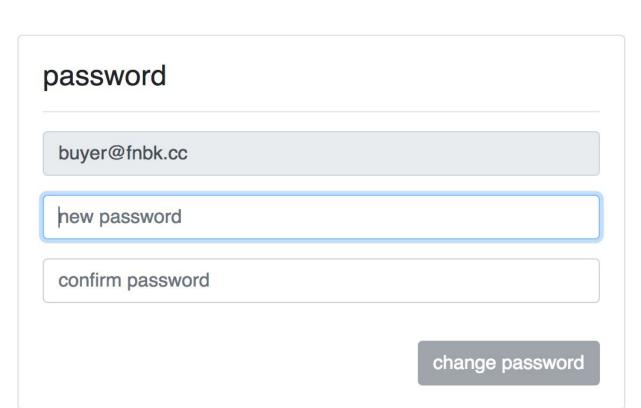# How to structure Software

ReactJS

How To Structure Software (React)

My Account

My Watchlist

Buyer Requests

Seller Offers

My Contracts

Let's Search

password

buyer@fnbk.cc

new password

confirm password

change password

Challenge 1: Big Fat Files
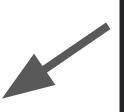
Solution 1: Old Good View & Controller

Separation of Concerns (SoC)

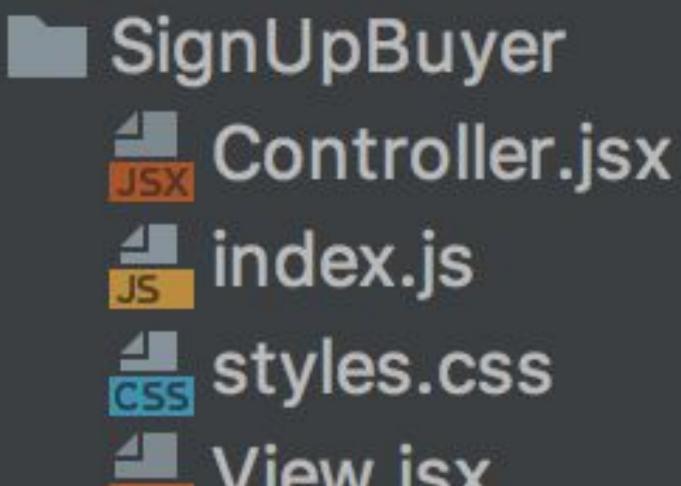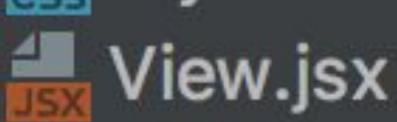- Presentation
- Logic

Big Fat File

view
(only props)

controller

```
import './styles.css';

const SignUpForm = ({
  email, password, error, onEmailChange, onPasswordChange, handleSignUp,
}) => (
  <Form onSubmit={handleSignUp} className="form-centered mt-5">
    <div className="text-center mb-4">
      <h1 className="h3 mb-5 font-weight-normal">sign up buyer</h1>
    </div>
    <FormGroup>
      <Input
        value={email}
        onChange={onEmailChange}
        type="email"
        id="inputEmail"
        className="form-control"
        placeholder="Email address"
        required
        autoFocus
      />
    </FormGroup>
    <FormGroup>
      <Input
        value={password}
        onChange={onPasswordChange}
        type="password"
        id="inputPassword"
        className="form-control"
        placeholder="Password"
        required
      />
    </FormGroup>
    <Button className="btn btn-lg btn-primary btn-block" type="submit">sign up</Button>
    { error && <Alert className="mt-2" color="secondary">{error}</Alert> }
  </Form>
);

SignUpForm.propTypes = {
  error: PropTypes.string.isRequired,
  email: PropTypes.string.isRequired,
  onEmailChange: PropTypes.func.isRequired,
  password: PropTypes.string.isRequired,
  onPasswordChange: PropTypes.func.isRequired,
  handleSignUp: PropTypes.func.isRequired,
};

export default SignUpForm;
```

```
const INITIAL_STATE = {
  error: '',
  email: '',
  password: '',
};

class Controller extends Component {
  constructor(props) {
    super(props);
    this.state = { ...INITIAL_STATE };
    this.handleSignUp = this.handleSignUp.bind(this);
  }

  async handleSignUp(e) {
    const that = this;
    const { email, password } = this.state;
    const { history } = this.props;

    e.preventDefault();
    auth.createUserWithEmailAndPassword(email, password)
      .then(userCred => {...})
      .then(() => history.push(ACCOUNT))
      .catch(err => {
        that.setState({ error: err.message });
      });
  }

  render() {
    const { error, email, password } = this.state;

    return (
      <View
        error={error}
        email={email}
        password={password}
        onEmailChange={e => this.setState({ email: e.target.value })}
        onPasswordChange={e => this.setState({ password: e.target.value })}
        handleSignUp={this.handleSignUp}
      />
    );
  }
}

/* eslint-disable react/forbid-prop-types */
Controller.propTypes = {
  history: PropTypes.object.isRequired,
};

export default withRouter(Controller);
```
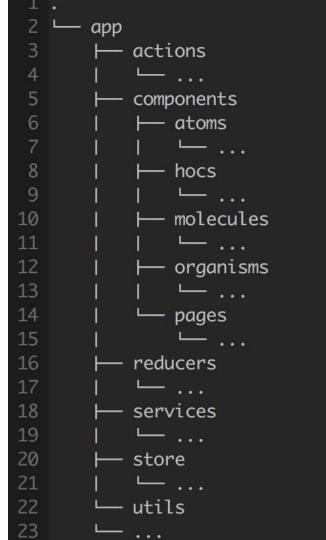
▼ 📁 SignUpBuyer
📄 Controller.jsx
📄 index.js
📄 styles.css
📄 View.jsx

