElementById("root");

createRoot(root).render(<App/>);
```

- **package.json**

```json
{
    "name": "tailwind",
    "version": "1.0.0",
    "scripts": {
        "dev": "vite dev"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "description": "",
    "type": "module",
```

```
    "dependencies": {
        "react": "^19.1.1",
        "react-dom": "^19.1.1"
    },
    "devDependencies": {
        "@vitejs/plugin-react-swc": "^4.1.0",
        "vite": "^7.1.6"
    }
}
```

- **npm**

```
npm run dev # http://localhost:5173/
```

2.4.8 tailwind

> npm install tailwindcss @tailwindcss/vite -D

- **package.json**

```
{
    "name": "tailwind",
    "version": "1.0.0",
    "scripts": {
        "dev": "vite dev"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "description": "",
    "type": "module",
    "dependencies": {
        "react": "^19.1.1",
        "react-dom": "^19.1.1"
    },
    "devDependencies": {
        "@tailwindcss/vite": "^4.1.13",
        "@vitejs/plugin-react-swc": "^4.1.0",
```

```
        "tailwindcss": "^4.1.13",
        "vite": "^7.1.6"
    }
}
```

- **vite.config.js**

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react-swc";
import tailwindcss from "@tailwindcss/vite";

export default defineConfig({
    plugins: [
        react(),
        tailwindcss(),
    ]
})
```

- **src/globals.css**

```
@import "tailwindcss";
```

- **src/main.jsx**

```
import { createRoot } from "react-dom/client";
import { App } from "./app.jsx";
import "./globals.css";

const root = document.getElementById("root");
createRoot(root).render(<App/>)
```

- **src/app.jsx**

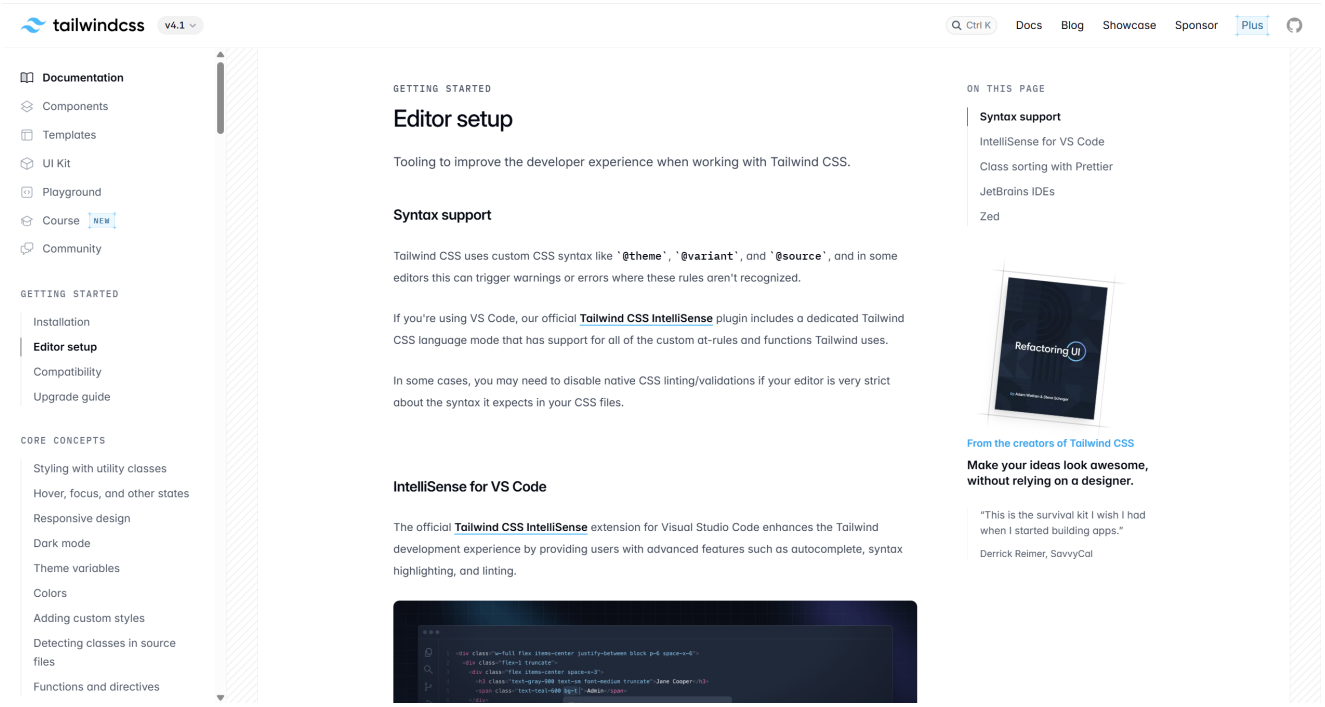
```jsx
export function App() {
    return (
        <div className="p-32 bg-rose-500">
            <h1 className="font-mono text-center text-white">
                <span className="text-4xl md:text-6xl lg:text-7xl duration-300">
                    tailwind
                </span>
            </h1>
        </div>
    )
}
```

