Uporaba zunanjih knjižnic

Gregor Jošt

Ponovitev

- Kaj so kavlji (hooks)
- Pravila uporabe hooks
- Kategorije hooks
- useState hook
  - immutable vs mutable value
- useEffect
  - Kdaj se ne potrebuje? Alternative?
- useContext
  - Kateri problem rešuje? Pravila uporabe?

Uporaba zunanjih knjižnic

Upravljanje s stanjem

Upravljanje s stanjem

- Za upravljanje s stanjem React ponuja dva hook-a:
  - **useState**
  - useReducer
- Opcijsko lahko vključimo tudi **useContext**
- Vključitev vseh treh hook-ov omogoča razvoj zmogljivega upravljanja s stanjem
- Če imamo opravka z „veliko“ globalnega stanja, je smiselno uporabiti temu namenjene knjižnice

Upravljanje s stanjem

- **Redux** (oz. **Redux Toolkit**) velja / je veljala za eno bolj priljubljenih knjižnic za upravljanje z globalnim stanjem
- Poenostavi proces ustvarjanja in upravljanja globalnega stanja
- Zmanjša obseg programskega koda, potrebnega za upravljanje s stanjem
- Takšen pristop naredi kodo lažjo za vzdrževanje

Upravljanje s stanjem

```
export const nanizankeSlice = createSlice({
  name: "nanizanke",
  initialState: [],
  reducers: {
    addSeries: (
      state: NanizankaType[],
      action: PayloadAction<NanizankaType>
    ) => {
      const { leto, naziv, stSezon } = action.payload;

      state.push({
        leto,
        naziv,
        stSezon,
      });
    },
  },
});

export const { addSeries } = nanizankeSlice.actions;

export default nanizankeSlice.reducer;
```

```
const SeriesList = () => {
  const nanizanke = useSelector((state: RootState) =>
state.nanizanke);

  return (
    <ul>
      {nanizanke.map(({ leto, naziv, stSezon }) => (
        <li key={naziv}>
          {naziv} ({stSezon}) - {leto}
        </li>
      ))}
    </ul>
  );
};

const AddSeries = () => {
  const dispatch = useDispatch();
  // useState ali useReducer za vnosna polja

  const handleSubmitSeries = (e: FormEvent<HTMLFormElement>)
=> {
    e.preventDefault();
    dispatch(addSeries(series));
  };

  return <form onSubmit={handleSubmitSeries}>{/* ...
*/}</form>;
};
```

Upravljanje s stanjem

- Knjižnice:
  - MobX,
  - Zustand,
  - Recoil,
  - XState

Uporaba zunanjih knjižnic

Pridobivanje podatkov

Pridobivanje podatkov

- Za pridobivanje (oddaljenih) podatkov oz. API klicev knjižnica React ne nudi specializiranih pristopov
- Vse lahko dosežemo s Fetch API in hook-i za upravljanje s stanjem
- Privzeto ne nudi predpomnjenja (caching)

Pridobivanje podatkov

```
function User({userId}: {userId: number}) {
  const [user, setUser] = useState();

  useEffect(() => {
    let ignore = false;

    async function startFetching() {
      const res = await fetch(`https://reqres.in/api/users/${userId}`);
      const data = await res.json();

      if (!ignore) {
        setUser(data.data);
      }
    }

    startFetching();

    return () => {
      ignore = true;
    };
  }, [userId]);

  return <h1>{user?.first_name} {user?.last_name}</h1>;
}
```

Pridobivanje podatkov

- Knjižnice:
  - TanStack Query (včasih znan kot React Query)
  - RTK Query, privzeto vključen v React Toolkit
  - SWR

Pridobivanje podatkov

```
function User({ userId }: { userId: number }) {
  const { data } = useQuery({
    queryKey: ["usersData"],
    queryFn: () =>
      fetch(`https://reqres.in/api/users/${userId}`)
        .then((res) => res.json())
        .then((res) => res.data),
  });

  return (
    <h1>
      {data?.first_name} {data?.last_name}
    </h1>
  );
}
```

```
function User({userId}: {userId: number}) {
  const [user, setUser] = useState();

  useEffect(() => {
    let ignore = false;

    async function startFetching() {
      const res = await
fetch(`https://reqres.in/api/users/${userId}`);
      const data = await res.json();

      if (!ignore) {
        setUser(data.data);
      }
    }

    startFetching();

    return () => {
      ignore = true;
    };
  }, [userId]);

  return <h1>{user?.first_name} {user?.last_name}</h1>;
}
```

