前 端 开 发

大 前 端 浪 潮 下 的 新 前 端

SuenYinKit

大 前 端 浪 潮 下 的 新 前 端

前因

后果

框架原理

领域衍生

刀 耕 火 种 的 时 代

为 什 么 ?

开发工程化

前后端耦合

前 端 语 言

HTML

使用标记定义页面元素 (Dom)

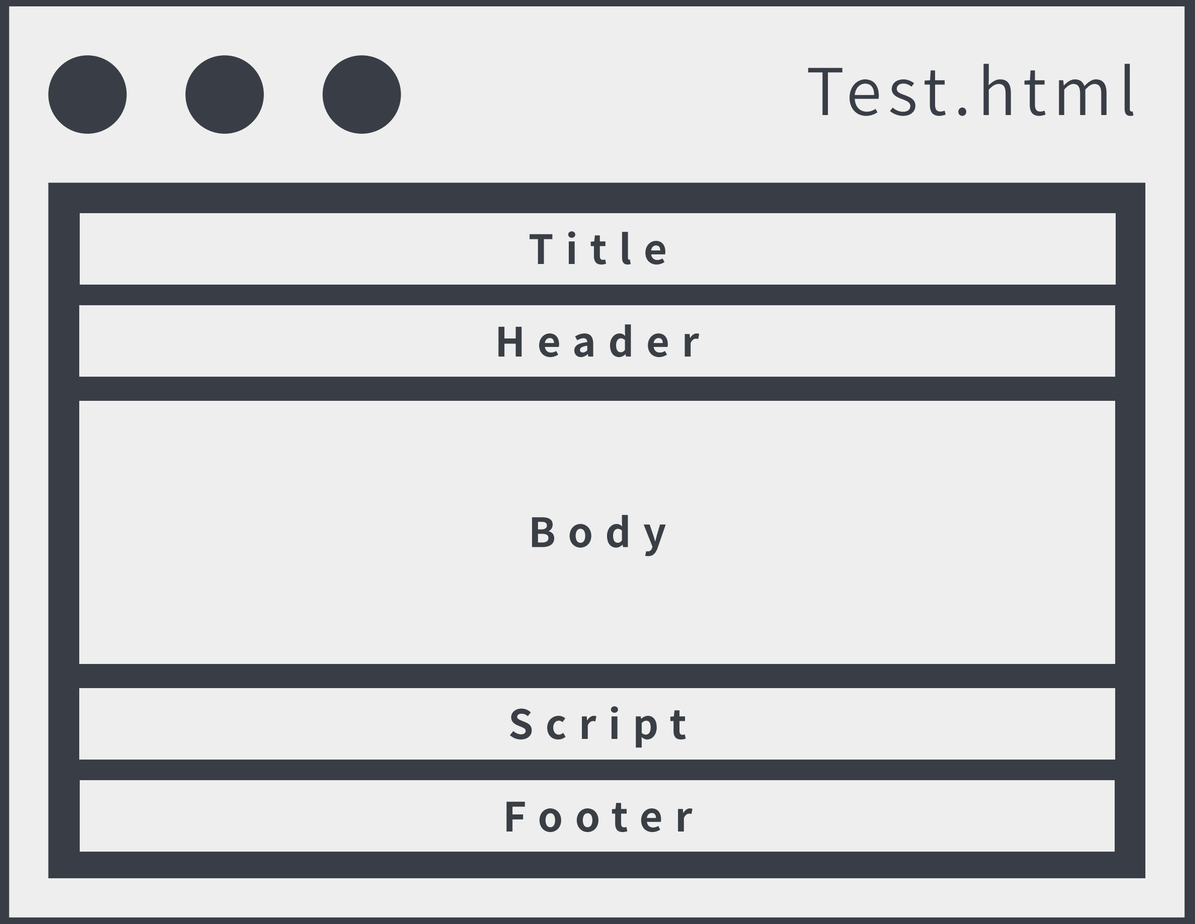
CSS

定义元素的样式 (Style)

JavaScript

对元素插入动作脚本 (Action)

前 端 如 何 工 作



Body

标记几乎所有页面元素

Header

标记页面信息与使用的样式

Script

标记页面所插入的脚本

前端工具库

46k+ 13k+

JQuery

B Bootstrap

E EasyUI

J JQueryUI

. . .

