React基础

01 简单介绍一下

React是一个构建UI的库

React不是一个框架，它只提供UI层面的解决方案
实际项目中需要结合其他的库(Redux,React-router等)提供完整解决方案

02 开发环境

Create React App

Create React apps with no build configuration.

- Getting Started – How to create a new app.
- User Guide – How to develop apps bootstrapped with Create React App.

Create React App works on macOS, Windows, and Linux.
If something doesn't work please file an issue.

tl;dr

```
npm install -g create-react-app

create-react-app my-app
cd my-app/
npm start
```

Then open http://localhost:3000/ to see your app.
When you're ready to deploy to production, create a minified bundle with `npm run build`.

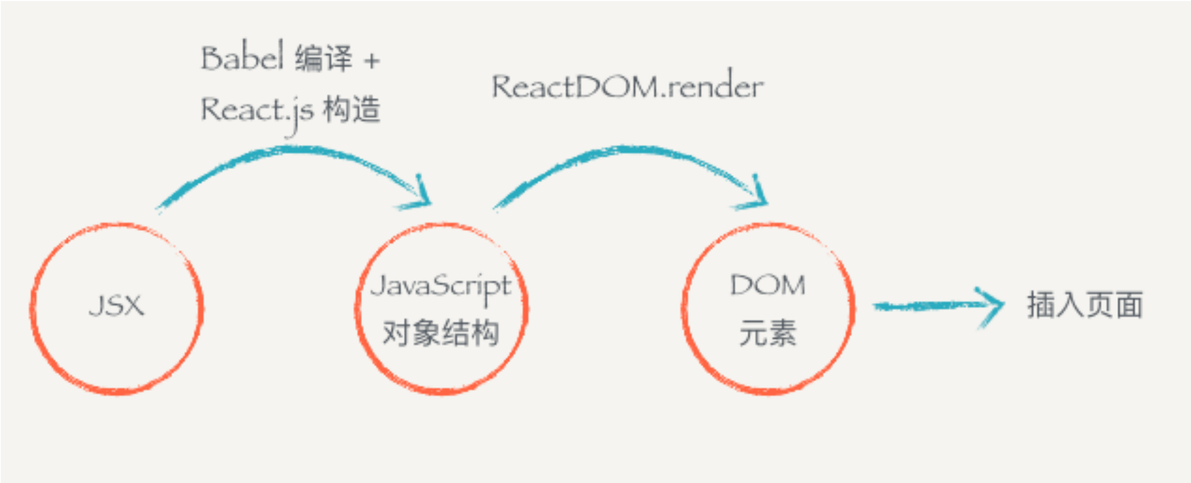
项目地址：https://github.com/facebookincubator/create-react-app

03 语法使用

JSX语法

长得像 HTML，但并不是 HTML

JSX其实就是JavaScript对象



如何用JavaScript表示DOM

```
<div class='box' id='content'>
  <div class='title'>Hello</div>
  <button>Click</button>
</div>
```

```
{
  tag: 'div',
  attrs: { className: 'box', id: 'content'},
  children: [
    {
      tag: 'div',
      arrts: { className: 'title' },
      children: ['Hello']
    },
    {
      tag: 'button',
      attrs: null,
      children: ['Click']
    }
  ]
}
```

表达式插入

```
...
render () {
  const word = 'is good'
  return (
    <div>
      <h1>React 小书 {word}</h1>
    </div>
  )
}
...
```

元素变量

```
...
render () {
  const isGoodWord = true
  const goodWord = <strong> is good</strong>
  const badWord = <span> is not good</span>
  return (
    <div>
      <h1>
        React 小书
        {isGoodWord ? goodWord : badWord}
      </h1>
    </div>
  )
}
...
```

条件返回

```
...
render () {
  const isGoodWord = true
  return (
    <div>
      <h1>
        React 小书
        {isGoodWord
          ? <strong> is good</strong>
          : <span> is not good</span>
        }
      </h1>
    </div>
  )
}
...
```

