PB138 Moderní značkovací jazyky

Týden 10 - React & forms

Kahoot

Základní HTML form

```
<form>
    <label for="Name">Name</label>
    <input
        id="name"
        name="name"
    />

    <button>Send</button>
</form>
```

- Default `form` method je `GET`.
- Default `form` action cílí na aktuální stránku (*"refresh"*).
- Jaké jsou výchozí typy `<input>` a `<button>` elementů?

Přepsání do Reactu

```
export const Form01HtmlOnly = () => {
    return (
        <form>
            <label htmlFor="name">Name</label>
            <input
                id="name"
                name="name"
            />

            <button>Send</button>
        </form>
    )
}
```

- Co když chceme přepsat výchozí chování HTML formu?

Ošetření submitu

```
export const Form02HandleSubmit = () => {
    const handleSubmit = (event: React.MouseEvent<HTMLElement>) => {
        event.preventDefault();
        console.log("Submit clicked");
    }

    return (
        <form>
            <label htmlFor="name">Name</label>
            <input
                id="name"
                name="name"
            />

            <button type="submit" onClick={handleSubmit}>Send</button>
        </form>
    )
}
```

- Co když chceme mít přístup k datům?

Držení stavu formu

```tsx
import React, {useState} from "react";

interface User {
    name: string
}

export const Form03State = () => {
    const [formData, setFormData] = useState<User>({name: ""});

    const handleSubmit = (event: React.MouseEvent<HTMLElement>) => {
        event.preventDefault();
        console.log("Data submitted!", formData)
    }

    return (
        <form>
            <label htmlFor="name">Name</label>
            <input
                id="name"
                name="name"
                value={formData.name}
                onChange={(e) => setFormData({...formData, name: e.target.value})}
            />

            <button type="submit" onClick={handleSubmit}>Send</button>
        </form>
    )
}
```

- Co když chceme ošetřovat správnost dat?

Validace

```tsx
import React, {useState} from "react";

interface User { name: string }
type UserErrors = { [K in keyof User]?: string }

export const Form04Validation = () => {
    const [formData, setFormData] = useState<User>({name: ""});
    const [errors, setErrors] = useState<UserErrors>({});

    const validate = () => {
        let errors: UserErrors = {}
        if (!formData.name) {
            errors.name = "Name cannot be empty"
        }
        return errors
    }
    const handleSubmit = (event: React.MouseEvent<HTMLElement>) => {
        event.preventDefault();
        const errors = validate()

        if (Object.keys(errors).length > 0) {
            setErrors(errors)
            console.log("Data is invalid!!!", errors)
            return
        }

        setErrors({});
        console.log("Data is valid!", formData)
    }

    return (
        <form>
            <label htmlFor="name">Name</label>
            <input
                id="name"
                name="name"
                value={formData.name}
                onChange={(e) => setFormData({...formData, name: e.target.value})}
            />

            <button type="submit" onClick={handleSubmit}>Send</button>
            <div style={{color: "red"}}>{errors.name}</div>
        </form>
    )
}
```

Možné usnadnění práce

- Nemusíme držet stav formu sami
- Nemusíme psát validaci sami

`react-hook-form` - form za nás

```tsx
interface User {
    name: string
}

export const Form05HookForm = () => {
    const { register, handleSubmit, formState: {errors} } = useForm<User>({defaultValues: {name: ""}});

    const onSubmission = (formData: User) => {
        console.log("Data is valid!", formData)
    }

    return (
        <form onSubmit={handleSubmit(onSubmission)}>
            <label htmlFor="name">Name</label>
            <input
                {...register("name", {required: "Name cannot be empty"})}
            />

            <button type="submit">Send</button>
            <div style={{color: "red"}}>{errors.name?.message}</div>
        </form>
    )
}
```

yup - validace za nás

Viděli jsme při validaci na backendu:

```
const schema = object({
  name: string().required(),
  description: string().min(10).default("Popis bude pridaný"),
  age: number().positive().required(),
  picture: string().default("/images/animal.webp"),
  type: string().matches(/(cat|dog)/).default("dog"),
  sex: boolean().required(),
  addedAt: date().required(),
});
```

Použijeme v useForm hooku:

```
const { register, handleSubmit, formState:{ errors } } = useForm({
    resolver: yupResolver(schema)
  });
```

Otázky?

- HTML `<form>`
- Using forms in React
- Using `react-hook-form` to handle forms
- Using `yup` for validation

Odpovědník

Week 10 - React & Forms

Dnešní úkol

Stáhněte si zip s úkolem, stáhněte si závislosti (`npm install`) a spusťte aplikaci (`npm run dev`).

Splňte následující kroky:

1. Odhalte, co ve *formuláři* chybí, i když je funkční.
2. Přidejte ošetření `onSubmit` .
3. Přepište formulář, aby používal `react-hook-forms` .
4. Přidejte validaci za použití knihovny (`yup` , `zod`).
5. Přidejte chybové hlášky v případě nevalidních dat.
6. Bonus: zkuste vhodně použít `watch` , `getValues` , `setValues` , `control` , `formState`