Introduction to ReactJS Workshop

Prerequisites

1. Laptop (Windows or Mac OSX)
2. Browser (recommend latest Chrome or Firefox)
3. Node.js
    a. Install via website: https://nodejs.org/en/download/
    b. Install via Package Managers: https://nodejs.org/en/download/package-manager/
4. create-react-app
    a. Install via npm: https://github.com/facebookincubator/create-react-app#getting-started
5. Code Editor or IDE
    a. VS Code: https://code.visualstudio.com/download
    b. Atom: https://atom.io/
    c. Brackets: http://brackets.io/
    d. WebStorm: https://www.jetbrains.com/webstorm/

Workshop Agenda

Part 1

1. Introductions
2. Workshop Goals
3. Introduction to ReactJS Presentation

Part 2

1. Create Application using Create React App
2. Install Dependencies
3. Application Review
4. Components
5. Props and State
6. Container Components
7. Routing

Part 3

1. Sample Data
2. Fetch Projects

3. Projects Table
4. Filter Projects Table
5. Install Developer Dependencies
6. Linting and Formatting

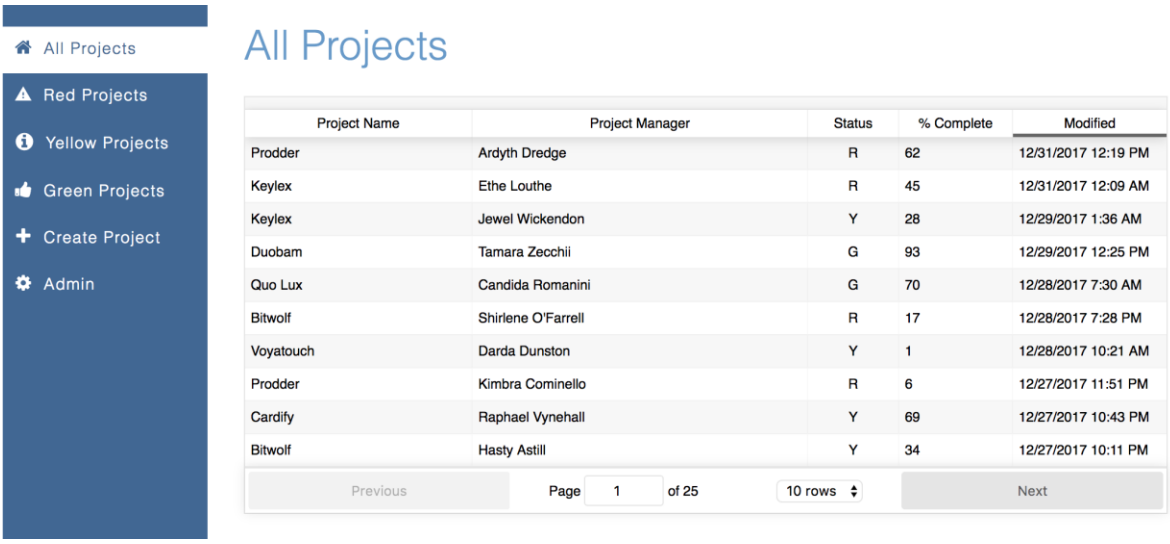
Part 1

Workshop Goals

1. Introduction to ReactJS
2. Create enterprise application
3. Use third-party components for integration
4. Discuss real-world development methods

Application Review

1. We will be building a Project Portfolio application.
2. This application has been built multiple times for different clients with different sets of functionality.



3. Features
    a. Sidebar: Allow the user to quickly search between different types of content. All Projects, Red, Yellow, or Green projects, Create Project, or Admin.
    b. Dashboard: Allow the user to visualize content.
    c. Table: Allow the user to view project information.

Part 2

Create Application

1. Open Terminal or Command Prompt
2. Navigate to location where you want to install project. I usually use Projects.
3. Run the following command.

```
create-react-app reactjs.workshop
```

4. Navigate into project folder

```
cd reactjs.workshop
```

5. Run one of the following commands to run application.

```
npm start
yarn start
```

Install Dependencies

1. Third-party packages can be installed to provide additional capability
2. Use npm or yarn or manage dependencies
3. Managed in package.json
4. Two locations:
    a. dependencies: Required to run. Used by the application during run-time.
    b. dev-dependencies: Are used during development. Examples include: unit tests, TypeScript, linting, formatting.
5. Install dependencies:
    a. axios: used for data fetching
    b. date-fns: used for date formatting
    c. prop-types: used to track Prop Types library
    d. react-icons: popular icon packs library
    e. react-router-dom: for using routes in application library
    f. react-table: for displaying content in a table
    g. styled-components: CSS-in-JS library
6. Run the following commands to install dependencies.

```
npm install axios
npm install date-fns
npm install prop-types
npm install react-icons
npm install react-router-dom
npm install react-table
npm install styled-components

yarn add axios
yarn add date-fns
yarn add prop-types
yarn add react-icons
```

```
yarn add react-router-dom
yarn add react-table
yarn add styled-components
```

