# 第一阶段面试题

## 一、MySQL数据库

### 1. MySQL中varchar与char的区别以及varchar(50)中的50代表的涵义？

**答案：varchar:65535个字节，变长字符串，查询速度较慢**

**char:255个字节，定长字符串，查询速度较快**

**varchar(50)：50个字符，**

**5.0以上版本50个字符，5.0以下版本是50个字节**

### 2. 简述浏览器打开www.codeboy.com显示页面，中间都经过哪些过程？

**答案：1.向DNS服务器获取域名对应的ip地址并返回给浏览器，通过ip地址去请求服务器，web服务器获取网页所需要的数据或文件，最后将完整的网页响应给浏览器(前端)。**

### 3. 如何解决MySQL中存储中文乱码问题？

**答案：1.sql脚本文件采用UTF8编码 2.客户端连接数据库时采用UTF8编码 3.服务器创建数据库时采用UTF8编码。**

### 4.Float和Double的区别是什么？

**答案：float浮点型 double 双精度**

## 二、JS基础

### 1. continue和 break有什么区别？

**答案：continue跳过本次循环，继续下一次循环**

**break终止整个循环体，不再继续执行**

**return 返回值，**

### 2. i++和++i的区别？ 计算：var n=5; 求 var num=n++ + ++n + n++ + ++n +n;

**答案：i++ 先参与运算，再自增**

**++i 先自增，再参与运算**

**5+5+7+7+9+9=42**

**n + n++ + ++n + n++ + ++n + n**

**5 5 7 7 9 9**

### 3. JavaScript都有哪些数据类型？

**答案：原始类型，也叫基本数据类型： 数值/字符串/布尔/undefined/null**

**引用类型:对象，数组，函数**

### 4. 自调用函数是什么？用于什么地方？

**答案：（function(){//函数体})(); 创建一个局部作用域，不造成全局污染**

### 5. slice和splice有什么区别？

**答案： slice(s,e):截取子数组或子字符串，s开始的下标，e结束的？？**

**splice(s,c,e1,e1)：即可以删除元素，也可以添加新的元素 s开始的下标，c删除的个数，e1**

### 6. typeof返回的类型有哪些？

**答案：number,string,undefined,object,boolean,function**

### 7. 取 1~11之间的随机数 (即包括1不包括11)？

**答案：Math.ceil(Math.random()\*10+1） 方法1**

**parseInt(Math.random()\*10+1) 方法2**

### 8. 什么是变量声明提前？

**答案：使用var关键词，声明的变量，会将声明提升到作用域的最前边**

### 9. push、pop、shift和unshift 区别？

**答案：pust/pop:是在数组尾部（栈顶）添加/删除**

**unshift/shift:是在数组的头部（栈底）添加/删除**

### 10. 例举强制类型转换和隐式类型转换

**答案：强制转换：Number()，parseInt(), parseFloat() toString()**

**隐式转换：字符串+数字 数字转换成字符串，拼接**

**字符串+布尔 布尔转字符串**

**数字+布尔 布尔转数字**

### 11. 函数声明与函数表达式的区别？

**答案：相同点：都是可以创建函数**

**不同点：**

**函数声明：存在声明提前； 函数表达式：不存在函数提前**

### 12. js中有哪些内置函数？(也叫对象) 11个

**答案：object,Array,Date,Math,String,Number,Boolean,Error,Function,RegExp(正则表达式),Global,**

### 13.列举出获取日期相关函数

**答案：**

### 14.Math相关函数 写了10个，实际上有很多，

**答案：Math.random() Math.ciel() Math.floor()**

### 15.null和undefined的区别？

**答案：undefined:是访问一个未初始化的变量时返回的值**

**null：是访问一个尚不存在的对象时所返回的值**

### 16.==和===有什么不同？

**答案：== 抽象相等，比较时，会先进行类型转换，然后再比较值**

**=== 严格相等，判断两个值，类型及数值完全相同**

### 17.setTimeout和setInterval的区别是什么？

**答案：setTimeout 一次性定时器，只执行一次**

**setInterval 周期性的定时器，会重复调用**

### 18. 请说出以下代码执行结果

for (var i = 0; i < 3; i++) {

setTimeout(function() {

console.log(i);

}, 0);

console.log(i);

}

**答案：**

### 19. 请说出（true+flase）>2+true的执行结果

**答案：**

### 20. 当前代码块输出结果是什么？

var z=10;

function foo(){console.log(z);}

(function(funArg){var z=20;funArg();})(foo);

**答案**：

### 21. setTimeout(function(){},10)表示什么意思？

**答案**：

## 三、NODEJS

### 1. 同步和异步有何区别？

**答案：同步：指发送请求，需要等待返回，然后再发送下一个请求，同一时间只能执行一个任务，有等待过程。**

**异步：指发送请求时，不需要等待返回，也可发送其它请求，不耽误执行其他任务，即不需要等待。**

### 2. NodeJS中有哪些类型模块，文件操作用哪一个模块？

**答案：**

### 3. 对NodeJS的优点和缺点提出自己的看法？

**答案：****优点：NodeJS设计思想就是以事件驱动，异步，非阻塞I/O密集型为核心，它提供的大多数的API都是基于事件 和异步的风格，所以非常适合处理高并发请求。（意思是同一时间有很多访问）,因此构建在node上的服务器相比其 它技术实现的服务器要好的多，另外，与node服务器交互的前端代码是由js来编写的，客户端与服务器端都用同 一种语言，降低成本。**

**缺点：** 不适合做CPU密集型的应用

### 4. 使用NodeJS完成登录功能(编写HTML页面和路由接口中的代码)？

**答案：**

## 第一阶段月考解析

**4.有数列为：5，4，10，8，15，16，（）。括号中应该填入的数字为（）。**

**解析：**

**奇数次时5的倍数 5\*n ----->20**

**偶数次时2\*（2的n次方）2\*2\*2\*2\*2------>32**

**5.有数列为：2，3，4，9，8，27，16，（）括号中应该填入的数字为（）**

**解析：**

**这是两个数列,分别是2,4,8,16（2的n次方）……及3,9,27(3的n次方)……**

**因此：2,3,4,9,8,27,16（ 81）**

**8.从0、1、2、7、9 五个数字中任选四个不重复的数字，组成的最大四位数和最小四位数的差是多少？**

**解析：**

**9721-1027=8694**

**11.有数列为：1，8，27，（）括号中应该填入的数字为**

**解析：1 = 1^3, 8 = 2^3, 27=3^3, 64= 4^3 （n的3次方）**

**12.甲、乙、丙三人各有一些糖果，甲、乙共有 81 颗，乙、丙共有 99 颗。已知三人中糖果最多的人的糖果数是最少人的2倍，那么丙有（）颗糖果**

**解析：x+y=81**

**y+z=99**

**2x=y ? ---->推出：x=21 y=54 z=45 符合答案选项**

**2y=z ?**

**2x=z ?**

**18.有数列为：1，1，2，3，（），8，13，21，（）。括号中应该填入的数字为**

**解析：斐波拉切数列 ：后一个数是前两个数之和**

**19.有数列为：81，27，9，3，（1）。括号中应该填入的数字为**

**解析：3的n次方 3的0次方为1**

**22.有数列为：3，5，11，29，（83）。括号中应该填入的数字为**

**解析：前一个数加上 2\*3的n次方**

**3+2\*3^0=5**

**5+2\*3^1=11**

**11+2\*3^2=29**

**29+2\*3^3=83**

**23.一个整数除以2余1，用所得的商除以5余4，再用所得的商除以6余1。用这个整数除以60，余数是多少？**

**解析：**

**设这个整数为x，由题意可知，它除以2余1，**

**所以设x=2a+1；**

**又a除以5余4，所以设a=5b+4；**

**又b除以6余1，所以设b=6c+1；**

**即a=5b+4=5（6c+1）+4=30c+9；**

**所以x=2a+1=2（30c+9）+1=60c+19；**

**所以这个整数除以60，商为c，余数是19．**

**26.有数列为：1.6，2.12，4.24，（8.48），16.96。括号中应该填入的数字为**

**解析： 1.6 --->1和6分别乘以2得到后一个数，依次类推。**

**28.一袋白、蓝、红球共 157 个，其中蓝球比红球的2倍多3个，白球比蓝球和红球的总和还多37个。问蓝球有多少个？**

**解析：**

**x+y+z=157**

**2z+3=y**

**y+z+37=x**

**解出：篮球41**

**30.有三个自然数,分别是一位数,两位数和三位数,这三个数的乘积为1212,则三个数之和**

**解析：**

**1\*12\*101=1212**

**1+12+101=114**

**38.有数列为：2，6，13，39，15，45，23，（）。括号中应该填入的数字为**

**解析：**

**因为这道题中2和6是3倍的关系，13和39是3倍的关系，15和45也是3倍的关系，那么23的3倍就是69了**

**40.有2匹马、3头牛和4只羊，它们各自的总价都不满10000文钱（古时的货币单位）。如果2匹马加上1头牛，或者3 头牛加上1只羊，或者4只羊加上1匹马，那么它们各自的总价都正好是10000文钱了。问：马的单价是多少钱？**

**解析：**

**设马的单价是x,牛的单价是y,羊的单价是z**

**2x+y=10000……(1)**

**3y+z=10000……(2)**

**4z+x=10000……(3)**

**解出 x=3600 y=2800 z=1600**

# 第二阶段面试题解析

## 一、HTML

### 1. img属于什么类型的元素？有什么特点？

答案：行内元素，显示方式是行内元素，但是可以修改尺寸(设置高宽)

### 2. 结构标记有哪些 与div的相同不同？

答案：header footer aside section article(文字布局) nav

相同：做网页布局

不同点：具有语义性，有利于阅读和维护代码

### 3. 行内元素，块元素，空元素有哪些

答案： 行内：img span a b i u ...

