컴포넌트 디자인 패턴과 커스텀 훅

Chap 1. 컴포넌트의 구성 요소와 커스텀 훅

컴포넌트의 3가지 구성

```jsx
import { useState } from "react";

function Bookmark() {
  const [currentTab, setCurrentTab] = useState("heart");

  const onClickHeartTabHandler = () => setCurrentTab("heart");
  const onClickGoodsTabHandler = () => setCurrentTab("Goods");

  return (
    <>
      <div>HEAD</div>
      <BookmarkTab
        state={currentTab}
        handler={(onClickHeartTabHandler, onClickGoodsTabHandler)}
      />
      <SMainWrap>{state === "heart" ? <Goods /> : <Station />}</SMainWrap>
      <BottomNav />
    </>
  );
}

export default Bookmark;

const SMainWrap = styled.main`
  padding: 0 16px;
`;

```

기능

상호작용

UI

기능, 상호작용, UI는 컴포넌트에 종속적이면 좋을까?

커스텀 훅이란

Hook은 React 16.8 버전에 새로 추가되었습니다.
Hook은 클래스 컴포넌트를 작성하지 않아도 state와 같은 특징들을 사용할 수 있다.
자신만의 Hook을 만들면 컴포넌트 로직을 함수로 뽑아내어 재사용할 수 있습니다.

→ 리액트 공식문서 (레거시) 발췌

커스텀 훅을 작성한다

= 기능과 상호작용을 재사용할 수 있다

컴포넌트의 3가지 구성

```jsx
import { useState } from "react";

function Bookmark() {
  const [currentTab, setCurrentTab] = useState("heart");

  const onClickHeartTabHandler = () => setCurrentTab("heart");
  const onClickGoodsTabHandler = () => setCurrentTab("Goods");

  return (
    <>
      <div>HEAD</div>
      <BookmarkTab
        state={currentTab}
        handler={(onClickHeartTabHandler, onClickGoodsTabHandler)}
      />
      <SMainWrap>{state === "heart" ? <Goods /> : <Station />}</SMainWrap>
      <BottomNav />
    </>
  );
}

export default Bookmark;

const SMainWrap = styled.main`
  padding: 0 16px;
`;

```

기능

상호작용

UI

컴포넌트의 3가지 구성

```jsx
import { useState } from "react";

function useManageTab() {
  const [active, setActive] = useState("bookmark");
  const onClickTabHandler = (name) => {
    setActive(name);
  };
  return [active, onClickTabHandler];
}
export default useManageTab;
```

