web前端

# 项目导入

1.windows系统常用的快捷键

Alt+Tab: 切换窗口

Window+d: 显示/隐藏桌面

Window+r: 打开运行窗口

Window+e: 打开’资源管理器’

Alt+F4: 关闭当前的窗口

练习: 不使用鼠标情况下完成，进入c:/xampp/htdocs目录，创建demo.html文件，使用编辑器打开，输入内容，保存并使用浏览器运行改文件；删除改文件。

2.软件工程

程序设计

Ada – bug – debug

软件作坊

IBM – Dos – 管理磁盘 – Microsoft - Windows

软件工程

给定成本、进度的前提下，开发出具有有效性、可靠性、可维护、可重用、可追踪的软件产品。

3.软件的生命周期

软件过程/软件开发流程

**(1)软件定义期**

1)可行性研究阶段

技术、人力、设备、时间、资金、回报、政策、风俗…

2)需求分析阶段

功能性需求和非功能性需求

**(2)软件开发期**

3)概要设计阶段——架构师

子系统、功能模块、各自功能、模块间的接口

4)详细设计阶段——设计师/产品经理

页面、主体内容、属性、对象、方法

5)编码实现阶段

UI设计师: 设计效果图

前端工程师: 把效果图转换成html/css/js代码

后端工程师: 为前端页面提供所需要的数据

6)软件测试阶段——软件测试工程师

**(3)软件维护期**

7)项目部署阶段

将项目部署到服务器

8)项目维护阶段

4.学子商城功能性需求分析

前台子系统: www.codeboy.com

后台子系统: www.codeboy.com/admin/login.html

|  |
| --- |
| (1)前台: 用户模块、商品模块、购物车模块  (2)后台: 用户模块、商品模块、订单模块 (3)移动端: 用户模块、商品模块、购物车模块 |

5.web项目中的服务器

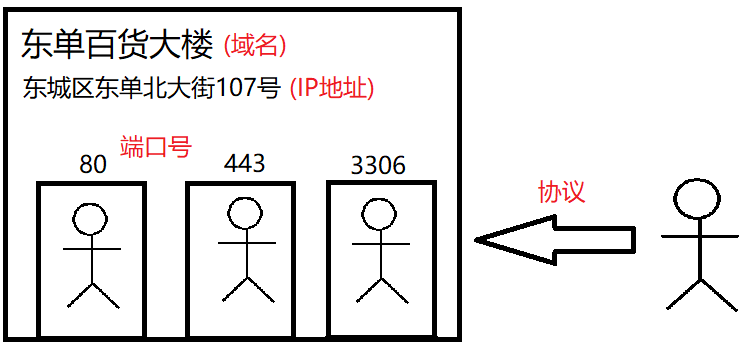
Server: 为客户端提供各种服务的功能强大的计算机

访问服务器

(1)服务器地址: 域名/IP地址

(2)对应服务的端口号

(3)提供该服务所使用的协议



https://www.jd.com:443

http://www.codeboy.com:80

http://101.96.128.94:80

6.访问自己电脑上的web服务器

启动web服务 webServer/start.cmd

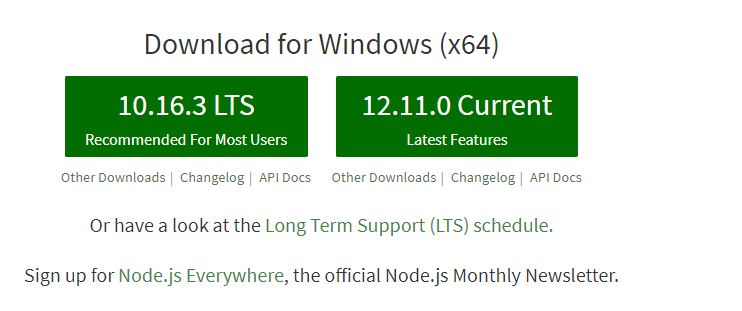
<http://127.0.0.1:80> 通过自己电脑ip访问

<http://localhost:80> 通过自己电脑的域名访问

练习: 在public目录下创建文件4.html，随便输入一些内容，分别使用域名和IP访问改文件

该服务器需要安装的软件——NodeJS

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



选择10.16.3LTS

查看是否安装

运行 -> cmd -> node -v

获取自己电脑在局域网的IP地址

在命令行下输入 ipconfig

查看 以太网适配器 以太网 IPv4地址

172.163.100.113

<http://172.163.100.113:80> 通过局域网IP地址访问服务器，所有局域网的计算机都可以访问

# MYSQL

## 服务器

### 1.数据的存储形式

内存/文件/第三方服务器/数据库服务器

### 2.什么是数据库

数据库就是按照一定的形式来组织、存储数据，目的为了对数据进行操作——增删改查

(1)数据库发展历史

网状数据库 - 层次型数据库 – 关系型数据库 – 非关系型数据库(NoSQL)

(2)关系型数据库逻辑结构

|  |
| --- |
| Server – Database – Table – Row-Column  服务器 - 数据库 – 表 - 行 – 列 |

### 3.MySQL数据库

Oracle分支: MySQL

Martin分支: MariaDB

Xampp

|  |
| --- |
| 服务器套装，包含有多个服务器，mysql、Apache…  https://www.apachefriends.org/ |

(1)mysql部署结构

**服务器:** 负责存储-维护数据 —— 银行机房

C:/xampp/mysql/bin/mysqld.exe 启动文件

确保端口3306不被占用

**客户端:** 负责向服务器发起增删改查——AMT机

C:/xampp/mysql/bin/mysql.exe 客户端工具

(2)使用客户端连接服务器端

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

-h host 主机

-P port 端口

-u user 用户

-p password 密码

mysql -uroot 简写形式

### 4.mysql管理命令

show databases; 显示服务器上当前所有的数据库

use 数据库名称; 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 表名称; 描述表中都有哪些列(表头)

quit; 退出服务器的连接

练习: 使用root以外的用户登录，查看数据库有哪些

练习: 分别查看test和phpmyadmin都有哪些表

练习: 查看phpmyadmin中以下表有哪些列

Pma\_\_favorite

Pma\_\_recent

Pma\_\_history

### 5.SQL语句

SQL: Structured Query Language，结构化查询语言，用于操作**关系型数据库**服务器，对数据执行增删改查。

SQL命令的两种执行方式

(1)交互模式：客户端输入一行回车，服务器端执行一行，适用于临时性的查看数据

(2)脚本模式：客户端把要执行的命令编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/../01.sql 回车

练习: 编写脚本文件02.sql，显示所有的数据库，进入数据库phpmyadmin，查看当前数据库所有的表，查看表pma\_\_recent中有哪些列；如果测试没问题，尝试把中间一行写错，看结果有哪些影响

SQL命令语法规范

(1)一行语句可以跨越多行，以英文的分号作为结尾

(2)假如某一条语句出现语法错误，则此条语句以及后边所有语句不再执行

(3)SQL命令不区分大小写，习惯上关键字大写，非关键字小写

(4)分为单行注释(#...)和多行注释(/\*….\*/)，注释的内容不会被服务器所执行

常用的SQL命令

(1)丢弃指定的数据库，如果存在的话

DROP DATABASE IF EXISTS jd;

(2)创建新的数据库

CREATE DATABASE jd;

(3)进入创建的数据库

USE jd;

(4)创建保存数据的表

CREATE TABLE student(

sid INT,

name VARCHAR(8),

sex VARCHAR(1),

score INT

);

(5)插入数据

INSERT INTO student VALUES('1','scott','m','87');

(6)查询数据

SELECT \* FROM student;

(7)删除数据

DELETE FROM student WHERE sid='3';

(8)修改数据

UPDATE student SET name='guo',score='89' WHERE sid='4';

练习: 编写脚本文件04\_tedu.sql，先丢弃再创建数据库tedu，进入该数据库。

练习: 在tedu数据库中创建保存员工数据的表emp，包含员工的编号eid、姓名name、地址addr、电话phone

练习: 往员工表emp中插入4条数据，查询结果

练习: 编写脚本文件05\_dangdang.sql，先丢弃再创建数据库dangdang，进入该数据库，创建保存图书数据的表book，包含编号bid，标题title；插入3数据，查询结果。

课后任务

(1)复习今天内容

(2)练习

编写脚本文件xz.sql，先丢弃再创建数据库xz，进入该数据库，创建保存用户数据的表user，包含uid，用户名uname，密码upwd，邮箱email，电话phone，性别sex，真实姓名userName，注册时间regTime，是否在线isOnline，插入4条件数据，删除1条，修改1条，查询结果。 '2019-10-20'

(3)预习mysql中的列类型 手册11章

### 1.计算机存储字符

#### (1)如何存储英文字符

ASCII: 总共有128个，对所有的英文字符进行了编码。

Latin-1: 总共有256，兼容ASCII码，同时对欧洲符号进行了编码，mysql默认使用这种编码。

#### (2)如何存储中文字符

GB2312: 对常用的6千多汉字进行了编码，兼容ASCII

GBK: 对2万多汉字进行了编码，兼容GB2312

BIG5: 台湾繁体字编码，兼容ASCII

Unicode: 对世界上主流国家常用的语言进行了编码，兼容ASCII，不兼容GB2312,GBK,BIG5等；具体分为UTF-8，UTF-16，UTF-32

#### (3)解决mysql存储中文乱码

脚本文件另存为的编码

客户端连接服务器端的编码(set names utf8)

