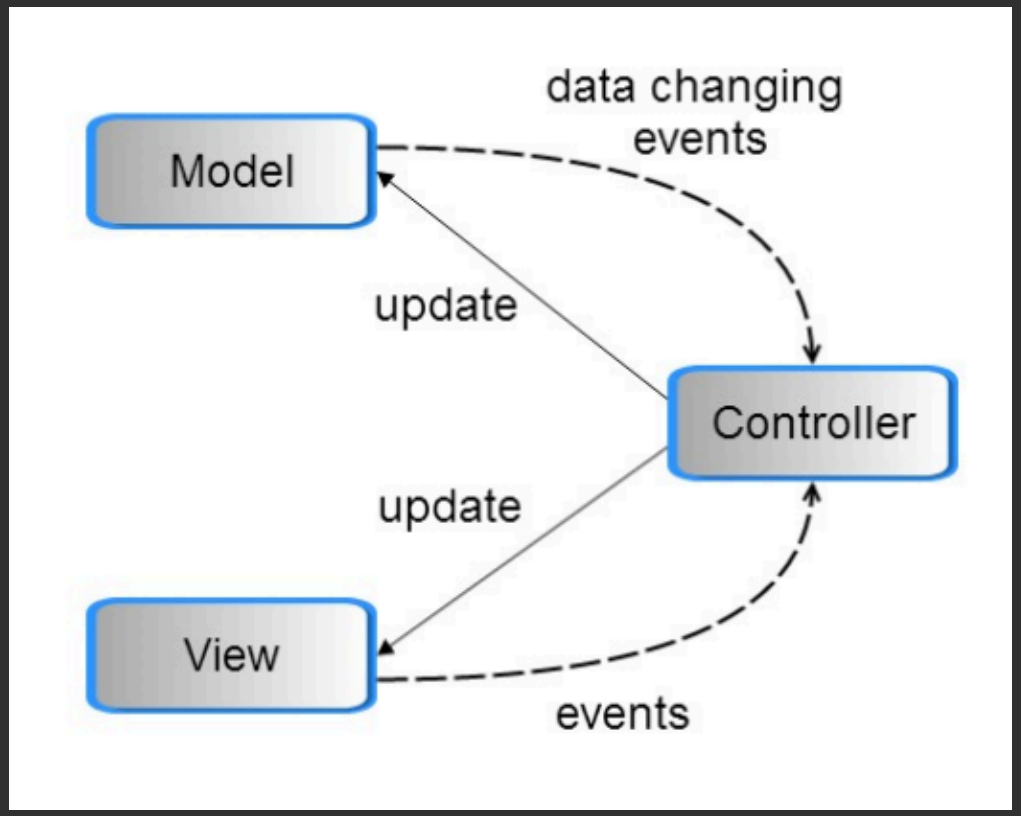
React

前端框架一般基于MVC模式



React 用称之为Component 的来代替了View 层

什么是React？

React 是一个 Facebook 和 Instagram 用来创建用户界面的 JavaScript 库。（--中文官方）

初识Demo　Hello world

http://jsfiddle.net/reactjs/69z2wepo/

//without jsx

http://jsfiddle.net/reactjs/5vjqabv3/

JSX

JSX 是javascript XML——一种在React组件内部构建标签的类xml语法。

使用**jsx**的好处

- 更加熟悉
- 更加语义化
- 更加直观

它能定义简洁且我们熟知的包含属性的树状结构语法。

JSX

举个例子

```
var HelloWorld = React.createClass({
  render: function() {
    return (
      <p>
        Hello, <input type="text" placeholder="Your name here" />!
        It is {this.props.date.toTimeString()}
      </p>
    );
  }
});
```

http://localhost:8010/react/index1.html

JSX

离线转换工具(依赖npm)

```
npm install -g react-tools

jsx --watch src/ build/
```

数据流

在Reactjs中，数据的流向是单向的—从父节点传递到子节点。

两种数据传递属性

--props　可以使用任意类型的数据传递给组件
--state　　每个组件都拥有自己的state,可以用来确定一个元素的视图状态。

数据流--props

例:

```
var InputReact = new React.createClass({
    render: function () {
      return (
        <input value = {this.props.name} name='uname' />
      );
    }
});

React.render(<InputReact name='jime' />,
document.getElementById('example'));
```

http://localhost:8010/example2.html

注意:
一个组件绝对不可以自己修改自己的props

数据流--props

针对props的验证方式(非强制):

```
…
propTypes:{
        …
}
```

可以为组件添加getDefaultProps函数来设置属性的默认值

```
…
getDefaultProps: function () {
        return {
                age:12
        }
}
```