03. 开发工具

> https://tailwindcss.com/docs/editor-setup

3.1 IDE

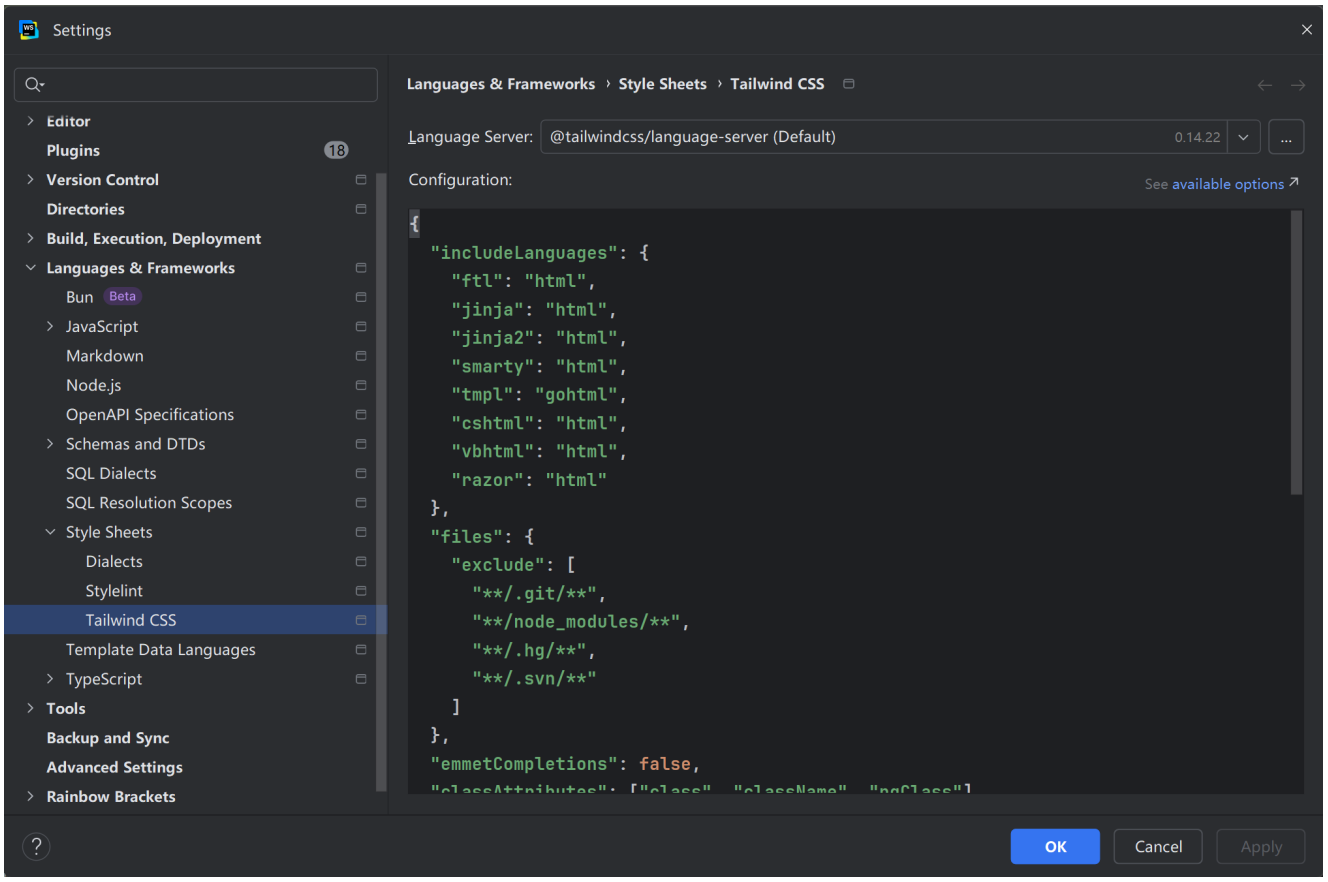
- **ui**



WebStorm 内置 Tailwind Language Server 语言服务引擎

- ☑ 推荐最新版本：https://www.jetbrains.com/webstorm/download/other.html
- ☑ 引擎自动开启、无需其它配置、可自动提示 Tailwind、可检测类选择器

