1、js模拟java和C语言的类的特点，有类才有作用域

2、GO：浏览器的垃圾回收对象

3、eval：浏览器提供的一个API【一个方法】，可以将字符串转换成对象，不安全，(不常用)

4、栈内存的特点（FILO）：先进后出，后进先出； first in last out

5、所有的JS都在栈的主线程执行

6、js执行环境

1、全局环境

1）、创建全局环境：window对象

2）、创建全局环境上下文(作用域) 【EO】 execution object

3）、创建变量对象（死对象：因为此时的变量都是undefined或者null，此时不能操作对象），【变量对象VO】variable object

4）、预编译：JS代码预编译（解析JS代码、先检查是不是JS代码，是的话就开始检查语法是否有错误，如果没有错误则开始变量提升，然后赋值，赋值之后可操作，变成活动对象，如果错误了，直接跳过，执行下一块代码，接着从3-4） 【活动对象AO】 activetion object

3到4的过程就是变量对象VO到活动对象AO的过程

2、局部环境

1）、创建局部环境

2）、创建局部环境上下文(作用域) 【EO】 execution object

3）、创建局部变量对象（死对象：因为此时的变量都是undefined或者null，此时不能操作对象），【变量对象VO】variable object

4）、预编译：JS代码预编译（解析JS代码、先检查是不是JS代码，是的话就开始检查语法是否有错误，如果没有错误则开始变量提升，然后赋值，赋值之后可操作，变成活动对象，如果错误了，直接跳过，执行下一块代码，接着从3-4） 【活动对象AO】 activetion object

3、eval环境（不常用）

1）、创建eval环境

2）、创建eval环境上下文(作用域) 【EO】 execution object

3）、创建eval对象（死对象：因为此时的变量都是undefined或者null，此时不能操作对象），【变量对象VO】variable object

4）、预编译：JS代码预编译（解析JS代码、先检查是不是JS代码，是的话就开始检查语法是否有错误，如果没有错误则开始变量提升，然后赋值，赋值之后可操作，变成活动对象，如果错误了，直接跳过，执行下一块代码，接着从3-4） 【活动对象AO】 activetion object

7、简单数据（基本数据）类型：Number数字、String字符串、Symbol唯一、Boolean布尔、undefined未定义、null空

8、引用数据类型：Objcet对象、Array数组、Date()时间对象、RegExp()正则表达式、Function函数.

9、基本数据类型和引用数据类型的区别：基本数据类型存放在栈里，存值；引用数据类型存放在堆里，存地址（16进制的字符串）

10、字典查询：通过一些状态码匹配一些对应的汉字，就是字典查询

11、函数的属性:

name：是函数的属性，只读，不可以被修改

callee：是arguments对象的一个属性，指向自身函数，可以配合使用递归，只能在自己的函数体内调用，解决了一些函数名冲突的问题

caller：是函数的属性，指向直接父级函数【废弃了】

callee和caller主要是配合递归函数进行一些操作

arguments：是对象，用来收集实参的一个集合，有数组的特点；length；

length：形参的个数

this：函数的上下文

12、事件流

规则：1、先捕获，再冒泡；2、事件源处于事件冒泡阶段

分为两个阶段（三个阶段）：

1）、事件捕获 ——网景公司提出

2）、事件源（目标元素的事件源）

3）、事件冒泡 ——微软提出

13、addEventListener（）；

1）、W3C浏览器（标准浏览器）提供的事件监听方法

2）、参数：1、事件类型；2、回调函数（需要些具体的监听代码）3、布尔值 true/false 可以省略，默认false冒泡，true为事件捕获

14、attachEvent（）；

1）、IE浏览器提供的事件监听方法

2）、参数：1、事件类型；2、回调函数（需要些具体的监听代码）3、布尔值 true/false 可以省略，默认false冒泡，true为事件捕获

15、标准浏览器：谷歌Google、火狐FireFox（firbug甲壳虫---调试工具）、欧朋、Safari

IE浏览器：怪异盒模型，兼容性很差，好多方法、属性都无法shim（优雅降级、兼容性无法解决）

16、事件监听的缺点：一直监听，就算不调用了，还存在于内存中，占用内存单元，消耗性能，容易造成堆栈溢出 解决：移除事件监听

17、let：声明变量的，let是有独立作用域，不存在变量提升，不允许重复声明

const：声明常量的，具有独立作用域，不存在变量提升。不允许重复声明

18、闭包：

函数嵌套函数，内部函数可以访问外部函数的变量，外部函数不能访问内部函数的变量，形成了一个独立的作用域空间（安全气泡）

19、闭包作用：

1）、保存：保存变量的值

2）、保护：把变量保护起来，不让其他人访问，避免全局环境污染

缺点：闭包使用过多，会造成变量不能被浏览器垃圾回收机制（GO）回收，造成内存占用过多，容易堆栈溢出

20、JS中哪些是闭包？

自执行函数，匿名函数，函数嵌套函数

21、构造函数和实例对象和原型对象三者的关系

原型对象是构造函数内部存在的，实例对象是new实例化之后得到的

22、在JS中，构造函数一般都是大写，构造函数想要正常使用，必须要实例化（new）

23、构造函数的内部执行过程：

1）、隐式创建一个空对象

2）、把当前函数的上下文（作用域/this）指向当前隐式创建的空对象

3）、将函数的参数赋值给当前隐式创建的空对象

4）、隐式返回当前对象

24、原型对象：

原型对象（prototype）存在于构造函数内，原型对象.是用来封装一些公共的属性、方法等，为了实现数据共享，减少代码冗余，提高性能

25、\_\_proto\_\_:实例对象原型链，浏览器使用

26、new 内部执行过程

1）、new触发了构造函数的内部执行,隐式创建一个空对象

2）、将构造函数的上下文指向当前的new创建的空对象

3）、将prototype 完全赋值给 \_\_proto\_\_，\_\_proto\_\_ = prototype

4）、隐式返回当前new创建的对象

27、当前原型内的constructor指向当前原型所在的构造函数

28、原型链：隐式原型，是一条作用域链，将内部作用域和外部作用域连接起来，形成了一条链

29、缓存：优点：直接从本地获取数据、主要是为了减轻服务器压力，为了开发方便，可以复用；缺点：不能实时更新状态

30、前端所有的资源都是静态资源，前端所有的静态资源都是从后台请求过来的

31、304:304状态码内，内容没有变化，则缓存下来，返回304，每次都向服务器请求，获取最新数据，然后浏览器解析并渲染页面。

阻止浏览器缓存：时间戳、随机数、让变量随时变化

32、Connection：keep-alice， connection是保持连接的意思、keep-alice缓存

33、cache，前后端交互时，缓存的字段

34、本地缓存：（WEB Storage）：[5种]

存储特点：存的都是字符串

浏览器想要获取本地存储中的数据，必须反序列化[JSON.parse（）]

localStorage（）：本地存储

         特点：长期存储，不管浏览器是否打开，都一直存储在内存中，一直存在，除非手动删除，不与后台进行交互，存储数据的方式：存字符串   要序列化，存储数据的大小：5M。最大 可以存20条数据：100M

sessionStorage():本地存储

特点：会话级存储，打开浏览器，则存进去数据，关闭浏览器，存储的数据自动消失，不与后台进行交互，存储数据的方式：存字符串   要序列化，存储数据的大小：5M最大可以存20条数据：100M

