考试时间: 满分:100分 开卷

一 基础

1. 什么是JS？JS有什么特点？

Js是JavaScript的简称，是网景（Netscape）公司开发的一种基于客户端浏览器、面向（基于）对象、事件驱动式的网页脚本语言。

特点：（1）JavaScript 被设计用来向 HTML 页面添加交互行为。

（2）JavaScript 是一种脚本语言（脚本语言是一种轻量级的编程语言）。

（3）JavaScript 由数行可执行计算机代码组成。

（4）JavaScript 通常被直接嵌入 HTML 页面。

（5）JavaScript 是一种解释性语言（就是说，代码执行不进行预编译）。

（6）所有的人无需购买许可证均可使用 JavaScript。

2.什么是JS解析引擎？有什么作用？

JavaScript解析引擎就是能够―读懂JavaScript代码，并准确地给出代码运行结果的程序,JavaScript引擎本身也是程序，代码编写而成。

3.下面JS代码的输出结果是：

var a=1;

var b=(a++)+a;

alert(b);

结果是3

4.解释下面代码:

var result=[]; //var表示声明一个变量，var声明的变量会有变量提升。等号表示将等号右边的值取出并赋值给等号左边的变量。将一个空数组赋值给新声明叫result的变量空间，当前在全局声明的变量等于是给window创建了一个属性，这个变量叫做全局变量。  
function foo(){ //function表示声明一个函数，名为foo的函数，函数声明也会隐式提升到最前面，在变量提升的前面，与变量提升不同的是函数内部的表达式也会提前声明出来，所以函数可以在函数声明前使用，不会报错。当前在全局声明的函数等于是给window创建了一个方法。  
var i= 0;//在函数内部声明一个名叫i的变量空间，在函数内部也会变量提升，但是只会在函数内部变量提升，不会在全局环境里隐式声明。将0赋值给新声明名为i的空间，这个变量叫做局部变量，在函数外部不能直接使用，要在外部使用需要使用闭包。  
for (;i<3;i=i+1){result[i]=function(){document.write(i+"<br/>");}}};//因为在全局声明了i这个变量，所以for循环执行第一条语句可以不写。For循环执行第二条语句是判断语句即布尔表达式，若其值为false，将不会再执行for循环内部的代码。当执行判断语句后会执行for循环内部的代码，最后是执行i的自增表达式，依此循环。For 循环内部的代码表示：每次循环将等号后面的函数表达式放入result这个数组作为数组的元素，因为for循环三次，所以当for循环执行结束后，result数组内部有三个元素且内容相同。在for循环结束的同时，i最后会再进行一次自增，所以在for循环结束时，i的值为3.  
foo();写出函数名并加上小括号表示执行foo这个函数。执行完这个函数后，result数组内放有三个存有函数代码的元素，其函数不会执行，因为等号表示将右边的值赋给左边的变量，所以只进行了赋值操作，没有执行函数，并且for循环结束后i为3.  
result[0](); //通过数组的下标result【下标】的方式表示数组每一个元素，因为数组元素内部存放的是函数，所以result[]后面加上小括号表示执行这个函数。所以将会在页面上打印i的值并且换行。  
result[1]();  
result[2]();

结果是在页面上打印：3

3

3

5.在控制台上随机打印出你们班3位同学的名字

<script type="text/javascript">

function showName() {

var arr = ['安可', '宴亮', '杨成', '张昭', '韩聪', '倪霖', '杨梅', '刘春梅', '袁毅'];

//console.log(parseInt(Math.random()\*9));

var arrNum = new Array();

function random() {

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

arrNum[i] = parseInt(Math.random()\*9);

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

if(arrNum[j]==arrNum[i]){

while (true) {

arrNum[i] = parseInt(Math.random()\*9);

if(arrNum[j]!=arrNum[i])

break;

}

}

}

}

return arrNum;

}

for(var a of random()){

console.log(arr[a]);

}

}

showName();

</script>

二 高级

1. 在界面上展示出下面效果(画的不标准,想象小圆都一样大,围绕大圆,都是实心的):大圆的半径R:80px left:300px top:300px 小圆的半径r:20px)