- **ui**

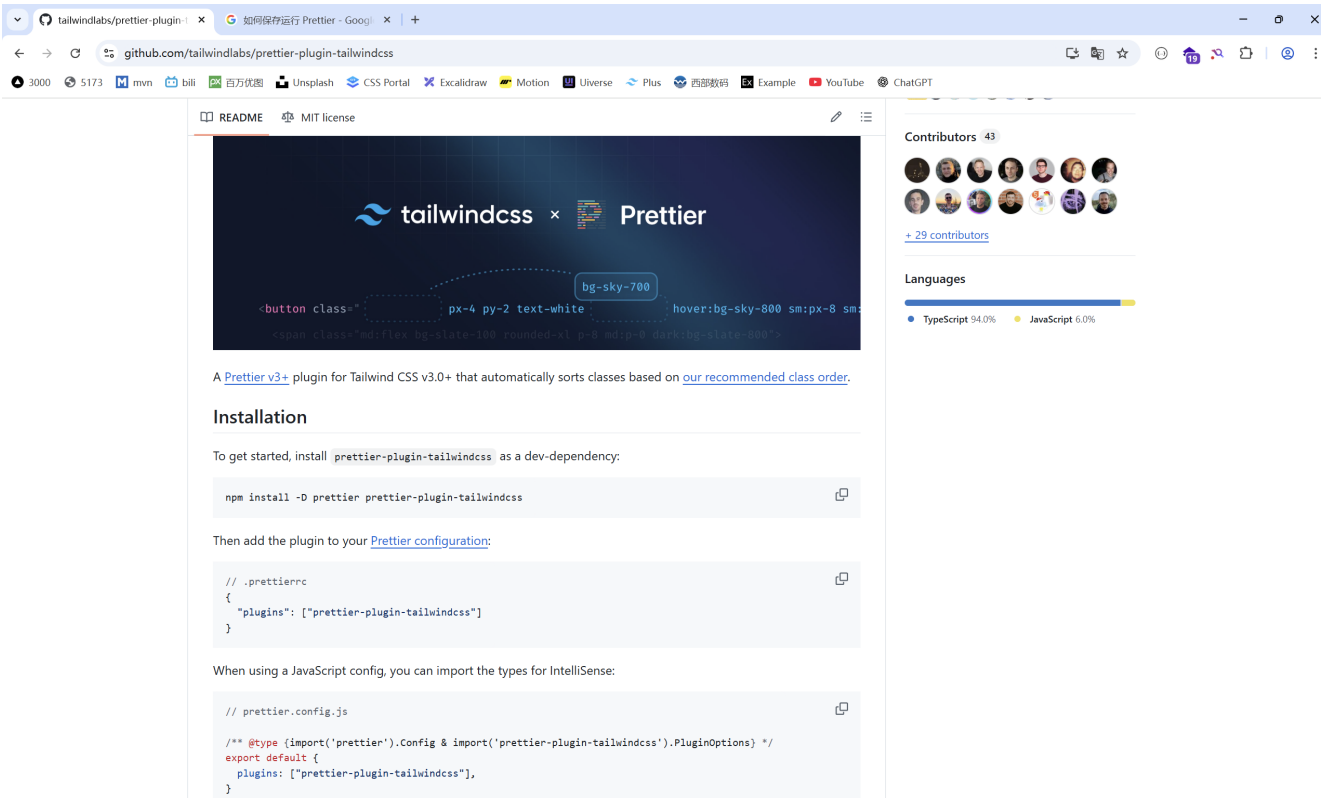


- **ui**

3.2 格式化

> https://github.com/tailwindlabs/prettier-plugin-tailwindcss

- ui



- npm

```
npm install prettier prettier-plugin-tailwindcss -D
```

- prettier.config.js

```
export default {
    plugins: ["prettier-plugin-tailwindcss"],
}
```

- **npx**

```
npx prettier . --write
```

> 推荐人为控制书写顺序 [位置 | 盒子模型 | 排版 | 视觉 | 其它]

- **src/app.jsx**

```
export function App() {
    return (
        <div className="p-32 bg-slate-200">
            <div className="relative size-48 bg-rose-500">
                <div className="absolute left-1/2 top-1/2 -translate-1/2
                                size-1/2
                                bg-purple-700
                                animate-spin"/>
            </div>
        </div>
    )
}
```

