Recap

useEffect

ComponentDidMount

When a child is born

`useEffect(callback, [])`

ComponentDidUpdate

He faces ups and downs of life

`useEffect(callback, [depedencies])`

ComponentWillUnmount

He just dies off!

`useEffect(callback, [])`

callback returns array that will be called when component will be unmounted!

Event Handlers Registeration

State is a single source of truth. There will be a single component updating the state.

```
import React from "react";

export default function Parent() {
  const [name, setName] = React.useState("");

  function handleChange(name) {
    setName(() => name);
  }
  return (
    <>
      <h1>Parent Component</h1>
      <strong>Name: {name}</strong>
      <Child nameChangeHandler={handleChange} />
    </>
  );
}

function Child(props) {
  function changeParentState() {
    props.nameChangeHandler("John");
  }
  return (
    <>
      <h1>Child Component</h1>
      <strong></strong>
      <button onClick={changeParentState}>Change Name</button>
    </>
  );
}
```

- State Lift up: Move state to its parent component to share with its siblings or other
- We use `useReducer` and `useContext` for this purpose. We can also use `redux` for state management.

Forms in React

Controlled Components

- Does not have its own state. State cam from its parent as props
- Data input => Get and set in general, not in react
- We have to set the values by ourselves using state
- Reason: **Single source of truth**

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

export class SignupClass extends Component {
  constructor(props) {
    super(props);
    this.state = {
      fullname: "",
      email: "",
      gender: "",
      about: "",
      male: false,
      react: false,
      typescript: false,
      language: ["urdu", "english"],
      country: "",
    };

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

  handleChange(event) {
    // Event Synthesize object: event
    let { name, value, type, checked, selectedOptions } = event.target;
    if (type === "checkbox") value = checked;
    else if (name === "language")
      value = Array.from(selectedOptions, (option) => option.value);
    this.setState({ [name]: value });
  }

  render() {
    return (
      <>
        <h1>Register</h1>
        <p>{JSON.stringify(this.state)}</p>
        <form>
          <label>Name: </label>
          <input
            type="text"
            name="fullname"
            value={this.state.fullname}
            onChange={this.handleChange}
          />

          <label>Email: </label>
          <input
            type="email"
            name="email"
            value={this.state.email}
            onChange={this.handleChange}
          />

          <label>About me</label>
          <textarea
            type="text"
            name="about"
            value={this.state.about}
            onChange={this.handleChange}
          />

          <label>Skills</label>
          <label>React:</label>
          <input
            type="checkbox"
            name="react"
            checked={this.state.react}
            onChange={this.handleChange}
          />
          <label>Typescript:</label>
          <input
            type="checkbox"
            name="typescript"
            checked={this.state.typescript}
            onChange={this.handleChange}
          />

          <label>Gender</label>
          <input type="radio" name="male" value="Male" />
          <label>Male</label>
          <input type="radio" name="male" value="female" />
          <label>Female</label>

          <label>Language</label>
          <select
            name="language"
            value={this.state.language}
            onChange={this.handleChange}
            multiple
          >
            <option value="urdu">Urdu</option>
            <option value="english">English</option>
            <option value="vietnamese">Vietnamese</option>
            <option value="japanese">Japanese</option>
          </select>
          <input type="submit" value="Submit" />
        </form>
      </>
    );
  }
}
```

```
export default SignupClass;
```

Formik Docs

Basics

Docs

```
npm install formik --save
```

or

```
<!-- Add this to the bottom of html page -->
<script src="https://unpkg.com/formik/dist/formik.umd.production.min.js"></script>
```

```
import React from "react";
import { useFormik } from "formik";

export default function Basic() {
  const formik = useFormik({
    initialValues: {
      email: "",
    },
    onSubmit: (values) => {
      console.log(values);
      alert(JSON.stringify(values, null, 2));
    },
  });
  return (
    <form onSubmit={formik.handleSubmit}>
      <label htmlFor="email">Email Address</label>
      <input
        id="email"
        name="email"
        type="email"
        onChange={formik.handleChange}
        value={formik.values.email}
      />
      <button type="submit">Submit</button>
    </form>
  );
}
```

```
import React from "react";
import { useFormik } from "formik";

export default function Basic() {
  const formik = useFormik({
    initialValues: {
      firstName: "",
      lastName: "",
      email: "",
    },
    onSubmit: (values) => {
      console.log(values);
      alert(JSON.stringify(values, null, 2));
    },
  });
  return (
    <form onSubmit={formik.handleSubmit}>
      <label htmlFor="firstName">First Name</label>
      <input
        id="firstName"
        name="firstName"
        type="text"
        onChange={formik.handleChange}
        value={formik.values.firstName}
      />
      <label htmlFor="lastName">Last Name</label>
      <input
        id="lastName"
        name="lastName"
        type="text"
        onChange={formik.handleChange}
        value={formik.values.lastName}
      />
      <label htmlFor="email">Email Address</label>
      <input
        id="email"
        name="email"
        type="email"
        onChange={formik.handleChange}
        value={formik.values.email}
      />
      <button type="submit">Submit</button>
    </form>
  );
}
```

Validation

