

MYSQL

MySQLDay01:

1.数据的存储方式

特定的文件/内存/第三方服务器/数据库服务器

2.什么是数据库

数据库就是按照特定的形式来组织，存储数据，最终是为了数据的操作方便——增删改查

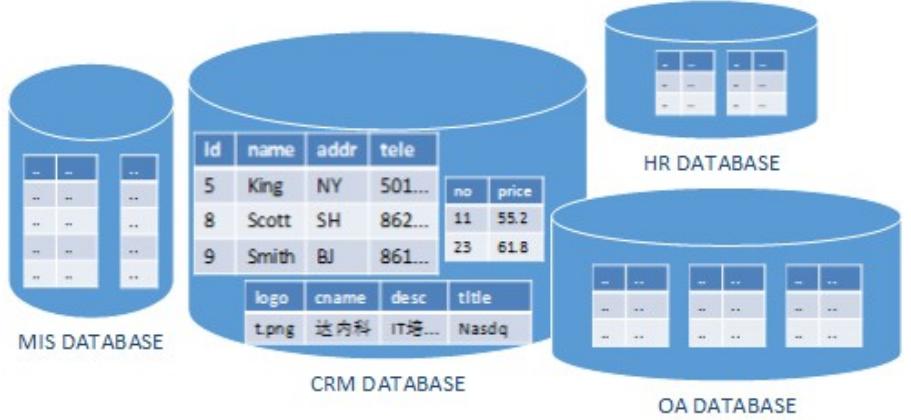
(1)数据库的发展历史

网状数据库 -> 层次型数据库 -> 关系型数据库 -> 非关系型数据库 (MongoDB)

(2)关系型数据库的逻辑结构

Server -> Database -> Table -> Row -> Column

服务器 数据库 数据表 行 列



3.mysql 数据库

马丁: MariaDB

oracle: mysql

xampp 服务器套装软件

包含有多款服务器, 例如 Apache、mysql...

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

(1) 部署结构

服务器端: 负责存储/维护数据——银行的数据库服务器

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

确保端口 **3306** 不被占用

客户端: 负责连接服务器, 对数据进行增删改查的操作——ATM 机

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

(2) 使用客户端连接服务器端

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

`-h` host 服务器(IP地址/域名) 127.0.0.1/localhost

`-P` port 端口

`-u` user 用户名 root 管理员用户

`-p` password 密码

简写形式 `mysql -uroot`

4. 常用的管理命令

`quit;` 退出

`show databases;` 显示当前服务器下所有的数据

`use 数据库名称;` 进入指定的数据库

`show tables;` 显示当前数据库下所有的表

`desc 表名称;` 描述当前表中都有哪些列

所有的管理命令都要加英文分号结尾。

`pma_history`

`pma_favorite`

pma_recent

5.SQL 命令

SQL: 结构化查询语言，是用来操作关系型数据库，主要是对数据进行增删改查。

SQL 命令的两种执行方式

(1) **交互模式**

客户端输入一行，点击回车，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 SQL 命令写在一个脚本文件中，然后一次性的提交给服务器，适用于批量的增删改查。

```
mysql -uroot<C:/xampp/..../01.sql 回车
```

SQL 的语法规规范

- (1) 一条 SQL 命令可以跨越多行，以英文的分号为结尾
- (2) 假如某一条 SQL 命令出现语法错误，则此条命令后所有代码不执行
- (3) SQL 命令中可以使用单行注释 (#...) 和多行注释 (/* ... */)
- (4) SQL 命令不区分大小写，习惯上关键字大写，非关键字小写

6. 常用的 SQL 命令

(1) 丢弃数据库，如果存在

```
DROP DATABASE IF EXISTS qq;
```

(2)创建新的数据库

```
CREATE DATABASE qq;
```

(3)进入数据库

```
USE qq;
```

(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', 'ccy', 'b', '85');
```

(6)查询数据

```
SELECT * FROM student;
```

MySQLDay02:

1.SQL 命令

(1)删除数据

```
DELETE FROM user WHERE uid='3';
```

(2)修改数据

```
UPDATE user SET upwd='777777',userName='龟田依然' WHERE uid='2';
```

2.计算机如何存储字符

(1)如何存储英文字符

ASCII 对所有的英文字母及符号进行了编码，总共有 128 个

a b c.... 979899

Latin-1 对所有欧洲符号进行了编码，总共有 256 个，兼容 ASCII

(2)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

GBK 对两万多汉字进行了编码，兼容 GB2312

BIG5 台湾繁体字编码

Unicode 对世界上主流国家常用的语言进行了编码，具体分为 3 种存储方案，UTF-8，UTF-16，UTF-32

3.解决 mysql 中文乱码

mysql 默认使用 Latin-1 编码

需要统一使用 UTF-8

(1)脚本文件另存为的编码

(2)客户端连接服务器端使用的编码

SET NAMES UTF8;

(3)服务器端创建数据库使用的编码

CREATE DATABASE xz CHARSET=UTF8;

3.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(

id 列类型

);

(1)数值型——可以不加引号

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，最大 3.4×10^{38} ，可能产生误差。

DOUBLE 双精度浮点型，占 8 个字节，可能产生误差

DECIMAL(M,D) 定点小数，小数点不会变化，几乎也不会产生误差，M 代表总的有效位数，D 代表小数点后的有效位数。

BOOL/BOOLEAN 布尔型，通常用于存储执行两个值的数据，TRUE/FALSE，真正存储的时候 TRUE 转为 1，FALSE 转为 0。

因为 mysql 中没有真正的布尔型，最终会自动转为微整型 **TINYINT**。

TRUE 和 FALSE 是关键字，使用的时候不能加引号。

(2) 日期时间型——必须加引号

DATE 日期型 2020-12-25

TIME 时间型 16:36:25

DATETIME 日期时间型 2020-12-25 16:36:25

(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 的最大值是 65535

CHAR(M) 定长字符串，可能产生空间浪费，操作速度相对快，往往存储一些固定长度的数据，例如手机号码，身份证号等固定长度的数据，M 的最大值是 255

TEXT(M) 大型变长字符串，M的最大值是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

浮点型

123456.789*10^-1

12345.6789

1234.56789*10

123.456789*10^2

计算机常用的单位

TB GB MB KB BYTE(字节) BIT 1BYTE=8BIT(位)

十进制

1 2 3 4 5 6 7 8 9 10

二进制

1 10 11 100 ...

使用合理的列类型

```
CREATE TABLE t1(  
    id INT,  
    age TINYINT,  
    phone CHAR(11),  
    price DECIMAL(4,2), #99.99  
    sex BOOL, # TRUE/1 ->男 FALSE/0 ->女  
    article VARCHAR(10000),  
    ctime DATE  
);
```

MySQLDay03:

1.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，否则被认为非法的插入。

(1)主键约束——PRIMARY KEY

声明了主键约束的列上不允许插入重复的值，一个表中只能有一个主键，通常加在编号列上，查询的时候会按照主键编号从小到大排序，会加快查找速度。

NULL 是一个暂时无法确定的值，无法确定当前手机号码，邮箱...无法确定一个人的工资..

NULL 是关键字，不能加引号

主键约束不允许为 NULL

(2)唯一约束——UNIQUE

声明了唯一约束的列上，不允许插入重复的值，但允许插入 NULL，并且多个 NULL。一个表中可以出现多个唯一约束。

(3)非空约束——NOT NULL

声明了非空约束的列上禁止为 NULL

(4)检查约束——CHECK

可以对要插入的数据进行自定义的验证

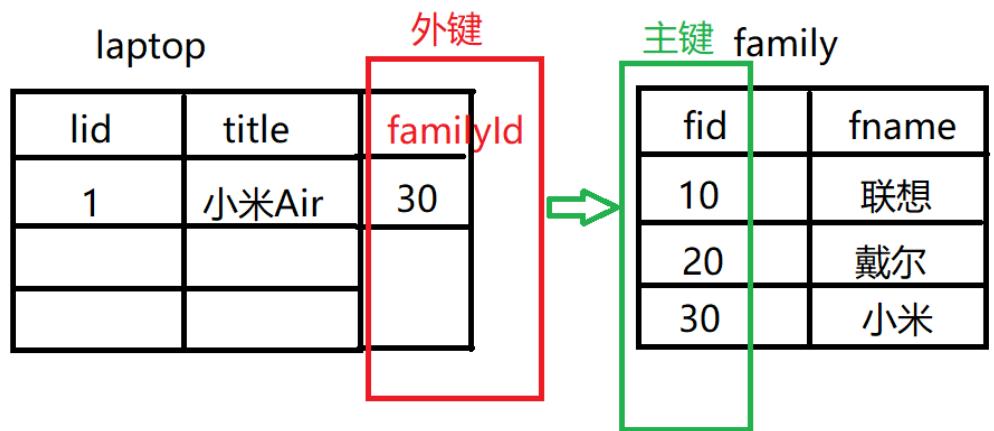
```
CREATE TABLE t1(  
    salary DECIMAL(7,2) CHECK(salary>=0)  
);
```

mysql 不支持检查约束，认为会对服务器造成一定的压力，降低数据的插入速度。可以使用 JS 代替

(5)外键约束——FOREIGN KEY

声明了外键约束的列上，取值必须在另一个表的主键列上出现过，而且两者的列类型要保持一致。允许为 NULL

FOREIGN KEY(familyId) REFERENCES family(fid)



(6)默认值约束——DEFAULT

mysql 可以使用 DEFAULT 关键字设置默认值，具体应用有两种方式

```
INSERT INTO laptop VALUES(1,'灵越',DEFAULT...);
```

```
INSERT INTO laptop(lid,title,spec) VALUES(2,'thinkpad','商务版');
```

2.自增列

AUTO_INCREMENT: 自动增长，假如某个列声明了自增列，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 插入。

注意事项：自增列必须添加在主键约束的列上，并且是整数编号的列。

数据库中不保存文件(图像,文件,视频)，保存的是文件的路径。

img/range.png range.zip range.avi

3.简单查询

(1)查询特定的列

示例: 查询所有的员工的编号和姓名

```
SELECT eid,ename FROM emp;
```

练习: 查询所有员工的姓名, 性别, 生日, 工资

```
SELECT ename,sex,birthday,salary FROM emp;
```

(2)查询所有的列

```
SELECT * FROM emp;
```

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

(3)显示不同的记录

示例: 查询出都有哪些性别的员工

```
SELECT DISTINCT sex FROM emp;
```

练习: 查询出员工都分布在哪些部门

```
SELECT DISTINCT deptId FROM emp;
```

(4)给列起别名

示例: 查询出所有的员工编号和姓名, 起汉字别名

```
SELECT eid AS 编号,ename AS 姓名 FROM emp;
```

练习：查询出所有员工的姓名，性别，生日，工资，使用一个字母作为别名

```
SELECT ename AS a,sex AS b,birthday c,salary d FROM emp;
```

说明：AS关键字可以省略的。

(5)查询时执行计算

示例：计算 $6+5*7.57-8/3.25$

```
SELECT 6+5*7.57-8/3.25;
```

练习：查询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 2000，年终奖 20000，查询出所有员工的姓名及其年薪，使用汉字别名

```
SELECT ename AS 姓名,(salary+2000)*12+20000 AS 年薪 FROM emp;
```

(6)查询结果集排序

示例：查询出所有部门的数据，结果集按照编号从小到大排序

```
SELECT * FROM dept ORDER BY did ASC; #ascendant 升序
```

示例：查询出所有部门的数据，结果集按照编号从大到小排序

```
SELECT * FROM dept ORDER BY did DESC; #descendant 降序
```

练习：查询出所有员工的数据，结果集按照工资的降序排列

```
SELECT * FROM emp ORDER BY salary DESC;
```

练习：查询出所有员工的数据，结果集按照年龄从大到小排列

```
SELECT * FROM emp ORDER BY birthday ASC;
```

练习：查询出所有员工数据，结果集按照姓名升序排列

```
SELECT * FROM emp ORDER BY ename ASC;
```

按照字符串排序其实是按照对应的 Unicode 码来排列的

如果不加排序规则，默认是按照升序排列：“ASC 可以省略不写”

练习：查询出所有员工数据，结果集按照工资降序排序，如果工资相同按照姓名升序排列

```
SELECT * FROM emp ORDER BY salary DESC, ename/(ename ASC);
```

练习：查询出所有员工数据，所有女员工显示在前，如果性别相同，按照生日降序排列。

```
SELECT * FROM emp ORDER BY sex/(sex ASC),birthday DESC;
```

(7)条件查询

示例：查询出编号为 5 的员工数据

```
SELECT * FROM emp WHERE eid=5;
```

练习：查询出姓名为 david 的姓名，性别，生日。

```
SELECT ename,sex,birthday FROM emp WHERE ename='david';
```

练习：查询出 20 号部门下的员工有哪些

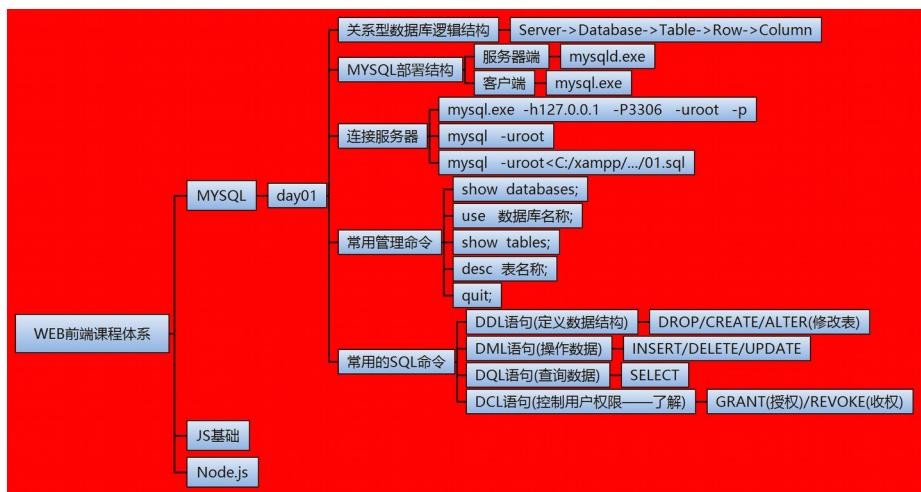
```
SELECT * FROM emp WHERE deptId=20;
```

练习：查询出所有的女员工有哪些

```
SELECT * FROM emp WHERE sex=0;
```

课后任务

查看学子商城数据库设计中的 `xz_laptop` `xz_laptop_pic` `xz_laptop_family`



1.简单查询

(1)条件查询

练习：查询出工资在 7000 以上的男员工有哪些

```
SELECT * FROM emp WHERE salary>7000 AND sex=1;
```

练习：查询出 1991 年后出生的员工有哪些

```
SELECT * FROM emp WHERE birthday>'1991-12-31';
```

> < >= <= != 不等于

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 5000~7000 之间的员工有哪些

```
SELECT * FROM emp WHERE salary>=5000 AND salary<=7000;
```

```
SELECT * FROM emp WHERE salary BETWEEN 5000 AND 7000;
```

练习：查询出工资在 5000 以下和 7000 以上的员工有哪些

```
SELECT * FROM emp WHERE salary<5000 OR salary>7000;
```

```
SELECT * FROM emp WHERE salary NOT BETWEEN 5000 AND 7000;
```

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 号部门和 30 号部门的员工有哪些

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

(2)模糊条件查询

示例：查询出员工姓名中含有字母 e 的员工有哪些

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

练习：查询出员工姓名中以 e 结尾的员工有哪些

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

练习：查询出员工姓名中倒数第 2 个字符为 e 的员工有哪些

```
SELECT * FROM emp WHERE ename LIKE '%e_';
```

% 匹配任意 0 个或者多个字
符 >=0

_ 匹配任意1个字符 =1
以上两个匹配符只能结合着
关键字 LIKE 使用。

(3)分页查询

查询的结果集有太多的记录，一次显示不完，可以使用分页查询。

需要有两个已知的条件：当前页码和每页的数据量

计算开始查询的值=(当前的页
码-1)*每页的数据量

语法格式

SELECT * FROM emp LIMIT **开始查询的值,每页的数据量**;

练习：假设每页显示5条数据，查询出前3页每一页数据

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;

练习：假设每页显示8条数据，查询出前两页每页数据

SELECT * FROM emp LIMIT 0,8;

SELECT * FROM emp LIMIT 8,8;

2.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(ename) FROM emp; #15
```

```
SELECT COUNT(deptId) FROM emp; #14
```

推荐使用 **主键列** 或者 *

```
SELECT COUNT(eid) FROM emp;
```

练习：查询出所有男员工的数量

```
SELECT COUNT(eid) FROM emp WHERE sex=1;
```

聚合函数

函数，就是一个功能体，接收若干个数据，返回处理的结果。

COUNT() 求数量 / SUM()

求总和 / AVG() 平均

MAX() 最大 / MIN() 最小

练习：计算所有女员工的工资总和

```
SELECT SUM(salary) FROM emp WHERE sex=0;
```

练习：查询出所有员工的平均工资

```
SELECT AVG(salary) FROM emp;
```

练习：查询出年龄最大的员工的生日

```
SELECT MIN(birthday) FROM emp;
```

练习：查询出工资最高的员工的工资是多少

```
SELECT MAX(salary) FROM emp;
```

分组查询中只能查询分组条件和聚合函数的结果

示例：查询每个部门员工的平均工资，最高工资，最低工资

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

练习：分别查询出男女员工的数量，平均工资，工资总和分别是多少。

```
SELECT sex,COUNT(eid),AVG(salary),SUM(salary) FROM emp GROUP BY sex;
```

示例：查询日期中的年份

```
SELECT YEAR('2020-4-3');
```

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

示例：查询出研发部的所有员工

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 号部门的所有员工

```
SELECT * FROM emp WHERE deptId=10;
```

综合：

```
SELECT * FROM emp WHERE deptId=(SELECT did FROM dept WHERE dname='研发部');
```

子查询是多个 SQL 语句的嵌套，把一个的结果作为
另一个的条件。

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——6000

```
SELECT salary FROM emp WHERE ename='tom';
```

步骤 2：查询出工资比 6000 高的员工

```
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 年出生的员工有哪些

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

综合：

```
SELECT * FROM emp WHERE YEAR(birthday)=(SELECT YEAR(birthday) FROM emp WHERE ename='tom') AND  
ename!='tom';
```

(3) 多表查询

每条数据中所查询的列是分布在不同的表中

示例：查询出每个员工的姓名及其部门名称

```
SELECT ename,dname FROM emp,dept;
```

错误：产生笛卡尔乘积

需要添加查询的条件

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

这种查询方法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

新增多表查询语法

(1) 内连接——和之前的查询结果一样

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

(2) 左外链接

左侧表中所有的记录都显示，先写哪个表哪个就是左。

```
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

右侧表中所有的记录都显示，后写哪个表哪个就是右

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

OUTER 关键字可以省略

(4) 全连接

FULL JOIN mysql 不支持

左外和右外的结果联合到一起

UNION 合并相同的记录

UNION ALL 不合并相同的记录

```
(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);
```

学习一门语言的基本步骤

(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)通用的小程序

(8)函数和对象

(9)第三方的库和框架

(10)实用的项目

有基础的做程序员必做 50 题

[HTTPS://WENKU.BAIDU.COM/VIEW/AF66E2F14AFE04A1B071DE42.HTML](https://wenku.baidu.com/view/af66e2f14afe04a1b071de42.html)

推荐书籍

《JAVASCRIPT 高级程序设计》

JavaScript

JavaScriptDay01:

1.JS 概述

(1)历史

1995 年，JS 最早出现了 Netscape 的浏览器

1996 年，IE3 中也出现了 JS，叫 JScript

1997 年，出现了 ECMA 组织，制定了统一的标准 ECMAScript

2009 年，JS 遵循了 CommonJS 规范，开始向服务器端发展——Node.js

(2)现状

既可以运行在客户端的浏览器中，也可以运行在服务器端，实现例如 Java，php 等后端语言的功能。

(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

(4)应用场景

浏览器端的交互效果

服务器端的操作，例如创建 web 服务器，数据库的操作

2.搭建开发环境

(1)浏览器中自带的 JS 解释器

谷歌浏览器

(2)服务器端安装 Node.js

下载地址 nodejs.org

安装成功后，在 cmd 下 node -v 回车查看版本号

(3)运行 JS 代码

浏览器

创建 01.js 和 01.html 两个文件，把 01.js 引入到 01.html 中，双击使用浏览器打开 html 文件即可运行 js。

```
<script src="01.js"></script>
```

Node.js

在 cmd 下 node C:/xampp/.../01.js 回车运行

3.JS 的书写规范

区分大小写

每行结束的分号可加可不加，建议都加

分为单行注释(//...)和多行注释(/ * ... * /)

JavaScriptDay02:

1. 变量

x=1 y=2

变量是用来存储数据的容器

```
var name='然哥';
```

使用 var 关键字声明变量，接着把数据保存到声明变量中。

(1) 变量的命名规则

变量名称由字母、数字、下划线、美元符号(\$)组成，不能以数字开头。

不允许使用关键字作为变量名

liran li_ran liRan

username user_name userName

可读性

(2) 变量的赋值

var salary;

如果变量只是声明未赋值，则存储的数据为 undefined

变量允许多次赋值，并赋不同类型的值，这也是弱类型语言的特点。

(3) 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。

2.常量

```
const sex='男';
```

声明常量必须赋值，一旦声明后不允许重新赋值。

3.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

10进制 1 2 3 4 8 9 10 .. 15

8 进制 1 2 3 4 7 10 11 12..

16 进制 1 2 3 4 ... 8 9 a ... f 10

8 进制是以 0 开头的数字，例如 010 -> 8

16 进制以 0x 开头的数字，例如 0xf -> 15 不区分大小写

浮点型

3140 -> 3.14e3

31.4e2

0.00314 -> 3.14e-3

检测数据类型 `typeof`

(2) 字符串型

所有被引号所包含的数据都是字符串型，不区分单双引号。

查看任意一个字符的 Unicode 码

'然'.**charCodeAt()**

(3)布尔型

只有两个值 true/false，表示真/假

通常只有两个结果的数据可以使用布尔型，例如是否登录、是否注册、是否为会员..

(4)未定义型

只有一个值 undefined，声明了变量未赋值则为 undefined

(5)空

只有一个值 null，类型为 object，常和引用类型的数据一起使用

4.数据类型转换

(1)隐式转换

①数值+字符串 数值转为字符串型

```
1+'2' //'12'
```

②数值+布尔型 布尔型转为数值型, true 转为 1, false 转为 0

```
2+true //3
```

```
2+false //2
```

③字符串+布尔型 布尔型转为字符串型

```
'3'+false //'3false'
```

练习：查看以下程序的运行结果。

```
var a=2,b=true,c='tedu';
```

```
console.log(a+b+c); //'3tedu'
```

```
console.log(b+c+a); //'truetedu2'
```

```
console.log(c+a+b); //'tedu2true'
```

JS 中加号的作用

①作为加法运算

②字符串之间的拼接

减、乘、除的隐式转换

如果运算符两端不是数值，自动调用 Number 函数转为数值；

(2) 强制转换

① 强制转为数值型

Number()

如果要转换的数据为字符串，并含有非数字，则返回 NaN(Not a Number)，不是一个数字，它和任何值执行数学运算都返回 NaN.

