1.什么是工厂模式/单例模式/观察者模式

工厂模式：对象的调用者和对象的创建者过程分离，当对象调用者需要对象时，直接向工厂请求即可，从而避免了对象的调用者与对象的实现，类似编码方式耦合，以提高系统的可维护性、可扩展性。

单例模式：单例：对象只能实例化一次就叫单例

                 字面量的对象声明（），其实在设计模式中可以看作是一种单例模式。所谓单例模式，就是永远保持对象的一个实例。

观察者模式：观察者模式又叫发布订阅模式（Publish/Subscribe），它定义了一种一对多的关系，让多个观察者对象同时监听某一个主题对象，这个主是对象的状态发生变化时就会通知所有的对空者对象，使得它们能够自动更新自己。

如：小孩哭了，（我们不知道他什么时候哭，于是通过等待观察，观察到哭了后），表示饿了，开始喂奶了。

2.什么是面向对象， 面向对象有什么特点(js中是没有多态的,多态分为重载(js没有)和重写)

面向对象：是一种编程思想，（高级的编程思想）。

好处：

    变量私有化，防止命名空间的污染，各个功能独立，便于维护

    缺点：代码变多，开发周期变长，成本增加。

特点:js中是没有多态的,多态分为重载(js没有)和重写.

3.构造函数有几种继承方式，分别是什么

一：通过改变父类的执行环境

二，通过call继承

三，通过apply继承

四--原型链继承

五--混合继承（解决以上缺点问题）

4.什么是作用域 js中有哪几种作用域?

作用域：  作用域： js代码可以执行的地方

   三种作用域：全局，函数，块级

5.作用域链是什么?

作用域链: 当一个作用域使用一个变量时, 如果当前变量已经存在,直接使用,

         如果当前作用域不存在,向上一级作用域查找,直到找到全局作用域, 如果还没有报错.

6.什么是原型链?

原型链: 每一个实例对象的\_\_proto\_\_指向其构造函数的prototype,原型相连构成了原型链

    对象查找属性时, 先从自身查找,如果没有从\_\_proto\_\_查找, 找到找到没有\_\_proto\_\_, 还没有返回undefined

7.Es6中如何声明一个类， 如何实现继承

1. Class 之间可以通过extends关键字实现继承， 这比 ES5 的通过修改原型链实现继承， 要清晰和方便很多。

      2 类的 prototype 属性和 \_\_proto\_\_ 属性

      3 Extends 的继承目标  extends关键字后面可以跟多种类型的值。

8.书写ajax的四步

**1.创建XMLHttpRequest**

**2.连接服务器**

**3.向服务器端发送请求**

**4.接受服务器的返回**

function ajax(url,fnSucc,fnFaild){

    // 参数;1.连接服务器的地址 2.成功时函数 3，失败时函数

    // 1. 创建xmlhttprequest对象

    var xmlHttp = null;

    if(window.XMLHttpRequest){

        xmlHttp = new XMLHttpRequest();//ie6及以下不兼容，针对主流浏览器

    }else{                             // 针对ie5, ie6

        xmlHttp = new ActiveXObject("Microsoft,XMLHTTP");

    }

    //2. 连接服务器

    xmlHttp.open('GET',url,true);  //xmlhttp.open(method,url,async)

    // method 包含 get 和 post两种方法 url主要是文件或资源的路径，async参数为true(代表异步)或者false（代表同步）

    //3.向服务器端发送请求

    xmlHttp.send();

    //4.接受服务器的返回

    xmlHttp.onreadystatechange = function(){

        if(xmlHttp.readyState == 4){   // 判断交互是否成功

            if(xmlHttp.status == 200){      //获取服务器返回的数据  获取纯文本数据

                //alert('成功：'+xmlHttp.responseText);

                fnSucc(xmlHttp.responseText);

            }else{

                //alert('失败');

                if(fnFaild){

                    fnFaild();

                }

            }

        }

    }

}

9.数据中存在student\_score表，字段分别为id（自增长）， student\_name（学生姓名）,  score（分数）,  mark（备注），书写sql

查询表格所有数据：select \* from student\_score

把分数大于90分的学生，备注修改为优秀：“Update student\_score Set mark ='优秀' Where score >=90”

插入一条学生的成绩，姓名：李楠，分数：68分，备注：良好：insert into student\_score {student\_name,score.mark}value{"李楠"，68，"良好"}

删除备注为退学或者分数为0分的数据：delete \*  student\_score Set mark='退学' score=0;

