**const**

const React = require('react')

**import**

import React from 'react'

Here are the definitions of each but I am still not sure which to use.

[import](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import)

The import statement is used to import functions, objects or primitives that have been exported from an external module, another script, etc.

[const](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const)

The const declaration creates a read-only reference to a value. It does not mean the value it holds is immutable, just that the variable identifier cannot be reassigned

There are a number of React ES6 examples that start

**import** React **from** 'react';

**class** **MyComponent** **extends** **React**.**Component** { */\* ... \*/* }

This seems weird, as it implies that if React were itself written in ES6, the "react" module would have a default export with a number of properties:

**export** **default** {

Component: ReactComponent

*// ...*

}

rather than a named export:

**export** {ReactComponent **as** Component};

To work with a named export, the import would need to look like

**import** \* **as** React **from** 'react';

*// or*

**import** {Component} **from** 'react';

Using Babel today, both named and default import statements work because React isn't an ES6 module. If/when react is written in ES6, one of the two forms of import won't work unless there are redundant (both default and named) exports.

my-navbar.jsx

import React from 'react';

import Navbar from 'react-bootstrap/lib/Navbar';

export default class MyNavbar extends React.Component {

render(){

return (

<Navbar className="navbar-dark" fluid>

...

</Navbar>

);

}

}

main-page.jsx

import React from 'react';

import ReactDOM from 'react-dom';

import MyNavbar from './comp/my-navbar.jsx';

export class MyPage extends React.Component{

render(){

return(

<MyNavbar />

...

);

}

}

ReactDOM.render(

<MyPage />,

document.getElementById('container')

);

Using webpack and following their tutorial, I have main.js:

import MyPage from './main-page.jsx';

|  |  |
| --- | --- |
|  | Wrapping components with braces if no default exports:  import {MyNavbar} from './comp/my-navbar.jsx';  or import multiple components from single module file  import {MyNavbar1, MyNavbar2} from './module'; |

1. Declared the component you were exporting as the 'default' component that this module was exporting: export default class MyNavbar extends React.Component { , and so when Importing your 'MyNavbar' you DON'T have to put curly braces around it : import MyNavbar from './comp/my-navbar.jsx';. Not putting curly braces around an import though is telling the document that this component was declared as an 'export default'. If it wasn't you'll get an error (as you did).
2. If you didn't want to declare your 'MyNavbar' as a default export when exporting it : export class MyNavbar extends React.Component { , then you would have to wrap your import of 'MyNavbar in curly braces: import {MyNavbar} from './comp/my-navbar.jsx';

I think that since you only had one component in your './comp/my-navbar.jsx' file it's cool to make it the default export. If you'd had more components like MyNavbar1, MyNavbar2, MyNavbar3 then I wouldn't make either or them a default and to import selected components of a module when the module hasn't declared any one thing a default you can use: import {foo, bar} from "my-module";where foo and bar are multiple members of your module.

Definitely read the MDN doc it has good examples for the syntax. Hopefully this helps with a little more of an explanation for anyone that's looking to toy with ES 6 and component import/exports in React.