React02

生成项目: `create-react-app 项目名(只支持小写)`

启动项目: `npm start`

作业

```
// rcc
import React, { Component } from "react";

export default class App extends Component {
  items = ["吃饭", "睡觉", "写代码"];

  // 页面的数据会动态变化, 则此数据一定是放在 state 中
  state = {
    item: "",
    tags: ["SQL", "HTML", "MYSQL", "VUE", "ANGULAR", "REACT", "PYTHON"],
  };

  showTags = () => {
    return this.state.tags.map((item, index) => {
      return (
        <li key={index}>
          <span>{item}</span>
          <button onClick={this.delTag.bind(this, index)}>删除</button>
        </li>
      );
    });
  };

  delTag(index) {
    // splice 会直接修改 数组
    this.state.tags.splice(index, 1);
    // this.setState({ tags: this.state.tags });

    // setState: 有两个作用 -- 更新UI && 更新数据
    this.setState({});
  }

  showItems() {
    return this.items.map((item, index) => {
      return (
        <button key={index} onClick={() => this.chooseItem(index)}>
          {item}
        </button>
      );
    });
  }

  // 参数: 项目的序号
  chooseItem(index) {
```

```
    // 需求: 根据序号 找到对应的项目名, 然后动态显示到页面上 --- 数据变UI变 -- state
    this.setState({ item: this.items[index] });
    // setState: 有两个任务 --  更新UI 和 更新数据
  }

  render() {
    return (
      <div>
        <p>作业1</p>
        <h4>
          丽莎小姐姐晚上的娱乐项目:
          <span style={{ color: "blue" }}>{this.state.item}</span>
        </h4>
        <div>{this.showItems()}</div>

        <hr />
        <ul>{this.showTags()}</ul>
      </div>
    );
  }
}
```

双向数据绑定

```
// rcc
import React, { Component } from "react";

// input 双向数据绑定
// vue中: v-model
// ng中: ngModel
// react: js原生方式实现
export default class App extends Component {
  // 双向绑定:
  // 方向1: 数据变 UI变 -- 必须state配合
  // 方向2: UI变 数据变
  state = { word: "666" };

  // 接收事件传参: 凡是通过事件触发的函数 都会默认接收到事件本身作为最后一个参数!
  inputChanged(e) {
    // 想要看打印, 必须调用 persist() 方法
    e.persist();
    console.log(e);

    // 读取输入框中的新值
    let word = e.target.value;
    this.setState({ word });
  }

  render() {
    return (
      <div>
        {/* 由于输入框的value绑定了值, 所以人为录入效果失效! */}
        {/* 此时必须明确的告知: 此输入框 是 只读的 还是 双向绑定的? */}
```

```
        {/* readOnly: 明确表示当前输入框 是只读的 */}
        <input type="text" value={this.state.word} readOnly />
        {/* 使用 onChange 来监听用户的录入操作; 实现双向绑定 */}
        <input
          type="text"
          value={this.state.word}
          onChange={this.inputChanged.bind(this)}
        />
        <br />
        {/* onChange事件触发的是 箭头函数, 所以箭头函数会接收 事件传参 */}
        <input
          type="text"
          value={this.state.word}
          onChange={(e) => this.inputChanged(e)}
        />
        {/* (e) => { this.inputChanged(e) } */}

        <br />
        <b>输入框的内容: {this.state.word}</b>
      </div>
    );
  }
}
```

条件渲染

```
// rcc

import React, { Component } from "react";

// if 条件渲染
// vue: v-if    ng: *ngIf
// react: 使用原生的if 即可
export default class App extends Component {
  // if判断会造成 UI的动态变化, 所以必须配合state
  state = { score: 50 };

  showRes() {
    // if判断 必须要配合state;  当state发生更新时, 系统会自动检测 与state 有关的代码 并进行刷新
    if (this.state.score < 60) {
      return <h2>还不够, 必须找东哥提升...</h2>;
    } else if (this.state.score >= 60 && this.state.score < 80) {
      return <h2>刚及格, 必须再找东哥...</h2>;
    } else {
      return <h2>不错不错, 基础很扎实!</h2>;
    }
  }

  render() {
    return (
      <div>
        <h4>丽莎小姐姐如果去找东哥学习, 就会得到 {this.state.score} 分</h4>
```

```
        <button onClick={() => this.setState({ score: this.state.score + 10 })}>
          找东哥一次
        </button>

        {this.showRes()}
      </div>
    );
  }
}
```