> **Note**
>
> ☑ https://codeguide.bootcss.com/

- **ui**

声明顺序

相关的属性声明应当归为一组，并按照下面的顺序排列：

1. Positioning
2. Box model
3. Typographic
4. Visual
5. Misc

由于定位（positioning）可以从正常的文档流中移除元素，并且还能覆盖盒模型（box model）相关的样式，因此排在首位。盒模型排在第二位，因为它决定了组件的尺寸和位置。

其他属性只是影响组件的 *内部* 或者是不影响前两组属性，因此排在后面。

完整的属性列表及其排列顺序请参考 Bootstrap property order for Stylelint。

```
.declaration-order {
  /* Positioning */
  position: absolute;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  z-index: 100;

  /* Box-model */
  display: block;
  float: right;
  width: 100px;
  height: 100px;

  /* Typography */
  font: normal 13px "Helvetica Neue", sans-serif;
  line-height: 1.5;
  color: #333;
  text-align: center;

  /* Visual */
  background-color: #f5f5f5;
  border: 1px solid #e5e5e5;
  border-radius: 3px;

  /* Misc */
  opacity: 1;
}
```

3.3 兼容性

> https://tailwindcss.com/docs/compatibility

- ui

GETTING STARTED

Compatibility

Learn about browser support and compatibility with other tooling.

Browser support

Tailwind CSS v4.0 is designed for and tested on modern browsers, and the core functionality of the framework specifically depends on these browser versions:

- **Chrome 111** *(released March 2023)*
- **Safari 16.4** *(released March 2023)*
- **Firefox 128** *(released July 2024)*

Tailwind also includes support for many bleeding-edge platform features like `field-sizing: content`, `@starting-style`, and `text-wrap: balance` that have limited browser support. It's up to you if you want to use these modern features in your projects — if the browsers you're targeting don't support them, simply don't use those utilities and variants.

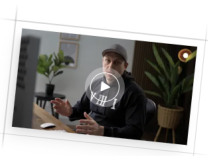
If you're unsure about the support for a modern platform feature, the Can I use database is a great resource.

Sass, Less, and Stylus

Tailwind CSS v4.0 is a full-featured CSS build tool designed for a specific workflow, and is not designed to be used with CSS preprocessors like Sass, Less, or Stylus

ON THIS PAGE

- Browser support
- Sass, Less, and Stylus
  - Build-time imports
  - Variables
  - Nesting
  - Loops
  - Color and math functions
- CSS modules
  - Scoping concerns
  - Performance
  - Explicit context sharing
- Vue, Svelte, and Astro



5-day mini-course

Build UIs that don't suck.

Short, tactical video lessons from the creator of Tailwind CSS, delivered directly to your inbox every day for a week.

Get the free course →

04. 样式初始化

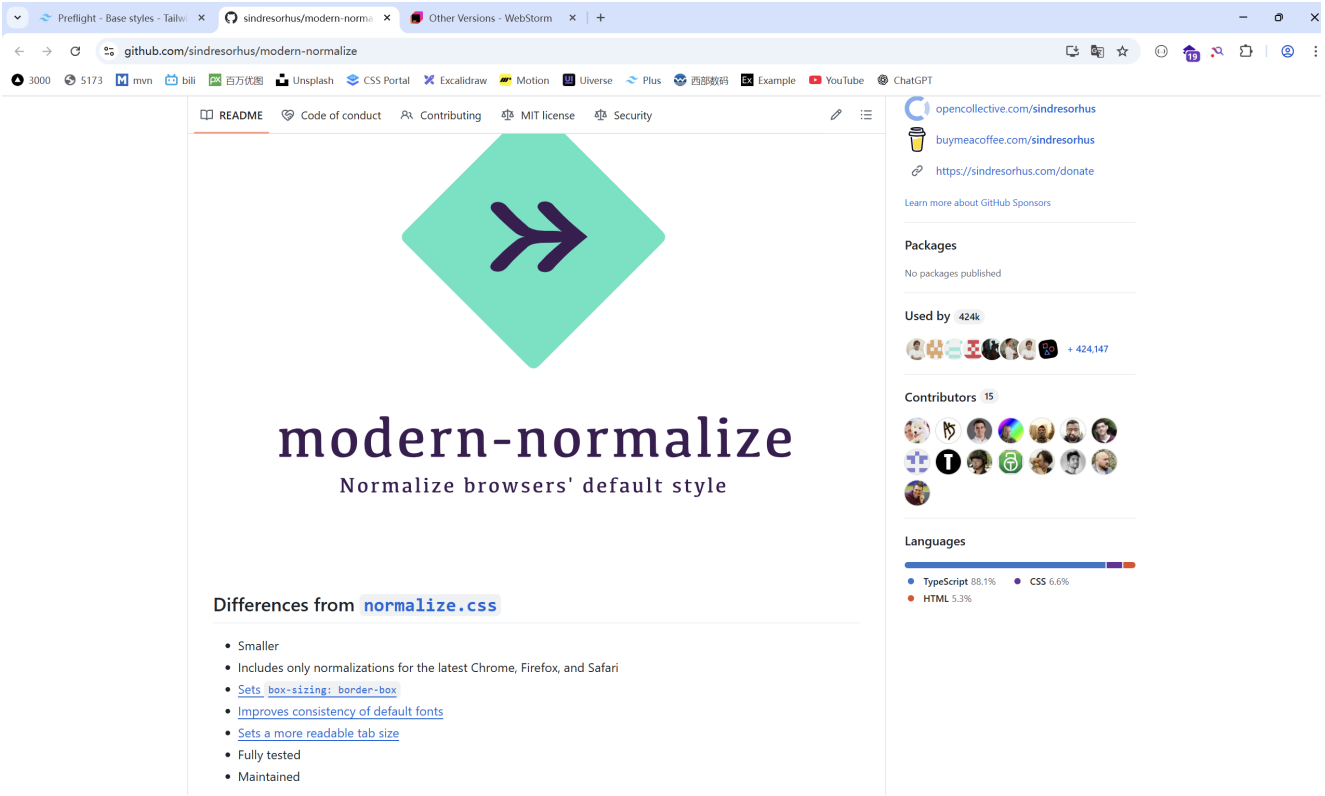
> https://tailwindcss.com/docs/preflight

