web前端面试宝典1（<https://github.com/markyun/My-blog/tree/master/Front-end-Developer-Questions>）

web前端面试宝典2（<https://github.com/h5bp/Front-end-Developer-Interview-Questions/tree/master/Translations/Chinese>）

面试题：

一：页面加载过程是什么？（输入一个URL发生了什么？）

URL解析——DNS服务器上进行查询——资源请求——浏览器解析

（url结构：[http://www.happymmall.com:80/get\_data.do?productId=1#title](http://www.happymmall.com/get_data.do?productId=1" \l "title)）

http是请求协议

[www.happymmall.com:80](http://www.happymmall.com/)域名和默认端口（可省略），域名是用来查找服务器位置的，端口就好比钥匙

get\_data.do 是路径，服务器接到请求以后，拿这个路径在服务器上定位资源位置的

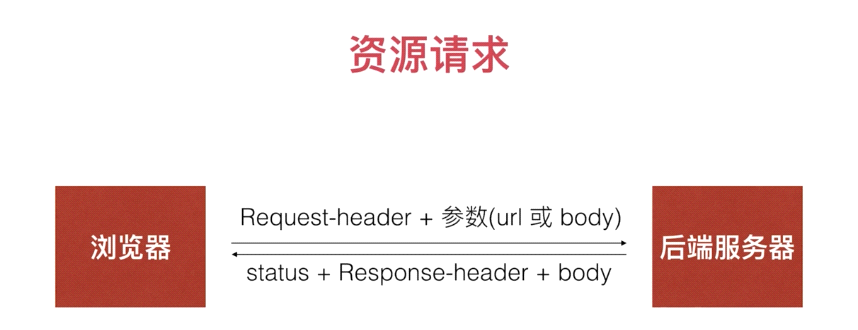
productId=1是请求参数

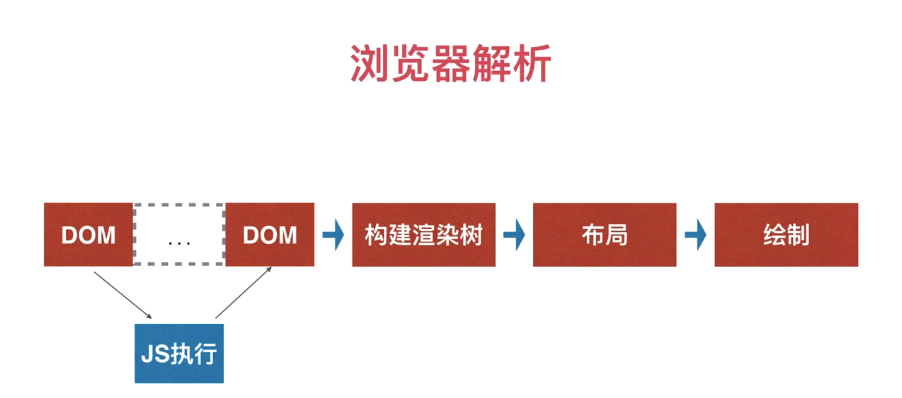
#titlehach 前端页面的锚点



DNS请求效率优化参考







二：什么是跨域？可参考<https://www.jianshu.com/p/b95813068827>

跨域，指的是浏览器不能执行其他网站的脚本。

解决办法一：jsonp(只支持get请求，不支持post请求)

解决办法二：前后端统一http的请求头。

解决办法三：使用代理工具。

解决办法四：使用服务器端代理插件http-proxy-middleware

在config文件中的index.js中的proxyTable

proxyTable: {//设置服务器代理

"/taokubuy": {

"target":"http://198.33.33.33:8000,

"changeOrigin":true,

"pathRewrite": {

"^/api":"" // api相当于一个别名，代指[http://198.33.33.33:8000](http://198.33.33.33:8000/)

}

}

},

比如你要请求跨域请求数据<http://198.33.33.33:8000/data> 只要请求/api/data即可

$.ajax({url:"/api/data",

success:function(res){

console.log(res)

}

})

同理：react也可以对proxy进行配置

三：this的作用域？

<https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/this>

三：单列以及多列布局？

关键点：左侧定宽+左侧浮动+右边距margin,右边不定宽不浮动

<http://blog.csdn.net/u012999771/article/details/49253059>

三：BFC是什么鬼？

就是一种布局方式，overflow:hidden;float:left;position:absolute;display:inline-block;可以用来解决浮动元素乱跑的问题

三：HTTP协议？

四：几个重要的宽高

offsetWidth:width+padding\*2+border\*2,

clientWidth:width+padding\*2

五：盒模型和怪异盒模型

盒模型包括margin,padding,border,content，在W3C下content部分不包含其他部分，而在ie下，content包含border和padding

六：渐进增强(progressive enhancement)和优雅降级(graceful degradation)之间的不同吗？

渐进增强（progressive enhancement）：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

优雅降级（graceful degradation）：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

例子：

.transition{ -webkit-transition: all .5s; -moz-transition: all .5s; -o-transition: all .5s; transition: all .5s; } 这个就是渐性增强

.transition{transition:all.5s;-o-transition: all .5s; 　-moz-transition: all .5s; -webkit-transition: all .5s;} 这个就是优雅降级

七：假若你有 5 个不同的样式文件 (stylesheets), 整合进网站的最好方式是?

根据样式书写规则，提取公共的样式，逐行合并，非公共的单独拧出来，然后打包压缩一下就行了，若每个文件都很大，就需要分模块化加载。

八：请说出三种减少页面加载时间的方法。

1. 减少HTTP请求；
2. 使用CDN
3. 尽可能少的操作DOM

九：css3动画和JS动画优劣？

1. css3动画只兼容Ie10+，js动画几乎兼容所有浏览器；
2. js动画更灵活，css3动画更简单
3. css3有天然的时间支持如（animationENd和transitionEnd）

十：doctype(文档类型) 的作用是什么？

doctype使浏览器按照dtd指定的渲染方式对页面进行渲染

十一：浏览器标准模式 (standards mode) 、几乎标准模式（almost standards mode）和怪异模式 (quirks mode) 之间的区别是什么？

所谓的标准模式是指，浏览器按W3C标准解析执行代码；怪异模式则是使用浏览器自己的方式解析执行代码

十二：继承

先执行父级的构造函数，然后再添加自己的属性和方法

主要实现原理：构造函数伪装和原型链继承父级的方法

十三：面向对象

特点：封装，继承，多态，抽象

理解：构造函数就是类，类就是构造函数，然后是给构造函数添加原型的方法

十四：事件捕获和事件冒泡

这里有个小知识点：使用事件委托可以提高页面的性能<https://www.cnblogs.com/Chen-XiaoJun/p/6210987.html>

十五：css选择器

Id，类型选择器，元素选择器，属性选择器

十六：===和==等号的区别

==之比较“转化成同一类型后的值”看“值”是否相等，===如果类型不同，其结果就是不等

十七：JSON对象和JS对象是否相同

JSON仅仅是一种数据格式，js表示类的实例。总而言之JSON是JS下的一种数据格式，他从属于JS，并且在处理JSON数据时可直接使用JS内置API

十八：canvas实现雪花效果

十八：闭包

简单的理解为定义在一个函数内部的函数

十八：竖直方向上居中

parentElement{

Position:relative;

}

childElement{

Position:absolute;

Top:50%;

Transform:translateY(-50%);

}

方法二:用flex布局

parentElement{

display:flex;

display:-webkit-flex;

align-items:center;

}

十九：手机响应式开发

<http://blog.csdn.net/zhangyufeng0126/article/details/78716059>

二十：jQuery中attr()和prop()的区别

对于html元素本身就有的固有属性，使用prop()方法;

对于html元素我们自己定义的DOM属性，在处理时使用attr();

二十：数据类型

7种：字符串，数字，数组，对象，布尔，undefined，null

NAN是，不是数字的数字类型

二十：构造函数伪装

主要在面向对象继承那一块使用。例子：Person.call(this,name,sex)

二十一：原型链

例如：Person.prototype = Worker.prototype （这里的等号相当于链条）切记在使用原型链的时候一定要加个for in循环

二十二：引用

例如：Person.prototype = Worker.prototype，

例如：

var arr1 = [1,2,3];

var arr2 = arr1;

arr2.push(4);

alert(arr1) 为1234

alert(arr2) 为1234

引用这是很危险的，解决办法

for(var i in arr1){

arr[i] = arr[i];

