# 概述

Windi CSS是下一代实用程序优先的 CSS 框架。请将 Windi CSS 视为Tailwind的按需替代品，它提供更快的加载时间、与 Tailwind v2.0 的完全兼容性以及一系列额外的酷功能。Windi CSS 支持Tailwind CSS的所有实用程序，无需任何额外配置。

# 配置Windi CSS

Windi CSS中的配置与Tailwind CSS中的配置相似，但具有额外的增强和功能。默认情况下，Windi CSS 将在您的项目根目录下搜索配置文件。有效的文件默认名称是：

* windi.config.ts
* windi.config.js
* tailwind.config.ts
* tailwind.config.js

实例配置：

import { defineConfig } from 'windicss/helpers'

import colors from 'windicss/colors'

import plugin from 'windicss/plugin'

export default defineConfig({

darkMode: 'class', // or 'media'

theme: {

extend: {

screens: {

'sm': '640px',

'md': '768px',

'lg': '1024px',

'xl': '1280px',

'2xl': '1536px',

},

colors: {

blue: colors.sky,

red: colors.rose,

pink: colors.fuchsia,

},

fontFamily: {

sans: ['Graphik', 'sans-serif'],

serif: ['Merriweather', 'serif'],

},

spacing: {

128: '32rem',

144: '36rem',

},

borderRadius: {

'4xl': '2rem',

},

},

},

plugins: [

plugin(({ addUtilities }) => {

const newUtilities = {

'.skew-10deg': {

transform: 'skewY(-10deg)',

},

'.skew-15deg': {

transform: 'skewY(-15deg)',

},

}

addUtilities(newUtilities)

}),

plugin(({ addComponents }) => {

const buttons = {

'.btn': {

padding: '.5rem 1rem',

borderRadius: '.25rem',

fontWeight: '600',

},

'.btn-blue': {

'backgroundColor': '#3490dc',

'color': '#fff',

'&:hover': {

backgroundColor: '#2779bd',

},

},

'.btn-red': {

'backgroundColor': '#e3342f',

'color': '#fff',

'&:hover': {

backgroundColor: '#cc1f1a',

},

},

}

addComponents(buttons)

}),

plugin(({ addDynamic, variants }) => {

addDynamic('skew', ({ Utility, Style }) => {

return Utility.handler

.handleStatic(Style('skew'))

.handleNumber(0, 360, 'int', number => `skewY(-${number}deg)`)

.createProperty('transform')

}, variants('skew'))

}),

require('windicss/plugin/filters'),

require('windicss/plugin/forms'),

require('windicss/plugin/aspect-ratio'),

require('windicss/plugin/line-clamp'),

require('windicss/plugin/typography')({

modifiers: ['DEFAULT', 'sm', 'lg', 'red'],

}),

],

})

然后，在你的 Vite 配置中安装插件：

vite.config.js

import WindiCSS from 'vite-plugin-windicss'

export default {

plugins: [

WindiCSS(),

],

}

最后，导入virtual:windi.css您的 Vite 条目：

main.js

import 'virtual:windi.css'

# 提取

## 提取判断

Windi CSS 依靠对源文件的静态扫描和提取来查找您的实用程序用法并按需生成等效的 CSS。您需要使用 Windi CSS 实用程序的静态全名才能正确检测它们。例如，不能静态提取字符串连接：

<div class="text-${ active ? 'green' : 'orange' }-400"></div>

请改用实用程序的全名：

<div class="${ active ? 'text-green-400' : 'text-orange-400' }"></div>

## safelist

有时您必须使用动态连接：

<div class="p-${size}"></div>

为此，您需要在 的safelist选项中指定可能的组合windi.config.js。

import { defineConfig } from 'windicss/helpers'

export default defineConfig({

safelist: 'p-1 p-2 p-3 p-4',

})

或者更灵活：

import { defineConfig } from 'windicss/helpers'

function range(size, startAt = 1) {

return Array.from(Array(size).keys()).map(i => i + startAt)