4.1 modern-normaliz

> https://github.com/sindresorhus/modern-normalize

- ☑ Tailwind 使用 modern-normaliz 初始化 HTML 及常用元素样式初始化
- ☑ node_modules/tailwindcss/preflight.css

- ui



- **preflight.css**

```css
*,
::after,
::before,
::backdrop,
::file-selector-button {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
    border: 0 solid;
}

h1,
h2,
h3,
h4,
h5,
h6 {
    font-size: inherit;
    font-weight: inherit;
}

img,
svg,
video,
canvas,
audio,
iframe,
embed,
object {
    display: block; /* 1 */
    vertical-align: middle; /* 2 */
}
```

4.2 @layer

> Tailwind 内置 theme | base | components | utilities 分层全局管理样式

- **globals.css**

```
@import "tailwindcss";
```

- **globals.css**

```
@layer theme, base, components, utilities;

@import "tailwindcss/theme.css" layer(theme);
@import "tailwindcss/preflight.css" layer(base);
@import "tailwindcss/utilities.css" layer(utilities);
```

4.3 @layer theme

> 主题层定义 Tailwind 内置各种全局变量、颜色、动画、间距 等 ...

- **node_modules/tailwindcss/theme.css**

```
@theme default {

    --color-red-50: oklch(97.1% 0.013 17.38);
    --color-red-100: oklch(93.6% 0.032 17.717);
    --color-red-200: oklch(88.5% 0.062 18.334);
    --color-red-300: oklch(80.8% 0.114 19.571);
    --color-red-400: oklch(70.4% 0.191 22.216);
    --color-red-500: oklch(63.7% 0.237 25.331);
    --color-red-600: oklch(57.7% 0.245 27.325);
    --color-red-700: oklch(50.5% 0.213 27.518);
    --color-red-800: oklch(44.4% 0.177 26.899);
    --color-red-900: oklch(39.6% 0.141 25.723);
    --color-red-950: oklch(25.8% 0.092 26.042);

    --blur-xs: 4px;
    --blur-sm: 8px;
    --blur-md: 12px;
    --blur-lg: 16px;
```

```
    --blur-xl: 24px;
    --blur-2xl: 40px;
    --blur-3xl: 64px;

    --spacing: 0.25rem; // html 16px * 0.25 ⟹ 4px
                        //              w-5 ⟹ width: 20px
                        //              p-5 ⟹ padding: 20px

    @keyframes spin {
        to {
            transform: rotate(360deg);
        }
    }

    --animate-spin: spin 1s linear infinite;

    ...

}
```

- **node_modules/tailwindcss/index.css**

```
@layer theme, base, components, utilities;

@layer theme {
    @theme default {
        ...
        --color-red-100: oklch(93.6% 0.032 17.717);
        --color-red-200: oklch(88.5% 0.062 18.334);
        --color-red-300: oklch(80.8% 0.114 19.571);
        ...
    }
}
```

- **globals.css**

```
@import "tailwindcss";

@theme {
    --font-serif: ui-sans-serif, system-ui, sans-serif;
}
```

4.4 @layer base

> 完成 HTML 样式初始化、全局样式默认内外边距、盒子模型 等 ...

