Otter Conner Donner

Monday, February 17, 2020 12:33 PM

Create app contrived from redux
Idea of app is to "help otter manage his OCD"
    -you are otter and you have compulsions you must fulfill
Compulsions are randomly generated. "monkey see, monkey do"
Website interface. Game.
Text based. Actions will trigger text notifications
Have a box with a log, kind of like terminal

<u>Code Structure</u>
"App" component is high level smart component. All other components are dumb.
Other components are the square component and knight component.

<u>App</u>
In other words, the board. Returns a div with the 64 squares inside, one of which has a knight. display is flex, so squares are put next to each other, and flexwrap: true, so row-by-row is filled

<u>Square</u>
returns a div with a width of 12.5% of parent container, so there should be 8 squares per row. div height is equal to width. Takes in several Props
- isBlack
- hasKnight: if true, renders a knight inside square. else, empty
- refID: id of react-dnd drag/drop

Components

<u>Knight</u>
usually would not be it's own component. But since this will be dragged, knight is its own component.
props:
- dragID

<u>Redux Store</u>
- state
  - position: position of the knight
    - positionReducer
      - responds to actiions of type: "SET_POSITION"
      - updates position to payload of accepted actions
- actions
  - types
    - SET_POSITION
  - action creators
    - setPosition([x,y])
- component connections
  - App
    - state
      - position
    - actions
      - setPosition

Redux

```
const [{ isOver }, drop] = useDrop({
    accept: ItemTypes.KNIGHT,
    drop: () => this.props.setPosition([x, y]),
    collect: monitor => ({
      isOver: monitor.isOver()
    })
  });
```

old drop initialization for square component

```
const [{ isDragging }, drag] = useDrag({
    item: { type: ItemTypes.KNIGHT },
    collect: monitor => ({
      isDragging: monitor.isDragging()
    })
  });
```

old drag initialization for knight component