注1:盒子的坐标js设置示例:

div1=document.qurerySelector(‘#div1’)

div1.style.position=’relative’

div1.style.left=’1px’

div1.style.top=’1px’

注2:答案写js代码,不用发html文档



<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>

</title>

<style media="screen">

div{

border-radius: 50%;

width: 40px;

height: 40px;

background: red;

position: absolute;

}

</style>

</head>

<body>

<div id="divBig"></div>

<div id="div1"></div>

<div id="div2"></div>

<div id="div3"></div>

<div id="div4"></div>

<div id="div5"></div>

<div id="div6"></div>

<div id="div7"></div>

<script type="text/javascript">

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

divBig.style.width = '160px';

divBig.style.height = '160px';

divBig.style.left = '300px';

divBig.style.top = '300px';

var div1 = document.querySelector('#div1');

div1.style.left = '360px';

div1.style.top = '260px';

var div2 = document.querySelector('#div2');

var a = Math.PI/2;

div2.style.left = Math.cos(a-(2\*Math.PI/7))\*100+360+'px';

div2.style.top = 360-Math.sin(a-(2\*Math.PI/7))\*100+'px';

var div3 = document.querySelector('#div3');

div3.style.left = Math.cos(4\*Math.PI/7-a)\*100+360+'px';

div3.style.top = 360+Math.sin(4\*Math.PI/7-a)\*100+'px';

var div4 = document.querySelector('#div4');

div4.style.left = Math.sin(Math.PI-(6\*Math.PI/7))\*100+360+'px';

div4.style.top = Math.cos(Math.PI-(6\*Math.PI/7))\*100+360+'px';

var div5 = document.querySelector('#div5');

div5.style.left = 360-Math.cos(3\*a-(8\*Math.PI/7))\*100+'px';

div5.style.top = Math.sin(3\*a-(8\*Math.PI/7))\*100+360+'px';

var div6 = document.querySelector('#div6');

div6.style.left = 360-Math.cos((10\*Math.PI/7)-3\*a)\*100+'px';

div6.style.top = 360-Math.sin((10\*Math.PI/7)-3\*a)\*100+'px';

var div7 = document.querySelector('#div7');

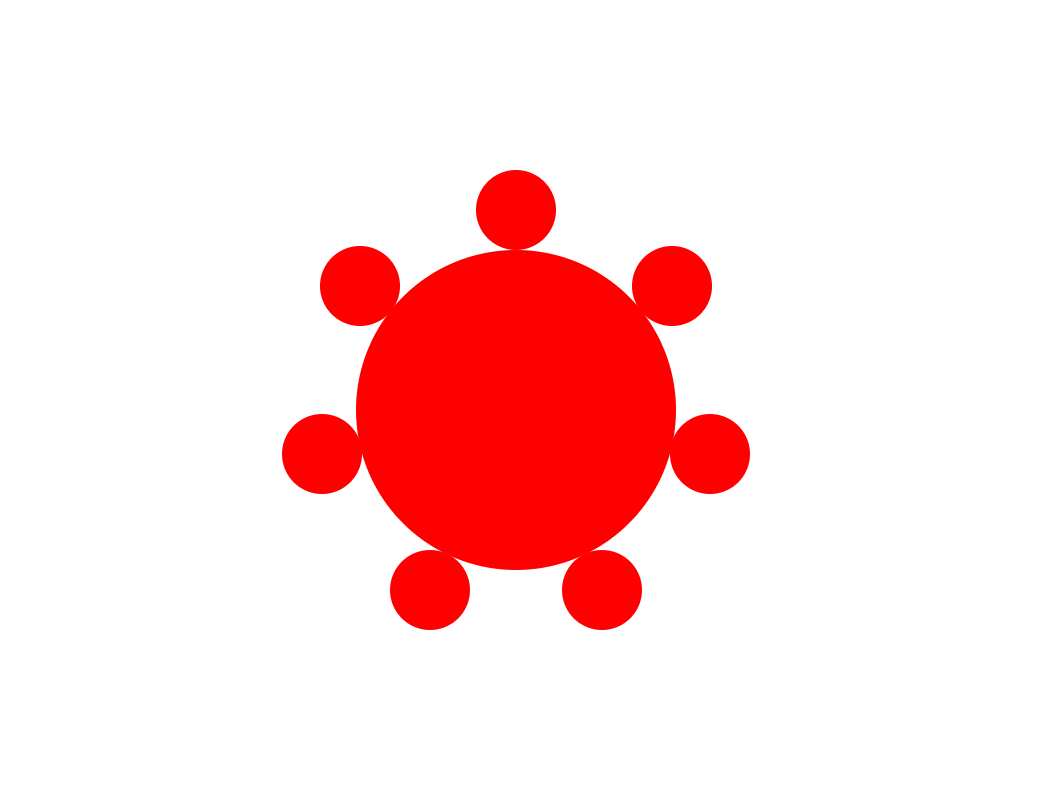
div7.style.left = 360-Math.cos((12\*Math.PI/7)-3\*a)\*100+'px';

