06. Props를 사용해 데이터 표시하기

지금까지 `<Header />` 컴포넌트를 재사용하면 항상 같은 내용을 보여주었습니다.

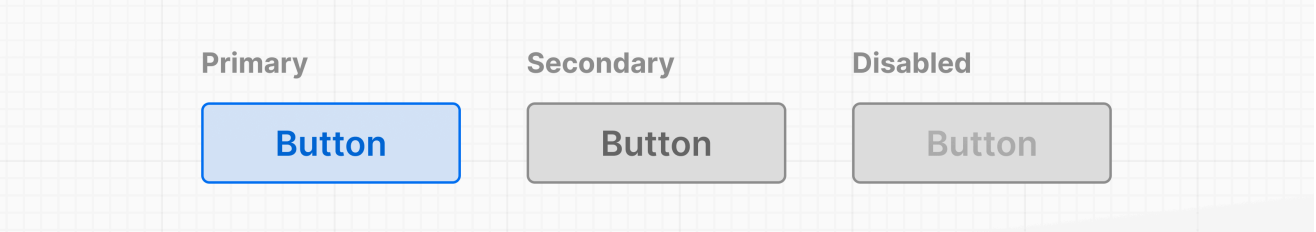
```
function Header() {
  return <h1>Develop. Preview. Ship.</h1>;
}

function HomePage() {
  return (
    <div>
      <Header />
      <Header />
    </div>
  );
}
```

하지만 다른 텍스트를 표시하거나, 외부 데이터 소스에서 가져온 정보를 표시하고 싶다면 어떻게 해야 할까요?

일반 HTML 요소들은 `src`, `href` 와 같은 속성(attributes)을 통해 동작을 바꿀 수 있습니다. 마찬가지로, React 컴포넌트에도 정보를 전달할 수 있는데, 이를 **props**(properties)라고 합니다.

예를 들어, 버튼 컴포넌트에는 다양한 변형(Primary, Secondary, Disabled)이 있을 수 있습니다:



JavaScript 함수처럼, React 컴포넌트도 **인자(arguments)**를 받아 동작이나 화면 출력을 변경할 수 있습니다. 부모 컴포넌트에서 자식 컴포넌트로 데이터를 전달하는 방식을 사용하는 것이죠.

> 💬 참고: React에서는 데이터가 **한 방향(Top → Bottom)** 으로만 흐릅니다. 이를 "one-way data flow"라고 부릅니다.

Props 사용하기

`HomePage` 컴포넌트에서 `Header` 컴포넌트로 `title` 이라는 **props**를 전달해보겠습니다:

```
function HomePage() {
  return (
    <div>
      <Header title="React" />
    </div>
  );
}
```

그리고 `Header` 컴포넌트는 첫 번째 매개변수로 props를 받을 수 있습니다:

```
function Header(props) {
  return <h1>Develop. Preview. Ship.</h1>;
}
```

`console.log(props)`를 찍어보면 다음처럼 보입니다:

```
function Header(props) {
  console.log(props); // { title: "React" }
  return <h1>Develop. Preview. Ship.</h1>;
}
```

구조 분해 할당(destructuring) 을 사용하면 코드를 더 깔끔하게 만들 수 있습니다:

```
function Header({ title }) {
  console.log(title); // "React"
  return <h1>Develop. Preview. Ship.</h1>;
}
```

이제 `h1` 안에 `{title}` 변수를 넣어볼까요?

```
function Header({ title }) {
  return <h1>title</h1>;
}
```

브라우저에서는 `title`이라는 글자 그대로가 출력됩니다. 이유는 React가 `title`을 문자열로 인식했기 때문입니다.

JSX 안에서 변수 사용하기

변수를 사용하려면 중괄호 `{}`로 감싸야 합니다:

```
function Header({ title }) {
  return <h1>{title}</h1>;
}
```

중괄호 `{}`를 사용하면 **JSX 안에서 JavaScript 표현식**을 삽입할 수 있습니다.

다른 예제들도 볼까요?

1. 객체 속성 접근

```
function Header(props) {
  return <h1>{props.title}</h1>;
}
```

2. 템플릿 리터럴 사용

```
function Header({ title }) {
  return <h1>{`Cool ${title}`}</h1>;
}
```

3. 함수 반환값 사용

```
function createTitle(title) {
  return title ? title : 'Default title';
}

function Header({ title }) {
  return <h1>{createTitle(title)}</h1>;
}
```

4. 삼항 연산자 사용

```
function Header({ title }) {
  return <h1>{title ? title : 'Default Title'}</h1>;
}
```

이렇게 하면 `title`을 안 넘겼을 때 기본값도 설정할 수 있습니다:

```
function Header({ title }) {
  return <h1>{title ? title : 'Default title'}</h1>;
}

function HomePage() {
  return (
    <div>
      <Header />
    </div>
```

```
  );
}
```

Props로 다양한 데이터 전달하기

이제 `title`만 바꿔서 여러 `Header`를 만들어봅시다:

```
function HomePage() {
  return (
    <div>
      <Header title="React" />
      <Header title="A new title" />
    </div>
  );
}
```

리스트 반복 렌더링하기

리스트 형태의 데이터를 화면에 출력할 때, JavaScript 배열 메서드를 사용할 수 있습니다.

`HomePage` 컴포넌트에 배열을 추가해봅시다:

```
function HomePage() {
  const names = ['Ada Lovelace', 'Grace Hopper', 'Margaret Hamilton'];

  return (
    <div>
      <Header title="Develop. Preview. Ship." />
      <ul>
        {names.map((name) => (
          <li>{name}</li>
        ))}
      </ul>
    </div>
  );
}
```

`map()` 메서드를 사용해 배열의 각 항목을 `<li>`로 매핑(mapping)하는 것이죠.

여기서도 중괄호 `{}`를 사용하여 JSX 안에서 JavaScript 코드를 작성했습니다.

하지만 여기서 React는 경고를 띄웁니다:

> "Warning: Each child in a list should have a unique 'key' prop."

리스트 항목에 **key**를 추가해 고유 식별자를 제공해야 합니다.

```
function HomePage() {
  const names = ['Ada Lovelace', 'Grace Hopper', 'Margaret Hamilton'];

  return (
    <div>
      <Header title="Develop. Preview. Ship." />
      <ul>
        {names.map((name) => (
          <li key={name}>{name}</li>
        ))}
      </ul>
    </div>
  );
}
```

참고:
key 값은 고유해야 합니다. 실제 애플리케이션에서는 ID 같은 진짜 고유값을 사용하는 것이 더 좋습니다.