User Input and Forms

Roland Guijt
Freelance consultant and trainer

@rolandguijt roland.guijt@gmail.com

Forms in JavaScript Applications

Write code that gets references to the input elements

Extract current values

Post to API

Convert internal state to React state

Controlled components

A Controlled Component (1/2)

```
const [ firstname, setFirstname ] = useState("Alice");

return (
  <input type="text" value={firstname} />
);
```

A Controlled Component (2/2)

```
const [ firstname, setFirstname ] = useState("Alice");

return (
  <input type="text" value={firstname}
      onChange={(e) => setFirstname(e.target.value)} />
);
```

Controlled Components and Forms

```
const [ firstname, setFirstname ] = useState("Alice");
const submit = (e) => {
  e.preventDefault();
  //submit firstname to API
};

return (
  <form onSubmit={submit}>
    <input type="text" value={firstname}
        onChange={(e) => setFirstname(e.target.value)}
/>
  </form>
);
```

Multiple Controlled Components and Forms

```
const [ person, setPerson ] = useState({ firstname: "Alice",
lastname: "Doe"});
const submit = (e) => {
  e.preventDefault();
  //submit person to API
};
return (
  <form onSubmit={submit}>
    <input type="text" value={person.firstname}
        onChange={(e) => setPerson({ …person, firstname:
e.target.value})} />
   <input type="text" value={person.lastname}
        onChange={(e) => setPerson({ …person, lastname:
e.target.value})} />
  </form>
```

A Common onChange Handler

```
const [ person, setPerson ] = useState({ firstname: "Alice",
lastname: "Doe"});
const change = ((e) => setPerson({ …person, [e.target.name]:
e.target.value });

return (
  <form onSubmit={submit}>
    <input type="text" name="firstname"
value={person.firstname}              onChange={change}/>
   <input type="text" name="lastname" value={person.lastname}
        onChange={change} />
  </form>
);)
```

textarea

HTML:

```
<textarea>
   Some text
</textarea>
```

React component:

```
<textarea value={state} onChange={change} />
```

select

HTML:

```
<select>
   <option value="option1">1</option>
   <option selected value="option2">2</option>
</select>
```

React component:

```
<select value={state} onChange={change} >
   <option value="option1">1</option>
   <option value="option2">2</option>
</select>
```

Uncontrolled Components

Controlled is the way to go

But work involved

When converting preexisting codebase

Quick and dirty

Temporary

An UnControlled Component

```
const Form = () => {
  const inputEl = useRef(null);
  const submit = (e) => {
    e.preventDefault();
    const inputValue = inputEl.current.value;
    //process inputValue
    };
  return (
    <form onSubmit={submit}>
      <input ref={ inputEl } type="text" />
      <input type="submit" value="Submit" />
    </form>
  );
};
```

UnControlled Components: defaultValue

```
<input ref={ inputEl } type="text" defaultValue={val} />
```

File Input

```
const Form = () => {
  const inputEl = useRef(null);
  const submit = (e) => {
    e.preventDefault();
    const selectedFile = inputEl.current.files[0].name;
    //process selectedFile
    };
  return (
    <form onSubmit={submit}>
      <input ref={ inputEl } type="file" />
      <input type="submit" value="Submit" />
    </form>
  );
};
```

Adding Form Functionality

Validation

Error messages

Handling form submission

State handling

External library

Formik (formik.org)

Next up:

Application Design