服务器端在创建数据库时的编码(chaset=utf8)

练习:编写脚本文件01\_sohu.sql，先丢弃再创建数据库sohu，设置编码为utf8，进入数据库，创建保存新闻数据的表news，包含nid，标题title，发表时间ctime，内容content，作者author，来源origin；插入若干条数据，在交互模式下查询结果

### 2.mysql中的列类型

创建数据表的时候，指定的列可以存储的数据类型

CREATE TABLE t1( nid 列类型 );

#### (1)数值型 —— 引号可加可不加

TINYINT 微整型，占1个字节，范围-128~127

SMALLINT 小整型，占2个字节，范围-32768~32767

INT 整型，占4个字节，范围-2147483648~2147483647

BIGINT 大整型，占8个字节

FLOAT 单精度浮点型，占4个字节，最多3.4e38，可能产生计算误差

DOUBLE 双精度浮点型，占8个字节，范围比BIGINT大的多

DECIMAL(M,D) 定点小数，不会产生计算误差，M代表总的有效位数，D小数点后的有效位数

BOOL 布尔型，只有两个值TRUE/1、FALSE/0，TRUE和FALSE不能加引号；在存储的时候使用TINYINT类型，具体的值是1和0

#### (2)日期时间型 —— 必须加引号

DATE 日期型 '2019-10-31'

TIME 时间型 '14:26:30'

DATETIME 日期时间型 '2019-10-31 14:26:30'

#### **(3)字符串型** —— 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最大是65535

CHAR(M) 定长字符串，可能产生空间浪费，操作速度相对的快，M最大值是255，常用于存储手机号，身份证号等一些固定长度的字符串

TEXT(M) 大型变长字符串，M最大是2G

|  |  |  |
| --- | --- | --- |
|  | VARCHAR(5) | CHAR(5) |
| a | a\0 | a\0\0\0\0 |
| ab | ab\0 | ab\0\0\0 |
| 一二三 | 一二三\0 | 一二三\0\0 |
| abcde | abcde | abcde |

1TB GB MB KB BYTE BIT(位)

1BYTE=8BIT

CREATE TABLE t1(

id INT,

age TINYINT,

commentCount INT,

price DECIMAL(6,2), #9999.99

phone CHAR(11),

article VARCHAR(5000),

birthday DATE,

sex BOOL

);

练习: 编写脚本文件02\_xz.sql，先丢弃再创建数据库xz，设置编码UTF8，进入该数据库，创建保存笔记本数据的表laptop，包含lid，标题title，价格price，库存量stockCount，上架时间shelfTime，是否为首页推荐isIndex，插入若干条数据。

练习: 编写脚本文件03\_tedu.sql，先丢弃再创建数据库tedu，设置存储的编码为UTF8，进入该数据库，创建保存部门数据的表dept，包含did，部门名称dname，员工数量 empCount，插入以下数据

10 研发部 3

20 运营部 2

30 市场部 2

创建保存员工数据的表emp，包含eid，姓名ename，性别sex，生日birthday，工资salary，所属部门编号deptId；插入若干条数据

### 3.列约束

mysql可以对要插入的数据进行特定的验证，只有满足条件才允许插入到数据表中，否则被认为非法的插入

例如：一个人的性别只能是男或者女，一个人成绩0~100

CREATE TABLE t1( id 列类型 列约束 );

#### (1)主键约束——PRIMARY KEY

声明了主键约束的列上的值不能出现重复，一个表中只能有一个主键，通常加在编号列；表中查询的记录会按照主键从小到大的顺序排列——加快查找速度。

声明了主键约束后就不能在插入NULL值

|  |
| --- |
| NULL 表示空，在插入数据时，无法确定要保存的数据。例如: 无法确定员工的生日、工资都可以使用NULL |

练习: 在部门中添加主键约束，并插入测试

#### (2)非空约束 —— NOT NULL

声明了非空约束的列上不允许插入NULL值

课后任务

(1)复习今天内容，重新编写案例代码

(2)练习

编写脚本文件xz.sql，创建数据库xz，设置编码utf8，进入数据库，创建保存笔记本分类的表laptop\_family，包含fid，分类名称fname，数量laptopCount

10 联想 2 20 戴尔 2 30 小米 3

创建保存笔记本数据的表laptop，包含lid，title，price，规格spec，详情detail，上架时间shelfTime，是否在售isOnsale，所属分类编号familyId；插入若干条数据

(3)预习mysql中的列约束

(1)主键约束——PRIMARY KEY

(2)非空约束——NOT NULL

#### (3)唯一约束——UNIQUE

声明了唯一约束的列上不允许插入重复的值，允许插入NULL，甚至多个NULL

|  |
| --- |
| NULL和任何值比较都不等，包括两个NULL比较 |

练习: 在商品表的商品名称这一列添加唯一约束，并插入数据测试，如果也不允许空。

#### (4)检查约束——CHECK

检查约束可以对插入的数据进行自定义验证

CREATE TABLE student(

score TINYINT CHECK(score>=0 AND score<=100)

);

mysql不支持检查约束，会降低数据的插入速度

#### (5)默认值约束——DEFAULT

可以使用DEFAULT关键字声明默认值，有两种方式可以应用默认值

INSERT INTO laptop\_family VALUES(70,'神州',DEFAULT);

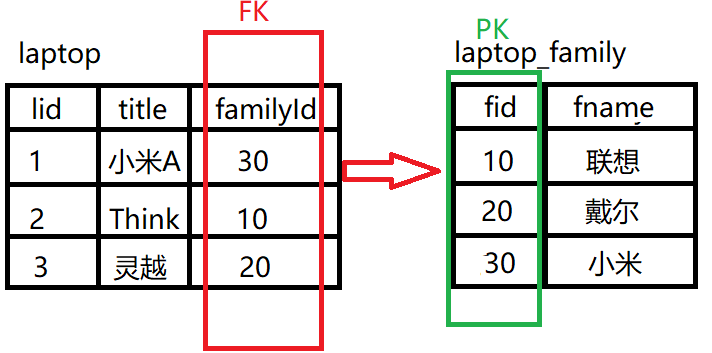
INSERT INTO laptop\_family(fid,fname) VALUES(80,'华为');

练习: 设置笔记本的默认价格为5199，分别应用两种默认值插入方式

#### (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列上出现过，两者的列类型要保持一致，允许插入NULL

FOREIGN KEY(外键列) REFERENCES 另表(主键列)



### 4.mysql中的自增列

AUTO\_INCREMENT: 自动增长，假如声明了自增列，无需手动赋值，直接赋值为NULL，会获取当前的最大值，然后加1

注意：

只适用于整数型的主键列上

自增允许手动赋值

练习: 编写脚本文件01\_tedu.sql，创建数据库tedu，进入该数据库，创建保存部门数据的表dept，包含did，dname，插入以下数据

10 研发部 20 市场部 30 运营部 40 测试部

创建保存员工数据的表emp，包含eid，ename姓名，sex，birthday生 日，salary工资，deptId所属部门编号；插入若干条数据

### 5.简单查询

#### (1)查询特定的列

示例: 查询所有员工的姓名、工资

SELECT ename,salary FROM emp;

练习: 查询出所有员工的编号、姓名、性别、生日

SELECT eid,ename,sex,birthday FROM emp;

#### (2)查询所有的列

SELECT \* FROM emp;

SELECT eid,ename,sex,birthday,salary,deptId FROM emp;

#### (3)给列起别名

示例: 查询所有员工的姓名和工资，使用汉字别名

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，使用汉字别名

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

练习: 查询所有员工的姓名和工资，使用一个字母作为别名

SELECT ename e,salary s FROM emp;

#### (4)显示不同的记录/合并相同的记录

示例: 查询出员工都分布在哪些部门

SELECT DISTINCT deptId FROM emp;

练习: 查询出都有哪些性别的员工

SELECT DISTINCT sex FROM emp;

#### (5)查询时执行计算

示例: 计算2+3-5\*6.4+3.6\*8.7

SELECT 2+3-5\*6.4+3.6\*8.7;

练习: 查询出所有员工的姓名及其年薪

SELECT ename,salary\*12 FROM emp;

练习: 假设每个员工的工资增加500，年终奖5000，查询出所有员工的姓名及其年薪，给列起汉字别名

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

#### (6)查询结果集排序

示例: 查询所有的部门，结果集按照部门编号升序排列

SELECT \* FROM dept ORDER BY did ASC;#ascendant

示例: 查询所有的部门，结果集按照部门编号降序排列

SELECT \* FROM dept ORDER BY did DESC;

describe 描述

descendant 降序

练习:查询所有的员工，结果集按照工资降序排列

SELECT \* FROM emp ORDER BY salary DESC;

练习: 查询所有的员工，结果集按照年龄从小到大排列

SELECT \* FROM emp ORDER BY birthday DESC;

练习: 查询所有的员工，结果集按照姓名升序排列

SELECT \* FROM emp ORDER BY ename;

练习: 查询所有的员工，结果集按照工资降序排列，如果工资相同按照姓名升序排列

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

练习: 查询所有的员工，结果集中女员工显示在前边，如果性别相同，按照年龄从大到小排列

SELECT \* FROM emp ORDER BY sex,birthday;

#### (7)条件查询

示例: 查询出编号为7的员工

SELECT \* FROM emp WHERE eid=7;