移动开发

PhoneGap

Cordova

Test.html

Title

Header（引入库 CSS）

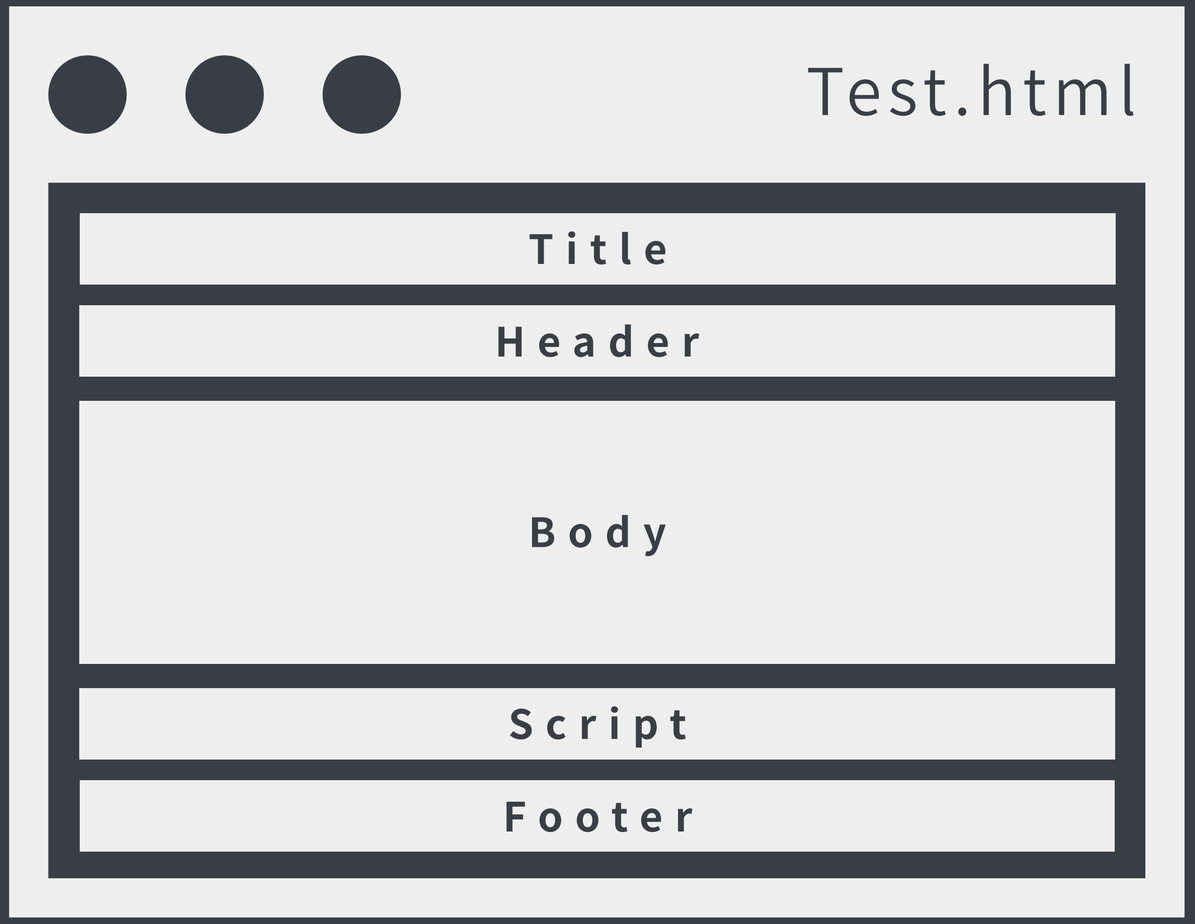
Body（引入库所需 Dom）

Script（引入库 JS）

Footer

有什么缺陷？

代码管理缺陷



臃肿不堪

- 代码复用
- 公共库管理
- 问题排查
- 页面性能

项 目 缺 陷

到处都是模板

前后端语言上的依赖

责任追查不便

改版 = 重写

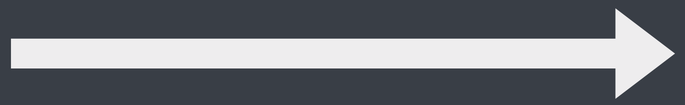
规 范 与 前 后 端 分 离

前 端 规 范

CommonJS

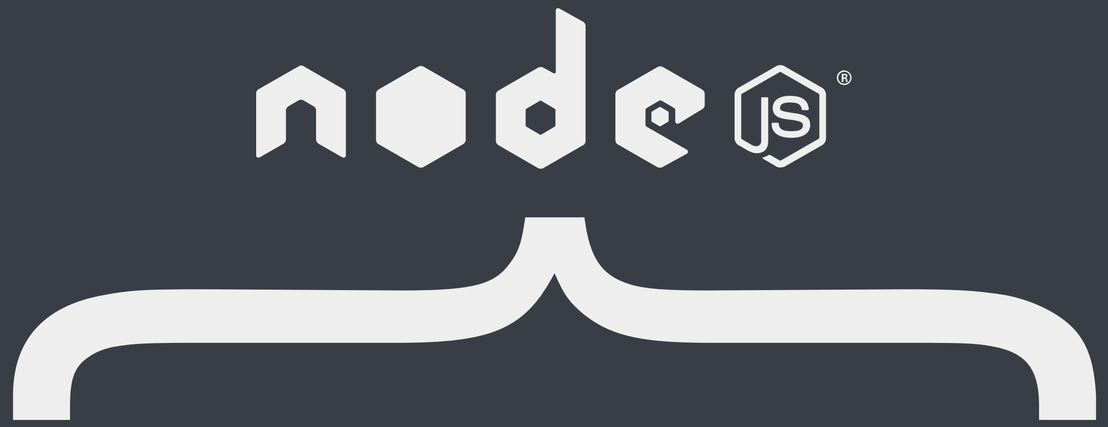