Review Application

1. Open source code editor
2. Open project folder

Folder	Description
public	Contains the main HTML file (index.html) and fav.icon. index.html includes a div that the react app will show up inside.
src	Contains all application source code. 1. index.js stores our main Render call from ReactDOM. It imports our App.js component that we start off with and tells React where to render it (remember that div with an id of root?). index.css stores the base styling for our application. 2. App.js is a sample React component called "App" that we get for free when creating a new app. We'll actually be deleting the entire contents of the file and starting over! App.css stores styling targeting that component specifically. Finally, logo.svg is just the React logo. 3. App.test.js is our first set of tests to run against our sample App component that we start off with. We won't be discussing testing during this workshop.
.gitignore	Defines list of folders and files you don't want Git to check into your source code repository (e.g. node_modules, build, coverage, etc.).
package.json	Holds miscellaneous metadata about the application including name, version, authors, license, dependencies, development dependencies, and scripts.
README.md	Markdown file for your project. By default, it is the create-react-app README; however, it is recommended you update for your application.
yarn.lock	A locked list of all dependencies of the application.
node_modules	Directory is where all of the dependencies get built/stored.

Component Overview

1. Open application
2. Remove all code from App.js

3. Add the following code to App.js: Note: class is a reserved word in JavaScript so className is used.

```
import React from 'react';
import './App.css';

const App = (props) => {
  return (<div className="App">Hello World!</div>);
}

export default App;
```

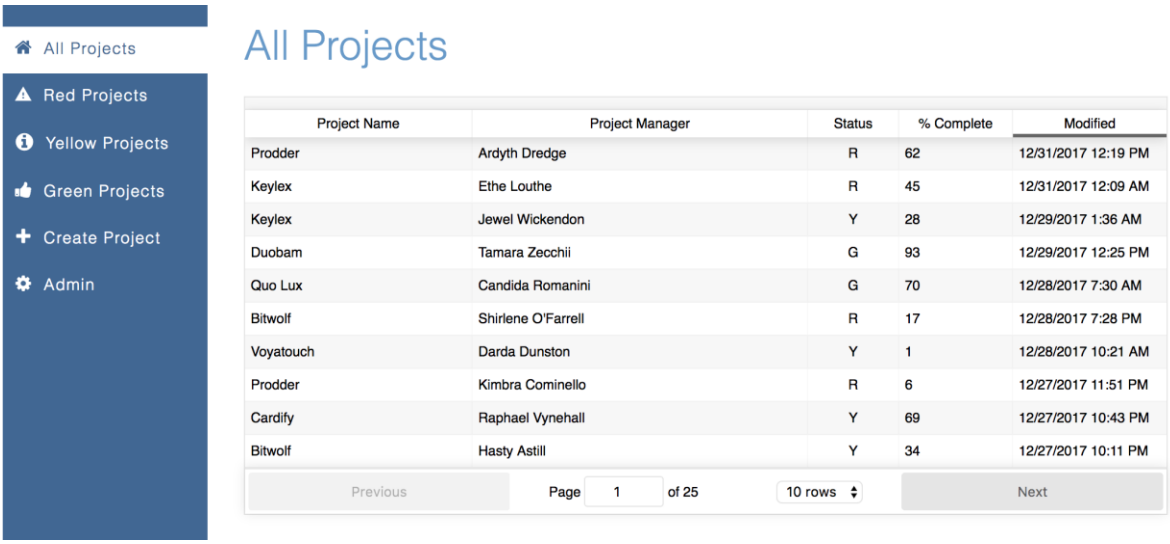
4. Save and review the application running in browser.
5. Update App.css

```
.App {
  border: 2px solid black;
  background: #f5f5f5;
  color: #333;
  margin: 20px;
  padding: 20px;
}
```

6. Save and review the application running in browser.

Application Review

1. We will be building a Project Portfolio application.
2. This application has been built multiple times for different clients with different sets of functionality.
3. Uses flexbox for responsive design.



