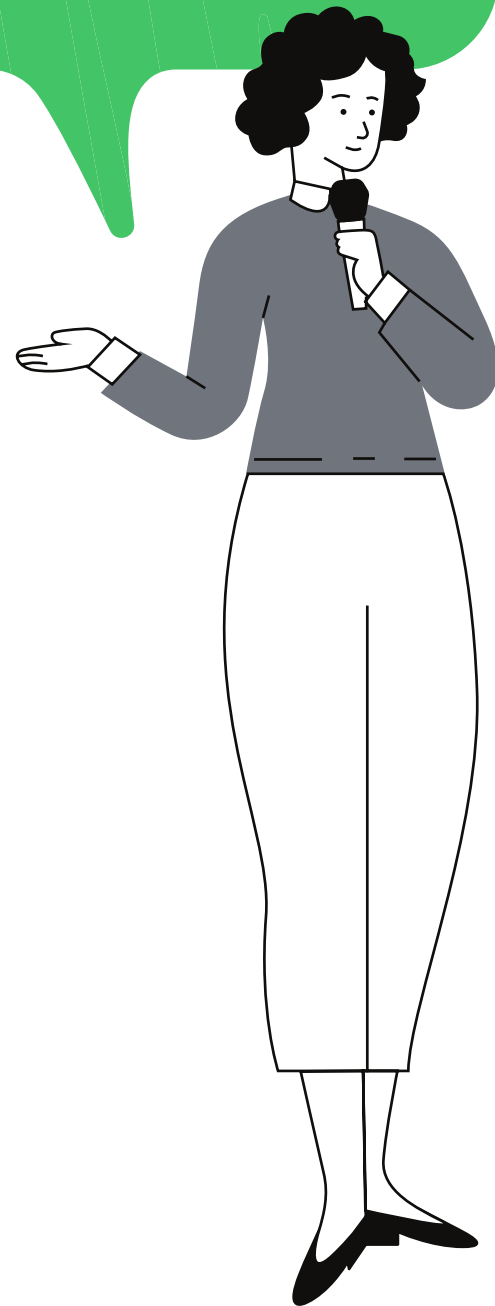


방학 2주차



Today's Study



1

Coding Test

2

SCSS

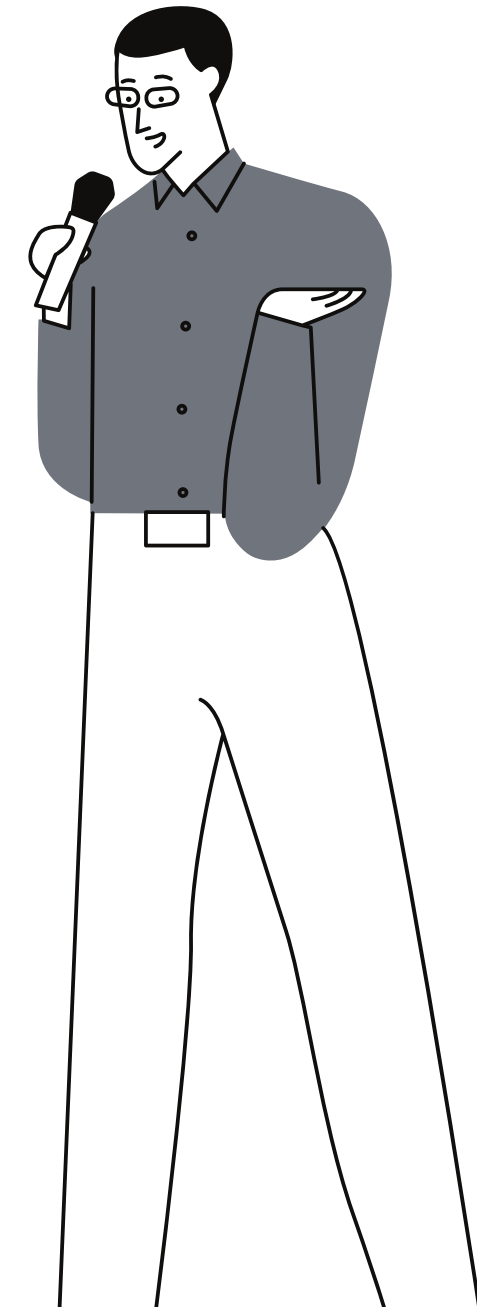
Coding Test

20230704

- 1 저번주 발표 이어서(발표 들으면서 평가지 작성)
- 2 다른 팀이 선택한 Lv.O 문제 풀기
- 3 풀 코드와 점수 캡처해서 김은서에게 카카오톡으로 전송 후 발표
- 4 발표는 각 팀에서 한 명만
- 5 발표 순서: 정지현조 → 나민경조 → 고선진조 → 김하경조
- 6 평가 점수 합계 후 1위 발표

Are you ready?

Let's Start!



각도기

문제 설명

각에서 0도 초과 90도 미만은 예각, 90도는 직각, 90도 초과 180도 미만은 둔각 180도는 평각으로 분류합니다. 각 `angle` 이 매개변수로 주어질 때 예각일 때 1, 직각일 때 2, 둔각일 때 3, 평각일 때 4를 return하도록 solution 함수를 완성해주세요.

- 예각 : $0 < \text{angle} < 90$
- 직각 : $\text{angle} = 90$