模块化标准

脱离浏览器运行



包管理工具

前 端 规 范



Gulp

Grunt

Express

koa

Webpack

工 程 化 后 的 前 端

Webpack

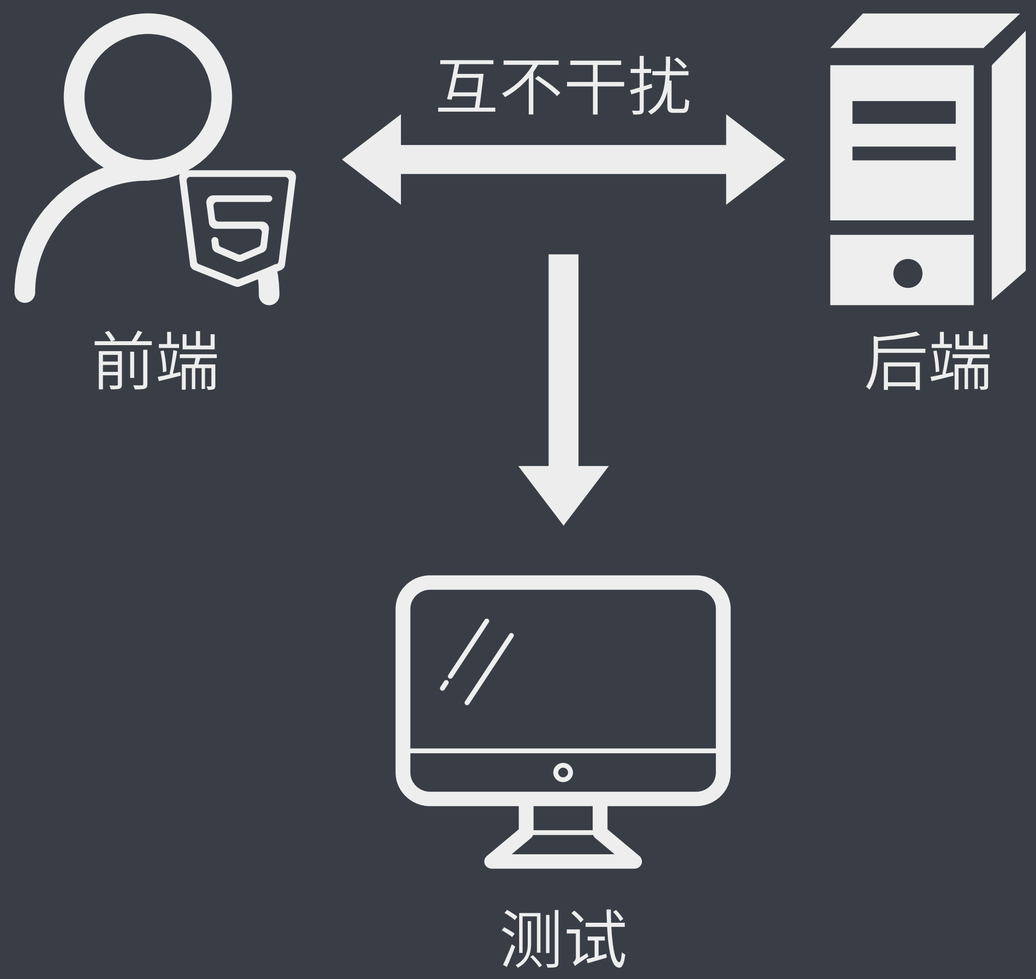
更好地管理依赖

更好地管理项目环境

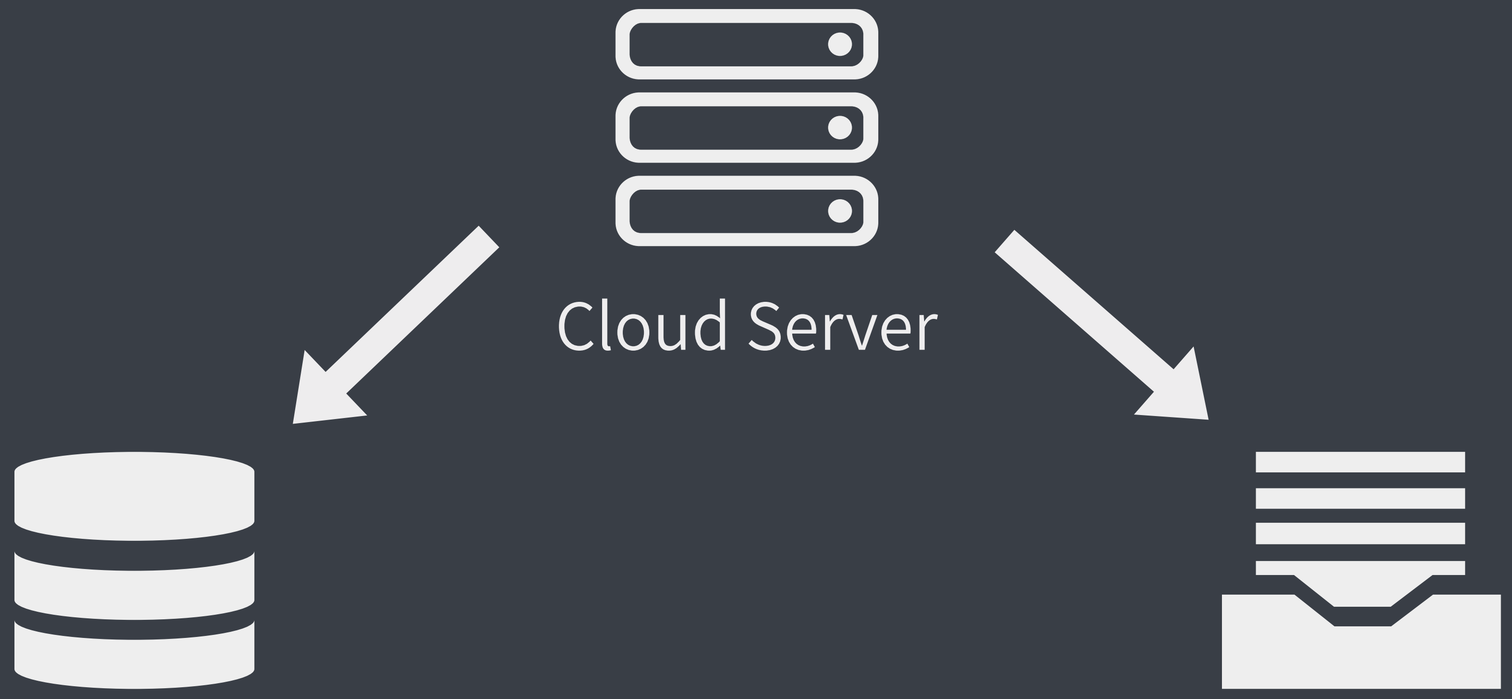
更高的开发速度 ……

前 后 端 分 离

前后端分离



前 后 端 分 离



BaaS (Backend as a server)

数据存储

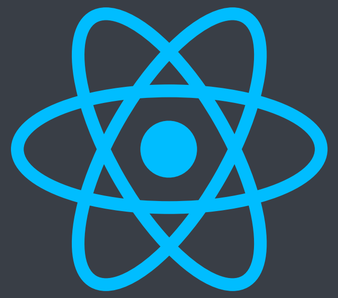
FaaS (Function as a server)

