Chapter 15. 스타일링

교재: 처음만난 리액트 (저자: 이인제, 한빛출판사)

Contents

- CHAPTER 15: 스타일링

Styling - 웹사이트의 전체 레이아웃 구성을 포함하여 버튼의 색깔, 테두리, 폰트 크기, 색상 등을 모두 포함하는 개념

리액트 컴포넌트를 어떻게 원하는 모양으로 만들고 원하는 곳에 배치하는 방법에 대해 학습

15.1 CSS

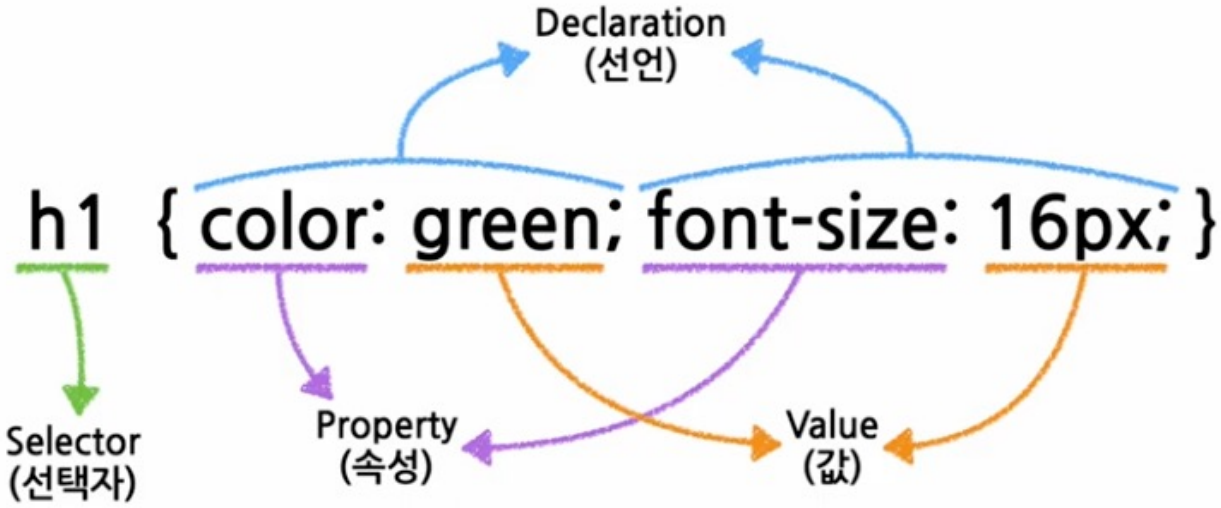
15.2 styled-components

15.3 (실습) styled-components를 사용하여 스타일링 해보기

SECTION 15.1 CSS

- **CSS (Cascading Style Sheets)**
  - HTML – 콘텐츠의 구조와 의미를 정의
  - CSS – 스타일과 레이아웃을 정의

  - CSS는 크게 선택자(selector)와 스타일로 구성



Selector - 스타일을 어떤 엘리먼트에 적용할지를 선택하게 해주는 것

SECTION 15.1 CSS

Selector의 유형

- Element selector
- ID selector
- Class selector
- Universal selector
- Grouping selector
- Element의 상태와 관련된 selector

SECTION 15.1 CSS

◦ Selector의 유형

- Element selector – 특정 HTML 태그를 선택

```css
h1 {
  color: green;
}
```

- ID selector – element에 고유한 id를 정의하고 이 id를 기반으로 선택

  #뒤에 id를 붙여서 사용

```
<div id="section">
  ...
</div>

#section {
  background-color: blue;
}
```

SECTION 15.1 CSS

◦ Selector의 유형

- Class selector – 여러 개의 element를 분류하기 위해 사용

  . 뒤에 클래스명을 붙여서 사용

```
<span class="medium">
  ...
</span>
<p class="medium">
  ...
</p>

.medium {
  font-size: 20px;
}

p.medium {
  font-size: 30px;
}
```

SECTION 15.1 CSS

◦ Selector의 유형

- 전체 선택자 (Universal selector) – 전체 엘리먼트에 적용하기 위한 선택자. * 를 사용

```
* {
  font-size: 20px;
  color: blue;
}
```