块级：div h1~h6 p ul ol dl 结构标记 pre

空 ：br hr

### 4什么是前端？

答案：用HTML、CSS、JS以及更多的框架技术，编写网站页面，App应用，微信小程序，游戏等应用，通过内容设计，交互动画，数据操作构建项目与用户的交互界面。前端工程师与UI设计师和后台工程师合作。实现设计意图并调用数据接口，也可以独立工作完成强用户交互。

## 二、CSS3

### 0. 框模型默认的计算方式是什么？要改变默认的计算方式用哪个属性及值？

答案：W3C标准：实际占地宽度=左右外边距+左右边框+左右内边距+width

实际占地高度=上下外边距+上下边框+上下内边距+height

box-sizing:border-box;

### 1. 隐藏网页中的元素有几种方式？有什么区别？

答案：display:none; 脱离文档流 不能使用onclick类的事件

visibility:hidden; 不脱离文档流

事件？绑定后由于不可见，不可以被选择，故无法点击

opacity:0; 不脱离文档流 可以使用onclick类的事件

### 2. 如何解决子元素上外边距溢出？

答案：内容生成 :before

在元素的第一个子元素处添加空的table

### 3. 常用设置透明度的方式

答案：opacity: 所有相关颜色全部随着变化

rgba(): 当前某一种颜色的透明度

### 4. 浮动会对父元素的高度带来什么影响？如何解决？

答案：高度坍塌

内容生成:after display:block clear:both

直接设置父元素的高度

直接添加一个空的块级

overflow:hidden (/auto ?)

父元素也浮动

### 5. 哪些元素默认有外边距？

答案：ul ol p h1~h6 body pre(预格式化)

### 6. transition 和 animation的区别?

答案：transition 过渡 一个状态到另一个状态从from到to

需要触发才能改变样式效果

animation 动画 很多帧(状态)，可以理解一个动画由多个transition构成；

不需要触发，也可以设置自动播放

### 7. 如何实现元素水平垂直居中

答案：1.flex justify-content:center; align-items:center;

2.定位+margin (50%,50%)+调整

3.位移+transform (50%,50%)+调整

### 7.子元素在父元素中居中显示有几中方法？

1.父元素设置为：position: relative;

子元素设置为：position: absolute;

left:50%;top:50%，

teansform：translate(50%,50%)

2.display: flex;//flex布局

justify-content: center;//使子项目水平居中

align-items: center;//使子项目垂直居中

3.display:flex//父元素

margin:auto//子元素

4.display: table-cell;//父元素

vertical-align: middle;//父元素

margin:auto;//子元素

### 8. 列举改变页面元素位置的定位方式和他们的区别？

答案：浮动：实现多个块级元素在一行中显示

绝对定位：可以实现内容的堆叠

(离他最近的已定位元素实现位置的偏移，如果没有则相对于body做初始化)

相对定位：实现元素自身位置的偏移

固定定位：将元素固定在可视区域，不随页面移动

### 9.实现时针转动效果需要用到CSS中哪些关键技术？

答案： 倒角，绝对定位，动画

### 10.设置行内元素居中对齐用什么属性？

答案： text-align:center;

### 11.如果要将表单中控件的值提交给服务器必须为控件设置什么属性？

答案：name和value

### 12. 盒子模型你是怎么理解的？

答案：元素框处理元素内容，内边距，外边距以及边框的计算方式。

盒子模型分为两种：W3C和IE标准

### 11. src和href的区别？

答案：href:是Hypertext Reference的缩写，表示超文本的引用。用来建立当前元素与文档之间的链接。并行下载该文档，不会停止对当前文档的处理。

src:是source的缩写，src指向的内容会嵌入到文档当前标签的位置，如img，script,iframe当浏览器解析到该元素时，会暂停浏览器的渲染，直到该资源加载完毕。

### 12. 说出link和import的区别？

答案：link一般在html头部引入外部样式。

import一般定义在CSS文件内部，并且一定要在其他所有规则之前，也就是一般写在文件的头部，并且专门拿来链接CSS的。

加载顺序：import比link延迟一些

## 三、AJAX

### 1. post方式异步向服务器提交数据时，需要发送前设置什么？提交的数据放在什么位置？

答案：设置消息头content-type

xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

放在send(form-data);

### 2. 什么是异步的数据加载？

答案：浏览器在向服务器发送请求时，不耽误用户在网页中做其他的事情

以无刷新的效果来更改页面局部的内容

### 3. 异步请求数据的步骤？

答案：创建异步对象，绑定监听事件，创建请求，发送请求

1).创建异步对象

var xhr=new XMLHttpRequest();

这种创建方式，不兼容IE8以下的版本

以下方式兼容IE8以下，但是很少用

if(window.XMLHttpRequest){

//如果有这个属性，说明是IE8以上的浏览器

var xhr=new XMLHttpRequest();

}else {

//如果没有该属性，说明是IE8以下的浏览器

var xhr=new ActiveXObject("Microsoft.XMLHttp")

}

### 4. 异步请求中满足什么条件才能取出响应的结果数据？

答案：1.xhr.readyState 属性

用于表示xhr对象的请求状态

一个有5个状态：(值变化4次)

0：请求尚未初始化

1：已经打开连接，请求正在发送中

2：接收响应头

3：接收响应主体

4：接收响应数据成功

2.xhr.onreadystatechange 监听事件

监听xhr.readyState值的改变:每改变一次,方法会调用一次;一共会调用4次

注意：只有当xhr.readyState==4的时候，是我们要接收响应的正确时刻

3.xhr.status 响应状态码

只有当响应状态码是200的时候，我们才接收响应

4.响应数据放在了xhr.responseText属性中

完整的写法：

xhr.onreadystatechange=function() {

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

var result=xhr.responseText;

console.log(result);

}

}

### 5. 如何理解json？

答案：json javascript object notation js对象表示法

即以js对象的数据格式表现出来的一个字符串

专门负责承载数据用的

JSON.parse 将json串转换成对象/数组

JSON.stringify 反过来转换成json

### 6. 列举常见的3和4开头的状态码？

答案：404 找不到资源文件

403 服务器禁止访问

400 请求方法不被允许

304 命中缓存 请求未被修改

302 响应重定向/临时

301 永久重定向 www.jd.com=>https://www.jd.com

### 7.http和https的区别？

答案：http 未加密的传输协议 ，明文传递

https 加密协议，提高安全性。

## 四. Bootstrap

### 1.用bootstrap中的栅格布局基本结构？默认一行多少列？

答案：div.container div>row>div.col-n

默认一行12列

### 2.如何编写响应式网页？

答案：1.媒体查询 @media screen and (...) {...}

2.viewport 设置视口

<meta name="viewport" content="width=device-width,initial-scale=1.0>

3.流式布局

float+flex 流式布局+弹性布局+媒体查询组成了响应式布局

4. 所有内容，包括文字/图片 使用相对尺寸 rem em...

### 3.同时监听多个scss文件的命令？

答案：node-sass -w scss文件夹 -o css文件夹

### 4. 在scss中如何定义混合器和引用？

答案：@mixin 混合器名称{...}

@include 混合器名称

1.一般混合器

1.定义混合器

@mixin 混合器名称{css代码}

2.调用混合器

选择器{@include 混合器名称;}

2.带参数的混合器

1.定义混合器

@mixin 混合器名称(参数1,参数2...){css代码含参数}

2.调用混合器

选择器{@include 混合器名称(实参1,实参2...);}

3.混合器使用的场合：css-hack

# 第三阶段

## 1.正则表达式

### 1.用正则表达式写微信的敏感词

答：(微|w(ei)?)\s\*(信|x(in)?)

第一个字：(微|w(ei)?) wei可以看做是一个整体ei是可有可无的，所有这里用？

中间：\s\* 可能有，也可能不只一个所以在这里用\*

第二个字：(信|x(in)?)