业务分离

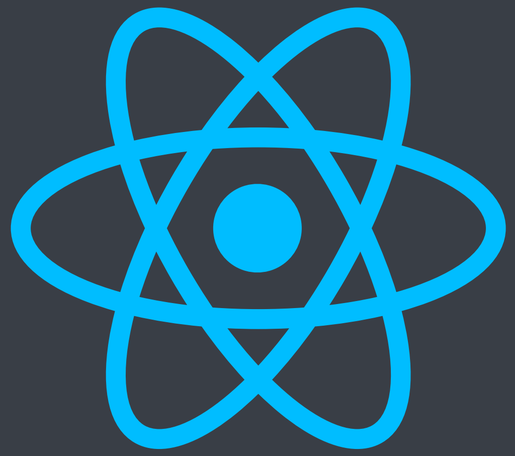
数 据 驱 动 型 框 架

数 据 驱 动 型 框 架

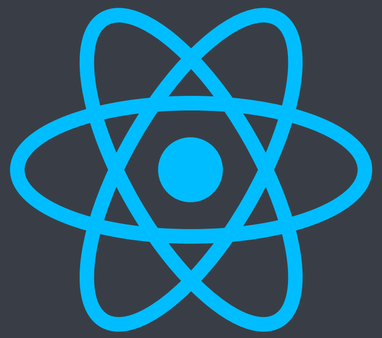
Regular.js



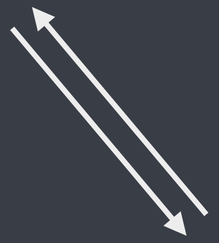
qreact.js



三 大 框 架



数据驱动



Model

ViewModel

优 点

更少的 Dom 操作

更灵活的交互实现

业务与 UI 分离

组件式视图分离

统一数据流控制

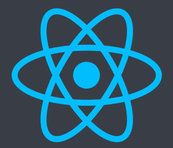
视图层抽象化

数据驱动原理

脏检查

数据模型的脏检查

view 抽象的脏检查

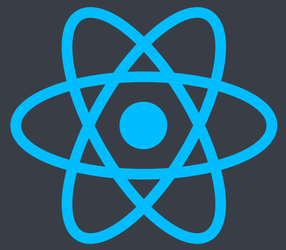


存取器

对象的计算属性

setter / getter 函数

数据模型的脏检查



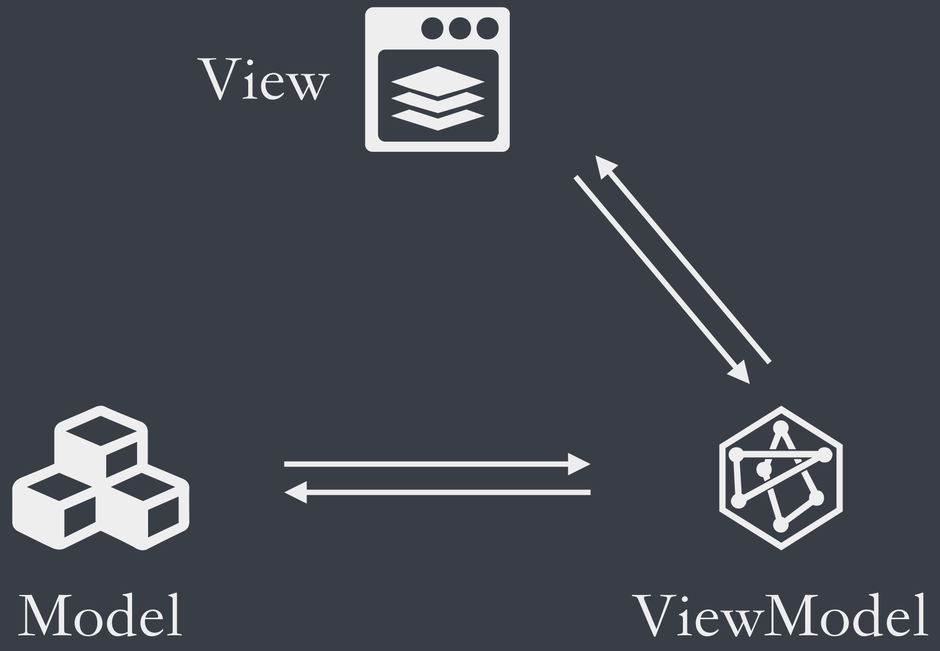
view 抽象的脏检查

对象的计算属性

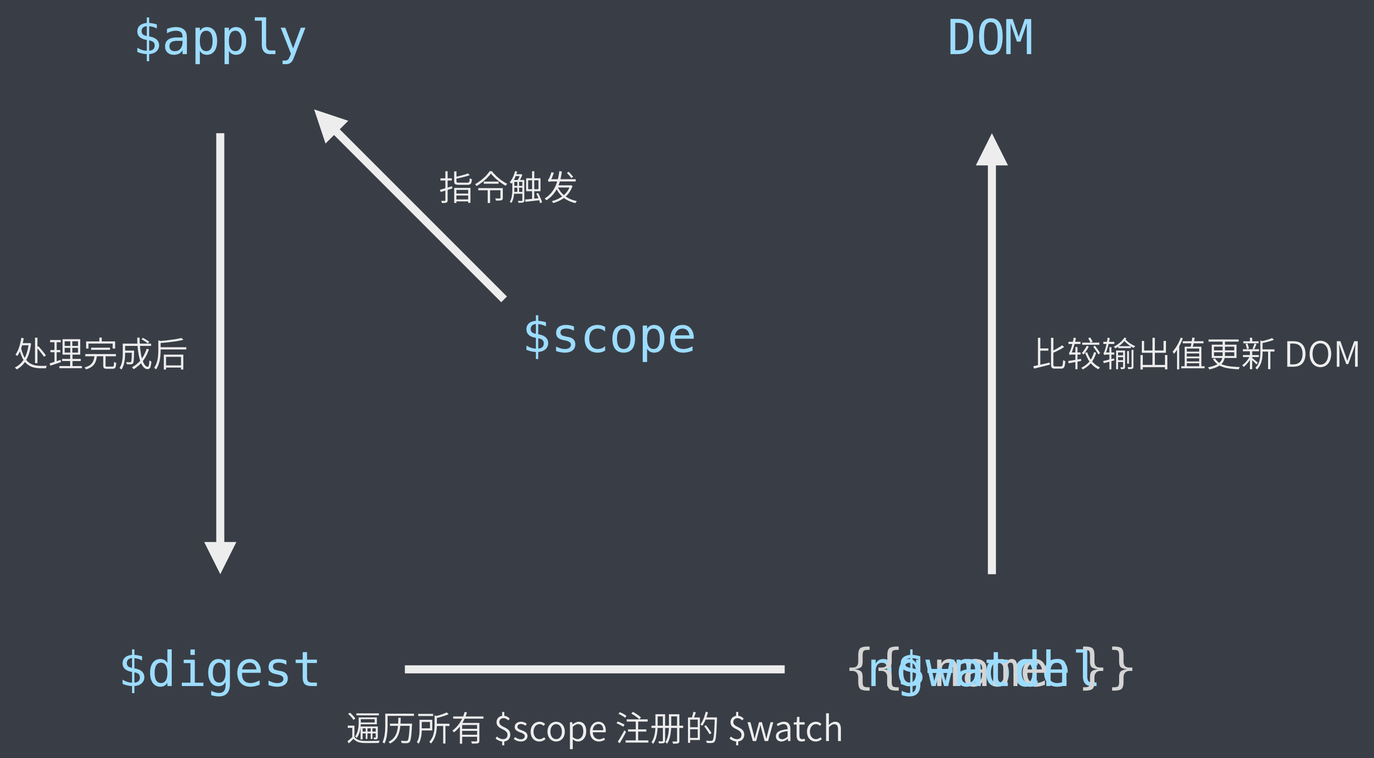
数 据 模 型 的 脏 检 查

```
<div ng-app="testApp" ng-controller="TestController">
    <p>名字 : <input type="text" ng-model="name"></p>
    <h1>Hello {{ name }} !!!</h1>
    <button ng-click="onDIVClick()">change to Flyme</button>
</div>
```

```
var testApp = angular.module('testApp', []);
testApp.controller('TestController', function ($scope) {
    $scope.name = 'MEIZU';
    $scope.onDIVClick = function () {
        $scope.name = 'Flyme';
    };
});
```