生命周期

```
import React, { Component } from "react";

class Son extends Component {
  state = { num: 1 };

  // vue的 mounted
  componentDidMount() {
    console.log("componentDidMount: 组件挂载时");
  }

  // 组件在显示期间，内容发生变更时
  componentDidUpdate(props, state) {
    // props: 父传入的参数
    // state: 自身的参数
    console.log("componentDidUpdate:", props, state);
  }

  // 面试题经常问的: 如果提高react的渲染效率?
  // 当数据变化时，会先询问此方法；  此方法的返回值将决定是否要 刷新UI/重新渲染 页面
  shouldComponentUpdate(props, state) {
    console.log(props, state);

    // 值显示 偶数
    if (state.num % 2 == 0) {
      return true;
    }

    return false; //不重新渲染页面
  }

  componentWillUnmount() {
    console.log("componentWillUnmount: 组件卸载时");
  }

  render() {
    return (
      <div>
        <h1>我是Son组件</h1>
        <button onClick={() => this.setState({ num: this.state.num + 1 })}>
          {this.state.num}
        </button>
```

```
      </div>
    );
  }
}

// 生命周期
export default class App extends Component {
  state = { show: false, age: 18 };

  showSon() {
    if (this.state.show) {
      return <Son age={this.state.age} />;
    }
  }

  render() {
    return (
      <div>
        <button onClick={() => this.setState({ show: !this.state.show })}>
          切换显示状态
        </button>
        <button onClick={() => this.setState({ age: this.state.age + 1 })}>
          年龄+1
        </button>
        {this.showSon()}
      </div>
    );
  }
}
```

网络请求

react与 vue 一样, 本身不具备网络请求功能模块, 需要采用第三方模块: axios

在项目目录下, 打开命令, 进行安装

```
npm i axios
```

```
+ axios@0.19.2
added 253 packages from 135 contributors, removed 226 packages and updated 1369 packages in 304.
012s
```

```
import React, { Component } from "react";

// 引入axios:  前提是 npm i axios 进行模块安装
// 安装之后, 最好是重启;  否则可能会找不到模块 报错!  module not found
import axios from "axios";

export default class App extends Component {
  // ts语言: 才有静态类型特征, 可以声明类型 出代码提示;    此处是js, 没有这些写法

  state = { data: null };
```

```
  componentDidMount() {
    let url = "https://api.apiopen.top/getImages";

    axios
      .get(url)
      .then((res) => {
        console.log(res);
        // 数据下载完毕后，需要页面实时刷新出：  数据变 UI变
        this.setState({ data: res.data });
      })
      .catch((err) => console.log(err));
  }

  showGirls() {
    // 网络请求的异步性：先判断有值再使用
    if (this.state.data) {
      return this.state.data.result.map((item, index) => {
        return <img key={index} src={item.img} style={{ width: "100px" }} />;
      });
    }
  }

  render() {
    return <div>{this.showGirls()}</div>;
  }
}
```

练习

音乐排行榜: https://api.apiopen.top/musicRankings


```
import React, { Component } from "react";

import axios from "axios";

import "./App.css";
```

```
export default class App extends Component {
  // 数据下载完毕后，页面要随着刷新出内容
  state = { data: null };

  componentDidMount() {
    let url = "https://api.apiopen.top/musicRankings";

    axios
      .get(url)
      .then((res) => {
        console.log(res);

        this.setState({ data: res.data });
      })
      .catch((err) => console.log(err));
  }

  showRank() {
    if (this.state.data) {
      return this.state.data.result.map((item, index) => {
        return (
          <div key={index}>
            <div className="cell">
              <img src={item.pic_s444} alt="" />
              <div>
                <span>{item.name}</span>
                <span>{item.comment}</span>
              </div>
            </div>

            {/* 添加content 中的内容 */}
            {item.content.map((item1, index1) => {
              return (
                <div key={index1} className="content">
                  <img src={item1.pic_big} alt="" />
                  <div>
                    <span>歌曲名: {item1.title}</span>
                    <span>作者: {item1.author}</span>
                    <span>专辑: {item1.album_title}</span>
                  </div>
                </div>
              );
            })}
          </div>
        );
      });
    }
  }

  render() {
    return <div>{this.showRank()}</div>;
  }
}
```

```
.cell {
  width: 500px;
  border-bottom: 1px solid gold;
  display: flex;
}

.cell > img {
  height: 120px;
}

.cell > div {
  display: flex;
  flex-direction: column;
}

.cell > div > span:first-child {
  font-size: 1.2em;
  color: blue;
}

.content {
  background-color: aliceblue;
  padding: 2px;
  display: inline-block;
  margin: 2px;
}

.content > div {
  display: flex;
  flex-direction: column;
}
```

新闻列表练习

接口地址:

