学习一门编程语言的基本步骤

* 了解背景知识：历史，现状，特点，应用场景
* 搭建开发环境，编写hello world
* 变量和常量
* 数据类型
* 运算符
* 逻辑结构
* 函数和对象
* 通用小程序
* 第三方的库和框架
* 个人项目

程序员必做50题

* JS概述
* 历史

1995年出现在网景公司的浏览器中

1995年，IE3中也出现了JS，叫做Jscript

1997年，出现了国际标准规范ECMAScript

2009年，出现了CommonJS规范，JS开始往服务器端发展，出现了Nodejs

* 现状

JS既可以运行在浏览器中，也可以运行在服务器端

* 特点

解释型语言，编译一行执行一行

弱类型语言

基于对象

跨平台，运行在不同的操作系统

* 应用场景

开发浏览器端的交互效果，进行服务器端的开发

* 搭建开发环境
* 浏览器

浏览器自带的JS解释器，推荐使用谷歌浏览器

火狐 谷歌 IE safari 欧朋

* 服务器端

安装Nodejs

[www.nodejs.org](http://www.nodejs.org/) 官网

打开命令行窗口

Node –v 回车 查看版本号

（3） 运行JS

创建01.JS和01.html两个文件，把js嵌入到html中

<script src=‘01.js’></script>

Nodejs

Node 空格 拖拽文件 回车

3 JS的语法规范

区分大小写

每行代码结束的分号可以省略

分为单行注释（//...）和多行注释（/\*...\*/）

* 变量

用于保存程序中的数据

* 声明变量

Var x=1;

使用关键字 var 声明变量，x是变量名称，在内存中开辟了一块空间，把数据保存进去。

练习：声明多个变量，分别保存一个员工编号，姓名，性别，生日，工资，并打印出

* 命名规范

变量名称可以由字母，数字，下划线，美元符号组成，不能以数字开头，不能使用关键字，不推荐使用汉字

多个单词组合作为变量名 可以用下划线分辨

userNameOne 驼峰命名法 每个单词首字母大写

（3） 变量赋值

变量声明未赋值则为undeined（未定义），变量可以多次赋值，赋不同类型的值，是弱类型语言的特点。

练习：声明多个变量分别保存一个人的成绩，声明一个变量保存总成绩，最后打印出总成绩。

* 一次声明多个变量

Var a=1,b=2

多个变量之间用逗号隔开

练习：声明多个变量保存商品的单价和数量，声明变量保存总价，打印出总价

课后任务

* 复习今天的内容，整理思维导图
* 重新编写SQL查询语句

练习： 交换两个变量中的数据

a=1 b=2 转换 a=2 b=1

3 预习JS第二天，数据类型，数据类型转换，运算符

1 常量

声明后必须赋值，不允许重新赋值

const userName=‘root’；

练习：声明常量保存圆周率，声明变量保存半径，分别计算园的周长和面积并打印出来

2 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，空

引用类型是对象，后边讲

（1）数值型

整数

十进制 1 2 3 4 5 6 7 8 9 10

八进制 1 2 ... 7 10 11 12

十六进制 1 2 ... 8 9 a-f 代表10-15

八进制特点 以0开头的数字

十六进制 以0x开头的数字，a-f代表10-15，不区分大小写

浮点数

3141.5-1

314.15

31.415e1

3.1415e2

tepeof 检测数据类型

（2）字符串型

被引号所包含的数据， 不区分单双引号

查看任意一个字符的Unicode码

‘a'.charCodeAt()

(3) 布尔型

只有两个值，分别是true和false，表示真和假

条件的结果会是布尔型，还可以用于保存状态，例如是否登录，是否注册等等...

（4）未定义型

只有一个值为undefined

声明了变量未赋值为undefinde

（5） 空

只有一个值为null,常和引用类型数据一起使用

3.数据类型转换

（1） 隐式转换

运行过程中自动产生的转换

① 数值+字符串 //数值转为字符串

2+‘3’ //‘23’

（2）数值+布尔型 //布尔型转为数值 true-1 false-2

2+true //3

2+false //4

（3）字符串+布尔型 //布尔型转为字符串

‘4’+true //‘4true'

练习 查看一下程序的运行结果

var a=1,b=true,c='tedu';

console.log(a+b+c);//2tedu

console.log(b+c+a);//truetedu1

console.log(c+a+b);//tedu1true

练习 假设从后端获取到了一个员工的编号，姓名，打印出以下格式，员工编号 姓名

js中加号（+）的作用

加法运算

执行字符串之间的拼接

NaN(Not a Number):不是一个数字，在将数据转化为数值型的时候，没有成功返回的结果。NaN和任何数字执行加减乘除等运算结果还是NaN

结论：隐式转换为数值自动调用Number函数

②强制转

（1）强制转为数值

Number('5a')//NaN

Number('5')//5

Number(true)//1

Number=(false)//0

Number(undefined)//NaN

Number(null）//0

(2)强制转为整型

parseInt(

用于把字符串和小数转为整型,其他转换都是NaN

parseInt(1.9)//1

parseInt(‘3.14')//3

parseInt(‘3.14a’)//3

parseInt(‘a3.14)//NaN

(3)强制转为浮点型

parseFloat()