练习: 查询出姓名叫king的员工的编号，姓名，工资

SELECT eid,ename,salary FROM emp WHERE ename='king';

练习: 查询出所有的女员工

SELECT \* FROM emp WHERE sex=0;

练习: 查询出20号部门下的员工有哪些

SELECT \* FROM emp WHERE deptId=20;

练习: 查询出工资为5000以上的员工有哪些

SELECT \* FROM emp WHERE salary>5000;

|  |
| --- |
| 比较运算符: > < = >= <= !=(不等于) |

练习: 查询出1991-1-1后出生的员工有哪些

SELECT \* FROM emp WHERE birthday>'1991-1-1';

练习: 查询出不在10号部门的员工有哪些

SELECT \* FROM emp WHERE deptId!=10;

练习: 查询出没有明确部门的员工有哪些

SELECT \* FROM emp WHERE deptId IS NULL;

练习: 查询出有明确部门的员工有哪些

SELECT \* FROM emp WHERE deptId IS NOT NULL;

练习: 查询出工资在6000以上的男员工有哪些

SELECT \* FROM emp WHERE salary>6000 AND sex=1;

练习: 查询出工资在5000~8000之间所有员工

SELECT \* FROM emp WHERE salary>=5000 AND salary<=8000;

SELECT \* FROM emp WHERE salary BETWEEN 5000 AND 8000;

练习:查询出工资在5000以下和8000以上的所有员工

SELECT \* FROM emp WHERE salary<5000 OR salary>8000;

SELECT \* FROM emp WHERE salary NOT BETWEEN 5000 AND 8000;

练习: 查询出1990年之前和1993年之后出手的员工有哪些

SELECT \* FROM emp WHERE birthday<'1990-1-1' OR birthday>'1993-12-31';

练习: 查询出1993年出生的员工有哪些

SELECT \* FROM emp WHERE birthday>='1993-1-1' AND birthday<='1993-12-31';

SELECT \* FROM emp WHERE birthday BETWEEN '1993-1-1' AND '1993-12-31';

练习: 查询出20号部门和30号部门的员工有哪些

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

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

练习: 查询出不在20号部门或者30号部门的员工有哪些

SELECT \* FROM emp WHERE deptId NOT IN(20,30);

|  |
| --- |
| AND/OR  BETWEEN.. AND../NOT BETWEEN..AND..  IS NULL/IS NOT NULL  IN() / NOT IN() |

#### (8)模糊条件查询

示例: 查询出姓名中含有字母e的员工

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

练习: 查询出姓名中以e以为的员工

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

练习: 查询出姓名中倒数第二个字符为e的员工

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

|  |
| --- |
| % 可以匹配任意个字符 >=0  \_ 可以匹配任意1个字符 =1  以上两个匹配符必须结合LIKE关键字使用 |

#### (9)分页查询

假如查询的结果集有太多的数据，一次显示不完，可以使用分页显示

需要有两个条件: 当前的页码值、每页的数据量

SELECT \* FROM emp LIMIT start,count;

start: 开始查询的值，从哪一条开始查询

count: 每页的数据量

|  |
| --- |
| start = (当前的页码值-1)\*每页的数据量 |

**注意事项: start和count必须是数值型，不能加引号。**

假设每页显示5条数据

第1页: SELECT \* FROM emp LIMIT 0,5;

第2页: SELECT \* FROM emp LIMIT 5,5;

第3页: SELECT \* FROM emp LIMIT 10,5;

假设每页显示6条数据，查询前3页

第1页: SELECT \* FROM emp LIMIT 0,6;

第2页: SELECT \* FROM emp LIMIT 6,6;

第3页: SELECT \* FROM emp LIMIT 12,6;

课后任务

(1)复习今天内容，删除代码保留注释，重新编写SQL语句

(2)查询出工资在8000以上的女员工的姓名、性别、生日，工资，结果集按照工资降序排列，显示前两个人。

### 6.复杂查询

#### (1)聚合查询/分组查询

示例: 查询所有员工的数量

SELECT COUNT(eid) FROM emp; #推荐写法

SELECT COUNT(\*) FROM emp; #推荐写法

练习: 使用员工的部门编号这一列计算员工数量

SELECT COUNT(deptId) FROM emp; #14

练习: 查询所有女员工的数量

SELECT COUNT(eid) FROM emp WHERE sex=0;

|  |
| --- |
| 聚合函数  函数是一个功能体，提供若干个数据，得到结果——饺子机  COUNT()/SUM()/AVG()/MAX()/MIN() |

练习: 查询所有员工工资的总和

SELECT SUM(salary) FROM emp;

练习: 查询出男员工的平均工资

SELECT AVG(salary) FROM emp WHERE sex=1;

SELECT SUM(salary)/COUNT(eid) FROM emp WHERE sex=1;

练习: 查询出工资最高的员工

SELECT MAX(salary) FROM emp;

练习: 查询出工资最低的员工

SELECT MIN(salary) FROM emp;

练习: 查询出年龄最大的员工生日

SELECT MIN(birthday) FROM emp;

|  |
| --- |
| 分组查询: 只能查询分组条件和聚合函数 |

示例: 查询出每个部门的员工的平均工资

SELECT deptId,AVG(salary),ename FROM emp GROUP BY deptId;

练习: 查询出男女员工的数量，最高工资，最低工资

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

|  |
| --- |
| year() 获取日期中的年份 |

练习: 查询出1991年出生的员工有哪些

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

#### (2)子查询

|  |
| --- |
| 把一个SQL语句的查询结果作为另一个SQL语句的查询条件 |

示例: 查询研发部的所有员工

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: 查询出10号部门的员工

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工资高的员工有哪些

步骤1:查询出tom的工资是多少 ——6000

SELECT salary FROM emp WHERE ename='tom';

步骤2:查询出比tom工资高的

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同一年出生的员工有哪些

步骤1: 查询出tom出生的年份——1990

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

步骤2: 查询出1990年出生的员工

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/跨表查询

示例: 查询出所有的员工姓名及其部门名称

SELECT ename,dname FROM emp,dept;

#### 避免笛卡尔积

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

|  |
| --- |
| 以上查询的结果是SQL-92中的查询方法，无法查询中没有部门的员工，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语法 |

#### (1)内连接——和SQL-92查询结果一致

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

#### (2)左外连接——左侧表中所有记录都显示

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

OUTER关键字可以省略

#### (3)右外连接——右侧表中所有记录都显示

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

OUTER关键字可以省略

#### (4)全连接

显示左侧和右侧表中所有的记录——FULL JOIN

mysql不支持全连接

UNION 合并相同的记录

UNION ALL 不合并相同的记录

SELECT ename FROM emp\_cn

UNION

SELECT ename FROM emp\_us;

**把左外连接的结果和右外连接的结果合并**

(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);

# JS\_BASIC

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)常量和变量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用的小程序

(8)函数和对象

(9)第三方的框架、库

(10)实用的项目

程序员必做50题

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

《JavaScript高级程序设计》第三版

## 1.JS概述

### (1)历史

1995年，JS最早出现在Netscape的浏览器中

1996年，IE3中也出现了JS，称为JScript

1997年，JS提交了ECMA组织，指定了标准规范ECMAScript

2009年，出现了CommonJS规范，JS遵循这种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

跨平台性

### (4)应用场景

制作浏览器端的交互效果

创建web服务器，操作文件，访问数据库等

## 2.JS的开发环境

(1)浏览器中自带JS解释器，每个浏览器厂商都有自己的JS解释器

谷歌 IE 火狐 Safari 欧朋

(2)服务器端NodeJS解释器，基于谷歌的V8引擎JS解释器

https://nodejs.org nodejs下载地址

node -v 查看当前nodejs的版本号

(3)执行JS代码

**浏览器端**

创建01.js和01.html文件

在01.html中引入01.js

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

**服务器端**

node C:/xampp/.../01.js 回车

## 3.JS语法规范

(1)区分大小写

(2)每行代码结尾的分号可加可不加，建议都加

(3)分为单行注释(//...)和多行注释(/\* ... \*/)

## 4.变量

用于存储数据的容器

x=1

### (1)声明变量

var x=1;

使用关键字var声明变量，变量名称x，存储的值是1

### (2)变量的命名规则

变量名称可以使用字母、数字、美元符号、下划线，其中不能以数字开头

不能使用关键字和保留字作为变量名称

### (3)变量的注意事项

var a;

变量只是声明未赋值，则结果为undefined

可以为变量多次赋值，并且赋不同类型的值，这也是弱类型语言的特点

### (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

练习: 声明三个变量分别保存语文，数学，总成绩，其中总成绩未赋值；计算语文和数学相加，并把结果赋值给总成绩，最后打印三个变量。

## 5.常量

一旦声明不能重新赋值，声明后必须赋值

const pi=3.14;