- 그룹 선택자 (grouping selector) – 여러 가지 선택자를 그룹으로 묶어 하나의 스타일을 적용하기 위해 사용하는 선택자

```
h1 {
  color: black;
  text-align: center;
}

h2 {
  color: black;
  text-align: center;
}
```

```
h1, h2 {
  color: black;
  text-align: center;
}
```

SECTION 15.1 CSS

◦ Selector의 유형

- 상태와 관련된 선택자
  - :hover – 마우스 커서가 엘리먼트 위에 올라왔을 때
  - :active – 주로 <a> 태그 에 사용. 엘리먼트가 클릭됐을 때
  - :focus – 주로 <input> 태그에 사용. 엘리먼트가 focus 되었을 때
  - :checked – radio, checkbox 같은 유형의 input 태그가 체크되어 있는 경우
  - :first-child, :last-child – 상위 엘리먼트를 기준으로 각각 첫 번째 child, 마지막 child일 경우

```css
button:hover {
  font-weight: bold;
}

a:active {
  color: red;
}

input:focus {
  color: #000000;
}
```

```css
option:checked {
  background: #00ff00;
}

p:first-child {
  background: #ff0000;
}

p:last-child {
  background: #0000ff;
}
```

SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

- 화면에 엘리먼트들을 어떻게 배치할 것인지 결정

  - display – 엘리먼트를 어떻게 표시할지에 관한 속성

```
div {
  display: none | block | inline | inline-block | flex;
}
```

    - none: 엘리먼트를 화면에 숨기기 위해 사용
    - block: 블록 단위로 엘리먼트를 배치. 새로운 줄에서 시작하여 위치한 곳 전체의 width를 차지
      <p>, , <h1>~<h6> 태그의 기본 속성
    - inline: 줄바꿈 없이 한 줄에 다른 엘리먼트들과 나란히 배치. width, height 속성은 무시됨
      , <a>, 태그의 기본 속성
    - inline-block: inline처럼 동작하면서 width, height, margin, padding 등 속성 지정 가능
      <button>, <input>, <select> 태그의 기본 속성
    - flex: 엘리먼트를 블록 레벨의 플렉스 컨테이너(flex container)로 표시하는 것
      내부에 다른 엘리먼트들을 포함

SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

- display: block

```
before
  <h1>H1</h1>
  <div>div</div>
  <p>P</p>
after
```

before

H1

div

P

after

- display: inline

```
before
  <a>A</a>
  <span>SPAN</span>
  <em>EM</em>
after
```

```
span {
  background: yellow;
  width: 200px;
  height: 50px;
  margin: 20px;
  padding: 10px;
}
```

before A SPAN *EM* after

SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

- display: inline-block

```
before
  <a>A</a>
  <span>SPAN</span>
  <em>EM</em>
after
```

```
span {
  display: inline-block;
  background: yellow;
  width: 200px;
  height: 50px;
  margin: 20px;
  padding: 10px;
}
```

SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

- visibility – 엘리먼트를 화면에 보여주거나 감추기 위해 사용하는 속성

```
div {
  visibility: visible | hidden;
}
```

- visible – 엘리먼트를 화면에 보여줌
- hidden – 엘리먼트를 화면에서 안 보이게 감춤
  display:none; 과 달리 화면에서 영역은 그대로 차지

SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

- position – 엘리먼트를 어떻게 위치시킬 것인지 정의

```
div {
  position: static | fixed | relative | absolute;
}
```

- static – (default) 엘리먼트를 원래 있어야 하는 위치(작성된 순서)에 배치
- fixed – 엘리먼트를 브라우저 window(뷰포트)에 상대적으로 위치하도록 배치
  채팅상담, 상담톡 등 화면의 고정 위치에 배치
- relative – 엘리먼트를 원래 위치를 기준으로 상대적으로 배치
- absolute – 엘리먼트를 절대 위치에 위치시킴
  이때 배치 기준은 position 속성이 static이 아닌 첫번째 상위 요소 기준
  조건에 맞는 상위 요소가 없다면 body 요소가 배치 기준