\*可有可无，多了不限 ？是可有可无，最多一次 +到少一次，多了不限

### 2.用正则表达式写邮箱格式(笔试题)

答：

### 3.查找敏感词的有几种方法？

String中的正则表达式：查找敏感词的四种方法

答：1.查找一个固定的敏感词

2.模糊查找一个敏感词的位置

3.查找敏感词内容(两种)

1>只查找一个敏感词的内容

2>查找所有敏感词的内容

### 1.高级替换

var str=" zhang dong "

//1.删除左边的空格

function left\_trim(str){

return str.replace(/^\s+/,"");

}

console.log(left\_trim(str))

//2.删除右边的空格

function right\_trim(str){

return str.replace(/\s+$/,"");

}

console.log(right\_trim(str));

//3.删除前后的空格 （+g）

function trim(str){

return str.replace(/^\s+|\s+$/g,"");

}

console.log(trim(str));

1.什么是闭包？

答：即重用一个变量，又保护变量不被篡改。

2.函数的创建方式？

答：1.声明方式 声明变提前，赋值留在原地；

2.用赋值的方式 仅函数名提前，函数体留在原地。在这里揭示了两个秘密：1.函数名本质就是一个普通的变量。2.而函数定义其实是一个对象(独立的存储空间)

3.用new创建 调用函数等效于执行函数中的代码段，如果不调用，是不会执行函数的内容的。

3.js有几种作用域，什么是作用域链？

答：js有两种作用域：1.全局作用域。2.函数作用域。

作用域链：是由多级作用域组成的链式内存结构。

4.鄙视题 闭包

function fun(){

for(var i=0,arr=[];i<3;i++){

arr[i]=function(){ console.log(i); }

}

return arr;

}

var funs=fun();

funs[0](); //3

funs[1](); //3

funs[2](); //3

## 2019/6/4

1.var number=2;

var obj={

number:4,

fnl:(function(){

this.number\*=2;

number\*=2;

var number=3

retrun function(){

this.number\*=2

number\*=3

console.log(number)

}

})

( )

}

## 2019/6/5

1.var a=0,b=0

function A(a){

A=function(b){console.log(a+b++)}

console.log(a)

}

A(1); //1

A(12); //13

A(100); //101

2. var m=true;

setTimeout(function(){m=false},3000);

while(m){}alert("ok"); //请问多久能弹出ok

3. var i=3;

function fn(){

i\*=2;

return function(n){console.log(n\*(++i))}

}

var f=fn()

f(3) //21

fn()(3) //45

f(4) //64

fn()(3) //99

# ECMAScipt核心

一.ECMAScipt核心  
--------------------------

//0.javascript类型  
  
--   
//'string' 'number' 'object' 'function' 'boolean' 'undefined'   
--js七种数据类型  
//String Object null undefined Array Boolean Number  
--复合数据类型  
//function object  
  
//1.JavaScript 变量可以是( AD )   
A.局部变量   
B.动态变量   
C.全局变量   
D.形参量  
//JavaScript中没有定义动态变量,形参量; JavaScript 变量可以是局部变量或全局变量.  
  
//2.使用JavaScript可以实现以下哪些操作（ ABCD ）  
A.网页特殊显示效果,动画效果等的制作  
B.实现用户表单的输入验证   
C.浏览器事件的触发和处理   
D.异步提交服务器数据  
  
//3.语句var arr=[a,b,c,d];执行后,数组arr中每项都是一个整数,下面得到其中最大整数语句正确的是哪几项( BCD )  
A.Math.max(arr)   
B.Math.max(arr[0], arr[1], arr[2], arr[3])   
C.Math.max.call(Math, arr[0], arr[1], arr[2], arr[3])   
D.Math.max.apply(Math,arr)  
  
//4.关于数组的说法正确的是（ ABC ）  
//A.数组是引用类型的对象，变量中保存的是一个空间地址，而不是数组实际的值   
//B.想要获取数组中的某个元素，可以通过中括号配合下标进行获取   
//C.通过length属性可以获取数组中的元素个数   
//D.arr[arr.length]的值是数组中最后一个元素的值 arr[arr.length-1]是最后一个值  
  
//5.下面属于javascript基本数据类型的有( ABCD )  
A.字符串   
B.数字 (数值型)  
C.null   
D.undefined   
  
//6.1对于slice说法正确的是  
// slice可以进行数组元素的截取   
// slice中的参数如果只有一位，表示从当前参数位置截取到数组结尾  
// slice()方法可提取字符串的某个部分,并以新的字符串返回被提取的部分  
// 所有主流浏览器都支持slice()  
//6.2对于slice说法不正确的是  
// slice中的参数不可以是负数  
// slice直接修改原数组   
// slice() 方法可从已有的数组中删除选定的元素  
// slice() 方法会改变原始数组  
  
//7.以下哪种方式可以创建3个空元素的数组（ BCD ）  
//A.var arr1=[ 3 ];   
//B.var arr2=["","",""]   
//C.var arr3=new Array(3);   
//D.var arr=new Array("","","");   
  
//8.javascript中跳出循环的关键字有( ABC )  
A.continue //在for循环中常用  
B.return //在function中常用  
C.break //在switch-case中常用  
D.each //不是  
  
//9.给网页添加javascript的方式有( ABC )  
//A.使用script标签,将javascript代码写到<script></script>之间   
//B.添加外部javascript文件   
//C.使用行内javascript   
//D.使用@import引入javascript文件   
  
//10.以下哪些选项可以体现JavaScript是弱类型语言的特点（ ACD ）\*B选项体现的是JavaScript语言无需编译的特点  
//A.声明变量时，不需要提前指定变量的数据类型   
//B.JavaScript语言边读取边执行，后读取的相同内容会覆盖先前读取的内容   
//C.同一个变量，先后可保存不同类型的数据   
//D.在运算时，程序会根据自身需要，隐式转换数据类型  
  
//11.(多选题)关于while循环和do-while循环，说法正确的是（ ACD ）  
//A.while和do-while循环一般适用于变量变化没有规律的循环  
//B.while和do-while的执行效果是一样的   
//C.如果第一次执行条件都不满足的话，仍想循环的话，我们可以使用do-while 结构   
//D.如果第一次条件满足，while和do-while的执行效果是相同的   
  
//12.下面代码有错误的是( ABCD )   
//JavaScript是对大小写敏感的,变量名,属性,函数,关键字都要区分大小写.所以AC打印的A跟a是不同的变量.BD的innerHTML属性写错.  
//A.var a=''1"; console.log(A);   
//B.document.getElementById("div").innerHtml   
//C.var A="2" console.log(a);   
//D.document.getElementById("Div").innerhtml   
  
//13.解释语言的特性有( AB )  
A.非独立   
B.效率低   
C.独立   
D.效率高性   
//解释性语言和编译性语言的定义: 计算机不能直接理解高级语言,只能直接理解机器语言,所以必须要把高级语言翻译成机器语言,计算机才能执行高级语言编写的程序. 翻译的方式有两种,一个是编译,一个是解释.两种方式只是翻译的时间不同. 解释性语言的定义: 解释性语言的程序不需要编译,在运行程序的时候才翻译,每个语句都是执行的时候才翻译.这样解释性语言每执行一次就需要逐行翻译一次,效率比较低. 现代解释性语言通常把源程序编译成中间代码,然后用解释器把中间代码一条条翻译成目标机器代码,一条条执行. 编译性语言的定义: 编译性语言写的程序在被执行之前,需要一个专门的编译过程,把程序编译成为机器语言的文件,比如exe文件,以后要运行的话就不用重新翻译了,直接使用编译的结果就行了(exe文件),因为翻译只做了一次,运行时不需要翻译,所以编译型语言的程序执行效率高.所以选AB  
  
//14.以下哪些值转为布尔值为false（ ABCD ）  
A.空字符串   
B.undefined   
C.null   
D.NaN   
//只有0，”“（空字符串），NaN，null，undefined这5个值转为布尔值时为false，其余值都转为true  
  
//15.以下有关关系运算的说法正确的是（ CD ）  
//A.“=”，“==”，“===”，三者都可以用来判断两值是否相等   
//B.“==”和“===“检测机制是一样的   
//C.”===“，叫做严格相等运算符，不允许进行类型转换   
//D.”1”==true，表达式的结果是true   
//A：“=”是赋值运算符，“==”是相等运算符，“===”是恒等运算符 B：“==”检测允许进行类型转换，而“===”不允许 C：参考B D：布尔值true首先转为数字1，接着字符串“1”转为数字1，最后结果为true  
  
//16.对于连接数组操作，说法正确的是（ ACD ）  
//A.将两个数组进行连接使用的方法是concat()   
//B.concat直接在原数组上进行修改   
//C.concat可以传入多个参数进行连接   
//D.concat会自动打散参数中的数组，以单个元素形式进行拼接   
//B错误，concat不修改原数组，总返回新数组  
  
//17.关于分支结构，说法正确的是（ ABCD ）  
//A.如果我们想要让程序根据不同的条件执行不同的操作，需要使用到分支结构   
//B.分支结构中的条件都默认转为布尔类型   
//C.如果想要中止并推出当前的分支结构，我们可以配合break一起使用   
//D.在switch...case...结构中，条件表达式和case进行的是“全等”比较   
  
//18.插入 Javacript 的正确位置是 ( ABC )  
//A.<body> 部分   
//B.<head> 部分   
//C.<body> 部分和 <head> 部分均可   
//D.<html>   
//插入 Javacript 的正确位置是<body> 部分和 <head> 部分都可以,~所以选ABC都正确.  
  