cookies：特点：存储字符串、可以存4kb、服务端进行交互、过期时间的设置

indexedDB:前端模拟的数据库，不常用

Web SQL: 前端模拟的数据库，不常用

35、对象内有一个toJSON，可以指定返回某个特定的属性，配合序列化使用的。

36、JSON.stringify(object)：序列化，就是将对象转换成字符串

37、JSON.parse()：反序列化，将字符串转换成浏览器能解析的对象

38、服务器：数据库专门存储数据，刷新数据也不会丢失

39、前端：WEB Storage 存储数据，刷新数据也不会丢失

40、前端存储分为两种

不管什么缓存，都是由服务器控制，（response headers）响应字段决定的

**强缓存**：直接缓存到本地，哪里需要当前数据，直接在本地获取即可，不用和服务端进行交互。

字段：cache-control和expires如果有的话，则说明当前缓存是强缓存，cache-control的优先级高于expires

强缓存的状态码：

200：发请求 => 对比，看看请求是否过期 => 过期 =>再次请求服务器=>变化了，更新=> 前端本地获取最新数据

**协商缓存**： 直接缓存到本地，根据一些过期时间等等的字段判断后台数据是否有更新，如果有更新，则前端浏览器获取后台返回的最新字段，根据字段更新前端缓存数据，然后可以使用

字段：etag：每个文件有唯一值，一旦文件有修改，则说明数据变化了，etag值跟着变化，每次值都是不一样的，hash值;last-modified：记录文件最后修改的时间，精确到秒

协商缓存的状态码：

304：发请求 => 对比，看看请求是否过期 => 没有过期 =>没有变化前端本地获取数据

41、函数类型：

回调函数

优点：让我们的代码更加灵活，可以实现封装很多的操作，

缺点：打乱了js的执行机制

高阶函数

参数为函数或者返回值为函数的函数是高阶函数

箭头函数

箭头函数是ES6的新语法，

没有this，借用了直接父级的上下文（this）

不能被用做构造函数

不能被实例化

（）（）自执行函数，不传参this指向数据window

递归函数：

直接或者间接的调动自己，这样的函数叫做递归函数

注意：必须要有结束条件，否则造成堆栈溢出

42、JS本身没有块级作用域，作用域都是模拟出来的

43、JS内作用域分为两种：

词法作用域[VO]：就是声明了还没有执行呢，此时他的this是undefined

执行作用域[AO]：执行时的作用域，此时才有确定当前上下文，当前this指针指向哪里，就有了作用域

作用域链：就是我们访问外部变量的通道，是 Scope chain（\_\_proto\_\_）作用域链

44、普通函数，this指向window（谁调用指向谁）

定时器，this指向window

对象内的this，谁调用指向谁（指向调用者）

事件监听this执行时看具体是谁调用、指向直接调用者

构造函数，this指向实例对象，new可以改变this指向

构造函数内部return对象，则this指向当前的return对象

构造函数内部只要return的是对象，则指向当前对象，否则还是指向实例对象

严格模式中，普通函数this指向undefined，不是window了

DOM对象的this指向当前调用者（指向的是一个DOM对象）

45、域名的组成：

四部分：

1、协议 例：http（网络协议）、https（网络协议）、ftp（上传协议）、file（本地协议）

2、域名 例：locallhost baidu

3、子域名 http：//zimo.niu.com niu：子域名 可以没有

4、端口号：

（默认隐藏的）前端默认的端口号是8080，四个数字组成

服务端默认端口号：443

本地服务器端口号是暴露出来的 localhost（127.0.0.1）是本地服务器

前端本地服务器：http：//localhost：8080

5、http和https都是建立在TCP传输协议之上的，http是明码传输数据，https是加密传输数据，https是在http请求的基础上。添加了SSL、TLS安全协议证书（要钱）

46、JS主线程执行栈，优先执行同步任务

47、同步任务执行结束了，然后才开始检查任务队列，执行异步任务

48、ajax：异步请求对象（）

概念：在不刷新页面的情况下，获取并刷新某块区域的数据（服务端数据）

作用：异步请求对象，进行前后端交互，是一个工具

缺点：破坏了JS执行机制；破坏了浏览器的回退按钮；地狱回调；对SEO搜索引擎支持较弱，数据比较难抓取

49、封装ajax步骤：

1）、创建异步请求对象，注意兼容性

标准：XMLHttpRequest IE：ActiveXObject

2）、建立连接（和服务端建立连接）

open(‘请求方式’,’请求地址’,true) true代表异步请求

3）、发送/提交数据

send()

get:

请求发送null对象就行，send(null)

请求头：setRequestHeader(‘Content-type’:’application/json;charset = utf-8’)

post：

请求发送正常参数 send(params)

请求头：(‘Content-type’,’application/x-www-farm-urlencoded; charset = utf-8’)