- 둔각 : $90 < \text{angle} < 180$
- 평각 : $\text{angle} = 180$

짝수의 합

문제 설명

정수 `n` 이 주어질 때, `n` 이하의 짝수를 모두 더한 값을 return 하도록 solution 함수를 작성해주세요.

1로 만들기

문제 설명

정수가 있을 때, 짝수라면 반으로 나누고, 홀수라면 1을 뺀 뒤 반으로 나누면, 마지막엔 1이 됩니다. 예를 들어 10이 있다면 다음과 같은 과정으로 1이 됩니다.

- $10 / 2 = 5$
- $(5 - 1) / 2 = 2$
- $2 / 2 = 1$

위와 같이 4번의 나누기 연산으로 1이 되었습니다.

정수들이 담긴 리스트 `num_list` 가 주어질 때, `num_list` 의 모든 원소를 1로 만들기 위해서 필요한 나누기 연산의 횟수를 return하도록 solution 함수를 완성해주세요.

문자열 붙여서 출력하기

문제 설명

두 개의 문자열 `str1`, `str2` 가 공백으로 구분되어 입력으로 주어집니다.

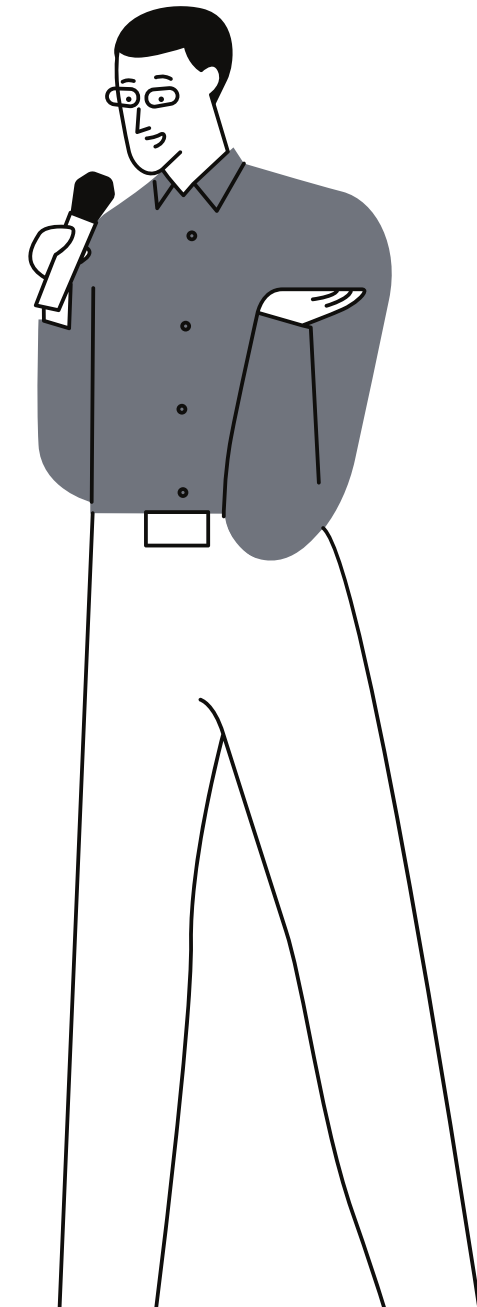
입출력 예와 같이 `str1` 과 `str2` 을 이어서 출력하는 코드를 작성해 보세요.