div7.style.top = 360-Math.sin((12\*Math.PI/7)-3\*a)\*100+'px';

</script>

</body>

</html>



2.详细解释下面的js代码

var a=1; // var表示声明一个变量，var声明的变量会有变量提升。等号表示将等号右边的值取出并赋值给等号左边的变量。将1赋值给新声明叫a的变量空间，当前在全局声明的变量等于是给window创建了一个属性，这个变量叫做全局变量。

function fn(){ //function表示声明一个函数，名为fn的函数，函数声明也会隐式提升到最前面，在变量提升的前面，与变量提升不同的是函数内部的表达式也会提前声明出来，所以函数可以在函数声明前使用，不会报错。当前在全局声明的函数等于是给window创建了一个方法。

a=b=2;//右边第一个等号表示等号左边的变量指向等号右边2这个值的地址空间，因为b在之前没有声明，所以会隐式在全局申明一个b 值为undefined；左边第一个等号表示将b指向的值使等号左边a这个变量也指向这个值的空间，使a，b都指向这个值的空间。

document.write(b) // 使b的值打印在页面上

}

fn(); // 执行fn这个函数。

alert(b) //弹出窗口显示b的值，因为b在函数内部没有用var声明，所以函数内部的b会在全局隐式声明，所以b其实是一个全局变量，所以函数外部alert能调到b指向的值。

3.说说这段JS代码在内存中的意义,每一个**单词,符号在内存中的意义**详细说明

var person = {name:”karen”,}

var表示声明一个变量名指向一个存储空间，因为等号右边的式用大括号括起来的键值对，所以声明的这个变量是一个对象，如果这个对象是在全局声明的那么这个对象就是全局的属性。大括号里面冒号左边表示对象属性的名称，冒号右边表示对象属性的值，要用引号括起来，对象中也可以储存方法，冒号前面是对象方法名称，冒号后面是方法这个函数的代码，对象内部每个属性或者方法需要用逗号隔开，在严格模式下，最后一个对象或者方法最后不能写逗号，所以这个表达式不能在严格模式下书写。

4.解释以下js代码

function fn(){

//function表示声明一个函数，名为fn的函数，函数声明也会隐式提升到最前面，在变量提升的前面，与变量提升不同的是函数内部的表达式也会提前声明出来，所以函数可以在函数声明前使用，不会报错。当前在全局声明的函数等于是给window创建了一个方法。

rank=function(){alert(“盖伦”)} //fn内部的静态方法,因为没有var所以这个rank变量会隐式声明在函数的外部。

return this //返回调用fn这个函数的空间

}

fn.rank=function(){alert(“赵信”)} //给fn这个构造函数添加名为rank的静态方法

fn.prototype.rank=function(){alert(“典韦”)} //给fn构造函数的原型添加一个方法

var rank=function(){alert(“项羽”)} // 声明一个叫rank的变量并指向等号后面的这个匿名函数，等于是给这个匿名函数取了个名字，方便调用

function rank(){alert(“琴女”)} // 声明了一个叫rank的函数

fn.rank() //fn调用内部的静态方法rank，因为之前外部添加了一个同名的静态方法，所以内部原有的静态方法被覆盖，所以这里弹出窗口显示 赵信

rank() // 因为var变量提升在function隐式提升的后面，所以rank（）使用的是隐式声明的变量调用的函数，所以这里弹窗显示 项羽

fn().rank() //这里表示是在执行fn（）这个函数并找rank（）函数，因为return this表示执行fn（）的window对象，并且fn函数内部的函数是隐式在全局环境声明的，所以他属于window的方法，所以上面表达式等同于this.rank（）即window.rank（）,因为fn（）的执行，将外部隐式声明的rank函数内部覆盖为函数内部的函数内容所以弹出窗口 盖伦

