重点 js 高级程序设计 3 - 7 章，10 - 13 章，21 章

重点图解 http，极客时间的趣谈网络协议，博客

css 权威指南 2、3、7、8、10 章，w3c，博客

重点 ES6 标准入门 2、3、7、11、14、16、18、19、22、23

mvvm 和 mvc 的区别，双向绑定原理，虚拟 dom、diff 算法，spa 原理。

深入浅出 nodejs

JavaScript:

说你所知道的全局对象:

全局对象就是Global对象，所有在全局作用域中定义的属性和函数都是Global对象的属性。诸如:isNaN(),isFinite(),parseInt(),parseFloat(),encodeURI(),encodeURIComponent,decodeURI(),decodeURIComponent(),eval()。

ECMAScript没有指出如何直接访问Global对象，但是web浏览器把这个全局对象当作window对象一部分实现。

获取Global对象的代码:var global = function(){return this;}();

另一个全局对象就是Math对象。常用方法:Math.min(),Math.max(),Math.ceil(),Math.floor(),Math.round(),Math.random()(生成一个随机数的函数：function selectFrom(lowerVlaue,upperValue){var choices = upperValue-lowerValue;return Math.floor(Math.random\*choices+lowerValue);})

其他方法如：Math.abs(),Math.pow(num,power),Math.sqrt(num)

判断数组的方法 :

Value instanceof Array;

Instanceof的实现原理就是左操作数的原型链上是否有包含右操作数构造函数的prototype属性

Array.isArray(value);

Value.constructor == Array

Object.protype.toString.apply(value);

Object.prototype.toString.apply([1,2]) //"[object Array]"

Object.prototype.toString.apply('str') //"[object String]"

Object.prototype.toString.apply(1) //"[object Number]"

Object.prototype.toString.apply(null) //"[object Null]"

Object.prototype.toString.apply() //"[object Undefined]"

Object.prototype.toString.apply(function(){}) //"[object Function]"

Object.prototype.toString.apply(true) //"[object Boolean]"

Object.prototype.toString.apply(new Object()) //"[object Object]"

typeof都有什么返回值:

‘string number undefined ‘boolean’ object function

"undefined"==undefined并根据这个问题扩展

False;

Undefined和null都不会进行类型转换，而“undefined”是一个String,undefined数据类型是undefined所以返回false;

Alert(null>=0)返回值

True.进行比较操作时，都是朝着数字转换，Number(null)=0,所以返回true

闭包

Js可以使用内部函数，就是函数的函数声明或函数表达式在另一个函数的函数体内，这些内部函数可以访问包含的内部函数的所有属性。当其中这样一个内部函数在包含的外部函数外调用时，形成闭包。

常见的几种使用方式：

1:

var Circle = function() {

var obj = new Object();

obj.PI = 3.14159;

obj.area = function( r ) {

return this.PI \* r \* r;

}

return obj;

}

var c = new Circle();

alert( c.area( 1.0 ) );

2:

var Circle={

"PI":3.14159,

"area":function(r){

return this.PI \* r \* r;

}

};

alert( Circle.area(1.0) );

3:

var Circle = new Object();

Circle.PI = 3.14159;

Circle.Area = function( r ) {

return this.PI \* r \* r;

}

alert( Circle.Area( 1.0 ) );

闭包的用途：

1. 匿名自执行函数

var data= {

table : [],

tree : {}

};

(function(dm){

for(var i = 0; i < dm.table.rows; i++){

var row = dm.table.rows[i];

for(var j = 0; j < row.cells; i++){

drawCell(i, j);

}

}

})(data);

创建了一个匿名函数并立即执行，函数执行完之后立即释放资源，外部无法引用它内部变量，不会污染全局变量。（污染全局变量(临时变量加入到全局变量中）会导致别的函数可能误用这些变量；造成全局对象过于庞大，影响访问速度(因为作用域链的原因))

1. 结果缓存。设想有一个处理过程很耗时的函数对象，每次调用都会花费很长时间，那么我们可以将计算的值值缓存起来，当调用函数时，先从缓存中找，找到直接返回即可，若找不到再进行计算，然后更新缓存并返回计算值。闭包之所以可以实现这点是因为它不会释放外部引用，所以函数内部值可以保留。

如下例子：

var data= {

table : [],

tree : {}

};

(function(dm){

for(var i = 0; i < dm.table.rows; i++){

var row = dm.table.rows[i];

for(var j = 0; j < row.cells; i++){

drawCell(i, j);

}

}

})(data);

1. 封装。实现私有方法和私有变量。

如下：

var person = function(){

//变量作用域为函数内部，外部无法访问

var name = "default";

return {

getName : function(){

return name;

},

setName : function(newName){

name = newName;

}

}

}();

print(person.name);//直接访问，结果为undefined

print(person.getName());

person.setName("abruzzi");

print(person.getName());

得到结果如下：

undefined

default

Abruzzi

1. 在类的继承中可以用到。通过原型继承后，给子类添加非静态方法。

var Jack = function(){};

//继承自Person

Jack.prototype = new Person();

//添加私有方法

Jack.prototype.Say = function(){

alert("Hello,my name is Jack");

};

闭包的特点：

1. 内部函数，函数申明或函数表达式在另一个函数函数体内
2. 内部函数可以访问包含的外部函数的所有属性
3. 即使外部函数被回收了，内部函数对外部的引用也不会从内存释放

• this 绑定（说了默认绑定，隐式绑定，显式绑定，new 绑定）浅析 Javascript 中 this 的指向 ( 箭头函数 )

1. 默认绑定:非严格模式下会绑定到window,严格模式下为undefined
2. 隐式绑定:当作为对象方法调用是，会绑定到对象。即有一个对象有一个函数，然后在该函数内使用了this,那么该对象调用时，this绑定的就是这个对象
3. 显示绑定：call,apply,bind
4. New绑定:当创建一个新对象时，将this绑定到新对象

• 函数柯里化（这里不是特别会，他讲给我听了）sum(4)()()(3) 结果等于 7 这样

原型链:

每个新函数都有一个prototype属性，该属性是一个指针，指向一个特定类型的原型对象，该原型对象包含特定类型的所有实例所共享的所有方法和属性。原型对象有一个constructor属性，是一个指向该prototype所在构造函数的指针。每个实例对象都有一个[[Prototype]]指针，指向该实例对象对应的原型对象。当让一个对象的prototype赋值为另外一个对象实例时，该对象的prototype就包含了一个可以指向另外一个对象的原型对象的[[Prototype]]指针。如果另一个原型又是另外一个对象的实例，那么该上述关系依然成立，如此层层递进，最终形成实例与原型的链条，就是原型链。

原型继承：就是让子类的原型对象赋值为父类的实例。

即Subtype.prototype = new Suptype();

这样所有父类的实例属性和方法就成了子类的原型属性和原型方法，同时子类又可以通过原型对象包含的[[prototype]]指针获取父类的原型属性和原型方法。

Var instance = new Subtype();

此时instance.constructor == Suptype,因为Subtype.prototype指向了Suptype.prototype,Suptype.prototype.constructor指向的是Suptype，所以instance.constructor现在指向了Suptype.

而且给Subtype.prototype添加私有方法或者重写Suptype.prototype的方法时，一定要写在Subtype.prototype = new Suptype()后面，因为赋值之后，Subtype.prototype的[[Prototype]]指针会指向Suptype.prototype。所以如果添加或者重写方法写在赋值语句之前，那么子类实例会找不到那些方法。

原型继承的缺点：1.原先的父类实例属性变成了子类的原型属性，如果这些属性是引用类型的话，会导致所有子类实例共享同一个引用类型。

2.无法通过子类构造函数想父类构造函数传递参数。

借用构造函数继承：

在子类的构造函数中调用父类的构造函数.如

function Subtype(name){Suptype.apply(this,name);this.age = 25;}

这样做的话既可以向父类的构造函数传递参数了，而且父类的实例属性会变成子类的实例属性。

缺点：1.无法实现函数复用。方法都在构造函数中定义了，那么每次新建一个子类实例都会将所有函数创建一次。

2.无法访问父类原型属性和方法。

组合继承：

在子类构造函数中调用父类构造函数，同时让子类原型对象赋值为父类实例，再令子类原型对象构造函数赋值为子类构造函数(因为将子类原型对象赋值为父类实例后，子类实例的constructor为父类构造函数，如果不重新赋值为子类构造函数就无法在子类构造函数调用父类构造函数了)。即

Function Subtype(name){Suptype.call(this,name);}

Subtype.prototype = new Suptype();

Subtype.prototype.constructor = Subtype;

缺点:当新建一个子类实例时，Suptype()执行了两次，一次在赋值给子类原型对象时，一次在子类的构造函数中。如果父类有一个实例属性如name,name在子类的原型对象上有一个name属性，同时在子类的实例属性中也有一个name属性(因为子类构造函数调用了父类构造函数)，子类实例的name属性屏蔽了子类原型的name属性。

寄生组合继承:

Function Subtype(){Suptype.call(this);}

Var prototype = Object.create(Suptype.prototype);

Prototype.constructor = Subtype;

Subtype.prototype = prototype;

这样就只调用了一次Suptype(),避免了在Subtype.prototype上创建不必要的、多余的属性。

class

Clone,copy

JavaScript 的垃圾回收机制？

垃圾收集器按照固定的时间间隔周期性的清除那些不再使用的变量，释放它们占用的内存。

主要有两种方式标记清除和引用清除。目前主要使用标记清除法。

标记清除就是垃圾收集器给所有在内存中的变量都加上标记，然后将执行环境中的变量和被环境中的变量引用的变量的标记清除，然后到了执行时间时，将带有标记的变量清除，释放它们占用的内存。

引用清除：就是记录每个值被引用的次数。当声明一个变量并将一个引用类型的值赋给该变量时，这个值的引用次数加一，当同一个值又被赋给另一个变量时，该值的引用次数又加一，当某个包含对该值引用的变量被赋给另外一个值时，这个值的引用次数减一，最后到了执行时间时，引用次数为一的值将被清除，释放占用的内存。

严重问题：循环引用；如果对象A中包含了一个指向对象B的指针，对象B包含了一个指向对象A的指针，那么这两个对象的引用次数永远不会为0.

事件绑定的多种方式，事件监听addEventListener的第三个参数是什么，取值各有什么意思？

事件：用户或者浏览器进行的操作如click、mouseover、keydown、load等。

事件处理程序：响应某个事件的函数。

DOM0级事件处理程序:

element.onclick = function(event){alert(event.type);

删除事件处理程序：element.onclick = null;

DOM2级事件处理程序:

Element.addEventListener(“click”,function(event){alert(event.type);},false);

好处是可以给同一个事件添加多个事件处理程序，按添加先后顺序执行。

第三个参数为true表示在捕获阶段(event.eventPhase=1)执行处理程序，为false表示在冒泡阶段(event.eventPhase=3)执行处理程序.

一般都是在冒泡阶段执行处理程序。

删除事件处理程序：element.removeEventListener(“click”,functionName,false);

阻止事件冒泡   阻止事件默认方式 事件代理

DOM事件流：捕获阶段(eventPhase=1)🡪处于目标阶段(eventPhase=2)🡪冒泡阶段(eventPhase=3)

Event对象有很多属性如bubbles(boolean事件是否冒泡),cancelable(boolean事件是否可以取消默认行为),currentTarget(正在处理事件的事件处理程序所在的元素 == this)，target(目标元素，即事件发生的具体元素),preventDefault(function,阻止事件的默认行为，在cancelable值为true时)，stopPropagation(function取消事件的进一步捕获或冒泡，当bubbles为true时)

事件代理(事件委托)：当页面中有多个元素绑定了同种类型的事件处理程序时，可以在尽量最高级别的元素上添加事件处理程序。这样当时间冒泡到了最高级别的元素时，再根据元素id(通过event或者window.event（IE浏览器）可以获取到target或者event.srcElement（IE浏览器），再通过target或者event.srcElement这个对象就可以获取元素的id属性了)就可以做出相应处理。  
6.7.有一段文字，里面有电话号码，电话号码是连续数字，需要把电话号码都替换成\*，说出多种做法(String.replace;for循环)

