作用域链是什么

1. 为了规范变量和函数的有序性
2. 作用域链只能向上访问
3. 不能向下访问

this对象的理解

1. this指向它的直接调用者
2. this指向new出来的实例对象
3. 如果是事件中，this指向触发事件的对象

Ajax原理

1. 用户和服务器交互的中间层
2. 通过XMLHttpRequest来向服务器发送异步请求，在通过JavaScript和DOM渲染页面
3. Ajax的整个操作包括，XMLHttpRequest，JavaScript，DOM。其中XMLHttpRequest核心

Ajax的优点

1. 异步化的操作提升了用户体验
2. 因为异步化，前端处理了一些服务器的工作，使后端工作量减少
3. 减少了大量用户时的服务器压力
4. 可以实现不刷新页面的展示数据

Ajax缺点

1. 不容易调试
2. 暴露了客户端和服务器之间交互的细节

同源限制

1、协议、端口、ip地址相同

2、同源限制为了防止黑客入侵

promise的了解

promise的四种状态

pending初始状态

fulfilled成功状态

rejected失败状态

then方法有两个参数，分别代表，成功状态后的操作和失败状态后的操作

js的基本数据类型和引用数据类型

1. 基本数据类型有：
   1. undefined、null、string、number、Booleansymbol
2. 引用数据类型
   1. object、array、function

js的内置对象

1. 封装数据对象
   1. Array String Number Boolean object
2. 其他对象
   1. Function Date

null和undefined的区别

1. undefined表示不存在这个值
2. undefined是变量定义了但是没有赋值
3. null表示变量定义了值是空的
4. 判断一个值知否为null时，要用===

JavaScript中的“user strict”

ES5增加的运行模式，这种模式使得JavaScript更严格的执行，减少了代码的怪异行为

JSON

1. stringify ，将对象转为JSON字符串
2. parse ，将JSON字符串转为对象
3. 是一种轻量级的数据格式转换
4. 格式件单，易于读写，带宽小

同步异步的区别

1. 同步：浏览器向服务器发送请求，浏览器得到数据并刷新页面
2. 异步：浏览器向服务器发送请求，浏览器得到数据页面不刷新

ES6的理解

1、新增模板字符串

2、箭头函数

3、arguments对象

4、promise对象

5、let和const

6、块级作用域

7、let命令实际上就是增加了块级作用域

面向对象编程

1. 继承
2. 多态
3. 封装
4. 易维护
5. 易扩展

判断js判断一个数组

1. instanceof 判断变量是否属于Array对象
   1. 变量 instanceof Array
2. constructor 返回属性的构造函数
   1. 变量.constructor == Array;
3. Array.isArray(变量) ES6新增方法

let和var的区别

1. let命令不存在提升
2. 在块级作用域中let是暂时性死区
3. let不能在块级作用域中重复声明同一个变量名
4. var存在变量提升

map和forEach的区别

1、map：map方法中需要return值，此方法会将旧数组中的每一项作相同的操作，并返回一个新数组

2、forEach：就是循环数组的方法，其中第一个参数为数组中每一项、第二个参数为下标

箭头函数和普通函数的区别

1. 普通函数的this指向该函数
2. 箭头函数中不存在arguments对象
3. 箭头函数不能使用new声明
4. 箭头函数的this指向全局window对象

数组遍历的方式

1. for （in）
2. for
3. forEach

let

1. 不存在变量提升
2. ES6新增的定义数据的方式
3. 在块级作用域中不允许重复声明
4. 存在暂时性死区

const

1. 声明一个只读的值
2. const声明的值不能更改
3. const是ES6新增的定义数据的方式

var

1. 声明变量的作用域在变量位置的上下文中
2. 存在变量提升
3. 可以修改
4. 不能定义已存在的内部变量名

JavaScript组成

DOM 访问html和xml的接口

BOM 浏览器窗口的交互对象和方法

ES 语言基础

js内置对象

String Number Boolean Object Array Symbol Promise Function Date Map Set

JavaScript的基本规范

1. 代码缩进、使用四个空格
2. 代码段使用花括号包裹
3. 语法使用分号结尾
4. 变量函数使用前先声明
5. 大写字母开头的是构造函数、全大写的为常量
6. 用{}定义对象、用【】定义数组

作用域

1. 在花括号区域内称为作用域
2. function的花括号中为函数作用域

变量提升

1. 变量提升为JavaScript引擎隐式提升作用域到顶部
2. 在块级作用域中存在变量提升，在找不到一个变量时，会向外一层一层的寻找，知道找到window对象时会停止，这种就叫做变量提升

数组方法

1. concat合并数组
2. join拼接转为字符串
3. sort排序
4. pop尾部删除
5. push尾部添加
6. Unshift头部添加
7. shift投不删除
8. reverse逆置数组
9. slice读取开始和结束位置的数组不包括end
10. splice（index，length）截取index位置开始到length长度的元素，会改变原数组，返回一个新数组
11. indexOf查找返回下标
12. includes查找返回布尔值

MVVM

model：数据模型

View：界面

ViewModel：View和model的桥梁

改变this指向的三种方法

1. call
2. bind
3. apply

相同点

都可以改变函数内部this指向

不同点

1. call和apply会调用函数，并且改变函数内部的this指向
2. call参数是分项传递，apply参数以数组
3. bind返回一个函数

this绑定分类

1. **默认绑定**，默认绑定window全局环境
2. **隐式绑定**，隐式的绑定到函数内
3. **隐式丢失**，函数内部丢失绑定对象，从而绑定到了window
4. **显式绑定**，通过**call() apply() bind()** 方法将this绑定到对象上
5. **new绑定**，通过new关键字，创建构造函数this绑定到构造函数中

