学习一门编程语言的基本步骤：

1. 了解背景知识，包括历史，现状，特点，应用场景
2. 搭建开发环境：编写hello world
3. 变量和常量，用于临时保存数据
4. 数据类型
5. 运算符
6. 逻辑结构
7. 通用小程序
8. 函数和对象
9. 第三方的库和框架
10. 实用的项目

**JS概述**

**1.**

（1）1995，JS最早出现在netscape的浏览器中，是浏览器的一个脚本

1996，IE3中出现JS，叫做Jscript

1997，ECMA组织制定了JS标准ECMAScript

2009，遵循CommonJS规范，开始向服务器端发展

（2）现状

既可以运行在浏览器端制作交互效果，也可以运行在服务器端

（3）特点

JS属于解释型语言，编一行执行一行，而不像编译型语言，需要一次性编译完成后执行

弱类型语言

基于对象

跨平台性。可以在所有操作系统运行，如Windows下，Mac下运行

（4）应用场景

制作浏览器端的交互效果

创建web服务器以及操作数据库

1. **JS的开发环境**

谷歌/火狐/欧朋/Safari/IE

1. 浏览器自带的JS解释器（又叫JS引擎）
2. 服务器端nodeJS解释器

<http://nodejs.org>

node –v查看当前nodeJS版本

1. 执行JS代码

浏览器：创建01.JS和01.HTML两个文件，在01.HTML文件中引入01.JS <script src=”01.js”></script>

nodeJS：window+r输入cmd打开，输入node空格+拖拽文件 回车

需要在脚本输入console.log() 才能打印

1. **JS语法规范**

JS区分大小写，分号可加可不加。单行注释//，多行/\* \*/

1. **变量**

用于存储数据的容器

1. 声明变量

Var eid=8; 使用关键字var来声明变量，变量名称eid，储存的值为8

1. 变量的命名规则

可以用字母，数字，下划线，美元符号，不能以数字开头。不能使用关键字作为变量名，例如var，if。

常用的命名法，多个连词之间可以用下划线或者驼峰命名 user\_name userName

1. 声明了变量未赋值，此时的变量值为undefined（未定义） var a;

可以多次为变量赋值，并且赋不同类型的值

一次性声明多个变量 var name=’’,adr=’’,phone=’’

练习：声明多个变量，分别保存语文，数学，总成绩，计算总成绩，并把结果赋值给总成绩，打印总成绩

var chinese=90,math=80,zong;

zong=chinese+math;

console.log(zong) 打印多个以,隔开

1. **常量**

一旦声明不允许重新赋值，不允许为空 const pi=3.14;

练习：声明常量保存中秋节和国家的面积（960万）

1. **数据类型**

分为原始类型和引用类型

原始类型分为数值型、字符串型、布尔型、未定义型（undefined）、空（null）

1. 数值型

分为整型和浮点型

整型在内存中占4个字节，浮点型占8个字节

课后练习：使用变量保存圆的半径，常量保存圆周率，声明变量保存面积和周长，最后计算并打印出来

Var r=5;

Const pi=3.14;

Length=2\*pi\*r;

Var Erea=pi\*r\*r;

Var lentgth=2\*pi\*r;

Console.log(length,erea)

day 06

**1.数据类型**

原始类型：数值型，布尔型，字符串型，未定义，空

1. 数值型

分为整型和浮点型

整型：

八进制 以0开头的数字， 例如 010，代表8

十六进制 以0X开头 0XF代表15

A~F代表10~15，不区分大小写

0XFF代表255 15\*16+15

浮点型分为定点小数和指数型小数

3.14e3 3140

3.15e-3 0.00314

检测数据类型 typeof

1. 字符串型

数据被引号包含就是字符串型，不区分单双引号

因此变量不能放在引号里

查看任意一个字符的Unicode码（汉字在数据库里的编码就是unicode码 ）

Console.log(‘达’.charcodeat())

Console.log(b,typeof b)看数据类型

1. 布尔型

True/false 在程序中表示真或者假，常用于是否的结果，例如是否登录，是否注册

两种：1是直接赋值，2是通过运算结果得到

1. 未定义型

只有一个值undefined

1. 空

只有一个值，叫null。手动输入null。Var b=null 常和引用类型一起使用

**2.数据类型转换**

**（1）隐式转换** 在运算过程中自动转换 var a1=1+’2’ console.log(a1)=12 加代表拼接 最后得到的是字符串string

1）数字+字符串 数字转成字符串 1+’2’=’12’

2）数字+布尔型 var b=true+8; console.log=(b)=9 加代表加法运算 布尔值转为数字（true转为1.false转为0），最后得到数字

3）字符串+布尔型 var c='1'+true; console.log(c)=1 true 加变成拼接，布尔型转为字符串，最后得到字符串 *\*\*凡是字符串参与的，最后都变成字符串*

*-\*/ 隐式转换 如果运算符的两端不是数字，则会自动调用number转换为数值型*

JS中加号的作用：执行加法运算；字符串的拼接（其中至少有一个值为字符串）

练习：var num1=3,num2=true,num3=’tedu’ ;

Console.log(num1+num2+num3); ‘4tedu’

Console.log(num2+num3+num1); ‘truetedu3 ’

Console.log(num3+num1+num2); ‘tedu3true’

Var b1=’2’-‘3’ -1

Var b2=’5’-true 4

Var b3=4\*’5’ 20

Var b4=’8a’-10; nan

Var b5=undefined+1 nan

Var b6=null-4 -4

**（2）强制转换**

1）强制转为数值型 number（数据）var c1=number(‘2’); console.log(c1); console.log(c1,typeof c1) nan和任何数加减乘除都是nan（not a number）

Number(‘2’) //2

Number(‘2a’) //nan

Number(true) //1

Number(undefined) //nan

Number(null) //0

练习：声明变量保存一个人的姓名，性别，年龄，打印“我的姓名叫？，性别？，年龄？”

Var name=’tom’;

Var sex=’男’;

Var age=18;

Console.log(‘我的姓名叫’+name+’,性别’+sex+’,年龄’+age);

练习：声明变量保存笔记本的名称，价格，打印商品的名称为？价格为？

Var title=’apple air’;

