react阶段测试题

1. 下列说法错误的是？(B)[单选题]
   A、React 中某个元素的 key可以确定其在同级元素中具有唯一性。
   B、setState用来修改组件状态，此过程是异步的。
   C、JSX是JavaScript的语法扩展，使用它可以高效构建UI。
   D、 React 中 refs 使我们可以访问 DOM 元素或者组件实例。

2. 下列关于组件的说法错误的是？(D)[单选题]
   A、React中组件可以将UI划分为独立、可重用的区块。
   B、组件常见的形式有函数和类两种形式。
   C、组件通过props获取父组件传递的数据，props是只读的。
   D、组件通过state维护组件状态，只有state变化会导致组件重新渲染。

3. 下列关于组件使用说法错误的是？(D)[单选题]
   A、展示组件关心组件看起来是什么，容器组件则更关心组件是如何运作的。
   B、React中通常使用受控组件实现表单，即包含表单元素的组件在state中保存输入的值，且每次事件回调中更新 state。
   C、高阶组件以组件为参数并返回一个新组件，它主要用于组件扩展和逻辑复用。
   D、通过prop.children可以获取外部传入内容，这里children是一个JSX或者JSX数组

4. 下列关于React生命周期的说法错误的是？(B)[单选题]
   A、componentDidMount在组件真正在被装载之后执行，它仅执行一次，可用于异步数据请求
   B、componentWillReceiveProps在组件接收到属性的时候调用，可用于组件初始化时处理属性
   C、shouldComponentUpdate在组件接受到新属性或者新状态的时候执行，它要求返回一个布尔值，可用于优化避免无谓执行渲染函数。
   D、组件在render函数中生成虚拟 DOM ，它会多次执行

5. 以下关于Redux说法哪项是错误的？(D)[单选题]
   A、Redux是JavaScript状态容器，提供可预测化的状态管理，主要是解决了组件间状态共享的问题
   B、Redux工作流程是view调用store的dispatch派发action，store通过reducer进行state操作，view通过store的subscribe订阅state变更，通过store的getState获取最新state
   C、Redux是通用状态容器，可用于React之外的框架。
   D、使用react-redux可以省掉状态变更订阅过程是因为使用了React的Context功能。

6. 以下关于react-router说法哪项是错误的？(C)[单选题]
   A、在react-router中仍然遵循一切皆组件思想，包括路由的配置
   B、在react-router中可能同时匹配多个路由并同时显示这些内容
   C、Route组件三个属性render/children/component是互斥的，优先执行component
   D、react-router没有提供路由权限控制功能，若要实现可以通过高阶组件扩展Route

7. 下列说法正确的是？(ABCD)[多选题]
   A、react-redux通过Context实现了store实例的传递
   B、react-redux通过Hoc扩展了组件状态变更检测及状态获取能力
   C、redux能够引入中间件是通过扩展store的dispatch实现的
   D、redux中修改状态的reducer必须是纯函数

8. 关于虚拟DOM，下列说法错误的是？(D)[单选题]
   A、虚拟DOM其实就是JavaScript对象，它能够描述 DOM 树的结构。
   B、虚拟DOM不一定能够提高性能表现。
   C、通过diff算法比较新旧虚拟DOM可以得到最少DOM操作。
   D、不使用虚拟DOM方案无法有效减少DOM操作。

9. 下列关于diff算法描述错误的是？(C)[单选题]
   A、把树形结构按照层级分解，只比较同级元素。
   B、拥有不同类型的两个元素将导致新树结构创建和替换。
   C、当比较同类型DOM元素时，仅需比较两者特性(attributes)差异并做更新即可。
   D、当比较同类型组件元素时，React更新属性后执行render并将新老vdom进行递归比对。

10. 编程题：如果你像下面这样使用组件Comp，那么它的定义是怎样的，选择正确的选项？ (AC)[多选题]

```
<Comp username="kaikeba">
  {(user) => user === null
    ? <Loading />
    : <Badge info={user} />}
</Comp>
```

A、

```
import React, { Component } from 'react'
import { fetchUser } from './api'

class Comp extends Component {
  state = { user: null }
  componentDidMount () {
    fetchUser(this.props.username)
      .then((user) => this.setState({user}))
  }
  render () {
    return this.props.children(this.state.user)
  }
}
```

B、

```
import React, { Component } from 'react'
import { fetchUser } from './api'

class Comp extends Component {
  render () {
    fetchUser(this.props.username)
      .then((user) => this.props.children(user))
  }
}
```

C、

```
import React, { useState, useEffect } from 'react'
import { fetchUser } from './api'

function Comp(props) {
  const [user, setUser] = useState(null);
  useEffect(()=>{
    fetchUser(props.username)
      .then((user) => setUser(user))
  }, []);
  return props.children(user)
}
```

D、

```
import React, { useState } from 'react'
import { fetchUser } from './api'

function Comp(props) {
  const [user, setUser] = useState(null);
  fetchUser(props.username)
      .then((user) => setUser(user))
  return props.children(user)
}
```