Tutorial React - Nico Rahm

Freitag, 1. Mai 2020 15:47

1. The Basics:

Every component returns a static html snippet, which will replace lateron the component usage.
This snippet is defined in the return value of the render() function.

After all components have been rendered the static html is hanged into the DOM by the "ReactDOM.render()" call.

2. Creating a component:

There are two different ways to create components.
One Way ist to create components as classes, wich are extending the class React.Component.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Jogi" />,
  document.getElementById('hello-example')
);
```

Every classbased component needs a render() function.
This is called when the component is used.

In a function based component, as in the following, an explicit render() function is not needed.
The html snippet has to be set as value in the return statement.

```
const Hello = (props) => {
  return(
    <div>
      Hello {props.name}
    </div>
  )
}

ReactDOM.render(
  <Hello name="Jogi"/>,
  document.getElementById("root")
);
```

3. Props:

No matter which type of component is used, the props apply to both.

In the props you can pass dynamic information to the component.
The props (Properties) are accessed like a JSON-Object and are implicit declared by component usage.

Components are used in the following html similar syntax:

```
< %ComponentName% %PropName% ={ %PropValue% } />
```

4. State:

The state is a very powerful tool in react.

States can be defined in the component context. If a state needs to be manipulated in a inner component, it has to be passed in the props of this inner component.
When a state changes, every component which uses this state will be rerendered and will show the current state.

```
class Timer extends React.Component {
  constructor(props) {
    super(props);
    this.state = { seconds: 0 };
  }

  tick() {
    this.setState(state => ({
      seconds: state.seconds + 1
    }));
  }

  componentDidMount() {
    this.interval = setInterval(() => this.tick(), 1000);
  }

  render() {
    return (
      <div>
        Seconds: {this.state.seconds}
      </div>
    );
  }
}
//....
```