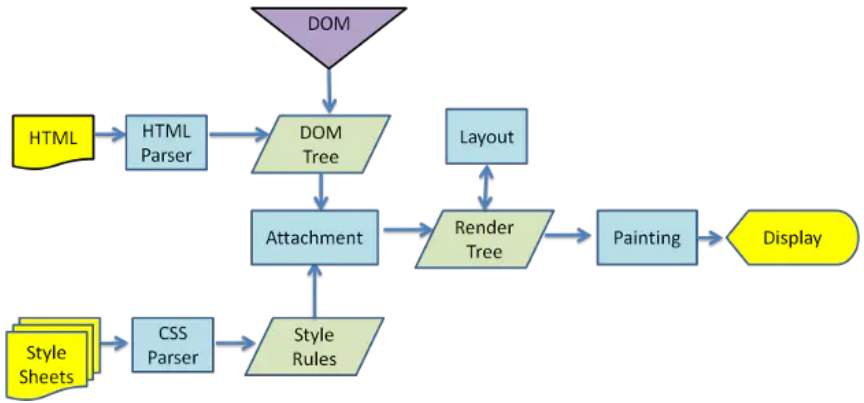

CSS 原子化
css 原子化，工程化，编程直觉化

CSS 对渲染的影响



当 css 文件加载时间为 5s 时，你访问页面，5s 内你会看到什么？ →

```
<!-- css -->
<link rel="stylesheet" href="./global.css">

<!-- html -->
<div class='bg-red'>
  我是一段文本内容
</div>
```

什么是 CSS 原子化

在 JS 中我们可以通过函数式编程的思想，把一个大函数分解成多个小的纯函数，然后小函数也可以相互组合生成其他我们所需的函数，这样我们可以最大程度上的复用代码，减少代码体积。

对 CSS 而言，我们也可以采用函数式编程的思想，也叫做 CSS 原子化。

```
// 来源于 A 页面
.list-item {
  display: flex;
  padding-top: 20px;
  margin-top: 20px;
  // ...其他样式
}
// 来源于 B 页面
.carbon-list-item {
  display: flex;
  margin-top: 20px;
  // ...其他样式
}
```

```
.flex {
  display: flex;
}

.pt-20px {
  padding-top: 20px;
}

.mt-20px {
  margin-top: 20px;
}
```

公用 CSS

```scss
// global.scss
@for $i from 1 through 10 {
  .m-#{$i} {
    margin: ($i * 1px);
  }
}
```

```css
// css 编译结果
.m-1 { margin: 1px; }
.m-2 { margin: 2px; }
/* ... */
.m-10 { margin: 10px; }
```

```html
<div class='m-1'>
</div>

<div class='m-10'>
</div>

<div class='m-100'>
</div>
```

```css
// css 编译结果
.m-1 { margin: 1px; }
.m-10 { margin: 10px; }
.m-100{ margin: 100px; }
```

CSS 原子化框架的作用

预置了所有 CSS 的原子化样式，或者说是所有的 CSS 原子化样式生成规则。

```
// n 等于 1,2,3,4...n-1,n

// margin
.m-n { margin: n; }
// padding
.p-n { padding: n; }
// width
.w-n { width: n; }
// height
.h-n { height: n; }
// flex
.flex { display: flex; }
// grid
.grid { display: grid; }
// ...省略
```

```
.content {
  display: flex;
  margin-top: 10px;
  padding-top: 10px;
  color: pink;
}
```

```
  <div class='content'>
    我是内容
  </div>
```

```
  <div class='flex mt-10px pt-10px color-pink'>
    我是内容
  </div>
```

CSS 原子化框架对比

Tailwindcss，Windicss，Unocss

CSS 的直觉性编程

以 tailwind 为例，我们想要设置边框，就直觉性的写了如下代码：

```
<!-- 不会生效 -->
<div class='border-10'>
  我的 border 是 10px
</div>
```

在 tailwind 中，border 默认提供了 2px，4px，6px，8px 几个尺寸。

```
// 全局样式
.border-10 {
  border-width: 10px;
}
```

```
// tailwind.config.js
module.exports = {
  theme: {
    borderWidth: {
      DEFAULT: '1px',
      '0': '0',
      '2': '2px',
      '3': '3px',
      '4': '4px',
      '6': '6px',
      '8': '8px',
      '10': '10px' // <-- here
    }
```

我们会发现，这样的编程是违反直觉性的，而且我们为了解决问题得反复查阅文档，然后再回来编写代码。

此时，我可能已经忘了我在哪块编程。

windicss，unocss

在 windicss 和 unocss 中，程序会以特定规则推测你编写的样式。

```
<div class='border-100px'>
  我的 border 是 100px
</div>
```

如果你是以 border 开头，程序会自动生成你后面写的尺寸：

```
.border-100px {
  border-width: 100px;
}
```