}

二十一：after()和append()的区别

after()是在元素节点外后面添加，append是在元素节点内部的后面添加

二十二：instanceof

用来判断谁是谁的实例

var arr1 = [1,2,3]

alert(arr1 instanceof Array) 就是true， 意思是说arr1是Array的实例

二十三：粘性布局

position:sticky;

二十四：内存泄漏

不再用到的内存，没有及时释放，就会造成内存泄漏（memory leak）。像大量的使用了闭包及全局变量

二十四：各种存储和缓存的理解及区别

cookies主要是用来解决请求头常带存储信息的问题，小于4K

本地存储localstorage永久存储，除非手动删除。

localStorage与sessionStorage在使用方法上是相同的，区别在于sessionStorage在关闭页面后即被清空。

离线缓存application cache本地缓存应用所需的文件

二十四：flex布局（很爱考的一道题）

垂直对齐方式

align-items : center;

<http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html?utm_source=tuicool>

二十五：数组的去重

var arr = [1,2,3,4,5,1,2,4];

function unique1(array){

var arr1 = [];

for(var i = 0;i < array.length; i++){

if(arr1.indexOf(array[i]) === -1){

arr1.push(array[i]);

}

}

return arr1;

}

alert(unique1(arr));

二十六：如何给一个不具备iterator接口的数据结构部署一个iterator接口？

二十七：rem和em和vm的区别

rem是基于HTML根元素来设置的62.5%，em是根据父级来设置的，vm是根据窗口来设置的

二十八：ES6定义变量的特点

1. 不能重复定义
2. 只能在块作用域下起作用
3. 在欲解析的时候是不会被提升的

二十九：箭头函数的特点

1. 没有独立的作用域
2. 不能用做构造函数
3. 没有prototype属性

三十：模板字符串的特点

1. 用反引号标识 ``
2. 支持多行字符串
3. 模板字符串支持变量和表达式

三十：冒泡排序

var subbleSort = function (arr){

for(var i =0;i<arr.length-1;i++){

for(var j = i+1;j<arr.length;j++){

If(arr[i]>=arr[j]){

var max = arr[i];

arr[i] = arr[j];

arr[j] = max;

}

}

}

return arr;

}

三十一：计算add(2)(3)

const add = (...args) =>{

let sum = args.reduce( (num1,num2) =>{

num1 + num2;

});

const add2 = (...args2) =>{

sum += args2.reduce( (num1,num2) =>{

num1 + num2;

});

return add2;

}

add2.valueOf = () => sum;

return add2;

}

add(2,3,4) // 9

add(2)(5) // 7

add(2,3)(5)(1) // 11

三十二：几种存储的Local Storage,Session Storage,Web SQL,Cookies,Cache Storage的区别

存储方式一：

<http://blog.csdn.net/LiuHanQingIT/article/details/78028469>

三十三：Redux的原理及用法

存储方式一：

<http://blog.csdn.net/LiuHanQingIT/article/details/78028469>

三十四：手些Ajax()的原理

function ajax(method, url, data, success) {

var xhr = null;

try {

xhr = new XMLHttpRequest();

} catch (e) {

xhr = new ActiveXObject('Microsoft.XMLHTTP');

}

if (method == 'get' && data) {

url += '?' + data;

}

xhr.open(method,url,true);

if (method == 'get') {

xhr.send();

} else {

xhr.setRequestHeader('content-type', 'application/x-www-form-urlencoded');

xhr.send(data);

}

xhr.onreadystatechange = function() {

if ( xhr.readyState == 4 ) {

if ( xhr.status == 200 ) {

success && success(xhr.responseText);

} else {

alert('出错了,Err：' + xhr.status);

}

}

}

}

三十五：call()和apply()的区别？

call方法参数将依次传递给借用的方法作参数, 而apply直接将这些参数放到一个数组中再传递。

当参数不明确时可用apply结合arguments使用

三十六：map()和forEach()的区别？

相同点：

都是循环遍历数组中的每一项，

每一次执行匿名函数时都支持三个参数，分别为item,index,arr(原数组可省略不写)

匿名函数内部的this都指的window

只能遍历有值的数组

不同点：

map()方法返回一个新的数组（不会改变原始数组），新数组中的元素为原始数组调用匿名函数处理后的值。

forEach()方法用于调用数组中的每个元素，将元素传给回调函数（这里没有谈及到返回值）会改变原来的数组

三十七：本周安排，

学会react的产品页及列表页，项目的上线过程，项目的启动过程，以及分页及项目再走一遍逻辑。

三十八：类数组对象arguments的妙用

<http://blog.csdn.net/qq_16339527/article/details/53231725>

三十九：JSON.stringify()的毛用

JSON.stringify() [从一个对象中解析出字符串]

var data={name:'goatling'}

JSON.stringify(data)

结果是：

'{"name":"goatling"}'

四十：Flexbox如何将页面底部固定在屏幕最下方

flex布局

四十一：获取数组中的最大值和最小值

alert(Math.min.apply(null,arr));//获取数组中最小值

alert(Math.max.apply(null,arr));//获取数组中的最大值

四十二：javascript事件流的理解（通俗的讲：事件的捕获，事件冒泡）

事件流分为三个阶段：捕获过程，目标过程，冒泡过程。一个完整的js事件流是从window开始，到目标对象，最后回到window的一个过程

四十三：IOS系统对position:fixed不兼容怎么办？

这个问题有待回答

四十四：如何快速复制一个数组

方法一：通过遍历。

方法二：用数组方法concat一个空数组。var a=[1,2,3];var c=[].concat(a);

方法三：arr.slice();

四十五：如何快速复制一个对象

方法一：通过 for (var key in obj) { }进行遍历。

方法二：通过var newObj = JSON.parse(JSON.stringify(obj))

vue框架的考点

一.vue的基础知识：

vue-router的钩子函数分为三类：全局钩子，单个路由里面的钩子，组件钩子

全局钩子： beforeEach， (to,from,next)三个参数,afterEach

单个路由里面的钩子: beforeEnter, beforeLeave

组件钩子: beforeRouterEnter, beforeRouterUpdate,beforeRouteLeave

vuex的钩子函数：

...mapState，...mapGetters，...mapMutations，...mapAction，...mapModule

vue的钩子函数：（10个）beforeCreate,created,beforeMount,mounted,beforeUpdate,updated,activated,deactivated,beforeDestory,destoryed

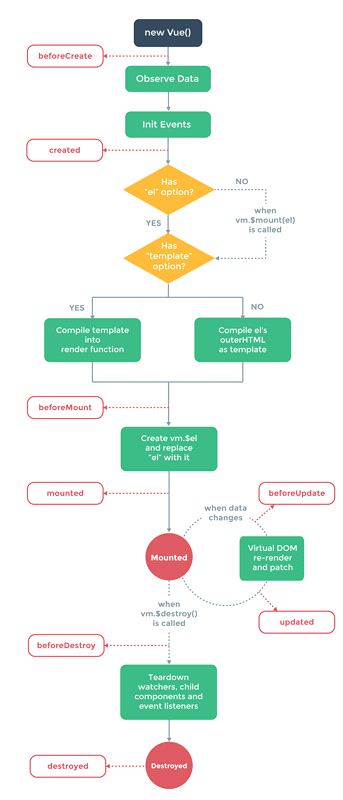
二．vue路由跳转参数的三种方式

如果通过this.$router.push()跳转的话有两种方式：name和params的结合，path和query的结合。

第三种就是在path中直接使用模板字符串进行传值

三．vue和react的优缺点:

四．vue的生命周期



五．兄弟组件如何传值：

可以利用mixin,也可以利用他们共同的父级，还可以用vuex的仓库

Router-view和router-link的区别

可以利用mixin,也可以利用他们共同的父级，还可以用vuex的仓库

六．vue如何做到深度监听？

immediate: true,

deep: true

七．vue中的懒加载和按需加载

懒加载：也叫延迟加载，即在需要的时候，才进行加载。

按需加载：需要的时候才加载

1．vue中路由懒加载

export default new Router({

routes: [

{

path: '/my',

name: 'my',

component: resolve => require(['../page/my/my.vue'], resolve) , // 懒加载

}

]

})

2．组件的懒加载

components: {

historyTab: resolve => {

require(['../../component/historyTab/historyTab.vue'], resolve)