'2a' // NaN

true // 1

false //0

undefined //NaN

null //0

②强制将字符串和小数转整型

`parseInt()`

如果要转换的字符串中开头是非数字，则返回 NaN

'm4.5' //NaN

'4.5m' //4

3.14 //3

'2' //2

③强制将字符串转浮点型

parseFloat()

'4.5m' //4.5

'2' //2

④强制将数值和布尔型转字符串（了解）

toString(n) n 表示转为字符串的同时设置的进制。

var num=3;

var str=num.toString();

5.运算符

表达式：由数据或由运算符连接的数据组成的形式。

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，在原来的基础之上加 1

-- 自减，在原来基础之上减 1

var d=c++; 先把 c 的值赋值给 d，c 再执行自增

var e=++c; 先让 c 的值执行自增，然后再把自增的结果赋值给 e

(2) 比较运算符

> < >= <= == (等于) != === (全等于) !==

返回一个布尔型的值

== 只是比较值是否相同，可能产生隐式转换

==== 不仅比较值，还会比较类型（建议使用）

`3>'10' //false` 字符串转为数值

`'3'>'10' //true` 字符串之间比较的是 Unicode 码

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

`3>'10a' //false`

`3<'10a' //false`

`3=='10a' //false`

`NaN` 和任何值比较(`>` `<` `>=` `<=` `==` `=====`)都是 `false`

`NaN==NaN //false`

JavaScriptDay03:

1.运算符

(1)逻辑运算符

`&&` 并且(逻辑与)，关联的两个条件都是 true，结果是 true，否则 false

`||` 或者(逻辑或)，关联的两个条件有一个是 true，结果是 true，否则 false

`!` 取反(逻辑非) 非真为假，非假为真

短路逻辑

`&&` 如果第一个条件为 false，不再执行第二个条件

`||` 如果第一个条件为 true，不再执行第二个条件

查看以下程序是否会报错

```
var num=5;  
  
num>3 && console.log(a);  
  
num<10 || console.log(a);
```

短路逻辑无需关注整体上是 true 还是 false，重点是看第二个表达式是否会执行。

(2)位运算符(了解)

模拟计算机底层运算过程，将 10 进制转成 2 进制然后运算，运算完以后再把结果转回成 10 进制。

1 2 3 4 5 6 7 8

1 10 11 100 101 110 111 1000

& 按位与，上下两位都是 1 结果为 1，否则 0

| 按位或，上下两位含有 1 结果为 1，否则 0

3&5 5|7

011 101

101 111

001 111

(3) 赋值运算符

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

计算赋值，先执行计算，然后再赋值。

(4)三目运算符

一目：由一个运算符连接的一个操作数据或表达式 `++` `--` `!`

二目：由一个运算符连接的两个操作数据或表达式

三目：由两个运算符连接的三个操作数据或表达式

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

2. 浏览器端的函数

`alert()` 弹出警示框

`prompt()` 弹出提示框(输入框)，需要使用变量来保存输入的值，如果什么也不输入点确定返回空字符串('')。如果点击取消返回 `null`

练习：两次弹出提示框，分别输入数字，计算两个数字相加的和，并以警示框的形式弹出。

逻辑结构

什么是程序？

程序=数据+算法

程序的执行方式：顺序执行、选择执行、循环执行

3.流程控制(选择执行)

(1)if语句

```
if(条件表达式){  
    语句 1;  
}
```

语句 2;

以下 5 个数据作为条件表达式出现默认为 false

0 NaN " undefined null

如果 if 后的大括号语句中只有一行语句，则大括号可以省略。

(2)if-else 语句

```
if(条件表达式){
```

语句 1;

```
}else{
```

语句 2;

```
}
```

(3)if-else 嵌套

```
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; //以上所有的条件都是  
    false  
}
```

(4)switch-case 语句

```
switch(表达式){ //表达式通常是一个变量  
    case 值 1: //如果表达式的值为值 1，执行后边的语句  
        语句 1;  
        break; //结束后续语句的执行  
    case 值 n:  
        语句 n;  
        break;  
    default: //如果以上表达式和所有值比较都是 false，执行下边语句
```

```
语句 n+1;  
}  
}
```

JavaScript Day04:

1.switch-case 语句

switch 在拿表达式和 case 后的值比较的时候，使用的是全等于(==)，要求值和类型都满足。

对比 switch-case 和 if-else 嵌套

switch-case 只能使用全等于的比较

if-else 嵌套可以使用各种比较，使用范围更加广泛

switch-case 结构上更为清晰合理，执行效率更高。

2. 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

循环条件：控制循环的次数

循环体：重复执行的相同或相似代码

(1) while 循环

```
while(循环条件){  
    循环体;  
}
```

(2) break

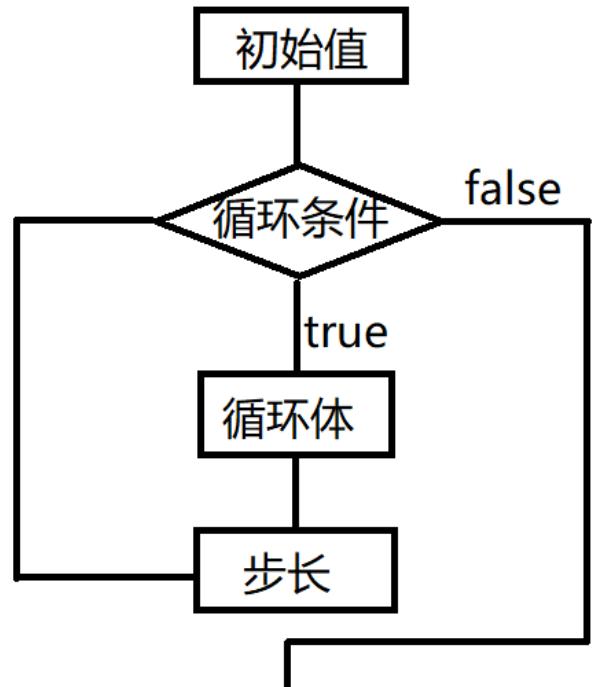
用于结束循环，出现后，就不再执行循环体，同时结束循环。

(3)do-while 循环

```
do{  
    循环体;  
}while(循环条件);
```

(3)for 循环

```
for(初始值;循环条件;步长){  
    循环体;  
}
```



3. continue 和 break

continue 出现后跳过循环体中剩余的内容，继续执行步长

break 结束循环，不再执行任何内容

练习：使用 continue 打印 1~100 之间所有的奇数（遇到偶数就跳过）

4. 循环嵌套

在一个循环体中，又存在其它的循环。

任意两种循环中都可以嵌套

JavaScript Day05:

1. 函数

Number/parselnt/parseFloat/alert/prompt...

函数分为系统函数和自定义函数

function: 功能体、函数、方法... 可以接收若干个数据，返回处理的结果——用于封装重复执行的代码

(1) 创建普通函数

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

调用函数

函数名称()

调用后，就会执行函数体中的代码。

(2) 创建带有参数的函数

```
function 函数名称(参数){ //用于  
接收传递的数据  
    函数体  
}
```

调用

函数名称(参数) //真正传递的数据

参数：创建函数时候的参数称为形参，调用函数时候的参数称为实参，实参会赋值给形参，实参的数量可以使 0 个或者多个，如果形参未赋值则为 undefined

(3) 创建带有返回值的函数

```
function 函数名称(参数){  
    函数体;  
    return 值; //返回值， 返回函数执  
行后的结果  
}
```

调用函数

函数名称(参数)

得到函数返回的结果。

return 用于将函数的执行结果返回，如果函数体中没有 return 或者 return 后不加任何值都返回 undefined。 return 执行后，就不再执行函数体中剩余的代码。

2. 对比 return 和 break

`return` 应用在函数中，一旦出现就会阻止函数体剩余代码的执行

`break` 应用在循环和 switch-case 语句，一旦出现就会结束循环或者 switch-case

3. 变量的作用域

全局作用域：在函数外使用 `var` 声明的变量，可以在任何作用域下访问。

函数作用域：在函数内使用 `var` 声明的变量，只能在该函数内访问。

注释事项：在函数作用域不加 `var` 声明的变量是全局变量。

变量的提升

JS 程序执行前，会将 `var` 声明的变量提升到所在作用域的最前边，但赋值不提升，还是在原来位置。

4. 函数的作用域

全局作用域：在全局下使用 `function` 创建的函数可以在任意作用域下调用

函数作用域：在函数下使用 `function` 创建的函数只能在所在函数作用域调用

函数提升

JS 程序在执行前，会将 function 创建的函数提升到所在作用域的最前边。

斐波那契数列

第 1 项和第 2 项的值是 1，从第 3 项开始，每项的值是前两项相加的和

1 1 2 3 5 8 13 21 ...

JavaScript Day06:

1. 递归

在函数内调用自身，本身是一个死循环。

如何使用递归：需要有跳出的条件，结合着 return。

递归本身就是循环

递归这种算法属于深层次的嵌套，对 CPU 的要求比较高，而 JS 只能使用 CPU 的其中一个内核。

2. 匿名函数

```
function (){ }
```

(1) 创建函数

函数声明

```
function fn(){ }
```

函数表达式，把匿名函数赋给变量，变量名称就是函数名称

```
var fun=function(){ }
```

调用

```
fun();
```

对比函数名称和函数名称()

函数名称()调用函数，得到的是函数的返回结果。

函数名称就是一个变量，保存了一个函数。

对比函数表达式和函数声明的区别

函数声明会将函数整个提升到所在作用域的最前边，可以先调用再创建。

函数表达式是直接通过变量来保存的，只是提升变量声明，必须先创建再调用。

(2)匿名函数自调用

如果变量出现在全局，会存在全局污染；需要放到函数中，这时候就在一个函数作用域下，防止全局污染。但是函数如果有名称，函数名称也是变量，照样存在污染，需要使用没有名称的函数（匿名函数）

需要自调用才会执行函数体代码

```
(function(){  
    //函数作用域下，防止全局污染  
})()
```

```
})();
```

(3)回调函数

就是把另一个函数以实参的形式传递，此时的实参在这就叫回调函数

```
function ran(madai){  
    console.log('然哥开始跑');  
    console.log('然哥到达第一棒的终点');  
    madai();  
}  
  
function dong(){  
}  
  
ran(dong); //传递实参 dong 给形参 madai madai()就会调用 dong 函数  
ran(function(){ })//传递实参匿名函数给形参 madai, madai()就会调用传递匿名函数
```

3.系统函数

parseInt/parseFloat/Number...

2/0

0 做除数返回 Infinity 无穷

isFinite() 检测一个值是否为有限值，如果是返回 true，否则就是 Infinity，返回 false

NaN

isNaN() 检测一个值是否为 NaN，隐式转为数值型，是 NaN 返回 true，不是返回 false

```
console.log('1+2')  console.log( 'typeof 3' );
```

字符串中的表达式无法执行

eval() 执行字符串中的表达式

```
console.log( eval('1+2') ) //3
```

4. 对象

对象属于引用类型数据

什么是对象？

一组属性和方法的集合。

然哥的属性有身高、体重、头发颜色、电话、地址... 方法有养兔子、摊煎饼、交女朋友、玩单杠....

手机的属性有颜色、品牌、大小...方法有打电话、看视频、玩游戏...

万物皆对象

对象是具体到某一个事物，例如然哥是对象，人不是对象。

(1) JS 中的对象

内置对象：JS 提供的对象

宿主对象：根据不同的执行环境划分的

自定义对象：用户创建的对象

(2)自定义对象创建

对象字面量

内置构造函数

自定义构造函数

(3)对象字面量创建对象

使用大括号创建一个空对象，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；属性名中的引号可加可不加，如果含有特殊字符必须加

```
var range={  
    name:'山田一然',  
    sex:'男',  
    'age':46,
```

```
'come@from':'八宝山'  
};
```

(4)访问对象中的属性

对象.属性名

对象['属性名']

如果属性名不存在，则返回 undefined

(5)内置构造函数创建对象

`new Object()` 创建一个空对象，需要单独添加每一个属性。

构造函数，是通过 `new` 来调用的函数，返回一个对象。

JavaScript Day07:

1.遍历属性

循环访问对象中的每一个属性

for-in

```
for(var key in 对象){  
    key //是一个变量，保存每一个属性名  
    对象[key] //通过属性名来获取对应的属性值  
}
```

2.检测对象中的属性

对象.属性名==undefined 存在 false 不存在 true

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

'属性名' in 对象 存在 true 不存在 false

3.对象中的方法

```
var ran={  
    name: '然哥',  
    play: function(){  
        this.name //this指代当前所在的对象  
    },  
};
```

调用方法 ran.**play()**;

4.原始类型和引用类型数据的存储

见文件 05_object.js 和 数据存储.png

5.数组

一组数据的集合，由多个元素组成，每个元素是一个数据。

一组商品、一组会员、一组爱好

数组可以放任何类型的数据，通常是放一组相同的数据。

(1) 数组字面量创建数组

[元素 1, 元素 2, 元素 3]

(2) 访问数组中的元素

数组[下标]

下标是自动分配的一个整数，第 1 个是从 0 开始。

(3) 内置构造函数创建数组

`new Array(元素 1, 元素 2, 元素 3)`

`new Array(3)` 创建数组，初始化长度为 3，可以添加更多个元素。

(4) 数组的长度

数组.length

在数组的末尾添加元素

数组[数组.length] = 值

(5)数组分类

索引数组：以数字作为下标

关联数组：以字符串作为下标，需要单独添加每个元素

(6)遍历数组

for-in

```
for(var key in 数组){  
    key 每个元素的下标  
    数组[key] 下标对应的元素  
}
```

循环

```
for(var i=0;i<数组.length;i++){  
    i 每个元素的下标  
    数组[i] 下标对应的元素  
}
```

只能遍历索引数组

JavaScript Day08:

1. 数组

(1) API

JS 预定义好的一些方法或者函数

toString() 数组转为字符串， 默认按照逗号连接为字符串

join() 数组转为字符串，默认按照逗号连接为字符串，可以按照指定的字符连接为字符串

concat(arr2,arr3..) 拼接多个数组

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；如果 end 为空截取到最后；如果是负数代表倒数。。

splice(start, count, v1,v2..) 删除数组中的元素，start 开始的下标，count 删除的数量，v1，v2...删除后补充的元素。如果 count 为空表示删除到最后，如果 start 为负值表示倒数。返回删除后的元素；原来的数组会发生变化

reverse() 翻转数组中的元素

sort() 对数组排序，默认按照首个字符 Unicode 排序

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

push() 在数组的末尾添加元素，返回数组的长度

pop() 删除数组末尾的一个元素，返回删除的元素

unshift() 在数组的开头添加元素，返回数组的长度

shift() 删除数组开头的一个元素，返回删除的元素

数组中能否再放数组

所有的员工要在一个数组下，对员工进行二次分类，按照部门分

(2)二维数组

数组中的元素还是数组

[[元素 1,元素 2], [元素 1,元素 2]]

访问元素 数组[下标][下标]

2.字符串对象

字符串本身属于是原始类型，属于**包装对象**

包装成对象以后，就具有了属性和方法。

JS 中提供 3 种包装对象：字符串、数值、布尔型

`new String()` 将数据转为字符串型，返回对象

`String()` 将数据转为字符串，返回字符串

对比 `toString` 和 `String`

`toString` 是对象下的一个方法，只能在相应的对象下使用，例如数组，数值型等对象下才可以使用。

`String` 是一个函数，可以将任意的数据转字符串，没有使用限制。

(1) 转义字符 \

转换字符本身的意义

\' 转为普通的引号

\n 转为换行符

\t 转为制表符，tab 键效果

...

练习：打印 hello \n world

(2) API

length 获取字符串的长度

charAt(n) 查找下标对应的字符，也可以使用数组写法 字符串[下标]

练习：遍历字符串，计算 n 出现的次数。

indexOf(str) 查找某个字符串首次出现的下标，找不到返回-1

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

toUpperCase() 英文字母转大写

toLowerCase() 英文字母转小写

slice(start, end) 截取字符串，start 开始的下标，end 结束的下标，如果 end 为空截取到最后，如果是负数表示倒数。

substring(start, end) 截取字符串，start 开始的下标，end 结束的下标，end 为空截取到最后，如果是负数自动转为 0

slice 和 substring 的区别

slice 允许使用负数表示倒数，substring 负数自动转为 0

slice 开始的下标必须小于结束的下标，substring 开始下标可以大于结束下标

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，如果 start 为负值表示倒数

split(str) 将字符串按照指定的字符切割为数组

练习：获取邮箱中的用户名和域名，按照@切割为两端。

range@tedu.cn

JavaScript Day09:

正则表达式

1. 匹配模式(了解)

用于查找、替换字符串

`search(/range/i)` 用于查找某个字符串，返回满足条件的第一个的下标，如果找不到返回-1

/字符串 / i 忽略大小写

`match(/range/ig)` 用于查找某个字符串，返回所有满足条件的字符串，格式为数组。

g 全局查找

`replace(/range/ig, '然哥')` 查找并替换字符串，返回替换后的字符串

2. Math 对象

PI 圆周率

`abs()` 绝对值

`ceil()` 向上取整

floor() 向下取整

round() 四舍五入取整

parseInt() 去除小数点后部分取整

max() 取一组数字最大值

min() 取一组数字最小值

pow(x,y) 计算 x 的 y 次方

random() 取随机数 $>=0 <1$

3.Date 对象

对日期时间的存储和计算

(1)存储

`new Date('2020/4/17 11:35:40')`

`new Date(2020,3,17,11,35,40)` 月份 0~11 对应 1月~12月

`new Date(1000)` 存储的是距离计算机元年的毫秒数，对应的日期时间

`new Date()` 存储的是当前操作系统的时间

(2)获取日期时间信息

`getFullYear/getMonth/getDate/getHours/getMinutes/`

`getSeconds/getMilliseconds/`

月份的值是 0~11 对应 1~12 月

`getDay` 获取星期的值 0~6 对应星期日~星期六

`getTime` 获得距离计算机元年的毫秒数

(3)转为本地字符串格式(了解)

`toLocaleString()` 2020-4-17 15:49:03

`toLocaleDateString()` 2020-4-17

`toLocaleTimeString()` 15:49:03

(4)修改日期时间

`setFullYear()/setMonth()/ setDate()/setHours()/setMinutes()`

`setSeconds()/setMilliseconds()/`

`setTime()` 设置距离计算机元年的毫秒数，会产生一个新的日期时间。

4.Number 对象

包装对象

`new Number()` 构造函数，将数据转为数值，返回对象

`Number()` 普通函数，将数据转为数值，返回数值——推荐用法

`toFixed(n)` 保留小数点后 n 位

`toString(n)` 转为字符串，可以设置进制，n 进制

5.Boolean 对象

`new Boolean()` 构造函数，将数据转为布尔型，返回对象

`Boolean()` 普通函数，将数据转为布尔型，返回布尔型——推荐用法

`toString()` 转为字符串

6.错误处理

`SyntaxError`: 语法错误，程序执行前的检查会出现，任何代码都不执行。

`ReferenceError`: 引用错误，使用了未声明的变量。

`TypeError`: 类型错误，调用的方法或者函数不存在。

`RangeError`: 范围错误，参数的使用超出了范围。

`throw` : 自定义错误

错误处理

```
try{
```

尝试执行，可能会有错误

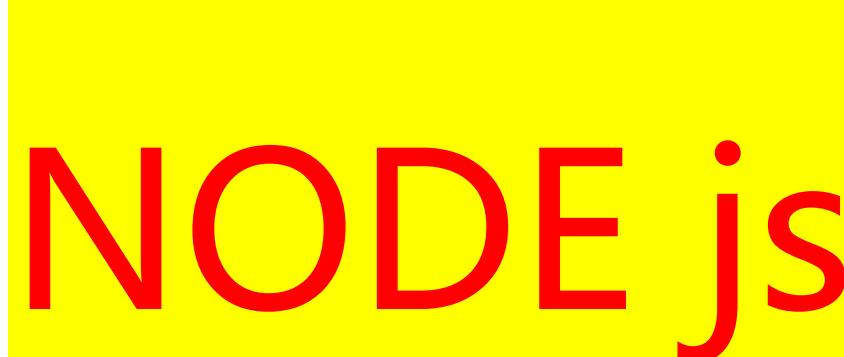
```
}catch(err){
```

一旦 try 中出现错误，则会自动执行这里边的代码

err 是 try 中产生的错误信息

执行相应的处理方案

```
}
```



nodejsDay01:

1.ES6 新特征

ECMAScript6 7 8

ES2016 2017 2018 2019

(1)块级作用域

let 声明的变量不存在变量的提升，不允许 let 反复声明同一个变量；

块级作用域下 let 声明的变量是局部变量

{ }就是块级作用域，还包括 if、else、for、while...下都属于块级作用域

(2)函数增强

可以在函数中的参数设置默认值

```
function fn(a,b,c=0){  
    console.log(a+b+c);  
}  
  
fn(1,2);
```

(3)箭头函数

是匿名函数的另一种写法，两者不完全相同

```
arr.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 return 形式，可以简写为

```
arr.sort( (a,b)=>a-b );
```

(4)模板字符串

解决了字符串的拼接 可换行!!!

```
`字符串 ${js 表达式}`
```

2.Node.js

(1)JS 和 Node.js 的区别

JS 运行在客户端浏览器，存储多个浏览器，代码有兼容性问题；Node.js 只有 V8 引擎一种，代码不存在兼容性问题。

两者有共同的内置对象，自定义对象，不同的宿主对象。

JS 用于开发浏览器端交互效果，Node.js 用于服务器端开发，例如 web 服务器的创建，数据库的访问等。

(2)Node.js 特点

属于单线程处理逻辑，在运行的时候只有一个 CPU 内核在工作

解决数万个并发连接

属于 I/O 密集型，就是输入输出，适合做基于社交网络的大规模的 web 应用

不适合做 CPU 密集型的应用，例如递归嵌套，数据分析，数据挖掘等..

官网 www.nodejs.org

中文镜像 www.nodejs.cn

(3) 使用 Node.js

脚本模式

```
node C:/xampp/.../01.js 回车
```

交互模式

```
node 回车 进入交互模式
```

两次 `ctrl+c` 退出交互模式

5. 全局对象

Node.js: `global`

交互模式下 `var` 声明的变量都属于全局下的变量，可以使用 `global` 访问，例如 `global.a`

脚本模式下 `var` 声明的变量不属于全局下的变量。不能使用 `global` 访问

JS: `window`

在 JS 中 `var` 声明的变量属于全局下的变量，可以使用 `window` 访问，

例如 `window.a`

6. `console` 对象

`console.log()` 输出，打印

`console.info()` 输出消息

`console.warn()` 输出警告

`console.error()` 输出错误

`console.time()` 开始计时

`console.timeEnd()` 结束计时

开始计时和结束计时的字符串要保持一致。

7. process 对象

进程，计算机中每个程序运行的时候，都是一个进程

`process.arch` 查看当前的 CPU 架构

`process.platform` 查看当前的操作系统

`process.version` 查看当前 nodejs 版本号

`process.pid` 查看当前进程的编号，由操作系统自动分配的

`process.kill()` 结束指定编号的进程，或者称为杀死进程

8. Buffer 对象

缓冲区、缓冲器，是内存中临时存储数据的区域，通常存储网络传输时的资源，例如在线视频..

`let buf=Buffer.alloc(5, 'abcde');` 创建 Buffer，设置大小为 5，存储的数据为 abcde，
每个汉字占 3 个字节

`buf.toString()`、`String(buf)` 将 Buffer 数据转字符串

9. 模块

模块是一个独立的功能体

Node.js 下每一个文件是一个模块，每个模块可以引入其它的模块，每个模块也可以被其它的模块所引入

Node.js 自动为每个文件添加了一个构造函数，红色代码就是自动添加的

```
(function (exports, require, module, __filename, __dirname) {  
    //程序员写的代码  
})
```

exports 导出的对象

require() 是一个函数，用于引入其它的模块

nodejsDay02:

1.模块中的参数

require 用于引入其它的模块

exports 导出的对象，module.exports 的别名。

module 当前的模块对象

`module.exports` 导出的对象，和 `exports` 两者指向同一个对象；这个是真正的导出对象

`_filename` 当前模块的绝对路径和模块名称

`_dirname` 当前模块的绝对路径

模块分为自定义模块、核心模块、第三方模块

2. 模块分类

	文件模块	目录模块
以路径开头	<code>require('./circle.js')</code> 常用于引入 自定义模块	<code>require('./02_ran')</code> 会到目录下寻找 <code>package.json</code> 中 main 属性对应的文件，如果找不到 则引入 <code>index.js</code>

不以路径开头	<code>require('querystring')</code> 常用于引入官方提供的 核心模块	<code>require('04_2')</code> 会到当前目录下的 <code>node_modules</code> 中寻找，如果找不到会一直往上一级寻找，直到顶级盘符目录，常用于 引入第三方模块
--------	---	---

3. 包和 npm

包就是一个目录模块，一个第三方模块就可以称作一个包

npm 是用来管理第三方模块的工具，包括下载，上传，更新，卸载....

npm -v 可以查看当前的 npm 版本，按照 nodejs 的时候附带着安装。

www.npmjs.com npm 官网

(1) 切换命令行的目录

`cd 完整路径` 回车

如果涉及到盘符的变化，需要添加

盘符名称：回车

在指定的目录下，按住 shift，单击鼠标右键->在此处打开 powershell 窗口

(2)使用 npm

npm install 包名称 下载安装包；会下载到命令行对应的目录，会将下载的包放到 node_modules 目录下，同时会生成一个 package-lock.json 文件，记录所下载的包的版本号。

4. 查询字符串模块 querystring

查询字符串是浏览器向 web 服务器发请求传递数据的一种方式，位于 URL 中

`http://www.jd.com/search?keyword=单杠&price=4999`

查询字符串模块是用于处理和解析查询字符串的。

`parse()` 将查询字符串解析为对象

练习：获取以下查询字符串中的数据

wd=china&place=beijing

5.URL 模块

URL: 统一资源定位，网络上的任何资源都有对应的 URL

`http://www.codeboy.com:9999/admin/login.html?uname=root`

协议 域名/IP 地址 端口号 文件的路径 查询字符串

URL 模块用于解析和处理 URL 的

`parse()` 将 URL 解析为对象，就可以得到各个部分

练习：获取以下 URL 中查询字符串的数据

`https://www.tmooc.cn:443/web/2003.html?cid=18&cname=nodejs`

步骤 1：先解析 URL 得到查询字符串

步骤 2：再次解析查询字符串，得到数据

6. 定时器模块

是一组全局 API，不需要引入模块

(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；单位是毫秒

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 1 次回调函数；单位是毫秒

清除

```
clearInterval( timer )
```

(3)立即执行定时器

开启

```
let timer=setImmediate( 回调函数 )
```

把回调函数放在事件队列的最前边执行，当主程序执行完就会执行事件对列中的回调函数

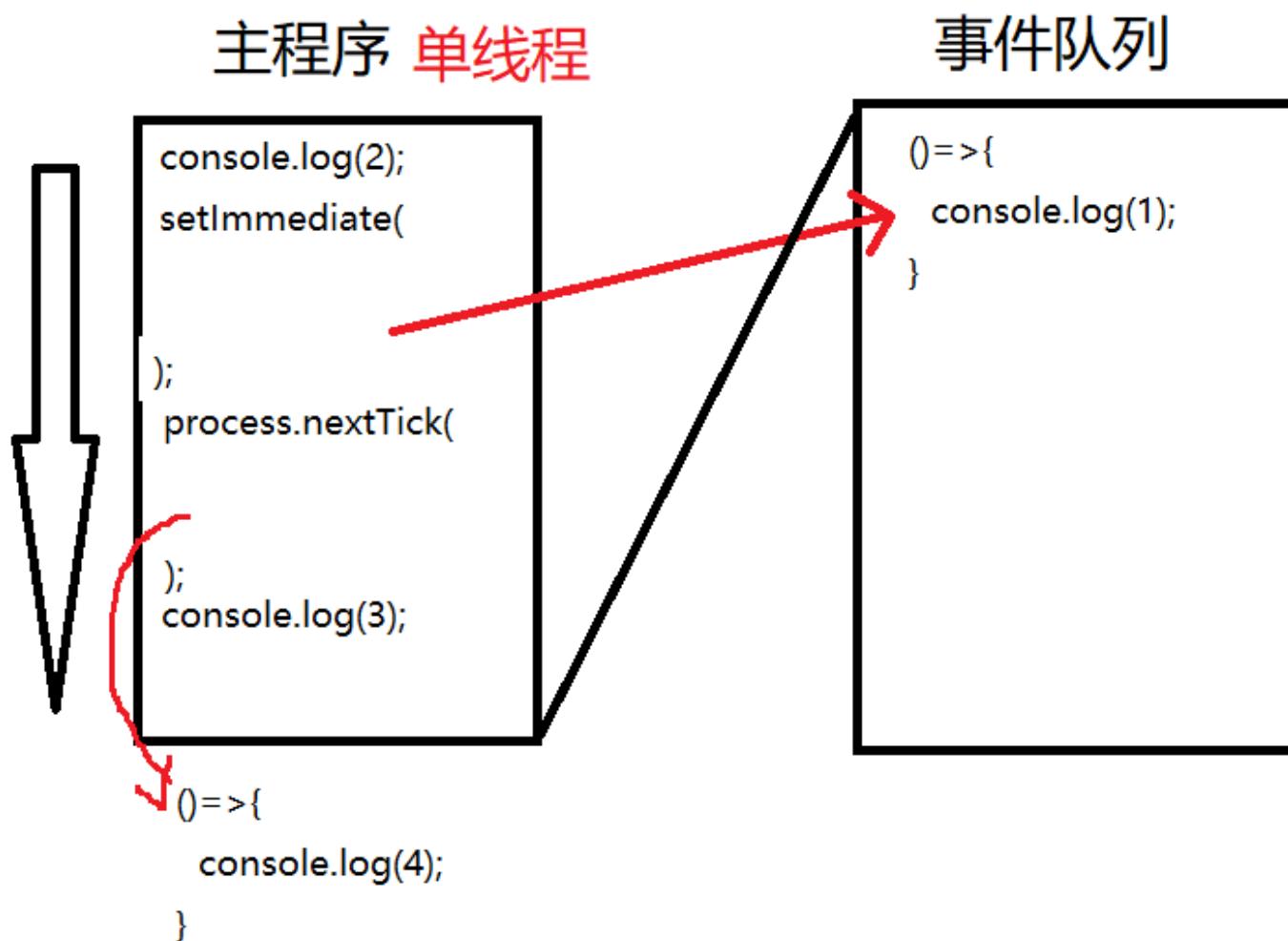
清除