4）、注册监听数据（监听服务器给前端响应的数据）onreadystatechange

条件： readyState：响应状态码，后台响应完成后，前端才可以获取数据并操作

status：http状态码 200 <= status < 300 || status == 304

50、readyState响应码：

0 － （未初始化）还没有调用send()方法

1 － （载入）已调用send()方法，正在发送请求

2 － （载入完成）send()方法执行完成，已经接收到全部响应内容

3 － （交互）正在解析响应内容

4 － （完成）响应内容解析完成，可以在客户端调用了

51、status：前端状态码200 是请求成功的状态码

52、不管什么请求，传递数据的过程中，都必须是字符串，因为http请求是超文本传输协议，注意：undefined

53、所有的JS都在栈内存的主线程内执行，主线程先执行同步任务，然后再去检查异步任务，有异步任务就执行，否则结束JS引擎执行（停止）

54、跨域：是浏览器产生的，因为安全性考虑，不让你随便拿数据（访问）

55、同源策略：浏览器产生的同源策略，是WEB浏览器的最基础的安全核心，如果一旦协议、域名、子域名、端口号中有一个不一样，则违反了同源策略，产生跨域问题，不让访问数据

56、同源：协议、域名、子域名、端口号都一样

57、跨域的解决方式：

1)、JSONP：

原理：动态的创建script标签，让src指向当前需要访问的地址，因为src属性指向的是外部地址，不受同源策略的限制

优点：创建的script，解决跨域问题；简单，使用回调函数的方式跨域的

缺点：只支持get请求，不支持其他请求

2)、cors跨域：

原理：设置响应头部，允许访问即可

3)、nginx跨域：

原理：反向服务器代理

4)、postMessage()

H5中新增的提交数据的方法，不受同源策略的限制

5)、document.domain + iframe （不常用）

原理：利用iframe的src属性进行跨域访问

缺点：只有在主域相同的时候才能使用

6)、location.hash + iframe （不常用）

原理：利用lacation.hash进行传值

7)、window.name + iframe （不常用）

原理：name值在不同的页面（甚至不同域名）

加载后依旧存在，并且可以支持非常长的 name值 （2MB）

8)、websocket：持久性请求

有状态的请求，只要连接上服务器，不主动断开连接则一直连接着服务器

多用于实时聊天，请求方式是【ws】，不受同源策略的限制  
  双向通道：双向通

58、window.location.href：获取地址栏全路径// "https://www.baidu.com/?tn=88093251\_85\_hao\_pg"

window.location.host：获取域名和端口// "www.baidu.com"

window.location.port：获取端口号// "" 隐藏的

window.location.protocol：获取协议// "https:"

window.location.hostname：获取当前域名//"www.baidu.com"

window.location.pathname：获取除了域名、协议、子域名、端口号之外的其他的路径// "/"

window.location.origin：获取域名、子域名、协议、端口号// "https://www.baidu.com"

window.location.search：获取地址栏的参数// "?tn=88093251\_85\_hao\_pg"

59、http请求是无状态的：不会一直连接着服务器，数据传输完毕就会断开连接

60、SYN(建立联机) ACK(确认) FIN(结束)  PSH(传送)

