1. 面试类
   1. 技术框架演变
   2. 最难解决问题
   3. 长列表页优化性能
   4. 前端是什么
   5. 前端会之后发展
   6. 后端服务会不会消失
   7. 浏览器工作原理 url
   8. 缓存分类
   9. http缓存分类
   10. 前端性能优化
   11. js加载方式（js的循环机制）

js加载分为同步和异步

js是单线程程序，一般情况下代码会一步一步执行，如果遇到错误，write，alert则会阻塞运行。

异步情况其实是把一部分代码放入任务队列中，让它延后执行，执行完主线程，任务队列的任务添加 到主线程加载执行，往复循环，这也是js的循环机制

异步执行顺序： 微任务：promist、宏任务：setinterval、 settimeout

* 1. node的异步回调
  2. 项目中遇到的坑

1. 在vue中，v-model.trim写在文本域中，按回车不会换行，因为trim把换行当成了空格
2. 在使用element-ui form表单时，遇到多个form，form里input按回车会触发表单提交。从而使页面会刷新
3. element ui upload终止上传 return false
4. element ui 下拉框不随着页面滚动而移动问题popper-append-to-body = "false" 加这个属性，加了下拉框就不会随页面滚动而滚动了。要改其样式就不能加scoped了
5. IE兼容问题：

1. new Date(“2019-02-22”)转换失败，需要换成斜杠格式的日期，如：var t = new Date (“2019/02/22”)

2. this.$route.query 如果参数太长，会导致ie页面奔溃，样式全部坍塌

* 1. webpack优化

1. devtool：eval 速度更快，缺点是不能正确显示console.log的打印行数
2. DllPlugin, HappyPack
3. 类似苹果高斯模糊遮罩效果

outline: 9999px solid rgba(0,0,0,.5); 设置这个盒子外的遮罩范围

backdrop-filter: blur(5px) 高斯模糊

1. 判断复杂数据类型

Object.prototype.toString.call({}) // "[object Object]"

1. vue $refs 获取元素距离顶部的距离

this.$refs.divRef.getBoundingClientRect().top

1. element ui 表单重置

this.$refs.form.resetFields();

// 单独验证某一个form-item

this.$refs.form1.validateField('point')

分解入参，变成数组

Array.prototype.slice.call(arguments) （element ui 每个组件也是有生命周期的，全局改动时可利用生命周期改动，如ElementUI.Select.mounted）

1. vue强制刷新页面，可代替this.$set()

this.$forceUpdate()

1. vs code git本地分支和远程对不上 报错：(would clobber existing tag)

用git ls-remote -t 查看远程tags

git tag -l查看本地tag

然后用 git tag -d xxx删除本地tag

最后远程拉取远程tags git fetch origin --prune-tags

参考地址： <https://www.cnblogs.com/little-ab/p/11460246.html>

1. 版本回退

git reset --hard c4718176a1de899bc754bde40c0fd3de081ec93a

git push –f

1. element ui switch开关字体在开关内显示

.switch-box {

// 设置宽度

/deep/.el-switch\_\_core {

width: 55px !important;

}

/deep/.el-switch\_\_label--left{

position: relative;

left: 60px;

color: #fff;

z-index: -1111;

}

/deep/.el-switch\_\_label--right{

position: relative;

right: 60px;

color: #fff;

z-index: -1111;

}

/deep/.el-switch\_\_label.is-active{

z-index: 1111;

color: #fff;

}

/deep/.el-switch\_\_label--right.is-active{

z-index: 1111;

color: #fff !important;

}

/deep/.el-switch\_\_label--left.is-active{

z-index: 1111;

color: #9c9c9c !important;

}

}

1. 云平台时间过滤 getDateTime('fulltime')
2. element ui upload终止上传 return false
3. position: -webkit-sticky;position: sticky;top: 0; 简单实现导航栏跟随置顶效果
4. watch 深度监听deep

watch 监听 '$route.meta', deep: true深度监听，只监听数组变化，属性变化监听无效

数组（一维、多维）的变化不需要通过深度监听，对象数组中对象的属性变化则需要deep深度监听。

watch: {

'$route.meta': {

handler (value) {

console.log(value)

value.businessBread && value.businessBread.length && (this.businessBread = value.businessBread)

value.pageDescrib && (this.pageDescrib = value.pageDescrib)

},

deep: true

}

}

1. 校验时间不交叉不重复

let timeCrossRules = (rule, value, callback) => {

if (value[0] === value[1]) {

return callback(new Error('时间区间不能为0'))

}

let dateValue = this.form.availables && this.form.availables.map(ele => ele.dateValue)

// 判断数组都不为空

if (dateValue.some((value) => { return !value })) {

return

}

// 排序

dateValue = dateValue.sort((next, last) => { return Number(next[0].split(':').join('')) - Number(last[0].split(':').join('')) })