课后任务

(1)复习今天内容，删除代码保留注释，重新编写代码，整理思维导图

(2)声明变量保存圆的半径，声明常量保存圆周率，声明两个变量分别保存计算的面积和周长。

(3)预习JS中的数据类型

## 6.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

### (1)数值型

分为整型和浮点型

八进制 以0开头的数字 例如 012 - 10

十六进制 以0X开头的数字 例如 0XF - 15

A~F 10~15 不区分大小写

浮点型分为定点小数和指数型小数

3.14e3

3.14

|  |
| --- |
| typeof 检测数据类型 |

### (2)字符串型

数据被引号所包含就是字符串型，不区分单双引号

查看任意一个字符的Unicode码

'a'.charCodeAt() -> 97

### (3)布尔型

只有两个值 true/false

在程序中表示真或者假

常用于一些是否的结果，例如是否登录、是否注册、是否在线、是否显示在首页推荐

### (4)未定义型

只有一个值 undefined

声明了变量未赋值结果为undefined

### (5)空

只有一个值 null

常和引用类型数据一起使用

## 7.数据类型转换

### (1)隐式转换

在运算的过程中，自动转换

1)数值型+字符串型 数值型转换字符串型

1+'2' - '12'

加号代表字符串拼接

2)数值型+布尔型 布尔型转换数值型

true - 1 false - 0

3+true - 4

3)字符串型+布尔型 布尔型转换字符串型

'5'+true - '5true'

|  |
| --- |
| JS中加号的作用  ①执行加法运算  ②执行字符串的拼接，前提是在加号的左右两端至少有一个数据为字符串  减、乘、除隐式转换  如果运算符的两端不是数字，则会自动调用Number转为数值型 |

练习: 查看以下程序的运行结果

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

console.log(num1+num2+num3);//'4tedu'

console.log(num2+num3+num1);//'truetedu3'

console.log(num3+num1+num2);//'tedu3true'

### (2)强制转换

1)强制转为数值型

Number(数据)

|  |
| --- |
| Number('2') //2  Number('2a') //NaN  Number(true) //1  Number(undefined) //NaN  Number(null) //0 |

NaN -> Not a Number 不是一个数字，NaN除了和字符串相加是拼接以外，其它运算都会返回NaN

2)将数据转为整型

parseInt()

|  |
| --- |
| parseInt(3.14) //3  parseInt('4a') //4  parseInt('a4') //NaN  parseInt(true) //NaN  parseInt(undefined) //NaN  parseInt(null) //NaN |

用于将小数和字符串转为整型，如果要转换的字符串以非数字开头，返回NaN

3)将数据转为浮点型

parseFloat()

和parseInt用法基本一样，只能将以数字开头的字符串转为浮点型。

4)将数值型和布尔型转为字符串型

toString()

var num=2;

num.toString() //'2'

## 8.运算符

表达式: 由运算符连接的操作数据，所组成的形式

运算符分为算术运算符、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)算术运算符

+ - \* / % ++ --

% 取余

++ 自增 在原来基础之上加1

-- 自减 在原来基础之上减1

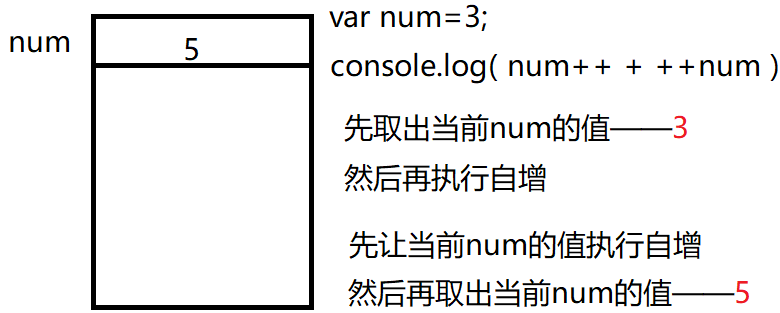
console.log(a++) 先打印a的值，然后执行自增

console.log(++a) 先执行自增，然后打印a的值

练习: 查看以下程序的执行结果

var num=3;

console.log( num++ + ++num );



### (2)比较运算符

> < >= <= ==(等于) != ===(全等于) !==

== 只是比较两个值是否相等

=== 不仅比较值，还会比较类型

3>'10' //字符串转为数值型

'3'>'10' //比较的是首字符的Unicode码

3->51 1->49

3>'10a' //false

3<'10a' //false

3=='10a' //false

NaN==NaN //false

NaN和任何值比较(> < >= <= == ===)都返回false

### (3)逻辑运算符

或者 || 关联的两个条件只要有一个为true，结果就是true，否则false

并且 && 关联的两个条件都是true，结果是true，否则false

非 ! 取反向

**逻辑短路**

&& 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

|  |
| --- |
| 注意事项：逻辑短路无需关注最终的结果是true，还是false；重点是看第二个条件有没有执行 |

练习: 查看以下程序是否会报错

var num=3;

num>5 && console.log(a);

num<1 || console.log(a);

### (4)位运算符(了解)

在执行计算的时候，计算机把数据转为二进制进行计算，计算完成后，再转回成十进制。

1 10 11 100 101 110 111 1000 1001 1010

2 4 8

10101=10000+100+1

16 + 4 + 1

100110=100000+100+10

32 + 4 + 2

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(^) 上下两位不同是1，相同是0

按位右移(>>) 删除二进制的最后一位，相当于在原来基础之上除以2，结果是小数的话再进行取整

按位左移(<<) 在二进制的最后补0，每次移动一位是原来的2倍

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，判断这个年份是否为软年，结合着逻辑短路，如果是闰年，打印'闰年'

能被4整除，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 if-else嵌套

1.运算符

### (5)赋值运算符

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

### (6)三目运算符

一目 一个运算符连接的一个操作数据 ++ -- ！

二目 一个运算符连接的两个操作数据 + - \* / =...

三目 两个运算符的连接的三个操作数据 ? :

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

## 9.浏览器端函数

alert() 弹出警示(消息)框

prompt() 弹出提示(输入)框，需要使用变量来保存输入的值，类型默认是字符串型；如果输入框中的内容为空，返回空字符串('')，如果点击取消，返回null

练习: 两次弹出提示框输入数字，计算任意两个数字相加的和，并把结果以警示框的形式弹出。

**程序**=数据+算法

程序执行分为顺序执行、选择执行、循环执行

## 10.流程控制——选择执行

### (1)if语句

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

满30减15

如果if后的语句只有一行，则大括号可以省略

|  |
| --- |
| 在条件表达式中，有一些值自动转为false  0 NaN '' undefined null |

练习: 声明变量保存个人签名内容，如果签名内容为空，设置签名内容为'这家伙很懒，什么也没留下'；打印签名内容。

### (2)if-else语句

|  |
| --- |
| if(条件表达式){  语句1;  }else{  语句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；假如购物卡内余额是800元，如果余额足以支付商品总价，警示框弹出'pay success'，否则警示框弹出'pay error'

07\_exercise.html 07\_exercise.js

### (3)if-else嵌套

|  |
| --- |
| if(条件表达式1){  语句1;  }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均为false  } |

(4)switch-case语句

是一种多项分支语句

|  |
| --- |
| switch(表达式){ //表达式通常是一个变量  case 值1:  语句;  break; //结束switch语句，不再往后执行  ...  default:  语句n; //如果表达式和case中的每个值比较均为false  } |

表达式在和case后的值比较的时候，使用的是全等于，要求值和类型都相等

**对比if-else嵌套和switch-case的区别**

相同点: 两者都可以用于多项分支语句

不同点: if-else嵌套既可以判断相等，也可以判断不等,适用范围更广泛；switch-case只能判断全等，结构上更为清晰，执行效率相对高

## 11.循环

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: 循环的次数

循环体: 重复的相同或者相似的代码

课后任务

(1)复习今天的内容，整理思维导图

(2)课后练习

使用switch-case语句来根据一个人的成绩进行评判

分数/10 取整

(3)预习循环语句 while do-while for 循环嵌套

### (1)while循环

|  |
| --- |
| while(循环条件){  循环体  } |

### (2)break

在循环中，break可以结束任何形式的循环

练习: 声明一个变量保存任意一个数字，无限循环弹出提示框并输入数字，如果输入的数字大于之前保存的数字，使用警示框提示'big'，如果输入的数字小于之前保存的数字，使用警示框提示'small'，否则警示框提示'right'，并结束循环。

03\_break.html

|  |
| --- |
| 检测一个值转为数值后是否为NaN  isNaN(数据)  是NaN -> true 不是-> false |

2.do-while循环

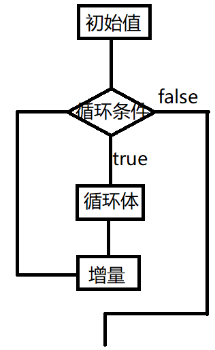
|  |
| --- |
| do{  循环体;  }while(循环条件); |

练习: 声明变量保存密码'123456'，无限循环弹出提示框并输入密码，如果输入正确结束循环。

04\_dowhile.html

### (3).for循环

|  |
| --- |
| for(初始值;循环条件;增量){  循环体  } |



### (4).break和continue

continue: 跳过后续的循环体，继续执行下一次循环

break: 结束循环，不会执行后续的循环

练习: 计算1~100之间所有偶数的和，如果遇到奇数就跳过

5.循环嵌套

在一个循环体中，还有其它的循环

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

课后任务

(1)复习今天内容，整理思维导图

(2)练习

完成九九乘法表

打印本世纪(2000~2100)前10个闰年

计算1~100之间所有整数的和，当和大于4000的时候，结束循环，并查看当前的和。

(3)预习JS中的自定义函数

## 12.函数

Number()/parseInt/parseFloat/isNaN...