脏 检 查 实 现


$apply
指令触发
处理完成后
$scope
DOM
比较输出值更新 DOM
$digest
{{name}}
遍历所有 $scope 注册的 $watch


dirty=false
$digest
dirty=true
$watch 输出值发生变化

脏检查优点 & 缺点

只看结果 不关心变化过程

不局限于数据储存的格式
实现比存取器容易

容易触发无谓的脏检查

性能难以保证

Angular 2

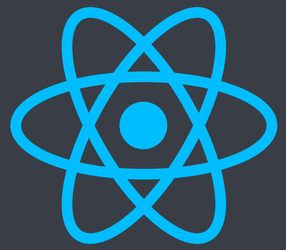
推荐阅读

Angular 脏检查机制研究

Angular 沉思录

Angular 脏检查深入分析

数据模型的脏检查



View 抽象的脏检查

对象的计算属性

View 抽象的脏检查

View 的抽象

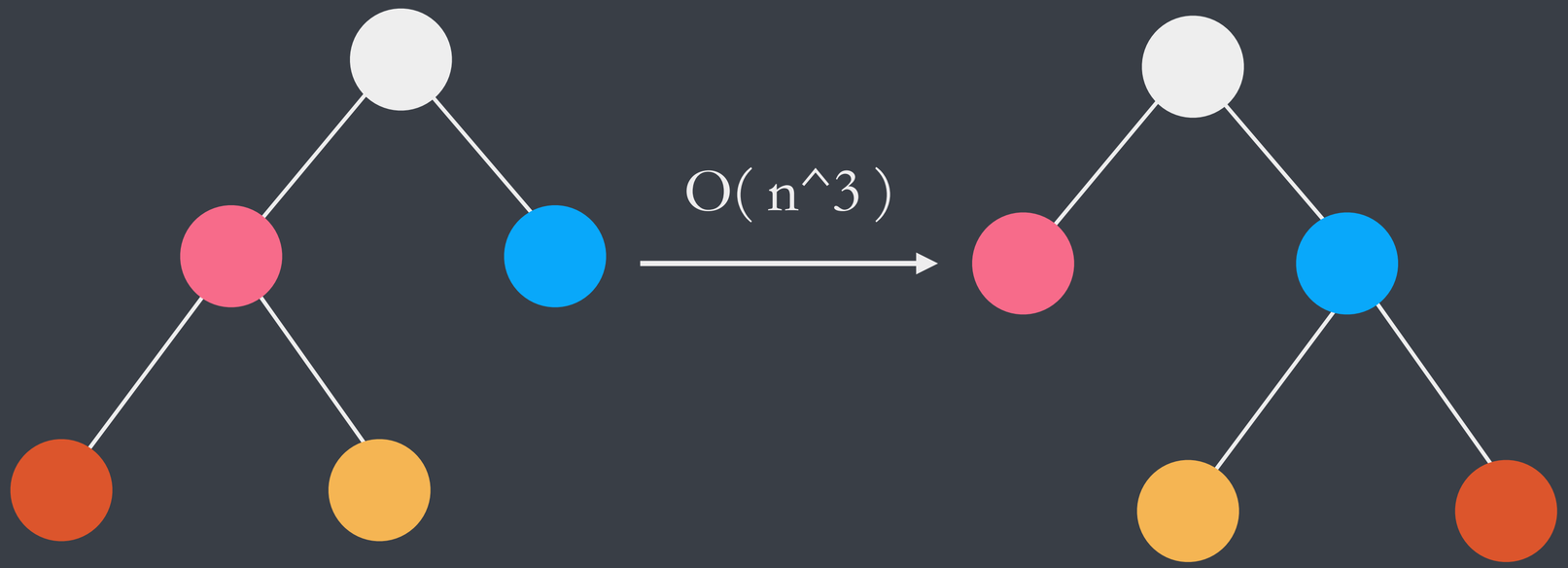
```
<ul id='list' style="width: 100%">
    <li class='item'>
        Item 1
    </li>
    <li class='item'>
        Item 2
    </li>
    <li class='item'>
        Item 3
    </li>
</ul>
```

```
let vDom = {
    tagName: 'ul',  // 节点标签名
    props: {  // DOM的属性，用一个对象存储键值对
        id: 'list',
        style: {
            width: '100%'
        }
    },
    children: [  // 该节点的子节点
        {
            tagName: 'li',
            props: { class: 'item' },
            children: ["Item 1"]
        }
        // omitted Item 2 and Item 3
    ]
}
```