数据流--state

State可以确定一个元素的视图状态

例：

http://localhost:8010/example3.html

例用方法：
getInitialState：function(){}
this.state
this.setState({})

只要state的值改变，render就会被调用。如果render函数的返回值有变化，虚拟DOM就会更新，真实的DOM也会被更新。

注意：千万不要使用this.state直接修改state的值，使用this.setState()方法修改。

事件处理

1.绑定事件处理器

React 标准化了事件对象，因此在不同的浏览器中都会有相同的属性。

支持的事件

焦点事件，表单事件，鼠标事件，触摸事件，UI事件，键盘事件，剪切板事件

例：

onChange， onKeyDown，onKeyPress，onKeyUp，onInput,onSubmit, onFocus,….

文档：http://www.reactjs.cn/react/docs/events.html

事件处理

事件和状态

```
var InputReact = React.createClass({
  getInitialState: function () {
    return {
      value:'这是一个例子'
    }
  },
  changeVal: function (e) {
  //  this.setState({value:e.currentTarget.value})
  },
  render: function () {
    var value = this.state.value;
    return (
      <div>
        <input value={value} onChange = {this.changeVal} />
      </div>
    );
  }
```

组件的复合

React 最激动人心的特性之一：可组合性（composability）

在React中，构建页面的基本单元是React组件。

优点：

可重用，易于维护，思路清晰

注意:

组件命名 首写字母大写，以与普通标签区分

组件的复合

示例：

```
var GlobalReact = React.createClass({
  render: function () {
    return (
      <div>
        <Avater />
        <UserName />
      </div>
    );
  }
});
```

```
var Avater = React.createClass({

  render: function () {
    return (
      <img src="https://ss3.baidu.com
    );
  }
});
```

```
var UserName = React.createClass({

  render: function () {
    return (
      <div>鹿晗</div>
    );
  }
});
```