},

}

3．全局懒加载

Vue.component('mideaHeader', () => {

System.import('./component/header/header.vue')

})

按需加载原因：首屏优化，第三方组件库依赖过大,会给首屏加载带来很大的压力，一般解决方式是按需要，引入组件。如：element-ui 按需加载的方法：

{

"presets": [

["es2015", { "modules": false }]

],

"plugins": [

[

"component",

[{

"libraryName": "element-ui",

"styleLibraryName": "theme-default"

}]

]

]

}

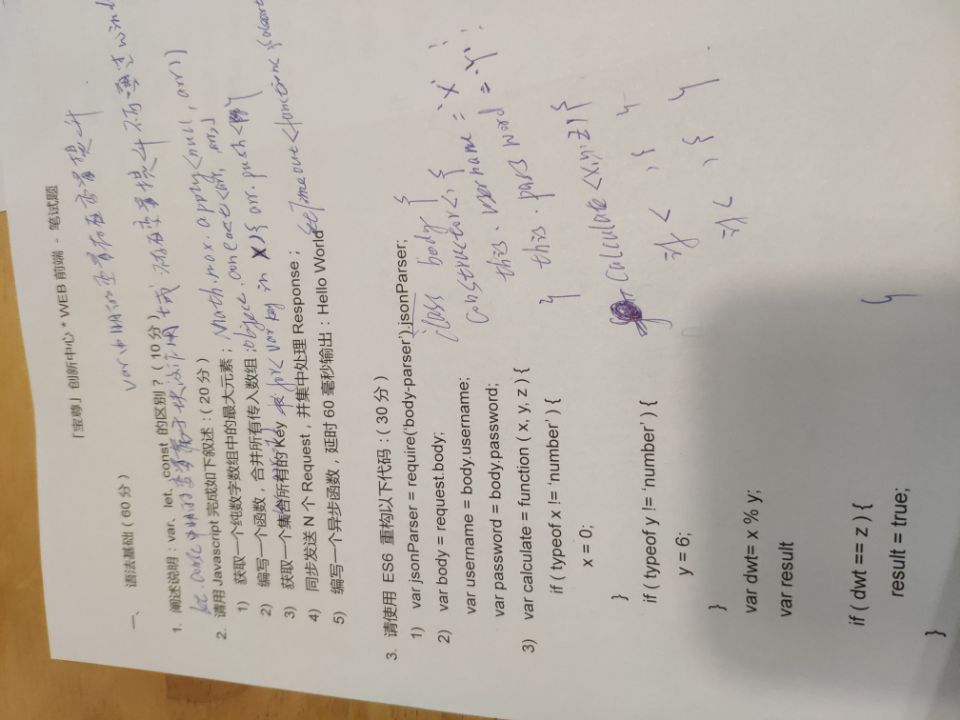
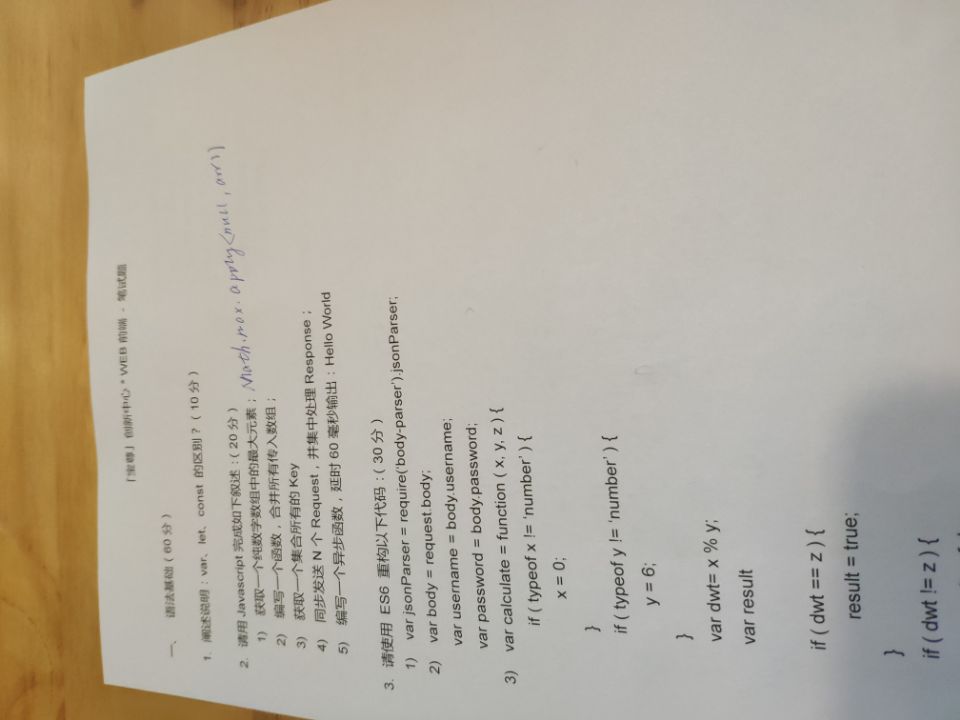
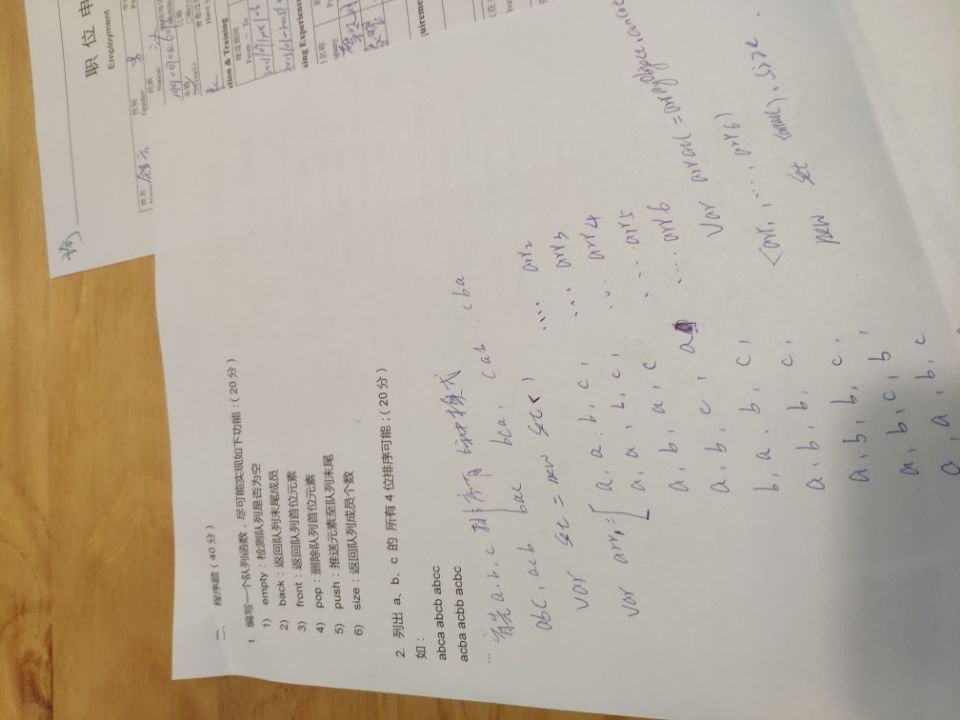
八．如何解决打包过大的问题？

immediate: true,

九．v-model实现数据的绑定的真实原理？

immediate: true,

十．真实的面试题！！！！！！

  
ES6知识的考点

<https://www.cnblogs.com/fengxiongZz/p/8191503.html>

一．箭头函数的特点：(背一遍)

1. 在使用=>{}定义函数的时候，this是指定义时所在的对象，而不是使用时所在的对象。（也就是说：非严格模式下，setTimeout中所执行函数中的this，永远指向window）

这个是使用原来的函数

class Animal {

constructor() {

this.type = "animal";

}

say(val) {

setTimeout(function () {

console.log(this); //window

console.log(this.type + " says " + val);

}, 1000)

}

}

var animal = new Animal();

animal.say("hi"); //undefined says hi

这个使用的是箭头函数

class Animal {

constructor() {

this.type = "animal";

}

say(val) {

setTimeout(() => {

console.log(this); //Animal

console.log(this.type + ' says ' + val);

}, 1000)

