React 特点

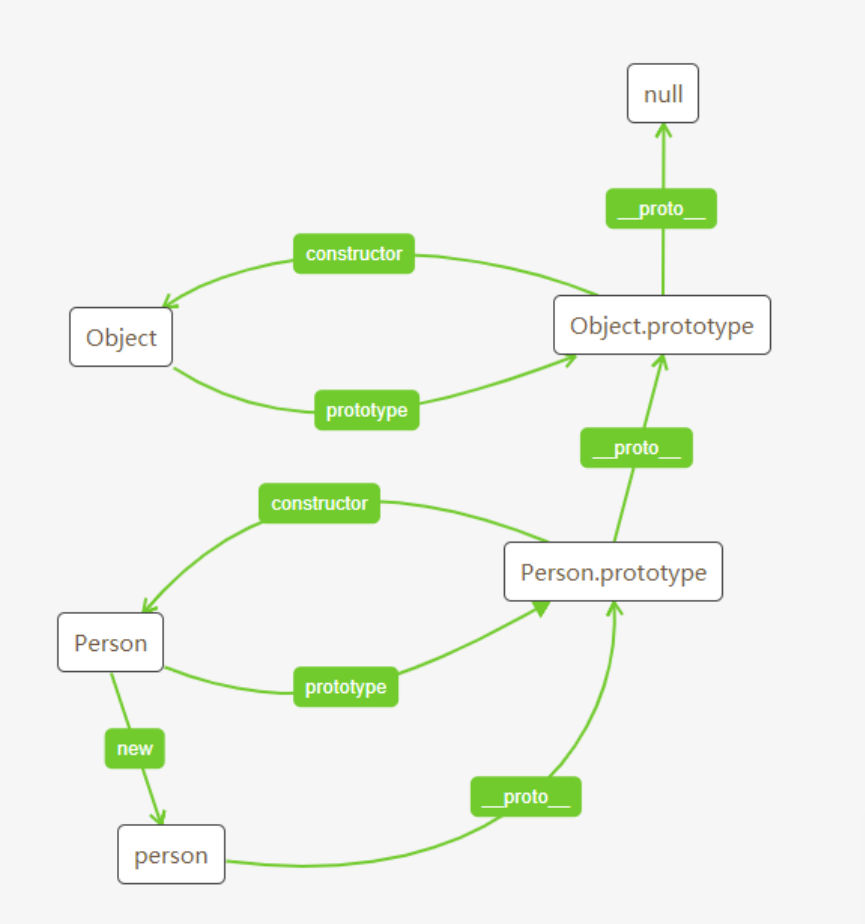
- 采用组件化模式, 声明式编码, 提高开发效率和组件复用率.
- 在 React Native 中可以使用 React 语法移动端开发.
- 使用 虚拟DOM+Diffing算法, 减少与真实 DOM 的互动.

虚拟 DOM	本质 Object 类型的对象, 一般对象. 虚拟 DOM 仅在 React 内部使用, 没有真实 DOM 那么多的属性. 虚拟 DOM 最后会被 React 转换为真实 DOM, 呈现在页面上.
真实 DOM	普通的 html 真实 DOM.

JSX 语法规则

- 定义虚拟DOM的时候, 不要写引号.
- 标签中混入JS表达式的时候要添加花括号{}.
  - 一定注意区分JS语句(代码)和JS表达式!
    - 表达式: 可以放在任何一个需要值的地方.(const x = 表达式;)
      - a+b
      - demo(1)
      - arr.map((num)=>{return num+1})
      - function test () {console.log('@')}
    - 语句(代码)
      - if () {}
      - for () {}
      - switch () {case:xxx}
- 使用 className 指定样式的类名.
- 内联样式表的写法: style={{ color: "green", fontSize: "29px" }}
- 有且仅有一个根标签,其他的标签必须闭合.
- 标签首字母
  - 小写字母 - 将标签改为 html 的同名元素, 若找不到同名元素则报错.
  - 大写字母 - 渲染对应的组件, 若找不到同名组件则报错.

原型链



React 核心库

```
    <body>
        <!-- 准备好一个容器 -->
        <div id="test"></div>
        <!-- 引入 react 核心库 -->
        <script type="text/javascript" src="../js/react.development.js"></script>
        <!-- 引入 react-dom，用于支持 react 操作 DOM -->
        <script type="text/javascript" src="../js/react-dom.development.js"></scr
        <!-- 引入 babel，用于将 jsx 转为 js -->
        <script type="text/javascript" src="../js/babel.min.js"></script>
        <script type="text/babel">
            /* 此处一定写 babel*/
            // 1. 创建虚拟 DOM
            const vDOM = <h1>Hello, React!</h1>; /*不要写引号*/
            // 2. 渲染虚拟 DOM 到页面
            ReactDOM.render(vDOM, document.getElementById("test"));
        </script>
    </body>
```

React 组件

- 函数式组件, 没有 state 的简单组件

```
function MyComponent() {
   return <h2>我是用函数定义的组件(适用于【简单组件】的定义)</h2>
}
ReactDOM.render(<MyComponent/>,document.getElementById('test'))
```

- 类式组件, 有 state 的复杂组件

```
class MyComponent extends React.Component {
    render(){
        return <h2>我是用类定义的组件(适用于【复杂组件】的定义)</h2>
    }
}
ReactDOM.render(<MyComponent/>,document.getElementById('test'))
```

组件实例的三大属性

- state (简写方式)

```
class Weather extends React.Component {
    //追加属性字段, 不需要写在构造器中
    state = { isHot: false, wind: "微风" };
    render() {
        const isHot = this.state.isHot;
        return <h2 onClick={this.changeWeather}>今天天气很{this.state.isHot ? "炎热
    }
    // 箭头函数没有自己的 this.
    // 函数体内的 this 会到外部去寻找 Weather 的实例对象.
    changeWeather = () => {
        const isHot = this.state.isHot;
        this.setState({ isHot: !isHot });
    };
}
ReactDOM.render(<Weather />, document.getElementById("test"));
```

- props - READ ONLY

```
<script type="text/babel">
    class Person extends React.Component {
        // 对标签属性的类型和必要性进行限制
        static propTypes = {
            name: PropTypes.string.isRequired,
            age: PropTypes.number,
            speak: PropTypes.func,
        };
        // 对标签的默认值进行限制
        static defaultProps = {
            gender: "male",
            age: 18,
        };
        render() {
            const { name, age, gender } = this.props;
            return (
                <ul>
                    <li>name: {name}</li>
                    <li>gender: {gender}</li>
                    <li>age: {age + 1}</li>
                </ul>
            );
        }
    }
    // 模拟一些数据
    const data = { name: "sheng", age: 20, gender: "male" };
    // 渲染组件到页面
    ReactDOM.render(<Person {...data} />, document.getElementById("test1"));
    ReactDOM.render(<Person name="linzi" age={20} gender="male" speak={speak} />,
</script>
```

- refs