H5新特性

Header、nav、main、article、section、aside、footer、hgroup、figure、figcaption、time、datalist、details、mark、progress、meter

ES5:

Document事件 document.addEventListener('DOMContentLoaded',function(){})

use strict严格模式

JSON.parse(text)：将json字符串转换成对象/数组

获取元素节点 querySelector(selector) querySelectorAll(selector)

bind() 用于将当前函数和指定对象绑定(改变this指向)，返回一个新的函数

classList：js 的方法

data自定义属性 datase

ES6：

变量声明： let const

解构赋值Destructuring var [a,b,c] = [1,2,3] var {a,b}={a:’html’,b:’css’}

字符串方法

include判断是否包含某个字符，返回布尔值

startsWith/endsWith 是否以某一字符开头/结尾

repeat(n) 得到字符串重复n次后的结果，n可以为小数，但不能为负数

for..of 遍历数组

Object.assign(obj1,obj2,…objN); Object.assign({a:1},{b:2},{b:4,c:3}); //{a:1,b:4,c:3}

箭头函数arrow function

去重数组 let items = new Set([1, 2, 3, 4, 5, 5, 5, 5]);

移动端布局方式：

1. 响应式布局，利用@media进行断点，在每个断点中编写css 百分比布局、vw|vh
2. Rem布局，会根据html的font-size大小进行转换
3. 弹性盒布局：是一种当页面需要适应不同的屏幕大小以及设备类型时确保元素拥有恰当的行为的布局方式

闭包：

1. 函数嵌套函数（闭包诞生）
2. 全局和局部作用域 正常情况下局部是能访问全局，但是全局是不能访问局部，因为闭包存在的，使全局能访问局部
3. 因为全局能访问局部，局部变量会垃圾回收，全局不会垃圾回收，这个局部可以升级为全局，那局部变量就不消失了，所以垃圾回收机制失效
4. 什么是垃圾回收机制，什么是全局和局部

Promise：

是异步编程的一种解决方案，如果回调嵌套太多就会形成回调地狱，promise就是为了解决回调地狱的，

var p1= new Promise(function(resolve, reject){

window.setTimeout(function(){

reject('error');

}, 2000)

})

p1.then(function(res){

console.log('ok -> ' + res)

}).catch(function(error){

console.log('error -> ' + error)

})

async await 基于es6 promise的基础上，作用跟promise一样

let start = async () => {

try{

let result = await sleep3();

console.log('asycn await => ' + result);

}catch(error){

console.log('asycn await => ' + error);

}

}

let start1 = () => {

sleep3().then((res) => {

console.log(res)

})

}

start();

面向对象：

面向对象是一个编程的思想，编程是为了解决问题，编程思想就是在解决问题的时候所用到的思想。

面向对象的特点：抽象、封装、继承、多态

New：

1. new会创建新对象
2. 将对象的\_\_proto\_\_属性指向函数的prototype属性
3. 在执行函数的时候把函数中的this指向这个对象
4. 返回这个对象

原型：

原型是构造函数的一个属性，值是一个对象，并且原型上的属性和方法会被构造器的实例共享

原型链：

构造器实例的\_\_proto\_\_属性指向的是构造器原型，所以构造器原型上的属性和方法能被实例访问到，假设A构造器的原型是B构造器的实例，B构造的原型是C构造器的实例，这样的话实例之间就形成一条由\_\_proto\_\_属性链接的原型链

在原型链低端的实例可以使用原型链高端的属性和方法

bind、call、apply的区别：

都是改变this的的指向，bind不会执行函数，call和apply会自动执行函数，call和apply的区别的传参的方式不一样，call()在第一个参数之后的 后续所有参数就是传入该函数的值，apply() 只有两个参数，第一个是对象，第二个是数组，这个数组就是该函数的参数。

前端性能优化：

1、压缩代码 2、制作精灵图 3、利用cdn内容分发网络 4、分离css和js 5、css放头部js放底部 6、避免回流和减少重绘 7、减少http请求 8、将内存存储多个服务器 9、使用懒加载、无线加载 10、减少DOM的操作 11、图片较小的使用base64编码格式 12、减少cookie的使用 13、使用Ajax缓存 14、浏览器缓存机制

Vue：

Vue 不支持 IE8 及以下版本

Vue是一个MVVM的js框架，即model、view、viewModel三层，双向绑定

绑定原理：

配置在data的数据，实例化组件时是会为每一个组件挂在相应的data数据，挂载的时候会通过objectDefineProperty进行处理，是每个数据都拥有getter和setter方法，数据改变就会执行setter方法，触发实例对应的watcher，然后创建新的虚拟DOM，更新视图

