

React Events & Functional Components - Events & Functions First

1. What are Events?

Events are interactions from the user:

- Click button
- Type in input
- Submit form
- Move mouse
- Press keyboard

When an event happens, we run a **function** to handle it.

2. Basic Event Handler Function

```
javascript

function MyComponent() {
  // This is the EVENT HANDLER FUNCTION
  const handleClick = () => {
    console.log('Button clicked!');
  };

  return (
    <button onClick={handleClick}>Click Me</button>
  );
}
```

What happens:

- User clicks button
 - `onClick` detects the click
 - `handleClick` function runs
 - Console shows: `"Button clicked!"`
-

3. All Common Events (Interview Important)

Mouse Events

javascript

```
function MouseEvents() {  
  const handleClick = () => console.log('Clicked!');  
  const handleDoubleClick = () => console.log('Double Clicked!');  
  const handleMouseEnter = () => console.log('Mouse Entered!');  
  const handleMouseLeave = () => console.log('Mouse Left!');  
  const handleMouseDown = () => console.log('Mouse Down!');  
  const handleMouseUp = () => console.log('Mouse Up!');  
  
  return (  
    <div>  
      <button onClick={handleClick}>Click</button>  
      <button onDoubleClick={handleDoubleClick}>Double Click</button>  
      <div  
        onMouseEnter={handleMouseEnter}  
        onMouseLeave={handleMouseLeave}  
        onMouseDown={handleMouseDown}  
        onMouseUp={handleMouseUp}  
        style={{ padding: '20px', border: '1px solid black' }}  
      >  
        Hover and Click  
      </div>  
    </div>  
  );  
}
```

Keyboard Events

javascript

```
function KeyboardEvents() {  
  const handleKeyDown = (event) => {  
    console.log('Key pressed:', event.key);  
  };  
  
  const handleKeyUp = (event) => {  
    console.log('Key released:', event.key);  
  };  
  
  return (  
    <div>  
      <input  
        type="text"  
        onKeyDown={handleKeyDown}  
        onKeyUp={handleKeyUp}  
        placeholder="Press any key"  
      />  
    </div>  
  );  
}
```

Form Events

javascript

```
function FormEvents() {  
  const handleChange = (event) => {  
    console.log('Input changed:', event.target.value);  
  };  
  
  const handleSubmit = (event) => {  
    event.preventDefault();  
    console.log('Form submitted!');  
  };  
  
  const handleFocus = () => {  
    console.log('Input focused!');  
  };  
  
  const handleBlur = () => {  
    console.log('Input lost focus!');  
  };  
  
  return (  
    <form onSubmit={handleSubmit}>  
      <input  
        type="text"  
        onChange={handleChange}  
        onFocus={handleFocus}  
        onBlur={handleBlur}  
        placeholder="Type here"  
      />  
      <button type="submit">Submit</button>  
    </form>  
  );  
}
```

Other Common Events

javascript

```
function OtherEvents() {  
  const handleScroll = () => {  
    console.log('Page scrolled!');  
  };  
  
  const handleContextMenu = (e) => {  
    e.preventDefault();  
    console.log('Right clicked!');  
  };  
  
  const handleDoubleClick = () => {  
    console.log('Double clicked!');  
  };  
  
  return (  
    <div  
      onScroll={handleScroll}  
      onContextMenu={handleContextMenu}  
      onDoubleClick={handleDoubleClick}  
      style={{ height: '100vh', border: '1px solid black' }}  
    >  
      Try: scroll, right-click, double-click  
    </div>  
  );  
}
```

4. Event Handler with Parameters

Sometimes you need to pass data to the function:

```
javascript
```

```
function ButtonsWithParams() {
  const handleClick = (buttonName) => {
    console.log(`${buttonName} button was clicked`);
  };

  const handleInputChange = (event) => {
    console.log('Current value:', event.target.value);
  };

  return (
    <div>
      <button onClick={() => handleClick('Button 1')}>Button 1</button>
      <button onClick={() => handleClick('Button 2')}>Button 2</button>
      <input onChange={handleInputChange} placeholder="Type..." />
    </div>
  );
}
```

5. Event Object - What You Get

Every event handler receives an `event` object with useful information:

javascript

```
function EventObject() {
  const handleClick = (event) => {
    console.log('Event type:', event.type); // "click"
    console.log('Target element:', event.target); // The button
    console.log('Coordinates:', event.clientX, event.clientY); // Position
  };

  const handleChange = (event) => {
    console.log('Input value:', event.target.value); // What user typed
    console.log('Input name:', event.target.name); // Input name attribute
  };

  return (
    <div>
      <button onClick={handleClick}>Click Me</button>
      <input name="username" onChange={handleChange} placeholder="Type..." />
    </div>
  );
}
```

6. Common Event Properties

Property	What it gives	Example
<code>event.type</code>	Type of event	"click", "change", "keydown"
<code>event.target</code>	The element that triggered event	The button, input, div
<code>event.target.value</code>	Value of input field	User's text
<code>event.target.name</code>	Name of input	"username", "email"
<code>event.key</code>	Which key was pressed	"Enter", "a", "Backspace"
<code>event.clientX, event.clientY</code>	Mouse position	123, 456
<code>event.preventDefault()</code>	Stop default behavior	Prevent form reload

7. Using Variables to Store Values (No State Yet)

Now we can store values in **regular JavaScript variables** and show them in the DOM:

```
javascript

function CounterWithVariable() {
  let count = 0; // Regular JavaScript variable

  const handleClick = () => {
    count = count + 1;
    console.log('Count:', count);
    // Problem: Screen doesn't update! (yet)
  };

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={handleClick}>Increment</button>
    </div>
  );
}
```

Problem: The variable `count` changes, but React doesn't re-render the screen!

