#### **目标:高级前端、大前端、前端全栈工程师**

[一、课程体系 1](#_Toc524105575)

[阶段1: 项目流程、数据库、JS基础、Node.js 2](#_Toc524105576)

[1、项目流程与数据库 2](#_Toc524105577)

[DAY01 软件周期 2](#_Toc524105578)

[DAY02 服务器&数据库 4](#_Toc524105579)

[DAY03 语句操作&列约束 7](#_Toc524105580)

[DAY04 列约束以及各种查询 10](#_Toc524105581)

[复习以及练习 16](#_Toc524105582)

[2、JavaScript基础 22](#_Toc524105583)

[DAY01 JavaScript介绍 22](#_Toc524105584)

[DAY02类型 27](#_Toc524105585)

[DAY03 if 31](#_Toc524105586)

[DAY04 switch for do循环 34](#_Toc524105587)

[DAY05 for循环 37](#_Toc524105588)

[DAY06函数 对象 40](#_Toc524105589)

[DAY07 数组 43](#_Toc524105590)

[DAY08 字符串操作 48](#_Toc524105591)

[DAY09 Date Number Boolean对象 51](#_Toc524105592)

[3、Node.js基础-成亮 53](#_Toc524105593)

[DAY01 全局对象 模块系统 53](#_Toc524105594)

[DAY02 包和npm以及其他模块 58](#_Toc524105595)

[DAY03 Node服务器搭建 61](#_Toc524105596)

[DAY04 中间件MySQL模块 63](#_Toc524105597)

[DAY05-DAY06项目练习 66](#_Toc524105598)

# 一、课程体系

**学习目标：高级前端、大前端、前端全栈工程师**

阶段1：项目流程、数据库、JS基础、Node.js

阶段2：HTML基础、AJAX、CSS、Bootstrap

阶段3：JS高级、BOM&DOM、jQuery、Vue.js

阶段4：VueUI、H5高级、Node.js

阶段5：Angular、React

阶段6：Ionic、PhoneGap、微信

2.学前准备

(1)全部技术内容从零开始、专业没要求

(2)英语单词从零开始、只要下功夫

(3)对时间有要求！早上-白天-晚上-夜间

(4)对键速有要求！—— 350次/分钟 TopSpeeder

3.Windows常用快捷键

Win+e：打开我的电脑

Win+d：显示/隐藏桌面

Win+r：打开“运行”窗口 calc/mspaint/cmd

Alt+Tab：在多个窗口间切换(从左往右)

Alt+Shift+Tab：在多个窗口间切换(从右往左)

Alt+F4：关闭当前窗口

午间任务：打开电脑，删除c:/xampp/htdocs/目录下所有的内容，重新创建一个目录SERVER\_DAY01，进入该目录，创建day01.docx文件，打开该文件，写些内容，保存退出。关闭所有窗口。 —— 全程不许碰鼠标！

打开我的电脑，在D盘下创建一个目录public，进入该目录，创建一个空白文本文档（.txt），重命名为index.html，使用Editplus软件打开该文档，在其中输入html:5，再点击Tab键，可以看到生成了很多内容，试着删除其中的第3~6行的内容，保存退出。

4.软件工程

Debug： Ada - Bug - Debug（调试程序）

IBM - DOS - Bill Gates - Microsoft – Windows

## 阶段1: 项目流程、数据库、JS基础、Node.js

### 1、项目流程与数据库

#### DAY01 软件周期

成亮

18514668981

chengliang@tedu.cn

1.软件工程

IBM -> DOS(Bill Gates) -> Microsoft -> Windows

软件工程学科

(1)软件过程

(2)软件测试

(3)软件度量

(4)软件质量保证

(5)软件标准

2.软件过程

软件的开发流程/软件生命周期

**(1)软件定义期**

①可行性研究阶段——《可行性研究报告》

技术、人力、设备、时间、资金、回报、政策、风俗

②需求分析阶段——《需求分析报告》

功能性需求、非功能性需求

**(2)软件开发期**

③概要设计阶段

子系统、模块、各自的功能

④详细设计阶段

主体内容、使用的程序、算法、数据库...

⑤编码实现阶段

设计师：出效果图

前端：把效果图转换成网页(html、css、js)

后端：为前端提供页面中所需要的数据

⑥测试阶段

**(3)软件维护期**

⑦项目部署阶段

部署到服务器

⑧项目维护阶段

|  |
| --- |
| 目标项目：学子商城(www.codeboy.com)  前台子系统：商品模块、用户模块、购物车模块  后台子系统：商品模块、用户模块、订单模块  移动端子系统：商品模块、用户模块、购物车模块 |

3.服务器

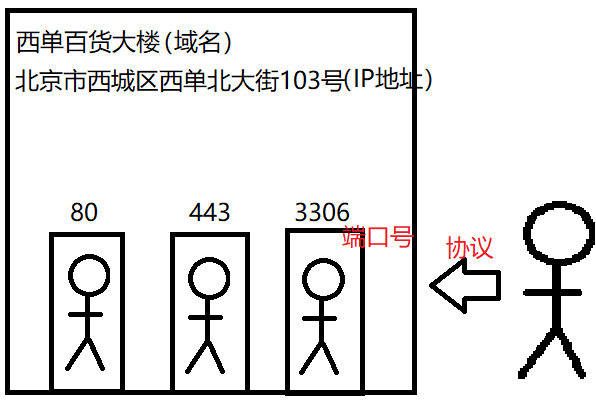
server：为客户端提供各种服务的功能强大的计算机。

访问服务器：

(1)服务的**域名/IP地址**

(2)每一个服务都有对应的**端口号**

(3)提供访问该服务所使用的**协议**



访问web服务器(提供网页服务)

http://www.codeboy.com:80

**访问自己电脑创建的web服务器**

启动web服务器 /webServer/start.cmd

http://127.0.0.1:80 通过服务器的ip地址

http://localhost:80 通过服务器的域名

练习：在public下创建4.html，随便输入内容，然后使用服务器来访问。

**获取自己电脑在局域网的IP地址**

在命令行输入 ipconfig

在浏览器中 http:// IP地址:80 回车

例如：http://172.163.100.163:80

练习：同桌之间尝试使用局域网IP地址访问对方服务器。

#### DAY02 服务器&数据库

复习

软件开发流程

软件定义期

可行性研究阶段

需求分析阶段

软件开发期

概要设计阶段

详细设计阶段

编码实现阶段

测试阶段

软件维护期

项目部署阶段

项目维护阶段

服务器

提供服务，功能强大的计算机

访问服务器下的某一个服务器

访问服务器 IP地址/域名 端口 协议

访问服务器下的web服务器

http://127.0.0.1:80

http://localhost:80

http://172.163.100.163:80

输入http://www.taobao.com 如何获取淘宝首页的。

源码下载地址

http://code.tarena.com.cn

用户名：tarenacode 密码：code\_2013

今日目标：

项目中如何保存数据——数据库 （难点：单词）

1.保存数据的几种形式

内存 临时保存数据

文件 不便于管理

第三方机构 腾讯云，百度云，阿里云

独立数据库服务器

2.数据库概述

Database：数据库，以特定的结构批量存储数据的软件。

数据库的发展史

网状数据库->层次型数据库->**关系型数据库**->非关系型数据库

|  |
| --- |
| 关系型数据库中数据的逻辑结构  Server->Database->Table->Row->Column  服务器->数据库->表->行->列 |

3.MySQL

MySQLDB分支：https://www.mysql.com/

MariaDB分支：https://mariadb.org/

xampp下载地址

https://www.apachefriends.org/download.html

|  |
| --- |
| xampp是一个服务器套装，包含多款服务器软件。 |
| d->Deamon 精灵，守护者  mysqld.exe 启动MySQL服务器 |

4.MySQL数据库系统：

(1)服务器端：

负责存储数据

C:/xampp/..../mysqld.exe

(2)客户端

负责向服务器发起增删改查的命令

C:/xampp/.../mysql.exe

(3)连接MySQL数据库服务器

mysql.exe -h127.0.0.1 -P3306 -uroot -p

简写： mysql -uroot

-h 服务器的IP地址/域名 可以使用localhost

-P 端口，MySQL服务使用3306端口

-u 用户

-p 密码

练习：尝试使用错误的信息连接数据库服务器。

5.MySQL管理命令

quit; 退出服务器连接

show databases; 显示数据库服务器上当前所有的数据库

use 数据库名; 进入指定的数据库

show tables; 显示当前数据库中所有的数据表

desc 表名; 描述表中有哪些列(表头)

练习：查看phpmyadmin数据库下以下表都有哪些列

pma\_\_favorite 2

pma\_\_history 6

pma\_\_recent 2

6.MySQL常用的SQL命令

SQL: Structured Query Language，结构化查询语言，用于操作关系型数据库服务器中的数据。——增删改查

SQL语言最早是由IBM提出，后提交给ISO，成为行业内标准语言。分为多个版本:SQL-87、SQL-92、SQL-99。当前标准的SQL语言被绝大多数的关系型数据库所支持。

|  |
| --- |
| SQL命令的两种执行方式  (1)交互模式：客户端连接MySQL，输入一行，回车，服务器执行一行。适用于临时性的查看数据  mysql -uroot 回车  (2)脚本模式：客户端把要执行的多行命令写在一个脚本文件中，一次性的提交给服务器执行。适用于批量的操作数据。  mysql -uroot < C:/xampp/...../02.sql |

练习：使用脚本模式向数据库服务器提交多行SQL命令，尝试写错其中一行，执行查看结果。

**SQL语法规范**

(1)每条SQL语句必须以英文分号作为结尾，一条语句可以跨越多行，见到分号认为结束。

(2)若第n条SQL语句有语法错误，则后续所有的语句不会再执行。

(3)SQL语句中不区分大小写。习惯上数据库关键字用大写，非关键字用小写。

(4)SQL语句还可以使用单行注释：#... 或者多行注释: /\*...\*/，注释的内容不会被服务器所执行。

**日常开发中常用的SQL命令**

(1)丢弃数据库，如果存在的话

DROP DATABASE IF EXISTS jd;

(2)创建新的数据库

CREATE DATABASE jd;

(3)进入刚刚创建的数据库

USE jd;

(4)创建数据表

CREATE TABLE student(

sid INT,

name VARCHAR(6),

sex VARCHAR(1),

score INT

);

(5)向数据表中插入记录

INSERT INTO student VALUES

('10','Tom','M','89'),('11','Kate','F','92').....;

(6)查询数据表所有的记录

SELECT \* FROM student;

练习：创建脚本文件05.sql；先丢弃数据库tedu，如果存在的话；创建数据库tedu;

练习：创建脚本文件06.sql; 先丢弃数据库xuezi，如果存在的话；创建数据表laptop，包含列有lid(编号)，title(标题)，price(价格)，number(数量)；插入3条记录，查询出所有记录。

课后任务：

(1)复习今天内容，保留注释，重新编写代码

(2)练习：

创建脚本文件xz.sql，重新创建数据库xuezi；丢弃数据库xuezi,如果存在。创建数据表user，包含列uid编号，uname用户名，upwd密码，email邮箱，phone电话，sex性别，user\_name真实姓名，registerTime注册时间。 插入3条记录，查询出所有记录。

(3)预习MySQL手册第11章列类型

#### DAY03 语句操作&列约束

复习

网状数据库，层次型数据库，**关系型数据库**，非关系数据库

MySQL

server->database->table->row->column

连接

mysql.exe -h127.0.0.1 -P3306 -uroot -p

mysql -uroot

脚本模式

mysql -uroot < C:/xampp/..../xz.sql

mysql管理命令

quit;/show databases;/use 数据库名;/show tables;/desc 表名;

SQL命令

CREATE DATABASE xz;

USE xz;

CREATE TABLE user(

uid INT,

...

);

INSERT INTO user VALUES(),()...;

SELECT \* FROM user;

1.MySQL中常用SQL命令

增删改查

(1)更改记录

UPDATE user SET

upwd='654321', email='wh@tedu.cn' WHERE uid='1';

练习：更改编号为2的数据，电话改为1318888888，性别改为'M',注册时间改为 '2017-3-1'

(2)删除记录

DELETE FROM user WHERE uid='3';

2.数据库中的中文乱码

**计算机中如何存储英文字符**

ASCII: 总共有128个，对所有的英文字符及符号进行了编码。

hello 72698977

Latin-1: 总共有256个，兼容ASCII，同时对欧洲符号进行了编码

**计算机中如何存储中文字符**

GB2312: 对6千多常用的汉字进行了编码，兼容ASCII码

GBK: 对2万多汉字进行了编码，兼容GB2312

BIG5: 台湾繁体字编码，不兼容GB2312和GBK；兼容ASCII码

Unicode：对世界上主要的语言的字符进行编码，兼容ASCII码，不兼容GBK，GB2312，BIG5等。具体分为UTF-8，UTF-16，UTF-32三种存储方案。

**中文乱码产生的根源**

MySQL默认使用Latin-1编码，而这个编码不能存储中文的。

**解决MySQL存储中文乱码**

(1)sql脚本文件另存为的编码

(2)客户端连接服务器使用的编码

(3)服务器端数据库存储数据使用的编码

都统一使用UTF8编码。

**设置editplus的默认编码：**

Tools->preferences->file->default encoding

练习：创建脚本文件01\_dangdang.sql；设置连接数据库编码utf8，创建数据库dangdang，进入该数据库，创建数据表book,包含bid编号，title标题，price价格，pubDate出版时间，isJia是否加价购(是->1，不是->0)；插入4本书的信息，修改第2条书的信息，删除第3条，查询结果。

解决存储中文

3.MySQL中列类型

列类型：创建表的时候，指定列可以存储的数据类型

CREATE TABLE book( bid 列类型 );

