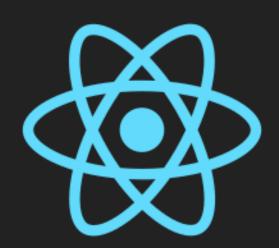
### REACT



#### THE GOAL FOR THIS SESSION

- 1. Understand how useRef works
- 2. Understand how forms work with state (controlled)
- 3. Understand how forms work without state (uncontrolled)

#### **AGENDA**

- 1. Mini Case: Forms and controlled components
- 2. useRef
- 3. Forms
  - uncontrolled with useRef
  - controlled with useState
  - form validation with useEffect?

## useRef

- https://reactjs.org/docs/hooksreference.html#useref
- https://beta.reactjs.org/learn/referencingvalues-with-refs
- The returned object will persist for the full lifetime of the component.
- A common use case is to access a child imperatively
- Or to store a value between renders

- If you pass a ref object to React with <div ref={myRef}/>, React will set its current property to the corresponding DOM node whenever that node changes.
- We use it when we really need to go into the DOM
- Like for forms

#### useRef vs useState

useRef	useState
Doesn't trigger re- render when you change it.	Triggers re-render when you change it.
Mutable—you can modify and update current's value	"Immutable"—you must use the state setting function

```
import { useRef } from "react";
                                                                 ()
 2
   export default function Form() {
     const form = useRef(null);
 4
 5
     function onSubmit(e) {
 6
 7
       if (form.current.checkValidity()) {
 8
 9
10
11
     return (
       <form onSubmit={onSubmit} ref={form}>
12
         <input type="submit" />
13
      </form>
14
     );
15
16 }
```

Using useRef as a "global variable"

If the component is rerendered, the id is persisted

Link (content removed from PDF)

## FORMS

https://reactjs.org/docs/forms.html

What? class NameForm extends React.Component?

#### Working with forms is quite different in React

- 1. We store the initial value of each field in state
- 2. We set the value of each field to it's matching state
- 3. We attach an onChange listener to each field
- 4. Whenever the event fires, we update state
- 5. onSubmit we use state as the values (remember, we don't have direct access to the DOM)

```
import { useState } from "react";
                                                                export default function MyForm(props) {
     const [name, setName] = useState("");
     const [email, setEmail] = useState("");
 6
     const nameChanged = (e) => {
8
       setName(e.target.value);
     const emailChanged = (e) => {
       setEmail(e.target.value);
     const onSubmit = (e) => {
13
14
       e.preventDefault();
15
       postStuff({
         name name
```

## VALIDATION USEEFFECT

```
1 import { useState, useEffect, useRef } from "react";
                                                                2 export default function VanillaForm() {
     const [name, setName] = useState("");
 3
     const [cardnumber, setCardnumber] = useState("");
 5
     const [isValid, setIsValid] = useState(false);
     const form = useRef(null);
 6
 7
     useEffect(() => {
 8
       // prettier-ignore
       const isCreditCardValid =
10
         cardnumber.replaceAll(" ", "").length === 16;
11
12
       // prettier-ignore
       setIsValid(
13
         form.current.checkValidity() && isCreditCardValid
14
15
     } [name cardnumher]).
16
```

# 43/4