JS背景知识

1995年,JavaScript开始出现

1997年,JavaScript 提交给了ECMA,出现了ECMAScript标准规范.

2009年遵守Common JS,JS开始向后端发展---NODEJS

应用场景

浏览器端的交互效果

创建WEB服务器

开发移动APP

特点

可以使用任意的编辑器编写代码

弱类型语言

跨平台,跨操作系统

搭建开发环境

在客户端中直接运行(客户端)

需要借助HTML运行

<script src =”02.js”></script>

在nodejs下运行(服务器端)

JS书写规范

区分大小写

每行结束的分号,可加可不加.推荐都加上分号

分为单行 // 和多行注释 /\*….\*/

变量和常量

变量:就是用于存储数据的容器

1),声明变量使用关键字 var -> variable

变量的命名规则

1), 由字母,数字,美元符号$,下划线\_组成,不允许以数字,特殊符号开头.

驼峰命名法 username 第二个单词首字母大写

下划线命名法 user\_name 单词之间用下划线连接

**不能使用关键字或者保留字作为变量名**

**变量声明未赋值,此时变量的值为undefind(未定义)**

**弱类语言特点: 声明的变量可以保存任意类型的数据**

**常量**

**一旦声明就不能重新赋值**

**使用const 关键字声明**

**数据类型**

**原始类型**

**数值型,字符串型,布尔型,未定义型(undefined),null**

**1).数值型**

**整型**

**占用4个字节,8进制(以0开头),10进制,16进制(0x开头)**

**0 1 2 3 4 5 6 7 8 9 10 十进制**

**0 1 2 3 4 5 6 7 8 10 11 12 8进制**

**0 1 2 3 4 5 6 7 8 9 a b c d e f 10 16进制**

**浮点型**

**占用8个字节**

**普通小数**

**3.1456789 \* 1000🡪314.56789**

**指数型小数**

**3.1456789e3🡪3145.6789**

**2),字符串类型**

**数据被引号所包含**

**查看一个字符的Unicode码**

**Var str =’a’;**

**Console.log(str.charCodeAt());**

**3),布尔型**

**在程序中表示真或假的结果使用布尔型**

**True/false**

**例如:用户是否在线,是否为VIP,是否在售**

**比较运算的结果是布尔型的值**

**4),未定义型**

**声明了变量未赋值,结果就是undefined**

**未定义型只有一个值 undefined**

**5),null**

**用于释放内存空间,只有一个值.**

**数值型 🡪 number**

**字符串类型🡪 string**

**布尔型 🡪 Boolean**

**未定义型 🡪 undefined**

**空(null) 🡪 object(对象)**

**引用类型**

**数据类型 隐式转换**

**1),数字+字符串, 数字会转成字符串**

**1+'a' 🡪 '1a'**

**2),数字+布尔型,布尔型会转成数字 true 1 / false 0**

**1 + true = 2 1+ false = 1**

**3),字符串+布尔型,布尔型会转成字符串**

**'hello'+true = 'hellotrue'**

**总结**

JS中+ - \* / 号的应用

1),执行加法运算

2),执行字符串拼接

3),任何数据和字符串执行加法运算,结果都是字符串

会尝试将符号两端的数据转为数值型进行运算,如果数据中出现了非数字,结果为NaN🡪 Not a Number(数值类型为Number)

数据类型🡪强制转换

1),将任意的类型的数据转成整型,如果转换失败,返回NaN

parseInt();

2),将任意类型数据转化成浮点型,转换失败返回NaN

parseFloat();

3),将任意类型数据转为数值型

Numer('1.5') //1.5

Number('1.5a')//NaN

如果要转化的字符串中含有非数字,则返回NaN

4),将任意类型数据转为字符串

转换成字符串 tostring() 方法括号中 还可以转换成2 8 16进制

运算符

定义:由运算符连接操作的数据,所组成的结构成为表达式.

1),算术运算符

+ - \* /

%(取余)

//取奇数 --> 和2取余的结果为1

//取偶数 --> 和2取余的结果为0

++ 子增,在原来的基础之上加1, -- 自减,在原来的基础上-1

Console.log(num++); 先打印num的值,再自增

Console.log(++num);先自增,再打印num的值.

var a = 3;

console.log(a++ + ++a);//8

console.log(a-- + --a);//8

2),比较运算符

> < >= <= == != ===(全等) !==(不全等)