**(1)数值类型——可以不加引号**

TINYINT: 微整型，占用1个字节，-128~127

SMALLINT: 小整型，占用2个字节，-32768~32767

INT: 整型，占用4个字节，-2147483648~2147483647

BIGINT: 大整型，占用8个字节，范围很大

FLOAT: 单精度浮点型，占4个字节，3.4E38，范围比INT大的多，但是精度不如INT，可能产生误差。

DOUBLE: 双精度浮点型，占8个字节，1.79E308，范围比BIGINT大的多，但是精度不如BIGINT，可能产生误差。

DECIMAL(M,D): 定点小数，不会产生计算误差舍入；M代表总的有效位数(不包含小数点)，D代表小数点后有效位数。

BOOL: 布尔型，取值只能是TRUE/1、FALSE/0，数据库在存储的时候使用TINYINT(1或0); **TRUE和FALSE不能加引号**。

219.85

25000.32

123456.789\*10^-1

12345.6789

1234.56789\*10^1

123.456789\*10^2

12.3456789\*10^3

**(2)字符串类型——必须加引号**

VARCHAR(M): 变长字符串，不会产生空间浪费，操作速度相对慢，M不能超过65535。

CHAR(M): 定长字符串，可能产生空间浪费，操作速度相对快，M不能超过255。

TEXT(M): 大型变长字符串，M不超过2G

**(3)日期时间类型——必须加引号**

DATETIME: 日期时间类型 '2018-11-2 15:18:30'

DATE: 日期类型 '2018-11-2'

TIME: 时间类型 '15:18:30'

|  |  |  |
| --- | --- | --- |
|  | CHAR(5) | VARCHAR(5) |
| a | a\0\0\0\0 | a\0 |
| ab | ab\0\0\0 | ab\0 |
| abc | abc\0\0 | abc\0 |
| 达内 | 达内\0\0\0 | 达内\0 |

age TINYINT

empCount SMALLINT

commentCount Int

price DECIMAL(6,2) #9999.99

phone CHAR(11)

blog VARCHAR(8000)

isVip BOOL

练习：创建脚本文件02\_xuezi.sql；创建数据库xuezi，设置保存的字符编码为utf8；创建保存笔记本信息的表laptop，包含列lid编号，title标题，price价格，stockCount库存量，shelfTime上架时间，isIndex是否显示在首页。插入4条记录

练习：创建脚本文件03\_tedu.sql；创建数据库tedu，设置存储的编码；创建保存部门信息的表dept，包含(did，dname部门名称，empCount员工数量)，插入3条数据

10 研发部 3

20 市场部 2

30 运营部 1

创建保存员工信息的表emp，包含(eid，ename姓名，sex，birthday生日，salary工资，deptId所属部门编号)，插入对应的员工数据。

4.列约束

MySQL可以对插入的数据进行特定的检验，只有满足条件的数据才能插入成功，否则会提示“非法插入”，“禁止插入”。

例如：插入人的年龄范围是0~100之间，商品的价格大于0...

MySQL提供了六种列约束

(1)主键约束——PRIMARY KEY

声明了主键约束的列上不能插入重复的值，如果主键的数据是数值会自动按照由小到大的顺序排列——加快查找速度。注意：一个表中最多有一个主键。

|  |
| --- |
| MySQL中的NULL含义：空的，空缺的；  指应该有某个数据，但暂时还不能确定具体的指；  例如：还没到发年终奖，未决定员工的部门，未确定部门经理 |

课后任务：

(1)复习今天内容，重新编写代码

(2)练习：

创建脚本文件xuezi.sql，创建数据库xuezi，使用utf编码；创建保存笔记本分类表xz\_laptop\_family(fid，fname分类名称，laptopCount 笔记本数量)

10 戴尔 3

20 小米 1

30 苹果 2

创建笔记本信息表xz\_laptop，包含(lid，title，price，detail介绍，shelfTime上架时间，isOnsale是否在售，familyId所属分类)，插入6条记录。

#### DAY04 列约束以及各种查询

复习

更改记录

UPDATE emp SET salary=5000,sex=0 WHERE eid=5;

删除记录

DELETE FROM emp WHERE eid=6;

中文乱码

Latin-1

Unicode，utf8

sql脚本文件，客户端设置连接使用的编码，服务器存储数据使用的编码

列类型

数值型

TINYINT(-128~127) SMALLINT(-32768~32767) INT(-2147483648~2147483647) BIGINT FLOAT DOUBLE DECIMAL(6,2) 9998.99 BOOL

字符串型

VARCHAR(M) 65535

CHAR(M) 255

TEXT

日期时间型

DATE TIME DATETIME

列约束

主键约束 PRIMARY KEY

今日目标

列约束

简单查询

1.列约束

(1)主键约束——PRIMARY KEY

(2)唯一约束——UNIQUE

声明了唯一约束的列上不能插入重复的值，但可以插入NULL，而且可以插入多个NULL

(3)非空约束——NOT NULL

声明非空约束的列上不能插入NULL值，如果插入的是NULL也不会报错，会显示空内容。

(4)默认值约束——DEFAULT

可以使用DEFAULT关键字为列声明默认值；有以下两种用法：

INSERT INTO xz\_laptop VALUES(1,'小米air', DEFAULT...);

INSERT INTO xz\_laptop(lid,title) VALUES(2,'苹果pro');

(5)检查约束

检查约束可以对插入的数据范围进行验证

CREATE TABLE student(

age TINYINT CHECK(age>18 AND age<60)

);

MySQL不支持，会降低插入和更新等操作效率。

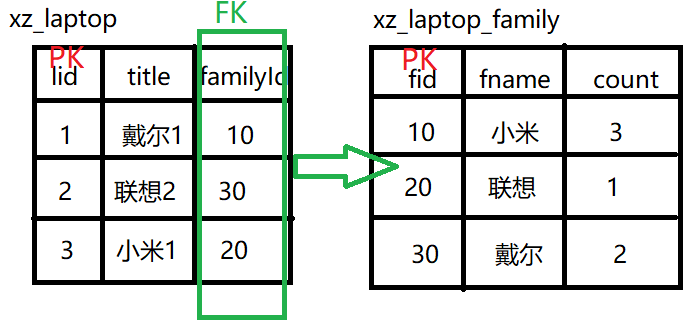
(6)外键约束——FOREIGN KEY

声明了外键约束的列，取值必须在另一个表的主键上出现过

取值可以是NULL;

**注意：**外键的列类型要和另外一个表主键的列类型一致。

FOREIGN KEY(familyId) REFERENCES xz\_laptop\_family(fid)



|  |
| --- |
| 所有的列约束中，主键约束推荐使用——可以提高查询的效率；其它的列约束是否使用根据项目而定，可以不用——会影响数据的插入和更新效率。 |

2.MySQL中的自增列

AUTO\_INCREMENT: 自动增加，加入一个列声明了自增列，无需手动赋值，直接指定为NULL，会自动获取当前的最大值，新纪录会执行+1然后插入。

注意：

允许手动赋值

只用于整数型的主键列上

练习：使用列约束；

创建脚本文件02\_tedu.sql；创建数据库tedu

创建部门表 dept，包含did，dname部门名称；

10 研发部 20 市场部 30 运营部 40 测试部

创建员工表 emp，包含eid，ename，sex，birthday生日，salary工资，deptId隶属的部门编号

插入14条记录，分属于10/20/30部门，其中最后一个员工不属于任何一个部门。

3.项目中如何存储日期时间数据

2018-11-25

2018年11月25日

11-25-2018

25/11/2018

**存储的是距离计算机元年的毫秒数，使用BIGINT类型。**

计算机元年 1970-1-1 0:0:0

1秒钟=1000毫秒

数据库存储1000 1970-1-1 0:0:1

1000\*60 1970-1-1 0:1:0

1000\*60\*60 1970-1-1 1:0:0

2018年距离计算机元年的毫秒：48\*365\*24\*60\*60\*1000

男 女

男孩 女孩

男人 女人

man woman

girl boy

male female

1 0

4.简单查询

**(1)查询所有的列**

SELECT \* FROM emp;

\*代表所有的列

**(2)查询特定的列**

示例：查询所有员工的姓名，生日，工资。

SELECT ename,birthday,salary FROM emp;

练习：查询所有员工的编号，姓名，性别，工资

SELECT eid,ename,sex,salary FROM emp;

**(3)给列取别名**

示例：查询所有员工的姓名和工资，列名使用汉字。

SELECT ename AS 姓名,salary AS 工资 FROM emp;

练习：查询所有员工的编号，姓名，性别，生日；全部使用中文别名。

SELECT eid AS 编号,ename AS 姓名,sex AS 性别,birthday AS 生日 FROM emp;

练习：查询所有员工的编号，姓名，性别，生日；全部使用1个英文字母作为别名。

SELECT eid a,ename b,sex c,birthday d FROM emp;

|  |
| --- |
| AS关键字可以省略 |

**(4)只显示不同的值/合并相同的项**

练习：查询公司都有哪些性别的员工

SELECT DISTINCT sex FROM emp;

练习：查询公司员工都分布在哪些部门

SELECT DISTINCT deptId FROM emp;

**(5)查询时执行运算**

示例：计算5+3-7/2\*9

SELECT 5+3-7/2\*9;

练习：查询所有员工的姓名及其年薪，并且起别名。

SELECT ename AS 姓名,salary\*12 AS 年薪 FROM emp;

练习：假设所有员工工资加500，年终奖是5000，查询出姓名和年薪，并且起别名。

SELECT ename AS 姓名,(salary+500)\*12+5000 AS 年薪 FROM emp;

(6)查询的结果排序

示例：查询所有员工的信息，结果按照工资由低到高排序

SELECT \* FROM emp ORDER BY salary ASC;

示例：查询所有员工的信息，结果按照工资由高到低排序

SELECT \* FROM emp ORDER BY salary DESC;

练习：查询所有员工的信息，结果按照年龄由大到小。

SELECT \* FROM emp ORDER BY birthday ASC;

练习：查询所有员工的信息，结果按照姓名升序排序。

SELECT \* FROM emp ORDER BY ename ASC;

练习：查询所有员工的信息，按生日由大到小，若生日相同，再按照姓名升序排列。

SELECT \* FROM emp ORDER BY birthday DESC,ename ASC;

练习：查询所有员工信息，按照工资由大到小排序，要求女员工必须排在男员工的前边。

SELECT \* FROM emp ORDER BY sex,salary DESC;

|  |
| --- |
| asc -> ascend 升序  desc -> descend 降序  如果不加排序规则，默认是按照升序排列（asc）。  ORDER BY可以按照**数值、日期、字符串(**字符的编码)来排序 |

(7)条件查询

示例：查询编号为5的员工工资

SELECT \* FROM emp WHERE eid=5;

|  |
| --- |
| SQL语句中支持的比较运算符：  = != > < >= <=  等于 不等于 大于 小于 大于等于 小于等于 |

练习：查询出工资大于等于6000的员工所有信息。

SELECT \* FROM emp WHERE salary>=6000;

练习：查询出所有女员工的信息；

SELECT \* FROM emp WHERE sex=0;

练习：查询出员工姓名为king的所有信息

SELECT \* FROM emp WHERE ename='king';

练习：查询出不在10号部门的员工的信息。

SELECT \* FROM emp WHERE deptId!=10;

练习：查询出1990年之前出生的员工信息

SELECT \* FROM emp WHERE birthday<'1990-1-1';

练习：查询出1993年之后出生的员工信息

SELECT \* FROM emp WHERE birthday>'1993-12-31';

练习：查询出没有明确部门的员工信息

#SELECT \* FROM emp WHERE deptId=NULL; 错误

SELECT \* FROM emp WHERE deptId IS NULL;

|  |
| --- |
| 注意: NULL不能和任何值进行等于/不等于判定，包括NULL自己。可以使用 IS NULL 和 IS NOT NULL |

练习：查询所有有明确部门的员工信息

SELECT \* FROM emp WHERE deptId IS NOT NULL;

练习：查询出10号部门所有女员工的信息

SELECT \* FROM emp WHERE deptId=10 AND sex=0;

|  |
| --- |
| AND(并且) OR(或者) |

练习：查询出10号和30号部门所有员工信息

SELECT \* FROM emp WHERE deptId=10 OR deptId=30;

练习：查询出工资在6000~8000所有员工信息

SELECT \* FROM emp WHERE salary>=6000 AND salary<=8000;

SELECT \* FROM emp WHERE salary BETWEEN 6000 AND 8000;

|  |
| --- |
| between ... and ... >= 第1个值 <= 第2个值  not between ... and ... |

练习：查询出不在6000~8000所有员工信息。

SELECT \* FROM emp WHERE salary<6000 OR salary>8000;

SELECT \* FROM emp WHERE salary NOT BETWEEN 6000 AND 8000;

练习：查询出1990年出生的员工信息。

SELECT \* FROM emp WHERE birthday>='1990-1-1' AND birthday<='1990-12-31';

SELECT \* FROM emp WHERE birthday BETWEEN '1990-1-1' AND '1990-12-31';

练习：查询10号和20号和30号部门的员工信息。

SELECT \* FROM emp WHERE deptId=10 OR deptId=20 OR deptId=30;

SELECT \* FROM emp WHERE deptId IN(10,20,30);

练习：查询出不在10号和20号部门的员工信息

SELECT \* FROM emp WHERE deptId NOT IN(10,20);

(8)模糊条件查询

示例：查询出姓名中含有字母e的员工信息

SELECT \* FROM emp WHERE ename LIKE '%e%';

练习：查询出姓名以e结尾的员工信息

SELECT \* FROM emp WHERE ename LIKE '%e';

练习：查询出姓名中倒数第2个字符为e的员工信息

SELECT \* FROM emp WHERE ename LIKE '%e\_';

|  |
| --- |
| % 任意多个字符 >=0  \_ 任意一个字符 =1  上述两个符号不能和=连用，必须使用LIKE |

(9)分页查询

分页显示：假如查询的结果集中有太多的记录，一次显示不完，可以分多页显示。

|  |  |  |
| --- | --- | --- |
| 页码 | 开始 | 结束 |
| 第1页 | 0 | 4 |
| 第2页 | 5 | 9 |
| 第3页 | 10 | 14 |
| 第4页 | 15 | 19 |

开始 = (页码-1)\*每页数量

SELECT \* FROM emp LIMIT start,count;

start 开始的值

count 查询的数量

示例：每页大小是5，查询第1页的数据

SELECT \* FROM emp LIMIT 0,5;

练习：每页大小是5，查询第2页的数据

SELECT \* FROM emp LIMIT 5,5;

练习：每页大小是5，查询第3页的数据

SELECT \* FROM emp LIMIT 10,5;

练习：每页大小是5，查询第4页的数据

SELECT \* FROM emp LIMIT 15,5;

**注意：**LIMIT 后边的两个数值不能添加引号。

综合：

SELECT \* FROM emp

WHERE ...

