REACT HOOKS -USESTATE

We will learn how to make our apps more interactive.

To do that we will learn the concept of STATE.

State is one of the fundamental concept in react

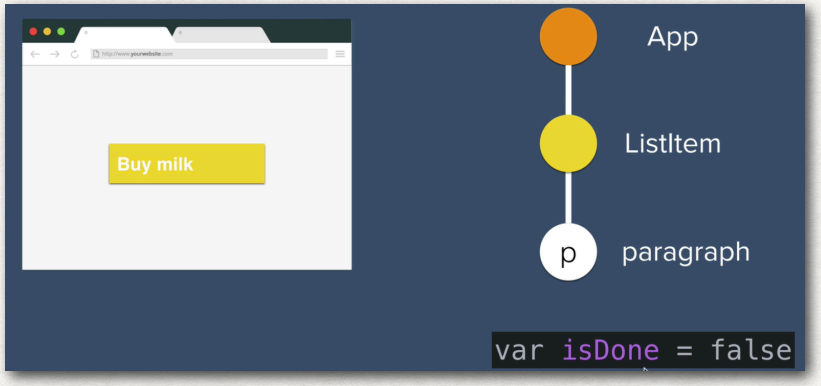
UI is actually a function of the state

When the state of a UI changes, we see different behavior of the app.

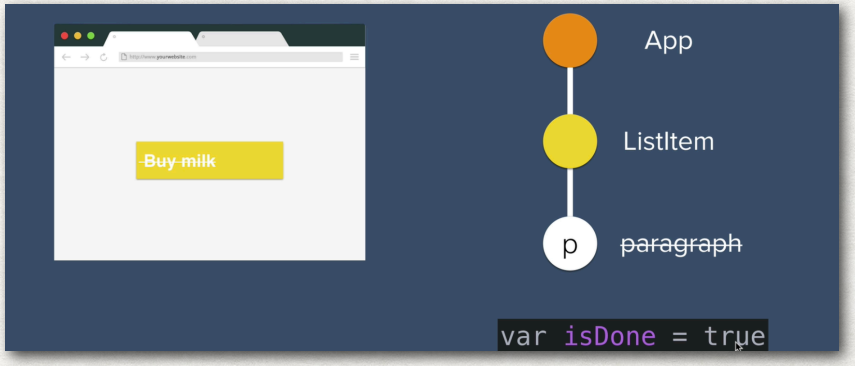
One of the way to change the state is, kept track of the condition of a variable

We will change the sate of the element depending on the isDone variables state.

In this practice we as we click on a button, we will increate the value on the number



USER CLICKS THEN CONDITION CHANGES



REACT HOOKS -USESTATE

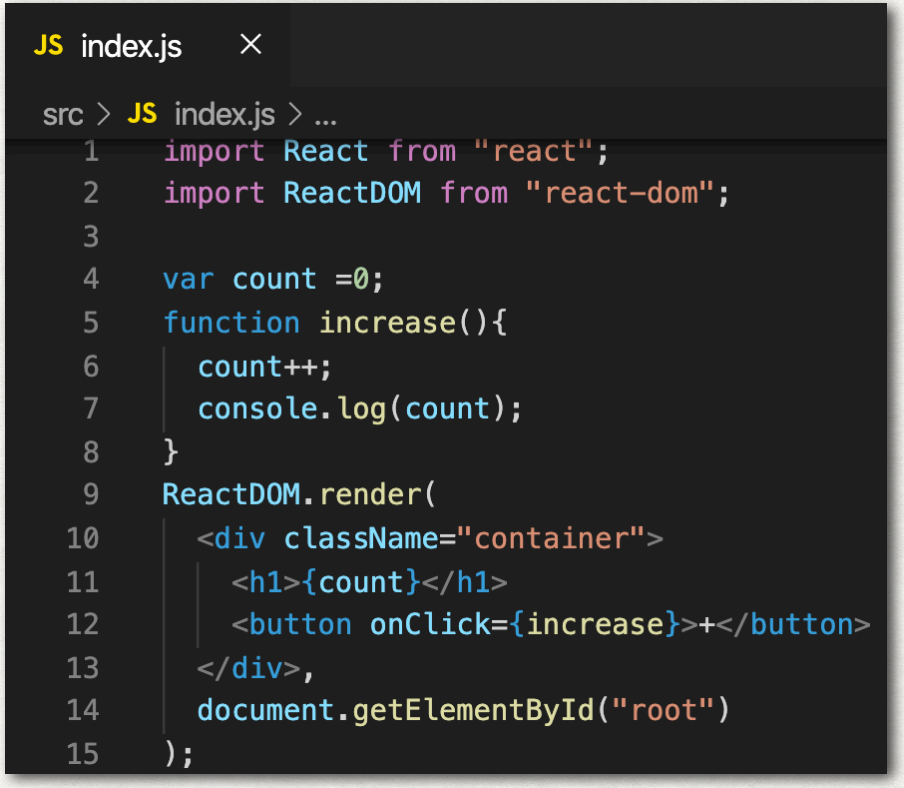
Import ten-usestate-hook,npm install react-scripts start,npm install,npm start