比较运算符结果返回布尔型的数据

console.log(1 > 3);//false

console.log(2 =='2');//ture,比较值是否相等

console.log(2 === '2');//false,比较值和类型是否相等

console.log('3'.charCodeAt());// unicode 51

console.log('10'.charCodeAt());// unicode 49

console.log('3'>'10');//true

console.log('乾隆' > '康熙');//false

console.log('乾隆'.charCodeAt());//unicode 20094

console.log('康熙'.charCodeAt());//unicode 24247

字符串字符,比较的是Unicode码

‘张无忌’>‘张翠山’ //false

如果首字符相同,会比较第2个字符的Unicode码

console.log(3 > '10a'); //false

console.log(3 < '10a');//false

console.log(3 == '10a');//false

console.log(3 !== '10a');//ture

console.log(NaN == NaN);//任何值和NaN相比较都是false

3),逻辑运算符

并且 && 关联的两个条件都满足,结果ture, 否则 false

或者 || 关联的两个条件满足其一,否则 false

非 ! 取反

//判断一个人是否处于工作年龄 -->18岁以上,60岁以下

var age = 27;

console.log(age>=18 && age<=60);//true

//练习:判断分数是否为中等 70-80以下

var score = 90;

console.log(score>= 70 && score <80);//false

//练习:判断购票半价标准 12岁以下,65岁以上

var age = 15;

console.log(age<12 && age > 65);

//取反

console.log(!true);//false

console.log(!(3<2));//true

逻辑短路

&& 如果第1条件为false的时候, 就不需要执行第2条件

|| 如果第1条件为ture的时候,就不需要执行第2个条件

//练习 判断一个人的年龄是否是成年人,如果是,打印成年人

var age = 18;

age >= 18 && console.log('成年人');

if(age >=18){

console.log('成年人');

}

else{

console.log('未成年 or 挂了');

}

位运算符(了解)

在使用位运算符的时候,将数字转为二进制然后运算

1 10 11 100 101 110 111 1000 1001 1010 1100...

1 2 3 4 5 6 7 8

101010 换成10进制是多少

32 8 2 10 = 2e1 100= 2e2 1000 = 2e3 10000= 2e4

按位与(&) 3&5

3 011

5 101

1. 上下两位都是1,结果是1, 否则是0

5&7 1 10 11 100 101 110 111 1000 1001 1010 1100 1101 1110 10001 10010

5🡪101

7🡪111

101 🡪 5

按位或 (|) 3|5 上下两位含有1,结果是1,否则0

011

101

111🡪7

按位异或(^) 3^5 上下两位不同为1,相同是0

011

101

110🡪6

按位左移(<<) 15 <<1 🡪 1111 🡪11110 🡪 30

在最后补0 , 不断变大

按位右移(>>) 15>>1 🡪 1111 🡪 111 🡪7

删除最后一位,不断变小

赋值运算符

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

//练习:所有商品打八折,假如总价1000,计算应付多少

var price = 1000;

price \*=.8;//0.8中的0可以省略

console.log('最终付款:',price);

**三目运算符**

**一目运算符: ++ -- !**

**二目运算符: + - \* / % < > <= >= != == === && || !== & | << >> += = -= \*= /= %=**

**三目运算符:由3个表达式或数据组成的结果**

**条件表达式(结果为true/false) ? 表达式1: 表达式2:**

**如果条件表达式为true,执行表达式1**

**如果条件表达式为false,执行表达式2**

**浏览器端函数**

**Alert() 弹出警示窗口**

**Prompt() 弹出提示窗口**

//练习:使用提示窗口 弹出两个值 ,然后相加,并打印结果; 注意数据类型

var a = prompt('please input first number');

var b = prompt('please input second number');

console.log(typeof(a));

console.log(typeof(b));

a = parseInt(a);

b = parseInt(b);

console.log(a + b);

**流程控制**

**程序 = 数据 + 算法**

**程序的执行方式:**

**顺序执行,选择执行,循环执行**

**1), if语句**

**语句1**

**If(条件表达式){**

**语句2**

**}**

**If后选择执行的语句,如果只有一行,可省略大括号**

**语句3**//如订满30,减15

if(price>=30){

//在当前基础之上减15

price -=15;

}

console.log('价格为:',price);

//练习:超市购物,满1000打8折,计算应付金额

var total = 1001;

if(total>=1000){

total \*=.8;

};

console.log('付款:',total);

//练习:在个人签名中如果签名为空,就显示这家伙很懒,否则正常显示签名内容

var details = '';

if(details ==''){

console.log('这家伙很懒,啥也没写');

}

else{

console.log(details);

}

var a = '';

//出现false的特殊情况

//undifiend null NaN 0 '' !''

if(a == ''){

console.log('用户名不能为空')

}

if (!a){

console.log('用户名不能为空');

}

**2),if-else**

**语句1;**

**If(条件表达式){**

**语句2**

**}else{**

**语句3**

**}**

**语句4**

**如果条件表达式为true,执行语句2;否则执行条件3**

//练习:声明变量保存商品总额,如果总额满1000,打8折,打印应付金额;付款:声明变量保余额

// 如果余额小于应付金额,打印余额不足,否则打印支付成功

var price = 850;

if(price >= 1000){

price \*= .8;

}

console.log('应付金额为:',price)

var money = 1000;

console.log('包里只有:',money);

if(money >= price){

var cheap = money - price;

console.log('成功支付:',price,'包里还剩:',cheap);