组件的复合—从下向上传递数据

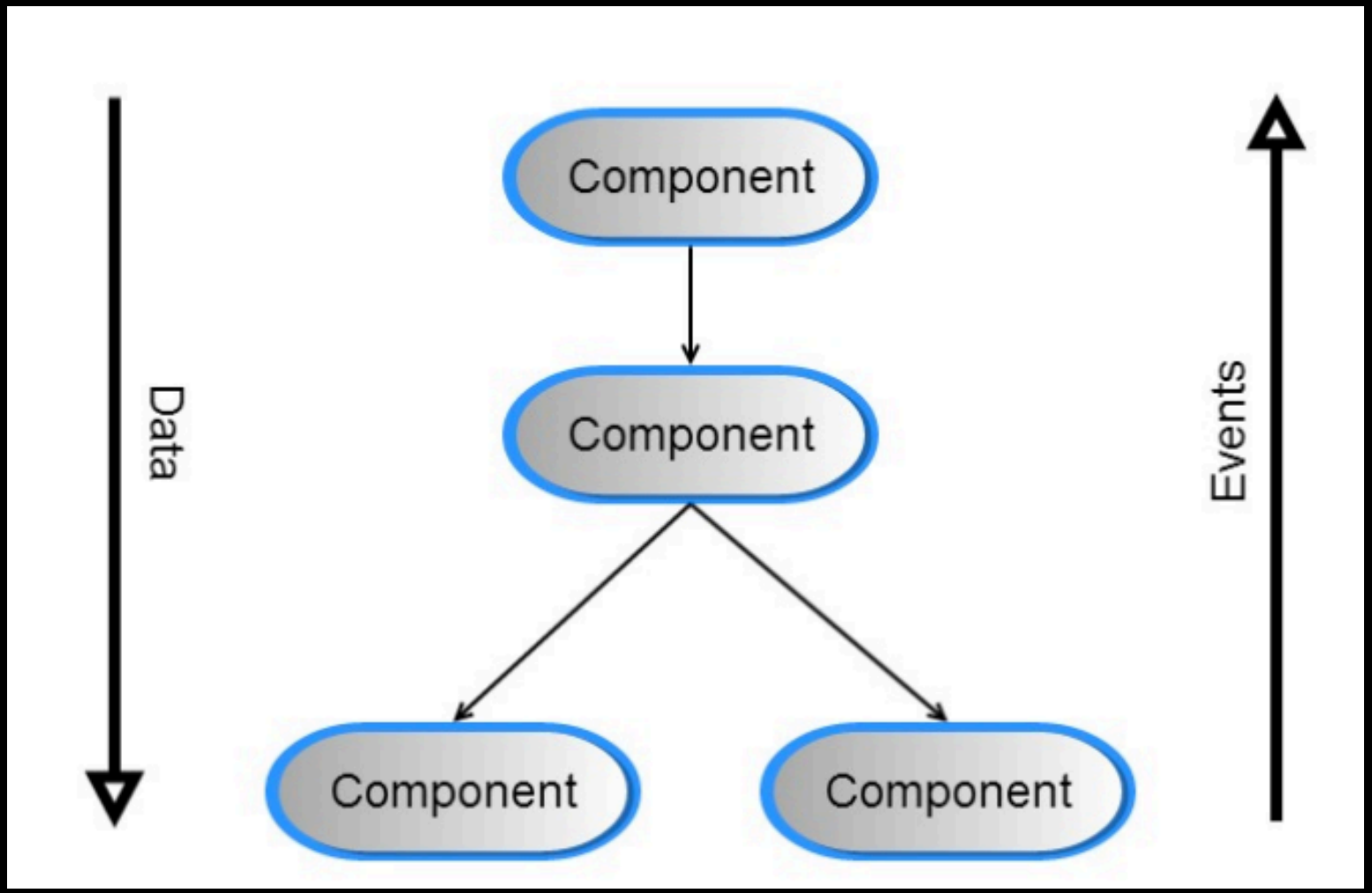
子组件向父级传递状态

```
var SubReact = React.createClass({

  clickMe: function () {
    this.props.ClickSubReact();
  },
  render: function () {
    return (
      <button onClick = {this.clickMe}>btn</button>
    );
  }

});
```

```
var ParentReact = React.createClass({
  clickSub: function () {
    console.log('click sub React');
  },
  render: function (){
    return (
      <div>
        <SubReact ClickSubReact={this.clickSub} />
      </div>
    );
  }
})
```

组件的复合—从下向上传递数据

数据从父到子地传递, 事件触发后从子到父地传递



DOM操作

访问受控的DOM节点—通过ref属性实现。

```
render: function () {
  return (
    <div>
      <input valu="test" name="uname" ref='unamdom' />
    </div>
  );
}
```

整合非React类库

组件的生命周期

实例化：

一个实例初次被创建时,你会看到下面这些方法会依次被调用：

getDefaultProps
getInitialState
componentWillMont (会在 component 渲染完成且已经有了 DOM 结构的时候被调用)
render
componentDidMount

对于该组件类的所有后续应用，你将会看到下面的方法依次被调用。

getInitialState
componentWillMont
render
componentDidMount

组件的生命周期

存在期：

随着应用状态的改变，以及组件逐渐受到影响，你将会看到下面的方法依次被调用：

compoentWillReceiveProps
shouldComponentUpdate
componentWillMont
render
componentDidMount

