React - Optimize Performance

Case 1 - Preventing Wasted Renders with Complex Props

Create new object reference before delete a key.

```
function handleDelete(card) {
  const clonedCards = { ...cards };
  delete clonedCards[card.id];

  setCards(clonedCards);
}
```

Class

```
export class Summary extends React.Component {
  shouldComponentUpdate(nextProps) {
    const oldKeys = Object.keys(this.props.cards);
    const newKeys = Object.keys(nextProps.cards);

    console.log({
      oldLength: oldKeys.length,
      newLength: newKeys.length
    });

    return oldKeys.length !== newKeys.length;
  }
}
```

Function

```
export const Summary = React.memo(function Summary(props) {
  const cards = Object.values(props.cards);

  const distances = { max: 0, min: 100000 };
  cards.forEach(currentCard => {
    cards.forEach(compareCard => {
      if (compareCard === currentCard) {
        return;
      }
      const distance = levenshtein(currentCard.label, compareCard.label);

      distances.max = Math.max(distances.max, distance);
      distances.min = Math.min(distances.min, distance);
    });
  });

  return (
```

```
  );
}, (p1, p2) => Object.keys(p1.cards).length === Object.keys(p2.cards).length;
```

Case 2 - Preventing Wasted Renders in a Simple Component

Class

```
export class AddButton extends React.PureComponent {
  render() {
    const { onClick } = this.props;

    return (
      <button
        onClick={onClick}
```

Function

Before

```
<AddButton onClick={() => setIsAddOpen(true)} />
```

```
const showDialog = useCallback(() => setIsAddOpen(true), []);
```

After

```
<AddButton onClick={showDialog} />
```

Caching Expensive Operation Results

```
useMemo(
```

Lazy Loading Components

```
useCallback,
lazy,
Suspense,
} from "react";
```

Component:

```
const AddModal = lazy(() => import("./AppModal"));
```

Fallback:

```
const ModalLoader = () => <div className="Modal-loader">Loading</div>;
```

```jsx
{isAddOpen && (
  <Suspense fallback={<ModalLoader />}>
    <AddModal
      isOpen={isAddOpen}
      onClose={() => setIsAddOpen(false)}
      onAdd={cardText => {
        const updatedCards = positionCards(
          addCard(cards, cardText),
          width,
          height
        );
        setCards(updatedCards);
      }}
    />
  </Suspense>
```