}

}

var animal = new Animal();

animal.say("hi"); //animal says hi

1. 箭头函数不能用作构造函数，不能使用new命令。
2. 不能使用arguments对象。
3. 不需要function关键字创建函数
4. 省略return关键字

二．let和const申明变量(背一遍)

1. let和const申明的变量具有块级作用域，通常情况下不会发生变量提升。const定义的常量值，不能够重新赋值，如果值是一个对象，可以改变对象里面的属性值。
2. let申明的变量不能通过window.变量名进行访问。
3. For(let x..)的循环每次迭代都为x创建新的绑定

三．class构造函数及构造函数的继承(背一遍)

class Animal {

constructor() {

this.type = 'animal'

}

says(say) {

console.log(this.type + 'says' + say)

}

}

let animal = new Animal()

animal.says('hello') // animal says hello

class Cat extends Animal {

constructor() {

super()

this.type = 'cat'

}

}

let cat = new Cat()

cat.says('hello') // cat says hello

因为在ES6中，子类的构造函数必须含有super函数，

4.模板字符串(背一遍)

将表达式嵌入字符串中进行拼接，用${}来界定。

5.ES6新增的类型(背一遍)

Set是唯一无序的集合，和数组有点类似，但是没有索引,不会有重复数据，因此可以用来给数组去重.Set默认不做类型转换。虽然NaN在js中不想等,但Set内部认为他们是相同的

set声明: let set = new Set(); 即创建了一个空的set

set赋值: let set = new Set(['张三','李四','王五']); 输出结果为:张三，李四，王五

set去重:let set = new Set(['张三','李四','王五','张三','李四'] 输出结果仍然为:张三，李四，王五

set属性:size属性求set集合的大小(长度) console.log(set.size)集合的大小为3

set方法:

add方法，表示新增一个元素到set中，如:set.add('赵六')则输出结果为张三，李四，王五，赵六

delete方法，表示删除一个元素，返回的结果为true或者false。如:set.delete('张三')

has方法，判断某个元素是否在set这个集合中，如：set.has('李四')返回结果为true

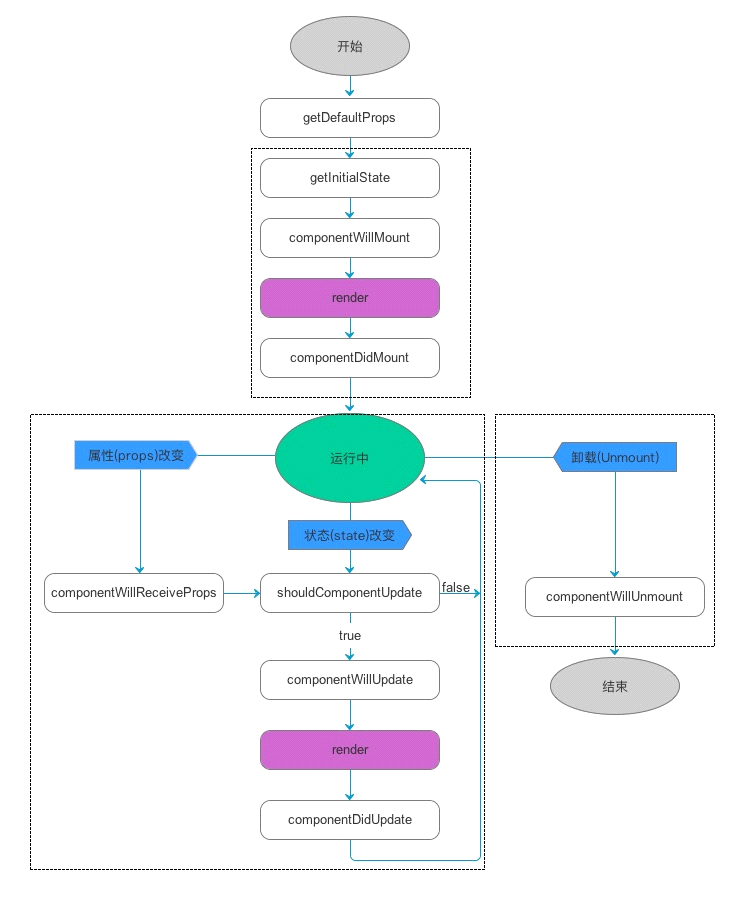
clear方法，表示清除集合中所有的元素，没有任何返回值。如:set.clear()则将元素全部清除。

还有一些map，forEach，Proxy（代理对象），class关键字，filter（过滤），Promise，Object.assign()，extends，super()， 解构赋值，扩展运算符，模板字符串，箭头函数。

Webpack构建工具的使用

面试题啊，面试题啊 面试题啊 面试踢啊免死谜团乃是啊 附魔附魔

React钩子函数的使用(背一遍)

getDefaultProps ——getInitialState——componentWillMount——render——componentDidMount——componentWillReceiveProps——shouldComponentUpdate——componentWillUpdate——render——componentDidUpdate——Unmount——componentWillUnmount

1. react的跳转及传参问题(背一遍)

第一种情况——通过params进行传参：

路由需要进行配置

<Route path='/sort/:id' component={sort}></Route>

Link处传参

<Link to={'/sort/' + '2'} activeClassName = 'active'></Link>

通过js方式进行传参

this.props.router.push('/sort/' + '2')

子组件sort页面接收参数的方式：

通过this.props.params.id

第二种情况——通过query进行传参：

路由不需要进行配置：

<Route path='/sort' component={Sort}></Route>

HTML方式进行传递

<Link to={{path:'/sort', query: {name: '张三'}}}>

通过JS的方向进行传递

this.props.router.push({path: '/sort', query: {name: '王二'}})

子组件sort页面通过如下方式进行获取参数

this.props.location.query.name

第三种情况——通过state进行传参：（不同之处：state传入的参数是保密的，query传递的参数是公开的）

路由不需要进行配置：

<Route path='/sort' component={Sort}></Route>

HTML方式进行传递

<Link to={{path:'/sort', state: {name: '林冲'}}}>

通过JS的方向进行传递

this.props.router.push({path: '/sort', state: {name: '李逵'}})

子组件sort页面通过如下方式进行获取参数

this.props.location.state.name

1. React父组件往子组件传值：父组件的state通过props传入子组件(背一遍)

父组件代码片段：

constructor(props){

this.state = {

message: "我是父组件传来的"

}

}

render() {

return (

<div style={{background: "red", padding: "30px"}}>

<Son msg={this.state.message}/>

</div>

)

}

子组件代码片段：

<div>

父组件传到子组件的信息：<span>{this.props.msg}</span>

</div>

1. React子组件往父组件传值（思想是：父组件将传参的函数通过props传递给子组件，子组件可以调用这个函数，并传入新的值）(背一遍)

父组件片段代码：

constructor(props){

super(props)

this.state={

visible:false,

}

}

// 进入

goIn(){

this.setState({

visible:true

})

}

// 取消

cancel(mode){

console.log(mode)

this.setState({

visible:mode

})

}

render(){

return(

<div style={{background:"red",padding:"30px"}}>

{

this.state.visible ?

<div style={{background:"yellow"}}>

<Son cancelFn={mode=>this.cancel(mode)}/>

</div>

:

<div style={{background:"blue"}}>

<button onClick={this.goIn.bind(this)} style={{width:"100px",height:"50px"}}>进入</button>

</div>

}

</div>

)

}

子组件片段代码：

class Son extends React.Component{

render(){

console.log(this.props.msg)

return(

<div>

<button onClick={()=>{this.props.cancelFn(false)}}>返回</button>

父组件传到子组件的信息：<span style={{background:"white"}}>{this.props.msg}</span>

</div>

)

}

}

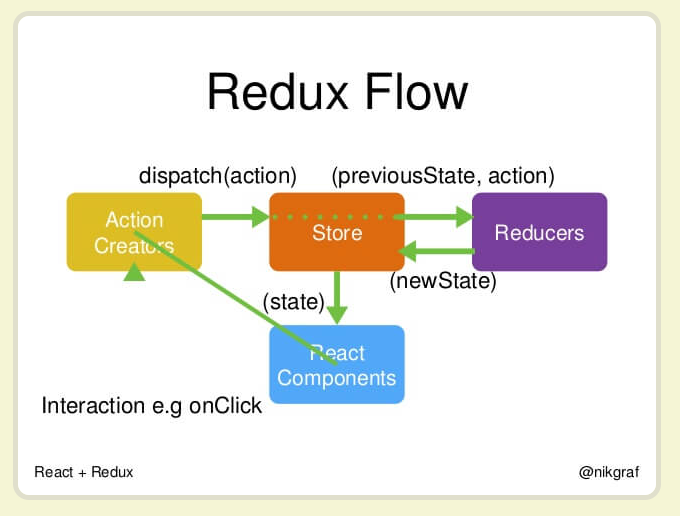
Redux的原理(背一遍)