函数分为系统函数和自定义函数

function: 功能体，函数(方法)，接收若干个数据，返回处理的结果——用于封装多次执行的代码

### (1)创建普通函数

|  |
| --- |
| function 函数名称(){  函数体; //封装的代码  } |

调用

函数名称()

练习: 创建函数，封装计算两个相加，调用多次

练习: 创建函数，封装计算1~100之间所有整数的和，并调用多次

### (2)创建带有参数的函数

|  |
| --- |
| function 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) //实际要传递的数据

参数: 创建函数时的参数称为形参，调用时的参数称为实参，实参会赋值给形参，多个参数用逗号隔开；实参的个数可以是0个或者多个，如果未赋值，形参的值为undefined

练习: 创建函数，计算1~任意一个数值之间所有整数的和，调用多次

练习: 计算任意两个年份之间的闰年个数，调用多次

### (3)创建带有返回值的函数

|  |
| --- |
| function 函数名称(参数列表){  函数体;  return 值; //返回值，函数调用后的返回结果  } |

调用

函数名称(参数列表)

返回值(return): 如果函数中没有return，或者return后没有任何的值，则返回undefined；return执行后，函数体后续的代码就不会再执行

练习: 创建函数，传递两个参数，返回任意两个数字的最大值

练习: 创建函数，传递三个参数，返回任意三个数字的最大值

练习: 创建函数，传递一个参数，根据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 其它-错误的状态码

|  |
| --- |
| 对比break和return  break用于结束switch语句和循环  return 用于结束函数，函数体后续的代码不再执行 |

练习: 创建函数，传递一个参数，查看任意一个年份是否为闰年，如果是返回true，不是返回false

练习: 创建函数，传递一个参数，计算1~任意数字之间所有整数阶乘的和。

## 13.变量的作用域

### (1)作用域

变量和函数可访问范围，分为两种

全局作用域: 在函数以外使用var关键字声明的变量，可以在任意位置访问到

函数作用域: 在函数内使用var关键字声明的变量，只能在函数内部访问

### (2)变量提升

JS程序在执行前，会将var声明的变量提升到所在作用域的最前边，但赋值还是在原来的位置

## 14.函数的作用域

函数的可访问范围，和变量的作用域一样，也分为全局函数和局部函数；全局函数可以在任意位置访问，局部函数只能在所在作用域访问

函数提升

和变量提升一样，JS程序执行前，会将function关键字创建的函数提升到所在作用域的最前边

## 15.递归

在函数内调用自身，本身是一个无限循环

用法：需要有结束条件，结合着return

斐波那契数列

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

第1,2项的值1，从第3项开始，每一项的值都是前两项相加的和

课后任务

(1)复习今天内容，整理思维导图

(2)练习

使用递归计算斐波那契数列的第n项

使用普通函数(循环)计算斐波那契数列的第n项

计算第10 30 50项的值

(3)预习**匿名函数**，对象

## 16.匿名函数

function (){ }

### (1)创建函数

|  |
| --- |
| 函数声明  function fn(){ } |
| 函数表达式  var fun=function(){ } |

**函数名称和函数名称()的区别**

fun/fn fun()/fn()

函数名称是一个变量，对应的是函数的结构

函数名称()是函数的调用，会执行函数体，得到返回值

**对比函数声明和函数表达式的区别**

函数声明创建的函数存在提升，可以在所在作用域任意位置调用

函数表达式创建的函数只存在变量的提升，必须先创建再调用

练习: 使用函数表达式创建函数，计算任意两个数字之间所有整数的和，并返回结果

### (2)匿名函数自调用

目的: 创建函数作用域，防止污染全局

|  |
| --- |
| (function(形参列表){  //函数体，封装的代码不会污染全局  })(实参列表); |

### (3)回调函数

将匿名函数以实参的形式传递

|  |
| --- |
| function fn(a){  //调用fn的时候，匿名函数会赋给形参a，a就是函数名称；a()就是调用传递的匿名函数  }  fn( function(){ ... } ) |

## 17.全局函数

Number()/parseInt()/parseFloat()/isNaN()

isFinite() 检测一个值是否为有限值 true->有限值 false->无限值

1/0 -> Infinity 无穷，0做除数返回无限值

encodeURI() 编码一个字符串为URI

decodeURI() 解码URI为一个字符串

编码和解码主要是针对于汉字和部分英文字符

eval() 执行字符串中的表达式

练习: 弹出提示框，输入一组字符串表达式，使用eval来执行字符串中的表达式 05\_global.html

## 18.对象

属于引用类型数据

对象: 是一组属性(property)和方法(method)的集合

一个人: 属性有身高，体重，姓名...方法有唱歌，敲代码...

一部手机: 属性有品牌，尺寸，颜色...方法有看视频，玩游戏，办公...

万物皆对象

### (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划分，分服务器端和浏览器端

自定义对象: 用户创建的对象

### (2)自定义对象

对象字面量

内置构造函数

自定义构造函数

### (3)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，如果含有特殊字符比价添加

练习: 创建一个员工对象，包括编号，姓名，性别，生日，工资，所属部门编号

练习: 创建一个手机对象，包含颜色，品牌，尺寸，内存大小

### (4)访问对象中的属性

对象.属性名

对象['属性名']

如果属性不存在，返回undefined

练习:创建图书的对象，包含编号，标题，作者，价格，出版时间；修改图书的价格，打印图书的作者，添加图书的出版社

### (5)使用内置构造函数创建对象

new Object() 创建一个空对象

需要访问对象中属性，并添加每一项

练习: 创建一个汽车对象，包含品牌，型号，颜色，长度，宽度

### (6)遍历对象中的属性

依次访问对象中的每个属性，是一个循环

|  |
| --- |
| for(var key in 对象){  key 代表对象中每个属性名  对象[key] 属性名对应的属性值  } |

练习: 创建对象，包含有若干个成绩，遍历对象中的属性，计算总分

### (7)对象中的方法

|  |
| --- |
| var person={  name:'tom',  say: function(){  this.name //this 默认指代当前所在的对象  }  }  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

创建一个计算器对象，包含有加减乘除4个方法，每个方法中都需要传递两个数字，返回结果；最后调用这四个方法

(3)预习数组

### (8)检测属性是否存在

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

'属性名' in 对象 true -> 存在 false -> 不存在

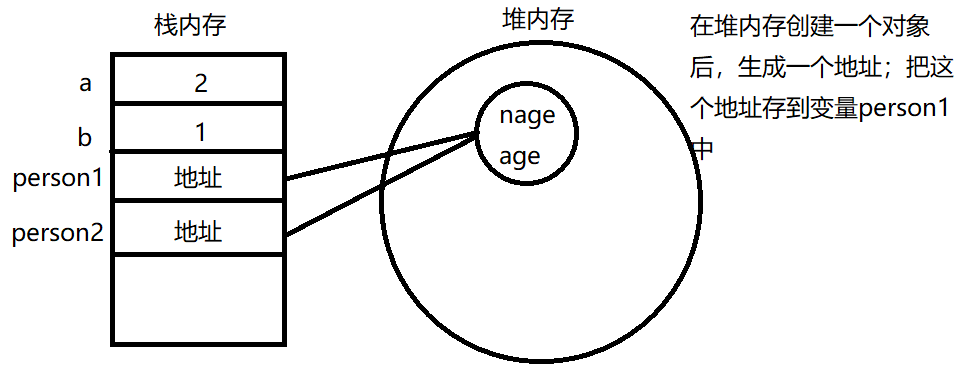
对象.hasOwnProperty('属性名') true->存在 false->不存在

练习: 检测对象中是否含有price属性，如果有，在原来基础之上加1000；检测对象中是否含有pic属性，如果没有，添加该属性并赋值为 'default.png'

### (9)引用类型的存储

引用类型数据存储在堆内存中，并生成一个地址，把这个地址存储在栈内存中

null用于设置变量不再指向任何的引用类型数据，也就相当于释放一个对象



数组就是一组数据的集合

## 19.数组

### 19.1创建数组

#### (1)数组字面量

[元素1,元素2...]

