Web 3

Formulieren

HO
GENT

Gebruikte technologieën

- Formik

HO
GENT

Forms

- **Forms** worden gebruikt zodanig dat onze **gebruikers data** kunnen **ingeven** op onze **webapplicatie**
- In **React** maken we ook gebruik van een **HTML form element** om een **form** aan te maken

```
<form onSubmit={addNote}>
  <input />
  <button type="submit">save</button>
</form>
```

HO GENT

Forms (2)

```
const addNote = (event) => {
  event.preventDefault();
  console.log('button clicked', event.target);
}
```

```
<form onSubmit={addNote}>
  <input />
  <button type="submit">save</button>
</form>
```

HO GENT

Controlled components

- **Hoe** kunnen we nu de **data uit** onze **input** elementen **uithalen** via React?
- In **HTML forms** houden de `input`, `textarea` en `select` elementen **typisch** hun **eigen state** bij
- In **React** willen we dat de **state** van **React** de enige **betrouwbare state** is (single source of truth)
- De form elementen zullen dus de **state** van **React** moeten **gebruiken** (**Controlled components**)

https://reactjs.org/docs/forms.html#controlled-components

HO GENT

Controlled components (2)

```
const App = (props) => {
  const [notes, setNotes] = useState(props.notes);
  const [newNote, setNewNote] = useState('a new note');

  const handleNoteChange = (event) => {
    console.log(event.target.value);
    setNewNote(event.target.value);
  }

  return (
    <form onSubmit={addNote}>
      <input value={newNote} onChange={handleNoteChange} />
      <button type="submit">save</button>
    </form>
  )
}
```

FORMIK

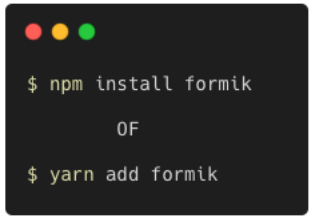
HO
GENT

FORMIK

Formik

- **Formik** is een **framework** dat het ons toelaat om **formulieren** op te **bouwen** in **React zonder veel problemen**
- Gericht op de **3 grootste** “**problemen**” met formulieren in React
  - **Waarden in** en **uit** de **form state** halen
  - **Validatie** en **foutmeldingen**
  - **Afhandelen** van het **versturen** van formulieren

https://formik.org



HO GENT

Formik – useFormik()

- We kunnen gebruik maken van een hook dat er voor zorgt dat we alle Formik state en helper methodes terug krijgen

```
import { useFormik } from 'formik';

// Component ...

const formik = useFormik({
  initialValues: {
    firstName: "",
    lastName: ""
  },
  onSubmit: values => {
    console.log(values);
  },
});

// const { handleChange, handleSubmit, handleBlur, errors, values } = formik;

// ...
```

HO GENT

Formik – useFormik() (2)

```
<form onSubmit={formik.handleSubmit}>
  <label htmlFor="email">Email</label>
  <input id="email"
         name="email"
         type="email"
         onChange={formik.handleChange}
         value={formik.values.email}   />
  <button type="submit">Submit</button>
</form>
```

https://formik.org/docs/examples/basic

HO GENT

yup

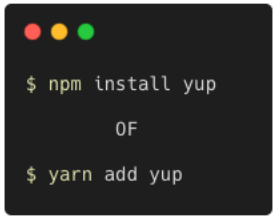
HO
GENT

yup

Yup

- Yup is een **JavaScript schema builder** voor het **valideren** van **waarden** uit **formulieren** (React / React Native)
- In **combinatie** te **gebruiken** met **Formik** (anders validators zelf schrijven)

https://github.com/jquense/yup



HO GENT

Yup – schema

- **Validatieschema aanmaken** via Yup

```
import * as Yup from 'yup';

const schema = Yup.object().shape({
  name: Yup.string().required(),
  age: Yup.number().required(),
  email: Yup.string().email(),
  created: Yup.date().default(() => { return new Date();})
});
```

Yup – koppeling Formik

- Je kan het **Yup schema** dan **koppelen** aan **Formik** zodanig dat de **validatie** kan **gebeuren** in het **Formik** gebeuren

```
const { handleChange, handleSubmit, values } = useFormik({
  validationSchema: schema,
  //...
});
```

HO GENT