填空题

Promise构造函数有哪些静态方法：pedding、fulfilled、rejected、seeled

map对象中增删查的方法有：增set、删delete、查get、has检查是否存在

Promise中实例对象的方法有：then()、catch()

创建Object的方式有：利用字面量、利用Object.create({})、利用构造函数、利用构造函数加原型

Object构造函数的方法有：keys()获取键 、values()获取值、create()创建对象、is()判断两个对象是否相等、assign()合并对象、getPrototypeOf() 判断属于哪个构造函数

ES6新增的关键字：Symbol、Set、Map、 Promise、 arguments 、let

js实现继承的方案有：借助构造函数、借助原型链、借助构造函数加原型

ES6新增的实现异步编程的方案有Promise对象

ES6中可以使用 async 和 await 实现异步编程

Set集合中add方法用来新增元素，delete用来删除元素，has方法用来判断元素是否存在，forEach方法用来循环迭代元素

JQuery中使用html() 方法显示html代码，show() 方法用来将节点显示，hide（）方法将节点隐藏。val（）方法获取节点的value值，click()方法用来定义点击事件

Django中用来启动项目的命令：python manage.py runserver

Django中用来生成迁移文件的命令为：python manage.py makemigrations 应用名称

Django中生成数据库文件的命令：python manage.py migrate

Django中创建自子应用的命令：python manage.py startapp 应用名称

在模板字符串中使用 ${} 来传递变量或方法

在使用匿名函数时 可以使用 箭头函数 代替

ES6中新增find() 方法，可以查找元素是否存在，存在返回元素值

flat() 方法可以将多维数组转换为一维数组

Symbol的含义为 独一无二、唯一

Symbol的方法有for , keyfor

常见状态码500表示，服务器内部错误。404表示，未找到资源。200表示，访问成功

Promise中实例方法有 then catch ，静态方法有resolve、reject、all、any

Symbol解决了对象中属性相同时不覆盖的问题

异步编程的终极解决方案，async、await

JavaScript中欧异步操作有，Promise、Ajax、事件绑定、回调函数、定时器

解决跨域的方案有django中设置setting（corsheaders和中间件corsheaders.middlware.CorsMiddlware）、jsonp

jQuery中ajax的方法有，$.ajax $.get $.post $.getJOIN $.getScript

使用await失败后抛出异常通过，try、catch来捕捉

简答题

1. JavaScript原型，原型链。有什么特点？

js中每个对象初始化的时候都会生成一个\_\_proto\_\_属性，\_\_proto\_\_指向的就是对象的原型。当该对象没有\_\_proto\_\_属性时，就会向上寻找，而上一个\_\_proto\_\_还有它自己的\_\_proto\_\_属性，直到找不到时返回undefined，这种寻找的过程称作 原型链

通过构造函数实例化的对象，有prototype属性，这个属性指向它的构造函数

1. this绑定的方式有几种，场景

默认绑定：默认绑定到window对象上

隐式绑定：隐式的绑定到构造函数中

显示绑定：通过对象的方法apply、bind、call将this指向改变到某一个对象上

new绑定：通过new关键字，绑定到实力对象上

1. new操作符干了什么

创建了一个空对象，并且this指向这个对象

将构造函数的属性和方法都通过this指向new出来的对象

新创建的对象由this引用，最后隐式的返回this

1. 普通函数和箭头函数的区别

普通函数的this指向该函数，箭头函数的this指向上一层的this

普通函数有arguments对象，箭头函数没有arguments对象

普通函数可以通过new关键字构造，箭头函数不可以作为构造函数

1. prototype、proto、constructor三者关系

\_\_proto\_\_是每个对象的私有属性它指向原型

prototype 指向它的原型

constructor指向构造函数

1. 定义一个js 类 Person 属性：姓名 年龄 性别 方法：吃饭、跑步

定义一个Student类 属性：姓名 年龄 性别 学号 方法：学习

要求Student类继承自Person类， 在Student类定义的学习方法中调用父类的吃饭方法

自行创建对象进行测试

   class Person {

        constructor(username, age, sex) {

            this.username = username;

            this.age = age;

            this.sex = sex;

        }

        run = function () {

            return '跑步';

        }

        eat = function () {

            return '吃饭';

        }

    }

    class Student extends Person {

        // 没有constructor时this指向当前继承状态后的Student，有constructor时this指向当前没有继承的Student

        study = function () {

            let eat = this.eat();

            return '学习' + eat;

        }

    }

    var s1 = new Student('张三', 20, '男')

    console.log(s1);

    console.log(s1.study());

1. Ajax请求的过程

通过XMLHttpRequest来向服务器发送异步请求，然后服务器 返回数组，通过JavaScript和DOM操作将服务器返回的数据渲染到页面上

1. 讲述为什么要有同源策略

同源是域名、端口、协议都相同的一种安全协议。

同源策略为了防止黑客攻击，防止数据泄漏

1. Ajax优缺点

优点：异步化的操作提高了用户体验、降低了服务器的负载、前端替代服务器了一些工作

缺点：暴露了交互的细节，不易调试

1. 同步异步区别

同步：浏览器向服务器申请数据，浏览器拿到数据并刷新页面从而将数据展示到页面中

异步：浏览器拿到数据不用刷新页面也可以将数据展示到页面中

1. 如何中断Promise链
2. reject()
3. 抛出异常
4. 什么是回调地狱、什么情况下会产生回调地狱
5. 将上一个方法的返回值，作为下一个方法的参数，多层调用
6. 实例化ajax时会产生回调地狱
7. 列举django中间件的5个方法？以及django中间件的应用场景？

init调用父类方法

process\_request()请求中间件时第一个执行的方法

process\_response() 每次请求后返回的数据

process\_template\_response() 每次请求后返回的视图

process\_exception() 每次请求后抛出的异常