ORDER BY...

LIMIT...

练习：查询出所有的男员工工资最高的前三个人。

课后任务：

(1)复习今天内容

(2)删除所有的代码，保留注释，重新编写SQL语句。

#### 复习以及练习

复习：

软件过程：

软件定义期：可行性研究、需求分析

软件开发期：概要设计、详细设计、编码实现、测试

软件维护期：项目部署、项目维护

MySQL中常用的SQL语句：

show databases;

use 库名;

show tables;

desc 表名;

show warnings; #显示上一条语句产生的警告

show create table 表名; #显示创建表时所用的语句

-------------------------------------

drop database if exists 库名;

create database 库名 charset=utf8;

use 库名;

create table 表名( 列名 列类型 列约束 );

insert into 表名 values('值');

delete from 表名 where 条件;

update 表名 set 列名='值' where 条件;

select \* from 表名;

列类型：

数值类型：tinyint/smallint/int/bigint/decimal/bool

日期时间类型：date/time/datetime

字符串类型：char/varchar/text

列约束：

唯一/非空/**主键**/默认值/检查/外键

简单查询：

查询特定的列 select ename,sex from emp

给列取别名 select ename as n,sex s from emp

显示不同的值 select distinct sex from emp

执行运算 select salary\*12 from emp

单条件查询 select \* from emp where salary>6000

多条件查询 select \* from emp where deptId=10 and sex=0

模糊条件查询 select \* from emp where ename like '%E\_\_'

复杂查询：

1.MySQL中的简单查询 —— 查询结果的排序

示例：查询出所有员工信息，要求按工资由小到大排序

SELECT \* FROM emp

ORDER BY salary ; #默认是由小到大

#ORDER BY salary ASC; #ascendant，升序

示例：查询出所有员工信息，要求按工资由大到小排序

SELECT \* FROM emp

ORDER BY salary DESC ; #descendant，降序

练习：查询出所有员工信息，按照姓名由小到大排序

SELECT \* FROM emp

ORDER BY ename ;

练习：查询出所有员工信息，按照姓名由大到小排序

SELECT \* FROM emp

ORDER BY ename DESC;

练习：查询出所有员工信息，按照年龄由大到小排序

SELECT \* FROM emp

ORDER BY birthday ASC ; #生日由小到大排列

练习：查询10号部门中员工的所有信息，按工资由大到小排列

SELECT \* FROM emp

WHERE deptId=10

ORDER BY salary DESC ;

#注意：where子句和order by子句的顺序问题

练习：查询出所有员工信息，按工资由大到小排列；工资相同的，再按姓名由小到大排列

SELECT \* FROM emp

ORDER BY salary DESC, ename ASC ;

练习：查询所有员工信息，按生日由大到小排列——前提是：所有女员工必须排在所有男员工之前

SELECT \* FROM emp

ORDER BY sex ASC , birthday DESC ;

2.MySQL中的简单查询 —— 分页查询 —— 重点&面试重点

分页查询：用于分批次的显示大量的数据。语法：

SELECT .... FROM ... WHERE ... ORDER BY ....

LIMIT start, count ;

(1)start：是一个数字，表示从哪一行开始获取数据

(2)count：是一个数字，表示一次最多获取的行数，即页面大小

假设每页显示5行记录，则

第1页： LIMIT 0, 5

第2页： LIMIT 5, 5

第3页： LIMIT 10, 5

第4页： LIMIT 15, 5

第5页： LIMIT 20, 5

....

第n页： LIMIT (n-1)\*5, 5

练习：分页查询出员工信息，页面大小为5，显示第1页

SELECT \* FROM emp LIMIT 0, 5 ;

练习：分页查询出员工信息，页面大小为5，显示第2页

SELECT \* FROM emp LIMIT 5, 5 ;

练习：分页查询出员工信息，页面大小为5，显示第3页

SELECT \* FROM emp LIMIT 10, 5 ;

练习：分页查询出员工信息，页面大小为5，显示第4页

SELECT \* FROM emp LIMIT 15, 5 ;

3.MySQL复杂查询 —— 子查询

示例：查询出“研发部”所有员工的信息

步骤1：查询研发部的部门编号 => 10

SELECT did FROM dept WHERE dname='研发部';

步骤2：查询部门编号为10的员工信息

SELECT \* FROM emp WHERE deptId = 10;

综合上述两步为一条语句：

SELECT \* FROM emp WHERE deptId =( #父查询

SELECT did FROM dept WHERE dname='研发部' #子查询

) ;

练习：查询出工资比TOM高的所有员工信息

步骤1：查询TOM的工资 => 6000

SELECT salary FROM emp WHERE ename='TOM';

步骤2：查询工资大于6000的员工信息

SELECT \* FROM emp WHERE salary>6000;

综合上述两步：

SELECT \* FROM emp WHERE salary>(

SELECT salary FROM emp WHERE ename='TOM'

);

练习：查询出比TOM年长的所有员工信息

步骤1：查询TOM的生日 => 1990-1-1

SELECT birthday FROM emp WHERE ename='TOM';

步骤2：查询出生日小于1990-1-1的员工所有信息

SELECT \* FROM emp WHERE birthday<'1990-1-1';

综合上述两步：

SELECT \* FROM emp WHERE birthday<(

SELECT birthday FROM emp WHERE ename='TOM'

);

练习：查询出与TOM同年出生的所有员工信息

步骤1：查询TOM的生日中的年份 => 1990，提示：用year( )运算

SELECT YEAR(birthday) FROM emp WHERE ename='TOM';

步骤2：查询出生日中年份等于1990的员工所有信息

SELECT \* FROM emp WHERE YEAR(birthday)='1990';

综合上述两步：

SELECT \* FROM emp WHERE YEAR(birthday)=(

SELECT YEAR(birthday) FROM emp WHERE ename='TOM'

);

4.MySQL复杂查询 —— 聚合/分组查询

函数：Function，就是一个功能体，可以接收若干数据，加以处理，得到最终希望的结果——类似饺子机。

MySQL提供了几个聚合函数：

COUNT()：计算若干数据的个数

SUM()：计算若干数据的总和

AVG()：计算若干数据的平均值

MAX()：计算若干数据的最大值

MIN()：计算若干数据的最小值

示例：查询出员工的总数

SELECT COUNT(eid) FROM emp; #14

SELECT COUNT(deptId) FROM emp; #13，NULL不算数

SELECT COUNT(sex, deptId) FROM emp;

SELECT COUNT(ename, sex, deptId) FROM emp;

SELECT COUNT(\*) FROM emp; #推荐写法

示例：查询出所有员工工资的总和

SELECT SUM(salary) FROM emp;

练习：查询出所有员工工资的平均值（用两种方法）

SELECT SUM(salary) / COUNT(salary) FROM emp;

SELECT AVG(salary) FROM emp;

练习：查询出所有员工工资的最大值和最小值

SELECT MAX(salary), MIN(salary) FROM emp;

练习：查询出工资高于所有人平均工资的员工的数量 —— 难度系数：5星

步骤1：查询出所有员工的平均工资 => 6842

SELECT AVG(salary) FROM emp;

步骤2：查询出工资大于6842的员工的数量

SELECT COUNT(\*) FROM emp WHERE salary>6842;

综合上述两步：

SELECT COUNT(\*) FROM emp WHERE salary>(

SELECT AVG(salary) FROM emp

);

示例：查询出每个部门的编号及该部门员工的数量——先分组再聚合计算

SELECT deptId, COUNT(\*) FROM emp

GROUP BY deptId ; #按照部门编号分组

|  |
| --- |
| 注意：分组查询要么查询分组条件列，要么是其它列的聚合函数——不能直接查询其它列。 |

练习：按照性别分组，查询每组中工资的最大值、最小值、平均值

SELECT sex, MAX(salary), MIN(salary), AVG(salary)

FROM emp

GROUP BY sex;

5.MySQL复杂查询 —— 跨表/多表查询 —— 了解

查询结果集中的数据来自于多个不同的表。

示例：查询每个员工的姓名及其所在部门的名称

SELECT ename, dname

FROM emp, dept ; #笛卡尔积！错误！

-----------------------------------------------------------

SELECT ename, deptId, did, dname

FROM emp, dept

WHERE deptId = did ; #跨表查询必须有连接条件

注意：上述写法是SQL-92版本中的写法。存在一定的缺陷：

(1)无法显示没有部门的员工 —— KING

(2)无法显示没有员工的部门 —— 测试部

SQL-99标准对此进行了改变，但是SQL变得很复杂，有四种写法：

**(1)内连接查询：inner join ... on ...**

SELECT ename, dname

FROM emp INNER JOIN dept

ON deptId=did ;

#内连接的效果与SQL-92标准完全一样

**(2)左外连接查询： left outer join ... on ...**

SELECT ename, dname

FROM emp LEFT OUTER JOIN dept

ON deptId=did ; #13+1

#可以显示出“左侧表”中所有的记录！即使右侧表中没有对应的记录

**(3)右外连接查询：right outer join ... on ...**

SELECT ename, dname

FROM emp RIGHT OUTER JOIN dept

ON deptId=did ; #13+1

#可以显示出“右侧表”中所有的记录！即使左侧表中没有对应的记录

**(4)全外连接查询：full join ... on ...**

SELECT ename, dname

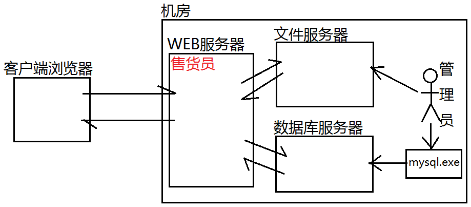
FROM emp FULL JOIN dept

ON deptId=did ; #13+1+1

#可以显示出“右侧表”和“右侧表”中所有的记录！—— MySQL不支持

|  |
| --- |
| 课外小知识：如何解决MySQL不支持全连接的问题 —— 使用结果集的合并  select ename, salary from emp\_us ;  select ename, salary from emp\_cn ; |
| 结果集的合并：UNION，把两条查询语句的结果合并为一个大结果  (select ename, salary from emp\_us)  UNION  (select ename, salary from emp\_cn) ;  两个结果集中的相同数据只显示一次 |
| (select ename, salary from emp\_us)  UNION ALL  (select ename, salary from emp\_cn) ;  两个结果集中的相同数据各自显示 |
| 练习：查询出员工姓名及其所在部门的名字，要求必须显示出所有的员工和所有的部门：左外连接合并上右外连接  (  SELECT ename, dname  FROM emp LEFT OUTER JOIN dept  ON deptId=did  )  UNION  (  SELECT ename, dname  FROM emp RIGHT OUTER JOIN dept  ON deptId=did  ) ; |

6.WEB服务器



WEB服务器：负责接收客户端请求，查找需要的数据/文件，给客户端以响应。Web服务器的分类：

**(1)静态WEB服务器**：提供的内容任何时间任何人访问都是一样的

常见的静态WEB内容：HTML/CSS/JS/图片/视频/Flash...

**(2)动态WEB服务器**：提供的内容不同时间不同人访问可能不同，一般都需要访问数据库、复杂运算、访问其它服务器。

常见的动态WEB技术：

JSP = HTML + JAVA

PHP = HTML + php

ASP.NET = HTML + C#

Node.js

Python

.....

### 2、JavaScript基础

#### DAY01 JavaScript介绍

复习

|  |
| --- |
| 标准SQL语句分类  **(1)DDL: Data Define Language 定义数据**  CREATE/DROP/ALTER(修改)  **(2)DML: Data Manipulate Language 操作数据**  INSERT/DELETE/UPDATE  **(3)DQL: Date Query Language 查询数据**  SELECT  **(4)DCL: Date Control Language 控制用户权限**  GRANT(授权)/REVOKE(收权) |

CREATE TABLE emp(

eid INT PRIMARY KEY AUTO\_INCREMENT

);

INSERT INTO emp VALUES(NULL....)