`http://101.96.128.94:9999/mfresh/data/news_select.php?pageNum=1`

参数: pageNum 代表页数

App.js

```
import React, { Component } from "react";
import Axios from "axios";
```

```
import "./News.css";

export default class App extends Component {
  state = { data: null };

  // 某个变量在多个方法中使用，则应该提取成 成员属性；  达到复用效果
  url = "http://101.96.128.94:9999/mfresh/data/news_select.php?pageNum=";

  componentDidMount() {
    Axios.get(this.url + 1).then((res) => {
      console.log(res);

      this.setState({ data: res.data });
    });
  }

  showNews() {
    if (this.state.data) {
      return this.state.data.data.map((item, index) => {
        return (
          <div key={index}>
            <span>{item.title}</span>
            {/* react采用原生的函数方式，没有vue和ng 的 {xxx | xxx } 写法 */}
            <span>{this.date(item.pubTime)}</span>
          </div>
        );
      });
    }
  }

  // ng 和 vue 提供管道 来对 双标签的内容进行变化;
  // react 使用原生的js函数来解决
  date(timestamp) {
    let ts = parseInt(timestamp); //时间戳必须是数字类型
    let date = new Date(ts);

    let year = date.getFullYear();

    let month = date.getMonth() + 1; //此方法给的是0~11  +1才可以变1~12
    if (month < 10) month = "0" + month; // 1 -> 01

    let day = date.getDate();
    if (day < 10) day = "0" + day; // 1 -> 01

    return `${year}-${month}-${day}`;
  }

  // 获取指定页数数据
  getData(pno) {
    Axios.get(this.url + pno).then((res) => {
      console.log(res);

      this.setState({ data: res.data });
    });
  }

  showPages() {
    if (this.state.data) {
```

```
      let arr = [];

      for (let i = 1; i <= this.state.data.pageCount; i++) {
        // react希望用户写 === 代替 == , 更加严格标准;  ==后台会黄色警告
        if (this.state.data.pageNum === i) {
          arr.push(
            <span key={i} className="page-cur page">
              {i}
            </span>
          );
        } else {
          arr.push(
            <span key={i} className="page" onClick={this.getData.bind(this, i)}>
              {i}
            </span>
          );
        }
      }

      return arr;
    }
  }

  // 显示上一页
  showPrev() {
    // 网络请求结束后，才能使用
    if (this.state.data) {
      // 当前是第一页，则上一页不能点
      if (this.state.data.pageNum === 1) {
        return <span className="pg-disable">上一页</span>;
      } else {
        return (
          <span
            className="pg"
            onClick={() => this.getData(this.state.data.pageNum - 1)}
          >
            上一页
          </span>
        );
      }
    }
  }

  // 下一页
  showNext() {
    if (this.state.data) {
      if (this.state.data.pageNum === this.state.data.pageCount) {
        return <span className="pg-disable">下一页</span>;
      } else {
        return (
          <span
            className="pg"
            onClick={this.getData.bind(this, this.state.data.pageNum + 1)}
          >
            下一页
          </span>
        );
      }
```

```
    }
  }

  render() {
    return (
      <div className="news">
        <div className="news-content">{this.showNews()}</div>
        <div className="news-pages">
          {this.showPrev()}
          {this.showPages()}
          {this.showNext()}
        </div>
      </div>
    );
  }
}
```

News.css

```
.news {
  width: 800px;
  margin: 0 auto;
}

.news-content > div {
  border-bottom: 1px dashed gray;
  padding: 5px;
  display: flex;
  justify-content: space-between;
}

.news-pages {
  margin-top: 10px;
  text-align: center;
}

.page,
.pg {
  padding: 2px 10px;
  border: 1px solid gray;
  display: inline-block;
  margin: 0 2px;
}

.page:hover,
.page-cur {
  cursor: pointer;
  background-color: orange;
  color: white;
  border-color: orange;
}

.pg:hover {
  border-color: orange;
  color: orange;
  cursor: pointer;
```

```
}

.pg-disable {
  padding: 2px 10px;
  border: 1px solid lightgray;
  color: lightgray;
  display: inline-block;
  margin: 0 2px;
}

* {
  user-select: none;
}
```

父子传参 和 子父传参

```
import React, { Component } from "react";

class Son extends Component {
  // props: 此属性是用来接收 父子传参的

  render() {
    return (
      <div>
        <p>子父传参: 选择喜欢的选手:</p>
        <div>
          <button onClick={() => this.props.choose("The Shy")}>The Shy</button>
          <button onClick={() => this.props.choose("Rookie")}>Rookie</button>
          <button onClick={() => this.props.choose("Faker")}>Faker</button>
          <button onClick={() => this.props.choose("丽莎")}>丽莎</button>
        </div>
      </div>
    );
  }
}

// 子父传参
export default class App extends Component {
  state = { player: "" };

  // 父传递一个方法 给子， 方法中的this必须永远指向父
  // 两种方式： 1.箭头函数 2.普通bind this
  choosePlayer = (player) => {
    this.setState({ player });
  };

  render() {
    return (
      <div>
        {/* 利用父子传参的写法，把函数传给子 */}
        <Son choose={this.choosePlayer} />
        <p>儿子喜欢的选手是: {this.state.player}</p>
      </div>
```

```
    );
  }
}
```

周五要使用的资源, 提前下载

之前让下载过!

```
链接：https://pan.baidu.com/s/1kdaVOFeR3eTSJDOdiIjRlQ
提取码：ddqk
```

作业

下方两个接口都是可以使用的, 任选一个接口来制作

- 第一个是获取 gif 动图
- 第二个是获取视频

```
<video src="视频地址" controls />
```

```
https://api.apiopen.top/getJoke?page=1&count=10&type=gif
https://api.apiopen.top/getJoke?page=1&count=10&type=video

参数: page是页数; count是每页数量;type是类型
```