Chapter 16: Stateless Functional Components

Section 16.1: Stateless Functional Component

Components let you split the UI into *independent*, *reusable* pieces. This is the beauty of React; we can separate a page into many small reusable **components**.

Prior to React v14 we could create a stateful React component using `React.Component` (in ES6), or `React.createClass` (in ES5), irrespective of whether it requires any state to manage data or not.

React v14 introduced a simpler way to define components, usually referred to as **stateless functional components**. These components use plain JavaScript functions.

For example:

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

This function is a valid React component because it accepts a single `props` object argument with data and returns a React element. We call such components **functional** because they are literally JavaScript *functions*.

Stateless functional components typically focus on UI; state should be managed by higher-level “container” components, or via Flux/Redux etc. Stateless functional components don’t support state or lifecycle methods.

Benefits:

1. No class overhead
2. Don't have to worry about `this` keyword
3. Easy to write and easy to understand
4. Don't have to worry about managing state values
5. Performance improvement

Summary: If you are writing a React component that doesn't require state and would like to create a reusable UI, instead of creating a standard React Component you can write it as a **stateless functional component**.

Let's take a simple example :

Let's say we have a page that can register a user, search for registered users, or display a list of all the registered users.

This is entry point of the application, `index.js`:

```
import React from 'react';
import ReactDOM from 'react-dom';

import HomePage from './homepage'

ReactDOM.render(
    <HomePage/>,
    document.getElementById('app')
);
```

The `HomePage` component provides the UI to register and search for users. Note that it is a typical React component including state, UI, and behavioral code. The data for the list of registered users is stored in the `state` variable, but our reusable `List` (shown below) encapsulates the UI code for the list.

`homepage.js`:

```
import React from 'react'
import {Component} from 'react';

import List from './list';

export default class Temp extends Component{

    constructor(props) {
        super();
        this.state={users:[], showSearchResult: false, searchResult: []};
    }

    registerClick(){
        let users = this.state.users.slice();
        if(users.indexOf(this.refs.mail_id.value) == -1){
            users.push(this.refs.mail_id.value);
            this.refs.mail_id.value = '';
            this.setState({users});
        }else{
            alert('user already registered');
        }
    }

    searchClick(){
        let users = this.state.users;
        let index = users.indexOf(this.refs.search.value);
        if(index >= 0){
            this.setState({searchResult: users[index], showSearchResult: true});
        }else{
            alert('no user found with this mail id');
        }
    }

    hideSearchResult(){
        this.setState({showSearchResult: false});
    }

    render() {
        return (
            <div>
                <input placeholder='email-id' ref='mail_id'/>
                <input type='submit' value='Click here to register'
onClick={this.registerClick.bind(this)}/>
                <input style={{marginLeft: '100px'}} placeholder='search' ref='search'/>
                <input type='submit' value='Click here to register'
onClick={this.searchClick.bind(this)}/>
                {this.state.showSearchResult ?
                    <div>
                        Search Result:
                        <List users={[this.state.searchResult]}/>
                        <p onClick={this.hideSearchResult.bind(this)}>Close this</p>
                    </div>
                    :
                    <div>
                        Registered users:
```

```
                        <br/>
                        {this.state.users.length ?
                            <List users={this.state.users}/>
                            :
                            "no user is registered"
                        }
                    </div>
                }
            </div>
        );
    }
}
```

Finally, our **stateless functional component** `List`, which is used display both the list of registered users *and* the search results, but without maintaining any state itself.

`list.js`:

```
import React from 'react';
var colors = ['#6A1B9A', '#76FF03', '#4527A0'];

var List = (props) => {
    return(
        <div>
            {
                props.users.map((user, i)=>{
                    return(
                        <div key={i} style={{color: colors[i%3]}}>
                            {user}
                        </div>
                    );
                })
            }
        </div>
    );
}

export default List;
```

Reference: https://facebook.github.io/react/docs/components-and-props.html