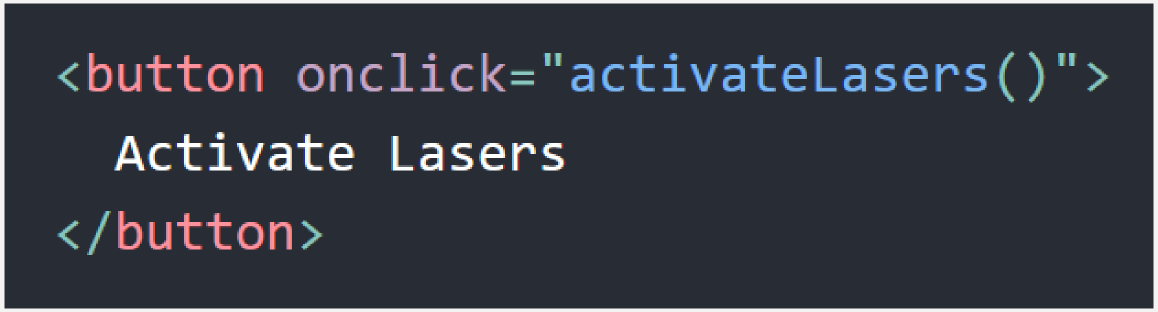
React

Handling Events

Introduction

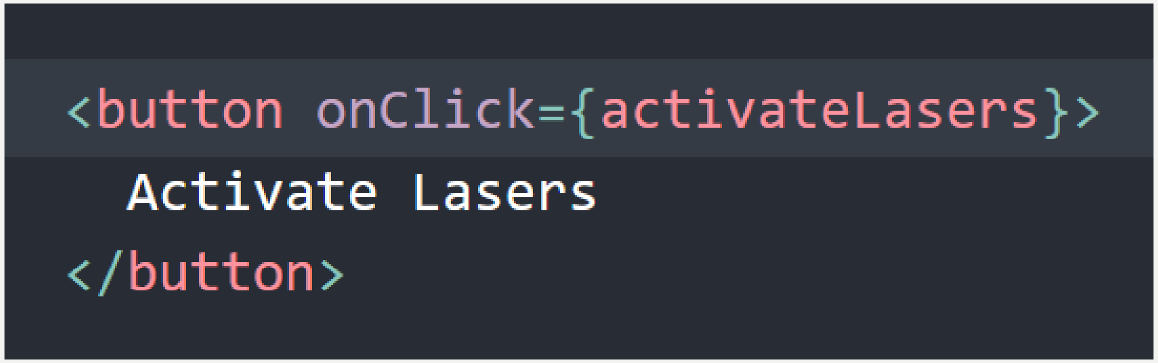
- Handling events with React elements is very similar to handling events on DOM elements. There are some syntax differences:
  - React events are named using camelCase, rather than lowercase.
  - With JSX you pass a function as the event handler, rather than a string

HTML



```html
<button onclick="activateLasers()">
  Activate Lasers
</button>
```

