Milestone 5
Frontend

Michael John Canson,
Chiu Wong,
Paul Borst

Overview

- Frontend Stack
- Code Structure
- Implementations

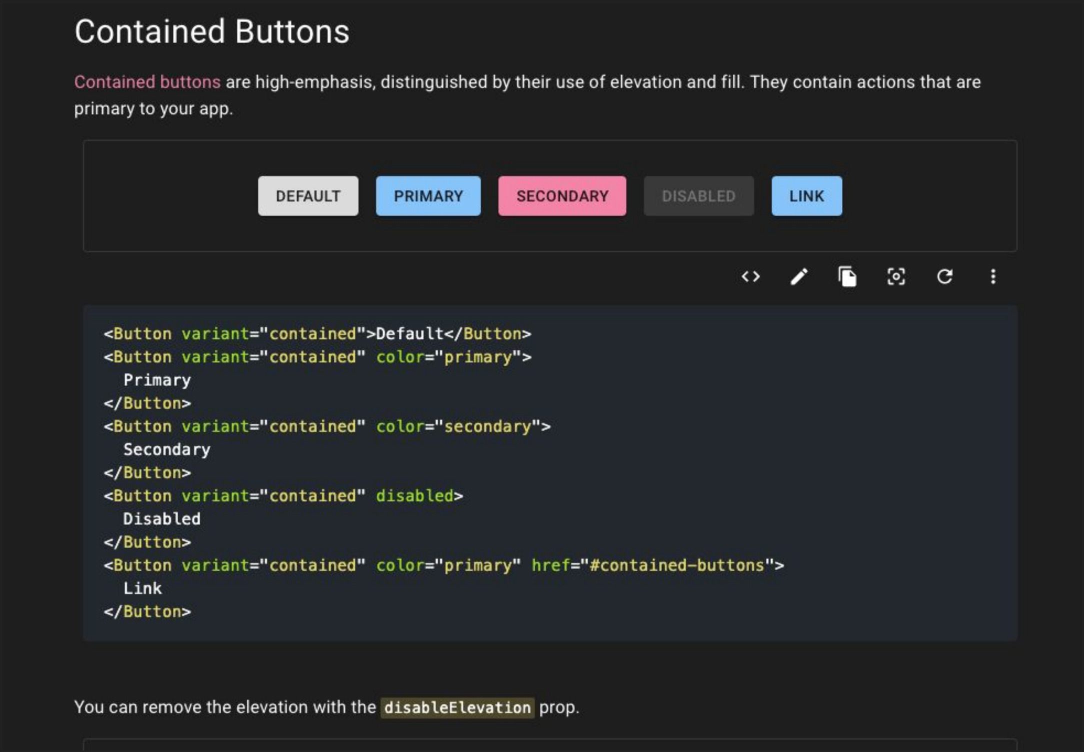
Frontend Stack

- Node.js
  - Asynchronous, event-driven JS runtime dev platform
- React.js (UI/UX)
  - Open-source frontend JS library
  - Material-ui (React UI framework)
- Axios
  - Handles fetch requests or to save data
- Redux
  - State container
- Jest (Unit testing) & Enzyme (testing utility)

File Structure

- Files are grouped by file type
  - Components
    - Header, appbar, lists components, etc...
  - Pages
    - Containers that hold multiple components
    - Dashboard, exercise library, PT Profile
  - Redux
    - State managing files
  - Assets
    - Logos, photos, etc...