4. Features
    a. Sidebar: Allow the user to quickly search between different types of content. All Projects, Red, Yellow, or Green projects, Create Project, or Admin.
    b. Dashboard: Allow the user to visualize content.

c. Table: Allow the user to view project information.

Application Components

1. Below is a list of visual components we will create in our application.
    a. Page Title: Display the title of the page. Default to All Pages.
    b. Page Header: Display a page title component. Separated for adding components in the future.
    c. Sidebar: Display list of links to navigate between different types of content.
    d. Table: Display content in tabular format (i.e. columns and rows). Table will allow for paging.
2. Below is a list of non-visual components we will create in our application.
    a. Page Container: Simple div with a max-width and centered in page.
    b. Page Content: Simple div with responsive design attributes attached.
    c. Page Header: Contains the page title.
    d. Page Template: Contains the container, sidebar, page header, and page content components.
    e. Home Page Container: page with dashboard and table.
    f. Create Page Container: page with content to create a project.
    g. Admin Page Container: page with content to administer settings.
    h. Unknown Page Container: page with content display a route was unknown (i.e. page not found).

Create Components #1

1. Create components folder under src.
2. Create PageContainer.js file (under src/components) and enter the following code.

```
import styled from 'styled-components';

const PageContainer = styled.div`
  max-width: 1024px;
  margin: auto;
`;

export default PageContainer;
```

3. Create PageContent.js file (under src/components) and enter the following code.

```
import styled from 'styled-components';

const PageContent = styled.div`
  margin-left: 180px;
  padding-left: 2rem;
  display: ${props => (props.displayBlock ? 'block' : 'flex')};
  flex-wrap: wrap;
  min-height: calc(100vh - 164px);
`;
```

```
export default PageContent;
```

4. Create PageTemplate.js file (under src/components) and enter the following code.

```
import React from 'react';
import PropTypes from 'prop-types';
import PageContainer from './PageContainer';
import PageContent from './PageContent';

const PageTemplate = props => (
  <PageContainer>
    <PageContent displayBlock="true">{props.children}</PageContent>
  </PageContainer>
);

PageTemplate.propTypes = {
  children: PropTypes.node
};

PageTemplate.defaultProps = {
  children: ''
};

export default PageTemplate;
```

5. Update App.js file (under src) with the following code.

```
import React from 'react';
import './App.css';
import PageTemplate from './components/PageTemplate';

const App = props => <PageTemplate>Hello World!</PageTemplate>;

export default App;
```

6. Save all files and review application in browser.

Create Components #2

1. Create Sidebar.js file (under src/components) and enter the following code.

```
import React from 'react';

const Sidebar = (props) => (
  <div>
    sidebar content
  </div>
)

export default Sidebar;
```

2. Create PageTitle.js file (under src/components) and enter the following code.

```
import React from 'react';
import PropTypes from 'prop-types';
import styled from 'styled-components';
```

```
const StyledPageTitle = styled.h1`
  color: #6699cb;
  font-weight: 100;
  font-size: 36px;
  margin-top: 0;
`;

const PageTitle = props => <StyledPageTitle>{props.title}</StyledPageTitle>;

PageTitle.propTypes = {
  title: PropTypes.string.isRequired
};

export default PageTitle;
```

3. Create PageHeader.js file (under src/components) and enter the following code.

```
import React, { Fragment } from 'react';
import PropTypes from 'prop-types';
import styled from 'styled-components';

import PageTitle from './PageTitle';

const StyledHeader = styled.div`
  margin-left: 180px;
  margin-top: 1rem;
  padding-left: 2rem;
  display: flex;
  justify-content: space-between;
  width: calc(100% - 180px);
`;

const PageHeader = props => (
  <StyledHeader>
    <Fragment>
      <PageTitle title={props.title} />
      {props.children}
    </Fragment>
  </StyledHeader>
);

PageHeader.propTypes = {
  title: PropTypes.string.isRequired,
  children: PropTypes.node
};

PageHeader.defaultProps = {
  children: ''
};

export default PageHeader;
```

4. Update PageTemplate.js file (under src/components) with the following code.

```
import React from 'react';
import PropTypes from 'prop-types'
import PageContainer from './PageContainer';
import PageHeader from './PageHeader';
import Sidebar from './Sidebar';
```

```
import PageContent from './PageContent';

const PageTemplate = props => (
  <PageContainer>
    <Sidebar />
    <PageHeader title={props.title} />
    <PageContent displayBlock="true">{props.children}</PageContent>
  </PageContainer>
);

PageTemplate.propTypes = {
  title: PropTypes.string.isRequired,
  children: PropTypes.node
};

PageTemplate.defaultProps = {
  children: ''
}

export default PageTemplate;
```