Var price=5000;

Console.log(‘笔记本的名称为’+title+’,价格是’+price);

2）将数据转为整型 parseInt

Var num1=parseInt(3.14); console.log(num1) //3

Var num2=parseInt(‘5.8’);console.log(num2) //5

Var num3=parseInt(‘4.7a’);console.log(num3) //4

Var num4=parseInt(‘a4’);console.log(num4) //NaN

True,null,undefined 都会是NaN *只要开头都是数字，就会转化为整型，开头为非数字，就会转换为NaN*

3）将数据转为浮点型 parseFloat

和parseInt的用法一样，只是转的数据为浮点型

4）将数值和布尔型转字符串 toString

var n=3;

console.log( n.toString(),typeof n.toString())

**3.运算符**

表达式：由运算符连接的操作数据，组成的形式

运算符分为算术运算符、比较运算符、逻辑运算符、位运算符、赋值运算符、三目运算符

**（1）算术运算符** + - \* / % ++ --

% 取余（取模） console.log(5%2); //1 console.log(2%5); //2

++ 自增 在原来的基础上加1 var a=1; a++; console.log(a); //2 var a=1; a++; console.log(a++) // 2 (先打印a的值再执行自增) ++a，是自增再打印

--自减 在原来的基础上减1 var b=4; b--; console.log(b); //3

**（2）比较运算符**

> < <= >= == != ===（全等于） !==（不全等于）

Console.log(3>2); //true

Console.log(2==’2’) //true

Console.log(2===’2’) //false

Console.log(0==false) //true

Console.log(0===false) //false

== 只是比较值是否相等 ===不仅比较值，还会比较类型

Console.log(3>’10’); false 字符串和数值比较，字符串转为数值

Console. log(‘3’>’10’); true 比较的是首字母的Unicode码

练习：3>’10a’ false

3<’10a’ false

3==’10a’ false

把字符串转为数值，console.log( number(‘10a’)); //NaN 任何值和NaN比较结果都是false

**（3）逻辑运算符**

并且 && 关联的两个都是true，结果才是true，否则false

或者 || 关联的两个只需要一个是true，结果就是true，否则是false

非 ！ 取反 !true ->false

练习：声明变量保存用户名和密码，如果用户名为root，并且密码为123456，打印结果为true，否则打印false

Var a=’root’;

Var b=’123456’;

a===’root’ && b===’123456’;

console.log(‘true’);

Var a=’root’;

Var b=’123456’;

Console.log(a===’root’ && b===’123456’);

（逻辑短路）

Var a=’root’;

Var b=’123456’;

(a===’root’ && b===’123456’) && console.log(‘true’);

（三目运算符）

Var a=’root’;

Var b=’123456’;

(a===’root’ && b===’123456’) ? console.log(‘true’) : console.log(‘false’);

（if语句）

Var a=’root’;

Var b=’123456’;

If(a===’root’ && b===’123456’){console.log(‘true’)};

练习：声明变量保存用户输入的值，该值为用户名‘dangdang’或者该值为手机号码18111111，如果有一个条件满足打印true，否则打印false

Var input=’dangdang’;

Console.log(input===dangdang || input===’181111111’);

**逻辑短路**

&& 当第一个条件为false，就不再执行第二个条件

|| 当第一个条件为true，就不再执行第二个

注意：逻辑短路无需关注结果为true or false，重点看第二个表达式是否会执行

练习：声明变量来保存年龄，如果满18岁，打印“成年人”

Var age=18;

Age>=18 && Console.log(‘成年人’)

Var age=18;

Age>=18;

Console.log(‘成年人’)

**（4）位运算符**

在执行计算的时候，计算机会把数据转成二进制来进行计算

1. 10 11 100 101 110 111 1000 1001 1010

100110=100000+100+10=32+4+2=38

110111=100000+10000+100+10+1=32+16+4+2+1=55

十进制转二进制：25-> 11001 25=16+8+1

按位与（&）：console.log(5&7); //5 先转2进制，101和111，上下比较，1和1就是1,1和0就是0（上下两位都是1，是1，否则是0）

Console.log(11&19); //3

按位或（|）：上下两位含有1，结果是1，否则为0

按位异或（^）：上下两位不同为1，相同为0

*课后任务：27&49=17 43&18=57 15|19=31*

*声明变量保存年份，判断这个年份是否为闰年，结合逻辑短路，如果是闰年，打印“闰年”*

*四年一润，能被4整除，并且不能被100整除，或者能被400整除*

*Var year=2019;*

*(year%4===0 && year%100!==0 || year%400===0) && console.log(‘闰年’)*

day 07

运算符

**（1）位运算符**

按位右移>>当前的值除以2取整

按位左移<<当前的值乘以2

**（2）赋值运算符**

= += -= \*= /= %=

**（3）三目运算符**

单目运算符：只需要一个数据和运算符

a++ a-- !a

双目运算符：需要两个数据和运算符

= < > ==

三目运算符：需要三个数据和运算符

条件表达式： ? 表达式1 : 表达式2;

//声明两个变量分别保存用户名和密码，如果用户名为root，密码为123456，打印登录成功，否则打印失败

var a=lalala;

var b=123456;

(a==='root' && b==='123456') ? console.log('登陆成功') : console.log('登陆失败');

2.浏览器端函数

Alert() 弹出警示框

Prompt() 弹出提示框。需要使用变量来保存输入的值，类型是字符串。只能在浏览器端运行

练习：两次弹出提示框，输入数字，计算两次输入的和并打印出来

var num1=Prompt(‘input num1’);

var num2=prompt(‘input num2’);

num1=Number(num1);

num2=Number(num2);

console.log(num1+num2);

程序=数据+代码

程序执行分为顺序执行，选择执行，循环执行

**（1）if 语句 if(条件表达式){语句1} 语句2;**

如果if后的大括号只有一行，可以省略

**False的几种情况：0 NaN ‘’ undefined null ，其他情况为true（这5种情况不执行，其他情况不执行） !3>5 3是true（因为除开以上5种情况都是false），非3就是false，false就是0，就是0>5**