Material-UI



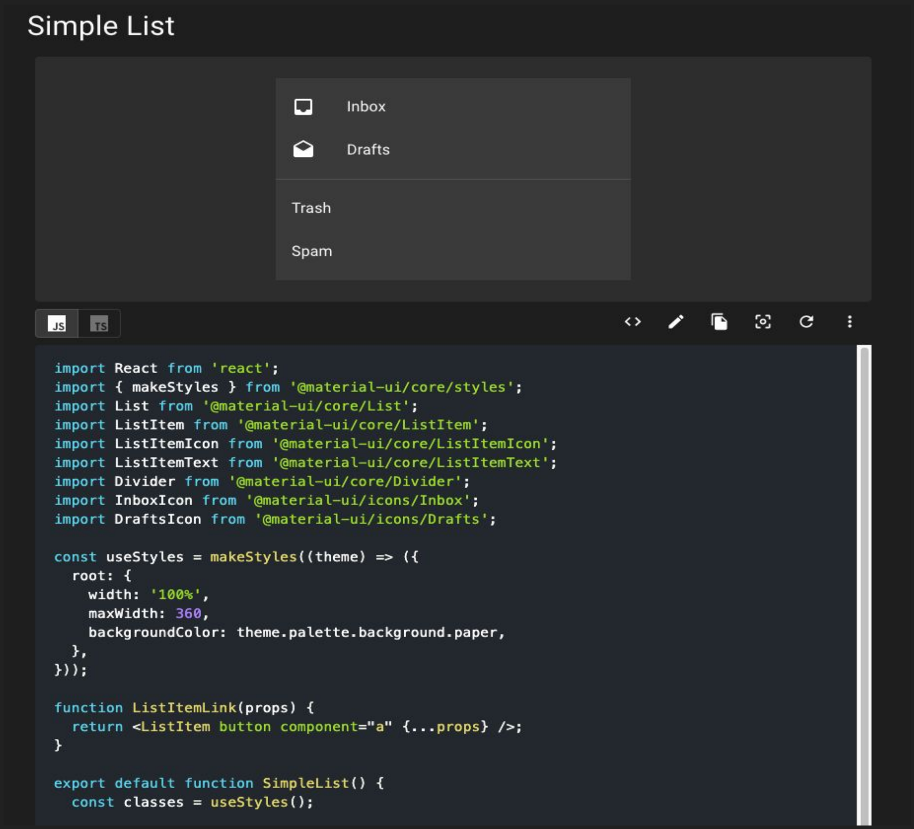
Contained Buttons

Contained buttons are high-emphasis, distinguished by their use of elevation and fill. They contain actions that are primary to your app.

```
<Button variant="contained">Default</Button>
<Button variant="contained" color="primary">
  Primary
</Button>
<Button variant="contained" color="secondary">
  Secondary
</Button>
<Button variant="contained" disabled>
  Disabled
</Button>
<Button variant="contained" color="primary" href="#contained-buttons">
  Link
</Button>
```

You can remove the elevation with the `disableElevation` prop.

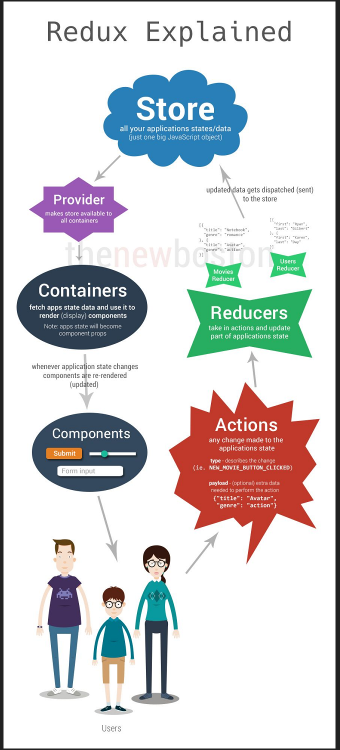
Material-UI



Simple List

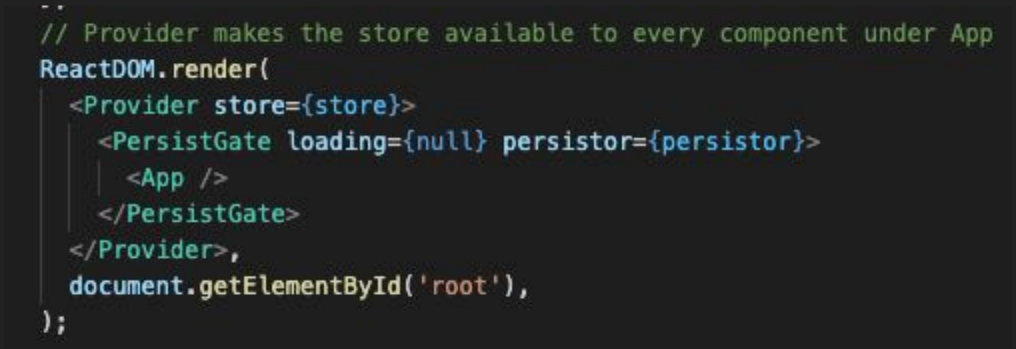
```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import List from '@material-ui/core/List';
import ListItem from '@material-ui/core/ListItem';
import ListItemIcon from '@material-ui/core/ListItemIcon';
import ListItemText from '@material-ui/core/ListItemText';
import Divider from '@material-ui/core/Divider';
import InboxIcon from '@material-ui/icons/Inbox';
import DraftsIcon from '@material-ui/icons/Drafts';

const useStyles = makeStyles((theme) => ({
  root: {
    width: '100%',
    maxWidth: 360,
    backgroundColor: theme.palette.background.paper,
  },
}));

function ListItemLink(props) {
  return <ListItem button component="a" {...props} />;
}

export default function SimpleList() {
  const classes = useStyles();
```