Now I got everything inside my index.js and rending a single div with h1 and button

Our goal is the button should trigger and update the number. We want count change so create a count variable

We want to trigger + button. In HTML, button on-click attribute to trigger click action BUT in JS it is onClick.

So use **onClick** attribute for the button and assign a function called increase. Increate function will increate the value of count by 1 when we clock on the + button. onClick will trigger an action and update the element



REACT HOOKS -USESTATE

In App.jsx.js,

1. Create App component and return the elements.

2.Create a variable count, initialize and use it in the h1 as js object

3.Use onClick attribute in the button and set it to a function called increase

4.Create increase function to increment the count

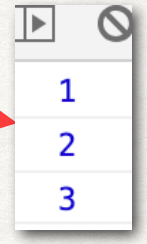
In index.js, call the App component

Now as we click the button, we should see the count is increasing on the console, But it is not rendering on the UI.

index.js | App.jsx

components > App.jsx > ...

```
1   import React from "react";
2
3   let count=0;
4   function increase(){
5     count++;
6     console.log(count);
7   }
8
9   function App() {
0     return  <div className="container">
1     <h1>{count}</h1>
2     <button onClick={increase}>+</button>
3   </div>;
4   }
5
6   export default App;
```



index.js | App.jsx

src > index.js

```
1   import React from "react";
2   import ReactDOM from "react-dom";
3   import App from './components/App'
4
5   ReactDOM.render(
6     <App />,
7      document.getElementById("root")
8   );
```

REACH HOOKS-USESTATE

1. In App.jsx, in the App function, Instead of var count = 0; we use useState hook

```
const state = useState(123);// 123 is starting state
console.log(state[0]);//return 123 on the console
```

2. import React, { useState } from "react";

3. To display on the <h1 element: <h1>{state[0]}</h1>. This me
   Whenever useState updated the ui automatically be updated

4. This is has coded. We will Destructure arrays and objects

5. To destrucre useState, let's map useState initial value to an array value:Below will render the initial value:

```
const [count] = useState(123);//assign 123 for count initial
<h1>{count}</h1>//Render count in the h1.UI SHOULD RENDER SAME
```

6. HOW DO WE INCREASE COUNT AS WE CLICK ON COUNT BUTTON???

7. useState(initialValue,function)

```
const [count,setCount] = useState(123);
function increase(){
  setCount(count+1)//increate the initial value one by one
}
<button onClick ={increase}>+</button>
```

6.

dex.js | JS App.jsx

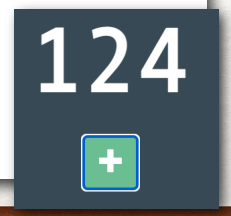
components > JS App.jsx > ...

```
import React, { useState } from "react";

// let count=0;
// function increase(){
//   count++;
//   console.log(count);
// }

function App() {
  const [count,setCount] = useState(123);
  function increase(){
    setCount(count+1)
  }
  return  <div className="container">
  <h1>{count}</h1>
  <button onClick={increase}>+</button>
</div>;
}

export default App;
```