커스텀 훅

기능 상호작용

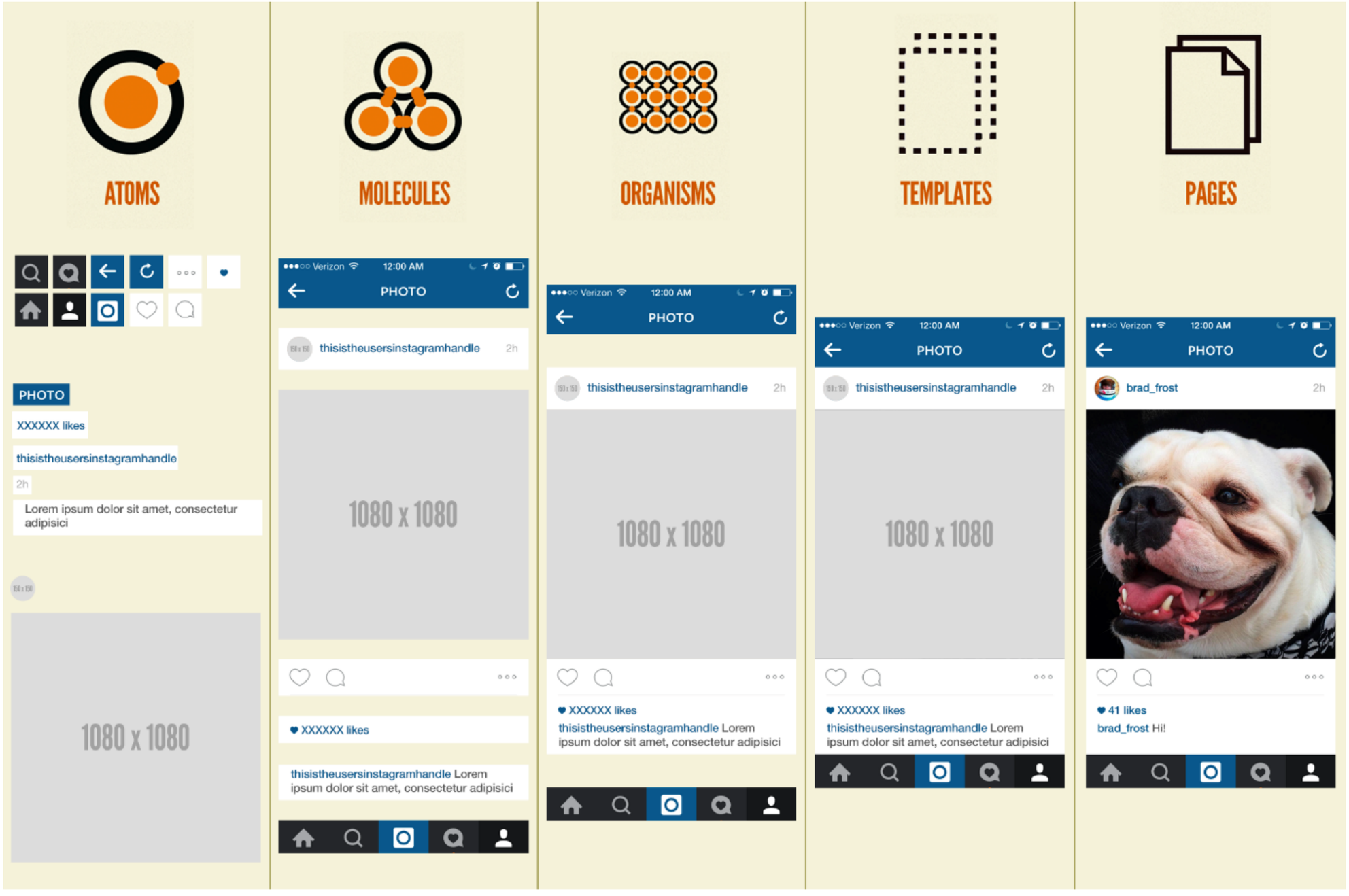
```jsx
function Bookmark() {
  const [state, handler] = useManageTab();
  return (
    <>
      <div>HEAD</div>
      <BookmarkTab state={state} handler={handler} />
      <SMainWrap>{state === "heart" ? <Goods /> : <Station />}</SMainWrap>
      <BottomNav />
    </>
  );
}
export default Bookmark;
```

컴포넌트

UI

Chap 2. 컴포넌트 디자인 패턴

아토믹 디자인 패턴


ATOMS
MOLECULES
ORGANISMS
TEMPLATES
PAGES
PHOTO
XXXXXX likes
thisistheusersinstagramhandle
2h
Lorem ipsum dolor sit amet, consectetur adipisici
1080 x 1080
Verizon
12:00 AM
PHOTO
thisistheusersinstagramhandle
2h
XXXXXX likes
thisistheusersinstagramhandle Lorem ipsum dolor sit amet, consectetur adipisici
brad_frost
41 likes
brad_frost Hi!

아토믹 디자인 패턴엔 단점이 없을까?

디자이너가 비슷한 컴포넌트를 만들면...

해당 위치에 시간 표기, 드롭다운 설정

각각 있는 컴포넌트 2개 만들어주세요!

Ctrl C + Ctrl V 후 컴포넌트 변경했는데..