}

export default defineConfig({

safelist: [

range(3).map(i => `p-${i}`), // p-1 to p-3

range(10).map(i => `mt-${i}`), // mt-1 to mt-10

],

})

## 扫描

当开发服务器/构建过程开始时，Windi CSS 将扫描您的源代码并提取实用程序使用情况。默认情况下，它会扫描src/带有扩展名的文件vue, html, mdx, pug, jsx, tsx。如果要启用/禁用对其他文件类型或位置的扫描，可以使用include和exclude选项进行配置：

import { defineConfig } from 'windicss/helpers'

export default defineConfig({

extract: {

// accepts globs and file paths relative to project root

include: [

'index.html',

'src/\*\*/\*.{vue,html,jsx,tsx}',

],

exclude: [

'node\_modules/\*\*/\*',

'.git/\*\*/\*',

],

},

})

## 预检

预检（样式重置）也可以通过扫描按需启用。您可以在配置中完全禁用它：

import { defineConfig } from 'windicss/helpers'

export default defineConfig({

preflight: false,

})

或者通过安全列表显式启用它：

import { defineConfig } from 'windicss/helpers'

export default defineConfig({

preflight: {

safelist: 'h1 h2 h3 p img',

},

})

# 价值自动推断

Windi CSS 实用生成使用的 CSS 样式，因此您可以在类中使用任意值并根据程序生成相应的样式。

<!-- sizes and positions -->

<div class="p-5px mt-[0.3px]"></div>

<!-- colors -->

<button class="bg-hex-b2a8bb h-20px w-20px"></button>

<button class="bg-[hsl(211.7,81.9%,69.6%)] h-20px w-20px"></button>

<!-- grid template -->

<div class="grid-cols-[auto,1fr,30px] h-10px"></div>

## 数字

p-{float} -> padding: {float/4}rem;

p-2.5

p-3.2

CSS：

.p-2\.5 {

填充： 0.625rem ；

}

.p-3\.2 {

填充： 0.8rem ;

}

## 尺寸

// {size} should be end with rem|em|px|vh|vw|ch|ex

p-{size} -> padding: {size};

p-3px

p-4rem

CSS：

.p-3px {

填充： 3px ;

}

.p-4rem {

填充： 4rem ；

}

## 分数

w-{fraction} -> width: {fraction -> percent};

w-9/12

CSS：

.w-9\/12 {

宽度: 75% ;

}

## 颜色

text-{color} -> color: rgba(...);

border-hex-{hex} -> border-color: rgba(...);

text-cyan-400

border-hex-6dd1c7

CSS：

. border-hex-6dd1c7 {

--tw-border-opacity: 1;

border-color: rgba(109, 209, 199, var(--tw-border-opacity));

}

.text-cyan-400 {

--tw-text-opacity: 1;

color: rgba(34, 211, 238, var(--tw-text-opacity));

}

## 变量

可以传递变量名称，这与 CSS 变量结合使用非常有用。

bg-${variableName}

bg-$test-variable

CSS：

.bg-\$test-variable {

background-color: var ( --test-variable ) ;

}

@layer base {

:root {

--color-base: theme('colors.white');

--test-variable: theme('colors.black');

--color-off-base: theme('colors.gray.50');

--color-text-muted: theme('colors.gray.600');

--color-text-muted-hover: theme('colors.gray.500');

--color-primary: theme('colors.blue.800');

--color-secondary:theme('colors.blue.200');

}

}

## 网格模板

grid-cols-[auto,1fr,30px]

grid-cols-[auto,1fr,30px]

CSS：

.grid-cols-\[auto\2c 1fr\2c 30px\] {

grid-template-columns : auto 1fr 30px ;

}

# 动作分组

通过用括号对它们进行分组来为相同的变体应用实用程序。

<div class="hover:(bg-gray-400 font-medium) bg-white font-light"/>

bg-blue-200 font-light p-2

hover:(bg-gray-400 font-medium)

CSS：

.bg-blue-200 {

--tw-bg-opacity : 1 ;

