React：

1. 组件化（react的核心）
2. 性能非常好
3. 生命周期

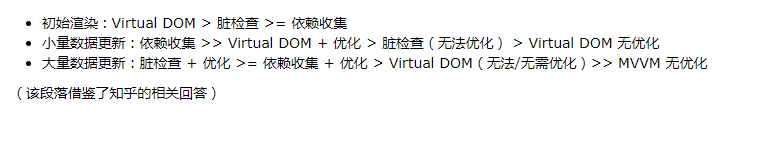




1. 可维护性：可以基于flux或redux的架构设计，确定性的store很容易定位问题，易于查找bug
2. Virtual DOM，运行效率高：解决了以往大量操作DOM带来的性能问题

在React中，render执行的结果得到的并不是真正的DOM节点，结果仅仅是轻量级的JavaScript对象，我们称之为virtual DOM

  虚拟DOM是React的一大亮点，具有batching(批处理)和高效的Diff算法。这让我们可以无需担心性能问题而”毫无顾忌”的随时“刷新”整个页面，由虚拟 DOM来确保只对界面上真正变化的部分进行实际的DOM操作。在实际开发中基本无需关心虚拟DOM是如何运作的，但是理解其运行机制不仅有助于更好的理解React组件的生命周期，而且对于进一步优化 React程序也会有很大帮助。



Redux：

1. 概念：Redux 是 JavaScript 状态容器，提供可预测化的状态管理。可以让你构建一致化的应用，运行于不同的环境（客户端、服务器、原生应用），并且易于测试。Redux 除了和 React 一起用外，还支持其它界面库。它体小精悍（只有2kB）且没有任何依赖。
2. 图解：



3、设计思想：a、web应用是一个状态机，视图和状态一一对应

b、所有状态（state）保存在一个对象中（store）

1. 单一状态树：所有的状态state对象（数据）都以树的形式储存在唯一的store中

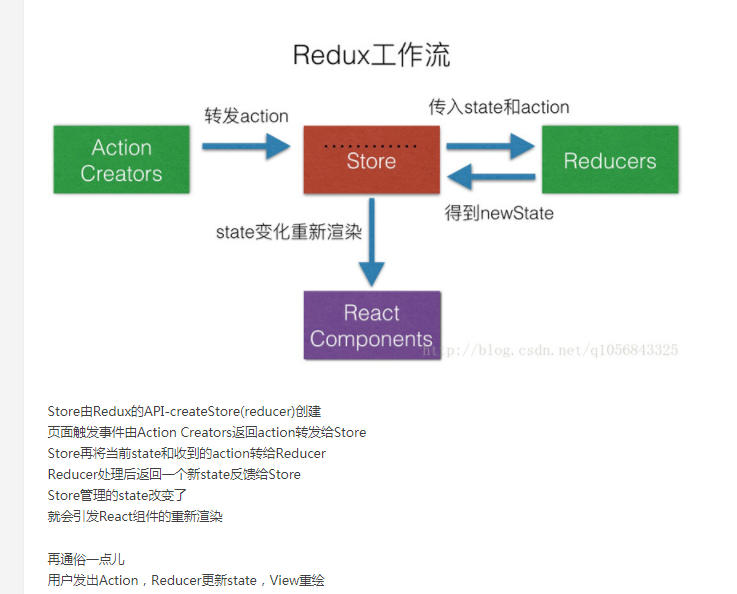
页面中的任何变动，首先都要去改变状态树（store），然后以某种形式成现在页面（一个视图对应一个状态对象）

