XSPACE Web App Front-End Design Guidelines

10/08/2018

Contents

Branding and Corporate Identity

`Roboto` is the primary font in the system.

Defaults

```
// default entire stylesheet to sans-serif
* {
    font-family: "Roboto", Arial, sans-serif;
}

//primary heading
h1 {
    font-size: 2.0em;
```

```
}

h2 {
    font-size: 1.5em;
}

h3 {
    font-size: 1.25em;
}

h4 {
    font-size: 1.125em;
}

h5 {
    font-size: 1em;
}

h6 {
    font-size: .875em;
}

p {
    font-size: 1em;
}
```

Colors

```
* Green - #27AE60
* Blue - #00A3FF
* Red - #EB5757
* Yellow - #F2C94C
* Dark Panels - #333333
```

Container vs Page vs Component

When to use pages, ask yourself…

- Does it need authentication to access?
- Does it require the user to perform another activity?
- Will the user understand what activity you are granting?

What is a container?

- A container is a **view controller** that bridges data from the React state tree or other possible pages, components, and containers

When to use a containers, ask yourself…

- Will I need to make multiple pages to interact with the same data?
- Will all of those pages need to be authenticated?
- Will this need to be a front-end endpoint? If so, add container to

App.jsx and create a **page**.

What is a page?

- A page is a window for activity

When to use components, ask yourself…

- Will I use this again somewhere else?
- Are the behaviors different when the user is logged in?
- Is it resistant to changes in the data model, or does is it flexible?

What is a component?

- A component is a designated layout for a behavior to accomplish the activity.

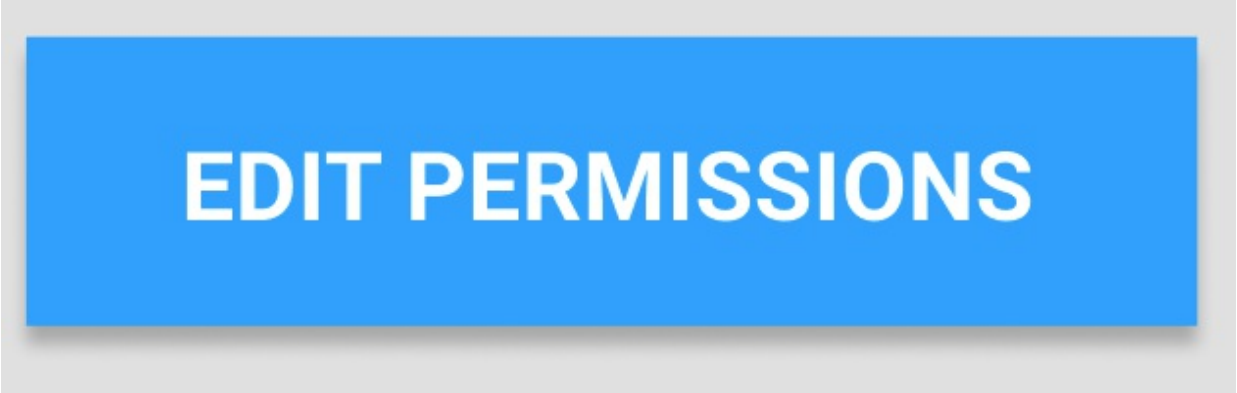
Component Design

Buttons

Using MDBReact Guidelines, specific components follow certain design guidelines.

Activity Button

- This is a material design bootstrap button, placed on the background of a page. The shadow creates distance and emphasis to create the gesture of an action.



Component Button

- Used agasint a white background of another component or layout, but not the page level. This illustrates the button belongs to a sub-activity.

Dropdowns

Dropdowns have a more complex behavior. You may have more than one dropdown in your page. To indicate that a dropdown is closed by default, use the component's **state tree**. React provides a dropdown view by default.

Setup

```
    constructor(props) {
        this.state = {
            dropdownOpen: false
        }

```

```
        this.toggleForm = this.toggleForm.bind(this);
    }
```

Toggle

```
  toggleForm(){
    this.setState({
      dropdownOpen: !this.state.dropdownOpen
    });
  }
```

View

- Note how `Dropdown isOpen={this.state.dropdown}` and `toggle={this.toggleForm}` correspond to the previous two code snippets.

```
    let upcType = (
      <Dropdown isOpen={this.state.dropdownOpen} toggle={t
his.toggle}>
        <DropdownToggle nav caret><p>{this.state.upcType}<
/p></DropdownToggle>
        <DropdownMenu>
            <DropdownItem onClick={this.select}>UPC-A</Dro
pdownItem>
            <DropdownItem onClick={this.select}>UPC-E</Dro
pdownItem>
```

```
            <DropdownItem onClick={this.select}>EAN-8</Dro
pdownItem>
            <DropdownItem onClick={this.select}>EAN-13</Dr
opdownItem>
        </DropdownMenu>
      </Dropdown>
    )
```

Custom Dialogs (Modal)

Dialogs have the highest level of complexity. In order to create a dialog, it must be in the view tree of the component. **NOTE**: Modals will block UI, so handle with care!