//19.关于浏览器内核描述错误的是( CD )  
//A.Presto内核的浏览器:Opera7及以上版本   
//B.Webkit内核的浏览器:Safari、Chrome。   
//C.Gecko内核的浏览器:IE、Maxthon、TT、The World等   
//D.Trident内核的浏览器:Netcape6及以上版本、FireFox、MozillaSuite/SeaMonkey   
//使用Trident内核的浏览器:IE、Maxthon、TT、The World等; 使用Gecko内核的浏览器:Netcape6及以上版本、FireFox、MozillaSuite/SeaMonkey; 使用Presto内核的浏览器:Opera7及以上版本; 使用Webkit内核的浏览器:Safari、Chrome.CD明显写反版本.  
  
//20.以下哪些选项属于JavaScript的特点（ ACD ）  
//A.运行在客户端浏览器或者独立安装的脚本解释引擎中   
//B.预编译语言，先编译后执行 错误：JavaScript不需要预编译，由JS引擎解释执行，边读取边执行  
//C.弱类型语言，在运算时，程序会根据自身需要隐式转换数据类型   
//D.基于对象   
  
//21.属于javascript中逻辑运算符的是( ABC )  
//A.&&   
//B.!   
//C.||   
//D.===   
//逻辑运算符用于测定变量或值之间的逻辑包括:&&、||、!; D属于比较运算符,用于判断全等(值和类型).  
  
//22.1-String 对象属性有( ABD )  
A.constructor   
B.length   
C.substring   
D.prototype   
//没有substring属性; constructor 对创建该对象的函数的引用; length 字符串的长度; prototype 允许您向对象添加属性和方法.  
//22.2-属于Boolean 对象方法的是( ABC )  
A.toSource()   
B.toString()   
C.valueOf()   
D.prototype()  
//属于Boolean 对象方法的是ABD.prototype无此方法, 但是有prototype.是Boolean 对象属性.  
  
  
//23.以下哪种方式可以创建3个空元素的数组（ BCD ）  
//A.var arr1=[ 3 ];   
//B.var arr2=["","",""]   
//C.var arr3=new Array(3);   
//D.var arr=new Array("","","");   
//BCD正确，均能创建3个空元素的数组，而A选项创建的包含一个元素数字3的数组  
  
//24.下列获取数组字符串方法错误的是( ABC )  
//A.var cars = <"Saab", "Volvo", "BMW">;   
//B.var cars = [Saab, Volvo, BMW];   
//C.var cars = ("Saab", "", "BMW");   
//D.var cars = ["Saab:", "", "BMW"];   
//AC..数组定义要用“[” B.数组定义字符串类型要加"双引号".  
  
//25.关于使用字符串方法描述正确的是( ABCD )  
//A.search() 方法 用于检索字符串中指定的子字符串,或检索与正则表达式相匹配的子字符串,并返回子串的起始位置   
//B.replace() 方法 用于在字符串中用一些字符替换另一些字符，或替换一个与正则表达式匹配的子串   
//C.test() 方法用于检测一个字符串是否匹配某个模式，如果字符串中含有匹配的文本,则返回 true,否则返回 false   
//D.exec() 方法用于检索字符串中的正则表达式的匹配   
  
//26.属于String HTML 包装方法的是( BCD )  
A.trim()   
B.big()   
C.bold()   
D.strike()   
//trim()属于String 对象方法. HTML 包装方法返回加入了适当HTML标签的字符串.  
  
//27.如何规避javascript多人开发函数重名问题()  
//首先是通过命名规范,比如根据不同的开发人员实现的功能,在函数名加前缀,虽然函数名看起来复杂,发布的时候还是可以替换,从而优化.   
//还有一种办法是,每个开发人员都把自己的函数封装到类中,然后调用的时候即使函数名相同,但是因为是要类.函数名来调用,所以也减少了重复的可能性.  
  
//28.语句var arr=[a,b,c,d];执行后,数组arr中每项都是一个整数,下面得到其中最大整数语句正确的是哪几项( BCD )  
var arr=[1,2,3,4];  
A.Math.max(arr) //没设置参数 NaN  
B.Math.max(arr[0], arr[1], arr[2], arr[3]) //4  
C.Math.max.call(Math, arr[0], arr[1], arr[2], arr[3]) //4  
D.Math.max.apply(Math,arr) //4  
//因为函数 Math.max(x);的参数是Number类型,可以是小数,整数,正数,负数或者是0,如果不是上面所述类型就会返回NaN.  
  
//29.以下哪个选项是正确的（ AC ）  
A.Number(null); //结果为0 正确  
B.parseInt(true); //结果为1 错误 NaN  
C.parseFloat("99.99px"); //结果为99.99 正确  
D.Number(undefined); //结果为0 错误 NaN  
//B：parseInt会先将括号内的数据转为字符串格式再读取数字，所以B结果应为NaN D：Number(undefined)的结果为NaN  
  
//30.下列哪两项可以创建函数( BC )  
//A.function=myFunction(){……..}   
//B.function myFunction(){…...}   
//C.myfunction = function(){…….}   
//D.myFunction(){….}   
//B 函数声明语法定义函数 C 函数表达式定义函数 函数实际上是对象,函数名是一个指向函数对象的指针,不会与某个函数绑定. 解析器会率先读取函数声明,并使其在执行任何代码之前可用:至于表达式,则必须等到解析器执行到它所在的代码行,才会真正被解释执行.  
  
//31.以下代码运行结果正确的是（ AD ）  
var o={x:1};var p=null;  
A.o&&o.x //=>1   
B.p&&p.x //=>1   
C.o&&o.x //=>null   
D.p&&p.x //=>null   
//根据短路逻辑，o&&o.x中o为真值，因此返回值为o.x，即返回1； p&&p.x中p为假值，因此将p返回，即返回null,而不去计算p.x  
  
//32.对于连接数组操作，说法正确的是（ ACD ）  
//A.将两个数组进行连接使用的方法是concat()   
//B.concat直接在原数组上进行修改   
//C.concat可以传入多个参数进行连接   
//D.concat会自动打散参数中的数组，以单个元素形式进行拼接  
//concat不修改原数组，总返回新数组  
  
//33.关于数据类型转换，说法正确的是（ AC ）  
//A.数据类型转换分隐式转换和强制转换两种   
//B.隐式转换会实际改变数据的值   
//C.将一个值隐式转为数字，调用的是Number()函数   
//D.可以使用toString()将null转为字符串   
//B：隐式转换是临时性的操作，不会实际改变数据的值，而强制转换是永久操作，一旦执行，数据的值就被替换为转换后的结果 D：null和undefined没有toString()方法  
  
//34.循环结构的三要素指的是（ BCD ）  
A.循环结构   
B.循环条件   
C.循环变量   
D.循环体   
//循环结构的三要素指的是 ①循环条件：允许循环再次进入循环的条件 ②循环变量：循环条件中用作比较的变量 ③循环体：要反复执行的同一段代码  
  
//35.在 JavaScript 中,关于使用以下条件语句说法正确的是( ABCD )  
//A.if 语句 - 只有当指定条件为 true 时,使用该语句来执行代码   
//B.if...else 语句 - 当条件为 true 时执行代码,当条件为 false 时执行其他代码   
//C.if...else if....else 语句 - 使用该语句来选择多个代码块之一来执行   
//D.switch 语句 - 使用该语句来选择多个代码块之一来执行   
  
//36.声明变量var hello = "hello",以下那种方法可以使我们正确将其输出（ BD ）  
A.alert( Hello );   
B.document.write( hello );   
C.console.log ( HELLO );   
D.console.log ( hello );   
//JavaScript是区分大小写的语言。  
  
//37.对以下函数说法正确的是（ ABCD ）  
function fun(x){ if(x<=1) return 1; return x\*fun(x-1); }  
A.该函数的函数名是fun   
B.该函数可以用来计算阶乘   
C.该函数是递归函数   
D.该函数有局部变量   
//本题函数是个计算阶乘的递归函数（即调用自身的函数），函数参数x即函数内的局部变量.  
  
//------------------  
//二.ES6 新特性(了解)  
//------------------  
//1.JavaScript只有indexOf方法,可以用来确定一个字符串是否包含在另一个字符串中.ES6新特性中有( ABD )新方法  
A.includes()   
B.startsWith()   
C.repeat()   
D.endsWith()   
//repeat方法返回一个新字符串,表示将原字符串重复n次 includes()：返回布尔值,表示是否找到了参数字符串; startsWith()：返回布尔值,表示参数字符串是否在源字符串的头部; endsWith()：返回布尔值,表示参数字符串是否在源字符串的尾部.  
  
//2.extends后面可以接驳任意合法且拥有prototype属性的构造函数,它可以是( ABCD )  
A.一个函数调用   
B.一个对象上的属性访问   
C.一个包含一个函数或类的变量   
D.一个普通的函数   
//extends后面可以接驳任意合法且拥有prototype属性的构造函数,它可以是: 另一个类 源自现有继承框架（译者注：作者指的是原型继承，即使在JavaScript中类继承的本质也是原型继承）的近类函数 一个普通的函数 一个包含一个函数或类的变量 一个对象上的属性访问 一个函数调用.  
  