1. 核心概念：Action、Reducer、Store

6、API



1. 工作流程



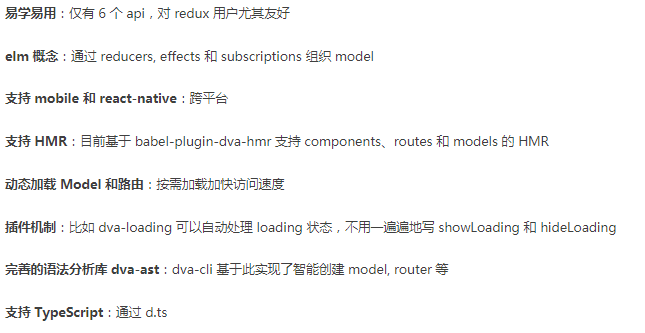
1. react-redux流程：

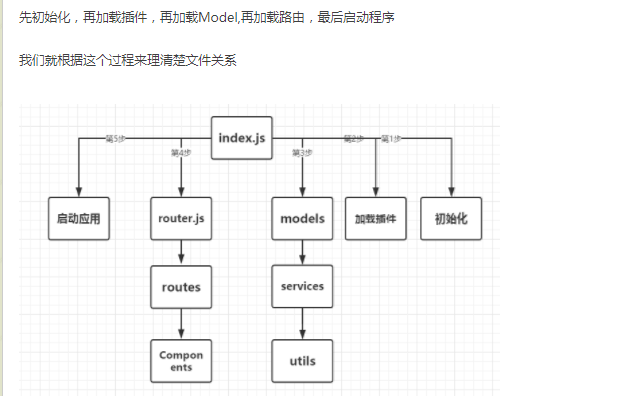


React-dva：

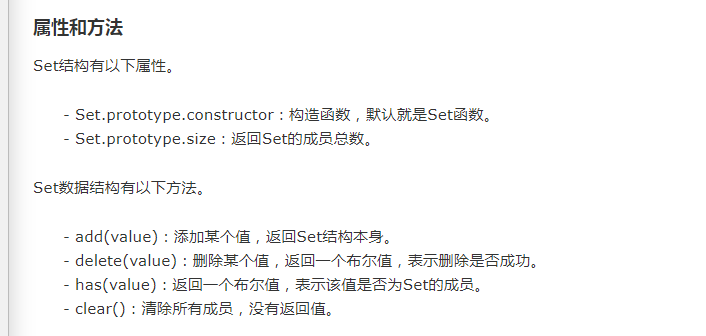
1. 介绍：[dva](https://github.com/dvajs/dva) 是一个基于 react 和 redux 的轻量应用框架，概念来自 elm，支持 side effects、热替换、动态加载、react-native、SSR 等，已在生产环境广泛应用。
2. dva的亮点： 引进了model这个概念，将以前一堆松散的概念和文件组织到一起了，这才是关键

React+redux+redux-saga中，每新增1个组件或者说新增1个页面，都要新增3个文件，sagas，reducers,action各自对应1个文件，编写的时候不停的切换，极其的麻烦，而且文件多了之后，文件的管理不是很方便

1. 为啥用dva：
2. 运行流程：



ES6

1. let：所声明的变量，只在let命令所在的代码块内有效。避免了变量提升。为JavaScript新增了块级作用域。
2. Const：也用来声明变量，但是声明的是常量。一旦声明，常量的值就不能改变。
3. Set：它类似于数组，但是成员的值都是唯一的，没有重复的值。

Array.from方法可以将Set结构转为数组。

Set结构的默认遍历是其values方法，可直接用for...of循环遍历Set。

Set结构也有keys和entries方法，与Map结构保持一致

Set可以使用扩展运算符

1. Map

……

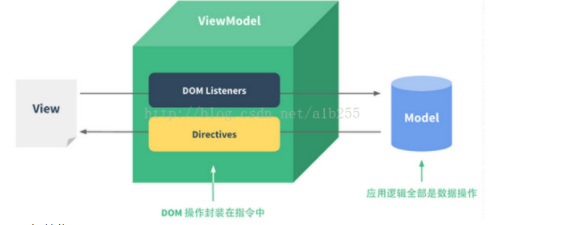
Flux



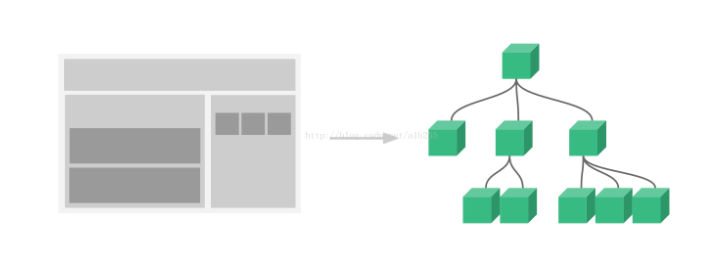
Vue核心思想

1. 双向绑定（双向数据流）

Vue是一种MVVM框架。而DOM是数据的一个种自然映射。传统的模式是通过Ajax请求从model请求数据，然后手动的触发DOM传入数据修改页面。Vue中，Directives对view进行了封装，当model里的数据发生变化是，Vue就会通过Directives指令去修改DOM。同时也通过DOM Listener实现对视图view的监听，当DOM改变时，就会被监听到，实现model的改变，实现数据的双向绑定