练习：

查询出所有男员工工资最高的前3个人。

SELECT \* FROM emp WHERE sex=1 ORDER BY salary DESC LIMIT 0,3;

1.复杂查询

(1)聚合查询/分组查询

示例：查询出所有员工的数量

SELECT COUNT(eid) FROM emp;

SELECT COUNT(\*) FROM emp; #推荐写法

练习：使用员工的姓名这一列查询员工数量

SELECT COUNT(ename) FROM emp;

练习：使用员工的部门编号这一列查询员工数量

SELECT COUNT(deptId) FROM emp;

练习：查询所有男员工的数量

SELECT COUNT(\*) FROM emp WHERE sex=1;

|  |
| --- |
| 聚合函数  函数就是一个功能体，提供数据，产出结果。——饺子机  COUNT(...)/SUM(...)/AVG(...)/MAX(...)/MIN(...) |

练习：查询出所有员工的工资总和是多少

SELECT SUM(salary) FROM emp;

练习：查询出所有员工的平均工资是多少

SELECT SUM(salary)/COUNT(\*) FROM emp;

SELECT AVG(salary) FROM emp;

练习：查询出工资最高的员工工资是多少

SELECT MAX(salary) FROM emp;

练习：查询出工资最低的员工工资是多少

SELECT MIN(salary) FROM emp;

练习：查询出年龄最大的员工

SELECT MIN(birthday) FROM emp;

|  |
| --- |
| 分组查询: 只能查询分组的条件和聚合函数。 |

示例：查询出每个部门的员工数量是多少。

SELECT deptId,COUNT(\*) FROM emp GROUP BY deptId;

练习：查询出男女的员工的平均工资，最高工资，最低工资是多少

SELECT sex,AVG(salary),MAX(salary),MIN(salary) FROM emp GROUP BY sex;

函数补充：

|  |
| --- |
| YEAR(...) 获取日期中的年份  MONTH(...) 获取日期中的月份  DAY(...) 获取日期中的日 |

示例：查询出1991年出生员工的所有信息

SELECT \* FROM emp WHERE YEAR(birthday)=1991;

练习：查询出3月份出生的员工所有信息

SELECT \* FROM emp WHERE MONTH(birthday)=3;

练习：查询出5号出生的员工所有信息

SELECT \* FROM emp WHERE DAY(birthday)=5;

(2)子查询

**把一个SQL语句的查询结果作为另外一个SQL语句的查询条件**

示例：查询出研发部所有的员工信息

步骤1：查询出研发部的部门编号——10

SELECT did FROM dept WHERE dname='研发部';

步骤2：根据研发部的部门编号10查询员工信息

SELECT \* FROM emp WHERE deptId=10;

综合：

SELECT \* FROM emp WHERE deptId=(

SELECT did FROM dept WHERE dname='研发部'

);

练习：查询出比tom工资高的员工的所有信息

步骤1：查询出tom的工资——6000

SELECT salary FROM emp WHERE ename='tom';

步骤2：查询出工资比6000高的员工所有信息

SELECT \* FROM emp WHERE salary>6000;

综合：

SELECT \* FROM emp WHERE salary>(

SELECT salary FROM emp WHERE ename='tom'

);

练习：查询出和tom同一年出生的员工信息

步骤1：查询出tom的出生的年份是多少——1990

SELECT YEAR(birthday) FROM emp

WHERE ename='tom';

步骤2：查询出出生年份为1990年的员工信息

SELECT \* FROM emp WHERE YEAR(birthday)=1990;

综合：

SELECT \* FROM emp WHERE YEAR(birthday)=(

SELECT YEAR(birthday) FROM emp

WHERE ename='tom'

);

(3)多表查询

示例：查询出所有员工的姓名及其部门名称。

SELECT ename,dname FROM emp,dept;

错误：产生笛卡尔积！

多表查询如何避免产生笛卡尔积，添加查询条件

SELECT ename,dname FROM emp,dept WHERE deptId=did;

|  |
| --- |
| 上述多表查询语法是SQL-92中，无法查询出没有部门的员工，也无法查询出没有员工的部门。  SQL-99中提出了新的多表查询方法。 |

**(1)内连接 INNER JOIN...ON...** 和SQL92结果一致

SELECT ename,dname FROM emp INNER JOIN dept ON deptId=did;

**(2)左外连接 LEFT OUTER JOIN...ON...**

SELECT ename,dname FROM emp LEFT OUTER JOIN dept ON deptId=did;

查询结果是左侧所有的记录都显示； OUTER可以省略

**(3)右外连接 RIGHT OUTER JOIN...ON...**

SELECT ename,dname FROM emp RIGHT OUTER JOIN dept ON deptId=did;

查询结果是右侧所有记录都显示；OUTER可以省略

**(4)全连接 FULL JOIN**

显示左侧和右侧所有记录——MySQL不支持

UNION 合并相同的项；

UNION ALL 不合并相同的项；

(SELECT ename FROM emp\_us)

UNION

(SELECT ename FROM emp\_cn);

(SELECT ename,dname FROM emp LEFT OUTER JOIN dept ON deptId=did)

**UNION**

(SELECT ename,dname FROM emp RIGHT OUTER JOIN dept ON deptId=did);

2.学习一门编程语言的基本步骤

(1)了解背景知识：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

(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)第三方库、框架

(10)实用的项目

有基础的学员

程序员必做50题

https://wenku.baidu.com/view/af66e2f14afe04a1b071de42.html

3.JS概述

(1)历史

1995年，JS最早出现在Netscape的浏览器中

2009nian ,JS遵循CommonJS规范，开始向服务器端发展。

(2)现状

既可以运行在客户端浏览器，也可以运行在服务器端。

(3)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

3.JS的执行环境

(1)浏览器自带的JS解释器

(2)NODEJS下的JS解释器

https://nodejs.org nodejs下载地址

在命令行下 node -v 查看系统中nodejs版本号

(3)执行JS代码

**浏览器：**

创建01.js和01.html

在01.html中引入01.js

|  |
| --- |
| <script src="01.js"</script> |

**NODEJS下：**

node C:/xampp/..../01.js

4.JS代码规范

(1)区分大小写

(2)每行代码结束的分号可加可不加，提倡都加

(3)分为单行注释(//...)和多行注释(/\*...\*/)

5.变量

变量就是用于存放数据的容器。

x=1 y=2

(1)声明变量

**var a=1;**

练习：声明多个变量，分别保存员工的编号，姓名，性别，生日，工资，部门。

(2)变量的命名规则

变量名称是以字母、数字、美元符号($)、下划线组成的；不能以数字开头。

多个连词之间的命名方法(下划线命名法、驼峰命名法)

user\_name userName

不能使用关键字和保留字作为变量名



(3)变量的注意

var a;

变量声明后未赋值，此时的值是undefined

使用未声明的变量会报错

(4)一次性声明多个变量

var a=1,b=2,c;

多个变量之间用逗号隔开。

3.常量

一旦声明不能重新赋值。

例如：春节的日期、中秋节的日期、圆周率...

关键字： **const** PI=3.14;

4.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

8进制，以0开头，例如012 -> 10

16进制，以0X开头，例如0XF -> 15

a~f 代表 10~15 不区分大小写

0XFF -> 255

课后任务：

(1)复习今天的内容，删除注释重新编写代码

(2)练习：

①使用常量保存圆周率，使用程序分别计算半径为5(变量)的30度/60度/90度的弧度是多长。

②使用变量分别保存商品的单价为12.5,数量为20；单价为30，数量为8.5；计算总和。

(3)预习JS的数据类型

#### DAY02 类型

复习

开发环境

浏览器 js

服务器 nodejs

变量声明

var $\_c1=3;

字母，数字，美元符号，下划线

var a; undefined

a=1;

var a,b=2,c=5;

常量声明

const PI=3.14;

数据类型

原始类型 和 引用类型

数值型、字符串型、布尔型、未定义型、空null

学习一门编程语言的基本步骤

(1)了解背景知识：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

(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)第三方库、框架

(10)实用的项目

1.数据类型

(1)数值型

分为整型和浮点型

8进制(012以0开头) 16进制(以0X开头) 10进制

1234.56 -> 1.23456\*10^3 -> 1.23456E3

**typeof(数据) 检测数据类型**

(2)字符串型

数据被引号包含就是字符串类型；不区分单双引号。

查看一个字符的Unicode编码

'a'.charCodeAt() //97

(3)布尔型

在程序中表示真或者假的结果

true/false

常用于一些是否的结果，是否登录，是否注册，是否是会员，是否在售

isLogin=true; isOnsale=false;

(4)未定义型

声明了变量未赋值，结果就是undefined

(5)空

用于释放(销毁)一个引用类型的地址，只有一个值null

2.数据类型转换

(1)隐式转换

①数字+字符串： 数字被转成字符串

1+'a' //'1a'

②数字+布尔型： 布尔型被转成数字 true->1 false->0

1+true //2

1+false //1

③布尔型+字符串： 布尔型转成了字符串

true+'hello' //'truehello'

**JS中加号(+)的作用**

执行加法运算

执行字符串的拼接

**使用 - \* /执行运算**

尝试将运算符的两端转成数值型，如果含有非数字则返回NaN(Not a Number)，不是一个数字。

NaN类型number

练习：查看以下程序的运行结果

var num1=3, num2=true, num3='tedu';

num1+num2+num3 //'4tedu'

num2+num3+num1 //'truetedu3'

num3+num1+num2 //'tedu3true'

(2)强制转换

①将任意类型转为整型

parseInt( )

parseInt('1.5a') //1

从开头查找数字，遇到非数字或者小数点；返回前边的数字;如果开头是非数字，返回NaN。

②将任意类型转为浮点型

parseFloat()

parseFloat('1.5a') //1.5

转换规则和parseInt类似，区别在于遇到小数点继续往后查找数字。

③将任意类型转为数值型

Number()

Number('1.5a') // NaN

如果要转换的数据中含有非数字，返回NaN

true -> 1 false->0

④数值型和布尔型转为字符串型

toString()

var num=10;

num.toString(); //'10'

num.toString(16); //a

如果要转换的数据是数值，可以设置进制(8,16,2)

3.运算符

由运算符连接操作的数据，所组成的形式就是表达式

(1)算术运算符

+ - \* / % ++ --

% 取余

++ 自增，在原来的基础之上加1

-- 自减，在原来的基础之上减1

console.log(num++) //先打印num的值，再执行自增

console.log(++num) //先执行自增，再打印num的值

(2)比较运算符

> < >= <= == != ===(全等于) !==(不全等于)

返回一个布尔型的结果

== 只是比较两个值是否相同

=== 不仅比较值，还会比较类型是否相同

3>'10' //false

数字和字符串比较，字符串转成数字。

'3'>'10' //true

比较首个字符的Unicode码，如果首个字符，则比较第二个字符

'3'->51 '1' -> 49

3>'10a' //false

3<'10a' //false

3=='10a' //false

'10a' -> NaN

NaN和任何数比较(> >= < <= ==) 都返回false

NaN==NaN 返回false

(3)逻辑运算符

&& 并且 || 或者 ! 非

返回一个布尔型的结果

&& 关联的两个条件都满足，结果是true，否则false

|| 关联的两个条件只需满足其一，结果是true，否则false

! 取反 !false -> true !true -> false

练习：声明两个变量保存用户名和密码，如果用户名是'root'，并且密码是'123456'，打印true，否则打印false；

练习:声明一个变量保存年龄，如果年龄大于90岁，或者年龄小于3岁，打印true，否则打印false

**逻辑短路**

&& 当第一个条件为false的时候，就不需要再执行第二个条件

|| 当第一个条件为true的时候，就不需要再执行第二个条件

练习：以下两个语句是否会报错。

|  |
| --- |
| var num=3;  num>5 && console.log(a);  num<1 || console.log(a); |

(4)位运算符(了解)

在执行运算符的时候，会把数字转成二进制进行运算

1 10 11 100 101 110 111 1000 1001 1010

2 4 8

110101=100000+10000+100+1

32 16 4 1

25 = 16+8+1

10000+1000+1

11001

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(^) 上下两位不同为1，相同为0

按位右移(>>) 删除二进制的最后一位，大概变小到原来一半

按位左移(<<) 在二进制的最后添加一个0，增加一倍

(5)赋值运算符

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

练习：声明变量保存商品的价格，打五折，计算价格。

(6)三目运算符

单目运算符：只需要一个数据或者表达式

a++ a-- !false

双目运算符：需要两个数据或者表达式

+ - \* / % > < >= <= == != === !==

&& || & | ^ >> << = += -= \*= /= %=

三目运算符：需要三个数据或者表达式

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；

如果条件表达式为false，执行表达式2；

练习：声明两个变量，分别保存两个数字；比较两个数字的大小，打印最大值的变量名。

练习：声明两个变量，分别保存用户名和密码；如果用户名是root，并且密码是123456，打印“登录成功”，否则打印“登录失败”

课后任务：

(1)复习今天内容，整理当天的思维导图

(2)练习：

声明一个变量保存年份，判断这个年份是否为闰年，如果是打印“是闰年”，否则打印“不是闰年”

闰年：4年一个闰年(能被4整除，和4取余为0)，并且不能被100整除，或者能被400整除。

(3)预习逻辑结构(if if-else..)

#### DAY03 if

复习

原始类型数据

数值型 整型和浮点型 3.14 3.14e2

字符串型

'1' 'undefined'

'a'.charCodeAt()

布尔型

true/false

未定义型

undefined

空

null

typeof(2) number/string/boolean/undefined/object

数据类型转换

隐式转换

数值 + 字符串 = 字符串

数值 + 布尔型 = 数值

布尔型 + 字符串 = 字符串

\* / - 将数据转为数值型 NaN

强制转换

整型 parseInt('a1a') true/false -> NaN

浮点型 parseFloat('a1.5a') true/false -> NaN

数值型 Number('1.5a') true/false -> 1/0

转字符串(数值和布尔型) 15.toString(2)

运算符

算术运算符 + - \* / % ++ --

比较运算符 > < >= <= == != === !==

逻辑运算符 或且非 || && ! 逻辑短路

位运算符 & | ^ >> <<

赋值运算符 = += -= \*= /= %=

三目运算符 条件表达式 ? 表达式1 : 表达式2

学习一门编程语言的基本步骤

(1)了解背景知识：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

(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)第三方库、框架

(10)实用的项目

练习：

声明一个变量保存年份，判断这个年份是否为闰年，如果是打印“是闰年”，否则打印“不是闰年”

闰年：4年一个闰年(能被4整除，和4取余为0)，并且不能被100整除，或者能被400整除。

4年一闰，100年不闰，400年再闰

1.浏览器端函数

alert() 弹出警示框(消息框)

prompt() 弹出提示框(输入框)，需要使用变量来接收输入的值；值的类型是字符串型。

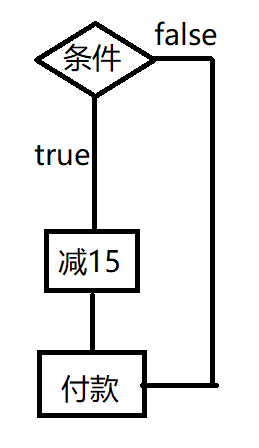
2.流程控制

程序 = 算法 + 数据

程序分为顺序执行、选择执行、循环执行

(1)if语句

满30减15



|  |
| --- |
| 语句1;  if(条件表达式){  语句2;  }  语句3; |

执行流程：

①执行语句1

②执行条件表达式，如果为true，执行语句2；false跳过

③执行语句3

注意：如果if后的大括号语句中只有一行，是可以省略大括号。

|  |
| --- |
| if(age>=18)  console.log('成年人'); |

在if语句的条件表达式中，有一些值默认就是false

0、NaN、''、undefined、null

(2)if-else语句

|  |
| --- |
| 语句1;  if(条件表达式){  语句2;  }else{  语句3;  } |

执行过程：

①执行语句1

②执行条件表达式，如果是true，执行语句2；如果是false，执行语句3

练习：使用弹出提示框分别输入商品的**价格**和**数量**，判断商品**总价**是否满500，如果满500打八折；使用变量保存当前**余额**为600，如果总价足够支付，则打印支付成功，否则打印余额不足。 05\_exercise.html 05\_exercise.js