61、任务队列的特点：只要是排队的任务，必须所有的执行完成

62、任务队列的进出规则：先进先出，后进后出（FIFO）

63、简单请求：post、get、head

非简单请求：put、delete

64、GUI线程（UI线程）负责页面的渲染工作，调用paint()画出页面，页面渲染过程和JS执行线程互斥

65、现代交互，就是异步解决方案

axios:前后端交互的一个库【插件】，基于Promise封装的

ajax：前后端交互的工具，创建异步请求对象

Promise():是ES6提出的前后端交互的一种工具，用来解决ajax解决不掉的地狱回调

是一个对象、也是一个构造函数，接受一个回调函数作为参数，这个函数叫做函数解析器，就是executor，

Promise对象，如果给成功的回调函数传的是字符串，而不是一个函数的话，则会发生值穿透现象

executor接受两个参数，一个是成功的回调函数，一个是失败的回调函数

特点：

 1、不可逆，只能从准备状态到成功，或者从准备状态到失败，过程不可逆

有三种状态：

1、准备状态  pending

2、成功     resolved

3、失败     rejected

2、每个then和catch函数都重新返回一个新的Promise对象-----then和catch为了可以链式调用？

3、链式调用、解决了ajax的地狱回调问题

async await：终极异步操作，后面根的必须是异步代码【必须跟Promise对象】

async 后的任何数据都代表是异步的了

await 必须在async内部，await代表的等待返回结果，有结果了再执行下一步，没有返回结果，则一直等待

66、JS动画就是改变当前元素的left、top、right、bottom值 缺点：卡顿

67、currentStyle：IE提供currentStyle可以获取到当前元素当前属性值

68、getComputedStyle：标准浏览器（w3c）获取当前元素当前属性值

69、Content-Type格式：

text/html ： HTML格式

text/plain ：纯文本格式

text/xml ：  XML格式

image/gif ：gif图片格式

image/jpeg ：jpg图片格式

image/png：png图片格式

application/xhtml+xml ：XHTML格式

application/xml     ： XML数据格式

application/atom+xml  ：Atom XML聚合格式

application/json    ： JSON数据格式

application/pdf       ：pdf格式

application/msword  ： Word文档格式

application/octet-stream ： 二进制流数据（如常见的文件下载）

application/x-www-form-urlencoded ： <form encType=””>中默认的encType，form表单数据被编码为key/value格式发送到服务器（表单默认的提交数据的格式）

70、构造函数原型方法：是加在prototype的方法

静态方法：构造函数内部的方法，不能被继承，只能自己用（一般情况下），通过函数名调用，不被实例化，静态方法也可以使用

promise对象的静态方法：

race():可以接收一个数组作为参数，返回的是一个新的Promise对象，数组内是多个promise实例，谁执行的快，最先得到结果，则race()执行结果就是那个Promise对象

举例：Promise.race([p1,p2,p3,p4]) ===> p3的Promise对象 // {}

all():接收一个数组作为参数，结果返回一个数组，数组内是全部的Promise对象，每一个Promise对象执行完毕才返回结果

resolve():成功的回调

reject():失败的回调

Promise对象的原型方法：

then():成功的回调

catch():失败的回调

finally():异常处理

71、ES6 class类是function的语法糖，原理还是function模拟出来的

语法糖：写法简单，实现的功能一样，用起来舒服

72、异步代码：（I:input O:output）

AJAX，定时器，回调，I/O操作  Promise  async-await【终极异步解决方案】

73、事件循环：

1）、所有的JS都在栈的主线程执行

2）、执行栈优先执行同步任务，从上到下

3）、等主线程执行完成同步代码了，主线程空闲了，才去检查任务队列中是否有异步任务，有则拿出来到执行栈（主线程）执行，当前任务执行完毕，继续执行下一个任务，直到任务队列中没有任何任务，停止执行

4）、优先检查上一个宏任务中是否有遗留下来的微任务，有就执行，没有就执行当前宏对象，然后在执行当前宏任务内的微任务

5）、一直循环3、4步骤，形成一个事件循环，直到任务队列中没有任务

74、宏任务：

setTimeout、setInterval、setImmediate[立即执行函数]、requestAnimationFrame、I/O、UI渲染

75、微任务：

Promise.then()、Promise.catch()、finally、MutationObserver[监听器]、 process.nextTick()、Promise对象本身是异步操作，但是Promise函数体内同步执行、Object.observe()[监听对象]

76、async和await后面返回的是promise对象或者一个值，当事件循环执行的过程中，遇到await的时候，立马返回结果，然后之后的同步代码都扔到微任务队列去，宏任务扔到下一个宏任务中去，如果async中没有遇到await的时候，则当同步代码执行