//3.var foo = 'bar'; var baz = {foo}; baz // {foo: "bar"} // 等同于 var baz = {foo: foo}; 上述代码中,以下说法错误的是( BD )  
A.ES6允许直接写入变量和函数,作为对象的属性和方法   
B.ES6不允许在对象之中,不写属性值   
C.ES6允许在对象之中,只写属性名,不写属性值   
D.以上说法均有误  
//ES6允许直接写入变量和函数,作为对象的属性和方法.这样的书写更加简洁,上面代码表明,ES6允许在对象之中,只写属性名,不写属性值.这时,属性值等于属性名所代表的变量.  
  
//4.查下如下代码,说法正确的是：  
class ColorPoint extends Point {   
 constructor(x, y, color) { super(x, y); // 调用父类的constructor(x, y)   
 this.color = color; } toString() {   
 return this.color + ' ' + super.toString(); // 调用父类的toString()  
 } }  
A.constructor方法和toString方法之中,都出现了super关键字,它在这里表示父类的构造函数   
B.constructor方法和toString方法之中,都出现了super关键字,用来新建父类的this对象   
C.子类必须在constructor方法中调用super方法,否则新建实例时会报错   
D.如果不调用super方法,子类就得不到this对象   
//以上说法均正确.  
  
//5.ES6规范定义了一个全新的全局构造函数:代理(Proxy),它可以接受两个参数( BC )  
//A.事件对象（Event）   
//B.目标对象（target） √  
//C.句柄对象（handler） √  
//D.节点对象（document）   
//ES6规范定义了一个全新的全局构造函数:代理（Proxy）,它可以接受两个参数:目标对象（target）与句柄对象（handler）示例: var target = {}, handler = {}; var proxy = new Proxy(target, handler);  
  
//6.ES6推荐设置原型对象的方法有( AC )  
A.Object.setPrototypeOf //√  
B.proto\_\_   
C.\_\_proto\_\_ //√  
D.Object.setPrototypesOf  
//Object.setPrototypeOf方法的作用与\_\_proto\_\_相同,用来设置一个对象的prototype对象.它是ES6正式推荐的设置原型对象的方法.  
  
//7.ES6 新特性中修饰器只能用于( AB )  
A.类的方法 //√  
B.类 //√  
C.类的函数   
D.函数   
//修饰器只能用于类和类的方法,不能用于函数,因为存在函数提升.  
  
//8.ES6 新特性中严格模式有哪些限制( ABCD )  
//A.变量必须声明后再使用 函数的参数不能有同名属性,否则报错 不能使用with语句 不能对只读属性赋值,否则报错   
//B.不能使用前缀0表示八进制数,否则报错 不能删除不可删除的属性,否则报错 不能删除变量delete prop,会报错,只能删除属性delete global[prop]  
//C.eval不会在它的外层作用域引入变量 eval和arguments不能被重新赋值 arguments不会自动反映函数参数的变化 不能使用arguments.callee 不能使用arguments.caller   
//D.禁止this指向全局对象 不能使用fn.caller和fn.arguments获取函数调用的堆栈 增加了保留字（比如protected、static和interface）  
//面这些限制,模块都必须遵守.  
  
//9.是function的简写形式,支持哪( AC )种形式  
A.statement //√  
B.this   
C.expression //√  
D.null   
//=> 是function的简写形式,支持expression 和 statement 两种形式.同时一点很重要的是它拥有词法作用域的this值,帮你很好的解决this的指向问题,这是一个很酷的方式,可以帮你减少一些代码的编写,语法: ([param] [, param]) => { statements } param => expression,  
  
10.在ES6中,模块的功能主要由( AC )组成  
A.export //√  
B.name   
C.import //√  
D.url   
到目前为止,javascript (ES5及以前) 还不能支持原生的模块化,大多数的解决方案都是通过引用外部的库来实现模块化.比如 遵循CMD规范的 Seajs 和AMD的 RequireJS.在ES6中,模块将作为重要的组成部分被添加进来.模块的功能主要由 export 和 import 组成.每一个模块都有自己单独的作用域,模块之间的相互调用关系是通过 export 来规定模块对外暴露的接口,通过import来引用其它模块提供的接口.同时还为模块创造了命名空间,防止函数的命名冲突.

# 获得dom元素宽高的方法

//来自百度, 供参考:

//关于页面don元素的宽高\*

// 第一种情况就是宽高都写在样式表里，就比如#div1{width:120px;}。

// 这种情况通过#div1.style.width拿不到宽度，

// 而通过#div1.offsetWidth才可以获取到宽度。

// 第二种情况就是宽和高是写在行内中，比如style="width:120px;"，

// 这种情况通过上述2个方法都能拿到宽度。

//其他:

//网页可见区域宽:

document.body.clientWidth;

//网页可见区域高:

document.body.clientHeight;

//网页可见区域宽:(包含边线)

document.body.offsetWidth;

//网页可见区域高:(包含边线)

document.body.offsetHeight;

//网页正文全文宽:

document.body.scrollWidth;

//网页正文全文高:

document.body.scrollHeight;

//网页被卷去的高:

document.body.scrollTop;

//网页被卷去的左:

document.body.scrollLeft;

//网页正文部分上:

window.screenTop;

//网页正文部分左:

window.screenLeft;

//屏幕分辨率的高:

window.screen.height;

//屏幕分辨率的宽:

window.screen.width;

//屏幕可用工作区的高度:

window.screen.availHeight;

//屏幕可用工作区的宽度:

window.screen.availWidth;

//HTML精准定位:scrollLeft scrollWidth clientWidth offsetWidth

scrollHeight; //获取对象的滚动高度

scrollWidth; //获取对象的滚动宽度

scrollLeft; //设置或获取位于对象左边界的和窗口中目前可见内容的最左端之间的距离

scrollTop; //设置或获取位于对象最顶端和窗口中可见内容的最顶端之间的距离

offsetHeight; //获取对象相对于版面或由offsetParent属性指定的父坐标的高度

offsetWidth; //获取对象相对于版面或由offsetParent属性指定的父坐标的宽度

offsetLeft; //获取对象相对于版面或由offsetLeft属性指定的父坐标的计算机左侧位置

offsetTop; //获取对象相对于版面或由offsetTop属性指定的父坐标的计算机顶端位置

event.clientX; //相对文档的水平坐标

event.clientY; //相对文档的垂直坐标

event.offsetX; //相对容器的水平坐标

event.offsetY; //相对容器的垂直坐标

document.documentElement.scrollTop; //垂直方向滚动的值

event.clientX+document.documentElement.scrollTop; //相对文档的水平坐标+垂直方向滚动的量

# JavaScript导图

## 1.什么是

## 2.变量

1.什么是变量？

内存中存储\*一个\*数据的存储空间，再起一个名字

可以重用一个数据

只要一个数据可能被反复使用时，都要保存在变量中

2.什么是声明一个变量？

在内存中创建一块存储空间，再起一个名字

任何变量在使用前，必须先声明

3.变量名的命名规则？

1.字母，数字，下划线和$组成，但不能以数字开头

2.不能使用保留字

3.见名知义

4.驼峰命名

5.默认值，仅声明，未赋值的变量，默认是undefined

6.简写 var 变量1，变量2....

4.什么是常量？

一旦创建，值不能改变的量

只要程序中的一个值，任何情况下都不能改变

const常量名=值;

强调：常量名必须全大写，创建常量时必须立该赋值。

5.强行修改常量的值？

1.在最新的Chrome浏览器中，已经内置了对常量赋值的检查。

2.在ES3标准中，普通运行模式下，强行给常量赋值，不报错，也不让修改，静默失败，

解决方法：启用严格模式。

要求：将所有静默失败升级为错误！

## 3.数据类型

1.什么是数据类型？

数据在内存中的存储格式

不同类型的数据，可执行的操作不一样

不同的操作，要求专门的存储结构支持

2.数据类型分几类？

1.原始类型：值直接保存在变量本地的数据类型

unmber string boolean null undefined

2.引用类型：值无法保存在变量本地的复杂数据类型，如果一个复杂的数据结构，同时保存多个数据，就不能直接保存在变量中，存储在window之外的一块独立存储空间，独立存储空间中可同时存储多个数据，拥有唯一的地址，变量名仅保存存储空间的地址。

3.什么是number？

专门存储数字(即包含整数，也包含小数)的类型

如果一个数值可能需要比较大小和进行数学计算时用number

程序中只要不加引号的数字，自动就是number类型。

底层都是二进制存储：n.toString(2)

存储方式：整数占4字节，浮点数占8字节，存储空间和数值大小无关。

4.什么是string?

专门存储一串用于显示的文字

用于记录一串文字，不用做比较和计算。

凡是加引号的一串字符直接称为字符串类型。

存储方式：

unicode对全球主要语言的每个字编一个号

因为计算机不认识字符，只认识数字

范围：

“0”-“9” 48-47

“A”-“Z” 65-90

"a"-"z" 97-122

汉字 \u4e00-\u9fa5

每个字母/数字字符，占1字节

每个汉字，占2字节

其他：utf-8 字母数字1字节，汉字3字节

GBK 字母数字1字节，汉字2字节

5.boolean

专门表示判断结果的，只要作为判断条件的结论时，就用boolaean.

