Tailwind CSS

Tailwind는 유틸리티 클래스 조합 + JIT 컴파일 + 모바일 퍼스트 변형 + 컨테이너 쿼리로, 빠르고 예측 가능한 반응형 UI를 만드는 도구다.

---

1) 오늘 배운 핵심

주제	내용
Utility-first CSS	`p-4`, `bg-blue-500` 같은 작은 유틸리티를 조합해 UI 구성
동작 원리	소스(HTML/JSX 등)를 스캔 → 쓰인 클래스만 **JIT**로 CSS 생성 → PostCSS가 최종 CSS로 출력
상태 변형 (variants)	`hover:`, `focus:`, `active:`, `first:`, `odd:` 등
반응형 디자인	모바일 퍼스트(min-width) 브레이크포인트(`sm`, `md`, `lg`, `xl`, `2xl`)
컨테이너 쿼리	뷰포트가 아니라 **부모/조상 컨테이너 크기 기준**으로 스타일링(`@container`, `@md`, `@max-lg` 등)
임의값과 단위	`@min-[475px]`, `@max-[960px]`, `w-[50cqw]` 등 **즉석 값**과 **컨테이너 쿼리 단위(cqw, cqh...)**
Tailwind Merge	클래스 충돌을 똑똑하게 병합(`twMerge` / 가벼운 결합은 `twJoin`)

2) Utility-first 정리

장점

- 빠른 스타일링
- 일관된 디자인
- 낮은 CSS 중복/누수로 스코프 고민 적어짐

전통적인 방식 vs Tailwind

```
<!-- 전통적인 방식 -->
<button class="btn">Save</button>
/ * .btn { padding: 8px 12px; background: #3b82f6; ... } * /

<!-- Tailwind 방식 -->
<button class="px-3 py-2 bg-blue-500 text-white rounded">Save</button>
```

3) Tailwind의 구동

- 설정 파일 기반: 디자인 시스템을 `tailwind.config.js`에 정의
- JIT 컴파일: "쓰는 것만 빌드" → 번들 최소화, 실시간 피드백 빠름
- PostCSS 플러그인: `@tailwind base; components; utilities;`를 실제 CSS로 변환

> 사용하지 않은 클래스는 자동 제거되어 **최종 CSS가 작다**는 점이 유리하다.

4) 상태 변형(Variants)

```
<button class="bg-violet-500 hover:bg-violet-600 focus:outline-2
focus:outline-offset-2 focus:outline-violet-500 active:bg-violet-700">
  Save changes
</button>

<ul class="divide-y">
  <li class="first:pt-0 last:pb-0">Item 1</li>
  <li class="odd:bg-gray-50 even:bg-gray-100">Item 2</li>
</ul>

<div class="group">
  <button class="group-hover:bg-red-500">Hover me!</button>
</div>
```

기타: `visited:`, `disabled:`, `checked:`, `focus-within:`, `placeholder:` 등.

5) 반응형 디자인(Responsive)

5-1. 모바일 퍼스트 & 브레이크포인트

- 기본(접두사 없음)은 **모바일**
- 더 큰 화면에서 바꾸고 싶으면 `sm:`, `md:`, `lg:`, `xl:`, `2xl:` 사용
- 예: `md:flex`(≥ 768px에서 `display:flex`)

```
<div class="p-4 md:flex">
  <img class="w-full md:w-48 md:shrink-0" src="/img.jpg" alt="" />
  <div class="mt-4 md:mt-0 md:ml-6">
    <h2 class="text-xl font-bold">Title</h2>
    <p class="text-gray-600">Responsive example…</p>
  </div>
</div>
```

기본 브레이크포인트(min-width)

접두사	최소 너비
`sm`	≥ 40rem (640px)
`md`	≥ 48rem (768px)
`lg`	≥ 64rem (1024px)
`xl`	≥ 80rem (1280px)
`2xl`	≥ 96rem (1536px)

5-2. 구간 타깃과 범위

- **구간 한정**: `md:max-xl:flex` → `md` 이상 **그리고** `xl` 미만에서만 적용
- **단일 구간만**: `md:max-lg:...` → 오직 `md` 구간만

