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#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

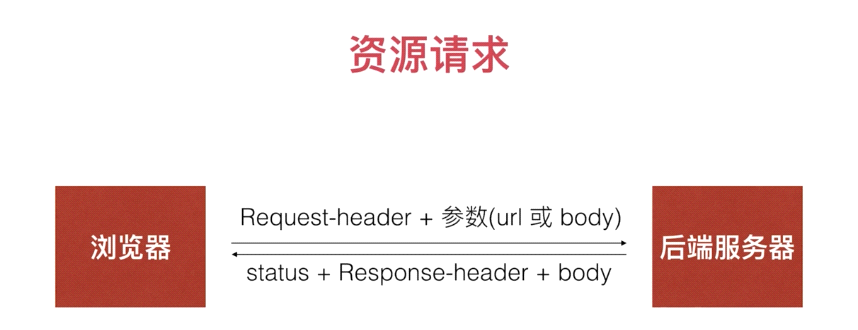
productId=1是请求参数

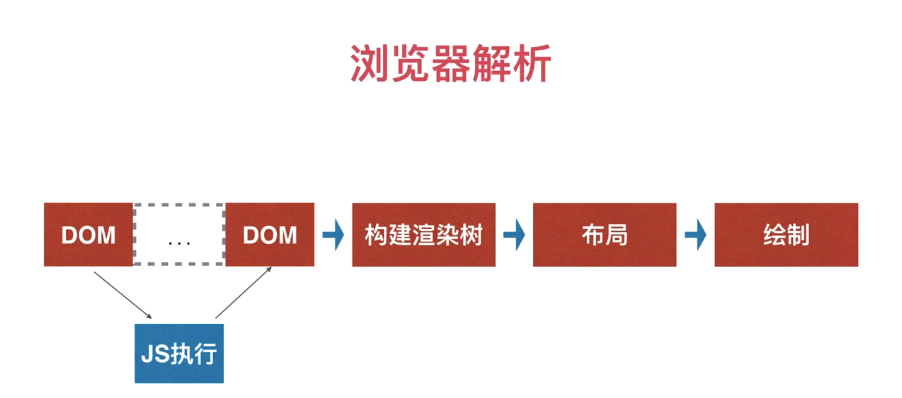
#titlehach 前端页面的锚点



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元素本身就带有的固有属性，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

