|  |  |
| --- | --- |
| Materi | Nilai |
| Pembelajaran React-JS Video ke 21 - 28  (Tugas semester 4 #9) | 95 |

**Saya Sudah Belajar dan Mengerti dan Saya BISA**

1. Mengimport file bootstrap.min.css di dalam folder node\_modules>folder bootstrap>folder dist>folder css>file bootstrap.min.css.
2. Exact pada route digunakan untuk memastikan route hanya merender component yang memiliki path dan location.pathname yang cocok, jika tidak ada yang cocok, maka route yang di render yaitu route terakhir dengan komponen NotFound.
3. Mengimport BrowserRouter yang ada di dalam react-router-dom untuk pengaturan dari router.

Cth :

*import* {BrowserRouter *as* Router, Route} *from* "react-router-dom";

      <Router>

        <Route path='/' component={Front} exact/>

        <Route path='/admin' component={Back}/>

        <Route path='/home' component={Front}/>

      </Router>

Artinya, import BrowserRouter dan digunakan sebagai / alias Router yang di dalam nya terdapat Route yang diambil dari react-router-dom. Terdapat <Router> an <Route> yang mengatur path, apabila dituliskan path “/”, maka akan menjalankan komponen Front dan exact akan mencari dan merender komponen dengan nama Front , jika dituliskan path “/admin”, maka akan menjalankan komponen Back, dan apabila dituliskan path “/home”, maka akan menjalankan komponenen Front.

1. Element kosong / fragment adalah element yang digunakan untuk membungkus child element tampa menambahkan node ke dalam DOM.
2. Mengconvert elemen html ke dalam JSX dengan menggunakan snippet html to JSX, caranya adalah memblog bagian html yang akan diubah ke dalam JSX, lalu klik kanan, dan klik convert to JSX.
3. Komponen <link> adalah komponen react-router yang bertindak seperti tag <a>, tetapi tidak memuat ulang halaman dengan tautan target baru. Di dalam komponen <link> memiliki komponen to untuk menopang pemanggilan dari web.

Cth :

 <Link to="/kategori">

            <li className="list-group-item">Kategori</li>

 </Link>

Artinya, jadi apabila <li> Kategori di klik, maka akan mengarahkan link ke dalam halaman /kategori.

1. Nested route adalah route bersarang.

Cth :

const { url, path } = useRouteMatch();

  <Link to={`${url}/kategori`}>

      <li className="list-group-item">Kategori</li>

  </Link>

Artinya, di dalam const terdapat url dan path, serta useRouteMatch digunakan untuk mengambil nilai route yang digunakan dan disimpan ke dalam url. Di dalam tag <li> Kategori, yang berisi tag <Link> yang memanggil $url/kategori, maka ketika <li> Kategori di klik, maka URL yang muncul adalah URL dengan memanggil localhost:3000/admin/kategori. Jadi, /kategori masuk ke dalam /admin. Karena /admin adalah route yang dipilih menggunakan useRoouteMatch dan disimpan ke dalam url, maka ketika <li> Kategori di klik, akan menjalankan $url yang berisi /admin lalu disambung dengan /kategori.

1. useRouteMatch digunakan untuk menentukan apakah URL cocok dengan jalur route.
2. Switch digunakan untuk membungkus Route, dimana hanya akan merender satu Route saat path cocok dengan URL.
3. Mengambil path dari URL yang di klik menggunakan useRouteMatch() :

const { path } = useRouteMatch();

*return* (

      <Switch>

        <Route path={`${path}/:isi`}>

          <Content />

        </Route>

      </Switch>

Artinya, terdapat const yang berisi path, serta useRouteMatch digunakan untuk mengambil nilai path yang digunakan dan disimpan ke dalam path. Tag <Switch> digunakan untuk membungkus Route agar hanya merender satu Route saat path cocok dengan URL. Di dalam tag <Route> terdapat path yang memanggil $path yang disambungkan dengan variabel isi. Setelah itu, menampilkan isi dari <Content> yang diambil dari dile Content.js.

Jadi, ketika mengklik <Link>, maka nilai route akan diambil dan disimpan ke dalam $url, setelah itu, nilai path akan diambil dan disimpan ke dalam $path, lalu disambungkan dengan variabel yang dibuat dan akan menampilkan isi yang sesuai dengan link yang diklik.

