**JS基础**

JavaScript是一种运行在客户端 的脚本语言

java 编译语言 java虚拟机 java虚拟机不认识程序猿写的java代码 xxx.java -(编译)-> xx.class -->执行.class文件

js 脚本语言 浏览器 浏览器认识js代码,不需要编译的过程 xx.js --> 直接在浏览器中解释执行,解释一行,执行一行

### **JavaScript和HTML、CSS的区别**

1.HTML：提供网页的结构，提供网页中的内容

2.CSS: 用来美化网页

3.JavaScript: 可以用来控制网页内容，给网页增加动态的效果

## **2. JavaScript的组成 （ECMAScript DOM BOM）**

### **BOM - 浏览器对象模型**

一套操作浏览器功能的API

通过BOM可以操作浏览器窗口，比如：弹出框、控制浏览器跳转、获取分辨率等

### **DOM - 文档对象模型**

一套操作页面元素的API

DOM可以把HTML看做是文档树，通过DOM提供的API可以对树上的节点进行操作

## **3.JavaScript的书写位置**

1.写在script标签中

2.写在外部js文件中，在页面引入

注意点 ===> 引用外部js文件的script标签中不可以写JavaScript代码

## **4.变量(重点)**

1.什么是变量 ==== > 变量是计算机内存中存储数据的容器

2.为什么要使用变量 ==== > 使用变量可以方便的获取或者修改内存中的数据(便于计算)

3.如何使用变量

1.var声明变量 ==== > var age;

2.变量的赋值 ==== > var age;age = 18;

3.同时声明多个变量 === > var age, name, sex; age = 10; name = 'zs';

4.同时声明多个变量并赋值 === > var age = 10, name = 'zs';

### **4.变量的命名规则和规范**

### **1.**规则 - 必须遵守的，不遵守会报错

### 2.由字母、数字、下划线、$符号组成，不能以数字开头

### 3.不能是关键字和保留字，例如：for、while。区分大小写

**注意点:**

1.变量名必须有意义

2.遵守驼峰命名法。首字母小写，后面单词的首字母需要大写。例如：

userName、userPassword

5.**获取变量的类型（typeof）**

**console.log(typeof age); // 'number'**

**4.数据类型(重点)**

js中有6种数据类型:Number、String、Boolean、Undefined、Null和Object（对象是引用数据类型）

我们一般把数据类型分为两个类,分别是:基本数据类型和引用数据类型

### **4.1 简单数据类型（**Number、String、Boolean、Undefined、Null**）**

**（1）Number类型**

1.1具体数字 ， 浮点数（浮点数的精度问题）

浮点的最数值高精度是 17 位小数，但在进行算术计算时其精确度远远不如整数

var result = 0.1 + 0.2; // 结果不是 0.3，而是：0.30000000000000004

console.log(0.07 \* 100); //7.000000000000001

1.2数值范围

最小值：Number.MIN\_VALUE，这个值为： 5e-324 科学技术法 //5乘以10负324方

最大值：Number.MAX\_VALUE，这个值为： 1.7976931348623157e+308

无穷大：Infinity 关键字

无穷小：-Infinity 关键字

1.3数值判断

NaN：not a number

NaN 与任何值都不相等，包括他本身

isNaN: is not a number

#### **（2）String类型**

**2.1**字符串字面量 （'程序猿'，'程序媛', "黑马程序猿"）

**2.2**转义符

#### **（3）Boolean类型**

#### 3.1 Boolean字面量： true和false，区分大小写

#### 3.2计算机内部存储：true为1，false为0

#### **Undefined和Null类型**

#### 1.undefined表示未定义，变量只声明的时候值默认是undefined

2.null表示一个空，变量的值如果想为null，必须手动设置 （返回的是一个对象）

### **4.2 引用数据类型（**Object **）**

## **5. 数据类型转换(重点)**

### 5.1**转换成字符串类型**

1. toString() var num = 5; console.log(num.toString());
2. String() ==== 》 强制转换

String()函数存在的意义：有些值没有toString()，这个时候可以使用String()。比如：undefined和null

1. 接字符串方式 '123' + 456 // '123456'