10.Php连接数据库分几步，每一步做什么？

1.连接数据库； $coon = new mysqli('数据库的地址', '用户名称', '用户密码', '连接的数据源', 数据库端口号);

  2. 准备一条SQL语句 ：  $sql = "";

   3.执行sql语句，如果是查询语句，成功返回结果集对象；如果不是，成功执行为true，执行失败为false  $result = $coon -> query($sql);

     （$result 返回的是一个对象）

    4. 判断返回是否执行成功 $row = $result -> fetch\_assoc();

11.状态码2xx开头的通常指什么意思？4xx的什么意思？5xx的什么意思？

    2xx:操作成功收到，分析、接受      4xx:请求包含一个错误语法或不能完成    5xx:服务器执行一个完全有效请求失败

12.获取dom元素的几种方式，哪种方式获取的是集合，哪种获取的是单个元素。哪些有兼容问题？

    1、通过ID选取元素（getElementById）

1）使用方法：document.getElementById("domId")

             其中，domId为要选取元素的id属性值

2）兼容性：低于IE8版本的IE浏览器对getElementById方法的实现是不区分元素ID号的大小写的，并且会返回匹配name属性的元素。

     2、通过名称name选取元素（getElementsByName）

1）使用方法：document.getElementsByName("domName")

              其中，domName为要选取元素的name属性值

2）说明：a. 返回值是一个nodeList集合（区别于Array）

         b. 和ID属性不一样，name属性只在少数DOM元素中有效（form表单、表单元素、iframe、img）。这是因为name属性是为了方便提交表单数据而打造的。

          c. 为form、img、iframe、applet、embed、object元素设置name属性时，会自动在Document对象中创建以该name属性值命名的属性。所以可以通过document.domName引用相应的dom对象

3）兼容性：IE中ID属性值匹配的元素也会一起返回

    3、通过标签名选取元素（getElementsByTagName）

1）使用方法：element.getElementsByTagName("tagName")

            其中，element是有效的DOM元素（包括document）

            tagName是DOM元素的标签名

2）说明：a. 返回值是一个nodeList集合（区别于Array）

                b. 该方法只能选取调用该方法的元素的后代元素。

                c. tagName不区分大小写

                d. 当tagName为\*时，表示选取所有元素（需遵从b.规则）

              e. HTMLDocument会定义一些快捷属性来访问标签节点。如：document的images、forms、links属性指向<img>、<form>、<a>标签元素集合，而document.body和document.head总是指向body和head标签（当未显示声明head标签时，浏览器也会创建document.head属性）

     4、通过CSS类选取元素（getElementsByClassName）

1）使用方法：element.getElementsByClassName("classNames")

      其中，element是有效的DOM元素（包括document）

      classNames是CSS类名称的组合（多个类名之间用空格，可以是多个空格隔开），

      如element.getElementsByClassName("class2 class1")将选取elements后代元素中同时应用了class1和class2样式的元素（样式名称不区分先后顺序）

2）说明：   a. 返回值是一个nodeList集合（区别于Array）

                  b. 该方法只能选取调用该方法的元素的后代元素。

3）兼容性：IE8及其以下版本的浏览器未实现getElementsByClassName方法

**5、通过CSS选择器选取元素**

1）使用方法：document.querySelectorAll("selector")

其中，selector为合法的CSS选择器

2）说明：a. 返回值是一个nodeList集合（区别于Array）

3）兼容性：IE8及其以下版本的浏览器只支持CSS2标准的选择器语法

13.检测数据类型的终极模式（用call的形式）

function checkType(data) {

            var str = Object.prototype.toString.call(data);

            return str.match(/\w+/g)[1];

        }

14.如何会形成闭包? 闭包的优点,缺点;

     闭包（closure）是javascript的一大难点，也是它的特色。很多高级应用都要依靠闭包来实现。

function f1(){

　　　　var n=999;

　　　　function f2(){

　　　　　　alert(n); // 999

　　　　}

　　}

优点：闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中，不会在f1调用后被自动清除。

缺点：（1）由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。

（2）闭包会在父函数外部，改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

15.解析函数

var name = "ling";

function sayName(){

   var a = {

     name:"zhang",

     sayName:getName

  };

  function getName(){

            console.log(this.name);