Uporaba zunanjih knjižnic

Usmerjanje

Usmerjanje

- Usmerjanje (**routing**) knjižnica React ne nudi
  - Lahko prikazujemo posamezne komponente s pogojnim izrisanjem, a to ne nadomesti usmerjanje
- Ogrodja, kot so Next.js ali Remix imajo to podporo vgrajeno
- Usmerjanje omogoča tudi deljenje kode in „lazy“ nalaganje

Usmerjanje

- Knjižnice:
  - React Router
  - TanStack Router

Usmerjanje

```
import Root, {
  loader as rootLoader,
  action as rootAction,
} from "./routes/root";

const router = createBrowserRouter([
  {
    path: "/",
    element: <Root />,
    errorElement: <ErrorPage />,
    loader: rootLoader,
    action: rootAction,
    children: [
      {
        path: "contacts/:contactId",
        element: <Contact />,
      },
    ],
  },
]);
```

```
ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <RouterProvider router={router} />
  </React.StrictMode>
);
```

Uporaba zunanjih knjižnic

Upravljanje obrazcev

Upravljanje obrazcev

- Večina aplikacij React je še vedno enostranskih (single page application ali SPA)
- Naložijo le eno stran, preko katere se dinamično prikazujejo novi podatki
- Pri potrditvi obrazcev podatki ne posredujejo neposredno iz obrazca na strežnik
- Namesto tega podatke iz obrazca zajamemo na strani odjemalca in jih na strežnik posredujemo s pomočjo programske kode JavaScript

Upravljanje obrazcev

- V primeru knjižnice React lahko obrazce oz. vnosna polja obrazcev obravnavamo na dva načina
  - Brskalniku lahko dovolimo, da obravnava večino elementov obrazca in se na podatke sklicujemo v ustreznem dogodku (npr. ob potrditvi obrazca)
  - Popoln nadzor and vrednostjo vnosnih polj dosežemo tako, da neposredno posodabljamo vrednosti
- Prvi pristop je implementiran v obliki **nenadzorovanih komponent**, ker React ne nastavlja vrednosti
- Drugi pristop pa uporablja **nadzorovane komponente**, saj React aktivno posodablja vrednosti

Upravljanje obrazcev

- Uporabnik lahko vnese podatke preko ene od naslednjih komponent:
  - **<input>**
  - **<select>**
  - **<textarea>**

Upravljanje obrazcev

```
function Obrazec() {
  const [disable, setDisabled] = useState<boolean>(true);
  const [errors, setErrors] = useState<Oseba>({
    ime: "",
    priimek: "",
  });
  const [form, setForm] = useState<Oseba>({ ime: "", priimek: "" });

  useEffect(() => {
    const napake = validacijaObrazca(form);
    if (napake.ime || napake.priimek) setDisabled(true);
    else setDisabled(false);
    setErrors(napake);
  }, [form.ime, form.priimek]);

  const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setForm((values) => ({ ...values, [e.target.name]: e.target.value }));
  };

  const handleSubmit = (e: React.FormEvent<HTMLFormElement>) => {
    e.preventDefault();
    console.log("Vrednost obrazca: ", form);
  };

  return <form onSubmit={handleSubmit}>{/* ...  */}</form>;
}
```

Upravljanje obrazcev

```
const validacijaObrazca = (obrazec: Oseba): Oseba => {
  const napake: Oseba = { ime: "", priimek: "" };
  for (const lastnost in obrazec) {
    const imaVrednost = !!obrazec[lastnost];
    const dolzinaZnakov = obrazec[lastnost].length;
    const { required, min, max } = shema[lastnost];

    if (required && !imaVrednost) {
      napake[lastnost] = `Polje ${lastnost} ne sme biti
prazno!`;
    }

    if (dolzinaZnakov < min || dolzinaZnakov > max) {
      napake[
        lastnost
      ] = `Polje ${lastnost} mora vsebovati najmanj ${min}
in največ ${max} znakov!`;
    }
  }
  return napake;
};
```

Upravljanje obrazcev

- Knjižnice:
  - React Hook Form
  - Formik

Upravljanje obrazcev

```
const schema = yup
  .object({
    firstName: yup.string().required(),
    age: yup.number().positive().integer().required(),
  })
  .required();

type FormData = yup.InferType<typeof schema>;
```

```
export function UserForm(): JSX.Element {
  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm<FormData>({
    resolver: yupResolver(schema),
  });
  const onSubmit = (data: FormData) => console.log(data);

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <input {...register("firstName")} />
      <p>{errors.firstName?.message}</p>

      <input {...register("age")} />
      <p>{errors.age?.message}</p>

      <input type="submit" />
    </form>
  );
}
```