}else{

console.log('余额不足,咱还是回家呆着吧');

}

//练习: 声明变量保存用户名和密码,如果用户名为root并且密码是123456,打印登陆成功;

// 否则打印登陆失败

var userNanme = 'root';

var userPwd = '123456';

if(userNanme == 'root'){

if(userPwd == '123456')

console.log('登陆成功');

else

console.log('密码错误');

}else

console.log('用户名或密码错误');

userNanme == 'root', userPwd == '123456' ?

console.log('登陆成功'):console.log('登陆失败');

**3) if-else嵌套**

//判断周几,想吃什么

var day = 7;

if(day == 0){

console.log('周日,我们吃火锅吧');

}else if(day == 1){

console.log('周一,我们吃牛排吧');

}else if(day == 2){

console.log('周二,我们吃工作餐吧');

}else if(day == 3){

console.log('周三,我们吃霸王餐吧')

}else if(day == 4){

console.log('周四,我们吃点小面')

}else if(day == 5){

console.log('周五,快放假了,晚上聚一下吧');

}else if(day == 6){

console.log('周六,大吉大利,晚上吃鸡')

}else{

console.log('周',day,'你吃个锤子');

}

**4),switch-case**

**Switch(表达式){**

**Case 1:**

**执行语句1..**

**Break;**

**Case 2:**

**执行语句2;**

**Break;**

**Default:**

**执行语句N+1//以上所有的条件都不满足**

**}**

**Break 终止后续语句的执行,在比较的过程中使用的是全等于===**

//订单状态 1-等待付款 2-等待发货 3-运输中 4-已签收 -5已取消

//练习:声明变量储存状态码,根据状态码打印不同的显示,如果以上都不满足,提示'错误状态码'

var order\_status = 2;

switch(order\_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;

case 4:

console.log('已签收');

break;

case 5:

console.log('已取消');

break;

default:

console.log('错误状态码');

}

**对比if-else嵌套和switch-case的区别**

**相同点:两者都可以用于多项分支语句,都可以判断是否相等的情况**

**不同点:switch-case 在比较相等的时候 使用全等于 ===**

**If-else 可以选择使用任意的运算符.**

**Switch-case 结构更为清晰合理,执行效率更高一些**

**3,循环-while**

**循环:一遍又一遍执行相同或者相似的代码**

**循环执行的两个要素:**

**1),循环条件:循环可以继续执行的条件**

**2),循环体:要重复执行的相同或者相似代码**

**While(循环条件){**

**循环体**

**}**

//练习:循环打印5~10

// 循环打印30~20

var i = 5;

while (i < 11) {

console.log(i);

i += 1;

}

var a = 30;

while (a >= 20) {

console.log(a);

a -= 1;

}

//练习 计算1~10之间的所有整数的和

var total = 0;

var i = 0;

while (i <= 10) {

total += i;

i++;

}

console.log(total);

**复习今天的内容,整理思维导图**

**2),课后练习**

**计算1-100之间所有偶数的和**

**计算1-100之间所有能被3整除的数字的和**

**计算 1- 10直接所有数字的乘积**

**3),预习do-while 和for 循环**

**While嵌套if else**

**猜大小游戏**

var num = 10;

while (true) {

var num1 = prompt('please input number');

num1 = parseInt(num1);

if (num1 > num) {

alert('big');

} else if (num1 < num) {

alert('small');

} else if (num == num1) {

alert('bingo');

break;

} else {

alert('wrong');

}

}

**do while循环**

**do{**

**循环体**

**}while(循环条件)**

//循环产生0~9

var i = 0;

do {

console.log(i)

i += 1;

} while (i <= 10);

//打印2 4 6 8 10

var i = 0;

var sum = 0;

do {

sum += i;

i += 2;

} while (i <= 10)

console.log(sum);

var i = 0;

var sum = 0;

do {

if (i % 2 == 0) {

sum += i;

}

i += 1;

} while (i <= 10);

console.log(sum);

//计算1到100之间的和

var i = 1;

var sum = 0;

do {

sum += i;

i += 1;

} while (i <= 100);

console.log(sum);

//1到10 的乘积

var i = 1;

var sum = 1;

do {

sum \*= i;

i += 1;

} while (i <= 10);

console.log(sum);

//打印90.85.80....50

var i = 90;

do {

console.log(i);

i -= 5;

} while (i >= 50);

//计算1-100自检所有奇数的和

//打印1-100之间所有能被7整除的数字

var i = 1;

var sum = 0;

do {

if (i % 2 == 1) {

sum += i;

}

i += 1;

} while (i <= 100);

console.log(sum);

var i = 1;

do {

if (i % 7==0) {

console.log(i);

}

i += 1;

} while (i <= 100);