평가표		점수	
1	모든 문제를 풀이하였나요?(총 30점,문제 개당 10점씩)		
2	모든 문제를 발표하였나요?(총 30점,문제 개당 10점씩)		
3	1팀 퀴즈를 맞추었나요? (5점)		한 팀당 하나씩 !
	2팀 퀴즈를 맞추었나요? (5점)		
	3팀 퀴즈를 맞추었나요? (5점)		
	4팀 퀴즈를 맞추었나요? (5점)		
4	1팀 문제를 다른 방식으로 풀이 했나요? (10점)		
	2팀 문제를 다른 방식으로 풀이 했나요? (10점)		
	3팀 문제를 다른 방식으로 풀이 했나요? (10점)		
	4팀 문제를 다른 방식으로 풀이 했나요? (10점)		
5	하나의 문제를 여러 개의 풀이로 하였나요?(풀이 당 3점씩)		
총점		00점	합계

Break Time

Are you ready?

Let's Start!



1 SCSS(SASS)란 무엇인가

2 SCSS와 SASS 차이

3 SCSS 개념 & 실습

SCSS(SASS).

1

- CSS를 편리하게 이용할 수 있도록 도와주며 추가 기능도 있는 확장판
- CSS를 확장하는 스크립팅언어

2

- CSS 파일 크기가 커지면 유지보수의 어려움 발생 ⇒ 해소

3

- CSS는 불필요한 선택자, 연산 기능 한계, 구문 부재의 문제점 ⇒ 해소

기존 CSS

```
header{
  width: 85%;
  margin: auto;
  padding: 35px 0;
  display: flex;
  align-items: center;
  justify-content: space-between;
}
header img{
  width: 150px;
  cursor: pointer;
}
header ul li{
  list-style: none;
  display: inline-block;
  margin: 0 20px;
  position: relative;
}
header ul li::after{
  content: '';
  height: 3px;
  width: 0;
  background: #009688;
  position: absolute;
  left: 0;
  bottom: -10px;
  transition: 0.5s;
}
header ul li:hover::after{
  width: 100%;
}
header ul li a{
  text-decoration: none;
  color: #fff;
}
```

SCSS로 작성

```
body{
  header{
    width: 85%;
    margin: auto;
    padding: 35px 0;
    display: flex;
    align-items: center;
    justify-content: space-between;
    img{
      width: 150px;
      cursor: pointer;
    }
    ul{
      li{
        list-style: none;
        display: inline-block;
        margin: 0 20px;
        position: relative;
        &:after{
          content: '';
          height: 3px;
          width: 0;
          background: #009688;
          position: absolute;
          left: 0;
          bottom: -10px;
          transition: 0.5s;
        }
        &:hover{
          &:after{
            width: 100%;
          }
        }
        a{
          text-decoration: none;
          color: #fff;
        }
      }
    }
  }
}
```