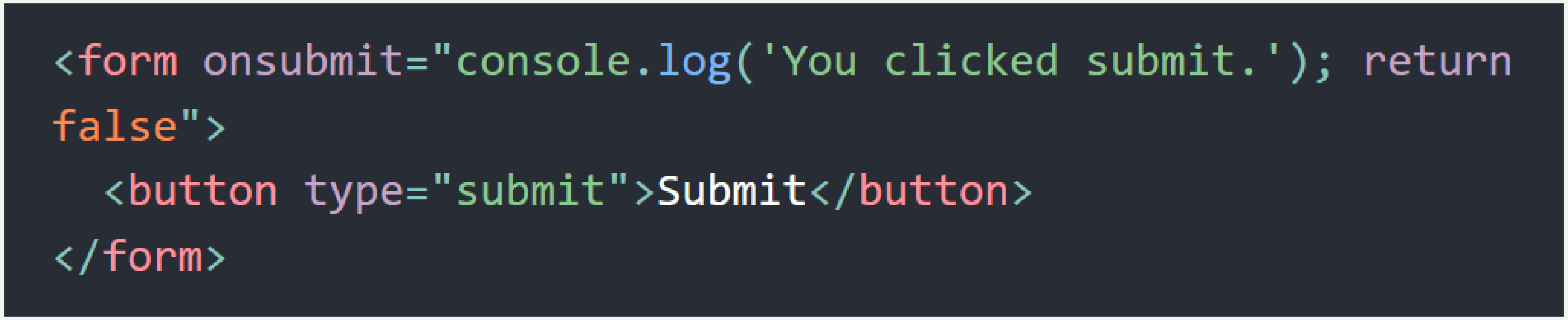
React



```jsx
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

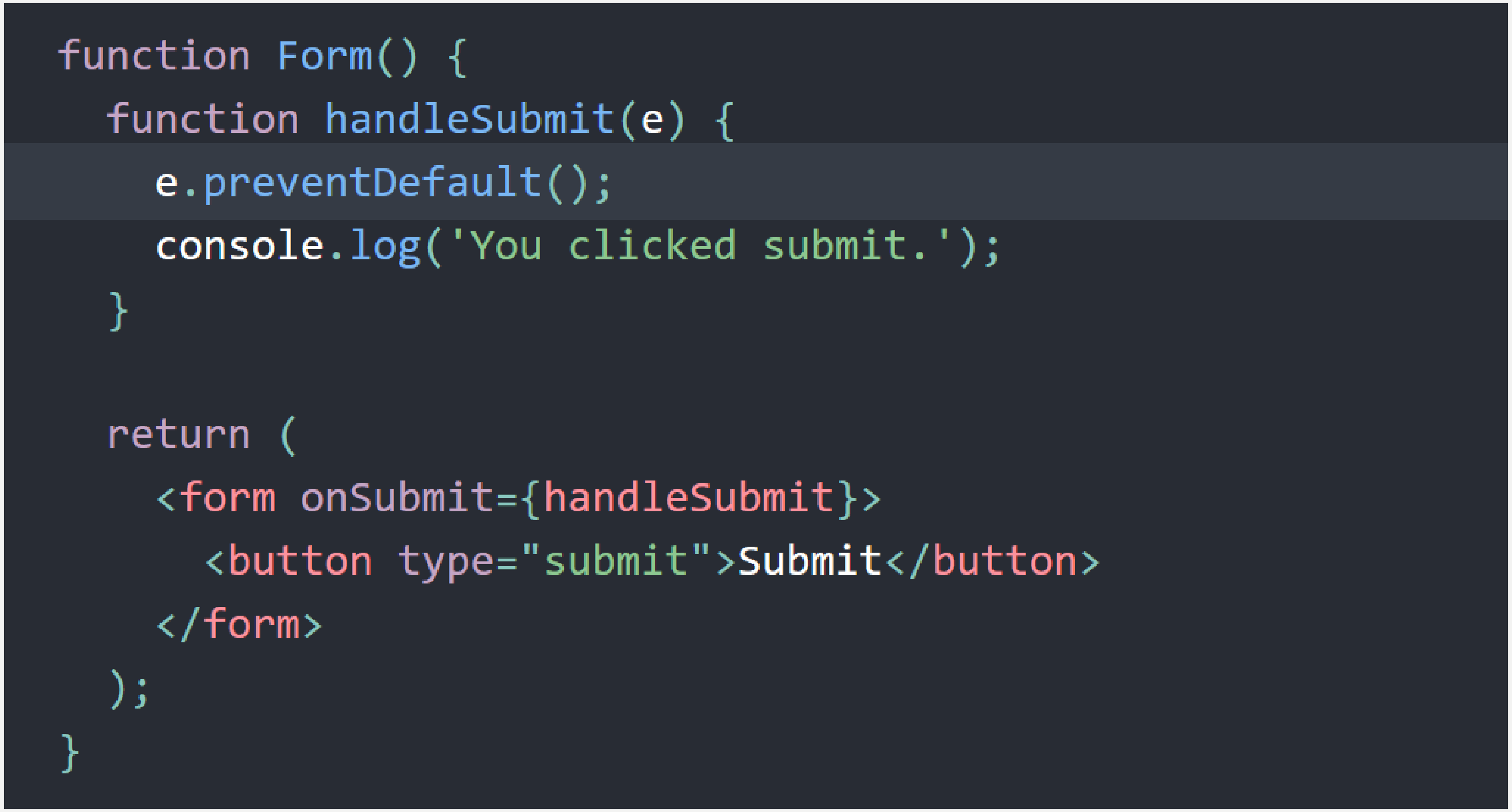
- Default behavior of element is slightly different in HTML and React.

HTML