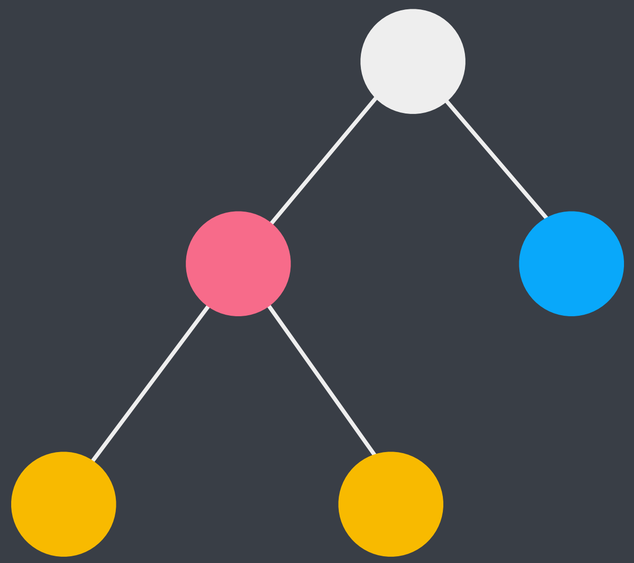
Dom 结构 ⟶ JS 对象（Virtual Dom）

Dom 更新

diff 算法



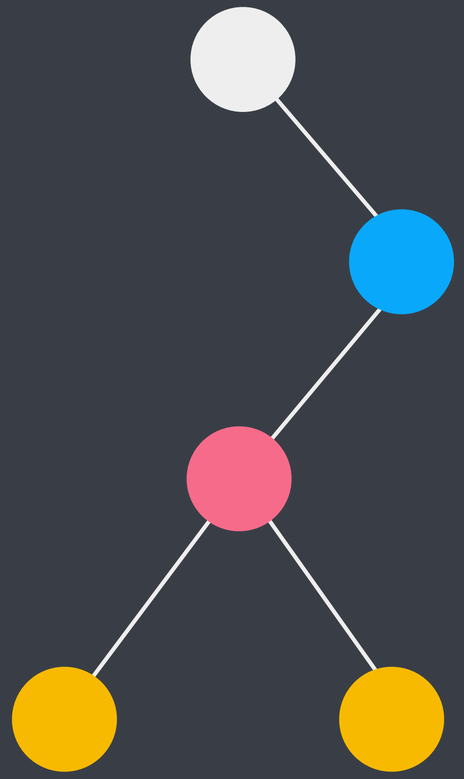
diff 算法



diff 策略

O(n^3) ⟶ O(n)

diff 算法

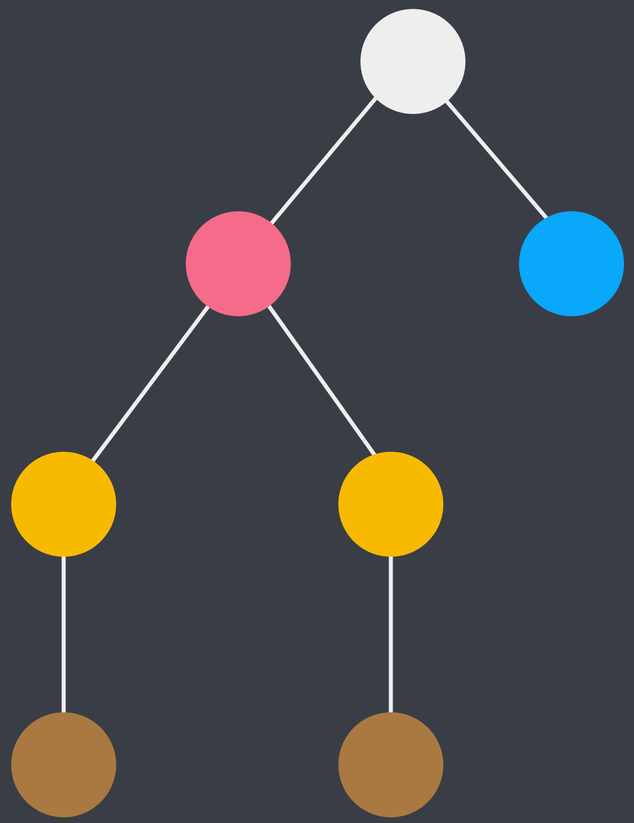


diff 策略

O(n^3) ⟶ O(n)

Web UI 中 DOM 节点跨层级的移动操作特别少，可以忽略不计

diff 算法



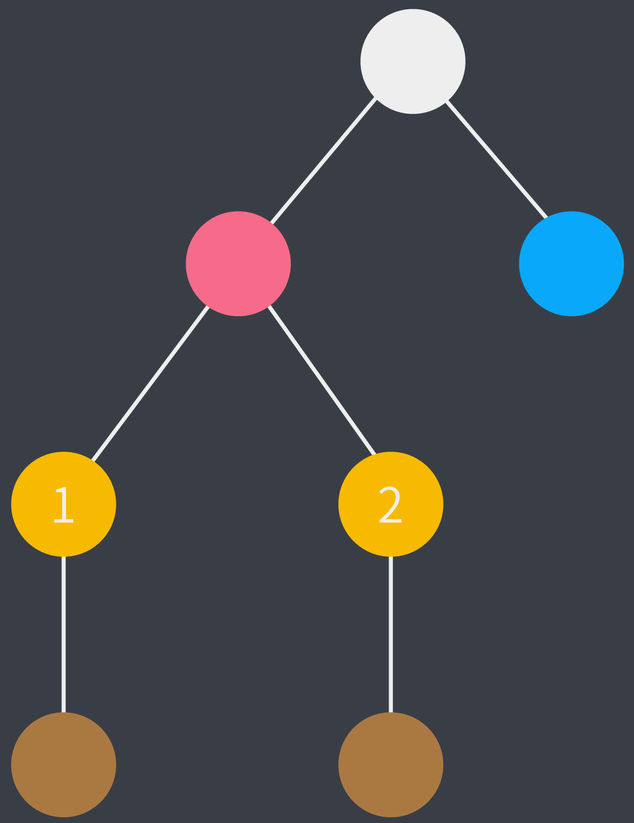
diff 策略

O(n^3) ⟶ O(n)