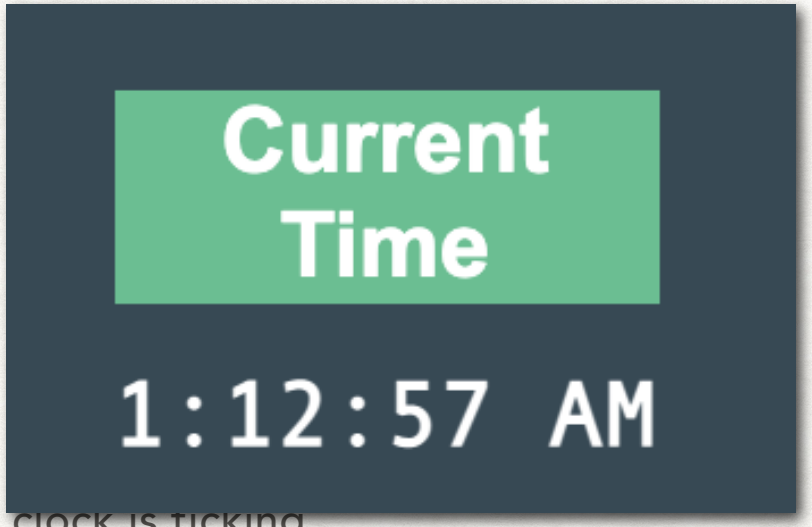
REACH HOOKS-USESTATE PRACTICE

YOUR TURN

Add a - sign and decrease the function

REACH HOOKS-USESTATE PRACTICE

RENDER THE CURRENT local TIME ON THE UI



Steps :

1. Create a button with text: Current Time
2. Style the button to match the style
3. Create an h1 and to get the current time.
4. Make sure to use create needed objects and the clock is ticking

NOTE : USE setInterval(function) TO SET RENDER THE UPDATE TIMES

REACT HOOKS-USESTATE PRACTICE

1. Create a button with text: Current Time

1. Style the button to match the style

```
<button style = {
  {fontSize:"30px",marginTop:"20%",width:"60%"}
  }>Current Time</button>
```

2. Create an h1 and to get the current time.

3.

```
const now = new Date().toLocaleTimeString();
console.log(now);
<h1 style = {{fontSize:"30px"}}>{now}</h1>
```

```
//OR
const [time,setTime] = useState(new Date().toLocaleTimeString());
console.log(time);
```

```
//Creating a setInterval function that will update setTime every 1000 milisecond
//setInterval(setTime, 1000);//=> State Hook functions doesnt work in setInterval.
//That is why, we need a new function
setInterval(updateTime, 1000);
function updateTime(){
  const newTime = new Date().toLocaleTimeString();
  setTime(newTime);
}
<h1 style = {{fontSize:"30px"}}>{time}</h1>
```

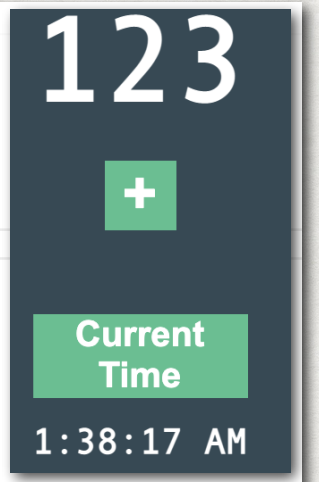
components > JS App.jsx > ...

```
import React, { useState } from "react";

// let count=0;
// function increase(){
//   count++;
//   console.log(count);
// }
function App() {
  const [count,setCount] = useState(123);
  function increase(){
    setCount(count+1)
  }
  // const now = new Date().toLocaleTimeString();
  // console.log(now);
  // const [time,setTime] = useState(now);
  //OR
  const [time,setTime] = useState(new Date().toLocaleTimeString());
  console.log(time);
  //Creating a setInterval function that will update setTime every 1000 milisecond
  //setInterval(setTime, 1000);//=> State Hook functions doesnt work in setInterval.
  //That is why, we need a new function
  setInterval(updateTime, 1000);
  function updateTime(){
    const newTime = new Date().toLocaleTimeString();
    setTime(newTime);}

  return  <div className="container">
  <h1>{count}</h1>
  <button onClick={increase}>+</button>
  <br />
  <button style = {
    {fontSize:"30px",marginTop:"20%",width:"60%"}
    }>Current Time</button>
  <h1 style = {{fontSize:"30px"}}>{time}</h1>
</div>;}

export default App;
```



