incompatible-library

Validates against usage of libraries which are incompatible with memoization (manual or automatic).

중요합니다!

These libraries were designed before React's memoization rules were fully documented. They made the correct choices at the time to optimize for ergonomic ways to keep components just the right amount of reactive as app state changes. While these legacy patterns worked, we have since discovered that it's incompatible with React's programming model. We will continue working with library authors to migrate these libraries to use patterns that follow the Rules of React.

Rule Details

Some libraries use patterns that aren't supported by React. When the linter detects usages of these APIs from a known list, it flags them under this rule. This means that React Compiler can automatically skip over components that use these incompatible APIs, in order to avoid breaking your app.

```
// Example of how memoization breaks with these libraries
function Form() {
  const { watch } = useForm();

  // ❌ This value will never update, even when 'name' field changes
  const name = useMemo(() => watch('name'), [watch]);
```

```
  return <div>Name: {name}</div>; // UI appears "frozen"
}
```

React Compiler automatically memoizes values following the Rules of React. If something breaks with manual `useMemo`, it will also break the compiler's automatic optimization. This rule helps identify these problematic patterns.

자세히 살펴보기

Designing APIs that follow the Rules of React

자세히 보기

Invalid

Examples of incorrect code for this rule:

```
// ❌ react-hook-form `watch`
function Component() {
  const {watch} = useForm();
  const value = watch('field'); // Interior mutability
  return <div>{value}</div>;
}

// ❌ TanStack Table `useReactTable`
function Component({data}) {
  const table = useReactTable({
    data,
    columns,
    getCoreRowModel: getCoreRowModel(),
  });
  // table instance uses interior mutability
  return <Table table={table} />;
```

주의하세요!

MobX

MobX patterns like `observer` also break memoization assumptions, but the linter does not yet detect them. If you rely on MobX and find that your app doesn't work with React Compiler, you may need to use the `"use no memo" directive`.

```
// ❌ MobX `observer`
const Component = observer(() => {
  const [timer] = useState(() => new Timer());
  return <span>Seconds passed: {timer.secondsPassed}</span>;
});
```

Valid

Examples of correct code for this rule:

```
// ✅ For react-hook-form, use `useWatch`:
function Component() {
  const {register, control} = useForm();
  const watchedValue = useWatch({
    control,
    name: 'field'
  });

  return (
    <>
      <input {...register('field')} />
      <div>Current value: {watchedValue}</div>
    </>
  );
```

```
}
```

Some other libraries do not yet have alternative APIs that are compatible with React's memoization model. If the linter doesn't automatically skip over your components or hooks that call these APIs, please file an issue so we can add it to the linter.

이전
immutability

다음
preserve-manual-memoization