**For 循环**

**1.执行初始值,**

**2.判断值是否满足条件,**

**3.执行循环体,**

**4.执行自增(减),**

**5.重新执行第二步**

//打印1-100之间所有能被3整除的数字

for (var a = 1; a <= 100; a++) {

if (a % 3 == 0) {

console.log(a);

}

}

//计算20到50之间能被2整除的所有整数的和

var sum = 0;

for (var i = 20; i <= 50; i++) {

if (i % 2 == 0) {

sum += i;

}

}

console.log(sum);

//打印2 4 6 8

for (var num = 1; num <= 8; num++) {

if (num % 2 == 0) {

console.log(num);

}

}

//打印80 77 74 71 ... 50

for (var num = 80; num >= 50; num -= 3){

console.log(num);

}

//计算1~20之间所有偶数的乘积

var sum = 1;

for (var num = 1; num <= 20; num++){

if (num % 2 == 0) {

sum \*= num;

}

}

console.log(sum);

//打印本世纪(2000-2100)所有的闰年

for (var year = 2000; year <= 2100; year++){

if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) {

console.log(year+'年是闰年');

}

}

var str = '';

for (var a = 1; a <= 5; a++){

str += a + '\*5=' + a \* 5 + ' ';

}

console.log(str)

**可以写成这种形式**

**For循环初始值可以写在外边,但是分号;要保留**

**循环条件也可以省略,但是会变成无限循环**

**For循环的自增(减)可以写在循环体中,但是分号;不能省略**

var i = 1;

for (; ;){

console.log(i);

if (i == 10) {

break;

}

i++;

}

**5.break和continue**

**Break:结束循环,不在执行其他循环**

**Continue:跳过本次循环,继续下一次循环**

//循环1到10,如果遇到5就跳过

for (var a = 1; a <= 10; a++){

//如果a = 5,跳过本次循环

if (a == 5) {

continue;//跳出循环体,执行自增

}

console.log(a);

}

//打印1-100之间的偶数,用continue

for (var num = 1; num <= 100; num++){

if (num % 2 != 0) {

continue;

}

console.log(num);

}

//打印1,2,5,7,10,11,13,14

for (var num = 1; num <= 100; num++){

if (num % 3 == 0 || num % 4 == 0) {

continue;

}

console.log(num);

}

//计算新中国成立后一共有多少个闰年

var num = 0;

for (var year = 1949; year <= 2018; year++) {

if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) {

num = num + 1;

}

}

console.log('一共有' + num + '个闰年');

//计算本世纪所有的闰年,打印前10个

var num = 0;

for (year = 2000; year <= 2100; year++) {

if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) {

num += 1;

console.log(year + '年为闰年')

if (num ==10)

break;

}

}

console.log('已统计' + num + '个闰年');

// 1-100之间的和,大于4000结束循环,打印和的值和当前加到哪个数字

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

sum += i;

if (sum > 4000) {

console.log(sum, i)

break;

}

}

//1/10+2/9+3/8+4/7+5/6

//循环调教中有多个表达式,则最后一个起作用

for (var a = 1, b = 10,result = 0; a <= 10,b>=1; a++,b--){

console.log(a + '/' + b + '=' + a / b);

result += (a / b);

}

console.log(result);

**循环嵌套**

//打印5行5列的\*

//外层循环:控制循环的行数

for (var a = 1; a <= 5; a++){

var str = '';

//内层循环:控制循环的列数

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

//把每次循环产生的\* 放入到空字符串中

str += '\*';

}

console.log(str);

}

//打印6行8列的%

for (var i = 1; i <= 6; i++){

var str = '';

for (var b = 1; b <= 8; b++) {

str += '%'+' ';

}

console.log(str);

}

//打印直角三角形形状的星

/\*

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*/

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

var str = ''

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

str += '\*';

}

console.log(str);