  }

  getName(); //this.name = ? ling

  a.sayName(); //this.name = ?  zhang

  getName.call(a);//this.name = ?zhang

}

      sayName(); //undefind

16. 数组去重方式。（最少三种）

**一、循环遍历法（传统思路）**

最简单粗暴的算法，新建一个空数组，然后遍历原数组，将不在新数组中的项添加到新数组，最后返回新数组

      function compare(arr){

    var newarr=[];//新建空数组

    for(var i=0; i<arr.length;i++){//遍历原数组

        var isadd=true;//设置标记

        for(var j=0; j<newarr.length;j++){//遍历新数组

            if(arr[i]===newarr[j]){isadd=false;break;}//如果当前原数组中的项，在新数组中存在，则标记为‘不添加’，并跳出新数组遍历        }

        if(isadd){newarr.push(arr[i]);}//若标记为‘添加’（即遍历新数组后，没有发现相同项），则添加进新数组    }

    return newarr;//返回新数组

}

**二、排序比较法（巧妙转换）**

相对第一种方法，需要每次遍历新旧两个数组，效率不高的情况。此方法先对原数组进行排序，这样每次只要与新数组的最后一项比较即可，大大提高效率

function sortarr(arr){

    var arrsort=arr.sort();//对原数组进行排序

    var newarr=[];//新建空数组

    newarr.push(arrsort[0]);//将排序后数值的第一项给添加到新数组

    for(var i=1;i<arrsort.length;i++){//遍历排序后的数组

        if(arrsort[i]!==newarr[newarr.length-1]){newarr.push(arrsort[i])}//若当前项与新数组最后一项不同，这添加到新数组

    }

    return newarr;//返回新数组

}

**三、对象属性法（另辟新径）**

相对前两种方法通过比较的方式，此方法主要利用了对象属性（key值）不可重复的特性，将数组中各项赋值给一个新对象，重复将自动覆盖。最后从对象上遍历出属性即可。

function attribute(arr){

    var obj={};//新建空对象

    var newarr=[];//新建空数组

    for(var i=0; i<arr.length;i++){obj[arr[i]]=null;}//将原数组项作为对象的key进行赋值

    for( var attr in obj){newarr.push(Math.floor(attr));}//遍历对象，取出key值添加到新数组

    return newarr;//返回新数组

}

17.Get请求和Post请求的区别？

        get:

        1.安全性较低:携带的信息,拼接在Url中.

        2.一次性传输的数据是比较小

        3.一般是向服务器请求数据

        4.get请求会有缓存问题

        post:

        1.安全性较高:携带的信息,在body中.

        2.理论上传输数据没有限制.

        3.设置数据,使用post请求.

        4.post请求没有缓存问题.

18.当你用ajax调用一个接口， 如何查看该接口返回的数据？如何查看该请求发送的数据？

1.首先可以在浏览器里面请求头里面查看ajax发送过去的数据，响应头能查看接口返回的数据

2.在ajax通过回调函数打印出来

19.什么是同源策略, 为什么要有同源策略

浏览器最核心最基本的安全策略

所谓同源是指传输协议，域名，端口相同

20解决跨域的方式你知道几种?