(3)if-else的嵌套

用于判断多种情况

|  |
| --- |
| 语句0;  if(条件表达式1){  语句1;  }else if(条件表达式2){  语句2;  }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都是false  } |

执行流程：

①执行语句0

②执行条件表达式1

如果条件表达式1是true，执行语句1

如果条件表达式1是false，执行条件表达式2

如果条件表达式2是true，执行语句2

如果条件表达式2是false，执行条件表达式n

③以上所有的条件表达式为false，执行语句n+1

练习：声明变量保存订单的状态码（1,2,3,4,5...）

1-等待付款 2-等待发货 3-运输中 4-已签收 5-已取消

根据状态码打印对应内容；不存在的状态码打印‘非法的状态’

2.switch-case语句

是一种特殊的分支语句，可以根据一个表达式的不同值，来选择执行不同的程序。

|  |
| --- |
| 语句0;  switch(表达式){  case 1: //如果表达式的值是1  语句1;  break; //终止，不会再往后执行其它语句  ....  case n:  语句n;  break;  default:  语句n+1;  } |

**注意：**在case中表达式和值的比较使用的是全等于比较，要求值和类型都满足结果才是true。

**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句。

不同点：if-else可以判断相等或者不等的情况，使用范围更广；switch-case只使用全等的情况，结构上更为清晰，执行效率相对高。

3.循环执行

循环：就是一遍又一遍执行**相同或者相似**的代码。

循环的两个要素

循环的条件：重复的次数

循环体：重复执行的相同或者相似代码

(1)while循环

|  |
| --- |
| while(循环的条件){ //是一个布尔型的值  循环体  } |

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

使用while循环打印 11~19之间所有的整数

使用while循环打印 1~100之间所有的奇数(使用if判断是否为奇数)

使用while循环打印 20 22 24 26 28 30

使用while循环计算1~100之间所有的整数和

(3)预习do-while和for循环

《JavaScript高级程序设计》第三版 红宝石书

#### DAY04 switch for do循环

复习

弹出警示框 alert()

弹出提示框 var a=prompt()

程序执行：顺序执行，选择执行，循环执行

if(条件表达式){ 语句； }

if(条件表达式){ 语句1; }else{ 语句2; }

if(条件表达式1){ 语句1; }else if(条件表达式2){ }.....else{ }

var type='群众'

switch(type){ case '团员': ... break; ... default: }

循环执行

一遍又一遍执行相同或者相似的代码

循环条件

循环体

学习一门编程语言的基本步骤

(1)了解背景知识：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

(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)第三方库、框架

(10)实用的项目

1.break关键字

break可以结束任何形式的循环

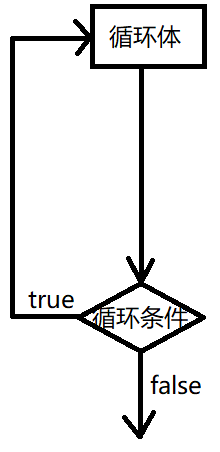
练习：使用无线循环形式来计算1~10所有数字相乘的结果。

练习：使用变量保存1个数字，循环弹出提示框的形式输入数字，如果数字猜大了，警示框提示'big'；如果数字猜小了，警示框提示'small'；否则提示'right'，结束循环。

03\_break.html 03\_break.js

说明： **''(空字符)在和数字比较的时候，转成0**

2.do-while循环



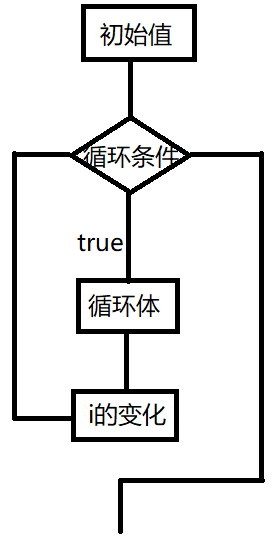
|  |
| --- |
| do{  循环体  }while(循环条件); |

不管循环条件是否为true，都会执行一次循环体。

练习：声明变量保存密码'123456'; 循环弹出提示框输入密码，如果输入正确，警示框弹出'login success'，结束循环。

05\_dowhile.js 05\_dowhile.html

3.for循环



1~10

|  |
| --- |
| for(初始值;循环条件;i的变化){  循环体  } |
| ①执行初始值  ②判断循环条件  ③如果循环体是true执行循环体，是false结束循环  ④如果执行了循环体，执行i的变化  ⑤重新执行第2步 |

4.break和continue

break: 结束循环，后续不会再执行其它的循环

continue: 跳过本次循环，继续下一次循环

练习：计算1~100之间所有偶数的和(遇到奇数跳过)

练习：打印1,2,5,7,10,11,13,14,17....98

打印1~100之间，去除所有能被3或4整除的数。

练习：计算新中国成立后一共有多少个闰年。

练习：打印本世纪的前10个闰年

练习：计算1~100之间的和，当和大于4000的时候，结束循环，打印总和以及当前i的值。

5.循环嵌套

while，do-while，for循环三者之间可以相互嵌套。

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

打印倒着的99乘法表

9\*9=81 8\*9=72.

8\*8=64

....

1\*1=1

(3)预习JS中自定义函数

#### DAY05 for循环

复习

while(循环条件){ 循环体 }

do{ 循环体 }while(循环条件)

for(初始值;循环条件;i的变化){ 循环体 }

break 和 continue

|  |
| --- |
| var i=0;  var sum=0;  do{  i++;  if(i%2==0){  continue;  }  if(i%5==0){  break;  }  sum+=i;  }while(i<10) |

循环嵌套

学习一门编程语言的基本步骤

(1)了解背景知识：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

(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)第三方库、框架

(10)实用的项目

学习目标

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(掌握)

1.函数

parseInt()/parseFloat()/typeof()...

分为系统函数和自定义函数

自定义函数

function：功能体，函数，可以接受若干个数据，返回处理的结果。用于封装反复执行的代码。—— 饺子机

(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  } |

调用

函数名称();

练习：使用函数封装10+20的结果，并打印出来，调用3次。

练习：使用函数封装计算1~100所有数字的和，打印结果，调用3次。

(2)创建带有参数的函数

|  |
| --- |
| function 函数名称(参数列表){ //形参->形式上的参数  函数体  } |

调用

函数名称(参数列表); //实参->实际的参数

**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，调用函数时的参数成为实参，调用的时候，实参的值会赋给形参。

形参本质上就是一个声明了的变量，但未赋值。

练习：创建函数getSum，传递1个参数，计算1~任意数字之间的和。

练习：创建函数getRun，传递2个参数，打印任意两个年份之间的所有闰年个数。

(3)带有返回值的函数

|  |
| --- |
| function 函数名称(参数列表){  函数体  return 返回值;  } |

调用

函数名称(参数列表)

return表示函数执行后，所返回的结果。

注意：

①如果没有return或者return后没有返回值，结果都是undefined。

②return可以返回任意类型的数据

③return后的所有代码都不会被执行

练习：创建函数add，传递3个参数，返回3个数字的和。

练习：创建函数getMax，传递2个参数，返回最大值。

练习：创建函数getMax，传递3个参数，返回最大值。

练习：创建函数isRun，传递1个参数(年份)，是闰年返回true，不是闰年返回false。

练习：创建函数getStatus，传递1个参数(订单状态码)，返回对应的中文；

1-待付款 2-待发货 3-运输中 4-已签收 5-已取消 其它状态码-无法查询

**对比return、break和continue**

return 返回结果，用于终止函数的执行，常用于函数中。

break 用于结束循环，结束switch语句；

continue 跳过当前循环，继续下一次循环。

2.变量的作用域

(1)作用域

变量或者函数的可访问范围

分为两种：

**全局作用域**：在全局作用域下声明的变量可以在任意位置访问到。

**函数(局部)作用域**：函数作用域下声明的变量只能在函数内部访问。

注意：在函数内部使用var关键字声明的变量是局部变量，不适用var关键字声明的变量是全局变量。

(2)变量声明提升

JS程序执行前，使用var关键字声明的变量会提升到**所在作用域**的最前边；但赋值还是在原来的位置。

|  |
| --- |
| console.log(a);  var a=1; //var a;会提升到前边; 而 a=1还是在当前位置; |

3.函数的作用域

函数和变量类似，也分为全局作用域和函数(局部)作用域

全局作用域：在全局作用域下创建的函数可以在任意位置调用

函数(局部)作用域：在函数(局部)作用域下创建的函数只能在函数内部调用

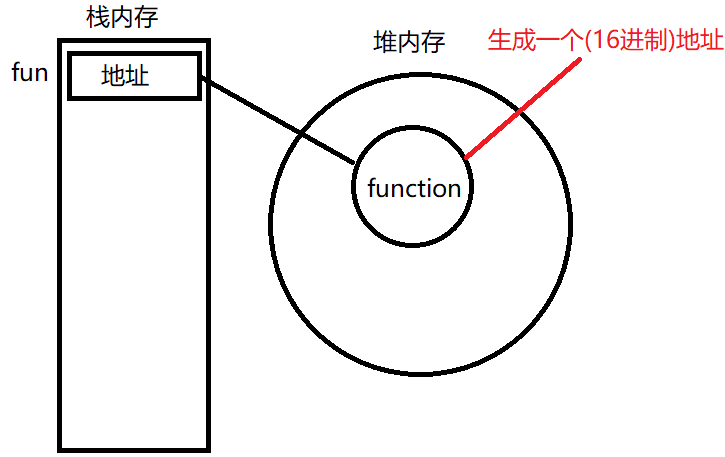
**函数声明提升**

和变量一样，JS在程序执行前，把使用function声明的函数提升到最前边。

|  |
| --- |
| function fun(){  return 1;  } |

函数调用 fun() 获取函数的返回值(return后的值)

函数名称 fun 保存的是函数在堆内存中的地址



4.递归的调用

在函数的内部调用自身

递归要有跳出条件，需要结合return使用。

斐波那契数列

1 1 2 3 5 8 13 21 34 55 89 144

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

创建函数，传递1个参数，使用递归计算斐波那契数列的第n项是多少

(3)预习JS的对象

#### DAY06 函数 对象

复习

学习一门编程语言的基本步骤

(1)了解背景知识：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

(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)第三方库、框架

(10)实用的项目

1.匿名函数

没有名字的函数 function(){ }

|  |
| --- |
| 创建函数——函数声明  function 函数名称(){} |

(1)创建函数——函数表达式

|  |
| --- |
| var 函数名称=function(形参列表){  函数体  return 返回值;  }  调用：函数名称(实参列表) |

**对比函数声明和函数表达式的区别**

函数声明存在函数提升，在任何的位置都可以调用

函数表达式不存在函数提升，必须先创建再调用

练习：使用函数表达式来计算1~100之间的和。

(2)匿名函数自调用

创建函数(局部)作用域，防止造成全局污染。

|  |
| --- |
| (function(){  函数体//就是局部作用域,创建的变量和函数都是不能被外部访问  })(); |

(3)作为回调函数

把匿名函数作为实参传递给形参，此时的形参就是函数名称

|  |
| --- |
| function fn(num){  //num就是函数的名称  num(); //调用传递的匿名函数  }  fn(function(){ ... }); |

练习：创建函数add，传递2个参数，2个参数都是以匿名函数的形式传递；在匿名函数中分别返回1个数字。

最后在add中计算两个数字相加。

2.系统函数(全局函数)

encodeURI 对一个url进行编码

decodeURI 对一个已经编码的url进行解码

parseInt 将数据转为整型

parseFloat 将数据转为浮点型

isNaN 检测一个数据是否为NaN 是->true 不是->false

isFinite 检测一个数据是否为有限值，是->true 不是->false 1/0 -> Infinity 无限值

eval 执行字符中的表达式 eval('1+2') -> 3

练习：使用弹出提示框输入一组运算，使用eval执行这组运算。 06\_eval.html 06\_eval.js

3.对象

是一种引用类型的数据，存储在堆内存中。

对象：是一组**属性**(property)和**方法/功能**(method)的集合

哪些是对象？

一个手机: 属性有颜色，品牌，尺寸，厚度... 功能有打电话、发短信、照相、聊天、游戏....

一辆汽车：属性有品牌、外观、车型... 功能有代步、撞人、拉货、取暖、乘凉...

(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来划分

自定义对象：自己创建的对象

(2)创建自定义对象

①对象字面量(直接量)

②内置构造函数

③自定义构造函数

4.使用对象字面量创建对象

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，如果出现特殊字符，必须添加引号

|  |
| --- |
| var phone={ color:'red', 'made-in':'china' }; |

练习：创建一个部门对象，属性有部门编号，名称，员工数量

练习：创建一个员工对象，属性有编号，姓名，性别，生日，工资，所在部门

5.使用内置构造函数创建对象

|  |
| --- |
| var book=new Object(); //创建一个空对象  book.id=103; //添加属性，属性名不能添加引号  book['titlte']='三国演义'; //添加属性，属性名必须加引号；如果不加引号，会被认为是变量. |

练习：创建一个汽车对象，属性有编号，品牌，颜色，价格

练习：创建一个电脑对象，属性有型号，品牌，尺寸，产地

6.访问对象中的属性

(1)获取属性值

|  |
| --- |
| emp.eid  emp['ename']  如果要获取的属性名不存在，返回undefined |

(2)遍历对象中的属性(for-in)

获取对象中每一个属性名，进而获取属性值

|  |
| --- |
| for(var key in emp){  //emp 要遍历的对象  //key 要遍历的每一个属性名  //emp[key] 通过属性名获取对应的属性值  } |

**注意：**for-in不能遍历预定义(JS默认为每一个对象添加的)属性。

练习：创建一个商品对象，包含编号，标题，价格，是否在售(isOnsale)，商品库存量(stockCount)，使用for-in遍历所有的属性。

7.检测对象中是否含有某个属性

'属性名' in 对象 // true->存在 false->不存在

对象.hasOwnProperty('属性名') //true->存在 false->不存在

对象.属性名=== undefined //true->不存在 false->存在

8.对象中的方法

|  |
| --- |
| var person={  name: 'tom',  say: function(){  this.name //this指代当前的对象  }  }  person.say(); //调用对象中的方法 |

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，添加属性半径，圆周率；添加方法计算周长(getLength)和计算面积(getArea)；分别返回圆的周长和面积

(3)预习JS中的数组

#### DAY07 数组

学习目标：

数组(重点)

1.什么是数组？

程序 = 数据 + 算法

数组也是一种数据类型，是数据的集合，可以包含其它任意的数据类型。

数组是通过所在位置的顺序号来访问每一个元素的，顺序号是从0开始，最大序号是 元素个数-1。

**2.定义一维数组**

(1)数组字面量

var arr = [ 元素1, 元素2…. ];

(2)构造函数 new Array()

var arr = new Array('tom','jerry'….)

