# 李文华

## Server

### Day01介照服务器

Email：[liwenhua@tedu.cn](mailto:liwenhua@tedu.cn)

QQ：1759496573

1.如何解决遇到困难？

(1)自己分析，动手解决 —— F12

(2)请兄弟姐妹一起分析解决

(3)请项目经理帮助解决

(4)给文华老师发邮件、QQ

TopSpeeder： 350次/分钟

2.课程体系 —— 大前端/全栈工程师

阶段1（10天-8）：数据库服务器 + Web服务器

阶段2（20天-5）：HTML + CSS + JS基础

阶段3（20天-9）：JS高级 + DOM + jQuery

阶段4（20天-7）：Bootstrap + H5高级 + Vue.js

阶段5（20天-9）：移动端

阶段6（10天-8）：高阶框架

3.Windows中的常用快捷键

Windows+E：打开文件管理器

Windows+D：显示/隐藏桌面

Alt+Tab：在多窗口间切换(向右)

Alt+Shift+Tab：在多窗口间切换(向左)

Windows+R：打开“运行”窗口，继续敲入cmd、mspaint、calc、notepad….

Alt+F4：关闭当前窗口

练习：纯键盘操作，删除C:\xampp\htdocs下所有的内容

4.常用的编程快捷键

Ctrl+Alt+↓：复制鼠标当前行内容

Alt+↑/↓：向上/下移动当前行内容

Ctrl+D：删除当前行

Ctrl+/：注释当前行

练习：熟练使用上述快捷键

5.五个月中的课程项目

**课上演示项目：学子商城**

PC版的前台：www.codeboy.com/index.html

PC版的后台：www.codeboy.com/admin/login.html

**每人的专有项目：自己选题、自己实现、自己部署**

Windows系统：个人计算机，不适合用作服务器

Mac系统：个人计算机，不适合用作服务器

Linux/Unix系统：服务器操作系统

Bill Gates - Microsoft： IBM - DOS - Windows

#### 6.软件工程之：软件过程 —— 小重点

**软件定义期：**

(1)可行性研究阶段（项目经理） 技术、政策、资金、时间、人力、软硬件可行性

(2)需求分析阶段（需求分析师） 功能、性能、安全、可用性等方面的需求

**软件开发期：**

(3)概要设计阶段（架构师） 软硬件、网络、技术、功能模块、数据

(4)详细设计阶段（设计师） 模块进一步细分

(5)开发阶段（UI工程师+前端工程师+后台工程师）

(6)测试阶段（测试工程师）

**软件维护期：**

(7)部署实施阶段（运维工程师）

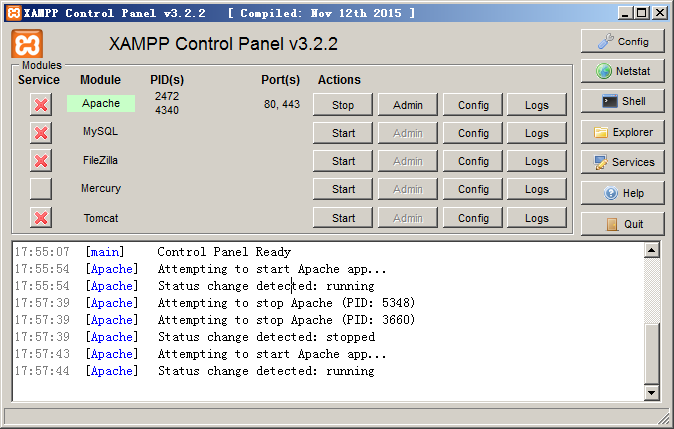
(8)维护阶段(运维/开发工程师)

7.练习：搭建自己的Web服务器

(1)服务器端：下载并安装一款Web服务器软件，如Apache Httpd，作用相当于商店里的售货员

网址：http://www.xampp.org/

(2)服务器端：启动Web服务器，保证80端口被打开，相当于售货员开始上班



(3)服务器端：编写网页，保存在特定的可以被客户端访问到的目录下，c:/xampp/htdocs(相当于商店中的货柜)

===================================

(4)客户端：通过浏览器的地址栏访问服务器

http://172.163.0.212

|  |
| --- |
| 查看自己电脑的IP地址： 运行 => cmd => ipconfig |

练习：编写一个自己的网页，保存在Web服务器上，让同桌来访问

### Day02

复习：

软件过程/生命周期——三个时期八个阶段

软件定义期：

(1)可行性分析阶段

(2)需求分析阶段

软件开发期：

(3)概要设计阶段

(4)详细设计阶段

(5)开发阶段(UI+前端+后台)