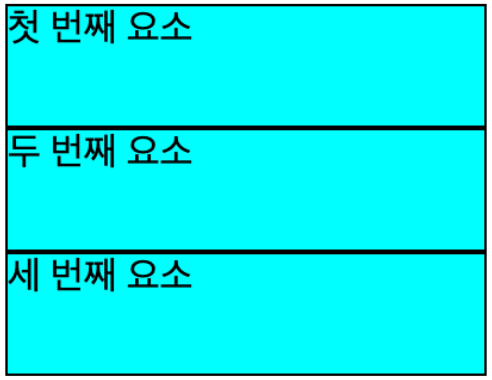
SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

- position: static

```html
<main>
  <div>첫 번째 요소</div>
  <div>두 번째 요소</div>
  <div>세 번째 요소</div>
</main>
```

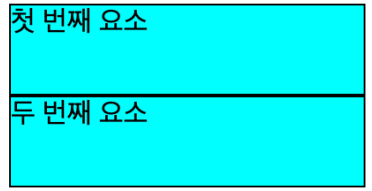
```css
div {
  position: static;
  width: 200px;
  height: 50px;
  border: 1px solid;
  background: cyan;
}
```

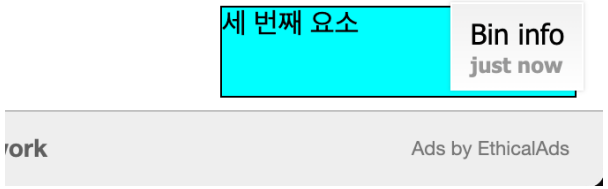


- position: fixed

```html
<main>
  <div>첫 번째 요소</div>
  <div>두 번째 요소</div>
  <div id="third">세 번째 요소</div>
</main>
```

```css
div {
  position: static;
  width: 200px;
  height: 50px;
  border: 1px solid;
  background: cyan;
}

div#third {
  position: fixed;
  bottom: 8px;
  right: 16px;
}
```





SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

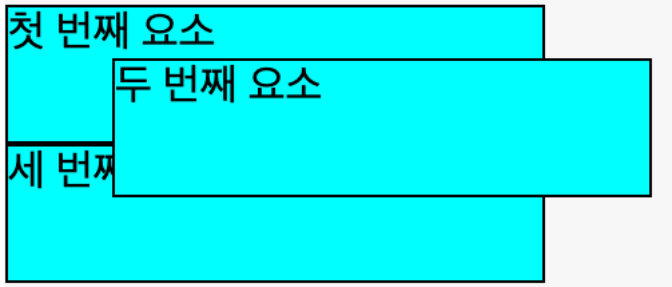
- position: relative

```
<main>
  <div>첫 번째 요소</div>
  <div class="second">두 번째 요소</div>
  <div>세 번째 요소</div>
</main>
```

- position: absolute

```
<main>
  <div>첫 번째 요소</div>
  <div id="second">두 번째 요소</div>
  <div>세 번째 요소</div>
</main>
```

```
div {
  position: static;
  width: 200px;
  height: 50px;
  border: 1px solid;
  background: cyan;
}

div.second {
  position: relative;
  top: 28px;
  left: 48px;
}

div#second {
  position: absolute;
  top: 28px;
  left: 48px;
}
```



SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

- 플랙스박스(Flexbox)
  - 기존 CSS 레이아웃 사용의 불편한 부분을 개선하기 위해 등장
  - 컨테이너와 아이템으로 구성

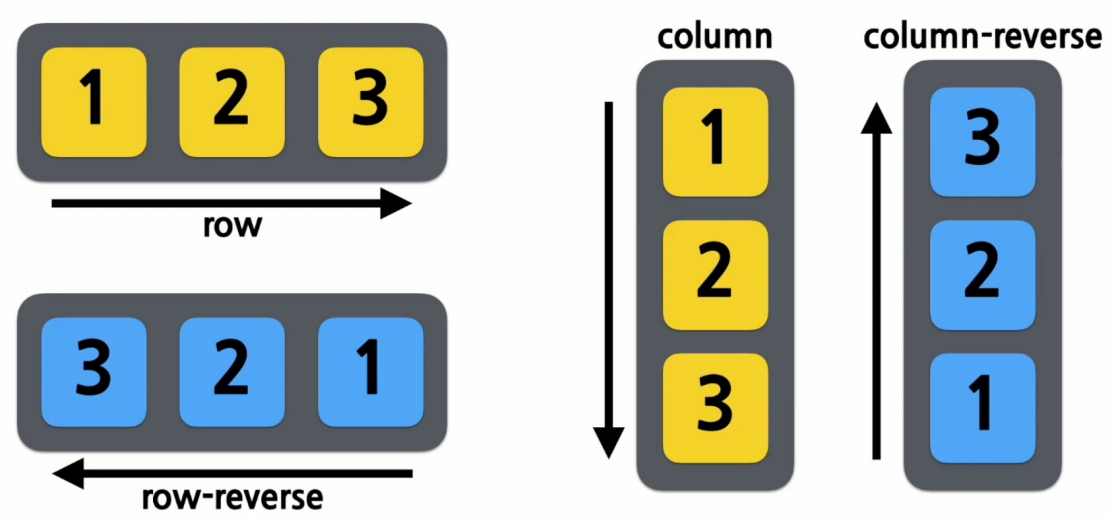

```
div {
  display: flex;
  flex-direction: row | column | row-reverse | column-reverse;
  align-item: stretch | flex-start | center | flex-end | baseline;
  justify-content: flex-start | center | flex-end | space-between | space-around;
}
```

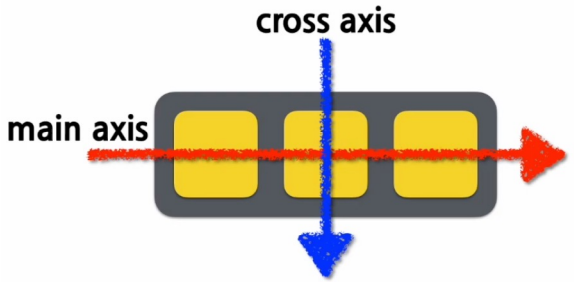
SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

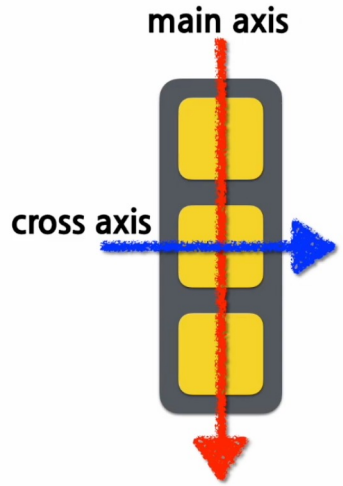
○ flex-direction

- row: (default) 아이템을 행(row)을 따라 가로 순서대로 왼쪽부터 배치
- column: 아이템을 열(column)을 따라 세로 순서대로 위쪽부터 배치
- row-reverse: 아이템을 행(row)의 역(reverse)방향으로 오른쪽부터 배치
- column-reverse: 아이템을 열(column)의 역(reverse)방향으로 아래쪽부터 배치