```html
<form onsubmit="console.log('You clicked submit.'); return
false">
  <button type="submit">Submit</button>
</form>
```

React

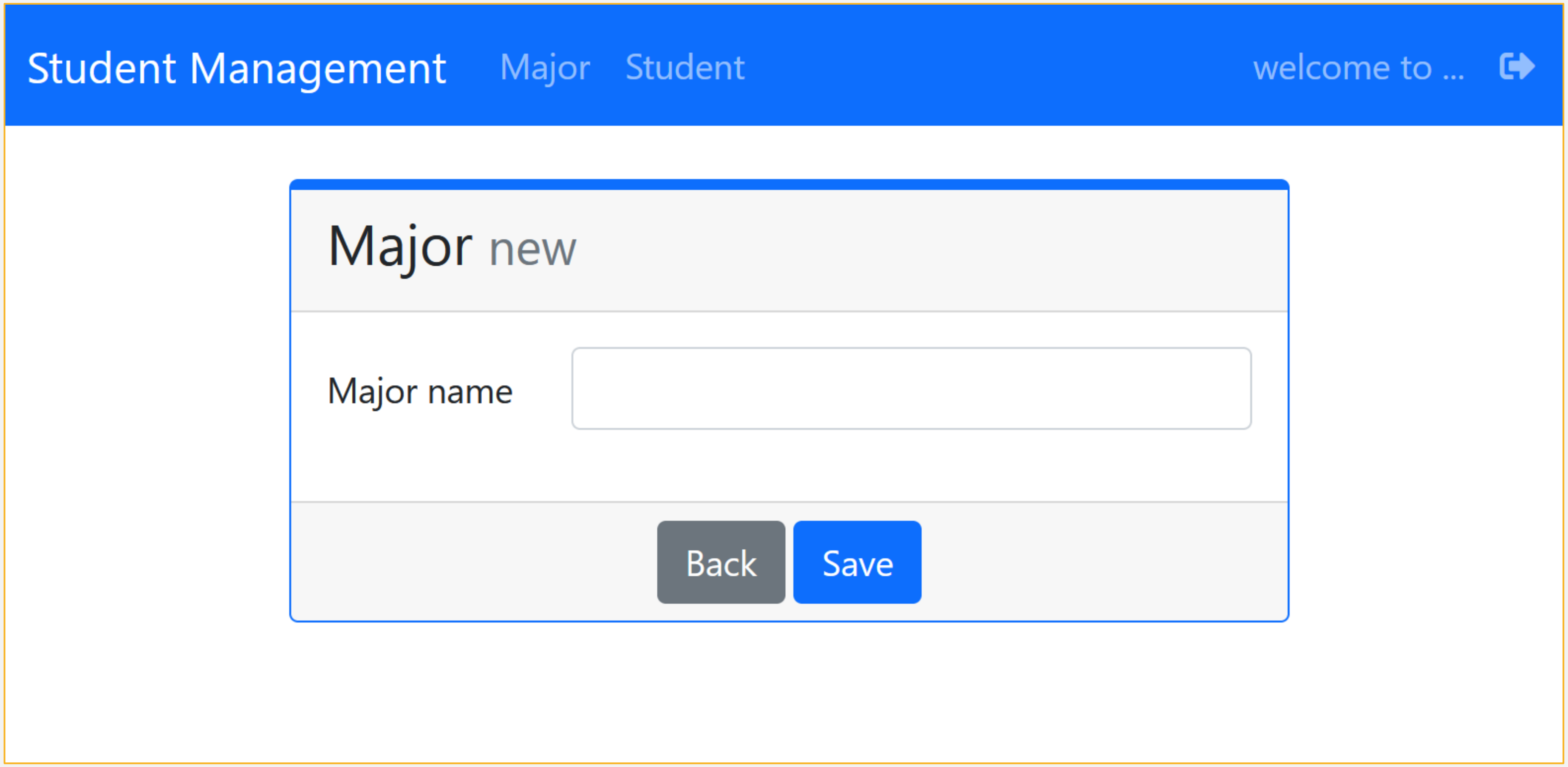


```jsx
function Form() {
  function handleSubmit(e) {
    e.preventDefault();
    console.log('You clicked submit.');
  }

  return (
    <form onSubmit={handleSubmit}>
      <button type="submit">Submit</button>
    </form>
  );
}
```

Event of Back Button on MajorEdit

```
import { useParams, useNavigate } from "react-router-dom";

const MajorEdit = () => {
  const { id } = useParams<{ id: string }>();

  const navigate = useNavigate();