只有两个值 true false

6.undefined 空

专门由程序自动为一个变量赋值，程序员很少主动使用

7.null 空

专门给程序员用于手动清空一个变量之用。

## 4.数据类型转换

## 5.运算符和表达式

## 6.函数

## 7.全局函数

## 8.分支结构

## 9.循环结构

## 10.数组

什么是数组：内存中连续存储多个数据的存储空间，再起一个名字

为什么要用数组：连续存储的一组数据，可极大提高程序的执行效率，便于维护和查找

何时：只要存储多个数据都要用数组

### 1)创建

1. 创建空数组：

何时：如果创建数组时，暂时不知道数组的内容

数组直接量：var arr=[]

用new var arr=new Array();

2. 创建数组同时初始化数组元素

何时：如果创建数组时，已经知道数组内容

数组直接量：var arr=[值1，值2,.....]

用new var arr=new Array(值1,值2,....);

3. 创建n个空元素的数组

何时：如果创建数组时只知道数组的元素个数，暂时不知道内容时

用new var arr=new Array(n)

### 2)访问

一、元素：保存在数组中的一个数据

二、下标：数组中唯一标识元素存储位置的序号

默认从0开始，依次递增，连续不重复，到length-1结束

三、arr[i]

单个元素的用法和单个变量完全一样

数组其实是一组变量的集合，再起一个统一的变量名

三个不限制：

1.不限制元素的数据类型

2.不限制下标越界：

取值：不报错，返回undefined; 赋值：不报错，自动在指定位置创建新元素

稀疏数组：下标不连续的数组。

自动将length调整到最大下标+1

3.不限制元素个数

可随时在任意位置添加新元素

添加新元素后，都会自动改变length属性为最大数字下标+1

### 3).length属性

规定了数组理论上的元素个数

始终等于最大下标+1，自动维护

固定套路：

获取最后一个元素：arr[arr.length-1]

获取倒数第n个元素：arr[arr.length-n]

末尾追加一个新元素：arr[arr.length]=值

缩容：

删除末尾一个元素：arr.length--

删除末尾n个元素：arr.length-=n

清空数组：arr.length=0

### 4)数组是引用类型

不直接存储在变量本地，一个变量只能存1个值

实际存储在变量之外，每个数组都有一个唯一的地址值，变量中只保存数组的地址值，也称变量引用 着数组

按值传递：两个变量间赋值或将变量传给函数参数时，其实只是将原变量中的值复制一个副本给对方

原始类型：修改新变量，不影响原变量

引用类型：通过新变量修改数组，同样会影响原变量

为什么：将原变量中的地址值复制给新变量，结果：两个变量引用同一个数组。

### 5)遍历

什么是：依次取出数组中每个元素的值，执行相同的操作

何时：只要对数组中每个元素执行相同操作时

如何：索引数组

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

arr[i] //当前数组元素

}

### 6)关联数组

什么是关联数组：可自定义下标名称的数组，VS索引数组：下标都是数字的数组

为什么：索引数组的数字下标没有意义，只能通过遍历查找指定的元素，查找速度受数组元素个数和元素位置的影响

何时：希望通过下标名称快速查找某个元素时，无需遍历，不受元素个数和元素存储位置的影响。

创建(2步)

创建空数组：var hash=[]

向数组中添加新元素：hash["下标名(key)"]=值(value)

访问

hash["下标名(key)"]

用法同访问索引数组中的元素

特点：

.length属性始终为0

无法使用索引数组的API

遍历

for(var key in hash){

key //仅获取当前下标名称

hash[key] //获取当前元素值

}

in依次取出关联数组中每个下标名称保存在变量key

固定套路

仅获取hash中的所有key

var keys=[];

var i=0;

for(keys[i++] in hash);

//结束后: keys中保存了hash的所有key

### 7)数组API

1. 转字符串

String(arr)

将arr中每个元素转为字符串，用逗号链接(拍照)

arr.join("连接符")

将arr中每个元素转为字符串，可自定义连接符

固定套路

无缝拼接：arr.join("")

判断空数组：arr.join("")===""

动态生成页面元素：

var html=“<ANY>”+arr.join("</ANY><ANY>")+"</ANY>"

elem.innerHTML=html;

2. 强调： 都无权修改原数组，只能返回新数组，必须用变量接住返回值

拼接

var newArr=arr1.concat(值1,值2,arr2,....)

将值1,值2,arr2中的每个元素，拼接到arr1结尾

强调:可打散数组类型参数

选取：

var subArr=arr.slice(starti,endi+1);

选取arr中starti位置到endi位置的元素，组成新数组返回

强调:凡是两个参数都是下标的API，含头不含尾

简写

负数参数：倒数第n个，本质：自动执行length-n

省略第二个参数：从starti一直选取到结尾

省略全部两个参数：复制整个数组

用途：将类数组对象转化为数组对象

Array.prototype.slice.call(arguments)

相当于：arguments.slice()

固定套路:

获得i位置开始的n个元素：arr.slice(i,i+n)

3. 删除元素：

arr.splice(starti,n)

删除arr中starti位置开始的n个元素

强调:直接修改原数组，不用考虑含头不含尾

简写：

支持负数参数，表示倒数第n个

省略第二个参数，删除starti位置后所有元素

其实有返回值

返回被删除的元素组成的临时数组

var deletes=arr.splice(starti,n)

插入元素：

arr.splice(starti,0,值1,值2,...)

在arr中starti位置插入新值，原starti位置的值及其之后的值被向后顺移

强调:不支持打散数组类型参数，如果插入子数组，会变成二维数组

替换：

arr.splice(starti,n,值1,值2...)

先删除arr中starti位置的n个元素，再在starti位置插入新元素

强调:删除的元素个数不一定和插入的元素个数一致

固定套路：广告轮播

移除开头的n个元素拼到结尾：imgs=imgs.concat(imgs.splice(0,n))

移除结尾的n个元素拼到开头：imgs=imgs.splice(-n).concat(imgs)

4. 翻转：arr.reverse()

5. 手写排序:冒泡，快速，插入

冒泡：依次比较相邻两个数，如果前数>后数，就交换两个数的位置

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

for(var i=0;i<arr.length-r;i++){

if(arr[i]>arr[i+1]){

arr[i]=[arr[i+1],arr[i+1]=arr[i]][0];

}

}

}

排序 arr.sort()

将arr中每个元素转为字符串，再按字符串升序排列

问题：只能按字符串升序排列

解决:

自定义比较器函数(2步)

1. 定义比较器函数：专门比较两个值大小的函数

2个要求

1.两个参数a,b

2.返回值

a>b,返回正数

a<b,返回负数

a==b,返回0

比如：

最简单的数字升序比较器function compare(a,b){return a-b;}

2. 将比较器函数作为对象传入sort方法

arr.sort(compare)

强调:回调函数

将一个函数，作为对象传入另一个函数内，被另一个函数使用

传入回调函数时，不要加()

因为不是立刻调用！也不是只调用一次！而是交给别人去调用

其实都会简写为: arr.sort(function(a,b){return a-b;})

ES6:

arr.sort((a,b)=>a-b)

降序排列

颠倒比较器函数返回值的正负号

比如：

数字降序比较器function compare(a,b){return b-a;}

6. 栈和队列

何时：只要按照顺序使用数组中的元素时

js中没有专门的栈和队列的类型，都是用普通数组模拟的

栈stack

一端封闭只能从另一端进出的数组

FILO

何时：希望始终使用最新进入数组的元素时

如何：

结尾出入栈：

入：arr.push(值) 结尾入；arr[arr.length]=值

出：var last=arr.pop()

开头出入栈：

入：arr.unshift(值)

arr.splice(0,0,值)

出：var first=arr.shift() 开头出

强调:开头入栈和结尾入栈的顺序是相反的

队列queue

只能从结尾进入，从开关出的数组

FIFO

何时：只要按照先来后到的顺序使用数组元素时

如何：结尾入，开头出

### 8)二维数组

二维数组

什么是：数组中的元素内容，又是一个子数组

何时：1.保存横行竖列的二维数据

2.对大的数组中的元素，再进行细分类

创建：2步

1.先创建空数组，再赋值

var arr=[];

var[i]=[值1，值2，...]

2.再创建数组同时，初始化子数组

var arr=[

[值1，值2，...]，

[值1，值2，...]

]

访问：arr[r][c]

用法和普通数组元素的用法完全一样

强调：任何情况下下标r不能越界，会报错!