注意：JSX没有if...else语法

渲染列表

```
const users = [
  { username: 'Jerry', age: 21, gender: 'male' },
  { username: 'Tomy', age: 22, gender: 'male' },
  { username: 'Lily', age: 19, gender: 'female' },
  { username: 'Lucy', age: 20, gender: 'female' }
]

class Index extends Component {
  render () {
    const usersElements = [] // 保存每个用户渲染以后 JSX 的数组
    for (let user of users) {
      usersElements.push( // 循环每个用户，构建 JSX，push 到数组中
        <div>
          <div>姓名：{user.username}</div>
          <div>年龄：{user.age}</div>
          <div>性别：{user.gender}</div>
          <hr />
        </div>
      )
    }

    return (
      <div>{usersElements}</div>
    )
  }
}

ReactDOM.render(
  <Index />,
  document.getElementById('root')
)
```

注意：JSX语法没有for语法支持，但是可以用map

组合嵌套

```
class Title extends Component {
  render () {
    return (
      <h1>React 小书</h1>
    )
  }
}

class Header extends Component {
  render () {
    return (
      <div>
        <Title />
      </div>
    )
  }
}
```

事件监听

```
class Title extends Component {
  handleClickOnTitle () {
    console.log('Click on title.')
  }

  render () {
    return (
      <h1 onClick={this.handleClickOnTitle}>React 小书</h1>
    )
  }
}
```

```
class Title extends Component {
  handleClickOnTitle (e) {
    console.log(e.target.innerHTML)
  }

  render () {
    return (
      <h1 onClick={this.handleClickOnTitle}>React 小书</h1>
    )
  }
}
```

注意：事件绑定中要注意this的变化，bind或lambda

组件的state和setState

```
import React, { Component } from 'react'
import ReactDOM from 'react-dom'
import './index.css'

class LikeButton extends Component {
  constructor () {
    super()
    this.state = { isLiked: false }
  }

  handleClickOnLikeButton () {
    this.setState({
      isLiked: !this.state.isLiked
    })
  }

  render () {
    return (
      <button onClick={this.handleClickOnLikeButton.bind(this)}>
        {this.state.isLiked ? '取消' : '点赞'} 👍
      </button>
    )
  }
}
...
```