Redux Implementation


Redux Explained
Store
all your applications states/data
(just one big JavaScript object)
Provider
makes store available to all containers
Containers
fetch apps state data and use it to render (display) components
Note: apps state will become component props
whenever application state changes components are re-rendered (updated)
Components
Submit
Form input
Users
Actions
any change made to the applications state
type - describes the change
(ie. NEW_MOVIE_BUTTON_CLICKED)
payload - (optional) extra data needed to perform the action
{"title": "Avatar", "genre": "action"}
Reducers
take in actions and update part of applications state
Movies Reducer
Users Reducer
[{ "title": "Notebook", "genre": "romance" }, { "title": "Avatar", "genre": "action" }]
[{ "first": "Ryan", "last": "Gilbert" }, { "first": "Karen", "last": "Day" }]
updated data gets dispatched (sent) to the store
thenewboston

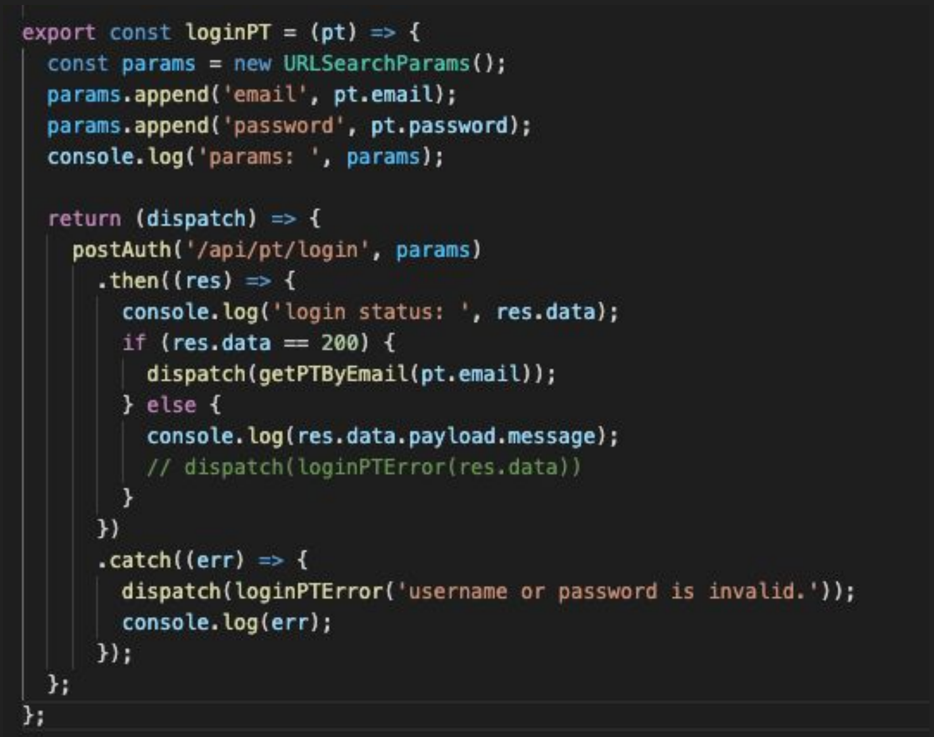
- Axios calls inside **actions**
  - Less axios calls inside functional components
- Allows data to be sent and passed around to different components globally
  - Patient list, workout plans, PT info, etc…
- Redux Files:
  - **Provider** component wraps App container allowing access to the **store** component, which holds all fetched data
  - **Reducers** in charge of what data is being saved and updated inside the **store**
  - Components trigger **actions** which is then taken in by **reducers**