遍历：

for(var r=0;r<arr.length;r++){

for(var c=0;c<arr[r].length;c++){

arr[r][c] //当前元素

}

}

## 11.内置对象

## 12.string

## 13.正则表达式(不属于JS，但是JS要使用)

什么是：规定一个字符串中字符出现规律的规则

何时用：1.按规则模糊查找多种关键词时

2.用规则验证用户输入的格式时

1）关键字的原文是最简单的正则表达式

2）字符集

何时：只要一位字符有多个备选字时[备选字符列表]

强调：一个字符集只能匹配一位字符

简写：[0-9]数字

[a-z]小写字母

[A-Z]大写字母

[a-zA-Z] 大小写字母

[a-zA-Z0-9] 大小写字母和数字

[\u4e00-\u9fa5] 汉字

除了：[^47]

3）预定义字符集

\d [0-9] ; \w [[a-zA-Z0-9]]; \s 空字符，空格，tab . 通配符

4）量词

何时：只要规定一个字符集出现的次数时

如何：字符集量词

强调：默认仅修饰相邻的前一个字符集

有明确数量的边界：

字符集{n,m} 至少n个，最多m个

字符集{n,} n个以上

字符集{n} 必须n个

没有明确数量的边界：

字符集？ {0,1} 有没有都可以，最多一个

字符集\* {0,} 有没有都可以，多了不限

字符集+ {1,} 最少一个，多了不限

\*可有可无，多了不限 ？是可有可无，最多一次 +到少一次，多了不限

5）选择与分组

1.分组(多个规则)

何时：

1. 希望一个量词同时修饰多个字符集时 身份证号： \d{15}(\d\d[0-9Xx])?

2. 希望分段获取或处理字符串中部分子内容时 格式化生日：(\d{4})(\d{2})(\d{2})

2.选择(规则1|规则2) |优先级最低

何时：在两种规则间任选其一匹配 微信：(微|w(ei)?)\s\*(信|x(in)?)

6）指定匹配的位置

^ 字符串开头 ^\s+

$ 字符串结尾 \s+$

开头或结尾的空字符 ^\s+|\s+$

\b 单词边界 ^ $ 空字符 标点

7）密码强度

6~8位字母，数字的组合，至少包含一个大写字母和一位数字

^(?![a-z0-9]+$)(?![A-Za-z]+$)[A-Za-z0-9]{6,8}$

## 14.RegExp

## 15.Math

## 16.Date

## 17.Error

## 18.Function

### 一、什么是function函数？

函数其实是一个封闭一段代码的对象，函数名其实仅是引用函数对象的一个变通变量。

### 二、为什么用函数？

可以实现代码重用。

### 三、何时用？

只要一项任务，可能被反复使用，都要定义为函数，反复使用函数。

### 四、创建函数：(三种方式)

1.声明方式

function 函数名(形参列表){函数体;return返回值}

会被声明提前 hoist

什么是声明提前 ？

在开始执行程序前，引擎会首先查找var声明的变量和function声明的函数，将其提前到当前作用域的顶部集中创建，赋值留在原地。

鄙视时：

凡是看到先使用，后声明，都是在声明提前，先改为声明提前之后的程序，再判断输出。

解决问题：

1.所有变量和函数的声明都放在当前作用域的顶部

2.ES6可用let代替var，要求当前作用域中let a之前不允许出现未声明的a.

3.函数 var 函数名=function(参数列表){函数体;return返回值} 不会被声明提前。(赋值方式)

2.赋值方式

var 函数名=function(参数列表){函数体;return返回值}

不会被声明提前。

3.用new

var函数名=new Function("参数名1","参数名2",..."函数体";return返回值)

参数：接收函数执行时必须的数据的变量，

可让函数更灵活

只要函数必须某些数据才能正常执行

返回值：函数的执行结果

如果调用者需要获得函数的返回值。

### 五、调用

让引擎按照函数的步骤清单，执行任务

强调：函数不调用不执行，只有调用才执行

### 六、重载(overload)

1.什么是重载？

相同函数名，不同参数列表的多个函数

在调用时，根据传入参数的不同，自动选择匹配的函数执行。

2.为什么用重载？

用重载是为了减少API的数量，减轻调用者的负担。

3.什么时候用重载？

只要一项任务，需要根据不同的参数，执行不同的操作时，

4.怎么用重载？

问题：JS语法不支持重载，JS不允许多个同名函数同时存在。

解决：每一个函数内，都有一个arguments对象接住所有传入函数的参数值，根据arguments的元素内容或元素个数，判断执行不同的操作。

arguments：函数调用时，自动创建的，自动接收所有传入函数的参数值。

类数组对象：长的像数组的对象，

类数组vs数组

相同：1.下标 2.length 3.for/for of遍历

不同：类型不同，数组是Array类型 类数组对象是Object类型，类数组对象无法使用数组的API。

### 七、匿名函数

1.什么是匿名函数？

函数创建时，没有指定函数名，使用后自动释放！

2.为什么用匿名函数？

节约内存，划分临时作用域，避免全局变量。

3.什么时候用？

只要一个函数只用一次，

1.回调callback，将一个函数作为参数传入另一个函数内，被其他函数调用。

比如：

1.arr.sort(function(a,b){return a-b;})

ES6: arr.sort((a,b)=>a-b)

2.str.replace(/reg/g,function(kw,$1,$2,..){return替换值})

ES6：str.replace(/reg/g,(kw,$1,$2,....)=>{return替换值})

2.自调

定义函数后自己调用自己，调用后，立刻释放

什么时候用？

定义一个临时作用域，减少使用全局变量，避免全局污染

怎么用？

(function(参数列表){函数体;return返回值})(参数值列表)

+function(参数列表){函数体;return返回值}(参数值列表)

### 八、作用域和作用域链

一、作用域(scope)：

变量的可用范围，其实是一个对象

包括：

1.全局作用域对象window 保存全局变量

优点：可反复使用

缺点：随处可用，全局污染。

建议：尽量少用全局变量。

2.函数作用域对象：AO，保存局部变量，

包括：1.参数变量 2.函数内var出的变量

优点：仅函数内可用，不会被污染

缺点：不可重用。

二、函数的生命周期

1.开始执行程序前:

创建ECS，专门保存正在调用的函 数的执行环境 的数组。

首先在ECS中添加浏览器主程序的执行环境main

创建全局作用域对象window,所有全局变量都是存在window对象中

main执行环境引用window

2.定义函数时：

用函数名声明全局变量

创建函数对象，封闭函数定义

函数对象的scope属性，指函数创建时的作用域

函数名变量引用函数对象

3.调用函数时：

向ECS中压入本次函数调用的执行环境元素

创建本次函数调用时的函数作用域对象(AO)

在AO中创建所有局部变量：1.形参变量2.函数内用var声明的变量

设置AO的parent属性引用函数的scope属性指向的父级作用域对象

函数的执行环境引用AO

变量的使用顺序：

1.先在AO中找局部变量 2.AO中没有才去父级作用 域中找

4.调用后：

函数的执行环境出栈，导致AO释放，导致AO中的局部变量一同被释 放。

三、作用域链(scope chain)

由各级作用域逐级引用，形成的链式结构

保存着所有的变量

控制着，变量的使用顺序。

函数中，没有用任何对象/this就直接访问的变量，在作用域链中找

### 九、闭包(closure)

1.什么是闭包？

即重用变量，又保护变量不被污染/篡改的机制

2.为什么要用闭包？

1.全局变量 优点：可重用。缺点：易被污染。

2.局部变量 优点：仅函数内可用，不会被污染。 缺点：不可重用。

3.什么时候用？

即重用变量，又保护变量不被污染。

4.如何用(3步)：

补充：

1.外层函数：包裹受保护的变量和操作变量的内层函数

2.外层函数要返回内层函数的对象(三种)：

1.return function(){...}

2.直接给全局变量赋值一个内部function

3.将内部函数保存在一个对象的属性或数组元素中

return[function,function,function]

return{fun:function(){...}}0

3.调用外层函数，用外部变量接住返回的内层函数对象，形 成闭包。

5.闭包如何形成？

外层函数调用后，外层函数的作用域被内层函数引用着,无法释放。

外层函数的作用域，因为是保存在内层函数的私密的scopes通讯录里，所以外人都不知道！且通过全局，绝对无法修改。

6.闭包的缺点：

1.比普通函数占用更多内存，多的是外层函数的作用对象(AO)始终存在。

2.容易造成内存泄漏

解决：释放闭包，将引用内层函数对象的外部变量设置为null，导致内层函数对象被释放，导致外层函数的AO被释放。

7.鄙视：画简图2步

1.找受保护的变量，确定外层函数调用后，受保护变量的最终值

2.找操作受保护的变量的内层函数对象

结论：同一次外层函数调用，返回的内层函数对象，共用同一个受保护的变量。

## 19.OOP

### 一、面向对象三大特点：

封装：创建一个对象，集中存储一个事物的属性和功能

继承：父对象中的成员，子对象无需重复创建，就可直接使用

多态：同一事物，在不同情况下，表现出不同的状态

### 二、封装--创建对象

1.创建对象也称为封装：

将一个事物的属性和功能集中定义在一个对象中

事物的属性会成为对象的属性，其实就是保存在对象中的普通变量

事物的功能会成为对象的方法，其实就是保存在对象中的普通函数

对象的成员：属性和方法统称为成员。

2.为什么用封装？

便于维护

3.什么时候用封装？

只要使用面向对象，都要先创建对象，再按需求调用对象的方法执行操作。

4.如何？

1>创建一个单独的对象

1.用对象直接量

var obj={

属性名:属性值,

... : ... ,

方法名: function(){

... this.属性名...

