React 程序开发

--- React 基础语法

河北师范大学软件学院
Software College of Hebei Normal University

内容提纲

- JSX 语法
- 元素渲染
- 组件
- 事件处理

JSX 语法

- JSX
  - JavaScript 和 XML 结合的一种格式
  - 利用 HTML 语法来创建虚拟 DOM
  - 实例：
    - const element = <h1>Hello, world!</h1>;
  - 在 JSX 中使用表达式
    - const num = 100;
    - const element = <h1> { num } </h1>;

JSX 语法

- React 元素
  - 实际上就是一个普通的对象

```
let eleObj = {
        type : 'div',
        props : {
                children : ['hello','world'],
                className : 'red',
                id : 'box'
        }
}
```

demo01

JSX 语法

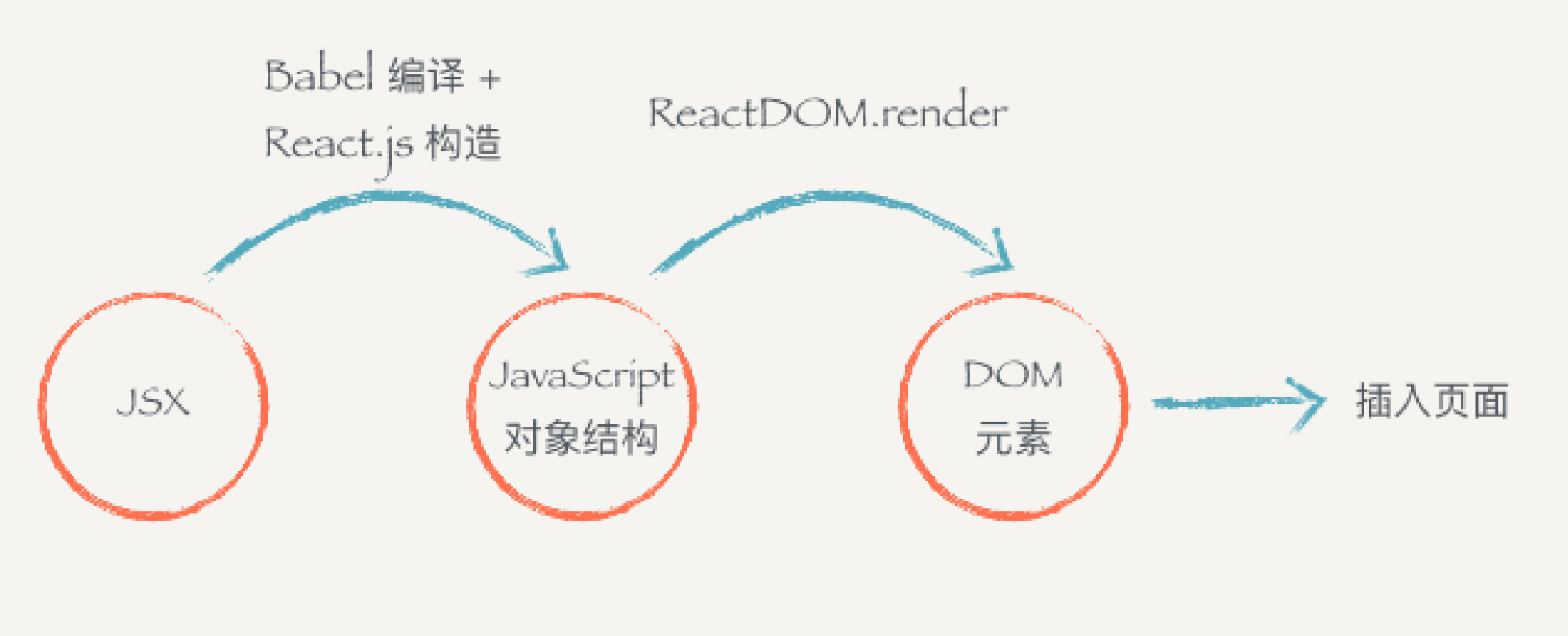
- Babel 编译
  - 把 JSX 转换成一个名为 React.createElement( ) 的方法调用，返回 JavaScript 对象
  - React.createElement(type [, props] [, ...children])
    - type：必需，元素名称
    - props：可选，元素属性
    - children：可选，子节点