Challenge 2: File Structure

So many different names…

I don't have a clue what's going on?

Please show me the documentation!

```
 1  .
 2  └── app
 3      ├── actions
 4      │       └── ...
 5      ├── components
 6      │   ├── atoms
 7      │   │       └── ...
 8      │   ├── hocs
 9      │   │       └── ...
10      │   ├── molecules
11      │   │       └── ...
12      │   ├── organisms
13      │   │       └── ...
14      │   └── pages
15      │           └── ...
16      ├── reducers
17      │       └── ...
18      ├── services
19      │       └── ...
20      ├── store
21      │       └── ...
22      └── utils
23              └── ...
```

Solution 2: Use Your Website Layout

```
 2  ├── Footer
 3  │       └── ...
 4  ├── Header
 5  │       └── ...
 6  ├── Main
 7  │    ├── Account
 8  │    │    ├── Menu
 9  │    │    └── Page
10  │    │         ├── Contract
11  │    │         ├── Offer
12  │    │         ├── Profile
13  │    │         ├── Request
14  │    │         └── Watchlist
15  │    ├── Landing
16  │    ├── SignUpBuyer
17  │    ├── SignUpSeller
18  │    └── ...
19  └── ...
```

```
const App = () => (
    <div>
        <Header />
        <Main />
        <Footer />
    </div>
);
```

```jsx
const Main = () => (
  <div>
    <Switch>
      <Route exact path={SEARCH} component={Search} />
      <Route exact path={PUBLIC_PROFILE} component={PublicProfile
      <Route exact path={SIGN_UP_SELLER} component={SignUpSeller}
      <Route exact path={SIGN_UP_CUSTOMER} component={SignUpBuyer
      <Route exact path={PASSWORD_FORGET} component={PasswordForg
      <Route exact path={PUBLIC_PROFILE} component={PublicProfile
      <Route component={Account} />
    </Switch>
  </div>
);

export default Main;
```

```jsx
const Page = () => (
  <div className="card">
    <div className="card-body">
      <Switch>
        <Route path={ACCOUNT} component={Profile} />
        <Route path={WATCHLIST} component={Watchlist} />

        <Route path={NEW_REQUEST} component={NewRequest} />
        <Route path={SHOW_REQUEST} component={ShowRequest} />
        <Route path={REQUESTS} component={ListRequests} />

        <Route path={NEW_OFFER} component={NewOffer} />
        <Route path={SHOW_OFFER} component={ShowOffer} />
        <Route path={OFFERS} component={ListOffers} />

        <Route path={NEW_CONTRACT} component={NewContract} />
        <Route path={SHOW_CONTRACT} component={ShowContract} />
        <Route path={CONTRACTS} component={ListContracts} />
      </Switch>
    </div>
  </div>
);
```

Question: What are your Challenges?

Challenge 3: Dependencies everywhere!

Aaaahhhh, all components are interconnected. Please help me!

Software Cell to the rescue ...