

Renderowanie różnego obrazka w zależności od tego czy zmieni się jakaś wartość no w state

Ogólnie obrazki renderuje się przed dekalacją funkcji komponentu więc jeśli dany komponent ma renderować obrazek w zależności od tego jakie dostanie dane (które np. może zmieniać dynamicznie jakiś button) to wtedy trzeba przekazać podmieniony obrazek jako cały komponent

załóżmy że masz nawigację i ma ona dane itemów nawigacji zdefiniowane tak:

```
src > components > Navigation > components > NavItemsList > data > NavItemsListData.ts > NavItemsListData
1  import { NavItemType } from '../NavItem/data/NavItemType';
2  import defaultImage from 'assets/images/Nav/heroines/default.jpg';
3  import yui from 'assets/images/Nav/heroines/yui.jpg';
4  import asagi from 'assets/images/Nav/heroines/asagi.jpg';
5  import kaori from 'assets/images/Nav/heroines/kaori.jpg';
6
7  export const NavItemsListData: NavItemType[] = [
8    {
9      label: 'history',
10     sectionToScroll: 'history',
11   },
12   {
13     label: 'characters',
14     sectionToScroll: 'characters',
15     dropdown: [
16       {
17         label: 'yui',
18         to: 'yui',
19         description:
20           'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tristique scelerisque.',
21         image: yui,
22       },
23       {
24         label: 'asagi',
25         to: 'asagi',
26         description:
27           'Lorem Ipsum is simply dummy text of the printing and typesetting industry.',
28         image: asagi,
29       },
30       {
31         label: 'kaori',
32         to: 'kaori',
33         description:
34           'Aenean sagittis sapien erat, a consequat libero finibus id. Orci varius natoque penatibus et.',
35         image: kaori,
36       },
37     ],
38     dropdownDefaultData: {
39       image: defaultImage,
40       description: 'default description',
41     },
42   },
43 ]
```

Importujemy obrazki jakie mają się wyświetlać jako osobne komponenty i przekazujemy te komponenty jako klucz "image" w danych obiektu danego nav itemu