练习: 创建数组，包含有多个笔记本的品牌名称

练习: 创建数组，包含有多个城市名称

#### (2)访问数组中的元素

数组[下标] 下标是0开始

练习: 创建数组，添加一组汽车的品牌名称，修改第2个元素，在数组的末尾添加两个元素

#### (3)内置构造函数

new Array(元素1,元素2...)

new Array(5) 初始化数组的元素个数为5，需要给每个元素单独赋值

#### (4)数组的长度

数组.length 获取数组元素的个数

作用: 在数组的末尾添加新的元素

数组[ 数组.length ] = 值

练习: 创建一个空数组，使用数组长度属性添加星期日~星期六这些数据

#### (5)数组的分类

索引数组: 以0及以上的整数为下标

关联数组: 以字符串为下标，只能单独添加每个元素

### 19.2.遍历数组

#### (1)for-in

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

#### (2)循环

|  |
| --- |
| for(var i=0;i<数组.length;i++){  i 下标  数组[i] 下标对应的元素  }  只能遍历索引数组 |

练习: 声明变量保存多个数字，计算总和总和总和

练习: 创建数组，包含有多个姓名，把姓名为tom的元素，改为'汤姆'

练习: 创建数组，包含有一组数字，获取数组中的最大值

练习: 创建函数，传递一个参数(一组工资)，返回平均工资

练习: 创建函数，传递两个参数(数组，要查找的值)，如果在数组中查找到了这个值，返回该元素的下标；如果找不到返回-1

### 19.3.数组API

API 应用程序编程接口，预定义好的一些函数或者方法

toString() 将数组元素转为字符串，只能按照逗号分隔元素

join('|') 将数组元素转为字符串，可以按照指定的字符分隔元素,不修改原素组，需要重新定义变量接收

concat(arr1,arr2..) 拼接多个数组，返回一个新的数组，不修改原素组，需要重新定义变量接收

slice(start,end) 截取数组中的元素，start开始的下标，end结束的下标，不包含end本身，如果是负数表示倒数，如果end为空截取到最后 (含头不含尾，从头截到尾)

练习:创建数组，包含a-g，每个字母是一个元素，分别截取cd，f，b；把截取的结果拼接成一个新的数组

|  |
| --- |
| 如何学习API  (1)API的作用  (2)API中需要多个参数，哪些必须，哪些可选  (3)API返回值，是否对原来的数组有影响 |

splice(start,count,value1,value2...) 删除数组中的元素，start开始的下标，count删除的数量，value..删除后补充的元素;如果count为空表示删除到最后，value为空表示不补充任何元素；返回删除的元素，原数组会发生化

课后任务

(1)复习今天内容，整理思维导图

(2)练习

使用遍历数组翻转数组中的元素

使用冒泡排序将数组中的元素进行从小到大排序

(3)预习字符串对象及其API

reverse() 翻转数组中的元素，返回翻转后的数组

sort() 对数组中的元素排序，默认是按照Unicode排序，返回排序后的数组

|  |
| --- |
| sort(function(a,b){  return a-b; //从小到大  //return b-a; 从大到小  }) |

push(value..) 在数据的末尾添加元素，返回数组的长度，原数组会发生变化

pop() 删除数组末尾的一个元素，返回删除的元素，原数组会发生变化

unshift(value..) 在数组的开头添加元素，返回数组的长度，原数组会发生变化

shift() 删除数组开头的一个元素，返回删除的元素，原数组会发生变化

### 19.4二维数组

数组中的每个元素也是数组

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

访问二维数组中的元素 arr[下标][下标]

## 20.字符串

包装对象: 目的就是让原始类型的数据像引用类型一样，具有属性和方法

JS提供了3种包装对象: String Number Boolean

将任意的数据强制转为字符串

new String() 构造函数，类型为对象，使用的时候和字符串没有区别

String() 普通函数，类型为字符串

### (1)转义字符 —— \

转换字符本身的意义

用法: 在要转义的字符前加反斜杠 \

\' 将引号转义成普通的字符

\n 将字符n转义成换行符

\t 将字符t转义成制表符(Tab键的效果)

...

练习：打印出以下路径 C:\xampp\js\string.js

### (2)API

length 获取字符串的长度

charAt() 获取下标对应的字符，也可以使用数组的访问形式 字符串[下标]

charCodeAt() 获取某个字符的Unicode码

indexOf(value) 查找某个字符，返回下标，如果找不到返回-1

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

练习: 声明变量保存邮箱，检查邮箱中是否含有@，如果有返回‘合法的邮箱’，否则返回‘非法的邮箱’

toUpperCase() 将英文字母转大写

toLowerCase() 将英文字母转小写

练习: 声明变量保存4个英文字母作为验证码，无限循环弹出提示框并输入密码(不区分大小写)，如果输入正确结束循环

06\_code.js 06\_code.html

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，如果为负数表示倒数

练习: 声明变量保存邮箱，分别截取邮箱的用户名和域名

tom1993@163.com

substr(start, count) 截取字符串，start开始的下标，count截取的长度；如果start为负数，表示倒数；如果count为空，表示截取到最后；

练习: 声明变量保存身份证号，截取年月日，性别，打印 '1997年09月14日 性别男'

110246199709147690

substring(start,end) 截取字符串，start开始的下标，end结束的下标，如果start大于end，则位置自动交换；如果为负数，自动转为0；如果end为空，截取到最后

**对比slice和substring**

slice中允许使用负数，substring中负数自动转为0；

slice中下标start要求小于end，substring中如果start小于end，则自动交换位置

练习：把英文单词的首字母转大写，其余的字母转小写

'hElLo' -> 'Hello'

split(sep) 将字符串按照指定的字符串切割为数组，sep是指定的字符串

练习: 使用split来获取用户名和域名

tom1234@163.com

### (3)匹配模式(了解)

作用：用于查找、替换字符串

match(value) 用于查找匹配的字符串，返回所有满足条件的字符串，格式为数组 /china/ig

i->ignore 忽略大小写

g->global 全局查找

search(value) 用于查找满足条件的第一个字符串，返回下标，找不到返回-1. 功能类似于indexOf

replace(value1,value2) 查找并替换，value1要查找的字符串，value2要替换的字符串

## 21.Math对象

PI 取圆周率

abs() 取绝对值

ceil() 向上取整

floor() 向下取整

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() 取随机数 >=0 <1

课后任务

(1)复习今天内容，整理思维导图

(2)练习

将一句英文的每个单词首字母大写，其余字母小写

'hOw aRe yOu' -> 'How Are You'

获取0~9之间的随机数

(3)预习Date对象

## 22.Date对象

用于对日期时间的存储和计算

### (1)创建Date对象

new Date('2019/10/18 10:23:30')

new Date(2019,9,18,10,23,30) //1~12月 范围0~11

new Date() 存储当前所在操作系统时间

new Date(1523600000000) 存储的是距离计算机元年毫秒数对应的日期时间

2019/10/18 2019年10月18日 2019-10-18

### (2)获取日期时间

getFullYear/getMonth/getDate/getHours/getMinutes

getSeconds/getMilliseconds/

getDay 获取星期 0~6 星期日~星期六

getTime 获取距离计算机元年的毫秒数

练习：创建Date对象，保存当前的系统时间，打印

'今天是2019年10月18日 11:17:05 星期五'

### (3)获取本地字符串格式

toLocaleString()

toLocaleDateString()

toLocaleTimeString()

### (4)设置日期时间

setFullYear()/setMonth()/setDate()/setHours()/

setMinutes()/setSeconds()/setMilliseconds()

setTime() 设置后，可能各部分的日期时间都发送变化

### (5)拷贝Date对象

var d1=new Date();

var d2=new Date(d1); //拷贝d1对象

练习: 创建Date对象，保存员工的**入职时间**'2019/10/18'，3年后合同到期，计算出**到期时间**；合同到期前1个月续签合同，如果是周末，提前到周五，计算**续签时间**；打印出3个时间的本地日期字符串格式。

## 23.Number对象

new Number() 将数据转为数值，返回对象

Number() 将数据转为数值，返回数值

toFixed(n) 保留小数点后n位

toString(n) 将数值转为字符串，n表示转字符串同时设置的进制

## 24.Boolean对象

new Boolean() 将数据转为布尔型，返回对象

Boolean() 将数据转为布尔型，返回布尔型

!!值 将数据转为布尔型

## 25.错误处理

语法错误(SyntaxError): 错误的使用了中文，缺少括号

引用错误(ReferenceError): 使用未声明的变量

类型错误(TypeError): 把一个变量或者对象中属性当做函数或者方法。

范围错误(RangeError): 参数的使用超出了范围

自定义错误: throw '错误信息'

|  |
| --- |
| 说明：第一种错误出现后，所有的代码不再执行；其它的错误出现后，会影响后续代码的执行 |

错误处理

|  |
| --- |
| try{  尝试执行可能出现错误的代码(引用/类型/范围/自定义)  }catch(err){  如果try中出现错误，才会执行  err：捕获到错误信息  执行其它的代码(解决错误)  }  不会阻止后续代码执行 |

## 26.ES6

ECMA

ECMAScript ES

<http://es6.ruanyifeng.com/> ES6电子书

### (1)块级作用域

使用let关键字声明的变量不存在变量提升，只能在块级作用域下访问

|  |
| --- |
| 块级作用域: 在大括号中，包括if, else, switch,while,do-while,for  注意：let关键字不允许反复声明 |

练习: 计算1~100之间所有整数的和，声明变量使用let关键字

课后任务

(1)复习今天内容，整理思维导图

(2)练习

红球：1~33之间随机取6个，不能重复，并从小到大排序

蓝球：1~16之间随机取1个，可以和之前的红球重复

把获取到7个数字放到数组。

(3)预习es6 函数中参数增强，箭头函数，模板字符串

nodejs 模块、查询字符串、同步和异步、回调函数、线程和进程、http协议

### (2)箭头函数

回调函数的另一种写法，简化了匿名函数，但不等价于匿名函数

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

如果函数中只有一行代码，并且是return形式的，则可以进一步简化为 sort( (a,b)=>a-b )

练习: 创建函数，传递两个参数，实参都是回调函数(返回1个数字)；最后在创建的函数中计算两个数字相加，并打印出来。

### (3)函数增强

给参数设置默认值

|  |
| --- |
| function add(a,b,c=0){  } |

通常设置默认值，都是从后往前

### (4)模板字符串

解决了字符串的拼接

|  |
| --- |
| ` 任何形式的字符串 ${JS表达式} ` |

练习: 创建对象，保存员工的数据(编号，姓名，性别1/0，生日，工资)，使用模板字符串打印员工的数据

编号:

姓名:

...

