React compiler

Automating optimisation

28/08/2025

SOMMAIRE

What is the React Compiler?

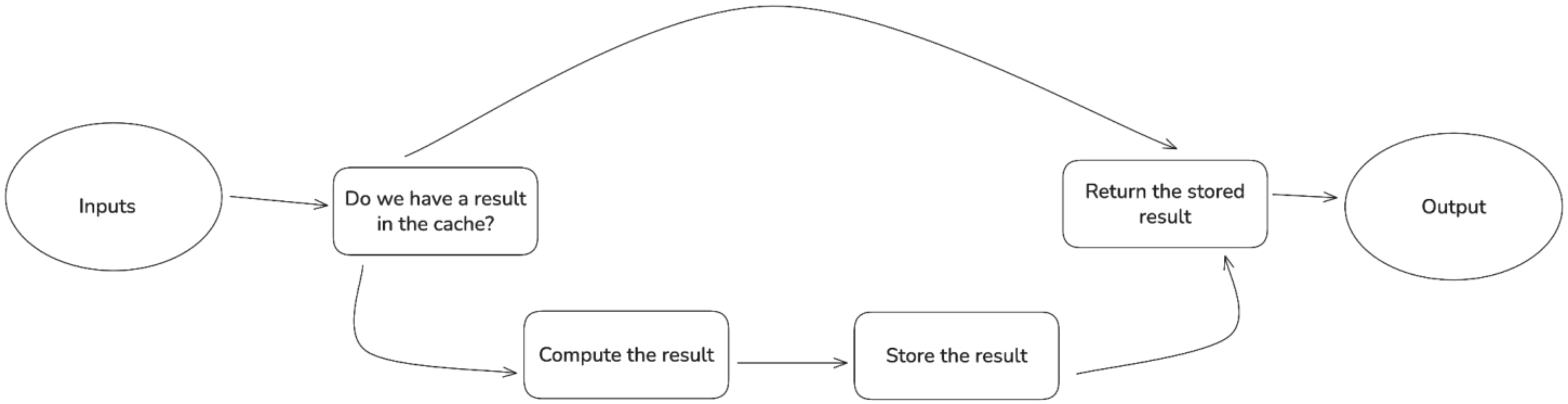
React Compiler

- Built time tool
- Automatically optimize your React application
- Skip unnecessary re-render
- Based on “Rules of React”
- Integration with Eslint

memo, useCallback and useMemo

Optimizing re-rendering, one step at a time

Memoization



Before the compiler

React primitives

for manual optimisations

memo

Used for memoizing a component.
Won’t re-render if props or state did not change.

useCallback

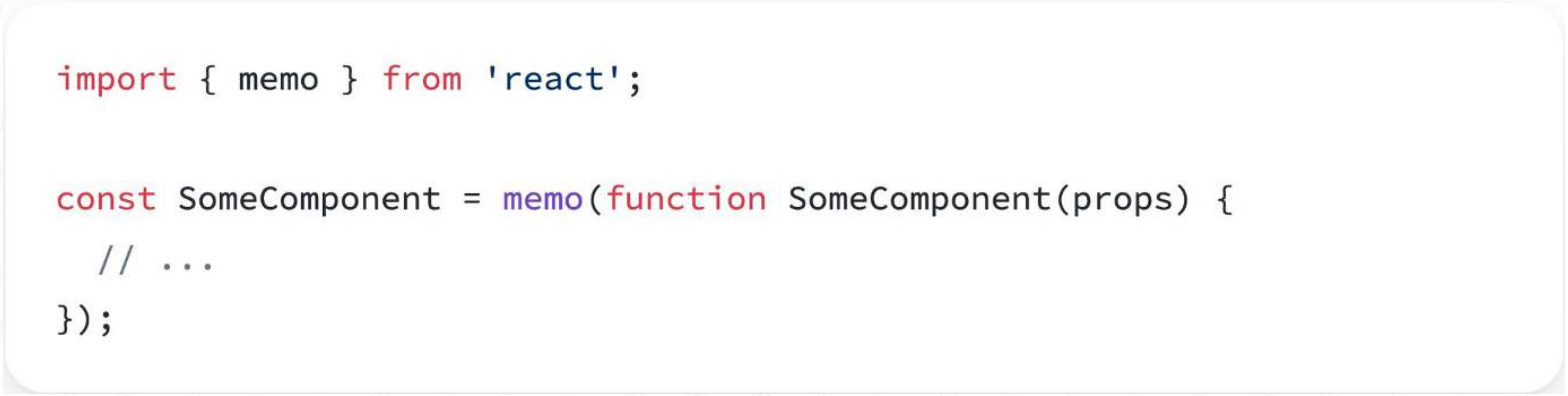
Used for memoizing a function.
Redefine the function only if one of the dependency has changed.

useMemo

Used for memoizing a value.
Useful when heavy computation is involved.
Will re-evaluate if one of the dependency has changed.