```
clearImmediate(timer)
```

`process.nextTick(回调函数)`

把回调函数放在事件队列之前，主程序的后边执行

事件队列：用于放置一组回调函数，当主程序执行完以后，排队执行。



nodejsDay03:

1.文件系统模块

用于文件的操作；文件包括文件形式和目录形式

(1)查看文件的状态

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

同步

异步

`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

`s` 文件的状态

`isFile()` 是否为文件

`isDirectory()` 是否为目录

(2) 创建目录

`mkdirSync(path)` / `mkdir(path, callback)`

(3) 移除目录

`rmdirSync(path)` / `rmdir(path, callback)`

练习：使用两种方法移除目录 `mydir` 和 `mydir2`

(4) 读取目录

`readdirSync(path)` / `readdir(path, callback)`

`callback`

`err`

result 读取的结果，格式为数组

(5) 创建文件/写入文件

`writeFileSync(path, data) / writeFile(path, data, callback)`

path 文件的路径

data 要写入的数据

如果文件不存在，则会创建文件，然后写入；

如果文件已经存在，则会覆盖内容写入

(6) 追加写入文件

`appendFileSync(path, data) / appendFile(path, data, callback)`

data 要写入的数据

如果文件不存在，则会创建文件，然后写入；

如果文件已经存在，则会在末尾追加写入

(7) 读取文件

`readFile(path, callback) / readFileSync(path)`

`callback`

`err`

`data` 读取到的数据，格式为 `buffer`

(8) 删除文件

`unlink(path, callback) / unlinkSync(path)`

(9) 检测文件(目录)是否存在

`existsSync(path)`

如果文件存在返回 `true`，不存在返回 `false`

练习：如果文件 2.txt 存在，则同步删除该文件；如果目录 mydir 不存在，则创建该目录。

2. 同步和异步

同步：按照顺序执行，执行完前边的代码才会执行后边，会阻止后续代码执行，通过返回值获取结果

异步：在线程池中单独取一个线程执行，不会阻止后续代码的执行，通过回调函数获取结果

3. 文件流

通常用于大文件的读取和拷贝

fs.createReadStream(path) 创建读取的流，path 是文件的路径

```
//创建可读取的流
```

```
let rs=fs.createReadStream('1.rar');
```

```
//事件：一旦读取到数据流，自动触发回调函数，  
//data 事件名称，固定用法  
rs.on('data',(chunk)=>{  
    //chunk 就是获取到的数据，分段的  
})  
  
//事件：一旦读取结束，自动触发回调函数；end 事件名称，固定用法  
rs.on('end',()=>{  
});
```

fs.createWriteStream(path) 创建可写入的流，path 要写入的文件的路径，会自动创建

```
//大文件的拷贝  
//可读取的流  
let rs=fs.createReadStream('1.rar');  
//可写入的流(创建要写入的文件)  
let ws=fs.createWriteStream('2.zip');  
//把读取的流通过管道添加到写入流  
//pipe 管道  
rs.pipe(ws);
```

4.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，表示要向服务器获取哪些内容

Request Method: 请求的方法 get 获取得 to post 传递给别人

Status Code: 响应的状态码

1**: 正在响应，还没有结束

2**: 成功的响应

3**: 响应的重定向，跳转到另一个 URL

4**: 客户端请求错误

5**: 服务器端错误

(2)响应的头信息(response)

Content-Type: 响应的文件类型，例如：html 类型为 text/html

Location: 要跳转的 URL，需要结合状态码 3** 使用

(3)请求头信息

暂无

(4) 请求主体

用于存放传递的数据；只有传递数据的时候才会显示。

5. http 模块

用于创建 web 服务器

```
//引入 http 模块
const http=require('http');

//创建 web 服务器
const app=http.createServer();

//设置端口 8080
app.listen(8080);
```

//web 服务器通过事件接收请求，一旦请求自动触发回调函数

```
app.on('request', (req, res)=>{
```

//req 请求的对象

req.url 请求的 URL

req.method 请求的方法

//res 响应的对象

res.writeHead(状态码, 头信息) 设置响应的状态码和头信息, 头信息可以为空

res.write() 设置响应的内容, 显示到网页中的内容

res.end() 结束并发送响应

```
)
```

nodejsDay04:

1. express 框架

基于 Node.js 平台，快速、开放、极简的 WEB 开发框架

www.expressjs.com.cn 中文版网站

属于第三方模块，需要安装

npm install express

```
const express=require('express');
const app=express(); //创建 web 服务器
app.listen(8080); //设置端口
```

(1) 路由

在 express 中，浏览器向 web 服务器发来请求， web 服务器根据请求的方法和请求 URL 做出响应。只能处理特定的请求

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

 req 请求的对象

 req.method 获取请求的方法

 req.url 获取请求的 URL

req.query 获取查询字符串传递的数据

req.params 获取路由传参的数据

 res 响应的对象

res.send() 设置响应的内容并发送

res.redirect() 响应的重定向，跳转到另一个 URL

res.sendFile() 响应文件，需要使用绝对路径 `_dirname`

});

浏览器传递方式	具体格式	在路由中接收方式
查询字符串	http://127.0.0.1:8080/mysearch?kw=dell	req.query
路由传参	http://127.0.0.1:8080/package/mysql	在 URL 中设置接收的名称 app.get('/package/:pname', ()=>{ req.params // 获取数据，格式为对象 });

post 传递数据	URL 中不可见	在路由中，通过流的方式传递 req.on('data',(chunk)=>{ chunk 就是获取到的数据，分 段的 转字符串后为查询字符串，需 要使用 querystring 解析为对象 });
-----------	----------	--

2. 路由器

实际项目中，不同模块下的路由出现了相同的 URL

用户模块

列表 (get /user/list) 删除(get /user/delete) 修改..

商品模块

列表 (get /product/list) 删除(get /shopping/delete) 修改..

为了管理同一个模块下的路由，防止出现 URL 的冲突，为了团队协作，每个人可以独立开发某个模块下的功能，可以使用路由器来解决。

①创建路由器对象

```
//引入 express  
const express=require('express');  
  
//创建路由器对象  
const router=express.Router();  
  
//添加模块下所有的路由  
router.get('/list',(req,res)=>{  
    res.send('这是用户列表');  
});
```

```
//导出路由器对象  
module.exports=router;
```

②在服务器下引入并挂载

```
const userRouter=require('./user.js');  
  
//挂载并添加前缀  
app.use('/user',userRouter );
```

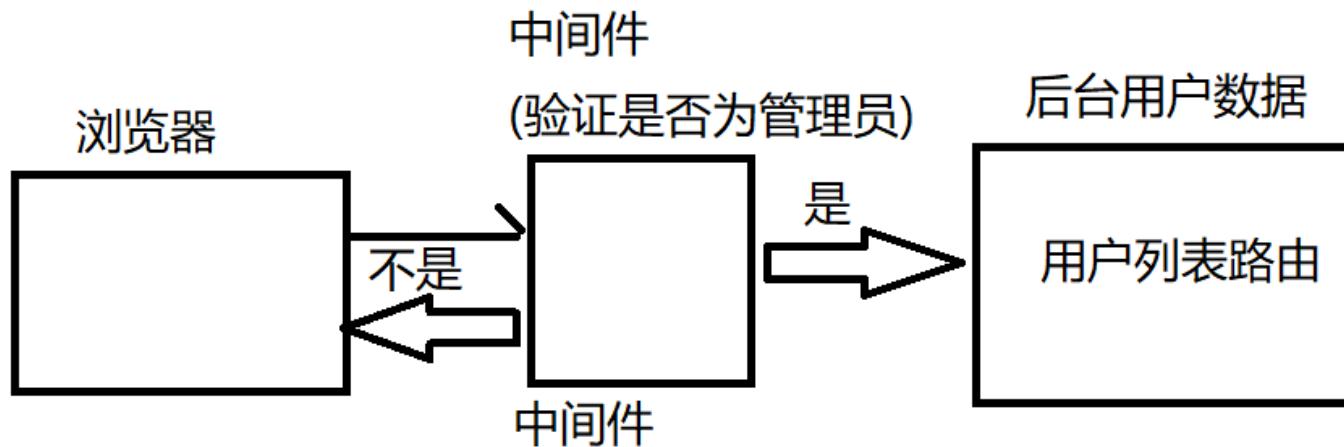
nodejsDay05:

中间件

mysql 模块

1. 中间件

浏览器向服务器发请求(路由)，中间件可以拦截到请求，也可以做出响应，或者往后请求路由。



中间件分为应用级中间件、路由级中间件、内置中间件、第三方中间件、错误处理中间件。

(1) 应用级中间件

也称为自定义中间件，就是一个函数

```
app.use( '/list', (req,res,next)=>{  
    next() //是一个函数， 表示执行下一个中间件或路由  
});
```

一旦请求到 URL， 自动执行回调函数。

练习：创建购物车路由（get /shopping）,传递商品的单价 price，在中间件中实现对商品打九折；最后在路由中响应'商品的最终价格为：xxx'



(2) 路由级中间件

路由器的使用

(3) 内置中间件

只有一个内置中间件

托管静态资源：如果浏览器端要请求静态资源(html, css, js, 图像)，**不需要使用路由响应文件**，而是自动去寻找所需要的文件。

app.use(express.static('要托管的目录'));

练习：创建 web 服务器，托管静态资源到 public 目录下，在目录下创建注册文件 reg.html，点击提交，向服务器发送请求(post /myreg)；在路由中获取数据
04_three.js

(uname, upwd, email, phone)

用户	
密码	
邮箱	
电话	
提交	

(4)第三方中间件

需要下载安装并引入到所在的 web 服务器

以 body-parser 为例

```
//1.引入 body-parser 中间件(第三方模块)
```

```
const bodyParser=require('body-parser');
```

```
//2.应用中间件
```

```
//urlencoded，可以将 post 请求的数据解析为对象，内部也会使用查询字符串模块
```

```
//extended:false 不使用扩展的第三方的 qs 模块，而是使用核心模块 querystring  
app.use( bodyParser.urlencoded({  
    extended:false  
}) );  
  
//3.在路由中获取 post 请求的数据，格式为对象  
req.body
```

2.mysql 模块

在 nodejs 下用于操作数据库

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

```
use 数据库名称;
```

```
INSERT INTO emp VALUES(...);
```

```
DELETE FROM emp WHERE eid=3;
```

```
UPDATE emp SET phone='1311',email='a@qq.com' WHERE eid=1;
```

```
SELECT * FROM emp;
```

先下载并引入 mysql 第三方模块

(1)建立普通连接

```
const c=mysql.createConnection({  
    host:'127.0.0.1',//IP 地址  
    port:'3306', //端口  
    user:'root', //用户名  
    password:"", //密码  
    database:'tedu' //连接后进入的数据库  
});
```

(2)执行 SQL 语句

c.query(sql, callback)

sql 要执行的 SQL 语句

callback 用于获取执行的结果

err 可能产生的错误

result 具体的执行结果

(3) 使用连接池 (推荐连接方式)

只需要创建有限的连接个数，供所有的访问使用。

```
const pool=mysql.createPool({  
    host:'127.0.0.1',  
    port:'3306',  
    user:'root',  
    password:"",  
    database:'tedu',  
    connectionLimit:20 //连接池的大小, 默认 15 个  
});
```

在让用户提供查询条件的时候，结果用户拼接了 SQL 命令，造成了数据暴露；——称为 SQL 注入

SELECT * FROM emp WHERE eid=5 OR 7=7;

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

如何避免

对用户提供的值进行过滤。

mysql 模块提供了占位符(?)对用户提供的值进行过滤。

nodejsDay06:

学子商城后端接口

接口：后端给前端提供的数据，前端向数据库发出的请求都是通过接口来完成，例如登陆、注册、用户列表、删除用户、修改用户..



练习：在 xz 目录下创建 web 服务器文件 app.js，创建 web 服务器，托管静态资源到 public 目录下，包含文件 user_reg.html
(uname, upwd, email, phone)，点击提交 (post /user/reg)

练习：在用户路由器模块 user.js 下，创建路由器对象，添加用户注册的路由(post /reg)，响应‘注册成功’；最后导出路由器；在服务器 app.js 下引入该路由器(`require('./router/user.js')`)，并挂载，添加前缀 /user

练习：在 web 服务器 app.js 下应用 body-parser 中间件，在路由获取 post 请求的数据 `req.body`

练习：在 xz 目录下创建自定义模块 pool.js，引入 mysql 模块，创建连接池对象并导出；在路由器 user.js 下引入该模块(`require('../pool.js')`)，执行 SQL 语句，将用户的数据插入到数据库 xz 下 xz_user 表中。

练习：在 public 下创建用户登陆的网页 login.html

在public下创建用户登陆的网页login.html 点击提交 post /user/login

用户名	<input type="text"/>
密码	<input type="password"/>
提交	

- 创建对应的路由
1.获取数据
2.验证是否为空
3.执行SQL语句

创建修改用户资料的public下 user_update.html

编号	<input type="text"/>
邮箱	<input type="text"/>
电话	<input type="text"/>
姓名	<input type="text"/> user_name
性别	<input type="text"/> gender
提交	

- 点击提交 请求(get /user/update)
创建对应路由
1.获取数据 (req.query)
2.验证是否为空 (批量验证)
3.执行SQL语句 (修改)

分页查询

当前的页码、每页的数据量

```
SELECT * FROM xz_user LIMIT start, count;
```

start 开始查询的值 = (当前页码-1)*每页数据量

count 每页的数据量

start 和 count 不能是字符串型，必须数值型

```
SELECT * FROM xz_user LIMIT 0, '2';
```

在public目录下创建用户列表的路由user_list.html

当前页码	<input type="text"/>	pno
每页数据量	<input type="text"/>	count
<input type="button" value="提交"/>		

点击提交 get /user/list

创建路由

1. 获取数据
2. 如果页码为空默认为1，如果每页数据量为空默认为2
3. 计算开始查询的值
4. 执行SQL语句

在public下创建文件user_detail.html

编号	<input type="text"/>	<input type="button" value="提交"/>
----	----------------------	-----------------------------------

点击提交 /user/detail get

创建对应的路由

1. 获取数据
2. 验证为空
3. 执行SQL语句

在public下创建文件user_delete.html

编号	<input type="text"/>	<input type="button" value="提交"/>
----	----------------------	-----------------------------------

点击提交 /user/delete get

创建对应的路由

1. 获取数据
2. 验证为空
3. 执行SQL语句

<D:\第一阶段 复习用\NODEJS\接口说明文档.docx>



HTMLDay01:

day01

自我介绍

WX: Ran_Lee0210

一.第二阶段课程安排 (20 天)

1.HTML5 Basic(2 天)

HTML5，制作页面的标签

搭建网页结构，完成网页的骨架和
内容

2.ajax 异步的数据交互 (3 天)

Ajax一共4步

js 代码，异步发送请求，访问服务
器，接收响应数据

3.ajax 项目 (2 天)

4.CSS3 (4+2)

5.Bootstrap 框架 (4 天)

6.boot 项目 (3 天)

HTML

一.html 的基础知识

1.html 的历史

早期，html 泛指前端网页技术

2014.9 发布了 html5---->大前端技术

html5 是 html4 和 XHTML1.0 版本的升级

<input type="text"> <input type="text" />

html5 升级之后，单标签，带不带/都行

或

2.web 与 internet

internet：全球性计算机互联网，俗称：互联网，因特网，交际网

www 服务：world wide web 万维网服务，网站服务

3.internet 上的应用程序

1.C/S 结构

c:client 客户端

s:server 服务器

代表: qq, app 游戏

2.B/S 结构

b:browser 浏览器

s: server 服务器

代表: 所有的网站, 页游

3.c/s 和 b/s 的区别

① c/s 需要下载安装客户端, b/s 不需要

② c/s 需要更新, 停机更新。b/s 的更新不会影响用户体验

③我们的工作主要是 b/s

3.web 运行原理

web：运行在 internet 上只用 b/s 结构的应用程序-----俗称网站

internet:为 web 运行提供了网络环境

web 的运行原理:

基于**浏览器和服务器**以及**通信协议**来实现数据的传输和展示

①通信协议：规范了数据是如何打包和传递的，让浏览器和服务器都认识这个数据

http https 浏览网页

ftp 上传下载

file 本地文件协议

②服务器

a.功能：

存储数据

接收浏览器请求，并给出响应

提供程序的运行环境

具备一定的安全性

b.产品

1.apache (php)

2.Tomcat (java)

3.IIS (微软 .net iis sqlserver)

c.技术

1.php

2.java

3..net

4.python

5.nodejs

③浏览器

a.功能

发送请求

接收响应，把代码解析成图形页面

b.产品

chrome

safari

firefox

opera

IE--edge

c.技术

Html5 css3 javascript

二.HTML 快速入门

1.什么是 HTML

HTML 是什么

HyperText Markup Language

超文本 标记 语言

<关键字></关键字>

标记--->尖括号

语言：有语法

2.HTML 的特点

1.以.html 或者.htm 为后缀

2..html 保存在服务器，由浏览器解析执行

3.使用带有<>的标记去标识

4.在网页可以执行js 脚本

3.HTML 的基础语法

1.标记，标签，元素，对象，节点

把关键字放到尖括号中，让其具备一定的功能

标记的分类

1.双标记（封闭类型标记）

<关键字> 内容区域 </关键字>

2.单标记（非封闭类型，空标记）

<关键字> 或者 <关键字/>

2. 嵌套

在双标记的内容区域中，编写其他标签，让多个标签的功能进行叠加

<关键字>

<关键字 1></关键字 1>

</关键字>

① 注意嵌套顺序

② 注意正确的缩进

3. 标签的属性和值

属性和值是用来修饰元素的，属性的值，推荐使用双引号包裹

<关键字 属性 1="值 1" 属性 2="值 2".....> </关键字>

属性的分类

①标准属性，所有元素都有的属性

`id`: 定义元素在页面中唯一标识

`title`: 鼠标悬停在元素上，显示的文本

`style`: CSS 中，定义内联样式属性

`class`: CSS 中，引用类选择器样式的属性

②专属属性，只对某元素或者某些元素有效的属性

附加知识点

语义错误：不影响代码运行结果的错误

语法错误：代码直接报错

4.注释

代码中，不被解析运行的文本

```
<!-- 我是注释 -->
```

注意

1.注释中不能嵌套注释

2.标签的<>内不能写注释

三.HTML 文档结构详解

1.html 的文档构成

①文档声明

```
<!DOCTYPE html> 告诉浏览器，我当前页面代码是使用 h5 规则编写，请使用 h5 的规则解析
```

②网页结构

<html> 代表网页的开始和结束，一个页面中，有且只有一对 html 标签

<head></head> 网页的头部，定义网页全局信息

<body></body> 网页的主体，所有要显示在页面中的东西，都要写在 body 中

</html>

③ <head></head> 中的设置

<title></title> 网页标题

<meta> 定义全局信息，元数据

<meta charset="utf-8">

<meta name="description" content="网页描述"/>

<meta name="Keywords" content="网页关键字"/>

<style></style> 定义网页内部样式 css 内容

<link> 引用外部样式 css 外部

<script></script> 定义或者引用js文件

④ body 标签

页面的主体，浏览器可视区域的部分

属性： bgcolor="black" 背景颜色

text="red" 文本颜色

⑤学习 html 的技巧

<关键字 属性 1="值" 属性 2="值"></关键字>

学习 html，就是学习固定的关键字和功能

和固定的属性以及对应的效果

四.文本标签

1.标题标签

```
<h1>1号标题</h1>
```

```
<h2>2号标题</h2>
```

```
<h3>3号标题</h3>
```

```
<h4>4号标题</h4>
```

```
<h5>5号标题</h5>
```

```
<h6>6号标题</h6>
```

功能：字体加粗，独占一行，字号改变(h1最大，h6最小)

上下有垂直的空白间距

属性：align 设置标签中文本的水平对齐方式

取值：left(默认，缺省)、center、right

2.段落标签

```
<p></p>
```

功能：独占一行，上下有垂直间距，是专门盛放文本的标记

属性：align 取值：left(默认，缺省)、center、right

3.换行标记

空格折叠现象，html 被运行的时候，会把所有的空格和回车解析成一个空格显示

换行
或者

4.特殊字符

空格 &nbsp

< <

> >

× ×

® ®

© ©;

¥ ¥;

HTMLDay02:

day02

四.文本标签

5.水平线

<hr> 或者 <hr/>

属性：

size="10" 设置水平线的粗细，如果不写单位默认是px

width="50%" 设置水平线的宽度，

取值：以px为单位的数字

% 父元素宽度的%

color="pink" 设置水平线的颜色

align="left" 设置水平线的水平对齐方式 left/center(默认值)/right

6. 预格式化标签

html 有空格折叠现象

但是写在<pre></pre>中的所有空格和回车，都会保留下来

<pre></pre>

7. 文本样式标签

<i></i> 斜体

 粗体

<s></s> 删除线

<u></u> 下划线

H5 建议使用有语言意义的标签，替代没有语言意义的标签

标签带语义的好处，在被搜索的时候，有语义的标签更容易被找到

 上标

 下标

文本样式标签的特点，共用一行

8. 分区元素

① 块分区

<div></div> 是用来做页面结构布局
独占一行

②行分区

 同一行文本，有不同样式的的时候，用 span，把不同样式的文本括起来
与其他行内元素共用一行

9. 元素默认的显示方式

块级元素	行内元素	行内块元素	table
单独成行	与其他行内元素和行内块共用一行	与其他行内元素和行内块共用一行	一种非常特殊的显示方式

h1~h6 p div hr pre	span i em b strong...	input	
支持 align 属性	不支持 align 属性	不支持 align 属性	

五.图像和链接

1.img 的使用

```
<img src=""> 单标签  
属性 src="图片资源路径/url"
```

2.URL

Uniform Resource Locator 统一资源定位符，俗称资源路径

3.URL 的表现方式

①绝对路径

完整的路径

绝对路径用于使用网络资源

使用网络资源的优点，不占用本地服务器的存储空间

缺点，资源不稳定

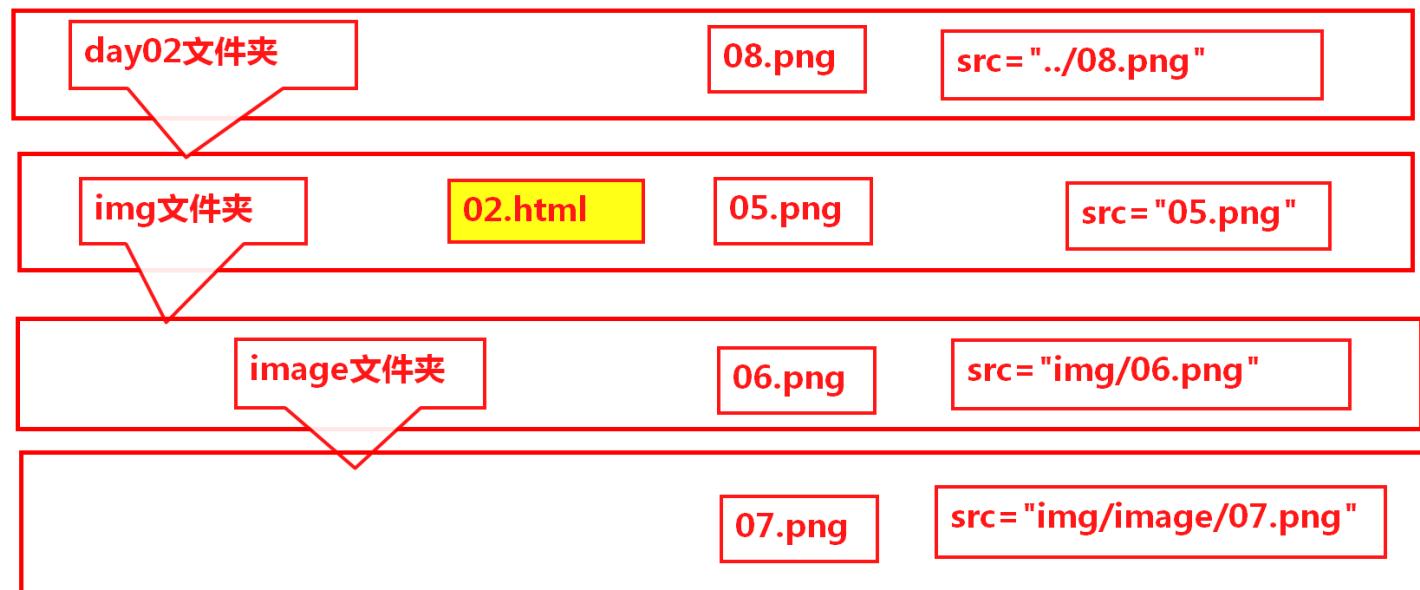
为了支持网络资源的优点，屏蔽缺点----->图床

本地资源可以使用绝对路径，但是项目中不允许

②相对路径

本服务器资源，使用相对路径

1. 兄弟关系，直接写兄弟名称 `src="05.png"`
2. 兄弟的儿子，先写兄弟名称，用/进入兄弟，写兄弟儿子的名称 `src="img/06.png"`
3. 父亲的兄弟，使用../进入父级，在选中父级的东西 `src="../08.png"`



4. img 的其他属性

`src=""` 图片资源的 url

`alt=""` 图片加载失败时，显示的文本

`title=""` 鼠标悬停时显示的文本

`width=""` 设置图片的宽高

`height=""`

注意，如果设置的宽高比，与图片默认宽高比不同，图片会发生失真

解决方案：宽高只设置一个，另外一个会自动适应

5.链接

`<a>`

`<a href>`

没有 `href` 属性，`a` 标签的功能失效

属性 `href=""` 指向要跳转的资源

`target="" _self` 默认缺省值，在本页面打开新链接

`_blank` 新开选项卡页面，打开链接

6.a 标签的其他表现形式

1. 下载压缩包

