Day-8

☆ props.children

→ props.children is a special prop, automatically passed to every component, that can be used to render the content included between the opening and closing tags when invoking a component.

→ This components are identified as "Boxes".

→ Example:

```
function Wrapper(props) {
    return <div>{props.children}</div>
}

function App() {
    return (
        <Wrapper>
            <h1> Hello! </h1>
            <p> .... </p>
        </Wrapper>
    )
}
```

So, when we write props.children, we can pass any content btw the opening and closing tags. So, it will take <h1>, <p> content and render it between the <div> of Wrapper component.

* Types of children

1) String literals

<Wrapper> Hello World! </Wrapper>

→ Rules:

(1) JSX removes whitespaces at the beginning and end of a line, as well as blank lines.

```
<div>    Hello    </div>

<div>
    Hello
</div>
```

↳ Output: Hello

(2)
```
<div>

        Little Lemon
    </div>
```

↳ Output: Little Lemon (ignores space)

(3)
```
<div>
    Hello
    World
</div>
```
↳ Output: Hello World (Changes it into single space)

2) JSX elements

```
<Wrapper>
    <Title />
    <Body />
</Wrapper>
```

$\longrightarrow$ Nested components.

We can also pass different types of children combinations together.

3) Javascript Expressions

```
<Wrap> {`Little Lemon`} </Wrap>
```

4) Functions ✓ (Accepted)

5) Booleans, nulls and undefined are ignored.

```
<div> {undefined} </div>
<div> {true} </div>
<div />
```

They don't render anything.

☆ Manipulating children dynamically in JSX.

→ Two API's (we will work with these two)

- → React.cloneElement
- → React.children

Miracle
Pg.No :
Date: / /20

(1) React.cloneElement

→ React top-level API. Used to manipulate and transform elements.

→ Top-level API refers to the way you import these functions from react package.

(1) React global Object

React.cloneElement(...)

(2) As a named Import.

import {cloneElement} from 'react'

cloneElement(...)

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

$\text{cloneElement}(\underset{\uparrow}{\text{element}}, \underset{\uparrow}{[\text{props}]})$

the element you want to clone

additional props you want to add.

What we can do?

(1) Modify children properties
(2) Add to children properties.
(3) Extend functionality of children

Miracle
Pg No: ______
Date: / /20

(2) React. children

→ Top-level API, useful for children manipulation.

→ Provides utilities for dealing with the props. children data structure.

→ Most imp. method : map function

React. children. map (children, callback)

invokes a func in every child of children prop.

Similar to map in React arrays.

Returns a new element

★ Spread Attributes

→ Spread Operator (...)

When we want to keep the previous data as it is and add new data or modify any data we use spread operator.

e.g.
```
const order = {
      id: '1',
      name: '',
      item: 'pizza'
      price: '',
}
```

```
const orderAmen = {
    ...order,          → Everything else as it is.
    item: 'burger'     → Item property changed.
}                        pizza → burger
```