(6)测试阶段

软件维护期：

(7)部署阶段

(8)维护阶段

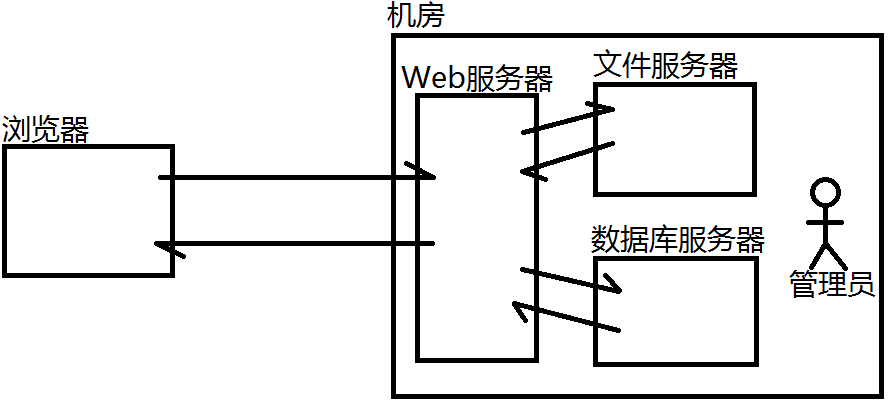
|  |
| --- |
| XAMPP = Apache + MySQL + PHP |

初次使用Web服务器：

服务器端：安装XAMPP、启动Web服务器、保存网页文件

客户端：在地址栏中输入服务器的地址加以访问

学子商城概要设计——项目部署结构图



学子商城概要设计——数据结构图

今日目标：

数据库服务器的使用 —— 比较容易理解，但有一定的单词记忆量

date：日期

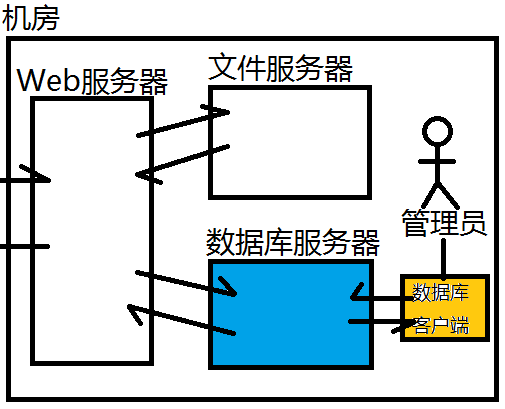
data：数据 database

1.数据库服务器

当前主流的数据库是——关系型数据库(RDBMS)。分为两部分：

**RDBMS服务器软件**：负责存储和管理数据(对于人来看都是乱码)

**RDBMS客户端软件**：负责向服务器发起增删改查命令(显示出人能读懂的操作结果)



|  |
| --- |
| 数据库服务器中的数据结构：  SERVER => DATABASE => TABLE => ROW => COLUMN |
| MySQL -> SUN / Oracle  -> MariaDB |

Deamon 精灵，守护者，精灵程序，守护程序，服务器程序

mysqld httpd ftpd

2.使用MySQL数据库

(1)服务器端：下载并安装MySQL服务器端软件

C:\xampp\mysql

(2)服务器端：启动MySQL服务器端软件

C:\xampp\mysql\bin\mysqld.exe

保证3306被打开即可

==============================

(3)客户端：下载并安装MySQL客户端软件

C:\xampp\mysql\bin\mysql.exe

(4)客户端：启动客户端，连接到服务器上去，向服务器发送各种操作指令

C:\xampp\mysql\bin\mysql.exe -uroot -p

若连接服务器时未指定用户名，则默认是来宾/匿名账户！

root后不能加分号！！！ 还要注意大小写！！

还可以向服务器导入一个脚本文件，把其中的SQL命令一次性提交给服务器： mysql -uroot < c:/f1/x.sql

#### 3.MySQL中的常用管理命令

注意：

(1)所有的SQL命令都必须以英文的分号结尾！

(2)一条命令可以书写在多行中

(3)SQL命令不区分大小写！！

(4)编写SQL命令时可以使用单行注释: #.... 还可以使用多行注释： /\* .... \*/

常用的MySQL管理命令：

(1)quit; 断开到服务器的连接

(2)show databases; 显示服务器上已有的所有数据库

(3)use 库名; 开始使用指定的数据库/进入指定的数据库

(4)show tables; 显示当前库中有哪些表

练习：用匿名账户连接到MySQL服务器，查看有哪些库，断开连接

练习：用root账户连接到MySQL服务器，查看有哪些库，断开连接

练习：用root账户连接到MySQL服务器，查看有哪些库，每个库中各有多少个表

练习：用root账户连接到MySQL服务器，查看有哪些库，每个库中各有多少个表 —— 把所有命令书写在一个文本文档中，一次性提交给服务器执行

exit 出口/退出 exist 存在

form 表单 from 从....