```
<a href="1.zip">下载</a>
```

2. 打开系统自带写邮件的软件

```
<a href="mailto:邮箱地址">写邮件</a>
```

3. 使用 a 标签调用 js 代码

```
<a href="javascript:方法名称()">调用 js 代码</a>
```

4. 回到页面顶部

```
<a href="#">to top</a>
```

注意，在做项目的时候，`href` 如果不知道跳转到哪里，可以使用#占位

尽量不要出现空的 href=""

7.锚点

在页面上，设定锚点，不管页面滚动到哪里，点击 a 标签，页面就跳转回锚点位置

①设置锚点

任意元素的 id 值，都可以当做锚点

<any id="id 值">

②跳转到锚点

就可以跳转到锚点

注意，锚点会在页面的地址栏显示

练习

02_img.html 中添加一个锚点

如何在 03_flag.html 中的 a 跳转去 02_img 的锚点

六.表格

表格本意，是有规律的显示数据

但是 10 年之前，table 被广泛的用于布局

由于表格布局，简单，可是 table 加载效率低，现在被淘汰了

现在布局，不使用 table。只有在学习的时候，使用 table 布局

1.表格语法

```
<table>  
    <tr>    table row  
        <td></td>  table data  
        .....  
    </tr>
```

```
</table>
```

练习，完成一个 4*4 的表格，要求 td 中填写 1~16

2. 属性

①. table 的属性

border="1" 添加边框

width="200px" 设置宽高

height="200px"

bordercolor="purple" 设置边框颜色

bgcolor="pink" 设置背景颜色

align="center" 表格本身的水平对齐方式 left/center/right

(块级元素的 align 属性，是让元素内部的文本水平居中)

cellpadding="20px" 设置单元格内边距(边框到内容之间的距离)

`cellspacing="20px"` 设置单元格外边距(边框到边框之间的距离)

② tr 的属性

`align="right"` 设置内容的水平对齐方式 left/center/right

`valign="bottom"` 设置内容的垂直对齐方式 top/middle/bottom

`bgcolor="yellow"`

③ td/th 的属性

`width="200"`

`height="200"`

`align="right"`

`valign="top"`

`bgcolor="red"`

3.table 显示方式的特点

所有表格都有特点

- 1.默认每一行列数相同
- 2.不同行的同一列，宽度统一
- 3.同一行所有列，高度统一

html 的 table 的特点

- 1.table 的尺寸比较特殊

如果设置的宽高比较大，内容比较少，最后的尺寸以设置的为准

如果设置的宽高比较小，内容比较多，最后的尺寸以内容为准

- 2.表格的渲染方式

读取整个表格，放到内存中，再从内存中取出，渲染到页面上

所以，**表格的渲染效率非常低**

- 4.不规则的表格

列合并 colspan="n" 在当前列向右合并 n 列, n 包含自己

要把被合并的列删除

行合并 rowspan="n" 在当前单元格向下合并 n 个单元格, n 包含自己

要把被合并的单元格删除

5. 表格的可选标记

1. 表格的标题

<caption></caption> 要求紧紧挨着 table 标签写, 写在 tr 外层

2. 行/列的标题

<th></th> 替代 td 使用, 有文本加粗居中的效果

6. 表格的复杂应用

① 行分组---页面中不可见

<thead></thead> 表头

<tbody></tbody> 表主体

<tfoot></tfoot> 表脚

如果表格没有做分组，浏览器默认添加一个 tbody，把所有 tr 放入 tbody 中

②表格嵌套

表格可以嵌套其他任意元素

但是所有嵌套在表格中的内容，只能放在 td/th 中

HTML Day03:

day03

六.表格

七.列表

早期列表就是用来有条理的显示数据

现在无序列表经常用来布局

1.列表的组成

①列表类型

有序列表

 order list

无序列表

 unorder list

②列表项

 list item

2.列表的属性

1.有序列表的属性

type="" 设置列表项的类型

1 (默认) a A l i

start="" 指定起始编号 默认值 1

2.无序列表的属性

type=""设置列表项的类型

disc 默认值 实心小黑圆

circle 空心圆

square 小方块

none 去除标识

3.列表的嵌套

列表中，可以嵌套任意元素(包括列表)

但是，所有被嵌套的内容，必须放在li 中

ol/ul 的直接子元素只能是 li (语义要求)

4. 定义列表

对一个名词或者事物的解释说明，可以使用定义列表

<dl> 定义列表

<dt></dt> 要解释说明的名称

<dd></dd> 解释说明的内容

</dl>

八. 结构标记

结构标记跟 div 一模一样，只不过关键字不同

带有语义，增强代码的可读性，

另外在有语义的元素中，编写文本，

会优先被搜索到

常用的结构标记

1.<header></header>

定义网页的头部，或者某个区域的顶部

2.<footer></footer>

定义网页的脚部，或者某个区域的底部

3.<nav></nav>

定义页面的导航栏

4.<aside></aside>

定义页面侧边栏

5.<section></section>

定义网页主体

6.<article></article>

定义与文本相关的内容，评论，回复

九.表单(重点*****)

1.作用

- 1.提供了可视化的输入控件
- 2.自动的收集用户输入的信息，并发送请求给服务器

注意：form 自带发送请求的功能

ajax 不使用 form 表单

2.表单 <form></form>

属性

① action=""

定义表单提交时发生动作，要把请求提交给哪个接口

如果没有值，默认提交给本页面

② method=""

定义表单的提交方式，设置请求方法

get 默认 特点：明文提交，提交的信息会在地址栏显示

提交数据有大小限制，最大 2kb

向服务器要数据的时候，使用 get

post 特点：隐式提交，提交的信息不会在地址栏显示，会在请求主体中显示

数据提交没有大小限制

给服务器传递大量数据的时候，使用 post

delete/put/option

③ enctype=""

设置表单数据的编码方式

取值

application/x-www-form-urlencoded 默认值，允许将任意字符提交给服务器

text/plain 允许提交普通字符给服务器

任意字符，所有字符

普通字符，不能带符号 = & ^ % \$ #....都不行

multipart/form-data 允许提交文件给服务器

3.表单控件，在 form 表单中，能够与用户进行数据交互的可视化元素

①分类

input 元素 基础的 9 种，h5 新 input10 种

textarea 多行文本域

select/option 下拉选择框

其他元素元素

② input 基础 9 种元素

<input/> 或者 <input>

所有 input 元素都有的属性

1.type="" 设置 input 元素的具体类型，默认缺省值是文本框

text

2.name="" 表单提交请求，没有 name 属性的元素，不会被收集数据，也不会被提交

在 ajax 中，不需要 form 表单，那么就失去表单自动收集数据发送请求的能力

ajax 使用 input 的时候，不需要 name 属性了

在使用单选和多选按钮时，需要用 name 属性来分组

3.value="" 表单提交请求时，某个控件真正提交的值

特例：所有按钮的 value 不会被提交，只是修改按钮显示的文本

4.disabled 禁用 无值属性 控件的 value 只能看不能改，不能提交

③ input 详解

`type="text"` 文本框

`type="password"` 密码框

属性

`maxlength="3"` 设置数据的最大长度

`readonly` 只读，无值属性，只能看不能改，但是可以提交

`placeholder=""` 占位提示符

`type="submit"` 提交按钮，将当前表单中的数据，自动收集整理，并发送给服务器

`type="reset"` 重置按钮，将当前表单中的数据，恢复到初始化状态

`type="button"` 没有任何功能，就是通过事件调用js

h5 新出的按钮标签，增加了代码的可读性

`<button></button>` 功能同 submit

单选按钮

`type="radio"`

多选按钮

type="checkbox"

属性

name 属性有两个功能， 1.提交数据

2.为控件分组， 实现单选

value 必须写， 不然提交时候， 不管选中谁都是 on

checked 默认选中

隐藏域

type="hidden"

用于， 想把数据提交给服务器， 又不想被用户看到

文件选择框 ---上传文件

type="file"

前提

form 的 method="post"

enctype="multipart/form-data"

属性 multiple 无值属性，选取多个文件

④ textarea 多行文本域

```
<textarea name="addr" cols="5" rows="3"></textarea>
```

cols="5" 初始显示时，限制一行显示 5 个字符

rows="3" 限制显示三行

这个设置不准确，会根据硬件不同而改变

⑤ 下拉选择框(下拉选)

```
<select name="city">  
    <option>北京</option>  
    .....  
</select>
```

`select` 的属性

`name` 控件的名称

`size` 显示 `option` 数量，默认值为 1，取值大于 1，变为滚动选择框

`multiple` 多选

`value` 1.如果 `option` 没有 `value`, `select` 的 `value` 是被选中的 `option` 的内容区域的文本

2.如果 `option` 有 `value`,`select` 的 `value` 是被选中的 `option` 的 `value` 值

`option` 的属性

`value`

`selected` 下拉选择框的默认选中

nodejsDay04:

day04

九.表单(重点*****)

3.表单控件，在 form 表单中，能够与用户进行数据交互的可视化元素

⑥其他表单元素

1.label

功能：可以在 form 中替代 span 使用（不强制）

可以与表单控件进行关联

```
<label for="关联控件的 id"></label>
```

2.控件分组

<fieldset> 为控件分组

```
<legend>分组标题</legend>
```

```
</fieldset>
```

3. 浮动框架

在一个 html 中引入其他 html

```
<iframe></iframe>
```

属性

src="要引入的 html 的 url"

width="100%"

height="1000px"

frameborder="0" 去除边框

4. H5 新表单元素 10 个

① 邮箱

```
<input type="email" name="mail">
```

提交的时候，验证邮箱格式，格式规则是@前后有内容

②搜索类型

```
<input type="search" name="s1">
```

自带快速删除的小叉叉

③ url 类型

```
<input type="url" name="u1">
```

验证有没有 http://, 以后 http://后面带字符

④电话号码类型

在移动设备上，文本框获取焦点，自动弹出虚拟键盘

5.数字类型

```
<input type="number" name="n1" >  
max="20" 最大值  
min="10" 最小值  
step="3" 步长
```

6.范围类型

提供一个选择数据范围的滑块

```
<input type="range" name="r1" >  
max="20" min="10" step="3"
```

7.颜色类型

提供一个取色器

```
<input type="color" name="c1">
```

8.日期类型

```
<input type="date" name="d1">
```

9.月份类型

```
<input type="month" name="m1">
```

10.星期类型

```
<input type="week"  
name="w1">
```



AJAXDay01:

一.http 协议

1.url

url 结构：协议+主机名称+端口号+目录结构+文件名称

url 的完整结构

<scheme>://<user>:<pwd>@<host>:<port>/<path>;<params>?<query>#<flag>

① scheme

方案，协议，以哪种方式获取服务器的资源

不区分大小写

常见的协议

常见协议	默认端口号	协议基本作用
FTP	21	文件上传、下载
SSH	22	安全的远程登录
TELNET	23	远程登录
SMTP	25	邮件传输
DNS	53	域名解析
HTTP	80	超文本传输
POP3	110	邮件接收
HTTPS	443	加密传输的HTTPS

②<user>:<pwd>

后台管理的登录，要把账号和密码写在 url 中
现在这种写法基本没人用了

③<host>

主机名称 域名 www.baidu.com ip 101.111.98.12

④<port>

端口号，就是计算机上软件对外提供服务器的柜台号

⑤<path>

路径，资源在服务器上具体的存放位置

⑥<params>

参数，跟踪状态的参数 session/cookie (必须会的知识点)

⑦<query>

查询字符串

⑧<flag>

锚点

2.http 协议

HyperText Transfer Protocol 超文本传输协议

规范了数据是如何打包和传递的(专门用来传输 html 文件)

http 的历史

HTTP 协议历史与标准

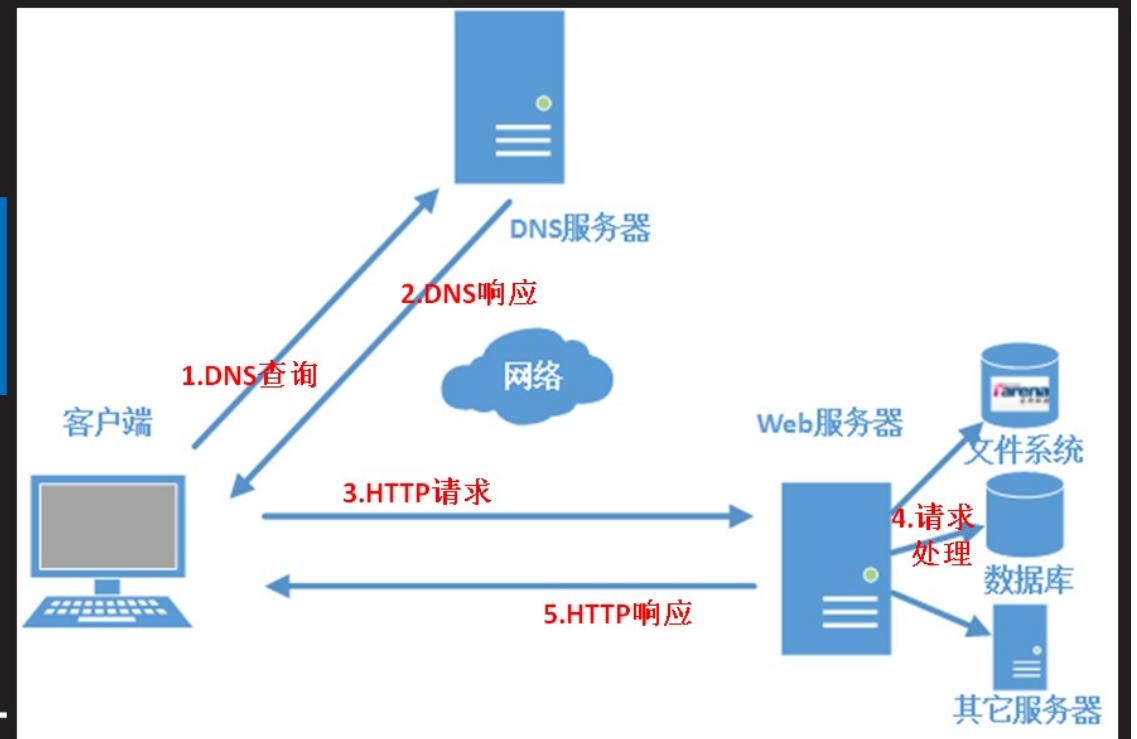
Tare
达

- **HTTP/0.9** : 1991年制定，有严重设计缺陷，只支持GET方法，不支持MIME类型，很快被HTTP/1.0取代。
- **HTTP/1.0** : 1996年制定，支持多种请求方法，支持多媒体对象，得到广泛应用。
- **HTTP/1.0+** : 支持持久连接、虚拟主机、代理连接等新特性，成为非官方的事实标准。
- **HTTP/1.1** : 1999年制定，校正HTTP中的设计缺陷，性能优化，删除一些不好的特性。
- **HTTP-NG (或HTTP/2.0)** : 关注HTTP协议的性能优化以及更强大的服务逻辑远程执行框架，研究工作仍在进行中。

3 完整的 web 请求原理

请求与响应流程

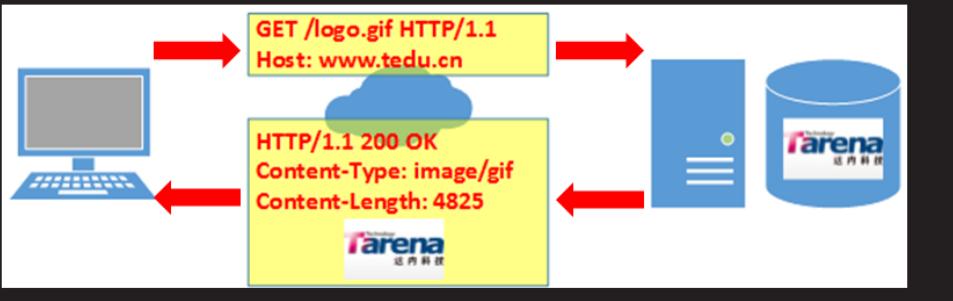
Iarena
达内科技



4. 报文/消息 Message

请求与响应流程 (续1)

- Message , 消息/报文 , 是在HTTP客户端与服务器间传递的数据块。
- HTTP协议规定 , 消息必须符合**特定的格式**才能被彼此理解。
- Request Message : 客户端向服务器发送的请求消息。
- Response Message : 服务器端根据客户端的请求消息 , 返回给客户端的响应消息。



1. 请求消息 Request Message (请求起始行, 请求头, 请求主体)

2. 响应消息 Response Message(响应起始行, 响应头, 响应主体)

5. Request Message

① 请求起始行

请求方法	
http 原生的请求方法	restful Api 请求规则
get 上限 2kb,查询字符串， 向服务器要数据的时候用 req.query 接收	get 无请求主体， 所有的查询模块 select req.params 接收
post 隐式提交（请求主体） , 给服务器发送数据的时候 req.body 接收	post 有请求主体 所有的新增模块 insert req.body 接收
put 有请求主体 往服务器上存放资源的时候使用 req.body 接收	put 有请求主体 所有的更新模块 update req.body 接收

`delete` 查询字符串

从服务器上删除资源的时候使用

`req.query` 接收

`option` 预请求

在使用第三方支付接口

`delete` 没有请求主体

所有的删除模块 `delete`

`req.params` 接收

http 协议版本 1.1

请求的 url

②请求头

1.Host: www.tmooc.cn

浏览器告诉服务器，本浏览器要请求的主机名称

2.Connection: keep-alive

浏览器告诉服务器，请开启持久连接

3.User-Agent: Mozilla/5.0

浏览器告诉服务器，当前浏览器的信息和版本号

4.Accept-Encoding: gzip, deflate

浏览器告诉服务器，我这个浏览器可以接收的压缩文件的格式

5.Accept-Language: zh-CN,zh;q=0.9,en;q=0.8

浏览器告诉服务器，我这个浏览器可以接收的自然语言

6.Referer: <http://www.tmooc.cn/>

浏览器告诉服务器，当前请求来自于哪个页面

③请求主体

formdata

6.Response Message

①响应起始行

1.http 版本号

2.响应状态码

200 成功 404 请求资源找不到 500 服务器代码错误

3.原因短句

②响应头

1.Date: Thu, 30 Apr 2020 07:27:56 GMT

服务器告诉浏览器，具体的响应时间

2.Connection: keep-alive

服务器告诉浏览器，已经开启了持久连接

3.Content-Type: text/html

服务器告诉浏览器，响应主体的类型

text/html html 文件

text/css css 文件

text/plain 普通文本

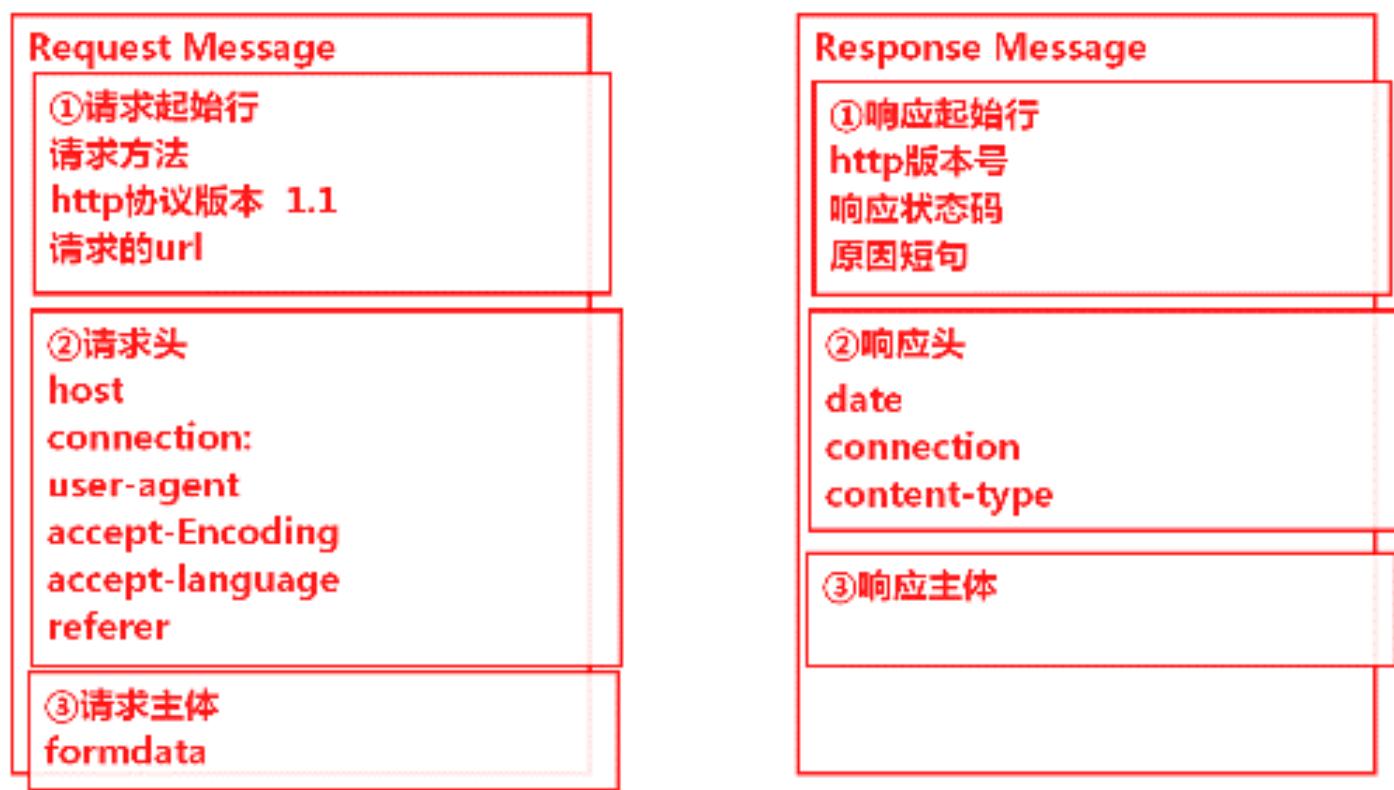
application/javascript js 文件

application/json json 字符串

application/xml xml 字符串

image/jpeg png gif 图片文件

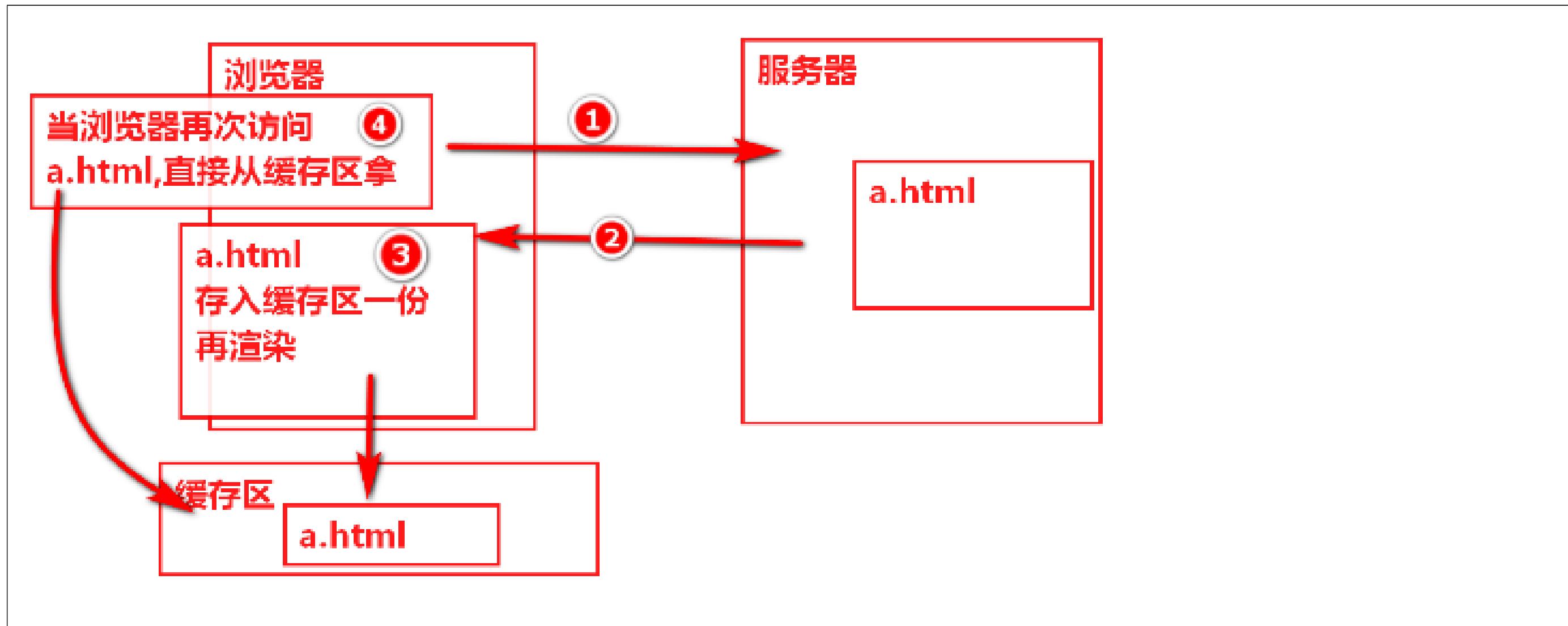
③响应主体



7.缓存

客户端将服务器响应回来的数据进行自动的保存，保存到缓存区

当再次访问这个数据的时候，直接从缓存区读取，不发请求



缓存的优点

- 1.减少冗(rong)余的数据传输，节省客户端流量
- 2.节省服务器带宽
- 3.降低了对服务器资源的消耗和运行要求
- 4.降低了由于远距离传输而造成加载延迟

设置需要前后台代码配合

与缓存相关的消息头的设置

Cache-Control: no-cache/max-age=0 不缓存

max-age=3600 单位秒,新鲜度---过期

