React State vs Props Blog

Das React Function Component und Properties

React Komponenten werden über Properties von aussen modifiziert. Damit kann man eine Komponente leichter wiederverwenden, indem man das Verhalten über Properties steuert.Die einfachste aller Components ist die Function Component, es handelt sich hier um eine einfache JavaScript Funktion. Ein Beispiel:function SayHello(props) {
  return <h1>Hello {props.name}</h1>
}Diese Funktion ist eine gültige React-Komponente, da sie ein einzelnes Objektargument "props" akzeptiert und ein React-Element zurückgibt.props ist intern betrachtet ein JavaScript Object analog einer Key,Value Map oder JSON Struktur.Das folgende Script zeigt die Anwendung der SayHello Function Component:ReactDOM.render(<SayHello name="James"/>, document.getElementById('root'));Das gesamte Beispiel sehen sie nachfolgend als Lösung.

Lösung React Function Component und Properties

Das HTML Script React Function Component und Properties:<!DOCTYPE html>
<html>
  <head>
  <meta charset="UTF-8" />
  <title>React Function Component</title>
  <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
  <script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
  <script type="text/babel">
  function SayHello(props) {
  return <h1>Hello {props.name}</h1>;
  }
  </script>
  </head>
  <body>
  <div id="root"></div>
  <script type="text/babel">
  ReactDOM.render(<SayHello name="James"/>, document.getElementById('root'));
  </script>
  </body>
</html>Die laufende Lösung zu diesem Blog und React Function Component und Properties finden Sie hier

Das React Class Component State und Properties

Die klassische ES6 Rect Class Component arbeitet auch mit Properties (props), welche via constructor zuweisbar sind.Der Blog React Hello Component zeigte eine solche HelloComponent auf.Class Components haben aber noch einen internen Status den state. Mit dem state wird der interne Zustand der Instanz einer Komponente abgespeichert, es handelt sich um Attribute.Properties (props) sind read only, aber der state kann ändern. Jede state Anpassung muss über die Methode setState() propagiert werden.Wir erweitern unsere HelloComponent mit dem internen state und speichern dort die verwendete Background- und Foreground-Color:class HelloComponent extends React.Component {
  constructor(props) {
  super(props);
  this.state = {color: "red", background: "black"};
  }
  ...
}Die render Methode berücksichtig die CSS Farben aus dem state:render() {
  const style = {};
  style.background = this.state.background;
  style.color = this.state.color;
  return <div>
  <h1 style={style}>Hello {this.props.name}</h1>
  </div>
}Wir erweitern unser Beispiel und damit die Komponente mit 3 Buttons und ändern mit jedem Button Click die Foreground-Color. Das folgende Listing zeigt die Buttons:render() {
  const style = {};
  style.background = this.state.background;
  style.color = this.state.color;
  return <div>
  <h1 style={style}>Hello {this.props.name}</h1>
  <p/>
  <button style={{background: "red", padding: "1em"}} >red</button>
  <button style={{background: "green", padding: "1em"}} >green</button>
  <button style={{background: "blue", padding: "1em"}} >blue</button>
  </div>
}Das Beispiel zeigt die

HelloComponent mit den 3 Buttons:Das folgende Script zeigt die komplette HelloComponent mit dem Binding der Button Events über die Methode changeState() und dem state-Wechsel inkl. Propagation via setState() Call:class HelloComponent extends React.Component {
 constructor(props) {
 super(props);
 this.state = {color: "red", background: "black"};
 }
 render() {
 const style = {};
 style.background = this.state.background;
 style.color = this.state.color;
 return <div>
 <h1 style={style}>Hello {this.props.name}</h1>
 <p/>
 <button style={{background: "red", padding: "1em"}} onClick={() => this.changeStyle('red')}>red</button>
 <button style={{background: "green", padding: "1em"}} onClick={() => this.changeStyle('green')}>green</button>
 <button style={{background: "blue", padding: "1em"}} onClick={() => this.changeStyle('blue')}>blue</button>
 </div>;
 }
 changeStyle(color) {
 this.state.color = color;
 this.setState(this.state);
 }
 }Bei der onClick Methode handelt es sich um eine Lambda Expression (quasi anonyme Funktion). Die Methode changeStyle() löst die Propagation via setState(this.state) aus, so dass die Komponente neu gerendert und entsprechend aktualisiert wird.

Lösung React Component State und Properties

Das HTML Script React Component State und Properties:<!DOCTYPE html>
<html>
 <head>
 <meta charset="UTF-8" />
 <title>Hello Component</title>
 <script src="https://unpkg.com/react@16/umd/react.development.js"></script>
 <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
 <script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
 <script type="text/babel">
 class HelloComponent extends React.Component {
 constructor(props) {
 super(props);
 this.state = {color: "red", background: "black"};
 }
 render() {
 const style = {};
 style.background = this.state.background;
 style.color = this.state.color;
 return <div>
 <h1 style={style}>Hello {this.props.name}</h1>
 <p/>
 <button style={{background: "red", padding: "1em"}} onClick={() => this.changeStyle('red')}>red</button>
 <button style={{background: "green", padding: "1em"}} onClick={() => this.changeStyle('green')}>green</button>
 <button style={{background: "blue", padding: "1em"}} onClick={() => this.changeStyle('blue')}>blue</button>
 </div>;
 }
 changeStyle(color) {
 this.state.color = color;
 this.setState(this.state);
 }
 }
 </script>
 </head>
 <body>
 <div id="root"></div>
 <script type="text/babel">
 ReactDOM.render(<HelloComponent name="James"/>, document.getElementById('root'));
 </script>
 </body>
</html>Die laufende Lösung zu diesem Blog und React Component State und Properties finden Sie hier

Feedback

War dieser Blog für Sie wertvoll. Wir danken für jede Anregung und Feedback

Kontakt

Simtech AG
Finkenweg 23
3110 Münsingen
Schweiz

Impressum

Das Copyright für sämtliche Inhalte dieser Website liegt bei Simtech AG, Schweiz.
Beachten Sie auch unsere Hinweise zum Urheberrecht, Datenschutz und Haftungsauschluss.
Jeder Hinweis auf Fehler nehmen wir gerne entgegen.

Copyright

2024 Simtech AG, All rights reserved, Powered by stack.ch written in Golang by Daniel Schmutz

https://www.simtech-ag.ch/setstate