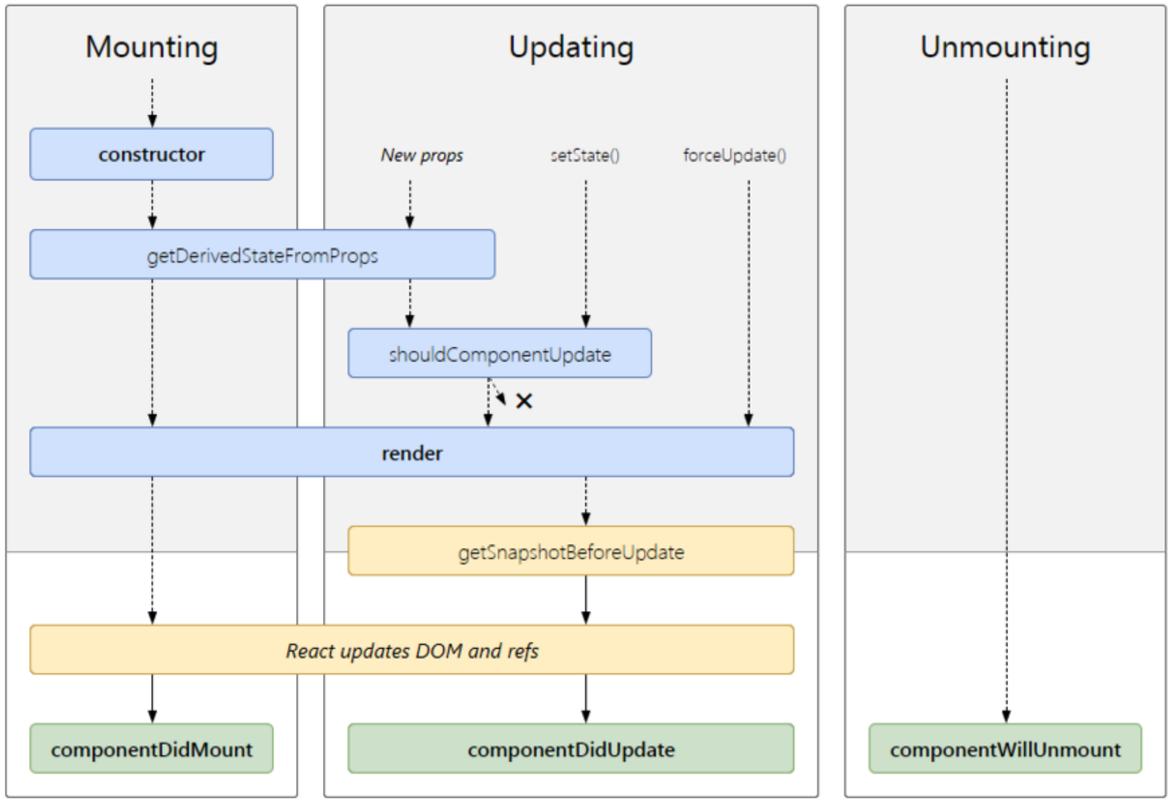
组件的props

```
class LikeButton extends Component {
  constructor () {
    super()
    this.state = { isLiked: false }
  }

  handleClickOnLikeButton () {
    this.setState({
      isLiked: !this.state.isLiked
    })
  }

  render () {
    const likedText = this.props.likedText || '取消'
    const unlikedText = this.props.unlikedText || '点赞'
    return (
      <button onClick={this.handleClickOnLikeButton.bind(this)}>
        {this.state.isLiked ? likedText : unlikedText} 👍
      </button>
    )
  }
}
```

```
class Index extends Component {
  render () {
    return (
      <div>
        <LikeButton likedText='已赞' unlikedText='赞' />
      </div>
    )
  }
}
```

组件管理

组件声明周期


Mounting
Updating
Unmounting
constructor
New props
setState()
forceUpdate()
getDerivedStateFromProps
shouldComponentUpdate
render
getSnapshotBeforeUpdate
React updates DOM and refs
componentDidMount
componentDidUpdate
componentWillUnmount

常用技巧

ref和组件中的DOM操作

```
class AutoFocusInput extends Component {
  componentDidMount () {
    this.input.focus()
  }

  render () {
    return (
      <input ref={(input) => this.input = input} />
    )
  }
}

ReactDOM.render(
  <AutoFocusInput />,
  document.getElementById('root')
)
```

props.children&容器组件

```
class Card extends Component {
  render () {
    return (
      <div className='card'>
        <div className='card-content'>
          {this.props.children}
        </div>
      </div>
    )
  }
}
```

```
ReactDOM.render(
  <Card>
    <h2>React.js 小书</h2>
    <div>开源、免费、专业、简单</div>
    订阅：<input />
  </Card>,
  document.getElementById('root')
)
```

dangerouslySetHTML

```
class Editor extends Component {
  constructor() {
    super()
    this.state = {
      content: '<h1>React.js 小书</h1>'
    }
  }

  render () {
    return (
      <div className='editor-wrapper'>
        {this.state.content}
      </div>
    )
  }
}
```


```
...
  render () {
    return (
      <div
        className='editor-wrapper'
        dangerouslySetInnerHTML={{__html: this.state.content}} />
    )
  }
...
```

Style

```
<h1 style='font-size: 12px; color: red;'>React.js 小书</h1>
```

```
<h1 style={{fontSize: '12px', color: 'red'}}>React.js 小书</h1>
```

```
<h1 style={{fontSize: '12px', color: this.state.color}}>React.js 小书</h1>
```

