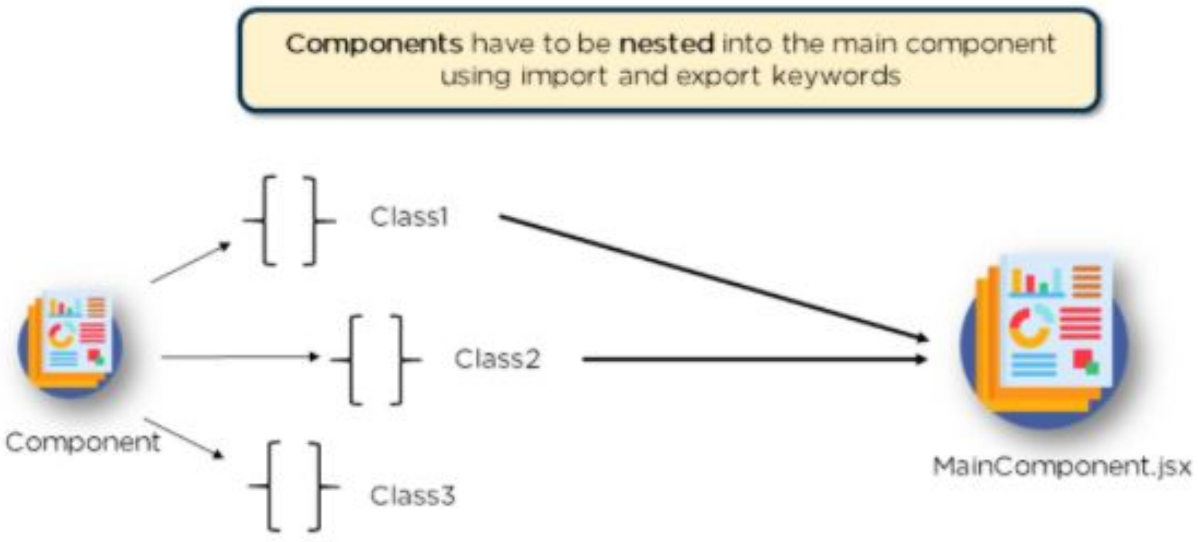
Nesting Of Components

In React, we can nest components inside within one another. This helps in creating more complex User Interfaces. The components that are nested inside parent components are called child components. Import and Export keywords facilitate the nesting of the components.

- **Export**: This keyword is used to export a particular module or file and use it in another module.
- **Import** - This keyword is used to import a particular module or file and use it in the existing module.




```
import { Class1, Class2 } from "./Component";
```

A lot of the power of ReactJS is its ability to allow the nesting of components. Take the following two components:

```
var React = require('react');
var createReactClass = require('create-react-class');

var CommentList = reactCreateClass({
  render: function() {
    return (
      <div className="commentList">
        Hello, world! I am a CommentList.
      </div>
    );
  }
});

var CommentForm = reactCreateClass({
  render: function() {
    return (
      <div className="commentForm">
        Hello, world! I am a CommentForm.
      </div>
    );
  }
});
```

You can nest and refer to those components in the definition of a different component:

```
var React = require('react');
var createReactClass = require('create-react-class');

var CommentBox = reactCreateClass({
  render: function() {
    return (
      <div className="commentBox">
        <h1>Comments</h1>
        <CommentList /> // Which was defined above and can be reused
        <CommentForm /> // Same here
      </div>
    );
  }
});
```

Further, the nesting can be done in **two** ways, which all have their own places to be used.

Say, **A**, **B**, and **C** are three Components,

- Nesting without using children: This is the style where **A composes B and B composes C**.
  - ❖ **Pros**:
    - ➢ Easy and fast to **separate UI** elements
    - ➢ Easy to inject **props down** to **children** based on the **parent** component's state
  - ❖ **Good if**:
    - ➢ B and C are just **presentational components**
    - ➢ B should be responsible for C's **lifecycle**

- ❖ **Cons**:
  - ➢ Less **visibility** into the composition **architecture**
  - ➢ Less **reusability**

- **Nesting using children**: This is the style where A composes B and A tells B to compose C. More power to parent components.

  - ❖ **Pros**:
    - ➢ Better components **lifecycle management**
    - ➢ Better **visibility** into the composition **architecture**
    - ➢ Better **reusuability**
  - ❖ **Cons**:
    - ➢ **Injecting props** can become a little **expensive**
    - ➢ **Less flexibility** and power in child components
  - ❖ **Good if**:
    - ➢ B should accept to **compose something different** than C in the future or somewhere else
    - ➢ A should control the **lifecycle** of C.