1. What are the principles of redux?

**First: the State Tree Principle**

everything that changes in app (data/ui state) is contained in a single object called the state or the *state tree*

**Second: the Action Principle**

State tree is redundant (cannot be modified or written to). To change the state, dispatch an *action*.

**Third: Pure and Impure Functions**

1. **What is pure and Impure functions?**

function square(x) {

return x \* x;

}

function squareAll(items) {

return items.map(square);

}

Pure functions are those whose return values depend only upon the values of their arguments. Pure functions don't have side effects like network or database calls. Pure functions also do not override the values of anything. In the above example, a new array is returned instead of modifying the items that was passed in.

**Impure:**

function square(x) {

updateXInDatabase(x);

return x \* x;

}

function squareAll(items) {

for (let i = 0; i < items.length; i++) {

items[i] = square(items[i]);

}

}

**URL:** [**https://gkedge.gitbooks.io/redux-course-notes/01-Intro\_and\_3\_Principles\_of\_Redux.html**](https://gkedge.gitbooks.io/redux-course-notes/01-Intro_and_3_Principles_of_Redux.html)

1. **What is ref’s when to use?**

In the typical React dataflow, [props](https://reactjs.org/docs/components-and-props.html) are the only way that parent components interact with their children. To modify a child, you re-render it with new props. However, there are a few cases where you need to imperatively modify a child outside of the typical dataflow. The child to be modified could be an instance of a React component, or it could be a DOM element. For both of these cases, React provides an escape hatch.

When to Use Refs

There are a few good use cases for refs:

* Managing focus, text selection, or media playback.
* Triggering imperative animations.
* Integrating with third-party DOM libraries.

Avoid using refs for anything that can be done declaratively.

For example, instead of exposing open () and close () methods on a Dialog component, pass an isOpen prop to it.

**URL:** [**https://reactjs.org/docs/refs-and-the-dom.html**](https://reactjs.org/docs/refs-and-the-dom.html)

[**https://hackernoon.com/refs-in-react-all-you-need-to-know-fb9c9e2aeb81**](https://hackernoon.com/refs-in-react-all-you-need-to-know-fb9c9e2aeb81)

**Use ‘ref’ only if its MUST , otherwise not …why?**

* It hinders in optimized working of Babel inline plugin.
* Using refs is kinda moving little away the react way of thinking which is once state changes, you re-render all the components of your UI that depend on that state. React will take care of making sure only the right bits of the DOM are updated, making the whole thing efficient. You may eventually use refs in a Jquery fashion which is not the goal.

1. **How to access child method from parent in react?**

We can use this by using the refs.

Example: const { Component } = React;

const { render } = ReactDOM;

class Parent extends Component {

render() {

return (

<div>

<Child ref={instance => { this.child = instance; }} />

<button onClick={() => { this.child.getAlert(); }}>Click</button>

</div>

);

}

}

class Child extends Component {

getAlert() {

alert('clicked');

}

render() {

return (

<h1>Hello</h1>

);

}

}

render(

<Parent />,

document.getElementById('app')

);

1. Benefits of stateless components.

**Performance:** Stateless components will bring performance gain since there is no state and lifecycle of a component. React doesn’t need to have unnecessary check and memory allocations which eventually brings performance boost.

**More focused on presentation**: React is forcing us to write components which are used only for presentational purpose, these components are useful for dumb presentational purpose only, that focus on UI rather than behavior. Technically speaking its normal tendency to go for stateful components, doing that way we are making our code easily hackable where every damn component keeps its own local state. Functional components allowing us to have pure components which just focus on presentation. whereas the state is maintained by few higher-level components.

**No this Keyword:** stateless component is just a function. Thus, all the annoying and confusing quirks with Javascript’s this keyword are avoided.

**URL:** <https://hackernoon.com/react-stateless-functional-components-nine-wins-you-might-have-overlooked-997b0d933dbc>

<http://knowledge-cess.com/advantages-of-react-stateless-functional-components/>

1. When does the V-Dom gets created?

*Render is where the Virtual DOM gets re-build and the diffing happens*

Link: <https://medium.com/@gethylgeorge/how-virtual-dom-and-diffing-works-in-react-6fc805f9f84e>

1. ReactDom why do we need it ?

The react-dom package provides DOM-specific methods that can be used at the top level of your app and as an escape hatch to get outside of the React model if you need to. Most of your components should not need to use this module.

* [render()](https://reactjs.org/docs/react-dom.html#render)
* [hydrate()](https://reactjs.org/docs/react-dom.html#hydrate)
* [unmountComponentAtNode()](https://reactjs.org/docs/react-dom.html#unmountcomponentatnode)
* [findDOMNode()](https://reactjs.org/docs/react-dom.html#finddomnode)
* [createPortal()](https://reactjs.org/docs/react-dom.html#createportal)

1. What is ReactDom.render and how it works?

Reading and writing to the DOM is slow. Reading and writing to JavaScript objects is faster. The React virtual DOM is a JavaScript object. React never reads from the real DOM, and it only writes to the real DOM if needed.

Whenever you use functions like getElementById, you're reading from the DOM. When you change any of those elements, change classes, change content, you're writing to the DOM so you read and you write to the DOM.

React only interacts with the virtual DOM, that JavaScript object. When we use getDOMNode, we'll get state information from the virtual DOM.

When we call the render function, React will update the virtual DOM which will push only those necessary changes to the real DOM.

URL: <https://www.quora.com/How-does-ReactDOM-render-work>

<https://reactjs.org/docs/react-dom.html#finddomnode>

1. How can you access dom elements in react?

By using findDomNode or refs.

1. What is findDOMNode() ?

React.findDOMNode(components).

This method is useful for reading values out of the DOM, such as form field values and performing DOM measurements. In most cases, you can attach a ref to the DOM node and avoid using findDOMNode at all.

Note:

1. findDOMNode only works on mounted components (that is, components that have been placed in the DOM).

2. findDOMNode cannot be used on functional components