2. Refs, Keys, Event Handling, Forms

Refs

Allow you to directly interact with DOM nodes/React elements created in the render function

```
class MyDiv extends React.Component{
        constructor(props){
                super(props);
                // creates reference object (unassigned)
                this.myRef = React.createRef();
        }
        componentDidMount(){
                this.myRef.current.innerHTML = "Div Changed";
                this.myRef.current.style.backgroundColor = "yellow";
        }
        render(){
                return(
                        // assigning ref to div element
                        <div ref={this.myRef}>My Div</div>
                )
        }
}
```

Callback Refs

Instead of using `React.createRef()`, you can pass a function into your `ref` attribute. This function is called with your DOM element when it is mounted/updated.
If the element is removed from the DOM, the function is called with `null`.

```
class ListWithRefs extends React.Component {
  constructor(props) {
    super(props);
    this.listItemRefs = {}; // Object to store refs for multiple items
  }

  setListItemRef = (id, element) => {
    if (element) {
      this.listItemRefs[id] = element; // Save the ref to the corresponding DOM node
    } else {
      delete this.listItemRefs[id]; // Clean up the ref if the element is removed
    }
  };

  focusItem = (id) => {
    if (this.listItemRefs[id]) {
      this.listItemRefs[id].focus();
    }
```

```
  };

  render() {
    const items = [1, 2, 3];
    return (
      <div>
        {items.map((id) => (
          <input
            key={id}
            type="text"
            ref={(element) => this.setListItemRef(id, element)}
          />
        ))}
        <button onClick={() => this.focusItem(1)}>Focus Input 1</button>
        <button onClick={() => this.focusItem(2)}>Focus Input 2</button>
        <button onClick={() => this.focusItem(3)}>Focus Input 3</button>
      </div>
    );
  }
}

root.render(
  <div>
    <ListWithRefs/>
  </div>
)
```

Keys

- Allow React to keep track of specific elements -> if an item is updated/removed, only that item needs to be re-rendered instead of an entire list
- **NOTE**: Each child in an array or iterator should have a unique "key" prop
- Example for key: array indices (discouraged)

```
import React from "react";
import ReactDOM from "react-dom";

class ItemList extends React.Component {
  render() {
    const items = ["Apple", "Banana", "Cherry", "Date"];
    return (
      <div>
        <h1>Fruit List</h1>
        <ul>
          {items.map((item, index) => (
            // Key is set to the item name, which should be unique
            // could also do an id and name field, and set key = {item.id}
            <li key={item}>{item}</li>
          ))}
        </ul>
      </div>
```

```
    );
  }
}

const root = ReactDOM.createRoot(document.getElementById("container"));
root.render(<ItemList />);
```

Event Handling

- React event handling -> Synthetic events
- Cannot prevent default behaviour by returning `false`; you need to call `event.preventDefault`
- Event object passed to handler function: SyntheticEvent -> wrapper over DOMEvent
- Define this in constructor itself

```
var txt, ev;
  class MyDiv3 extends React.Component{
    constructor(props){
      super(props);
      this.setRef= (el)=>{ this.myRef=el};
      this.showChar=(event)=>{
        if(event.shiftKey)
          txt="<span style='color:red'>"+event.key+
          "</span>"
        else
          txt=event.key
        this.myRef.innerHTML+=txt
        console.log(event);
        event.persist();
        ev=event;
        //return=false;
        event.preventDefault();
        //event.stopPropagation();

      }
    }
     render(){
    return(
      <div>
        <input onKeyPress={this.showChar} type="text"/>
        <h1 ref={this.setRef}/>
      </div>
    )
   }
  }
```

Forms

- Values of `input`, `textarea` and `select` are controlled using state variables and set only using setState
- This is called controlled component

```
  textarea> Default Value </textarea>  can be changed to
<textarea value={this.state.value} />

<select>
<option select value=“optionselected”> Default Value </option>
</select>
can be changed to
<select value={this.state.value}>
<option value=“optionselected”>Default Value</option>
</select>
```

```
var txt,ev;
                        class BMICalc extends React.Component{
                                constructor(props){
                                        super(props);
                                        this.state={
                                                name:'Bob',
                                                age:22
                                        }

this.handleChange=this.handleChange.bind(this);

this.handleSubmit=this.handleSubmit.bind(this);
                                }
                                handleChange=function(event){
                                        var name=event.target.name;
                                        var value=event.target.value
                                        this.setState({
                                                [name]:value
                                        })
                                }
                                handleSubmit=function(event){
                                        console.log("Name entered:"+this.state.name);
                                        console.log("Age entered:"+this.state.age);
                                        event.preventDefault();
                                }
                                render(){
                                        return(
                                                <form onSubmit={this.handleSubmit}>
                                                        <label>
                                                        Name:
                                                        </label>
                                                        <input name="name" value=
{this.state.name} onChange={this.handleChange} type="text"/>
                                                        <label>
                                                        Age:
                                                        </label>
                                                        <input name="age" value=
{this.state.age} onChange={this.handleChange} type="text"/>
                                                        <input type="submit"
```

```
value="submit"/>
                                                    </form>
                                                )
                                            }
                                        }
      const root = ReactDOM.createRoot(document.getElementById('root'));
      root.render(
        <BMICalc />
      );
```