

Tailwind CSS로 개발자가 만드는 멋진 UI 스타일링

Create a great web Ui Design styling with Tailwind CSS



코딩웍스 Tailwind CSS 강의 소개



[Tailwind CSS 강의소개] Tailwind CSS 장점과 단점.. 그리고 중심 포인트

{ Tailwind CSS 코드 vs 일반 CSS 코드 비교 }

Tailwind CSS 중심 포인트... 지저분한 HTML을 극협하는 코딩웍스 본인이 강의를 만드는 동안 너무 재밌게 만들며 그 매력에 빠졌다는 것! 특히 주변 개발자에게 Tailwind CSS를 강력 추천하고 싶다는 것!



Kristen Ramos

kristen.amos@example.com

```

<body class="h-screen flex justify-center items-center bg-gray-100">
  <div class="bg-white rounded-lg flex gap-4 border-b border border-slate-200 p-[1.25rem] shadow-md">
    
    <div>
      <b class="font-[500]">Kristen Ramos</b>
      <span class="block text-gray-400 mt-[5px]">Kristen.amos@example.com</span>
    </div>
  </div>

```

Tailwind HTML

▲ Tailwind CSS 코드

```

body {
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #lightgray;
}
.crew {
  background-color: #fff;
  border-radius: 10px;
  display: flex;
  gap: 15px;
  padding: 20px;
  box-shadow: 3px 3px 5px rgba(0,0,0,0.2);
}
.crew img {
  width: 50px;
  border-radius: 50%;
  border: 1px solid #ddd;
}
.crew div b {
  font-weight: bold;
}
.crew div span {
  display: block;
  color: #aaa;
  margin-top: 5px;
}

```

일반 CSS

◀ 일반 CSS 코드

```

<div class="crew">
  
  <div>
    <b>Kristen Ramos</b>
    <span>kristen.amos@example.com</span>
  </div>
</div>

```

HTML

Tailwind CSS의 장점

- 신속한 개발로 개발 시간을 단축할 수 있다.
- 클래스 이름을 고민할 필요가 없다.
- 일관성 있는 디자인 시스템을 적용하기 좋음.
- 반응형 레이아웃 작업이 매우 간편하다.
- 컴포넌트 활용으로 지저분한 HTML 코드를 보 완할 수 있다.
- 유지 관리하기 좋음.
- 프레임워크의 독립성(다른 프레임워크와 병용)

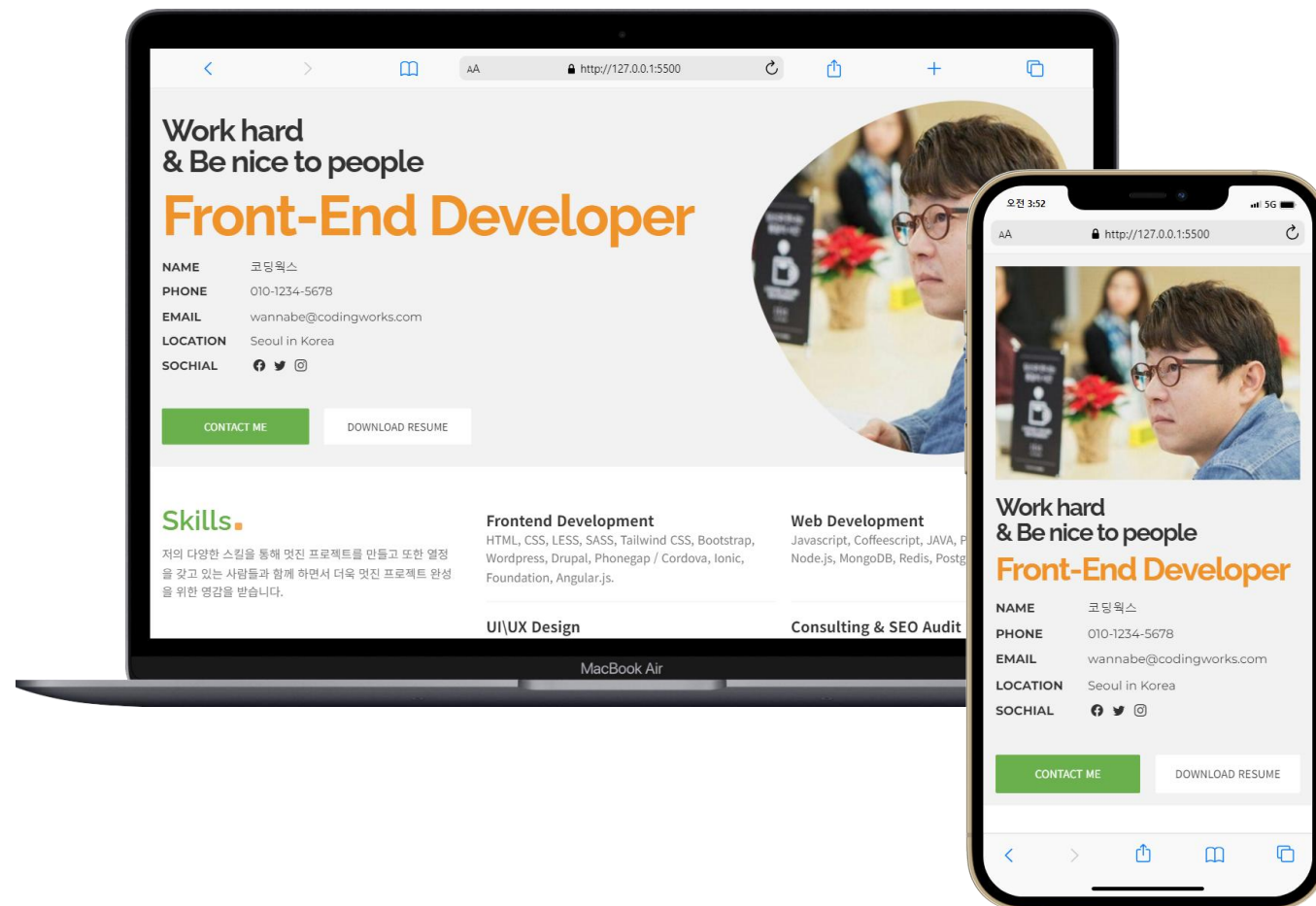
Tailwind CSS의 단점

- 학습을 위해 시간이 필요해 Learning Curve가 있음.(특히 개발자들에게...)
- 익숙해지기 전까지 초반 높은 문서 의존도 높음.
- 편리하고 빠르지만 지저분해 지는 HTML 코드로 코드 가독성이 많이 떨어짐.
- 미세한 CSS 디자인 곧, 복잡한 스타일에 제약
- CLI 설치 환경을 만들고 컴파일해야 하는 번거로움이 있다

[Tailwind CSS 강의소개] 코딩웍스 Tailwind CSS 강의 특징들

 코딩웍스(Coding Works)

- 경험 많은 퍼블리셔가 진행하는 Tailwind CSS 수업
- 충실한 내용과 예제를 바탕으로 16시간 이상의 알찬 수업내용과 수업량(수업시간이 긴 이유는..)
- Tailwind CSS 공식사이트에서 코드 복사해서 수업하지 않고 처음부터 모두 직접 코딩하는 방식
- Tailwind CSS 모든 이론내용을 다루고 개별적으로 자세하게 설명 후 코딩하면서 학습
- Tailwind CSS 이론을 바탕으로 경력자 퍼블리셔도 만족할 만한 다양하고 예쁜 웹 UI 실전 예제들 제작
- 개발자 취업과 경쟁력을 위해 Tailwind CSS로 개발자 경력 이력서 & 포트폴리오 소개 반응형 웹 제작(진짜 학습 결과물)



Tailwind CSS로 개발자가 만드는 멋진 UI 스타일링

Create a great web Ui Design styling with Tailwind CSS



Tailwind CSS를 시작하기 위한 필독 사항

🎯 [Tailwind CSS 핵심 이론] Tailwind CSS를 시작하기 위한 필독 사항

{ CDN 사용법과 비주얼스튜디오코드에서 Tailwind CSS IntelliSense 세팅 }

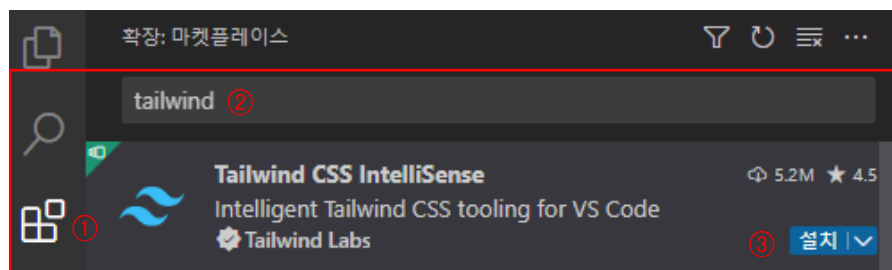
```
index.html
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <h1 class="text-3xl font-bold underline">
    Hello world!
  </h1>
</body>
</html>
```

<head>..</head> 사이에 복사해서 붙여넣기 합니다.

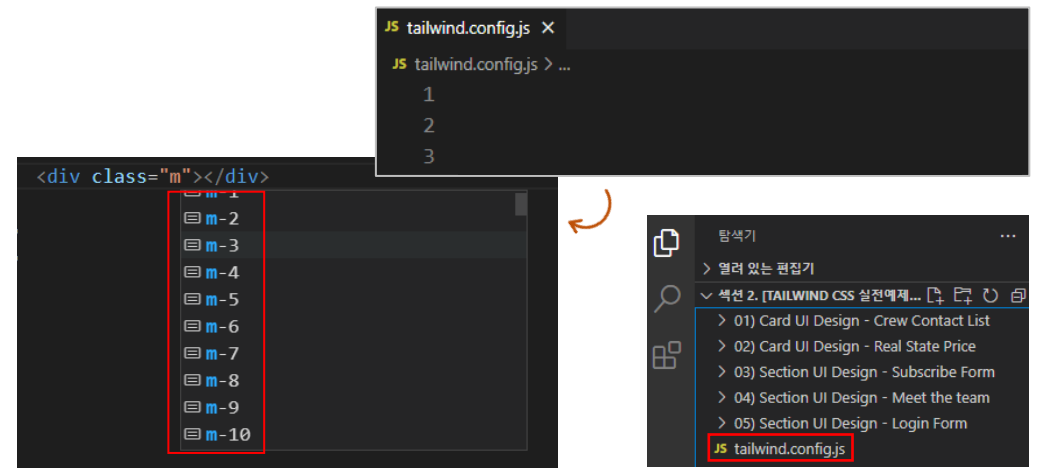
```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>Tailwind CSS IntelliSense 세팅</title>
  <!-- Tailwind CSS CDN -->
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div class="m"></div>
</body>
</html>
```

- 예를 들어 m이라고 치면 m으로 시작하는 Tailwind CSS의 모든 클래스 네임을 아래에 보여주기 때문에 쉽게 찾아 선택할 수 있습니다.
- 하지만 루트(root) 폴더에 config 파일을 만들어 주지 않으면 Tailwind CSS 클래스 네임을 아래에 보여주지 않기 때문에 **반드시 tailwind.config.js** 파일을 만들어야 합니다.
- **tailwind.config.js** 파일 내에 아무것도 없어도 루트에 동일한 파일이 있는 것을 체크해서 익스텐션이 작동합니다.
- 위의 절차가 완료되면 Tailwind CSS IntelliSense가 정상적으로 작동하지만 아닌 경우 비주얼스튜디오코드를 재실행 합니다.

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



- ① 확장 버튼(익스텐션 설치) 클릭
- ② tailwind 검색
- ③ 설치 클릭



{ Tailwind CSS의 기본이 되는 크기 단위 값, REM 단위와 비율 단위 }

REM 단위는 R(Root) 곧, html 또는 body에 설정되어 있는 폰트 사이즈를 기준으로 모든 것을 상대적(em)으로 결정되는 단위

- 좀 더 쉽게 이해해 보면.. html 또는 body의 폰트 사이즈가 16px이면 1rem은 16px이고, html 또는 body의 폰트 사이즈가 20px이면 1rem은 20px입니다.
- Tailwind CSS에서 기본적으로 16px을 기준으로 하므로 1rem은 16px이 됩니다. (0.75rem은 12px, 0.5rem은 8px, 0.25rem은 4px)

고정 너비 값(Fixed widths)

속성명-1	속성명 : 0.25rem;	/* 4px */
속성명-2	속성명 : 0.5rem;	/* 8px */
속성명-3	속성명 : 0.75rem;	/* 12px */
속성명-4	속성명 : 1rem;	/* 16px */

w-1	width: 0.25rem;	/* 4px */
w-2	width: 0.5rem;	/* 8px */
w-3	width: 0.75rem;	/* 12px */
w-4	width: 1rem;	/* 16px */

w는 width의 속성명의 첫 글자입니다. 예를 들어 margin 속성의 첫 글자는 m이고 padding은 p입니다. 곧, w에 m 또는 p를 넣으면 margin이나 padding을 조절하게 됩니다.

속성명 첫 글자-[단위] 속성명 : 단위; /* 단위 */

- w-[23px] → width: 23px;
- mt-[50%] → margin-top: 50%;

비율 너비 값(Percentage widths)

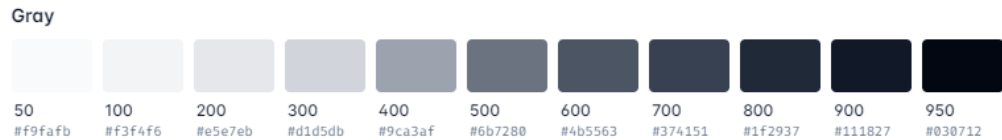
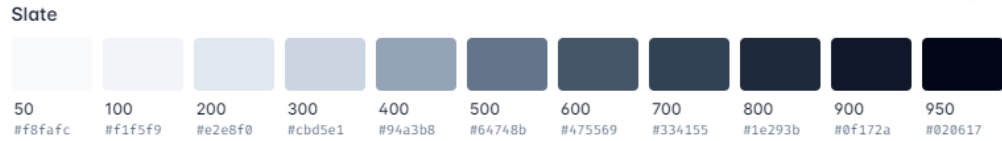
w-1/2	w-1/2
w-2/5	w-3/5
w-1/3	w-2/3
w-1/4	w-3/4
w-1/5	w-4/5
w-1/6	w-5/6
w-full	

w-분할비율 → w-1/4 이면 너비를 4분의 1로 나눈다 곧, 25%로 나누게 됩니다. w 대신에 h를 넣으면 높이를 분할비율로 나누는 것이 됩니다.

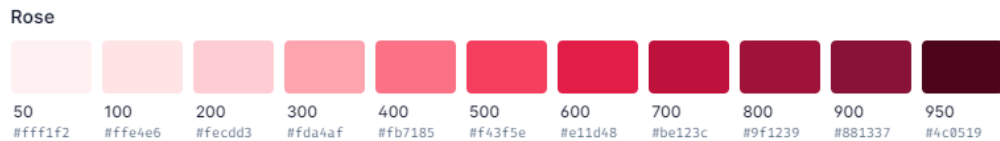
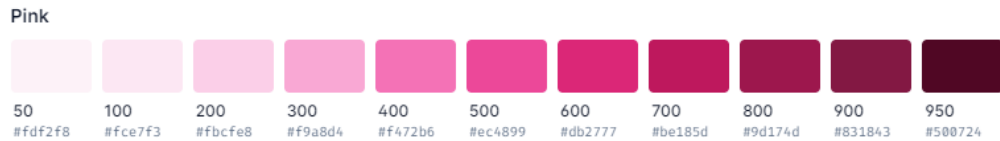
🎯 [Tailwind CSS 핵심 이론] Tailwind CSS를 잘 하기 위한 필독 사항(2)

{ Tailwind CSS에서 사용하는 색상 체계(Color System) }

↪ Tailwind CSS에서 Color System은 배경색을 포함해서 색상을 사용하는 모든 요소에서 동일하게 적용



숫자가 작아질수록 흐려 짐 ← → 숫자가 커질수록 진해 짐



▲ Tailwind CSS에서 클래스로 정해 놓은 색상은 총 16가지
(다양한 방법으로 Tailwind CSS가 만든 22가지 색상 외 모든 색상을 사용할 수 있음)
<https://tailwindcss.com/docs/customizing-colors>

🎨 색상이름 종류(22개)

slate, gray, zinc, neutral, stone, red, orange, amber, yellow, lime, green, emerald, teal, cyan, sky, blue, indigo, violet, purple, fuchsia, pink, rose

🎨 색상이름 다음에 50,100,200,300,400,500,600,700,800,900,950

ex) text-색상이름-50 또는 text-색상이름-950

ex) border-색상이름-500

ex) decoration-색상이름-500

ex) bg-색상이름-500

50~950 사이라고 만약 550을 사용하면 아무 변화가 없습니다. Tailwind CSS에서 550이라는 이름을 포함한 색상 값을 만들어 놓지 않았기 때문입니다. 반드시 11가지 단계의 숫자를 정확히 써야 합니다.

ex) text-pink-550 이렇게 사용하면 적용되지 않습니다.