CSS로 자동 변환

```
body header {
  width: 85%;
  margin: auto;
  padding: 35px 0;
  display: flex;
  align-items: center;
  justify-content: space-between;
}
body header img {
  width: 150px;
  cursor: pointer;
}
body header ul li {
  list-style: none;
  display: inline-block;
  margin: 0 20px;
  position: relative;
}
body header ul li:after {
  content: "";
  height: 3px;
  width: 0;
  background: #009688;
  position: absolute;
  left: 0;
  bottom: -10px;
  transition: 0.5s;
}
body header ul li:hover:after {
  width: 100%;
}
body header ul li a {
  text-decoration: none;
  color: #fff;
}
```

SASS

SCSS

문법

들여쓰기

중괄호 {}

속성 구분

줄 바꿈

세미콜론 ;

특징

SASS보다 더 넓은 범용성과 CSS 호환성

CSS

```
nav ul {  
  margin: 0;  
  padding: 0;  
  list-style: none;  
}  
nav li {  
  display: inline-block;  
}  
nav a {  
  display: block;  
  padding: 6px 12px;  
  text-decoration: none;  
}
```

SASS

```
nav  
  ul  
    margin: 0  
    padding: 0  
    list-style: none  
  
  li  
    display: inline-block  
  
  a  
    display: block  
    padding: 6px 12px  
    text-decoration: none
```

SCSS

```
nav {  
  ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
  }  
  
  li { display: inline-block; }  
  
  a {  
    display: block;  
    padding: 6px 12px;  
    text-decoration: none;  
  }  
}
```


vscode내 scss 사용방법

Step01. Variables(변수)

- 스타일시트 전체에서 **재사용**하려는 **정보**를 **저장**
- 색상, 글꼴 스타크 또는 재사용하고 싶은 CSS 값과 같은 것을 저장
- **\$ 기호**를 **사용**하여 무언가를 **변수로 만듦**
- 브랜드 색상으로 작업하고 사이트 전체에서 일관성을 유지할 때 매우 강력

```
<header>
  <h1>SCSS</h1>
</header>
<div class="container">
  <div class="variables">
    <p>1. Variables</p> <!-- 메인 제목은 24px로 고정 -->
    <ul> <!-- 중요한 내용은 빨간색으로 고정 -->
      <li>스타일시트 전체에서 재사용하려는 정보를 저장</li>
      <li>색상, 글꼴 스타크 또는 재사용하고 싶은 CSS 값과 같은 것을 저장</li>
      <li>$ 기호를 사용하여 무언가를 변수로 만듦</li>
      <li>브랜드 색상으로 작업하고 사이트 전체에서 일관성을 유지할 때 매우 강력</li>
    </ul>
  </div>
  <div class="nesting">
    <p>2. Nesting</p> <!-- 메인 제목은 24px로 고정 -->
    <ul> <!-- 중요한 내용은 빨간색으로 고정 -->
      <li>HTML과 동일한 시각적 계층 구조를 따르는 방식으로 CSS 선택기를 중첩</li>
      <li>CSS를 정리하고 가독성을 높이는 좋은 방법</li>
    </ul>
  </div>
</div>
```

```
$mTitle: 24px; // 메인 제목은 24px로 고정
$remember: red; // 중요한 내용은 빨간색으로 고정

.container{
  p{
    font-size: $mTitle;
  }
  .variables{
    ul{
      li{
        &:nth-child(3){
          color: $remember;
        }
      }
    }
  }
  .nesting{
    ul{
      li{
        &:nth-child(1){
          color: $remember;
        }
      }
    }
  }
}
```

Step02. Nesting(중첩)