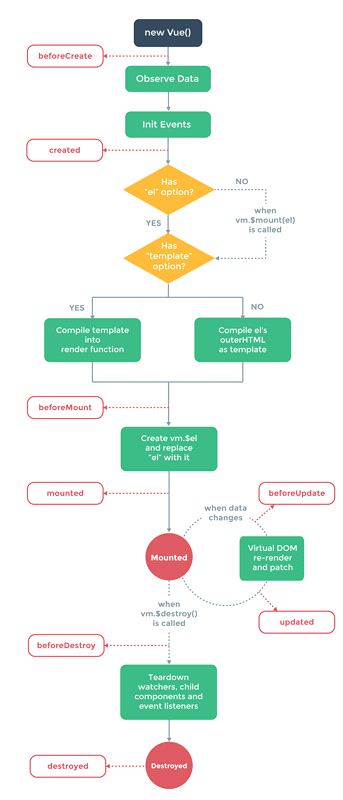
二．vue路由跳转参数的三种方式

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

懒加载定义：也叫延迟加载，即在需要的时候，才进行加载。

https://www.jianshu.com/p/b323dadfeda9

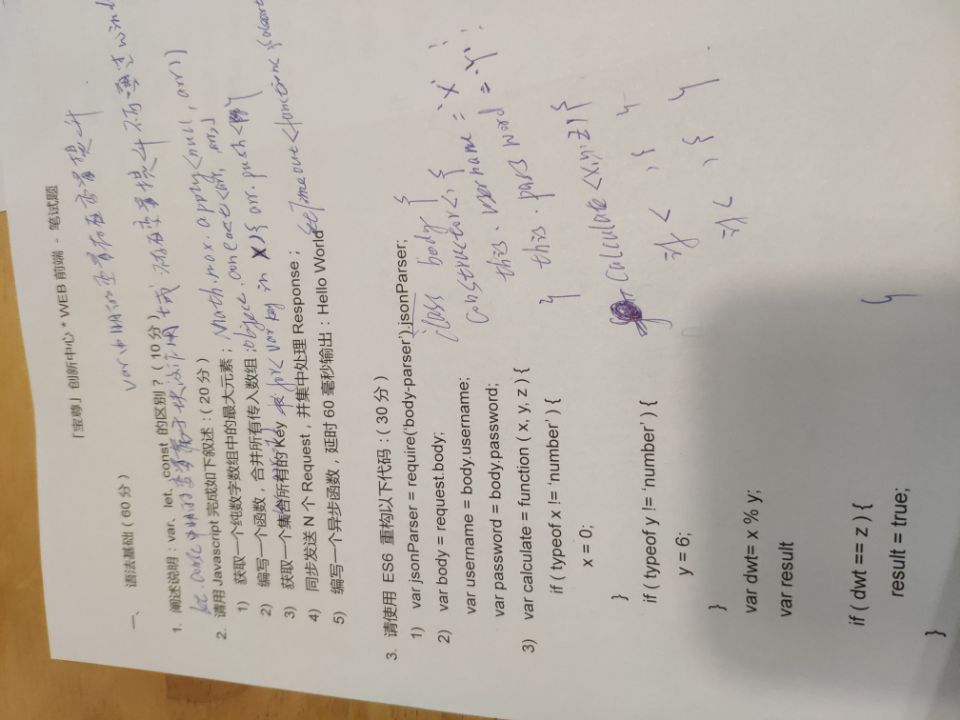
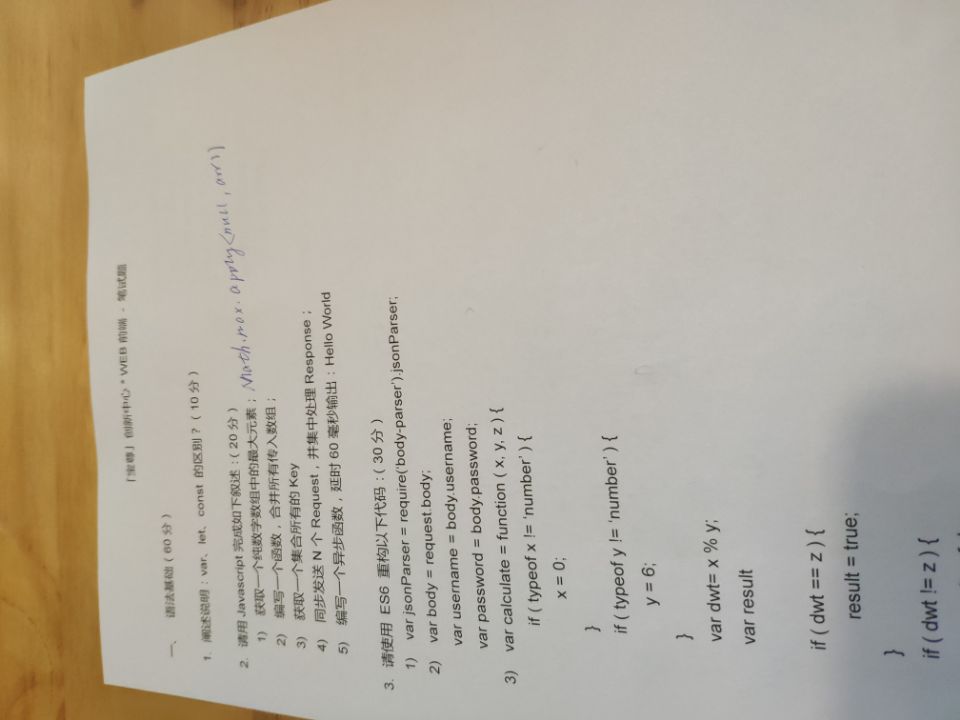
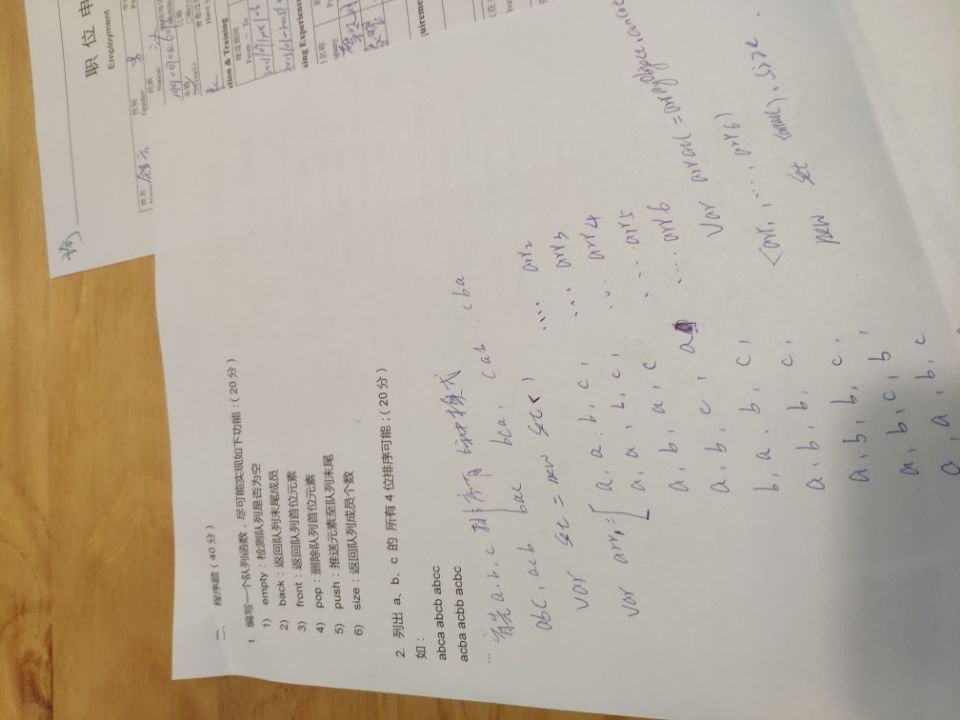
八．如何解决打包过大的问题？