USESTATE HOOK PRACTICE-HOMEWORK

Import eleven-usestate-hook-practice

npm install react-scripts start, npm install, npm start

Or goal to to render the clock on the page and see the clock is running

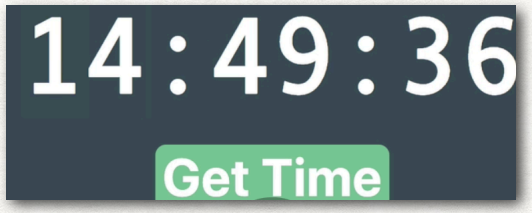
We break down the steps to learn the steps

TASK:

Part 1 : when we click on Get Time we should see the latest updated time

Part 2: we should always the updated time like a clock ticking

NOTE : USE setInterval(function) TO SET RENDER THE UPDATE TIMES




```
//Challenge:
//1. Given that you can get the current time
using:
let time = new Date().toLocaleTimeString();
console.log(time);
//Show the latest time in the <h1> when the Get
Time button
//is pressed.

//2. Given that you can get code to be called
every second
//using the setInterval method.
//Can you get the time in your <h1> to update
every second?

//e.g. uncomment the code below to see Hey
printed every second.
// function sayHi() {
//   console.log("Hey");
// }
// setInterval(sayHi, 1000);
```

USESTATE PRACTICE HOMEWORK

1. In App.jsx, creates constt now to get het current time

```
const now = new Date().toLocaleTimeString();
console.log(now);
```

1. In App.jsx, create a new function to destructure useState array that will hold the initial value of the time and a function that will update the time

```
  const [time, setTime]=useState(now);//starting value now. time=now
  console.log({time});
```

3. Use time, which has the update time in the <h1> to render it:

```
<h1>{time}</h1>
```

4. We will update the time when button clicked Use onClick and creat a function to update:

```
<button onClick={updateTime}>Get Time</button>
```

5. Create the updateTime function:

```
  function updateTime(){
    const newTime = new Date().toLocaleTimeString();
    setTime(newTime);
  }
```

5. Now whenever I click on the on Get Time button, I will get he new time. PART1 IS DONE. TEST
6. We will use setInterval function to display int he first line of the App function

```
  setInterval(updateTime, 1000);//MEANS update updateTime function every 1000 millisecond
```

7. It will create a new constant set to the current time and update our time variable that is used in h1 every single second! So we will see dynamic refreshing time

```
import React, { useState } from "react";

function App() {

  setInterval(updateTime, 1000);

  const now = new Date().toLocaleTimeString();
  // a(now);

  const [time, setTime] = useState(now);
  // console.log({time});

  function updateTime(){
    const newTime = new Date().toLocaleTimeString();
    setTime(newTime);
  }

  return (
    <div className="container">
      {/* <h1>TIME</h1> */}
      <h1>{time}</h1>
      <button onClick={updateTime}>Get Time</button>
    </div>
  );
}

export default App;
```

EVENT HANDLING IN REACT

Import twelve-event-handling-in-react,

npm install react-scripts start, npm install, npm start

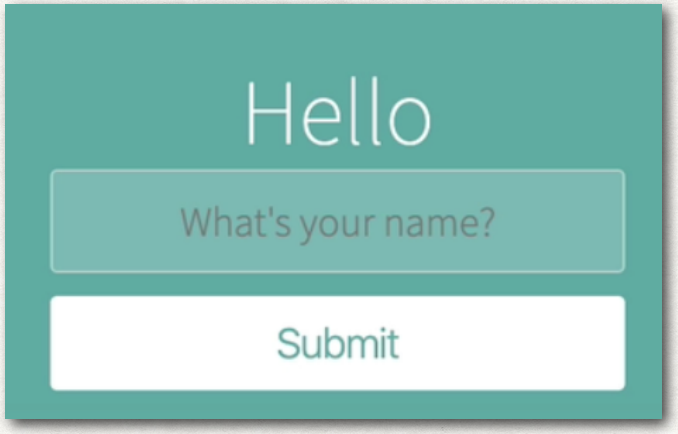
We will learn handling event in react

When the mouse is in the submit button, it turns black, when we move away, it turns white.

We will do it by handling this mouse over event.