# NODEJS

## 1.NodeJS概述

基于谷歌V8引擎，运行在服务器端的JS解释器

对比JS和NodeJS

(1)JS运行在浏览器端，存在有多个解释器，容易产生兼容性问题；而NodeJS运行在服务器端，只有V8引擎一种解释器，不存在兼容性问题

(2)两者有相同的内置(ES)对象、自定义对象；不同的宿主的对象

(3)JS用于浏览器端的交互效果，NodeJS用于服务器端的操作，包括web服务器的创建，数据库访问，文件操作等

NodeJS执行方式

脚本模式 node C:/xampp/.../01.js 回车

交互模式 node 回车 进入交互模式

退出交互模式 两次ctrl+c 或者 .exit

适用场景

属于I/O密集型，基于社交网络的大规模web应用

## 2.全局对象

NodeJS: global

在交互模式下声明的变量和创建的函数都属于全局global下的

在脚本模式下声明的变量和创建的函数都不属于全局global下的

JS: window

在JS文件中声明的变量和创建的函数都属于全局window下的

使用全局对象window访问变量a和函数fn

## 3.console对象

标准输出

console.log() 输出日志

console.info() 输出消息

console.warn() 输出警告

console.error() 输出错误

console.time('自定义字符串') 开始计时

console.timeEnd('自定义字符串') 结束计时

说明：开始计时和结束计时的字符串保持一致

练习: 使用while、do-while、for循环相同次数，查看耗时

## 4.process对象

进程对象

process.arch 查看当前CPU架构

process.platform 查看当前的操作系统

process.env 查看当前的环境变量

process.version 查看当前nodejs的版本号

process.pid 查看当前进程的编号

process.kill() 结束某个编号的进程

## 5.Buffer对象

缓冲区: 在内存中存储数据的区域，常用于存储网络传输时的资源

创建Buffer

let buf=Buffer.alloc(5,'abcde')

把buffer转为字符串

String(buf)

buf.toString()

## 6.模块

模块：就是一个独立的功能体

在NodeJS下，每个文件都是一个模块；每个模块可以引入其它的模块，也可以被其它的模块所引入

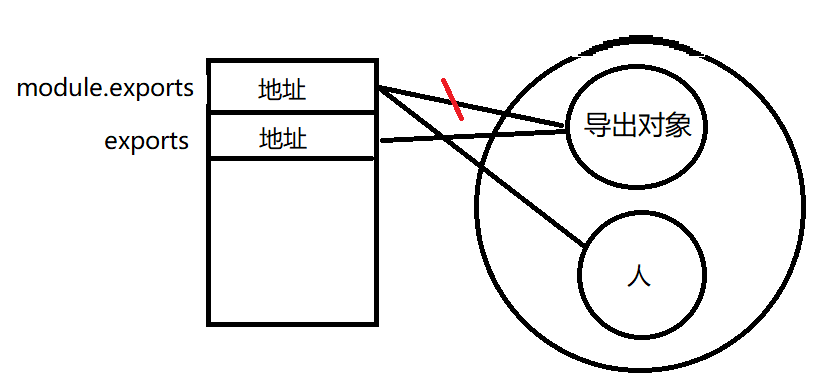
具体分为**自定义模块、核心模块、第三方模块**

NodeJS自动为每个文件添加了一个构造函数，所有的代码默认都是被这个函数所包含

### 6.1模块系统

|  |
| --- |
| (function (exports, require, module, \_\_filename, \_\_dirname) {  程序员写的代码  }) |
| require: 函数，引入一个模块  module: 对象，当前模块对象 module.exports: 导出对象  exports: 导出的对象别名  \_\_filename: 当前模块的绝对路径和文件名称  \_\_dirname: 当前模块的绝对路径 |

dir -> directory 目录 file 文件



### 6.2.模块引入分类

|  |  |  |
| --- | --- | --- |
|  | 以路径开头 | 不以路径开头 |
| 文件模块 | require('./circle.js')  常用于引入自定义模块 | require('url')  常用于引入官方提供的核心模块 |
| 目录模块 | require('./02\_2')  到02\_2目录下寻找package.json文件main属性对应的文件，如果找不到再去找index.js | require('04\_2')  到当前目录下的node\_modules目录中寻找04\_2目录，如果找不到会到上一级目录，直到顶级目录；常用于引入第三方模块 |

练习: 创建文件模块03\_1.js，引入当前目录下的03\_2目录模块，在03\_2目录中含有文件fun.js，导出一个函数add（计算任意两个数字相加），在03\_1.js调用该函数。

练习: 在05目录下创建文件05\_1.js，引入不带路径的目录模块05\_2，在05\_2中含有文件index.js，导出一个函数(计算任意两个数字相乘)，在05\_1.js中调用

### 6.3.包和npm

包(package): 就是一个目录模块

CommonJS规范，JS向服务器端发展所遵循的规范，例如模块的引入，导出都是由这个规范来的

npm: 包管理管理工具

**切换目录**

cd 完整路径 回车

进入不同盘符，需要添加 盘符名称: 回车

在指定目录下空白区域，按住shift键，单击鼠标右键，选择'在此处打开powershell窗口'

**npm官网**

www.npmjs.com

**使用npm下载安装包**

npm install 包名称

npm init -y 初始化一个package.json，项目说明文件，其中会记录安装的包相关信息

npm install 自动安装package.json中记录的包

### 6.4.查询字符串模块

查询字符串: 浏览器向web服务器发请求，传递数据的一种方式，位于URL中?后的部分

[https://search.jd.com/Search?keyword=手机&enc=utf-8](https://search.jd.com/Search?keyword=%E6%89%8B%E6%9C%BA&enc=utf-8)

parse() 将查询字符串解析为对象

stringify() 将一个对象格式化为查询字符串

练习: 获取查询字符串中商品的价格和名称

price=4999&title=dell

### 6.5.URL模块

URL: 网址，用于向服务器发请求

|  |
| --- |
| http://www.codeboy.com:80/admin/login.html?uname=root&upwd=123456 |

parse() 将URL解析为对象

format() 将对象格式化为URL

练习: 获取以下URL中查询字符串的数据

https://www.tmooc.cn:443/course/web.html?cid=1909&cname=js

①解析URL为对象，获取到查询字符串

②解析查询字符串为对象，获取到数据

### 6.6.定时器(Timers)模块

(1)一次性定时器

|  |
| --- |
| 开启  let timer=setTimeout(回调函数, 间隔时间);  当间隔时间到了，会执行1次回调函数  清除  clearTimeout(timer) |

(2)周期性定时器

|  |
| --- |
| 开启  let timer=setInterval(回调函数,间隔时间);  每隔一段时间，执行一次回调函数  清除  clearInterval(timer); |

练习: 打印3次hello后，清除定时器

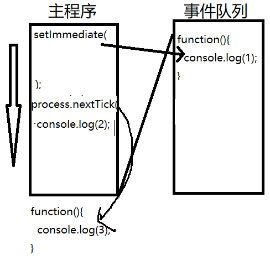
(3)立即执行定时器

|  |
| --- |
| 开启  let timer=setImmediate(回调函数)  清除  clearImmediate(timer) |

在事件队列的最前边执行

|  |
| --- |
| process.nextTick(回调函数) |

在主程序的后边执行



### 6.7.同步和异步

同步: 相当于程序的顺序执行，会阻止后续代码执行，只有前边的代码执行完，才会执行后边的

异步: 不会阻止后续代码的执行，异步的形式是通过回调函数的形式，在事件队列中执行回调函数

## 7.文件系统(fs)模块

用于操作文件的

文件: 包含文件形式和目录形式

### (1)创建目录

mkdir(path, callback) / mkdirSync(path)

path 目录的路径

callback 回调函数

err 错误信息

课后任务

(1)复习今天内容，整理思维导图

(2)练习

创建main.js文件，引入目录模块home，在home中含有文件fun.js，导出一个函数，传递1个参数，创建文件，在main.js中调用该函数

fn('mydir')

(3)预习http协议，http模块

### (2)查看文件的状态

异步/同步

fs.stat(path, callback)/fs.statSync(path)

path 文件的路径

callback 回调函数，用于获取结果

err 可能产生的错误

stats 具体的文件的状态

|  |
| --- |
| 异步获取结果是通过回调函数的参数  同步获取结果是通过返回值 |

### (3)读取目录

fs.readdir(path,callback)/readdirSync(path)

path 目录的路径

callback 回调函数

err 错误

result 读取的结果，格式为数组

### (4)移除目录

fs.rmdir(path,callback)/fs.rmdirSync(path)

### (5)写入文件

fs.writeFile(file,data,callback)/fs.writeFileSync(file,data)

file 要写入的文件

data 要写入的数据

如果文件不存在，会先创建文件然后写入

如果文件已经存在，会先清空内容然后写入

### (6)追加写入

fs.appendFile(file,data,callback)/

fs.appendFileSync(file,data)

如果文件不存在，会创建文件然后写入

如果文件已经存在，会在末尾写入

练习: 创建数组，包含多个姓名，使用同步方法把数组中的数据写入到文件student.txt中

遍历数组得到每个元素

### (7)读取文件

fs.readFile(file,callback)/fs.readFileSync(file)

callback

err

data 读取的数据，格式为buffer

### (8)删除文件

fs.unlink(path,callback)/fs.unlinkSync(path)

### (9)检测文件是否存在

fs.existsSync(path)