}

}

ES6:

var obj={

属性名:属性值,

... : ... ,

方法名(){

... this.属性名...

}

}

何时：在创建对象时，已知对象的成员

问题： 对象自己的方法，访问自己的属性，如果不加this，仅会在作用域链中找，不会在对象中找

解决：this.属性名，今后，只要对象自己的方法要访问自己的属性，必须用this.属性名。

强调：不带this.的变量，在作用域链中查找

this.属性在当前对象和当前对象的原型链中找

2.用new

1.先创建一个空对象

var obj=new Object(); {}

简写：new可省略，()省略，但不能同时省略，可推广到其它内置对象

2.为新对象添加新属性：

obj.属性名=属性值

obj.方法名=function(){

...this.属性名....

}

3.何时：在创建对象时，暂时不知道对象的成员

4.揭示：js中一切对象底层都是关联数组

相同：obj.属性名 等效于 obj["属性名"] 只要访问对象的属性时，属性名是变化的变量，就只能用[变量]

for in 遍历每个成员：

for(var key in obj){

key //当前属性名

obj[key] //当前属性值

}

访问不存在的属性，不报错，返回undefined

强行给不存在的属性赋值，不报错，自动添加该属性

不同：类型不同，API不通用

2>问题：反复创建多个相同结构的对象时，会造成大量重复的代码

3>解决：用构造函数反复创建多个相同结构的对象

3、用构造函数

1.什么是构造函数？规定一类对象统一结构的函数

2.何时？反复创建多个相同结构的对象

3.作用？ 1.描述统一的结构 2.将空对象构建成要求的结构

4.如何？(2步)

\*1.定义构造函数

function 类型名(属性参数){

this.属性名=属性参数;

this.方法名=function(){

... this.属性名 ...

} //js中强烈不推荐将方法定义在构造函数中

}

\*2.用new调用构造函数

实例化一个对象

var obj=new类型名(属性值)

/1.创建一个新的空对象

/2.设置新的子对象的\_\_proto\_\_继承构造函数的prototype对象

/3.调用构造函数，将构造函数中的this自动替换成当前新对象，构造函数规定的属性添加到新对象中，并将传入的参数值保存在新对象的新属性中

/4.返回新对象的地址保存到变量中

优点：代码重用

问题：无法节约内存,放在构造函数中的方法定义，每new一次，都会创建函数对象副本

解决：继承

screen recorder

### 三、this

什么是？

自动引用正在调用当前方法的.前的对象

为什么？

不用this的普通变量，只能在作用域链中查找，无法进入对象中

一般对象的变量名可能变化，所以不能写死在对象的方法内部

何时？

只要希望去当前对象中找属性值时，就必须写this.属性名

obj.fun() fun中的this-->obj

new Fun() Fun中的this-->正在创建的新对象

fun()和匿名函数自调 this默认->window

类型.prototype.fun fun中的this->将来调用fun的.前的子对象，子对象一定是当前类型

如果this不是想要的，fun.call(替换this的对象)，相当于对象.fun()

### 四、访问对象成员

访问属性：对象.属性名,用法和普通变量完全一样

特殊：如果访问对象的属性时，属性名需要动态拼接，只能用obj[属性名]

调用方法：对象.方法名(),用法和普通函数完全一样。

### 五、继承

1.什么是？

父对象中的成员，子对象无需重复创建即可直接使用

2.何时？

只要多个子对象拥有相同的属性值或方法定义时，仅需要集中定义在父对象中一份，所有子对象共用即可。

3.为什么？

两个优点：代码重用，节约内存。

4.如何？

js中的继承都是继承原型对象，原型继承

#### 5.原型对象

什么是：集中保存同一类型的所有子对象共有成员的父对象

何时：只要多个子对象，拥有相同的成员时，都要将相同的成员集中保存在原型对象中一份即可

如何：(3步)

1>不用创建：(赠送的)

买一赠一，定义构造函数同时，已经自动创建了该类型的原型对象

构造函数.prototype指向原型对象

原型对象.constructor指向构造函数

2>new的第二步：（自动的）

创建一个新子对象，都会自动设置子对象的 \_ \_proto\_ \_继承构造函数的原型对象

3>向原型对象中添加共有成员：构造函数.prototype.成员=值

自有属性和共有属性区别？

自有属性：直接保存在对象本地的属性

共有属性：保存在原型对象中，所有子对象共有的属性

操作：

\*读取：子对象.属性

\*修改：自有属性，必须用 子对象.属性名=值

共有属性，必须用 构造函数.prototype.属性名=值

判断：

\*自有：var bool=obj.hasOwnProperty("属性名")

\*共有：不是自有&& obj.属性名!==undefined

或：属性名 in obj 判断属性名是否在obj的原型链中

\*否则,就是没有

内置对象的原型对象：

其实内部对象类型Array.Date....都是构造函数

每种类型都有自己的原型对象：Array.prototype,Date.prototype,...

内置对象的原型对象中保存了所有该类型的子对象共用的API

因定套路：解决旧浏览器无法使用新API的问题2步：

1.判断：if(typeof内置类型.prototype.API!=="function")

if(!("API"in内置类型.prototype))

2.添加：

内置类型.prototype.方法=function(参数){

...this //获得将来调用方法的.前的对象

}

#### 6.原型链(prototype chain)

由多级父对象逐级继承形成的链式结构

1>保存了：所有对象的属性

2>控制着：对象成员的使用顺序，1.先用自有属性2.自己没有，才沿原型链向父级找3.原型链上没有，返回undefined

3>vs作用域链：

保存了：局部和全局变量

控制着变量的使用顺序：

1.先用局部变量

2.局部没有，才去作用域链上找

3.找不到，报错。

4>总结：

所有不用.的变量都在作用域链上找，作用域链中的变量不用，可直接访问

所有对象.访问的属性都保存在原型链上，原型链上的属性必须用"对象."才能访问。

#### 7.判断对象的类型(5种方法)

0>typeof 只能区分基础类型和function,不能进一步区分对象的类型

1>var bool=类型.prototype.isPrototypeOf(child) 不仅能检查直接父对象

，而且检查整个原型链

2>var bool=child instanceof构造函数。不仅检查直接对象，而且检查整个原型链。

问题：检查整个原型链

解决：3>检查内置class属性

Object.prototype.toString.call(obj)=="[object类型名]"

obj.toString()

更严格：class属性直接保存在对象本地，只在创建对象时确定类型，创建对象后，不随继承关系的改变而改变。

如果检查数组类型：4> var bool=Array.isArray(obj) ES5 IE9+

原理和Object.prototype.toString.call一样

鄙视：方法定义在原型对象中，还是定义在构造函数对象上？

答：如果方法仅限当前类型的子对象可用，其他类型的对象不可用，就定义在原类型对象中，必须当前类型的子对象才能调用。

如果方法不确定将来调用它的对象类型，就定义在构造函数对象上，不需要任何对象实例，即可用构造函数名直接调用。

### 六、多态

同一个方法，在不同情况下表现出不同的状态。有两种情况：

#### 1.重载

#### 2.重写(override)

如果了对象觉得父对象的成员不好用，可在本地定义同名自有成员，来覆盖父对象中的成员。

### 七、自定义继承（4）

1.仅修改两个对象间的继承关系 ：

1>获得子对象的父对象 var father=Object.getPrototypeOf(child)

2>设置子对象继承指定的父对象Object.setPrototypeOf(child,father)

2.修改构造函数原型对象，来修改所有子对象的父对象

构造函数.prototype=father

时机：必须紧跟在构造函数定义之后，开始创建子对象之前

3.仅基于现有父对象，创建子对象，并扩展自有属性：Object.create()

创建新子对象，继承父对象，扩展子对象自有属性

var child=Object.create(father,(

属性名:{四大特性},

...:..

))

鄙视：模拟实现Object.create() 在导图里有公式

4.两种类型间的继承

何时：如果发现多个类型拥有部分相同的属性结构和方法定义，都要抽象父类型

如何：2步

1>定义抽象父类型

相同的属性结构定义在父类型的构造函数中

相同的方法定义在父类型的原型对象中

2>让子类型继承父类型

1.在子类型构造函数中借用父类型构造

extends :让父类型构造函数帮助添加相同部分的属性定义||子类型构造函数仅负责添加独有的属性定义即可

错误:直接调用父类型构造函数,this->window,父类型中的属性都泄露到全局

正确：父类型构造.call(this, 参数1,参数2,...) 简写：父类型构造.apply(this, arguments)

鄙视:call vs apply

相同:都是强行借用一个本来无法调用的函数，并临时替换函数中this为指定对象

不同:

call：传入借用函数的参数，必须单独传入，逗号分隔

apply：传入借用函数的参数，放在一个数组中整体传入；可自动打散数组类型参数

2.让子类型原型对象继承父类型原型对象

inherits

Object.setPrototypeOf(子类型构造.prototype.父类型构造.prototype)

## 20.ES5

## 21.ES6

# ==end==