**Get全栈技能点**

1. **前端框架对比**
   1. Anaular提供的更多是一整套解决方案，React和vue 更像是一个生态；
   2. Vue与React对比：

不同点：React:1)更适用于大型应用和更好的可测试性,2)更大的生态圈带来的更多支持和工具；Vue:1)模板和渲染函数的弹性选择,2)简单的语法及项目创建,3)更快的渲染速度和更小的体积。

相同点：核心特点(数据驱动/双向绑定/状态管理)虚拟DOM实现快速渲染,轻量级,响应式组件,服务端渲染,易于集成路由工具、打包工具状态以及状态管理工具。

* 1. DOJO:很重,有丰富的组件和很多的核心库;

Spine和EmBer以及BackBone.js:都是典型的MVC框架,主要通过Controller做核心的控制,业务控制和事件触发都是通过Controller实现;

prototype和jQuery:主要针对函数库的,jQuery基于DOM操作的函数库;

Require.Js:模块化的开发框架,减轻了文件的加载,异步请求,结构更清晰;

React和Vue以及Antular:最近流行的MV\*框架.

1. **VUE语法**
   1. vue-cli构建SPA应用:

npm install –g vue-cli

vue init webpack-simple demo1 // 构建简单项目

vue init webpack demo2 // 构建常规(复杂)项目

* 1. class和style绑定：

class对象语法：

<div :class = "{'aclass': isActive, 'bclass': hasErroe}" > </div>

class数组语法：

<div :class = "[ classA, classB ]" > </div>

data: {

classA：'class-a'，

classB: 'class-b'

}

style绑定：

<div :style= "{color: activeColor, font-size: font1 + 'px'}" > </div>

data: {

activeColor：'#ccc'，

classB: '14'

}

* 1. vue事件处理器：

v-on:click.stop (或@click.stop) : 阻止冒泡;

@click.stop.prevent : 阻止默认事件;

@click.self : 只给其本身绑定事件,其子项不绑定;

@click.once : 只绑定一次事件.

@click.keyup事件：

@click.keyup.enter : 捕捉回车键,即回车键可触发;

@click.keyup.delete: 捕捉'删除'和'退格'键;

tab/esc/space/up/down/left/right

* 1. 前端路由的优缺点：

优点：

用户体验好，不需要每次从服务器全部获取,快速展现给用户

缺点：

* + - * 不利于SEO
      * 使用浏览器的前进/回退键时会从新发送请求，没有合理的利用缓存
      * 单页面无法记住之前滚动条滚动条的位置，无法前进/后腿的时候记住滚动条的位置
  1. route-link与route-view

<route-link></route-link> 跳转,编程式路由,类似this.$router.push({path: ''}) 或者a标签

<route-view></route-view> 组件渲染

* 1. 嵌套路由

注意:子路由定义时,path前不要加'/'

// router.js

routers: [

{

path: '/goods',

name: 'goodsList',

component: GoodsList,

children: [

path: 'title', // 嵌套的子路由,path路径前面不要加 /

name: 'title',

component: Title

]

}

]

// GoodsList.vue

<template>

<span>这是父页面展示的内容</span>

<!-- 跳转指定的path地址需要是绝对路径地址,前面带上/ -->

<router-link to="/goods/title">跳转子路由</router-link>

<div>

<router-view></router-view>

</div>

</temepalte>

* 1. 编程式路由

this.$router.push( 'goods' )

this.$router.push( { path: 'goods' } )

this.$router.push( { path: 'goods', query: {'id: 123'} } )

this.$router.push( { path: 'goods', params: { 'id: 123'} } )

this.$router.go(-1)

1. ES6语法
2. AMD、CMD、CommonJs、ES6 对比

AMD: AMD是RequireJS在推广过程中对模块定义的规范化产出.

异步模块定义, 特点是：依赖前置(在需要的时候,在最前面定义好)。

// 定义需要依赖的模块 package/lib

define(['package/lib']), function () { // 通过回调函数获取依赖的文件

function foo () {

lib.log('hello world')

}

}

return { // 返回处理后的结果

foo: foo

}

CMD: CMD是SeaJS在推广过程中对模块化定义的规范化产出.(SeaJS是淘宝官方提供的)

同步模块定义, 特点: 依赖就近 (哪里需要, 哪里require, 即用即返回)

// 所有模块都通过define定义

define(function (require, exports, module) {

// 通过require引入依赖

var $ = require('jquire') ;

var Spinning = require('./spinning') ;

})

CommonJS: module.exports规范, 浏览器不支持, 一般是服务端使用的(nodeJs推荐使用)

通过require引入, 通过module.export输出

const path = require('path')

module.exports = (filePath) => {

let aaa = path.extname(filePath)

return aaa

}

ES6 => export/ import

import org from './components/org.vue'

export default {

name: 'aaa',

components: {

org: org

}

}

1. test

// 7-2

**1-介绍**

Node.js is a JavaScript runtime built on Chrome's V8

Node.js uses an event-driven, non-blocking I/O model

**12--global全局变量之Buffer**

(1)关于Buffer: Buffer用于处理二进制数据流;实例类似整数数组,大小固定；c++代码在V8堆外分配物理内存

(2) 创建Buffer的方法

// 创建一个长度为 10、且用 0 填充的 Buffer。

const buf1 = Buffer.alloc(10);

// 创建一个长度为 10、且用 0x1 填充的 Buffer。

const buf2 = Buffer.alloc(10, 1);

// 创建一个包含 [0x1, 0x2, 0x3] 的 Buffer。

const buf4 = Buffer.from([1, 2, 3]);

// 创建一个包含 UTF-8 字节 [0x74, 0xc3, 0xa9, 0x73, 0x74] 的 Buffer。

const buf5 = Buffer.from('test');

// 创建一个包含 Latin-1 字节 [0x74, 0xe9, 0x73, 0x74] 的 Buffer。

const buf6 = Buffer.from('test', 'latin1');

(3)常用静态的属性和方法 Buffer.byteLength Buffer.isBuffer() Buffer.concat