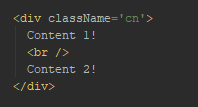
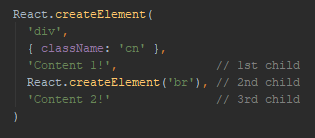
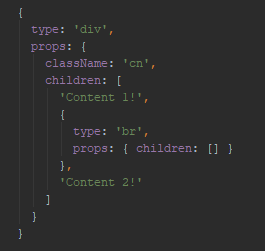
**JSX**



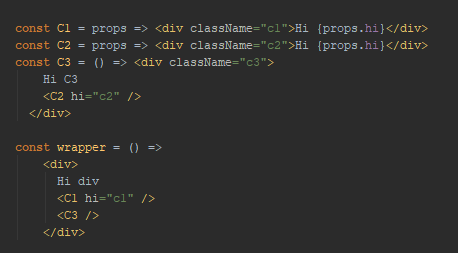
**JS**



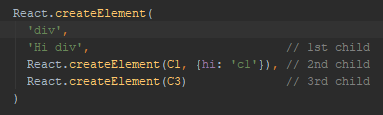
**VDOM**



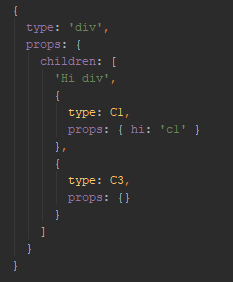
JSX



JS



VDOM



C1 and C3 have attachments

After a Virtual DOM object is built, ReactDOM.render will try to transform it into a DOM node our browser can display according to those rules:

* If a type attribute holds a string with a tag name—create a tag with all attributes listed under props.
* If we have a function or a class under type—call it and repeat the process recursively on a result.
* If there are any children under props—repeat the process for each child one by one and place results inside the parent’s DOM node.

Rebuilding the DOM – reconciliation or diffing

Conditions:

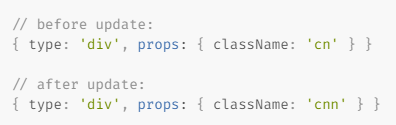
1. Node or parent Node component changes props
2. Node or parent Node component setState() is called
3. Node or parent Node component forceUpdte() is called

**Scenario 1: type is a string, type stayed the same across calls, props did not change either**



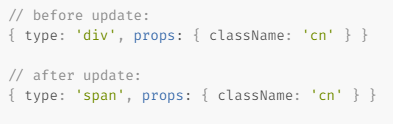
DOM stays the same

**Scenario 2: type is still the same string, props are different**



React knows how to change its properties through standard DOM API calls, without removing the node from a DOM tree

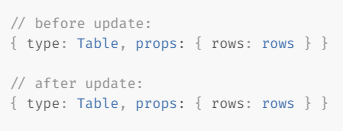
**Scenario 3: type has changed to a different String, or from String to a component**



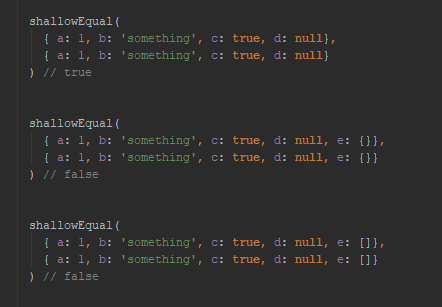
React would not even try to update our node: old element will be removed (*unmounted*) **together with all its children**. React uses === (triple equals) to compare type values, so they have to be the same *instance*of the same class or the *same* function.

**Scenario 4: type is a component**

If type is a reference to a function or a class, and we started tree reconciliation process, then React will always try to look inside the component to make sure that the values returned on render did not change (sort of a precaution against side-effects). Rinse and repeat for each component down the tree

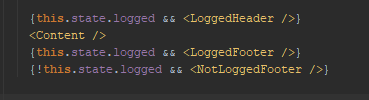


**shallow equality**

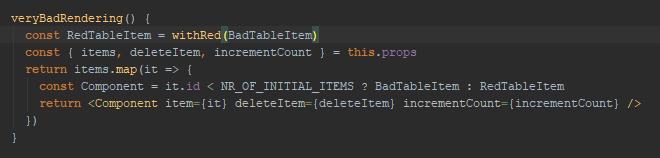


**Best practices:**

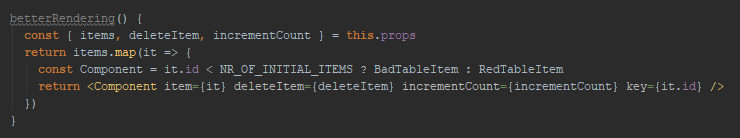
1. Use **short circuit boolean evaluation** – components will mount/unmount



1. Don’t create **HOCs** inside render() – components will mount/unmount



1. Use **key** when rendering an array – components could mount/unmount if index and type are different



1. Use **shouldComponentUpdate()** or **PureComponent** implementation – only components with differing props will be updated
2. Don’t provide **new** functions as component props – it will make unnecessary updates as props will differ in **shallow comparison**



1. Sometimes better to pass object properties rather object itself in props – it could fail shallow equality