有一个2n长的数组，现在要分成n个2长度的数组，把每个数组的最大值相加，怎么给出最优解

    我说的是先排序，让下标0和下标1 成一组，下标2和下标3成一组

   面试官问为什么这么

原型链，*proto(就是[[Prototype]]* 和 prototype 的区别

attribute 和 property 的区别？

Attribute是特性，property是属性。属性分为数据属性和访问器属性。

数据属性有4个特性来描述它的行为，分别是Configurable(能否delete属性，能否修改属性的特性，能否将数据属性改为访问器属性)、Enumerable(能否通过for-in遍历,用户自定义的属性默认为true,constructor、isPrototypeOf、propertyIsEnumerable、toString、toLocalString、valueOf默认为false)、Writable(能否修改属性的值,默认为true)、Value（这个属性的数据值，默认为undefined）;

访问器属性有4个特性来描述它的行为，分别是Configurable、Enumerable、Get(在读取属性时调用，默认为undefined)、Set(在写入属性时调用，默认为undefined)

正则对象test方法和exec方法的不同，分别返回什么？

Test方法返回的是一个boolean值，模式与参数匹配返回true,不匹配就返回false.

Exec方法返回的是一个数组，另外该数组还有两个特殊属性index,input，index是匹配项在模式中的索引，input是应用正则表达式的字符串，数组第一项是模式的第一个匹配项，其他项是与模式中捕获组(正则表达式中用括号括起来的部分)匹配的字符串。

正则 ？ 的用法有哪些（正向预查和 0 次或一次）?：有使用过吗，没有

正向预查(只要前面的不要后面的部分)如 RegEx为 window(?=[\d.]+\b)

反向预查(只要后面的部分)RegEx为(?<=CNY:)\d+\.\d+(匹配的字符串必须满足前面的模式，然后又要满足后面的模式，但是前面的模式匹配项不返回)

匿名函数的使用场景，说了回调函数，箭头函数，或者闭包，然后就没继续问了，是答得有问题吗?

回调函数，箭头函数，或者闭包,函数表达式;

主要作用是当匿名函数执行完毕，会释放资源，外部环境不能访问内部变量，避免全局变量污染。

深拷贝和浅拷贝，深拷贝的实现以及应用场景:

简单的来说就是如果B拷贝了A，如果A发生了变化而B没有发生相应变化，则为深拷贝，拷贝了A的所有层级属性，如果B的值随着A变化而变化则为浅拷贝。

深拷贝只针对引用类型。

深拷贝实现:

1:

function deepClone(obj){

let objClone = Array.isArray(obj)?[]:{};

if(obj && typeof obj==="object"){

for(key in obj){

if(obj.hasOwnProperty(key)){

//判断ojb子元素是否为对象，如果是，递归复制

if(obj[key]&&typeof obj[key] ==="object"){

objClone[key] = deepClone(obj[key]);

}else{

//如果不是，简单复制

objClone[key] = obj[key];

}

}

}

}

return objClone;

}

let a=[1,2,3,4],

b=deepClone(a);

a[0]=2;

console.log(a,b);

2.

function deepClone(obj){

let \_obj = JSON.stringify(obj),

objClone = JSON.parse(\_obj);

return objClone

}

let a=[0,1,[2,3],4],

b=deepClone(a);

a[0]=1;

a[2][0]=1;

console.log(a,b);

array 方法有哪些，分成两类的话怎么分

按是否改变原数组分类。

改变：sort reverse pop push shift unshift splice

不改变：concat,join,slice,,every,some,filter,map,forEach,reduce,reduceRight,toString,toLocalString,valueOf

两个数组比较，判断是否有相同元素。说了一个 set，一个 obj

假设里面有对象，只要对象属性相同就算相同该怎么做，说了一个 JSON.stringify 转换为字符串，不过好像不是他要的答案，不知道还有更好的方式吗

双层for循环或者将两个数组去重后arr1.concat(arr2).filter((value,index,arr)=>{arr.indexOf(value)==arr.lastIndexOf(value)}

如果数组里面有对象的话，在上述双层for循环中加个判断元素是否为对象，是对象的话再通过for-in或者Object.keys().sort.toString()比较两个字符串是否相同或者Object.getOwnPropertyNames.sort.toString()比较两个字符串是否相同

 JSON 序列化的缺点（不能复制函数，正则，date 这些对象，还有循环引用的问题）

不能序列化变量、函数、实例对象。

getElementsByClassName 和 querySelectorAll 的区别

getElementsByClassName传入的是一个字符串，返回的是一个动态(对DOM文档的实时查询)的NodeList;(ByTagName,ByName返回也是动态NodeList),

querySelectorAll传入的是一个CSS选择符，返回的是一个静态NodeList;

为啥 js 放在 body 最后（阻塞渲染）

渲染阻塞。当浏览器遇到<script>标签时会停止处理页面，执行javascript代码，然后再继续解析和渲染页面，在使用src属性加载javascript文件时也会这样，浏览器会先下载然后解析javascript然后再解析渲染页面。

跨域，主要让讲了一下 jsonp

Jsonp:在该页面中引入一个js文件，在加载这个js文件时会执行在url中指定的js函数，然后在那个js函数中返回json数据即可。

Window.name:在一个window的生命周期内，该window载入的所有页面都共享一个window.name属性。比如a.html想要获取在另外一个域内data.html的数据。Data.html内使一个js函数设置window.name为json格式字符串。在a.html页面内通过一个隐藏的iframe,设置该iframe的src为data.html,然后a.html在加载该iframe时即可获取到window.name值，同时要设置iframe的src与a.html为同一个域，这样a.html才能读取到window.name值。

Window.name值只能为字符串，且大小为2M.

Document.domains

cors

js 解包拆包，就是字符串调用 String 对象方法的过程，一开始没理解，后面才懂

1. 创建一个String类型实例
2. 在实例上调用指定的方法
3. 销毁这个实例

如var s1 = “hello world”;var s2 = s1.split(“ “);解包操作就相当于:

Var s1 = new String(“hello world”);

Var s2 = s1.split(“ “);

S1 = null;

bind的作用与实现

microtask和macrotask的区别

symbol类型的用途

Symbol本质上作为一种唯一标识符，可用作对象的唯一属性名，这样其他人就不会改写或者复写你设置的属性值。

Let name = Symbol(“id”);

Let name2 = Symbol(“id”);

即使是同一个key创建的Symbol对象也不同。而且设置Symbol对象对属性名的话，for…in也获取不到属性名，只能通过Object.getOwnPropertySymbols()获取到Symbol对象数组。如：

let name = Symbol("name");

let age = Symbol("age");

let person = {

[name] : "jack",

[age] : 25,

}

console.log(Object.getOwnPropertySymbols(person))

[Symbol(name), Symbol(age)]

Symbol有一个全局注册登记方法，根据key无就创建，有就直接返回.