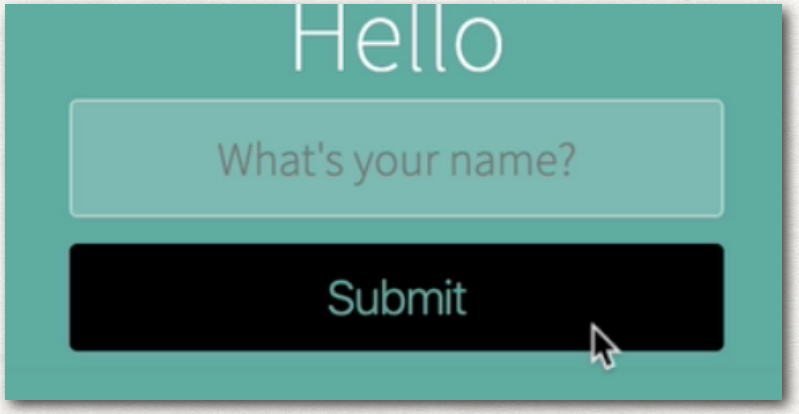
How do you detect when use interact with the app or a component?

How to use react render different things after these user interaction?



We ll use conditional rendering, inline styling, mouse events, managing state in this lesson.

CHECK OUT:



EVENT HANDLING IN REACT

In App,jsx, we just have an h1, input type text, placeholder that is asking for the name to type in, and submit button

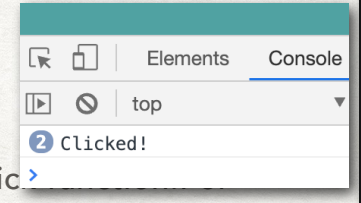
1. Let's start with onClick on the button:Add onClick on the button

```
<button onClick={handleClick}>Submit</button>
```

2.In App function, create the handleClick function to tell the handleClick what to do:

```
function handleClick(){console.log("Clicked!") }
```

Test button. When you click the button, you should now see the log:



3. Now that we can detect button click, we can write code in handleClic
example we can change the heading when button is clicked.

```
 <h1> {headingText} </h1>
```

4. Use useState hook, because we have initial value, which is Hello, and a new value, which is "Submitted"

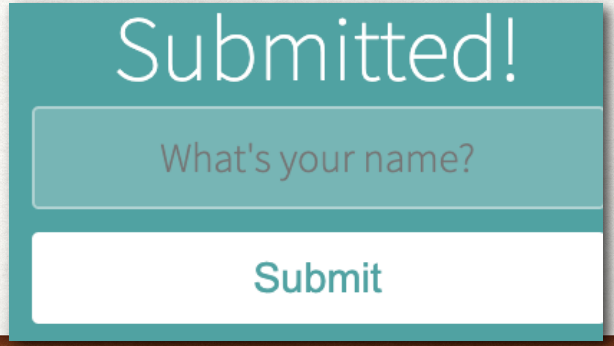
```
const [headingText,setHeadingText]=useState("Hello");
```

5. Add setHeadingText("Submitted" function in the handleClick() function to change the value after click

```
function handleClick(){setHeadingText("Submitted!");
  // To render the text conditionally we can add ternary in the handle click function:
  // headingText === "Hello"?setHeadingText('Clicked'):setHeadingText('Hello')}
```

App.jsx

```
1  import React,{useState} from "react";
2
3  function App() {
4    const [headingText,setHeadingText]=useState("Hello");
5    function handleClick(){setHeadingText("Submitted!");}
6    return (
7      <div className="container">
8        <h1> {headingText} </h1>
9        <input type="text" placeholder="What's your name?" />
10       <button onClick={handleClick}>Submit</button>
11     </div>
12   );
13 }
14
15 export default App;
```



EVENT HANDLING IN REACT

has the list of events

will detect when we hover over the element, that will detect when we move away from the element

1.Remember we can change the appearance of html element using style attribute.style = {{backgroundColor:"black"}}

```
<button style = {{backgroundColor:"black"}}
```

This'll make the button always black. We need to render background color black or red based on a condition, mouse is over or mouse is out, etc.