flex-direction이 row인 경우

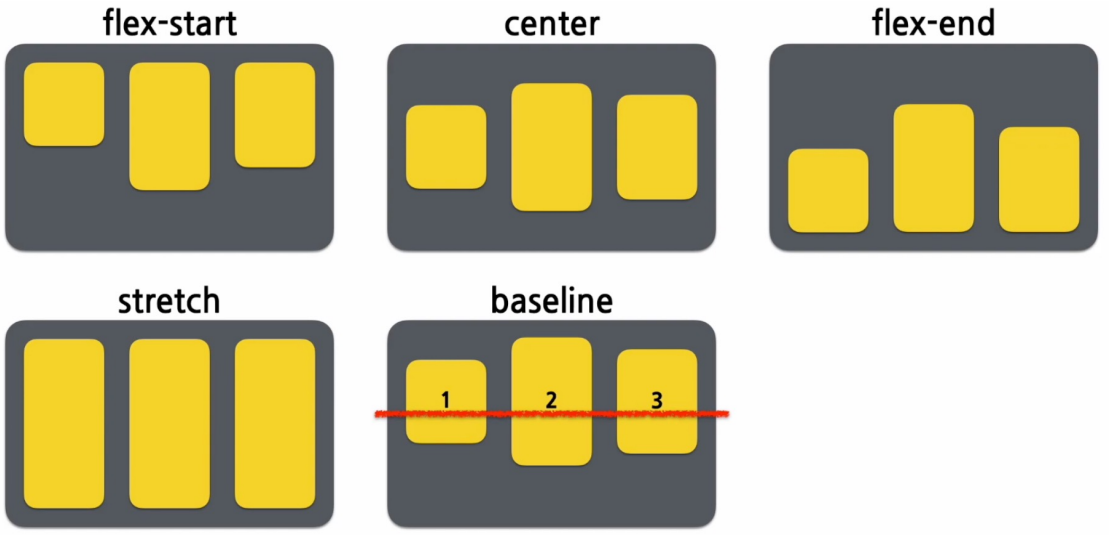


flex-direction이 column인 경우

SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

- align-items: 컨테이너 내에서 아이템을 정렬하는 방법(cross axis 기준)
  - stretch: (default) 아이템을 늘려서(stretch) 컨테이너를 가득 채움
  - flex-start: cross axis의 시작 지점으로 아이템을 정렬
  - center: cross axis의 중앙으로 아이템을 정렬
  - flex-end: cross axis의 끝 지점으로 아이템을 정렬
  - baseline: 아이템을 baseline 기준으로 정렬

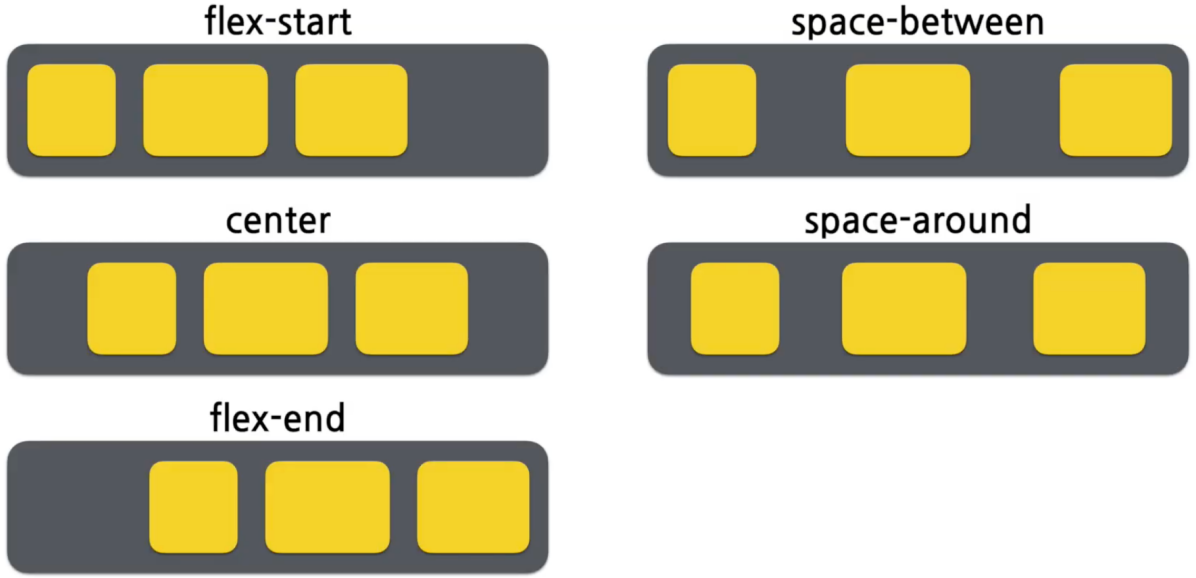


SECTION 15.1 CSS

◦ 레이아웃과 관련된 속성

◦ justify-content: 컨테이너 내에서 아이템을 나란히 맞추는 방법(main axis 기준)