4.常用的SQL命令

SQL: Structured Query Language，结构化查询语言，专用于操作(增删改查)关系型数据库服务器中的数据，是ISO指定的一门行业标准语言。

**SERVER => DATABASE => TABLE => ROW => COLUMN**

今日重点——常用的SQL语句

(0)设置客户端所提及的SQL命令所用的字符集

SET NAMES UTF8;

(1)试着丢弃指定的数据库

DROP DATABASE IF EXISTS 库名;

(2)新建一个数据库

CREATE DATABASE 库名 CHARSET=UTF8;

(3)进入指定的库

USE 库名;

(4)创建一个新的数据表

CREATE TABLE 表名( 列名 类型, 列名 类型, 列名 类型);

(5)向表中插入一行数据/记录

INSERT INTO 表名 VALUES( 值, 值, 值 );

提示：一行中数据值的数量必须与表声明中列的数量相同；字符串和日期数据必须用英文单引号或双引号括起来，数字可括也不括。

(6)从表中删除一行记录

DELETE FROM 表名; #删除所有的记录行，仍保留表本身

DELETE FROM 表名 WHERE 条件子句; #只删除满足条件的记录行

(7)修改表中的一行记录——用的比较少，最不好记忆！！

UPDATE 表名 SET 列名=值, 列名=值, 列名=值 ; #修改所有的行

UPDATE 表名 SET 列名=值, 列名=值, 列名=值 WHERE 条件子句; #只修改满足条件的记录行

(8)查询表中已有的所有数据

SELECT \* FROM 表名;

练习：

试着丢弃数据库tedu，如果存在的话；

重建新的数据库tedu，使用UTF8字符集；

进入数据库tedu;

创建保存学生信息的表student，包含编号、姓名、性别、入学分数、生日；

查询出已有的表;

练习：

设置SQL语句的字符集为UTF8;

试着丢弃数据库xuezi，如果存在的话；

重建新的数据库xuezi，使用UTF8字符集；

进入数据库xuezi;

创建保存笔记本信息的表laptop，包含编号、标题、价格、购买数量、上架日期；

插入三行记录;

查询出笔记本信息表中所有的数据

课后练习：根据下面的注释编写需要的SQL语句

(1)设置SQL语句的字符集为UTF8;

(2)试着丢弃数据库sohu，如果存在的话；

(3)重建新的数据库sohu，使用UTF8字符集；

(4)进入数据库sohu;

(5)创建部门信息表dept，包含部门编号、部门名称、部门人数；

(6)插入如下的三行记录：

10 研发部 3

20 市场部 2

30 运营部 2

(7)创建员工信息表emp，包含员工编号、员工姓名、工资、入职日期、所在部门编号

(8)插入八个员工的信息，要求10号部门3名员工，20号部门2名员工，30号部门2名员工

(9)“运营部的一名员工离职”：请从员工表中删除该行记录，部门表中运营部人数-1

(10)“研发部一名员工转岗到市场部”：请修改员工表中该员工所在部门编号，再修改部门表中研发部员工数量-1，市场部员工数量+1

### day03 SQL,MySql

复习：

数据库的种类：网状、层次、关系型、非关系型

关系型数据库的结构：

服务器端：c:/xampp/mysql/bin/mysqld.exe

客户端：c:/xampp/mysql/bin/mysql.exe

关系型数据库中的数据在逻辑上的结构：

SERVER=>DATABASE=>TABLE=>ROW=>COLUMN

常用的MySQL管理命令——MySQL数据库专用：

QUIT

SHOW DATABASES;

USE 库名

SHOW TABLES;

常用SQL命令——大多数数据库通用：

DROP DATABASE IF EXISTS 库名;

CREATE DATABASE 库名 CHARSET=UTF8;

USE 库名;

CREATE TABLE 表名(列名 类型, 列名 类型);

INSERT INTO 表名 VALUES(值, 值);

DELETE FROM 表名 WHERE 条件;

UPDATE 表名 SET 列=值, 列=值 WHERE 条件;

SELECT \* FROM 表名;