👤 사용자 정의 설정

ex) text-[#2980b9]

ex) border-[crimson]

ex) decoration-[#f1c40f]

ex) bg-[yellowgreen]

👤 투명도 조절하기(RGBA)

ex) 검정색 bg-black → bg-[rgba(0,0,0,1)]

ex) 투명도 적용된 검정색 → bg-[rgba(0,0,0,0.5)]

띄어쓰기 불가

🎯 [Tailwind CSS 핵심 이론] Tailwind CSS를 잘 하기 위한 필독 사항(3)

{ Tailwind CSS에서 사용하는 방향에 관한 클래스 네임 체계 }

pt-단위 padding-top: 단위;

pr-단위 padding-right: 단위;

pb-단위 padding-bottom: 단위;

pl-단위 padding-left: 단위;

px-단위 padding-left: 단위;
padding-right: 단위;

py-단위 padding-top: 단위;
padding-bottom: 단위;

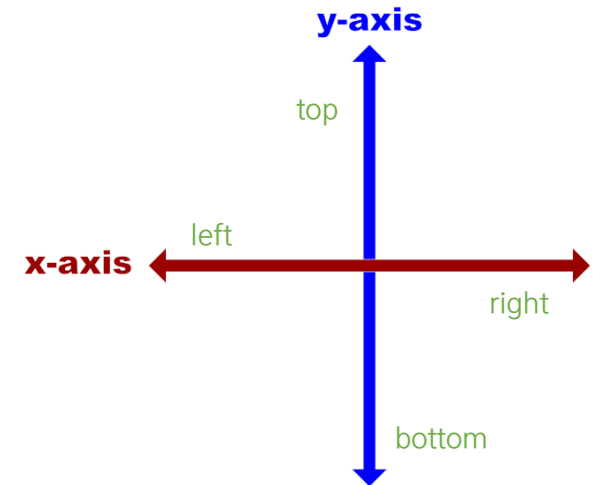
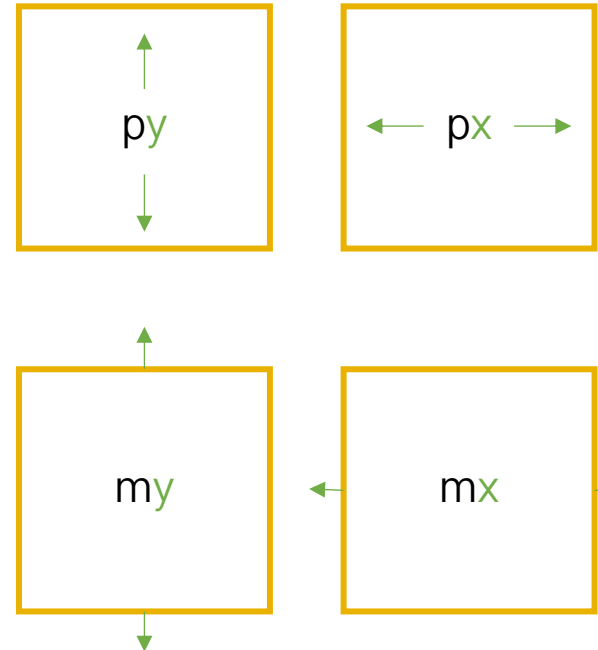
margin, border 등 4군데 방향을 사용하는 속성은 동일한 클래스 네임 시스템을 사용합니다.

x를 사용하면 x축을 기준으로
상하를 한번에 조절

y를 사용하면 y축을 기준으로
좌우를 한번에 조절

x축 또는 y축을 명시하지 않는 경우 한번에 상하좌우를 조절

- p-[20px]: 상하좌우 한번에 20픽셀 패딩 조절
- m-[20px]: 상하좌우 한번에 20픽셀 마진 조절



Tailwind CSS로 개발자가 만드는 멋진 UI 스타일링

Create a great web Ui Design styling with Tailwind CSS



Tailwind CSS 핵심이론

[Tailwind CSS 핵심 이론] 글자 서식 & 색상 - Typography

Font Size - 글자 크기



- **text-**문자(Tailwind에 지정된 이름)
ex) text-sm, text-lg, text-xl, text-9xl
- **text-**[단위]
ex) text-[14px]

Font Style - 글자 기울임



- **italic**: 글자 기울임
ex) italic
- **not-italic**: 글자 기울임 안함
ex) not-italic

Font Weight - 글자 굵기



- **font-**문자(Tailwind에 지정된 이름) or 숫자(100~900)
ex) font-thin, font-normal, font-bold
- **font-**[숫자]
ex) font-[600]

Letter Spacing - 글자 간격

- **tracking-**문자(Tailwind에 지정된 이름)
ex) tracking-tighter, tracking-wide
- **tracking-**[단위]
ex) tracking-[0.25em]

Line Clamp - 문단 끝 말 줄임



- **line-clamp-**숫자(1~6, Tailwind CSS가 정함)
ex) line-clamp-1, line-clamp-6
- **line-clamp-**[숫자]
ex) line-clamp-[7], line-clamp-[13]

Line Height - 줄 간격



- **leading-**숫자 or 문자(Tailwind에 지정된 이름)
ex) leading-3, leading-10, leading-loose
- **leading-**[단위]
ex) leading-[3rem], leading-[3px]

Boost your conversion rate

Nulla dolor velit adipisicing dui excepteur esse
in dui nostrud occaecat mollit incididunt
deserunt sunt. Ut ut sunt laborum ex occaecat...

[Tailwind CSS 핵심 이론] 글자 서식 & 색상 - Typography

List Style Type - 목록 불릿(bullet) 표시하기



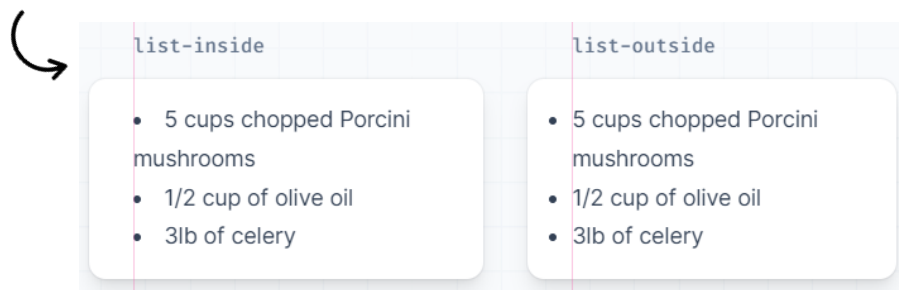
- **list-문자**(Tailwind에 지정된 이름)
ex) **list-disc**, **list-decimal**, **list-square**(작동불가)
- **list-[문자]**
ex) **list-[square]**, **list-[upper-alpha]**

태그	리스트 모양
ul	disc, circle, square
ol	decimal, decimal-leading-zero, lower-roman, upper-roman, lower-greek, lower-latin, upper-latin, armenian, georgian, lower-alpha, upper-alpha

▲ 위의 표처럼 ol, ul에 구분해서 사용할 필요는 없습니다. ol, ul 모두 적용됩니다.

List Style Position - 목록 Bullet을 안쪽으로 넣기

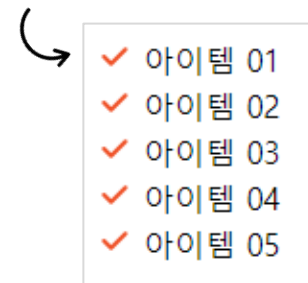
- **list-inside** : 목록 Bullet을 안쪽으로 넣기
- **list-outside** : 목록 Bullet을 바깥쪽으로 넣기
ex) **list-inside**, **list-outside**



List Style Image - 목록 Bullet을 이미지로 대체

- **list-image-[url('이미지 경로')]**
ex) **list-image-[url('img/icon.png')]**

▲ 이미지 경로를 둘러싼 따옴표는 생략 가능



[Tailwind CSS 핵심 이론] 글자 서식 & 색상 - Typography

Text Align - 글자 또는 문단의 수평 정렬



- **text-**문자(Tailwind에 지정된 이름)
ex) `text-center`, `text-left`, `text-right`, `text-justify`

Text Color - 글자 또는 문단의 색상



- **text-**색상-채도(Tailwind에 지정된 숫자)
ex) `text-blue-500`, `text-sky-950`
ex) `text-transparent` : 투명
ex) `text-black` : 검정색 / ex) `text-white` : 흰색
- **text-**[색상 값 or 색상 이름]
ex) `text-[#50d71e]`, `text-[yellowgreen]`
- **text-**색상-채도/백분율(Tailwind에 지정된 색상에 투명도)
ex) `text-blue-950/50` → 글자 색상 채도 950의 50%
ex) `text-blue-950/10` → 글자 색상 채도 950의 10%
- 투명도 조절은 `opacity` 속성으로도 조절 가능함

Text Decoration - 글자 또는 문단에 줄 긋기



- **underline** : 텍스트에 아래 라인 그리기
- **overline** : 텍스트에 위 라인 그리기
- **line-through** : 텍스트에 중앙 라인 그리기
- **no-underline** : 텍스트에 라인 그리기 않기



`underline`

The quick brown fox jumps over the lazy dog.

`overline`

The quick brown fox jumps over the lazy dog.

`line-through`

The quick brown fox jumps over the lazy dog.

`no-underline`

The quick brown fox jumps over the lazy dog.

[Tailwind CSS 핵심 이론] 글자 서식 & 색상 - Typography

Text Decoration Color - 글자 또는 문단에 줄 색상



- **decoration-색상-채도** (Tailwind에 지정된 숫자)
ex) `decoration-blue-500`, `decoration-sky-950`
ex) `decoration-transparent` : 투명
ex) `decoration-black` : 검정색 `decoration-white` : 흰색
- **decoration-[색상 값 or 색상 이름]**
ex) `decoration-[#50d71e]` `decoration-[yellowgreen]`

Text Decoration Style & Thickness & Offset

글자 또는 문단에 줄 스타일, 두께, 떨어짐 정도

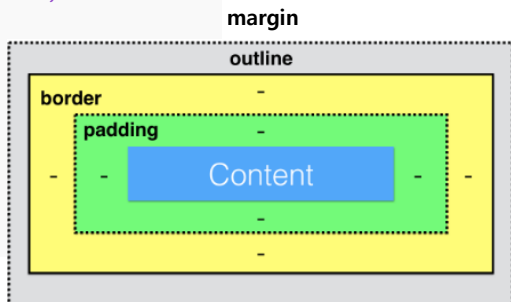


- **decoration-solid** : 텍스트에 실선 모양
- **decoration-double** : 텍스트에 이중선 모양
- **decoration-dotted** : 텍스트에 점선 모양
- **decoration-dashed** : 텍스트에 대쉬선 모양
- **decoration-wavy** : 텍스트에 물결선 모양

Outline Width, Color, Style - 아웃라인 테두리



- **outline-0(0, 1, 2, 4, 8)** - 아웃라인 두께
- **outline-none, outline, outline-dashed, outline-dotted, outline-double** (아웃라인 스타일)
- **decoration-색상-채도** (Tailwind에 지정된 숫자)
- **decoration-[색상 값 or 색상 이름]** - 아웃라인 색상
- **outline-offset-0(0, 1, 2, 4, 8)** - 아웃라인 떨어짐 정도



- **decoration-숫자** (Tailwind CSS가 정해 숫자)
ex) `decoration-1`, `decoration-8`
- **decoration-[단위]**
ex) `decoration-[3px]`

- **underline-offset-숫자** (Tailwind CSS가 정해 숫자)
ex) `underline-offset-0`, `underline-offset-8`

[Tailwind CSS 핵심 이론] 글자 서식 & 색상 - Typography

Text Transform - 영문 대소문자 변경

- **uppercase** : 대문자로 변경
- **lowercase** : 소문자로 변경
- **capitalize** : 단어 첫 단어를 대문자로 변경
- **normal-case** : 대소문자로 변경하지 않음

Text Overflow - 말 줄임

- **truncate** : 말 줄임

→

```
overflow: hidden;
text-overflow: ellipsis;
white-space: nowrap;
```

학교교육 및 평생교육을 포함한 교육제도와 그 운영, 교육재정 및 교원의 지위에 관한 기본적인 사항은 법률로 정한다. 국회는 상호원조 또는 안전보장에 관한 조약, 중요한 국제조직에 관한 조약.

→ 학교교육 및 평생교육을 포함한 교육제도와 그 운영, 교육재정 및 교원의 지위에 ...

Text Wrap - 줄 바꿈 설정

- **text-nowrap** : 부모요소 너비에 넘치는 내용 줄 바꿈 안함

CodingWorks Online Class

Lorem ipsum dolor sit amet consectetur adipisicing elit. Reprehenderit ullam error minus hic cum est dignissimos

nec
corr

CodingWorks Online Class

Lorem ipsum dolor sit amet consectetur adipisicing elit. Reprehenderit ullam error minus hic cum es

CodingWorks Online Class

Lorem ipsum dolor sit amet consectetur adipisicing elit. f

Vertical Align - 인라인블록 요소 수직 정렬

- **align-top** : 상단 정렬
- **align-middle** : 중앙 정렬
- **align-bottom** : 하단 정렬

[Tailwind CSS 핵심 이론] 글자 서식 & 색상 - Typography

Font Family - 원하는 글꼴 사용하기



대통령은 내우·외환·천재·지변 또는 중대한 재정·경제상의 위기에 있어서 국가의 안전보장 또는 공공의 안녕질서를 유지하기 위하여 긴급한 조치가 필요하고 국회의 집회를 기다릴 여유가 없을 때에 한하여 최소한으로 필요한 재정·경제상의 처분을 하거나 이에 관하여 법률의 효력을 가지는 명령을 발할 수 있다.

→ Noto Sans KR

CodingWorks Tailwind CSS Online Class

대통령은 내우·외환·천재·지변 또는 중대한 재정·경제상의 위기에 있어서 국가의 안전보장 또는 공공의 안녕질서를 유지하기 위하여 긴급한 조치가 필요하고 국회의 집회를 기다릴 여유가 없을 때에 한하여 최소한으로 필요한 재정·경제상의 처분을 하거나 이에 관하여 법률의 효력을 가지는 명령을 발할 수 있다.

→ Noto Sans KR

CodingWorks Tailwind CSS Online Class

대통령은 내우·외환·천재·지변 또는 중대한 재정·경제상의 위기에 있어서 국가의 안전보장 또는 공공의 안녕질서를 유지하기 위하여 긴급한 조치가 필요하고 국회의 집회를 기다릴 여유가 없을 때에 한하여 최소한으로 필요한 재정·경제상의 처분을 하거나 이에 관하여 법률의 효력을 가지는 명령을 발할 수 있다.

→ Single Day

CodingWorks Tailwind CSS Online Class

대통령은 내우·외환·천재·지변 또는 중대한 재정·경제상의 위기에 있어서 국가의 안전보장 또는 공공의 안녕질서를 유지하기 위하여 긴급한 조치가 필요하고 국회의 집회를 기다릴 여유가 없을 때에 한하여 최소한으로 필요한 재정·경제상의 처분을 하거나 이에 관하여 법률의 효력을 가지는 명령을 발할 수 있다.

→ Montserrat

CodingWorks Tailwind CSS Online Class

대통령은 내우·외환·천재·지변 또는 중대한 재정·경제상의 위기에 있어서 국가의 안전보장 또는 공공의 안녕질서를 유지하기 위하여 긴급한 조치가 필요하고 국회의 집회를 기다릴 여유가 없을 때에 한하여 최소한으로 필요한 재정·경제상의 처분을 하거나 이에 관하여 법률의 효력을 가지는 명령을 발할 수 있다.

→ Black Han Sans

대통령은 내우·외환·천재·지변 또는 중대한 재정·경제상의 위기에 있어서 국가의 안전보장 또는 공공의 안녕질서를 유지하기 위하여 긴급한 조치가 필요하고 국회의 집회를 기다릴 여유가 없을 때에 한하여 최소한으로 필요한 재정·경제상의 처분을 하거나 이에 관하여 법률의 효력을 가지는 명령을 발할 수 있다.