虚拟DOM：

通过在js中实现一套基于js的DOM模型树，需要操作DOM的时候会操作虚拟DOM，在对比虚拟DOM的不同进行真实DOM的更新，减少DOM的操作

实现步骤：

创建与真实DOM完全对应的虚拟DOM，当数据更新，根据更新后的数据再次创建新的虚拟DOM，利用diff算法将新的虚拟DOM和旧的虚拟DOM进行对比，，在视图中更新新的真实DOM，一些时间会随着虚拟DOM的消失而解绑，不会造成内存泄漏

配置属性：el，data，methods，computed，watch（immediate,deep），component，filter

Vue指令：v-for（加上key提高效率）、v-model、v-if、v-show、v-else-if和v-else、v-bind等同于：、v-on等同于@、v-cloak、v-html、v-text、v-pre、v-once

mixin： 引入方法 mixins:[...]

生命周期：

beforeCreate、created、beforeMount、mounted、beforeUpdate、updated、beforeDestroy、destroyed

created获取数据，轮播图Vue.nextTick中执行轮播图代码 config里配置代理服务器

vue-router：

vue-router会去监听浏览器地址栏的变化，当地址栏变化之后，进行对应组件的切换

创建一个router文件夹里面配置路由表、路由模式等等redirect重定向到not-found

将路由实例注入到 根实例中

配置路由出口，其实就是利用router-view指定路由在哪里切换

$route为当前router跳转对象里面可以获取name、path、query、params等

$router为VueRouter实例，想要导航到不同URL，则使用$router.push 、go、replate方法

router-link上的active-class可以指定在路由切换的时候切换哪个类名，exact严格匹配模式

to记录跳转路径，replate不记录 tag制定router-link为什么标签

路由传参：

路由的参数分为两种，一种是路径参数，需要先在路由表中配置，而且必须传

/detail/:id，接收参数的时候利用this.$route.params来接收

第二种参数为search参数，不需要配置也不一定非得传，依靠的是querystring字符串利用this.$route.query来接收

Vuex：状态管理工具

store中又可以分为多个小的部分：state，getters，actions，mutations

state，是一个纯对象，存储需要管理的状态，在computed里接受

mutations, 更改state只有唯一的一种方式！！！！利用commit的方法来进行执行

actions， 注意异步操作只能放在actions，利用dispatch的方法来进行执行

getters， 有的时候需要根据一条现有的状态派生新状态，与computed相识

view里还可以利用mapState、mapMutations、mapActions、mapGetters方便调用

Vue守卫：全局路由守卫beforeEnter，beforeLeave，beforeRouteEnter等

let need\_user\_state = [ 'mine' ]

router.beforeEach((to, from ,next) => {

let need\_us = need\_user\_state.some(name => to.name === name)

if ( need\_us && !store.state.commons.user\_state ) {

next('/login');return false;

}

next();

})

React：

生命周期：

getDefaultProps，getInitialState，componentWillMount，render，componentDidMount

componentWillreceivePorps，shouldComponentUpdate，componentWillUpdate

componentDidUpdate，componentWillUnMount

解决移动端点击延时300ms的问题

zepto.js 是一个轻量级的移动端的类库，其实我们在移动端开发中基本不使用jquery

touch.js/hammer.js这样的第三方手势库也很常用

引入fastclick库来解决300ms延迟的问题

点透bug产生的特定场景：

dom元素排列形成了上下层关系，上层元素绑定了tap事件，并且tap之后使其消失（类似于遮罩层），下层元素绑定是click事件（例如列表中的item包裹了a标签），当我们点击上层元素触发上层元素的tap事件之后，上层元素消失，但是这个时候，下层元素的click事件也会被触发，所以经常会出现这样的情况，当我们点击上层遮罩层的时候，页面突然跳转了，就是因为下层的a标签的click事件被触发了

解决方式：

1. 上下都用tap
2. 使用缓动动画，使上层元素被tap之后以动画的形式消失
3. 使用中间层，在上层元素和下层元素之间创建一个透明的层，给这个透明层添加click并且消失，点击上层元素之后，触发了中间的透明层的click事件，透明层消失了，下层元素被保护了
4. 全部使用click，并使用fastclick来解决300ms延迟的问题
5. h5新特性