- flex-start: main axis의 시작 지점으로 아이템을 정렬
- center: main axis의 중앙으로 아이템을 정렬
- flex-end: main axis의 끝 지점으로 아이템을 정렬
- space-between: main axis 기준으로 첫 아이템은 시작점, 마지막 아이템은 끝 지점에 맞추며, 중간에 있는 아이템들 사이의 간격이 일정하게 유지되도록 맞춤
- space-around: main axis를 기준으로 각 아이템의 주변(around) 간격(space)을 동일하게 맞춤



SECTION 15.2 styled-components

- **styled-components**
  - **CSS 문법을 그대로 사용하면서 결과물을 스타일링된 컴포넌트 형태로 만들어주는 오픈소스 라이브러리**

- styled-components 설치하기 (터미널)

```
npm install styled-components@latest
```

- npm 패키지 관련 명령어
  - 패키지 설치 – install, i

    npm install <패키지명>@버전 (특정 버전을 설치할 경우)

    npm install –g <패키지명> (패키지를 전역으로 설치할 경우 –g 옵션)
  - 패키지 업데이트 – update

    npm update <패키지명>
  - 패키지 삭제 – uninstall

    npm uninstall <패키지명>

SECTION 15.2 styled-components

- **styled-components 예제**

```
import styled from 'styled-components';

const Wrapper = styled.div`
  padding: 1em;
  background: grey;
`;

const Title = styled.h1`
  font-size: 1.5em;
  color: white;
  text-align: center;
`;

function MainPage(props) {
  return(
    <Wrapper>
      <Title>안녕, 리액트</Title>
    </Wrapper>
  );
}

export default MainPage;
```

SECTION 15.2 styled-components

◦ styled-components 기본 사용법

- styled-components는 태그드 템플릿 리터럴을 사용하여 구성 요소의 스타일을 지정

- 리터럴(Literal)

```
let number = 20;
```

소스코드의 고정된 값

```
// 정수 리터럴 (Integer literal)
const myNumber = 10;

// 문자열 리터럴 (String literal)
const myStr = 'Hello';

// 배열 리터럴 (Array literal)
const myArr = [];

// 객체 리터럴 (Object literal)
const myObject = {};
```

SECTION 15.2 styled-components

styled-components 기본 사용법

- 템플릿 리터럴(template literal)
  - 리터럴을 템플릿 형태로 사용하는 자바스크립트 문법
  - backticks(`)를 사용하여 문자열을 작성하고, 그 안에 대체 가능한 expression을 넣는 방법

- untagged/tagged 템플릿 리터럴
  - untagged template literal - 문자열을 여러 줄에 걸쳐서 작성하거나 포매팅을 위해 사용
  - tagged template literal - 앞에 있는 태그 함수를 호출하여 결과를 리턴

```
myFunction`string text ${expression} string text`;
```

SECTION 15.2 styled-components

◦ styled-components 기본 사용법

- tagged 템플릿 리터럴
  - 태그 함수의 파라미터는 expresion으로 구분된 문자열 배열과 expresion이 순서대로 들어감

```
const name = 'Jeonghun';
const region = 'Seoul';

function tagFunction(strings, nameExp, regionExp) {
  let str0 = strings[0];
  let str1 = strings[1];
  let str2 = strings[2];

  return `${str0}${nameExp}${str1}${regionExp}${str2}`;
}

const output = tagFunction`제 이름은 ${name}이고, 사는 곳은
${region}입니다.`;

console.log(output);
// 제 이름은 Jeonghun이고, 사는 곳은 서울입니다.
```

SECTION 15.2 styled-components

◦ styled-components 기본 사용법

- **styled-components는 태그드 템플릿 리터럴을 사용하여 CSS 속성이 적용된 리액트 컴포넌트를 만듦**

  - backticks(`)로 둘러싸인 문자열 부분에 CSS 속성을 넣음
  - 태그 함수 위치에 styled.<HTML 태그> 형태로 사용
  - HTML 태그에 CSS 속성들이 적용된 형태의 리액트 컴포넌트가 만들어짐

```
import styled from 'styled-components';

const Wrapper = styled.div`
  padding: 1em;
  background: grey;
`;
```

