Title: Do Hooks replace render props and higher-order components?

Answer:

React Hooks

Hooks were designed to replace `class` and provide another great alternative to compose behavior into your components. Higher Order Components are also useful for composing behavior. Hooks encapsulate the functionality to easily reusable functions

```
const [active, setActive] = useState(defaultActive)
```

There are few build-in Hooks

```
import {
  useState,
  useReducer,
  useEffect,
  useCallback,
  useMemo,
  useRef,
  ...
} from 'react'
```

Higher Order Components

A Higher Order Component (HOC) is a component that takes a component and returns a component. HOCs are composable using point-free, declarative function composition.

Example: logger API

```
import React, { useEffect } from 'react'

const withLogging = Component => props => {
  useEffect(() => {
    fetch(`/logger?location=${ window.location}`)
  }, [])
  return <Component {...props } />
}
export default withLogging
```

To use it, you can mix it into an HOC that you\'ll wrap around every page:

```
import React from 'react'
import withAuth from './with-auth.js'
import withLogging from './with-logging.js'
import withLayout from './with-layout.js'

const page = compose(
  withRedux,
  withAuth,
  withLogging,
  withLayout('default'),
)
export default page
```

To use this for a page

```
import page from '../hocs/page.js'
import MyPageComponent from './my-page-component.js'

export default page(MyPageComponent)
```

Tags: react-hooks