### 5.2 **转换成数值类型**

（1）Number() ==== 》强制转换

**Number()可以把任意值转换成数值，如果要转换的字符串中有一个不是数值的字符，返回NaN**

var num = Number("123ab") //NaN

var num1 = Number(null); //0

var num2 = Number(undefined) //NaN

1. parseInt() ==== > 开头要使数字，转换成整数

//把字符串转换成整数

var num1 = parseInt("12.3abc"); // 返回12，如果第一个字符是数字会解析直到遇到非数字结束

var num2 = parseInt("abc123"); // 返回NaN，如果第一个字符不是数字或者符号就返回NaN

var num3 = parseInt("123ab"); // 返回123,如果数字开头则返回开头部分的数字

### parseFloat() ====> 把字符串转换成浮点数

parseFloat会解析第一个. 遇到第二个 . 或者非数字结束

var num1 = parseFloat("12.3"); // 返回12.3

（4）+，- 等运算

var str = '500'; console.log(+str); // 500 console.log(-str);

### 5.3 **转换成布尔类型**

(1) Boolean() ==== 强制转换

0 ' ' null undefined NaN //会转换成false 其它都会转换成true

1. ! 运算符

!"123" // false

!!0 //false

(3)其他类型转布尔的规则：

字符串转布尔：空的字符串一定是false，其他是true

数字转布尔：0和NAN 是false，其他是true

Null转布尔：false

Undefined转布尔：false

JS 基础---- day02

## **1. 操作符( 运算符)(重点)**

### **1.1算术运算符 （+ - \* /）**

### 1.2 **一元运算符**

一元运算符：只有一个操作数的运算符

++ 自身加1 -- 自身减1

### **1.3逻辑运算符(布尔运算符)**

&& 与 两个操作数同时为true，结果为true，否则都是false

|| 或 两个操作数有一个为true，结果为true，否则为false

! 非 取反

&& 与

"123"&&1 //1 两个操作数同时为true，返回后面一个操作数,

""&&123 //"" 有一个是false,则返回那个为false的操作数,

""&&0 //"" 两个都是false,则返回前一个

|| 或

"123"||1 //1 两个操作数同时为true，返回第一个操作数,

""||123 //123 有一个是true,则返回那个为true的操作数,

""||0 //0 两个都是false,则返回后一个

! 非

!1 //false 取反

### **1.4关系运算符(比较运算符)**

< > >= <= == != === !==

==与===的区别：

==只进行值得比较

===类型和值同时相等,先比较类型

!== 操作数不相等或不同类型时返回true

**当比较运算涉及类型转换时,JavaScript 会按以下规则对字符串，数字，布尔类型的操作数进行操作**:

1.当比较数字和字符串时，字符串会转换成数字值。

2.如果其中一个操作数为布尔类型，那么布尔操作数如果为true，那么会转换为1，如果为false，会转换为整数0

x y == ===

0 false true false

"" false true false

"" 0 true false

"0" 0 true false

"17" 17 true false

null undefined true false

null 0 false false

null false false false

undefined false false false

0 NaN false false

"foo" NaN false false

NaN NaN false false

1. null == undefined 返回的是true

2. null 和undefined 和其他类型判断相等的时候,不转换类型

3. 一般只要有NaN参与的运算,返回都是NaN NaN转换成boolean 是false . NaN != NaN 返回的是true;

注意:NaN != NaN 的时候返回的是true

### **1.5赋值运算符** = += -= \*= /= %=

### **1.6运算符的优先级 (** 优先级从高到底 )

1. () 优先级最高

2. 一元运算符 ++ -- !