Redux Provider



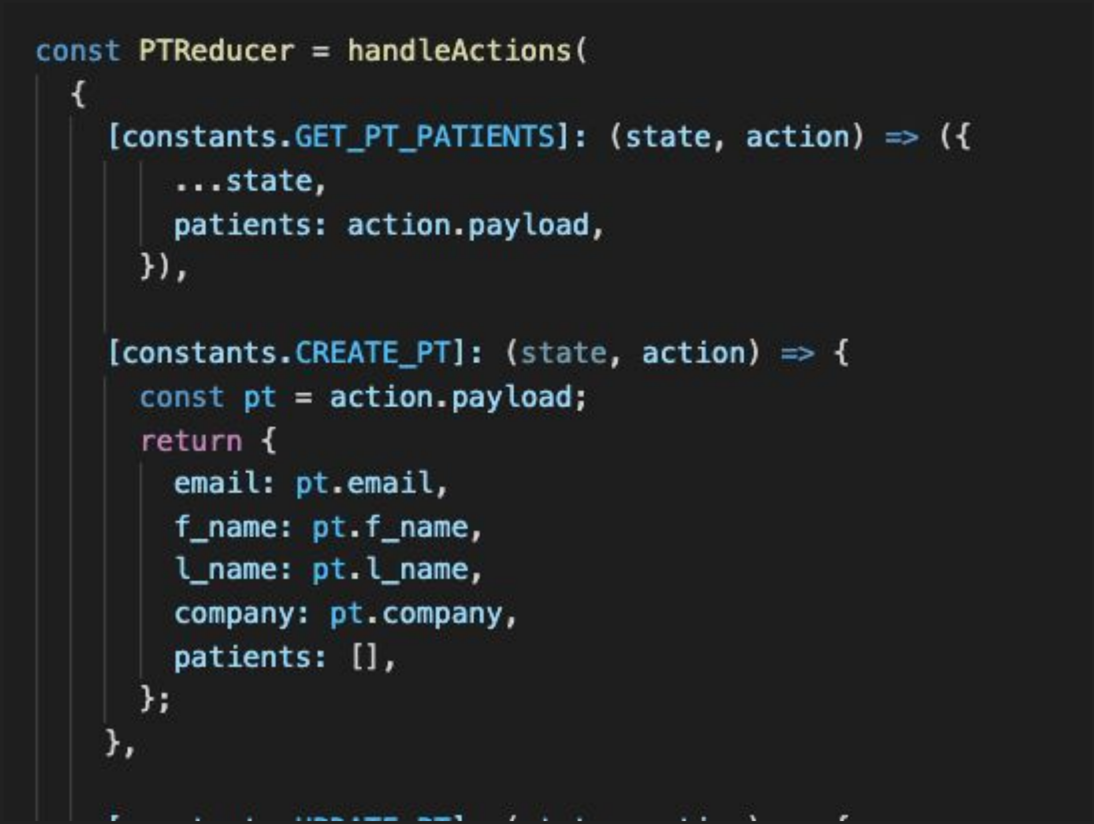
```
// Provider makes the store available to every component under App
ReactDOM.render(
  <Provider store={store}>
    <PersistGate loading={null} persistor={persistor}>
      <App />
    </PersistGate>
  </Provider>,
  document.getElementById('root'),
);
```

Redux Action



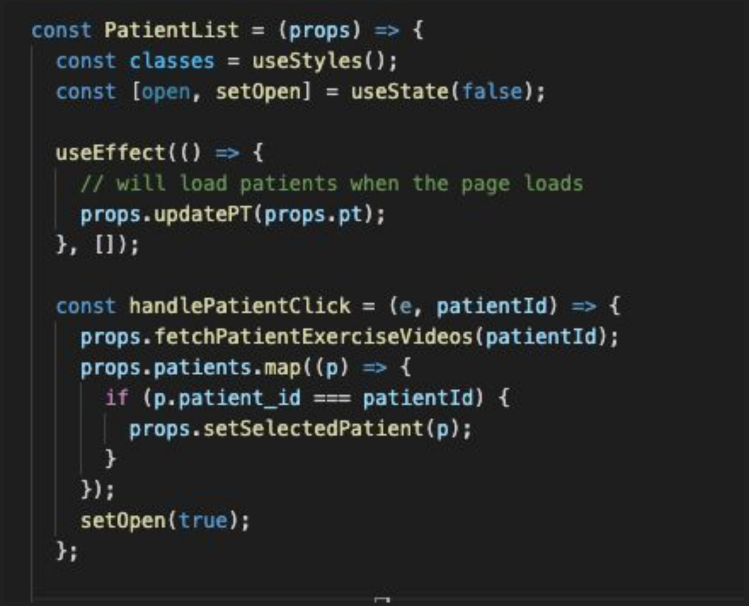
```
export const loginPT = (pt) => {
  const params = new URLSearchParams();
  params.append('email', pt.email);
  params.append('password', pt.password);
  console.log('params: ', params);

  return (dispatch) => {
    postAuth('/api/pt/login', params)
      .then((res) => {
        console.log('login status: ', res.data);
        if (res.data == 200) {
          dispatch(getPTByEmail(pt.email));
        } else {
          console.log(res.data.payload.message);
          // dispatch(loginPTError(res.data))
        }
      })
      .catch((err) => {
        dispatch(loginPTError('username or password is invalid.'));
        console.log(err);
      });
  };
};
```