- Sass를 사용하면 **HTML과 동일한 시각적 계층 구조를 따르는 방식**으로 CSS 선택기 중첩 가능
- CSS를 정리하고 가독성을 높이는 좋은 방법

```
<body>
  <header>
    <h1>SCSS</h1>
  </header>
  <div class="container">
    <div class="variables">
      <p>1. Variables</p>
      <ul> <!-- 중요한 내용 -->
        <li>스타일시트 전
        <li>색상, 글꼴 스
        <li>$ 기호를 사용
        <li>브랜드 색상으
      </ul>
    </div>
    <div class="nesting">
      <p>2. Nesting</p>
      <ul> <!-- 중요한 내용 -->
        <li>HTML과 동일한
        <li>CSS를 정리하
      </ul>
    </div>
  </div>
</body>
```

```
style.css
1  .container p {
2    font-size: 24px;
3  }
4  .container .variables ul li:nth-child(3) {
5    color: red;
6  }
7  .container .nesting ul li:nth-child(1) {
8    color: red;
9  } /*# sourceMappingURL=style.css.map */

style.scss
1  $mTitle: 24px; // 메인 제목은 24px로 고정
2  $remember: red; // 중요한 내용은 빨간색으로 고정
3
4  .container{
5    p{
6      font-size: $mTitle;
7    }
8    .variables{
9      ul{
10       li{
11         &:nth-child(3){
12           color: $remember;
13         }
14       }
15     }
16   }
17   .nesting{
18     ul{
19       li{
20         &:nth-child(1){
21           color: $remember;
22         }
23       }
24     }
25   }
26 }
```

Step03. Partials(부분)

- 한 파일의 콘텐츠를 다른 파일에 포함
- 파일을 CSS에 포함시키면 런타임에 추가 HTTP 호출 필요✕
- 파일 확장자 지정할 필요 없는 경우 Sass는 자동으로 .sass 또는 .scss 파일을 의미한다고 가정

@import를 권장하지 않는 이유

@import 규칙에는 여러가지 심각한 문제가 있습니다.

- @import는 모든 변수, 믹스인, 함수들을 글로벌하게 접근할 수 있습니다.
- 이로 인해 어디에 변수, 믹스인, 함수등이 정의되어 있는지 파악하기 어렵습니다.
- 그렇기 때문에 이름 충돌을 피하기 위해 모든것들 앞에 접두사를 두어 구분해야합니다.
- 각 스타일 시트가 실행되고 @import 될 때마다 CSS가 방출되므로 컴파일 시간이 길어지고 출력이 비대합니다.

Step04. Modules(모듈)

- 모든 Sass를 하나의 파일에 작성할 필요❌ ⇒ 사용 규칙 사용해 원하는 대로 분할
- **다른 Sass 파일 모듈로 로드** ⇒ 파일명을 기반으로 하는 네임스페이스 사용해 Sass 파일에서 **해당 변수, 믹스인 및 함수 참조**
- 파일을 사용하면 컴파일된 출력에 해당 파일이 생성하는 CSS도 포함
- styles.scss에서 **@use 'base';** 사용(파일을 사용할 때는 **파일 확장자 포함 필요❌**)

```
background.scss
1 $bck: #e2e2e2;
2
3 body{
4   background-color: $bck;
5 }

style.scss
1 $mTitle: 24px; // 메인 제목은 24px로 고정
2 $remember: red; // 중요한 내용은 빨간색으로 고정
3
4 @use 'background';
5
6 .container{
7   p{
8     font-size: $mTitle;
9   }
10  .variables{
11    ul{
12      li{
13        &:nth-child(3){
14          color: $remember;
15        }
16      }
17    }
18  }
19  .nesting{
20    ul{
21      li{
22        &:nth-child(1){
23          color: $remember;
24        }
25      }
26    }
27  }
28 }

style.css
1 body {
2   background-color: #e2e2e2;
3 }
4
5 .container p {
6   font-size: 24px;
7 }
8 .container .variables ul li:nth-child(3) {
9   color: red;
10 }
11 .container .nesting ul li:nth-child(1) {
12   color: red;
13 } /*# sourceMappingURL=style.css.map */
```

Step05. Mixins(믹스인)

- 사이트 전체에서 **재사용할 CSS 선언 그룹** 만듦
- Sass를 매우 깔끔하게 유지, 값을 전달하여 믹스인을 더 유연하게 만듦
- **@mixin** 지시문을 사용하고 이름 지정
- **괄호 안에 \$theme 변수** 사용하여 원하는 테마 전달
- 믹스인을 생성한 후에는 **@include로 시작**하고 그 뒤에 믹스인 이름을 붙이는 CSS 선언으로 사용

The screenshot displays a web browser on the left and a code editor on the right. The browser shows the 'SCSS' documentation page, which includes sections on Nesting, Partials, and Modules. The code editor shows the 'index.html' file with HTML structure and the 'style.scss' file with SCSS code. The HTML structure includes a header, a container with a nesting example, a partials section, and a modules section. The SCSS code defines a theme mixin and includes it in the nesting, partials, and modules sections.

