React所代表的组件化思想

## 组件化思想：

组件化表明web界面只是组件的一个容器，诸多具有低耦合、高内聚性的组件构造起了完整的界面。组件化思想并非空穴来风，历经了YUI、EXTJS和如今的Web Component。组件化开发代表了前端开发中的模块划分思想，也是面向对象思想的封装特性的体现。每一个组件都是独立的一个类，它具有类本身的私有变量和属性（state），对外提供接口函数和调用方法（props），web应用通过调用组件（新建类的实例）来实现组件的复用，而组件调用者只需要关注组件的输入和输出。和面向对象思想中的“万物皆对象”同出一辙，整个web应用就是包含了多个层级的子组件的组件。

## 组件化的优势：

①标记鲜明，容易维护：

只需要对组件本身进行维护，不会影响到其他组件的正常使用，具有高内聚和低耦合的优点。

②调试方便，便于协同开发：

由于整个系统是通过组件组合起来的，在出现问题的时候，可以用排除法快速定位问题组件（组件化的系统界面逻辑更简单）

③可以提高组件的复用性，并且能够很好的保持在系统开发汇总样式和风格的一直性（比如最基础的输入框组件、文本框组件等）

④避免了CSS样式全局名称污染的问题，一个组件内的样式以对象的形式存在于其文件作用范围内，直接以style方式绑定到了组件对应的dom节点。

## 组件化的不足：

①组件直接相互独立，包括样式互不影响，导致了通用样式无处安放（组件本身集成了一个DOM节点所需要的html、css、js），由此，还引发了文本、样式和行为相融合的与传统开发中文本、样式和行为分离背道而驰的问题。

## 组件化开发注意事项：

虽然在组件化的设计模式中，页面中都是由组件所构成的，但是组件也应该进行区分：对于无需以来其他子组件的基础组件，应该封装成标签文件以供调用；对于有依赖其他子组件或者由其他子组件构成的组件，在考虑是否封装的时候应慎重，因为一旦选择封装的方式，那么其props则需要由上一级调用它的父组件给予，以此类推，最顶端的父组件所携带的数据量容易存在过大的问题。

## 组件化思想和MVC设计模式

组件化的开发思维和MVC设计模式并不具备太高的可比性，MVC是完整的以分离表现、数据、控制的框架，着重于从技术的角度对UI进行分离，实现松耦合；而组件化开发则是开发者从功能的角度出发，将UI分成不同的组件，每个组件都独立封装，实现独立的功能。React仅仅负责表现层的组件设计和划分，需要配合Redux等Flux框架才能够比较好的实现web架构的设计。React并不是典型的MVC模式。