练习：

根据下面的注释编写需要的SQL语句

(1)设置SQL语句的字符集为UTF8; 9:38

(2)试着丢弃数据库tedu，如果存在的话；

(3)重建新的数据库tedu，使用UTF8字符集；

(4)进入数据库tedu;

(5)创建班级信息表banji，包含bid、bname、stuCount；

(6)插入如下的三行记录：

10 WEB1706 3

20 WEB1707 2

30 WEB1708 2

(7)创建学生信息表student，包含sid、sname、score、schoolDate、bjId

(8)插入七个学生的信息，要求10号班级3名，20号班级2名，30号班级2名员工

(9)“WEB1708的一名学生休学”：请从学生表中删除该行记录，部门表中WEB1708班人数-1

(10)“WEB1706一名学生留级到WEB1707”：请修改学生表中该学生所在班级编号，再修改班级表中WEB1706数量-1，WEB1707数量+1

今日目标：

(1)SQL语句的分类 —— 了解

(2)MySQL中的列类型 —— 重点

(3)MySQL中的列约束 —— 重点

1.SQL语句的分类

SQL语句分为四大组：

**(1)DDL：Data Define Language**，数据定义语言(定义列)

包括：CREATE、DROP、ALTER、TRUNCATE

**(2)DML：Data Manipulate Language**，数据操作语言(操作行)

包括：INSERT、DELETE、UPDATE

**(3)DQL：Data Query Language**，数据查询语言

包括：SELECT

**(4)DCL：Data Control Language**，数据控制语言(控制权限)

包括：GRANT、REVOKE

|  |
| --- |
| 面试题：SQL语句中，delete和truncate区别在哪儿？ |

2.MySQL中的列类型 —— 重点

CREATE TABLE 表名 ( 列名  **列类型** )

提示：不同的数据库中列类型是不同的——MySQL中的列类型是最丰富的。

1个bit：可以保存一个0或1的空间

1个byte：字节，1个字节=8个bit

00000000

00000000 00000000

10000000 00000000 00000000 00000001

**(1)数值类型 —— 引号可用可不用**

TINYINT 微整数，1个字节， -128~127

SMALLINT 小整数，2个字节，-32768~32767

INT 整数，4个字节，-2147483648~2147483647

BIGINT 大整数，8个字节，很大的数

FLOAT(M,D) 单精度浮点数，4个字节，1234.5678 1.2346x10^38

DOUBLE(M,D) 双精度浮点数，8个字节，1.2343x10^308

DECIMAL(M,D)定点小数，可以保证精确计算

BOOL 布尔，真假类型，只能取值为TRUE/1或FALSE/0，底层等价于TINYINT(1)

**(2)日期时间类型 —— 必须用引号括起来**

DATE 日期，数据形如 '2010-5-15'

TIME 时间，数据形如 '10:22:3'

DATETIME 日期时间类型，数据形如 '2010-5-15 12:10:5'

**(3)字符串类型 ——必须用引号括起来**

CHAR(M) 定长字符串，可能产生空间浪费问题，但存取速度快，M不能超过255

VARCHAR(M) 变长字符串，不会产生空间浪费，但存取速度慢，M不能超65535

TEXT(M) 变长字符串，M不能超过4G

|  |
| --- |
| CHAR(6) 即使数据长度不够，也要用\0占位  'a\0\0\0\0\0'  'ab\0\0\0\0'  'abc\0\0\0'  'abcd\0\0'  'abcde\0'  'abcdef' |
| VARCHAR(6) 实际占用的空间可能是 字符个数+1  'a\0'  'ab\0'  'abc\0'  'abcd\0'  'abcde\0'  'abcdef' |

练习：删除并重建数据库xuezi，表laptop(lid、title、price、spec/规格、shelfTime/上架时间、isOnsale/是否特价、stockCount/库存数量)，使用恰当的列类型。试着插入4行记录。

练习：删除并重建数据库xuezi，表user(uid, uname, upwd, email, phone, avatar, userName, sex, isOnline, registerTime )，使用恰当的列类型。试着插入4行记录。

3.MySQL中的列约束

Constraint：约束，表中某个列上的数据往往都是有限制的，如sex CHAR(1)只能存“男”或“女”、score DECIMAL(4,1)不能存-500.9，phone CHAR(11)必须符合电话号码的格式要求、uname不能有重复值、upwd不能为空....

