React项目实战（一）

课堂目标

1. 登录组件
2. Antd Form体验
3. 高阶组件设计
4. setState原理
5. 登录体验设计

知识要点

1. antd组件库使用
2. 高阶组件
3. cloneElement
4. setState

起步

antd表单体验

https://ant.design/components/form-cn/

核心逻辑

```
@Form.create()
class App extends React.Component {
  handleSubmit = (e) => {
    e.preventDefault();
    this.props.form.validateFieldsAndScroll((err, values) => {
      if (!err) {
        console.log('Received values of form: ', values);
      }
    });
  }

  render() {
    return <div>
      <Form onSubmit={this.handleSubmit} action="">

        {this.props.getFieldDecorator('nickname', {
          rules: [{ required: true, message: 'Please input your nickname!', whitespace:
true }],
        })(
          <Input />
        )}

      </Form>

    </div>
  }
}
```

1. 高阶组件Form.create为组件赋予getFieldDecorator和validateFieldsAndScroll方法
2. getFieldDecorator设置key和rules
3. validateFieldsAndScroll验证结果

Kform组件设计

提供setRule和validate两个接口，分别用来设置rule和验证表单

```
export default function KForm(Comp){
    return class NewComp extends React.Component{

        constructor(props){
            super(props)
        }

        setRule = (key,rule,Inputcomp)=>{
            return <div>
                {React.cloneElement(Inputcomp,
                    {
                        name:key,
                    })}

            </div>
        }
```

```
        validate = ()=>{

        }
        render(){
            return <Comp {...this.props} kvalidate={this.validate} krules=
{this.setRule}></Comp>
        }
    }
}
```

使用

```
@KForm
export default class Login extends React.Component {
  handleSubmit = () => {
    if (this.props.kvalidate()) {

    }
  }
  render() {

    return (
      <div className={styles.login}>
        <div className={styles['login-container']}>
          <img src={Logo} alt="" />
          {this.props.krules(
            'username',
            [{ required: true, message: '请输入用户名' }, { min: 5, message: '用户名长度不
得少于5' }],
            <Input
              placeholder="Enter your username"
              prefix={<Icon type="user" />}
            />
          )}
          {this.props.krules(
            'passwd',
            { required: true, message: '请输入密码' },
            <Input
              type='password'
              placeholder="Enter your password"
              prefix={<Icon type="lock" />}
            />
          )}
          <div className={styles.submit}>
            <Button type='primary' block onClick={this.handleSubmit}>登录</Button>
          </div>
        </div>
      </div>

    );
```

```
  }
}
```

React.cloneElement

```
React.cloneElement(
  element,
  [props],
  [...children]
)
```

返回扩展后的组件，添加参数 事件等等

绑定事件

```
setRule = (key, rule, Inputcomp) => {
  this.rules[key] = rule;
  const suffix = this.state[key] ? (
    <Icon type="close-circle" onClick={() => this.emitEmpty(key)} />
  ) : null;
  return (
    <div>
      <p style={{ color: "red" }}>{this.state[key + "Message"]}</p>
      {React.cloneElement(Inputcomp, {
        suffix,
        name: key,
        onChange: this.handleChange,
        value: this.state[key]
      })}
    </div>
  );
};

handleChange = e => {
  let { name, value } = e.target;
  this.setState({
    [name]: value
  });
};

emitEmpty = key => {
  this.setState({
    [key]: ""
  });
};
```

验证

```
validateInput(key){
    let rule = this.rules[key]
    rule = Array.isArray(rule)?rule:[rule]

    let ret = rule.some(r=>{
        if(r.required){
            if(!this.state[key]){
                this.setState({
                    [key+'Message']:r.message
                })
                return true
            }else{
                this.setState({
                    [key+'Message']:''
                })
            }
        }
        if(r.min){
            if(this.state[key].length<r.min){
                this.setState({
                    [key+'Message']:r.message
                })
                return true
            }else{
                this.setState({
                    [key+'Message']:''
                })
            }
        }

    })
    return ret

}
validate = ()=>{
    const rets = Object.keys(this.rules).map(key=>{
        return this.validateInput(key)
    })
    return rets.every(v=>!v)
}
handleChange = (e)=>{
    let {name,value} = e.target
    this.setState({
        [name]:value
    })
    this.validateInput(name)
}
```

setState

一个小bug

输入第一个字母，还是提示不存在，输入第二个才有效果，这涉及到setState的异步机制

看个小demo

```
class App extends Component {
    constructor() {
        super();
        this.state = {
            num: 0
        }
    }
    componentDidMount() {
        for ( let i = 0; i < 100; i++ ) {
            this.setState( { num: this.state.num + 1 } );
            console.log( this.state.num );    // 会输出什么?
        }
    }
    render() {
        return (
            <div className="App">
                <h1>{ this.state.num }</h1>
            </div>
        );
    }
}
```

这是React的优化手段，但是显然它也会在导致一些不符合直觉的问题（就如上面这个例子），所以针对这种情况，React给出了一种解决方案：setState接收的参数还可以是一个函数，在这个函数中可以拿先前的状态，并通过这个函数的返回值得到下一个状态。

```
componentDidMount() {
    for ( let i = 0; i < 100; i++ ) {
        this.setState( prevState => {
            console.log( prevState.num );
            return {
                num: prevState.num + 1
            }
        } );
    }
}
```

setState修改后触发

setState第二个参数，就是setState结束后的回调

```
handleChange = (e)=>{
    let {name,value} = e.target
    this.setState({
        [name]:value
    },()=>{
        this.validateInput(name)
    })
}
```

登录体验

1. 进度条
2. 出错聚焦输入框
3. 状态提示
    1. 明确提示的弹窗
    2. 等待2秒，还没结果，提示用户继续等一会
    3. 等待5秒，提示网速不太好
    4. 等待7秒，告知表单提交失败
4. 加密传输
    1. http是明文
    2. md5加密密码
    3. 随机盐
5. 数据缓存
6. https

回顾