# 练习：switch实现成绩大于60合格，否则不合格

**方法一：**

var num = 50;

switch(parseInt(num/10)){ #parseint 取整

case 6:

case 7:

case 8:

case 9:

case 10:

console.log('成绩合格');

break;

default:

console.log('成绩不合格');

break;

}

**方法二：**

var num = 50;

switch(true){

case num >= 60:

console.log('成绩合格');

break;

default:

console.log('成绩不合格');

break;

}

# while循环：

var n=1;

while(n<6){

document.write(n++ + "<br/>");

// alert(n++);

}

# Do…while循环

var m=1;

do{

document.write(m++ + "<br/>");

}while(m<11)

# 练习： 银行利率5%，1000块钱本金涨到5000需要多少年

var year = 0;

var mony = 1000;

while(mony<=5000){

mony \*=1.05;

year++;

}

console.log(year);

# 练习：判断成绩是否合格，若输入值不合法重新输入

while(true){

var score = prompt("please input a number(0-100)"); #弹出输入框

if(score >=0 && score <=100){

if(score<60 || score >= 0){

alert("不及格");

break;

}else if(score >= 60 || scor115015e <=100){

alert("及格");

break;

}

}

alert("输入错误，重新输入");

}

# 练习：求1-100之间质数的和

var sum=0;

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

if(n%2 != 0){

console.log(n)

sum += n;

}

};

console.log("奇数之和为：" + sum);

# 练习：求1-100之间7的倍数的个数和综合

var sum=0;

var n=0;

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

if(i%7 == 0){

n++;

sum +=i;

}

}

console.log("个数：" + n);

console.log("总和：" + sum);

# 练习：求水仙花数（三位数，个位十位百位三次幂之和等于本身）

for(var i=100;i<=999;i++){

var x = parseInt(i/100); //百位数

var y = parseInt(i%100/10); //十位数

var z = parseInt(i%10); //个位数

var num = x\*x\*x+y\*y\*y+z\*z\*z;

if(num == i){

console.log(i);

};

};

# 练习：判断用户输入的数是否为质数

while(true){

var score = prompt("请输入大于1正整数");

if(isNaN(score) === false){

if(score <=1){

alert("非法输入，请重新输入！");

}else{

for(var n=score-1;n>=1;n--){

if(score%n === 0 && score != 2){

alert("不是质数");

break;

}else if((n === 2 && score%n != 0) || score == 2){

alert("质数");

break;

}

};

break;

};

}else{

alert("非法输入，请输入数字");

}

}

# 练习：for循环画图形

var num = 1;

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

for(var y=0;y<num;y++){

document.write("\*")

}

num++;

document.write("<br/>");

}

# 练习：打印九九乘法表

<script type="text/javascript">

var x = 1;

for(x;x<=9;x++){

for(var y=1;y<=x;y++){

document.write("<span>" + x + "\*" + y + "=" + x\*y + "</span>");

}

document.write("<br/>")

}

</script>

<style type="text/css">

span{

display: inline-block;

width:5rem;

text-align: center;

}

</style>

# 联系：打印1-100之间的质数

方法一：

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

if(y == 2){

document.write(y + "<br/>");

}

for(var x=2;x<y;x++){

if((y%x != 0 && x == y-1) || y == 2){

document.write(y + "<br/>");

}else if(y%x == 0 && x<y){

break;

}

}

}

方法二：

for(var i=2;i<=100;i++){

var flag=true;

for(var j=2;j<i;j++){

if(i%j == 0){

flag=false;

break;

}

}

if(flag){

console.log(i);

}

}

# Continue和break的使用

1、给循环取名为outer，并在停止outer

outer:

for(var i=0;i<2;i++){

console.log(i);

for(var j=10;j<12;j++){

console.log("a");

break outer;

}

}

# 程序测试-计时器

console.time("test");

for(var i=2;i<=100;i++){

var flag=true;

for(var j=2;j<i;j++){

if(i%j == 0){

flag=false;

break;

}

}

if(flag){

console.log(i);

}

}

console.timeEnd("test");

# 对象

1. 对象的分类
   1. 内建对象

String、number、math等EES标准中定义的

* 1. 宿主对象

Bom、Dom、console、document等浏览器提供的

* 1. 自定义对象

开发人员自己创建的对象

2、增删改查

var obj = new Object();

// 增

obj.name = "许国栋";

obj.sex = "男";

obj.age = 22;

// 查

console.log(obj);

// 改

obj.name = "钩子";

// 删

delete obj.sex;

console.log(obj);

1. 属性名为数字

var obj = new Object();

obj["123"] = 789;

console.log(obj["123"]);

