Front End Development

by : Travis Quigg

What is front end development?

- Html, CSS?
- Deliver the GUI for the user

Technologies

- React, Redux, Angular, Vue(new)
- Connect the dots: Axios, Angular has its own way of doing things
- CSS, and to make life easier BootStrap (replaces flexbox/grid)

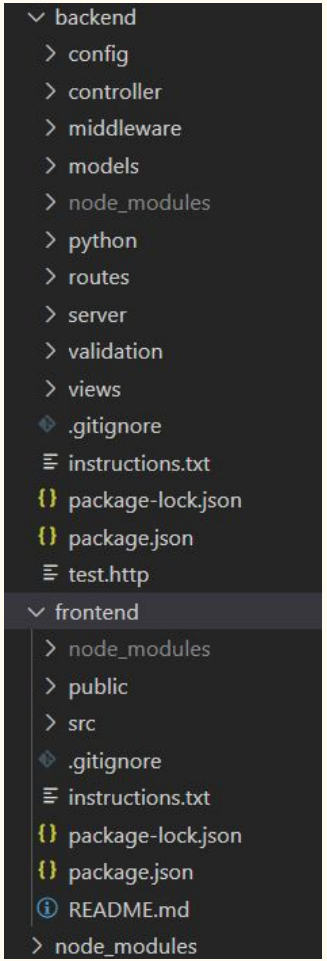
What we will focus on

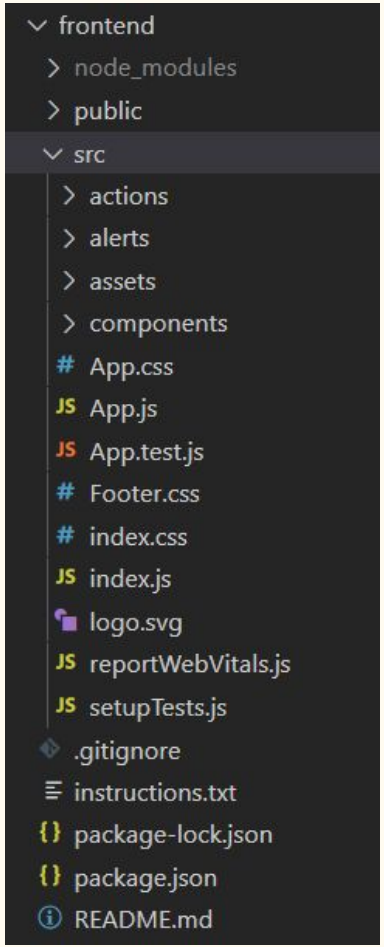
- React
- Large career prospects for React/Redux developers

What is React/Redux

- Part of the MERN stack
- React is made up of components
- Redux is a state handler eg: Amazon's add to cart feature

Structure


backend
config
controller
middleware
models
node_modules
python
routes
server
validation
views
.gitignore
instructions.txt
package-lock.json
package.json
test.http
frontend
node_modules
public
src
.gitignore
instructions.txt
package-lock.json
package.json
README.md
node_modules


frontend
node_modules
public
src
actions
alerts
assets
components
App.css
App.js
App.test.js
Footer.css
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
instructions.txt
package-lock.json
package.json
README.md

What is a component?

- Html can be made into blocks to perform iterations or to create modularity.
- For example: navBar.jsx is a component and will be called through the app.js file in the root directory of the frontend or my-app folder.
- Make a .jsx file and place it into your components folder

- **frontend/src/components/filename.jsx**

A simple component

-class based/ normal function

src:ReactJS Class Based Components - GeeksforGeeks

```
import React, {component} from 'react';
import '../assets/helloWorld.css'


//Always capitalize the First letter. Case sensitive
const HelloWorld = () =>{

    return(
        <div className="helloworld">

            <h1>Hello World!</h1>

        </div>
    )

}

export default HellowWorld;
```

javascript

```
import React from "react";

class App extends React.Component {
  render() {
    return <h1>GeeksForGeeks</h1>;
  }
}

export default App;
```

Output:

To break it down

- Cd ../ moves out of the folder
- Always do return(HTMLcode) when creating modularity
- Creating a const makes it immutable
- Components are separated into a components folder.
- Finally export default <filename> to allow app.js to construct it

App.js structure

- Npm/yarn i/install
- React-router-dom

rontend > src > JS index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

ontend > src > JS App.js > ...

```
import './App.css';
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Footer from './components/Footer.jsx';
import Login from './components/Login.jsx';

//Routes replaces switches in new update
//if you want an element in each page place outside of Routes
//Remember React is case sensitive capitalize, capitalize, Capitalize

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path= {"/login"} element=
        {
          <div className="App-body-login">
            <Login />
          </div>
        }/>
      </Routes>

      <Footer />
    </BrowserRouter>
  );
}

export default App;
```

Props

```
function Tool(props) {
  const name = props.name;
  const tool = props.tool;
    return (
      <div>
        <h1>My name is {name}.</h1>
        <p>My favorite design tool is {tool}.</p>
      </div>
    );
}

export default Tool
```

Another Example

Src:

How to pass props in React (koalatea.io)

```
function App() {
  const blogPost = {
    title: 'How to pass prop to components',
    description: 'Your component library for ...',
  };
  return (
    <div>
      <BlogCard title={blogPost.title} description={blogPost.description} />
    </div>
  );
}

function BlogCard({ title, description }) {
  return (
    <div>
      <h1>{title}</h1>
      <p>{description}</p>
    </div>
  );
}
```

