

🚩 Current Skill DOM Events

## DOM Events

Let's take a trip to the past and remember DOM events:

- Events on a website are always interacting with the browser. For example a mouse click, file loading on the browser or swiping the image left or right.

Handling events with React elements is very similar to handling events on DOM elements. There are some syntactic differences:

- React events are named using **camelCase**, rather than lowercase.
- With JSX you pass a function name as the event handler rather than a string.

We can find many events such as **onMouseMove**, **onChange**, **onClick** and many more.

## Events/Functions

Since we are writing JavaScript, React events are written in camelCase syntax:

**onClick** instead of **onclick**.



React event handlers are written inside curly braces:

**onClick={handleClick}**

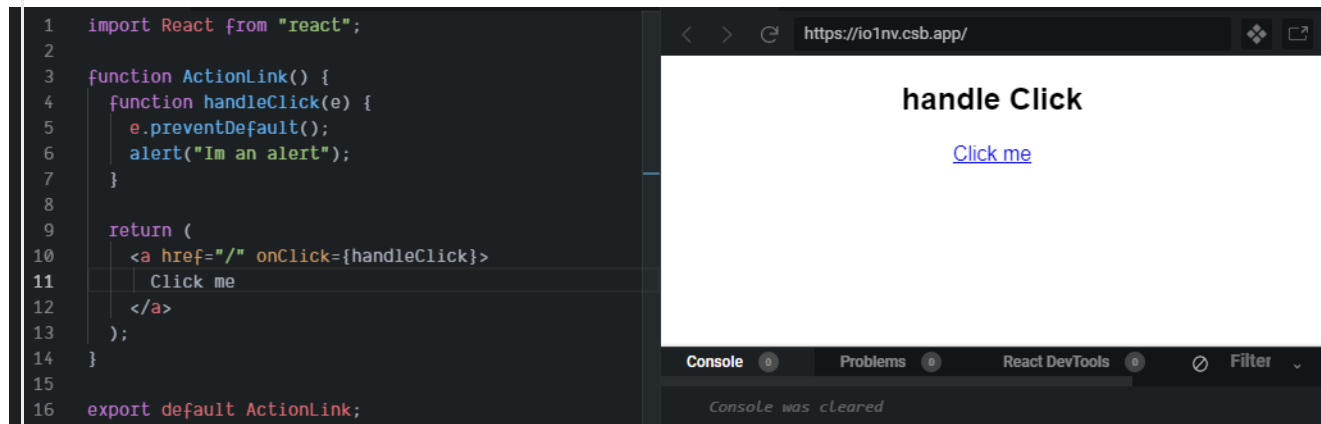
```
const ActionLink = () => {  
  const handleClick = e => {  
    e.preventDefault();  
    console.log("The link was clicked.");  
  };  
  
  return (  
    <a href="/" onClick={handleClick}>  
      Click me  
    </a>  
  );  
};
```



# Events/ Functions

In the example above, the **onClick** attribute is our event handler, and it is added to the target element in order to assign the function to be executed when that element is clicked. The **onClick** attribute is set to the **handleClick** function, which sends an alert message.

In simpler terms, this means that whenever the button is clicked on, the **handleClick** function is called which displays the alert box.



The screenshot shows a code editor on the left and a browser window on the right. The code editor contains the following code:

```
1 import React from "react";
2
3 function ActionLink() {
4   function handleClick(e) {
5     e.preventDefault();
6     alert("Im an alert");
7   }
8
9   return (
10    <a href="/" onClick={handleClick}>
11      Click me
12    </a>
13  );
14 }
15
16 export default ActionLink;
```

The browser window shows the URL <https://io1nv.csb.app/> and the text "handle Click" above a blue link that says "Click me". The browser's developer tools are open, showing the "Console" tab with the message "Console was cleared".

## Events/ Functions props

There are many situations when you're writing with React and you'll want to pass a function as a prop.