}

**函数相关(重点)**

**变量的作用域(重点)**

**函数的作用域(重点)**

**递归调用(了解)**

1. **函数:**

**parseInt()/typeof()/alert()/prompt()/parseFloat()**

**分为系统函数(js提供的)和自定义函数**

**自定义函数:**

**Function:功能体,可以接收若干个数据,返回处理的结果.用于封装需要反复执行的代码-----好比一个制作饺子的饺子机**

**1.1创建普通函数**

**Function 函数名称(){**

**函数体 🡪 要封装的反复执行的代码**

**}**

**1.2创建带有参数的函数**

**Function 函数清楚(参数列表){**

**函数体**

**}**

**调用:函数名称(参数列表)**

**参数列表:可以是0个或者是多个数据,之间用逗号隔开;创建函数时的参数称为’形参’,负责站位,在调用时的参数称为”实参”.**

**“实参”代表传递的数据,会赋值给形参.**

**注意:形参本质就是变量,但未赋值**

**1.3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**

**}**

**返回值就是函数体执行完,产生的结果**

**调用:函数名称(参数列表);**

**注意:**

**函数没有return 调用函数会返回 undefined;**

**函数中有retun,但是没有值,结果还是会返回undefined**

**Return可以返回任何类型的数据**

**对比break,continue,return用法**

**Break结束循环,结束switch-case中某一个情况**

**Continue跳出某一次循环,继续执行下一次循环**

**Return 终止函数的执行**

**2,变量的作用域**

**2.1作用域: 变量或者函数的访问(使用)范围.**

**局部(函数)作用域:函数内使用 var 声明的变量只能在函数内部生效**

**全局作用域:函数以外声明的变量,可以在任意位置生效**

**注意:函数内不使用VAR 关键字声明的变量是全局变量**

**2.2变量声明提升**

**Js程序在执行前,首先检查使用var 关键字声明的变量,提升到所在作用域的最前边,赋值还是在原来的位置.**

**2.3原始数据类型存储**

**计数据类型的数据存储在栈内存中,赋值一个原始类型的数据,会创建一个内存空间,把原来的值复制过去,和原来的值没有任何关系**

**3函数的作用域**

**和变量的作用域一样,也是分为局部作用域和全局作用域**

**全局作用域下创建的函数可以在任意作用域调用.**

**局部作用域下创建的函数只能在所在作用域下调用.**

**3.1函数声明提升**

在程序执行前,函数创建也会提升到最前边

如果两个函数的名称相同,后边创建的函数会覆盖前边的函数

**3.2函数在内存中的存储**

**函数存储在栈内存中**

**Function(){}**

**Fn() 表示调用函数,获取函数的返回值**

**Fn 存储的是函数在栈内存中的地址**

**4.递归**

**在函数内部调用自身**

**使用递归的时候需要有结束(renturn)**

**1创建函数 -- 函数表达式**

//创建一个匿名函数,把函数的地址存储到变量fun里

//创建函数 -- 函数表达式 此时的fun就是函数的名称

var fun = function () {

console.log('hello');

}

**调用函数: 函数名称(实参列表)**

**对比函数声明和函数表达式的区别**

**函数声明在函数提升,在任意的作用域下合法位置都可以.**

**函数表达式不存在函数提升,只能先声明再调用**

**2,匿名函数**

**Function(){}**

**匿名函数不能单独使用.**

**2.1 匿名函数用途—创建函数**

**2.2 匿名函数的自调用,防止污染全局**

**(function (){**

**//代码块**

**})()**

**3回调函数**

**将匿名函数以实参的形式传递给形参**

**Function fn(cb){**

**cb();**

**}**

**Fn(function(){函数体})//要调用此函数的名称(function(){函数体}**

**//function()就是上面的cb(第一个) 是实参**

**// cb是形参**

**全局函数:**

**parseInt()**

**parseFloat()**

**encodeURI() 对路径进行编码 –中文**

**decodeURI() 对路径进行解码 –中文**

**eval() 执行函数 可以做一些基本运算**

**isNaN() 检测一个值是否为NaN, 是 true 不是false**

**isFinite() 检测一个值是否为有限值 true false**

**4对象**

**对象:是一组具体的属性和方法(功能)的集合**

**哪些是对象?**

**具体某一个手机:属性:颜色,内存,屏幕的尺寸,品牌,摄像头….**

**功能:打电话,上网,发微信,看电影,听音乐**

**具体某一个人:属性:性别,年龄,身高,体重,肤色..**

**功能:吃饭,睡觉,打牌,工作,学习,写代码,唱歌**

**具体某一张床:属性:长宽,材质,单双,颜色,软硬…**

**功能:睡觉,休息,放东西**

**具体某一辆车:品牌,型号,排量,颜色,类型**

**功能:代步,载客,睡觉,运输**

**万物皆对象**

**1 Js中的对象**

**内置对象(js提供好的)**

**宿主对象(分为浏览器和服务器宿主对象)**

**自定义对象(自己创建对象)**

**2自定义对象**

**2.1对象字面量(直接量)**

**使用大括号创建空对象,**

**属性名和属性值之间用冒号隔开**

**多组属性之间用逗号隔开**

**属性名中引号可加可不加,如果特殊字符必须加引号.**

**3,给内置函数添加属性**

//使用内置构造函数创建对象

//创建空对象

var computer = new Object();

//添加属性

computer.color = 'grey';//语法1

computer.size = 15.4;

computer['price'] = 18688;//语法2

computer['made-in'] = 'CHINA';//有特殊符号时,必须使用中括号['']加引号''添加属性

console.log(computer);

上面两种语法既可以给内置构造函数创建的对象添加属性,也可以给对象字面量创建的对象添加属性

4获取属性

对象.属性名

对象[‘属性名’]

5.遍历对象(循环)

遍历可理解为循环,获取每一个属性

for in

**for(var key in emp){**

**//key 表示所有要遍历的属性名**

**//emp 表示要遍历的对象**

**emp[key] //表示要获取属性名想对应的值**

**}**

1. **检测对象中是否存在某个属性**

**1.1属性名’in 对象 console**.**log**('name' in person);

1.2 对象.hasOwnProperty(‘属性名‘);