rank()// 因为var变量提升在function隐式提升的后面，所以rank（）使用的是隐式声明的变量调用的函数，无法调用函数声明的函数，但是因为上面fn（）的执行，将外部隐式声明的rank函数内部覆盖为函数内部的函数内容，所以这里弹出的窗口显示和上面相同 盖伦

new fn.rank() // 因为优先级的原因，先看成员访问，因为小点左边fn可以执行访问，并且new无参优先级低于成员访问，所以new后面的是fn构造函数内的静态方法，再通过new创建这个静态方法的对象，并且弹出窗口 赵信

new fn().rank() //因为优先级的原因，先看成员访问，因为小点左边fn（）无法执行访问操作，并且new有参数和成员访问优先级相同，所以先执行new fn（），先声明一个新对象，然后由对象调方法，对象能够调用的方法是通过对象内\_\_proto\_\_这个原型属性指向构造函数的prototype，所以这个调用的函数是通过prototype添加的方法，所以这里弹出窗口 典韦 「(new fn()).rank()」

new new fn().rank() // 因为优先级的原因，先看成员访问，因为小点左边fn（）无法执行访问操作，并且new有参数和成员访问优先级相同，所以先执行new fn（），所以变成 new 对象.rank（），再看成员访问，因为小点左边对象能够访问内部成员，并且new无参优先级低于成员访问，所以执行成员访问，方法调用对象内部通过prototype添加的方法，再执行new有参并执行方法，所以弹窗 典韦 「new ((new fn()).rank)()」

1. 设计一个工具,具有如下功能:

①:如果传入字符串,检测字符串的敏感词(“中国共产党”,”广电总局”),并用\*代替(敏感词的字符个数与\*数量相等)

②:如果传入数组,检测数组中是否有字符串(不包含数值类型的字符串),如果有字符串,将原数组中的字符串首字母大写(字母大写,汉字变粗),没有字符串不做操作

③:如果传入其他类型数据,则返回 ”超出检测范围”.

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta charset="utf-8">
5. <title></title>
6. </head>
7. <body>
8. <script type="text/javascript">
9. function fn(num) {
10. if (typeof(num) === 'string') {
11. function change(num){
12. if(num.includes("中国共产党")||num.includes("广电总局")){
13. if(num.includes("中国共产党")){
14. num=num.replace("中国共产党","\*\*\*\*\*");
15. //console.log(num);
16. }
17. if(num.includes("广电总局")){
18. num=num.replace("广电总局","\*\*\*\*");
19. //console.log(num);
20. }
21. if(num.includes("中国共产党")||num.includes("广电总局")){
22. return change(num);
23. }
24. }
25. return num;
26. }
27. return change(num);
28. } else if (num instanceof Array) {
29. //console.log('it is array');
30. for (var i = 0; i < num.length; i++) {
31. var n = Number(num[i]);
32. if (isNaN(n)) {
33. //console.log(num[i]);
34. if (num[i].charCodeAt(0) > 255) {
35. //console.log(num[i]);
36. var a = num[i].split("");
37. for (var j = 0; j < a.length; j++) {
38. //console.log(a);
39. if (j == 0) {
40. //document.write((a[j].big()));
41. a[j] = a[j].big();
42. }
43. }
44. num[j] = a.join("");
45. //document.write(a.join(""));
46. } else {
47. var b = num[i].split("");
48. for (var j = 0; j < b.length; j++) {
49. //console.log(b);
50. if (j == 0) {
51. //document.write((b[j].toUpperCase()));
52. b[j]= b[j].toUpperCase();
53. }
54. }
55. num[i] = b.join("");
56. }
57. }
58. }
59. return num;
60. } else {
61. alert("超出检测范围");
62. }
63. }
64. console.log(fn('321广电总局123中国共产党中国共产党'));
65. console.log(fn([1, 2, '1', 'asd', '超出检测']));
66. </script>
67. </body>
68. </html>
69. 详细说明ES6与ES5的函数之间的区别(至少6种,详细说明(写法,用法,底层含义等等))
70. Es5不能直接为函数的参数指定默认值，只能通过在函数内部对传参进行判断并赋值例如：y=y||’world；对y所传的值进行判断，若y没有值，则给y一个默认值world。但是这样的缺点是如果y的值对应的布尔值是false，那么y还是会被赋默认值，这样就出错了。