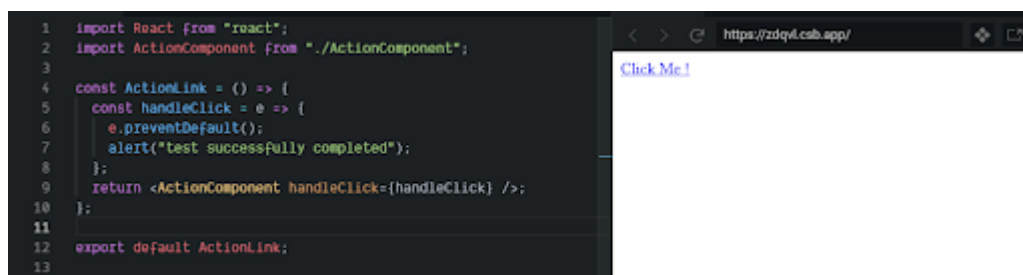
Usually it's to pass down to a child component so that the child can notify the parent of an event.

In this example, we have passed **handleClick** function from **ActionLink** component as props to



**ActionComponent**.

We wait from **ActionComponent** to trigger an action , in this case it's an alert message.



The screenshot shows a code editor on the left and a browser window on the right. The code editor contains the following code:

```
1 import React from "react";
2 import ActionComponent from "./ActionComponent";
3
4 const ActionLink = () => {
5   const handleClick = e => {
6     e.preventDefault();
7     alert("test successfully completed");
8   };
9   return <ActionComponent handleClick={handleClick} />;
10 };
11
12 export default ActionLink;
```

The browser window shows the URL <https://zdzql.csb.app/> and a blue link that says "Click Me!".



## Events/ Functions props test



```

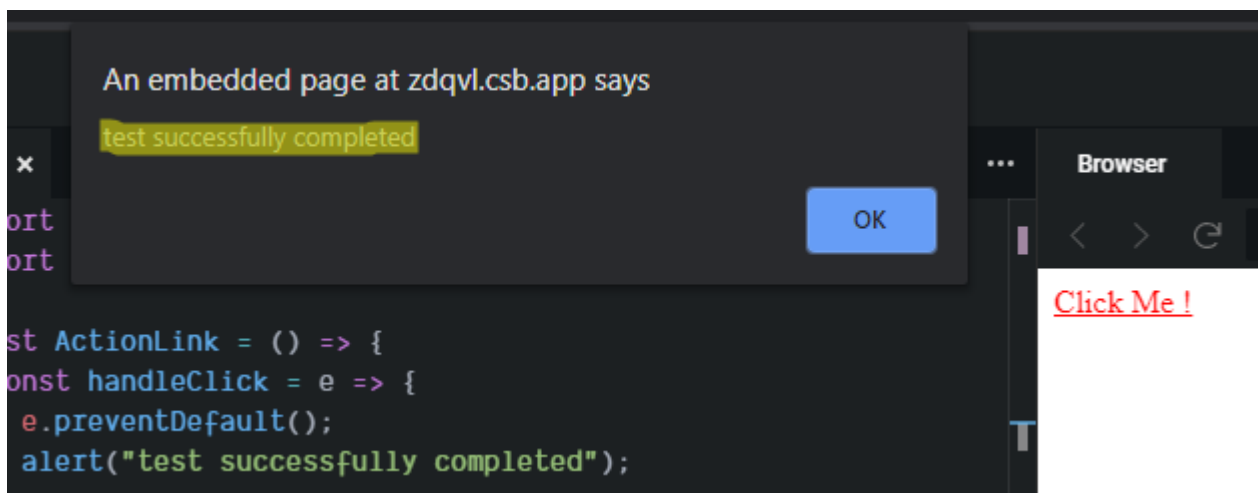
1 import React from "react";
2 import "./styles.css";
3
4 const ActionComponent = ({ handleClick }) => {
5   return (
6     <a href="/" onClick={e => handleClick(e)}>
7       Click Me !
8     </a>
9   );
10 };
11
12 export default ActionComponent;
13

```

Click Me !

In ActionComponent, what we expect when we click the Click Me! Link is to send an alert message from handleClick which is defined inside the parent component ActionLink.

## Events/ Functions props test



As we had expected, the alert gets triggered and the test is successfully passed!

## Events/ Conclusion

Handling events is a very important aspect to learn.

Since applications are based on user interactions, handling gives us the power to control any interaction that happens between the user and our application.

React puts forward all the DOM events handlers that exist only with a little difference in the way we call them (the use of the camel case naming.)

< Previous

next >