Stateless Components From Stateful Components

Stateful and Stateless

In React, a stateful component is a component that has one or more `state` property. While a stateless component doesn't have any `state` property.

```
//This is a stateful component
class Store extends React.Component {
 constructor(props) {
  super(props);
  this.state = { sell: 'anything' };
}

//This is a stateless componenet
class Week extends React.Component {
 render() {
  return <h1>Today is {this.props.day}!
</h1>;
}
```

Programming Pattern

One of the most common programming patterns in React is to use stateful parent components to maintain their own state and pass it down to one or more stateless child components as props. The code snippet above shows a basic example.

```
//This is a stateless child component.
class babyYoda extends React.Component {
  render() {
    return <h2>I am {this.props.name}!</h2>;
  }
}
//This is a stateful Parent element.
class Yoda extends React.Component {
  constructor(props) {
    super(props);
    this.state = { name: 'Toyoda' };
  }
//The child component is rendering
information passed down from the Parent
component.
  render() {
    return <babyYoda name={this.state.name}
/>;
  }
}
```

Changing props and state

In React, a component should never change its own `props`. Attempting to do so would result in doing nothing because a component cannot change its own `prop`. In order to change a prop, you would need to use a parent component that passes them down. State, on the other hand, is the opposite of props. `state` is used to store information that a component itself can update. The code snippet above shows what not to do with `props`.

```
// Never attempt to do this.
this.props.car = 'millennium falcon';
```