虚拟DOM树:

什么是: 仅包含可能变化的节点和可能变化的属性的树结构

<body>

<div id=”app”>  
 <img src=”logo.png”> alt title id ….

<h1>{{uname}}</h1>id class title name

<hr>  
 <h2>{{score}}</h2>

</div>

{

el:”#app”,

children:[

{el:”h1”, innerText:uname},

{el:”h2”,innerText:score}

]

}

为什么: 内容少，便于快速遍历比较不同

如何发挥作用:

当data中模型变量改变时，会通知虚拟DOM树

虚拟DOM树先缓存本次的修改在元素对象上

将一批修改生成新的DOM子树和原虚拟DOM树做对比。

一旦发现不同的元素或内容，就只更新有修改的元素

虚拟DOM树中，封装了传统DOM API: createElement() appendChild() .innerHTML，避免了程序员编写大量重复的代码。

懒加载

Vue首屏加载非常慢

原因: 当打包应用时，将所有JavaScript代码打包在一个文件中，导致js代码非常庞大，严重影响页面加载速度。

解决:

1. 配置打包工具，将组件分别打包到不同的js代码块中

build/webpack.base.conf.js

output:{

path: config.build.assetsRoot,

filename:’[name].js’,

//新增

chunkFilename:”[name].js”,

publicPath: process.env.NODE\_ENV===”production”

?config.build.assetsPublicPath

:config.dev.assetsPublicPath

}

1. 当路由请求到该组件时，才动态加载组件的内容

路由字典中，路由配置和以前完全一样

但是在引入组件对象时:

~~import Index from ‘@/views/Index.vue’~~

改为  
 const Index=()=>import(‘@/views/Index.vue’)//仅定义函数

//暂未执行

//暂时不引入Index.vue

当用户在vue中请求当前组件对应的路由地址时，由vue-router自动调用加载函数，动态请求Index.vue组件对象