Następnie w komponencie który trzyma w sobie btn zmieniający obrazek oraz sam komponent wyświetlający ten obrazek tworzymy state oraz metody które zmieniają ten state (ten obrazek):

```
src > components > Navigation > components > NavItemDropdown > NavItemDropdown.tsx > NavItemDropdown

27 } => {
28   const [activeImage, setActiveImage] = useState(dropdownDefaultData?.image);
29   const [activeDescription, setActiveDescription] = useState(
30     dropdownDefaultData?.description,
31   );
32
33   const changeActiveImage = (image: string) => setActiveImage(image);
34   const resetActiveImage = useCallback(
35     () => setActiveImage(dropdownDefaultData?.image),
36     [dropdownDefaultData?.image],
37   );
38
39   const changeActiveDescription = (description: string) =>
40     setActiveDescription(description);
41   const resetActiveDescription = useCallback(
42     () => setActiveDescription(dropdownDefaultData?.description),
43     [dropdownDefaultData?.description],
44   );
45
46   const handleOnMouseEnter = useCallback((image: string, desc: string) => {
47     changeActiveImage(image);
48     changeActiveDescription(desc);
49   }, []);
50
51   const handleOnMouseLeave = useCallback(() => {
52     resetActiveImage();
53     resetActiveDescription();
54   }, [resetActiveImage, resetActiveDescription]);
55
56   const DropdownItems =
57     dropdownData !== undefined
58     ? dropdownData.map((item) => (
59       <NavItemDropdownItem
60         to={item.to}
61         key={item.label}
62         onClickFunction={toggleMobileNavVisibility}
63         onMouseEnter={() =>
64           handleOnMouseEnter(item.image, item.description)
65         }
66         onMouseLeave={handleOnMouseLeave}

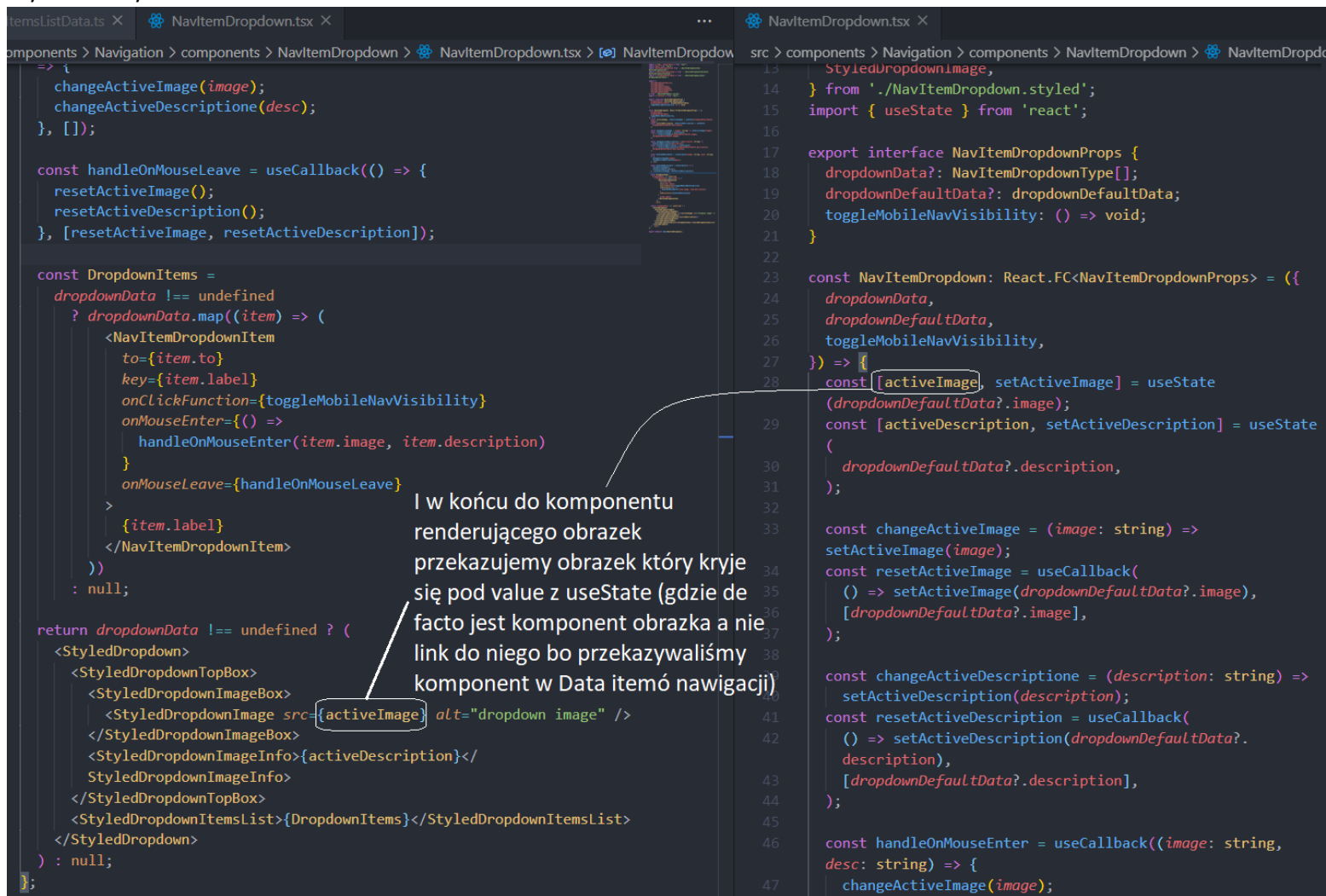
```

W komponencie który trzyma w sobie komponenty wyświetlające obrazek oraz btn który zmienia wyświetlany obrazek tworzy useState i przekazujemy domyślną wartość, domyślny link do obrazka etc (dropdownDefaultData też umieszczamy w danych itemów nawigacji, zobacz wcześniejszy screen)

Następnie piszemy metody dla btn które zmieniają wartość w state na nowy, aktualny obrazek (będą one de facto zmieniać nie tyle string czy nazwę ale komponent obrazka sam w sobie bo przekazywalismy komponent przecież)

Przekazujemy funkcje zmieniające defaultowy obrazek na inny do Btn czy innego elementu, który ma zmieniać obrazek