Let name = Symbol.for(“name”); // 创建

Let name = Symbol.for(“name”); // 该key对应的Symbol对象已存在，直接返回

通过Symbol对象获取key

Let key = Symbol.keyFor(name); // 返回“name”

proxy怎么拦截放置对象被外界访问

怎么判断循环列表

const用途，const a = {age: 22}, a.age = 23; 对吗

const声明一个只读变量，一旦声明必须立即初始化。

Const与let一样都只在声明的块级作用域有效，且没有没量提升；

Const只是保证变量指向那个内存地址不能变，并不是值不能改变，所以a.age=23是对的。

用过react和angular吗

后台拿数据是使用什么？原生的ajax对象了解吗？原生的ajax用promise封装一下要怎么做？

new这个操作做了哪些事情？

1. 创建一个新对象new Object();
2. 将构造函数的作用于赋给新对象(this指向这个新对象)
3. 执行构造函数中的代码(执行constructor)
4. 返回新对象

有一个列表，实现点击某一个li标签，弹出元素内容（没答好）

给每一个li添加onclick事件处理程序，当触发时li[i].onclick =()=>alert(li[i].innerHTML)

或者采用事件委托，在ul上添加监听程序，ul.onclick = function(event){

Alert(event.tartget.innerHTML);}

假设现在有一个数组，长度是99位，元素是1-100里边的值，数组是无序的也是不重复的，怎么快速的去找到这个数组和1-100相比缺的元素？（没答上来）

ES6:

es6的 let const

const 是常量，值不可修改，声明时就得初始化，不能在声明后再初始化。和let一样是块级作用域，不能变量提升。Const值为引用类型时只是地址不能变，引用类型的值可以发生变化。

Var可以不初始化，默认为undefined,不是块级作用域，可以变量提升。

es6的箭头函数（多层嵌套时的this）

箭头函数this的绑定是看this所在的函数时定义在哪个对象下，在哪个对象下，this就绑定到哪个对象。

多层嵌套时，看上一层this绑定到哪一个对象，这一层的this也绑定到那一个对象

promise 和 async await

promise 使用场景有哪些。promise 里面 return 一个 string，和在 resolve 一个 string 的区别，这题也答错了，说成了 return string 返回的也是 promise 对象。然后又问在 then 里面 throw 一个 error，怎么捕捉，说了第二个参数和 catch，也说错了，用第二个参数捕捉不到这个 error，让我面完试试，果然捕捉不到

异步编程的题目

promise的编程题

es6的新特性

let const 模板字符串

解构 let nums = [1,2,3];let [x,y,z] = nums;console.log(`${x} ${y} ${z}`);

对象字面量简写:

Let name = “jack”;let age=20;

Let person = {name,age}; console.log(person); // {“name:”jack”,”age”:20}

For…of循环

展开运算符…

箭头函数

Class

class Plane {

//constructor方法虽然在类中,但不是原型上的方法,只是用来生成实例的.

constructor(numEngines) {

this.numEngines = numEngines;

this.enginesActive = false;

}

//原型上的方法, 由所有实例对象共享.

startEngines() {

console.log('starting engines…');

this.enginesActive = true;

}

}

console.log(typeof Plane); //function

就是用原型模式创建对象，属性都放在constructor内，原型共享方法都放在class内(与constructor同级)

如果在方法前面加上static关键字，代表是静态方法，实例不能继承，但是实例可以直接通过类名访问。注意如果static方法内用了this，那么this绑定就是定义时所在的对象。

静态方法可以被继承，子类可以直接通过子类类名访问父类的静态方法。

Super,extend

Generator函数：

Function\* gen(){

Yield 12;

Yield 20;

}

Let t = gen(); // 返回一个iterator(),通过next()遍历

t.next(); // 12

t.next(); // 20

t.next(); // undefined

next()返回的数据结构：{value:””,done:false/true(如果遍历结束为true，否则为false)}

HTML、CSS:

HTML5 webStorage

Webstorage分为localStorage和sessionStorage，他们除了生命周期不一样其他都一样。SessionStorage的声明时在回话存在期间，会话如果结束sessionStorage内容也会销毁。localStorage的内容则会存在本地，始终存在。

SessionStorage大小约为5M,localStorage大小约为20M

Api:

setItem(key,value) getItem(key) removeItem(key) clear() key(index)(根据索引返回key名)

当存储数据发生变化时会出发storage事件

window.addEventListener(“storage”,onStorageChange);

回调函数接受一个event对象作为参数。这个event对象的key属性，保存发生变化的键名。

function onStorageChange(e) {

console.log(e.key);

}

除了key属性，event对象的属性还有三个：

oldValue：更新前的值。如果该键为新增加，则这个属性为null。

newValue：更新后的值。如果该键被删除，则这个属性为null。

url：原始触发storage事件的那个网页的网址。

值得特别注意的是，该事件不在导致数据变化的当前页面触发。如果浏览器同时打开一个域名下面的多个页面，当其中的一个页面改变sessionStorage或localStorage的数据时，其他所有页面的storage事件会被触发，而原始页面并不触发storage事件。可以通过这种机制，实现多个窗口之间的通信。所有浏览器之中，只有IE浏览器除外，它会在所有页面触发storage事件。

强缓存、协商缓存

缓存是代理服务器或者客户端磁盘存储的资源文件副本。

浏览器缓存就是浏览器在本地磁盘存储用户最近请求过的文件，当用户再次访问同一页面时，就可以直接从缓存中读取文件。

浏览器在第一次请求发生后，再次请求时：

浏览器会先获取该资源缓存的header信息，根据其中的expires和cache-control判断是否命中强缓存），若命中则直接从缓存中获取资源，包括缓存的header信息，本次请求不会与服务器进行通信；