```
<meta http-equiv="Cache-Control" content="max-age=3600">
```

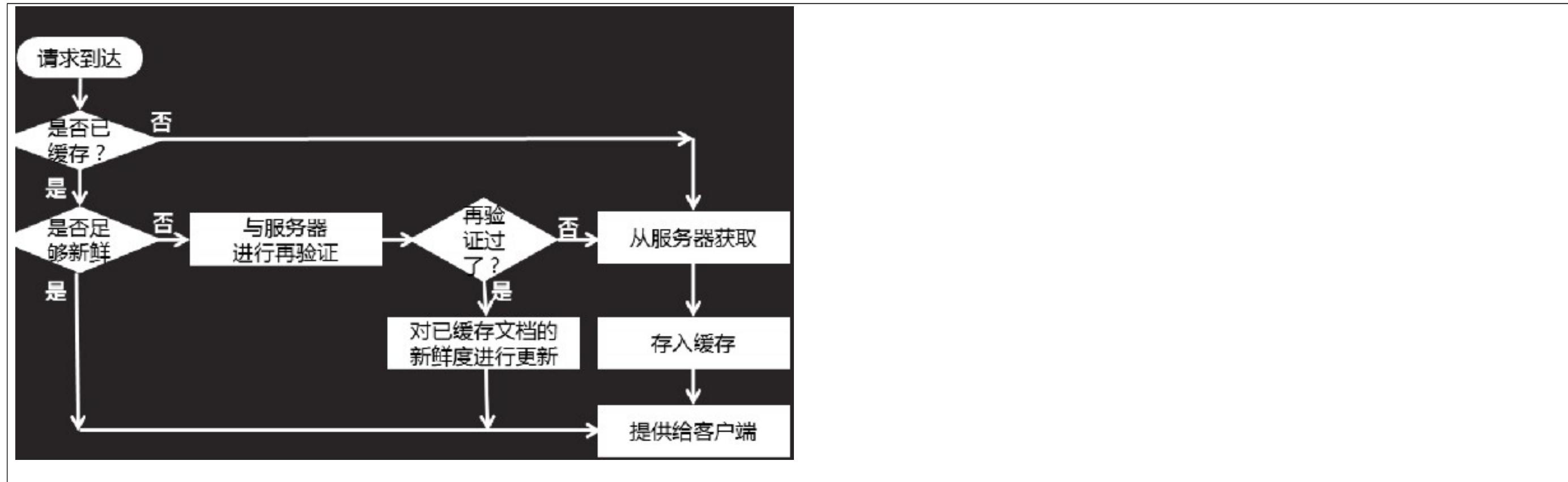
缓存流程

1. 请求--->没缓存--->连接服务器--->存缓存--->浏览器渲染

2. 请求--->有缓存--->够新鲜--->使用缓存--->浏览器渲染

3. 请求--->有缓存--->不新鲜--->连服务器验证--->不新鲜--->连服务器--->存缓存--->浏览器渲染

4. 请求--->有缓存--->不新鲜--->连服务器验证--->新鲜--->更新客户端的新鲜度--->浏览器渲染



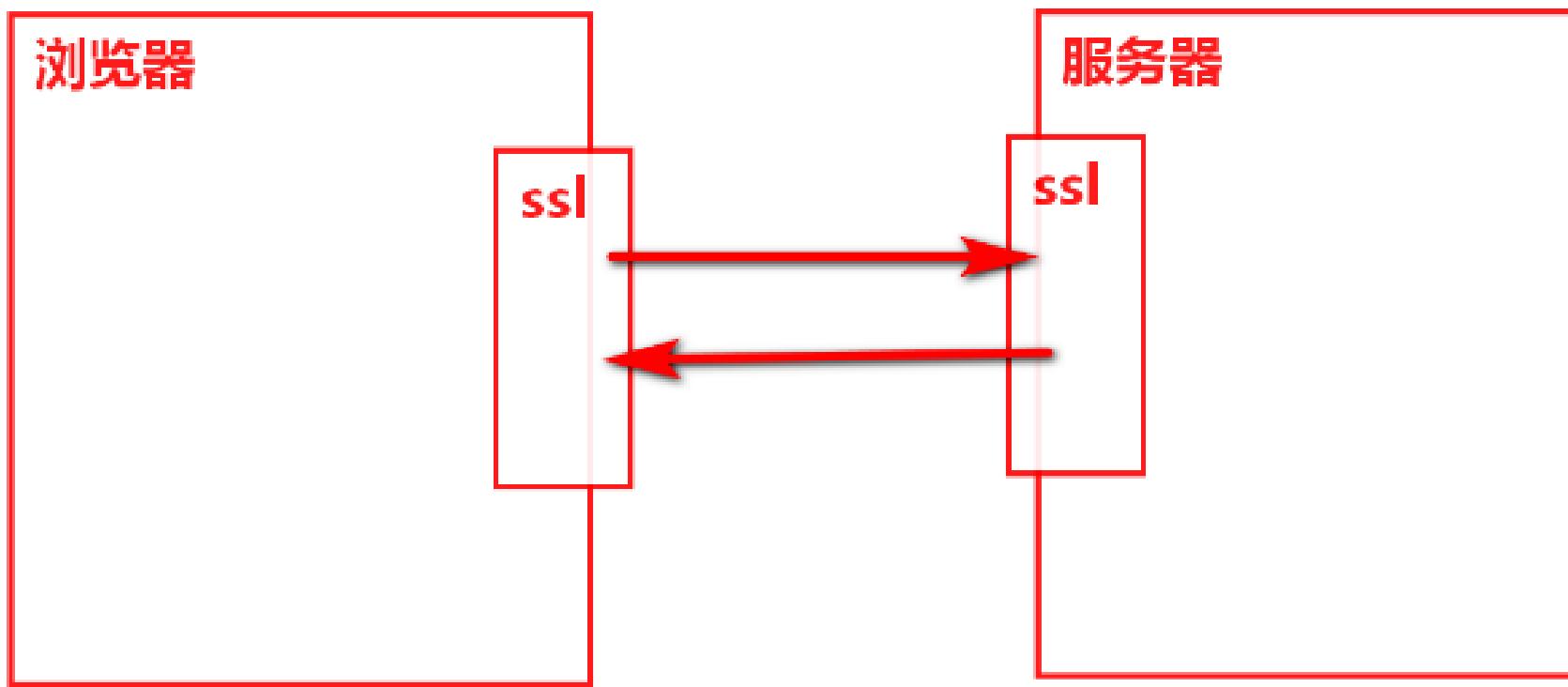
8.http 连接的性能优化

- 1.减少连接的创建次数(开启持久连接)
- 2.减少请求的次数 (使用缓存)
- 3.提供服务器端运行速度
- 4.尽可能减少响应数据的长度

9.安全的 http 协议

[https 443](https://www.iana.org)

添加了一个 SSL 层：专门为数据通信提供安全支持的



AJAX

一. 简易的 dom 操作（很费劲，原因：js 基础忘了）

1. 什么是 dom

完整 JavaScript 由 3 部分组成

1.js 核心 ECMA Script ES6

2.DOM Document Object Model 文档对象模型

把 html 看做文档对象，使用 js 操作标签对象

3.BOM Browser object Model 浏览器对象模型

使用 js 操作浏览器

2.为什么学习 ajax 之前要学一些 dom 操作

form 表单，点击提交，会自动的收集用户填写的数据

ajax 不使用 form，就没有自动收集数据的能力

需用我们使用 dom，操作 input 元素，手动收集数据

3.使用 dom 操作元素的步骤

1.获取元素对象

2.使用元素对象调用方法和属性 对象.属性 对象.方法()

① input 获取 value, 修改 value

```
var in_obj=document.getElementById("元素的 id");
```

获取值 var v=in_obj.value

修改值 in_obj.value="123";

②双标签获取内容区域, 修改内容区域

```
var div_obj=document.getElementById("d1");
```

div_obj.innerHTML

4.双标签的 innerHTML 详解

可以动态使用js 修改 html 结构

```
div_obj.innerHTML=<h1>123</h1>;
```

AJAXDay02:

一. 简易的 dom 操作

1. 使用事件，调用 js 代码

2. 通过 `document.getElementById("id")` 获取元素对象

`<input type="...." name="" id="">`

form 表单有自动收集整理数据的能力，元素必须有 name 属性，才会被收集

使用 ajax 不用 form 表单，就失去自动收集数据的能力，不需要使用 name
input 需要使用 id 属性。

但是，当对单选框和多选框进行操作，还需要使用 name 属性

此时的 name 用做分组

ps: ES6 对使用 id 获取元素对象做了简化

直接使用元素 id 充当元素对象

3.input 的值通过 value 属性来操作

4.所有双标签的内容区域，使用 innerHTML 属性来操作

5.innerHTML 可以赋值为 html 结构，改变页面布局

1.事件

单击事件 `onclick=""`

页面加载事件 页面一加载，马上调用 js 方法

必须添加在 body 中，`onload=""`

针对密码框和文本框的事件

获取焦点事件 `onfocus=""`

失去焦点事件 `onblur=""`

二.AJAX

1.同步 Synchronous

在一个任务进行的过程中，不能开启其他任务

同步访问：浏览器在向服务器发送请求的时候，浏览器只能等待服务器的响应，不能做其他的任何事

出现场合：

1. 浏览器地址栏输入 url
2. a 标签跳转
3. 表单的提交

2. 异步 ASynchronous

能够同时进行多个任务

异步访问：浏览器在发送请求的同时，还可以让用户在页面上做其他操作

出现场合：

1. 用户名重复的验证
2. 抖音评论

3.百度搜索联想

3.什么是 ajax

ASynchronous Javascript and XML json

ajax 的本质，使用 js 提供的异步对象

完成异步的发送请求

并接收响应

4.ajax 的使用步骤（原生 ajax）

1. 创建异步对象

```
var xhr=new XMLHttpRequest();
```

2. 创建请求

```
xhr.open("method","url",isAsy);
```

method: 请求方法

url: 请求的路径

isAsy: 是否使用异步请求 true--异步 false--同步

3. 发送请求

```
xhr.send(formdata);
```

参数为请求主体，没有请求主体，使用 null 或者不填

4. 接收响应数据

xhr.readyState---->xhr 的状态

有 5 个值 0 请求尚未初始化

- 1 请求正在发送
- 2 接收到了响应头
- 3 正在接收响应主体
- 4 响应主体接收完毕

```
xhr.onreadystatechange=function(){  
    if(xhr.readyState==4 && xhr.status==200){  
        var r=xhr.responseText;  
    }  
}
```

5.原生 http, get 方法带参数的请求

通过查询字符串传递参数

常见错误

1.跨域错误

```
✖ Access to XMLHttpRequest at  01_ajax.html:1  
  'http://127.0.0.1:8080/ajax/test' from origin  
  'null' has been blocked by CORS policy: No  
  'Access-Control-Allow-Origin' header is  
  present on the requested resource.
```

跨域错误

网页 file:///D:/web2003/day05/01_ajax.html

接口 http://127.0.0.1:8080/ajax/test

AJAXDay03:

二.AJAX

5.原生 http, get 方法带参数的请求

PS restful 的请求方法和 http 原生请求方法的区别

restful ---get/delete 操作一模一样，使用/: params 接收参数

http-----get/delete 使用? query 接收参数

/ajax/http_login?uname=abd&upwd=123

req.query.uname

6.restful 的 get 方法

后台

```
router.get("/restful_login/:uname&:upwd",(req,res)=>{
    var _uname=req.params.uname;
    var _upwd=req.params.upwd;
```

) ;

前台

xhr.open 的 url `ajax/restful_login/\${_uname}&\${_upwd}` 没有冒号

注意，restful 规则的接口

1.多个参数的连接符号，可以是 & 或者 -。但是需要前后台统一！！

2.restful 的 get 和 delete 请求，参数为空会报 404

强制在前台写非空验证

3.要对用户做友好的提示

练习，根据 uid 查询某个用户

7.json

后台得到 object，在传递给前台的过程中，自动的转换成 string 类型

这个 string 类型格式，是以 js 对象为基准的字符串

json javascript object notation js 对象表示法

以 js 对象为格式的字符串，简称 json 串

1.json 串是人为出现的么？ -----> 自动转换为 json 串 2

2.string 类型，只能用 string 的 API 操作，不能使用 js 对象的操作模式

json 解析

把 json 串转换成 js 对象，使用 js 对象的操作来处理数据

```
var obj=JSON.parse(json 串);
```

后续对 obj 进行操作就可以了

学习节点

restful get 请求

restful delete 请求

json 解析

post 有请求主体，insert 用 post

put 有请求主体，update 用 put

8.post 请求

http 的 post/put

restful 的 post/put

操作和代码一模一样

1.要创建请求主体，放在 send 中发送

```
var formdata=`uname=${$uname}&upwd=${$upwd}`;  
xhr.send(formdata);
```

2.要设置请求头信息，设置为可以传递所有字符

```
xhr.setRequestHeader("Content-Type","application/x-www-form-  
urlencoded");
```

这句代码，必须放在 xhr.open 和 xhr.send 之间

3.put 和 post 操作一模一样，需要修改请求方法的名称

restful 要求，什么操作用什么请求方法

get-----查询

post----新增

put-----修改

delete--删除

错误总结

: Can't set headers after they are sent.

在一个接口中，运行了两次 res.send();

AJAXday04:

day07

AJAX 项目

1. 结构

创建路由文件 pro.js，并且在 app.js 中挂载为 "/pro"

创建静态资源文件夹 pro, 并且在 app.js 中托管

2. 接口规范 restful

登录模块 get /pro/v1/login/:uname&:upwd router.get("/v1/login/:uname&:upwd", ()=>{});

响应数据 1 登录成功 0 登录失败

列表模块 get /pro/v1/list router.get("/v1/list", ()=>{});

响应数据 json

根据 uid 删除用户 delete /pro/v1/del/:uid router.delete("/v1/del/:uid", ()=>{});

响应数据 1 删除成功 0 删除失败

根据 uid 查询用户 get /pro/v1/search/:uid router.get("/v1/search/:uid", ()=>{});

响应数据 json 查询成功 0 未查询到

根据 uid 修改信息 put /pro/v1/update router.put("/v1/update", ()=>{});

提高任务

根据 uname 查询用户是否存在 onblur

注册-----逻辑复杂

3.登录

4.列表

5.删除

6.根据 uid 查询用户

7.根据 uid 修改用户信息

作业

- 1.完成 ajax 项目的修改模块
- 2.提高.根据 uname 查询用户是否存在 onblur
- 3.提高.注册



CSSday01:

一.css3 概述

1.什么 css

cascading style sheets

层叠样式表，简称样式表

2.css 的作用

html 负责网页的结构搭建，和数据的展示----->丑，素颜

css 负责网页的修饰----->亚洲 4 大邪术

3.css 与 html 属性的使用原则

由于 html 属性，没有可重用性，没有可维护性

所有，w3c 建议我们尽量的使用 css 的方式来取代 html 的属性

css 样式的优点：

1.提升代码的重用性(有限的提升，没有提升到极致， scss)

2.提高了可维护性

二.CSS 的语法规规范

1.使用样式的方式

①行内样式

在标签的 style 属性中，编写样式声明

样式声明---->样式属性：样式值

多个样式声明之间用分号隔开

```
<div style="color:red;background-color:yellow;font-size:50px">行内样式的样式</div>
```

内联样式，没有可重用性，没有可维护性

所有项目中基本不用，只有在学习和测试的时候使用

f12 中，内联样式，显示在 element.style 中

3 个常用样式属性

color:颜色的单词 文本颜色

background-color:颜色的单词 背景颜色

font-size:px 为单位的数字 字号大小

②内部样式

在 head 标签中，编写 <style></style> 标签

在 style 中， 选择器{样式声明;}

多个样式之间用分号隔开，但是注意，选择器{}结尾，一定不能添加；

内部样式，只能在当前页面中进行样式代码的重用，重用效果十分有限

所以，内部样式，在项目中使用较少，学习和测试使用较多

③外部样式

单独编写一个.css 文件

在 html 的 head 中使用 link 标签调用这个.css 文件

```
<link rel="stylesheet" href="my.css"/>
```

这样就能做到广泛的代码重用了，项目中使用最多

2.css 的特性

①继承性

很多样式是可以被内部元素继承的

文本，水平对齐可以被继承

宽高，边框，内外边距是不能被继承

例外：a 标签的字体颜色不会被继承

②层叠性

多个样式声明，同时作用到一个元素上

如果这些**样式不冲突**，所有样式都有效

③优先级---css 默认优先级

默认优先级顺序--从高到低

高 内联样式

内部样式，外部样式，遵循就近原则

低 浏览器默认样式 (不同浏览器给元素定义的默认样式不同，需要 css reset 统一样式)

f12 显示样式的顺序，是遵循 css 默认优先级排序

④ css 的! important 规则

在样式取值和; 号之间，添加! important;

作用，提高当前样式的优先级----->比内联都高

内联样式，不能设置! important

三.基础选择器

1.选择器的作用

规范了页面中哪些元素能够使用定义好的样式

选择器就是一个条件，符合这个条件的元素，可以使用定义好的样式

2.选择器详解

①通用选择器

*{样式声明}

但是，项目中少用！！！ * 效率极低

不多的用法中，*{margin:0;padding:0;} 清空所有元素的内外边距，css reset

②元素选择器,标签选择器

一般作为设置一类元素的公共样式

标签关键字{样式声明}

```
h3{color:red;}
```

③ id 选择器

匹配页面中某一个 id 值，专属定制

```
<any id="id 值">
```

```
#id 值{样式声明}
```

id 选择器在项目中，单独使用的情况不多！

[一般作为后代和子代选择器起始部分](#)

④类选择器

类选择器的定义 .类名{样式声明;}

```
<any class="类名"></any> any:代表任意标签
```

1.类名的要求

不能用数字开头

能够包含 字母 数字 _ -

见名知意

2.调用多个类名，每个类名中间使用空格分开（多类）

```
<any class="c1 c2 c3....."></any>
```

3. 分类选择器

① 元素选择器.类选择{样式声明}

调用这类名的，这个标签才能使用此样式

② .类名1.类名2

同时调用了这两个类名的元素会应用此样式

练习

03_ex.html

h2 标签，内容假文，此 h2 id 为 text1

使用 id 选择器设置 文本颜色 purple，背景 pink

字体为斜体 font-style:italic;

再使用 类选择器，设置字体颜色为 red

此时 f12 观察

最后使用标签选择器，设置文本颜色为 black, 背景为 yellow

f12 观察

坑，由于选择器权值问题，不执行就近原则了

⑤ 群组选择器

选择器 1, 选择器 2, 选择器 3,{样式声明}

```
.c1,#d1,h2,div{color:red}
```

⑥后代选择器

通过元素的后代关系，来匹配元素

后代，一级嵌套或多级嵌套

选择器 1 选择器 2 选择器 3{样式声明}

⑦子代选择器

通过元素的子代关系，来匹配元素

子代，一级嵌套

选择器 1 > 选择器 2 > 选择器 3{}

注意，子代选择器和后代选择器可以混写

⑧伪类选择器

匹配某一个元素的某一种状态

未访问状态 :link{}

已访问状态 :visited{}

激活状态 :active{}

鼠标悬停状态 :hover{}

被激活时的状态 :focus{}

练习 04_ex

一个 a 标签，内容随意，设置一下样式

1. 访问过后的状态 文本 orange

2. 被激活的状态 文本 purple

3. 鼠标悬停的状态 文本 red

4. 未被访问 文本 yellow

在一个 a 标签上同时设置 4 个伪类，需要按照一定的顺序编写，不然会冲突

爱恨原则 love hate

:link :visited :hover :active

⑨选择器有权值

权值代表一个选择器的重要程度

权值越高，选择器越重要，优先级越高

!important > 1000

内联样式 1000

id 选择器 100

类选择器 10

伪类选择器 10

元素选择器 1

通用选择器* 0

继承的样式 无

权值的特点：

1. 当一个选择器中，含有多个选择器时，需要将所有选择器的权值进行相加，然后比较。

权值大的优先显示。

2. 权值相同，就近原则

3. 群组选择器的权值，不相加，个算个的

4. ! important 直接获取最高权值

5. 选择器的相加结果，不会超过其最大数量级

(1W个1相加也不会超过10)

四. 尺寸和边框

1. 尺寸属性

width:

height:

取值：1.px 为单位的数字

ps：附加知识点----单位

1.px 像素

2.in 英寸 $1\text{in}=2.54\text{cm}$

3.pt 磅值 $1\text{pt}=1/72\text{in}$ 设置字号大小

4.cm

5.mm

-----相对单位-----

6.em 相对于父级设置的值是 1em

7.rem 相对于 html 设置的值的倍数，如果 html 没设置，默认 $1\text{rem}=16\text{px}$

8.% 一般是父元素的百分比

CSSday02:

四.尺寸和边框

1.尺寸

width 1.以 px/rem 为单位的数字

2.% 父元素的宽度的百分比

3.块级不设置宽度，宽度为父元素的宽度 100%

height 1.以 px/rem 为单位的数字

2.% 如果父元素是没有定义高度，那么%失效

如果父元素定义了高度，那么就是父元素高度的%

3.块级不设置高度，高度靠内容撑开

max-width

min-width

max-height

min-height

取值，1.px/rem 为单位的数字

2.max-width:100%;

图片有默认宽度，是图片自己的尺寸

可以缩放，但是最大不能超过自带宽度的 100%

2. 页面中允许设置尺寸的元素

块级元素	行内元素	行内块 input
单独成行	与其他行内和行内块共用一行	与其他行内和行内块共用一行
宽高有效	行内元素设置宽高无效	设置宽高有效

不设置宽高 默认宽是父元素 100% 默认高靠内容撑开	默认宽高是 靠内容撑开	默认自带宽高 但是每一个浏览器， 默认给 input 的宽高不同， 需要 css reset
-----------------------------------	----------------	---

3. 溢出的处理

溢出：当内容比较大，元素区域比较小的时候，就会发生溢出效果

坑：连贯的，没有空格的文本，浏览器解析成一个单词，不会自动换行

溢出默认是纵向的

`overflow: visible;` 默认值，溢出部分显示

`hidden` 溢出部分隐藏

`scroll` 强制添加滚动条

`auto` 只有溢出的时候，在溢出方向有滚动条，不溢出没有

`overflow-x`: 定义 x 轴有没滚动条

`overflow-y`: 定义 y 轴有没滚动条

注意: 1.`overflow` 底层非常非常复杂, 对我们来说没有意义

`overflow` 可以解决很多特殊情况, 但是不推荐用

2.如果想要把默认的纵向溢出, 改成横向溢出怎么办? ?

父元素宽度小, 子元素宽度大, 在子元素中添加内容

PS: 附加知识点, 合法的颜色值

1.颜色的英文单词

2.#后面跟 6 个 16 进制的数字

#rrggbb #ff0000

3.第二项的简写 #aabbcc #abc

#f00 #0f0 #00f #ff0 #0ff #f0f #000 #fff #666 #999

4.十进制定义单位

rgb(r,g,b)

5.带透明的颜色

rgba(red,green,blue,alpha); alpha:透明度 0~1 1 不透明 0 全透明

6.全透明的关键字 相当于 rgba(red,green,blue,0)

transparent

7.摄影师的颜色取值方式(知道有就可以)

`hsl(h,s,l)` h:色段 s:饱和度 l:亮度

4.边框

①简写方式

`border:width style color;`

同时设置 4 个方向的边框

`width:边框宽度`

color:边框颜色

style: 边框的样式 solid 实线

dashed 短线的虚线

dotted 原点的虚线

double 双实线

最简方式 border:style;

border:0; 清除边框

②单方向的定义

border-top/right/bottom/left: width style color;

③单属性定义

border-color:#0ff;