5. Create Table.js file (under src/components) and enter the following code.

```
import React, { Fragment } from 'react';

const Table = () => <Fragment>table content</Fragment>;

export default Table;
```

6. Save all files and review application in browser. Not much going on yet. ☺

Create Container Components

1. Create HomePageContainer.js file (under src/components) and enter the following code.

```
import React from 'react';
import PropTypes from 'prop-types';
import PageTemplate from './PageTemplate';
import Table from './Table';

const HomePageContainer = props => (
  <PageTemplate title={props.title}>
    <Table title={props.title} />
  </PageTemplate>
);

HomePageContainer.propTypes = {
  title: PropTypes.string.isRequired
};

export default HomePageContainer;
```

2. Create AdminPageContainer.js file (under src/components) and enter the following code.

```
import React from 'react';
import PropTypes from 'prop-types';
import PageTemplate from './PageTemplate';
```

```
const AdminPageContainer = props => (
  <PageTemplate title={props.title}>Admin Page Content</PageTemplate>
);

AdminPageContainer.propTypes = {
  title: PropTypes.string.isRequired
};

export default AdminPageContainer;
```

3. Create CreatePageContainer.js file (under src/components) and enter the following code.

```
import React from 'react';
import PropTypes from 'prop-types';
import PageTemplate from './PageTemplate';

const CreatePageContainer = props => (
  <PageTemplate title={props.title}>Create Page Content</PageTemplate>
);

CreatePageContainer.propTypes = {
  title: PropTypes.string.isRequired
};

export default CreatePageContainer;
```

4. Create UnknownPageContainer.js file (under src/components) and enter the following code.

```
import React from 'react';
import PageTemplate from './PageTemplate';

const UnknownPageContainer = () => {
  const title = '[UNKNOWN PAGE]';
  return (
    <PageTemplate title={title}>Page Not Found</PageTemplate>
  );
};

export default UnknownPageContainer;
```

5. Create index.js file (under src/components) and enter the following code. This file will be used as the entry point to all modules outside of components.

```
import HomePageContainer from './HomePageContainer';
import CreatePageContainer from './CreatePageContainer';
import AdminPageContainer from './AdminPageContainer';
import UnknownPageContainer from './UnknownPageContainer';

export { HomePageContainer, CreatePageContainer, AdminPageContainer,
UnknownPageContainer };
```

6. Update App.js file (under src) with the following code.

```
import React from 'react';
import './App.css';
import { HomePageContainer } from './components';
```

```
const App = () => <HomePageContainer />;

export default App;
```

7. Save all files and review application in browser. Notice the warning in the Console?
8. Update App.js file (under src) with the following code.

```
import React from 'react';
import './App.css';
import { HomePageContainer } from './components';

const App = () => <HomePageContainer title="Hello World" />;

export default App;
```

9. Save all files and review the application in browser. Phew! That's better.

Routing

1. Review routing.
2. Application routes.

Route	**Description**
/	Root route page. Display all projects.
/red	Display only red projects page.
/yellow	Display only yellow projects page.
/green	Display only green projects page.
/create	Display create project page.
/admin	Display admin page.
<<unknown>>	Page not found.