  const handleBack = () => {
    navigate("/major");
  };
```

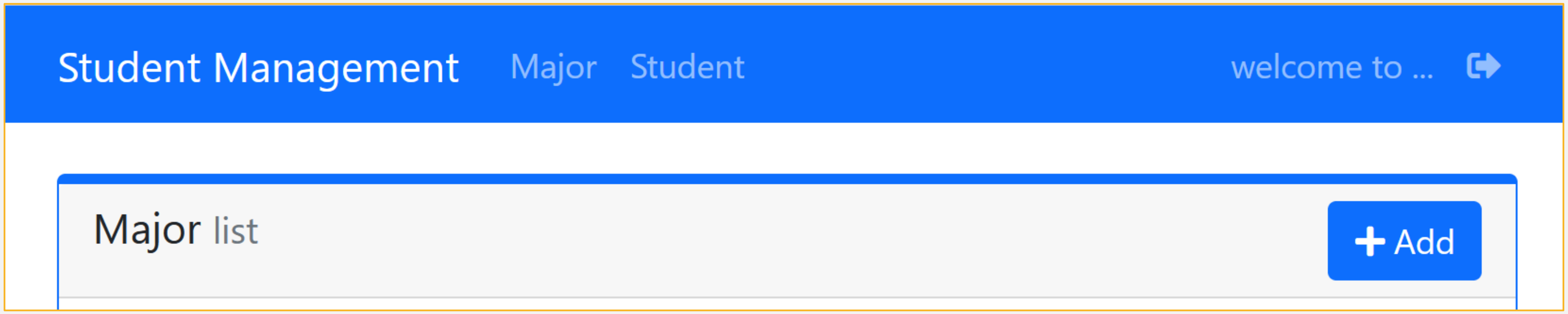
```
<button type="button" className="btn btn-secondary me-1"
  onClick={handleBack}>
  Back
</button>
```


Student Management
Major
Student
welcome to ...
Major new
Major name
Back
Save

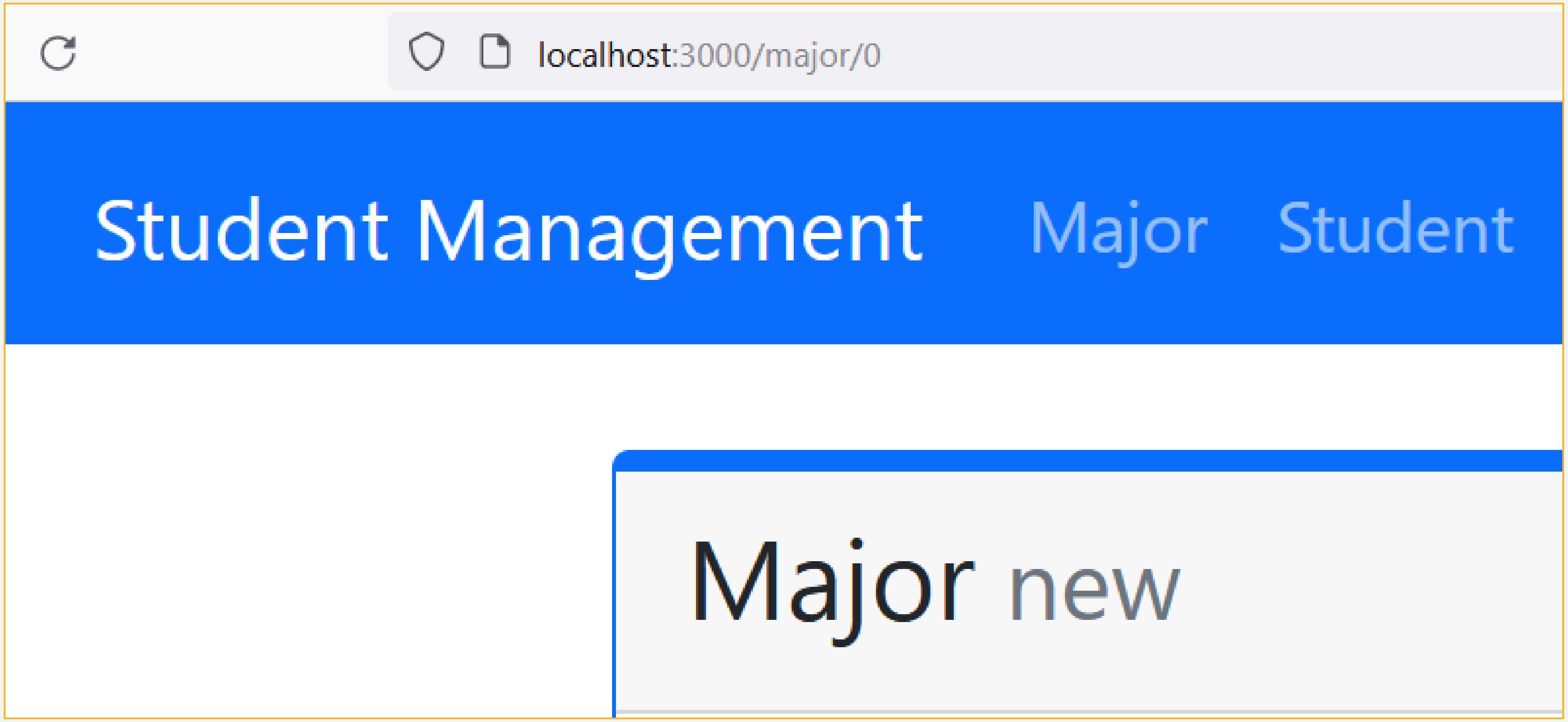
Event of Add Button on Major