→ YClover-Bold

```
tailwind.config = {
  theme: {
    extend: {
      fontFamily: {
        sans: ['Noto Sans KR', 'Arial', 'sans-serif'],
        // sans가 기본 상속 폰트이므로 전체 폰트바꾸려면
        // sans 재지정 후 맨앞에 원하는 폰트 넣기
        notosans: ['Noto Sans KR', 'sans-serif'],
        singleday: ['Single Day', 'cursive'],
        montserrat: ['Montserrat', 'sans-serif'],
        blackhansans: ['Black Han Sans', 'sans-serif'],
        yclover: ['YClover-Bold']
      }
    }
  }
}
```

▲ tailwind.config.js

```
/* Google Web Fonts CDN */
@import url('https://fonts.googleapis.com/css2?family=Noto+Sans+KR:wght@400;500&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Single+Day&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@300;400&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Black+Han+Sans&display=swap');
```

```
/* Noonu Web Fonts CDN */
@font-face {
  font-family: 'YClover-Bold';
  src:
    url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_231029@1.1/YClover-Bold.woff2') format('woff2');
  font-weight: 700;
  font-style: normal;
}
```

▲ style.css

- ① tailwind.config.js에 사용할 폰트 생성
- ② style.css에 사용할 웹 폰트 CDN 생성
- ③ index.html에 tailwind.config.js와 style.css 링크
- ④ HTML에서 사용할 곳에 클래스 네임 입력

```
<head>
<meta charset="UTF-8">
<title>Font Family(한글 영문 커스텀 폰트 사용하기)</title>
<link href="css/style.css" rel="stylesheet">
<!-- Tailwind CSS CDN -->
<script src="https://cdn.tailwindcss.com"></script>
<!-- tailwind config Import -->
<script src="script/tailwind.config.js"></script>
</head>
```

▲ index.html

```
<!-- 특정 폰트를 지정한 경우 : notosans -->
<p class="my-5 font-notosans"> ... </p>
```

▲ font-설정된 폰트이름

구글 웹 폰트 또는 노누 폰트에서 사용할 폰트 CDN을 복사해서 붙여넣기 합니다.

Border Radius - 모서리 둥글게 하기



- **rounded**-문자(Tailwind에 지정된 이름)
ex) rounded-md, rounded-lg, rounded-full
- **rounded**-[단위]
ex) rounded-[50px]
- **rounded**-방향-문자 **rounded**-방향-[단위]
ex) rounded-t-lg, rounded-b-lg, rounded-r-lg
ex) rounded-t-[50px]

Border Width - 테두리 두께



- **border**-숫자(Tailwind에 지정된 숫자)
ex) border-0, border, border-2, border-8
- **border**-[단위] **border**-방향-숫자
ex) border-[5px]
ex) border-x-2, border-y-4, border-x-[3px]

Border Style - 테두리 모양



- **border-solid** : 실선 테두리
- **border-dashed** : 대쉬 테두리
- **border-dotted** : 점선 테두리
- **border-double** : 이중선 테두리
- **border-none** : 테두리 없음

Border Color - 테두리 색상



- **border**-색상-채도(Tailwind에 지정된 숫자)
ex) border-blue-500, border-sky-900
ex) border-transparent : 투명, border-black : 검정색, border-white : 흰색
- **border**-[색상 값 or 색상 이름]
ex) border-[#50d71e]
ex) border-[crimson]

[Tailwind CSS 핵심 이론] 간격과 크기 - Spacing & Sizing

padding - 요소 안쪽의 여백(보더 안쪽)

- p-0, px-0, py-0
- pr-0, pl-0, pb-0, pt-0
- p-[5px] px-[10px] py-[15px]

margin - 요소 바깥쪽의 여백(보더 바깥쪽)

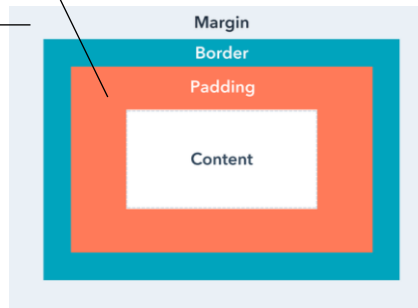
- m-0, mx-0, my-0
- mr-0, ml-0, mb-0, mt-0
- m-[5px] mx-[10px] my-[15px]

Width - 너비

- w-0, w-full, w-screen
- w-1에서 w-96 w-1/2에서 w-11/12
- w-[200px], w-[35%]

Height - 높이

- h-0, h-full, h-screen
- h-1에서 h-96
- h-1/2에서 h-11/12
- h-[200px], h-[35%]



px-[단위] : 좌우 패딩
py-[단위] : 상하 패딩
mx-[단위] : 좌우 마진
my-[단위] : 상하 마진

Size - 정사각형 만들기

- size-0 width: 0px; height: 0px;
- size-0.5 ~ size-96 0.125rem; /* 2px */ ~ 24rem; /* 384px */
- size-1/2 ~ size-11/12 50% ~ 91.666667%;
- size-full width: 100%; height: 100%;
- size-[단위] ex) size-[500px]



Min Width & Min Height - 요소의 최소 최대 너비와 높이

- min-w-[단위] ex) min-w-[300px]
- min-h-[단위] ex) min-h-[300px]
- max-w-[단위] ex) max-w-[600px]
- max-h-[단위] ex) max-h-[200px]

▲ Min Width & Min Height는 당장 학습할 필요는 없고 필요한 경우 학습하면 됩니다.

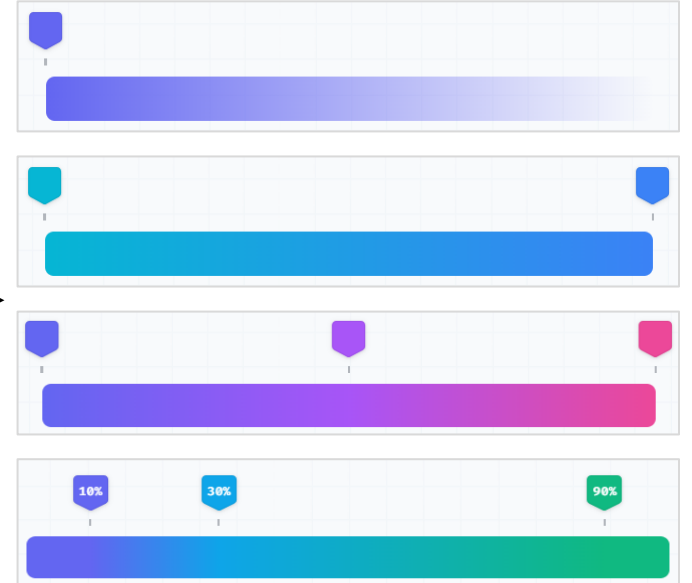
[Tailwind CSS 핵심 이론] 배경 색상 - Background Color ✖

Background Color - 배경 색상

- `bg-transparent`, `bg-black`, `bg-white`
- `bg-색상-채도`
ex) `bg-rose-800`
- `bg-[색상값 or 색상이름]`
ex) `bg-[#50d71e]` `bg-[crimson]`
- `bg-색상-채도/숫자(1~100)`
ex) `bg-rose-800/50` → 투명도 50%

Gradient Color - 그라디언트 만들기

- `from-색상-채도` → 시작 색상
- `to-색상-채도` → 종료 색상
- `via-색상-채도` → 중간 색상
- `bg-gradient-to-방향`
ex) `bg-gradient-to-r` → 왼쪽에서 오른쪽 방향으로
- `from-퍼센트`
ex) `from-10%` → 10%까지 퍼짐



Background Image - 배경 이미지

- `bg-[url('이미지 경로')]`
ex) `bg-[url('img/pattern-01.png')]`

배경 이미지 넣고 사용할 수 있는 속성

Background Repeat, Size, Position, Attachment

- Background Size : 배경 이미지 채우기 → `bg-cover`, `bg-contain`
- Background Repeat : 배경 이미지 반복 → `bg-no-repeat`, `bg-repeat-x`, `bg-repeat-y`
- Background Position : 배경 이미지 위치 → `bg-bottom`, `bg-center`, `bg-left` ~ `bg-top`
- Background Attachment : 배경 이미지 고정 → `bg-fixed`

[Tailwind CSS 핵심 이론] Layout & Position ✨

Aspect Ratio - 비율 만들기

- `aspect-square` - `aspect-ratio: 1 / 1;`
- `aspect-video` - `aspect-ratio: 16 / 9;`
- `aspect-[4/3]`

Columns - 자동 컬럼 만들기

- `columns-1` 에서 `columns-12`

Box Sizing - 너비와 높이에 패딩,보더 포함

- `box-border` : 너비와 높이에 패딩,보더 포함
- `box-content` : 너비와 높이에 패딩,보더 포함 안함

Display - 요소의 특성

- `block` : 블록 요소로 변경
- `inline-block` : 인라인블록 요소로 변경
- `inline` : 인라인 요소로 변경
- `hidden` : 요소 사라지게 하기
- `flex` : 플렉스 선언하기
- `grid` : 그리드 선언하기

Float & Clear - 우측 좌측 배치

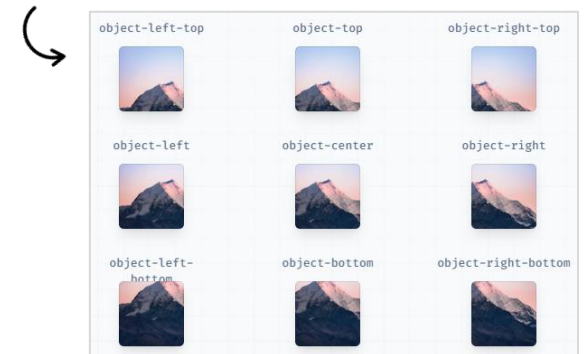
- `float-right` - `float: right;`
- `float-left` - `float: left;`
- `float-none` - `float: none;`
- `clear-left` - `clear: left;`
- `clear-right` - `clear: right;`
- `clear-both` - `clear: both;`

Object Fit - 이미지 부모요소 맞추기

- `object-cover` - `object-fit: cover;`

Object Position - 이미지 위치 이동

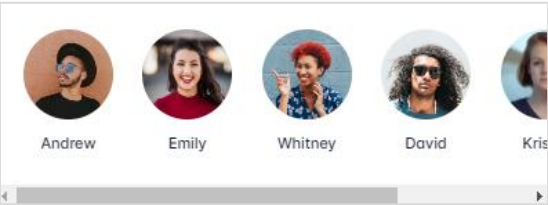
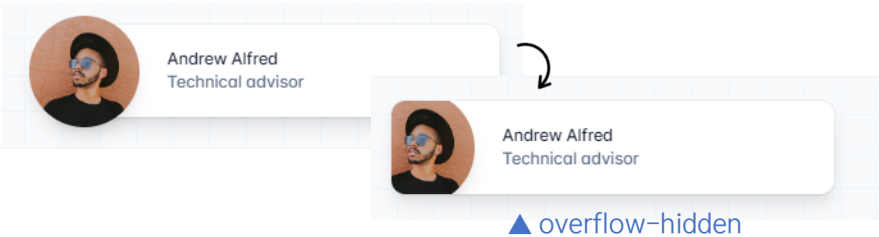
- `object-bottom`, `object-center`, `object-left`, `object-left-bottom`, `object-left-top`, `object-right`, `object-right-bottom`, `object-right-top`, `object-top`



[Tailwind CSS 핵심 이론] Layout & Position ✨

Overflow – 요소 밖으로 넘친 것 가리기

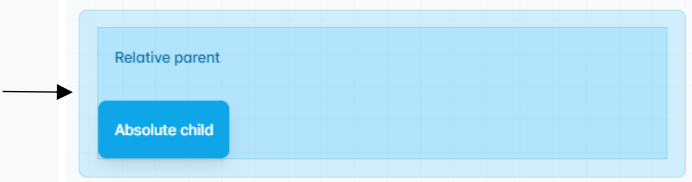
- `overflow-auto` – `overflow: auto;`
- `overflow-hidden` – `overflow: hidden;`
- `overflow-x-hidden` – `overflow-x: hidden;`
- `overflow-y-hidden` – `overflow-y: hidden;`



▲ `overflow-x-scroll`

Position – 요소의 절대 위치 & 상대 위치

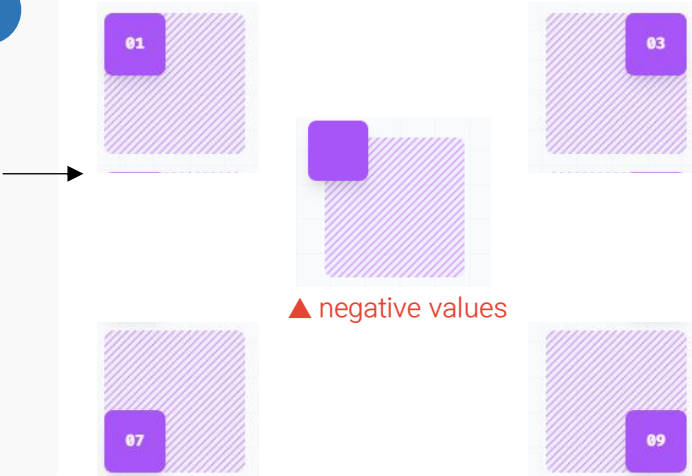
- `static(기본값)` – `position: static;`
- `fixed` – `position: fixed;`
- `absolute` – `position: absolute;`
- `relative` – `position: relative;`
- `sticky` – `position: sticky;`



- 부모요소에 `relative`, 자식요소에 `absolute`
- 부모요소에 `absolute`가 있으면 이미 부모요소가 됨

Position – 요소의 절대 위치 & 상대 위치

- `top-0` – `top: 0px;`
- `right-0` – `right: 0px;`
- `bottom-0` – `bottom: 0px;`
- `left-0` – `left: 0px;`
- `top-[단위]`
ex) `top-[3px]` `top-[-50px]` `-top-[50px]`
ex) `left-[50%]` `top-[-100%]`



▲ `top`, `right`, `bottom`, `left`, `z-index`는 포지션 속성이 있어야만 작동합니다.

[Tailwind CSS 핵심 이론] Layout & Position ✨

Visibility - 요소 안보이게 하기

- **visible** : `visibility: visible;`
- **invisible** : `visibility: hidden;`

Z-Index - 요소 쌓는 순서

- **z-0** `z-index: 0;`
- **z-10** `z-index: 10;`
- **z-20** `z-index: 20;`
- **z-30** `z-index: 30;`
- **z-40** `z-index: 40;`
- **z-50** `z-index: 50;`
- **z-auto** `z-index: auto;`
- **z-[숫자]** ex) `z-[100]`



Opacity - 요소 투명하게 하기

- **opacity-0 ~ opacity-100**
- **opacity-[숫자 또는 백분율]**
ex) `opacity-[0.67]`
ex) `opacity-[50%]`

요소를 안보이게 하는 3가지 방법과 차이점

- **hidden** `display: none;`
- **invisible** `visibility: hidden;`
- **opacity-0** `opacity: 0;`

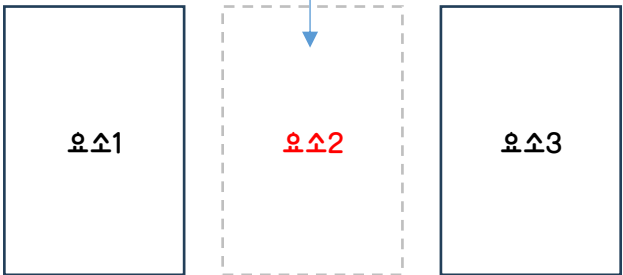


표 만들기 - Table 관련 속성

- **border-collapse**(보더 합치기) `border-collapse: collapse;`
- **border-separate** `border-collapse: separate;`
- **border-spacing-0 ~ border-spacing-96**
- **border-spacing-[단위]** ex) `border-spacing-[10px]`
- **border-spacing-x-[단위]** ex) `border-spacing-x-[5px]`
- **border-spacing-y-[단위]** ex) `border-spacing-y-[5px]`
- **table-auto** `table-layout: auto;`
- **table-fixed**(너비 일정하게 배분) `table-layout: fixed;`
- **caption-top**(캡션 표 위로 배치) `caption-side: top;`
- **caption-bottom**(캡션 표 아래 배치) `caption-side: bottom;`

State	City
Indiana	Indianapolis
Ohio	Columbus
Michigan	Detroit

[Tailwind CSS 핵심 이론] 그림자, 투명도 - Box Shadow, Opacity

Box Shadow - 도형 그림자 만들기

- `shadow-sm` `box-shadow: 0 1px 2px 0 rgb(0 0 0 / 0.05);`
- `shadow` `box-shadow: 0 1px 3px 0 rgb(0 0 0 / 0.1), 0 1px 2px -1px rgb(0 0 0 / 0.1);`
- `shadow-md` `box-shadow: 0 4px 6px -1px rgb(0 0 0 / 0.1), 0 2px 4px -2px rgb(0 0 0 / 0.1);`
- `shadow-lg` `box-shadow: 0 10px 15px -3px rgb(0 0 0 / 0.1), 0 4px 6px -4px rgb(0 0 0 / 0.1);`
- `shadow-xl` `box-shadow: 0 20px 25px -5px rgb(0 0 0 / 0.1), 0 8px 10px -6px rgb(0 0 0 / 0.1);`
- `shadow-2xl` `box-shadow: 0 25px 50px -12px rgb(0 0 0 / 0.25);`
- `shadow-inner` `box-shadow: inset 0 2px 4px 0 rgb(0 0 0 / 0.05);`
- `shadow-none` `box-shadow: 0 0 #0000;`