    1.   jsonp跨域原理：

      1、动态创建一个script标签，添加到body中

      2、设置script标签的src属性值，属性值就是一个接口

      3、通过接口上的某个参数，向服务器传递一个函数，通过

          这个回调函数 接收服务器返回的数据

   通过jsonp格式获取的服务器数据 大多都是object类型，不需要转换

    2.cors跨域：

    用户服务器的跨域请求数据

   php中设置头信息

   header("Access-Control-Allow-Origin:\*");//允许所有请求跨域访问该接口

21. 简述jsonp的原理，为何可以实现跨域？

jsonp跨域原理：

      1、动态创建一个script标签，添加到body中

      2、设置script标签的src属性值，属性值就是一个接口

      3、通过接口上的某个参数，向服务器传递一个函数，通过

          这个回调函数 接收服务器返回的数据

   通过jsonp格式获取的服务器数据 大多都是object类型，不需要转换

22.本地存储有什么,三种有什么区别

        1.sessionStorage

        2.localStorage

        3.cookier

相同点：都保存在浏览器端，同源的

不同点：

①传递方式不同

cookie数据始终在同源的http请求中携带（即使不需要），即cookie在浏览器和服务器间来回传递。

sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。

②数据大小不同

cookie数据还有路径（path）的概念，可以限制cookie只属于某个路径下。

存储大小限制也不同，cookie数据不能超过4k，同时因为每次http请求都会携带cookie，所以cookie只适合保存很小的数据，如会话标识。

sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。

③数据有效期不同

sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；

localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；

cookie只在设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭。

④作用域不同

sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；

localStorage 在所有同源窗口中都是共享的；

cookie也是在所有同源窗口中都是共享的。

Web Storage 支持事件通知机制，可以将数据更新的通知发送给监听者。

Web Storage 的 api 接口使用更方便。

23.说说你对cookie的理解

  在程序中，会话跟踪是很重要的事情。理论上，一个用户的所有请求操作都应该属于同一个会话，而另一个用户的所有请求操作则应该属于另一个会话，二者不能混淆。例如，用户A在超市购买的任何商品都应该放在A的购物车内，不论是用户A什么时间购买的，这都是属于同一个会话的，不能放入用户B或用户C的购物车内，这不属于同一个会话。而Web应用程序是使用HTTP协议传输数据的。HTTP协议是无状态的协议。一旦数据交换完毕，客户端与服务器端的连接就会关闭，再次交换数据需要建立新的连接。这就意味着服务器无法从连接上跟踪会话。即用户A购买了一件商品放入购物车内，当再次购买商品时服务器已经无法判断该购买行为是属于用户A的会话还是用户B的会话了。要跟踪该会话，必须引入一种机制。Cookie就是这样的一种机制。它可以弥补HTTP协议无状态的不足。在Session出现之前，基本上所有的网站都采用Cookie来跟踪会话。

24.什么是静态网站？ 什么是动态网站？

静态页面: 页面没有连接数据库, 没有数据交互

特点: 页面内容没有人修改, 页面的内容不会发生改变

动态页面: 页面通过后台连接数据库, 有数据交互

特点: 即使页面内容没有发生改变, 如果数据库的内容改变了, 页面内容也会发生改变

25.什么是前后端分离？ 什么是后台渲染页面？

前后端分离：

        前端：整个页面显示以及页面的交互逻辑。用ajax和node作为交互。其中node作为中间层。

        后端：提供api接口，利用redis保存session，与数据库交互。

后台渲染页面:

     1.加载：根据请求的URL进行域名解析，向服务器发起请求，接收文件（HTML、JS、CSS、图象等）

    2.解析：从html解析出DOM tree，解析css产生css规则树（计算css样式），js，通过DOM api和Css api操作DOM tree和CSS ruletree。

    3. 渲染：解析完成后，浏览器引擎会通过DOM tree和CSS RuleTree来构造Rendering tree（渲染树）（不包含Header等不需要显示的东西），也      就是将CSS rule加到每一个DOM结点上（或者叫做Frame）。定位坐标和大小，是否换行等。

    4.绘制：最后调用操作系统的Native GUI的API绘制.

26.前后端分离有什么好处？

前后端分离有什么好处

    1.提高开发效率

    2.提高工作效率

    3.避免前后端代码错误无法修复.

27.后端渲染有什么好处？

前后端渲染的区别：

前端渲染：

指的是后端返回json数据，前端利用预先写的html模板，循环读取json数据，拼接字符串，并插入页面。

好处：网络传输数据量小。不占用服务端运算资源（解析模板），模板在前端（很有可能仅部分在前端），改结构变交互都前端自己来了，改完自己调就行。

坏处：前端耗时较多。占用（一部分、少部分）客户端运算资源（解析模板）。前端代码多点，毕竟包含模板代码了么。

后端渲染：

前端请求，后端用后台模板引擎直接生成html，前端接受到数据之后，直接插入页面。

好处：前端耗时少，即减少了首屏时间，模板统一在后端。前端（相对）省事，不占用客户端运算资源（解析模板）

坏处：占用服务器资源。

28.聊聊前端优化

1.减少页面的回流和重绘（改变页面的结构：高，宽，行高）

    解决：    1.渲染大量数据时，可以使用文档碎片

              2.尽量减少dom操作，尽量不改变dom结构

              3.首屏加在，后面按需加载。

2.减少http请求

解决：           1.小的图片可以合成雪碧图（精灵图）

                 2.设置缓存

3.减少服务器压力

    1.前后端分离，使用ajax 获取数据 ，部分业务逻辑移到前端实现

    2.数据分页显示

29.聊聊前端自动化

30.网址中输入一个url的经过(或者一个http请求的经过)

31.http 和 https的区别

32.什么是深拷贝, 什么是浅拷贝.如何实现深拷贝.

33.Js中的异步函数有哪些

34.聊聊你对Promise的理解(可以从定义,功能,方法, 解决什么问题说)

35.宏任务, 微任务, 事件循环

36.浏览器垃圾回收机制

37.浏览器缓存机制