链滴

Recompact 应用浅析

作者：HuixiaZhang
原文链接：https://ld246.com/article/1545296626347
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

recompact是一款帮助你管理组件内部状态、属性、生命周期等的高阶组件库在某些场景下合理的使用他们会帮你构建更简单的代码，你还可以利用它帮你性能提升，最小化setState带来的开销。

场景 1

setState只需渲染页面单个组件，减少性能开销

```
return (
  <View style={Styles.wrap}>
    <GradientButton disabled={this.state.disabled} text={"tap"} btnType={"btn_m"} onPress={t
is._login}  />
  <View/>
);
```

这个场景下，GradientButton组件 依赖于disabled控制是否点击状态。

在常规思路下我们必须通过setState改变disabled来达到目的。

但是这里有一个问题，setState会带来整个页面的渲染开销，假设当前页面元素非常复杂，而且你不定各个页面其他的元素是否对state做了性能优化，这种情况下为了改变一个按钮的状态而去渲染整个面实在不是一个好的选择。

因此通过recompact来优化它，

```
render() {
  const enhance = compose(
      withState('disabled', 'setDisabled', false),
      withHandlers({
          disable: props => disabled => {
              props.setDisabled(disabled)}})
```

```
  )
  const Button = enhance(({disabled, disable}) => {
      this._disable  =  disable
      return <GradientButton onRef={ref=>this._ref=ref} text={"登录"} btnType={"btn_l"} disabl
d={disabled} onPress={this._login}  />
  })

  return (
    <View style={Styles.wrap}>
      <Button/>
    </View>
  );
}
```

在需要时调用this._disable(true)即可改变状态，通过recompact我们将state放在一个组件内部管理，不需要在外部调用setState带来不必要的开销

场景 2

setStatehook 生命周期方法异步渲染单个组件

```
componentDidMount(){
  NativeModule.RNBridge.getSwitchConfig(res => {
    if(res){
      res = JSON.parse(res);
      this.setState(bidDescriptionText: res.CommonConfig.BidDescriptionText);
    }
  })
}
render() {
  return (
    <View style={Styles.wrap}>
      <Text>{this.state.bidDescriptionText}</Text>
    </View>
  );
}
```

应用recompact

```
_generateWithStateText() {
  if (this._WithStateText) return this._WithStateText
  const setStateWithHandler = recompact.withStateHandlers(
    ({initialText = null}) => ({
      bidDescriptionText: initialText
    }),
    {
      setText: ({bidDescriptionText}) => (text) => ({
        bidDescriptionText:  text
      })
    }
  );
  let OnLoadText = null
  const WithStateText = setStateWithHandler(({bidDescriptionText, setText}) => {
    if (OnLoadText === null)
```

```
      OnLoadText = recompact.lifecycle({
        componentDidMount: ()=>{
          NativeModule.RNBridge.getSwitchConfig((res) => {
            if (res) {
              res = JSON.parse(res);
              setText(res.CommonConfig.BidDescriptionText)
            }
          });
        },
        componentWillUnmount: ()=>{}
      })(Text)
     return (<OnLoadText style={Styles.score_wrap_text}>{bidDescriptionText}OnLoadText>)
   });
   this._WithStateText = WithStateText
   return  WithStateText
}
render() {
  return (
    <View style={Styles.wrap}>
      <WithStateText/>
    </View>
  );
}
```

WithStateText用变量存储，避免重复初始化