3. Update Sidebar.js file (under src/components) with the following code. Best to copy from Github.
   https://raw.githubusercontent.com/spietrek/workshop.reactjs.1/master/src/components/Sidebar.js

```
import React from 'react';
import { NavLink } from 'react-router-dom';
import styled from 'styled-components';
import HomeIcon from 'react-icons/lib/fa/home';
import WarningIcon from 'react-icons/lib/md/warning';
import InfoCircleIcon from 'react-icons/lib/fa/info-circle';
import ThumbsUpIcon from 'react-icons/lib/fa/thumbs-up';
import PlusIcon from 'react-icons/lib/fa/plus';
```

```
import CogIcon from 'react-icons/lib/fa/cog';

const StyledSidebar = styled.section`
  width: 180px;
  min-height: 100vh;
  position: fixed;
  top: 0;
  background-color: #3F6695;
  font-size: 14px;
  overflow: hidden;
  font-weight: 200;
`;

const NavMenu = styled.ul`
  padding: 0;
  margin: 20px 0;
`;

const NavListItem = styled.li`
  list-style: none;
`;

const NavListLink = styled(NavLink)`
  color: #fff;
  text-decoration: none;
  width: 100%;
  display: inline-block;
  padding: 10px;
  letter-spacing: 1px;

  &:hover,
  &:active,
  &:focus {
    background-color: #fff;
    color: #3F6695;
  }
`;

const HomeIconLink = styled(HomeIcon)`
  font-size: 16px;
  margin-right: 10px;
  padding-bottom: 5px;
`;

const WarningIconLink = styled(WarningIcon)`
  font-size: 16px;
  margin-right: 10px;
  padding-bottom: 5px;
`;

const InfoCircleIconLink = styled(InfoCircleIcon)`
  font-size: 18px;
  margin-right: 10px;
  padding-bottom: 5px;
`;

const ThumbsUpIconLink = styled(ThumbsUpIcon)`
  font-size: 16px;
  margin-right: 10px;
  padding-bottom: 5px;
`;
```

```
const PlusIconLink = styled(PlusIcon)`
  font-size: 16px;
  margin-right: 10px;
  padding-bottom: 5px;
`;

const CogIconLink = styled(CogIcon)`
  font-size: 16px;
  margin-right: 10px;
  padding-bottom: 5px;
`;

const NavListLinkActive = {
  backgroundColor: '#fff',
  color: '#3F6695',
};

const Sidebar = () => (
  <StyledSidebar>
    <NavMenu>
      <NavListItem>
        <NavListLink
          activeStyle={NavListLinkActive}
          to="/"
          className="NavListLink"
          exact
        >
          <HomeIconLink />
          All Projects
        </NavListLink>
      </NavListItem>
      <NavListItem>
        <NavListLink
          activeStyle={NavListLinkActive}
          to="/red"
          className="NavListLink"
        >
          <WarningIconLink />
          Red Projects
        </NavListLink>
      </NavListItem>
      <NavListItem>
        <NavListLink
          activeStyle={NavListLinkActive}
          to="/yellow"
          className="NavListLink"
        >
          <InfoCircleIconLink />
          Yellow Projects
        </NavListLink>
      </NavListItem>
      <NavListItem>
        <NavListLink
          activeStyle={NavListLinkActive}
          to="/green"
          className="NavListLink"
        >
          <ThumbsUpIconLink />
          Green Projects
        </NavListLink>
```

```
      </NavListItem>
      <NavListItem>
        <NavListLink
          activeStyle={NavListLinkActive}
          to="/create"
          className="NavListLink"
        >
          <PlusIconLink />
          Create Project
        </NavListLink>
      </NavListItem>
      <NavListItem>
        <NavListLink
          activeStyle={NavListLinkActive}
          to="/admin"
          className="NavListLink"
        >
          <CogIconLink />
          Admin
        </NavListLink>
      </NavListItem>
    </NavMenu>
  </StyledSidebar>
);

export default Sidebar;
```

4. Create AppRoutes.js file (under src) and enter the following code. Best to copy from Github. https://raw.githubusercontent.com/spietrek/workshop.reactjs.1/master/src/AppRoutes.js

```
import React from 'react';
import { HashRouter, Route, Switch } from 'react-router-dom';
import {
  HomePageContainer,
  CreatePageContainer,
  AdminPageContainer,
  UnknownPageContainer
} from './components';

const renderMergedProps = (component, ...rest) => {
  const finalProps = Object.assign({}, ...rest);
  return React.createElement(component, finalProps);
};

const PropsRoute = (
  { component, ...rest } // eslint-disable-line react/prop-types
) => (
  <Route
    {...rest}
    render={routeProps => renderMergedProps(component, routeProps, rest)}
  />
);

const AppRoutes = props => (
  <HashRouter>
    <div>
      <Switch>
        <PropsRoute
```

```
          exact
          name="AllProjectsPage"
          title="All Projects"
          path="/"
          component={HomePageContainer}
          {...props}
        />
        <PropsRoute
          name="RedProjectsPage"
          title="Red Projects"
          path="/red"
          component={HomePageContainer}
          {...props}
        />
        <PropsRoute
          name="YellowProjectsPage"
          title="Yellow Projects"
          path="/yellow"
          component={HomePageContainer}
          {...props}
        />
        <PropsRoute
          name="GreenProjectsPage"
          title="Green Projects"
          path="/green"
          component={HomePageContainer}
          {...props}
        />
        <PropsRoute
          name="CreatePage"
          title="Create Project"
          path="/create"
          component={CreatePageContainer}
          {...props}
        />
        <PropsRoute
          name="AdminPage"
          title="Admin Settings"
          path="/admin"
          component={AdminPageContainer}
          {...props}
        />
        <PropsRoute component={UnknownPageContainer} {...props} />
      </Switch>
    </div>
  </HashRouter>
);

export default AppRoutes;
```