Text Shadow - 텍스트 그림자 만들기

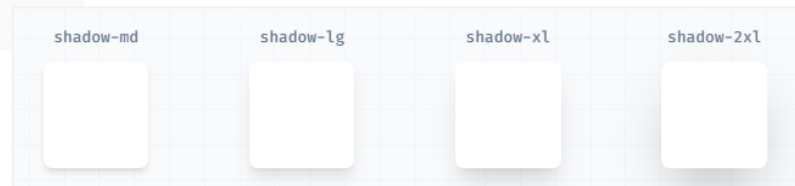
- `[text-shadow:x축이동_y축이동_퍼짐정도_색상]`
ex) `[text-shadow:5px_5px_0_orange]`
ex) `[text-shadow:-5px_-5px_0_crimson]`
ex) `[text-shadow:0_0_20px_black]`
ex) `[text-shadow:0_0_20px_#000000]`
ex) `[text-shadow:0_0_20px_rgba(0,0,0,0.7)]`

[참고사항] 텍스트 쉐도우

- 띄어쓰기 사용하면 작동 안함(띄어쓰기 금지)
- Tailwind CSS v3.4.1 공식사이트에서는 텍스트 쉐도우 사용법은 볼 수 없음

Box Shadow Color - 그림자 색상

- `shadow-black`
- `shadow-white`
- `shadow-색상-채도`
ex) `shadow-rose-500`
ex) `shadow-rose-500/30`
- `shadow-[색상 값 or 색상 이름]`
ex) `shadow-[#333333]`



Text with shadow
Text with shadow

[Tailwind CSS 핵심 이론] 트랜스폼 - Transforms

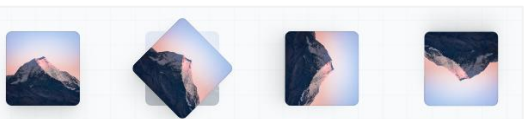
Scale - 크기

- `scale-0` `transform: scale(0);`
- `scale-x-0` `transform: scaleX(0);`
- `scale-y-0` `transform: scaleY(0);`
- `scale-[숫자]` ex) `scale-[1.7]`
- `scale-x-[숫자]` ex) `scale-x-[1.5]`
- `scale-y-[숫자]` ex) `scale-y-[2]`



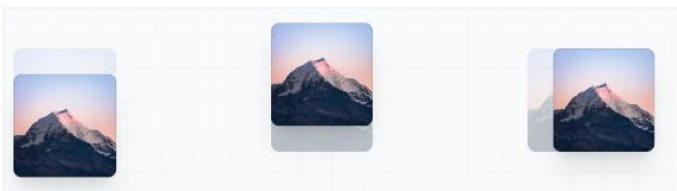
Rotate - 회전

- `rotate-[숫자]` ex) `rotate-[45deg]`
- `rotate-[-숫자]` ex) `rotate-[-45deg]`



Translate - 이동

- `translate-x-0` `transform: translateX(0px);`
- `translate-y-0` `transform: translateY(0px);`
- `translate-x-[숫자]` ex) `translate-x-[100px]`
- `translate-y-[-숫자]` ex) `translate-y-[-100%]`



Skew - 기울기

- `skew-x-0` `transform: skewX(0deg);`
- `skew-y-0` `transform: skewY(0deg);`
- `skew-x-[각도]` ex) `skew-x-[17deg]`
- `skew-y-[각도]` ex) `skew-y-[-17deg]`



Transform Origin - 회전과 크기 시작점

- `origin-center` `transform-origin: center;`
- `origin-top` `transform-origin: top;`
- `origin-top-right` `transform-origin: top right;`
- `origin-right` `transform-origin: right;`
- `origin-bottom-right` `transform-origin: bottom right;`
- `origin-bottom` `transform-origin: bottom;`
- `origin-bottom-left` `transform-origin: bottom left;`
- `origin-left` `transform-origin: left;`
- `origin-top-left` `transform-origin: top left;`
- `origin-[백분율_백분율]` ex) `origin-[33%_75%]`



[Tailwind CSS 핵심 이론] Transitions & Animation

Transition - 변화 과정 보여주기

- `transition-all` `transition-property: all;` → 트랜지션 여부 결정
- `duration-0 ~ duration-1000`
- `duration-[숫자ms]` ex) `duration-[2000ms]` → 트랜지션 지속시간
- `ease-linear` `transition-timing-function: linear;` → 트랜지션 속도변화
- `ease-in` `transition-timing-function: cubic-bezier(0.4, 0, 1, 1);`
- `ease-out` `transition-timing-function: cubic-bezier(0, 0, 0.2, 1);`
- `ease-in-out` `transition-timing-function: cubic-bezier(0.4, 0, 0.2, 1);`
- `delay-0 ~ delay-1000`
- `delay-[숫자ms]` ex) `delay-[2000ms]` → 트랜지션 지연시간

Animation - CSS 애니메이션

- `animate-spin` 회전 애니메이션
- `animate-ping` 퍼지는 듯한 애니메이션
- `animate-pulse` 맥박 뛰는 듯한 애니메이션
- `animate-bounce` 아래로 바운스 되는 애니메이션

• `animate-[애니메이션이름_지속시간_이징_반복여부]`
ex) `animate-[wiggle_1s_ease-in-out_infinite]`

Tailwind CSS에서 지정된 애니메이션은 매우 제한적입니다.

다양한 CSS 애니메이션을 적용하기 위해서는 CSS 파일에 CSS 키프레임 애니메이션을 만들고 예시처럼 html에 클래스를 적용하면 가능합니다.

Save Changes

Save Changes





[Tailwind CSS 핵심 이론] 상호작용 - Interactivity

Accent Color - 체크박스, 라디오버튼 디자인

- `accent-transparent`, `accent-black`, `accent-white`
- `accent-색상-채도`
ex) `accent-rose-800`
- `accent-[색상값 or 색상이름]`
ex) `accent-[#50d71e]` `accent-[crimson]`

☒ Browser default ☒ Customized

Cursor - 커서 포인터 변경

- `cursor-pointer` `cursor: pointer;` 
- `cursor-wait` `cursor: wait;` 
- `cursor-help` `cursor: help;` 
- `cursor-not-allowed` `cursor: not-allowed;` 

▲ 커서 모양은 매우 다양하지만 꼭 필요한 것들만 기억하세요.

Pointer Events - 마우스 이벤트 받기

- `pointer-events-none` `pointer-events: none;`
- `pointer-events-auto` `pointer-events: auto;`

`pointer-events-auto`

Q Search

`pointer-events-none`

Q Search

Appearance - 폼 요소 기본모양

- `appearance-none` `appearance: none;`

Yes

Default browser styles applied

Yes

Remove default browser styles

User Select - 마우스 드래그 텍스트 선택

- `select-none` `user-select: none;`

The quick brown fox jumps over the lazy dog.

Resize - textarea 리사이즈

- `resize-none` `resize: none;`
- `resize-y` `resize: vertical;`
- `resize-x` `resize: horizontal;`
- `resize` `resize: both;`

Scroll Behavior - 부드럽게 스크롤

- `scroll-smooth` `scroll-behavior: smooth;`

▲ a태그를 클릭했을 때 a태그의 id 이름과 같은 곳을 찾아 갈 때 부드럽게 스크롤링 되면서 이동하기

[Tailwind CSS 핵심 이론] 플렉스 레이아웃 - Flex

부모요소에 쓰는 속성

- `display: flex`
- `flex-direction`
- `gap`
- `flex-wrap`
- `justify-content`
- `align-items`
- `flex-wrap`
- `align-content`

자식요소에 쓰는 속성

- `flex & grow`
- `order`
- `align-self`

```
<div class="flex">
  <div>01</div>
  <div>02</div>
  <a href="#none">03</a>
  <span>04</span>
  <p>05</p>
</div>
```

→ 부모요소
(flex 클래스 주면 플렉스 배치 시작)

→ 자식요소

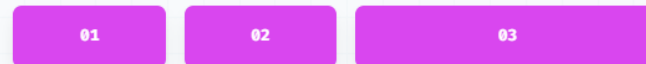
▲ 부모요소에 플렉스 선언되면 태그와 관계없이 모든 자식요소는 마진 없는 인라인블록 요소가 됩니다.

Flex - 자식요소 너비 자동 설정

- `flex-1` `flex: 1 1 0%;`
- `flex-none` `flex: none;`

Flex Basis - 자식요소 너비 고정 설정

- `basis-0` `flex-basis: 0px;`
- `basis-1/2 ~ basis-11/12`
- `basis-full` `flex-basis: 100%;`
- `basis-[단위]`
ex) `basis-[200px]`



```
<div class="flex flex-row">
  <div class="basis-1/4">01</div>
  <div class="basis-1/4">02</div>
  <div class="basis-1/2">03</div>
</div>
```

Flex Grow - 너비를 비율로

- `grow` `flex-grow: 1;`
- `grow-0` `flex-grow: 0;`
- `grow-[정수]` ex) `grow-[2]`



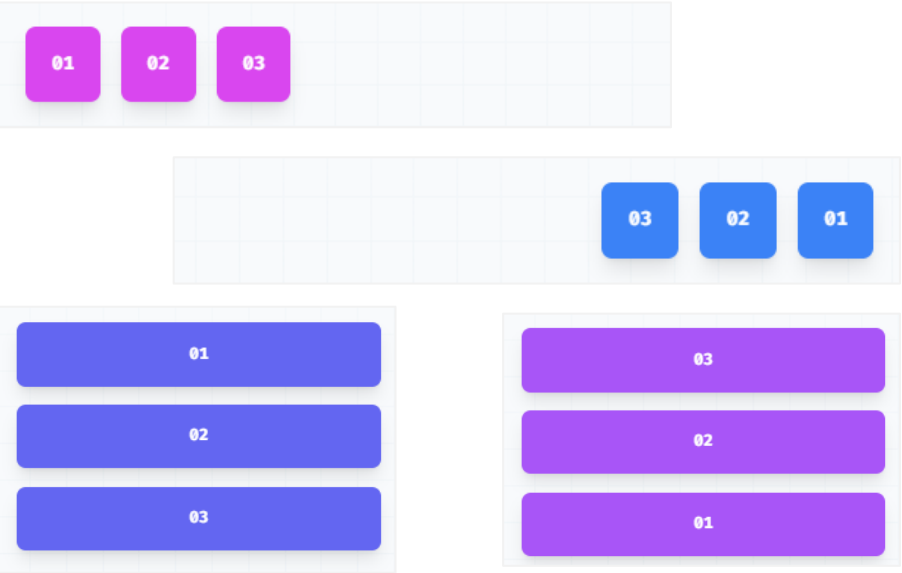
Order - 순서 설정

- `order-1 ~ order-12`
- `order-[정수]` ex) `order-[13]`
- `order-first` `order: -9999;`
- `order-last` `order: 9999;`



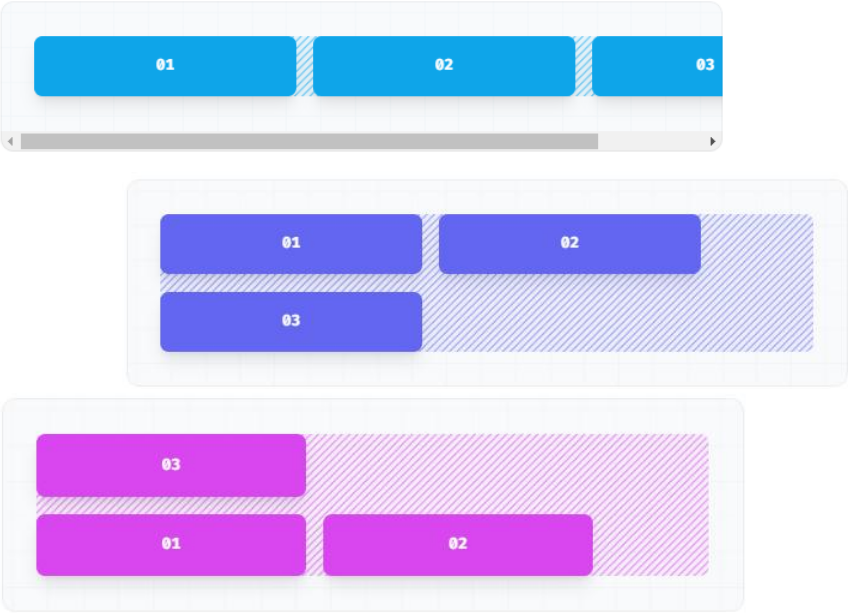
Flex Direction - 배치 방향(가로, 세로)

- flex-row(기본값) flex-direction: row;
- flex-row-reverse flex-direction: row-reverse;
- flex-col flex-direction: column;
- flex-col-reverse flex-direction: column-reverse;



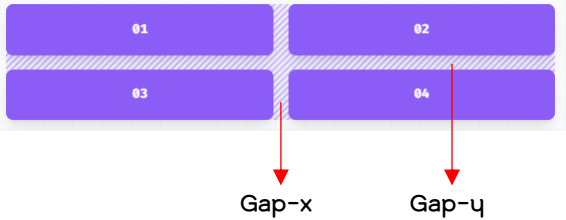
Flex Wrap - 부모요소 너비에 줄 바꿈 여부

- flex-wrap flex-wrap: wrap;
- flex-wrap-reverse flex-wrap: wrap-reverse;
- flex-nowrap(기본값) flex-wrap: nowrap;



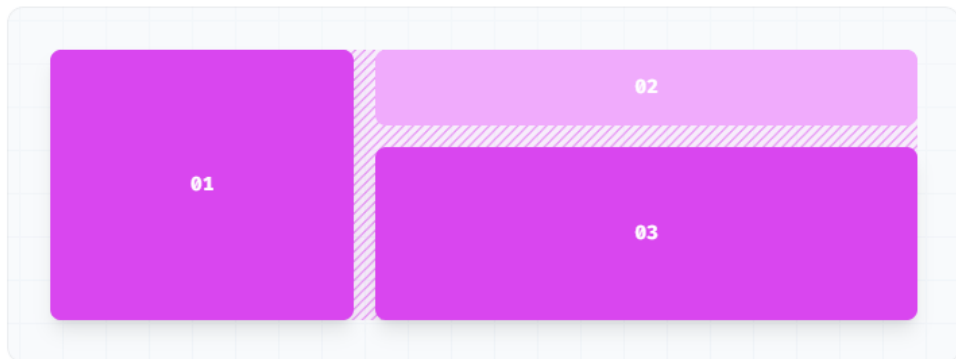
Gap - 자식요소 간격

- gap-0 gap: 0px;
- gap-x-0 column-gap: 0px;
- gap-y-0 row-gap: 0px;
- gap-1 ~ gap-96
- gap-[단위]
ex) gap-[20px]



[Tailwind CSS 핵심 이론] [Flex, Grid] - Flex & Grid 차이 이해하기

- 실무에서 만들게 되는 대부분의 레이아웃은 Flex로 충분히 구현합니다. Grid의 경우 사용 빈도는 적지만 아주 특별한 레이아웃에 사용하면 됩니다.
- Grid는 사용해봤다... 사용할 수 있다... 정도면 충분합니다. 대부분의 실무 작업은 Flex 레이아웃 배치로 충분합니다.