内容提纲

- JSX 语法
- **元素渲染**
- 组件
- 事件处理

JSX 语法

- React 元素渲染过程



元素渲染

- “根” DOM 节点
  - 首页中添加一个 id="root" 的
  - 节点所有内容都将由 React DOM 来管理
- 元素渲染
  - 将 React 元素传递给 ReactDOM.render( ) 方法将其渲染到页面上
  - ReactDOM.render(ele,document.getElementById('root'));

demo02

元素渲染

- 更新渲染元素
  - React 元素是不可变的
  - React 元素被创建后，无法改变其内容或属性

```
function tick() {
        const ele = <div>{new Date( ).toLocaleTimeString( )}</div> ;
        ReactDOM.render(ele, document.getElementById('root'));
}
setInterval(tick, 1000);
```

  - **使用 React 的 DOM 比较算法进行高效的更新**

demo03

内容提纲

- JSX 语法
- 元素渲染
- **组件**
- 事件处理

组件

- 组件
  - React 组件是小的，可复用的代码片段
  - 从概念上看就像是函数，可以接收任意的输入值（称之为“props”）
  - 返回一个 React 元素用于渲染页面
- 组件定义方式
  - 函数定义
  - 类定义

组件

- 函数定义组件
  - 接收单一的 props 对象，返回一个 React 元素
  - props 是组件的输入内容， 从父组件传递给子组件的数据(属性)
  - 注意：props 是只读的；组件名称必须以大写字母开头

```
function Hello( props ) {
        return <h1>Hello { props.name }</h1>
}
ReactDOM.render(
        <Hello name=“ React ”/>, document.getElementById('root')
);
```

demo04

组件

- 类定义组件
  - React 提供了 React.Component 抽象基础类
  - 直接引用 React.Component 几乎没意义，通常是继承它
  - 至少定义一个 render( ) 方法

```
class Hello extends React.Component {
        render( ) {
                return <h1>Hello, {this.props.name}</h1>;
        }
}
```