**console**.**log**(person.**hasOwnProperty**('age'));

**1.3对象.属性名 === undefined**

**console**.**log**(person.name === undefined);*//false 属性存在 true 属性不存在*

**2,对象中的方法**

var person = {

name: 'tom',

age: '18',

**sing**: function () {

//this引用当前的对象

**console**.**log**('我叫'+this.name+' 我喜欢唱歌');

},*//这是一个方法*

**run**: function **run**() {

return this.name+' is running'

}//这也是一个方法

}

*//调用对象中的方法*

person.**sing**()

var res = person.**run**();

**console**.**log**(res);

**3,对象在内存中的存储**

**对象属于引用型类型数据,存储在堆内存中,会在栈内存中保存存储地址.**

**复制一个对象的变量相当于复制了一份地址.**

var person = {

name: 'tom',

age:22

}

*//把person对象的地址复制一份给person2*

*//person和person2指向的是同一个对象*

var person2 = person;

person.name = 'jack';

*//通过person改变name的值,person2对应的name的值也发生了改变*

**console**.**log**(person2.name);*//jack*

*//练:通过person2改变对象的age属性,查看person.age是否也发生了变化*

person2.age = 25;

**console**.**log**(person.age);*//25*

**数组**

**按照特定的数据结果来保存数据,是数据的集合**

**1,创建数组—数组字面量**

**[元素1,元素2,元素3,元素N]**

**在数组中可以存放任意类型的数据**

*//创建数组 - 内置构造函数*

var emp = new **Array**(5);

*//new一个数组emp,初始化长度为5*

emp[0] = '得分后卫';

emp[1] = '小前锋';

emp[2] = '组织后卫';

emp[3] = '大前锋';

emp[4] = '中锋';

emp[5] = '教练'

**console**.**log**(emp);

for (var x in emp) {

**console**.**log**(x + ' --> ' + emp[x]);

}

Array(6) ["得分后卫", "小前锋", "组织后卫", "大前锋", "中锋", "教练"]

0 --> 得分后卫

1 --> 小前锋

2 --> 组织后卫

3 --> 大前锋

4 --> 中锋

5 --> 教练

**获取数组长度**

**数组.length**

var city = new Array(5);

city[0] = '北京';

city[1] = '上海';

city[2] = '广州';

city[3] = '深圳';

city[4] = '杭州';

console.log(city);

city[5] = '重庆';

*//自动获取下一个数组的下标:就是当前数组的长度*

city[city.length] = '成都'

console.log(city);

city[2] = '南京';

console.log(city);

**数组分类**

**索引数组**

**以0和正整数作为下标的数字**

**关联数组**

**以字符串作为下标的数据,只能单独添加每一个元素.**

ar car = new Array();

car['cid'] = '1001';

car['brand'] = '北京汽车';

console.log(car);

**数组遍历**

**For-in 既可以遍历索引数组,也可以遍历关联数组**

var piano = ['推荐', '热点', '娱乐', '科技', '汽车', '体育', '军事', '国际', '时尚', '游戏']

for (var i in piano) {

console.log(i + ' --> '+piano[i])

}

for (var k = 0; k < piano.length; k++){

console.log(k + ' --> '+piano[k])

}

**数组中的方法**

**数组属于JS中的内置函数,既然是对象,肯定有属性和方法**

**API🡪应用程序接口**

**就是JS提供好的方法**

**Join() 将数组转为字符串,可以按照指定的字符来拼接起来**

**Concat() 将多个数组组合到一起**

var a = ['111', '222'];

var b = ['333', '444'];

var c = ['555', '666'];

console.log(a.concat(b, c));

**slice() 截取数组中的元素**

*var arr = ['科技', '体育', '汽车', '美容','模特','餐饮','旅游'];*

*//截取特定*

*console.log(arr.slice(2, 4));//截取2-3 两个*

*console.log(arr.slice(-3));//截取倒数三个*

*console.log(arr.slice(2));//没有结束下标,表示截取到最后*

**splice() 删除(增加)数组的元素**

//arr.splice(2);//删除元素,从2开始到结束

console.log('start -->'+arr);

//console.log( arr.splice(2, 4));//从下边2开始,往后删除4个元素

//console.log(arr.splice(1, 2, 'add1'));//第三个'add1'为添加元素

//删除'2',在删除的位置添加新闻

//arr.splice(2, 1, '新闻');

reverse()

翻转数组元素

var arr = [1, 2, 3, 4, 5, 6];

console.log(arr.reverse());

sort()

对数组元素排序

var arr = [23, 6, 78, 9, 45];

//console.log(arr.sort());//默认是按照unicode码从小到大排序

console.log(arr.sort(function (a, b) {

return a - b;

//return b – a;

}));//运用回调函数,按照从小到大排序,第二种返回方式从大到小

push()