8. Showing Variables in DOM - Manual DOM Update

We can manually update the DOM using JavaScript:

```
javascript

function CounterWithDOM() {
  let count = 0;

  const handleClick = () => {
    count = count + 1;
    console.log('Count value:', count);

    // Manually update the DOM
    document.getElementById('counter').textContent = count;
  };

  return (
    <div>
      <p>Count: <span id="counter">0</span></p>
      <button onClick={handleClick}>Increment</button>
    </div>
  );
}
```

How it works:

1. Click button
 2. `count` variable increases
 3. `console.log` shows the value
 4. `document.getElementById('counter').textContent = count;` updates the screen
-

9. More DOM Examples

Changing Input Value in DOM

```
javascript
```

```
function InputDisplay() {  
  let inputValue = "";  
  
  const handleChange = (event) => {  
    inputValue = event.target.value;  
    console.log('Input value:', inputValue);  
  
    // Update display in DOM  
    document.getElementById('display').textContent = inputValue;  
  };  
  
  return (  
    <div>  
      <input onChange={handleChange} placeholder="Type..." />  
      <p>You typed: <span id="display"></span></p>  
    </div>  
  );  
}
```

Toggling Classes with DOM

javascript

```
function ToggleWithDOM() {  
  let isVisible = true;  
  
  const handleToggle = () => {  
    isVisible = !isVisible;  
    console.log('Visible:', isVisible);  
  
    // Update DOM element class  
    const element = document.getElementById('box');  
    if (isVisible) {  
      element.style.display = 'block';  
    } else {  
      element.style.display = 'none';  
    }  
  };  
  
  return (  
    <div>  
      <button onClick={handleToggle}>Toggle Box</button>  
      <div id="box" style={{ width: '100px', height: '100px', backgroundColor: 'blue' }}>  
        Box  
      </div>  
    </div>  
  );  
}
```

Changing Colors with DOM

javascript

```
function ColorChangerWithDOM() {  
  let currentColor = 'white';  
  
  const handleColorChange = (color) => {  
    currentColor = color;  
    console.log('Current color:', currentColor);  
  
    // Update background color in DOM  
    document.getElementById('colorBox').style.backgroundColor = color;  
  };  
  
  return (  
    <div>  
      <button onClick={() => handleColorChange('red')}>Red</button>  
      <button onClick={() => handleColorChange('blue')}>Blue</button>  
      <button onClick={() => handleColorChange('green')}>Green</button>  
  
      <div  
        id="colorBox"  
        style={{  
          width: '200px',  
          height: '200px',  
          backgroundColor: currentColor,  
          marginTop: '20px',  
          transition: '0.3s'  
        }}  
      >  
        Color Box  
      </div>  
    </div>  
  );  
}
```

List Counter with DOM

javascript

```

function ListCounterWithDOM() {
  let itemCount = 0;

  const handleAddItem = () => {
    itemCount = itemCount + 1;
    console.log('Items added:', itemCount);

    // Update count display
    document.getElementById('count').textContent = itemCount;

    // Add new item to list
    const newItem = document.createElement('li');
    newItem.textContent = `Item ${itemCount}`;
    document.getElementById('list').appendChild(newItem);
  };

  return (
    <div>
      <button onClick={handleAddItem}>Add Item</button>
      <p>Total Items: <span id="count">0</span></p>
      <ul id="list"></ul>
    </div>
  );
}

```

10. DOM Methods You Need to Know

Method	What it does	Example
<code>document.getElementById(id)</code>	Get element by ID	<code>document.getElementById('box')</code>
<code>.textContent</code>	Change text inside element	<code>el.textContent = 'Hello'</code>
<code>.innerHTML</code>	Change HTML inside element	<code>el.innerHTML = '<p>Hello</p>'</code>
<code>.style.property</code>	Change CSS style	<code>el.style.color = 'red'</code>
<code>.classList.add()</code>	Add CSS class	<code>el.classList.add('active')</code>
<code>.classList.remove()</code>	Remove CSS class	<code>el.classList.remove('active')</code>
<code>.appendChild()</code>	Add child element	<code>parent.appendChild(child)</code>
<code>.removeChild()</code>	Remove child element	<code>parent.removeChild(child)</code>

Method	What it does	Example
<code>.addEventListener()</code>	Add event listener	<code>el.addEventListener('click', fn)</code>

11. Summary - Variables + DOM Updates

The Flow:

1. Create variable to store data
2. Create function to handle event
3. Update variable in function
4. Use `console.log` to see value
5. Use `document.getElementById()` and `.textContent/.style` to update screen

javascript

// Pattern:

```
let myVariable = 'initial value';
```

```
const handleEvent = () => {
  myVariable = 'new value'; // Update variable
  console.log('Value:', myVariable); // Check console
  document.getElementById('myId').textContent = myVariable; // Update screen
};
```

```
return (
  <div>
    <p id="myId">{myVariable}</p>
    <button onClick={handleEvent}>Update</button>
  </div>
);
```

Next Step: After mastering this → Learn **React State (useState)** which automatically updates the screen when variables change!