1. 灵活使用属性名

var obj = new Object();

obj["123"] = 789;

obj["nihao"] = "你好";

var n = "nihao";

console.log(obj[n]);

1. 检查obj中是否有123属性，有则true，无则false

obj["123"] = 789;

console.log("123" in obj);

5、引用和赋值（基础数据类型保存在栈内存中，引用数据类型保存在堆内存中）

var name = "zrr";

var a = name;

name = "zll";

console.log(name); //zll

console.log(a); //zrr

var obj = new Object();

obj.age = 10;

var num = obj.age;

obj.name = 20;

console.log(obj.age); //20

console.log(num); //20

1. 创建对象

var obj = {name:"xuguodong",age:12};

console.log(obj);

# 函数

1. 定义函数

var fun = new Function("console.log('fun');");

fun();

function fun2(){

console.log("fun2");

}

fun2();

var fun3 = function(){

console.log("fun3");

};

fun3();

1. 函数返回值（return后的语句不会执行）

function add(a,b){

var sum = a+b;

return sum;

}

var result = add(1,99);

console.log(result);

1. 判断数字是否为偶数

function isOu(num){

return num % 2 == 0;

}

var result = isOu(4);

console.log(result);

1. 实参可以是任何值，比如函数

function hello(o){

console.log("我叫" + o.name + ",今年" + o.age + "了,我是" + o.gender + "人,我住在" + o.address);

}

var obj = {

name:"孙悟空",

age:"18",

address:"花果山",

gender:"男"

};

function fun(a){

a(obj);

}

fun(hello);

hello()

调用函数

相当于使用函数的返回值

Hello

函数对象

相当于使用函数对象

1. 立即执行函数

(function(){

console.log("hello everyone");

})();

(function(a){

console.log("a=" + a);

})(123)

1. 枚举对象中的键值对

var obj = {

name:"孙悟空",

age:10,

gender: "男"

}

for(var n in obj){

console.log(n);

console.log(obj[n]);

}

1. 全局作用域
   1. 全局作用域中创建的变量会作为window对象的属性保存
   2. 全局作用域中创建的函数会作为window对象的方法保存

var a=10;

console.log(window.a);

function fun(){

console.log("我是函数");

}

window.fun();

window.alert("hello");

8、函数作用域定义变量时如果未使用var则为全局作用域

var a=20;

console.log("a = "+a);

}

fun();

function fun1(){

console.log("a = "+a);

var a=30;

}

fun1();

function fun2(){

a=40;

console.log("a = "+a);

}

fun2();

function fun3(a){

a=123;

console.log("a = "+a);

}

fun3(a);

# This

当以函数形式调用时，this指向window

当以方法形式调用时，this指向调用方法的对象

var fun = function(){

console.log(this.name)

}

var obj = {

name:"憨皮",

sayHello: fun

};

var name = "憨憨";

obj.sayHello(); //憨皮

fun(); //憨憨

# 构造函数

function createPerson(name,age,gender){

var obj = new Object();

obj.name = name;

obj.age = age;

obj.gender = gender

obj.sayName = function(){

console.log("我叫" + name);

}

return obj;

}

var obj1 = createPerson("小明","22","男");

obj1.sayName(); //我是小明

var obj2 = createPerson("小白");

obj2.sayName(); //我是小白

# 创建构造函数

构造函数和普通函数的区别是调用方式不同

普通函数

Var obj = fun（）；

构造函数

Var obj = new fun（）；

构造函数执行流程

1. 创建一个新的对象
2. 将新建对象设置为函数中的this
3. 逐行执行函数中的代码
4. 将新建的对象作为返回值返回

任何对象和object做instanceof都为true

function Person(name,age){

this.name = name;

this.age = age;

this.sayName = fun;

}

function fun(){

console.log(this.name)

}

function Dog(name){

this.name = name,

this.sayName = fun;

}

var obj = new Person("小徐",20);

console.log(obj);

obj.sayName();

var obj1 = new Dog("timi");

console.log(obj1);

obj1.sayName();

console.log(obj1 instanceof Dog); //ture,检查obj1是否为Dog的实例

console.log(obj1 instanceof Object); //true

var obj2 = new Dog("deny");

console.log(obj1.sayName == obj2.sayName); //true

# 原型对象

创建的每一个函数都会有一个prototype属性，它指向原型对象

普通函数调用prototype没有任何作用

构造函数调用prototype

function myClass(){

}

var fun = new myClass();

console.log(fun.\_\_proto\_\_);

console.log(fun.\_\_proto\_\_ == myClass.prototype);

var fun2 = new myClass();

console.log(fun2.\_\_proto\_\_ == fun.\_\_proto\_\_);