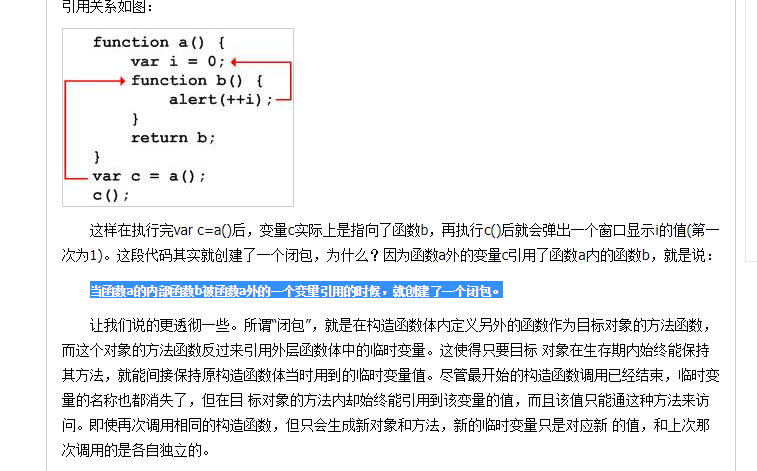


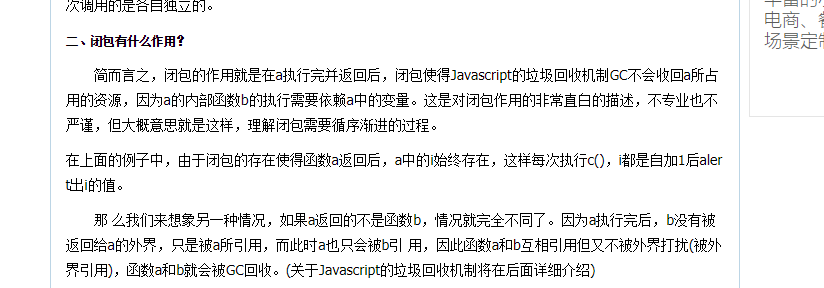
1. 组件化

组件化实现了扩展HTML元素，封装可用的代码。页面上每个独立的可视/可交互区域视为一个组件；每个组件对应一个工程目录，组件所需要的各种资源在这个目录下就近维护；页面不过是组件的容器，组件可以嵌套自由组合形成完整的页面。

JS

1. 闭包





1. 原型链与原型对象

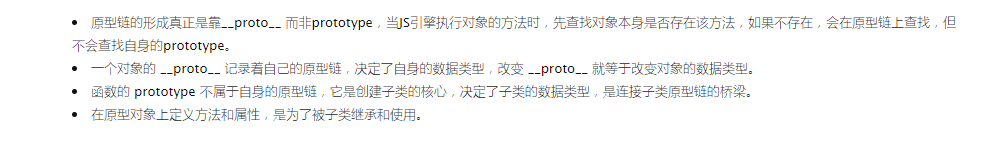
**函数对象**：**通过 new Function 产生的对象是函数对象，其他对象都是普通对象。**