如此这般，我们就可以依据自己的直觉编写 CSS。

自定义规则

```
// tailwind.config.js
const _ = require('lodash')
const plugin = require('tailwindcss/plugin')

module.exports = {
  theme: {
    rotate: {
      '1/4': '90deg',
      '1/2': '180deg',
      '3/4': '270deg',
    }
  },
  plugins: [
    plugin(function({ addUtilities, theme, e }) {
      const rotateUtilities = _.map(theme('rotate'), (value, ke
        return {
          [`.${e(`rotate-${key}`)}`]: {  transform: `rotate(${v
        }
      })
      addUtilities(rotateUtilities)
    })
  ]
}
```

编译结果：

```
.rotate-1\/4 {
  transform: rotate(90deg);
}
.rotate-1\/2 {
  transform: rotate(180deg);
}
.rotate-3\/4 {
  transform: rotate(270deg);
}
```

在 tailwind 中，我们想添加一个自定义规则，是如此复杂，而 windi 采用的是和 tailwind 一样的插件系统，每次编写规则可能都是痛苦面具😖😖😖。

UnoCSS 中的自定义规则

```
rules: [
  ['m-1', { margin: '0.25rem' }]
]
```

当在用户代码库中检测到 m-1 时，就会生成如下 CSS：

```
.m-1 { margin: 0.25rem; }
```

想要使其动态化，可以将匹配器修改为正则表达式，将主体改为一个函数：

```
rules: [
  [/^m-(\d+)$/, ([, d]) => ({ margin: `${d / 4}rem` })],
  [/^p-(\d+)$/, (match) => ({ padding: `${match[1] / 4}rem` })]
]
```

其中，回调函数的第一个参数为正则 matcher，所以你可以对它进行解构以获得正则表达式的匹配组。

我们假设写了如下代码：

```
<div class="m-100">
  <button class="m-3">
    <icon class="p-5" />
    My Button
  </button>
</div>
```

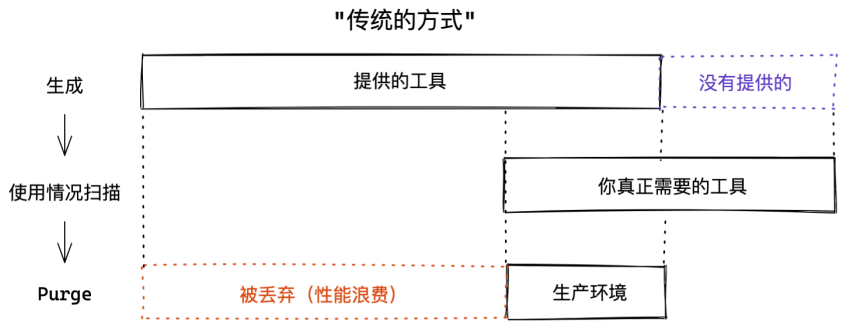
就会生成相应的 CSS：

```
.m-100 { margin: 25rem; }
.m-3 { margin: 0.75rem; }
.p-5 { padding: 1.25rem; }
```

这样编写规则真是异常轻松😄 😄 😄。

传统方式

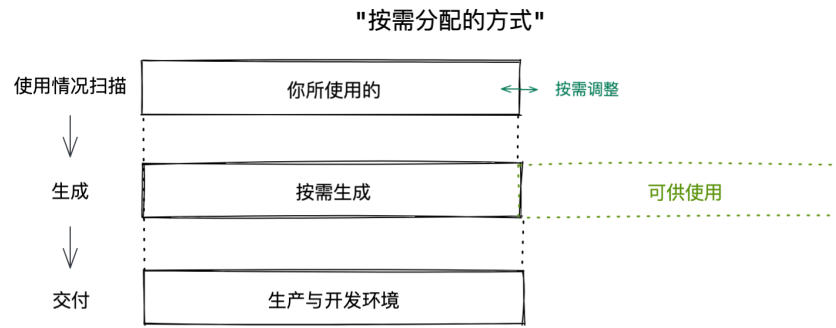
tailwind 采用了传统的生成方式



传统的方式不仅会消耗不必要的资源（生成了但未使用），甚至有时更是无法满足你的需求，因为总会有部分需求无法包含在内。

按需

windi,uno 采用了按需的方式



通过调换 "生成" 和 "扫描" 的顺序，"按需" 会为你节省浪费的计算开销和传输成本，同时可以灵活地实现预生成无法实现的动态需求。另外，这种方法可以同时在开发和生产中使用，提供了一致的开发体验，使得 HMR (Hot Module Replacement, 热更新) 更加高效。

扫描模式

Windi CSS 和 Tailwind JIT 都采用了预先扫描源代码的方式。下面是一个简单示例：

```
import glob from 'fast-glob'
import { promises as fs } from 'fs'

// 通常这个是可以配置的
const include = ['src/**/*.{jsx,tsx,vue,html}']

async function scan() {
  const files = await glob(include)

  for (const file of files) {
    const content = await fs.readFile(file, 'utf8')
    // 将文件内容传递给生成器并配对 class 的使用情况
  }
}

await scan()
// 扫描会在构建/服务器启动前完成
await buildOrStartDevServer()
```