3. 算数运算符 先\* / % 后 + -

4. 比较(关系)运算符 > >= < <=

5. 相等运算符 == != === !==

6. 逻辑运算符 先&& 后||

7. 赋值运算符

## **2. 表达式和语句**

### **表达式**

结果是一个值的就是表达式.有可能是运算、函数调用、有可能是数据或者一个变量。表达式可以放在任何需要返回值的地方。

### **语句**

语句可以理解为一个行为，循环语句和判断语句就是典型的语句。一个程序有很多个语句组成，

一般情况下用 ; (分号)分割一个一个的语句

语句中可能包含表达式

## **3.流程控制(重点)**

程序的三种基本结构

### **顺序结构 :** 从上到下执行的代码就是顺序结构( 结构图)

**程序默认就是由上到下顺序执行的**

### **分支结构 ：**根据不同的情况，执行对应代码( 结构图)

### **循环结构** 循环结构：重复做一件事情

## **分支结构(重点)**

### **4.1 if语句**

语法结构（3种）

判断一个年份是闰年还是平年

闰年：能被4整除，但不能被100整除的年份 或者 能被400整除的年份

### **三元运算符（**是对if……else语句的一种简化写法**）**

表达式1 ? 表达式2 : 表达式3

### **4.2 switch语句**

格式：

显示星期几

素质教育（把分数变成ABCDE）千万不要写100个case

### **4.3布尔类型的隐式转换**

流程控制语句会把括号里的值隐式转换成布尔类型

转换为true 非空字符串 非0数字 true 任何对象

转换成false 空字符串 0 false null undefined NaN

**注意：**

brea不可以省略，如果省略，代码会继续执行下一个case(造成case穿透)

switch 语句在比较值时使用的是全等操作符, 因此不会发生类型转换（例如，字符串'10' 不等于数值 10）

## **5. 循环结构(重点)**

在javascript中，循环语句有四种，while、do..while、for、for..in循环。

注意 :for..in循环用于遍历对象

### **5.1while语句**

**基本语法：**

打印100以内 7的倍数

打印100以内所有偶数

打印100以内所有数的和

打印100以内所有偶数的和

打印100以内的奇数

打印100以内的奇数的和

### **5.2 do...while语句**

do..while循环和while循环非常像，二者经常可以相互替代，但是do..while的特点是不管条件成不成立，都会执行一次。

**基本语法：**

求100以内所有3的倍数的和

使用do-while循环：输出询问“我爱你，嫁给我吧？”，选择“你喜欢我吗？(y/n):"，如果输入为y则打印”我们形影不离“，若输入为n,则继续询问

### **5.3 for语句**

while和do...while一般用来解决无法确认次数的循环。for循环一般在循环次数确定的时候比较方便

for循环语法：

打印1-100之间所有数

求1-100之间所有数的和

求1-100之间所有数的和的平均值

求1-100之间所有偶数的和

同时求1-100之间所有偶数和奇数的和

打印正方形

打印直角三角形

打印9\*9乘法表

斐波那契====？（兔子的规律为数列，1，1，2，3，5，8，13，21）

### **continue和break**

break:立即跳出整个循环，即循环结束，开始执行循环后面的内容（直接跳到大括号）

continue:立即跳过当前循环，继续下一次循环（跳到i++的地方）

案例：求1-100之间不能被7整除的整数的和（用continue）

求200-300之间所有的奇数的和（用continue）

求200-300之间第一个能被7整数的数（break）

**调试**

- 过去调试JavaScript的方式

- alert()

- console.log()

- 断点调试

断点调试是指自己在程序的某一行设置一个断点，调试时，程序运行到这一行就会停住，然后你可以一步一步往下调试，调试过程中可以看各个变量当前的值，出错的话，调试到出错的代码行即显示错误，停下。

- 调试步骤

浏览器中按F12-->sources-->找到需要调试的文件-->在程序的某一行设置断点

- 调试中的相关操作

Watch: 监视，通过watch可以监视变量的值的变化，非常的常用。

F10: 程序单步执行，让程序一行一行的执行，这个时候，观察watch中变量的值的变化。

F8：跳到下一个断点处，如果后面没有断点了，则程序执行结束。

tips: 监视变量，不要监视表达式，因为监视了表达式，那么这个表达式也会执行。

JS基础 ===== Day03

## **1.数组(重点)**

### **1.1数组的概念**

所谓数组，就是将多个元素（通常是同一类型）按一定顺序排列放到一个集合中，那么这个集合我们就称之为数组 （数组是一个Array对象 ）

**注意**:数组可以存放任意的数据，并且数组的长度可以动态的调整。

### **1.2数组的创建方式**

1. 常规方式(利用Array构造函数方式):

var myCars=new Array();

myCars[0]="Saab";

myCars[1]="Volvo";

myCars[2]="BMW";

1. 简洁方式(利用Array构造函数方式):

var myCars=new Array("Saab","Volvo","BMW");

1. 字面量(最常用):

// 创建一个空数组

var arr1 = [];

// 创建一个包含3个数值的数组，多个数组项以逗号隔开

var arr2 = [1, 3, 4];

// 创建一个包含2个字符串的数组

var arr3 = ['a', 'c'];

// 可以通过数组的length属性获取数组的长度

console.log(arr3.length);

