# 第一章 React简介

React本质上是一个“状态机”，可以帮助开发者管理复杂的随时间而变化的状态。它以一个精简的模型实现了这一点。React只关心两件事：

1、更新DOM

2、响应事件

React不处理Ajax、路由和数据存储，也不规定数据组织的方式。它不是一个Model-View-Controller框架。如果非要问它是什么，它就是MVC里的“V”。react的精简允许你将它集成到各种各样的系统中。事实上，它已经在数个MVC框架中被用来渲染视图了。

在每次状态改变时，使用JavaScript重新渲染整个页面会异常慢，这应该归咎于读取和更新DOM的性能问题。React运用一个虚拟的DOM实现一个非常强大的渲染系统，在React中对DOM只更新不读取。

React就像高性能的3D游戏引擎，以渲染函数为基础。这些函数读入当前的状态，将其转换为目标页面上的一个虚拟表现。只要React被告知状态有变化，它就会重新运行这些函数，计算出页面的一个新的虚拟表现，接着自动地把结果转换成必要的DOM更新来反映新的表现。

乍一看，这种方式应该比通常的JavaScript方案------按需更新每一个元素-----要慢，但React确实是这样做的：它使用了非常高效的算法，计算出虚拟页面当前版本和新版本间的差异，基于这些差异对DOM进行必要的最少更新。

React赢就赢在最小化了重绘，并避免了不必要的DOM操作。这两两点都是公认的性能瓶颈。

用户界面越复杂，就越容易发生这样的情况---------一个用户交互触发一个更新，而这个更新触发另外一个更新，一个接一个。如果没有恰当地把这些更新放到一起的话，性能就会大幅降低。更糟糕的是，有时候DOM元素在达到最终状态前，会被更新好多次。

React的虚拟表示差异算法，不但能够把这些问题的影响降到最低，还能简化应用的维护成本。当用户输入或者有其他更新导致状态改变时，我们只要简单地通知React状态改变了，它就能自动化地处理剩下的事情。我们无须深入到详细的过程中。

React在整个应用中只使用单个事件处理器，并且会把所有的事件委托到这个处理器上。这一点也提升了React的性能，因为如果有很多时间处理器也会导致性能问题。