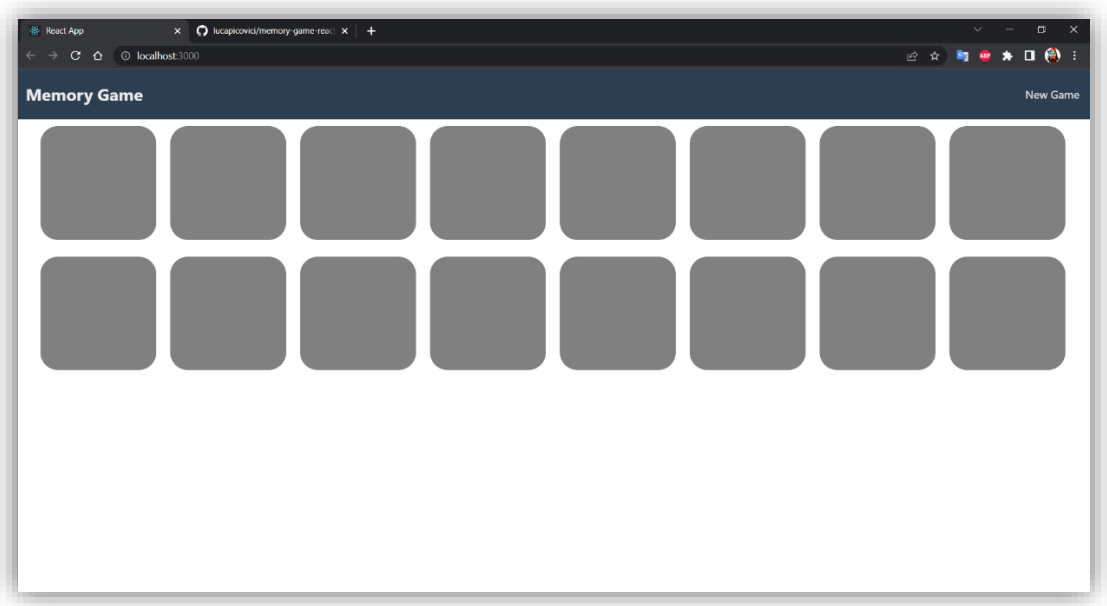
Luca Picovici

Proof of Concept Memory Game using React

For this project I've created a memory game with cards using JavaScript and React. The game is simple, the goal is to click on a card and then on another based on the color, until all cards are matched. Clicking on a New Game will reset and shuffle all cards.

The array of cards is defined as a JS object with the id, card state (hiding, showing or matching) and the background color.

At every new game, the cards array is shuffled and the card states are set to "hiding". The array is saved into the React's application state using "this.setState()" function.




```
render() {
  const cards = this.state.cards.map((card) => (
    <Card
      key={card.id}
      showing={card.cardState !== CardState.HIDING}
      backgroundColor={card.backgroundColor}
      onClick={() => {this.handleClick(card.id)}}
    />
  ));

  return(
    <div>
      <Navbar onNewGame={this.handleNewGame}/>
      {cards}
    </div>
  );
}
}
```

The React application is made from one parent component, called “App” which contains another large component “MemoryGame” which contains the Navbar and the cards themselves.

The card component, the navbar and the App itself have their own CSS file with the design choices.

Below is the MemoryGame React component which is a class that extends “Component”. The constructor is necessary to take any props from the parent components and to initialize the state.

```
export default class MemoryGame extends Component {
  constructor(props) {
    super(props);

    // The cards that we will use for our state.
    let cards = [ ···
    ];
    cards = shuffle(cards);
    this.state = {cards, noClick: false};

    this.handleClick = this.handleClick.bind(this);
    this.handleNewGame = this.handleNewGame.bind(this);
  }

  handleClick(id) {···
  }
  handleNewGame() {···
  }

  render() {···
  }
}
```

src > JS Card.js > ...

```
1   import React from 'react';
2   import PropTypes from 'prop-types';
3   import './Card.css';
4
5   const Card = (props) => {
6     let style = {};
7     if (props.showing) {
8       style.backgroundColor = props.backgroundColor;
9     }
10
11    return (
12      <div
13        onClick={props.onClick}
14        className='card-container'
15        style={style}
16      />
17    );
18  }
19
20  Card.propTypes = {
21    showing: PropTypes.bool.isRequired,
22    backgroundColor: PropTypes.string.isRequired,
23    onClick: PropTypes.func.isRequired
24  };
25
26  export default Card;
```

The Card component renders an empty div which is later styled by the CSS to make it a square of a certain colour. The div receives three props, the function onClick which deals with the logic of what happens when the user clicks, the prop className necessary for styling and the style prop.

In React one can impose requirements on the props, like in the Card component where the prop “showing” has to be required and a Boolean, the prop “backgroundColor” which has to be a string and the prop “onClick” which has to be a function to make sure the application works correctly and for easier future maintenance. All this information is then exported as a single component, ready to be included in other components.