1. useParams() digunakan untuk mengambil nilai dari path yang dilewati dan disimpan ke dalam variabel yang ditulis setelah nama path (${path}/:isi) => nilai dari $path disimpan ke dalam variabel isi.

Cth :

const { isi } = useParams();

      <h4>{isi}</h4>

Artinya, pada file Main.js memanggil Content dan terdapat variabel isi yang dituliskan setelah $path. Lalu pada file Content.js terdapat const isi dan useParams digunakan untuk mengambil nilai variabel yang dilewati. Lalu setelah mendapatkan nilai dari variabel isi, maka variabel isi ditampilkan menggunakan h4.

1. Hook adalah fungsi yang digunakan untuk mengaitkan state dan fitur – fitur lifecycle React daari function component.
2. useState dalah sebuah hook yang akan mengembalikan nilai dari state. useState berisi [variabel, function pengisi variabel]. useState digunakan karena kita tidak bisa mengirim variabel langsung ke dalam return tanpa menggunakan useState. Data dimasukkan ke dalam function pengisi variabel, dan ditampilkan menggunakan variabel.
3. Effect adalah operasi di dalam sebauh fungsi yang memengaruhi hal di luar fungsi itu.
4. useEffect(fn, deps) adalah fungsi yang di dalamnya mengandung effect (fn) dan input (deps). Effect (fn) adalah fungsi yang akan kita berikan ke dalam useEffect agar dipanggil setelah komponen selesai melakukan render. Dan input (deps) adalah array berisi identifier baik variabel atau fungsi yang akan memanggil effect (fn) kembali saat nilai identifier mengalami perubahan. Nilai dari input (deps) bersifat opsional, bisa diisi atau bisa diabaikan.

Ketikan nilai dari input (deps) tidak ditambahkan, maka fungsi yang diberikan ke dalam useEffcet akan dipanggil setiap komponen melakukan render.

1. Mengambil data dari dalam database menggunakan useEffect

Cth :

*import* { link } *from* "../Axios/link";

const Kategori = () => {

useEffect(() => {

    async function fetchData() {

      const request = *await* link.get("/kategori");

      setIsi(request.data);

   }

    fetchData();

  }, []);

Artinya, mengimport data dari link yang ada di dalam file link pada folder Axios, dimana link berisi url yang menyambungkan kepada lumen yang berisi data – data di dalam database. Setelah itu, terdapat const Kategori, di dalamnya terdapat fungsi useEffect dengan async function fetchData() dan terdapat const request yang berisi mengambil data dari {link} dan disambungkan dengan /kategori menggunakan fungsi get() yang digunakan untuk mengambil data. Setelah itu fuction setIsi mengambil nilai dari data pada request, function setIsi digunakan untuk mengisi variabel isi . Lalu, function fetchData dipanggil agar bisa ditampilkan.

1. Membuat form menggunakan react-hook-form.

Cth :

const { register, handleSubmit, reset, errors } = useForm();

function simpan(data) {

    console.log(data);

    reset();

  }

 <form onSubmit={handleSubmit(simpan)}>

            <div className="mb-3">

              <label htmlFor="kategori" className="form-label">

                Kategori

              </label>

              <input

                type="text"

                className="form-control"

                name="kategori"

                placeholder="kategori"

                ref={register}

              />

            </div>

 </form>

Artinya, terdapat const yang berisi objek register, handleSubmit, reset, dan error yang mengambil nilai dari useForm(). Register digunakan untuk memasukkan / mendaftarkan input ke dalam hook. handleSubmit digunakan untuk mengecek data yang akan dimasukkan apakah sesuai dengan peraturan. Lalu reset digunakan untuk mengosongkan form ketika button submit di klik. dan errors digunakan untuk menampilkan pesan karena gagal validasi.

Setelah itu, di dalam function simpan yang parameter berisi data, terdapat console.log yang akan menampilkan data yang dikirimkan. Lalu setelah menampilkan data, maka isi dari form akan dikosongkan.

Pada form terdapat objek onSubmit yang berisi objek handleSubmit yang akan menjalankan function simpan. Lalu di dalam ref, terdapat register yang digunakan untuk memasukkan input ke dalam hook.

Jadi, nilai dari form yang dimasukkan disimpan ke dalam variabel data pada function simpan, lalu divalidasi menggunakan handleSubmit, dan didaftarkan menggunakan register, setelah itu ketika button submit diklik, maka variabel data akan ditampilkan di dalam console.log da nisi dari form akan dihilangkan menggunakan reset().