首先，用户发出 Action，即store.dispatch(action);

然后，Store 自动调用 Reducer，并且传入两个参数：当前 State 和收到的 Action。 Reducer 会返回新的 State

State 一旦有变化，Store 就会调用监听函数store.subscribe(listener)

listener可以通过store.getState()得到当前状态



父组件如何调用子组件的方法：(背一遍)

在子组件身上绑定一个ref=’A’,父组件就这么调用：this.refs.A.Fn()

React和Vue如何做单页应用和多页应用

在子组件身上绑定一个ref=’A’,父组件就这么调用：this.refs.A.Fn()

VUE的双向绑定原理及实现

vue数据双向绑定的核心原理：通过Object.defineProperty()进行数据劫持，结合发布者-订阅者模式的方式来实现的。

主要是，通过以下3个步骤，来实现数据的双向绑定：

1.实现一个监听器Observer，用来劫持并监听所有属性，如果有变动的，就通知订阅者。

2.实现一个订阅者Watcher，可以收到属性的变化通知并执行相应的函数，从而更新视图。

3.实现一个解析器Compile，可以扫描和解析每个节点的相关指令，并根据初始化模板数据以及初始化相应的订阅器。

MVC模式和MVVM模式的区别：

MVC模式是单向绑定!

Model 🡪 View

MVVM是双向绑定!

Model ⇆ View

Vue面试题

生命周期函数面试题

1.什么是 vue 生命周期

2.vue生命周期的作用是什么

3.第一次页面加载会触发哪几个钩子

4.简述每个周期具体适合哪些场景

5.created和mounted的区别

6.vue获取数据在哪个周期函数

7.请详细说下你对vue生命周期的理解？

vue路由面试题

1.mvvm 框架是什么？

2.vue-router 是什么?它有哪些组件

3.active-class 是哪个组件的属性？

4.怎么定义 vue-router 的动态路由? 怎么获取传过来的值

5.vue-router 有哪几种导航钩子?

6.$route 和 $router 的区别

7.vue-router响应路由参数的变化

8.vue-router传参

9.vue-router的两种模式

10.vue-router实现路由懒加载（ 动态加载路由 ）

vue常见面试题

1.vue优点

2.vue父组件向子组件传递数据？

3.子组件像父组件传递事件

4.v-show和v-if指令的共同点和不同点

5.如何让CSS只在当前组件中起作用

6.<keep-alive></keep-alive>的作用是什么?

7.如何获取dom

8.说出几种vue当中的指令和它的用法？

9. vue-loader是什么？使用它的用途有哪些？

10.为什么使用key

11.axios及安装

12.axios解决跨域

13.v-modal的使用

14.scss的安装以及使用

15. 请说出vue.cli项目中src目录每个文件夹和文件的用法？

16.分别简述computed和watch的使用场景

17.v-on可以监听多个方法吗

18.$nextTick的使用

19.vue组件中data为什么必须是一个函数

20.vue事件对象的使用

21 组件间的通信

22.渐进式框架的理解

23.Vue中双向数据绑定是如何实现的

24.单页面应用和多页面应用区别及优缺点

25.vue中过滤器有什么作用及详解

26.v-if和v-for的优先级

27.assets和static的区别

28.列举常用的指令

29.vue常用的修饰符

30.数组更新检测

31.Vue.set视图更新

32.自定义指令详解

33.vue的两个核心点

34.vue和jQuery的区别

35 引进组件的步骤

36.Vue-cli打包命令是什么？打包后悔导致路径问题，应该在哪里修改

37.三大框架的对比

38. 跨组件双向数据绑定

39.delete和Vue.delete删除数组的区别

40.SPA首屏加载慢如何解决

41.Vue-router跳转和location.href有什么区别

42. vue slot

43.你们vue项目是打包了一个js文件，一个css文件，还是有多个文件？

44.vue遇到的坑，如何解决的？

45.Vue里面router-link在电脑上有用，在安卓上没反应怎么解决？

46.Vue2中注册在router-link上事件无效解决方法

47.RouterLink在IE和Firefox中不起作用（路由不跳转）的问题

48.axios的特点有哪些

49.请说下封装 vue 组件的过程？

50.vue 各种组件通信方法（父子 子父 兄弟 爷孙 毫无关系的组件）

51.params和query的区别

52. vue mock数据

53 vue封装通用组件

54.vue初始化页面闪动问题

55.vue禁止弹窗后的屏幕滚动

56.vue更新数组时触发视图更新的方法

57.vue常用的UI组件库

58. vue如何引进本地背景图片

59. vue如何引进sass

60.vue修改打包后静态资源路径的修改

vuex常见面试题

1.vuex是什么？怎么使用？哪种功能场景使用它？

2.vuex有哪几种属性

3.不使用Vuex会带来什么问题

4.Vue.js中ajax请求代码应该写在组件的methods中还是vuex的actions中？

5.vuex一个例子方法

6.Vuex中如何异步修改状态

7.Vuex中actions和mutations的区别

vue项目实战

1.顶部悬停效果

2.电话本列表效果（ 右边字母分类 上下滑动 旁边字母显示高亮）

3.vue做代理

4.Vue路由切换时的左滑和右滑效果示例

ES6面试题

ES6新增方法面试题

1.let const var比较

2.反引号（`）标识

3.函数默认参数

4.箭头函数

5.属性简写

6.方法简写

7.Object.keys()方法，获取对象的所有属性名或方法名

8.Object.assign ()原对象的属性和方法都合并到了目标对象

9.for...of 循环

10.import和export

11.Promise对象

12.解构赋值

13.set数据结构（可用于快速去重）

14.Spread Operator 展开运算符(...)

15.字符串新增方法

16.深拷贝和浅拷贝的区别及深拷贝的方法？

答：假设B复制了A，当修改A时，看B是否会发生变化，如果B也跟着变了，说明这是浅拷贝，

原理：

浅拷贝只是增加了一个指针指向已存在的内存地址，

深拷贝增加了一个指针并且申请了一个新的内存，这个增加的指针指向这个新申请的内存，

17.堆和栈的区别？

18.jsx的优点

ES6数组面试题

1.forEach()

2.map()

3.filter()

4.reduce()

5.some()

6.every()

7.all()方法

ES6编程题

1.使用解构，实现两个变量的值的交换

2.利用数组推导，计算出数组 [1,2,3,4] 每一个元素的平方并组成新的数组。

3.使用ES6改下面的模板

4.把以下代码使用两种方法，来依次输出0到9？

react面试题

react生命周期面试题

1.react 生命周期函数：

getDefaultProps()

getInitialState()

componentWillMount()

componentDidMount()

componentWillReceiveProps()

shouldComponentUpdate()

componentWillUpdate()

componentDidUpdate()

componentWillUnmount()

2.react生命周期中，最适合与服务端进行数据交互的是哪个函数：

componentDidMount：在这个阶段，实例和dom已经挂载完成，可以进行相关的dom操作

3.当组件发生更新时，组件的生命周期调用顺序：

componentWillReceivProps --- shouldComponentUpdate --- componentWillUpdata --- render --- componentDidUpdate

4.shouldComponentUpdate 是做什么的，（react 性能优化是哪个周期函数？）：

shouldComponentUpdate 手动判断组件是否要更新，根据组件的应用场景设置函数的合理返回值，能够帮我们避免不必要的更新

5.指出(组件)生命周期方法的不同：

6.shouldComponentUpdate 是做什么的，（react 性能优化是哪个周期函数？）：

shouldComp 18616268167

7.shouldComponentUpdate 是做什么的，（react 性能优化是哪个周期函数？）：

shoul

react 基础面试题

1.react 中 keys 的作用是什么？

keys是React，在操作列表中元素，被修改,添加,或者删除的辅助标识。在reactDiff算法中，React会借助元素的Key值，来判断该元素是新创建的，还是被移动而来的。React会保存这个辅助状态,从而减少不必要的元素渲染。