//在原型中添加属性

myClass.prototype.a = 123;

console.log(fun.a);

myClass.prototype.sayHello = function(){

alert("hello,这是原型");

};

fun.sayHello();

检查对象或者原型中是否有属性

function fun(){

}

var mc = new fun();

fun.prototype.name = "我是原型中的name";

console.log("name" in mc);

检查对象自身是否有属性

function fun(){

}

var mc = new fun();

fun.prototype.name = "我是原型中的name";

mc.age = 18;

console.log(mc.hasOwnProperty("name")); //false

console.log(mc.hasOwnProperty("age")); //true

hasOwnProperty是在原型的原型对象中定义的属性

console.log(mc.\_\_proto\_\_.\_\_proto\_\_.hasOwnProperty("hasOwnProperty"));

object对象的原型没有原型

console.log(mc.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_); //null

# toString

function Person( name, age ){

this.name = name,

this.age = age

}

//修改toString方法

Person.prototype.toString = function(){

return this.name

}

var per = new Person("悟空",18);

console.log(per)

var perString = per.toString();

console.log(perString); //[object Object]

console.log(per.\_\_proto\_\_.\_\_proto\_\_.hasOwnProperty("toString")); //true

var per1 = new Person("八戒");

console.log(per1.toString()); //八戒

# 垃圾回收

JS自动回收机制会自动回收，前提是赋值为NULL

var obj = new Object();

obj = null;

# 数组

数组也是对象

var arr = [1,2];

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

arr[arr.length] = i+3;

}

console.log(arr)

var arr1 = [1,2];

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

arr1.push(i+3);

}

console.log(arr1);

var arr = [function(){alert(1);},function(){alert(2);}];

arr[1]();

数组的方法（增删改查）

var arr = ["张","李","王"];

// 增加,末尾加

var result = arr.push("钟","吕");

console.log(arr);

console.log(result);

//前面加

var result2 = arr.unshift("郑");

console.log(arr);

console.log(result2);

// 删除,末尾删

var result1 = arr.pop();

console.log(arr);

console.log(result1);

//前面删

var result3 = arr.shift()

console.log(arr);

console.log(result3);

遍历数组，提取出符合条件的条目

function per(name,age){

this.name = name;

this.age = age;

}

var arr1 = new per("孙悟空",20);

var arr2 = new per("猪八戒",30);

var arr3 = new per("沙和尚",25);

var allArr = [arr1,arr2,arr3];

function getAdult(arr){

var newArr = [];

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

if(arr[i].age < 28){

newArr.push(arr[i]);

}

}

return newArr;

}

var result = getAdult(allArr);

console.log(result);

forEach

var arr = ["a","b","c"];

arr.forEach(function(num){

console.log(num)

});

Slice和Splice

var arr = ["a","b","c","d","e"];

var result = arr.slice(0,2); //中间取一段

console.log(result);

var result2 = arr.slice(2); //从2往后面的全要

console.log(result2);

var result3 = arr.slice(-2); //保存最后两个

console.log(result3);

var result1 = arr.splice(0,2,"A"); //第0个开始删除2个,开始位置添加A

console.log(arr);

数组中重复的值去除

var arr = [1,2,1,3,2];

var newArr = [];

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

if(newArr.length >= 1){

for(var j=0;j<newArr.length;j++){

if(arr[i] == newArr[j]){

break;

}else if(j == newArr.length-1){

newArr.push(arr[i]);

}

}

}else{

newArr.push(arr[i]);

}

}

console.log(newArr);

数组的其他方法

//链接数组

var arr1 = ["孙悟空"];

var arr2 = ["猪八戒"];

var arr3 = ["沙和尚"]

var arr = arr1.concat(arr2,arr3,"唐僧");

console.log(arr); //["孙悟空","猪八戒","沙和尚","唐僧"]

//数组转换为字符串

var result = arr.join("-");

console.log(result);

console.log(typeof result); //string

//颠倒数组,影响原数组

arr.reverse();

console.log(arr);

//排序

var arr4 = ["a","c","b"];

arr4.sort();

console.log(arr4);

//对数字排序时添加回调函数,返回值大于等于零顺序不变,小于零交换位置

var arr5 = [5,4,2,6];

arr5.sort(function(a,b){

return a - b;

})

console.log(arr5);