Setup

- Most of the activity in a dialog is equally dependent on the state tree of the page or component that it is contained in. Visibility state is not necessary.

```
    constructor(props) {
        this.toggleDialog = this.toggleDialog.bind(this);
    }
```

Toggle

- Not a significant difference compared to dropdown

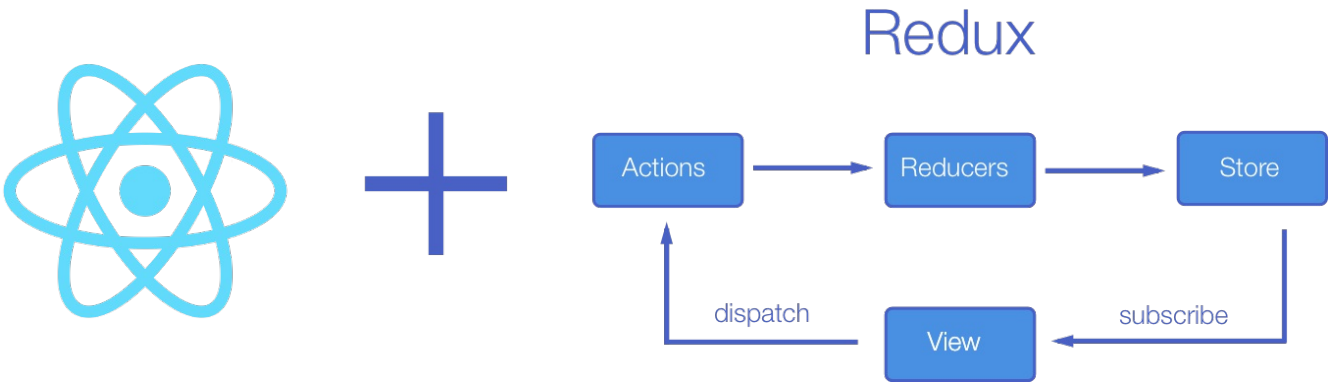
```
  toggleDialog() {
    this.setState({
      modal: !this.state.modal
    });
  }
```

View

- Note how `Modal isOpen={this.state.modal}` and `toggle={this.toggleForm}` correspond to the previous two code snippets. Any additional information in regards to dismissal of dialog can be found Link to MDBReact Modals

```
    <Container>
        ...
        <Modal isOpen={this.state.modal}>
          <ModalHeader toggle={this.toggleDialog}>Creating
 A New Product</ModalHeader>
          <ModalBody>
            Please wait, we are processing your submission
...
          </ModalBody>
        </Modal>
        ...
    </Container>
```

RSAA and the React State Tree



There are three methods that the front end use to promise the data to the front end.

- `fetch`
- `axios`
- `dispatch()` through react-redux and RSAA, as shown above

When to use fetch

- Use `fetch()` for **GET** data that does not need to be persisted and can be disposed of after viewing
- Be aware of back-nav and loss of information

When to Axios

- Use when a **GET** request to another API that doesn't require a token
- Use when a **POST** request to another API that doesn't require a token

- Safer to wrap with React Component state
- Better at handling other request types in addition to Content-Type: Application/JSON

```
axios.get('/user?ID=12345')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

When to use RSAA (Redux Standard API-Calling Actions)

- Must use if data requires access token
- When data needs to be persisted across the app
- Need to block acccess to data if user is unauthorized
- Long upload jobs

Below is an RSAA action

```
//actions/model.js

import { RSAA } from `redux-api-middleware`;
{
  [RSAA]: {
```

```
    endpoint: 'http://www.example.com/api/users',
    method: 'GET',
    types: ['X_REQUEST', 'X_SUCCESS', 'X_FAILURE']
  }
}
```

Here is a matching reducer.

```
//reducers/model.js
export default (state=initialState, action) => {
  switch(action.type) {

    case products.X_REQUEST:
      return state
    case products.X_SUCCESS:
      return {
        ...state,
        product: action.payload
      }
    case products.X_FAILURE:
      return {
        ...state,
        product: action.payload
      }
      ...
```

For the retamining of the redux steps follow this guide.