如果没有命中强缓存，浏览器会发送请求到服务器，该请求会携带第一次请求返回的有关缓存的header字段信息（Last-Modified/IF-Modified-Since、Etag/IF-None-Match）,由服务器根据请求中的相关header信息来对比结果是否命中协商缓存，若命中，则服务器返回新的响应header信息更新缓存中的对应header信息，但是并不返回资源内容，它会告知浏览器可以直接从缓存获取；否则返回最新的资源内容

Expires

该字段是http1.0时的规范，它的值为一个绝对时间的GMT格式的时间字符串，比如Expires:Mon,18 Oct 2066 23:59:59 GMT。这个时间代表着这个资源的失效时间，在此时间之前，即命中缓存。这种方式有一个明显的缺点，由于失效时间是一个绝对时间，所以当服务器与客户端时间偏差较大时，就会导致缓存混乱。

Cache-Control

Cache-Control是http1.1时出现的header信息，主要是利用该字段的max-age值来进行判断，它是一个相对时间，例如Cache-Control:max-age=3600，代表着资源的有效期是3600秒。cache-control除了该字段外，还有下面几个比较常用的设置值：

no-cache：不使用本地缓存。需要使用缓存协商，先与服务器确认返回的响应是否被更改，如果之前的响应中存在ETag，那么请求的时候会与服务端验证，如果资源未被更改，则可以避免重新下载。

no-store：直接禁止游览器缓存数据，每次用户请求该资源，都会向服务器发送一个请求，每次都会下载完整的资源。

public：可以被所有的用户缓存，包括终端用户和CDN等中间代理服务器。

private：只能被终端用户的浏览器缓存，不允许CDN等中继缓存服务器对其缓存。

Cache-Control与Expires可以在服务端配置同时启用，同时启用的时候Cache-Control优先级高。

Event loop（宏任务微任务）

参考：

<https://blog.csdn.net/lc237423551/article/details/79902106>和[https://juejin.im/post/59e85eebf265da430d571f89#heading-4](https://juejin.im/post/59e85eebf265da430d571f89" \l "heading-4)

怎样用 meta 怎么实现不从缓存中获取资源

<meta http-equiv="Cache-control" content="no-cache">

<meta http-equiv="Cache" content="no-cache">

<meta http-equiv="Pragma" content="no-cache">

<meta http-equiv="Expires" content="0">

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=8">

CSS 中 inline 元素可以设置 padding 和 margin 吗？

Block元素的特点:

1. 换行
2. 行高以及上下边距可以设置
3. 默认宽度为整个容器的100%，若设置宽度则显示设置的宽度
4. 常见：p，div,p,h1...h6,ol,ul,dl,table,address,blockquote,form

Inline元素特点：

1.和其他元素都在一行上（不换行）

2.行高以及顶和底边距不可改变

3.宽度就是它的文字或者图片的宽度，不可改变

4.常用的行内元素：<span><a><label><strong><i><em>等

Inline-block特点：

1. 和其他元素在同一行(不换行)
2. 可以设置宽高、行高，顶端和底部边距
3. 常见：img input

左边固定，右边自适应的两列布局，右面的 DOM 顺序在左边前面，说下实现？

<div class='container'>

<div class='right'></div>

<div class='left'></div>

</div>

1. flex实现

.container  {

   display:flex;

   flex-direction:row-reverse;

}

.left:{

   width:200px;

   height:500px

   background-color:'red'

}

.right:{

   flex:1,

   height:500px;

   background-color:'blue'

}

2.

 .container {

            position: relative; // 这句一定要，定位以祖宗中第一个非static为准进行绝对定位

        }

        .left {

            width: 200px;

            height: 300px;

            position: absolute;

            top: 0;

            left: 0;

            background-color:red;

        }

        .right {

            padding-left: 200px;

            width: 100%;

            height: 300px;

            background:blue

        }

CSS box-sizing 的值有哪些？

Content-box和border-box，默认为content-box;

Content-box是w3c盒模型，元素高度为content;

Border-box是ie盒模型，元素高度为content+padding+border

Header 头 Set-cookie: http-only 是干什么用的？

cookie只能用于http或者https传输，不能通过js操作

前端性能优化？

1. 页面级优化

1.减少http请求数

2.合理的设置缓存。能缓存的越多越好，缓存时间越久越好。

3.资源合并与压缩。如果可以尽可能的将外部脚本、样式合并为一个。另外css、js、image都可以通过相应工具进行压缩。

4.将外部脚本放在body底部，

5.将css放在head中。有的浏览器可能还未下载css文件就开始渲染页面了，导致用户体验很差，放在太靠下的位置又可能会导致浏览器渲染时间推迟。

1. 代码级优化

1.减少dom操作，增加、修改、删除dom或对dom集合进行操作很耗时。

2.减少访问HTMLCollection,因为这是一个动态集合，每次访问都会进行动态查询。

3.少用with。因为使用with会增加作用域链的长度，增加查询作用域链时间。

4.减少作用域链查找，减少闭包使用

5.多个字符串拼接时少用+=，如let s1 = “name”;let s2 = “name2”;let s3 += s1;s3+= s3;6.就可以写成s3 += s1+s2;

6.或者将要拼接的字符串放在数组中，最终通过join(“”)拼接

7.优化循环性能，将全局变量属性放在局部变量中

8.避免循环引用，如一个dom对象有一个方法，在这个方法继续操作这个dom，这样会造成内存泄漏。

CSS BFC 原理？

Bfc就是块级格式化上下文。布局规则是：

1. 内部box会在垂直方向，一个接一个的放置
2. Box垂直方向距离有margin决定，同一个bfc内的两个相邻box的margin不会重叠
3. 每个元素的margin box左边与包含块的border box的左边接触。即使存在浮动也会如此
4. Bfc区域不与float box重叠
5. Bfc里面的元素不会影响外面的元素，外面的元素也不会影响里面的元素
6. 计算bfc高度时浮动元素也参与计算

触发BFC:

1. 根元素
2. Float不为none
3. Overflow不为visible
4. Display为inline-block或flex或table-cell或table-caption或inline-flex
5. Position为absolute或fixed