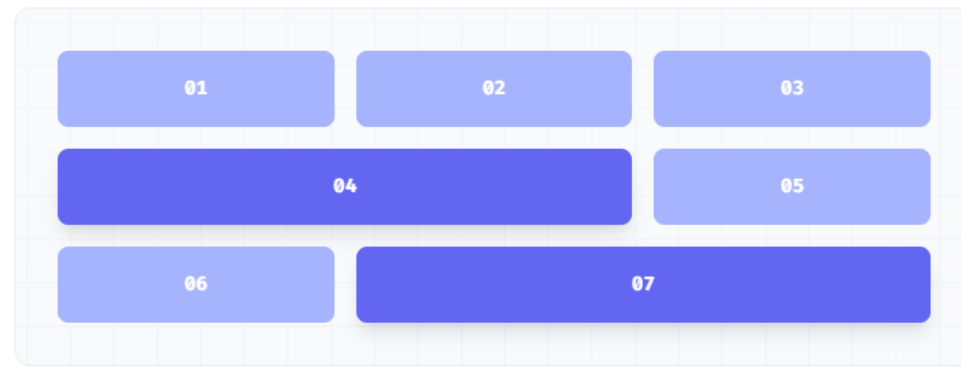
▲ 가장 많이 제작하는 위의 레이아웃을 만들기 위해 Flex와 Grid 모두 가능합니다. 단, Flex의 경우 html 구조가 하나 더 필요하고 Grid의 경우 그렇지 않은 것이 가장 큰 특징입니다.

```
<div class="entire-wrap">
  <div>01</div>
  <div class="sub-wrap">
    <div>02</div>
    <div>03</div>
  </div>
</div>
```

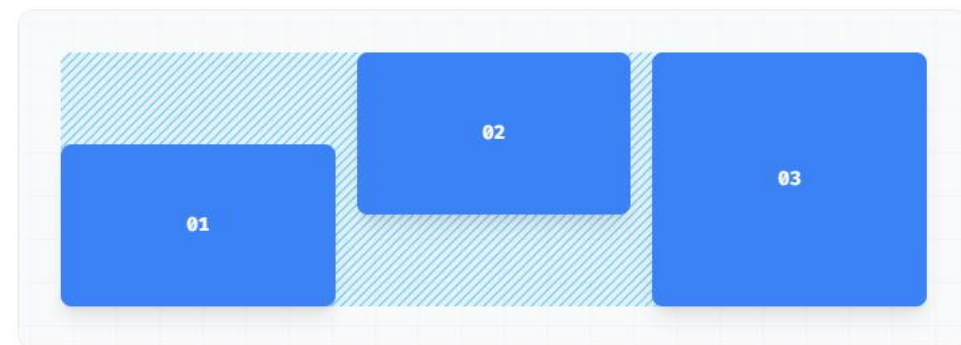
- Flex의 경우** .entire-wrap이 자식요소 div 2개를 가로 배치하고 나서 .sub-wrap이라는 부모요소 div가 필요합니다. 그리고 .sub-wrap안에 div 2개는 블록 요소이므로 그냥 놔두면 자동으로 세로 배치 됩니다.
- Flex의 경우** 자식요소의 너비를 정해주어야 함. ex) flex-1

```
<div class="entire-wrap">
  <div>01</div>
  <div>02</div>
  <div>03</div>
</div>
```

- Grid의 경우** .entire-wrap이 자식요소 div 3개를 가로 배치하고 첫째 div를 아래로 확장해서 Flex처럼 .sub-wrap이라는 부모요소 div가 필요하지 않습니다.
- Grid의 경우** 기본적으로 자식요소는 좌우로 펼쳐지는 stretch 상태가 됨



▲ 위의 레이아웃 예시와 같은 경우 Flex로 만들게 되면 html 구조가 조금 더 복잡해 집니다. (04,05 부모요소로 묶기 / 06, 07 부모요소로 묶기)



▲ 위의 레이아웃 예시와 같은 경우 Flex만으로는 불가능합니다. 하지만 위의 레이아웃을 만들 경우는 흔하지 않습니다.

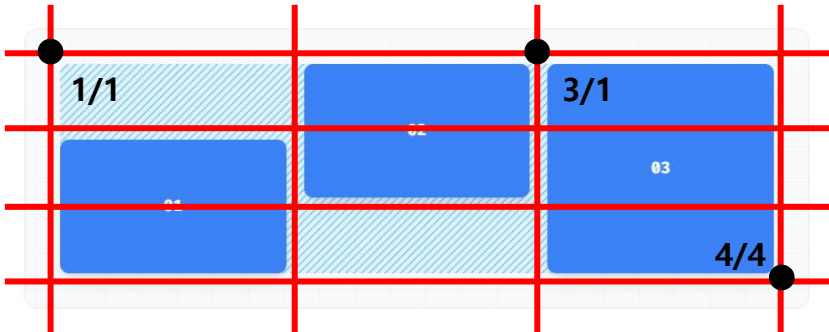
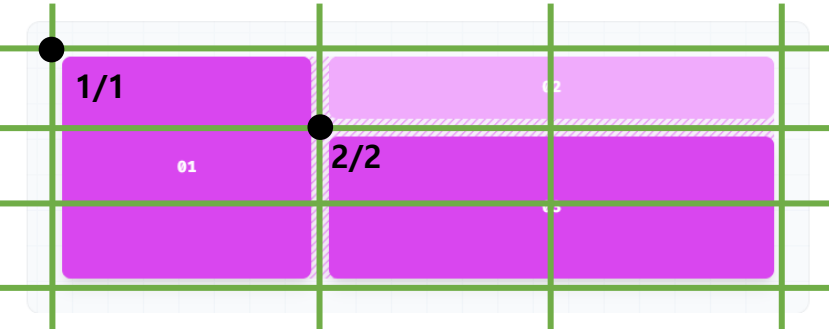
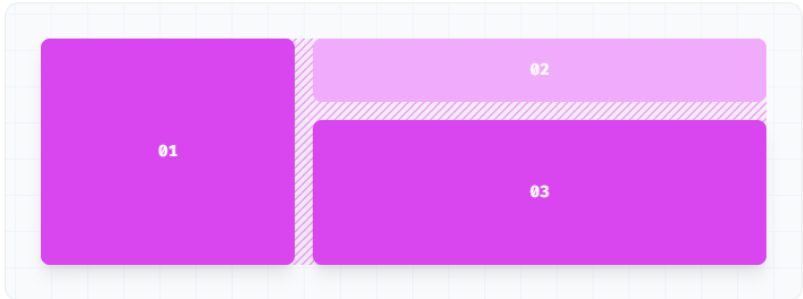
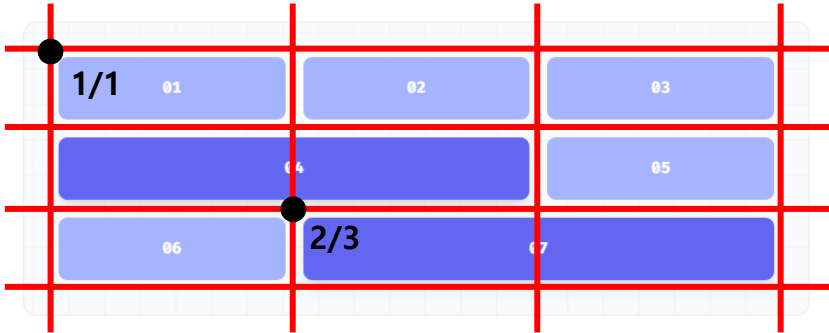
[Tailwind CSS 핵심 이론] [Grid System] - Grid 시스템 이해하기



1/1 → 컬럼시작 1, 로우 시작 1

2/3 → 컬럼시작 2, 로우 시작 3

(가로 세로 교차하는 일종의 좌표)



[Tailwind CSS 핵심 이론] 그리드 레이아웃 - Grid

부모요소에 쓰는 속성

- display: grid
- grid-template-column, grid-template-row
- grid-gap, grid-column-gap, grid-row-gap
- justify-items
- align-items
- justify-content
- align-content

자식요소에 쓰는 속성

- justify-self, align-self
- grid-column-start / grid-column-end / grid-column
- grid-row-start / grid-row-end, grid-row, grid-column, grid-row
- z-index
- grid-row
- order (Flex와 동일함)

```
<div class="grid">
  <div class="bg-red-500 h-[100px] rounded-md">01</div>
  <div class="bg-red-500 h-[100px] rounded-md">02</div>
  <div class="bg-red-500 h-[100px] rounded-md">03</div>
  <div class="bg-red-500 h-[100px] rounded-md">04</div>
</div>
```

▲ Grid는 Flex와 달리 grid 클래스를 준다고 바로 가로 배치가 되지 않습니다. grid 클래스 네임 준 후 적절한 컬럼의 개수를 주어야 합니다.

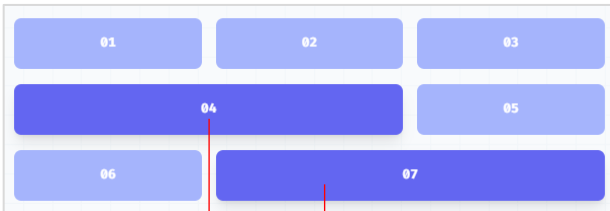
부모요소에 Grid 클래스 주고 컬럼 개수 설정해서 그리드 배치 시작

- grid-cols-1 grid-template-columns: repeat(1, minmax(0, 1fr));
- ... → 이것만 알면 됩니다!
- grid-cols-12 grid-template-columns: repeat(12, minmax(0, 1fr));
- grid-cols-[단위_minmax(단위,_1fr)_단위] ex) grid-cols-[200px_minmax(900px,_1fr)_100px]

[Tailwind CSS 핵심 이론] 그리드 레이아웃 - Grid

Grid Column Start / End - 컬럼 시작/종료

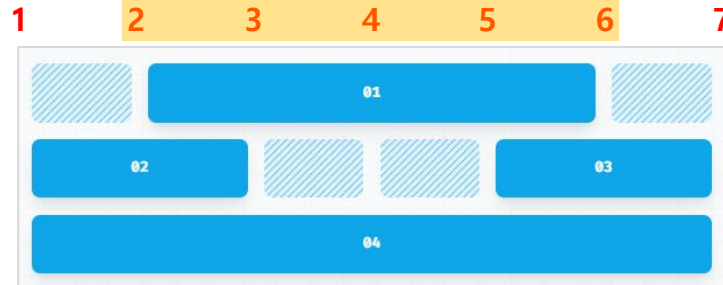
- col-span-1 ~ col-span-12
- col-span-full
- col-start-1 ~ col-start-13
- col-end-1 ~ col-end-13



`<div class="col-span-2">04</div>`

```
<div class="grid grid-rows-3 grid-flow-col gap-4">
  <div class="row-span-3 ..." >01</div>
  <div class="col-span-2 ..." >02</div>
  <div class="row-span-2 col-span-2 ..." >03</div>
</div>
```

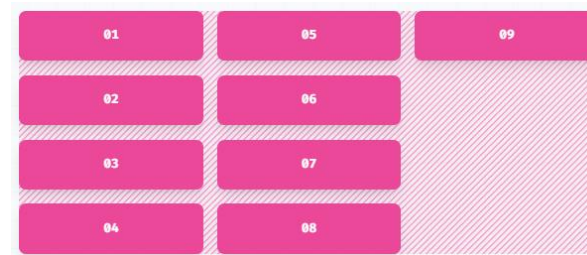
컬럼시작 지점이 2에서 시작하고 4개까지 확장한다.



```
<div class="grid grid-cols-6 gap-4">
  <div class="col-start-2 col-span-4 ..." >01</div>
  <div class="col-start-1 col-end-3 ..." >02</div>
  <div class="col-end-7 col-span-2 ..." >03</div>
  <div class="col-start-1 col-end-7 ..." >04</div>
</div>
```

Grid Template Rows - 그리드 열(가로) 개수

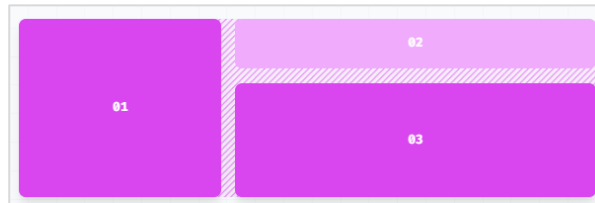
- grid-rows-1 ~ grid-rows-12



```
<div class="grid grid-rows-4 grid-flow-col gap-4">
  <div>01</div>
  <!-- ... -->
  <div>09</div>
</div>
```

Grid Row Start / End - 열(가로) 시작/종료

- row-span-1 ~ row-span-12
- row-span-full
- row-start-1 ~ row-start-13
- row-end-1 ~ row-end-13



[Tailwind CSS 핵심 이론] 정렬하기 - Flex & Grid

Justify Content – 한 줄일 경우 수평 정렬

- justify-start
- justify-end
- justify-center
- justify-between
- justify-around

justify-content: flex-start;

justify-content: flex-end;

justify-content: center;

justify-content: space-between;

justify-content: space-around;

01 02 03

01 02 03

Align Items – 한 줄일 경우 수직 정렬

- items-start
- items-end
- items-center

align-items: flex-start;

align-items: flex-end;

align-items: center;

01

01

01

Align Content- 여러 줄일 경우 수직 정렬

- content-center
- content-start
- content-end
- content-between
- content-around

align-content: center;

align-content: flex-start;

align-content: flex-end;

align-content: space-between;

align-content: space-around;

01 02 03 04 05

01 02 03 04 05

01 02 04 05

Align Self – 개별 수직 정렬

- self-start
- self-end
- self-center

align-self: flex-start;

align-self: flex-end;

align-self: center;

※ Align Self는 자식요소에 사용하는 속성

Flex로 부모요소 내 자식요소 센터링(Centering)

자식요소

Rapidly build modern websites without ever leaving your HTML. "Best practices" don't actually work.

요소들이 한 줄일 경우 수평/수직 정렬

요소 한 줄일 경우가 실무에서 자주 있으니 열심히 학습하세요!!

요소들이 여러 줄일 경우 수직 정렬

실무에서 매우 자주 사용하는 경우

Grid를 쓰면 Place Items 속성을 사용해서 한번에 자식요소를 센터링할 수 있습니다.

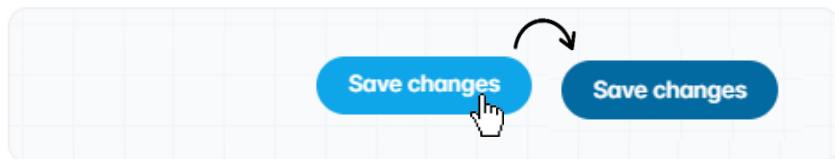
class="grid place-items-center"

tailwindcss

[Tailwind CSS 핵심 이론] 가상클래스 - Hover, Focus, Active, Content

Hover - 마우스가 올라가면 나타날 변화

- 마우스가 올라가 요소에 클래스 이름으로 hover:변화할 CSS 상태를 표시하면 됩니다.
- 최초 배경색은 sky-500에서 마우스 올리면 sky-900으로 변화
ex) bg-sky-500 hover:bg-sky-900

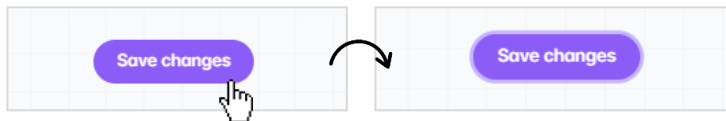


```
<button class="bg-sky-500 hover:bg-sky-700 ..." >  
  Save changes  
</button>
```

CSS에서 퍼블리셔가 사용할 수 있는 가상클래스의 종류는 정말 많습니다. 하지만 개발자 입장에서 Tailwind CSS를 하면서 그 수많은 가상클래스를 사용하는 건 현실적으로 무리입니다. 역시나 Tailwind CSS 공식사이트에서도 Hover, Focus, Active, Content 정도만 다루고 크게 비중 있게 다루지 않으니 가상클래스에 대한 부분 학습은 이 정도 선이면 일단 충분하다고 생각합니다.

Focus - 마우스가 올라가면 나타날 변화

- 마우스가 버튼, 인풋 등 폼(form) 요소를 클릭하면 발생되며 클래스 이름으로 focus:변화할 CSS 상태를 표시하면 됩니다.
- ex) outline-none focus:outline-[2]



Active - 클릭하고 떼지 않은 상태의 변화

- 마우스가 요소를 클릭하고 마우스를 떼지 않으면 발생되며 클래스 이름으로 active:변화할 CSS 상태를 표시하면 됩니다.
- ex) bg-sky-500 active: bg-sky-900

Content - before after

- 요소에 before after로 도형 또는 텍스트를 추가할 수 있는 가상클래스
- after:content-['내용']
- before:content-['내용']
- ex) after:content-['♥']
ex) before:content-['new']

New CodingWorks Online Class UPDATE

- CodingWorks Online Class
- 위처럼 new와 update라는 텍스트를 a태그 내에 넣지 않고 content를 통해 넣고 스타일링 합니다.
- 단점은 일반 CSS로 만드는 것보다 훨씬 손이 정말 더럽게 많이 가서 가성비 많이 떨어집니다. 사용하라 강추하기는 좀...