存在返回true，不存在返回false

练习: 如果文件2.txt存在，使用同步方法删除该文件

如果目录mydir不存在，使用同 步方法创建该目录

### (10)文件流

fs.createReadStream(path) 创建可读取的流

|  |
| --- |
| //1.读取文件  readStream.setEncoding('utf8');//设置编码  readStream.on('data', function(chunk){ }); //监听是否有数据流，一旦有数据，自动将数据放入到参数chunk，数据是分段  readStream.on('end', function(){ });//监听是否读取结束 |
| //2.拷贝文件  let writeStream=fs.createWriteStream(**path**);//创建写入流，同时也会创建文件  readStream.pipe(writeStream);//把读取流通过管道添加到写入流，完成了文件的拷贝 |

## 8.http协议

浏览器和web服务器之间的通信协议

### (1)通用头信息

Request URL: 向服务器获取哪些内容

Request Method: 请求的方法 get(获取)/post(发送数据)

Status Code: 响应的状态码

2\*\*：成功的响应

3\*\*：响应的重定向，执行跳转

4\*\*：客户端请求错误

5\*\*：服务端错误

Remote Address：请求的服务器的IP地址和端口号

### (2)响应头信息

Content-Length: 响应的文件的大小，单位通常是字节

Content-Type:响应的文件类型

Content-Encoding: 服务器端的压缩形式 gzip

Connection: 建立连接的方式 keep-alive 持久连接

Transfer-Encoding: 传输方式，chunked 分段传输

Location: 跳转的URL，常结合着状态码的3\*\*使用

### (3)请求头信息

Accept: 浏览器接收的文件类型有哪些

Accept-Encoding: 接收的压缩类型有哪些

User-Agent: 客户端使用的浏览器

### (4)请求主体

可有可无，用于浏览器向服务器发送的数据

## 9.http模块

既可以创建web服务器，也可以模拟浏览器向其它的web服务器发请求

### (1)模拟浏览器

http.get(url, callback) 向服务器发请求

url 请求的URL

callback 回调函数，用于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', function(chunk){ })

监听服务器端是否有数据响应

chunk 就是响应的数据，分段传输的，格式buffer

### (2)创建web服务器

|  |
| --- |
| const http=require('http'); //引入http模块  let app=http.createServer();//创建web服务器  app.listen(8080); //设置端口号  //接收请求，并作出响应  app.on('request',function(req,res){  req 请求的对象  req.url 请求的URL,显示端口号后的部分  req.method 请求的方法  res 响应的对象  res.writeHead(状态码, 头信息对象) 设置响应的状态码和头信息  res.write() 设置响应的内容  res.end() 结束并发送响应  }) |

http://127.0.0.1:8080

http://localhost:8080

课后任务

(1)复习今天内容，整理思维导图

(2)创建web服务器，设置端口8080，接收请求，根据请求作出响应

/list 响应 'this is list page'

/index 响应 ‘this is home page’

/study 跳转 http://www.tmooc.cn

其它 响应状态码404 响应内容 404 not found

(3)预习express 第三方模块

## 10.使用zlib模块对文件压缩

|  |
| --- |
| const zlib=require('zlib'); //引入压缩模块  Content-Encoding:gzip //告诉浏览器内容为gzip压缩，收到以后会进行解压缩  let gzip=zlib.createGzip(); //创建gzip压缩  fs.createReadStream(文件路径).pipe(gzip)  .pipe(res);//以流的方式读取文件并添加到压缩，把压缩响应到浏览器 |

## 11.express框架

基于NodeJS平台，快速、开放、极简的web开发框架

安装 npm install express

|  |
| --- |
| const express=require('express'); //引入第三方模块let app=express();//创建web服务器  app.listen(8080); //设置端口 |

### (1)路由

浏览器向web服务器发请求，web服务器根据请求来作出响应，这个过程就是路由来完成

路由三要素: 请求方法、请求URL、回调函数

res 响应对象

send() 设置响应的内容并发送

redirect() 响应的重定向，跳转到另一个URL

sendFile() 响应文件并发送，需要使用绝对路径\_\_dirname

练习: 创建路由

/list 响应 '这是商品列表'

/index 响应 '这是首页'

/ 跳转到 /index

/

req 请求对象

method 请求的方法

url 请求的URL

query 获取传递的查询字符串，并解析为对象

params 获取路由传参的数据，格式为对象

|  |
| --- |
| 路由传参  app.get('/package/:pname',function(req,res){  req.params //获取路由传参数据  });  浏览器传递  http://127.0.0.1:8080/package/mysql  mysql就是实参，就是要传递的数据 |

### (2)获取post请求的数据

|  |
| --- |
| req.on('data', function(chunk){//事件:监听是否有数据传递，将传递的数据放入到回调函数的参数chunk  chunk //格式为buffer，转字符串后为查询字符串，需要解析为对象  }) |

练习: 创建路由，请求方法get，请求URL /reg，响应文件reg.html，html中包含有(用户名，密码，手机)

练习: 创建订单详情路由，请求的方法get，URL /order，使用路由传参传递订单的编号oid和订单的日期date，把这两项数据发送到浏览器中

练习: 创建路由

/list 响应 '这是商品列表'

/index 响应 '这是首页'

/ 跳转到 /index

/admin 响应 响应文件admin.html

### (3)对比get和post请求

|  |  |  |  |
| --- | --- | --- | --- |
|  | 传递方式 | 获取方式 | 其它 |
| get | 查询字符串(可见，容易被浏览器缓存) | req.query | 速度快，安全性相对低，数据量小 |
| post | 请求主体(不可见) | req.on('data',..) | 速度慢，安全性相对高，数据量大 |

商品模块

列表 /list 删除 /delete

/**product**/list /**product**/list

用户模块

列表 /list 删除 /delete

/user/list /user/delete

## 12.路由器

路由在使用中，可能出现不同模块下相同的URL，为了团队协作，将同一个模块下的所有路由放到路由器下来管理

### (1)创建路由器

|  |
| --- |
| const express=require('express');  let router=express.Router();//1.创建路由器对象  router.get('/list',function(req,res){ //2.往路由器中添加路由  res.send('这是用户列表');  });  module.exports=router; //3.导出路由器对象 |

### (2)在服务器中引入并使用

|  |
| --- |
| const userRouter=require('./user.js');//引入路由器  app.use('/user',userRouter);//在服务器下挂载路由器，并给所有url添加前缀 /user 访问 /user/list |

课后任务

(1)复习今天内容，整理思维导图

(2)练习: 创建商品路由器模块，添加路由；在服务器下引入并使用

(3)预习express中间件， 回顾之前mysql知识

## 13.中间件

用来处理请求，最终是要为路由服务

应用级中间件、路由级中间件、内置中间件、第三方中间件

### (1)应用级(自定义)中间件

app.use(url, function(req,res,next){ })

url 表示中间件所要拦截的请求URL，如果为空，表示要拦截所有的请求

next 是一个函数，用于执行下一步（可能是下一个中间件，也可能是路由）

练习: 创建路由，方法get url /shopping，传递商品的价格price，在中间件中判断，如果价格满100打9折，最后在路由中响应 '商品的价格: \*\*\*'

### (2)路由级中间件

路由器的使用

### (3)内置中间件

express中只有一个内置中间件

把所有的静态资源(html,css,js,图像...)托管到指定的目录，如果浏览器要请求这些文件，自动到该目录下寻找，不需要再写路由响应

app.use( express.static('目录') )

练习：创建web服务器，托管静态资源到public目录，包含文件login.html；点击提交，再次向服务器发请求(post，/mylogin)，在路由中获取表单中的数据04\_three.js

### (4)第三方中间

|  |
| --- |
| //1.引入body-parser  const bodyParser=require('body-parser');  //2.使用中间件，可以将post请求数据解析为对象  app.use( bodyParser.urlencoded({  extended:false //不使用第三方的qs模块，而是使用querystring模块解析为对象  }) );  //3.在路由中获取post请求数据  req.body |

其它中间件: compression用于压缩、cors 跨域、session 用于登录..

## 14.mysql模块

建立连接

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

SQL语句

INSERT INTO emp VALUES(...);

DELETE FROM emp WHERE eid=5;

UPDATE emp SET ename='..',sex=1 WHERE eid=2;

SELECT \* FROM emp;

|  |
| --- |
| //创建连接对象  let connection=mysql.createConnection()  //执行连接  connection.connect()  //执行SQL命令  connection.query(sql,callback)  sql 要执行的SQL命令  callback 回调函数，用于获取执行的结果 |

SQL注入

SELECT \* FROM user WHERE uname='root' AND upwd='123456' OR 1=1;

DELETE FROM user WHERE uid=4 OR 1=1;

**使用占位符(?)，对用户提供的数据进行过滤**

INSERT INTO emp VALUES(?,?...)

练习: 往员工表中插入数据

|  |
| --- |
| 插入、修改、删除返回对象  查询返回数组 |

使用连接池

|  |
| --- |
| let pool=mysql.createPool() //创建连接池对象  poo.query(sql, callback) //执行SQL命令 |

课后任务

(1)复习今天内容整理思维导图

(2)课后练习

创建web服务器，托管静态资源到public目录，创建文件add.html,点击提交，向服务器发请求(get, /add)，把数据插入到tedu数据库下的dept表中，响应“部门添加成功”