如果要一个块元素在手机屏幕中垂直居中，css怎么写因为说到fixed所以提到css脱离文档流说一个问一个

Float:使用float脱离文档流，盒子漂浮在原来在文档流中位置上方，其他盒子会无视这个元素，但其他盒子中的文字依然会为这个脱离文档流的盒子让出位置。

Absolute：一直向上找指导一个父类的position不为static(默认为static)时，这个脱离的盒子就以它为相对位置，然后通过left,right,top,bottom进行浮动

Fixed：以浏览器窗口为相对位置，通过left,right,top,bottom进行浮动

Relative：半脱离文档流。以在原文档流中的位置为相对位置，可以通过left,right,top,bottom进行浮动,但还在原来文档流中占据空间。

垂直居中:

父容器position为relative,子块元素position为absolute，top:0,bottom:0

子块元素设置vertical-align为middle

Css3的transform:translate(-50%,-50%)

<https://www.cnblogs.com/chengzp/p/layout.html>

http 和 https 的区别，说了一下传输过程和对称非对称加密

css 伪类是什么（没答出来）<https://www.cnblogs.com/ghost-xyx/p/3763669.html>

:active 将样式添加到被激活元素

:hover 当鼠标浮动到元素上方时，向元素添加样式

:focus 当元素获得焦点时，向元素添加样式

:first-child 将样式添加到元素的第一个子元素

:last-child 将样式添加到元素的最后一个子元素

:before 在某元素之前插入一些内容(在文档流中的前后插入)

:after 在某元素之后插入一些内容(在文档流中的前后插入)

get 和 post 的区别，说了 post 更消耗资源，又问那为什么不用 get 发送数据? <https://www.cnblogs.com/logsharing/p/8448446.html>

1. get通过url传递参数，post通过request body参数
2. get比post更加不安全，因为参数直接暴露在url上，不能传递敏感信息
3. get只支持ASCII字符，post支持任意格式
4. get只支持url编码(application/x-www-form-urlencoded)，post支持任意编码方式
5. get的url长度有限制(不同浏览器长度不同，ie为2k,火狐为65536),post没有长度限制
6. get请求参数会完整保存在历史记录里，post不会
7. get请求会被浏览器主动cache，而post请求不会，除非手动设置
8. get和post都是基于tcp/ip协议，所以发送请求是都会通过tcp数据包传输数据，但是get只产生一个tcp数据包(firefox除外)，post产生两个tcp数据包，所以post更加耗时点。但是在网络环境好的时候两者时间没什么区别，网络环境不好的时候，post方式传递的两个数据包在验证数据完整性上有更大优点。

cookie 和 localstorage 区别，为什么不用 localstorage 验证用户身份（也没答上来）

前端路由和服务端路由

html 语义化，对语义化的理解，有哪些好处，哪些标签，什么用，

<https://www.cnblogs.com/fliu/articles/5244866.html>

html语义化:根据内容的结构化(内容语义化),选择合适的标签(代码语义化)便于开发者阅读和写出更优雅的代码以及便于浏览器爬虫和机器解析。

为什么要语义化:

1. 为了在没有css的情况下页面也能呈现出很好的内容结构、代码结构
2. 提升用户体验
3. 有益于seo优化：爬虫依赖标签来确定上下文和各个关键字权重，有助于爬虫获取更多的信息
4. 方便其他设备解析(如屏幕阅读器、移动设备)以意义的方式来渲染网页
5. 便于团队开发和维护，语义更具可读性

新增哪些标签：

1. header代表网页或section的页眉。作为整个页面或一个内容块的标题
2. footer代表网页或section的页脚。在整个页面或任意section底部，包含该节的基本信息。
3. hgroup代表网页或者section的标题。就是将标题h1,h2..放在hgroup内，如果只有一个标题不需要用hgroup
4. nav代表页面的导航链接区域，用于的主要导航部分
5. section代表页面中的“节”或段
6. article代表在网页中自成一体的内容。如一个帖子，一篇博客文章。
7. aside作为article的附属信息。如广告、作者信息等，最典型的是用作侧边栏

提了 seo，问了我一下 seo，了解的不太多，简单说了下

html 盒模型，怪异盒模型和标准盒模型的应用场景

css 不定宽高垂直居中（就说了两个）定宽高呢

不定宽高:

1. 设置.father{position:relative} .son{position:absolute;top:50%;transform:translateY(-50%);}
2. 使用flex布局,设置 .father{display:flex;align-items:center}
3. 设置.father{position:relative} .son{position:absolute;top:0;bottom:0;margin:auto;}
4. 利用css伪类，设置.father:after{ content: '';display: inline-block;height: 100%;vertical-align: middle;} .son{ display: inline-block;vertical-align: middle;}

定宽高.son{ position: absolute;

left: 50%;

top: 50%;

width:200px;

height:100px;

margin-left:-100px;

margin-top:-50px;}

前端需要哪些技能，了解哪些东西。提到了 node，问了下 node 在前端的用处，commonjs 和服务端渲染（只是知道有这么个东西）

https 具体的传输过程，其实不是问题，他接触的不太多，所以让我讲一下

dns 解析过程，也是接触的不太多，让讲一下

html 语义化的理解

引入 css 有哪几种方式<https://www.jianshu.com/p/e272c98c58d9>

1. 外部样式表引入 <link rel=”stylesheet” type=”text/css” href=”xxx.css/>或者@import url(“xxx.css”)/@import “xxx.css”
2. 内部样式表 在head部分<style></style>
3. 内联样式<p style=”color:red;font-size:20” ></p>
4. 属性样式表(已废弃，避免使用) <img src="a.png" width=100 height=200 >

css 的单位有哪些，你常用的有哪些? <http://www.divcss5.com/css3-style/c33196.shtml>

1.px：绝对单位，页面按精确像素展示

2.em：相对单位，如果自身定义了font-size按自身来计算,如果自身没有设置则基准点为父节点字体的大小一直向上找直到浏览器为止（浏览器默认字体是16px），整个页面内1em不是一个固定的值。

3.rem：相对单位，可理解为”root em”, 相对根节点html的字体大小来计算

css 优先级

!import>内联样式>id选择器样式>类选择器样式>元素选择器样式

css中的水平居中的不同方式

1. inline元素直接用text-align即可
2. block元素如果定宽高margin:0 auto;