memo
for components

- List the dependencies
- Wrap the component with *`memo`*
- Optional: pass a function to indicate whether the rerender needs to happen based on previous and current props



```javascript
import { memo } from 'react';

const SomeComponent = memo(function SomeComponent(props) {
  // ...
});
```

useCallback

for functions

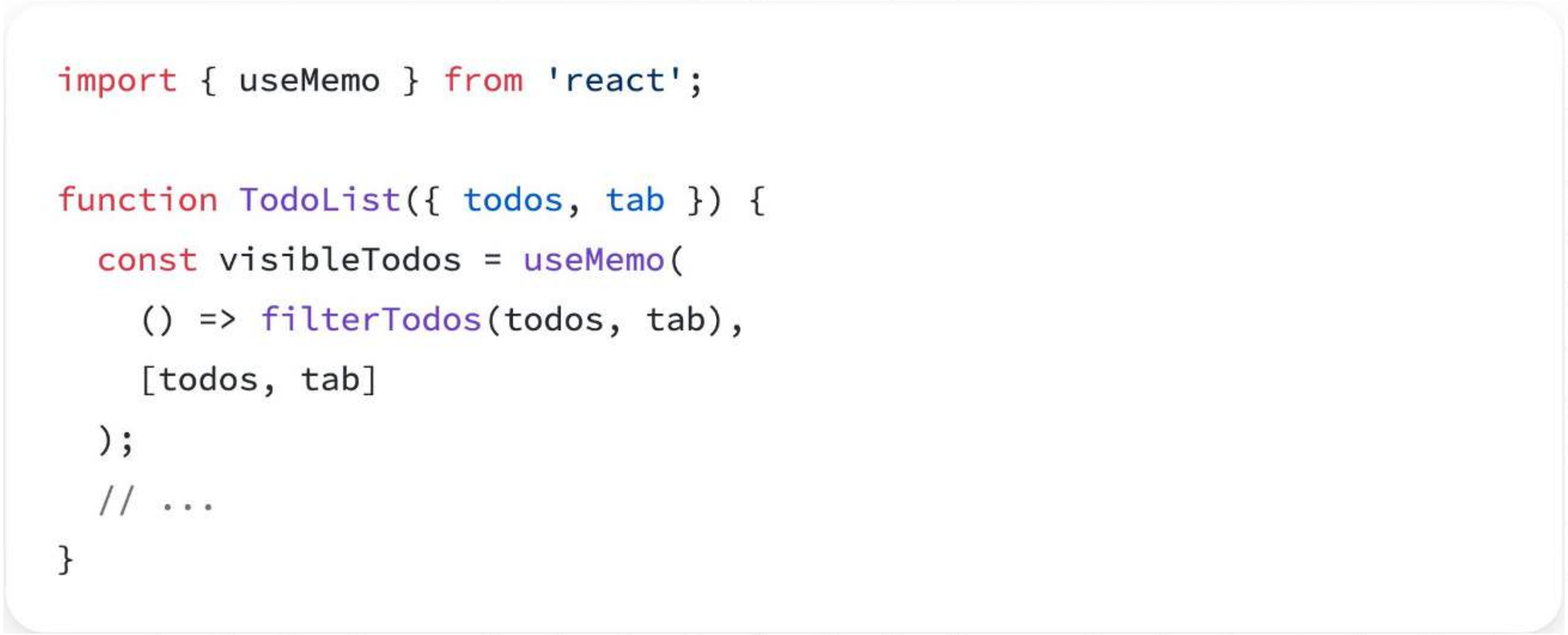
- List the dependencies
- Wrap the function with *`useCallback`*
- Add the relevant dependencies to the dependency array

```
import { useCallback } from 'react';

export default function ProductPage({ productId, referrer, theme }) {
  const handleSubmit = useCallback((orderDetails) => {
    post('/product/' + productId + '/buy', {
      referrer,
      orderDetails,
    });
  }, [productId, referrer]);
```

useMemo
for values

- List the dependencies
- Wrap the value with *`useMemo`*
- Add the relevant dependencies to the dependency array



```
import { useMemo } from 'react';

function TodoList({ todos, tab }) {
  const visibleTodos = useMemo(
    () => filterTodos(todos, tab),
    [todos, tab]
  );
  // ...
}
```

What’s gonna change with the React Compiler?

How does React “knows” what to re-render?

Children? Props? State? All of the above?

Current behavior

1. A state change happens in a parent component P
2. React re-renders the component P
3. P defines a variable that need heavy processing to init its value
4. P defines a function to be used as a a props of a child component C (e.g. event handler)
5. When rendering the component will trigger a rerender of all the child components including a child component C
6. C rerenders

Telling React what to re-render?

Manual memoization.

With manual memoization

1. A state change happens in a parent component P
2. React re-renders the component P
3. P contains a variable that need heavy processing to init its value
    a. We identified that this can be memoized
    b. Manually wrapped into useMemo()
    c. No dependency has changed so the heavy processing don’t need to happen
4. P defines a function to be used as a a props of a child component C (e.g. event handler)
    a. We identified that this can be memoized
    b. Manually wrapped into useCallback()
    c. No dependency has changed so the heavy processing don’t need to happen
5. When rendering the component will trigger a rerender of all the child components including a child component C
6. C props did not change so no rerender needs to happen

Letting React figure it out.

The Compiler.

With the React compiler

1. A state change happens in a parent component P
2. React re-renders the component P
3. P contains a variable that need heavy processing to init its value
   a. The value is cached automatically
4. P defines a function to be used as a a props of a child component C (e.g. event handler)
   a. The function is cached automatically
5. When rendering the component will trigger a rerender of all the child components including a child component C
6. C props did not change so no rerender needs to happen
   a. The component is cached automatically

How do I start using it?

https://react.dev/learn/react-compiler

https://playground.react.dev

It's all about plugins

Babel plugin

To run the compile step on your codebase

Eslint plugin

To enforce the "Rules of React" and be ready to use the compiler

Runtime for React<19

If you're not using React 19, there's a plugin available for older runtimes https://react.dev/learn/react-compiler/installation https://www.npmjs.com/package/react-compiler-healthcheck

You can start using the linting rules now!

Even if you don't plan on using the compiler before release.

- Some good practices about rendering are now handled by the eslint plugin
- The day you want to use the compiler, your codebase will most likely be ready
- You don't need to turn the compiler on for all your app and can start the migration progressively

Questions?

Thank you!

Thomas Huchedé
Consultant
Zenika Singapore
thomas.huchede@zenika.com

Illustrations by Storyset / Open-source icons and logos by Pictogrammers