用于将字符串转换为浮点型

parseFloat('5.18');//5.18

parseFloat('4.7a');//4.7

parseFloat('a4.7');//NaN

parseFloat('3b');//3

（4）强制将数值和布尔型转为字符串

var num=5;

//将转换后得到的字符串保存到变量str中

var str=num.toString();

4.运算符

表达式：由数据或者由运算符所连接的操作数据组成的形式

运算符分为六大类

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算数运算符

+ - \* / % ++ --

%（取余）

++（自增） 在原来基础之上加1

--（自减） 在原来基础之上减1

var a=1

var b=a++；//先把a的值赋给b，然后a在执行自增

var c=2

var d=++c，//先让c的值进行自增，然后将自增的结果赋给d

练习 查看以下程序的运行结果

var num=8

var num1=num--;

var num2=--num;

console.log(num)

//面试题

//+ -正负，隐式转换

console.log(5 + + ‘3')；//8

//自增和自减，隐式转换为数值

var a=‘2';

a++；

console.log(a);//3

②比较运算符

> < >= <= ==（等于） !=(不等于) ===（全等于） ！==（不全等于）

== 等于 ，比较值是否相同

=== 全等于，先比较类型，再比较值（推荐）

3>‘10’数字和字符串（布尔型）比较，隐式转换为数值

‘3’>‘10'两个字符串比较，比较的是首个字符的unicode码

‘3’->51 ‘1’->49

3>‘10’//false

3<‘10’//false

3==‘10’//false

NaN==NaN //false

NaN和任何值比较（ > < >= == ===)都是false

③逻辑运算符

&& (逻辑与) 表示并且，关联的两个条件都是true，结果是true，否则是fslse

|| （逻辑或） 表示或者,关联的两个条件至少有一个为true,否则false

! （逻辑非） 非 取反

console.log（！false）//true

短路逻辑

&& 当第一个条件为false，不再执行第二个条件

|| 当第一个条件为true，不再执行第二个条件

短路逻辑无需关注整体上是true还是false，重点是看第二部分有没有执行

练习：查看一下程序是否报错

var num=5;

num>8 && console,log(a);

课后任务

（1） 复习今天内容，整理思维导图

（2） 练习：声明变量保存任意一个年份，使用短路逻辑，如果是闰年，打印‘闰年’

4年一闰，100年不闰，400年再闰

能被4整除，并且不能被100整除，或者能被400整除

（3）预习js day3

④ 位运算符（简单了解）

模拟计算机底层的运算过程，先将值转为二进制然后再运算，运算完再把结果转回成十进制

1 2 3 4 5 6 7 8 9

1 10 11 100 101 110 111 1000 1001

& 按位与，上下两位都是1结果是1，否则是0

| 按位或，上下两位含有1结果是1，否则是0

^ 按位异或，上下两位不同是1，否则是0

3&5 5|7 5&7 9^3

011. 101 101 1001

101. 111 111 0011

001 111 101 1010

1 7 5 10

>> 按位右移 去掉最后的若干位

<< 按位左移 在最后补若干个0

7>>1 让7往右移动一位 符号后数字决定去掉几位

去掉最后一位 -> 11

⑤ 赋值运算符

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

计算赋值 ：先执行计算，在执行赋值

练习：声明变量保存商品的价格，让价格执行打八折，最后打印价格

⑥三目运算符

一目运算符：由一个运算符连接了一个操作数据或者表达式 ++ -- ！

二目运算符：由一个运算符连接了两个操作数据或者表达式

三目运算符：由两个运算符连接了三个操作数据或者表达式 ? :

条件表达式 ? 表达式1 : 表达式2

如果条件表达式为true，执行表达式1，否则执行表达式2

5.浏览器端函数

alert() 弹出警示框

prompt() 弹出提示（输入框），需要使用变量保存用户输入的值，类型是字符串型，如果点击取消返回显示 null

练习：要求弹出两次提示框，分别输入数字，最后要计算两个数字相加的和并打印出来

程序=数据+算法

程序的执行方式分为顺序执行、选择执行、循环执行

6.流程控制（选择执行）

（1）if语句

满30减20

if（条件表达式）{

语句块

}

如果if后语句中的只有一行语句，可以省略大括号。

以下数据作为条件表达式出现，隐式转换为fslse

0 ‘’ 两个引号空字符 NaN undefined null

声明一个变量，保存用户的签名内容，如果签名内容为空，设置签名内容‘这家伙很懒，什么也没留下’，最后打印签名内容

（2）if-else语句

if（条件表达式）{

语句块1

}else{

语句块2

}

（3）if-else 嵌套

if(条件表达式1){

语句块

}else if(条件表达式2）{

语句块2

}else{

语句块n

} //以上所有的条件都是false

（4）switch-case

根据一个表达式的值，选择执行对应的语句块，属于是一种特殊的多项分支语句

switch(表达式){

case 值1：//会将表达式和值1比较

语句块1

break；//结束，跳出switch-case语句

case 值2：

语句块2

break;

default:

语句块n

}

switch后的表达式在case后的值比较的时候只能进行全等于（===），要求值和类型都相同

练习：声明变量保存任意一个城市名称，根据城市的名称打印对应的美食

课后任务：

①复习今天内容，整理思维导图

②练习：弹出两次提示框，分别输入商品的单价和数量，计算出总价，如果总价满1000打九折，假设会员卡内有余额800，如果足以支付，警示框弹出‘pay success’,否则警示框弹出‘pay error’。

③预习JS第四天，循环

1.循环执行

循环是一遍又一遍执行的相同或者相似的内容

循环条件：可以控制循环的次数

循环体：要重复执行的相同或者相似内容

（1）while循环

while（循环条件）{

循环体

}

（2）break

在循环体中执行后，就会跳出循环。

（3）do-while循环

do{

循环体

}while(循环条件）；

练习：循环弹出提示框，声明变量保存用户输入的值，如果输入正确强制结束循环，警示框弹出‘login success’;如果输入的次数超过三次，强制结束循环，提示警示框‘game over’.

(4)for循环

for(初始值；循环条件；增量){

循环体

}

2.break和continue

用于循环体中

break 跳出循环，结束，不再执行任何循环中的内容

continue 跳过某一次循环，不执行某一次中剩余的循环体，还还执行下一次的循环

练习：打印1-100之间所有的奇数，遇到偶数就跳过

3.循环嵌套

在一个循环体的内部，还有其他的循环；任意两个循环之间可以相互嵌套

课后任务：

1.复习今天内容

2.练习

计算1-100之间所有整数的和，一旦总和超过4000，强制结束循环，打印当前和sum+=i

循环打印出2000-2100之间的前10个闰年

使用循环嵌套打印九九乘法表

1.函数 16号

Number()/parselnt()/parseLloat()/alert()/prompt()..

函数：是一个功能体，提供若干个数据，返回处理的结果，为了封装重复执行的代码。

函数分为系统函数和自定义函数

（1）创建普通函数

function 函数名称（）{

函数体——封装的要重复执行的代码

}

调用

函数的名称加（）

每调用一次，就执行一次函数体中的代码

练习：创建函数 getSum,在函数体中封装计算1-100之间所有整数的和并打印出来，调用多次

（2）创建带有参数的函数

dunction 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）

参数：创建函数是的参数称为形参，调用函数时的参数称为实参；多个参数之间用逗号隔开，实参的数量可以多于或者少于形参的数量，如果形参未赋值则为undefined

练习：创建函数getSun，在函数体中计算1-任意数字之间所有整数的和并打印出来，调用多次

练习：创建函数getSun，计算出任意两个年份之间所有的闰年个数并打印出来，调用多个参数

（3）创建带有返回值的函数

function 函数名称（参数列表）{

函数体

ruturn 值 //返回值，函数调用后得到的结果

}

调用

函数名称（参数列表）

return,返回，用于将函数调用的结果返回出去；ruturn后不加值或者再函数中不写rutron，则返回undefined; return一旦执行，跳出函数，结束函数的调用

练习：创建函数getMax1，传递任意两个数字，返回最大值

练习：创建函数getMax2，传递任意三个数字，返回最大值

练习：创建函数getStatus,传递订单的状态码，返回对应的中文状态。

① 等待付款 ② 等待发货 ③ 运输中

④ 已签收 ⑤已取消

使用swtich-case

对比return和break

return 用于函数中，一旦执行就会跳出函数，结束函数的调用

break 用于 switch-case 语句和循环，一旦执行就会跳出switch-case或者循环，结束相应语句的执行

练习：创建函数isRun，传递任意一个年份，查看是否为闰年，返回一个布尔型的值

2.变量的作用域

作用域： 分为全局作用域和函数作用域，具体指变量和函数的可访问范围

全局变量：在函数外声明的变量，叫全局变量，在任何作用域下都可以访问到

局部变量：在函数内使用var声明的变量，叫局部变量，只能在当前的作用域下访问到

在函数中不加var声明的变量是全局变量，不推荐，后期严格模式下会报错

变量提升：JS程序执行前，会将var声明的变量提升到所在作用域的最前面，只是提升证明，赋值不提升

3.函数的作用域

全局函数：在全局的作用域下创建的函数，可以在任意作用域下访问

局部函数：在函数作用域下创建的函数，只能在所在的作用域下访问

函数提升：

JS程序执行前，会将整个函数提升到所在作用域的最前面，可以先写调用，再写创建

练习：创建一个函数，能否在创建之前调用到

4.递归（提前了解）

在函数内调用自身这个函数，本质是一个死循环

斐波那契数列：第1项和第2项的值为1，从第3项开始，每项的值是前两项相加的值

1 1 2 3 5 8 13 ...

课后任务

（1）复习今天内容，整理思维导图

（2）练习：使用递归循环打印三次

练习：创建函数，传递任意一个数字，返回斐波那契数列中该项的值

使用循环

（3）预习js第六天 匿名函数、对象