# Arguments对象

方法创建时，不但会创建this，还会创建arguments，arguments类似数组但不是数组。

调用函数时，我们传递的实参都会在Arguments中保存。要使用的时候可以在Argument中取出来。

function func(){

console.log(arguments);

console.log(arguments instanceof Array); //false

console.log(Array.isArray(arguments)); //false

console.log(arguments.length); //2

console.log(arguments.callee); //打印函数中的所有代码

console.log(arguments.callee == func); //true

}

func("hello","world");

# Date对象

计算机以时间戳形式保存时间

//封装当前代码执行的时间

var d = new Date();

console.log(d);

//创建一个指定的时间,2012年12月6日 11点10分30秒

var d2 = new Date("12/06/2012 11:10:30");

console.log(d2);

console.log(d.getDate()); //获取当前日期是几号

console.log(d.getMonth() + 1); //获取当前月份

console.log(d.getFullYear()); //获取当前年份

console.log(d.getDay()); //获取当前是礼拜几

console.log(d.getTime()); //获取时间戳，从格林威治1970年1月1日0时0分0秒到当前日期所花费的毫秒数,1秒=1000毫秒

console.log(Date.now()); //获取当前时间戳

# Math

// 圆周率

console.log(Math.PI);

// 绝对值

console.log(Math.abs(-10));

//上舍入

console.log(Math.ceil(1.4));

// 下舍入

console.log(Math.floor(1.1));

// 四舍五入

console.log(Math.round(1.3));

//生成0-1之间的随机数

console.log(Math.random());

// 生成x-y之间的数

var x=20;

var y =30;

console.log(Math.round(Math.random()\*(y-x))+x);

// 获取多个数的最大值

console.log(Math.max(10,20,30,15,12));

// 获取多个数的最小值

console.log(Math.min(10,20,30,15,12));

// 返回x的y次幂

console.log(Math.pow(2,7));

// 开方

console.log(Math.sqrt(4));

# 包装类

// String 基本数据类型转换为String对象

// Boolean 基本数据类型转换为Boolean对象

// Number 基本数据类型转换为Number对象

var num = new Number(3);

var str = new String('hello');

var bool = new Boolean(true);

console.log(typeof num);

console.log(typeof str);

console.log(typeof bool);

num.hello = 'abc';

console.log(num.hello); //abc

var a = 3;

a.hello = "world";

console.log(a.hello); //undefined

# 字符串的方法

var str = "hello";

//字符串是以字符数组的形式保存["h","e","l","l","o"]

console.log(str[0])

console.log(str.length); //获取字符串长度

var result = str.charAt(0); //返回指定位置字符

console.log(result);

var result1 = str.charCodeAt(0); //获取unicode编码

console.log(result1);

var result2 = String.fromCharCode(0x2692); //根据unicode字符编码获取文字,十六进制

console.log(result2);

var str1 = str.concat("你好","世界"); //拼接字符串

console.log(str1);

var result3 = str.indexOf("h"); //查找字符串返回第一次出现的索引,-1表示没找到

console.log(result3);

var result4 = str.indexOf("l",3); //从第四个数开始找 l

console.log(result4);

var result5 = str.lastIndexOf("l"); //从后面开始找

console.log(result5);

var result6 = str.slice(0,2); //截取字符串,从0个开始截取2个字符

console.log(result6);

var result7 = str.slice(2); //除去前面两个字符后面的都截取下来

console.log(result7);

var result8 = str.substring(1,2); //截取字符串,不能传递负值,自动调整参数位置

console.log(result8);

var str2 = "xuguodong";

var result9 = str2.split("u"); //拆分字符串

console.log(result9);

var result99 = str2.split(""); //每个字符都拆分开

console.log(result99);

var result10 = str2.toUpperCase(); //转换成大写

console.log(result10);

var str3 = "XUguodong";

var result11 = str3.toLowerCase(); //转换成小写

console.log(result11);

# 正则表达式

var reg = new RegExp("a");

var str = "a";

var result = reg.test(str);

console.log(result); //true

var reg1 = new RegExp("a");

console.log(reg1.test("A")); //false,查找大写的A

var reg2 = new RegExp("a","i");

console.log(reg2.test("A")); //true,忽略大小写

var reg3 = new RegExp("xuguodong");

console.log(reg3.test("xuguodongxu")); //true

// 使用字面量创建表达式

var reg = new RegExp("a","i");

var exp = /a/i; //和上面的表达式效果一样

console.log(exp.test("Abcd")); //true

console.log(exp.test("name")); //true

//查找字符串中有a或b

var exp1 = /a|b/; //等效于/[ab]/

console.log(exp1.test("cde")); //false

console.log(exp1.test("ab")); //true

console.log(exp1.test("name")); //true

//查找字符串中是否有字母

var exp2 = /[a-z]/; //[a-z]任意小写字母，[A-Z]任意大写字母，[A-z]任意字母

console.log(exp2.test("123456")); //false

console.log(exp2.test("hantian123")); //true

console.log(exp2.test("ABC")); //false

//查找以a开头，以c结尾，中间d，e的字符串

var exp3 = /a[d,e]c/;