```
border-style:solid;
```

```
border-width:10px;
```

④单方向单属性的定义

```
border-right-color
```

```
border-top-style
```

...

一共 12 个

⑤倒角

把元素的 4 个直角都倒成圆角

border-radius: 圆的半径

取值 1.px 为单位的数字

2.% 50%就是一个圆

单角定义

border-top/bottom-left/right-radius

⑥阴影

box-shadow:h-shadow v-shadow blur spread color;

h-shadow 阴影的水平偏移距离

v-shadow 阴影的垂直偏移距离

blur 阴影的模糊程度

spread 阴影大小

color 阴影颜色

最简写法 box-shadow:h-shadow v-shadow;

内部阴影

外部阴影，向外扩散

内部阴影，往元素内部扩散

⑦轮廓

`outline: width style color;`

通常，我们只是清除轮廓

`outline:0;`

五.框模型(盒子模型)

框模型---元素在页面上实际占地空间的计算公式

元素实际占地宽度：

左外边距+左边框+左内边距+内容区域宽度+右内边距+右边框+右外边距

元素实际占地高度：

上外边距+上边框+上内边距+内容区域高度+下内边距+下边框+下外边距

外边距：边框以外的区域

内边距：边框到内容区域的距离

1. 外边距 margin

改变 margin 的值，元素有位移效果

调整元素与其他元素的间距

调整元素在页面上位置

① 简写语法

margin: v1 设置 4 个方向外边距

margin-top

margin-right

margin-bottom

margin-left

取值：距离某个方向多远

1.以 px 为单位的数字

2.% 上右下左都是父元素宽度的百分比

3.auto 只能设置左右，对上下外边距无效

让块级元素本身，水平居中

只对设置了宽度的块级元素有效

默认情况，左右冲突以左为准

上下冲突以上为准

②取多个值

margin:v1; 同时设置 4 个方向外边距

margin:v1 v2; v1 上下 v2 左右 margin:0 auto; margin:100px auto;

margin:v1 v2 v3; v1 上 v2 左右 v3 下 margin:50px auto 100px;

margin:v1 v2 v3 v4;

练习

05_ex.html

两个 div, 宽高都是 200px, 不同背景色

要求, 两个 div 之间有 50px 的距离

坑?

③外边距的特殊情况

1.外边距的合并

两个垂直外边距相遇时，会合并成一个外边距

取值以大的为准

解决方案：1.设计的时候规避

2.单独写一个上外边距或者下外边距

2.关于块级元素，行内元素，行内块的完整总结

块级元素	行内元素	行内块
独占一行	与其他行内和行内块共用一行	与其他行内和行内块共用一行
设置宽高有效	设置宽高无效	设置宽高有效
默认宽，父元素宽度 100% 默认高，内容撑开	靠内容撑开	浏览器会给一个默认的宽高，不同浏览器的 值不同

4个方向外边距有效

上下外边距无效

4个方向外边距都有效

注意，一行中，一个行内块使用上下外边距改变位置时，会连带着同一行其他行内元素和行内块一起移动

CSSday03:

五.框模型(盒子模型)

1.外边距 margin

③外边距的特殊情况

1.外边距的合并

两个垂直外边距相遇时，会合并成一个外边距

取值以大的为准

解决方案：1.设计的时候规避

2.单独写一个上外边距或者下外边距

2.关于块级元素，行内元素，行内块的完整总结

块级元素	行内元素	行内块
独占一行	与其他行内和行内块共用一行	与其他行内和行内块共用一行
设置宽高有效	设置宽高无效	设置宽高有效
默认宽，父元素宽度 100%	靠内容撑开	浏览器会给一个默认的宽高，不同浏览器的值不同
默认高，内容撑开		
4 个方向外边距有效	上下外边距无效	4 个方向外边距都有效

		注意，一行中，一个行内块使用上下外边距改变位置时，会连带着同一行其他行内元素和行内块一起移动
--	--	--

3. 浏览器会默认给一些元素外边距

body h1~h6 p ul ol pre

body 4个方向 8px

ul/ol 上下外边距 16px 左内边距 40px

4. 外边距溢出

在特殊的情况下，给子元素设置上外边距，会作用到父元素上

特殊情况：1. 父元素没有上边框

2. 子元素内容区域的上边沿和父元素内容区域的上边沿重合

解决方案：

- 1.给父元素加上边框，弊端，增加了父元素实际占地高度
- 2.给父元素添加上内边距,弊端，增加了父元素实际占地高度
- 3.给父元素设置 overflow:hidden/auto 元素如果想溢出显示，就冲突了
- 4.给父元素添加大儿子，一个空的<table></table>

2.内边距 padding

内边距变大，感觉上是元素变大了

但是元素的内容区域没有变，元素占地空间变大

内边距有颜色，与元素背景色相同

使用契机：1.把元素撑开

2.特殊情况下，border-box，元素之间的距离，用内边距设置，来欺骗用户

语法

`padding:v1;` 同时设置 4 个方向

`padding:v1 v2;` v1:上下 v2: 左右

`padding:v1 v2 v3;` v1:上 v2: 左右 v3: 下

`padding:v1 v2 v3 v4;` 上右下左

`padding-top`

`padding-right`

`padding-bottom`

`padding-left`

注意, `padding` 没有 `auto`

3. 更改盒子模型的计算方式

默认盒子模型的计算方式

左外边距 + 左边框 + 左内边距 + width + 右内边 + 右边框 + 右外边距

使用 box-sizing:修改盒子模型计算方式

box-sizing: 取值，关注 width 的范围

content-box; 默认值 设置的 width 就是内容区域的宽度

border-box; 设置的 width 是 border 以内所有内容

此时盒子模型计算公式为：左外边距+width+右外边距

使用场合，使用%定义尺寸的时候，一定要注意

有可能要更改盒子模型计算方式

六.背景

1.背景颜色

background-color:合法的颜色值

2.背景图片

background-image: url(09.png) ;

3. 背景图片的平铺

background-repeat:

默认值 repeat

不平铺 no-repeat

x 轴平铺 repeat-x

y 轴平铺 repeat-y

4. 背景图片的定位

background-position:

取值 1. 关键字 x y; x:left/center/right y:top/center/bottom

2. % x% y%;

3. px xpx ypx;

特殊情况，取一个值；这个值就是 x 的值，y 轴默认居中

5.设置背景图的大小

background-size:

取值 x y 以 px 为单位的数字

x% y%

一个值 同时设置 x 和 y

关键字一个值 contain 包含，容器包含背景，要把背景图完全显示

cover 覆盖，背景图片完全覆盖容器

6.背景图片的固定

background-attachment:

scroll;默认值，背景图位置，随着滚动条滚动

fixed 固定，背景图位置，不随着滚动条滚动，永远在当前位置（相对于 body 固定）

但是，只会在本元素中显示，离开本元素就看不到了

背景图的定位，会相对 body 去定位

7. 背景的简写方式

background: color image repeat attachment position;

背景图大小，不在简写中 background-size 需要单独设置

最简方式 background: color;

background:url();

七. 渐变 gradient

渐变是指多种颜色，平缓变换的一个效果

渐变的主要因素，色标 (颜色 出现位置)

一个渐变至少有 2 个色标，最后一个色标一定不能添加逗号

渐变分类

1.线性渐变

2.径向渐变

3.重复渐变

1.线性渐变

background-image: linear-gradient(方向,色标 1,色标 2.....);

方向： 1.关键字,写终点

to top ↑ 0deg

to right → 90deg

to bottom ↓ 180deg

to left ← 270deg

2.角度 deg

色标中位置

1.% 在方向上，按百分比显示颜色

2.px 在方向上的具体数值

3.不写位置，平均分配颜色

2.径向渐变

background-image: radial-gradient(半径 at 圆心, 色标 1, 色标 2.....);

半径：以 px 为单位的数字, 只有在色标的位置是%的时候，半径才生效

圆心：1.x y 以 px 单位的数字

2.x% y%

3.关键字 x:left/center/right y:top/center/bottom

3.重复渐变

把线性渐变或者径向渐变反复的运行

1.线性的重复渐变

```
background-image: repeating-linear-gradient(to right,#000 0%,#0f0 1%,#000 2%);
```

2.径向的重复渐变

```
background-image: repeating-radial-gradient(100px at center center,#000 0%,#faa  
1%,#000 2%);
```

CSSDay04:

七.渐变 gradient

1.线性渐变

```
background-image: linear-gradient(to right,色标 1,色标 2);
```

2.径向渐变

3.重复渐变

4.渐变的浏览器兼容性问题(css hack)

要在 linear-gradient 之前添加前缀

-webkit- chrome/safari

-moz- firefox

-o- opera

-ms- IE

background: -moz-linear-gradient(top, #27b1f6 0%, #0aa1ed 100%);

background: -webkit-linear-gradient(top, #27b1f6 0%, #0aa1ed 100%);

background: -o-linear-gradient(top, #27b1f6 0%, #0aa1ed 100%);

background: -ms-linear-gradient(top, #27b1f6 0%, #0aa1ed 100%);

background: linear-gradient(to top, #27b1f6 0%, #0aa1ed 100%);

如果添加内核，线性渐变的方向，必须写起点 top 从上往下

不添加内核前缀，写终点 to top 从下往上

八.文本格式化

1.字体属性

①字号大小

font-size:

取值：px/pt/em/rem 为单位的数字

②字体系列

font-family

font-family:"chiller","mv boli","jokerman";

③字体粗细

font-weight:

lighter 细 300

normal 普通 400

bold 粗体 600

bolder 更粗 700

数字取值，不带单位的 100 的整倍数，不能超过 1000

④字体样式

`font-style: normal/ italic` 斜体;

⑤小型大写字母

`font-variant: small-caps;`

⑥字体属性的简写方式

`font: style variant weight size family;`

最简方式 `font: size family;`

2.文本样式属性

①文本颜色

color:

②文本的水平对齐方式

text-align:

left 默认值

center

right

justify 两端对齐

`text-align` 和 `margin:0 auto` 的区别

`text-align: center` 是让元素内部的文本居中显示

可以让内部的行内元素和行内块水平居中

不能让块级元素水平居中

`margin:0 auto;` 让块级元素本身居中显示

③行高

如果行高大于字号，那么文本将在行高的范围内，垂直居中显示

让文本在容器中垂直居中显示

`line-height: 1.`以 px/em/rem 为单位的数字

2.不带单位的数字，是字号大小的倍数

通常我们会把行高设置为容器的高度
折行的文本，不建议使用行高设置垂直居中

④文本的线条修饰

text-decoration: underline; 下划线

overline 上划线

line-through 删除线

none 去掉线条 (a 标签去掉下划线)

⑤首行缩进

text-indent: 100px;

⑥文本阴影

text-shadow:h-shadow v-shadow blur color;

h-shadow 水平偏移

v-shadow 垂直偏移

blur 阴影的模糊

color 阴影颜色

九.表格的属性

1.表格常用属性

① table

尺寸，边框，背景，文本，内外边距都可以使用

但是边框只设置 table 最外层的边框

② tr，项目中，对 tr 设置 css 非常少

③ td

尺寸，边框，背景，字体文本，内边距

外边距无效

`vertical-align: top/middle(默认值)/bottom`

2. 表格的特有样式属性

① table

1. 边框合并 `border-collapse: collapse/separate` 默认值

2. 边框间距替代 td 的外边距

首先，要求表格的边框是分类状态 `border-spacing: 数字 px;`

3. 表格的标题 `caption-side: bottom/top` 默认值

4. 设置表格的显示规则

自动表格布局	固定布局
<code>table-layout:auto</code> 默认值	<code>table-layout:fixed;</code>

单元格大小会自动适应内容	单元格大小按照设置的尺寸显示 不读取到内存中，而是直接渲染
自动布局灵活	布局比较死板
自动布局需要先把整张表格读取 到内存中	少了存取内存的步骤 渲染效率高
再一次性渲染 当表格庞大/复杂时，渲染效率低	
适用于不确定每列大小，并且不 复杂的表格	适用于确定每列单元格大小的表 格

② tr/td/th

vertical-align: 让内部文本垂直对齐方式

top/middle/bottom

十.定位(*****)

1.什么是定位，控制元素在页面中的位置

2.定位分类

1.普通流定位

2.浮动定位

3.相对定位

4.绝对定位

5.固定定位

3.普通流定位（默认文档流）

1. 每个元素在页面上都有自己的空间（盒子模型）

2. 每个元素都是从父元素的左上角开始渲染

3. 块级元素独占一行，从上往下排列

4. 行内元素和行内块元素共用一行，从左往右排列

4. 浮动定位

让块级元素横向显示

`float: none;` 不浮动

`left` 左浮动

`right` 右浮动

1. 当前元素脱离文档流

2. 在发生浮动的当前行，向左/右对齐

浮动的特点

- 1.元素浮动，会脱离文档流
- 2.浮动元素，浮动之后，会停靠在父元素的边缘，或者其他已浮动元素的后面
- 3.父元素放不下所有浮动元素，放不下的浮动元素会自动换行
- 4.左浮动可以使用左外边距移动元素
右浮动要使用右外边距移动元素
- 5.浮动解决块级元素横向显示

5.浮动引发的特殊情况

①浮动元素占位的问题

浮动元素，会在浮动方向上，占满位置，不让已换行后续的浮动元素占据

②没设宽度的块级元素发生浮动，浮动之后的宽度靠内容撑开

③元素一旦浮动将变成块级元素

④文本，行内元素，行内块不会被浮动元素压在下面，而是巧妙的避开

6.清除浮动（清除之前浮动元素带来的效果）

一个元素设置了 clear: 就不会因为之前元素浮动，而上前补位

clear:left 清除左浮动元素的影响

right 清除右浮动元素的影响

both 左右都清除

7.高度坍塌

父元素没定义高，所有子元素都浮动，父元素失去高了

所有子元素脱离文档流，父元素在文档流中找不到子元素，就没有高了

解决方案

1.给父元素添加高度，但是不知道具体高度的时候，不能用

2.父元素也浮动，弊端，会影响父元素的兄弟元素和父级的父级

3.overflow:hidden/auto 弊端，可以解决，但是不推荐，溢出显示的需求不能做

4.在父级 div 的最后，追加一个空的子元素(块级)，给这个元素设置 clear:both

总结，子元素数量少，可以用 1 和 4 (推荐 1)

子元素数量多，或者数量不确定，一定用 4

面试，一定回答 4

总结，**脱离文档流**意味着 4 件事

1.不占据页面空间

2.后续未脱离文档流的元素上前补位

3.没设置宽度就脱离文档流，宽度靠内容撑开

4.所有脱离文档流的元素，都变成块级

CSSDay05:

day12

十.定位

1.默认文档流定位

2.浮动

一旦元素脱离文档流，会发生四件事情

- ①元素不占据页面空间
- ②后续未脱离文档流的元素上前补位
- ③元素如果没有设置宽度，脱离文档流之后，宽度靠内容撑开
- ④元素脱离文档流之后变成块级

高度坍塌

父元素没有设置高度，内部子元素都浮动，父元素就失去高

1.如果子元素少，可以直接给父元素添加高度

2.给父元素追加一个小儿子（空的块级元素，并设置 clear:both）

十一.其它常用样式属性

1.显示方式

元素在页面中成块级，行内，行内块，table 等规则显示

一个元素可以通过 display 属性来设置显示方式

display:block; 块级

inline; 行内

inline-block; 行内块

table; 表格

none; 隐藏，脱离文档流的隐藏

2.显示效果

隐藏 visibility:hidden; 隐藏，不脱离文档流的隐藏

显示 visibility: visible;

3. visibility:hidden 和 display:none;的区别

visibility:hidden; 不脱离文档流的隐藏

display:none; 脱离文档流的隐藏

4.透明度

rgba(x,x,x,0) transparent

opacity:0~1 0全透明，1是不透明

rgba 只让当前颜色透

opacity 让元素包括元素内部所有与颜色相关的都透明

5.光标的设置

根据操作系统的不同，光标效果不同

cursor:default; 箭头

pointer; 小手

crosshair 十字

wait 等待加载

help 帮助问号

text 文本 I

6.垂直对齐方式

vertical-align

适用场合

2 表格 设置表格内部文本的垂直对齐方式

3 top/middle/bottom

4 行内块 设置的前后文本（行内元素）与当前行内块的垂直对齐方式

5 middle(默认)/top/bottom

6 img 设置的前后文本（行内元素）与当前 img 的垂直对齐方式

baseline(默认值)/top/bottom/middle

7. 列表相关的样式

1. list-style-type: 设置列表标识类型

默认值 disc

none

square

circle

2. list-style-image: url(1.jpg) 把列表项设置成图片，但是要用小图

3. 设置列表项的位置

list-style-position:

outside 默认值，在 li 外侧，ul 的左内边距显示列表项

inside; 在 li 中显示列表项

4. 简写方式

list-style-type img position;

最简方式：list-style-type/img/position;

最常用的方案 list-style:none; 去除列表项

十二. 定位（相对定位，绝对定位，固定定位）

1. 定位的属性

position:

取值: static 静态, 默认文档流 默认值

relative 相对定位

absolute 绝对定位

fixed 固定定位

注意: 如果一个元素设置了 position 属性, 并且取值为 relative/absolute/fixed 中的一种时

1. 此元素被称为**已定位元素**

2. 解锁 5 个以 **px 为单位** 控制位置的**偏移属性**

left 距离左侧的距离

right 距离右侧的距离

top 距离顶部的距离

bottom 距离底部的距离

上下冲突，左右冲突，以上和左为准

z-index

2. 相对定位

position: relative， 配合偏移属性使用

不脱离文档流

相对本身原始位置做位移

使用场合

1. 类似于 margin， 做元素的位置的微调

2. 作为绝对定位元素的， 祖先级已定位元素使用

总结：如果一个元素设置了 relative， 但是没写偏移属性， 跟没定位的效果是一样的， 不会影响其他元素

3. 绝对定位

position: absolute; 配合偏移量使用

1. 绝对定位脱离文档流变成块级

2. 相对谁定位？

① 当祖先元素，没有已定位元素时，绝对定位元素相对 body 左上角去偏移

② 绝对定位元素相对于"最近的", "已定位的", "祖先级元素"左上角去偏移

3. 如果需求关注于绝对定位的位置，那么"最近的", "已定位的", "祖先级元素"用什么定位合适

4. 固定定位

永远相对于 body 左上角定位

position: fixed 配合偏移属性使用

5. 堆叠顺序

z-index 只能用于已定位元素上

默认的堆叠顺序，是后写的元素比先写的元素堆叠顺序高

问题：如何确认先后，html 结构的先后顺序

默认的堆叠顺序是多少值 z-index:0；

z-index:整数 默认值就是 0

1.定位的堆叠不能与浮动相比较，定位默认比浮动高，设置为负数比浮动低

2.堆叠顺序对父子关系无效，子元素永远在父元素的上面

CSS CORE

CSS COREDay01:

一.复杂选择器

1.兄弟选择器

兄弟选择器只能找后面的元素，不能前面的

有同一个父级的同级元素，称为兄弟元素

1相邻兄弟选择器，通过一个选择器找到紧紧挨着的符合另一个选择器弟弟 所有

选择器 1 + 选择器 2{}

②通用选择器，匹配一个选择器，后面所有符合条件的弟弟 效率低

选择器 1 ~ 选择器 2{}

CSS COREDay02:

day13

CSS CORE

一.复杂选择器

1.兄弟选择器

相邻兄弟选择器 选择器 1 + 选择器 2{}

通用兄弟选择器 选择器 1 ~ 选择器 2{}

2. 属性选择器(为 dom 准备的，精准的定位到元素)

根据元素的属性，精准的找到元素

我们可以通过 html 自带的属性，做属性选择器

还可以自定义 html 属性

[attr]{}

[attr1][attr2]

[attr=value]{}

elem[attr]{}

elem[attr1][attr2]{}

elem[attr1 = value1][attr2 = value2]{}

对属性值进行模糊查询

[attr[^]=value]{ } 匹配属性值以 value 开头的元素

[attr\$=value]{ } 匹配属性值以 value 结尾的元素

[attr*=value]{ } 匹配属性值中包含 value 的元素

[attr~=value]{ } 匹配属性值中包含 value 这个独立单词的元素

3. 伪类选择器

:link :visited :hover :active :focus

① 目标伪类

匹配锚点被激活的元素

:target{ }

② 结构伪类

根据元素的结构关系来匹配元素

1. elem:first-child{} 匹配的是 elem 的父元素的第一个儿子，这个儿子还必须是 elem 元素

2. elem:last-child{} 匹配的是 elem 的父元素的最后一个儿子，这个儿子还必须是 elem 元素
3. elem:nth-child(n) 匹配的是 elem 的父元素的第 n 个儿子，这个儿子还必须是 elem 元素
n 从 1 开始

③空选择器(纯丁克)

:empty{} 匹配内部没有任何元素的标签

任何元素：元素，文本，空格，回车

④独生子女

匹配属于其父元素的唯一子元素

:only-child{} 只认标签元素，空格，文本，回车不计算在内

⑤否定伪类

:not(selector) 符合 selector 条件的都不要

4. 伪元素选择器

改变元素内部一部分内容的样式

注意：w3c 最新的要求，写双：::

但是对于新要求发布时间点之前的伪类，还可以使用单：

1.:first-letter 或者 ::first-letter

匹配元素的首字符的样式

2.:first-line 或者 ::first-line

匹配元素的首行的样式，如果与首字符样式冲突，以首字符为准

3. ::selection 匹配用户选择文本样式

1.只能用双： 内部样式只能写文本颜色和背景颜色，其它样式无效

2.对首字符无效

5. 内容生成

使用 css 的方式，去修改添加 html 显示的内容

1. 使用 content 添加内容，只能设置文本 "", 还能使用 url(添加图片)

2.添加元素默认是一个行内元素，可以设置所有的样式。

添加这个元素的位置在当前元素的内容区域范围内最前面或最后面

:before 或者::before 内容区域范围内最前面

:after 或者::after 内容区域范围的最后面

使用内容生成能够解决的问题

1.解决外边距溢出

```
#d3:before{  
    content:"";  
    display:table;  
}
```

2.高度坍塌

```
#parent::after{  
    content:"";/*空元素*/
```

```
display: block; /*块级*/  
clear:both; /*清除浮动*/  
}
```

二.弹性布局 (重要程度*****)

浮动让块级元素横向显示

弹性布局可以替代浮动来做布局

1.弹性布局的相关概念

弹性容器

定义了 `display:flex` 的那个元素。容器内部的直接子元素进行弹性布局

弹性项目

真正发生弹性布局的元素们

定义了 `display:flex` 的那个元素的儿子们

主轴

项目们排列方向的一根轴

主轴有 4 条

x 轴 2 条

主轴的起点在左侧，主轴的终点在右侧

主轴的起点在右侧，主轴的终点在左侧

y 轴 2 条

主轴的起点在顶部，主轴的终点在底部

主轴的起点在底部，主轴的终点在顶部

交叉轴--2 条

永远与主轴垂直相交的一根轴

x:交叉轴起点在左侧，终点在右侧

y:交叉轴起点在顶部，终点底部

2.弹性语法

将元素设置成弹性容器，那么此元素所有的直接子元素就会变成弹性项目，按照弹性布局的方式排列

`display: flex;` 把块级元素设置成弹性容器

`display: inline-flex;` 把行内元素设置弹性容器

元素变成弹性容器（爹）后，`text-align`, `vertical-align` 失效

元素变成弹性项目（儿）后，`float/clear` 失效

元素变成弹性项目后，不管是块级还是行内都可以设置尺寸

3.容器属性（一次设置所有项目的显示方式）

①主轴方向的设置

默认主轴 x 轴，从左往右

`flex-direction:`

`row` x, 从左往右 默认值

row-reverse x,从右往左

column y,从上往下

column-reverse; y, 从下往上

②设置项目换行

主轴空间放不下所有项目，要不要换行

flex-wrap:

nowrap 默认值，不换行

wrap 换行

wrap-reverse; 翻转换行（会改变交叉轴默认的对齐方式）

③上面两个属性的简写

flex-flow:direction wrap;

④设置项目在主轴上的排列方式

justify-content

`flex-start` 项目在主轴的起点对齐

`center;` 主轴的中心对齐

`flex-end` 主轴的终点对齐

`space-around` 每个项目的左右外边距相等

`space-between` 两端对齐

⑤设置项目在交叉轴上排列方式

`align-items:`

`flex-start` 项目在交叉轴的起点对齐

`center;` 交叉的中心对齐

`flex-end` 交叉的终点对齐

`baseline` 项目中的文本，基线要对齐

`stretch` 项目在交叉轴方向上不写尺寸，项目充满交叉轴

4.项目属性（一次设置一个项目的显示方式）

①项目排列顺序

order: 整数，值越大离主轴起点越远

②单独设置一个项目的交叉轴对齐

align-self:

flex-start 项目在交叉轴的起点对齐

center; 交叉的中心对齐

flex-end 交叉的终点对齐

baseline 项目中的文本，基线要对齐

stretch 项目在交叉轴方向上不写尺寸，项目充满交叉轴

auto 使用容器设置的 align-items 的值

③ flex-shrink

当主轴方向显示不下所有项目时，项目是否压缩，如何压缩

flex-shrink: 默认值 1

取正整数，最小值为 0 不缩

把主轴方向不够的部分，按照几个项目的取值进行压缩

④ flex-grow

当主轴方向有剩余空间，项目是否增长，如何增长

flex-grow: 默认值 0; 不增长

把主轴剩余的空间，按照几个项目的取值进行分配

⑤ flex-basis

一个项目在主轴方向的基本尺寸

他的优先级大于 width/height

flex-basis: %;

px 为单位的数字

⑥ flex-grow, flex-shrink, flex-basis 的简写方式

flex:0 0 25%;

弹性容器

定义了**display:flex**的那个元素。容器内部的直接子元素进行弹性布局

弹性项目

真正发生弹性布局的元素们
定义了**display:flex**的那个元素的儿子们

主轴

项目们排列方向的一根轴

主轴有**4条**

x轴**2条**

主轴的起点在左侧，主轴的终点在右侧
主轴的起点在右侧，主轴的终点在左侧

y轴**2条**

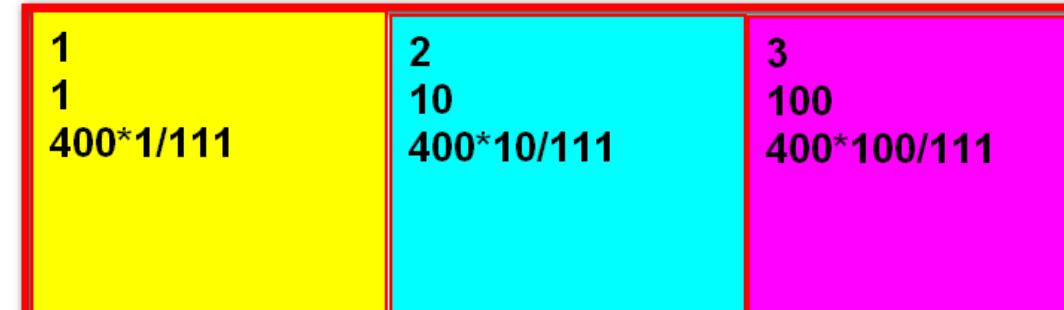
主轴的起点在顶部，主轴的终点在底部
主轴的起点在底部，主轴的终点在顶部

交叉轴--**2条**

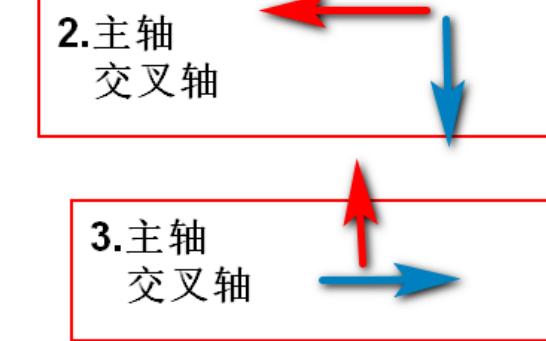
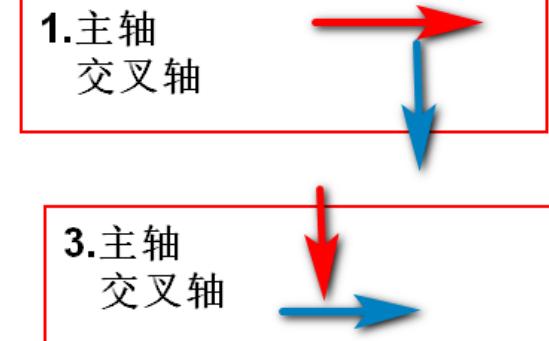
永远与主轴垂直相交的一根轴

x:交叉轴起点在左侧，终点在右侧

y:交叉轴起点在顶部，终点底部



600 1000 富裕400



CSS COREDay02:

二.弹性布局

三.CSS Hack

