Memoization

Shawn Leberknight

Software Engineer - **Somo Global**

So what is it?

Memoization is an optimization technique used in many programming languages to reduce the number of redundant, expensive function calls.

How does it work?

- Caching the values that the function returns after its initial execution.
- If the input values remain the same, the memoized function returns the cached response.
- The program does not have to recalculate anything.

Simple Example

Before memoization

```
function sum(a, b) {
 console.log('add')
 return a + b
}
console.log(sum(1, 2))
console.log(sum(1, 2))

// will output the following:
// add
// 3
// add
// 3
```

After memoization

```
function sum(a, b) {
 console.log('add')
 return a + b
}

const memSum = memoize(sum)
console.log(memSum(1, 2))
console.log(memSum(1, 2))

// will output the following:
// add
// 3
// 3
```

Some live coding!!!

Some other things to remember

Network & database calls

- Be wary of using memoization for this and know the possible issues
- Expire cache after given period of time
- Clear cache manually if needed

React.memo & useMemo

React.memo

```
const MyComponent = React.memo(function MyComponent(props) {
  /* render using props */
})
```

React.memo example

```
class CounterComponent extends Component {
  state = { buttonPressedCount: 0 };
  render() {
    const { buttonPressedCount } = this.state;
    return (
      <div className="new-component">
        <h4>Button Pressed Count: {buttonPressedCount}</h4>
        <button
          onClick={() =>
            this.setState({ buttonPressedCount: buttonPressedCount + 1 })
          }
        >
          Increase Count
        </button>
        <Banner type="info" />
      </div>
    );
  }
}
```

Banner component

```
const Banner = props => {
  const { type } = props;

  if (type === "info") {
    return <div className="info-banner">I am an info banner</div>;
  }
}
```

Banner component with memo

```
const Banner = React.memo(props => {
  const { type } = props

  if (type === "info") {
    return <div className="info-banner">I am an info banner</div>
  }
})
```

useMemo

```
const memoizedValue = useMemo(() => {
  return computeExpensiveValue(a, b), [a, b]
})
```

useMemo example

```
const List = useMemo(
  () =>
  listOfItems.map(item => ({
    ...item,
    itemProp1: expensiveFunction(props.first),
    itemProp2: anotherPriceyFunction(props.second)
  })),
  [listOfItems]
)
```

Resources

- https://dev.to/nas5w/what-is-memoization-4lod
- https://medium.com/better-programming/react-memo-vs-memoize-71f85eb4e1a
- https://codeburst.io/understanding-memoization-in-3-minutes-2e58daf33a19
- https://reactjs.org/docs/hooks-reference.html#usememo
- https://reactjs.org/docs/react-api.html#reactmemo
- https://www.digitalocean.com/community/tutorials/react-usememo
- https://www.digitalocean.com/community/tutorials/react-learning-react-memo