// 如果前一个数组的第0个小于当前数组的第1个，则return出来，为true，就表示有重复和交叉

let dataFlag = dateValue.some((value, index, array) => {

if (index > 0) {

return Number(dateValue[index][0] && dateValue[index][0].split(':').join('')) < (Number(dateValue[index - 1] && dateValue[index - 1][1].split(':').join('')))

}

})

// 为true，就表示有重复和交叉

if (dataFlag) {

callback(new Error('时间不能交叉或重叠'))

}

callback()

}

1. JSON.parse(JSON.stringify())的弊端

1.无法解决循环引用的问题 如：

const a = {val: 2}

a.targe = a

2. 无法拷贝一些特殊对象，诸如 RegExp, Date, Set, Map等

3. 无法拷贝函数

var obj = {a: 1, b: {a: 1, b: 2}, c: /a/, d: function (){}}

JSON.parse(JSON.stringify(obj))

// {a: 1, b: {a: 1, b: 2}, c: {}}

1. 完整的深拷贝函数 （见神三元 掘金）
2. 普通深拷贝 （缺点是无法解决循环引用）

deepcopy = function (source) {

if (!source) {

return source

}

let sourceCopy = source instanceof Array ? [] : {}

for (let item in source) {

sourceCopy[item] =

typeof source[item] === 'object' ? deepcopy(source[item]) : source[item]

}

return sourceCopy

}

或

function deepClone (tar) {

if (typeof tar === 'object') {

let cloneTar = Array.isArray(tar) ? [0] : {}

for (const key in tar) {

cloneTar[key] = deepClone(tar[key])

}

return cloneTar

} else {

return tar

}

}

1. 可以用函数parseInt(3).toString(2)，将10进制的3转换为2进制
2. 如何在element 表单验证validator的验证规则里传参（再套一层函数）

let pureNumber = (msg) => {

return (rule, value, callback) => {

const valReg = /^[0-9]\*$/

if (!valReg.test(value) && !!value) {

return callback(new Error(`${msg}应为纯数字！`))

} else {

callback()

}

}

}

1. lodash的常用方法

1. \_.omit(obj, [ 'type' ]) // 拿到obj里除了type属性组成的对象

2. \_.debounce(function () {...}, 1000, { 'leading': true, 'trailing': false }) // 防抖

3. 判断两个对象是否相等

\_.isEqual(obj1, obj2)

4. 取一个对象里有相同的属性并赋值 pick: 赋值， keys： 拿相同的属性

var model = {

fname: null,

lname: null

}

var credentials = {

fname: 'xyz',

lname: 'abc',

age: 23

}

var result = \_.pick(credentials, \_.keys(model))

console.log(result, 'ddddd')

5.拿到相同属性的值，并保留原来的值

let tar = {a: '', b: '', c: '', e: ''}

let obj = {a: '1', b: '2', c: '3', d: '5'}

console.log(this.getEqualData(tar, obj), 'ddddd')

getEqualData (tar, obj) {

tar = \_.cloneDeep(tar)

obj = \_.cloneDeep(obj)

for (const item of Object.keys(tar)) {

for (const ite of Object.keys(obj)) {

if (item === ite) {

tar[item] = obj[ite]

}

}

}

return tar

},

1. 合并代码，只有一次记录

git merge [branch name] --no-commit

1. element ui 下拉框不随着页面滚动而移动问题

popper-append-to-body = "false" 加这个属性，加了下拉框就不会随页面滚动而滚动了。要改其样式就不能加scoped了

1. some、every

some: 遇到一个true就为true，全为false才为false

every： 全为true才为true， 碰到一个false则为false

1. 创建一个虚拟标签，获取标签信息，如音频 图片等

new Image()

beforeAvatarUpload (file, typeList = ['jpeg'], size = 5) {

const isType = typeList.some((ele) => file.type === `image/${ele}`)

const isLtM = file.size / 1024 / 1024 < size

let alertMsg = ''

if (!isType) {

let msg = typeList.reduce((pre, cur) => {

if (cur === 'jpeg') {

return pre + '，jpg，jpeg'

} else {

return pre + '，' + cur

}

}, '')

alertMsg = `请上传${msg.substr(1)}格式的图片！`

// this.$message.error(`请上传${msg.substr(1)}格式的图片！`)