I na końcu przekazujemy do komponentu renderującego obrazek value z useState w której de facto siedzi komponent obrazka bo przecież przy deklarowaniu danych itemów nawigacji przekazywaliśmy komponenty obrazków a nie tylko same linki do nich czy ich nazwy:



```
NavItemDropdown.tsx
import { useState } from 'react';
import { NavItemDropdownType } from './NavItemDropdown.type';
import { StyledDropdownImage } from './NavItemDropdown.styled';
import { NavItemDropdownItem } from './NavItemDropdownItem';
import { StyledDropdownTopBox } from './NavItemDropdown.styled';
import { StyledDropdownImageBox } from './NavItemDropdown.styled';
import { StyledDropdownImageInfo } from './NavItemDropdown.styled';
import { StyledDropdownItemsList } from './NavItemDropdown.styled';

const handleOnMouseLeave = useCallback(() => {
  resetActiveImage();
  resetActiveDescription();
}, [resetActiveImage, resetActiveDescription]);

const DropdownItems =
  dropdownData !== undefined
    ? dropdownData.map((item) => (
        <NavItemDropdownItem
          to={item.to}
          key={item.label}
          onClickFunction={toggleMobileNavVisibility}
          onMouseEnter={() =>
            handleOnMouseEnter(item.image, item.description)
          }
          onMouseLeave={handleOnMouseLeave}
        >
          {item.label}
        </NavItemDropdownItem>
      ))
    : null;

return dropdownData !== undefined ? (
  <StyledDropdown>
    <StyledDropdownTopBox>
      <StyledDropdownImageBox>
        <StyledDropdownImage src={activeImage} alt="dropdown image" />
      </StyledDropdownImageBox>
      <StyledDropdownImageInfo>{activeDescription}</StyledDropdownImageInfo>
    </StyledDropdownTopBox>
    <StyledDropdownItemsList>{DropdownItems}</StyledDropdownItemsList>
  </StyledDropdown>
) : null;
};

NavItemDropdown.styled.tsx
import { styled } from 'styled-components';
import { NavItemDropdownType } from './NavItemDropdown.type';
import { NavItemDropdownItem } from './NavItemDropdownItem';
import { NavItemDropdownProps } from './NavItemDropdown.type';

export interface NavItemDropdownProps {
  dropdownData?: NavItemDropdownType[];
  dropdownDefaultData?: dropdownDefaultData;
  toggleMobileNavVisibility: () => void;
}

const NavItemDropdown: React.FC<NavItemDropdownProps> = ({
  dropdownData,
  dropdownDefaultData,
  toggleMobileNavVisibility,
}) => {
  const [activeImage, setActiveImage] = useState(
    (dropdownDefaultData?.image);
  );
  const [activeDescription, setActiveDescription] = useState(
    (dropdownDefaultData?.description);
  );

  const changeActiveImage = (image: string) =>
    setActiveImage(image);
  const resetActiveImage = useCallback(
    () => setActiveImage(dropdownDefaultData?.image),
    [dropdownDefaultData?.image],
  );

  const changeActiveDescription = (description: string) =>
    setActiveDescription(description);
  const resetActiveDescription = useCallback(
    () => setActiveDescription(dropdownDefaultData?.description),
    [dropdownDefaultData?.description],
  );

  const handleOnMouseEnter = useCallback((image: string, desc: string) => {
    changeActiveImage(image);
    changeActiveDescription(desc);
  }, [changeActiveImage, changeActiveDescription]);

  return (
    <NavItemDropdownItem
      to={dropdownDefaultData?.to}
      key={dropdownDefaultData?.label}
      onClickFunction={toggleMobileNavVisibility}
      onMouseEnter={handleOnMouseEnter}
      onMouseLeave={handleOnMouseLeave}
    >
      {dropdownDefaultData?.label}
    </NavItemDropdownItem>
  );
};
```

I w końcu do komponentu renderującego obrazek przekazujemy obrazek który kryje się pod value z useState (gdzie de facto jest komponent obrazka a nie link do niego bo przekazywaliśmy komponent w Data itemów nawigacji)

Dzięki temu że przyciski będą zmieniać wartość w state komponentu NavItemDropdown to jak zmieni się state to nowa wartość state (nowy komponent obrazka – czyli de facto nowy obrazek) zostanie przekazany do komponentu wyświetlającego obrazek i zmieni się sam obrazek