用于绘画 canvas 元素

用于媒介回放的 video 和 audio 元素。

本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失；

sessionStorage 的数据在浏览器关闭后自动删除。

语意化更好的内容元素，比如 article、footer、header、nav、section

表单控件，calendar、date、time、email、url、search

1. h5语义化标签兼容性问题

使用Google的html5shiv包

1. 本地存储方式

Cookie：

大小：cookie的大小限制在4k。每个域名下cookie的个数现在在20个。

在客户端请求服务器端和服务器响应时，cookie始终被携带在http请求中，即使不需要(造成流量浪费)。这也是限制cookie大小的原因。

客户端可以通过document.cookie操作cookie，并不安全。

cookie可以设置过期时间、路径、域和httpOnly等字段。如果设置了过期时间，cookie会保存在硬盘里，知道到了设定的过期时间才会失效。若未设置过期时间，在浏览器窗口关闭时，cookie就失效了。路径和域两个字段限制了cookie的作用范围。httpOnly设置为true，则js不能通过document.cookie操作cookie。

localStorage：

对比着来，localStorage能保存更大的数据，标准浏览器是5Mb。

localStorage保存在客户端，不随着请求发送给服务器，避免了流量的浪费。

客户端可以通过：setItem、getItem方法访问localStorage。

并且，localStorage没有过期时间，如果不手动清除，数据就永远不会过期，一直保存在浏览器当中。

存储的信息在同一域中是共享的。

1. css3新特性：

CSS3实现圆角（border-radius），阴影（box-shadow）

对文字加特效（text-shadow、），线性渐变（gradient），旋转（transform）

transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);// 旋转,缩放,定位,倾斜

增加了更多的CSS选择器 多背景 rgba

媒体查询，多栏布局

border-image

1. 项目开发流程：

第一、客户提出网站建设的需求，第二、制定网站建设方案并与客户进行沟通，第三、设计方案达成共识，预付款项，第四、网站建设初稿，敲定细节，第五、网站建设完成，进行验收

6．Vue计算属性：

每一个计算属性都包含一个getter 和一个setter

计算属性是基于它们的依赖进行缓存的

7．http有几种请求方式

1、opions 返回服务器针对特定资源所支持的HTML请求方法 或web服务器发送\*测试服务器功能（允许客户端查看服务器性能）

2、Get 向特定资源发出请求（请求指定页面信息，并返回实体主体）

3、Post 向指定资源提交数据进行处理请求（提交表单、上传文件），又可能导致新的资源的建立或原有资源的修改

4、Put 向指定资源位置上上传其最新内容（从客户端向服务器传送的数据取代指定文档的内容）

5、Head 与服务器索与get请求一致的相应，响应体不会返回，获取包含在小消息头中的原信息（与get请求类似，返回的响应中没有具体内容，用于获取报头）

6、Delete 请求服务器删除request-URL所标示的资源\*（请求服务器删除页面）

7、Trace 回显服务器收到的请求，用于测试和诊断

8、Connect HTTP/1.1协议中能够将连接改为管道方式的代理服务器

8．GET和POST的区别：

1、get是从服务器上获取数据，post是向服务器传送数据。

2、在客户端， get方式在通过URL提交数据，数据在URL中可以看到；post方式，数据放置在HTML HEADER内提交

3、对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。

4、get方式提交的数据最多只能有1024字节，而post则没有此限制

5、安全性问题。正如在2中提到，使用get的时候，参数会显示在地址栏上，而 post 不会。所以，如果这些数据是中文数据而且是非敏感数据，那么使用 get ；如果用户输入的数据不是中文字符而且包含敏感数据，那么还是使用 post为好

8．Git常用命令总结：

git init，git clone，git status，git add，git commit，git branch，git checkout，git merge，git pull，git push

9．Ajax原生请求：

var ajax = new XMLHttpRequest();

ajax.open('get','getStar.php?starName='+name);

ajax.send();

ajax.onreadystatechange = function () {

if (ajax.readyState==4 &&ajax.status==200) {

　　console.log(ajax.responseText);//输入相应的内容

}

}

10．常见http请求的状态码：

404 - 请求的网页不存在

503 - 服务不可用

304 （未修改） 自从上次请求后，请求的网页未修改过。 服务器返回此响应时，不会返回网页内容。

10．浏览器的渲染

解析html以构建dom树 -> 构建render树 -> 布局render树 -> 绘制render树

11．Vue和React的区别

Vue在渲染过程中，会跟踪每一个组件的依赖关系，不需要重新渲染整个组件树。

React每当应用的状态被改变时，全部子组件都会重新渲染。这可以通过shouldComponentUpdate这个生命周期方法来进行控制。

vue-cli则有模板列表可选

Create React App逼迫你使用Webpack和Babel