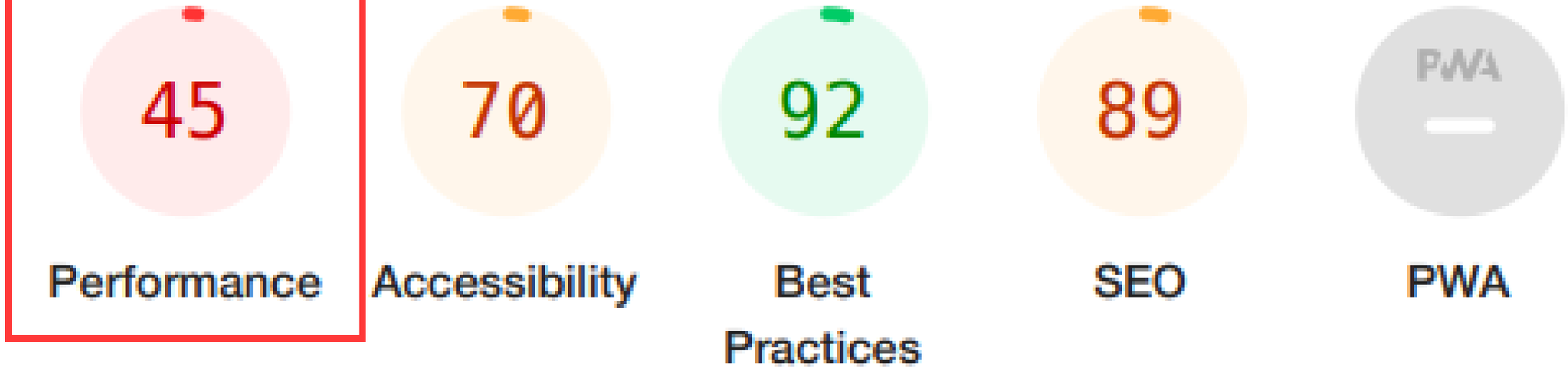
디자이너가 비슷한 컴포넌트를 만들면...

atomic 요소만 다른
molecules 컴포넌트 추가

이런 일이 종종 안 일어난다고 할 수 있을까?

컴포넌트의 나비효과

Lighthouse 성능 측정 결과



→ 처참한 웹 성능 → 사용자의 사이트 이탈 → 사업 망함

그렇다면 어떻게 컴포넌트를 만들어야 할까?

합성 vs 상속

React는 강력한 합성 모델을 가지고 있으며, 상속 대신 합성을 사용하여 컴포넌트 간에 코드를 재사용하는 것이 좋습니다.

→ React 공식문서 (레거시) 발췌

합성을 사용하게 되면...

```jsx
function BottomNavBtn({ children, url, active }) {
  return (
    <NavBtn to={url} active={active}>
      {children}
    </NavBtn>
  );
}

export default BottomNavBtn;

const NavBtn = styled(Link)`
  width: 100%;
  padding: 6px 8px;
  display: flex;
  flex-direction: column;
  align-items: center;
  border-top: 1px solid ${(props) => Palette[props.active]};
`;

const Palette = {
  black: theme.color.grayScale500,
  gray: theme.color.grayScale300,
};
```

컴포넌트에서 UI 구현부분이 간단해진다

합성을 사용하게 되면...

```jsx
return (
  <Nav>
    {Object.keys(icons).map((key) => (
      <BottomNavBtn
        key={key}
        url={icons[key].url}
        active={active === key ? "black" : "gray"}
      >
        {React.createElement(icons[key].icon, {
          active: active === key ? "black" : "gray",
        })}
        <BtnLabel content={icons[key].label} />
      </BottomNavBtn>
    ))}
  </Nav>
);
}
```

함수를 호출하는 컴포넌트에서 관리

= 컴파운드 컴포넌트 패턴과 유사

참고자료

합성 vs 상속 : https://ko.legacy.reactjs.org/docs/composition-vs-inheritance.html

아토믹 디자인 : https://yozm.wishket.com/magazine/detail/1531/

커스텀 훅 : https://ko.legacy.reactjs.org/docs/hooks-custom.html