5-3. 실전 패턴

```
<!-- 카드 그리드: 1 → 2 → 3열 -->
<ul class="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
  ...
</ul>

<!-- 내비게이션: 모바일 햄버거 / 데스크톱 가로 메뉴 -->
<header class="border-b">
  <div class="max-w-screen-xl mx-auto flex items-center justify-between p-
4">
    <a class="text-xl font-bold">Brand</a>
    <button class="md:hidden p-2 rounded hover:bg-gray-100" aria-
label="Menu">≡</button>
    <nav class="hidden md:flex gap-6">
      <a class="hover:text-blue-600">Docs</a>
      <a class="hover:text-blue-600">Blog</a>
      <a class="hover:text-blue-600">About</a>
    </nav>
  </div>
</header>

<!-- 타입/여백 스케일 업 -->
<article class="p-4 md:p-8">
  <h1 class="text-2xl md:text-3xl xl:text-5xl font-bold">Responsive Type
Scale</h1>
  <p class="mt-3 md:mt-4 xl:mt-6 text-gray-700">Body text…</p>
</article>
```

6) 컨테이너 쿼리(Container Queries)

6-1. 기본 사용

- 부모에 `@container` 부여 → 자식에서 `@sm`, `@md`처럼 **컨테이너 폭 기준** 변형 사용

```
<div class="@container">
  <div class="flex flex-col @md:flex-row">...</div>
</div>
```

- **최대/범위**도 가능: `@max-sm`, `@sm:@max-md`

```
<div class="@container">
  <div class="flex flex-row @max-md:flex-col">...</div>
```

```
</div>
```

- **명명된 컨테이너**: `@container/main` → 손자/증손자에서도 특정 컨테이너 기준으로 스타일

```
<div class="@container/main">
  <div class="flex flex-row @sm/main:flex-col">...</div>
</div>
```

6-2. 임의의 값 사용(Arbitrary Values)

- 테마에 등록하지 않고 일회성 컨테이너 크기를 사용

```
<div class="@container">
  <!-- 컨테이너가 475px 이상이면 가로 배치 -->
  <div class="flex flex-col @min-[475px]:flex-row">...</div>
</div>
```

- **최대 너비 조건** 예: `@max-[960px]`

6-3. 컨테이너 쿼리 단위

- 컨테이너 크기를 기준으로 동작하는 단위: `cqw`(너비%), `cqh`(높이%), `cqi`/`cqb`, `cqmin`/`cqmax`

```
<div class="@container">
  <div class="w-[50cqw]">...</div>
  <!-- 컨테이너 너비의 50% -->
</div>
```

6-4. 기본 컨테이너 크기 참조(내장 변형)

아래와 같이 **16rem(256px) ~ 80rem(1280px)** 범위의 명명된 단계가 제공됨:

변형	최소 너비
`@3xs`	16rem (256px)
`@2xs`	18rem (288px)
`@xs`	20rem (320px)
`@sm`	24rem (384px)
`@md`	28rem (448px)
`@lg`	32rem (512px)
`@xl`	36rem (576px)

변형	최소 너비
@2xl	42rem (672px)
@3xl	48rem (768px)
@4xl	56rem (896px)
@5xl	64rem (1024px)
@6xl	72rem (1152px)
@7xl	80rem (1280px)

사용 예

```
<div class="@container">
  <!-- md(≥28rem)에서 가로배치, 3xl(≥48rem)에서 간격 확대 -->
  <div class="flex flex-col @md:flex-row @3xl:gap-8">...</div>
</div>
```

6-5. 사용자 정의 컨테이너 크기

`@theme`로 컨테이너 단계 추가/수정 가능

```
@import 'tailwindcss';
@theme {
  --container-8xl: 96rem; /* 새 단계 추가 */
}
```

```
<div class="@container">
  <div class="flex flex-col @8xl:flex-row">...</div>
</div>
```

7) Tailwind Merge 사용법

- 클래스 문자열 병합 유틸
- **`twMerge`**: Tailwind 규칙을 이해하고 **충돌을 해소**(예: `p-2 p-4` → `p-4`)
- **`twJoin`**: 가벼운 문자열 결합(공백·falsy 제거)만, **충돌 처리 X**

```
import { twMerge, twJoin } from 'tailwind-merge';

const cls1 = twJoin('px-4', undefined, 'py-2', '  '); // "px-4 py-2"
const cls2 = twMerge('px-2 px-4', 'bg-blue-500', 'bg-red-500'); // "px-4
bg-red-500"
```

- 단순 합치기면 `twJoin`(빠름), 충돌 우려가 있으면 `twMerge`(안전)