17/03/23

Day 32

Parent-child data flow -

Passing data from the parent to the child components

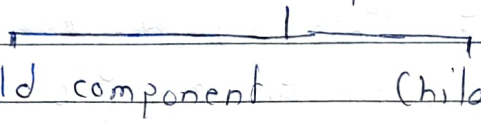
e.g

```
const data = {
    heading : " This is heading"
    callToAction : " an example"
}
function Example1() {
    return (
        <div>
            <ExampleHeading
            heading = {data.heading}
            callToAction = {data.callToAction}
            />
        </div
    )
}

function Example1(props){
    return (
        <h1> {props.heading} </h1>
        <h2> {props.callToAction} </h2>
    )
}

export default Example1;
```

The props states always flows from the parent to the child component and using props helps us avoid the need to change the data in several places. Instead we make change at data source, the parent and the updates will be applied to the child automatically.



Data flow in React

* The two main benefits of the unidirectional data flow are that it allows developer to:
  1. Comprehand the logic of React app more quickly
  2. Simplify the data flow.

e.g

Parent Component

```
function Dog(){
    return(
        <puppy name = "Max" bowlShape = "square
            bowlStatus = "full" />
    );
};
```

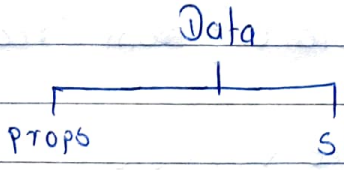
Child component

```
function Puppy(props){
    return(
        <div>
            {props.name} has <Bowl bowlShape
            ="square" bowlStatus="full" />
        </div>
    );
};
```

Grandchild component

```
function Bowl(props){
    return(
        <span>
            {props.bowlShape}-shaped bowl, and
            it's current {props.bowlStatus}
        </span>
    );
};
```

All the data in React can be divided into props data and states data



1) Props Data :

- It is data outside the components that it receive and works with but cannot mutate
- The prop data are belongs to the parent that renders the components.

2) State Data :

- It is data inside the components that it controls and can mutate.
- The state data belongs to the component itself.

* Hooks - Functions that let us hook into React state and lifestyle features from components
- Introduced in React Version 16.8

useState hook :

- Used to manage the state within a component and keep track of it, and it's built directly in React

Steps to use -

① import the useState from React

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

② Declare a state variable

```
const [state, setState] = useState(initialState)
```

③ Provide any name

e.g

```
const [showMenu, setShowMenu] = useState(false);
```

- Calling the useState hook does two things
  ① Creates a state variable with an initial value
  ② Creates a function to set that state variable's value.

Benefits of hooks - ① Readability
② Simplicity

Rules of using hooks
① We can only call hooks at the top level of our component or our own hooks
② We cannot call hooks inside loops or conditions.
③ We can only call hooks from react functions and not regular javascript functions.

4. State - Data in a components that determines behavior
   - Used to store data that affects the behavior of components
   - It allows components to stay in sync with each other and ensure that app behaves as intended.
   - The state is used to initialize the component properties.

Component
created
↓
Initial state
↓
Component properties initialized

- Components can be either stateful or stateless

1) Stateless component
   - App component with no state defined
   - It performs a single action, which is to render the text a stateless component and

```
function App() {
    return <h1> A completely stateless
        component </h1>;
};
```

2) Stateful component
   - This components also renders same texts but it references a variable to do so.

```
function App() {
    const [word, setWord ] = React.useState("Hello")
    return (
        <div>
            <h1> A state value : {word} </h1>
        </div>
    );
};
```