背景颜色： rgba（191、219、254 、var（--tw-bg-opacity））；

}

.hover\:bg-gray-400:hover {

--tw-bg-opacity : 1 ;

background-color：rgba ( 156 , 163 , 175 , var ( --tw-bg-opacity ) ) ;

}

.font-light {

font-weight: 300 ;

}

.hover\:font-medium:hover {

font-weight : 500 ;

}

.p-2 {

padding：0.5rem ;

}

# 快捷方式

当您处理类似的实用程序集时，重复出现是很常见的。我们提供了这种“快捷方式”功能，允许您提供实用程序名称的组合，您可以在应用程序中的任何地方重复使用这些名称，而无需重复自己。只需将该shortcuts字段添加到您的配置中：

export default {

theme: {

/\* ... \*/

},

shortcuts: {

'btn': 'py-2 px-4 font-semibold rounded-lg shadow-md',

'btn-green': 'text-white bg-green-500 hover:bg-green-700',

},

}

复杂实用程序也支持 CSS-in-JS 语法：

export default {

theme: {

/\* ... \*/

},

shortcuts: {

'btn': {

'color': 'white',

'@apply': 'py-2 px-4 font-semibold rounded-lg',

'&:hover': {

'@apply': 'bg-green-700',

'color': 'black',

},

},

'btn-green': 'text-white bg-green-500 hover:bg-green-700',

},

}

此配置添加的实用程序也可以直接包装在变体中，例如 sm:btn。此功能的目的类似于@apply指令，它将所有实用程序合并为一种样式。

# 响应式设计

在Windi CSS 中进行响应式设计是毫不费力的。通过简单地将变体前缀添加到您要使用的实用程序中，将自动生成相应的媒体查询md:或者lg:。

p-1 lg:p-2

CSS：

.p-1 {

padding: 0.25rem;

}

@media (min-width: 1024px) {

.lg\:p-2 {

padding: 0.5rem;

}

}

## 自定义范围

默认情况下，Windi CSS 的断点设计为 Mobile First。这意味着无前缀实用程序（如p-1）对所有屏幕尺寸生效，而带前缀实用程序（如）仅在指定断点及以上md:p-2生效。

当您想要将断点变体应用到多个实用程序时，在 Windi CSS 中，您可以使用Variant Groups功能来完成此操作，而无需重复自己。我们还提供了通过添加<和@前缀来更好地控制查询范围的能力：

* lg => 大于或等于此断点
* <lg => 小于此断点
* @lg => 正是此断点范围

lg:p-1

<lg:p-2

@lg:p-3

CSS：

@media (min-width: 1024px) {

.lg\:p-1 {

padding: 0.25rem;

}

}

@media (min-width: 1024px) and (max-width: 1279.9px) {

.\@lg\:p-3 {

padding: 0.75rem;

}

}

@media (max-width: 1023.9px) {

.\<lg\:p-2 {

padding: 0.5rem;

}

}

## 断点

|  | **默认** | **<前缀** | **@前缀** |
| --- | --- | --- | --- |
| sm | （最小宽度：640px） | （最大宽度：639.9px） | （最小宽度：640px）和 （最大宽度：767.9px​​） |
| MD | （最小宽度：768px） | （最大宽度：767.9px​​） | （最小宽度：768px）和 （最大宽度：1023.9px） |
| lg | （最小宽度：1024px） | （最大宽度：1023.9px） | （最小宽度：1024px）和 （最大宽度：1279.9px） |
| XL | （最小宽度：1280px） | （最大宽度：1279.9px） | （最小宽度：1280px）和 （最大宽度：1535.9px） |
| 2xl | （最小宽度：1536px） | （最大宽度：1535.9px） | （最小宽度：1536px） |

## 定制

可以在windi.config.js的配置文件中，定制。

import { defineConfig } from 'windicss/helpers'

export default defineConfig({

theme: {

screens: {

tablet: '640px',

laptop: '1024px',

desktop: '1280px',

},

},

})

# 黑暗模式

Windi CSS 具有开箱即用的暗模式支持。通过为实用程序添加变体前缀dark:，它们将仅在启用暗模式时应用。在以下示例中，Preview文本在浅色模式下为红色，在深色模式下为绿色。尝试使用它：

text-red-400 dark:text-green-400

我们有两种启用暗模式的模式，class模式和媒体查询模式。默认情况下，class启用模式。

## class模式

类模式可让您更好地控制何时应启用暗模式。

windi.config.js

export default {

darkMode: 'class',

// ...