A different way

src:[How to Use Props in React (freecodecamp.org)]

```
function Tool({name, tool}) {

    return (
      <div>
        <h1>My name is {name}.</h1>
        <p>My favorite design tool is {tool}.</p>
      </div>
    );

  }

  Tool.defaultProps = {
    name: "Designer",
    tool: "Adobe XD"
  }
export default Tool
```

States

- Managing data

```
const [loggingIn, setLoggingIn] = useState(true);
```

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }
```

Src: https://reactjs.org/docs/state-and-lifecycle.html

Uses

- Carousel
- Axios
- Many more

```
class HomeCarousel extends Component{

    constructor(){
        super();
        this.state = {
            pictures:[] ,
            usernames:[]
        }
    }
    componentDidMount(){
        fetch('https://randomuser.me/api/?results=7')
          .then(results => {
           return results.json()
          })
          .then(data => {
            let pictures = data.results.map((pic) => {
              return (
                <div className = "userThumbResults" key = {pic.results}>
                  <img className = "userPicThumb" src = {pic.picture.thumbnail} />
                 <h3 className = "featUserNames">{pic.name.first} {pic.name.last}</h3>
                  <h3 className = "picTitle">Title</h3>
                </div>
              )
            })
            this.setState({
              pictures: pictures
            })
            console.log("state", this.state.pictures)
          })
      }
    render(){
    return(
    <Carousel autoPlay>
```

Difference between props and states

- Props pass data
- States manage data

Event Handling

```
const [loggingIn, setLoggingIn] = useState(true);
```

```
if (loggingIn) {
    // axios
    axios.post('http://localhost:5000/app/login', data)
    .then(res => console.log(res.data))
    .catch(err => console.log(err));
    console.log("logging in");
}
else {
```

```
<input type = "submit" value ="Login" onClick={e => setLoggingIn(true)}>
```

Props w/ Events

-Some sneak peak tic tac toe code

```
class Board extends React.Component {
  renderSquare(i) {
    return (
      <Square
        value={this.props.squares[i]}
        onClick={() => this.props.onClick(i)}
      />
    );
  }

  render() {
    return (
      <div>
        <div className="board-row">
          {this.renderSquare(0)}
          {this.renderSquare(1)}
          {this.renderSquare(2)}
        </div>
        <div className="board-row">
          {this.renderSquare(3)}
          {this.renderSquare(4)}
          {this.renderSquare(5)}
        </div>
        <div className="board-row">
          {this.renderSquare(6)}
          {this.renderSquare(7)}
          {this.renderSquare(8)}
        </div>
      </div>
    );
  }
}
```

Links

- `<Link to={"/login"} className="loginButton"> Login </Link>`
- Links are important when you need to switch to different pages
- Eg: www.wordmouth.org/login
- You can also switch between links via app.js using Routes instead of the old style of Switches.
- Remember if you want every page to have a certain component like navbar/footer then you need to call it outside of Routes

Fragmenting

Fragments let you group a list of children without adding extra nodes to the DOM.

```
render() {
  return (
    <React.Fragment>
      <ChildA />
      <ChildB />
      <ChildC />
    </React.Fragment>
  );
}
```

Usage

```
class Table extends React.Component {
  render() {
    return (
      <table>
        <tr>
          <Columns />
        </tr>
      </table>
    );
  }
}
```

```
class Columns extends React.Component {
  render() {
    return (
      <div>
        <td>Hello</td>
        <td>World</td>
      </div>
    );
  }
}
```

results

```
<table>
  <tr>
    <div>
      <td>Hello</td>
      <td>World</td>
    </div>
  </tr>
</table>
```

Solution

```
class Columns extends React.Component {
  render() {
    return (
      <React.Fragment>
        <td>Hello</td>
        <td>World</td>
      </React.Fragment>
    );
  }
}
```

```
class Columns extends React.Component {
  render() {
    return (
      <>
        <td>Hello</td>
        <td>World</td>
      </>
    );
  }
}
```

Cool Prop Use with Fragmenting

```jsx
function Glossary(props) {
  return (
    <dl>
      {props.items.map(item => (
        // Without the `key`, React will fire a key warning
        <React.Fragment key={item.id}>
          <dt>{item.term}</dt>
          <dd>{item.description}</dd>
        </React.Fragment>
      ))}
    </dl>
  );
}
```

Mobile responsive navbar using fragmenting

```
const Navbar = () => {
    const isMobile = useMediaQuery({ maxWidth: 550 });
    return (
```

```
<div className="rightSide">
    {!isMobile &&
        <React.Fragment>
            <Link to={"/about"} className="aboutButton"> About </Link>
            <Link to={"/login"} className="loginButton"> Login </Link>
        </React.Fragment>
    }
    {isMobile &&
        <React.Fragment>
            <FontAwesomeIcon icon={faBars} className="hamburger" />
        </React.Fragment>
    }
```

Additional useful tools

- promises/try catch = exception handling (very useful for debugging your code)
- super() calls the constructor of the parent class
- Be familiar with json to perform handshakes
- Reducers, useEffect, Forms, and many more

Practice

Make a Tic Tac Toe game that logs whose turn it is and declares a winner at the end.