```
const Major = () => {
  const navigate = useNavigate();

  const showEditPage = (e: any, id: number) => {
    if (e) e.preventDefault();
    navigate(`/major/${id}`);
  };

  return (
```

```
<button type="button" className="btn btn-primary"
  onClick={() => showEditPage(null, 0)}>
  <i className="bi-plus-lg" /> Add
</button>
```

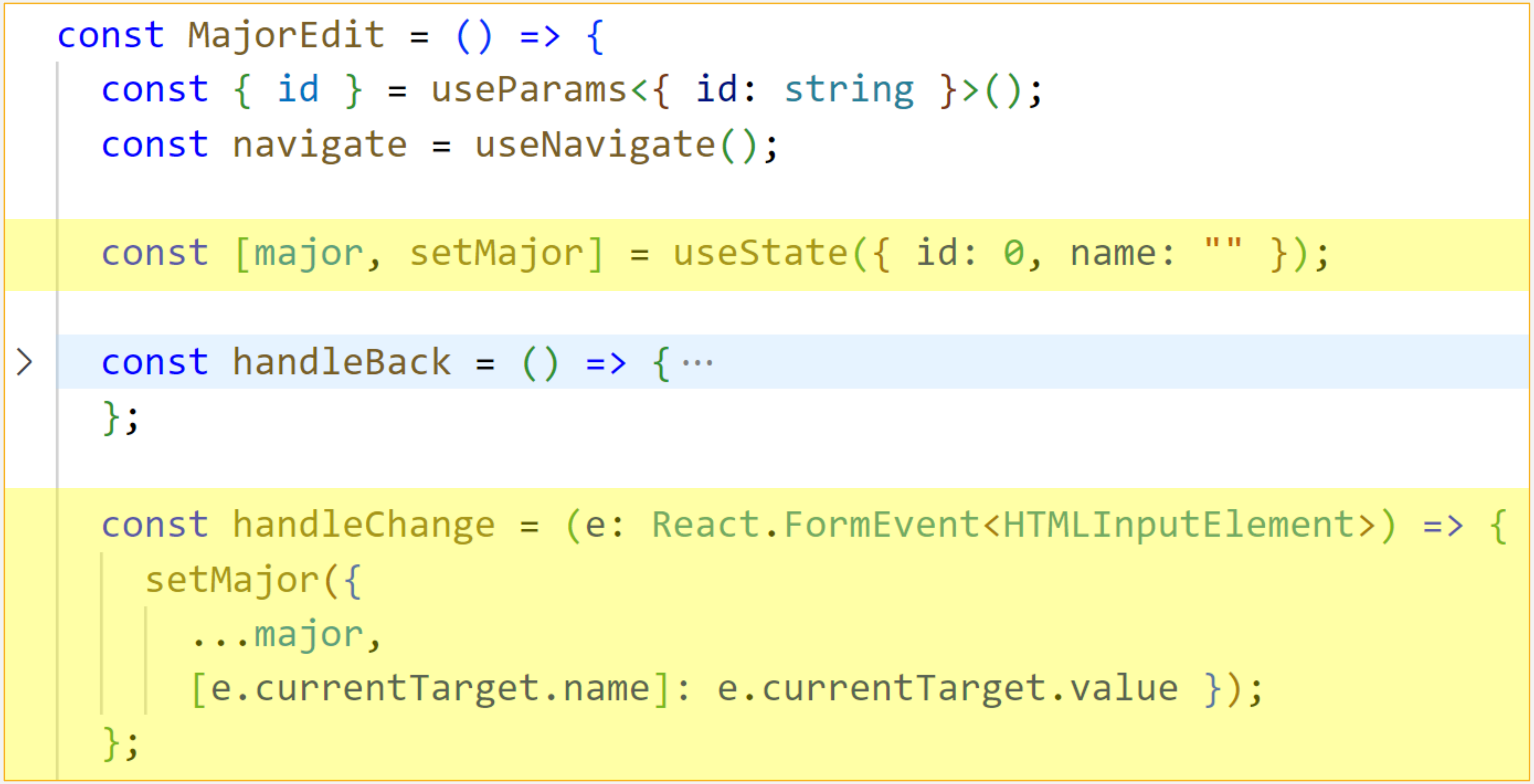


Major list

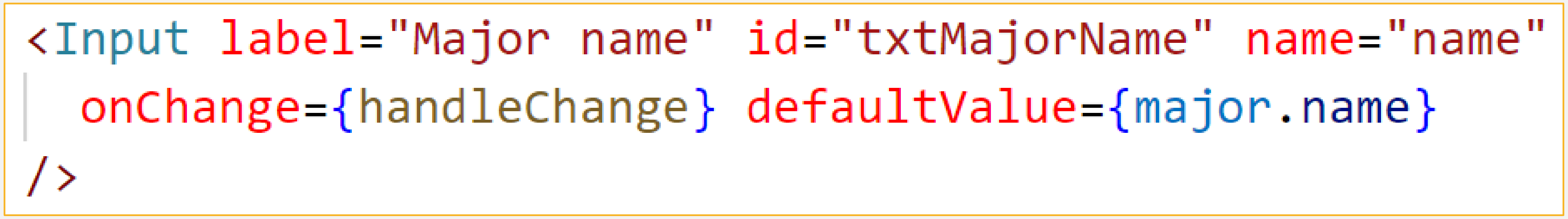


Major new

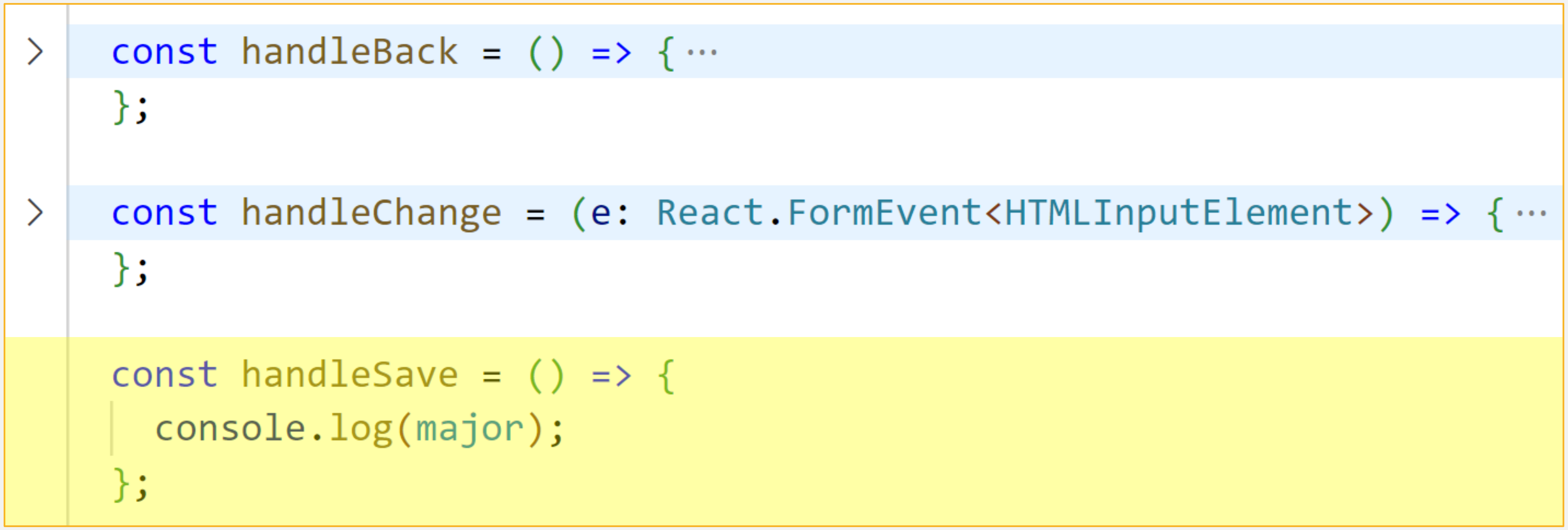
Handle onChange Event of Input



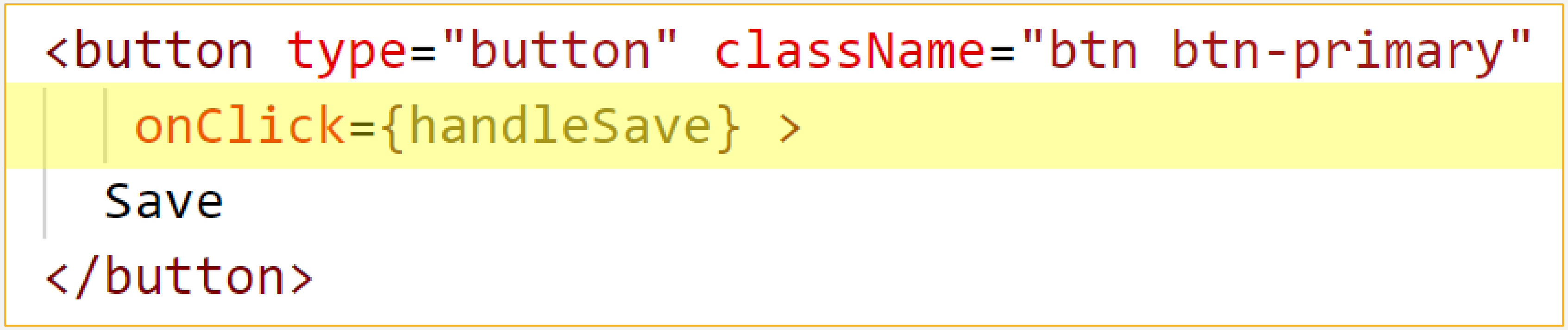
```tsx
const MajorEdit = () => {
  const { id } = useParams<{ id: string }>();
  const navigate = useNavigate();