// 可以设置length属性改变数组中元素的个数

arr3.length = 0;

### **1.3获取数组元素**

数组的取值

**格式：数组名[下标] 下标又称索引**

**功能：获取数组对应下标的那个值，如果下标不存在，则返回undefined。**

var arr = ['red',, 'green', 'blue'];

arr[0]; // red

arr[2]; // blue

arr[3]; // 这个数组的最大下标为2,因此返回undefined

### **1.4遍历数组**

遍历：遍及所有，对数组的每一个元素都访问一次就叫遍历。

### 数组遍历的基本语法：

### **1.5数组中新增元素**

### 数组的赋值

**格式：数组名[下标/索引] = 值;**

**如果下标有对应的值，会把原来的值覆盖，如果下标不存在，会给数组新增一个元素。**

var arr = ["red", "green", "blue"];

// 把red替换成了yellow

arr[0] = "yellow";

// 给数组新增加了一个pink的值

arr[3] = "pink";

求一组数中的所有数的和和平均值 [5,9,8,6,7,4,9]

求一组数中的最大值和最小值，以及所在位置

将字符串数组用|或其他符号分割转换成一个字符串

要求将数组中的0项去掉，将不为0的值存入一个新的数组，生成新的数组

翻转数组

冒泡排序，从小到大

将字符串数组用|或其他符号分割拼接成一个字符串 比如: [1,2,3,4,5] --> '1|2|3|4|5'

将数组中重复项去掉,返回一个新的数组 [1,3,4,5,4,3,9,7] --> [1,3,4,5,9,7]

判断一个数组中的值,是否都大于10 例如这个数组 [1,2,3,48,5,12],如果都大于则打印'都大于',否则打印'有未符合条件'

js基础===== 》 Day05

重点:

函数的创建

函数的使用

函数的三要素

函数中的arguments

## **函数(重点)**

### **1.为什么要有函数**

可以把一些重复的功能性的代码封装起来,可以重复使用,便于开发

### **2.什么是函数**

把一段相对独立的具有特定功能的代码块封装起来，形成一个独立实体，就是函数

函数是一个Function对象

//在代码中至少具有以下结构的就是函数

function(){}

### **函数的三要素**

### 3.1函数名 一般用于调用函数以及传递函数

### 3.2 参数 这个函数需要外部传入的数据

3.3 返回值 这个函数执行之后,需要返回的数据

### **函数的定义**

4.1 函数声明 （**注意:在预解析过程中,申明的函数也会被提升）**

function 函数名(){

// 函数体

}

4.2 函数表达式 （**注意:函数表达式的函数不会被提升**）

var fn = function() {

// 函数体

}

4.3 特点：函数声明的时候，函数体里的代码并不会执行，只有当函数被调用的时候才会执行。

### **5. 函数的调用**

5.1 调用函数的语法： ===== > 函数名();

5.2 作用

调用函数是为了执行函数体里面的代码。可以调用多次(重复使用)

### **6. 函数的参数**

语法： **函数内部是一个封闭的环境，可以通过参数的方式，把外部的值 传递给函数内部**

**带参数的函数声明**

function 函数名(形参1, 形参2, 形参...){

// 函数体

}

**带参数的函数调用**

函数名(实参1, 实参2, 实参3);

形参和实参

1. 形式参数：在声明一个函数的时候，为了函数的功能更加灵活，有些值是固定不了的，对于这些固定不了的值。我们可以给函数设置参数。这个参数没有具体的值，仅仅起到一个占位置的作用，我们通常称之为形式参数，也叫形参。
2. 实际参数：如果函数在声明时，设置了形参，那么在函数调用的时候就需要传入对应的参数，我们把传入的参数叫做实际参数，也叫实参。

### **函数的返回值**

当函数执行完的时候，并不是所有时候都要把结果打印。我们期望函数给我一些反馈（比如计算的结果返回进行后续的运算），这个时候可以让函数返回一些东西。也就是返回值。函数通过return返回一个返回值

**如果函数没有显示的使用 return语句 ，那么函数有默认的返回值：undefined**

**如果函数使用 return语句，那么跟再return后面的值，就成了函数的返回值**

**如果函数使用 return语句，但是return后面没有任何值，那么函数的返回值也是：undefined**

**函数使用return语句后，这个函数会在执行完 return 语句之后停止并立即退出，也就是说return后面的所有其他代码都不会再执行。**

求斐波那契数列Fibonacci中的第n个数是多少？ 1 1 2 3 5 8 13 21...