SCSS

Nesting

- Sass를 사용하면 HTML과 동일한 시각적 계층 구조를 따르는 방식으로 CSS 선택기 중첩 가능
- CSS를 정리하고 가독성을 높이는 좋은 방법

Partials

- 한 파일의 콘텐츠를 다른 파일에 포함
- 파일을 CSS에 포함시키면 런타임에 추가 HTTP 호출 필요 ❌
- 파일 확장자 지정할 필요 없는 경우 Sass는 자동으로 .sass 또는 .scss 파일을 의미한다고 가정

Modules

- 모든 Sass를 하나의 파일에 작성할 필요 ❌ ⇒ 사용 규칙 사용해 원하는 대로 분할
- 다른 Sass 파일 모듈로 로드 ⇒ 파일명을 기반으로 하는 네임스페이스 사용해 Sass 파일에서 해당 변수, 믹스인 및 함수 참조
- 파일을 사용하면 컴파일된 출력에 해당 파일이 생성하는 CSS도 포함
- styles.scss에서 @use 'base'; 사용(파일을 사용할 때는 파일 확장자 포함 필요 ❌)

index.html

```
<body>
  <header>
    <h1>SCSS</h1>
  </header>
  <div class="container">
    <div class="nesting"> <!-- 바탕: lightgray -->
      <h3>Nesting</h3>
      <ul>
        <li>Sass를 사용하면 HTML과 동일한
        <li>CSS를 정리하고 가독성을 높이는
      </ul>
    </div>
    <div class="partials"> <!-- 바탕: gray -->
      <h3>Partials</h3>
      <ul>
        <li>한 파일의 콘텐츠를 다른 파일에
        <li>파일을 CSS에 포함시키면 런타임에
        <li>파일 확장자 지정할 필요 없는 경
      </ul>
    </div>
    <div class="modules"> <!-- 바탕: lightgray -->
      <h3>Modules</h3>
      <ul>
        <li>모든 Sass를 하나의 파일에 작성할
        <li>다른 Sass 파일 모듈로 로드 ⇒ I
        <li>파일을 사용하면 컴파일된 출력에
        <li>styles.scss에서 @use 'base';
      </ul>
    </div>
  </div>
```

style.scss

```
1
2
3
4
5
6
7
8 @mixin theme($theme: lightgray){
9   background: $theme;
10  box-shadow: 0 0 1px rgba($theme, .25);
11 }
12 .nesting{
13   @include theme;
14 }
15 .partials{
16   @include theme($theme: gray);
17   color: #fff
18 }
19 .modules{
20   @include theme;
21 }
22
```


Step06. Extend(확장)/Inheritance(상속)

- **@extend**: 한 선택기에서 다른 선택기로 CSS 속성 집합 공유
- 플레이스홀더 클래스(%): 확장될 때만 인쇄되는 특수한 유형의 클래스, 컴파일된 CSS를 깔끔하게 유지
- 스타일에서 다른 곳에 중첩된 클래스를 확장✕ 가장 쉬운 방법

The screenshot shows a web browser on the left displaying the SCSS documentation page, and a code editor on the right showing SCSS code examples.