}

它检测父元素的class="dark"，通常您可以将其应用于html元素以使其全局工作。

<html>

<body>

<!-- Dark mode disabled -->

</body>

</html>

<html class="dark">

<body>

<!-- Dark mode enabled -->

</body>

</html>

您可以使用以下代码段使配色方案与用户的系统偏好相匹配，或者编写自己的逻辑来管理它。

if (window.matchMedia('(prefers-color-scheme: dark)').matches)

document.documentElement.classList.add('dark')

else

document.documentElement.classList.add('light')

## 媒体查询方式

在媒体查询模式下，它使用来自浏览器的内置@media (prefers-color-scheme: dark)查询，该查询始终与用户的系统偏好相匹配。

export default {

darkMode: 'media',

// ...

}

# RTL

Windi CSS 具有开箱即用的 RTL 支持，从v2.5.4。通过为实用程序添加变体前缀rtl:，它们将仅在启用 RTL 时应用。启用 RTL 很容易，您只需要dir="rtl"在html元素上应用即可。

<html>

<body>

<!-- RTL disabled -->

</body>

</html>

<html dir="rtl">

<body>

<!-- RTL enabled -->

</body>

</html>

# 重要前缀

可以为任何实用程序类添加前缀!以将它们设置为!important. 当您想要覆盖特定属性的先前样式规则时，这可能非常有用。

!text-green-300

# 指令

## @apply

用于@apply将任何现有的实用程序类内联到您的样式块中。当在 HTML 中找到想要提取到新组件的通用实用程序模式时，这很有用。

.btn {

@apply font-bold py-2 px-4 rounded;

}

.btn-blue {

@apply bg-blue-500 hover:bg-blue-700 text-white;

padding-top: 1rem;

}

如果您想要@apply一个现有的类并制作它!important，只需添加!important到声明的末尾：

.btn {

@apply font-bold py-2 px-4 rounded !important;

}

## @variants

可以通过将它们的定义包装在指令中来生成您自己的实用程序的屏幕变体、状态变体、主题变体。@variants

@variants focus, hover {

.rotate-0 {

transform: rotate(0deg);

}

.rotate-90 {

transform: rotate(90deg);

}

}

@variants dark {

.bg-color {

background-color: #1c1c1e;

}

}

## @screen

该@screen指令允许您创建按名称引用断点的媒体查询，而不是在您自己的 CSS 中复制它们的值。

@screen sm {

.custom {

@apply text-lg;

}

}

## @layer

该@layer指令设置每个类的应用顺序。有效层是base、components和utilities。

@layer components {

.components {

@apply bg-red-500;

}

}

@layer utilities {

.utilities {

max-width: 768px;

}

}

@layer base {

base {

margin-left: auto;

}

}

.normal {

margin-right: auto; /\* components by default \*/

}

## theme()

该theme()函数允许您使用点表示法访问您的配置值。

.btn-blue {

background-color: theme("colors.blue.500");

}

生成的 CSS：

.btn-blue {

background-color: #3b82f6;

}

# 辅助模式

使用此功能，您可以在 HTML 属性中编写 windi 类。大小，css大小有关可能会出现的问题，但g压缩下，可能会影响czip的大小。默认情况下，属性是的，首先在您的 Windi 配置中启用它。

windi.config.ts

import { defineConfig } from 'windicss/helpers'

export default defineConfig({

attributify: true,

})

并根据需要使用它们：

<button

bg="blue-400 hover:blue-500 dark:blue-500 dark:hover:blue-600"

text="sm white"

font="mono light"

p="y-2 x-4"

border="2 rounded blue-200"

>

Button

</button>