2. First use onMouseOver to turn the color to black when we hover over the button. Add button:

```
onMouseOver = {handleMouseOver}
```

3. And create the function to tell what this handleMouseOver will do :log something to test

```
function handleMouseOver(){console.log("Mouse Over");}
```

4. Next we will change the color of the button only when we handleMouseOver Function is called.

```
//isMouseOver initial value = false.
//setMouseOver will update the initial value
  const [isMouseOver,setMouseOver]=useState(false);
```

5. Use setMouseOver function inside the handleMouseOver function to set the new value true:

```
function handleMouseOver(){setMouseOver(true);}
```

6. We will change the button background color black when isMouseOver= true. Otherwise change it to white. Use Ternary.

```
<button style = {{backgroundColor:isMouseOver?"black":"white"}}
```

7. When we move mouse over button is still black. So Add on onMouseOut={handleMouseOut} to switch it back:

```
onMouseOut={handleMouseOut}
```

8. Create the handleMouseOut function to set the mouseOver false when mouse is moved out of the button

```
function handleMouseOut(){setMouseOver(false);}
```

App.jsx

src > components > App.jsx > App > handleMouseOut

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

function App() {
//create the state for headingText, with starting = Hello
//setHeadingText will be used to change the text
const [headingText,setHeadingText]=useState("Hello");
//starting value is false cause when we load the site we are not jovered
const [isMouseOver,setMouseOver]=useState(false);

function handleClick(){
  // console.log("Clicked!")
  //change the text to Submitted when we click on the Submit buttum.
setHeadingText("Submitted")
}
function handleMouseOver(){
  //This function will trigger whenever we hover over the button
  // console.log("Mouse Over")
  // Create the variable to hold the state to change the background color
  setMouseOver(true);
}
function handleMouseOut(){
  setMouseOver(false);
}
  return (
    <div className="container">
      {/* <h1>Hello</h1> */}
      <h1>{headingText}</h1>
      <input type="text" placeholder="What's your name?" />
      <button
      style = {{backgroundColor: isMouseOver ? "black":"white"}}
      onMouseOver = {handleMouseOver}
      onMouseOut={handleMouseOut}
      onClick={handleClick}>Submit</button>
    </div>
  );
}

export default App;
```

EVENT HANDLING IN REACT

Now when I move over the button, it turns black, move away it turns white.

And when you click on the submit button hello changes to Submitted

We used conditional rendering using ternary operator

In line styles. Using event listener like onMouseOver onMOuseOut

And setting and using the state!!!

REACT FORMS

Import the new thirteen-event-handlong-in-react, npm install react-scripts start,npm install,npm start

Will learn complex events, like handling form event.

Creating forms are very common in development like log in, registration, contact etc.

We will again set state and use state and **onChange** event for an input:

How we detect the change in the input?

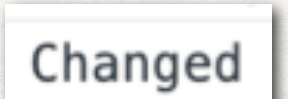
1.Add onChange event in the input and pass a function handleChange:

```
onChange={handleChange}
```

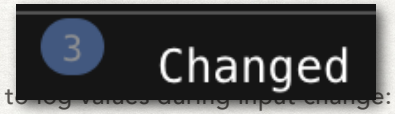
2.Create handleChange function and log to Test

```
function handleChange(){
console.log("Changed");}
```

What this does is whenever we enter a data change happens onChange triggers so the handleChange triggers.

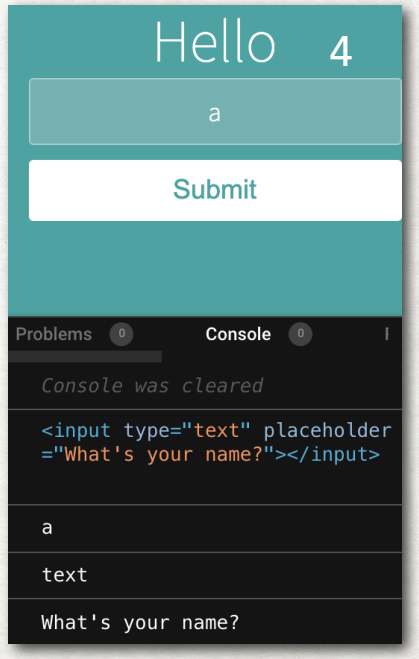


3. When input element triggers it passes an object to handleChange. We can catch that object by **event.** Then we can log various thing related to that event:



4. Add event to handleChange function, And use that event to use to log values during input change:

```
function handleChange(event){
    console.log(event.target)
    console.log(event.target.value);
    console.log(event.target.type);
    console.log(event.target.placeholder); }
```



App.jsx

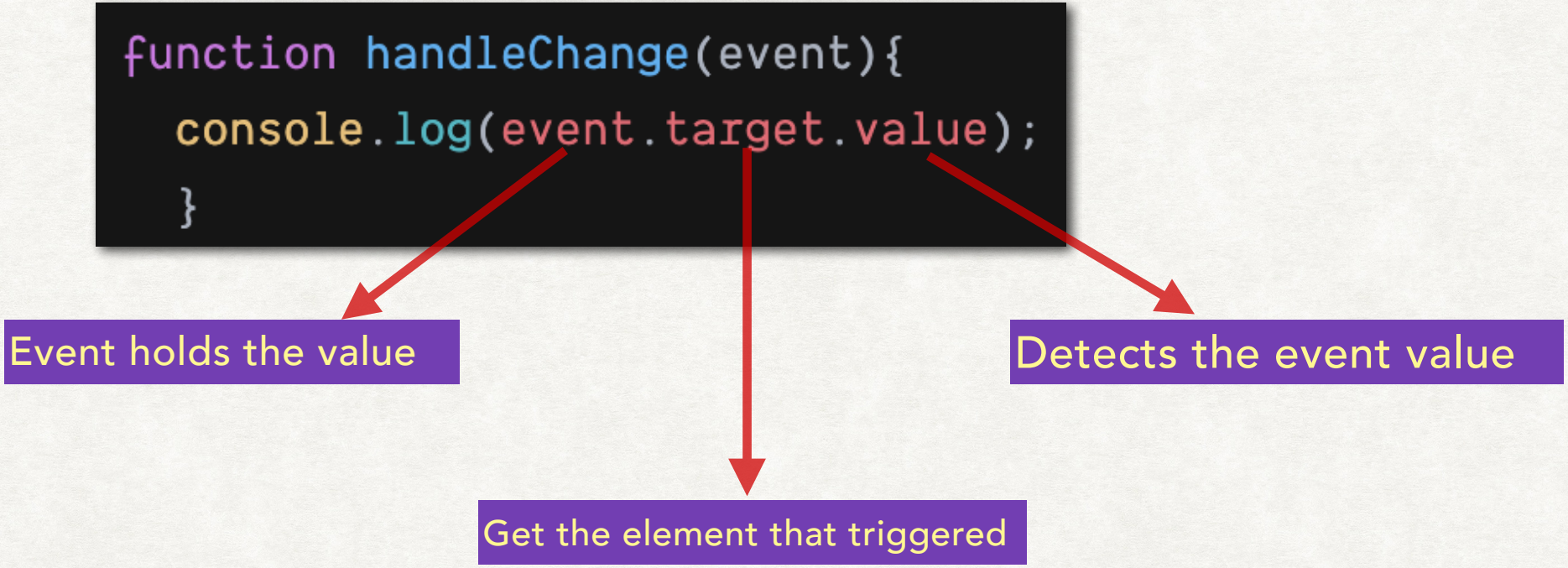
```jsx
import React from "react";

function App() {

  function handleChange(event){
    console.log(event.target)
    console.log(event.target.value);
    console.log(event.target.type);
    console.log(event.target.placeholder);}

  return (
    <div className="container">
      <h1>Hello </h1>
      <input
      onChange={handleChange}
      type="text"
      placeholder="What's your name?" />
      <button>Submit</button>
    </div>
  );
}
export default App;
```

REACT FORMS


function handleChange(event){
console.log(event.target.value);
}
Event holds the value
Get the element that triggered
Detects the event value

REACT FORMS

Now we are getting the data using onChange handler, but how can we use this data on the screen?

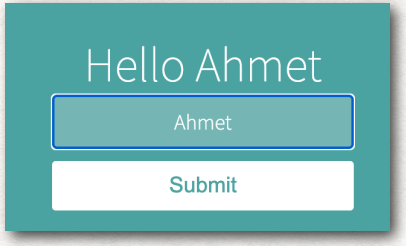
For example as we enter our name in the box, we want to see it on the heading. Again we can use state hook:

1. Create const with initial value is empty string.