SCSS

Nesting

- Sass를 사용하면 HTML과 동일한 시각적 계층 구조를 따르는 방식으로 CSS 선택기 중첩 가능
- CSS를 정리하고 가독성을 높이는 좋은 방법

Partials

- 한 파일의 콘텐츠를 다른 파일에 포함
- 파일을 CSS에 포함시키면 런타임에 추가 HTTP 호출 필요 ✕
- 파일 확장자 지정할 필요 없는 경우 Sass는 자동으로 .sass 또는 .scss 파일을 의미한다고 가정

Modules

- 모든 Sass를 하나의 파일에 작성할 필요 ✕ ⇒ 사용 규칙 사용해 원하는 대로 분할
- 다른 Sass 파일 모듈로 로드 ⇒ 파일명을 기반으로 하는 네임스페이스 사용해 Sass 파일에서 해당 변수, 믹스인 및 함수 참조
- 파일을 사용하면 컴파일된 출력에 해당 파일이 생성하는 CSS도 포함
- styles.scss에서 @use 'base'; 사용(파일을 사용할 때는 파일 확장자 포함 필요 ✕)

```
1
2
3
4
5
6 %stepShared{
7     background: lightcyan;
8     padding: 10px;
9     color: #000;
10 }
11 .nesting{
12     @extend %stepShared;
13 }
14 .partials{
15     @extend %stepShared;
16 }
17 .modules{
18     @extend %stepShared;
19     ul{
20         li{
21             &:nth-child(4){
22                 color: red;
23             }
24         }
25     }
26 }
```

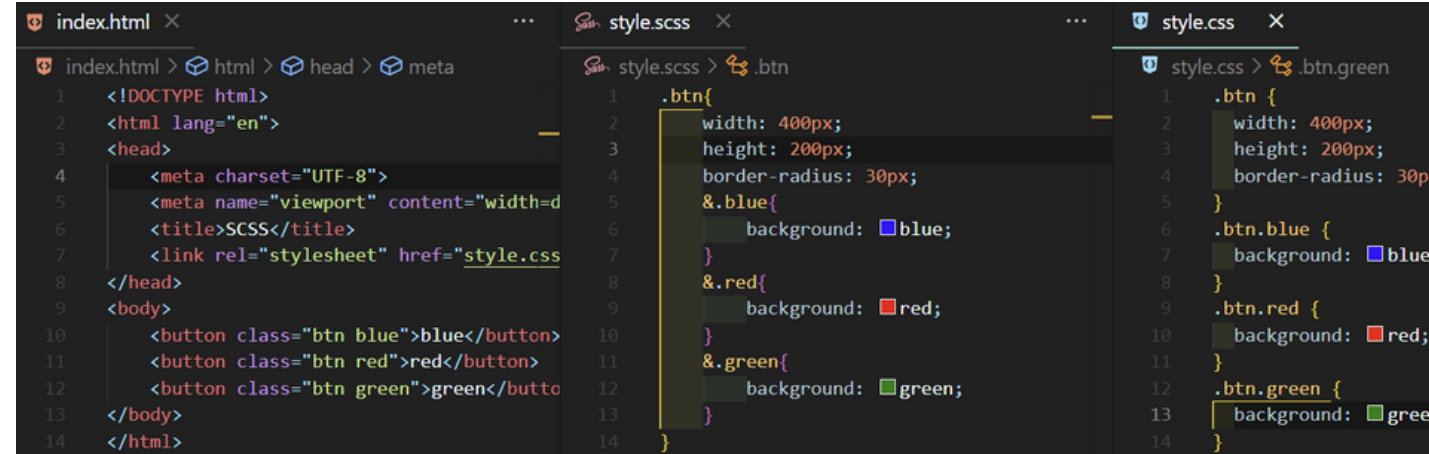
Step07. Operators(연산자)

- $+$, $-$, $*$, `math.div()`, `&`와 같은 몇 가지 표준 수학 연산자

Step08. 선택자(&, &:, &::)