- **node_modules/tailwindcss/index.css**

```
@layer theme, base, components, utilities;

@layer theme {
    ...
}

@layer base {
    *,
    ::after,
    ::before,
    ::backdrop,
    ::file-selector-button {
        box-sizing: border-box; /* 1 */
        margin: 0; /* 2 */
        padding: 0; /* 2 */
        border: 0 solid; /* 3 */
    }

    ...
}
```

- **globals.css**

```
@import "tailwindcss";

@layer base {
    h1 {
        font-size: 36px;
    }
}
```

4.5 @layer utilities

> utilities 表示内置程序类 p-5 | pt-2 | text-white 等 ... [通过解析生成]

☑ @layer utilities 应当定义 Tailwind 还没有内置的实用程序类

- **node_modules/tailwindcss/index.css**

```
@layer theme, base, components, utilities;

@layer theme {
    ...
}

@layer base {
    *,
    ::after,
    ::before,
    ::backdrop,
    ::file-selector-button {
        box-sizing: border-box; /* 1 */
        margin: 0; /* 2 */
        padding: 0; /* 2 */
        border: 0 solid; /* 3 */
    }

    ...
}
```

```
@layer utilities {
    @tailwind utilities;
}
```

- **node_modules/tailwindcss/utilities.css**

```
@tailwind utilities;
```

- **globals.css**

```
@import "tailwindcss";

@utility animmate-paused {
    animation-play-state: paused;
}

@utility animmate-delay-1500 {
    animation-delay: 3000ms;
}

@utility animmate-delay-3000 {
    animation-delay: 3000ms;
}
```

4.5 @layer components

> @layer components {} 层是预留给开发者编写的自定义类、在入口文件编写

- **src/globals.css**

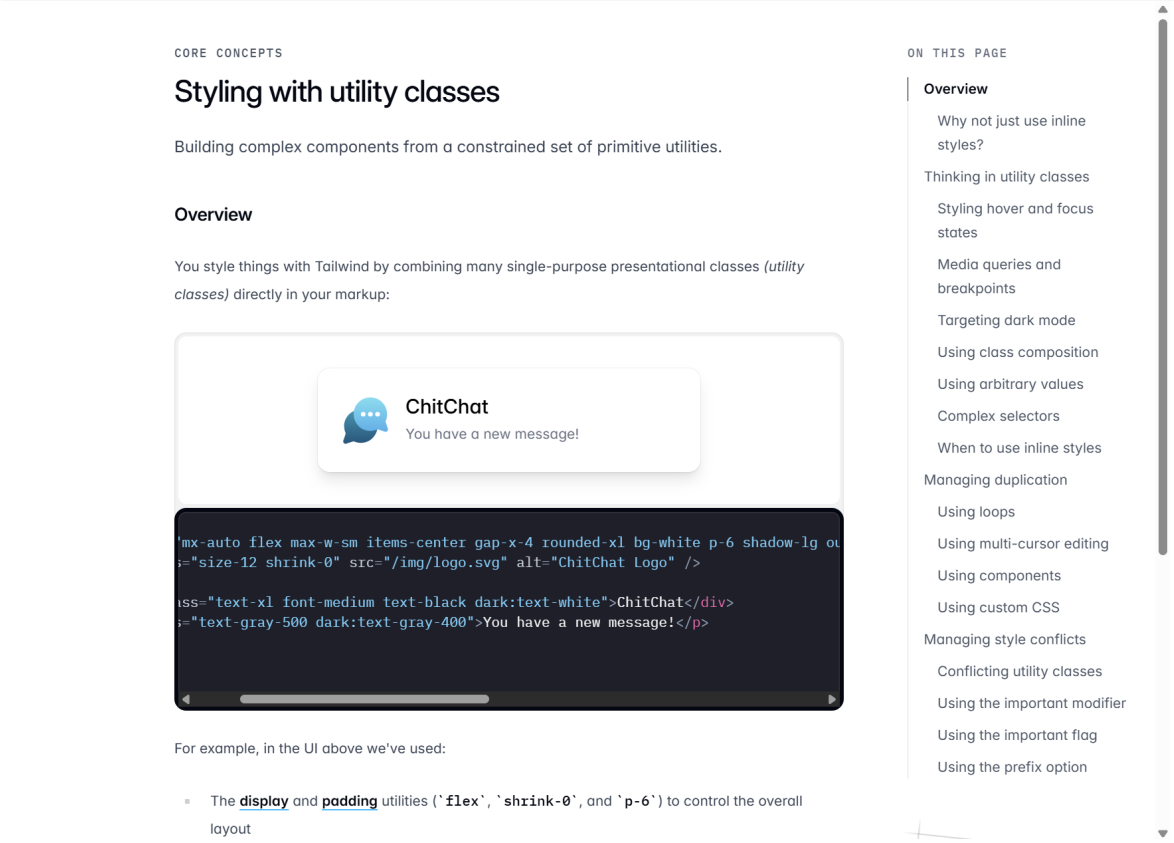
```
@import "tailwindcss";

@layer components {
    .btn {
        @apply px-3 py-2 bg-rose-500 text-white;
    }

    .btn-developer {
        background-color: blue;
        padding: 10px;
        color: white;
    }
}
```