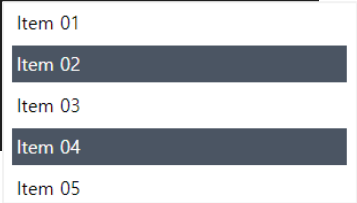
Pseudo-class reference를 당장은 보실 필요 없고 나중에 필요하면 보시면서 활용도가 있는 가상클래스를 Tailwind CSS 공식사이트에서 학습
<https://tailwindcss.com/docs/hover-focus-and-other-states#pseudo-class-reference>

[Tailwind CSS 핵심 이론] 자식선택자 & 자손선택자, 가상클래스(nth-child)

자식선택자(Child Selector) / 자식선택자 hover

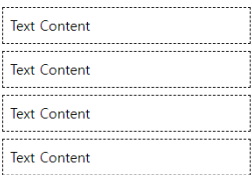
- 자식선택자란 부모요소 바로 밑의 요소
- 부모요소 클래스 네임에 &와 >를 사용 - 절대 띄어쓰기 불가 ex) [&>p]
- 자식선택자를 사용해서 Tailwind CSS의 대표적인 단점인 html이 지저분해 진다는 부분을 어느 정도는 상쇄할 수 있습니다.

```
<ul class="w-[300px] [&>li]:m-1 [&>li]:p-1 [&>li:nth-child(even)]:bg-gray-600 [&>li:nth-child(even)]:text-white">
  <li>Item 01</li>
  <li>Item 02</li>
  <li>Item 03</li>
  <li>Item 04</li>
  <li>Item 05</li>
</ul>
```



▲ 자식선택자 nth-child

▼ 자식선택자



```
<div class="w-[300px]">
  <p class="border border-black border-dashed p-2 m-2">Text Content</p>
  <p class="border border-black border-dashed p-2 m-2">Text Content</p>
  <p class="border border-black border-dashed p-2 m-2">Text Content</p>
  <p class="border border-black border-dashed p-2 m-2">Text Content</p>
</div>
```

▼ 자식선택자 hover



```
<div>
  <div class="border border-green-500 p-1 my-1">Children Element #01</div>
  <div class="border border-green-500 p-1 my-1">Children Element #02</div>
  <div class="border border-green-500 p-1 my-1">Children Element #03</div>
</div>
```

```
<div class="[&>div>div]:border [&>div>div]:border-green-500 [&>div>div]:p-1 [&>div>div]:my-1">
  <div>
    <div>Children Element #01</div>
    <div>Children Element #02</div>
    <div>Children Element #03</div>
  </div>
</div>
```

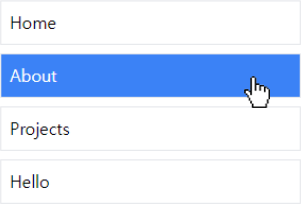
```
<div class="w-[300px] [&p]:border [&p]:border-black [&p]:border-dashed [&p]:p-2 [&p]:m-2">
  <p>Text Content</p>
  <p>Text Content</p>
  <p>Text Content</p>
  <p>Text Content</p>
</div>
```

[&]태그:테일윈드 클래스

자식요소가 다른 종류의 태그라면 [&]*를 사용할 수 있습니다.

```
<ul class="w-[300px] [&li]:border [&li]:p-2 [&li]:m-2 [&li]:capitalize [&li:hover]:text-white [&li:hover]:bg-blue-500">
  <li>Home</li>
  <li>About</li>
  <li>Projects</li>
  <li>Hello</li>
</ul>
```

[&]태그:hover:테일윈드 클래스



[Tailwind CSS 핵심 이론] 마우스 이벤트에 따른 부모 선택자, 형제 선택자 ✕

실전 예제 만들 때 형제 선택자, 부모 선택자는 매우 중요합니다. 확실히 이해하시고 활용할 수 있어야 합니다.

부모 선택자(group)

- ① 부모요소에 **group** 이라는 클래스 네임 주기
- ② 변화를 줄 자식요소에 **group-가상클래스:CSS상태** 만들기
가상클래스 종류와 CSS 상태는 제작자가 다양하게 변화 줄 수 있음
ex) `group-hover:block`
ex) `group-focus:hidden`

```
<a class="group" href="#none">
  <div>
    <h3 class="text-black group-hover:text-white">Lorem ipsum dolor sit amet.</h3>
    <p class="hidden group-hover:block">
      Lorem ipsum dolor sit amet consectetur, adipisicing elit. Quis sit quam eius
      dignissimos officiis laboriosam.
    </p>
  </div>
</a>
```

위의 예시처럼 하게 되면 부모요소인 a에 마우스가 올라가면 a의 자식요소인 h3의 글자색상이 변경되고, a의 또 다른 자식요소인 p태그의 내용이 안보인다고 보이는 상태로 바뀐다.

형제 선택자(peer)

- ① 형제요소에 **peer** 이라는 클래스 네임 주기
- ② 변화를 줄 형제요소에 **peer-가상클래스:CSS상태** 만들기
가상클래스 종류와 CSS 상태는 제작자가 다양하게 변화 줄 수 있음
ex) `peer-hover:visible`

```
<div>
  <h1 class="peer">Lorem ipsum dolor sit amet.</h1>
  <p class="hidden peer-hover:block">
    Lorem ipsum dolor sit amet consectetur adipisicing elit.
    Magni aspernatur sapiente autem doloremque. Quam,
    exercitationem.
  </p>
</div>
```

▼Tailwind CSS - Styling based on parent state (**group**-{modifier})

<https://tailwindcss.com/docs/hover-focus-and-other-states#styling-based-on-parent-state>

▼Tailwind CSS - Styling based on sibling state (**peer**-{modifier})

<https://tailwindcss.com/docs/hover-focus-and-other-states#styling-based-on-sibling-state>

Tailwind CSS 공식사이트에 사용방법이 영문으로 있습니다. 그런데 저도 이걸 이해하는데 꽤 걸렸습니다. 일반적인 CSS에서는 아무것도 아닌데 Tailwind CSS 시스템에서 구현하려고 하니깐 독특한 방법으로 해야 합니다. 개인적으로는 강의 자료 내용 정도로 일단 충분해 보입니다.

[Tailwind CSS 핵심 이론] 반응형 레이아웃 - Responsive Design

Container - 전체 레이아웃 반응형 레이아웃 자동 설정

- 웹 페이지 전체 레이아웃에서 가장 상단이 되는 div에 **container**라는 클래스 이름을 사용하면 전체 레이아웃 반응형 레이아웃 자동 설정해 줌.
- 브라우저 너비를 줄임으로서 해당 Breakpoint에 오면 자동으로 브라우저 사이즈를 Property에 맞춰 줍니다.

Layout		
Container		
Class	Breakpoint	Properties
container	None	width: 100%;
	sm (640px)	max-width: 640px;
	md (768px)	max-width: 768px;
	lg (1024px)	max-width: 1024px;
	xl (1280px)	max-width: 1280px;
	2xl (1536px)	max-width: 1536px;

Responsive Design - Breakpoint prefix로 반응형 조절

- sm, md, lg, xl, 2xl과 같은 Breakpoint prefix를 변화할 요소에 클래스로 넣어주어 브라우저 너비에 따라 반응형을 쉽게 제작할 수 있습니다.
- Targeting Mobile Screen(모바일 타겟 스크린)
 - 모바일 레이아웃은 Breakpoint prefix 접두사가 붙이지 않고 사용
 - 먼저 모바일 레이아웃 클래스를 지정한 후 더 넓은 레이아웃 Breakpoint prefix 접두사를 붙인 클래스를 지정

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet.

```
<!-- 반응형 Breakpoint prefix 기본 개념 -->
<div class="flex md:flex-row flex-col">
  <div class="bg-pink-500">Lorem ipsum do
  <div class="bg-pink-500">Lorem ipsum dolor sit amet.</div>
  <div class="bg-pink-500">Lorem ipsum dolor sit amet.</div>
</div>
```

Core Concepts

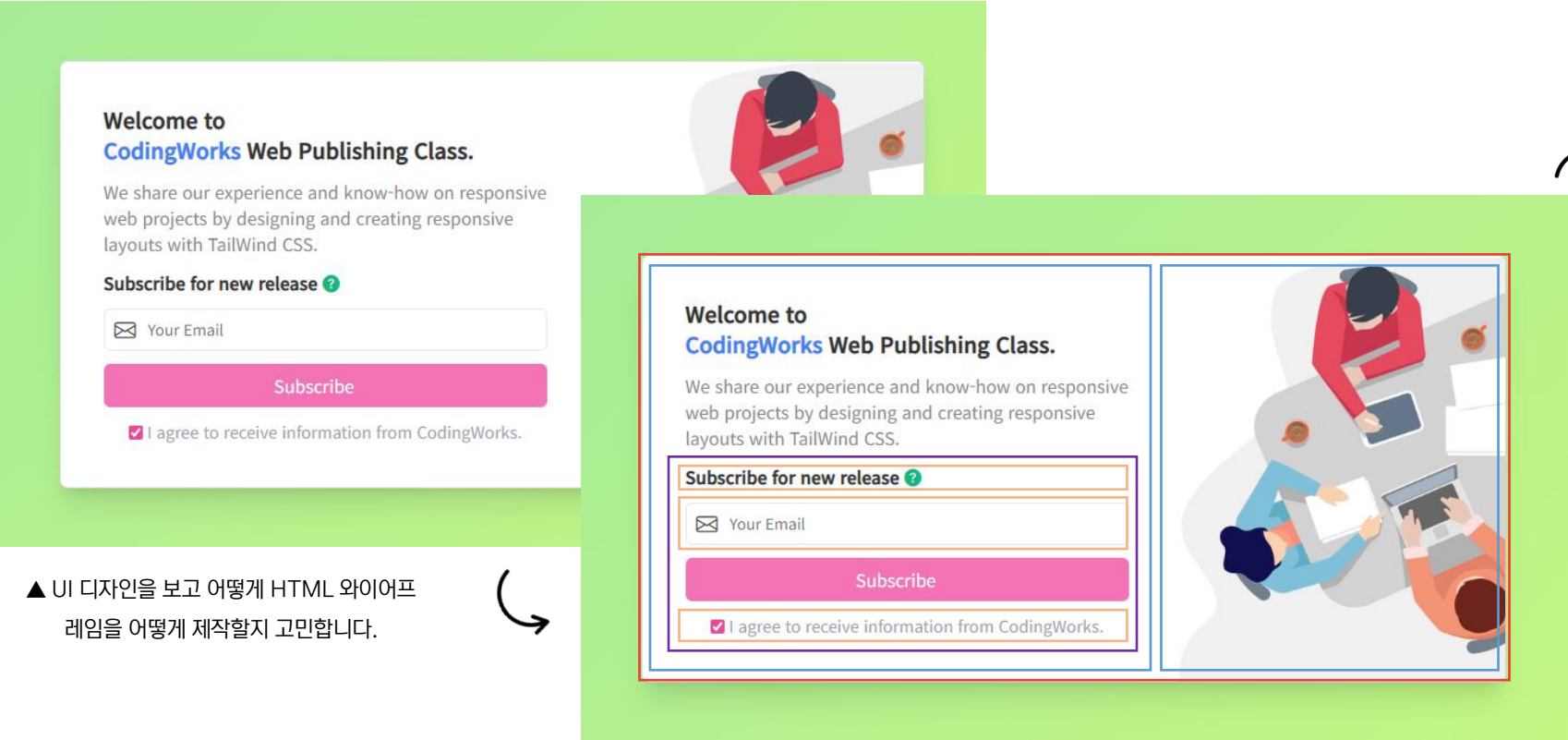
Responsive Design

Breakpoint prefix	Minimum width	CSS
`sm`	640px	`@media (min-width: 640px) { ... }`
`md`	768px	`@media (min-width: 768px) { ... }`
`lg`	1024px	`@media (min-width: 1024px) { ... }`
`xl`	1280px	`@media (min-width: 1280px) { ... }`
`2xl`	1536px	`@media (min-width: 1536px) { ... }`

- md:flex-row는 768px 이상일 때 flex-row
- Breakpoint prefix 접두사가 없는 것은 768px 미만일 때 flex-col 곧, 모바일 레이아웃

[Tailwind CSS 핵심 이론] HTML 와이어프레임 제작- Wireframe(1)

퍼블리싱 실력을 향상시키기 위해 필수적으로 HTML 와이어프레임 구조 제작을 하고 퍼블리싱을 해야 합니다.



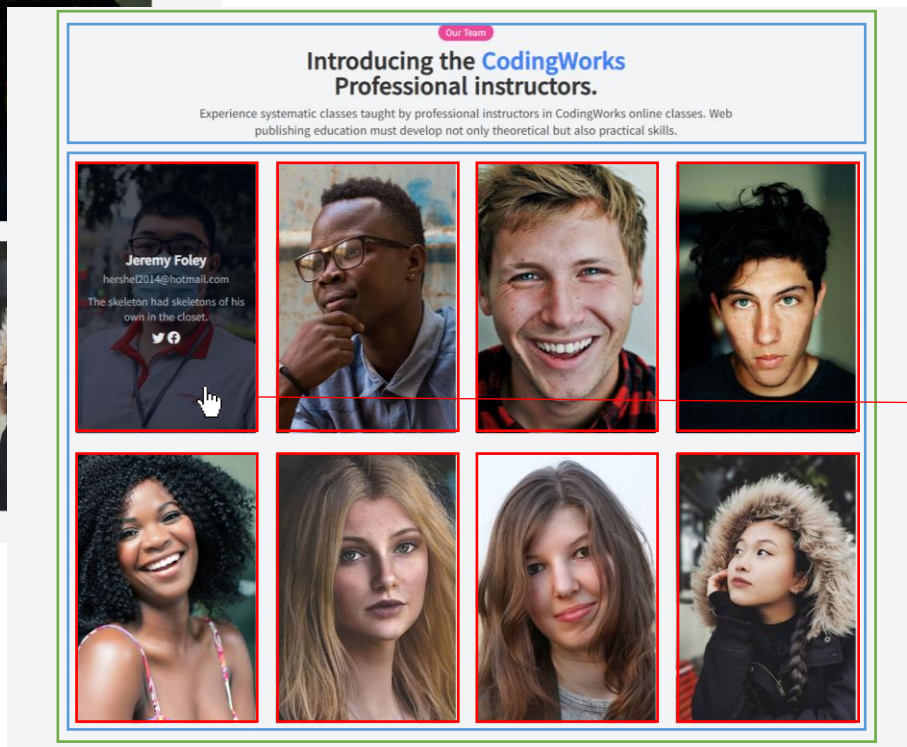
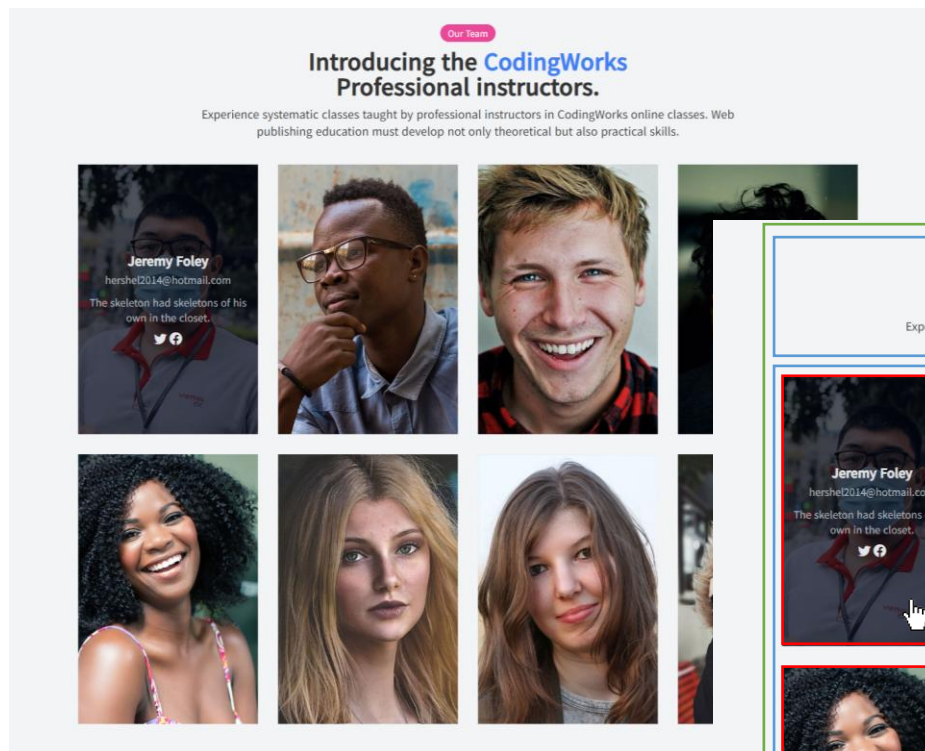
▲ UI 디자인을 보고 어떻게 HTML 와이어프레임을 어떻게 제작할지 고민합니다.