5. Update App.js file (under src) with the following code.

```
import React from 'react';
import './App.css';
import AppRoutes from './AppRoutes';

const App = (props) => (
  <div>
    <AppRoutes {...props} />
  </div>
```

```
)

export default App;
```

6. Update index.html fiel (under public) to have a document title of “Project Portfolio”.
7. Save all files and review application in browser.

Part 3

Sample Data

1. Need to create sample projects data.
2. Data will include the following columns.

Column Name	Description
id	The project unique ID.
projectName	The name of the project.
projectManager	The project manager name.
overallStatus	The overall status (G, Y, R).
percentageComplete	The project completion percentage (0 – 100).
modifiedDate	The project's last modified date.

3. Review Mockaroo (http://www.mockaroo.com/)
4. Install json-server package globally

```
npm i –g json-server
```

5. Update package.json

```
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject",
    "data-server": "json-server --watch public/db.json --port 3004"
  }
```

6. Create db.json file (under public) and add the following JSON (partial list). Best to pull from Github. https://raw.githubusercontent.com/spietrek/workshop.reactjs.2/master/public/db.json

```
{
  "projects": [
    {
      "id": 1,
      "projectName": "Stringtough",
      "projectManager": "Gwyn Tretwell",
      "overallStatus": "G",
      "percentageComplete": 65,
      "modifiedDate": "2017-12-16 22:59:13"
    },
    {
      "id": 2,
```

```
    "projectName": "Bigtax",
    "projectManager": "Dore Birtchnell",
    "overallStatus": "Y",
    "percentageComplete": 5,
    "modifiedDate": "2018-01-23 20:19:50"
  },
  {
    "id": 3,
    "projectName": "Voltsillam",
    "projectManager": "Cammi Steggals",
    "overallStatus": "R",
    "percentageComplete": 99,
    "modifiedDate": "2017-10-07 14:13:25"
  },
  {
    "id": 4,
    "projectName": "Regrant",
    "projectManager": "Bengt Paulsen",
    "overallStatus": "G",
    "percentageComplete": 49,
    "modifiedDate": "2018-01-06 11:13:10"
  },
  {
    "id": 5,
    "projectName": "Stronghold",
    "projectManager": "Jordana Cowie",
    "overallStatus": "Y",
    "percentageComplete": 86,
    "modifiedDate": "2017-11-05 17:05:59"
  },
  {
    "id": 6,
    "projectName": "Aerified",
    "projectManager": "Janessa Benduhn",
    "overallStatus": "Y",
    "percentageComplete": 3,
    "modifiedDate": "2017-12-09 11:21:47"
  },
  {
    "id": 7,
    "projectName": "Cardify",
    "projectManager": "Raphael Vynehall",
    "overallStatus": "Y",
    "percentageComplete": 69,
    "modifiedDate": "2017-12-27 22:43:28"
  },
  {
    "id": 8,
    "projectName": "Prodder",
    "projectManager": "Cassie Geeritz",
    "overallStatus": "Y",
    "percentageComplete": 62,
    "modifiedDate": "2017-11-28 20:46:08"
  },
  {
    "id": 9,
    "projectName": "Bamity",
    "projectManager": "Danette Tettley",
    "overallStatus": "Y",
    "percentageComplete": 22,
    "modifiedDate": "2018-01-05 16:54:32"
```

```
    },
    {
      "id": 10,
      "projectName": "Pannier",
      "projectManager": "Dimitri Margett",
      "overallStatus": "G",
      "percentageComplete": 65,
      "modifiedDate": "2018-01-22 20:13:49"
    }
  ]
}
```

7. Review db.json
8. Run command

```
yarn data-server
```

9. Review URL's in browser.
    a. http://localhost:3004/projects
    b. http://localhost:3004/projects/1

Fetch Projects

1. Create api folder under src.
2. Create fetchProjects.js file (under src/api) and enter the following code.

```
import axios from 'axios';
import format from 'date-fns/format';

const getFormattedDate = date => format(new Date(date), 'MM/DD/YYYY hh:mm A');
const getEditUrl = id => `/#/edit?projectId=${id}`;

const formatProjectsData = data => {
  const items = data.map(item => {
    const modifiedDate = getFormattedDate(item.modifiedDate);
    const editUrl = getEditUrl(item.id);
    return { ...item, modifiedDate, editUrl };
  });
  return items;
};

const fetchProjects = () => {
  const request = 'http://localhost:3004/projects';
  return axios(request)
    .then(response => {
      const { data } = response;
      console.log(data);
      return formatProjectsData(data);
    })
    .catch(error => {
      console.log(error);
      return [];
    });
};

export default fetchProjects;
```