```
import React from "react";
import { useFormik } from "formik";

export default function Basic() {
  const formik = useFormik({
    initialValues: {
      firstName: "",
      lastName: "",
      email: "",
    },
    validate: (values) => {
      const errors = {};
      if (!values.firstName) {
        errors.firstName = "Required";
      } else if (values.firstName.length > 15) {
        errors.firstName = "Must be 15 characters or less";
      }
      if (!values.lastName) {
        errors.lastName = "Required";
      } else if (values.lastName.length > 20) {
        errors.lastName = "Must be 20 characters or less";
      }
      if (!values.email) {
        errors.email = "Required";
      } else if (
        !/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i.test(values.email)
      ) {
        errors.email = "Invalid email address";
      }
      return errors;
    },
    onSubmit: (values) => {
      console.log(values);
      alert(JSON.stringify(values, null, 2));
    },
  });
  return (
    <form onSubmit={formik.handleSubmit}>
      <label htmlFor="firstName">First Name</label>
      <input
        id="firstName"
        name="firstName"
        type="text"
        onChange={formik.handleChange}
        value={formik.values.firstName}
      />
      {formik.errors.firstName ? (
        <div style={{ color: "red" }}>{formik.errors.firstName}</div>
      ) : null}

      <label htmlFor="lastName">Last Name</label>
      <input
        id="lastName"
        name="lastName"
        type="text"
        onChange={formik.handleChange}
```

```
          value={formik.values.lastName}
        />
        {formik.errors.lastName ? (
          <div style={{ color: "red" }}>{formik.errors.lastName}</div>
        ) : null}

        <label htmlFor="email">Email Address</label>
        <input
          id="email"
          name="email"
          type="email"
          onChange={formik.handleChange}
          value={formik.values.email}
        />
        {formik.errors.email ? (
          <div style={{ color: "red" }}>{formik.errors.email}</div>
        ) : null}
        <button type="submit">Submit</button>
      </form>
    );
  }
```

Visited Fields

```
import React from "react";
import { useFormik } from "formik";

export default function Basic() {
  const formik = useFormik({
    initialValues: {
      firstName: "",
      lastName: "",
      email: "",
    },
    validate: (values) => {
      const errors = {};
      if (!values.firstName) {
        errors.firstName = "Required";
      } else if (values.firstName.length > 15) {
        errors.firstName = "Must be 15 characters or less";
      }
      if (!values.lastName) {
        errors.lastName = "Required";
      } else if (values.lastName.length > 20) {
        errors.lastName = "Must be 20 characters or less";
      }
      if (!values.email) {
        errors.email = "Required";
      } else if (
        !/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i.test(values.email)
      ) {
        errors.email = "Invalid email address";
      }
      return errors;
    },
    onSubmit: (values) => {
      console.log(values);
      alert(JSON.stringify(values, null, 2));
    },
  });
  return (
    <form onSubmit={formik.handleSubmit}>
      <label htmlFor="firstName">First Name</label>
      <input
        id="firstName"
        name="firstName"
        type="text"
        onBlur={formik.handleBlur}
        onChange={formik.handleChange}
        value={formik.values.firstName}
      />
      {formik.touched.firstName && formik.errors.firstName ? (
        <div style={{ color: "red" }}>{formik.errors.firstName}</div>
      ) : null}

      <label htmlFor="lastName">Last Name</label>
      <input
        id="lastName"
        name="lastName"
        type="text"
        onBlur={formik.handleBlur}
        onChange={formik.handleChange}
        value={formik.values.lastName}
      />
      {formik.touched.lastName && formik.errors.lastName ? (
        <div style={{ color: "red" }}>{formik.errors.lastName}</div>
      ) : null}

      <label htmlFor="email">Email Address</label>
      <input
        id="email"
        name="email"
        type="email"
        onBlur={formik.handleBlur}
        onChange={formik.handleChange}
        value={formik.values.email}
      />
      {formik.touched.email && formik.errors.email ? (
        <div style={{ color: "red" }}>{formik.errors.email}</div>
      ) : null}
      <button type="submit">Submit</button>
    </form>
  );
}
```

Schema Validation with Yup

```
npm install yup --save

# or via yarn

yarn add yup
```

```
import React from "react";
import { useFormik } from "formik";
import * as Yup from "yup";

export default function MyYup() {
  const formik = useFormik({
    initialValues: {
      firstName: "",
      lastName: "",
      email: "",
    },
    validationSchema: Yup.object({
      firstName: Yup.string()
        .max(15, "Must be 15 characters or less")
        .required("Required"),
      lastName: Yup.string()
        .max(20, "Must be 20 characters or less")
        .required("Required"),
      email: Yup.string().email("Invalid email address").required("Required"),
    }),
    onSubmit: (values) => {
      console.log(values);
      alert(JSON.stringify(values, null, 2));
    },
  });
  return (
    <form onSubmit={formik.handleSubmit}>
      <label htmlFor="firstName">First Name</label>
      <input
        id="firstName"
        name="firstName"
        type="text"
        onBlur={formik.handleBlur}
        onChange={formik.handleChange}
        value={formik.values.firstName}
      />
      {formik.touched.firstName && formik.errors.firstName ? (
        <div style={{ color: "red" }}>{formik.errors.firstName}</div>
      ) : null}

      <label htmlFor="lastName">Last Name</label>
      <input
        id="lastName"
        name="lastName"
        type="text"
        onBlur={formik.handleBlur}
        onChange={formik.handleChange}
```

```
          value={formik.values.lastName}
        />
        {formik.touched.lastName && formik.errors.lastName ? (
          <div style={{ color: "red" }}>{formik.errors.lastName}</div>
        ) : null}

        <label htmlFor="email">Email Address</label>
        <input
          id="email"
          name="email"
          type="email"
          onBlur={formik.handleBlur}
          onChange={formik.handleChange}
          value={formik.values.email}
        />
        {formik.touched.email && formik.errors.email ? (
          <div style={{ color: "red" }}>{formik.errors.email}</div>
        ) : null}
        <button type="submit">Submit</button>
      </form>
    );
  }
```

ORM vs ODM

Seqeulite

Docs

```
npm install sequelize sqlite3
# or
yarn add sequelize sqlite3
```