递归: 函数体内调用自己

### **arguments的使用**

arguments对象是比较特别的一个对象，实际上是当前函数的一个内置属性。也就是说所有函数都内置了一个arguments对象，arguments对象中存储了传递的所有的实参。arguments是一个伪数组，因此及可以进行遍历

使用函数封装冒泡排序,返回排好序的数组

使用函数封装数组翻转,返回翻转后的数组

使用函数封装数组去重,返回去重后的数组(选做)

## **函数其它**

### **匿名函数===== 没有名字的函数**

匿名函数如何使用：

1.将匿名函数赋值给一个变量，这样就可以通过变量进行调用

2.匿名函数自调用 关于自执行函数（匿名函数自调用）的作用：防止全局变量污染。

3.把匿名函数作为参数,传递到另一个函数中(回调函数)

### **自调用函数 ====** 匿名函数不能通过直接调用来执行，因此可以通过匿名函数的自调用的方式来执行

(function () { alert(123);

})();

### **函数是一种数据类型**

function fn() {}

console.log(typeof fn); ===== >>>> function

(1.) 函数作为参数

因为函数也是一种类型，可以把函数作为另一个函数的参数，在另一个函数中调用

(2.) 函数做为返回值

因为函数是一种类型，所以可以把函数可以作为返回值从函数内部返回.

function fn(b) {

var a = 10;

return function () {

alert(a+b);

}

}

fn(15)();

js基础day06

重点:

作用域

作用域链

词法作用域规则

预解析

对象的创建

对象的使用

## **1.作用域(重点)**

**作用域：** 变量可以起作用的范围

**全局作用域:** 在script标签或者一个独立的js文件中

**局部作用域:** 函数可以构成一个局部作用域

### **全局变量和局部变量**

全局变量

在全局作用域中定义的变量叫做全局变量.全局变量在任何地方都可以访问到

局部变量

在局部作用域中定义的变量叫做局部变量。局部变量只有在定义该变量的函数中可以访问

（）不使用var声明的变量有可能是隐式全局变量，不推荐使用。

函数执行完之后,函数体的变量会销毁，全局变量关闭网页或浏览器才会销毁

### **词法作用域**

变量的作用域是在定义时决定而不是执行时决定，也就是说词法作用域取决于源码，通过静态分析就能确定，因此词法作用域也叫做静态作用域。

**在 js 中词法作用域规则:**

（1）函数内部允许访问函数外的数据.

（2）整个代码结构中只有函数可以限定作用域.

（3）作用域规则首先使用提升规则分析

（4）如果当前作用规则中有名字了, 就不考虑外面的名字

### **作用域链**

只有函数可以制造作用域结构， 那么只要是代码，就至少有一个作用域, 即全局作用域。凡是代码中有函数，那么这个函数就构成另一个作用域。如果函数中还有函数，那么在这个作用域中就又可以诞生一个作用域。

将这样的所有的作用域列出来，可以有一个结构: 函数内指向函数外的链式结构。就称作作用域链。

作用域链 图++++

## **2. 预解析(重点)**

JavaScript代码的执行是由浏览器中的JavaScript解析器来执行的。JavaScript解析器执行JavaScript代码的时候，分为两个过程：预解析过程和代码执行过程

预解析过程：

1. 把声明的变量提升到当前作用域的最前面，只会提升声明，不会提升赋值。

2.把声明的函数提升到当前作用域的最前面，只会提升声明，不会提升调用。

3.先提升var，在提升function

4.如果在同一个作用域中申明的变量和申明的函数重名,那么函数的优先级更高

### **全局解析规则**

### **函数内部解析规则**

### **变量提升**

定义变量的时候，变量的声明(使用var 申明的变量)会被提升到作用域的前面，变量的赋值不会提升

### **函数提升**

### JavaScript解析器首先会把当前作用域的函数声明(函数表达式提升的是前面的变量)提前到整个作用域的前面

## **3. 对象(重点)**

### **为什么要有对象**

1.对象也是我们存储数据的一种常用方式

2.面向对象编程(js高级部分)

### **什么是对象**

具体的事物，一个具体的事物就会有行为和特征。

### **JavaScript中的对象**