为了在开发期间提供 HMR，通常会启动一个 文件系统监听器：

```
import chokidar from 'chokidar'

chokidar.watch(include).on('change', (event, path) => {
  // 重新读取文件
  const content = await fs.readFile(file, 'utf8')
  // 将新的内容重新传递给生成器
  // 清除 CSS 模块的缓存并触发 HMR 事件
})
```

windi 相较于 tailwind 增加了按需，而 uno 相较于 windi 简化了很多步骤。

```
10/21/2021, 2:17:45 PM
1656 utilities | x50 runs

none                              8.75 ms /    0.00 ms
unocss       v0.0.0              13.72 ms /    4.97 ms (x1.00)
windicss     v3.1.9             980.41 ms /  971.66 ms (x195.36)
tailwindcss  v3.0.0-alpha.1    1258.54 ms / 1249.79 ms (x251.28)
```

UnoCSS

Shortcuts

假设我们想写出如下的样式：

```
.style {
  display: flex;
  width: 100px;
  height: 100px;
  align-items: center;
  color: pink;
}
```

我们会在代码中这样写

```
<div class='flex w-100px h-100px items-center color-pink'>
  我是内容
</div>
```

可能这是一个公用样式，好多地方都会用到。

我们可以直接配置快捷方式：

```
{
  shortcuts: [
    con: 'flex w-100px h-100px items-center color-pink',
    [/^con-(.*)$/, ([, c]) => `flex w-100px h-100px items-cente
  ]
}
```

于是我们就可以这样写了

```
<div class='con-pink'>
  我是内容
</div>
```

Attributify mode

```
<button class="bg-blue-400 hover:bg-blue-500 text-sm text-white font-mono font-light py-2 px-4 rounded border-2 border-blue-200 dar
  Button
</button>
```

经过属性化之后：

```
<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>
```

加入前缀标识：

```
<button
  un-bg="blue-400 hover:blue-500 dark:blue-500 dark:hover:blue-600"
  un-text="sm white"
  un-font="mono light"
  un-p="y-2 x-4"
  un-border="2 rounded blue-200"
>Button</button>
```

Theme

unocss 天然支持 dark 模式

```
<div class='text-white dark:text-white'>
  我是主题文字
</div>
```

生成的样式：

```
.text-white { /* 省略 */ }
.dark .dark\:text-white { /* 省略 */ }
```

也可以自己配置主题

```
theme: {
  dark: {
    white: 'pink'
  },
  compact: {
  }
}
```

不论是通过配置定义主题，还是以 dark：开头写样式，都是 ok 的。

CSS Directives

该样式只在本页面用，且不想添加 Shortcuts。

```
<div class='text-center my-0 font-medium;'>....</div>

<!-- 页面另一个位置 -->
<div class='text-center my-0 font-medium;'>....</div>
```

@apply

```
.custom-div {
  @apply text-center my-0 font-medium;
}
```

将会生成：

```
.custom-div {
  margin-top: 0rem;
  margin-bottom: 0rem;
  text-align: center;
  font-weight: 500;
}
```

开发中遇到的问题

结构不够清晰：

```pug
pug □ | You, last month | 1 author (You)
<template>
  <div class="h-100% p-20px ●bg-white">
    <div class="flex justify-between mb-20px">
      <a-button
        type="primary"
        @click="$router.push({ name: 'addAdvancedComputedToolConfig' })"
      >
        添加工具
      </a-button>

      <a-input-search
        v-model="searchName"
        placeholder="搜索工具名称"
        class="w-180px"
        @keydown.enter="getList"
        @input="getList"
      />
    </div>

    <div
      class="overflow-y-auto h-[calc(100%-50px)]"
    >
      <div
        v-for="item in filterData"
        :key="item.id"
        class="mb-20px"
      >
        <div
          class="text-14px"
          un-before="inline-block w-6px h-6px mr-8px mb-3px bg-[var(--color)] content-none b-rd-50%"
          :style="getRandomStyle()"
        >
          {{ item.toolGroupName }}
        </div>
        <div class="lt-xl-grid-cols-2 2xl-grid-cols-4 grid mt-16px gap-20px grid-cols-3">
```

提醒不够：

```
<PtIcon
  :name="it.toolIcon"
  width="16"
  class="mt-2p
/>
<div>
  <div class="flex items-center mb-8px text-16px font-600">
    <span
      class="line-clamp-1 text-ellipsis"
      :title="it.toolName"
    >{{ it.toolName }}</span>
```

```
/* layer: default */
.flex {
  display: flex;
}
```

改造了官方插件

class- 起手的会有代码高亮：

针对你选中的 unocss 片段，生成样式合集：