> **Note**
>
> ☑ @components 自定义类选择器、不具有任何特殊性、也不可 hover:btn 方式使用
> ☑ utilities 视作为 Tailwind 自身的一部分、可 hover:animmate-paused 使用

05. 核心机制

> https://tailwindcss.com/docs/styling-with-utility-classes

- **ui**



CORE CONCEPTS

Styling with utility classes

Building complex components from a constrained set of primitive utilities.

Overview

You style things with Tailwind by combining many single-purpose presentational classes *(utility classes)* directly in your markup:

ChitChat
You have a new message!

```
'mx-auto flex max-w-sm items-center gap-x-4 rounded-xl bg-white p-6 shadow-lg ou
;="size-12 shrink-0" src="/img/logo.svg" alt="ChitChat Logo" />

ıss="text-xl font-medium text-black dark:text-white">ChitChat</div>
;="text-gray-500 dark:text-gray-400">You have a new message!</p>
```

For example, in the UI above we've used:

- The **display** and **padding** utilities (`flex`, `shrink-0`, and `p-6`) to control the overall layout

ON THIS PAGE

- Overview
  - Why not just use inline styles?
- Thinking in utility classes
  - Styling hover and focus states
  - Media queries and breakpoints
  - Targeting dark mode
  - Using class composition
  - Using arbitrary values
  - Complex selectors
  - When to use inline styles
- Managing duplication
  - Using loops
  - Using multi-cursor editing
  - Using components
  - Using custom CSS
- Managing style conflicts
  - Conflicting utility classes
  - Using the important modifier
  - Using the important flag
  - Using the prefix option

4.1 规则

> ⟹ 通常一个 css property 对应一个类选择器、采用 [属性 + 对应值] 方式

- ☑ p-5 ⟹ padding: 20px;
- ☑ mt-5 ⟹ margin-top: 20px;
- ☑ text-white ⟹ color: white;
- ☑ bg-white ⟹ background-color: white;

> ⓘ **Note**
>
> - ☑ 原子选择器 有更高的重复利用率、减少打包体积 如 p-5 可反复独立使用
> - ☑ 原子选择器 避免大量的自定义类选择器增加命名负担、结构控制负担、避免优先级问题
> - ☑ 原子选择器 易于开发者之间、社区之间进行样式复制
> - ☑ 原子选择器 带有轻微的人为公共语义、极易在开发者之间达成心智共识
> - ☑ 内联选择器 属性及值需完整给出、较多属性会使得声明较长
> - ☑ 内联选择器 难以肉眼可见理解属性值含义
> - ☑ 内联选择器 无法完成 hover 聚焦、媒体查询、等浏览器交互效果

4.1.1 内联样式

> Tailwind 主张特殊情况使用内联样式、如条件变量拼接样式、没有内置的实用类

- src/app.jsx

```
export function App() {

    const styles = {
        backgroundImage: "linear-gradient(#fff 1px, transparent 1px), linear-gradient(to right, #fff 1px, transparent 1px)",
        backgroundSize: "20px 20px",
        backgroundPosition: "center center"
    };

    return (
        <div className="p-32 bg-slate-300">
            <div className="w-1/2 h-80 bg-purple-700" style={ { ...styles } }>
                <div className="size-full flex justify-center items-center">
                    <p className="font-mono text-6xl text-white">Tailwind</p>
                </div>

            </div>
        </div>
    )
}
```

4.1.2 样式重复

> 可采用 循环 | 列编辑 | 抽象组件 等思考控制样式重复

- src/app.jsx

```
export function App() {

    return (
        <div className="p-32 bg-slate-300">
            <div className="flex gap-x-5">
                <button className="px-5 py-3 text-white bg-purple-700 cursor-
pointer">tailwind</button>
                <button className="px-5 py-3 text-white bg-purple-700 cursor-
pointer">react</button>
                <button className="px-5 py-3 text-white bg-purple-700 cursor-
pointer">motion</button>
                <button className="px-5 py-3 text-white bg-purple-700 cursor-
pointer">next</button>
                <button className="px-5 py-3 text-white bg-purple-700 cursor-
pointer">zustand</button>
            </div>
        </div>
    )
}
```