练习：声明变量保存个人签名，判断签名是否为空，如果为空，赋值为“这家伙很懒，什么也没留下”，打印签名

var str=’’;

if(str===’’){console.log(‘这家伙很懒，什么都没留下’)}

练习：声明一个变量，保存一个值，如果不是数字，要求打印“请输入一个数字”

判断一个值是否为非数字 isNaN。 是NaN –> true 不是NaN ->false

Var gg=’18a’;

If(isNaN(gg)){

Console.log(‘请输入一个数字’)

}

**（2）if-else语句**

**If（条件表达式）{语句1} else {语句2}**

var sex=0;

if (sex===0)

{console.log('女')

}else{console.log('男')};

练习：声明变量保存年龄，如果满18，打印成年人，否则打印未成年人

var age=17;

if (age>=18)

{console.log('成年人')

} else {

console.log('未成年人')

};

练习：声明变量保存用户名和密码，如果用户名为root，密码为123456，打印登录成功，否则打印登陆失败

var a='root';

var b='123456';

if (a==='root' && b==='123456')

{console.log('登陆成功')

} else {

console.log('登陆失败')

};

练习：使用弹出提示框分别输入商品的价格和数量，如果总价满500打9折，声明变量保存卡内余额800，如果足以支付，

弹出警示框pay success，否则弹出警示框 pay error

var geshu=Prompt('输入商品个数');

var price=Prompt('输入商品价格');

var total=price\*geshu;

if (total>=500)

{total\*=0.9;

};

var money=800;

if (money>=total)

{alert('pay success')

}else {alert('pay error')};

**（3）if-else嵌套**

**If(条件表达式){语句1} else if(条件表达式n){语句n}else{语句n+1 // 以上所有的条件表达式均为false}**

var type='少先队员';

if (type==='党员')

{console.log('这个人是党员')

}else if(type==='团员'){

console.log('这个人是团员')

}else if(type==='群众'){

console.log('这个人是群众')

}else{console.log('不合法的政治面貌')}

练习：声明一个变量来保存订单的状态码，根据状态码打印对应的汉字内容 //1- 等待付款 //2- 等待发货 //3- 运输中 //4- 已签收 //5- 已取消 //其他- 无法追踪

var bb='9';

if (bb==='1')

{console.log('等待付款')

}else if (bb==='2')

{console.log('等待发货')

}else if (bb==='3')

{console.log('运输中')

}else if (bb==='4'){

console.log('已签收')

}else if (bb==='5')

{console.log('已取消')

}else{console.log('无法追踪')}

var j=500;

if (j<10)

{console.log('普通客户')

}else if (j<100)

{console.log('优质客户')

}else if (j<500)

{console.log('金牌客户')

}else

{console.log('钻石客户')

} ！！！！注意小括号里的，不要重复大于小于

**（4）switch-case 语句**

是一种特殊的分支语句，可以根据一个表达式不同的值来选择执行不同的语句

Switch(表达式){case 值1: 语句1;break; case 值n;break; default: 语句n+1; // 以上所有的比较都是false}

根据订单的状态码打印对应的汉字

Var status=3;

Switch(status) {case 1: console.log(‘等待付款’);

Break;

Case 2: console.log(‘等待发货’);

Break;

Case 3:console.log(‘运输中’);

Break;

Default:

Console.log(‘无法追踪’)

}

表达式和case后的值比较使用的是全等，三个等号。只有这一种比较，就是全等

var score=10;

score=parseInt(score/10);

switch (score)

{

case 5:console.log('不及格');

break;

case 6:console.log('较差');

break;

case 7:console.log('中等');

break;

case 8:console.log('良好');

break;

case 9:console.log('优秀');

break;

default:

console.log('烂');

};

**对比if-else嵌套和switch-case语句**

相同点：两者都可以用于多项分支语言；

不同点：if-else 既可以判断相等，又可以判断不等，范围更广；switch-case结构上更为清晰，执行效率更高

4.流程控制——循环执行

循环：一遍又一遍执行相同或者相似的代码

循环的两个要素：

循环条件： 控制循环的次数

循环体：重复执行的相同或者相似代码

1. while循环

while(循环条件){//是一个布尔型的值

循环体

}

day 08

**While循环**

*循环条件*：控制循环的次数

*循环体*：重复执行的相同或者相似代码

*语法结构：*while(循环条件){循环体} 当循环条件为true，就执行循环体，再回头判断循环条件，直到为false

循环10次（除了循环条件，需要确定的是初始值，还有增量）

Var i=0; //确定初始值

While(i<10){console.log(‘hello’);

I++; //打印出来就是10个hello

Var i=0;

While(i<10){

Console.log(i);

I++;

}

注意：先打印再自增（++写在console前面）

练习：打印50,55,60，65,70,75,80

var r=50;

while (r<=80)

{console.log(r);

r+=5;

}

练习：打印66 63 60 57 54 51

var f=66;

while (f>=51)

{console.log(f);

f=f-3;

}

练习：打印1~100之间所有的奇数，要求使用判断

var d=1;

while (d<100)

{if(d%2===1){

console.log(d);

}

d++;

}

练习：计算1~100之间所有整数的和

//声明一个变量保存所有数字的和

Var sum=0;

Var i=1;

While (i<=100) {

Sum=sum+i;

I++;

};

Console.log(sum);

练习：计算1~100之间偶数的和

Var sum=0;

Var i=1;

While(i<=100){if(i%2===0){

Sum+=I;

}

I++;

}

Console.log(sum);

练习：使用循环计算10的阶乘

Var jc=1;

Var i=10;

While(i>=1){

Jc=jc\*I;

I--;

};

Console.log(jc)

练习：使用无限循环计算1~100之间所有的整数和

var f=1;

var s=0;

while(true){

s+=f;

if(f===100){break;}

f++;

}

console.log(s)

**break:**

在循环中，可以结束任何形式的循环

练习：声明变量保存任一个数字，在无限循环中弹出提示框，如果输入的值大于之前的数字，警示框说明big，如果输入的值小于之前的数字，警示small，否则警示框说明right，

结束循环。

//先引入js

<script src=’’></script>

Var num=18;

While(true){var str=prompt(‘input a number’);