PropTypes和参数验证

```
class Comment extends Component {
  const { comment } = this.props
  render () {
    return (
      <div className='comment'>
        <div className='comment-user'>
          <span>{comment.username} </span>:
        </div>
        <p>{comment.content}</p>
      </div>
    )
  }
}
```

```
<Comment comment={1} />
```

```
▶ Uncaught TypeError: Cannot read property 'username' of undefined
    at Comment.render (index.js:108)
    at ReactCompositeComponent.js:796
    at measureLifeCyclePerf (ReactCompositeComponent.js:75)
    at ReactCompositeComponentWrapper._renderValidatedComponentWithoutOwner
    at ReactCompositeComponentWrapper._renderValidatedComponent (ReactCompo
```

PropTypes和参数验证

```
import React, { Component } from 'react'
import PropTypes from 'prop-types'

class Comment extends Component {
  static propTypes = {
    comment: PropTypes.object
  }

  render () {
    const { comment } = this.props
    return (
      <div className='comment'>
        <div className='comment-user'>
          <span>{comment.username} </span>:
        </div>
        <p>{comment.content}</p>
      </div>
    )
  }
}
```

```
▶ Warning: Failed prop type: Invalid prop `comment` of type `number` supplied to `Comment`, expected `object`.
    in Comment (at index.js:376)
```

HOC高阶组件

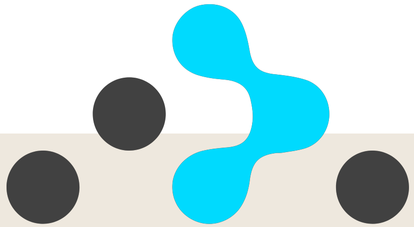
高阶组件就是一个函数，传给它一个组件，它返回一个新的组件。

```
import React, { Component } from 'react'

export default (WrappedComponent) => {
  class NewComponent extends Component {
    // 可以做很多自定义逻辑
    render () {
      return <WrappedComponent />
    }
  }
  return NewComponent
}
```

```
import React, { Component } from 'react'

export default (WrappedComponent, name) => {
  class NewComponent extends Component {
    constructor () {
      super()
      this.state = { data: null }
    }

    componentWillMount () {
      let data = localStorage.getItem(name)
      this.setState({ data })
    }

    render () {
      return <WrappedComponent data={this.state.data} />
    }
  }
  return NewComponent
}
```

04 路由react-router

react-router构成



REACT-ROUTER

核心路由组件和函数

REACT-ROUTER-DOM

浏览器环境所需的特定组件

REACT-ROUTER-NATIVE

react-native环境所需的特定组件

History

1

hashHistory

使用浏览器hash实现的前端路由模式，url中存在“#”，通常在老版浏览器或无服务端支持的时候使用

2

browserHistory

使用H5的history api实现的前端路由模式，url和正常网址一样，最常用的方式，但是需要服务端支持

3

memoryHistory

非DOM环境使用：react-native or ssr

使用

```
import React from 'react'
import {
  BrowserRouter as Router,
  Route,
  Link
} from 'react-router-dom'

const BasicExample = () => (
  <Router>
    <div>
      <ul>
        <li><Link to="/">首页</Link></li>
        <li><Link to="/about">关于</Link></li>
        <li><Link to="/topics">主题列表</Link></li>
      </ul>

      <hr/>

      <Route exact path="/" component={Home}/>
      <Route path="/about" component={About}/>
      <Route path="/topics" component={Topics}/>
    </div>
  </Router>
)
```

URL参数

```
import React from 'react'
import {
  BrowserRouter as Router,
  Route,
  Link
} from 'react-router-dom'

const ParamsExample = () => (
  <Router>
    <div>
      <h2>账号</h2>
      <ul>
        <li><Link to="/react-router">React Router</Link></li>
        <li><Link to="/leoashin">LeoAshin</Link></li>
        <li><Link to="/justjavac">justjavac</Link></li>
        <li><Link to="/reacttraining">React Training</Link></li>
      </ul>

      <Route path="/:id" component={Child}/>
    </div>
  </Router>
)

const Child = ({ match }) => (
  <div>
    <h3>ID: {match.params.id}</h3>
  </div>
)
```

课后作业

编写博客列表和详情的简单页面，详情页增加点赞功能

Thanks