  const [major, setMajor] = useState({ id: 0, name: "" });

  const handleBack = () => {···
  };

  const handleChange = (e: React.FormEvent<HTMLInputElement>) => {
    setMajor({
      ...major,
      [e.currentTarget.name]: e.currentTarget.value });
  };
```



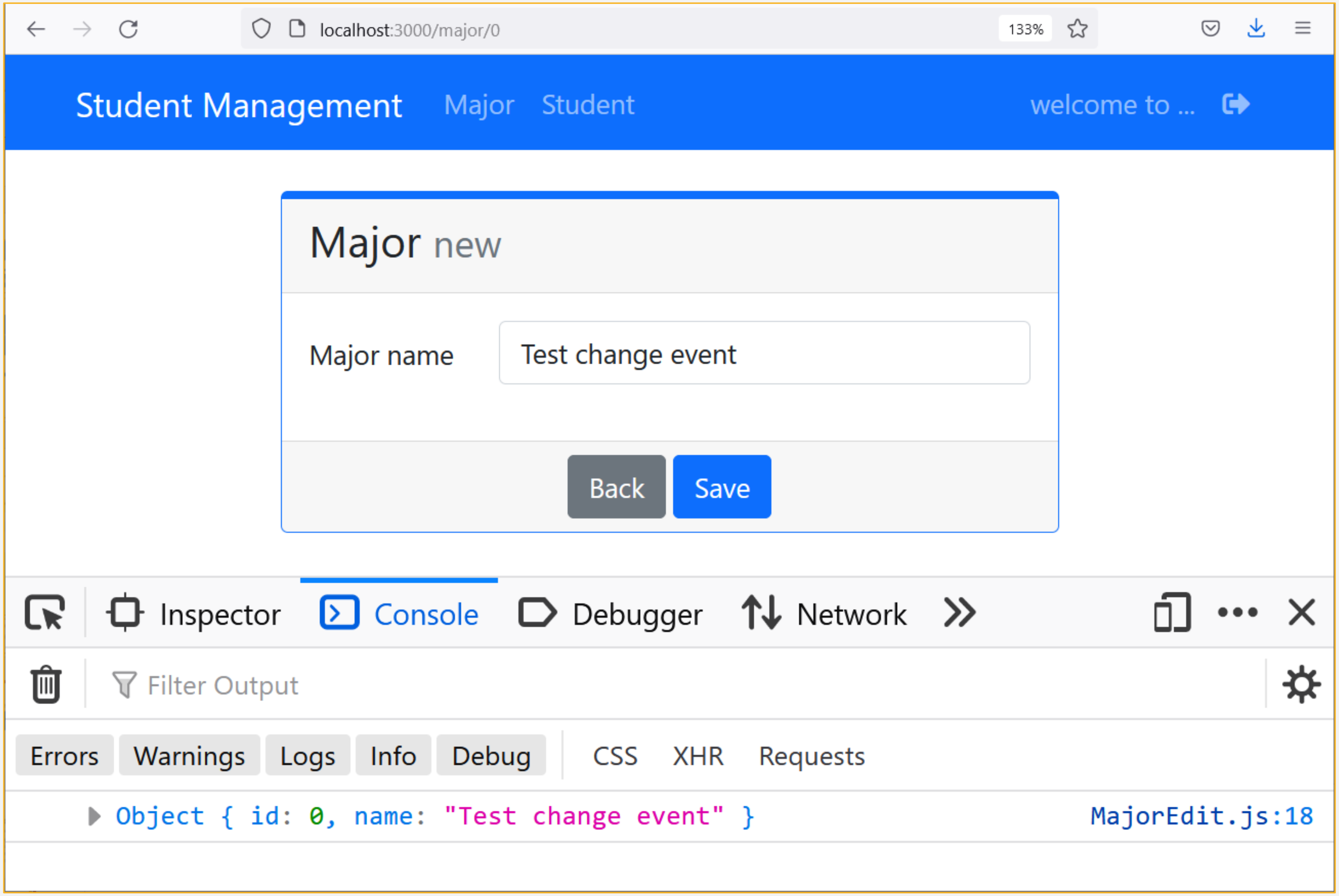
```tsx
<Input label="Major name" id="txtMajorName" name="name"
  onChange={handleChange} defaultValue={major.name}
/>
```