```
background:-webkit-linear-gradient();
```

```
background:-ms-linear-gradient();
```

```
background:-moz-linear-gradient();
```

```
background:-o-linear-gradient();
```

让同一份代码，可以在不同浏览器内核下，正常显示-----写 css hack

2015 年淘宝宣布，不再兼容 ie8 以下的版本

2017 年 bootstrap 推出 boot4,宣布不再兼容 ie8 以下的版本

细节知识，在 tmooc 回放的视频，正数第三个或者第四个

四.转换（重点*****）

1.什么是转换

改变元素在页面中的位置，大小，角度，以及形状

2D 转换，只在 x 轴和 y 轴上发生转换效果(4 个)

3D 转换,添加了 z 轴的转换效果(1 个)

2.语法

属性 transform: 转换函数 1 转换函数 2 转换函数 3.....; transform-function

3. 2D 转换(一共四个，注意前三个必须背)

①位移

transform: translate();

取值 1. translate(x); 指定元素在 x 轴上的位移

x: + → - ←

2.translateX(x); 同上

3. translateY(y) 指定元素在 y 轴上的位移

y: + ↓ - ↑

4. translate(x,y) 同时设置 x 轴和 y 轴的位移

②缩放，改变元素的尺寸

transform: scale();

取值 1.scale(v1)

同时缩放 x 轴和 y 轴的大小

v1>1 放大

v1=1 不变

0<v1<1 变小

-1<v1<0 变小并反转

v1=-1 不变并反转

v1<-1 放大并反转

2.scaleX(v1)

3.scaleY(v1)

4.scale(v1,v2) v1 单独设置 x 轴, v2 单独设置 y 轴

③旋转，改变元素的角度

transform: rotate(ndeg)

n + 顺时针 - 逆时针

注意：1. 旋转原点 transform-origin:x y; 转换原点会影响旋转的效果

取值，以 px 为单位的数字

%

关键字 x(left/center/right) y(top/center/bottom)

2. 旋转会带着坐标轴一起转，所以会旋转之后位移会受到影响

#d1 先 x 轴位移 300px. 再旋转 45 度

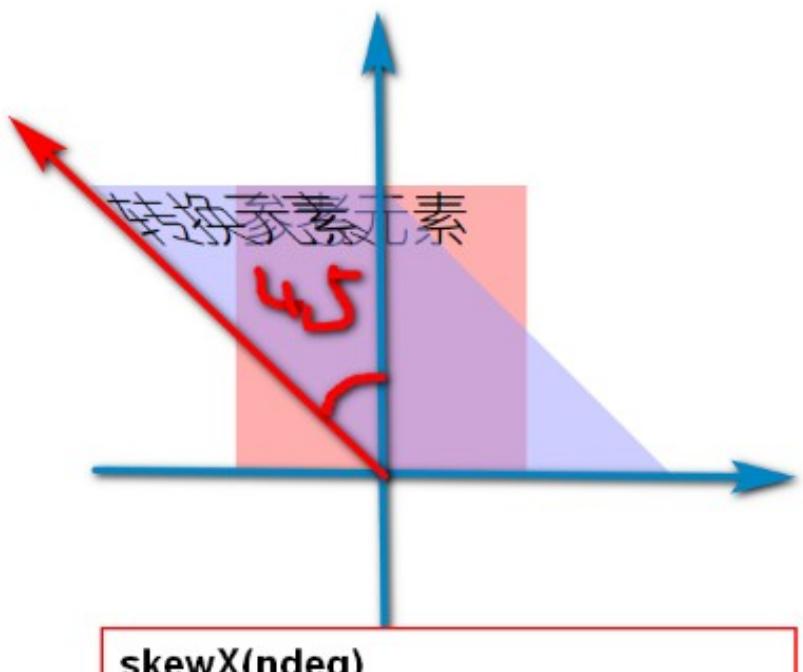
#d2 先旋转 45 度，再 x 轴位移 300px

④倾斜

易拉罐被踩了一脚，并且被拧了一下

`transform: skew();`

取值 1.`skew(x)`等同于 `skewX(x)`



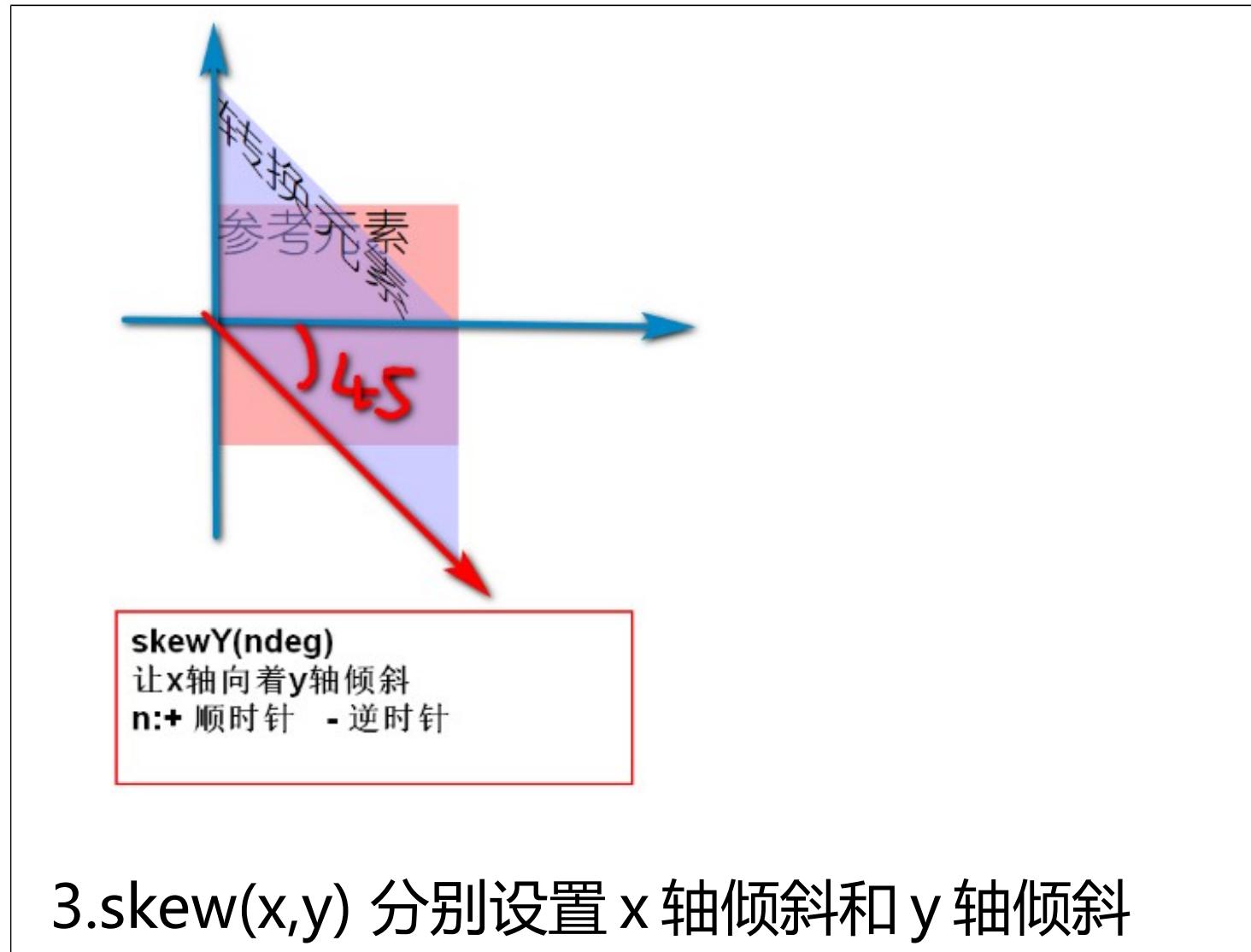
`skewX(ndeg)`

让y轴，向着x轴倾斜`ndeg`

`n:+` 逆时针

- 顺时针

2.`skewY(ndeg)`



3.skew(x,y) 分别设置 x 轴倾斜和 y 轴倾斜

4.3D 转换

3D 转换都是模拟的

①透视线距

模拟人的眼睛，到 3D 转换物体之间的距离

距离不同，效果就不同

perspective: 距离 设置在 3d 转换元素的父元素上

② 3d 旋转

transform:

- 1.rotateX(ndeg) 以 x 轴为中心轴旋转, 烤羊腿, 老式爆米花机
- 2.rotateY(ndeg) 以 y 轴为中心轴旋转, 旋转木马, 土耳其烤肉,钢管舞
- 3.rotateZ(ndeg) 以 z 轴为中心轴旋转, 电扇, 风车, 摩天轮
- 4.同时设置 x,y,z 的旋转角度 rotate3D(x,y,z,ndeg);

前 3 个参数, 代表 x,y,z 是否参与旋转, 和旋转的速率

五.过渡

css 的值, 在一段时间内平缓的变化 (把 css 的值变化过程显示出来)

css 的状态有两个, 开始时候的值, 和结束时候的值

1.语法

①设置参与过渡的属性

transition-property:

取值：1. 参与过渡的属性

2. 多个参与过渡的属性，用空格分开

3. all 所有支持过渡属性，都会参与

注意 1. transform/box-shadow 不管是否设置到过渡属性中，都会参与过渡

2. all 包括 颜色

大多数以数字取值的属性

阴影/转换

② 设置过渡的时长

transition-duration: s ms 为单位

③ 时间曲线函数

transition-timing-function:

取值的本质是贝塞尔曲线函数---->把几个特殊的取值变成了关键字

ease 默认值，慢速启动，中间小幅度加速，减速结束

linear 匀速

ease-in 慢速开始，缓慢的一直加速

ease-out 加速开始，一直减速

ease-in-out 慢速开始，中间先大幅加速，再大幅减速

④过渡的延迟

transition-delay: s / ms

⑤过渡代码的编写位置

写在起始的样式中，过渡效果，有去有回

写在激活过渡的伪类中，过渡效果，有去无回

⑥过渡简写方式

```
transition:property duration timing-function delay;
```

最简方式：

```
transition:duration;
```

六.动画

过渡	动画
两个 css 效果之间的变化	两个或者多个 css 效果之间的变化
必须使用伪类激活	可以刷新页面激活，也可以用伪类激活
两个 css 效果， 伪类激活	超过两个 css 效果， 页面刷新激活

1.什么是动画

是过渡的升级版，其实就是将 1 个或者多个过渡组合到一起的效果

2.关键帧

使用关键帧来控制动画中每个 css 效果，及其出现时间

关键帧：播放位置（播放的时间点）

在这个时间点上的样式

3. 使用动画的步骤

① 使用关键帧声明动画

```
@keyframes 动画名称{  
    from/0%{样式}  
    ....  
    to/100%{样式}  
}
```

② 调用动画

1.选择要调用的动画名称

animation-name:动画名称

2.设置动画的持续时间

animation-duration: s/ms

3.设置动画的时间曲线函数

animation-timing-function

ease

ease-in

ease-out

ease-in-out

linear

4.设置动画延迟执行

animation-delay: s/ms

animatie.css 的使用

1. 导入 css 文件

```
<link  
    rel="stylesheet"  
    href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.0.0/animate.min.css"  
/>
```

2. 把要使用的动画名称，设置在 animation-name 里

4. 动画的专有属性

① 动画的播放次数

animation-iteration-count

取值，1. 数字，具体的次数

2.无限 infinite

②动画的播放顺序

animation-direction: normal; 0%--->100% 默认

reverse 100%--->0% 倒叙

alternate 奇数次 0~100

偶数次 100~0

③动画的简写方式

animation: name duration timing-function delay count direction;

最简方式: animation: name duration;

5.动画的兼容性

如果要兼容低版本浏览器，需要在声明动画的时候做兼容

@keyframes 动画名称{}

@-webkit-keyframes 动画名称{}

@-ms-keyframes 动画名称{}

@-moz-keyframes 动画名称{}

@-o-keyframes 动画名称{}

七.CSS 优化

优化目的

- 1.减少服务器端的压力
- 2.提升用户体验

1.css 的优化原则

- 1.尽量的减少 http 请求个数
- 2.在页面顶部引入.css 文件
- 3.将 css 和 js 文件写入单独的文件中

BOOTSTARP

BOOTSTARPDay01:

过渡和动画的区别

过渡	动画
两个 css 状态之间的改变	两个或者多个 css 之间的改变
伪类激活	刷新页面和伪类激活

动画的填充状态

动画播放前后的填充状态，延迟时间内，和播放完毕后

`animation-fill-mode:`

forwards; 动画结束之后，保留在动画的最后一帧

backwards; 动画的延迟时间内，显示动画的第一帧

both; 前后都填充

七.CSS 优化

1.css 的优化原则

1. 尽量的减少 http 请求个数，精灵图

2. 在页面顶部引入.css 文件

3. 将 css 和 js 文件写入单独的文件中

2.css 代码优化

1. 合并样式（能用简写就不单独写，能用群组就不单独写）-----让 css 文件变小

2. 避免出现空的 src 和 href

3. 代码压缩

八.css reset 和 normalize.css

1.什么是css reset

不同浏览器，对元素的默认样式不同

在开发之前，要进行统一，这个统一的过程就是写css reset 例如： * { margin:0, padding:0}

css reset 的写法没有标准，工作你的领导让你用谁的你就谁的

css reset 比较霸道，会把很多元素的基本特性清除掉。

大家都是统一素颜，需要写什么样式需要重新写

很多元素失去了语义的特性

2.normalize.css

有一个标准范本

normalize.css 是一种 css reset 的替代方案

保留大部分元素语义，更新不同浏览器的 bug。

也是业内非常流行的一种方式

Bootstrap

<https://www.bootcss.com/>

一.响应式布局

响应式布局是 css 的技术，由于用 css 写非常繁琐

boot 对响应式布局做了封装

但是，我们需要先学习原始写法，再学习 boot 提供的简单写
法

1.什么是响应式网页

Responsive web page 响应式网页

可以根据**浏览设备的不同** (屏幕尺寸 横向屏幕 纵向屏幕 解析度),自动的改变布
局,图片,文字效果 不会影响用户体验

2.写响应式网页，必须做到的几件事情（针对初学者）

1.布局，不能写绝对值数据，要用相对单位 rem %

使用流式布局（默认文档流+浮动）和弹性，尽量不用定位

2.文本和图片随着设备大小变化而改变

3.使用媒体查询技术----css3 提供的技术

3.如何测试响应式

①使用真实设备

优点：真实可靠

缺点：成本高 测试任务量极大

②使用第三方的模拟器

优点：省钱，便捷

缺点：测试效果有缺陷，需要进一步验证

③使用 chrome 自带的模拟软件

优点：十分便捷

缺点：测试效果十分有限

④学习过程中，手动拖放浏览器尺寸就可以了

4. 编写响应式布局

① 移动端适配

```
<meta name="viewport"  
content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0">  
  
width=device-width 视口宽度为设备宽度---不需要横向滑屏  
  
initial-scale=1.0    初始化的时候，视口不能缩放  
  
maximum-scale=1.0  最大缩放比例 1.0---不能缩放  
  
user-scalable=0"    用户不能缩放视口
```

最简洁的视口设置方式

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

注意：我们在 pc 测试页面，不写视口不会有影响

②使用 css3 提供的媒体查询技术，编写响应式

Media Query: 可以自动的根据浏览设备不同，自动改变页面的尺寸，布局，大小，方向等

设备包含，硬件，屏幕尺寸，屏幕方向，解析度等等

使用方案：针对不同设备写不同的 css。

通过媒体查询技术，根据设备情况，调用符合条件的 css, 忽略其他 css

媒体查询会让代码量几何型的增加，复杂页面不适合使用响应式

③常用的设备规格

1.硬件条件

screen(pc/pad/phone) 移动设备

TV 电视

print 机器上的小窗口屏幕

2.屏幕尺寸的条件，市场上主流屏幕尺寸有 5 种

超大屏 xl 屏幕宽度 \geq 1200px min-width:1200px

大屏 lg 992px \leq 屏幕 $<$ 1200px min-width:992px and max-width:1199.99px

中屏幕 md 768px \leq 屏幕 $<$ 992px

小屏幕 sm 576px \leq 屏幕 $<$ 768px

超小屏 xs 屏幕 $<$ 576px

④语法总结

完整语法,按照屏幕尺寸写 5 套

```
@media screen and (min-width:992px) and (max-width:1199.00px){  
    div{.....}  
}
```

简写方式--boot 的响应式就是这样封装的

```
@media (max-width:575.99px){} xs  
@media (min-width:576px{}) sm-md/lg/xl  
@media (min-width:768px) {} md/lg/xl  
@media (min-width:992px{}) lg/xl  
@media (min-width:1200px) {} xl
```

如果有尺寸没有写，那么会发生向上兼容的问题

二.BootStrap

把很多样式封装成类，我们只需要调用类名就可以了

但是，封装是什么？提前写好的样式

在具体的项目中，提前写好的样式不能满足我们需求怎么办，

我们还是需要手写样式

1. 如何使用 boot

```
<!-- boot 是移动设备优先的框架，所以要添加视口设置 -->  
<meta name="viewport" content="width=device-width, initial-scale=1">  
<!-- 文件导入，有严格的顺序要求 -->  
<link rel="stylesheet" href="css/bootstrap.css">  
<script src="js/jquery.min.js"></script>  
<script src="js/popper.min.js"></script>  
<script src="js/bootstrap.min.js"></script>
```

还需要创建 div

```
<div class="container"></div>
```

2.全局 css 样式

boot 不支持 xs 超小屏 boot 1rem=16px

①容器

.container 居中对齐，左右 15px 内边距

在不同屏幕下，都有最大宽度----->定宽容器

. container-fluid 居中对齐，左右 15px 内边距, width:100%---->变宽容器

②按钮相关的 class

基本类 .btn 圆角，阴影， hover, focus, disabled 等等

按钮颜色 btn-danger/warning/success/primary/info/secondary/dark/light

镂空按钮 btn-outline-danger/warning.....

设置按钮的大小

btn-lg 大按钮

btn-sm 小按钮

btn-block 块级按钮

③图片相关的样式

rounded 0.25rem 的圆角

rounded-circle 50%的圆角

img-thumbnail 有内边距，右边框，有圆角的缩略图

img-fluid max-width: 100%;图片可以缩放，但是不能超过原始尺寸 100%;

响应式图片

④文本相关的样式

text-info/danger/warning..... text-muted 浅灰色

text-uppercase/ lowercase/ capitalize 大写/小写/首字母大写

font-weight-bold/normal/light 字体粗细的控制

.h1~~.h6 字号大小

文本对齐方式----->封装了响应式

text-justify

text-* -left/center/right/ *:sm/md/lg/xl

注意 1：boot 中封装的响应式，屏幕向上兼容

2: boot 没有封装 text-justify 的响应式

.text nowrap 文本不换行

.text-truncate 溢出的文本隐藏，显示的文本最后添加省略号

⑤列表相关的样式

boot 给我们提供了一个列表组，如果你的需求不需要列表组，就可以不用 ul 的类

list-unstyled 去除列表标识

list-group 弹性，y 轴主轴

li 的类

list-group-item 边框，内边距，第一个 li 的上圆角，最后一个 li 下圆角

list-group-item-danger/warning。 . 。 。 列表项的颜色

active 被选中的列表项

disabled 禁用的列表项

BOOTSTARPDay02:

Boot 三大重点 (手写媒体查询, 栅格布局, sass)

二.BootStrap

2.全局 css 样式 3

boot 不支持 xs 超小屏 boot 1rem=16px

3.辅助类 (所有元素都可以使用的样式)

①边框

基本类 border/border-top/ border-right/ border-bottom/
border-left

去除边框 border-0 / border-top-0/ border-right-0/ border-
bottom-0/ border-left-0

边框颜色 border-dark/danger/warning.....

②背景颜色

bg-info/danger/warning.....

bg-transparent

③浮动---带响应式

float-* -left/right/none *:sm/md/lg/xl

父元素解决高度坍塌，给父元素添加类名.clearfix

④圆角

rounded 4个角都是 0.25rem 圆角

rounded-left/right/bottom/top 把某个方向的两个角设置成圆角

rounded-circle 50%圆角

rounded-0 清除所有圆角

⑤内外边距的设置

外边距--245 个类名

m/mt/mr(mb/ml)/mx/my-* -0/1/2/3/4/5/auto

*:不加响应式/sm/md/lg/xl

0:0

1:0.25rem

2:0.5rem

3:1rem

4:1.5rem

5:3rem

内边距--215

p/pt/pr/pb/pl/px/py-*0/1/2/3/4/5

*:不加响应式/sm/md/lg/xl

⑥尺寸

w/h-25/50/75/100

尺寸经常要自己定义 .w-

95{width:95%}

4. 棚格布局 grid (重要 计算比例 (12/比例总和) *每份占比

*****)

把所有的布局看成一个 row, 一个 row 有 12 列

row 中布局宽度, 通过占多少个 col 来定义。

一个 row 如果超过 12 个 col, 会自动换行。

div.row

>div.col-n n:1~12

栅格的作用 1

快速完成布局的分布

源码

.row 弹性，x轴主轴，可换行，**左右-15px 外边距，通常要清掉**

.col-n n:1~12

左右+15px 的内边距，通常要清掉

使用 flex:0 0 %;中的 flex-basis 来设置每个项目在主轴上的空间

no-gutters 清除 row 左右外边和 col 左右内边距

栅格的作用 2-----快速完成响应式栅格

4个div, xl一行4个

lg一行3个

md一行2个

sm一行1个

web的页面布局一般3种

table 布局	div+css	boot 封装的栅格
简单，容易控制	语义正确，渲染效率高	简单，易控值，语义正确，渲染效率高，支持响应式
语义错误，渲染效率低	控制起来很麻烦	复杂的页面，不适合用栅格

其它 col 的使用方式

不带任何数字的.col 类

- 有几个 col 都平均分配，可以写出与 12 没有关系的数字
- 如果每个 col 内容足够少，一行可以超过 12 个 col

列偏移

offset-* -n *: sm/md/lg/xl n:0~11

原理，就是添加 margin-left

细节

一般 col 不能手动添加左右外边距来做缝隙

会把同一行其它 col 挤下去

5. 弹性布局

d-* -block/inline/inline-block/none/table/flex/inline-flex

*:sm/md/lg/xl

主轴方向： flex-* -row/column/row-reverse/column-reverse

主轴对齐方式 justify-content-* -start/end/center/ between/around

换行 flex-* -wrap/nowrap/wrap-reverse

交叉轴 align-items-* -start/center/end/baseline/ stretch

注意：只要外层是弹性，主轴 x 轴，可换行，主轴起点对齐

内部直接子元素，都可以直接用 col

在 boot 存在很多设置了弹性的类名，里面可以直接用 col

boot 的网络资源调用方式

1.进入 boot 官网 <https://www.bootcss.com/>

2.进入底部 bootCDN 的链接

BOOTSTARPDay03:

二.BootStrap

5.弹性布局

6.表单

①表单元素的排列方向

form-group 垂直表单

form-inline 水平表单

②表单元素的样式

文本框的类名 **form-control**

块级，w100,字体，边框，过渡

设置文本大小，上下内边距大小，最终效果，其实就是在设置 input 的高度

col-form-label-lg

col-form-label

col-form-label-sm

③表单要注意的事项

form-inline 可以替代栅格的 row，内部直接写 col

三.组件

boot 把项目中常用的功能性模块，进行了封装，就叫组件

我们可以快速完成组件，但是如果有特殊需求，还是需要我们自己修改
组件只需要背几个类名，其它的内容，只要能够对照笔记和文档写出来就行
使用 boot 封装的事件，需要关注两件事情

- 1.激发事件的元素和自定属性(boot 把调用事件的自定义属性固定了)
- 2.事件影响的目标

1.按钮组

btn-group 横向按钮组

btn-group-vertical 垂直按钮组

可以继续添加类名，设置按钮组大小 btn-group-lg/sm

2.下拉菜单

基本结构

div. dropdown 相对定位

>btn. dropdown-toggle 画出一个向下的倒三角

