Twitter Thread by George Moller

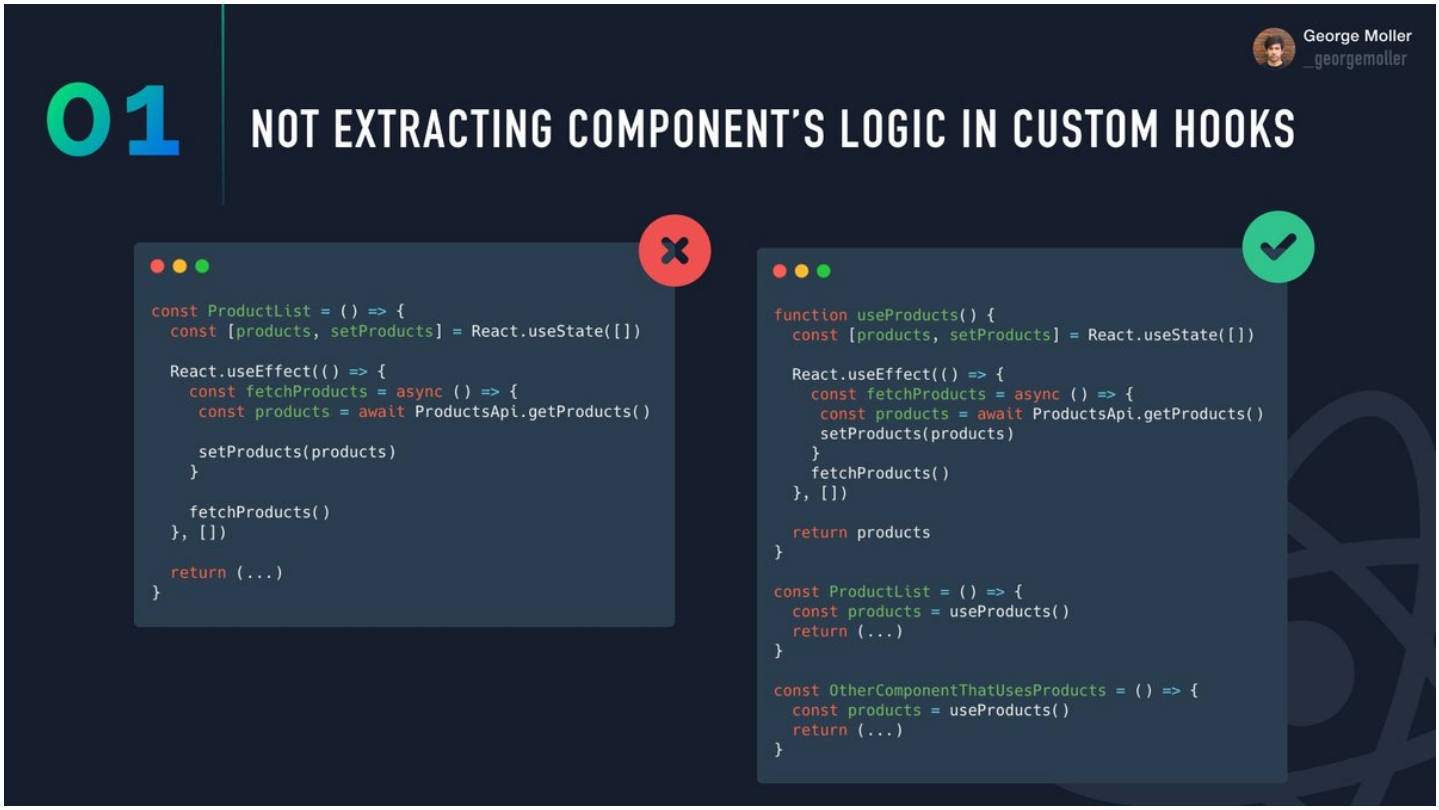
George Moller
@_georgemoller

7 common React mistakes that you should avoid.

