```
/* =============== Fragments, Portals & Refs =================
    1) JSX Limitations & Fragmest Helps us to overcome the limitations
    2) Geting a Cleaner DOM with Portals
    3) Working with Refs

// ================ JSX Limitations ===========

return (
    <h2> Hi There! </h2>
    <p> This does not Work :- </p>
);

you can't return more than one "root" JSX element
(you also can't store more than ore "root" JSX element in one variable);

Simple:

return (
    React.createElement('h2', {}, "Hi There!");
    React.createElement('p', {}, "This does not work :-");
);

React create element by it will not capable to send more element at time


The Solution: Always Wrap Adjacent Elements

return (
    <div>
        <h2> Hi there! </h2>
        <p> This Dose work Now </p>
    </div>
);

It simple return only one "root" element

******* A New Problem : "<div>" soup;

<div>
    <div>
        <div>
            <h2> some content - yeah, this can really Happend. </h2>
        </div>
    </div>
</div>

some time some many div coz problem rendering time consuming ...

1) To Some this unseserey div problem by return Wrapper
2) That wrapper only send the Childern element

******* Create Helper.js *******
const Wrapper = (props) =>{
    return props.children;
}
export default Wrapper;

3) import the helper file and and use as Wrapper to send root JSX one single variable
```

```
import Wrapper form './helper/helper';
function CourseItem(){
    return (
        <Wrapper>
        ...........
        ..........
        JSX
        .........
        ...........
        </Wrapper>
    );
}

****** Another Way to Remove unseserey <div> suop ******

=================== Introducing Fragments ===============

    1) first one Always Work and better

    return(
        <React.Fragment>
            <h2> Hi there! </h2>
            <p> This is Remove outer div </p>
        </React.Fragment>
    );

    OR

    return(
        <>
        <h2> Hi there </h2>
        <p> This is another example of Fragment use </p>
        </>
    );

    OR
    import React, {Fragment} from 'react';

    ...
    return(
        <Fragment>
            <h2> Hi there </h2>
            <p> This is another example of Fragment </p>
        </Fragment>
    );


===================== Understanding React Portals ============
portals under the ReactDOM
1) it helps us to render the component other place
2) After creating component use ReactDOM to place that component body close new createt index file element
3) ReactDOM.createPortal(<Component />, placeByID);
4) ReactDOM.createPortal(<SideBar />, document.getElementById("SideBar"));

================== Working with Ref ==============
Some time only Read the value use {useRef} hooks

it Alternative => useSate both rendering react Component Again

```

```

import React, {useRef} from 'react';

function Component(){
    const refObjName = useRef(); set defalult value

        function inputHandeler(event){
            event.preventDefault();
            console.log(refObjName); // it gives curent objec
            const getName = refObjName.current.value;
        }
    <form>
        <input ref={refObj} onChange={inputHandeler}/>
    </form>
}


*/
```