**Basic React Hooks**

There are 10 in-built hooks that was shipped with React 16.8 but the basic (commonly used) hooks include:

* [useState()](https://www.smashingmagazine.com/2020/04/react-hooks-api-guide/#useState)
* [useEffect()](https://www.smashingmagazine.com/2020/04/react-hooks-api-guide/#useEffect)
* [useContext()](https://www.smashingmagazine.com/2020/04/react-hooks-api-guide/#useContext)
* [useReducer()](https://www.smashingmagazine.com/2020/04/react-hooks-api-guide/#useReducer)

These are the 4 basic hooks that are commonly used by React developers that have adopted React Hooks into their codebases.

**useState()**

The useState() hook allows React developers to update, handle and manipulate state inside functional cpomponents without needing to convert it to a class component. Let’s use the code snippet below is a simple Age counter component and we will use it to explain the power and syntax of the useState() hook.

useState:

import React, { useState } from 'react'

export default function App() {

const [age, setAge] = useState(19);

const handleClick = () => setAge(age + 1)

return (

// eslint-disable-next-line

<div>

I am {age} Years Old

<div>

<button onClick={handleClick}>Increase my age! </button>

</div>

</div>

);

}

#### useEffect()

The useEffect() hook accepts a function that would contain effectual code.

useEffect() hook as **component mounting**, **updating** and **unmounting** — all combined in one function. It lets us replicate the lifecycle methods in functional components.

import React, {useState, useEffect} from 'react';

export default function App() {

//Define State

const [name, setName] = useState({firstName: 'name', surname: 'surname'});

const [title, setTitle] = useState('BIO');

//Call the use effect hook

useEffect(() => {

setName({firstName: 'Shedrack', surname: 'Akintayo'})

}, [])//pass in an empty array as a second argument

return(

<div>

<h1>Title: {title}</h1>

<h3>Name: {name.firstName}</h3>

<h3>Surname: {name.surname}</h3>

</div>

);

};

#### useContext()

The useContext() hook accepts a context object, i.e the value that is returned from React.createContext, and then it returns the current context value for that context.

import React from "react";

import ReactDOM from "react-dom";

const NumberContext = React.createContext();

function App() {

return (

<NumberContext.Provider value={45}>

<div>

<Display />

</div>

</NumberContext.Provider>

);

}

function Display() {

return (

<NumberContext.Consumer>

{value => <div>The answer to the question is {value}.</div>}

</NumberContext.Consumer>

);

}

ReactDOM.render(<App />, document.querySelector("#root"));

// import useContext (or we could write React.useContext)

import React, { useContext } from 'react';

// old code goes here

function Display() {

const value = useContext(NumberContext);

return <div>The answer is {value}.</div>;

}