创建数组，初始化数组元素

var arr = new Array(3)

创建数组，初始化大小为3，值为空.

练习：

创建一个数组country, 在数组中添加5个国家；

(3)数组元素的访问

数组名称[ 元素顺序号 ]

注意：元素顺序号是从0开始，最大的元素个数-1

3.对比undefined和null的区别

哪些情况出现undefined？

①创建了变量，没有赋值

②访问对象中，不存在的成员属性

③访问数组中，没有值的数组元素

null的作用？

用于释放一个变量；例如之前变量中存储了一个对象，或者存储了一个数组，都可以赋值为null来释放这个变量。

练习：

信息录入系统，通过提示框每次输入一个员工的姓名，输入完之后，打印出所有的员工；直到输入exit，退出提示框。

需要使用死循环，每次都执行弹出提示框；输入的内容，插入到数组中；当提示框中的值是exit，执行break终止循环。

while(true){ }

var names=[];

var str = prompt()

**4.获取数组的长度(个数)**

数组名称.length

**5.遍历数组元素**

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

练习：遍历一个数组，下标从大到小显示

①通过for循环遍历数组元素的下标可以实现遍历数组元素。

②通过for-in来遍历数组的属性来实现遍历数组元素，数组也是对象。

练习：

创建一个空数组(员工)，添加员工的编号，姓名，年龄，性别，电话，部门；最后通过for-in遍历数组

**数组的分类：**

数组分为索引数组(下标是顺序号)，关联数组(下标是自定义属性);

练习1：

声明函数，在函数中传递参数(实参是数组)，实现获取所传入数组中的最大值，并返回；

getMax([20,38,80,50,70]); -> 80

function getMax(arr){

var max=arr[0];

//比较

return max;

}

练习2:

定义函数，求数组中的最小值，并返回

练习3:

定义一个函数，查找数组中指定元素的位置，并范围

[26,38,72,97,85]

6.数组中的方法

toString( ) 返回数组转换的字符串

join([参数]) 返回字符串，参数是数组元素之间的分割符，默认是逗号(,)

concat(参数1,[参数2],[参数3]) 拼接多个数组，返回拼接后的数组，参数是所要拼接的数组

slice(参数1,[参数2]) 截取数组的元素，返回截取后的数组；参数1开始截取的位置，参数2结束的位置(为空，截取的最后)，(start ~ end-1)

splice(参数1,参数2,参数3…参数n) 删除数组的一部分；参数1是开始的位置，参数2是删除的长度；参数3以后是删除后，所要替换的新元素；

reverse() 翻转数组中的元素，会改变原来数组。

sort([参数]) 对数组进行排序，默认是按照Unicode码从小到大；参数是指定排序方式

数字[小-大] sort(function(a,b){ return a-b; });

数字[大-小] sort(function(a,b){ return b-a; });

push(参数) 往数组的最后添加元素，会改变原来数组。

pop() 从数组的最后删除一个元素，会改变原来数组。

unshift(参数) 往数组的最前边添加元素，会改变原来数组。

shift(参数) 从数组的第一个删除元素，会改变原来数组。

7.二维数组

二维数组就是数组的嵌套，数组中每一个元素也是数组。

课后练习：

创建省份信息大数组

每一个小数组都是一个省份的信息，第一项保存省份，后边的每一项是城市

最后遍历出整个数组

[

['山东',…..]

] 复习

匿名函数

function fn(){ } //函数声明

var fn=function(){ } //函数表达式

创建函数

匿名函数自调用

(function(a){ })(1);

回调函数

function add(num){ num(); }

add(function(){ var a=1; return a; });

系统函数

encodeURI/decodeURI/parseInt/parseFloat/isNaN/

isFinite/eval

对象

属性和方法的集合

字面量 内置构造函数 自定义构造函数

{name:'tom',age:18}

var person=new Object();

person.name='tom';

person['age']=18;

遍历对象属性

for(var key in person){ person[key] }

'name' in person

person.hasOwnProperty('age') //true false

person.name===undefined //false true

方法

var person={say:function(){},name:'tom',run:function(){} }

person.say();

学习一门编程语言的基本步骤

(1)了解背景知识：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

(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)第三方库、框架

(10)实用的项目

学习目标——数组

数组是由多个元素组成的集合。

每一个元素就代表一个数据。

1.创建数组

(1)数组字面量

[元素1,元素2...]

数组中可以存放任意类型的数据

(2)内置构造函数

|  |
| --- |
| new Array(5) //初始化数组的长度为5，可以继续添加第6个元素...  new Array(元素1,元素2....); |

访问数组中的元素

数组名称[下标]，下标是从0开始

练习：创建数组，初始化长度为5，添加5个课程。

练习：创建数组，直接添加5个员工的姓名。

2.获取数组元素的个数

数组名称.length

使用数组长度添加一个新的元素

数组[数组.length]=值

练习：创建一个空数组，使用数组长度添加3个国家名称。

3.数组的分类

数组分为**索引数组**和**关联数组**

索引数组：以整数作为下标

关联数组：以字符串作为下标

|  |
| --- |
| var arr=[]  arr['eid']=1;  arr['ename']='tom'; |

4.遍历数组中的元素

for循环，循环遍历数组元素的下标

|  |
| --- |
| var scrore=[85,79,93];  for(var i=0;i<score.length;i++){  i 代表数组元素的下标  score[i] 下标对应的元素  }  注意：只能遍历索引数组。 |

for-in

|  |
| --- |
| for(var key in arr){  key 数组的下标  arr[key] 下标对应的元素  }  注意：既可以遍历索引数组，也可以遍历关联数组。 |

练习：创建数组，保存所有员工的工资，使用for循环遍历计算平均工资。

练习：创建数组，保存4个汽车的品牌名称，把品牌名称为'宝马'改为'BMW';

使用循环来找到宝马的下标，使用下标改变元素的值为'BMW'

练习：创建数组，包含元素推荐、热点、娱乐、财经；分别使用for循环和for-in遍历该数组中的元素。

练习：创建一个函数getIndex，传递2个参数(数组，字符串)，返回字符串的下标，如果找不到该字符串的下标，则返回-1；

5.数组中的方法(API)

API-应用程序编程接口，预先定义好的函数/方法

toString() 将数组中的元素按逗号分隔成字符串

join('|') 将数组中的元素按照指定的字符分隔成字符串

concat(arr1,arr2...) 拼接两个或者更多的数组

slice(start, end) 截取数组中的元素，start开始的下标，end结尾的下标，不包含end本身；负数表示倒数。

splice(start, count,value1,value2....) 删除数组中的元素；start开始的下标，count删除的个数，value1,value...删除后补充的元素。

reverse() 翻转数组中的元素

sort() 对数组中的元素进行排序，默认按照Unicode由小到大

|  |
| --- |
| 对数字排序  sort(function(a,b){  return a-b; //由小到大  //return b-a; //由大到小  }) |

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

将数组中的元素进行翻转，不能使用reverse

['a','b','c','d'] -> ['d','c','b','a']

了解冒泡排序

(3)预习字符串下的方法(api)

#### DAY08 字符串操作

复习

创建

字面量 [ ]

内置构造函数 new Array('a','b'); new Array(5)

访问数组元素

arr.length

arr[arr.length]

下标 0 最后一个 arr.length-1

数组的分类

索引数组 下标是整数

关联数组 下标是字符串

遍历数组元素

for循环 遍历下标 0~arr.length-1

for-in

api、方法

转为字符串 toString() join('-')

拼接数组 arr1.concat(arr2,...)

截取数组元素 slice(start,end)

删除数组元素 splice(start,count,value1,value2...)

翻转数组元素 reverse()

数组元素排序 sort(function(a,b){ return b-a; })

学习一门编程语言的基本步骤

(1)了解背景知识：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

(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)第三方库、框架

(10)实用的项目

1.数组API(方法)

push() 往数组的末尾添加元素，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，返回数组的长度

shift() 删除数组开头的元素，返回删除的元素

2.二维数组

数组中的每一个元素也是数组

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

访问二维数组中的元素 arr[下标][下标]

3.字符串操作

包装对象：目的是让原始类型的数据也可以像引用类型一样，具有属性和方法。

JS提供了3种包装类型：String、Number、Boolean

将任意数据包装成字符串对象

new String( true ) 返回object类型

String(true) 返回字符串 'true'

(1)转义字符 —— \

转换字符的意义

\n 将字符n转义成换行

\t 将字符t转义成制表符(tab键)

\' 将引号转成普通的字符

...

练习：打印出现 welcome to chi\na

(2)字符串中常用方法

toUpperCase() 将英文字母转为大写

toLowerCase() 将英文字母转为小写

练习：初始化4个英文字母(有大小写)保存变量中，循环弹出提示框，输入4个字符(不区分大小写)，如果输入正确结束循环。 06\_code.html 06\_code.js

length 获取字符串的长度

charAt() 获取下标对应的字符

charCodeAt() 获取某个字符对应的Unicode码

练习：使用变量保存字符串“javascript”，获取字符“a”出现的个数。

indexOf(value, start) 查找某个字符串的下标，value是要查找的字符串，start开始查找的下标，默认是0，如果找不到返回-1

lastIndexOf(value) 查找某个字符串最后一次出现的下标，找不到返回-1

练习：声明变量保存字符串，检测该字符串是否为邮箱格式；如果是打印true，不是打印false；(查看是否含有@)

slice(start, end) 截取字符串，start开始的下标，end结束的下标，不包含end本身；如果end为空，截取到最后

substr(start, count) 截取字符串，start开始的下标，count截取的长度；如果count为空，截取到最后

substring() ?

练习：使用变量存储身份证号，获取其中年月日和性别。打印1997年05月20日 男

110230199705203310倒数第2位 偶数->女 奇数->男

练习：变量保存邮箱地址，分别截取邮箱的用户名和域名

jerry1995@sina.com.cn

练习：将一个英文单词首字母转大写，其余转小写

'hELlo' -> 'Hello'

split(sep) 按照指定的字符分隔为数组，sep是分隔符

练习：使用split获取邮箱的用户名和域名(jerry@qq.com)

练习：将一句英文所有单词的首字母转大写，其余字母转小写； 'hOW arE yOu' -> 'how are you'

['hOW', 'arE', 'yOu'] -> ['How', 'Are', 'You']

4.匹配模式(掌握)

作用：用于查找、替换字符串。

replace(value1, values) 查找并替换。value1要查找的字符串，value2要替换的字符串；value1可以使用字符串形式，也可以使用正则表达式形式 /china/ig

i -> ignore 忽略大小写

g -> global 全局查找

match(value) 用于查找匹配的字符串，返回一个数组

search(value) 用于查找满足条件的第一个字符的下标，如果找不到返回-1

5. Math对象

Math对象不需要使用new创建，可以直接使用。

PI 获取圆周率

abs() 取绝对值

floor() 向下取整

ceil() 向上取整

round() 四舍五入取整

max() 取一组数字的最大值

min() 取一组数字的最小值

pow(x,y) 取x的y次幂

random() 取随机 >=0 <1

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

创建数组，存储若干个员工的姓名，每次运行随机取出1个员工的姓名打印出来

创建数组，包含a-z, A-Z, 0-9元素；在数组中随机取4个字符，放到一个新数组中。 push

(3)预习Date对象

#### DAY09 Date Number Boolean对象 ES6

复习

push()/unshift()/pop()/shift()

a-z A-Z

[ [],[],[]... ] 数组[1][3]

字符串

包装对象 new String() var str='123';

方法：toUpperCase()/toLowerCase()/charAt()/indexOf()/

lastIndexOf()/slice(start,end)/substr(start,count)

/substring()/split()

匹配模式：查找，替换

replace(/china/ig,'中国')

match(/china/ig) 数组

search(/china/i) 返回下标

Math对象

PI/abs()/ceil()/floor()/round()/max()/min()/pow(x,y)/random()

学习一门编程语言的基本步骤

(1)了解背景知识：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

(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)第三方库、框架

(10)实用的项目

1.字符串方法

String.fromCharCode(index) 获取任意Unicode码对应的字符

substring(start,end) 截取字符串，start开始的下标，end结尾的下标，不包含end本身；如果两个参数是负数，自动转成0.

