React-2

上节课练习回顾

好友列表

```
<div class="friend-list">
    <dl class="friend-group expanded">
        <dt>家人</dt>
        <dd>爸爸</dd>
        <dd>妈妈</dd>
    </dl>
    <dl class="friend-group">
        <dt>朋友</dt>
        <dd>张三</dd>
        <dd>李四</dd>
        <dd>王五</dd>
    </dl>
    <dl class="friend-group">
        <dt>客户</dt>
        <dd>阿里</dd>
        <dd>腾讯</dd>
        <dd>头条</dd>
    </dl>
</div>
```

```
dl {
    margin: 0;
}
.friend-list {
    border: 1px solid #000000;
    width: 200px;
}
.friend-group dt {
    padding: 10px;
    background-color: rgb(64, 158, 255);
    font-weight: bold;
}
.friend-group dd {
    padding: 10px;
    display: none;
}
.friend-group.expanded dd {
    display: block;
}
```

```
let datas = {
    family: {
        title: '家人',
        list: [
            {name: '爸爸'},
```

```
            {name: '妈妈'}
        ]
    },
    friend: {
        title: '朋友',
        list: [
            {name: '张三'},
            {name: '李四'},
            {name: '王五'}
        ]
    },
    customer: {
        title: '客户',
        list: [
            {name: '阿里'},
            {name: '腾讯'},
            {name: '头条'}
        ]
    }
};
```

课程目标

- 掌握 setState 的各种使用情况
- 掌握 React 组件间通信
- 掌握 React 组件的生命周期
- 掌握受控组件的使用

课程内容

state 和 setState

- setState(updater, [callback])
    - updater: 更新数据 FUNCTION/OBJECT
    - callback: 更新成功后的回调 FUNCTION
    - 异步:react通常会集齐一批需要更新的组件，然后一次性更新来保证渲染的性能
    - 浅合并 Objecr.assign()
    - 调用 setState 之后，会触发生命周期，重新渲染组件

组件间通信

在 React.js 中，数据是从上自下流动（传递）的，也就是一个父组件可以把它的 state / props 通过 props 传递给它的子组件，但是子组件不能修改 props - React.js 是单向数据流，如果子组件需要修改父组件状态（数据），是通过回调函数方式来完成的。

- 父级向子级通信
  把数据添加子组件的属性中，然后子组件中从props属性中，获取父级传递过来的数据
- 子级向父级通信
  在父级中定义相关的数据操作方法(或其他回调), 把该方法传递给子级，在子级中调用该方法父级传递消息
- 案例：完善好友列表

跨组件通信 context - 扩展

- React.createContext(defaultValue);
  { Consumer, Provider } = createContext(defaultValue)
- Context.Provider 在父组件调用 Provider 传递数据
  - value 要传递的数据
- 接收数据
  - class.contextType = Context;
  - static contextType = Context;
    - this.context;
  - Context.Consumer

```
<Consumer>
        {(props)=>{
            console.log(props);
            return <div></div>
        }}
    </Consumer>
```

注意在使用不熟练时，最好不要再项目中使用 context，context一般给第三方库使用

组件的生命周期

所谓的生命周期就是指某个事物从开始到结束的各个阶段，当然在 React.js 中指的是组件从创建到销毁的过程，React.js 在这个过程中的不同阶段调用的函数，通过这些函数，我们可以更加精确的对组件进行控制，前面我们一直在使用的 render 函数其实就是组件生命周期渲染阶段执行的函数

生命周期演变

- 挂载阶段 （组件创建-->把组件创建的虚拟DOM，生成真实DOM，添加到我们的DOM树中）
  - constructor
  - static getDerivedStateFromProps(props)
    - 注意 this 问题
  - render
  - componentDidMount -- 处理副作用(请求)
- 更新阶段 -- 组件重新渲染
  - static getDerivedStateFromProps(props, state)
  - shouldComponentUpdate() -- 判断是否跟新
  - render()
  - getSnapshotBeforeUpdate()
  - componentDidUpdate() -- 处理副作用(请求)
- 卸载阶段
  - componentWillUnmount -- 删除添加在全局的一些信息或操作

受控组件

当想要获取表单的一些内部状态时，就可以将表单的内部状态和组件的状态进行绑定，这样就形成受控组件
受控组件: 让 表单控件 的内部状态 和我们 state 保持一致
非受控组件: 我们不需要同步 value 值(defaultValue，defaultChecked)

todoList 初实现

下节课内容

- 掌握React其他 API 使用：PureComponent、ref、children、dangerouslySetInnerHTML、key

练习

完成简易留言板效果

- 利用 React 类组件完成留言板效果
- 该效果具备以下功能
  - 将 state 中的数据展示到留言板上
  - 添加留言
  - 删除留言