在数组的最后添加元素

var arr = ['推荐', '热点', '新闻'];

console.log(arr.push('汽车'));//返回数组的长度

console.log(arr);

pop()

删除最后一个元素

console.log(arr.pop());//返回删除的元素值

二维数组

在数组中,每一个元素也可以是数组

获取数组中具体某一个元素

//大数组表示所有的城市

var city = [

['海淀区','朝阳区','西城区','宣武区'], //每一个小数组,代表一个省市下的城市

['深圳','广州','花都','潮州'],

['成都','绵阳','甘孜阿坝','泸州'],

['渝中区','巴南区','江津区','永川区']

];

console.log(city[2][1]);

console.log(city[1][2]);

数组也是引用类型

var arr = ['a', 'b', 'c', 'd'];

var arr1 = arr;

arr.push('tom');

arr1[0] = 'tom';

console.log(arr1);

console.log(arr);

字符串操作

包装对象:

目的就是原始数据像引用类型,具有属性和方法

本质上还是原始数据类型

JS提供三种包装对象 String Number Boolean

使用方式 new String(‘hello’);

1转义字符

把普通的字符,转成具有特定意义的字符

console.log('hello \n world');// \n 换行

console.log('It\'s a dog')// \'输出字符串中加个'

console.log('1\t2'); // \t为tab键功能 制表符

**2字符串对象常用方法**

var str = 'I Love My Family';

//如果要调用字符串对象下的方法,无需包装成对象

//1,将英文字符转为大写

console.log(str.toUpperCase());

//2,将英文字符转为小写

console.log(str.toLowerCase());

var test = 'hello world';

console.log(test.length);// 获取字符串的长度

console.log(test.charAt(2));//获取字符串 下标对应的字符

//遍历字符串

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

//将每一个字符转成unicode码,并打印

console.log(test.charAt(i).charCodeAt());

}

for (var i in test) {

console.log(i + ' --> ' + test[i]);

}

indexOf(value,start) 查找某个字符(串)对应的下标,value为查找值,start是从第几个下标开始找,不写默认为0.

var str = 'JavaScript';

console.log(str.indexOf('Script'));

//第一个参数查找的字符(串),从第2个开始的下标

console.log(str.indexOf('Script',3))

**4 匹配模式**

**作用:用于查找,替换字符串中的某些字符**

**Replace(value1,value2)用于查找字符串,并使用另一个字符替换.value1要查找的字符串,value2要替换的字符串**

**Value1 可以用匹配模式的写法 /china/gi**

**G 🡪 global 全局匹配**

**I 🡪 ignore 忽略查找大小写**

var str = 'china是世界上人口最多的国家,China的面积是960万平方公里,CHINA的互联网高度发达,china有好多的程序员';

//替换china为中国,这样只能替换第一个,不能往后继续查找

console.log(str.replace('china', '中国'));

//使用匹配模式来查找china --正则表达式

//g --> global

//i --> ignore

console.log(str.replace(/china/gi,'中华'));

**match(value)用于查找某个字符串出现的次数,返回数组.可以使用i和g**

console.log(str.match(/china/gi));

search(value)用于查找字符串,返回满足条件的第一个条件的下标,找不到返回-1,和indexOf()作用类似

console.log(str.search(/china/i));

课后任务:

复习今天内容,整理思维导图

课后练习:

将一句英文字符串中的单词首字母转为大写,其余字符转为小写.

‘we are falimy’ 🡪 ‘We Are Family’

预习match对象,data对象.

Math对象

*//math对象*

*//获取圆周率*

console.log(Math.PI);*//3.1415926..*

*//绝对值*

console.log(Math.abs(-5));*//5*

*//parseInt*

console.log(parseInt(3.5)); *//3*

*//四舍五入取整*

console.log(Math.round(3.5));

*//向上取整*

console.log(Math.ceil(3.35));*// 4*

*//向下取整*

console.log(Math.floor(3.6))*// 3*

*//求最大值*

console.log(Math.max(18, 29, 14, 99, 100, 20));*// 100*

*//求最小值*

console.log(Math.min(15, 2, 45, 29, 3)); *// 2*

*//取X的Y次幂*

console.log(Math.pow(3, 5));*// 243*

*//取随机数*

console.log(Math.random());

*//获取0-55之间随机整数*

var num = Math.random() \* 56;

*//向下取整(o.几到55.几)*

console.log(Math.floor(num));

**DATE对象**

**用于对日期和时间进行存储和计算**

1. **创建对象**

**New Date(‘2018/10/1 18:20:30’)**

**New Date(2018,9,1,18,20,30) //第二个参数月份范围 0-11**

**New date(1000) // 距离计算机元年 1000毫秒的时间**

**New Date() //系统当前时间**

**2),获取具体的时间信息**

console.log(d1.getFullYear());

console.log(d1.getMonth());*//获取的月份 0-11*

console.log(d1.getDate());

console.log(d1.getHours());

console.log(d1.getMinutes());

console.log(d1.getSeconds());

console.log(d1.getMilliseconds());