} else if (!isLtM) {

alertMsg = `请选择小于${size}M大小的图片!`

}

// 图片文件大小限制，限制宽高分别为1280px和800px

let \_this = this

let imgWidth = ''

let imgHight = ''

const viewport = new Promise(function (resolve, reject) {

let width = 100

let height = 10

let \_URL = window.URL || window.webkitURL

let img = new Image()

img.onload = function () {

imgWidth = img.width

imgHight = img.height

let valid = img.width === width && img.height === height

console.log(typeof img.width, img.height, valid, 'with')

alertMsg = valid ? alertMsg : '上传文件的图片大小不合符标准,宽需要为1280px，高需要为800px。当前上传图片的宽高分别为：' + imgWidth + 'px和' + imgHight + 'px'

isType && isLtM && valid ? resolve() : reject(new Error())

}

img.src = \_URL.createObjectURL(file)

}).then(() => {

return file

}, () => {

\_this.$message.warning(alertMsg)

return Promise.reject(new Error())

})

console.log(viewport)

return viewport

// return isType && isLtM

},

1. 自定义指令监听div宽高变化

directives: { // 使用局部注册指令的方式, 监听某元素宽高变化

resize: { // 指令的名称

bind (el, binding) { // el为绑定的元素，binding为绑定给指令的对象

let width = ''

let height = ''

function isReize () {

const style = document.defaultView.getComputedStyle(el)

if (width !== style.width || height !== style.height) {

binding.value() // 关键

}

width = style.width

height = style.height

}

el.\_\_vueSetInterval\_\_ = setInterval(isReize, 300)

},

unbind (el) {

clearInterval(el.\_\_vueSetInterval\_\_)

}

}

}

1. 重置vue当前data里的值

this.form = this.$options.data().form

1. IE兼容问题

1. new Date(“2019-02-22”)转换失败，需要换成斜杠格式的日期，如：var t = new Date (“2019/02/22”)

2. this.$route.query 如果参数太长，会导致ie页面奔溃，样式全部坍塌

1. 区分ie ie6 ie7 ie8的方法

var isIE=!!window.ActiveXObject;

var isIE6=isIE&&!window.XMLHttpRequest;

var isIE8=isIE&&!!document.documentMode;

var isIE7=isIE&&!isIE6&&!isIE8;

1. 如何平滑滚动到页面顶部

const scrollToTop = () => {

const c = document.documentElement.scrollTop || document.body.scrollTop;

if (c > 0) {

window.requestAnimationFrame(scrollToTop);

window.scrollTo(0, c - c / 8);

}

}

scrollToTop()

1. 用nodejs，将base64转化成png文件，或者相反

// 将base64格式转成png格式

const baseData = require('./data.js')

const fs = require('fs');

const pngPath = 'config/image/'+ Date.now() +'.png';

const base64 = baseData.replace(/^data:image\/\w+;base64,/, "");//去掉图片base64码前面部分data:image/png;base64

const dataBuffer = new Buffer(base64, 'base64'); //把base64码转成buffer对象，

fs.writeFile(pngPath, dataBuffer, function(err){ //用fs写入文件

if(err){

console.log(err, '------------------------err');

}else{

console.log('写入成功！');

}

})

// 将png或者其它格式转成base64

const path = require('path');

const mineType = require('mime-types');

let filePath = path.resolve('config/image/1586918583626.png');

let data = fs.readFileSync(filePath);

data = new Buffer(data).toString('base64');

let base64Data = 'data:' + mineType.lookup(filePath) + ';base64,' + data;

console.log(base64Data, 'base64Data')

// fs.writeFileSync(path.resolve('your/save/file/path'), base64);

1. 提交时，滚动到第一个验证提示处 scrollIntoView

toError () {

if (document.getElementsByClassName('is-error').length > 0) {

document.getElementsByClassName('is-error')[0].scrollIntoView({behavior: 'smooth', block: 'start', inline: 'nearest'});

}

}

1. 创建一个长度为10，值为undefined的数组

Array.apply(null, {length: 10})

Array.apply(null, Array(10))

1. 怎样把一个类数组变成数组

1.for of循环添加进去

2.Array.prototype.slice.call(类数组)

3.es6新方法 Array.from(类数组)

1. 防抖，节流

export let debounce = function (fn, time = 2000) {

let timeout = null;

return function () {

clearTimeout(timeout);

timeout = setTimeout(() => {

fn.call(this, arguments);

}, time);

};

};

export let throttle = function (fn, time = 2000) {

let canRun = true;

return function () {

let this\_ = this;

let arg = arguments;

if (!canRun) return;

canRun = false;

setTimeout(() => {

canRun = true;

fn.call(this\_, arg);

}, time);

};

};

1. element ui日期区间选择，限制选择区间

pickerOptions: {

disabledDate (time) {

// 区间为当前时间至一个月后

return (time.getTime() < Date.now() - 8.64e7) || (time.getTime() > new Date(new Date().setMonth(new Date().getMonth() + 1)).valueOf());