```
const handleBack = () => {···
};

const handleChange = (e: React.FormEvent<HTMLInputElement>) => {···
};

const handleSave = () => {
  console.log(major);
};
```



```
<button type="button" className="btn btn-primary"
    onClick={handleSave} >
  Save
</button>
```



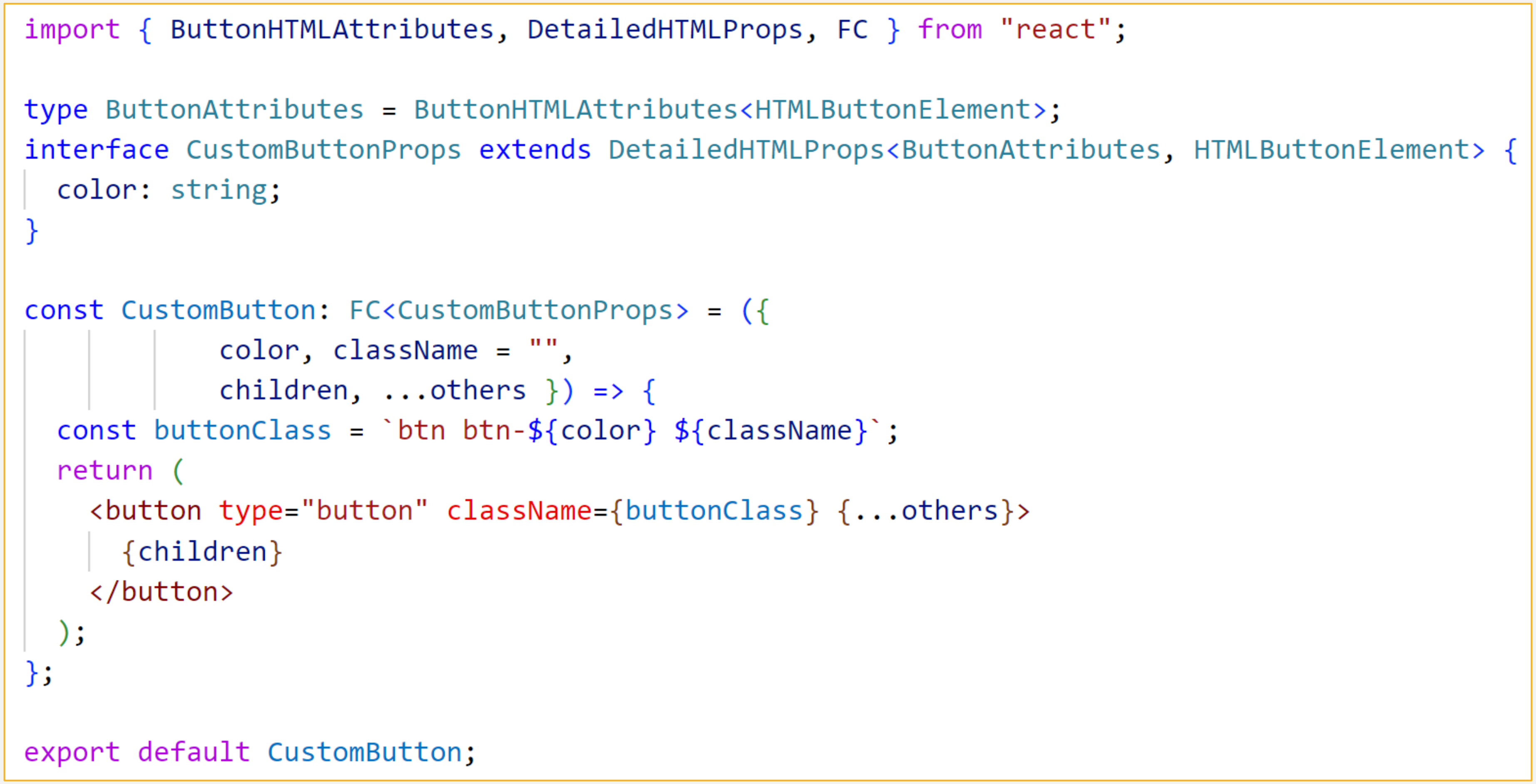
Major new

Major name: Test change event

```
Object { id: 0, name: "Test change event" }                MajorEdit.js:18
```

Add CustomButton Component

- In “components” folder, create a new file named “CustomButton.tsx”



```tsx
import { ButtonHTMLAttributes, DetailedHTMLProps, FC } from "react";

type ButtonAttributes = ButtonHTMLAttributes<HTMLButtonElement>;
interface CustomButtonProps extends DetailedHTMLProps<ButtonAttributes, HTMLButtonElement> {
  color: string;
}

const CustomButton: FC<CustomButtonProps> = ({
            color, className = "",
            children, ...others }) => {
  const buttonClass = `btn btn-${color} ${className}`;
  return (
    <button type="button" className={buttonClass} {...others}>
      {children}
    </button>
  );
};

export default CustomButton;
```

Modify Back Button on MajorEdit

```
<button type="button" className="btn btn-secondary me-1"
  onClick={handleBack}>
  Back
</button>
```

↓

```
<CustomButton color="secondary" className="me-1"
  onClick={handleBack}>
  Back
</CustomButton>
```

THE END