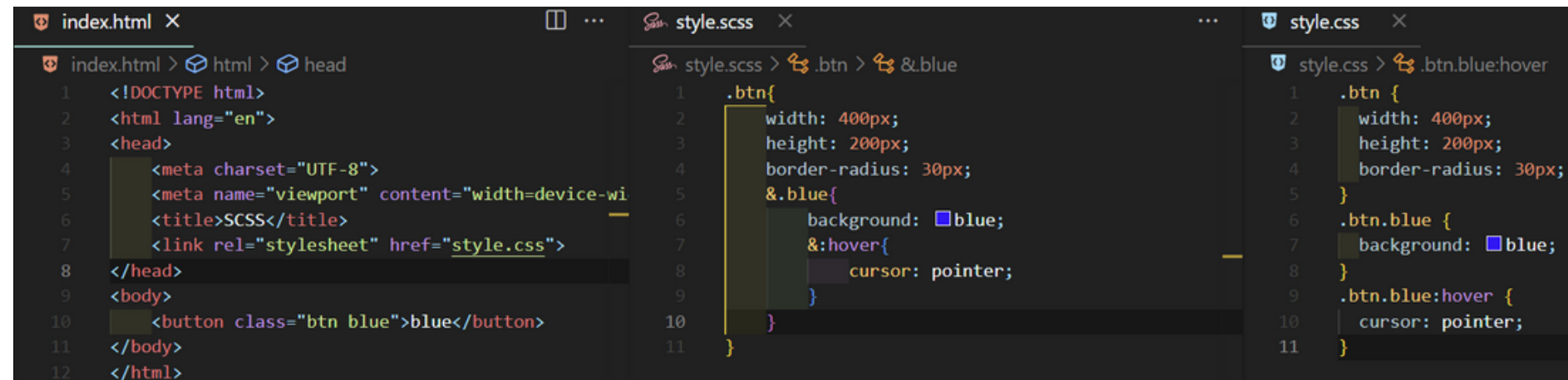
&

- 상위(부모) 선택자, **nesting** 기능



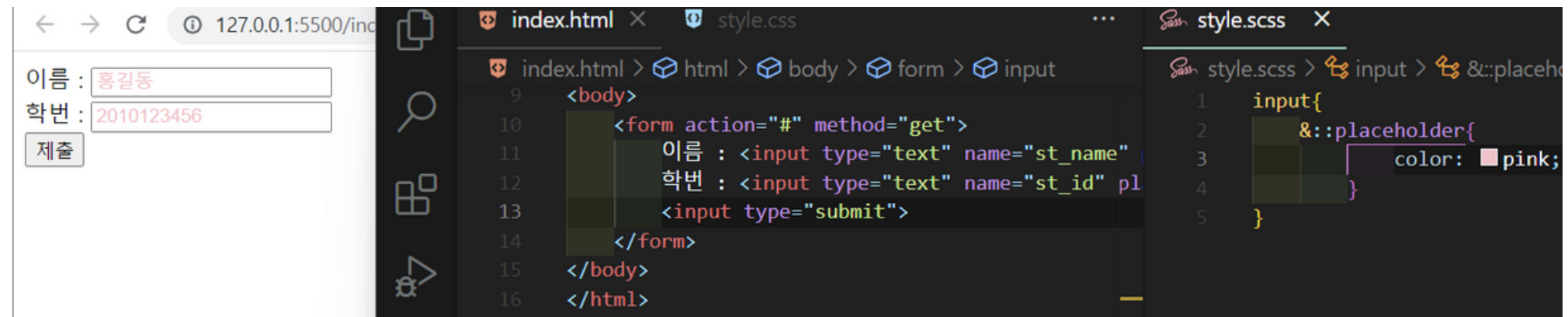
&

- 가상 선택자
- **:뒤에 특정 이벤트**를 적고 그 특정 이벤트가 발생할 때 나타날 동작을 적용



&::

- 가상 선택자
- **::뒤 placeholder**를 입력해서 **인풋창의 placeholder창 배경색 변경**



We're done!

A green speech bubble with a tail pointing downwards and to the left, containing white text.

Thank you for
participating. Have a
great day ahead.

과제는 저번주와 동일