▲ UI 디자인을 보고 어떻게 요소들을 그룹(group)으로 묶을지 박스를 그립니다.
(박스를 치는 기준은 가로로 2개 이상의 요소가 있을 때)

```
div
  div
    h2
    p
    form
      div
        div
          b
          div
            span > font-icon
            span
        div
          input[type=text]
          span > font-icon
        button
          span > font-icon
        label
          input [type=checkbox]
      div
        img
```

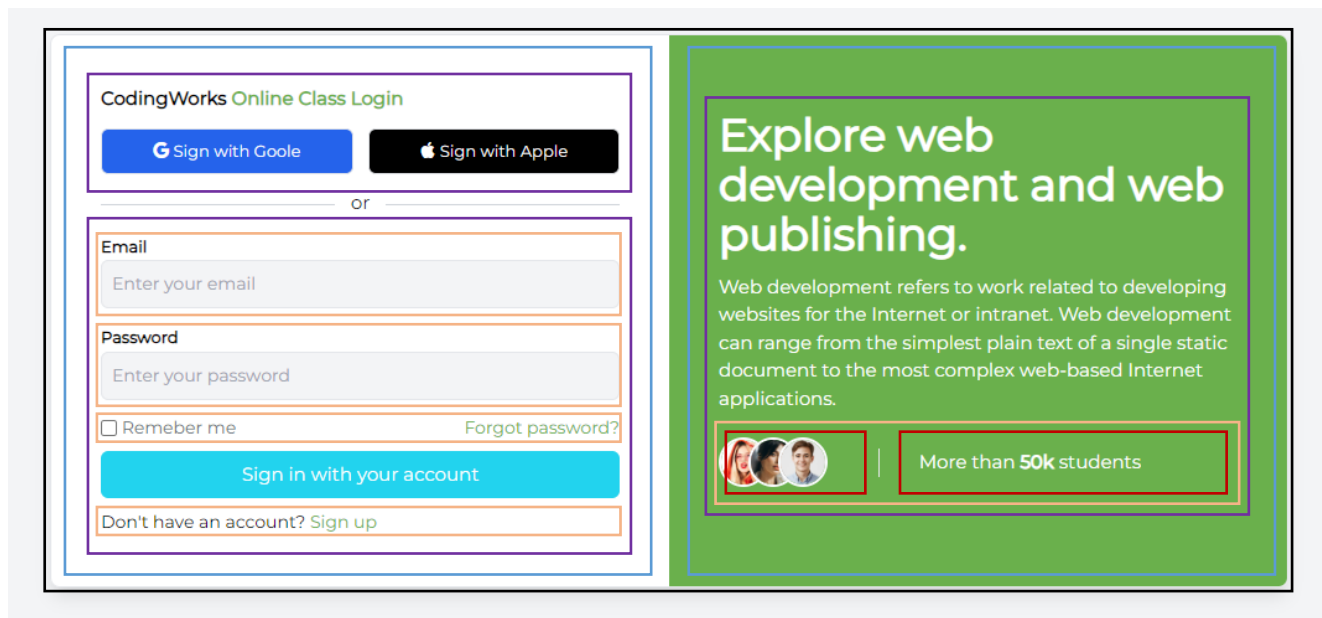
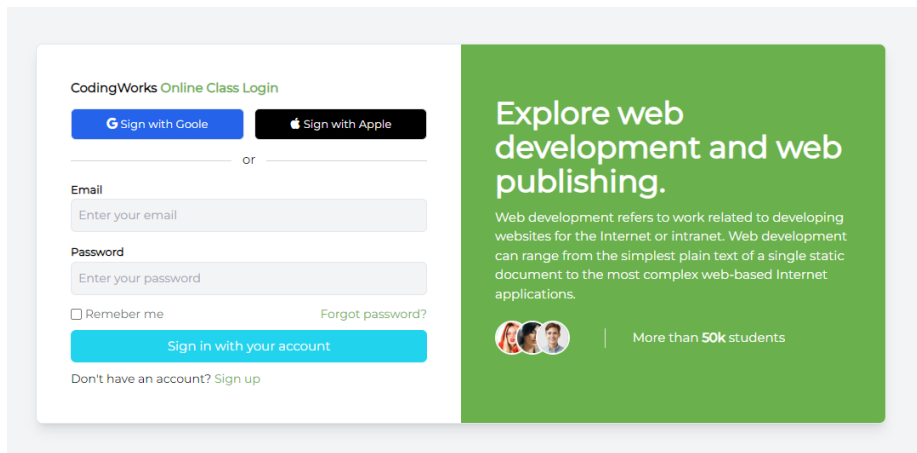
▲ 박스를 기준으로 어떤 태그를 사용할지 결정하면서 순수한 HTML 와이어프레임 구조만 작성합니다.

[Tailwind CSS 핵심 이론] HTML 와이어프레임 제작 - Wireframe(2)



```
section
  div
    span
    h1 > span
    p
  div
    div
      img
      div
        h3
        span
        p
        div
          (a > font-icon) * 2
```

[Tailwind CSS 핵심 이론] HTML 와이어프레임 제작- Wireframe(3)



```
section
  div
    div
      h3
      div
        button > font-icon
        button > font-icon
    span
    form
      div
        b, input[type=text]
      div
        b, input[type=password]
      div
        label > input[type=checkbox]
      button
      span > a
  div
    div
      h1
      p
      div
        div
          img*4
        div
          span > em
```

Tailwind CSS로 개발자가 만드는 멋진 UI 스타일링

Create a great web Ui Design styling with Tailwind CSS



[Tailwind CSS 핵심 이론] CDN에서 유틸리티 클래스/Config 활용하기

[Tailwind CSS 핵심 이론] CDN에서 유틸리티 클래스/Config 활용하기

SCSS에서

@mixin과 @include 사용하기

```
@mixin notice-button {
  font-size: 15px;
  width: 120px;
  padding: 7px;
  background-color: #fff;
  cursor: pointer;
}

.complete {
  @include notice-button;
  border: 1px solid royalblue;
  color: royalblue;
}

.loading {
  @include notice-button;
  border: 1px solid green;
  color: green;
}

.error {
  @include notice-button;
  border: 1px solid red;
  color: red;
}
```

SCSS

```
tailwind.config = {
  theme: {
    extend: {
      colors: {
        bgmain: '#e74c3c',
        textmain: '#f1c40f'
      }
    }
  }
}
```

▲ config에 색상 추가하기

HTML에서 클래스네임을 아래처럼 사용합니다.

- bg-bgmain, text-bgmain, border-bgmain

Lorem ipsum dolor sit amet consectetur adipisicing elit. Eius culpa, unde est ratione eos quia blanditiis temporibus nemo laudantium quo, vel quod, quaerat iusto. Amet voluptatem soluta in nam alias?

```
<div class="custom-box !text-base text-bgmain !border-4 !border-bgmain">
```

Tailwind CSS works by scanning all of your HTML files, JavaScript components, and any other templates for class names, generating the corresponding styles and then writing them to a static CSS file.

```
<div class="w-1/2 p-3 my-5 custom-box">
  Tailwind CSS works by scanning all of your HTML files, JavaScript
  components, and any other templates for class names, generating
  the corresponding styles and then writing them to a static CSS
  file.
</div>

<style type="text/tailwindcss">
  @layer utilities {
    .custom-box {
      @apply border-2 border-gray-200 shadow-lg rounded-md p-5
        text-center text-xl font-montserrat;
    }
  }
</style>
```

```
<div class="custom-box !text-base">
  Lorem ipsum dolor sit amet consectetur adipisicing elit.
</div>
```

- CSS에서 **important**
- .custom-box 안에 있는 text-xl을 text-base로 바뀜

Tailwind CSS 유틸리티 클래스 사용

- ① 자주 사용하는 CSS 속성을 하나의 클래스 골, 유틸리티 클래스를 만들어서 필요할 때 유틸리티 클래스 네임을 적용
- ② 기존의 클래스 중 수정이 필요할 속성이 있는 경우 CSS에서 !important 처럼 **!클래스네임** 을 사용하면 됩니다.

```
@layer utilities {
  클래스네임 {
    @apply 테일윈드 클래스 네임들
  }
}
```

▲ 유틸리티 클래스 생성하는 문법



[Tailwind CSS 핵심 이론] Tailwind CSS v4.0 소개

[Tailwind CSS v4.0 달라진 점]

- **새로운 고성능 엔진** : 전체 빌드는 최대 5배 더 빠르고, 증분 빌드는 100배 이상 더 빠르며 - 마이크로초 단위로 측정됩니다
- **최신 웹을 위해 설계됨** : 캐스케이드 레이어, @property로 등록된 사용자 지정 속성 및 color-mix()와 같은 최첨단 CSS 기능을 기반으로 구축되었습니다.
- **단순화된 설치** : 더 적은 종속성, 제로 구성, CSS 파일에 단 한 줄의 코드.
- **자사 Vite 플러그인** : 최대 성능과 최소 구성을 위한 긴밀한 통합.
- **자동 콘텐츠 감지** : 모든 템플릿 파일이 자동으로 발견되며, 구성이 필요하지 않습니다.
- **내장된 import 지원** : 여러 CSS 파일을 묶기 위해 추가 도구가 필요하지 않습니다.
- **CSS-first 구성** : JavaScript 구성 파일 대신 CSS에서 직접 프레임워크를 사용자 정의하고 확장하는 재창조된 개발자 경험.
- **CSS 테마 변수** : 모든 디자인 토큰이 네이티브 CSS 변수로 노출되므로 어디서나 액세스할 수 있습니다.
- **동적 유틸리티 값 및 변형** : 간격 척도에 어떤 값이 존재하는지 추측하거나 기본 데이터 속성과 같은 것에 대한 구성을 확장하지 마십시오.
- **현대화된 P3 컬러 팔레트** : 현대적인 디스플레이 기술을 최대한 활용하는 재설계된, 더욱 생생한 컬러 팔레트.
- **컨테이너 쿼리** : 컨테이너 크기에 따라 요소를 스타일링하기 위한 일류 API, 플러그인이 필요하지 않습니다.
- **새로운 3D 변환 유틸리티** : HTML에서 직접 3D 공간의 요소를 변환합니다.
- **확장된 그라디언트 API** : 방사형 및 원뿔형 그라디언트, 보간 모드 등.
- **@starting-style 지원** : 자바스크립트 없이 입력 및 종료 전환을 만드는 데 사용할 수 있는 새로운 변형.
- **Not-* 변형** : 다른 변형, 사용자 지정 선택자 또는 미디어 또는 기능 쿼리와 일치하지 않는 경우에만 요소를 스타일링합니다. (nth-* 변형 사용 가능)
- **더 많은 새로운 유틸리티와 변형** : 색조, 필드 크기 조정, 복잡한 그림자, 불활성 등.

Tailwind CSS v3.4와 v4.0 빌드 속도 비교

	v3.4	v4.0	Improvement
Full build	378ms	100ms	3.78x
Incremental rebuild with new CSS	44ms	5ms	8.8x
Incremental rebuild with no new CSS	35ms	192µs	182x



CSS 테마 변수(Adding custom styles) v3.4와 v4.0 비교

```
tailwind.config.js
/** @type {import('tailwindcss').Config} */
module.exports = {
  theme: {
    screens: {
      sm: '480px',
      md: '768px',
      lg: '976px',
      xl: '1440px',
    },
    colors: {
      'blue': '#1fb6ff',
      'pink': '#ff49db',
      'orange': '#ff7849',
      'green': '#13ce66',
      'gray-dark': '#273444',
      'gray': '#8492a6',
      'gray-light': '#d3dce6',
    },
  },
  fontFamily: {
    sans: ['Graphik', 'sans-serif'],
    serif: ['Merriweather', 'serif'],
  },
}
```

▲ v3.4에서 tailwind.config.js에 추가해서 사용

```
CSS
@theme {
  --font-display: "Satoshi", "sans-serif";

  --breakpoint-3xl: 1920px;

  --color-avocado-100: oklch(0.99 0 0);
  --color-avocado-200: oklch(0.98 0.04 113.22);
  --color-avocado-300: oklch(0.94 0.11 115.03);
  --color-avocado-400: oklch(0.92 0.19 114.08);
  --color-avocado-500: oklch(0.84 0.18 117.33);
  --color-avocado-600: oklch(0.53 0.12 118.34);

  --ease-fluid: cubic-bezier(0.3, 0, 0, 1);
  --ease-snappy: cubic-bezier(0.2, 0, 0, 1);

  /* ... */
}
```

▲ v4.0에서 input.css에 추가해서 사용

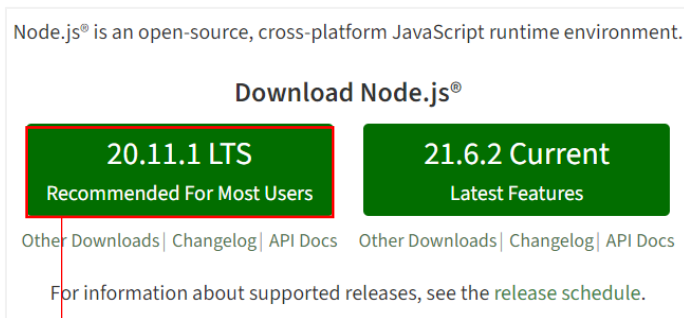


[Tailwind CSS 핵심 이론] Tailwind CSS v4.0에서 CLI 환경 구축(1)

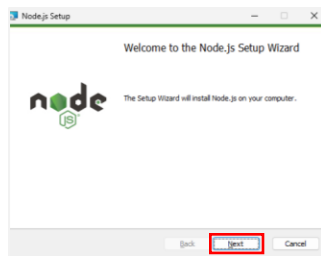
참고 웹사이트 URL

- Tailwind CSS v4.0 정식 출시 : 2025년 1월 22일
- Tailwind CSS v4.0 CLI Installation : <https://tailwindcss.com/docs/installation/tailwind-cli>
- Download Node.js : <https://nodejs.org/en>
- Tailwind CSS 4.0 소개(수코딩 님) : <https://www.youtube.com/watch?v=2Q7x3LNZPJs&t=3s>

Step #1) Node.js 설치하기



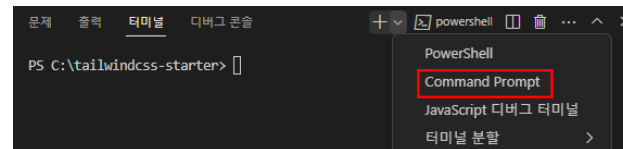
클릭해서 설치를 시작하고
계속 Next를 눌러서 설치를
마칩니다.



Step #2) Tailwind v4.0 CLI 환경설정 - 설치 경로 설정 및 폴더 생성

- C드라이브 루트에 tailwindcss-cli-setting 라는 폴더 만들기
(폴더는 아무 위치나... 폴더명도 아무거나.. 단, 띄어쓰기 없이 영문으로 해주세요.)
- tailwindcss-cli-setting를 루트 폴더로 VC에서 터미널 열기
만약 PowerShell 인 경우 Command Prompt 로 변경
- 커멘트 창에 `npm init` 입력 후 엔터(패키지 파일 생성되며 계속 나오는 항목은 그냥 엔터 엔터)
package.json 파일 생성됨

▼ 터미널 열기 단축키 Ctrl + Shift + `

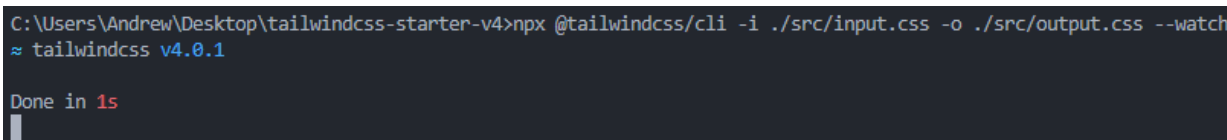


PS C:\tailwindcss-cli-setting> npm init



▲ package.json

- 커멘트 창에 `npm install tailwindcss @tailwindcss/cli` 입력 후 엔터(테일윈드 버전 4.0 확인) →
- 루트에 src 폴더 만들고 그 안에 input.css 파일 생성 후 input.css 파일에 `@import "tailwindcss";` 코드 붙여넣기
순서상 5)번 6)번은 순서가 변경되어도 상관 없음
- 커멘트 창에 `npm @tailwindcss/cli -i ./src/input.css -o ./src/output.css --watch` 입력 후 엔터(빌드 시작)





[Tailwind CSS 핵심 이론] Tailwind CSS v4.0에서 CLI 환경 구축(2)

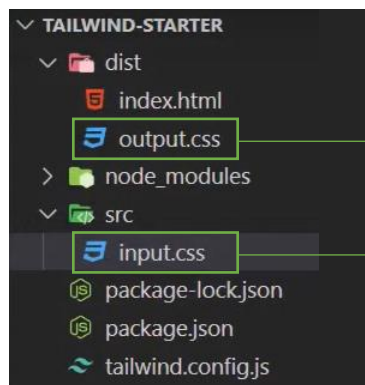
Step #3) Tailwind CLI 환경설정 - 소스 폴더 생성 및 경로 변경

① CLI 빌드 시작하기(아래 코드 커맨드 창에 붙여넣기 하고 엔터)

※ 빌드 종료 단축키 Ctrl + C

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

② CLI 빌드 시작하면 dist 폴더에 output.css 파일 자동 생성됨



복사하고 src를 dist로 반드시
변경 후 커멘트 창에 입력

Tailwind CSS 모든 클래스와 속성이
들어 있음

input.css 에서 작성된 유틸리티 클래스
와 일반적인 클래스와 속성이
output.css에 자동으로 쓰여 짐.