2.Date对象

用于对日期和时间进行存储和计算

(1)创建Date对象

new Date('2018/11/11 10:20:30')

new Date(2018,10,11,10,20,30) //第2个参数月份的范围0~11

new Date() 存储当前的系统时间

new Date(1000\*60\*60\*24) 存储的是距离计算机元年的毫秒数对应的日期时间

(2)获取Date对象的信息

getFullYear/getMonth/getDate/getHours/getMinutes

/getSeconds/getMilliseconds/getDay(星期0~6)

/getTime(距离计算机元年毫秒数)

(3)转为本地字符串

toLocaleString() // 年-月-日 时:分:秒

toLocaleDateString() //年-月-日

toLocaleTimeString() //时:分:秒

(4)设置Date对象的信息

setFullYear/setMonth/setDate/setHours/setMinutes

/setSeconds/setMilliseconds/

setTime 设置距离计算机元年毫秒数，一旦使用后，年月日时分秒都会受到影响

练习：创建对象保存员工的**入职时间**'2018-11-16'，3年后到期计算**到期时间**，合同到期前1个月续签合同，假如**续签时间**是周末，提前到周五。提前一周通知人事准备续签(**提醒时间**)

3.Number对象

new Number(值) 创建对象

将一个数据转为Number对象，本质还是数字。

Number.MAX\_VALUE 获取计算机能存储的最大值

Number.MIN\_VALUE 获取计算机能存储的最小值

toFixed(n) 保留小数点后n位

toString() 将数字转为字符串类型

数字+'' //隐式转为字符串类型

4.Boolean对象

new Boolean(值) 创建布尔对象，本质上将数据转为布尔型

Boolean(值) 转为布尔型

!!值 隐式将数据转为布尔型

toString() 将布尔型数据转为字符串

**null的作用**：用于释放(销毁)一个引用类型的数据。

5.错误处理

SyntaxError 语法错误，错误的使用了中文，缺少括号...程序不会执行。

ReferenceError 引用错误，使用未声明的变量

TypeError 类型错误，错误的使用了数据的类型，错误的使用括号

RangeError 范围错误，参数的使用超出了范围

new Array(-1)

引用错误、类型错误、范围错误出现后，会影响后续代码的执行。

**异常处理**

|  |
| --- |
| try{  尝试执行的代码，可能出现错误  }catch(err){  err：捕获的错误信息  处理错误的方案  } |

6.ES6新特征

**E**CMA**S**cript6

ES2017 ES2018...

(1)块级作用域——局部作用域

使用let关键字声明的变量，只能在块级作用域下访问，不能被外部访问。

块级作用域： { }、 for、while、do-while都是块级作用域

(2)箭头函数

是回调函数的另一种写法，和匿名函数不完全一致

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

如果箭头函数的函数体中只有一行代码，并且是以return形式，可以简化为

sort((a,b)=>a-b)

es6电子版

《ES6标准入门》第三版

http://es6.ruanyifeng.com

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

双色球：随机取1~33之间的6个数字不能重复，并且是从小到大排序，取1~16之间的1个数字（最后），组成一个新数组。

(3)预习服务器概念，NODEJS中的模块

### 3、Node.js基础

#### DAY01 全局对象 模块系统

学习目标：

(1)全局对象

(2)模块系统

1.全局对象

JS的全局对象 window

NodeJS的全局对象 global

**注意：**在nodejs交互模式下创建的变量或函数时全局的，而脚本模式下创建的属于局部的。

(1)console

console.time('LONG-LOOP');

检测的代码

console.timeEnd('LONG-LOOP');

作用：检测代码的运行时间。

练习：检测一组while的时间，对比for循环，哪个运行效率高。

(2)process

process.arch 查看CPU架构类型

process.platform 查看操作系统的类型

process.env 查看操作系统环境变量

process.cwd() 查看当前的工作路径

process.version/versions 查看当前NodeJS的版本信息

process.uptime() 查看nodejs运行时间

process.memoryUsage() 查看内存的使用情况

process.pid 获取当前进程的id

process.kill() 杀死某一个id对应的进程

(3)全局函数

定时器

①一次性定时器

setTimeout( func, delay ) 在一段时间后执行一次

func 要执行的任务，是一个回调函数，格式可以是箭头函数

delay 间隔的时间单位是毫秒

clearTimeout(timer) 清除一次性定时器

需要把开启的定时器放入到timer中。

②周期性定时器

setInterval(func, delay) 创建周期性定时器

func要执行的任务

delay间隔时间，单位毫秒

clearInterval(timer) 清除周期性定时器

需要把开启的定时器放入到timer中

练习：创建周期性定时器，打印10次'hello world'后，清除定时器。

③process.nextTick(()=>{ }) 开启定时器

④setImmediate(()=>{ });

clearImmediate(timer);

同步和异步

同步：程序按照顺序执行，先执行前边，后执行后边的。

异步：程序在执行的过程中，遇到异步函数等异步操作，会把这些操作放到主程序的最后边，主程序会跳过这个继续往后执行。

2.模块系统

|  |
| --- |
| 在nodejs中的文件中，任何一个js文件都是一个模块，在模块中都自动含有一个function  (function(exports,require,module,\_\_filename,\_\_dirname){  //\_\_filename 文件的完整路径和名称  //\_\_dirname 文件的完整路径  // require 用于引入其它的模块  //exports/module.exports 导出某些模块，是一个对象  **程序员自己编码的代码**  }) |

分为三种模块：核心模块、第三方模块、自定义模块；

(1)核心模块，直接引入 require('模块名称');

练习:

创建主模块09\_main.js

创建子模块09\_circle.js

在子模块中创建计算圆的周长和面积两个方法，并导出；

在主模块中引入子模块中的方法，并计算。

课后练习

创建主模块，引入子模块，调用方法计算周长和面积。

创建子模块，创建方法计算长方形的周长和面积，并导出

复习

学习一门编程语言的基本步骤

(1)了解背景知识：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

(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)第三方库、框架

(10)实用的项目

1.ES6新特征

(1)函数中的参数

ES6允许为形参设置默认值，如果没有传递实参，自动调用形参的默认值

(2)模板字符串

|  |
| --- |
| ` 在此之间可以写任意形式的代码 ${JS语法} ` |

练习：创建一个员工对象，包含姓名、性别、生日、工资；使用模板字符串打印员工的信息。

'姓名：\*\*，性别：\*\*，生日：\*\*，工资：\*\*'

2.nodejs概述

nodejs基于谷歌V8引擎(JS解释器)，运行在服务器端的语言，基于JS。

http://nodejs.cn 中文

http://nodejs.org 英文

**对比JS和NODEJS**

(1)JS运行在浏览器端，存在多个浏览器，容易产生兼容性的问题；而NODEJS在服务器端只有一个运行环境，不存在兼容性。

(2)两者都有内置(ES)对象、自定义对象、宿主对象(根据执行环境的不同)

(3)JS用于网页中的交互效果，而NODEJS用于服务器的数据库操作、文件操作...

**NODEJS的执行方式**

脚本模式 node c:/xampp/..../1.js

交互模式

node

两次ctrl+c 或者输入 .exit

3.全局对象

**NODEJS: global**

在交互模式下，声明的变量或者创建的函数都属于全局对下的，可以使用global访问，例如 var a=1; global.a

在文件中声明的变量或者创建的函数都属于是局部作用域下的，不能使用global来访问。

**JS: window**

在浏览器下，文件中声名的变量或者创建的函数都属于是全局作用域下的，可以使用全局对象访问；

例如: var a=1; window.a

(1)console对象

global.console.log() 打印消息

global.console.info() 打印消息

global.console.warn() 打印警告消息

global.console.error() 打印错误消息

global.console.time('自定义字符串') 开始计时

global.console.timeEnd('自定义字符串') 结束计时

自定义字符串前后要保持一致。

练习：使用计时查看for，while，dowhile循环10000的耗时。

(2)process对象

查看当前计算机的进程

process.arch 查看当前CPU架构 X64

process.platform 查看当前的操作系统 win32

process.env 查看当前计算机的环境变量

process.version 查看当前nodejs的版本号

process.pid 查看当前的进程编号

process.kill( ) 杀死某一个编号的进程

(3)Buffer对象

缓冲区：在内存中存储数据区域，存储网络传输时的资源

创建buffer

var buf=Buffer.alloc(5, 'abcde');

将buffer存储的数据转为普通字符

buf.toString()

(4)全局函数

parseInt/parseFloat/encodeURI/decodeURI/isNaN

isFinite/eval

①一次性定时器

|  |
| --- |
| 开启  var timer=setTimeout(回调函数，间隔的时间);  当间隔的时间到了，执行回调函数；单位是毫秒  清除  clearTimeout(timer); |

②周期性定时器

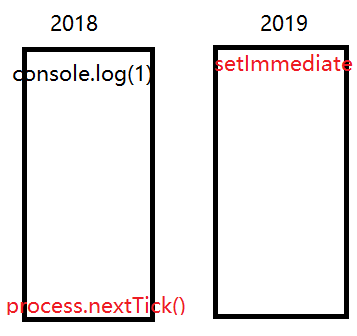
|  |
| --- |
| 开启  var timer=setInterval(回调函数，间隔的时候);  当间隔时间到了，执行回调函数；  清除  clearInterval(timer); |

练习：使用周期性定时器每隔3秒打印hello，打印三次后，清除定时器。

(3)立即执行(了解)

|  |
| --- |
| process.nextTick(回调函数)  在当前事件循环的结尾(2018年底)执行 |

|  |
| --- |
| var timer=setImmediate(回调函数)  clearImmediate(timer)  在下一个事件循环的开头(2019年初)执行 |



4.模块

模块可以理解为是一个功能体(积木块)

在NODEJS下模块分为自定义模块、核心模块(官方提供)、第三方模块

在NODEJS下，任意一个文件都是一个模块，文件中的代码默认是被一个构造函数所包含。

**以下代码中红色代码都是NODEJS自动为每一个文件添加的**

|  |
| --- |
| (function(exports,require,module,\_\_dirname,\_\_filename){  //程序员编码写的代码  }) |
| \_\_dirname 当前模块(文件)的完整路径  \_\_filename 当前模块(文件)的完整路径和文件名称  require() 引入一个模块  module 指代当前的模块对象  module.exports 当前模块的导出对象(公开)，可以供其它的模块使用的属性和方法  exports 等价于 module.exports |

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

创建两个模块 main.js(主模块), circle.js(功能模块)；在功能模块中创建两个函数，传递1个参数，分别获取圆的周长(getLength)和面积(getArea)，导出这两个函数；在主模块中引入功能模块，调用两个方法。

(3)预习 querystring、url、fs模块

#### DAY02 包和npm以及其他模块

复习

ES6新特征

函数的参数

function fn(a=1){ }

fn(2)

模板字符串

`${},' `

NODEJS

全局对象 global.fn() window

console log/info/warn/error/time('ab')/timeEnd('ab')

process arch/platform/env/version/pid/kill(1024)...

Buffer Buffer.alloc(5,'abcde')

setTimeout(回调函数, 间隔时间)/clearTimeout()

setInterval(回调函数, 间隔时间)/clearInterval()

process.nextTick(回调函数)

setImmediate(回调函数)/clearImmediate()

模块

自定义模块、核心模块、第三方模块

任意一个文件都是一个模块

exports require module.exports \_\_dirname \_\_filename

module.exports.a=1;

module.exports.fn=function(){ }

1.模块

|  |  |  |
| --- | --- | --- |
|  | 以路径开头 | 不以路径开头 |
| 文件模块 | require('./circle.js')  常用于用户自定义的模块，如果后缀名是.js的话，可以省略后缀名。 | require('querystring')  常用于引入官方提供的核心模块 |
| 目录模块 | require('./02\_2')  在02\_2目录下自动引入文件index.js；或者使用package.json文件声明main属性，来指定要引入的文件名称。 | require('04\_2')  要求引入的目录放在当前目录下node\_modules中。如果查找不到，则到上一级目录下查找，直到顶层目录。常用于第三方模块。 |

练习：创建模块03\_1.js，引入当前目录下的03\_2目录模块；在03\_2下创建test.js，导出一个函数fn(打印两个数字相加)，在03\_1.js中调用

练习：在05目录下创建模块05\_1.js，引入不带路径的目录模块05\_2，05\_2目录中含有hello.js文件(打印一句话)。

2.包和NPM

NPM: Node Package Manage

包(package): 就是一个目录模块，里边包含有多个文件，其中有一个文件命名为package.json的文件，是包说明文件。

自动下载 http://www.npmjs.com

**切换到下载的目录**

①cd 完整路径;

change directory

②在要下载的目录下，按住shift键，单击鼠标右键->在此处打开powershell窗口

**使用npm安装第三方包**

npm install 包的名称

3.核心模块

是NODEJS官方提供的模块，可以直接引入，不需要创建。

(1)查询字符串模块——querystring

浏览器向服务器发送请求，传递数据的一种方式

http://www.codeboy.com/product\_details.html?**lid=5&name=dell**

parse() 将查询字符串解析为对象

stringify() 将对象转换成查询字符串

练习：把百度搜索时的查询字符串解析为对象，获取关键词。

ie=utf-8&f=8&rsv\_bp=0&rsv\_idx=1&tn=baidu&wd=电脑

(2)URL模块

parse() 将url解析为对象

protocol 协议

hostname 主机(域名/ip地址)

port 端口

pathname 文件在服务器上的路径

query 查询字符串

format() 将对象转换成url

query属性对应的是对象

练习：浏览器请求的URL

https://www.tmooc.cn:3000/web/1810.html?sid=10&name=tom

获取URL中传递的sid和name的值

08\_exercise.js

(3)文件系统模块——fs

①fs.stat( path, callback )/fs.statSync(path) 查看文件的状态，通过回调函数来获取结果。

path 要查看的文件的路径

callback 回调函数，里边有两个参数

err 如果查看失败的错误信息

stats 文件的状态信息

isDirectory() 是否为目录

isFile() 是否为文件

**对比同步和异步的区别？**

同步会阻止后续代码的执行，只有方法执行完，才能继续执行后边的代码；是通过返回值来获取结果。

异步不会阻止后续代码的执行，把执行的结果放到整个程序的最后；是通过回调函数来获取结果。

②fs.mkdir(path, callback)/fs.mkdirSync(path) 创建目录

path 要创建的目录的路径

callback 回调函数，只有一个参数

err 如果创建失败的错误信息

③fs.rmdir(path,callback)/fs.rmdirSync(path) 删除目录

path 要删除的目录的路径

callback 回调函数，获取删除的结果

err 如果删除失败的错误信息

④fs.readdir(path,callback) 读取目录中的文件

callback

files 读取的文件，返回数组

⑤fs.writeFile(path, data, callback) 写入文件/创建文件

data 要写入的数据

如果文件不存在，则创建文件，然后写入；如果文件已经存在，则清空文件中的内容，然后写入。

课后任务

(1)复习今天内容，整理思维导图

(2)练习

创建目录mydir，在该目录下创建文件1.txt，写入1；创建文件2.txt，写入2; 读取mydir下所有的文件；删除mydir（自学删除文件unlink）。

整个过程全部使用同步方法。

(3)预习http协议

#### DAY03 Node服务器搭建

复习

模块分类

|  |  |  |
| --- | --- | --- |
|  | 以路径开头 | 不以路径开头 |
| 文件模块 | require('./circle.js')  自定义模块 | require('url')  核心模块 |
| 目录模块 | require('./mydir')  自动寻找index.js文件  package.json文件寻找main属性对应的文件 | require('mysql');  到node\_modules中寻找目录名称为mysql的目录 |

npm install 包名称;

querystring parse() stringify()

url parse() format()

protocol hostname port pathname query

fs fs.stat/fs.statSync fs.mkdir/mkdirSync

rmdir/rmdirSync readdir/readdirSync

readdir(path, callback) var res=readdir(path)

writeFile(path,data,callback)

