1.为所有字符串添加一个去处首尾空格的方法？

var str = " dbc defgh ";

function **kg**(t){

var reg=/(^\s+)|(\s+$)/g;

return t.**replace**(reg,'');

}

**console**.**log**( **kg**(str));

2.JS中的数据类型

String boolean undefined number object array null

3.JS中的作用域？

js中作用域主要分两部分：1.函数内作用域；2.函数外作用域。

函数外作用域有称为全局作用域，在全局作用域下声明的变量称为全局变量，可以在脚本的任何地方访问。函数内作用域又称为局部作用域，在局部作用域下声明的变量称为局部变量，只能在当前区域或当前作用域下的局部作用域访问。

提升现象:会将脚本的变量以及函数的声明（未赋值）提升到

当前所在作用域的最顶端。函数的提升比变量的提升更靠上。

注意：重复的命名 会有覆盖现象

降低现象:当全局变量和局部变量重名时，局部作用域代码的执行会优先使用当前作用域中的 声明的变量，这种现象称为变量降低。

4.简述JS中声明函数的几种方式？

(1)var声明变量的特点

  1.存在变量提升

  2.允许多次申明同一个变量

3.作用域区分全局（函数外） 以及局部（函数内） 同名变量 就近原则

  4.声明的全局变量 会作为window对象的属性

(2)let和const 都是用来申明变量的关键字 功能和 var 类似

let:不存在变量的提升，不允许多次赋值， 声明的全局变量不会作为window的对象属性 ，存在块级作用域。

const：常量 常变的量 特点用法与let 一模一样 无法二次赋值 否则就会报错 不允许只声明不赋值.

(4)Function:

声明函数：

function 后面跟上的是函数名(){

函数体，重复执行的代码块

}

匿名函数

function(){

函数体，重复执行的代码块

}

5.排序算法（冒泡、选择、插入）？

*// 冒泡排序：通过两两比较，将最大或最小的元素放在末尾实现排序。*从小到大

    for (**var** j = 0; j < arr.length-1; j++) {*//循环4次即可 不用和自己比较*

*// 剩5个数比4次 4个比3次 当j++时 减少一*

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

            if (arr[i] > arr[i + 1]) {*//如果第一个数比第二个大 那么交换两个数的位置*

**var** t = arr[i];

                arr[i] = arr[i + 1];

                arr[i + 1] = t;

            }

        }

    }

console.log(arr);

选择排序：首先在未排序的数组中找到最小（大）的元素，放在数组的起始位置，再从未排序的数组中继续找到最小（大）的元素，放在已排序元素的后面，重复，直到完成所有的排序 从小到大

**var** arr=[2,7,11,44,55,33,12,32,56,21];

        for(i=0;i<arr.length;i++){

**var** min = i ; *//声明一个最小数*

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

                if(arr[j] <arr[min]){

                    min = j;

                }

            }

**var** t = arr[i];

            arr[i] = arr[min];

            arr[min] = t;

        }

        console.log(arr);

插入排序：*// 从小到大排列*

var arr = [2, 3, 4, 1, 56, 78, 5];

**insertionStor**(arr);

function **insertionStor**(arr) {

var perIndex,*//前一个数的下标*

currse;*//取下标*

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

perIndex = i - 1;*//0 下标开始为0*

currse = arr[i];*//1 下标开始为1*

*// 限定 下标最小为0*

while (perIndex >= 0 && arr[perIndex] > currse) {

arr[perIndex + 1] =arr[perIndex];

perIndex--;

}

arr[perIndex + 1] = currse;

}

return **console**.**log**(arr);

}

6.闭包？

闭包：外层函数A外面调用它内部声明的函数B，并且内部函数B中还使用了外部函数A中声明的局部变量，这样形成的闭合环境称为闭包。

闭包的作用：闭包用来对变量进行永久存储，或者用来防止变量污染。

闭包的注意事项：正因为闭包会使变量永久的驻扎在内存中不被销毁，那么必然会造成内存的浪费，所以不要滥用闭包。

7.简述JS中创建对象的几种方式？

1.字面量创建

var obj={};//创建了一个空的对象

2 .通过系统提供的Object构造的函数进行创建

var obj = new Object();//创建了一个空的对象

3.工厂模式创建

4.通过自定义构造函数创建对象（自定义构造函数就相当于类）通过系统关键字 new 来创建对象。

8.this关键字？ ！

普通函数中：指向window

函数：

只存在于函数内，在函数外写this是无意义的this在函数内指向函数的拥有者。

对象方法：当前的实例对象

构造函数：实例对象

9.原型、原型链？

原型:

每个构造函数都存在一个属性：prototype（原型），值是一个对象，所有通过构造函数创建的实例，都可以访问创建它的构造函数的原型上面的属性和方法，我们一般将方法以及值固定不变的属性都放在构造函数的原型上。

prototype原型：

每个构造函数都有属于自己的原型，原型上的属性和方法可以供构造函数创建的实例对象访问使用；

当原型上的属性和构造函数中的属性命名冲突时，构造函数中的属性优先级更高。

isPrototypeOf（）方法用来判断当前原型是否是某个指定实例对象的构造函数的原型

hasOwnProperty()方法用来判断对象的属性来自于自身的本地属性还是原型上的属性

原型链：

在继承过程中，子类、父类、父类的父类等等之间原型会产生关联，形成链状结构称之为原型链

所有对象都存在一个属性\_\_proto\_\_,指向创建当前对象的构造函数的原型

10.简述JS中的继承？

根据规则或特征提炼准则的不同，类的范围有所不同，将多个类间的相同特征写给一个公共类，这个就称为父类。子类的实例对象可以访问父类中定义的属性和方法。称之为继承。js中的继承是单继承。一个子级只能有一个父类，并且子类不能再作为父类的父类。

构造函数继承：

通过call()方法和apply()方法实现的继承称为构造函数继承，只能继承到父类构造函数中的属性和方法.

构造函数创建对象，需要用到系统关键字new

无法继承到父类原型中的属性和方法。

原型上的继承：

（1）引用继承

可以实现对父类原型上的方法和属性的访问，但存在问题，因为赋值，那么从此A的原型就是B的原型，B的原型也就是A的原型.

对B原型的操作就会直接影响父级A的原型.这是不合理的.

(2) 拷贝继承

可以实现父类原型的方法和属性继承，但也存在一些问题

虽然解决了引用继承存在的问题。但是拷贝继承也是一次性的继承，如果父类的原型又做了拓展，子类想要继承就必须再次执行拷贝继承，那么会存在大量的冗余代码执行。

(3) 真引用继承

既可以实现原型上的继承，也可以实现构造函数上的继承.并且继承完成后，对子类的操作也不会影响到父类，对父类的操作，子类可以动态获取.

实现思路：将子类元素，指向父类的实例对象

11.简述DOM操作中获取元素的方法？（直接获取以及DOM遍历） ?