```
+ul.dropdown-menu display:none;
```

事件

button 中写自定义属性 data-toggle="dropdown"

不需要设置事件影响的目标

3.信息提示框

结构

div.alert 基本类，相对定位，边框，圆角，内边距

.alert-danger/warning..... 信息提示框的颜色

.alert-dismissible 负责修饰内部.close

>span.close 设置小叉叉的样式

事件

1.span data-dismiss="alert"

2.不需要设置事件影响的目标

4. 导航

① 水平导航

```
ul.nav 弹性，x轴，可换行，去掉左内边距，去掉点。  
.nav-justified 配合子元素 li.nav-item 让每个li等分显示  
>li.nav-item  
>a.nav-link 块级，hover等伪类
```

② 选项卡导航

```
基本结构  
ul.nav .nav-tabs 下边框，让内部所有link都有透明边框，hover边框显示  
>li.nav-item  
>a.nav-link .active 选中状态  
div.tab-content 本身没有样式  
>div.tab-pane 配合父元素 tab-content --->display:none;
```

.active 显示

事件：

1.a 标签上写事件 data-toggle="tab"

2.设置事件影响对象

boot 要求，被影响对象有 id

div.tab-pane#d1

在对应 a 标签的 href 中绑定这个 id href="#d1" 要带井号

③胶囊导航

基本结构

基本结构

ul.nav .nav-pills

>li.nav-item

>a.nav-link .active 选中状态

div.tab-content 本身没有样式

>div.tab-pane 配合父元素 tab-content ---> display:none;

.active 显示

事件

1.a 标签上写事件 data-toggle="pill"

2.设置事件影响对象

boot 要求，被影响对象有 id

div.tab-pane#d1

在对应 a 标签的 href 中绑定这个 id href="#d1" 要带井号

5. 导航栏（响应式导航栏）

div.navbar 弹性，x 轴，两端对齐，可换行，交叉轴居中

.navbar-expand-* x 轴，不可换行，主轴起点对齐 *:sm/md/lg/xl

>ul.navbar-nav 弹性，y 轴，

配合父元素的.navbar-expand-*, 让主轴轴方向变为 x

源码中的选择写法.navbar-expand-md .navbar-nav{flex-direction: row;}

如果这个选择器成立，主轴就是 x 轴，所有 li 横向显示

不成立，主轴就是 y 轴，所有 li 纵向显示

>li.nav-item

>a.nav-link

6. 折叠

基本结构

.btn

div. collapse ----> display:none;

事件

1. 按钮添加自定义属性 data-toggle="collapse"

2. 设置事件影响的对象，给 div 添加 id，并在 button 中指向 id

```
data-target="#demo"
```

7.卡片

```
<div class="card"> 弹性, y轴, 背景, 边框  
<div class="card-header"></div> 背景边框  
<div class="card-body"></div>  
<div class="card-footer"></div> 背景边框  
</div>
```

8.手风琴效果(卡片+折叠)

```
div#p1  
>div.card*3  
  >div.card-header>a  
  +div.collapse#c1  
    >div.card-body
```

事件

a 标签中 data-toggle="collapse" href="#c1"

每个 div.collapse 也要添加事件，才能在父元素中只展开一个

data-parent="#p1" 父级 id

9. 折叠导航栏

结构

div.navbar

.navbar-dark 让内部 a 标签，按钮，ul>li>a 字体颜色变为浅色

没有设置自己的深色背景，需要手动设置

.bg-dark

.navbar-expand-xl 1.让 button 在 xl 以上隐藏，在 xl 以下显示

2.让 ul. navbar-nav 的主轴在 xl 以上是 row, 在 xl 以下是 column

3.让 div .navbar-collapse 在 xl 以上显示，在 xl 以下隐藏

>a.navbar-brand 内边距，右外边距，字号，行高
+ button.navbar-toggler 定义内边距，字号，行高，背景，边框
 >span.navbar-toggler-icon 画 3 条线 收最外层 div.navbar-dark 变成浅色的 3
 条线
+div#d1.collapse 隐藏
 .navbar-collapse 响应式的隐藏和显示，配合最外层 div.navbar-expand-xl
>ul.navbar-nav
>li.nav-item
>a.nav-link

事件

- 1.button data-toggle="collapse"
- 2.设置受事件改变的元素 data-target="#d1"

BOOTSTARPDay04:

三.组件

9.折叠导航栏

10.媒体对象

div.media 弹性，x轴

>img

+div.media-body

11.焦点轮播图

1.图片部分

结构

div.carousel 相对定位

```
>div.carousel-inner 相对定位, w100, 溢出隐藏  
>div.carousel-item 相对定位, w100, display:none.  
.active display:block  
>img
```

事件

div.carousel"中, 写自定义属性 data-ride="carousel"

一定要给某一个 div.carousel-item 添加 active 类, 不然没有效果

data-ride="carousel"操作两件事, 1.让图移动 2.让轮播指示器的点移动

2.轮播指示器

结构

```
<ul class="carousel-indicators"> 让 ul 定位, 让 li 设置尺寸, 背景等  
<li data-target="#demo" data-slide-to="0" class="active"></li>
```

```
<li data-target="#demo" data-slide-to="1"></li>
<li data-target="#demo" data-slide-to="2"></li>
<li data-target="#demo" data-slide-to="3"></li>
</ul>
```

需要重写 li 的样式

```
.carousel-indicators li{
    width:0.8rem;height:0.8rem;
    border-radius: 50%;
    background-color: #fff;
    margin-left:0.25rem;
    margin-right: 0.25rem;
}
```

```
.carousel-indicators .active {  
    background-color: #0aa1ed;  
}
```

点击点点，让图片和指示器都转换

① li 事件 data-slide-to="0/1/2/3"

② 目标 li 最外层 div.carousel#demo data-target="#demo"

3. 左右箭头

结构

a.carousel-control-next 定义了 a 标签的位置和大小

> span.carousel-control-next-icon

a.carousel-control-prev

> span.carousel-control-prev-icon

样式重写

```
.carousel-control-prev,.carousel-control-next{  
    width:4%;height:20%;  
    background-color: #aaa;  
    border-radius: 0.25rem;  
    top:40%;  
}
```

```
.carousel-control-prev{left:0.25rem}
```

```
.carousel-control-next{right:0.25rem}
```

事件

```
a data-slide="next/prev"
```

```
设置目标 href="#demo"
```

四.其它常用组件

1.徽章

就是一个小的按钮

基本类 badge

颜色 badge-danger...

badge-pill 胶囊

2.巨幕

jumbotron 巨大的内边距，背景，圆角

3.面包屑导航

用于有层级关系的导航，现在不要求了

ul. breadcrumb 弹性，背景，圆角

>li. breadcrumb-item

我们可以重写每个 li 之间的连接符号

```
.breadcrumb-item + .breadcrumb-item::before{content: ">";}
```

4.分页条

ul.pagination 弹性

>li.page-item 给 a 添加前后的圆角

.active 选中

.disabled 禁用

>a.page-link 背景色，边框

5.进度条

div.progress 进度条的槽

>div.progress-bar 进度条的进度

.bg-danger... 进度条的颜色

.w-75

进度条的宽度

.progress-bar-striped 带条纹的进度条

progress-bar-animated 带条纹动画的进度条

SCSS

boot 中必须掌握的 3 个知识点

1.手写媒体查询

2.栅格布局

3.scss (js+css 的写法)

一.scss 的理论

css 和元素的属性使用规则，尽量使用样式来替代属性的写法。css 有一定的重用性，可维护性

使用 scss 动态样式语言可以让 css 的重用性做到极致

scss 提供了变量，嵌套，混合器，指令等高级功能

动态样式语言有下面这些

1.scss/sass 的升级版(scss 的语法更接近 css 语法) 写了 boot4

2.stylus

3.Less 写了 boot3

scss 是动态的 css 语言，语法与 js 和 css 十分相似

文件完成之后，会保存一个.scss 文件，scss 运行在后台，转换成 css 文件，把 css 发送给前台

1. 安装---阅读安装文档

2.scss 文件转换成 css 文件

路径问题

找到 scss 文件的路径 C:\Users\web\Desktop\web2003\day18\scss\01.scss

找到要转换成 css 的路径 C:\Users\web\Desktop\web2003\day18\css\01.css

打开 powershell 的路径 C:\Users\web\Desktop\web2003\day18>

①单文件转换

```
node-sass scss 文件路径 css 文件路径
```

②多文件转换(一次转换一个文件夹)

```
node-sass scss 文件夹 -o css 文件夹
```

③单文件监听转换 (scss 文件 ctrl+S 自动转换成 css)

```
node-sass -w scss 文件路径 css 文件路径
```

④多文件监听 (一次监听一个文件夹)

```
node-sass -w scss 文件 -o css 文件
```

二.scss 基本语法

1.变量

```
$my_color:#000;
```

变量使用\$声明，可以包含- _数字，字母，命名规则基本与 css 选择器相同。尽量做到见名知意

变量声明时，可以引用其他变量

scss 有作用域，原理与 js 变量作用域基本相同

2. 嵌套

1. 用嵌套的方式写选择器，减少代码量

```
#d1{  
    color:#123;  
    span{color:#321;}  
}
```

会生成

```
#d1 { color: #123; }

#d1 span { color: #321; }
```

2. 伪类的嵌套，必须有占位符&。不然伪类会和选择器之前有一个空格

```
a{color:red;

&:hover{color:blue;}
```

3. 群组的嵌套

```
nav,div,#d3,.c1{

a{color:red;}
```

生成 nav a, div a, #d3 a, .c1 a { color: red; }

4. 属性的嵌套

```
div{  
    border:{  
        style:solid;  
        color:#red;  
        width:100px;  
    }  
}
```

生成

```
div {  
    border-style: solid;  
    border-color: #red;  
    border-width: 100px; }
```

3.导入

scss 文件分为两种

- 1.全局 scss 文件 会在监听时被生成.css, 文件名不以下划线开头 bootstrap.scss
- 2.局部 scss 文件, 在被监听时, 不会生成相应的 css 文件, 是以下划线开头的 _breadcrumb.scss

注意: 1.全局 scss 使用@import "局部 scss 名称";导入局部 scss 文件

局部 scss 名称可以不带前面的下划线, 也不加后缀

2.被导入的 scss 文件中定义的变量, 函数, 可以被使用

ProjectBoot

ProjrctBootDay01:

SCSS

二.scss 基本语法

4. 继承

一个选择器可以继承另外一个选择器的所有样式

```
#d1{.....}
```

#d2{ **@extend** #d1;} 就可以继承 d1 的所有样式

但是现实形态是 #d1,#d2{....} 群组选择器

5. 混合器

把一堆 css 样式，封装进一个混合器

这些样式，哪里需要用，哪里就调用这个混合器

定义混合器 @mixin 混合器名称{样式声明}

调用混合器 在选择器中使用 @include 混合器名称;

带参数的混合器 @mixin 混合器名称(\$参数 1, \$参数 2){样式声明}

调用带参数混合器@include 混合器名称(值 1, 值 2);

6. 运算 (加减乘除模)

① 加

+ 除了做数学相加，还做字符串拼接

做字符串拼接的时候，有双引号的去拼接没有双引号的，结果有双引号

没有双引号去拼接有双引号的，结果没有双引号

+ 左边有双引号，结果就有双引号

②减

因为 css 的选择器命名方式， -号有可能会被误认为是变量名的一部分
所以，一般 - 号前后添加空格

③除

在 scss 中， /的作用，除了除法，还做分隔符
运算符两边有变量或者方法返回值，是除法
运算式被小括号包裹，是除法
除法运算式，是其它运算式的一部分，是除法

三.函数---scss 函数，与 js 函数很类似

1.数学函数

`round($v1)` 四舍五入

`ceil($v1)` 向上取整

`floor($v1)` 向下取整

`max($v1,$v2,$v3....)` 找出最大值

`min($v1,$v2,$v3....)` 找出最小值

`random()` 随机数

2.字符串函数

`unquote("abc")` 去除双引号

`quote(dbc)` 添加双引号

`to-upper-case("afdsfsd")` 转大写

`to-lower-case("AFDSF")` 转小写

3.自定义函数

```
@function 函数名称($a,$b){  
    @return $a+$b;  
}
```

function 和 return 关键字加@
参数加\$

四. 指令

if-else

条件可以不加双括号

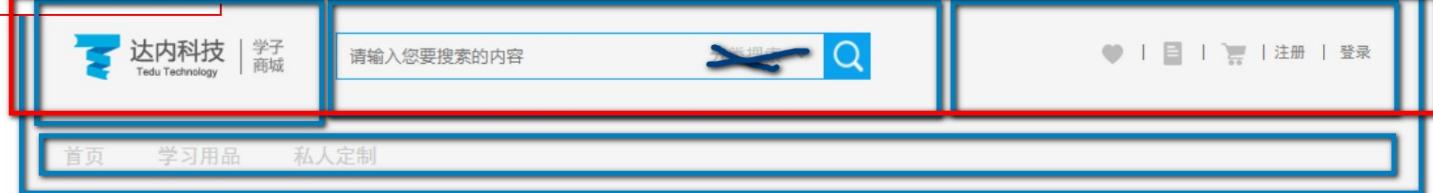
```
@if 条件{}  
@else if 条件{}  
@else{}
```

Boot 项目

要求使用 boot 完成学子页面（中间一些小细节，不要求 1:1 完成）

1.项目结构，创建项目文件夹 pro，复制图片资源和js 文件夹，css 文件夹

div.container

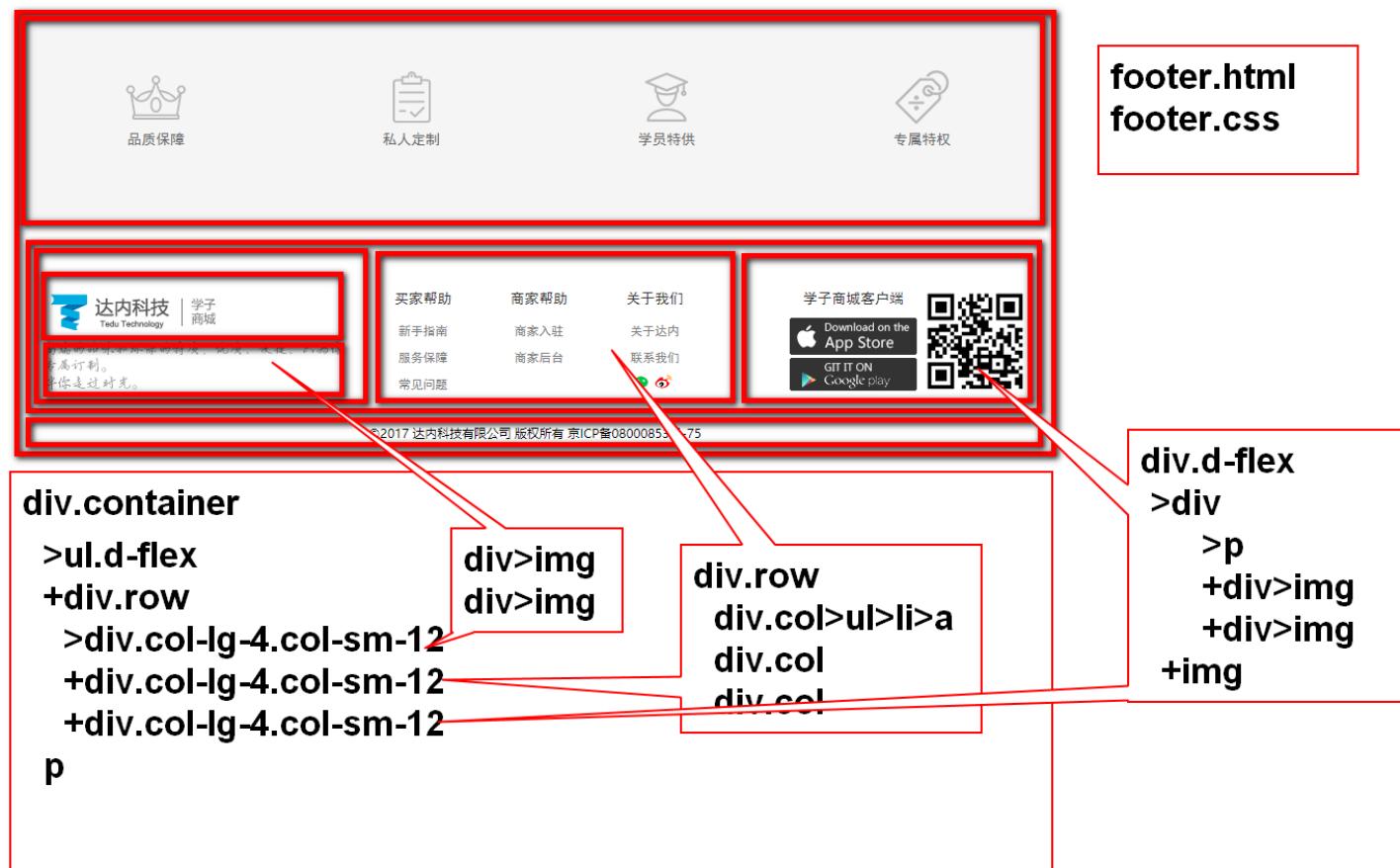


div.container

```
>div.row  
>div.col-sm-12.col-md-3 >a.navbar-brand>img  
+div.col-sm-12.col-md-5/6>div.input-group>input需要自己写css+a>img  
+div.col-sm-12.col-md-4/3>面包屑导航  
+ul.nav>li.nav-item>a.nav-link
```

使用boot完成页面。目的
1.熟练使用栅格
2.熟练使用媒体查询
3.熟悉boot样式



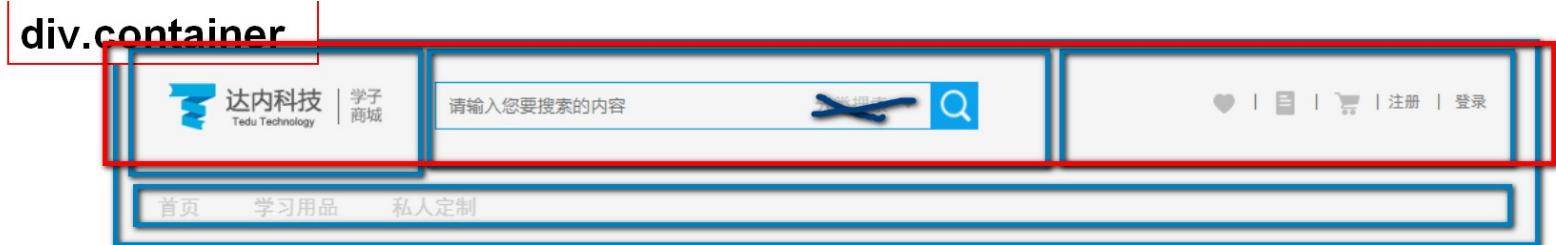


ProjectBootDay02:

Boot 项目

要求使用 boot 完成学子页面（中间一些小细节，不要求 1:1 完成）

1. 项目结构，创建项目文件夹 pro，复制图片资源和 js 文件夹，css 文件夹

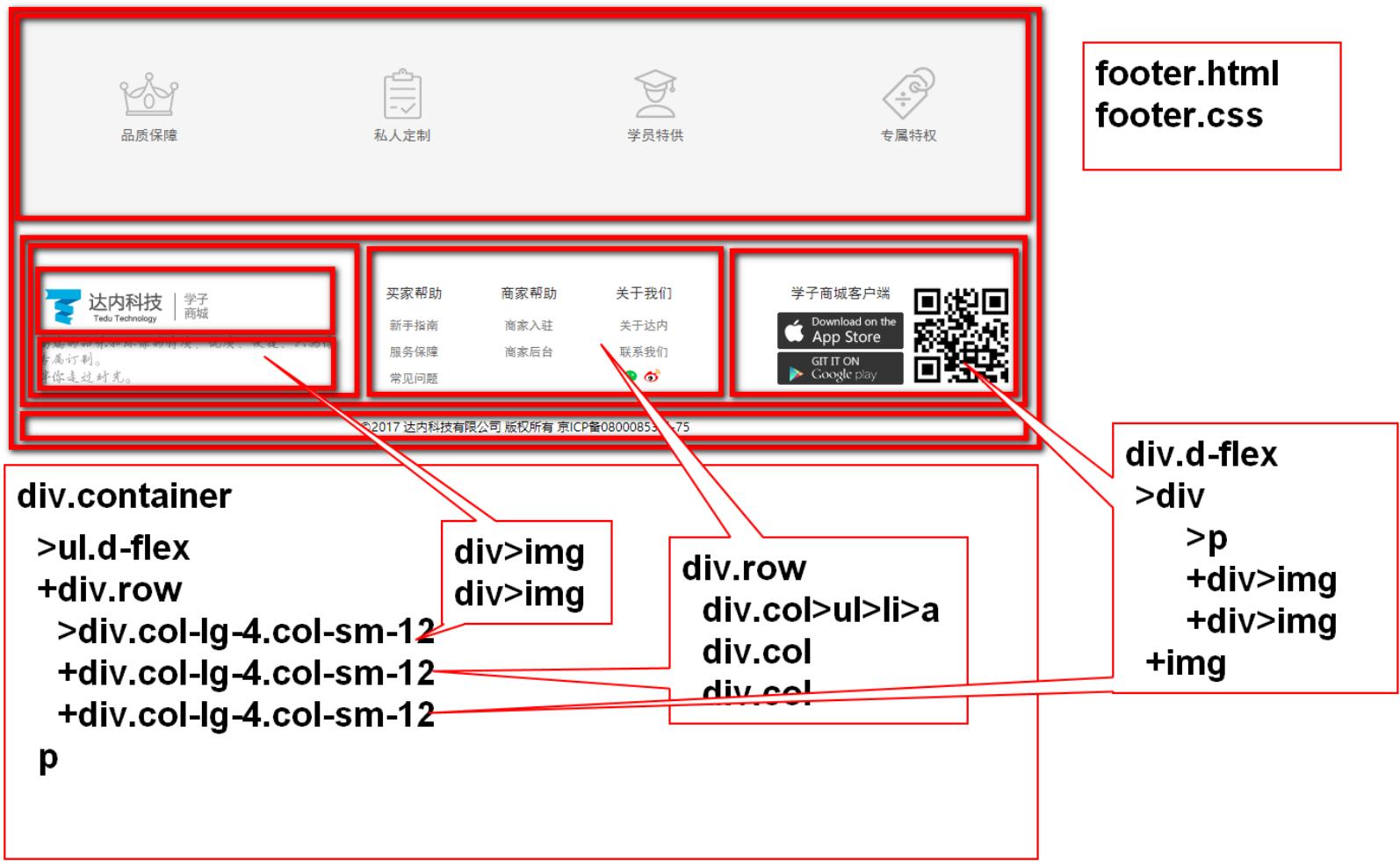


```
div.container  
>div.row  
>div.col-sm-12.col-md-3 >a.navbar-brand>img  
+div.col-sm-12.col-md-5/6>div.input-group>input需要自己写css+a>img  
+div.col-sm-12.col-md-4/3>面包屑导航  
+ul.nav>li.nav-item>a.nav-link
```

使用boot完成页面。目的
1.熟练使用栅格
2.熟练使用媒体查询
3.熟悉boot样式



2.底部



3. 轮播图和 1L

首页推荐 /1F

Apple MacBook Air 系列

酷睿双核i5处理器|256GB SSD|8GB内存|英特尔HD显卡620含共享显卡内存

¥ 6988.00

[查看详情](#)



小米Air 金属超轻薄

酷睿双核i5处理器|512GB SSD|2GB内存|英特尔HD独立显卡

¥ 3488.00

[查看详情](#)



联想E480C 轻薄系列

¥ 5399.00

[查看详情](#)



华硕RX310 金属超极本
¥ 4966.00

[查看详情](#)



联想小新700 电竞版游戏本
¥ 6299.00

[查看详情](#)



戴尔灵越燃7000 轻薄窄边
¥ 5199.00

[查看详情](#)

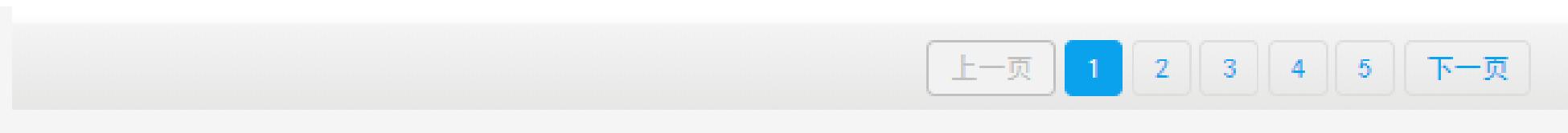


4.商品列表页。list.html list.css

导入头和脚

作业:

1.完成商品列表的页的分页



2.完成商品详情页

首页 > 学习用品 > 笔记本电脑 > 神舟战神Z7M-KP7GT

神舟(HASEE)战神Z6-KP7GT 15.6英寸游戏本笔记本电脑(i7-7700HQ 8G 1T+128G SSD GTX1050 1080P)
黑色
预约享5499抢! 【128G SSD+1T HDD】双硬盘, 春风“十”里, 期待是你!

学员售价: **¥ 5699.00**

服务承诺: *退货补运费 *30天无忧退货 *48小时快速退款 *72小时发货

客服: 联系客服

规格:

Z7M GT【i7 128G+1T GTX1050Ti】	战神Z7M【四核i7 GTX965M】
战神Z7M【四核i5 GTX965M】	Z6 GT【i5 128G+1T GTX1050】
战神G6【17.3英寸 GTX960M】	Z6 GT【i7 128G+1T GTX1050】

数量:

[立即购买](#) [加入购物车](#) [收藏](#)

The product page displays detailed information about the Hasee War God Z7M-KP7GT laptop, including its specifications, price, and purchase options. The main image shows the laptop open with Captain America's shield on the screen. Below the main image are five smaller thumbnails showing different views of the laptop. At the bottom right, there are three buttons for 'Buy Now', 'Add to Cart', and 'Favorites'.