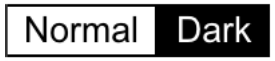
SECTION 15.2 styled-components

◦ styled-components의 props 사용하기

- 조건이나 동적으로 변하는 값을 사용해서 스타일링하기 위한 기능

(react component의 props와 같은 개념)

```
import styled from 'styled-components';

const Button = styled.button`
  color: ${props => props.dark ? 'white' : 'black'};
  background: ${props => props.dark ? 'black' : 'white'};
  border: 1px solid black;
`;

function Sample(props) {
  return (
    <div>
      <Button>Normal</Button>
      <Button dark>Dark</Button>
    </div>
  );
}

export default Sample;
```



SECTION 15.2 styled-components

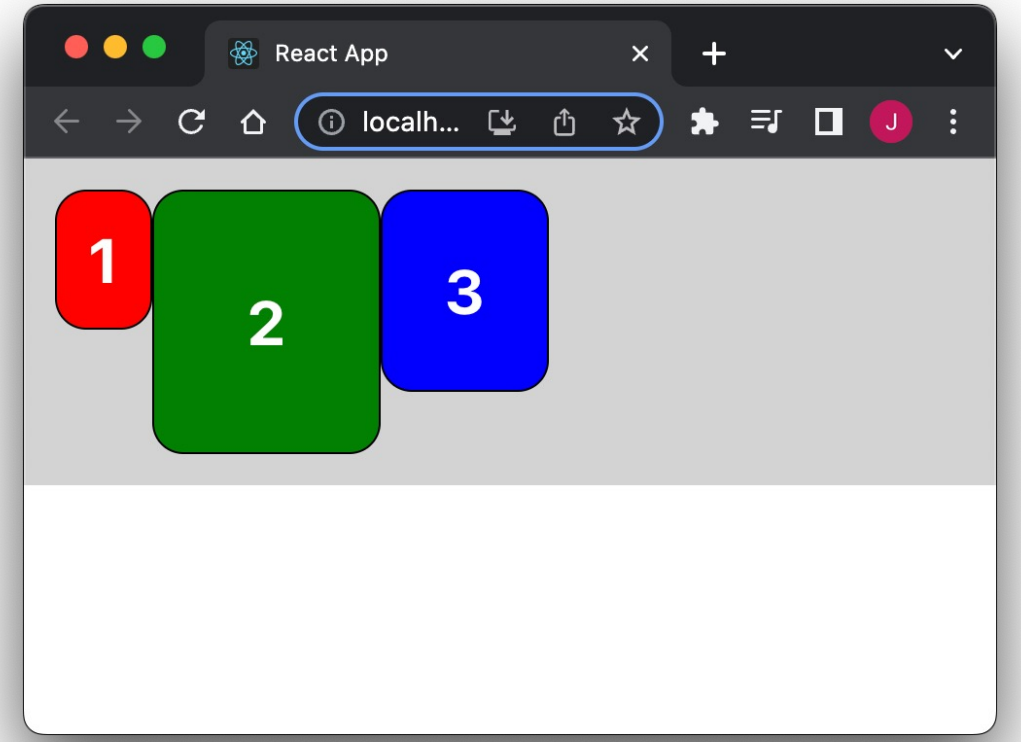
◦ styled-components의 스타일 확장하기

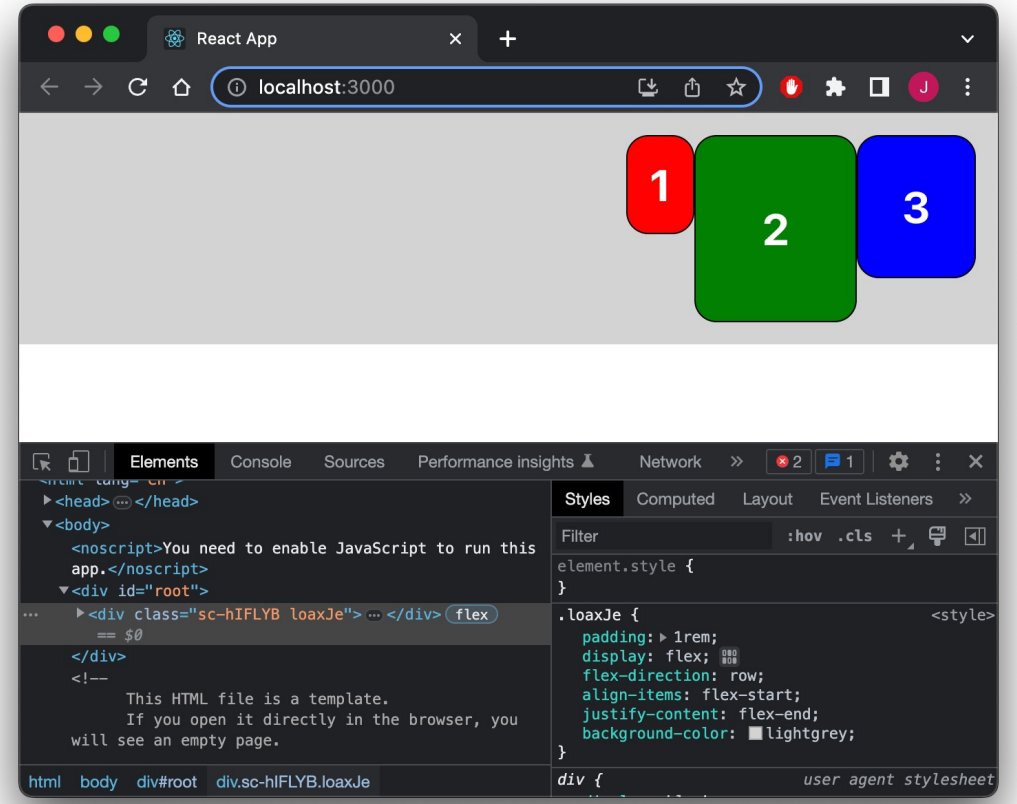
- 생성된 컴포넌트를 기반으로 추가적인 스타일을 적용할 수 있는 기능

```
import styled from 'styled-components';

const Button = styled.button`
  color: grey;
  border: 2px solid palevioletred;
`;

const RoundedButton = styled(Button)`
  border-radius: 16px;
`;

function Sample(props) {
  return (
    <div>
      <Button>Normal</Button>
      <RoundedButton>Rounded</RoundedButton>
    </div>
  );
}

export default Sample;
```

Practice 15.3 styled-components를 사용하여 스타일링해 보기

- 실행 결과





Practice 15.3 styled-components를 사용하여 스타일링해 보기

- **Blocks.jsx**

```
import styled from 'styled-components';