If(str>num){alert(‘big’)}else if(str<num){alert(‘small’);}else if(isNaN(str)){alert(‘input a number’)}else{alert(‘right’)};break;

}

**Do-while循环**

语法形式：do{循环体}while(循环条件)

练习：打印1~10之间所有的整数

var i=1;

do

{console.log(i);

i++}

while (i<=10);

练习：循环打印30~40之间所有的整数

var a=30;

do

{console.log(a);

a++;

}

while (a<=40);

练习：循环打印50~40之间所有的整数

var s=50;

do

{console.log(s);

s--;

}

while (s>=40);

练习：打印1~100之间能被7整除的数字

练习: 计算1~100之间所有能被3整除的数字之和

var v=1;

var sum=0;

do

{if(v%3===0){sum+=v;}

v++;

}

while (v<=100);

console.log(sum)

练习：声明变量保存密码‘123456’，无限循环弹出提示框，输入密码，如果输入正确结束循环

var qq='123456';

do

{var str=prompt('input');if(str===qq){

break;

}

}

while (true);

利用循环条件来判断驶入是否正确

Var pwd=’123456’;0

Do{var str=prompt(‘input password’)}while(str!=pwd)

**For 循环**

For(初始值;循环条件;增量){循环体} 增量仍然是最后执行的

练习：打印1~10之间的整数

for (var i=1;i<=10 ;i++ )

{console.log(i);

}

练习：打印80 75 70 65 60

for (var i=80;i>=60 ; i-=5)

{console.log(i)

}

练习：计算1~100之间所有奇数的和

for (var i=1,sum=0; i<=100; i++)

{if(i%2===1){sum+=i;}

}

console.log(sum)

练习：计算1~20之间所有能被3整除的数字的乘积

for (var b=1,f=1;b<=20 ; b++)

{if(b%3===0){f\*=b}

}

console.log(f)

练习：打印2000-2100之间所有的闰年

for (var i=2000; i<=2100; i++)

{if(i%4===0 && i%100!==0 || i%400===0){console.log(i)}

}

循环产生1~5，最后拼接起来

for (var d=1,str='';d<=5 ;d++ )

{str=str+d

}

console.log(str)

str+=i+’\*5=’+i\*5+’ ’

多个循环条件

for(var i=1,j=7;i<=10,j>=1;i++,j--){console.log(I,j)}

练习：计算1/20 +2/18 +……10/2

for (var d=1,m=20,sum=0; d<=10;d++,m-=2 )

{sum+=d/m

}console.log(sum)

Continue和break

Continue：跳过后面的循环体代码，执行下一次代码

Break：结束循环，不会执行其他循环

练习：打印1~10，不包含5

for (var i=0; i<=10; i++)

{if(i===5){continue}

console.log(i)}

练习：计算1~100之间所有偶数的和，遇到奇数跳过

for (var i=1,sum=0;i<=100 ; i++)

{if(i%2===1){continue}

sum+=i;

}

console.log(sum)

练习：打印1~100之间所有的整数，不包含所有能被3整除或者4整除的数字

for (var p=1; p<=100; p++)

{if(p%3===0 || p%4===0){continue}

console.log(p)

}

**循环嵌套**

外层：控制循环的行数

内层：控制循环的列数

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

//控制循环的行数

for (var j=1;j<=9 ;j++ )

{//控制每一行的列数

for (var i=1,str='';i<=9 ; i++)

{str+='\*'

}

console.log(str)

}

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

for (var i=1; i<=5;i++ )

{for (var j=1,str='';j<=i ;j++ )

{str+='\*'

}

console.log(str)

}

99乘法表

for (var i=1; i<=5;i++ )

{for (var j=1,str='';j<=i ;j++ )

{str+='\*'

}

console.log(str)

}

打印倒着的乘法表

for (var i=9;i>=1 ;i-- )

{for (var j=i,str='';j>=1 ; j--)

{str+=j+'\*'+i+'='+i\*j+' ';

}

console.log(str)

}

打印（2000-2100）的前十个闰年

for (var i=2000,sum=0;i<=2100 ; i++)

{if(i%4===0 && i%100!==0 || i%400===0){console.log(i);

sum++;if(sum===10){break;}}

}

计算1-100之间所有整数的和，当和大于4000，提前结束循环，打印当前的和

for (var i=1,sum=0;i<=100;i++ )

{sum+=i;

if(sum>4000){break;}

}

console.log(sum)

day 09

函数

函数分为系统函数和自定义函数

Function：功能体，函数（方法），接收若干个数据，返回处理的结果 ——用于封装多次执行的代码

1. 创建普通函数

Function 函数名称(){

函数体 //封装的代码

}

调用：函数名称()

练习：创建函数，计算两个数字相加的和（2.3），调用三次

Function ab(){console.log(2+3)}

Ab();

Ab();

Ab();

//练习：创建函数，封装计算1~100之间的所有整数的和，调用三次

function getsum(){

for (var i=1,sum=0;i<=100 ;i++ )

{sum+=i;

}

console.log(sum)

}

getsum();

getsum();

getsum();

（2）创建带有参数的函数

Function 函数名称(参数列表){//用于接收传递的数据 函数体}

调用：函数名称(参数列表) //实际要传递的数据

function add(a,b形参){

console.log(a+b)

}

add(2,3)实参，会赋值给形参

练习：创建函数，计算1~任意数字之间所有整数的和，调用多次

function cc(w){

for (var i=1,sum=0;i<=w ; i++)

{sum+=i

}

console.log(sum);

}

cc(50);

cc(60);

多个参数之间用逗号隔开，实参的值可以传递0个或者多个，如果未赋值，形参的值为undefined

练习：创建函数函数，计算任意两个年份之间的闰年个数

function hh(a,b){

for (var i=a,sum=0;i<=b ; i++)

{if(i%4===0 && i%100!==0 || i%400===0){sum++}

}

console.log(sum)

}

hh(2000,2009);

hh(1990,2000);

1. 创建带有返回值的函数

Function 函数名称（参数列表）{

函数体；

Return值；//返回值，函数调用后产出的结果

}

函数名称（参数列表）

如果函数中没有return，或者return后没有任何的值，则返回undefined，return执行后，后续的代码不会执行

练习：创建函数，传递两个参数，返回任意两个数字的最大值

function getmax(a,b){

if(a>b){return a}else{return b}

}

var res=getmax(3,6)

console.log(res)