77、input的控件必须设置成file类型，才是文件上传 【file是H5新增的表单控件】

multiple:设置是否允许多文件上传

accept:可以指定上传文件的类型

MIME:多文件类型

78、FileReader:电脑本地的文件系统

onload：文件读取完成的函数

readAsDataURL:BASE64格式的文件，传递给后台

readAsText:转成文本传递给后台

readAsArrayBuffer():就是一个数组，当前数组内存放的是二进制的数据

FileList:当前上传文件对象

0: 当前文件所有信息 name:文件名、size:文件大小、type:文件类型

length:上传文件的数量

79、generator函数 === 生成器函数

generator函数中的\*  代表的是async

yield:是打断执行，就是停止，如果想继续向下执行，必须调用next() 函数

yield就是await

generator函数执行返回一个迭代器对象 { value:值/undefined,done:false/true }

当done为true时，则说明当前generator函数执行完毕了，否则当前函数没有执行完成、当done为true时，则value为undefined

async和await的实现原理是generator函数

原型方法：

next():下调指针，执行下一步函数

return():一旦调用，则立马结束当前生成器函数

throw():汇集错误信息的   不用

80、将图片转换成base64，都是小于8kb的图片，因为大于8kb，转换后大于原始图片，没有提高性能

81、JS模块化

一般有四种：分别是AMD，CMD，CommonJS，ES6 Module

82、什么是JS的模块化？

JS 的模块化就是把项目中各种功能单独抽离出来，封装成一个单独的功能（单独JS文件），不同的JS文件之间可以相互引用，实现了JS与JS文件之间的交流

83、模块化优点：

减少http请求，优化性能

减少冗余代码，提高代码的复用性

集中管理，便于维护，便于测试，便于版本迭代

偏向于工程化

解耦，模块和模块之间尽量不要有任何关系

灵活，模块之间也可以自由组合

解决了命名冲突问题，避免了环境污染

可以多人协作开发，互不干扰

84、模块化的缺点

系统分层，调用链过长，消耗性能

模块间通信，模块间发送消息会很耗性能

85、AMD：是一种模块化规范，代表者是RequireJS

特点：异步加载、依赖前置、全部加载、通过define函数定义模块

定义模块：define函数定义

define(‘模块名？’,[‘依赖’]?,( 形参：用于接收依赖中的数据)=>{

暴露模块

return{

}

})

引入模块：define、require、requirejs

require({‘依赖’],function(形参：用于接收依赖中的数据) {

//使用依赖的代码即可

})

86、CMD：是一种模块化规范，代表者是 SeaJS ；页面中引入sea.js 然后引入入口js文件；既遵从AMD规范，有遵从CommonJs规范

特点：异步加载、没有依赖前置、按需加载、通过CommonJS暴露模块（module.exports）、通过define函数定义模块、就近原则（用到的地方再去引入模块）

定义模块：define函数定义

require：引入模块 exports：暴露模块 module：模块标识

define（（require，exports，module）=>{

//暴露模块

1、CommonJS默认暴露模块

module.exports = {}

2、CommonJS单个暴露方式

exports.xxx = 值

3、AMD暴露方式

return

}）

87、CommonJS：是一种模块化规范，代表者是Node.JS，运行在NODE中的，浏览器不支持。NODE环境默认有require模块

暴露模块：

module.exports = {}

exports.xxx = 值

引入模块： require

特点：运行在NODE环境中，浏览器不支持、同步加载

88、ES6 Module：ES6新增的一种模块化规范，页面中给script标签设置type="module",否则不会解析

单独暴露和引入：

暴露时：

1、export let xxx = 20;

引入单独暴露的变量时，有两种引入方式：

1、import { 暴露的变量名 } from './文件';

2、import \* as xxx from './文件';

默认暴露和引入：

暴露时：

export default {}

引入时：

import xxx from './文件'; 【xxx:可以自定义】

89、浅克隆：其实就是原来的数据复制一份，只是复制了值，地址没有变化，修改克隆之后的数据，能影响原来的数据

Object.create():浅克隆【对源对象的引用】

Object.assign():第一层数据深克隆，嵌套数据是浅克隆

第一个参数是目标对象，第二个参数是原对象

作用：将原对象赋值一份到目标对象中，返回目标对象

90、深克隆：完全复制一份原数据，在内存中开辟一个新的内存空间，修改克隆之后的数据，不会影响原数据

最粗暴的深克隆：

JSON.parse(JSON.stringify(obj)):不能克隆undefined、function、Symbol、BigInt()

JSON.parse(JSON.stringify(obj)):原数据中有NaN、+Infinity、-Infinity都会转成null

JSON.parse(JSON.stringify(obj)):原数据中有Regexp、Error()、会被转换成空对象()