2.React 中 refs 的作用是什么？

refs是React 提供的安全访问 DOM 元素或者某个组件实例的句柄。我们可以为元素添加 ref 属性，然后在回调函数中，接受该元素在 DOM 树中的句柄，该值会作为回调函数的第一个参数返回：

class CustomerForm extends Component{

handleSubmit = () => {

console.log('Input Value:'+this.input.value);

}

render(){

return (

<form onSubmit = {this.handleSubmit}>

<input type="text" ref={(input)=> this.input = input } />

<button type="submit">Submit</botton>

</form>

)

}

}

Input域中包含了一个ref属性，该属性声明的回调函数，会接收input对应的DOM元素，我们将其绑定到this指针上，以便在其他的类函数中，使用。

3.React中三种构建组件的方式？

方法一：无状态函数式组件：纯展示组件，只根据传入的props来负责展示，不涉及到state状态的操作

function HelloComponent(props) {

return <div>Hello {props.name}</div>

}

ReactDOM.render(<HelloComponent name="yourName" />, mountNode)

特点一：组件不会被实例化，整体性能得到提升。

特点二：组件不能访问this对象

特点三：无法访问，生命周期的方法

特点四：不会有副作用

方法二：React.createClass是ES5原生的，JavaScript来实现的React组件

例如：

var Greeting = React.createClass({

getInitialState: function () {

return {

work\_list: []

};

},

render: function () {

return (

<div>

<input type="text" ref="myWork" placeholder="What need to be done?" onKeyUp={this.Enter}/>

<ul>

{

this.state.work\_list.map(function (textValue) {

return <li key={textValue}>{textValue}</li>;

})

}

</ul>

</div>

);

},

Enter: function (event) {

var works = this.state.work\_list;

var work = this.refs.myWork.value;

if (event.keyCode == 13) {

works.push(work);

this.setState({work\_list: works});

this.refs.myWork.value = "";

}

}

});

特点一：React.createClass绑定函数方法导致不必要的性能开销

方法三：React.Component是ES6的形式来创建react的组件的

React.createClass和React.Component的主要区别是：React.createClass创建的组件，每一个成员函数的this，都有React自动绑定。

React.Component创建的组件，其成员函数，不会自动绑定this，需要手动绑定

4.调用 setState 之后发生了什么？

答：调用setState函数之后，react会将传入的参数对象与组件当前的状态合并，然后触发调和过程(即：reconciliation)。经过调和过程，React会以相对高效的方式，根据新的状态，构建React元素树。得到元素树之后，react会自动计算出新的树与老树的节点差异，然后根据差异，对界面进行最小化重新渲染。

5.react diff 原理（常考，大厂必考）

答：(1)把树形结构，按照层级进行分解，只比较同级元素。(2)列表结构的每个单元，添加唯一的key属性，方便比较。(3)react只会匹配相同的class组件。(4)合并操作，调用component的setState方法的时候，react会将其标记为dirty，等到每一个事件都执行结束，react检查所有标记dirty的component重新绘制。(5)选择性子树渲染，shouldComponentUpdate提高diff的性能。

5.为什么建议传递给 setState 的参数是一个 callback 而不是一个对象

答：因为this.props和this.state的更新可能是异步的，不能依赖它们的值，去计算下一个state.

7.除了在构造函数中绑定 this，还有其它方式吗？

答：(1)默认绑定，例如：

function foo() {

console.log(this.a);

}

var a = 1;

foo();

(2)隐式绑定，例如：

function foo() {

console.log(this.a);

}

var obj = {

a: 2,

foo: foo

}

var a = 1;

obj.foo();

(3)硬绑定，使用call, apply或者bind进行硬绑定。

(call与apply都属于Function.prototype的一个方法，其作用都是改变this的指向，不同点是：传递参数的方式不同，call()传递的参数必须列举出来，apply()传递的参数必须是数组)

function foo() {

console.log(this.a);

}

var obj = {

a: 1

}

foo.apply(obj)

(4)使用new进行绑定

function Foo(a) {

this.a = a;

console.log(this.a);

}

var obj = new Foo(2);

console.log(obj.a);

8.setState第二个参数，回调函数的作用

答：该函数会在setState函数调用完成，并且组件开始重新渲染的时候被调用。我们可以用该函数，来监听渲染是否完成。

9.(在构造函数中)调用 super(props) 的目的是什么？

答：super指的是父类构造函数，在调用父构造函数之前，不能在构造函数中使用this

10.简述 flux 思想:

首先，flux最大的特点就是数据的单向流动。flux将一个应用分成四个部分，分别是视图层view，动作层Action，派发器dispatcher，数据层store

主要流程：用户访问view，view会发出用户的action，dispatcher接收到Action，要求store做出相应的更新，store更新后会发出一个”change”事件，view接收到”change”事件后，更新页面。

11.在 React 当中 Element 和 Component 有何区别？

ReactElement是描述屏幕上，所见内容的数据结构，是对UI对象的描述。典型的ReactElement是利用JSX构建的声明式代码片段，然后转化为createElement的调用组合。ReactComponent则是可以接收参数输入，并且返回某个ReactElement的函数或者类。

12.描述事件在 React 中的处理方式。

使用箭头函数，使用组件方法（就是在constructor(props){}进行重新绑定），使用ES7属性初始化的语法（实质：也是利用箭头函数）

13.createElement 和 cloneElement 有什么区别？

React.createElement(

type, // 第一个参数type可以是标签名如div,span或者react组件

[props],

[…children]

)

React.cloneElement(

element, // 传入的参数element是一个React元素，而不是标签名或组件

[props],

[…children]

)

14.如何告诉 React ，它应该编译生产环境版本？

使用webpack的definePlugin方法将NODE\_ENV变量值设为production

15.Controlled Component 与 Uncontrolled Component 之间的区别是什么？

受控组件：引入原生的HTML表单元素时，将所有数据，托管到React组件状态中。

class ControlledForm extends Component {

state = {

username: ''

}

updateUsername = (e) => {

this.setState({

username: e.target.value,

})

}

handleSubmit = () => {}

render () {

return (

<form onSubmit={this.handleSubmit}>

<input

type='text'

value={this.state.username}

onChange={this.updateUsername} />

<button type='submit'>Submit</button>

</form>

)

}

}

非受控组件：DOM元素存放表单数据，并非存放在 React 组件的状态中，我们用 refs 来操控DOM元素

　　class UnControlledForm extends Component {

　　  handleSubmit = () => {

    　　console.log("Input Value: ", this.input.value)

 　　 }

 　　 render () {

  　　  return (

   　　   <form onSubmit={this.handleSubmit}>

      　　  <input

     　　     type='text'

　　          ref={(input) => this.input = input} />

　　        <button type='submit'>Submit</button>

 　　     </form>

　　    )

　　  }

　　}

16.hashRouter和historyRouter的区别？

HashRouter：

　　原理是：URL的hash，不需要由web server支持， #/react/route URL不能被web server读取，在web client这边，window,location.hash被react router解析成example.com/#/react/route

BrowserRouter：

　　原理是：H5的history API，IE9及以下不兼容，需要由web server支持，在web client这边window.location.pathname被react router解析成，example.com/react/route

react组件面试题

1.展示组件(Presentational component)和容器组件(Container component)之间有何不同？

展示组件关心组件看起来是什么。展示专门通过 props 接受数据和回调，并且几乎不会有自身的状态，但当展示组件拥有自身的状态时，通常也只关心 UI 状态而不是数据的状态。(子组件)  
容器组件则更关心组件是如何运作的。容器组件会为展示组件或者其它容器组件提供数据和行为(behavior)，它们会调用 Flux actions，并将其作为回调提供给展示组件。容器组件经常是有状态的，因为它们是(其它组件的)数据源。(父组件)

2.类组件(Class component)和函数式组件(Functional component)之间有何不同？

类组件允许你使用更多额外的功能，如组件自身的状态和生命周期钩子函数，也能使组件直接访问store 并维持状态  
当组件仅接收 props，并将组件自身渲染到页面时，该组件就是一个 ‘无状态组件(stateless component)’，可以使用一个纯函数来创建这样的组件。这种组件也被称为哑组件(dumb components)或展示组件