如果不定宽高:1)display:inline;text-align:center2)display:table;margin:0 auto;3)设置父元素position:relative子元素position:absolute;top:0;bottom:0;

html中的meta标签？<https://www.cnblogs.com/liangshuang/p/9001261.html>

1. 标题信息(类似http协议头)

<Meta http-equiv="Content-Type" Content="text/html; Charset=gb2312">

<Meta http-equiv="Cache-Control" Content="no-cache ">

<Meta http-equiv="Cache" Content="no-cache ">

<Meta http-equiv="Progma" Content="no-cache ">

<Meta http-equiv="Expires" Content="0">

1. 页面描述信息(方便搜索引擎搜索)

1、Keywords (关键字)

用法：<Meta name="Keywords" Content="关键词1,关键词2，关键词3,关键词4,……">

2、Description (简介)

说明：Description用来告诉搜索引擎你的网站主要内容。

用法：<Meta name="Description" Content="你网页的简述">

4、Author (作者)

用法：<Meta name="Author" Content="张三，abc@sina.com">

5、Copyright (版权)

<Meta name="Copyright" Content="本页版权归Zerospace所有。All Rights Reserved">

media-query用过吗？

媒介查询。有三种媒介类型 print(打印机)screen(电脑、手机、平板屏幕)all(所有媒体设备)

@media only screen and (min-width: 320px) {

#square {

background:red;

}

}

用css画一个三角形（没答上来）

<https://www.cnblogs.com/blosaa/p/3823695.html>

简单点的就是

.element{

Display:block;(如果用的块级元素就不需要设这个值)

Width:0;

Height:0;

Border-width:0 20px 20px 20px;(想要三角形朝那边就让那个值为0)

Border-style:solid;

}

清除浮动的方式:

如果元素设置了浮动，会脱离文档流，后面的元素会自动补位。

1. 设置父元素的overflow:hidden
2. Clear:left/right/both
3. 使用css伪类：

.clearFloat:after{

Display:block;

Visibility:hidden;

Clear:both;

Content:””;

Height:0;

Line-height:0;

}

css盒子模型，哪些元素会自动设置为怪异盒模型，如何设置为怪异盒模型

<https://blog.csdn.net/quguang65265/article/details/50970601>

如果没有声明<!DOCTYPE HTML>在ie6,7,8将会触发怪异模式

设置box-sizing:border-box;

css3新增属性<http://www.w3school.com.cn/css3/css3_border.asp>

border-image,border-radius,box-shadow

transition如：

<style>

div

{

width:100px;

height:100px;

background:yellow;

transition:width 2s;

}

div:hover

{

width:300px;

}

Anination如div

{

animation: myfirst 5s;

}

@keyframes myfirst

{

0% {background:red;}

25% {background:yellow;}

50% {background:blue;}

100% {background:green;}

}

说下http请求

defered 就是作用、特性、方法，然后嘴贱提到了es6的promise

算法：

什么是稳定排序和不稳定排序？哪些排序是稳定的，哪些是不稳定的？

比如说一个数组是[6,2,4,6]里面有两个6,进行升序排序后，如果第一个6仍然在第二个6前面就是稳定排序，如：冒泡排序、归并排序。如果第一个6到了第二个6后面就是不稳定排序，如：快速排序、堆排序、选择排序。

冒泡排序：时间复杂度为：n^2。思路为：

void bubblesort(int a[],int len )

{

for(int i = 0; i<len;++i)

for(int j = i+1; j<len;++j)

if(a[i]>a[j])

swap(a[i],a[j]);

}

遍历数组中每一个元素，将每个元素与后面的所有元素进行比较，如果比他大就交换位置，一次可以确定一个元素位置。

快速排序：时间复杂度为：nlogn。思路为：

先选取一个基准值，一般选数组第一个元素，将比基准值小的放在数组左边，比基准值大的放在数组右边，然后分别在左右两边进行递归操作。

void quicksort(int a[], int left, int right){

int first = left;

int last = right;

int key = a[left];

if(left>=right) return ;

while(first<last){

while(first<last && a[last]>=key)

last--;

a[first] = a[last];

while(first<last && a[first]<key)

first++;

a[last]= a[first]；

a[first] = key;

quicksort(a,left , first-1);

quicksort(a,first+1,right);

}

可以用挖坑填数发理解上面代码:设置起点first=left(数组索引，初始值为0)

Last=right(数组索引，初始值为数组长度-1),设置基准值key=a[left],设置基准值时相当于从数组中挖走了first位置的值，这时数组中first位置有了一个坑，然后从last往first遍历，直到找到一个比基准值key小的值，将这个位置的值挖出来放到first这个这个坑内，此时last位置又有了一个坑，于是又从first往last遍历，直到找到一个比基准值大的值，将这个值挖出来放在上一个坑last内。然后将key放在左边first这个可能。最后左右两边递归。

堆排序：时间复杂度为：nlogn。思路为：

所有的根节点均比所有的叶子节点大的为最大堆(降序排序)，小的为最小堆(升序排序)。每插入一个叶子节点，都会先将他放在对的最后位置，然后重调堆(让所有根节点均比叶子节点小或大。每次删除一个节点都是删除根节点(数组的第一个)然后将最后一个叶子节点放到根节点，然后进行重调。重调就是让所有的根节点均比所有叶子节点大或小。

了解哪些设计模式；

原型模式(原型模式创建类对象)、工厂模式(工厂模式创建类对象)、装饰器模式(String、Boolean、Number这三个包装器类)、观察者模式(事件监听)、单例模式(某对象全局只有一个实例，如window)

写个全排列 <https://blog.csdn.net/mxdzchallpp/article/details/74596858>

数据源a,b,c

1.拿出a，组合数据group为空，插入数据源a元素到组合数据group，此时group=[a]

2.拿出b，组合数据group拿出a，a和b组合，得到ab，把数据源b元素、ab插入组合数据group，此时group=[a,b,ab]