练习：创建函数，传递三个参数，返回任意三个数字的最大值

对比break和return。

Break用于结束switch语气结束循环，return用于结束函数后续代码执行

练习：创建函数，传递一个参数，要求查看任意年份是否为闰年，如果是闰年，返回true，或者返回false

function w(a){

if(a%4===0 && a%100!==0 || a%400===0){return true}else

{return false};

};

console.log(w(2019))

//计算1~任意数字之间所有整数的乘积

Function getjc(n){

For(var i=1,jc=1;i<=n;i++){jc\*=i}

Return jc

}

Console.log(getjc(5));

//计算1~任意数字之间所有整数的和，并返回结果

Day 7.11

1.函数的创建

|  |
| --- |
| 函数声明创建  function fn(){} |
| 函数表达式创建  Var fun=function(){} |

函数名称fun/fn本质上都是一个变量，只是保存的是函数

对比函数声明和函数表达式的区别

函数声明存在函数提升，可以在任意位置调用

函数表达式只是存在变量提升，需要先创建再调用

练习：使用函数表达式创建函数，计算任意两个数字之间所有整数的和，返回结果。

(2)匿名函数自调用

目的：创建函数作用域，防止污染全局。

|  |
| --- |
| (function(形参列表){  //函数体中的变量和函数都不能被外部访问  })(实参列表) |

(3)回调函数、

将匿名函数以实参形式传递，形参就是传递的匿名函数的名称

|  |
| --- |
| Function fn(a){  a();//调用fn的时候，匿名函数会赋值给形参a，a就是函数名称。  }  fn(function(){  return 2;  }) |

2.全局函数

parseInt()/parseFloat()/isNaN()

encodeURI(); 对URI中的汉字进行编码

decodeURI(); 对已编码的URI进行解码

isFinite(); 检测一个值是否为有限值。 是返回：ture，不是返回：false

2/0=Infinity 无穷大

eval() 执行字符串中的表达式（功能过于强大，不仅执行运算还可以执行代码）

练习：弹出提示框，输入一组运算，使用eval来执行输入的运算。

3.对象

是一组属性和方法的集合

一台电脑：属性颜色，大小，分辨率，方法有玩游戏…

1. js中的对象

内置对象：js提供的

宿主对象：根据不同的执行环境划分。

自定义的对象：自己创建的对象

1. 自定义对象

对象属性名与属性值之间用冒号隔开

使用大括号{}建立空对象

多组属性之间用逗号隔开

属性名中的引号可加可不加，如果含有特殊字符必须添加

练习：1.创建一个员工对象，包含有编号，姓名。性别，生日，工资，部门。

2.创建一个手机对象，包含有颜色，品牌，尺寸，内存。

(3)访问对象中的属性

对象.属性名

对象[‘属性名’]

如果属性名不存在，返回undefined

练习：创建图书对象，包含编号，标题，作者，价格，出版时间。修改图书的价格，打印图书的作者，添加出版社属性

(4) 使用内置构造函数创建对象

new Object() 创建一个空对象

需要单独添加每个属性

练习：创建一个汽车对象，包含品牌，颜色，价格，排气量。

(5)遍历对象的属性

|  |
| --- |
| for (var key in 对象){  key代表对象中的每一个属性名  对象[key] 每一个属性值  } |

练习：创建对象，包含有若干个成绩，遍历对象的属性，计算总成绩。

(6)对象中的方法

|  |
| --- |
| var person={  name:'汤姆',  say:function(){  //console.log('我叫'+person.name)  //console.log('我叫'+this.name) //this 指向当前的对象  console.log('我叫汤姆');  }  }  //调用对象中的方法  person.say(); |

课后任务：

1. 复习今天内容，整理思维导图
2. 练习

创建一个圆对象，含有半径和圆周率属性;添加计算周长和计算面积的方法，返回结果。

1. 课后预习数组创建，遍历，API（方法）。

Day 7.12

复习

1.回调函数

函数自调用

(Function(){})(实参)

回调函数

Fn(function(){

//可以写完整业务逻辑

})

2.对象

遍历属性：for(var key in 对象){

Key 代表对象中的每一个属性名

对象[key] 每一个属性值

}

一丶检测属性

1.对象.hasOwnProperty(‘属性名’) true存在 false 不存在

2.对象.属性===undefined true 不存在 false 存在

练习：创建一个手机对象，添加睡醒有品牌，颜色；添加方法用于介绍手机，

返回‘手机的品牌为？颜色？价格？’，如果价格属性不存在，设置价格为4999。

//原始类型存储（栈内存）

var a=1;

var b=a;

//改变a的值

a=3;

console.log(a);

//引用类型存储（堆内存）

/\*把对象car的地址复制给了car2，两个变

量指向同一个对象，无论使用一个变量

修改属性，另外一个也会改变。\*/

var car={

pinpai:'奥迪',

price:150000,

}

var car2=car;

car2.pinpai='红旗';

car.price=522000;

console.log(car);

console.log(car2);

//这里暂时没有找到复制到另一个地址的方法 -> 销毁对象，释放内存空间

//原理：给对象赋值为null，对象不再被地址所引用，自动销毁

car=null;//不再指向任何对象

car2=null;

console.log(typeof null);

数组

数组里有多个元素，是个元素集合，数组也是一个数据。

(1)创建数组

1) 数组字面量

2)new Array()

(2)访问

1) Array[下标]

2) Array.length

1. 获取数组长度

Array.length

1. 在末尾添加数组元素

Array[Array.length]=?;

(3)数组索引下标数字从0开始，因此数组最大索引号数字为Array.length-1。

(5)数组的分类

数组分为索引数组和关联数组。

索引数组以0和0以上的整数作为下标。

关联数组以字符串作为下标，单独添加元素。

|  |
| --- |
| //索引数组：数字作为下标  //关联数组：字符串作为下标  var emp=[];  emp['eid']=15;  emp['ename']='king';  emp['sex']='男';  console.log(emp); |