*//1.节点的获取*

*//html 作为文档的根节点，下面只有两个子节点，分别是head和body*

*//document.head获取head节点*

*//document.body获取body 节点*

*//获取第一个子节点*

*//firstChild*

*// console.log(cs[5].firstChild);*

*//获取最后一个子节点*

*//lastChild*

*// console.log(cs[1].lastChild);*

*// 父节点获取 一个元素节点的父节点不可能是一个文本节点，只能是元素节点*

*//parentNode*

*// console.log(document.body.parentNode);*

*//将页面上第3个li 标签 内容换成 找到你了*

*// 先找p的父级，再找父级的所有子级，取下标获取ul ，再找ul的子节点 ，取下标 找到相应的li*

*// p.parentNode.childNodes[5].childNodes[5].innerHTML='找到你了';*

*//同胞关系*

*//nextSibling 获取指定节点的后一个同胞节点*

*// console.log(cs[0].nextSibling);*

*//previousSibling 获取指定节点的前一个同胞节点*

*// console.log(previousSibling);*

*//元素节点的获取*

*//子元素节点：chlidren 只会获取指定节点的所有元素节点，没有任何兼容问题。推荐指数5颗星*

**console**.**log**(document.body.children);

*//父元素节点：parentElement 存在兼容问题 IE6、7、8 不支持*

**console**.**log**(p.parentElement);

*//同胞元素节点*

*//nextElementSibling 获取指定节点的下一个同胞元素节点*

*//previousElementSibling 获取指定节点的上一个同胞元素节点*

*//如果不存在指定关系的元素节点，返回null；存在兼容问题 IE6、7、8 不支持*

*//firstElementChild:获取指定节点的第一个元素子节点 存在兼容问题 IE6、7、8 不支持*

*//lastElementChild:获取指定节点的最后一个元素子节点 存在兼容问题 IE6、7、8 不支持*

*//特殊节点*

*//document.getElementsByName():获取带有指定name属性值的元素集合*

*//更多用于获取表单元素*

*//documentgetxxx 与 document.querySelectorxxx 区别*

*//1.get类型的需要的参数是具体值；query类型要的是选择器字符串*

*//2.get类型的没有兼容问题；query类型存在兼容问题 ie9 以下 不支持*

*//3.get类型的得到是动态的元素集合--同步更 新；query类型得到的是静态元素集合--不同步更新*

*//get类型的query类型的不仅仅只能用于document，也可以用于元素节点，但是查找获取的范围就是当前元素节点的范围*

*//body.querySelectorAll('span') 查找获取的是文档中所有的span标签*

*//div.querySelectorAll('span') 查找获取的是div中所有的span标签*

12.为一个div元素绑定一个点击事件的方法？（行间、0级、2级）

*1.元素行间绑定:通过元素的onclick 属性*

*局限性：*

*(1)不适合代码量较大的事件处理;*

*(2)如果要赋值函数 必须写函数的调用 导致函数中的this关键字 不再指向当前元素而是widow*

*<div onclick="alert('123');console.log(123)"></div> -->*

DOM0级绑定事件处理程序：

*var d=document.querySelector('div');*

*// d.onclick=function(){*

*// console.log(111);*

*// }*

DOM2级事件处理程序绑定（事件监听）

*DOM 2 级 事件处理程序绑定（事件监听）*

*添加事件监听：*

*addEventListener(type,fn,boolean);*

*参数1-type：string 事件类型，例如：点击事件 ‘click’ 必须参数*

*参数2-fn：函数 可以是匿名 或声明好的函数 表示事件触发时要执行的操作 必须参数、*

*参数3-boolean:布尔类型 表示事件被添加到那个事件流中 可选参数 默认值为false 为事件冒泡，true就是事件捕获*

*特点：1，2，3，4*

*(1)事件捕获的发生要优先于事件冒泡*

*(2)当为同一个元素多次绑定同一类型事件，不会发生改变，会按照绑定顺序依次执行*

*移除事件监听：*

*removeEventListener();*

*参数1-type：string 事件类型，例如：点击事件 ‘click’ 必须参数*

*参数2-fn：函数 可以是匿名 或声明好的函数 表示事件触发时要执行的操作 必须参数、*

*参数3-boolean:布尔类型 表示事件被添加到那个事件流中 可选参数 默认值为false 为事件冒泡，true就是事件捕获*

*(3)注意：如果绑定的事件需要移除，那么绑定时除理程序就不能使用匿名函数*

*(4)addEventListener removeEventListener 存在兼容问题 IE9以下 不支持 IE低版本使用DOM 2 级 为：attachEvent(),detachEvent();*

*不支持事件捕获。*

*使用方式：attachEvent('click',fn)*

*detachEvent('click',fn)*

*var ds = document.querySelectorAll('div');*

*// ds[0].addEventListener('click',function(){alert('red')},true);*

13.浏览器嗅探？

浏览器嗅探，主要通过window.navigator 对象的userAgent属性包含浏览器名称的相关信息去判断浏览器的类型，手机端有Mobile

14.什么是事件冒泡？什么是事件捕获？

事件冒泡：事件触发的顺序，从最确定元素向不确定的元素传播。

事件捕获：事件触发的顺序，从最不确定的元素向最确定的元素传播

15.如何阻止事件冒泡、默认事件？

阻止事件的冒泡

var e=e||window.event;

if(e.stopPropagation){//非ie

e.stopPropagation();

}else{//ie

e.cancelBubble=true;

}

阻止默认事件的发生 :

if(e.preventDefault){ 非IE下

e.preventDefault();

}else{//IE下

e.returnValue=false;

