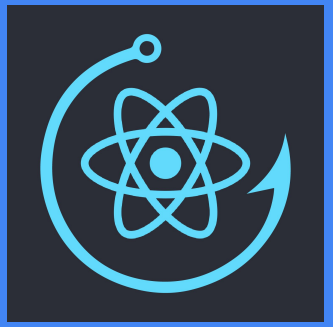
React Functional Components & Hooks

May 29th 2021

Functional Components

- Functional components are basic JavaScript functions. These are typically arrow functions but can also be created with the regular function keyword
- Sometimes referred to as “dumb” or “stateless” components as they simply accept data and display them in some form; that is they are mainly responsible for rendering UI
- React lifecycle methods (for example, componentDidMount) cannot be used in functional components
- There is no render method used in functional components
- These are mainly responsible for UI and are typically presentational only (For example, a Button component)
- Functional components can accept and use props

Hooks



- Hooks are functions that let you “hook into” React state and lifecycle features from function components
- React feature introduced in 16.8
- Let’s you use state and other feature without writing a class
- Less verbose
- Called inside a function
- Doesn’t work in class components or any vanilla JS function description (only works inside functional components)

https://www.elanandkumar.com/blog/react-comp-lifecycle-with-hooks/

useState

- useState returns a stateful value and a function to update it
- Accepts String, Array, Integer or Object
- Uses a linked list data structure

```
const [data, setData] = useState([]);
```

useEffect

- Used for Data Fetching, subscriptions, or manually changing the DOM
- Equivalent to componentDidMount and componentDidUpdate, componentWillUnmount
- Will only run after the DOM is applied or DOM mutation is done
- Runs every time a local state is changed
- The number of times it runs depends on the dependency array
- Can be used for cleanup

```
useEffect(() => {

    // do stuff

   // component update, subscriptions to events

    // console.log, alert, fetch

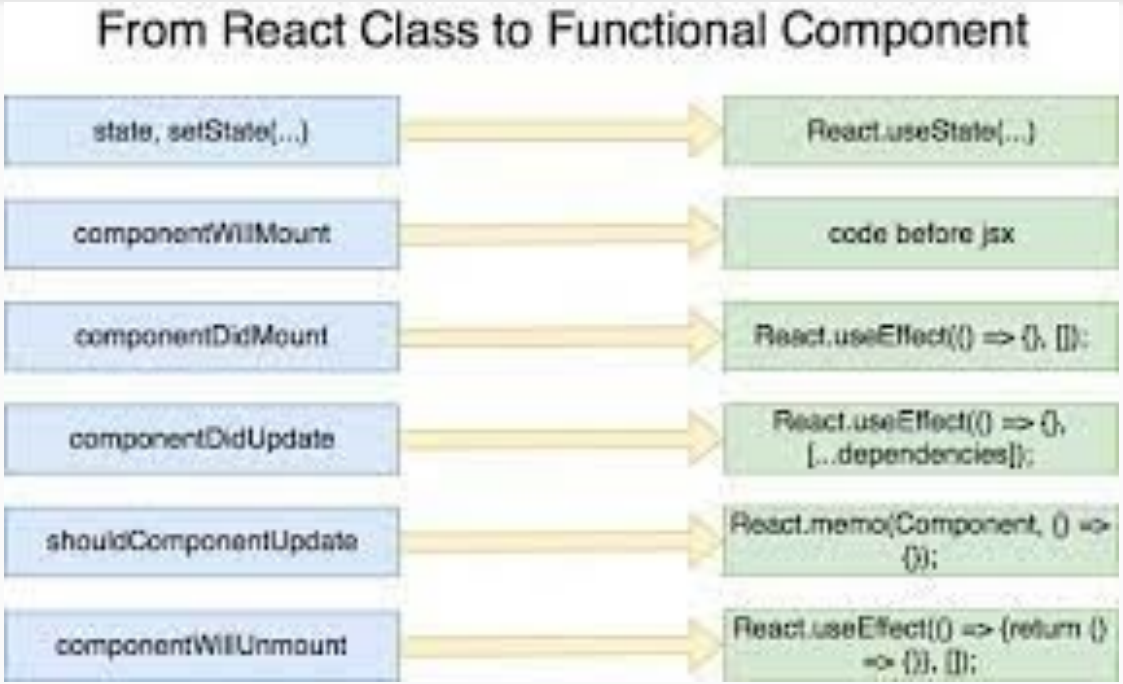
}, [dependencies]);
```

useMemo

- ```
  const memoizedValue = useMemo(() => computeExpensiveValue(a, b), [a, b]);
  ```
- Returns a memoized value
  - It allows you to **apply memoization to any value type** (not just functions)
  - It does this by accepting a function which returns the value and then that function is only called when the value needs to be retrieved
  - Similar side effects can be performed in Class component methods using componentDidUpdate or getDerivedStateFromProps
- You may rely on useMemo as a performance optimization, not as a semantic guarantee. In the future, React may choose to “forget” some previously memoized values and recalculate them on next render, e.g. to free memory for offscreen components

https://stackoverflow.com/questions/61930571/react-usememo-in-class-component
  ```

Converting from Class to Functional Component



useCallback

- Returns a memoized callback
- Pass an inline callback and an array of dependencies
- `bind(this)` is used in class components for a equivalent effect
- `useCallback(fn, deps)` is equivalent to `useMemo(() => fn, deps)`

```
useCallback(() => {
  // do stuff
  // console.log, alert, fetch
}, [dependencies]);
```

useRef

- ```
  const refContainer = useRef(initialValue);
  ```

- `useRef` returns a mutable ref object whose `.current` property is initialized to the passed argument (`initialValue`)

- This object exists outside of React's render cycle
- Refs can be used for accessing DOM nodes or React elements, and for storing mutable variables (like with instance variables in class components)
- Updating a ref is a side effect so it should be done only inside a useEffect (or useLayoutEffect) or inside an event handler
- The returned object will persist for the full lifetime of the component
- createRef is equivalent in class components
  ```

Cascading Style Sheets

CSS is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents

- A CSS file in which all class names and animation names are scoped locally by default
- SASS is detailed as "Syntactically Awesome Style Sheets". SASS is an extension of CSS3, adding nested rules, variables, mixins, selector inheritance, and more. It's translated to well-formatted, standard CSS using the command line tool or a web-framework plugin.

Available ways for styling

- Inline styling
- Using a CSS file
- Styled-components or Emotion npm package
- CSS Modules