所以另外的一种方法是：if（typeof y === ‘undefined’）{

y = ‘world’};

在es6中允许在函数的参数设置默认值，即可以直接写在参数定义的后面。例如：function fn(x,y=’world’){};参数变量是默认声明的，所以在函数的内部不能再用let或者const对这个变量再次声明，函数也不允许有同名函数理由同上。参数默认值是惰性求值，例如

let x = 99；

function foo(p = x + 1) {

console.log(p);

}

foo() // 100

x = 100;

foo() // 101

参数p的默认值是x+1，每次调用函数时，都会重新计算x+1，并不是默认p=100；

1. es5中无法在参数定义的时候设置默认值，所以也就不能在参数定义时使用解构赋值来对参数赋默认值，而在es6中，参数默认值可以与结构赋值的默认值结合起来使用。例如：

function foo({x, y = 5}) {

console.log(x, y);

}

foo({}) // undefined, 5

foo({x: 1}) // 1, 5

foo({x: 1, y: 2}) // 1, 2

foo() // TypeError: Cannot read property 'x' of undefined

上面代码使用了对象的解构赋值默认值，只有当函数foo的参数是一个对象时，变量x和y才会通过解构赋值而生成。如果foo调用时参数不是对象，变量x和y就不会生成，从而报错。如果对象没有y属性，y的默认值5才会生效。

1. 在es6中，一旦设置了参数的默认值，函数进行声明初始化时，参数会形成一个单独的作用域。等到初始化结束，这个作用域会消失，这种情况在不设置默认值时，是不会出现的。例如：

var x = 1;

function f(x, y = x) {

console.log(y);

}

f(2) // 2

参数y的默认值等于变量x，调用函数f时，参数会形成一个单独的作用域，在这个作用域里面，默认值变量x指向第一个参数x，而不是全局变量x，所以输出为2.

let x = 1;

function f(y = x) {

let x = 2;

console.log(y);

}

f() // 1

上面例子中，函数调用时，参数形成一个单独的作用域，因为变量x本身没有定义，并且内层何以调用外层的变量，所以参数默认值x指向外层的全局变量。

function f(y = x) {

let x = 2;

console.log(y);

}

f() // ReferenceError: x is not defined

上面这个例子中，参数形成一个单独的作用域时，参数作用域中的x无法指向函数内部的变量x，并且let声明变量前面有四区，所以参数作用域无法访问到数据，也没有全局变量x的存在，所以报错。

1. es5中，虽然函数中参数可以写无数个，但是数量很多的时候不方便书写和使用，如果写入了多余参数，可以用arguments对象找出，但是并不方便，所以在es6中引入了rest参数用于获取函数的多余参数，这样就不需要使用arguments对象了。Rest参数搭配的变量是一个数组，该变量多余的参数放入这个数组中。例如：

function add(n1,...values) {

let sum = 0;

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

sum += val;

}

return sum;

}

add(2, 5, 3) // 8

rest参数之后不能有其他参数，否则会报错。

1. 函数的name属性，返回该函数的函数名

function fn(){}

fn.name;//”fn”

name属性早被浏览器支持，在es6中，才被写入标准中。

在es6中name属性与es5中不同的在于，如果将一个匿名函数赋值给一个变量，es5的name属性会返会空字符串，而es6的name属性会返回实际的函数名。

1. 在es6新加入了箭头函数。例如

var f = v =>v;

f(20); //20

等同于：

var f = funcrtion(v){

return v;

}

f(20); //20

如果箭头函数的代码部分多余一条语句，就要使用大括号将它们括起来，并且使用return语句返回。由于大括号被解释为代码块，所以如果箭头函数直接返回一个对象，必须在对象外面加上括号。