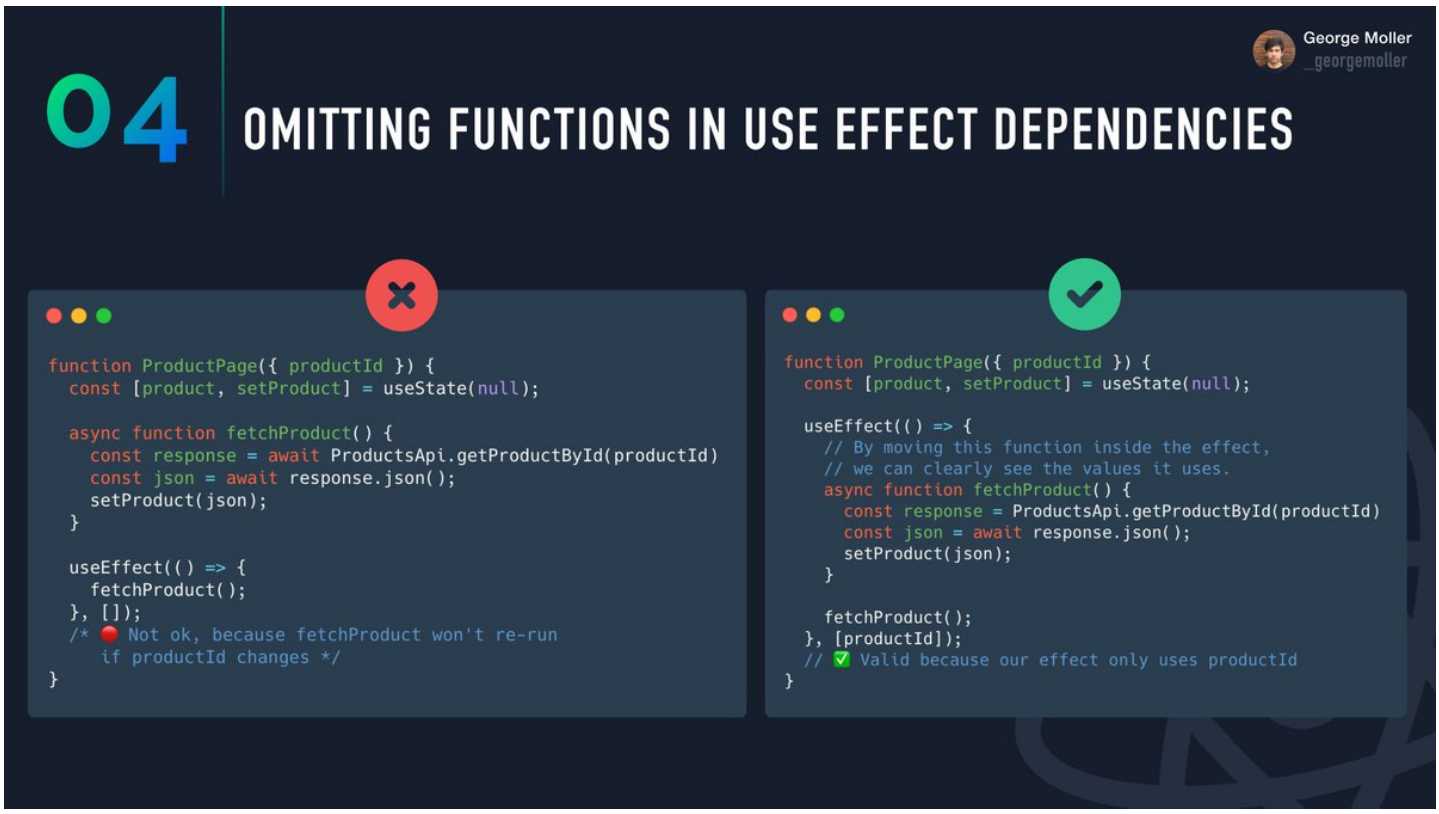
■ Thread ■



1/4


George Moller
_georgemoller
01
NOT EXTRACTING COMPONENT'S LOGIC IN CUSTOM HOOKS
const ProductList = () => {
const [products, setProducts] = React.useState([])
React.useEffect(() => {
const fetchProducts = async () => {
const products = await ProductsApi.getProducts()
setProducts(products)
}
fetchProducts()
}, [])
return (...)
}
function useProducts() {
const [products, setProducts] = React.useState([])
React.useEffect(() => {
const fetchProducts = async () => {
const products = await ProductsApi.getProducts()
setProducts(products)
}
fetchProducts()
}, [])
return products
}
const ProductList = () => {
const products = useProducts()
return (...)
}
const OtherComponentThatUsesProducts = () => {
const products = useProducts()
return (...)
}

2/4


George Moller
_georgemoller
04
OMITTING FUNCTIONS IN USE EFFECT DEPENDENCIES
function ProductPage({ productId }) {
const [product, setProduct] = useState(null);
async function fetchProduct() {
const response = await ProductsApi.getProductById(productId)
const json = await response.json();
setProduct(json);
}
useEffect(() => {
fetchProduct();
}, []);
/* 🔴 Not ok, because fetchProduct won't re-run
if productId changes */
}
function ProductPage({ productId }) {
const [product, setProduct] = useState(null);
useEffect(() => {
// By moving this function inside the effect,
// we can clearly see the values it uses.
async function fetchProduct() {
const response = ProductsApi.getProductById(productId)
const json = await response.json();
setProduct(json);
}
fetchProduct();
}, [productId]);
// ✅ Valid because our effect only uses productId
}

3/4


07
USING MEMO/useMemo too soon
George Moller
_georgemoller
A complete example of this can be found here:
https://overreacted.io/before-you-memo/

4/4


George Moller
_georgemoller
Did you find it useful?
Let me know in the comments.