4.遍历数组

（1）for-in

|  |
| --- |
| for(var key in score){  zong+=score[key];  }  console.log(zong);//可以遍历索引数组和关联数组 |

1. for循环

|  |
| --- |
| for (var i=0,total=0;i<score.length;i++)  {  total+=score[i];  }  console.log(total); |

只能遍历索引数组。

练习：创建数组，包含多个姓名，把姓名为‘King’的全部替换成‘金’。

|  |
| --- |
| var name=['金','king','s',];  for (var i=0;i<name.length;i++)  {  if(name[i]=='king'){  name[i]='金';  }  }  console.log(name); |

练习：创建数组，包含多个姓名，统计King出现的次数

|  |
| --- |
| var name=['金','king','s','king','king'];  for (var i=0,times=0;i<name.length;i++)  {  if(name[i]==='king'){  times++;  }  }  console.log(times); |

练习：打印数组中的最大值

|  |
| --- |
| var array=[12,15,85,885,885,44,25,54,545,212,54,5,454,8,48,12,4854,];  var max,b,c;  for(var i=1;i<array.length;i++){  max=array[i-1];  if(array[i-1]<array[i]){  max=array[i];  }  }  console.log(max); |

练习：创建函数，传递一个参数（数组），返回这组数字的平均值

5.数组API（方法）

API：应用程序编程接口，预定义好的一些方法和函数。

(1)toString():数组中的元素按照指定的字符分割为字符串

(2)join(‘-’):数组中的元素按照指定的字符连接为字符串，如果参数为空，默认按照逗号分割。

(3)concat(arr2,arr3):拼接多个数组

(4)slice(start,end):截取数组中的元素,start开始的下标，end结束的下标，不包含end;如果是负数，表示倒数第几个。

(5)splice(start,count,val1,val2..):删除数组中的元素。start:开始的下标，count：删除的长度，

val1,val2,..:在截断数组位置添加新的元素。

课后任务

1. 复习今天的内容，整理思维导图
2. 练习

创建函数，传递两个参数（数组，值），如果值存在于数组中，返回满足条件的第一个的下标，找不到返回-1.

创建数组，使用遍历数组翻转数组中元素

创建数组，包含一组数字，按从小到大排序输出，（冒泡排序）

（3）预习

复习

1. 数组翻转排列生成新的数组。
2. 数组按从小到大，从大到小排列（冒泡排序）。

Day7.15

## 数组复习及部分API

1. 数组翻转排列生成新的数组。

|  |
| --- |
| //遍历数组，找对应的下标0~3 1~2 ... (翻转数列)  var arr=['a','b','c','d','c'],arr2=[];  for (var i=0;i<arr.length;i++ )  {  arr2[i]=arr[arr.length-i-1]  }  console.log(arr2);  //API直接实现  console.log(arr.reverse()); |

2. 数组按从小到大，从大到小排列（冒泡排序）。

|  |
| --- |
| //练习：创建数组，包含一组数字，让数字实现从小到大排序（冒泡排序）  var arr=[23,9,78,6,45],max;  for (var i=0;i<=arr.length-1;i++ )  {  //比较4轮  // 0 4  // 1 3  // 2 2  // 3 1  // 4 0  for (var j=1;j<arr.length-i;j++ )  {  if(arr[i]>arr[i+j]){  max=arr[i];  arr[i]=arr[i+j];  arr[i+j]=max;  }  }  }  console.log(arr);  //API实现数字按大小排序  console.log(arr.sort());//sort默认按照Unicode排序  //按照数字从小到大排序  console(arr.sort(function(a,b){return a-b}));  //按照数字从大到小排序  console(arr.sort(function(a,b){return b-a})); |

总结：

（1）arr.sort(): 默认按照Unicode排序

1.按照数字从小到大排序

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

2. 按照数字从大到小排序

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

（2）arr.reverse(): 翻转数列

其他API：

push()：往数组末尾添加元素，返回数组的长度。

pop():删除数组末尾的元素，返回删除的元素。

unshift():往数组的开头添加元素，返回数组长度。

shift():删除数组开头的元素，返回删除的元素。

splice():往数组中间添加删除元素，没有返回值。例：splice(1,2,’aaa’,’bb’) 自数组元素arr[1]起，删除2个元素，并添加2个新元素。

3.二维数组

1.数组中的元素也是数组

|  |
| --- |
| var arr=[[],[],[],]; |

2.访问二维数组中的元素

arr[][];

## 字符串

### new String()和String()

定义字符串

var str=’java’;

new String(数据)

String(数据)

包装对象：目的将原始类型数据像引用类型数据，可以调用属性和方法。

js中有三种包装对象：String Number Boolean

将任意的数据转为字符串

new String(数据)：返回的是一个包装对象，使用和字符串一样使用。

String(数据):将数据强制转换成字符串。

### 转义字符

转换字符本身的意义

\’ 将引号转成普通字符

\n 将字符n转成换行符

\t 将字符t转成制表符（tab键）

练习：打印出现’ welcome to chi\na’

|  |
| --- |
| console.log('welcome to chi\\na'); |

### 字符串中的API

length 获取字符串的长度

charAt(下标) 获取下标对应的字符,也可以使用数组类似写法访问。 例：str[下标] / str. charAt(下标)

charCodeAt() 获取某个字符的Unicode码

indexOf(value,start) 查找某个字符串的下标，value：要查找的字符（串），start：开始的下标，默认为0；如果找不到默认返回-1。

lastIndexOf(value) 倒着查找某个字符（串）的下标，找不到返回-1

练习：声明一个变量保存邮箱，检测该变量中是否含有@。如果有，打印‘合法的邮箱’，没有打印‘非法的邮箱’

toUpperCase(); 将英文字母转大写

toLowerCase(); 将英文字母转小写

练习：声明变量保存4个英文字母（大小写），无限循环弹出提示框，输入验证码（不区分大小写），如果输入正确结束循环。

slice(start,end) 截取字符串，start开始的下标（包括），end结束的下标（不包括），如果只有一个参数，表示这个参数所示位置到末尾，负数表示倒数