1. handleSubmit{} digunakan untuk mengecek / memvalidasi sebelum mengirimkan data menggunakan onSubmit.
2. Reset() digunakan untuk mengosongkan form ketika button submit di klik.
3. Ref={register} digunakan untuk memasukkan input ke dalam hook.
4. Errors digunakan untuk menampilkan pesan ketika validasi gagal dilakukan . form yang diisi tidak sesuai dengan peraturan.
5. Untuk menggunakan error, maka di luar <input> diberi {errors.name input && “pesan yang ditampilkan”}. Dan di dalam register diberi ({validasi rules: true}), artinya baris form itu harus diisi.

Cth :

<input

      ref={register({ required: true })}

/>

{errors.kategori && "Kategori wajib diisi\*"}

Artinya, setelah input didaftarkan, maka diberi validasi rules, dan jika form dengan name kategori tidak diisi sesuai peraturan, maka akan menampilkan error dengan tulisan Kategori wajib diisi.

1. Mengirimkan data menggunakan post().

Cth :

 link.post("./kategori", data).then((res) => console.log(res));

Artinya, link yang berisi custom config dengan post() melakukan pengiriman data pada /kategori dan mengirimkan nilai pada variabel data. Setelah itu, respon akan disimpan ke dalam variabel res dan menampilkan res ke dalam console.log.

1. Menghapus data menggunakan delete().

Cth :

async function hapus(id) {

    const res = *await* link.delete("./kategori/" + id);

    setPesan(res.data.pesan);

  }

  fetchData();

Artinya, terdapat function asynchronous hapus dengan parameter id, const res berisi await yang menunda kode akan dijalankan sampai proses asynchonus berhasil, setelah proses asynchronous berhasil, maka await akan menghapus data dari {link} yang disambungkan dengan /kategori dan sesuai dari id yang dipilih dengan menggunakan fungsi delete. Setelah itu, function setPesan akan mengambil data dari objek pesan yang ada di dalam objek data pada variabel res. Lalu, setelah button hapus di klik, maka secara otomatis tampilan data akan berubah karena menjalankan function fetchData().

1. Window.confirm() digunakan untuk menambahkan jendela informasi.

Cth :

*if* (window.confirm("Apakah anda yakin menghapusnya?")) {

    }

Artinya, terdapat pengujian menggunakan if yang di dalamnya menjalankan fungsi window.confirm dengan peringatan yaitu Apakah anda yakin ingin menghapusnya?.

1. setValue() adalah fungsi yang digunakan untuk mengisi value dari form dengan memanggil nama form.

Cth :

 setValue("kategori", res.data[0].kategori);

Artinya, setValue memanggil form dengan name kategori dan memberi value dengan mengambil data dari variabel data yang ada di dalam variabel res dengan index 0 dan objek kategori.

1. Melakukan update data dengan menggunakan pilihan, jika true maka akan insert data dan jika pilihan false, maka akan melakukan update data.

Cth :

*if* (pilihan) {

      const res = *await* link.post("/kategori", data);

      setPesan(res.data.pesan);

    } *else* {

      const res = *await* link.put("/kategori/" + idkategori, data);

      setPesan(res.data.pesan);

      setPilihan(true);

    }

    reset();

    fetchData();

Artinya, terdapat pengujian menggunakan if, yaitu apabila pilihan bernilai trus, maka akan menjalankan const res yang mengirimkan data menggunakan post pada /kategori dan mengirimkan variabel data yang sudah dimasukkan ke dalam form. Apabila data berhasil dimasukkan, maka akan menjalankan function setPesan yang menampilkan objek pesan di dalam variabel data pada variabel res.

Tetapi, jika pilihan bernilai false, maka akan menjalankan const res yang mengupdate data menggunakan put pada /kategori sesuai dengan idkategori yang dipilih dan akan mengirimkan data. Lalu, jika data berhasil diupdate, maka akan menjalankan function setPesan yang menampilkan objek pesan di dalam variabel data pada variabel res. Dan mengembalikan nilai dari pilihan yaitu true.

Setelah itu, nilai dari form akan otomatis berubah karena menjalankan reset(), dan data akan ditampilkan secara otomatis menggunakan function fetchData().

**Saya Belum Mengerti**