const Wrapper = styled.div`
  padding: 1rem;
  display: flex;
  flex-direction: row;
  align-items: flex-start;
  justify-content: flex-start;
  background-color: lightgrey;
`;

const Block = styled.div`
  padding: ${(props) => props.padding};
  border: 1px solid black;
  border-radius: 1rem;
  background-color: ${(props) => props.backgroundColor};
  color: white;
  font-size: 2rem;
  font-weight: bold;
  text-align: center;
`;
```

```
const blockItems = [
  {
    label: '1',
    padding: '1rem',
    backgroundColor: 'red'
  },
  {
    label: '2',
    padding: '3rem',
    backgroundColor: 'green'
  },
  {
    label: '3',
    padding: '2rem',
    backgroundColor: 'blue'
  },
]
```

Practice 15.3 styled-components를 사용하여 스타일링해 보기

- **Blocks.jsx**

```
function Blocks(props) {
  return (
    <Wrapper>
      {blockItems.map((blockItem) => {
        return (
          <Block
            padding={blockItem.padding}
            backgroundColor={blockItem.backgroundColor}
          >
            {blockItem.label}
          </Block>
        );
      })}
    </Wrapper>
  )
}

export default Blocks;
```

[요약]

- styled-components
  - CSS 문법을 그대로 사용하면서 결과물을 스타일링된 컴포넌트 형태로 만들어주는 오픈소스 라이브러리
  - 컴포넌트 개념을 사용하기 때문에 리액트와 궁합이 잘 맞음
  - 태그트 템프릿 리터럴을 사용하여 구성 요소의 스타일을 지정