JSON.parse(JSON.stringify(obj)):如果原数据中有构造函数生成的数据，丢失当前构造函数的constructor

JSON.parse(JSON.stringify(obj)):如果原数据中有日期对象（Date）则会转换成日期字符串

91、迭代器对象：是一种新的数据结构，迭代器对象可以被for of进行循环的对象，可以被for of循环的对象是迭代器对象

Symbol.iterator:是一个迭代器的接口，有了此属性则当前对象可以被for of循环

92、JS特征：

继承：JS属性、方法都可以实现继承（复用）

.call：调用所有者对象作为参数的方法

封装：把相同的代码抽离出来，单独进行封装，形成一个公共方法

多态：封装一个函数，可以有多个功能

抽象：比较抽象，模拟JAVA中的抽象概念

93、ES5继承：构造函数继承

构造函数继承：

缺点：原型方法无法继承，只能继承自身属性

原型继承：

缺点：无法继承自身属性，构造器constructor的指向父级

组合继承：

缺点：通过构造函数和原型进行继承，构造器constructor的指向父级

寄生式组合继承：

缺点：构造器constructor的指向父级

完美继承：

优化：

手动更改子类的constructor指向

通过Object.create/Object.assign()，复制一份父类原型属性及方法给子类

94、ES6继承：class类继承：是ES6新增的API，是有了类的概念，有了块级作用域，更像编程语言，偏向于真正的强类型编程语言

class

1）、class类和构造函数完全没有关系

2）、class类是function的语法糖，有块级作用域，强类型、偏向于后台语言（java C）

3）、类中默认是严格模式 ‘use strict’

4）、constructor类中默认有的方法，可以声明，不声明的时候，类会自动创建一个constructor函数

5）、类中可以声明静态方法，可以直接通过类名调用，关键字是static（可以不用实例化）

6）、直接声明的方法是原型方法，加载prototype上的方法

7）、class继承：通过extends进行继承，通过super进行接收

8）、子类中如果要访问继承的数据，必须要在super调用之后，否则报错

9）、super以函数调用的时候，则当前super当前父类

10）、class继承过程：class进行继承时，通过extends 继承，super接收，当前super指向的是父类，触发了父类中的构造器（constructor），父类中的构造器把属性、方法传递给子类，并且触发了子类构造器的执行，把this改变到子类中去

11）、当super以对象调用时，在子类的静态方法中可以继承父类的静态方法

12）、当super以对象调用时，在子类的原型方法中可以继承父类的原型方法

13）、类不存在变量提升

95、canvas，H5新增画布 默认宽高300 x 150

可视化库（图表库）：echarts D3.js three.js highcharts.js

96、正则：其实就是一个算法验证及匹配是否符合条件，符合条件返回true，不符合返回false

有两种验证方法：

1、text()：验证是否匹配成功，成功了返回true，失败返回false

2、exec()：验证捕获，有就拿出来使用，如果没有返回null，

[123]：匹配字符集的集合，匹配123，如果匹配[^123]，除了123全部匹配

正则有两种表示方法

1、let reg = new RegExp() [不常用]

2、let reg = //；

写法：

+：匹配一次或多次 == {1,}

.：匹配所有字符

？：匹配0次或1次 == {0,1}

[]：匹配字符集的集合

{}：匹配的次数： 例：{0,10} ：匹配0-10次 {4，}：最少匹配四次 {5}：匹配五次

^：从头开始匹配

\*：匹配0次或多次 == {0，n}

$：匹配的结束边界

a-z：所有小写字母

A-Z：所有大写字母

0-9：所有数字

-[0-9]：所有的数字，负数

\d：代表数字 == 0-9

\D：代表的是非数字 == [^0-9] 此时^在[]内。表示取非

\w：代表的是组合 == [a-zA-Z0-9\_]

\W：代表[^a-zA-Z0-9]

\s：代表空格

\S：非空白字符

\n：换行

\r：回车

\\ == \

97、函数柯里化：  
    是一种编程思想，就是把复杂的东西简单化  
    具体表现为函数柯里化 === 把多个参数变为一个参数的函数