```
const [name,setName] = useState("");
```

2. Use setName function to update the name as we update the input

```
function handleChange(event){
 setName(event.target.value);}
```



3. In input, Then pass the name inside the h1 to display the entered updated new value

```
<h1>Hello {name}</h1>
```

So far this works, But Important note about the forms, In html , elements are responsible to handle their own state. We can use value attiribute to set the value with the event.target.value, which is the name variable:

4. In put, So add value property and set the the name(which is event.target.value), so input can always value

```
value={name}
```

This concept is called controlled component

Again we should update the value attribute with the variable that comes from our useState hooks, which is the updated value

App.jsx

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

function App() {
  const [name,setName] = useState("");

  function handleChange(event){
    console.log(event.target.value);
    setName(event.target.value);}

  return (
    <div className="container">
      <h1>Hello {name}</h1>
      <input
      onChange={handleChange}
      type="text"
      placeholder="What's your name?"
      value={name}/>
      <button>Submit</button>
    </div>
  );
}
export default App;
```

REACT FORMS

What if I want to type a name , and I don't want to see the instant update in the heading. I only want to see the entered value after I click on the submit button.

HINT1:Detect event in the submit button

HINT2:Need another variable to tract the state

HINT3:And play with the code until it works

1. In button, Add onClick listener on the submit button(We did similar before have)

```
<button onClick={handleClick}>Submit</button>
```

2. Create handleClick function: This will get the input value from event.target.value(name) that came from input and put in the h1. Let's use setHeading to update the name:

```
function handleClick(){
      setHeading(name);}
```

3. We need to create useState hook to hold the initial value and set the new heading value:

```
const [headingText,setHeading] = useState("");
```

4. In h1, add the new heading text . It will only display in the heading after user clicks on submit button. Only after When we Click button, we shiould take the value from event.target.value and put it it in the h1

```
 <h1>Hello {name} {headingText}</h1>
```

App.jsx

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

function App() {
  const [name,setName] = useState("");
  const [headingText,setHeading] = useState("");
  function handleChange(event){
    console.log(event.target.value);
    setName(event.target.value);}
  function handleClick(){
      setHeading(name);}
  return (
    <div className="container">
      <h1>Hello {name} {headingText}</h1>
      <input
      onChange={handleChange}
      type="text"
      placeholder="What's your name?"
      value={name}/>
      <button onClick={handleClick}>Submit</button>
    </div>
  );
}
export default App;
```

REACT FORMS

Lastly, normally we put inputs and submit button, we use html form component

```
<form>
        <input
          onChange={handleChange}
          type="text"
          placeholder="What's your name?"
          value={name}/>
        <button onClick={handleClick}>Submit</button>
</form>
```

2. In form, add onSubmit handler pass handleClick, add type="submit" to the button, Then delete onClick handler from button cause we no longer need it. This will refresh the page under submit button

```
    <form onSubmit={handleClick}>
          <input
            onChange={handleChange}
            type="text"
            placeholder="What's your name?"
            value={name}/>
          <button type="submit">Submit</button>
        </form>
```

3.In handleClick function, Add event.preventDefault(); Then we can use event handler and get the event to prevent the default behavior. Which is when click the button refreshing automactivcallt

```
function handleClick(event) {
    setHeading(name);
    event.preventDefault();}
```

Now after I click on submit, event handler prevents the default behaviors. DONE

App.jsx

```
1  import React, { useState } from "react";
2
3  function App() {
4    const [name, setName] = useState("");
5    const [headingText, setHeading] = useState("");
6    function handleChange(event) {
7      console.log(event.target.value);
8      setName(event.target.value);
9    }
10   function handleClick(event) {
11     setHeading(name);
12     event.preventDefault();}
13   return (
14     <div className="container">
15       <h1>
16         Hello {name} {headingText}
17       </h1>
18       <form onSubmit={handleClick}>
19         <input
20           onChange={handleChange}
21           type="text"
22           placeholder="What's your name?"
23           value={name}/>
24         <button type="submit">Submit</button>
25       </form>
26     </div>
27   );
28 }
29 export default App;
```