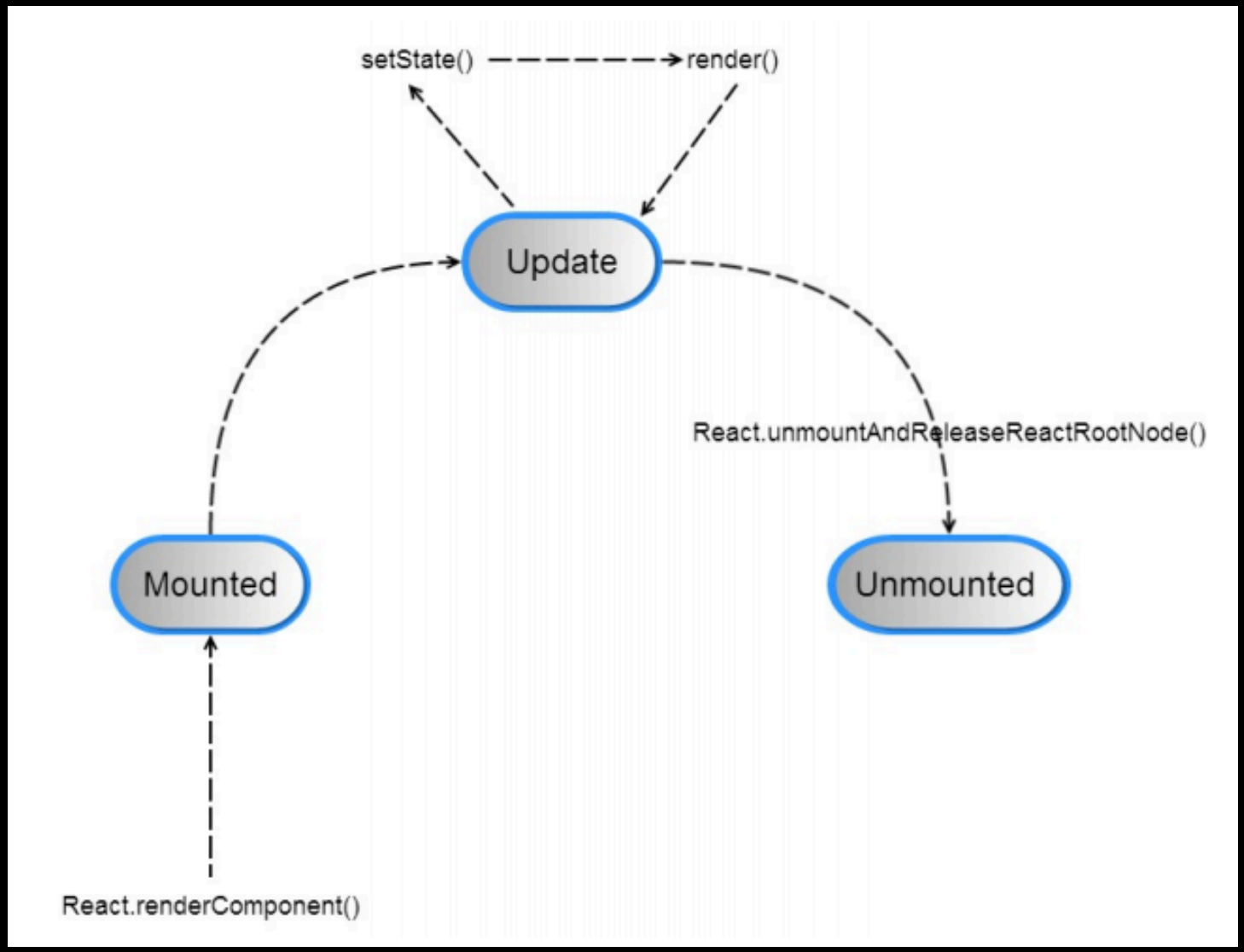
组件的生命周期

销毁&清理期：

当组件被使用完成后，将调用下面的方法，目的是给这个实例提供清理自身的机会：

componentWillMont

组件的生命周期


setState()
render()
Update
React.unmountAndReleaseReactRootNode()
Mounted
Unmounted
React.renderComponent()

其它知识点

- Mixin (可以定义多个组件中共用的方法 http://localhost:8010/example8.html)
- 表单(select,textarea)
- 动画
- 服务器渲染
- 开发工具(Browserify,watchify,webpack)
- 。。。

数据持久化以及事件传递

ReFlux.js

END