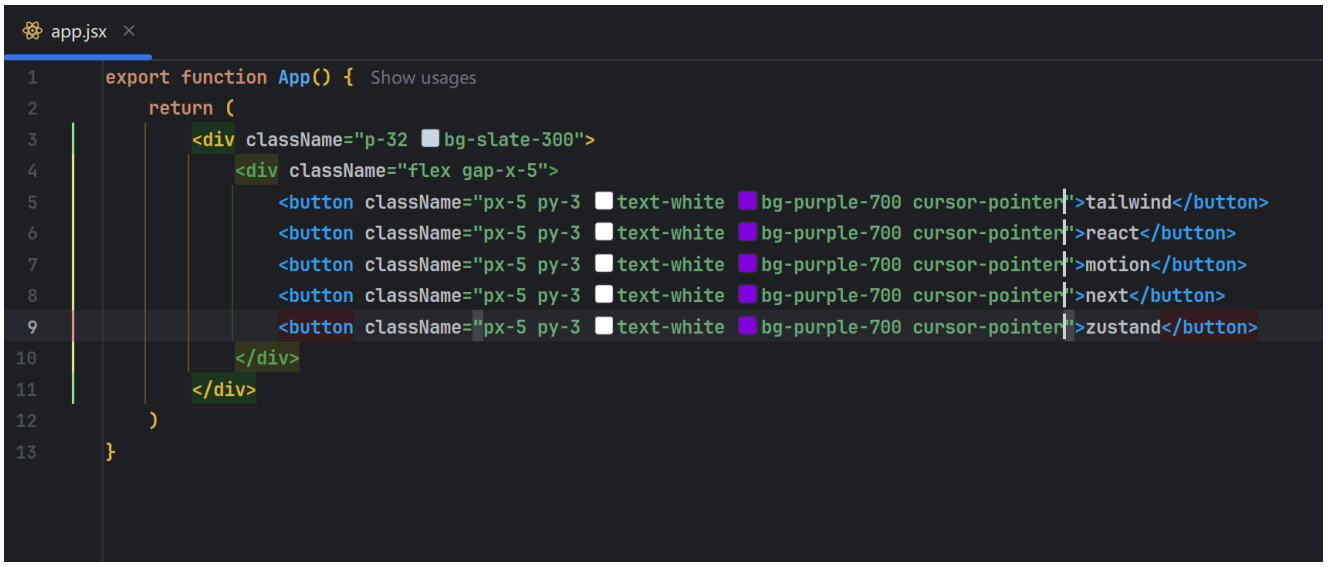
- **src/app.jsx [loop]**

```
export function App() {

    const titles = [
        "tailwind",
        "react",
        "motion",
        "next",
        "zustand",
    ];

    return (
        <div className="p-32 bg-slate-300">
            <div className="flex gap-x-5">
                {
                    titles.map((title) => (
                        <button className="px-5 py-3 text-white bg-purple-700
cursor-pointer"
                                key={ title }>
                            { title }
                        </button>
                    ))
                }
            </div>
        </div>
    )
```

```
}
```

列编辑 [ALT + SHIFT + ALT]

- **ui**



抽象组件

- **src/app.jsx**

```
function Button({ title }) {
    return (
        <button className="px-5 py-3 text-white bg-purple-700 cursor-pointer">
            { title }
        </button>
    )
}

export function App() {
    return (
        <div className="p-32 bg-slate-300">
            <div className="flex gap-x-5">
                <Button title={ "tailwind" }/>
                <Button title={ "react" }/>
```

```
                <Button title={ "motion" }/>
                <Button title={ "next" }/>
                <Button title={ "zustand" }/>
            </div>
        </div>
    )
}
```

- **src/app.jsx**

```
function Button({ title }) {
    return (
        <button className="px-5 py-3 text-white bg-purple-700 cursor-pointer">
            { title }
        </button>
    )
}

export function App() {

    const titles = [
        "tailwind",
        "react",
        "motion",
        "next",
        "zustand",
    ];

    return (
        <div className="p-32 bg-slate-300">
            <div className="flex gap-x-5">
                {
                    titles.map((title) => (
                        <Button title={ title } key={ title }/>
                    ))
                }
            </div>
        </div>
    )
}
```

06. 盒子大小

07. 背景

08. 间距

09. 弹性网格

10. 布局

11. 排版

12. 边框

13. 效果

14. 移动

15. 过渡动画

16. 互动

17. 表格

18. 作用

19. SVG