27/03/23

Day 41

JSX, components and Elements

JSX -

JSX is a syntax extension to javascript the React uses to describe what the UI should look like

Even though JSX look like HTML, it's essentially a more powerful abstraction that combines both markup and buisness logic into an entity called a component.

Tree of components

```
Component
        Component
        Component
        Component
                Component
                Component
Component.
```

JSX - eg

```
const buttonTitle = 'Submit'

return (
    <button className='button button-blue'>
        <span>
            {buttonTitle}
        </span>
    </button>
)
```

Element e.g

```
{
  type : 'button',
  props : {
      className : 'button button-blue',
      children : {
          type : 'span',
          children : 'Submit'
      }
  }
}
```

An element is just a way to represent the final HTML output as a plain object.

It consists primarily of two attributes, type and props.

Type defines the type of node such as button.

Props encompasses all the properties the component receive in a single object.

Eg

JSX

```
const Logout = () => (
    <div>
        <p> Are you sure? </p>
        <SubmitButton color='Blue'> Yes
        </SubmitButton>
    </div>
);
```

Element tree

```
{
    type : "div",
    props : {
        children : [
            {
                type : "p",
                props : {
                    children : "Are you sure?",
                },
            },
            {
                type : SubmitButton,
                props : {
                    color : "Blue",
                    children : "Yes",
                },
            },
        ],
    },
}
```

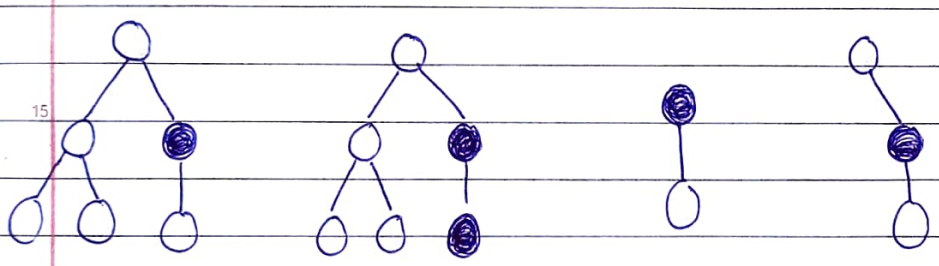
Steps involved when there is new change in UI.

Step 1) React will take all our JSX and produce a new UI representation as a tree of element.

Step 2) It will compare it with the previous representation that is kept in memory

Step 3) It will calculate the difference in a tree, recall that since each node in the tree is a javascript object.

Step 4) - Based on that difference it will apply the minimum no. of changes to the underlying dom nodes in order to process the update.



New UI → Compare → Calculate → Update

Children prop

When designing react components, one of the most important properties developers tend to overlook is the children prop.

The children prop which is a special property all components have the foundation for react powerful composition model.

There are two main features that enables component composition

1) Containment
2) Specialization

1) Containment - It refers to the fact that some components don't know their children ahead of time.

- This is especially common for components like a sidebar or a dialog.

2) Specialization - It defines components as being special cases of other components.

* React API

1) React.cloneElement

This is part of the react top-level API and it's used to manipulate and transform element.

Top-level API refers to the way we would import those functions from the react package.

We can either import react as a global object and the top of our file and access them as method on that object

or

Alternatively as named import

* React global object

```
import React from 'react';

React.cloneElement(....)
```

* Named import

```
import {cloneElement} from 'react'

cloneElement (....)
```

React.cloneElement effectively clones and returns a new copy of a provided element.

```
React.cloneElement(element, [props])
```

The first argument is the react element we would like to clone and second argument is the prompts that will be added and merged with original props passed into the component.

This API is useful to allow a parent to perform operations -
1) Modify children properties
2) Add to children properties
3) Extend functionality of children

2) React.children -

Another import top-level API useful for children manipulation

It provides utilities for dealing with props.children data structure

The most important method is the map function

React.children.map is very similar to the map function from arrays and invokes a function in every child contained within its children prop, performing a transformation and returning a new element.

React.children.map(children, callback)

4 Spread Operator (...)

The spread operator can be applied to different type of data type in javascript such as 1) arrays
2) Objects
3) String

Copying and Merging object are two main operations we can perform with these operator