Projects Table

1. Update HomePageContainer.js file (under src/components) with the following code.

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import PageTemplate from './PageTemplate';
import Table from './Table';
import fetchProjects from '../api/fetchProjects';

class HomePageContainer extends Component {
  static propTypes = {
    title: PropTypes.string.isRequired
  };

  constructor(props) {
    super(props);
    const { title } = props;
    this.state = {
      data: [],
      title
    };
  }

  componentDidMount() {
    this.fetchProjects();
  }

  fetchProjects = () => {
    fetchProjects().then(serverPlans => {
      this.setState({
        data: serverPlans
      });
    });
  };

  render() {
    const { title, data } = this.state;
    return (
      <PageTemplate title={title}>
        <Table title={title} data={data} />
      </PageTemplate>
    );
  }
}

export default HomePageContainer;
```

2. Update Table.js file (under src/components) with the following code.

```
import React, { Fragment } from 'react';
import PropTypes from 'prop-types';
import ReactTable from 'react-table';
import styled from 'styled-components';
import 'react-table/react-table.css';

const ExtendedReactTable = styled(ReactTable)`
```

```
  font-size: 12px;
`;

const TableResult = ({ data }) => (
  <Fragment>
    <ExtendedReactTable
      data={data}
      noDataText="No projects available"
      columns={[]}
      defaultSorted={[
        {
          id: 'modifiedDate',
          desc: true
        }
      ]}
      defaultPageSize={10}
      className="-striped -highlight react-table-container"
    />
  </Fragment>
);

const Table = props => (
  <Fragment>
    <TableResult {...props} />
  </Fragment>
);

TableResult.propTypes = {
  data: PropTypes.arrayOf(
    PropTypes.shape({
      projectName: PropTypes.string,
      projectManager: PropTypes.string,
      overallStatus: PropTypes.string,
      percentageComplete: PropTypes.number,
      modifiedDate: PropTypes.string
    })
  )
};

TableResult.defaultProps = {
  data: []
};

export default Table;
```

3. Save all files and review application in browser.
4. Create TableColumns.js file (under src/components) and add the following code.

```
const TableColumns = [
  {
    columns: [
      {
        Header: 'Project Name',
        accessor: 'projectName',
        maxWidth: 200
      },
      {
        Header: 'Project Manager',
        accessor: 'projectManager'
      },
      {
```

```
        Header: 'Status',
        accessor: 'overallStatus',
        maxWidth: 80,
        style: {
          textAlign: 'center'
        }
      },
      {
        Header: '% Complete',
        accessor: 'percentageComplete',
        maxWidth: 100
      },
      {
        Header: 'Modified',
        accessor: 'modifiedDate',
        maxWidth: 135
      }
    ]
  }
];

export default TableColumns;
```

5. Update Table.js file (under src/components) with the following code.

```
import React, { Fragment } from 'react';
import PropTypes from 'prop-types';
import ReactTable from 'react-table';
import styled from 'styled-components';
import 'react-table/react-table.css';
import TableColumns from './TableColumns';

const ExtendedReactTable = styled(ReactTable)`
  font-size: 12px;
`;

const TableResult = ({ data }) => (
  <Fragment>
    <ExtendedReactTable
      data={data}
      noDataText="No projects available"
      columns={TableColumns}
      defaultSorted={[
        {
          id: 'modifiedDate',
          desc: true
        }
      ]}
      defaultPageSize={10}
      className="-striped -highlight react-table-container"
    />
  </Fragment>
);

const Table = props => (
  <Fragment>
    <TableResult {...props} />
  </Fragment>
);

TableResult.propTypes = {
```

```
  data: PropTypes.arrayOf(
    PropTypes.shape({
      projectName: PropTypes.string,
      projectManager: PropTypes.string,
      overallStatus: PropTypes.string,
      percentageComplete: PropTypes.number,
      modifiedDate: PropTypes.string
    })
  )
};

