ToiletPaper #138

React forms without State

Author: Santo Pfingsten / Software Engineer / Office Leipzig

✕ Problem

The most common method of creating a form is with State. This is what it looks like with React Hooks:

```
export const RegistrationForm = ({ submitForm }) => {
    const [name, setName] = useState('');
    const [email, setEmail] = useState('');
    return (
        <form onSubmit={() => submitForm(name, email)}>
            <label>
                <b>Name: </b>
                <input value={name} onChange={(e) => setName(e.currentTarget.value)} />
            </label>
            <label>
                <b>E-Mail: </b>
                <input value={email} onChange={(e) => setEmail(e.currentTarget.value)} />
            </label>
            <button type="submit">Submit</button>
        </form>
    );
};
```

Creating a form in this way is firmly established in the React community. But such forms often consist of more than just 2 input fields and taking a closer look, you will notice that at least 2 new lines have to be added for every additional input field. If you imagine a form with 30+ entries, it is starting to become confusing. In addition, the function will be called again with each and every keystroke, which can also lead to performance problems.

✓ Solution

If you only need the form contents for submission (and e.g. validation only on the server side), you can omit the state. The values can then be accessed with the `FormData` class. `FormData` only needs the `form` element as constructor argument and can either be passed directly as `body` attribute to a `fetch` call or you can iterate over the contained values.

```
export const RegistrationForm = ({ submitForm }) => (
    <form onSubmit={(e) => submitForm(new FormData(e.currentTarget))}>
        <label><b>Name: </b><input name="name" /></label>
        <label><b>E-Mail: </b><input name="email" /></label>
        <button type="submit">Submit</button>
    </form>
);
function submitFormExample(formData: FormData) {
    formData.forEach((value, key) => console.log(key, value));
    for(const [key, value] of formData) { console.log(key, value); }
    fetch("/api/register", { body: formData, ... }).then(...);
}
```

By adding additional `name` attributes, you can identify them in the `FormData` object. They then appear in the `FormData` as `key`.

+ Further Aspects

- FormData documentation: https://developer.mozilla.org/en-US/docs/Web/API/FormData