console.log(exp3.test("adc")); //true

console.log(exp3.test("aec")); //true

console.log(exp3.test("afc")); //false

//查找除了abc以外的字符

var exp4 = /[^adc]/;

console.log(exp4.test("addc")); //false

console.log(exp4.test("affc")); //true

console.log(exp4.test("adc")); //false

# Split search replace match

//把字符串根据字母进行拆分

var str = "1a2b3c4d5";

var result = str.split(/[A-z]/);

console.log(result);

//查找字符串search

var str1 = "hello abc def adc";

var result1 = str1.search("def");

var result11 = str1.search("name");

console.log(result1); //10

console.log(result11); //-1

console.log(str1.search(/a[bd]c/));

//根据正则表达式把符合条件的内容提取出来match

var str2 = "1a2b3c4da5HY";

var result2 = str2.match("a");

console.log(result2);

//match只找第一个符合要求的内容，全局找设置全局匹配模式

console.log(str2.match(/[A-z]/g));

var str3 = "abcdkidlm"

var result3 = str3.replace("d","@");

console.log(result3); //abc@kidlm

console.log(str3.replace(/d/g,"@")); //abc@ki@lm

console.log(str3.replace(/d/g,"")); //删除d

# 正则表达式语法

// 通过量词配置字符出现的次数

var str = /a{3}/

var result = str.test("aaa");

console.log(result); //true

var str1 = /ab{3}/;

var result1 = str1.test("abbb");

console.log(result1); //true

var str2 = /a{1,3}bc/; //a出现1-3次

var result2 = str2.test("aabc");

console.log(result2);

var str3 = /ab+c/; //至少出现一个b

var str4 = /ab{1,}c/; //b至少出现一次以上

var str5 = /ab\*c/; //b有没有都行

var str6 = /ab?c/; //b只能有一个或者没有

// 检测字符串是否以a开头

var str7 = /^a/;

var result7 = str7.test("abc"); //true

console.log(result7);

//检测字符串是否以a结尾

var str8 = /a$/;

var result8 = str8.test("cba");

console.log(result8); //true

//正则表达式同时使用^和$，开头和结尾都是a，只能是一个a

var str9 = /^a$/;

var result9 = str9.test("");

console.log(result9);

var Reg = /./; // .表示任意字符

var Reg1 = /\./; //转义字符 \. 表示 .

console.log(Reg1.test("dfa"));

//检查字符串里是否有斜杠

var Reg2 = /\\/;

console.log(Reg2.test('ab\\'));

//构造函数的字符串中要显示 \. 要使用转义字符 \\.

/\*

\*\w 任意字母、数字、\_ [0-9] [A-z] \_

\*\W 除了字母、数字、\_

\*\d 任意数字[0-9]

\*\D 除了数字

\*\s 空格

\*\S 除了空格

\*\b 单词边界

\*\B 除了单词边界

\*/

var Reg3 = /\bchild\b/;

console.log(Reg3.test("hello child ")); //true

console.log(Reg3.test("hellochild"));

# 练习：判断字符串是否为电话号码

var phoneStr = "15970798564";

var phoneReg = /^1[3-9][0-9]{9}$/

console.log(phoneReg.test(phoneStr));

# 练习：把字符串中开头和结尾的空格去掉

// 去除开头和结尾空格，中间保留

var res = " he l lo ";

// 去除开头空格

// res = res.replace(/^\s\*/,"");

// 去除结尾空格

// res = res.replace(/\s\*$/,"");

res = res.replace(/^\s\*|\s\*$/g,"");

console.log(res);

# 练习：邮件的正则

// 电子邮件

// hell .nihao @ abc .com .cn

// 任意字母下划线 .任意字母下划线 @ 任意字母数字 .任意字母(2-5) .任意字母(2-5)

// \w{3,} (\.\w+)\* @ [A-z0-9]+ (\.[A-z]{2,5}){1,2}

var emailReg = /^\w{3,}(\.\w+)\*@[A-z0-9]+(\.[A-z]{2,5}){1,2}$/;

var email = "!abc.hello@163.com";

console.log(emailReg.test(email));

# DOM事件

<button id="btn">单击</button>

<script>

var btn = document.getElementById("btn");

btn.onclick = function (){

alert("干嘛");

}

</script>