**原型对象**：函数对象在创建时的属性，其中包括 prototype 和 \_\_proto\_\_， prototype 即原型对象，它记录着函数对象的一些属性和方法。

**原型对象的作用**：主要作用就是**继承**。 通俗一点讲，prototype 中定义的属性和方法都是留给自己的 “**后代**” 用的，因此，子类完全可以访问prototype中的属性和方法。

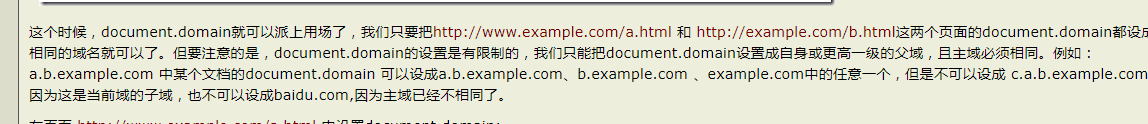
**原型链**：\_\_proto\_\_（引用父类的 prototype 对象），JS在通过 new 操作符创建一个对象的时候，通常会把父类的 prototype 赋值给新对象的\_\_proto\_\_属性，这样就形成了一代代传承...





1. 如何实现跨越：  
    一、通过jsonp跨越

二、**通过修改document.domain来跨子域**



三、使用window.name来进行跨域

四、使用HTML5中新引进的window.postMessage方法来跨域传送数据

4、判断浏览器类型（做浏览器兼容）：

navigator.userAgent; //取得浏览器的userAgent字符串

从中获取

CSS：

1. relative和absolute分别是相对于谁进行定位的：

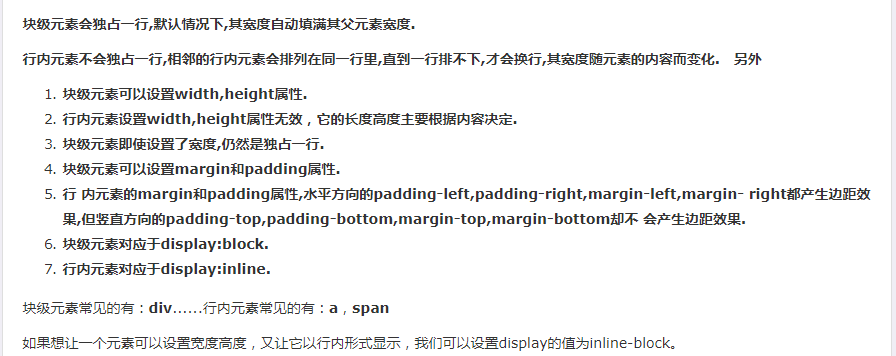


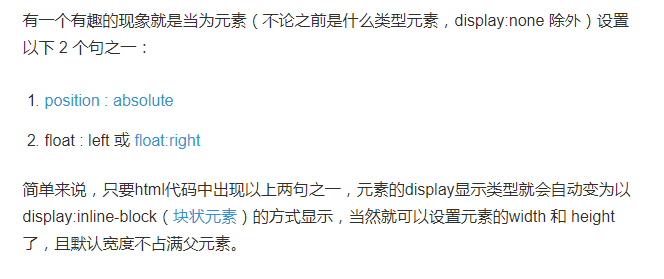
1. 盒子模型：

内容(content)、填充(padding)、边框(border)、边界(margin)， CSS盒子模式都具备这些属性。

这些属性我们可以用日常生活中的常见事物——盒子作一个比喻来理解，所以叫它盒子模式。

1. 块状元素与行内元素





1. css动画有哪些以及区别

transition关注的是CSS property的变化，property值和时间的关系是一个三次贝塞尔曲线。  
animation作用于元素本身而不是样式属性，可以使用关键帧的概念，应该说可以实现更自由的动画效果。  
至于实现动画效果用哪一种，我的感觉是要看应用场景，但很多情况下transition更简单实用些。

1. 浏览器是如何解析css和js文件的？为什么需要阻塞页面后续内容？

浏览器是并行加载css文件，串行加载js文件。如果你的javascript想操作后面的DOM元素，基本上来说，浏览器都会报错说对象找不到。因为Javascript执行时，后面的HTML被阻塞住了，DOM树时还没有后面的DOM结点。所以程序也就报错了。

1. css3弹性盒子

弹性盒子是 CSS3 的一种新的布局模式。

CSS3 弹性盒（ Flexible Box 或 flexbox），是一种当页面需要适应不同的屏幕大小以及设备类型时确保元素拥有恰当的行为的布局方式。

引入弹性盒布局模型的目的是提供一种更加有效的方式来对一个容器中的子元素进行排列、对齐和分配空白空间。