Web UI 中 DOM 节点跨层级的移动操作特别少，可以忽略不计

拥有相同类的两个组件将会生成相似的树形结构，拥有不同类的两个组件将会生成不同的树形结构

diff 算法



diff 策略

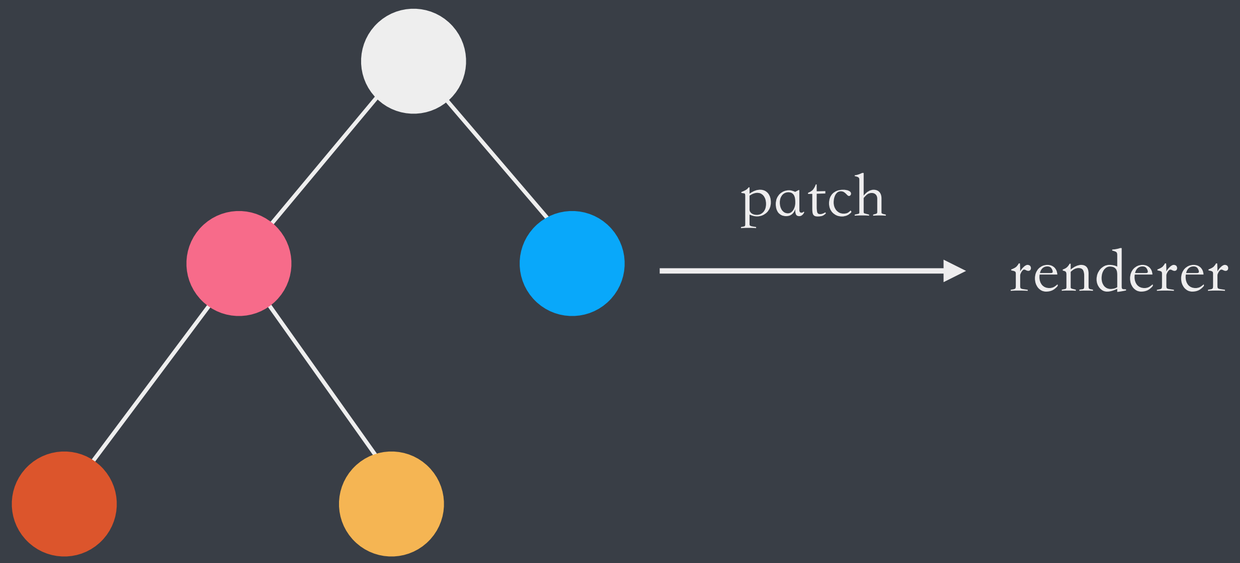
O(n^3) ⟶ O(n)

