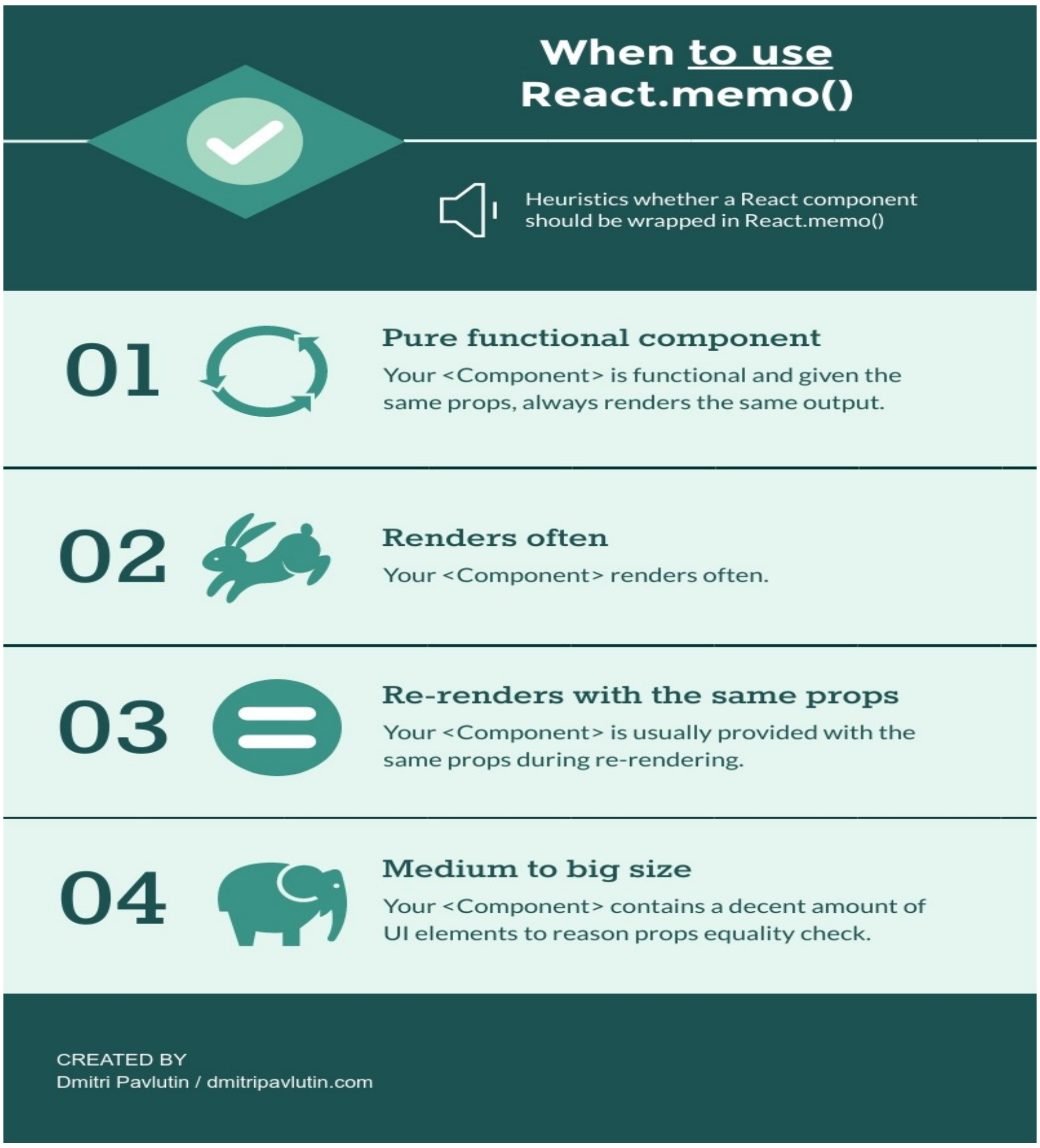
React Memo



When to use React.memo()

Heuristics whether a React component should be wrapped in React.memo()

01 **Pure functional component**
Your <Component> is functional and given the same props, always renders the same output.

02 **Renders often**
Your <Component> renders often.

03 **Re-renders with the same props**
Your <Component> is usually provided with the same props during re-rendering.

04 **Medium to big size**
Your <Component> contains a decent amount of UI elements to reason props equality check.

CREATED BY
Dmitri Pavlutin / dmitripavlutin.com

Using a memo will cause React to skip **rendering** a component if its **props** have **not changed**.

This can **improve performance**.

Problem

In this example, the Todos component re-renders even when the todos have not changed.

Example:

index.js:

```
import { useState } from "react";
import ReactDOM from "react-dom";
import Todos from "./Todos";

const App = () => {
  const [count, setCount] = useState(0);
  const [todos, setTodos] = useState(["todo 1", "todo 2"]);

  const increment = () => {
    setCount((c) => c + 1);
  };

  return (
    <>
      <Todos todos={todos} />
      <hr />
      <div>
        Count: {count}
        <button onClick={increment}>+</button>
      </div>
    </>
  );
};

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

const Todos = ({ todos }) => {
  console.log("child render");
  return (
    <>
      <h2>My Todos</h2>
      {todos.map((todo, index) => {
        return <p key={index}>{todo}</p>;
      })}
    </>
  );
};

export default Todos;
```

When you click the increment button, the Todos component re-renders.

If this component was complex, it could cause performance issues.
Solution
To fix this, we can use a memo.

Use memo to keep the Todos component from needlessly re-rendering.

Wrap the Todos component export in memo:

Example:

index.js:

```
import { useState } from "react";
import ReactDOM from "react-dom";
import Todos from "./Todos";

const App = () => {
  const [count, setCount] = useState(0);
  const [todos, setTodos] = useState(["todo 1", "todo 2"]);

  const increment = () => {
    setCount((c) => c + 1);
  };

  return (
    <>
      <Todos todos={todos} />
      <hr />
      <div>
        Count: {count}
        <button onClick={increment}>+</button>
      </div>
    </>
  );
};

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

import { memo } from "react";

const Todos = ({ todos }) => {
  console.log("child render");
  return (
    <>
      <h2>My Todos</h2>
      {todos.map((todo, index) => {
        return <p key={index}>{todo}</p>;
      })}
    </>
  );
};

export default memo(Todos);
```

Now the Todos component only **re-renders** when the todos that are **passed** to it through **props** are updated.