TableResult.defaultProps = {
  data: []
};

export default Table;
```

6. Save all files and review application in browser.

Filter Projects Table

1. Update HomePageContainer.js file (under src/components) with the following code.

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import PageTemplate from './PageTemplate';
import Table from './Table';
import fetchProjects from '../api/fetchProjects';

class HomePageContainer extends Component {
  static propTypes = {
    location: PropTypes.shape({
      hash: PropTypes.string,
      pathname: PropTypes.string,
      search: PropTypes.string,
      state: PropTypes.string
    }).isRequired,
    title: PropTypes.string.isRequired
  };

  constructor(props) {
    super(props);
    const { title } = props;
    this.state = {
      data: [],
      filteredData: [],
      title
    };
  }

  componentDidMount() {
    this.fetchProjects();
  }

  componentDidUpdate(prevProps) {
    if (this.props.location !== prevProps.location) {
      this.onRouteChanged();
    }
  }

  onRouteChanged = () => {
```

```
    const { location } = this.props;
    const { data } = this.state;
    const newFilter = this.getFilterValue(location);
    const filteredData =
      newFilter === '*'
        ? data
        : data.filter(project => project.overallStatus === newFilter);
    this.setState({
      filteredData
    });
  };

  getFilterValue = location => {
    switch (location.pathname) {
      case '/red':
        return 'R';
      case '/yellow':
        return 'Y';
      case '/green':
        return 'G';
      default:
        return '*';
    }
  };

  fetchProjects = () => {
    fetchProjects().then(serverPlans => {
      this.setState({
        data: serverPlans,
        filteredData: serverPlans
      });
    });
  };

  render() {
    const { title, filteredData } = this.state;
    return (
      <PageTemplate title={title}>
        <Table title={title} data={filteredData} />
      </PageTemplate>
    );
  }
}

export default HomePageContainer;
```

2. Save all files and review application in browser.

Install Developer Dependencies

1. Third-party packages can be installed to provide additional capability
2. Use npm or yarn or manage developer dependencies
3. Managed in package.json
4. Two locations:
    a. dependencies: Required to run. Used by the application during run-time.
    b. dev-dependencies: Are used during development. Examples include: unit tests, TypeScript, linting, formatting.

5. Install developer dependencies:
    a. babel-eslint: babel-eslint allows you to lint ALL valid Babel code with the fantastic ESLint.
    b. eslint: goal is to provide a pluggable linting utility for JavaScript
    c. eslint-config-airbnb: provides Airbnb's .eslintrc as an extensible shared config
    d. eslint-config-prettier: turns off all ESLint rules that are unnecessary or might conflict with prettier
    e. eslint-plugin-import: goal is to support linting of ES2015+ (ES6+) import/export syntax, and prevent issues with misspelling of file paths and import names.
    f. eslint-plugin-jsx-a11y: static AST checker for a11y rules on JSX elements
    g. eslint-plugin-prettier: ESLint plugin for prettier formatting
    h. eslint-plugin-react: React specific linting rules for ESLint
    i. prettier: is an opinionated code formatter
    j. prettier-eslint: formats your JavaScript using prettier followed by eslint --fix
6. Run the following commands to install dependencies.

```
npm install –D babel-eslint
npm install –D eslint
npm install –D eslint-config-airbnb
npm install –D eslint-config-prettier
npm install –D eslint-plugin-import
npm install –D eslint-plugin-jsx-a11y
npm install –D eslint-plugin-prettier
npm install –D eslint-plugin-react
npm install –D prettier
npm install –D prettier-eslint

yarn add –D babel-eslint
yarn add –D eslint
yarn add –D eslint-config-airbnb
yarn add –D eslint-config-prettier
yarn add –D eslint-plugin-import
yarn add –D eslint-plugin-jsx-a11y
yarn add –D eslint-plugin-prettier
yarn add –D eslint-plugin-react
yarn add –D prettier
yarn add –D prettier-eslint
```

Linting and Formatting

1. Update .eslintrc

```
{
  "parser": "babel-eslint",
  "extends": ["airbnb", "prettier"],
  "plugins": ["react", "prettier"],
  "parserOptions": {
    "ecmaVersion": 6,
    "sourceType": "module",
    "ecmaFeatures": {
      "jsx": true
    }
  },
  "env": {
```

```
    "browser": true,
    "es6": true
  },
  "rules": {
    "react/jsx-filename-extension": [1, { "extensions": [".js", ".jsx"] }],
    "no-debugger": 0,
    "no-console": 0
  }
}
```

2. Show demo of add img without alt tex.