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

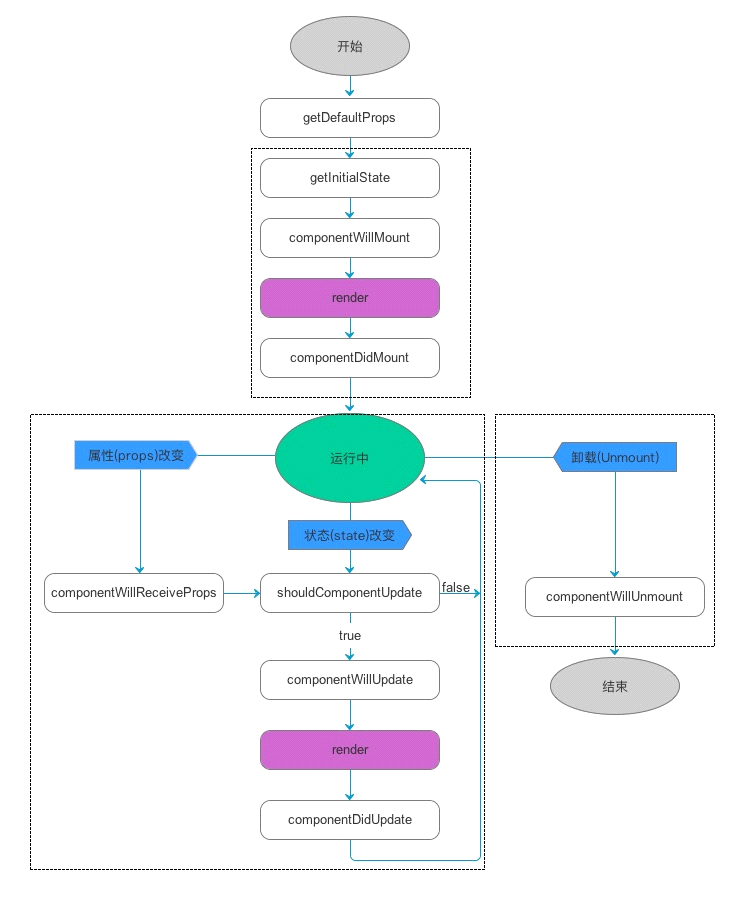
clear方法，表示清除集合中所有的元素，没有任何返回值。如:set.clear()则将元素全部清除。

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，字符串模板，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>

)

}

}

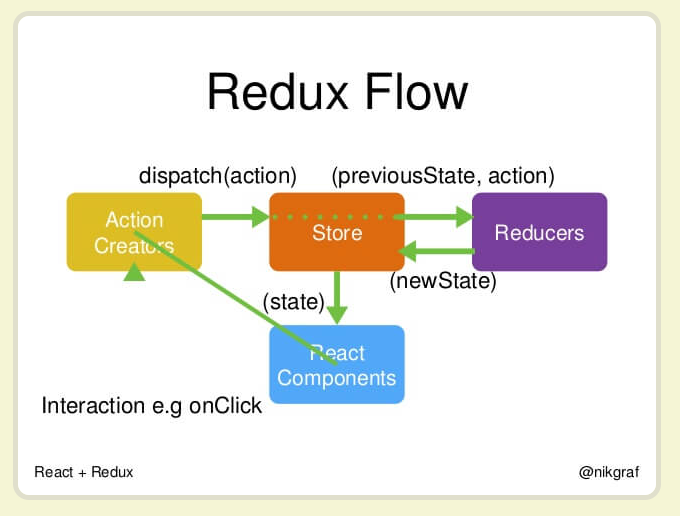
Redux的原理(背一遍)

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。

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**实现过程.**

我们已经知道实现数据的双向绑定，

// 首先要对数据进行劫持监听，

// 所以我们需要设置一个监听器Observer，

// 用来监听所有属性。

// 如果属性变化了，就需要告诉订阅者，Watcher看是否需要更新。

因为订阅者是有很多个，所以我们需要有一个消息订阅器Dep，来专门收集这些订阅者，然后在监听器Observer和订阅者Watcher之间进行统一管理的。

接着，

我们还需要有一个指令解析器Compile，

对每个节点元素进行扫描和解析，

将相关指令对应初始化成一个订阅者Watcher，

并替换模板数据或者绑定相应的函数，此时，当订阅者Watcher接收到相应属性的变化。就会执行对应的更新函数，从而更新视图。

因此接下来，我们执行以下3个步骤，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。

流程图如下：