1.文件系统模块——fs

①fs.unlink(path,callback)/fs.unlinkSync(path) 删除文件

②fs.existsSync(path) 判断文件是否存储

存在true 不存在false

练习：判断文件num.txt是否存在，如果不存在则创建，并初始化数字0；在原来的数字上加1。

③fs.readFile(path,callback)/fs.readFileSync(path) 读取文件

返回的数据是buffer形式。

④fs.appendFile(path,data,callback)

fs.appendFileSync(path,data)

追加写入，如果文件不存在则创建文件，如果文件已经存在，则在末尾写入数据。

练习：使用文件操作来创建文件user.txt，每次写入一个对象,

{uid:1, uname:'tom', upwd: '123456'}

2.http协议

是浏览器和web服务器之间的通信协议。

(1)通用头信息

Request URL: 请求的URL，要向服务端请求哪个文件。

Request Method: 请求的方法 get/post

Status Code: 响应的状态码

2\*\* 服务器成功的响应

3\*\* 响应的重定向，跳转到另一个网址

4\*\* 客户端错误

5\*\* 服务器端错误

Romote Address: 请求的远程服务器的IP地址和端口

(2)响应头信息

Connection: keep-alive；连接的方式：持续连接

Content-Type: text/html; 响应的文件类型

Content-Encoding: 响应的文件压缩形式

Transfer-Encoding:响应时的传输方式，chunked(分段传输)

Location: 响应时跳转的URL，通常结合着300系列状态码。

(3)请求头信息

Accept: 客户端接受的文件类型有哪些

Accept-Encoding: 客户端接受的文件压缩形式

Accept-Language: 客户端接受的语言类型

Connection: 客户端和服务器的连接方式，持续连接

(4)请求主体

可有可无，客户端向服务器端传递数据

3.http模块

可以模拟浏览器向服务器端发请求，也可以创建web服务器

(1)模拟浏览器

http.get(url, callback)

get 请求的方法

url 请求的网址

callback 回调函数，用来获取服务器端的响应

res 响应的对象

res.statusCode 获取响应的状态码

res.on('data', (buf)=>{ })

使用事件来获取服务器端响应的数据

buf是服务器端响应的数据，格式为buffer数据。

(2)创建web服务器

var server=http.createServer() 创建web服务器

server.listen(3000) 分配端口，监听3000端口的变化

server.on('request', (req,res)=>{ });

//接收浏览器的请求，是一个事件，一旦有请求，自动执行

req 请求的对象

url 请求的路径，显示端口后的部分

method 请求的方法，直接通过地址栏默认使用get方法

headers 请求的头信息

练习：创建web服务器，监听3001端口，使用事件监听浏览器的请求，打印请求的方法、URL；

http://127.0.0.1:3001/admin/login.html

http://localhost:3001/member/shopping.html

05\_server.js

res 响应的对象

write() 响应的内容为文本形式，向浏览器中写入文本。

writeHead( 302,{ } ) 设置响应的状态码和响应的头信息；如果要跳转需要设置 Location属性。

end() 响应结束

练习：创建web服务器，监听3000端口；接收浏览器端的请求； 06\_server.js

/login 响应文本 this is login page

/member 响应文本 this is member page

/ 跳转到 /member

如果以上都没有匹配的， 响应文本 404 not found

3.express框架

基于NODEJS，用于构建web服务器的框架

官网：www.expressjs.com.cn

安装：npm install express

|  |
| --- |
| const express=require('express');  var server=express();  server.listen(3000); |

(1)路由

浏览器向web服务器发来请求，web服务器要根据请求的方法和请求的URL来作出响应。

路由三要素：请求的方法、请求的URL、响应的内容

响应的对象(res)

res.send() 响应文本，只能响应一次send；如果是数字认为是状态码。

res.sendFile() 响应文件，必须使用绝对路径(\_\_dirname)

res.redirect() 响应的重定向

课后任务

(1)复习今天内容，整理思维导图

(2)练习

使用express创建web服务器，创建以下路由

get '/index' 发送文本"这是首页"

get '/login' 发送文件 "login.html"

post '/register' 发送文本"注册成功"

get '/' 跳转到 /index

(3)预习express的中间件，浏览器向服务器传递数据方式有哪些(post，查询字符串，路由传递)

#### DAY04 中间件MySQL模块

复习

fs模块

stat/mkdir/rmdir/readdir/writeFile/appendFile/readFile/

unlink/existsSync

http协议

通用头信息 请求头信息 响应头信息 请求的主体

http模块

http.get(url,(res)=>{

statusCode

res.on('data', (buf)=>{ })

})

web服务器

var server=http.createServer()

server.listen(3000)

server.on('request', (req,res)=>{  
 req url/method/headers

res write/writeHead(302,{ Location:'/detail' })/end()

})

express框架

创建web服务器

var server=express()

server.listen(3000);

创建路由

server.get('/login', (req,res)=>{

res.send()/sendFile(\_\_dirname+'/login.html')/redirect('')

});

1.路由中的请求对象

req.method 获取请求的方法

req.url 获取请求的URL

req.headers 获取请求的头信息

req.query 获取请求时以查询字符串形式传递的数据，返回格式为对象。

练习：创建文件03\_post.js，创建web服务器，新建路由(get /reg)，响应一个注册文件(reg.html)

2.post和get请求

get请求以查询字符串的形式传递数据，服务器端使用 req.query获取数据，结果是对象

post请求是通过表单提交(现阶段)的方式传递数据，服务器端通过事件形式获取数据(后期会有简单的方法)

|  |
| --- |
| req.on('data', (buf)=>{  获取的结果是buffer数据，需要使用查询字符串解析为对象  }) |

3.使用路由传递数据——路由传参

设置路由中接收的名称

|  |
| --- |
| server.get('/detail/:lid',(req,res)=>{  req.params //获取路由传递的数据，格式为对象  }) |

浏览器请求方式

http://127.0.0.1/detail/5

5就是传递的数据，使用lid来接收

练习：创建购物车的路由，请求的URL：/shopping，请求的方法get，传递商品的价格(price)和名称(pname)。

用户模块

用户列表 /list 详情 /detail 删除 /delete

/user/list /user/detail /user/delete

商品模块

商品列表 /list 详情 /detail 删除 /delete

/product/list /product/detail /product/delete

4.路由器

路由在使用过程中，不同的模块可能出现相同的URL，把同一个模块下的路由挂载到特定的前缀。

例如：商品模块下的路由挂载到/product，访问形式/product/list，用户模块下的路由挂载到/user，访问形式/user/list

路由器就是一个js文件，把同一模块下的路由放到一起。

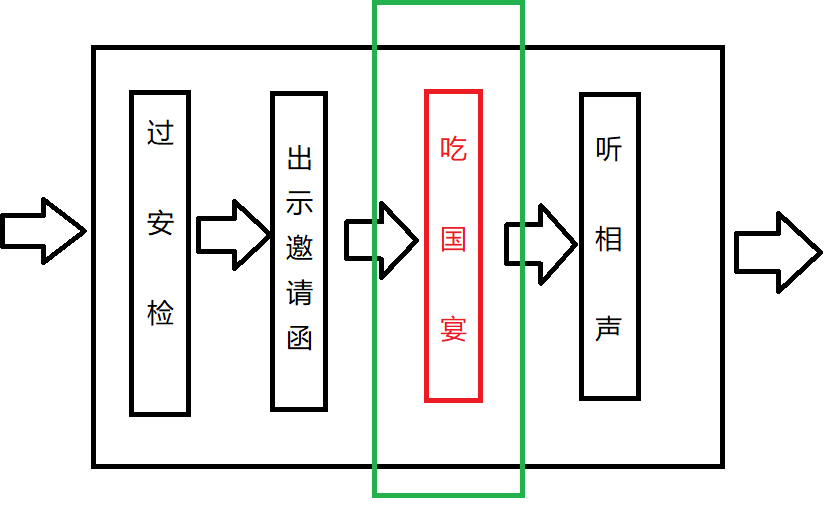
|  |
| --- |
| const express=require('express');  var router=express.Router(); //创建空的路由器对象  router.get('/list', (req,res)=>{ }); //往路由器中添加路由  module.exports=router; |

在web服务器下使用路由器

|  |
| --- |
| const userRouter=require('./user.js'); //引用路由器模块  server.use('/user', userRouter);//把路由器挂载到/user下，访问形式 /user/list |

练习：创建商品模块路由器(product.js)，添加路由商品列表(list)、商品删除(delete)、商品添加(add)，在web服务器引入，并挂载到/product

5.中间件



中间件的作用为主要的业务逻辑所服务。

**分为5个**

应用级中间件、路由级中间件、内置中间件、第三方中间件、错误级中间件

(1)应用级中间件

每一个中间件就是一个函数，需要配合其他的中间件或者路由使用。

server.use(回调函数) 拦截所有的路由

server.use('/detail', 回调函数) 拦截特定的路由

练习：创建路由(get，/view)响应当前的浏览次数，每次请求，响应的次数加1。

在函数外初始化一个变量，设置值为0; 在中间件中实现变量加1，在路由中响应变量。

(2)路由级中间件

用于在服务器中将路由器挂载到特定的URL

server.use('/user', userRouter);

(3)内置中间件

在express中只有一个内置的中间件

server.use( express.static('要托管的目录') )

托管静态资源到某一个目录，如果浏览器端要请求静态资源，则自动到这个目录下查找。

静态资源：html、css、客户端js、图像...

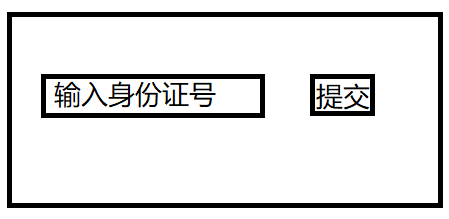
练习：将静态资源托管到files目录下，查看如果两个静态目录下有相同名称的文件，显示哪一个？

课后任务

(1)复习今天内容，复习MySQL中的SQL语句

(2)练习：

创建web服务器，托管静态文件（如下图），点击查询，服务器端获取输入的身份证号(中间件)，截取出生的年月日和性别；在路由中响应给浏览器。



(3)预习mysql模块的使用

https://www.npmjs.com/package/mysql

#### DAY05-DAY06项目练习

xz项目结构

node\_modules 项目所依赖的包

public 静态资源目录(html, css, img, 客户端js)

routes 项目中的路由器

app.js 项目入口文件

pool.js mysql连接池文件，在路由器中使用

练习：

用户检索，在 user\_query.html，输入框(用户编号)，点击提交，在客户端显示用户的这条数据。

select\*from xz\_user where uid=?;

课后练习：

创建修改用户信息(user\_update.html)，修改用户的user\_name,gender, phone, email四项，还需要提供用户的编号。检测输入的值，返回对应的状态码

复习

1.第三方中间件body-parser

将post请求的数据解析为对象

使用中间件

|  |
| --- |
| server.use(bodyParser.urlencoded({  extended:false  }))  urlencoded 可以将post请求的数据解析为对象  extended:false 不使用第三方中的qs模块，而是使用核心模块querystring将数据解析为对象 |

在路由中获取数据

req.body 返回一个数据的对象格式

**浏览器向服务器端传递数据的方式**

get req.query

post req.body

路由传参 req.params

2.MySQL模块

增 INSERT INTO emp VALUES(NULL,'tom'....)

删 DELETE FROM emp WHERE uid=5;

改 UPDATE emp SET uname='jerry',sex=1 WHERE uid=5;

查 SELECT \* FROM emp;

mysql.exe -h127.0.0.1 -P3306 -uroot -p

(1)普通连接mysql

var connection=mysql.createConnection({ }); 创建连接对象，参数中传递mysql服务器的主机、端口、用户名、密码、选择使用的数据库。

connection.connect(); 执行连接

connection.query(sql,callback); sql表示要执行的SQL语句，callback回调函数，用于获取SQL语句的执行结果。

connection.end(); 执行完所有SQL语句，关闭连接。

(2)使用连接池

var pool=mysql.createPool({ });创建连接池对象，需要提供服务器的主机、端口、用户名、密码、使用的数据库，以及设置连接池的大小 connectionLimit

pool.query(sql, callback); 执行SQL语句；sql是执行的SQL语句，callback获取SQL语句的执行结果。

练习：删除编号为5的员工的信息，查看结果；修改编号为7的员工的姓名为 '张三'，生日为'1980-5-1'，查看结果。

练习：创建web服务器，托管静态资源到public下，在该目录下创建add.html，点击提交按钮，向服务器端发送请求(/add，post)

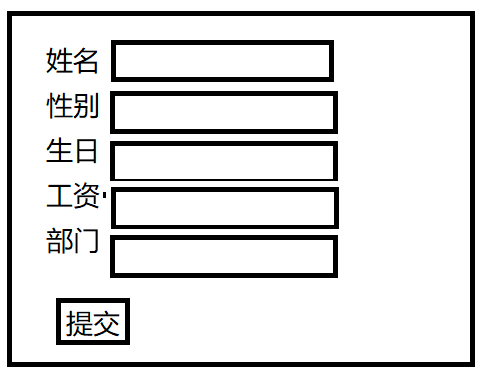
05\_add.js

课后任务

(1)复习今天内容，整理思维导图

(2)练习：

通过表单往数据库中插入员工信息 tedu下emp表



www.codeboy.com/pro