}

16.页面跳转的方式有哪些？

A herf

开新窗口：*window.open()方式*

*// window.history属性 主要包含了 浏览器历史记录相关信息 通常用于移动端页面的切换与返回*

*// 1.length 历史记录长度*

*// 2.back() 历史记录的前一个*

*// 3.forward() 前往历史记录的下一个*

*// 4.go() 参数要一个number 正数后跳 负数前跳 使用前提是已经产生了历史记录*

*// locatoin.href 是一个读写属性 可以为其赋予其他url来实现页面的跳转，*

*// 也是js中最为常用的一种页面跳转方式 跳转方式 打开当前窗口加载url*

*// location.href='http://www.baidu.com';*

*// location.assign() 方法也可以实现页面的跳转*

*// location.assign('http://www.baidu.com');*

*// 上面两种跳转页面的方式功能相同。没什么区别*

*// location.replace()将当前url替换成指定url，与上面两种方式的区别在于，不会产生新一条的历史记录*

*// location.replace('http://www.baidu.com');*

17.cookie、本地存储、会话存储的区别？

大小

　　cookie的存储大小不超过4KB

　　sessionStorage和localStorage可以达到5MB或更大；

数据存在时间：

　　　LocalStorage：关闭浏览器数据依旧会存在，除非主动删除其中数据；

　　 sessionStorage：当前页关闭则数据删除；

　　　cookie：根据用户设置的cookie保存时间来定，在设定时间到来之前都会一直存在；

数据与服务器之间的交互方式

　　　　cookie的数据会自动传递到服务器，服务的也可以写到客户端

　　　　LocalStroage和sessionStroage的数据仅保存在本地

18.JQ中的ready方法和window对象的load方法有何区别？

.执行时间

window.onload必须等到页面内包括图片的所有元素加载完毕后才能执行。

$(document).ready()是DOM结构绘制完毕后就执行，不必等到加载完毕。

编写个数不同

window.onload不能同时编写多个，如果有多个window.onload方法，只会执行一个

$(document).ready()可以同时编写多个，并且都可以得到执行

简化写法

window.onload没有简化写法

$(document).ready(function(){})可以简写成$(function(){});

19.简述你对ajax的理解？原生ajax的流程

ajax:不是一门新的编程语言，只是一种前后台交换数据的技术ajax也被称为前后台交换数据的艺术

ajax的特点：异步请求,局部更新,

ajax的核心：XMLHttpRequest对象

ajax的使用流程：

（1）创建请求对象XMLHttpRequest,存在兼容问题，IE5/6不支持，在IE5/6上需要使用ActiveXObject对象

var xhr;

if(window.XMLHttpRequest){

xhr = new XMLHttpRequest();

}else{

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

}

（2）准备请求

open()方法：存在三个参数

参数一：string，表示请求类型GET/POST；

参数二：string，表示请求的url地址；

参数三：bool，表示是否发送异步请

ajax使用get请求提交数据，需要将数据拼接在url后面

var url = 'http://localhost/php/php\_02/ajax.php?uname='+uname+'&pword='+pword;

xhr.open('get',url,true);

（3）发送请求：xhr.send();

ajax使用post请求提交数据，需要先建立请求头，将提交的数据放在send（）方法中发送 xhr.open('post','http://localhost/php/php\_02/ajax.php',true);

建立请求头：

Content-type用来设置发送数据的类型

application/x-www-form-urlencoded：表示发送的是表单数据（'key=value&key=value） xhr.setRequestHeader('Content-type','application/x-www-form-urlencoded');

发送请求

var data = 'uname='+uname+'&pword='+pword;

xhr.send(data);

（4）请求监听：

xhr.onreadystatechange = function(){

一共要监听两个状态

readyState:请求状态

一共有0-4五个阶段

0：表示请求对象已经创建，但是尚未准备请求

1：表示请求已经准备，但是尚未发送请求

2：表示请求已经发送，但是服务器尚未接受请求

3：表示服务器已经接受请求，但是尚未处理

4：服务器已经处理请求

status:服务器返回状态

200：表示请求处理成功，并且返回数据

404：表示请求的页面不存在

500：表示服务器的未知错误

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

请求成功并且可以接受服务器返回的数据

服务器返回的数据存放在responseText

console.log(xhr.responseText);

if(xhr.responseText === '登录成功'){

document.querySelector('h1').innerHTML = '登录状态：已登陆';

}

console.log(xhr.responseText)

}

}

20.数组和字符串操作方法？（10起步）

21.同步和异步？

2）同步任务

    同步任务是指在主线程上排队执行的任务，只有前一个任务执行完毕，才能继续执行下一个任务，当我们打开网站时，网站的渲染过程，比如元素的渲染，其实就是一个同步任务

（3）异步任务

    异步任务是指不进入主线程，而进入任务队列的任务，只有任务队列通知主线程，某个异步任务可以执行了，该任务才会进入主线程，当我们打开网站时，像图片的加载，音乐的加载，其实就是一个异步任务