▲ CLI 설치 후 최종적인 폴더 구조

```
> tailwindcss-starter-v4@1.0.0 dev
> npx @tailwindcss/cli -i ./src/input.css -o ./src/output.css --watch

~ tailwindcss v4.0.1

Done in 15s
```

▲ npm run dev 후 위 화면처럼 나와야 빌드가 정상적으로 작동합니다.

⑤ index.html에 기본 구조를 만들고 output.css를 링크하고
Tailwind CSS 적용 잘 되는지 확인하기

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="output.css">
</head>
<body>
</body>
</html>
```

⑥ CLI 빌드 시작하기 코드 쉽게 줄이기

package.json 파일에서 빌드 시작 코드를 쉽게 만듭니다.

```
{
  "name": "tailwindcss-starter-v4",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "dev": "npx @tailwindcss/cli -i ./src/input.css -o ./src/output.css --watch"
  },
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "@tailwindcss/cli": "^4.0.1",
    "tailwindcss": "^4.0.1"
  }
}
```

⑦ 커멘드 창에서 npm run dev 입력 후 엔터치면 CLI 빌드 시작됨

⑧ 파일명 변경하기(output.css ==> style.css)

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

- output.css를 style.css로 변경하고 빌드 종료하고 다시 빌드 시작하면 style.css 파일 생성됨.(output.css은 불필요하므로 삭제)
- 만약 style.css를 css 폴더에 넣어서 관리하고 싶으면 ./dist/css/style.css 라고 경로를 적어주면 됨.

dev가 아닌 다른 이름으로 바뀌도 됩니다.

v3.4에서는 이미 npm이 실행되고 있기 때문에 npx를 빼도 되지만, v4.0에서 npx를 빼면 빌드가 시작되지 않습니다.

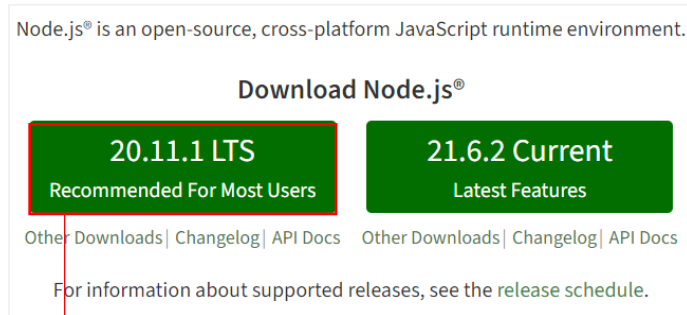
혹시라도 설치과정에서 문제가 생겨서 잘 안되는 경우는 루트 폴더 전체를 삭제하고 다시 차분하게 설치과정을 다시 해보세요. node.js는 다시 설치할 필요 없습니다.

[Tailwind CSS 핵심 이론] Tailwind CSS v3.4에서 CLI 환경 구축(1)

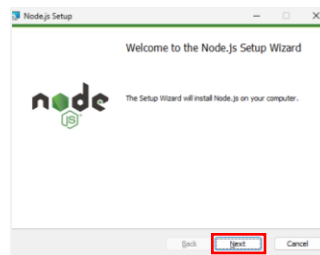
참고 웹사이트 URL

- Tailwind CLI Installation :
<https://v3.tailwindcss.com/docs/installation>
- Download Node.js : <https://nodejs.org/en>
- Node.js란?(코딩애플 님) :
<https://www.youtube.com/watch?v=pTm5E3jcOeY>
- Command Line Interface (CLI) 란?
<https://www.youtube.com/watch?v=IHvQcHEmLdc>

Step #1) Node.js 설치하기

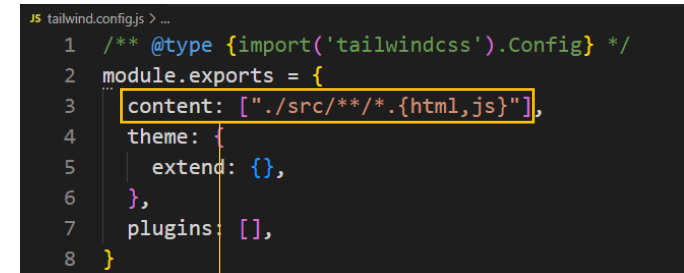
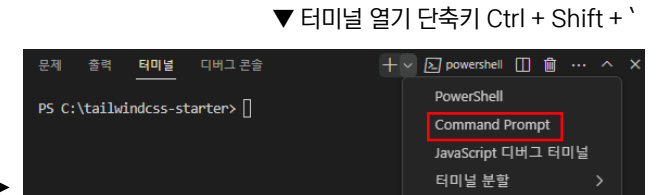


클릭해서 설치를 시작하고
계속 Next를 눌러서 설치를
마칩니다.



Step #2) Tailwind CLI 환경설정 - 설치 경로 설정 및 폴더 생성

- ① C드라이브 루트에 tailwindcss-cli-setting 라는 폴더 만들기
(폴더는 아무 위치나... 폴더명도 아무거나.. 단, 띄어쓰기 없이 영문으로 해주세요.)
 - ② tailwindcss-cli-setting를 루트 폴더로 VC에서 터미널 열기
만약 PowerShell 인 경우 Command Prompt 로 변경
 - ③ 커멘트 창에 `npm init` 입력 후 엔터(패키지 파일 생성되며 계속 나오는 항목은 그냥 엔터 엔터)
- package.json 파일 생성됨



- ④ 커멘트 창에 `npm install -D tailwindcss` 입력 후 엔터(node module 생성)
- ⑤ 커멘트 창에 `npx tailwindcss init` 입력 후 엔터(tailwindcss.config.js 생성)
- ⑥ 템플릿 경로 설정
`\"./src/**/*.html,js\"` 를 tailwindcss.config.js 파일의 content에 붙여넣기
- ⑦ 루트에 `dist` 폴더 만들고 그 안에 `index.html` 만들기
- ⑧ tailwindcss.config.js 파일 내 content 경로 변경
`content: [\"./src/**/*.html,js\"], ==> content: [\"./dist/**/*.html,js\"],`

커맨드창에서 npm init 이라고 입력하고 엔터 쳤을 때 아래처럼 오류를 출력하면 npm 환경 변수를 설정해주어야 합니다.
'npm'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는 배치 파일이 아닙니다.

- ① 윈도우 검색에서 '환경 변수' 라고 검색
- ② 환경 변수 > Path 클릭하고 편집 클릭 > 새로 만들기 클릭
- ③ C:\Program Files\nodejs\ 입력 후 확인

[Tailwind CSS 핵심 이론] Tailwind CSS v3.4에서 CLI 환경 구축(2)

Step #3) Tailwind CLI 환경설정 - 소스 폴더 생성 및 경로 변경

① 루트에 src 폴더 만들고 그 안에 input.css 만들기

② input.css에 아래 코드 붙여넣기

```
src/input.css

@tailwind base;
@tailwind components;
@tailwind utilities;
```

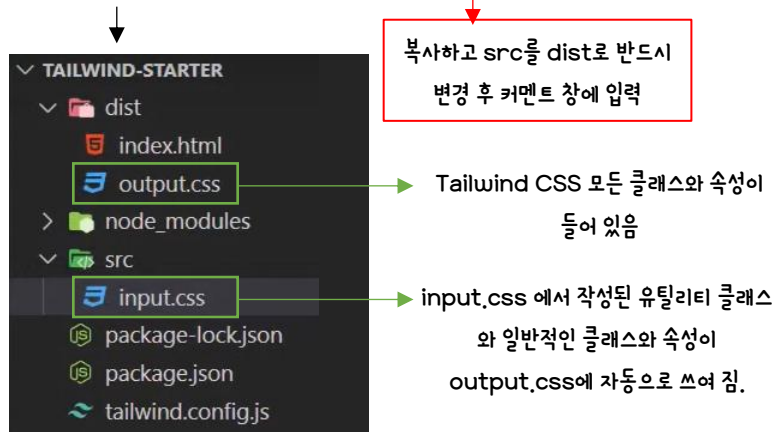
input.css에 유틸리티 클래스 넣으면 컴파일 되어 output.css에 일반적인 CSS로 변환되어 저장됨.

③ CLI 빌드 시작하기(아래 코드 커맨드 창에 붙여넣기 하고 엔터)

※ 빌드 종료 단축키 Ctrl + C

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

④ CLI 빌드 시작하면 dist 폴더에 output.css 파일 자동 생성됨



▲ CLI 설치 후 최종적인 폴더 구조

⑤ index.html에 기본 구조를 만들고 output.css를 링크하고 Tailwind CSS 적용 잘 되는지 확인하기

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="output.css">
</head>
<body>
  ~
</body>
</html>
```

⑥ CLI 빌드 시작하기 코드 쉽게 줄이기

package.json 파일에서 빌드 시작 코드를 쉽게 만듭니다.

```
package.json
package.json > ...
1 {
2   "name": "tailwindcss-cli-setting",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8   },
9   "author": "",
10  "license": "ISC"
11 }
```

⑦ 커맨드에서 `npm run dev` 입력 후 엔터치면 CLI 빌드 시작됨

⑧ 파일명 변경하기(output.css ==> style.css)

```
"dev": "tailwindcss -i ./src/input.css -o ./dist/output.css --watch"
```

- output.css를 style.css로 변경하고 빌드 종료하고 다시 빌드 시작하면 style.css 파일 생성됨.(output.css은 불필요하므로 삭제)
- 만약 style.css를 CSS 폴더에 넣어서 관리하고 싶으면 `./dist/css/style.css` 라고 경로를 적어주면 됨.

혹시라도 설치과정에서 문제가 생겨서 잘 안되는 경우는 루트 폴더 전체를 삭제하고 다시 차분하게 설치과정을 다시 해보세요.
node.js는 다시 설치할 필요 없습니다.

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "dev": "tailwindcss -i ./src/input.css -o ./dist/output.css --watch"
},
```

▲ test를 다른 이름으로 바뀌어도 되고.. 예시처럼 dev라는 이름으로 새로 만들어도 됩니다. 이미 npm이 실행되고 있기 때문에 npx를 빼도 됩니다. 넣어도 상관 없습니다.

[Tailwind CSS 핵심 이론] CLI 환경에서 유틸리티 클래스/웹 폰트 설정

Step #1) input.css에 웹 폰트 CDN 입력

```
@import url('https://fonts.googleapis.com/css2?family=Noto+Sans+KR:wght@400;500&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,100..900;1,100..900&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Nanum+Pen+Script&display=swap');

/* Noonu Web Fonts CDN */
@font-face {
  font-family: 'Pretendard-Regular';
  src: url('https://cdn.jsdelivr.net/gh/Project-Noonnu/noonfonts_2107@1.1/Pretendard-Regular.woff') format('woff');
  font-weight: 400;
  font-style: normal;
}

@font-face {
  font-family: 'YClover-Bold';
  src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_231029@1.1/YClover-Bold.woff2') format('woff2');
  font-weight: 700;
  font-style: normal;
}

@tailwind base;
@tailwind components;
@tailwind utilities;
```

▲ 웹 폰트 CDN 위치는 가장 상단에 위치시킵니다.

Step #2) tailwind.config.js에 font 코드 입력

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ["./dist/*.html", "js"],
  theme: {
    extend: {
      fontFamily: {
        sans: ['Noto Sans KR', "Arial", "sans-serif"],
        notosans: ['Noto Sans KR', 'sans-serif'],
        montserrat: ['Montserrat', 'sans-serif'],
        nanumpen: ["Nanum Pen Script", 'cursive'],
        pretendard: ['Pretendard-Regular'],
        yclover: ['YClover-Bold']
      }
    }
  },
  plugins: [],
}
```

Step #3) html 파일에 원하는 폰트 클래스 사용

```
<div class="my-5 font-montserrat">
  Tailwind CSS works by scanning all of your HTML files, JavaScript
  components, and any other templates for class names, generating the
  corresponding styles and then writing them to a static CSS file.
</div>

<div class="my-5 font-notosans">
  모든 사람은 노동시간의 합리적 제한과 정기적인 유급휴가를 포함하여 휴식과 여가의
  권리를 가진다.
</div>

<div class="my-5 font-nanumpen">
  모든 사람은 노동시간의 합리적 제한과 정기적인 유급휴가를 포함하여 휴식과 여가의
  권리를 가진다.
</div>

<div class="my-5 font-pretendard">
  모든 사람은 노동시간의 합리적 제한과 정기적인 유급휴가를 포함하여 휴식과 여가의
  권리를 가진다.
</div>

<div class="my-5 font-yclover">
  모든 사람은 노동시간의 합리적 제한과 정기적인 유급휴가를 포함하여 휴식과 여가의
  권리를 가진다.
</div>
```

이 선언의 어떠한 규정도 어떤 국가, 집단 또는 개인에게 이 선언에 규정된 어떠한 권리와 자유를 파괴하기 위한 활동에 가담하거나 또는 행위를 할 수 있는 권리가 있는 것으로 해석되어서는 아니된다.

Tailwind CSS works by scanning all of your HTML files, JavaScript components, and any other templates for class names, generating the corresponding styles and then writing them to a static CSS file.

모든 사람은 노동시간의 합리적 제한과 정기적인 유급휴가를 포함하여 휴식과 여가의 권리를 가진다.

모든 사람은 노동시간의 합리적 제한과 정기적인 유급휴가를 포함하여 휴식과 여가의 권리를 가진다.

모든 사람은 노동시간의 합리적 제한과 정기적인 유급휴가를 포함하여 휴식과 여가의 권리를 가진다.

모든 사람은 노동시간의 합리적 제한과 정기적인 유급휴가를 포함하여 휴식과 여가의 권리를 가진다.

[Tailwind CSS 핵심 이론] @layer로 Base, components, utilities 레이어 개념

Custom Styles(Custom Theme)

실전 예제 정도는 상관 없지만 프로젝트를 진행하다 보면 디자인 가이드에 맞게 전체적인 스타일을 정해야 합니다. 이런 경우 tailwind.config.js에 글꼴, 색상 등 theme(스타일)에 추가하여 초기 세팅을 합니다.

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ["./dist/*.html", ".js"],
  theme: {
    extend: {
      colors: {
        deepgreen: '#6ab04c', // 메인 색상
        deeporange: '#f0932b', // 포커스 색상
        darkblack: '#333333', // 기본 글자 색상
        dimblack: '#555555', // 필요시
        silverblack: '#777777', // 문단 글자 색상
        lightblack: '#999999', // 필요시
        linecolor: 'f8f8f8' // 그레이 구분선 색상
      },
      fontFamily: {
        sans: ['Noto Sans KR', 'Arial', 'sans-serif'], // 기본 글꼴
        notosans: ['Noto Sans KR', 'sans-serif'], // 메인 한글 글꼴
        raleway: ['Raleway', 'sans-serif'], // 메인 영어 글꼴
        montserrat: ['Montserrat', 'sans-serif'] // 서브 영어 글꼴
      }
    }
  },
  plugins: [],
}
```

base 레이어

Base 레이어는 Reset 규칙이나 Tailwind CSS에서 reset 시켜 놓은 HTML 요소 기본 스타일을 재정의

```
/* 기본(Base) 스타일 설정 */
@layer base {
  body {
    @apply font-notosans text-[16px] font-[400] text-darkblack
    leading-[1.6em];
  }
}
```

```
@tailwind base;
@tailwind components;
.title-at {
  @apply p-3;
}
.title-at > h4 {
  @apply text-[20px] font-bold mb-2;
}
.title-at > h4 > em {
  @apply not-italic text-base font-normal text-silverblack ml-2;
}
.title-at > p {
  @apply text-silverblack;
}
```

컴포넌트 코드를 @tailwind components; 바로 아래에 사용하면 @layer components를 사용하지 않아도 됩니다. 하지만 좀 더 일관성 있는 코드 관리를 위해서 추천드리지 않습니다.

components 레이어

컴포넌트 레이어는 카드, 버튼과 같이 재사용이 높은 스타일을 유틸리티 클래스를 만들 때 사용

```
/* 자주 사용하는 컴포넌트 클래스 설정 */
@layer components {
  .title-at > h4 {
    @apply text-[20px] font-bold mb-2;
  }
  .title-at > h4 > em {
    @apply not-italic text-base font-normal text-silverblack ml-2;
  }
}
```

utilities 레이어

Tailwind CSS가 제공하지 않은 유틸리티 클래스를 만들 때 사용

```
/* Tailwind CSS가 제공하지 않은 유틸리티 클래스 */
@layer utilities {
  .content-auto {
    content-visibility: auto;
  }
}
```