箭头函数可以与变量结构结合使用

const full = ({ first, last }) => first + 'xxxx ' + last;

// 等同于

function full(person) {

return person.first + ' xxxx' + person.last;

}

箭头函数的this对象，表示定义时所在的对象，而不是es5中使用时所在的对象

//----对比1

function foo1(n) {

this.tool=n

}

var f1=new foo1(function(){this.b=10})

f1.tool()

console.log(window.b)

console.log(f1.b)

//----对比2

function foo(n) {

this.tool=n

}

var f=new foo(() => this.a=10)

f.tool()

console.log(window.a)

console.log(f.a)

1. js中的基本数据类型与复杂数据类型分别是哪些? 都有什么特点?

基本数据类型：undefined、Null、boolean、number、string

复杂数据类型；object

undefined：如果值为定义，typeof返回的事undefined。Undefined类型只有一个值，即特殊的undefined。在使用var声明变量但未对其加以初始化时，这个变量的值就是undefined。不过一般建议尽量给变量初始化。

Null：需要注意，tyopeof（null）返回为object，因为特殊值null被认为是一个空的对象引用。Null类型是第二个直邮一个值的数据类型，这个特殊的值时null。从逻辑角度来看，null值表示一个空对象指针，而这也正是使用typeof操作符检测null时返回object的原因。

Boolean：该类型直邮两个字面量：true 和false，这两个值与数字值不是一回事，因此true不一定等于1，而fasle也不一定等于0.虽然boolean类型字面值直邮两个，但是js中所有类型的值都有与这两个boolean值等价的值，要将一个值转换为其他对应的boolean值，可以调用类型转换函数boolean（）；

Number：这种类型用来表示整数和浮点数值，有一种特殊的数值NaN，这个数值用语表示一个本来要返回数值的操作数未返回的情况。NaN与任何值都不相等，包括NaN本身。

String：字符串可以看成行为与基本类型相似的不可变引用类型。

Object：本质是一组无序的键值对组成的，在编程中使用的都是实例出来的对象。Object有连个重要的属性，prototype和constructor。每个object或者类都有自己的prototype。

1. 谈谈你对js中对象的理解(至少500字)

js中万物皆对象，在js中，任何一个变量，方法，函数等都叫对象，如同在现实中的具体事物的一种抽象。在数据上就是将无需的变量集合在一起，就像在浏览器中给对象\_\_proto\_\_这个原型属性，它的指向是更内层的对象原型，每个对象原型属性都会指向一个prototype，所以在数据上，对象是变量的集合。对象的属性可以是任何三种基本数据类型的，或者任何抽象数据类型，如另一个对象。对象属性通常是内部使用的对象方法的变量，但也可以适用于整个页面全局可见的变量。对象的方法是让对象做某件事，一个函数和一个方法，所不同的是一个function语句的一个独立的单元和方法被附加到对象，并可以通过这个关键字被引用之间的长别不大。方法可以用于一切从现实对象的屏幕上的内容，以对一组本的属性和参数执行。所有用户自定义的对象和内置对象被称为对象的对象的对象的后代。在js中，通过new运算符用于创建对象的实例，要创建一个对象，new运算符后面是构造方法。函数运行时会自动生成一个内部对象this，这个this对象只能在函数内部使用，但总指向调用它的对象，当this作为函数调用时，this会指向全局，例如：

var name = ‘enco’;

function person(){

return this.name;

}

console.log(person());

上面的这个例子在全局作用域中调用person（），此时的调用对象为window，因此this指向window。

当this作为对象的方法调用，那么this会指向调用这个方法的对象。

当this作为构造函数调用，构造函数使用new关键字创建一个实例，此时this指向实力对象。例如：

function person() {

return new person.prototype.init();

}

person.prototype = {

init: function() {

return this.name;

},

name: 'Brain'

};

console.log(person().name); //undefined

调用person，返回person.prototype.init()的一个实例，假设返回的这个实例名为a，那么此时的this指向的就是a，a作为person.prototype.init()的一个实例，那么所有定义在person.prototype.init()中的方法等都可以被a调用，但是name属性定义在person的原型中，而非init函数中，因此返回undefined。