3.组件中的状态(state)和属性(props)之间有何不同？

State 是一种数据结构，用于组件挂载时，所需数据的默认值。State 可能会随着时间的推移而发生变化，但多数时候，是作为用户事件行为的结果。

Props则是组件的配置。props 由父组件传递给子组件，并且就子组件而言，props 是不可变的。组件不能改变自身的 props，但是可以把其子组件的 props 放在一起，统一管理。Props 不仅仅是数据，回调函数也可以通过 props 传递。

4.何为受控组件(controlled component)？

表单元素，它的值通过 React 的状态来控制，这样的元素就被称为"受控元素"。

5.何为高阶组件(higher order component)？

高阶组件是一个没有负作用的纯函数，且该函数接受一个组件作为参数，并返回一个新的组件。

// 高阶组件

export default (WrappedComponent) => {

class NewComponent extends Component {

constructor() {

super();

this.state = {

username: ''

}

}

componentWillMount() {

let username = localStorage.getItem('username');

this.setState({

username: username

})

}

render() {

return <WrappedComponent username={this.state.username}/>

}

}

return NewComponent

}

// 高阶组件的使用

import React, {Component} from 'react';

import wrapWithUsername from './wrapWithUsername';

class Welcome extends Component {

render() {

return (

<div>welcome {this.props.username}</div>

)

}

}

Welcome = wrapWithUsername(Welcome);

export default Welcome;

6.应该在 React 组件的何处发起 Ajax 请求？

ajax请求放到componentDidMount 函数中执行，主要原因有下：

1.React 下一代调和算法 Fiber 会通过开始或停止渲染的方式优化应用性能，其会影响到 componentWillMount 的触发次数。对于 componentWillMount 这个生命周期函数的调用次数会变得不确定，React 可能会多次频繁调用 componentWillMount。如果我们将 AJAX 请求放到 componentWillMount 函数中，那么显而易见其会被触发多次，自然也就不是好的选择。

2.如果我们将 AJAX 请求放置在生命周期的其他函数中，我们并不能保证请求仅在组件挂载完毕后才会要求响应。如果我们的数据请求在组件挂载之前就完成，并且调用了setState函数将数据添加到组件状态中，对于未挂载的组件则会报错。而在 componentDidMount 函数中进行 AJAX 请求则能有效避免这个问题。

在componentDidMount()函数中发送ajax请求，拿到数据，通过setState()保存在state中，供给组件使用。当组件要卸载时，在componentWillUnmount()函数中，通过this.serverRequest.abort();将还没有完成的ajax请求停止。

7.react中组件传值

8.什么时候在功能组件( Class Component )上使用类组件( Functional Component )？

10.react 组件的划分业务组件技术组件？

11.装饰器

redux面试题

1.redux中间件

2.redux有什么缺点

3.了解 redux 么，说一下 redux 把

react性能比较面试题

1.vue和react的区别

2.react性能优化的方案

3.React 项目用过什么脚手架

4.介绍一下webpack webpack

5.如果你创建了类似于下面的 Twitter 元素，那么它相关的类定义是啥样子的？

6.为什么我们需要使用 React 提供的 Children API 而不是 JavaScript 的 map？

7.装饰器是什么东西？

8.https和http的区别

js面试题

1.简述同步和异步的区别

2.怎么添加、移除、复制、创建、和查找节点

3.实现一个函数clone 可以对Javascript中的五种主要数据类型（Number、string、Object、Array、Boolean）进行复制

4.如何消除一个数组里面重复的元素

5.写一个返回闭包的函数

6.使用递归完成1到100的累加：

function calculate(number) {

if( number === 1 ){

return 1;

}

return number + calculate(number-1);

}

console.log(calculate(100))

7.Javascript有哪几种数据类型

8.如何判断数据类型

9.console.log(1+‘2’)和console.log(1-‘2’)的打印结果

10.Js的事件委托是什么，原理是什么

11.如何改变函数内部的this指针的指向

12.列举几种解决跨域问题的方式，且说明原理

13.谈谈垃圾回收机制的方式及内存管理

14.写一个function ，清除字符串前后的空格

15.js实现继承的方法有哪些

16.判断一个变量是否是数组，有哪些办法

17.let ，const ，var 有什么区别

18.箭头函数与普通函数有什么区别

19.随机取1-10之间的整数

20.new操作符具体干了什么

21.Ajax原理

22.模块化开发怎么做

23.异步加载Js的方式有哪些

24.xml和 json的区别

25.webpack如何实现打包的

26.常见web安全及防护原理

27.用过哪些设计模式

28.为什么要同源限制

29.offsetWidth/offsetHeight,clientWidth/clientHeight与scrollWidth/scrollHeight的区别

30.javascript有哪些方法定义对象

31.说说你对promise的了解

32.谈谈你对AMD、CMD的理解

33.web开发中会话跟踪的方法有哪些

34.介绍js有哪些内置对象？

35.说几条写JavaScript的基本规范？

36.javascript创建对象的几种方式？

37.eval是做什么的？

38.null，undefined 的区别？

null表示没有对象，此处不应该有值，undefined表示此处应该有值，但实际上是没值得

39.[“1”, “2”, “3”].map(parseInt) 答案是多少？

[1, NaN, NaN] // parseInt(string, radix),string为要解析的字符串，radix可以理解为进制，答案所以为[1,NaN,NaN]

40.javascript 代码中的”use strict”;是什么意思 ? 使用它区别是什么？

41.js延迟加载的方式有哪些？

42.defer和async

43.说说严格模式的限制

44.attribute和property的区别是什么？

45.ECMAScript6 怎么写class么，为什么会出现class这种东西?

46.常见兼容性问题

47.函数防抖节流的原理

48.原始类型有哪几种？null是对象吗？

49.为什么console.log(0.2+0.1==0.3) //false

50.说一下JS中类型转换的规则？

51.深拷贝和浅拷贝的区别？如何实现

52.如何判断this？箭头函数的this是什么

53.== 和 ===的区别

54.什么是闭包

55.JavaScript原型，原型链 ? 有什么特点？

56.typeof()和instanceof()的用法区别

57.什么是变量提升

58.all、apply以及bind函数内部实现是怎么样的

59.为什么会出现setTimeout倒计时误差？如何减少

60.谈谈你对JS执行上下文栈和作用域链的理解

61.new的原理是什么？通过new的方式创建对象和通过字面量创建有什么区别？

62.prototype 和 proto 区别是什么？

63.使用ES5实现一个继承？

64.取数组的最大值（ES5、ES6）

65.ES6新的特性有哪些？

66.promise 有几种状态, Promise 有什么优缺点 ?

67.Promise构造函数是同步还是异步执行，then呢 ?promise如何实现then处理 ?

68.Promise和setTimeout的区别 ?

69.如何实现 Promise.all ?

70.如何实现 Promise.finally ?

71.如何判断img加载完成

72.如何阻止冒泡？

event.stopPropagation();

73.如何阻止默认事件？

event.preventDefault();

74.ajax请求时，如何解释json数据

75.json和jsonp的区别?

76.如何用原生js给一个按钮绑定两个onclick事件？

77.拖拽会用到哪些事件

78.document.write和innerHTML的区别

79.jQuery的事件委托方法bind 、live、delegate、on之间有什么区别？

80.浏览器是如何渲染页面的？

81.(document).ready()方法和window.onload有什么区别？

82.jquery中(document).ready()方法和window.onload有什么区别？

83. jquery中(document).ready()方法和window.onload有什么区别？

84.jquery中.get()提交和$.post()提交有区别吗？

85.对前端路由的理解？前后端路由的区别？

86.手写一个类的继承

87.XMLHttpRequest：XMLHttpRequest.readyState;状态码的意思

88.正则表达式常见面试题

89.call和apply的区别

1.给一个连字符串例如：get-element-by-id转化成驼峰形式。

2.匹配二进制数字

3.非零的十进制数字 (有至少一位数字, 但是不能以0开头)

4.匹配一年中的12个月

5.匹配qq号最长为13为

6.匹配常见的固定电话号码

7.匹配ip地址

8.匹配用尖括号括起来的以a开头的字符串

9.分割数字每三个以一个逗号划分

10.判断字符串是否包含数字

11.判断电话号码

12.判断是否符合指定格式

13.判断是否符合USD格式

14.JS实现千位分隔符

15.获取 url 参数

16.验证邮箱

17.验证身份证号码

18.匹配汉字

19.去除首尾的’/’