22.跨域？

跨域: 浏览器从一个域名的网页去请求另一个域名的资源时，域名、端口、协议任一不同，都是跨域

它是由浏览器的同源策略造成的，是浏览器施加的安全限制。

**跨域解决方案**

通过jsonp跨域  
document.domain + iframe跨域  
location.hash + iframe  
window.name + iframe跨域  
postMessage跨域跨域资源共享（CORS）  
nginx代理跨域  
nodejs中间件代理跨域  
WebSocket协议跨域

23.如何实现拖拽元素移动？

*//鼠标按下时*

*//事件对象*

box.**onmousedown** = function (event) {

*//获取它距离浏览器视窗的坐标*

var event = window.event || event;

*//获取 鼠标坐标*

x = event.clientX - box.offsetLeft;

y = event.clientY - box.offsetTop;

*// 在整个界面中移动 所以绑定给 document*

document.**onmousemove** = function (event) {

var event = window.event || event;

*//偏移量*

box.style.left = event.clientX - x + 'px';

box.style.top = event.clientY - y + 'px';

}

}

box.**onmouseup** = function () {

document.onmousemove = null;

}

24.如何实现活动倒计时倒？

*// （解题思路：*

*//  1.通过Date对象设置未来时间2018年7月7日*

*//  2.计算当前时间距离未来时间的时间差值*

*//  3.将时间差值通过换算得到对应的天、小时、分钟、秒*

*//  4.将所得天、小时、分钟、秒进行字符串拼接处理显示为要求格式*

*//  5.将上述4步放入计时器中不断的获取当前时间重新计算并显示）*

*//获取用户可操作的元素*

var box = document.**getElementById**('box');

function **K**(n){

n=n<10?'0'+n:n;

return n;

}

**setInterval**(function () {

*//当前的系统时间 检测不断变化的时间*

var date = new **Date**();

*//截止时间*

var end = new **Date**(2019, 10, 11, 0, 0, 0);

*//距离的总秒数*

var sums = **parseInt**((end - date) / 1000);

*//秒*

var s=**parseInt**(sums);

var m=**parseInt**(s/60);

var h = **parseInt**(m/60);*//获得小时数*

var day = **parseInt**(h/24); *//获得天数*

box.innerHTML = "距离抢购时间还剩："+ day + "天" + **K**(h-day\*24) + "小时" + **K**(m-h\*60) + "分钟" + **K**(s-m\*60) + "秒" ;

}, 1000)

25.说出至少10个数学方法并说明参数和用法？

// 1.Math.max() 返回参数中的最大数

// 2.Math.min() 返回参数中的最小数

// 3.Math.abs() 返回一个数的绝对值

// 4.Math.pow(x,y):计算x的y次幂

// 5.Math.floor() 对小数进行向下取整操作

// 6.Math.ceil() 对小数进行向上取整操作

// 7.Math.round() 对小数进行四舍五入取整

// 8.Math.random() 生成一个0-1 但包含1的随机数

// Math.sqrt() 方法返回数字x的平方根。

// Math.cbrt 方法用于计算一个数的立方根。

// Math.sin()

// Math.cos()

// trunc() 小数取整 返回整数部分 向下取整 floor()

// Math.hypot()

// Math.hypot方法返回所有参数的平方和的平方根。

// Math.sign() 用来判断一个数到底是正数、负数、还是零。对于非数值，会先将其转换为数值。

// 它会返回五种值。

// -参数为正数，返回 + 1；

// -参数为负数，返回 - 1；

// -参数为 0，返回0；

// -参数为 - 0，返回 - 0;