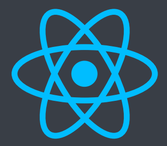
Web UI 中 DOM 节点跨层级的移动操作特别少，可以忽略不计

拥有相同类的两个组件将会生成相似的树形结构，拥有不同类的两个组件将会生成不同的树形结构

对于同一层级的一组子节点，它们可以通过唯一 id 进行区分

diff 算法





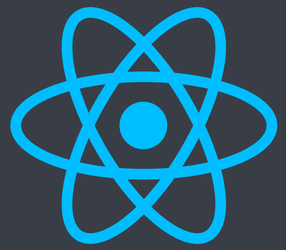
推荐阅读

不可思议的 react diff

虚拟 DOM Diff 算法解析

React 16 Fibet 源码速览

数据模型的脏检查



view 抽象的脏检查

对象的计算属性

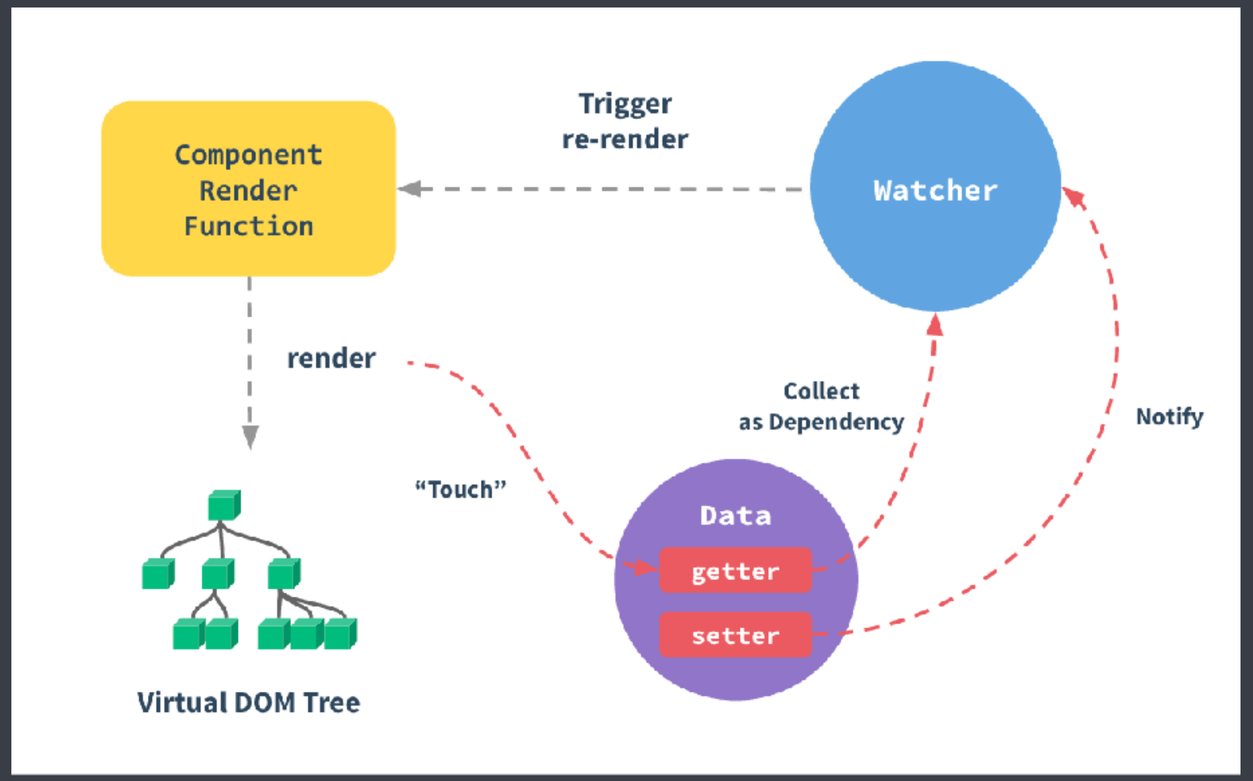
对象的计算属性

依赖收集

getter / setter

Depend

Watcher



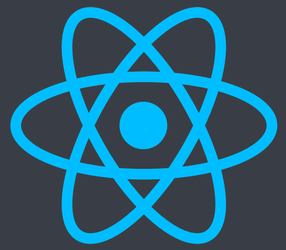
推荐阅读

Vue 官方 — 深入响应式原理

Vue 响应式原理探析

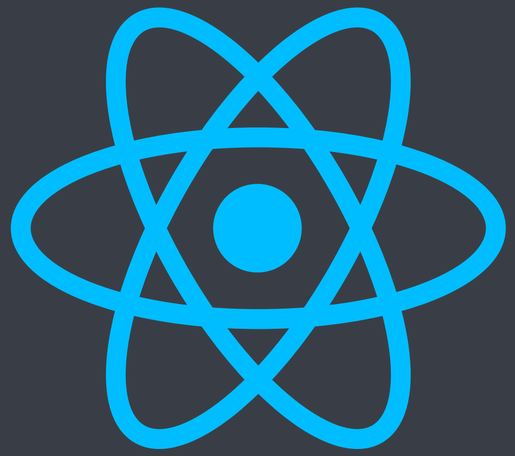
深入浅出基于“依赖收集”的响应式原理

数据模型的脏检查



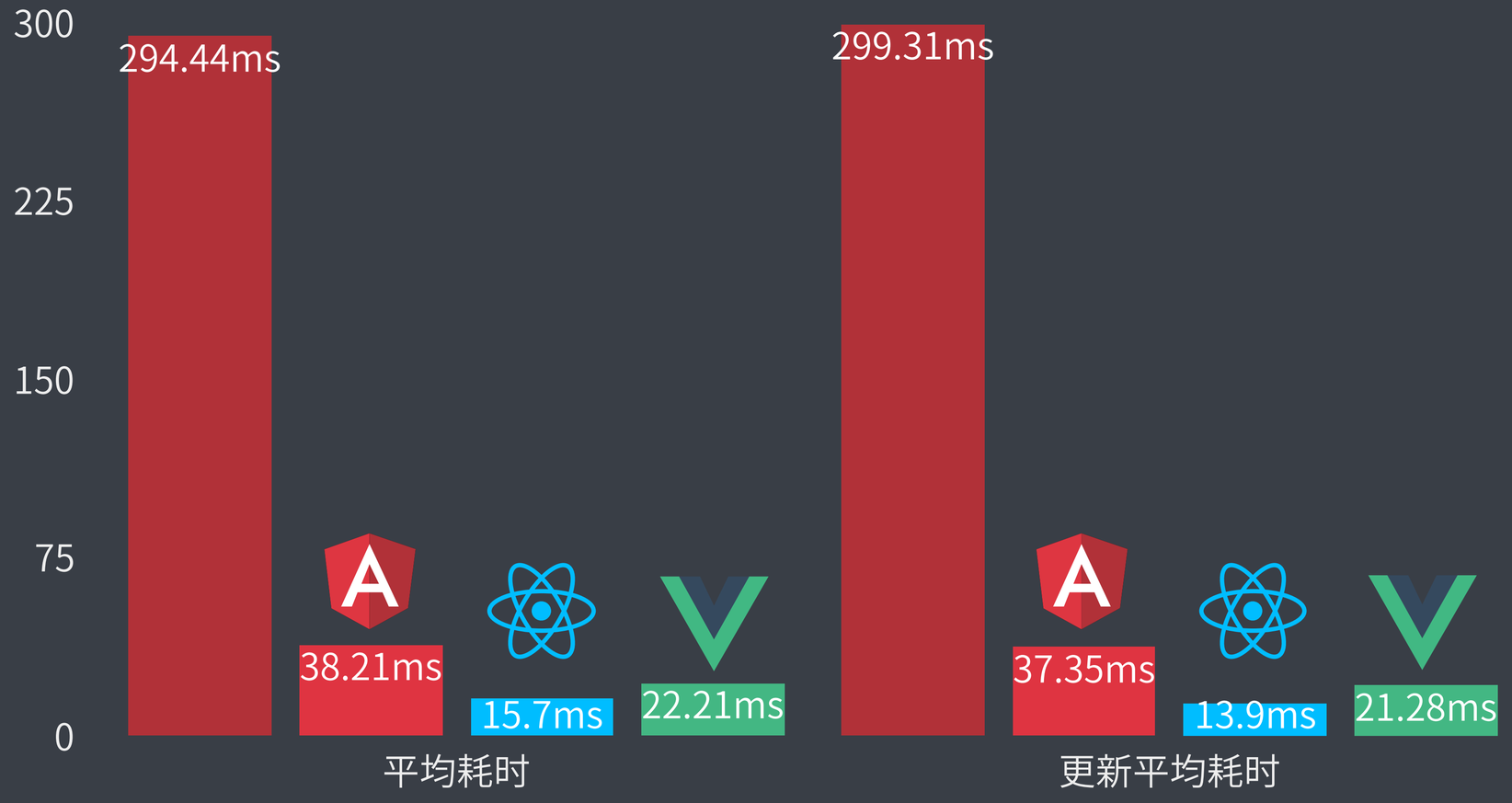
view 抽象的脏检查

对象的计算属性



性 能 差 异

连续渲染 60 次 1000 行随机数据表格



领 域 衍 生

领 域 衍 生

View

V-Dom

Web Android iOS

WEEX

React Native

感　谢　诸　位

耐　心　聆　听