练习：声明变量保存邮箱，分别截取邮箱的用户名和服务器名称[tom123@163.com](mailto:tom123@163.com)

substr(start,long) 截取字符串，start开始的下标（包括），long表示截取长度。如果只有一个参数，表示这个参数所示位置到末尾，负数表示倒数

substring() 功能与slice一样，区别在于：（1）substring参数不能为负数，为负数时自动设置为0。（2）slice两个参数start必须小于end，substring会自动识别两个参数大小进行截取。

//练习：声明变量保存一个单词（heLLo），将首字母转大写，其余转小写。用substring来写

split(sep) 将字符串按照指定的字符分为数组，sep指定的字符。

练习：使用split截取邮箱中的用户名和服务器名称[tom123@163.com](mailto:tom123@163.com)

### 匹配模式（掌握）

作用：用于查找，替换字符串

search(str) 用于查找满足条件的第一个字符串对的下标，如果找不到返回-1，类似indexOf。

/China/i i ->ignore 忽略大小写

match(str) 用于查找满足条件的第一个或者所有的，返回数组

/China/ig i ->ignore 忽略大小写 g->global 全局查找

|  |
| --- |
| var str='china是世界上人口最多的国家，China的互联网高度发达，CHINA有好多的程序员，欢迎来china学开发';  //查找china  //console.log(str.search('China'));  //正则表达式写法  //i ->ignore 忽略大小写  //console.log(str.search(/China/i));  //g -> global 全局的  console.log(str.match(/China/ig));  //查找并替换  console.log(str.replace(/china/ig,'中国')); |

replace(str1,str2) 用于查找并替换字符串，str1 要查找的字符串，str2 要替换的字符串

# math对象

没有字面量形式，不需要使用new创建，直接使用math

|  |
| --- |
| //圆周率  //console.log(Math.PI);  //绝对值  //console.log(Math.abs(18-20));  //取整 3.14  //向上取整  //console.log(Math.ceil(3.14));// ceil：楼上  //向下取整  //console.log(Math.floor(3.14));//floor:楼下  //四舍五入取整  //console.log(Math.round(3.44)); //round: v. 把(数字)四舍五入  //获取一组数字的最大值  //console.log(Math.max(23,6,78,45));  //获取一组数字的最小值  //console.log(Math.min(23,6,78,45));  //x的y次幂  //console.log(Math.pow(5,3));  //取随机>=0 0~0.999...  //console.log(Math.random());  //取0~9之间的一个整数  console.log(Math.floor(Math.random()\*10)); |

课后任务

（1）复习今天的内容，整理思维导图

（2）练习

1.将一句英文每个单词的首字母大写，其余小写

how are you

2.创建数组，包含0~9，a~z，随机取4个字符，放入到一个新数组中。

原理：36个整数 下标0~35 随机取4个下标

（3）预习Date对象

day 7.16

趣玩小项目：

双色球：

红球：在1~33之间随机取6个，不可重复。

篮球：在1~16之间随机取1个，可以和红球重复。

## Date对象

### 创建Date对象

new Date('2019/7/16 9:57:30'); 获取显示时间晚8个小时，东八区。

new Date(2019,0,32,9,57,30); 月份0~11，月日时等都可以超，超则进1.

new Date(); 获取当前的系统时间

new Date(147585261500); 从计算机元年1970.01.01 00:00+参数所示毫秒

### 获取存储的日期时间

|  |
| --- |
| //获取日期时间的各个部分  var d=new Date();  //获取年/月/日/小时/分/秒/毫秒  console.log(d.getFullYear());  console.log(d.getMonth()+1);  console.log(d.getDate());  console.log(d.getHours());  console.log(d.getMinutes());  console.log(d.getSeconds());  console.log(d.getMilliseconds());  //获取距离计算机元年的毫秒数  console.log(d.getTime());  //获取星期0~6  console.log(d.getDay()); |

练习：创建Date对象，保存2019/3/1，打印‘2019年03月01日 星期五’。

|  |
| --- |
| var d=new Date('2019/3/1');  var year=d.getFullYear();  var yue=d.getMonth()+1;  var ri=d.getDate();  var weekday=d.getDay();  var arr=['星期日','星期一','星期二','星期三','星期四','星期五','星期六',];  if(yue<10){  yue='0'+yue;  }  if(ri<10){  ri='0'+ri;  }  /\*  switch(weekday){  case 0:weekday='日'; break;  case 1:weekday='一'; break;  case 2:weekday='二'; break;  case 3:weekday='三'; break;  case 4:weekday='四'; break;  case 5:weekday='五'; break;  case 6:weekday='六'; break;  default: break;  }  \*/  console.log(year+'年'+yue+'月'+ri+'日 '+arr[weekday]); |

练习：计算当前系统时间距离2019/9/13还有？天？小时？分

|  |
| --- |
| //创建当前系统时间的对象和2019/9/13的对象  //计算两者相差的毫秒数 getTime()  var d=new Date('2019/09/13');  var now=new Date();  //var time=d.getTime()-now.getTime();  //console.log(time);  //两个对象相减，代表相差的毫秒数，计算机服务器里存储时间用的毫秒。  //console.log(d-now);  var time=d-now;  //把相差的单位转成秒  miao=Math.floor(time/1000);  //计算相差的天数，把相差的值换算成天  var tian=Math.floor(miao/(24\*60\*60));  console.log(tian);  //计算相差的小时  //去除time中的天数  var hoursMiao=miao%(24\*60\*60);  var hours=Math.floor(hoursMiao/(60\*60));  console.log(hours);  //计算相差的分  //去除hoursmiao中的小时数  var fenMiao=hoursMiao%(60\*60);  var fen=Math.floor(fenMiao/60);  console.log(fen); |