}

}

1. 文件流怎么下载excel文件

后台返回excel文件回来，转成blob格式下载，axios请求时就处理

import Axios from '@/assets/js/AxiosPlugin.js';

import FileSaver from 'file-saver';

export const exportLocationsData = data => {

return Axios({

responseType: 'blob',

method: 'post',

url: exportLocations,

data: data

})

.then(

res => {

let blob = new Blob([res.data], {type: 'application/vnd.ms-excel'});

FileSaver.saveAs(blob, '足迹地图点位信息.xlsx');

}

);

};

1. vue富文本 大神推荐： tinymce

管理后台富文本也是一个非常重要的功能，楼主在这里也踩了不少的坑。楼主在项目里最终选择了 [tinymce](https://github.com/tinymce/tinymce)

这里在简述一下推荐使用tinymce的原因：tinymce 是一家老牌做富文本的公司(这里也推荐 ckeditor，也是一家一直做富文本的公司，新版本很不错)，它的产品经受了市场的认可，不管是文档还是配置的自由度都很好。在使用富文本的时候有一点也很关键就是复制格式化，之前在用一款韩国人做的富文本summernote被它的格式化坑的死去活来，但 tinymce 的去格式化相当的好，它还有一个增值项目就是powerpaste,那是无比的强大，支持从word里面复制各种东西，都不会有问题。富文本还有一点也很关键，就是拓展性。楼主用tinymce写了好几个插件，学习成本和容易度都不错，很方便拓展。最后一点就是文档很完善，基本你想得到的配置项，它都有。tinymce也支持按需加载，你可以通过它官方的build页定制自己需要的plugins。 我再来分析一下市面上其它的一些富文本：

* [**summernote**](https://github.com/summernote/summernote) 先来说一个我绝对不推荐的富文本。这是一个韩国人开源的富文本(当然不推荐的理由不是因为这个)，它对很多富文本业界公认的默认行为理解是反起到而行的，而且只为用了一个dialog的功能，引入了boostrap，一堆人抗议就是不改。格式化也是差劲。。反正不要用！不要用！不要用！
* [**ckeditor**](https://github.com/galetahub/ckeditor) ckeditor也是一家老牌做富文本的公司，楼主旧版后台用的就是这个，今年也出了5.0版本，ui也变美观了不少，相当的不错，而且它号称是插件最丰富的富文本了。推荐大家也可以试用一下。
* [**quill**](https://github.com/quilljs/quill) 也是一个非常火的富文本，长相很不错。基于它写插件也很简单，api设计也很简单。楼主不选择它的原因是它对图片的各种操作不友善，而且很难改。如果对图片没什么操作的用户，推荐使用。
* [**medium-editor**](https://github.com/yabwe/medium-editor) 大名鼎鼎的medium的富文本(非官方出品)，但完成度还是不很不错，拓展性也不错。不过我觉得大部分用户还是会不习惯medium这种写作方式的。
* [**Squire**](https://github.com/neilj/Squire) 一个比较轻量的富文本，压缩完才11.5kb，相对于其它的富文本来说是非常的小了，推荐功能不复杂的建议使用。
* [**wangEditor**](https://github.com/wangfupeng1988/wangEditor) 一个国人写的富文本，用过感觉还是不错的。不过毕竟是个人的，不像专门公司做富文本的，配置型和丰富性不足。前端几大禁忌就有富文本 [为什么都说富文本编辑器是天坑?](https://www.zhihu.com/question/38699645)，不过个人能做成这样子很不容易了。
* [**百度UEditor**](http://ueditor.baidu.com/website/index.html) 没有深入使用过，只在一个angular1X的项目简单用过，不过说着的ui真的不好看，不符合当今审美了，官方也已经很久没跟新过了。

楼主列举了很多富文本但并没有列举任何 vue 相关的富文本，主要是因为富文本真的比想象中复杂，在前面的文章里也说过了，其实用 vue 封装组件很方便的，没必要去用人家封装的东西什么vue-quill vue-editor这种都只是简单包了一层，没什么难度的。还不如自己来封装，灵活性可控性更强一点。还有一点基于 vue 真没什么好的富文本，不像 react 有 facebook 出的 [draft-js](https://github.com/facebook/draft-js)，ory 出的 [editor](https://github.com/ory/editor)，这种大厂出的产品。

当然你也可以选择一些付费的富文本编辑器，作者自己公司里面有一个项目就使用了 [froala-editor](https://www.froala.com/wysiwyg-editor) 这款编辑器。不管是美观和易用性都是不错的，公司买的是专业版，一年也就 $349 ，价格也是很合理的，但其实省去的程序员开发陈本可能远不止这个价钱。

作者：花裤衩  
链接：https://juejin.im/post/593121aa0ce4630057f70d35  
来源：掘金  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

Ug build : 53931ms

My-vue-cli: 87395ms