0 项目导入 day01

1.软件工程

IBM->DOS(BillGates)->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

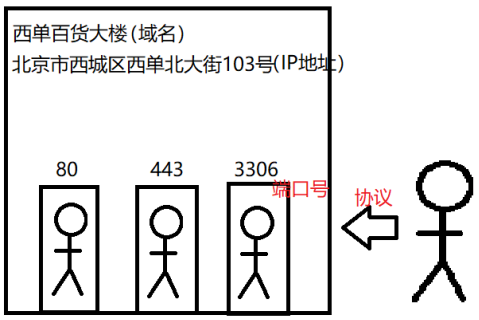
3.服务器 server：为客户端提供各种服务的功能强大的计算机。

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。

MySQL day01

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: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

练习：尝试使用错误的信息连接数据库服务器。

-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

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，查询所有记录。

课后任务：

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。

MySQL day02

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

(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：

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。

MySQL day03

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。

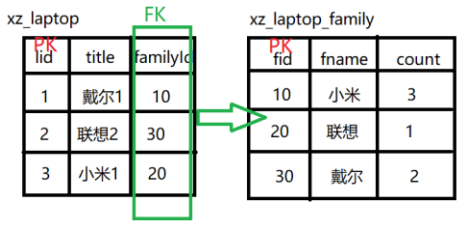
(6)外键约束——FOREIGN KEY

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 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

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：查询工资大于等于 6000 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：查询出所有男员工工资最高的前三个人。

JSDAY01

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;
```

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 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

a.解释型语言，编译一行执行一行。

b.弱类型语言 c.基于对象 d.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

标准关键字

break	case	catch	continue	default
delete	do	else	false	finally
for	function	if	in	instanceof
new	null	return	switch	this
throw	true	try	typeof	var
void	while	with	undefined	...

预保留的关键字

– class、int、float 等

(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：

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。

JSDAY02

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

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

a. 1 10 11 100 101 110 111 1000
2 4 8

b.110101=100000+10000+100+1
32 16 4 1

c. 25 = 16 + 8 + 1
11001 = 10000 + 1000 + 1

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：保存商品价格，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”

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。

JSDAY03

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。

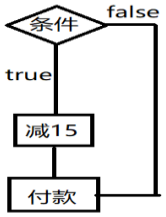
2.流程控制

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
}
```

练习：

使用 while 循环打印　11~19 之间所有整数

使用 while 循环打印　1~100 之间所有奇数(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

JSDAY04

1.break 关键字

break 可以结束任何形式的循环

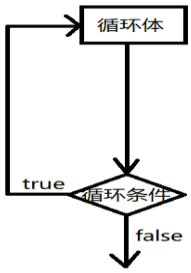
练习：使用无线循环形式来计算 1~10 所有数字相乘的结果。

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



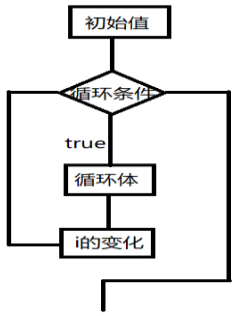

```
do{
   循环体
}while(循环条件);
```

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