（3）获取本地字符串格式

toLocaleString() //2019-7-16 14:23:06 根据本地时间格式，把 Date 对象的日期加时间部分转换为字符串。

toLocaleDateString() //2019-7-16 根据本地时间格式，把 Date 对象的日期部分转换为字符串。

toLocaleTimeString() // 14:23:06 根据本地时间格式，把 Date 对象的时间部分转换为字符串。

(4)设置日期时间（）

|  |
| --- |
| var d1=new Date('2019/7/16 14:28:30');  //设置30分钟以后  d1.setMinutes(58);  d1.setHours(17);  d1.setMonth(11);  //没有setDay方法  d1.setTime(25441545);  console.log(d1.toLocaleString());  //还有很多：setFullYear/setSeconds/setMilliseconds…. |

|  |
| --- |
| var d1=new Date('2019/7/16 14:28:30');  //var d2=d1; //d1和d2用的同一个地址（堆内存）  //复制一个d1对象,不指向同一个地址  var d2=new Date(d1);  //设置30分钟以后  d1.setMinutes(58);  d1.setHours(17);  d1.setMonth(11);  //没有setDay方法  //d1.setTime(25441545);  //3年后：先获取当前的年份，然后加3，把结果设置为当前的年份  d1.setFullYear(d1.getFullYear()+3);  //设置一星期后  d1.setDate(d1.getDate()+7);  console.log(d2.toLocaleString());  //练习：创建Date对象，保存员工的入职时间‘2019/7/16’，合同期为3年，计算到期时间  var d1=new Date('2019/7/19');  //到期时间，复制一个入职时间  var d2=new Date(d1);  //设置3年后到期  d2.setFullYear(d2.getFullYear()+3);  var d3=new Date(d2);  d3.setMonth(d3.getMonth()-1);  if(d3.getDay()===0){d3.setDate(d3.getDate()-2);}  else if(d3.getDay()===6){d3.setDate(d3.getDate()-1);}  console.log('ruzhishijian'+d1.toLocaleString()+'\ndaoqishijian'+d2.toLocaleString());  console.log(d3.getDay());  console.log(d3.toLocaleString()); |

## Number对象

new Number(数据) 将数据转为数值，返回对象（包装对象）

Number(数据) 将数据转为数值，返回数值

String()/toString()/num+'' 转字符串的方法

num.toString(8) 将数值转成8进制

Number.MAX\_VALUE 计算机能存储的最大值

Number.MIN\_VALUE 计算机能存储的最小值

num.toFixed(num) 保留一个数字小数点后n位

num.toString(n) 将数字转成字符串，设置进制n:2/8/16进制

## Boolean对象

new Boolean(数据) 将数据转为布尔型，返回对象

Boolean(数据) 将数据转为布尔型，返回布尔型

!!数据 隐式转为布尔型

false: 0 NaN '' undefined null

true:除了以上的几种都是true

## 错误处理

1. SyntaxError 语法错误：错误的使用了中文，缺少括号，缺少括号等；js在执行程序前，先检查是否有语法错误。

2. ReferenceError 引用错误：使用了未声明的变量，影响后边代码执行

3. TypeError 类型错误：错误的使用了括号，调用了不存在的函数或方法，影响后边代码执行。

4. RangeError 范围错误：参数的使用超出了范围，影响后边代码的执行

//var arr=new Array(-3);

5.throw 自定义错误，影响后边代码执行。

|  |
| --- |
| 错误处理（）  try  {  可能产生的错误  }  }  catch (err) //catch捕获错误:只有try里面报错，才会执行这里边的代码。err:收集错误信息  {  console.log(err);  num=10;  }  可以使用if语句代码代替 |

## ES6

ECMAScript6

<http://es6.ruanyifeng.com/> 阮一峰：ECMAScript6入门

(1)块级作用域

使用let关键字声明的变量，在块级作用域下，不能被外部访问，不存在变量提升。

|  |
| --- |
| 块级作用域：{}, if, if-else, while, do-while, for…都是 |

练习：计算1~100之间所有整数的和，使用let声明变量，计算总和。

|  |
| --- |
| let sum=0;  for (let i=1;i<=100;i++)  {  sum+=i;  }  console.log(sum); |

(2)箭头函数

是回调函数的另一种写法，和之前的匿名函数不完全一样

|  |
| --- |
| sort((a,b)=>{  return a-b;  } ) |

如果箭头函数的函数体中只有一行代码，并且是return形式的，可以简化为sort((a,b)=>a-b)

课后任务

(1)复习今天的内容，整理思维导图

(2)练习

计算2019/7/16 17:57:45 距离 2019/10/1 6:15:30 相差的天，小时，分钟，秒

(3)预习 服务器概念 node.js

1.ES6

(1)函数增强

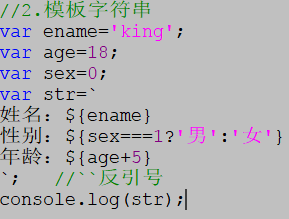
ES6 中允许给形参设置默认值，如果没有传递实参，则自动调用形参的默认值。

注意事项：默认值通常是从最后一个形参开始赋值。

|  |
| --- |
| function add(a,b,c=0) |

(2)模板字符串

|  |
| --- |
| `在模板字符串之间可以放任何形式的字符串 ${js表达式}`; |



练习：创建图书对象，包含有图书的编号，名称，价格，上架时间，是否在售（1/0）；使用模板字符串打印图书信息

|  |
| --- |
| var bid='1';  var title='红楼梦';  var price=25;  var shelfTime='2019-7-17';  var d=new Date(shelfTime);  var sale=1;  var book=`  编号：${bid}  名称：${title}  价格：${(price\*0.9).toFixed(2)}  上架时间：${d.getFullYear()}年${d.getMonth()+1}月${d.getDate()+'日'}  是否在售：${sale===1?'9折正在热卖中..':'已下架'}  `;  console.log(book); |

day 7.17

1.ES6

(1)函数增强

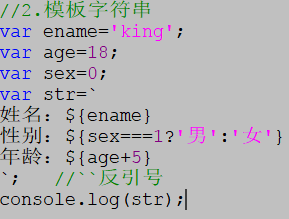
ES6 中允许给形参设置默认值，如果没有传递实参，则自动调用形参的默认值。

注意事项：默认值通常是从最后一个形参开始赋值。

|  |
| --- |
| function add(a,b,c=0) |

(2)模板字符串

|  |
| --- |
| `在模板字符串之间可以放任何形式的字符串 ${js表达式}`; |



练习：创建图书对象，包含有图书的编号，名称，价格，上架时间，是否在售（1/0）；使用模板字符串打印图书信息