*//星期 0-6 范围星期日--星期六*

console.log(d1.getDay());

console.log(d1.getTime());

**3本地化时间格式**

**toLocalString() 年-月-日 时:分:秒**

**toLocalDateString() 年-月-日**

**toLocalTimaString() 时:分:秒**

var d1 = new Date('2018/10/20 15:30:40');

*//本地化时间格式*

console.log(d1.toLocaleString());

*//年月日*

console.log(d1.toLocaleDateString());

*//时分秒*

console.log(d1.toLocaleTimeString());

d1.setFullYear(2008);*//设置年份为2018年*

d1.setMonth(11);*//实际显示月份+1*

*//设置日期为30 时分秒 8:30:50*

d1.setDate(30);

d1.setHours(8);

d1.setMinutes(30);

d1.setSeconds(50);

*//8月1号入职,合同3年,到期后是多久*

var workTime = new Date('2018/10/20');

var targetTime = new Date(workTime);

targetTime.setFullYear(targetTime.getFullYear() + 3);

console.log('入职时间为:' + workTime.toLocaleDateString()

+ '合同到期时间为:' + targetTime.toLocaleDateString());

*//续签合同,提前一个月续签,如果是周末,提前到周五*

var goon = new Date(targetTime);

goon.setMonth(goon.getMonth() - 1);

if (goon.getDay() == 0 ) {

goon.setDate (goon.getDate()-2)

} else if(goon.getDay() == 6 ){

goon.setDate (goon.getDate()-1)

}

console.log(goon.toLocaleDateString());

**Number对象**

**包装对象.本质上就是将数据转为数值型,返回结果是对象**

**New Number()**

**Number()**

**Number(true) -> 1 Number(false) -> 0**

**parseInt(true) ->NaN**

**Number.MAX\_VALUE 获取计算机所能存储的最大数字**

**Number.MIN\_VALUE 获取计算机所能存储的最小数字**

**超出最大最小,返回infinity**

**toFixed(n) 保留小数点后n位**

**toString() 将数值转为字符串类型**

**Boolean 对象**

**包装对象,返回对象**

**本质上就是将数据类型转换为布尔型**

**New Boolean()**

**Boolean()**

**!!数据**

**toString() 将布尔型转为字符串**

**错误处理**

**SyntaxError 语法错误,JS在执行程序前,先检查程序中是否有语法错误,如果有语法错误,则打印错误信息,不会执行任何JS代码.一般错误的使用了中文,缺少括号之类的.**

**ReferenceError 错误的使用了未声明的变量,会阻止后续代码的执行.**

**TypeError 类型错误,错误的使用了括号等..会阻止后续代码的执行**

**RangeError 范围错误,传递的参数超出了范围等…会阻止后续代码的执行**

**Try-catch 错误处理**

**Try{//尝试执行可能产生错误的代码**

**}**

**Catch(err){**

**//如果有错误,将错误信息捕获到err中**

**//错误处理方案**

**}**

**使用try-catch,即使有错误,也不会影响后续代码的执行**

/\*

//尝试执行,说明可能产生错误

try {

var b = a + 1;

//尝试执行的代码

var arr = new Array(-2);

} catch (err) { //尝试出现错误,捕获到err中

//解决方案

//console.log(err);

var b = 1 + 1;

}

console.log(b);

console.log('over');

\*/

/\*

计算所有成绩(语文,数学,英语)的和,如果成绩为空,默认设置为60,最好打印总和

\*/

try {

var chinese = 80,

math = 85,

english = 90,

sum;

//尝试计算总和,可能会出现错误

sum = chinese + math +english;

}

catch(err){

//console.log(err);

//解决防范:如果英语不存在,使用60

var sum = chinese + math + 60;

}

console.log(sum);

**ES6新特征**

**ECMAScript6 ES 2018 2017 2016**

**1,块级作用域**

**在块级作用域中,使用let声明的变量是局部变量,不能被外部方位**

**Let声明的变量不存在变量提升**

**块级作用域:{} for(){} while do while..**

**2,箭头函数**

**回调函数的另一种写法**

**如果箭头函数的函数体中只有一行代码,并且是含有return关键字,则可以省略大括号和return**

var arr = [18, 37, 59, 64, 81];

console.log('回调函数: '+arr.sort(function (a, b) {

return a - b;

}));

console.log('箭头函数: '+arr.sort((a, b) => {

return a - b;

}))

console.log('箭头函数简化 '+arr.sort((a, b) => a - b));

**3,函数中的参数**

//函数中的参数 ec6允许为形参设置默认值,如果没有传递实参,自动使用形参的默认值

function fn(a =5, b= 6, c = 4) {

console.log( a + b + c);

}

fn(1);// 11

4,模板字符串