20.判断日期格式是否符合 '2017-05-11’的形式，简单判断，只判断格式

21.判断日期格式是否符合 '2017-05-11’的形式，严格判断（比较复杂）

22.IPv4地址正则

23.十六进制颜色正则

24.车牌号正则

25.过滤HTML标签

26.密码强度正则，最少6位，包括至少1个大写字母，1个小写字母，1个数字，1个特殊字符

27.URL正则

28.匹配浮点数

浏览器/html/css面试题

1.什么是盒模型

2.行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？行内元素和块级元素有什么区别？

3.简述src和href的区别

4.什么是css Hack

5.什么叫优雅降级和渐进增强

6.px和em的区别

7.HTML5 为什么只写

8.Http的状态码有哪些

9.一次完整的HTTP事务是怎么一个过程

10.HTTPS是如何实现加密

11.浏览器是如何渲染页面的

12.浏览器的内核有哪些？分别有什么代表的浏览器

13.页面导入时，使用link和@import有什么区别

14.如何优化图像，图像格式的区别

15.列举你了解Html5. Css3 新特性

16.可以通过哪些方法优化css3 animation渲染

17.列举几个前端性能方面的优化

18.如何实现同一个浏览器多个标签页之间的通信

19.浏览器的存储技术有哪些

20.css定位方式

21.尽可能多的写出浏览器兼容性问题

22.垂直上下居中的方法

23.响应式布局原理

25.清除浮动的方法

26.http协议和tcp协议

27.刷新页面，js请求一般会有哪些地方有缓存处理

28.如何对网站的文件和资源进行优化

29.你对网页标准和W3C重要性的理解

30.Http和https的区别

31.data-属性的作用

32.如何让Chrome浏览器显示小于12px的文字

33.哪些操作会引起页面回流（Reflow）

34.CSS预处理器的比较less sass

35.如何实现页面每次打开时清除本页缓存

36.什么是Virtual DOM,为何要用Virtual DOM

37.伪元素和伪类的区别

38.http的几种请求方法和区别

39.前端需要注意哪些SEO

40.的title和alt有什么区别

41.从浏览器地址栏输入url到显示页面的步骤

42.如何进行网站性能优化

43.语义化的理解

44.HTML5的离线储存怎么使用，工作原理能不能解释一下？

45.浏览器是怎么对HTML5的离线储存资源进行管理和加载的呢

46.iframe有那些缺点？

47.WEB标准以及W3C标准是什么?

48.Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?

49.HTML全局属性(global attribute)有哪些

50.Canvas和SVG有什么区别？

51.如何在页面上实现一个圆形的可点击区域？

52.网页验证码是干嘛的，是为了解决什么安全问题

53.请描述一下 cookies，sessionStorage 和 localStorage 的区别？

54. CSS选择器有哪些？哪些属性可以继承？

55.CSS优先级算法如何计算？

56.CSS3有哪些新特性？

57.请解释一下CSS3的flexbox（弹性盒布局模型）,以及适用场景？

58.用纯CSS创建一个三角形的原理是什么？

59.常见的兼容性问题？

60.为什么要初始化CSS样式

61.absolute的containing block计算方式跟正常流有什么不同？

62.CSS里的visibility属性有个collapse属性值？在不同浏览器下以后什么区别？

63.display:none与visibility：hidden的区别？

64.position跟display、overflow、float这些特性相互叠加后会怎么样？

65.对BFC规范(块级格式化上下文：block formatting context)的理解？

66.为什么会出现浮动和什么时候需要清除浮动？清除浮动的方式？

67.上下margin重合的问题

68. 设置元素浮动后，该元素的display值是多少？

69.移动端的布局用过媒体查询吗？

70.CSS优化、提高性能的方法有哪些？

71.浏览器是怎样解析CSS选择器的？

72.在网页中的应该使用奇数还是偶数的字体？为什么呢？

73.margin和padding分别适合什么场景使用？

74.元素竖向的百分比设定是相对于容器的高度吗？

75.全屏滚动的原理是什么？用到了CSS的哪些属性？

76.什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的IE？

77. 视差滚动效果？

78.::before 和 :after中双冒号和单冒号有什么区别？解释一下这2个伪元素的作用

79.让页面里的字体变清晰，变细用CSS怎么做？

80. position:fixed;在android下无效怎么处理？

81.如果需要手动写动画，你认为最小时间间隔是多久，为什么？

82.li与li之间有看不见的空白间隔是什么原因引起的？有什么解决办法？

83.display:inline-block 什么时候会显示间隙？

84. 有一个高度自适应的div，里面有两个div，一个高度100px，希望另一个填满剩下的高度

85.png、jpg、gif 这些图片格式解释一下，分别什么时候用。有没有了解过webp？

86.style标签写在body后与body前有什么区别？

87.CSS属性overflow属性定义溢出元素内容区的内容会如何处理?

88.阐述一下CSS Sprites

89. 一行或多行文本超出隐藏

typeScript面试题

46．工作中遇到的问题

1.环境的问题，起不起来

2.打印生成PDF打印的问题

3.调用手机摄像头及上传图片的问题

4.使用ant design pro脚手架打包之后，包太大，用nginx压缩的问题

5.首页加载时间过长，白屏的问题

6.做移动端输入框聚焦，唤醒键盘时，遮住输入框的问题

47．JavaScript 中 call()、apply()、bind() 的用法及不同

var name = '小王', age = 17;

var obj = {

name: '小张',

objAge: this.age,

myFun: function (fm, t) {

console.log(this.name + '年龄' + this.age, '来至' + fm + '去往' + t);

}

}

var db = {

name: '德玛',

age: 99

}

obj.myFun.call(db,'成都','上海');　　　// 德玛 年龄 99 来自 成都去往上海

obj.myFun.apply(db,['成都','上海']); // 德玛 年龄 99 来自 成都去往上海

obj.myFun.bind(db,'成都','上海')(); // 德玛 年龄 99 来自 成都去往上海

obj.myFun.bind(db,['成都','上海'])();　// 德玛 年龄 99 来自 成都, 上海去往 undefined

call()、apply()、bind() 都是用来重定义 this 这个对象的！，上图例子this指向的是db这个对象

call()， apply()， bind()的不同：

1.传参不同

2.call的参数是直接放进去的，中间用逗号','隔开

3.apply的参数必须放在数组里传进去

4.bind()返回的是一个函数，它的传参方式与call()的传参方式一样

47．js如何获取一个月的天数

function days(year,month){

        var dayCount;

        now = new Date(year,month, 0);

        dayCount = now.getDate();

        return dayCount;

}

alert(days(2014,7))

// 获取当前时间的下一个月：

new Date(new Date().setMonth(new Date().getMonth() + 1))

<script>

/\*\*

\* 获取上一个月-第二种

\* @date 格式为yyyy-mm-dd的日期，如：2019-01-25

\* 对日期做处理，获取2019-03-28，29，30，31，都会返回2019-02-28

\*

\*/

function getPreMonth(date) {

var arr = date.split('-');

var year = arr[0]; //获取当前日期的年份

var month = arr[1]; //获取当前日期的月份

var day = arr[2]; //获取当前日期的日

var month2 = parseInt(month) - 1;

if (month2 == 0) {

year = parseInt(year) - 1;

month2 = 12;

}

var days2 = new Date(year, month2, 0);

days2 = days2.getDate();

if (day > days2) {

day = days2;

}

if (month2 < 10) {

month2 = '0' + month2;

}

var t2 = year + '-' + month2 + '-' + day;

return t2;

}

/\*\*

\* 获取下一个月-第二种

\* @date 格式为yyyy-mm-dd的日期，如：2019-01-25

\*

\*/

function getNextMonth(date, nextFlewMonth) {

      var arr = date.split('-');

      var year = arr[0];  // 获取当前日期的年份

      var month = arr[1]; // 获取当前日期的月份

      var currentDay = arr[2];   // 获取当前日期的日

      month = parseInt(month) + nextFlewMonth;

      if (month >= 13) {

        year = parseInt(year) + 1;

        month = month - 12;

      }

      var haveDays = new Date(year, month, 0);

      haveDays = haveDays.getDate();

      if (Number(currentDay) > haveDays) {

          currentDay = haveDays;

      }

      if (month < 10) {

          month = '0' + month;

      }

      var changeTime = '' + year + month + currentDay;

      return changeTime;

}

console.log(getNextMonth('2018-11-01', 3))

</script>