// -其他值，返回NaN。

26.post和get请求的区别？如何选择？

区别：

get请求会将请求提交的参数暴露在url上，不安全；而post请求相对安全

get请求所发送的数据量要远远小于post请求

get请求简单粗暴，直接将参数拼接在url后面发送请求，所以get请求请求的速度要比post请求快

应用场景：

如果单纯的请求数据，并不向服务器发送数据，那么通常选择get请求。

如果需要向服务器发送大量数据，或者发送的数据中包含敏感信息通常使用post请求。

27.移动端事件？

touchstart : 触摸开始（手指放在触摸屏上）

touchmove : 拖动（手指在触摸屏上移动）

touchend : 触摸结束（手指从触摸屏上移开）

touchenter ：移动的手指进入一个dom元素。j

touchleave ：移动的手指离开一个dom元素。

还有一个touchcancel，是在拖动中断时候触发。

28.mouseover和mouseenter的区别？

鼠标移入事件

(1) over优先于enter 触发

(2) over支持事件冒泡 enter不支持事件冒泡

29.变量提升？

js中的变量及函数会存在提升现象，会将脚本的变量以及函数的声明（未赋值）提升到

当前所在作用域的最顶端。函数的提升比变量的提升更靠上。注意：重复的命名 会有覆盖现象

30.常做的代码及页面优化？

(1) 使用精灵图 减少服务器的请求次数

(2) 资源合并与压缩

如果可以的话，尽可能的将外部的脚本、样式进行合并，多个合为一个。另外，CSS、Javascript、Image都可以用相应的工具进行压缩，压缩后往往能省下不少空间。

（3）延迟加载 懒加载

防止页面一次性向服务器响应大量请求导致服务器响应慢，页面卡顿或崩溃等问题。

（4）将css放在HEAD中

如果将 CSS放在其他地方比如 BODY中，则浏览器有可能还未下载和解析到 CSS就已经开始渲染页面了，这就导致页面由无 CSS状态跳转到 CSS状态，用户体验比较糟糕。除此之外，有些浏览器会在 CSS下载完成后才开始渲染页面，如果 CSS放在靠下的位置则会导致浏览器将渲染时间推迟。

1. if将概率大的放在上面。
2. For循环，减少循环次数。
3. 封装函数。

31.列举一些常做的兼容处理？

（1）target属性：返回正真触发当前事件的元素

IE9以下没有target,需要使用srcElement

var el = e.target || e.secElement;

（2）阻止事件的冒泡

在一些情况下 我们不希望事件冒泡的发生，需要手动阻止 父子级 不做同一件事情

非IE下：stopPropagation(); 停止传播

IE下：：cancelBubble 赋值为true

var wrap=document.querySelector('#wrap');

var body=document.body;

wrap.onclick=function(e){

var e=e||window.event;

//兼容写法

if(e.stopPropagation){

e.stopPropagation();

}else{

e.cancelBubble=true;

}

console.log('wrap');

}

body.onclick=function(){

console.log('body');

}

阻止默认事件的发生 :

if(e.preventDefault){ 非IE下

e.preventDefault();

}else{//IE下

e.returnValue=false;

}

滚轮事件的兼容处理：

wheelEvent(wrap, function (dir) {

if(dir){

console.log('向上滚动')

}else{

console.log('向下滚动');

}

})

function wheelEvent(el, callback) {

//1.判断浏览器

// 非火狐

if (el.onmousewheel === null) {

//非 火狐

el.onmousewheel =function(e){

var e=e||window.event;

var dir;

if(e.wheelDelta>0){

dir=true; //向上

}else{

dir=false; //向下

}

callback(dir);//滚动的方向

};

} else {

//ff

el.addEventListener('DOMMouseScroll', function(e){

var dir;

if(e.detail>0){

//向下

dir=false;

}else{

//向上

dir=true;

}

callback(dir);

}, false);

}

}

32.上传文件到服务器？

inp[1].onclick = function () {

//1. 获取 用户选择要上传的图片数据(需要t)通过file类型表单的files属性来读取

var file = inp[0].files[0];//下标0 对应的是file对象

console.log(file);

// 2. 开始ajax

var xhr = window.XMLHttpRequest ? new XMLHttpRequest() : new ActiveXObject('Microsoft.XMLHTTP');

xhr.open('post', './upload.php', true);

//3. 通常情况下 发送post请求 需要建立请求头 .

// 上传文件的时候，不建立请求头，让其自己判断发送文件的类型

// 处理数据，将数据进行统一格式包装 FormData:h5 新增对象 后台可识别的数据对象

var data = new FormData();

// 通过append()方法，来向 FormData对象添加内容

data.append('file', file);//第一个 后台根据 ‘file’读取的东西 第二个 变量名叫file

// $file = $\_FILES['file'];

xhr.send(data);

//4.请求监听

xhr.onreadystatechange = function () {

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

console.log(xhr.responseText);

}

}

}