Redux Reducer

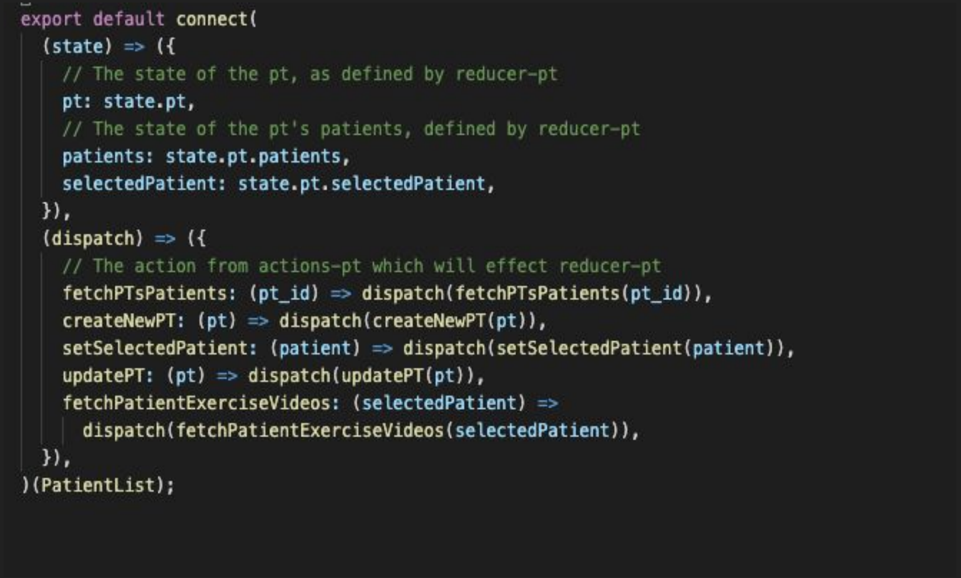


```
const PTReducer = handleActions(
  {
    [constants.GET_PT_PATIENTS]: (state, action) => ({
      ...state,
      patients: action.payload,
    }),

    [constants.CREATE_PT]: (state, action) => {
      const pt = action.payload;
      return {
        email: pt.email,
        f_name: pt.f_name,
        l_name: pt.l_name,
        company: pt.company,
        patients: [],
      };
    },

    [constants.UPDATE_PT]: (state, action) => {
```

Redux Implementation



```
const PatientList = (props) => {
  const classes = useStyles();
  const [open, setOpen] = useState(false);

  useEffect(() => {
    // will load patients when the page loads
    props.updatePT(props.pt);
  }, []);

  const handlePatientClick = (e, patientId) => {
    props.fetchPatientExerciseVideos(patientId);
    props.patients.map((p) => {
      if (p.patient_id === patientId) {
        props.setSelectedPatient(p);
      }
    });
    setOpen(true);
  };
```



```
export default connect(
  (state) => ({
    // The state of the pt, as defined by reducer-pt
    pt: state.pt,
    // The state of the pt's patients, defined by reducer-pt
    patients: state.pt.patients,
    selectedPatient: state.pt.selectedPatient,
  }),
  (dispatch) => ({
    // The action from actions-pt which will effect reducer-pt
    fetchPTsPatients: (pt_id) => dispatch(fetchPTsPatients(pt_id)),
    createNewPT: (pt) => dispatch(createNewPT(pt)),
    setSelectedPatient: (patient) => dispatch(setSelectedPatient(patient)),
    updatePT: (pt) => dispatch(updatePT(pt)),
    fetchPatientExerciseVideos: (selectedPatient) =>
      dispatch(fetchPatientExerciseVideos(selectedPatient)),
  }),
)(PatientList);
```