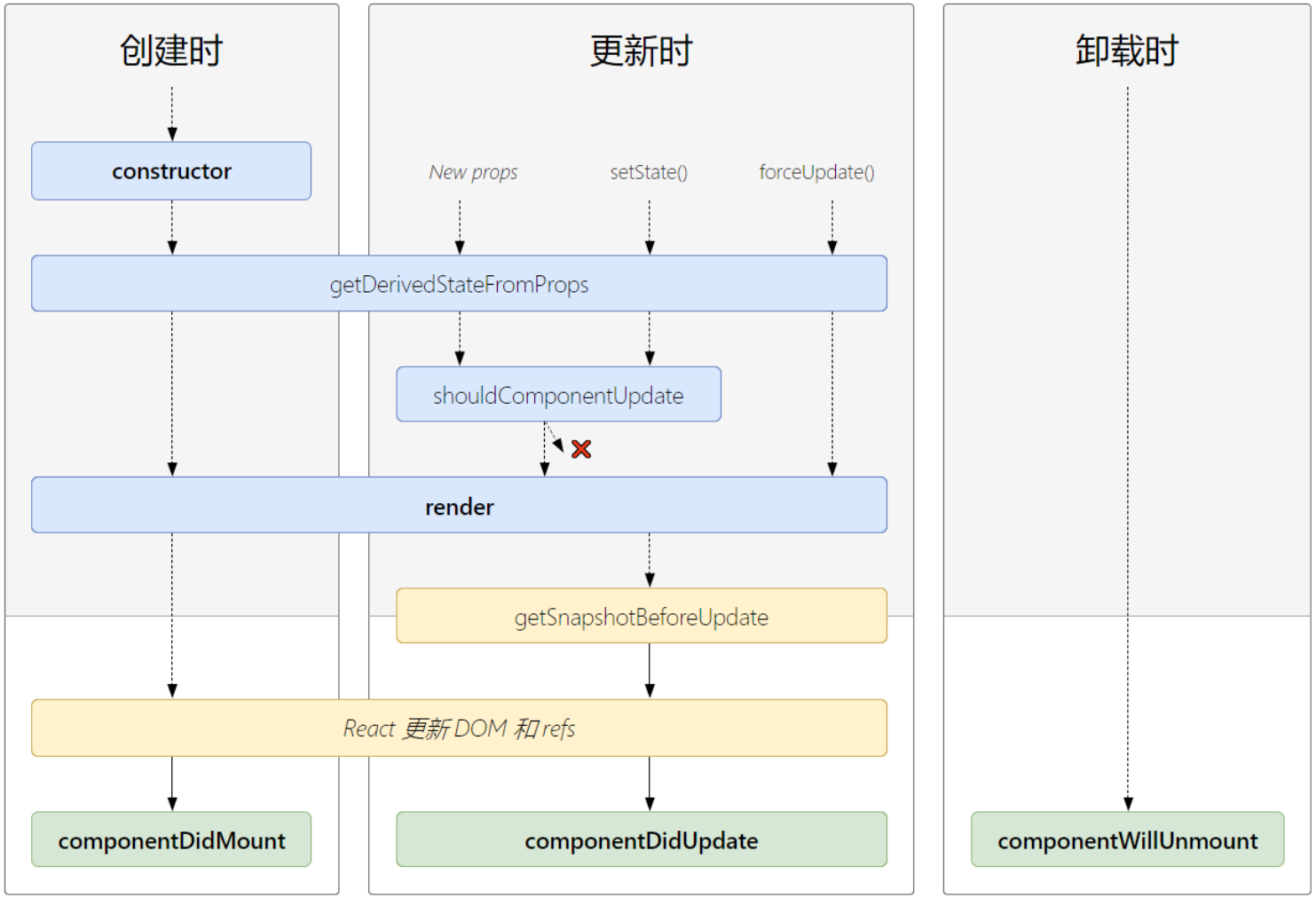
组件

- State（状态）
  - 私有的、完全受控于当前组件，组件外部是无法进行修改的
  - 类定义的组件特有的属性
  - 状态的声明（构造函数是唯一能够初始化 this.state 的地方）

```
class Hello extends React.Component {
        constructor( ){        // ES6 对类的默认方法
                super();       // 将父类中的 this 对象继承给子类
                this.state = {name:’React’}
        }
        render( ) { return <h1>Hello { this.state.name }</h1>; }
}
```

demo05

组件生命周期


创建时
更新时
卸载时
constructor
New props
setState()
forceUpdate()
getDerivedStateFromProps
shouldComponentUpdate
render
getSnapshotBeforeUpdate
React 更新 DOM 和 refs
componentDidMount
componentDidUpdate
componentWillUnmount

组件生命周期

- 生命周期函数
  - 指在某一个时刻组件会自动执行的函数
  - 只在类定义的组件中才有生命周期函数，函数方式定义的没有
  - 周期生命周期包含的阶段
    - 初始化
    - 挂载
    - 更新
    - 卸载
    - 错误

组件

- 初始化
  - constructor( )
    - 会在其装载之前被调用
    - 在函数内应该在任何其他的表达式之前调用super(props)，否则，this.props 在构造函数中将是未定义的
  - 作用:
    - 初始化状态，通过赋值一个对象到 this.state
    - 绑定事件处理函数到一个实例

demo06

组件

- 挂载
  - static getDerivedStateFromProps( )
    - 组件实例化后或接受新属性时将会被调用
    - 应该返回一个对象来更新状态，或者返回 null 来表明新属性不需要更新任何状态
  - render( )
    - 类组件唯一必须的方法
  - componentDidMount( )
    - 组件挂载后立即调用，发送请求的好地方

组件

- 更新
  - static getDerivedStateFromProps( )
  - shouldComponentUpdate( )
    - 当接收到新属性或状态时，在渲染前被调用，返回布尔值
  - render( )
  - getSnapshotBeforeUpdate( )
    - 在最新的渲染输出提交给 DOM 前将会立即调用
    - 该函数返回的任何值将会 作为参数被传递给componentDidUpdate
  - componentDidUpdate( )

组件

- 卸载
  - componentWillUnmount( )
- 错误处理
  - componentDidCatch( )

内容提纲

- JSX 语法
- 元素渲染
- 组件
- **事件处理**

事件绑定

- 事件绑定
  - React 事件绑定属性的命名采用驼峰式写法，而不是小写
  - 采用 JSX 语法需传入一个函数作为事件处理函数，而不是一个字符串(DOM 元素的写法)

```
handleClick = ( ) => { }
….
<button onClick={ handleClick }>
        Click
</button>
```

demo07

this

- 事件处理函数绑定 this
  - 类的方法默认是不会绑定 this 的
  - 通过 bind 绑定 this （两种形式）

```
constructor( ){
        super();
        this.handleClick = this.handleClick.bind( this );
}
<button onClick={ this.handleClick.bind( this ) }>
        Click
</button>
```

this

- 事件处理函数绑定 this （续）
  - 箭头函数

```
class Hello extends React.Component {
        handleChange = (  ) => {  }
        render( ) {
                return (
                        <button onClick={ this.handleClick }>
                                Click
                        </button>
                )
        }
}
```

事件处理函数传参

- 传参(两种形式)
  - 函数声明时事件对象作为最后一个参数传入
  - 箭头函数的事件对象显示传入；bind 会隐式传入

```
deleteRow = ( id, e ) => {  }
…
<button onClick={(e) => this.deleteRow(id, e)}>
        Delete Row
</button>
<button onClick={this.deleteRow.bind(this, id)}>
        Delete Row
</button>
```

demo08

Thank You

河北师范大学软件学院
Software College of Hebei Normal University