3.拿出c，组合数据group拿出a、b、ab，分别与c组合，分别得到ac、bc、abc，把数据源c元素、ac、bc、abc插入组合数据group，此时group=[a,b,ab,c,ac,bc,abc]

var data = ['a','b','c','d'];

function getGroup(data, index = 0, group = []) {

var need\_apply = new Array();

need\_apply.push(data[index]);

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

need\_apply.push(group[i] + data[index]);

}

group.push.apply(group, need\_apply);

if(index + 1 >= data.length) return group;

else return getGroup(data, index + 1, group);

}

console.log(getGroup(data));

框架：

了解哪些框架

说说React、Vue、Angular的区别

React是Facebook推出的，Vue是尤雨溪推出的，Angular是google推出的。

React用的es6,vue用的es5和es6,angular用的typescript

React采用的单项数据绑定，vue和angular采用的双向数据绑定。

MVVM：

MVVM是Model-View-ViewModel的简写。即模型-视图-视图模型。【模型】指的是后端传递的数据。【视图】指的是所看到的页面。【视图模型】mvvm模式的核心，它是连接view和model的桥梁。它有两个方向：一是将【模型】转化成【视图】，即将后端传递的数据转化成所看到的页面。实现的方式是：数据绑定。二是将【视图】转化成【模型】，即将所看到的页面转化成后端的数据。实现的方式是：DOM 事件监听。这两个方向都实现的，我们称之为数据的双向绑定。总结：在MVVM的框架下视图和模型是不能直接通信的。它们通过ViewModel来通信，ViewModel通常要实现一个observer观察者，当数据发生变化，ViewModel能够监听到数据的这种变化，然后通知到对应的视图做自动更新，而当用户操作视图，ViewModel也能监听到视图的变化，然后通知数据做改动，这实际上就实现了数据的双向绑定。

MVP模式(用于android开发):

M：model提供数据

V：View展示界面。是将activity的UI逻辑抽象为View接口

P：Presenter处理业务逻辑。是将activity的业务逻辑部分抽象为Presenter接口。

React 生命周期；<https://www.cnblogs.com/qiaojie/p/6135180.html>

1. 初始化
   1. getDefaultProps
   2. getInitState
   3. componentWillMount
   4. render
   5. componentDidMount
2. 更新
   1. props变化会执行componentWillReceiveProps,然后props变化或者state变化会出发shouldComponentUpdate(nextProps,nextState),如果为true的话执行componentWillUpdate—》Render🡪componentDidUpdate
3. 销毁 componentWillUnmount🡪结束

父子组件之间如何通信？子组件之间如何通信？

父子组件之间通过state和props通信。

子组件之间可以利用状态提升通过state和props实现子组件之间互相通信。也可以通过redux通信。

react的context

全局变量。父组件写static childContextType={data:ProtoTypes.string} getChildContext(){let name=”我是全局变量”return name},子组件写static contextType={data:Prototypes.string}然后直接通过this.context.name就可以调用

redux是怎么进行工作的？

像前面的context可以用来存放全局变量，当时它容易被任意修改，所以很不安全。Redux就是规定了用户可以进行哪些修改，并且进行修改时要声明。Dispatch就是用来进行修改操作的，他的参数为action,action规定了可以进行哪些修改，然后根据不同的方法对state进行修改。

为了方便使用封装了createStore函数，这个函数接收一个参数reducer，reducer就相当于与之前的dispatch，但是reducer在函数内部进行了一个判断，如果state不存在就创建一个state对象，然后根据action.type的不同类别进行修改。

Store有三个方法：getState()(获取state),dispatch(执行修改),subscribe(设置监听函数)

插件/工具：

webpack 怎么配置 ？

1. 创建webpack.config.js文件
2. 通过npm下载webpack插件。Npm install webpack –save.下完后会在项目目录下自动创建node\_modules文件夹(里面放的是所有依赖插件)
3. 配置webpack.config.js文件：

3.1配置入口文件：以对象形式配置入口文件:entry:{index:’./index’}或者以数组形式配置入口文件：entry:[‘./index’]

3.2配置文件出口output:{path(输出路径):’./dist’,filename(输出的文件名):’[name].bundle.js’},

babel 怎么配置

3.3配置加载器loaders(需要下载不同类别的加载器如css、js、png等等)loaders:[{test: /\.js?$/, loaders: ['babel-loader']}]

3.4 Resolve:{

//别名配置(配置之后可以在导入文件时直接使用别名;可以不配置别名)

Alias:{‘d3’:’d3/d3.js’},}

}

Babel怎么配置？

1. 生成package.json文件 npm init –y
2. 安装babel-cli。要全局安装和项目安装。Npm install babel-cli -g;npm install babel-cli –save;
3. 安装转码规则。Npm install babel-preset-es2015 --save;
4. 在根目录下创建.babelrc文件。{“presets”(转码规则):[“es2015”],”plugins(插件支持):[]}

loader中的东西弄完了，webpack最终输出什么了?

Loader可以对模块的源代码进行转换。Loader可以使你在import或加载模块时预处理文件。可以将其他语言(如TypeScript)转为JavaScript。

自我介绍：

* nodejs 了解什么（做过几个小爬虫）
* 具体用了哪些东西
* git （简历上的）怎么删除一个远程分支
* git常用命令？
* Git init git add . git commit -m git remote add myname <https://sdf.git> git reflog git log git push myname master
* Git reset –hard versioncode(通过git reflog可以得到)

项目：

* •项目
* 说说实习做的项目   项目中主要用的那些技术   如果现在有新的需求要做一个网站，你的技术选型是？
* 还有就是还是看项目看你的回答问问题，所以不要自己挖坑
* 自我介绍（我聊了聊学校的经历，以及项目经历）
* 项目的技术选型以及架构思想

HR面：

* 问了下学习成绩，考虑过读研吗？为什么选择工作？
* 问了下项目的情况和学习经历？
* 有没有拿到其他公司的 Offer？
*  了解我们公司吗？用过我们的产品吗？感觉怎么样？
*  了解我们工作时间和待遇吗？
*  问了下家庭情况？
*  你所期望的薪资价位？
*  还有什么想问我的？
* 你有用过拼多多么  你觉得前端方面拼多多有哪些不足

大概流程:

1. 自我介绍
2. 介绍项目
3. 提问(根据项目提问或直接问上面的这些问题)