MySQL中有如下约束类型：

**(1)PRIMARY KEY**：主键约束

语法： 列名 类型 PRIMARY KEY,

声明为主键的列上的值唯一(不能出现重复值)，且不能为空(NULL)，表中的记录会自动按照主键列上的值有小到大排序——一个表中至多只能有一个列声明为主键列。

**(2)UNIQUE：**唯一约束

语法： 列名 类型 UNIQUE,

声明为UNIQUE的列上不允许出现重复值，但可以出现NULL，且可以有多个NULL

**(3)NOT NULL：**非空约束

语法： 列名 类型 NOT NULL,

声明为NOT NULL的列上不允许出现空值

**(4)CHECK**：检查约束

语法： 列名 类型 CHECK(条件),

例如： score DECIMAL(4,1) CHECK(score<=100 AND score>=0)

提示：MySQL默认不支持检查约束

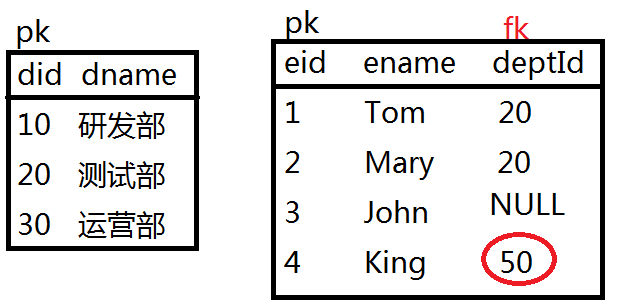
**(5)FOREIGN KEY：**外键约束

语法：

deptId INT,

FOREIGN KEY(deptId) REFERENCES dept(did)

声明为外键的列上出现的值必须在另一个表的主键列上存在！



注意：外键列上的值可以重复、有可以为空。有的项目设计师不喜欢用外键——因为它会降低INSERT/UPDATE语句的执行速度。

**(6)DEFAULT：**默认值约束

语法：列名 类型 DEFAULT 值,

若某个列上未指定明确的值，将自动使用默认值

CREATE TALBE user(uid INT, sex CHAR(1) DEFAULT '男');

INSERT INTO user VALUES(1, '男');

INSERT INTO user VALUES(2, '女');

INSERT INTO user VALUES(3, DEFAULT); #3号用户姓别为男

INSERT INTO user(uid) VALUES(4); #4号用户性别为男

|  |
| --- |
| 面试题：请问一列声明为PRIMARY KEY 和 声明为 UNIQUE NOT NULL的区别？ |

|  |
| --- |
| AUTO\_INCREMENT：  MySQL数据库专有的关键字，只能用于INT PRIMARY KEY列上，用于实现“自增列”，若赋值为NULL，则会查找当前已有的最大值，+1用于新的记录行。 |

练习：

删除并重建数据库tedu

创建部门信息表： dept

did 部门编号，主键

dname 部门名称，非空

location 部门所在地，非空，默认值为‘北京’

empCount 员工数量，非空，默认值为 0

创建员工信息表： emp

eid 员工编号，主键

ename 员工姓名，非空

idCardNum 身份证号，非空，唯一

salary 默认值为NULL

deptId 所在部门的编号，必须参考部门表的编号

课后练习：

删除并重建数据库xuezi；

创建笔记本型号表laptop\_family，包含如下列：

fid 型号编号，自增主键

fname 型号名称，非空

laptopCount 该型号下笔记本的数量，非空，默认值为1

插入3个笔记本型号，如“戴尔燃7000”、“联想小新”、“MacBook”

创建笔记本信息表laptop，包含如下列：

lid 笔记本编号，自增主键

title 标题，非空

price 价格，非空，默认值为99999.99

pic 图片，非空，默认值为'img/default.png'

isOnsale 是否特价，非空，默认为FALSE

familyId 所属型号编号，外键，参考型号表的主键

插入若干条笔记本信息，并尝试违反各种约束的插入语句。

### Day04 简单和复杂查询

复习：

数据库和SQL语句

(1)DDL: CREATE/DROP/ALTER/TRUNCATE

(2)DML: INSERT/DELETE/UPDATE

(3)DQL: SELECT

(4)DCL: GRANT/REVOKE

|  |
| --- |
| 课下任务：  MySQL的管理员root，如果设置登录密码 |
| 如何MySQL创建新的管理员账户并设置权限 |

MySQL中的列类型：

数值类型：

TINYINT/SMALLINT/INT/BIGINT

FLOAT/DOUBLE/DECIMAL

BOOL

日期时间类型：

DATE/TIME/DATETIME

字符串类型：

CHAR(M)/VARCHAR(M)/TEXT(M)

MySQL中的列约束：

(1)PRIMARY KEY

(2)UNIQUE

(3)NOT NULL

(4)CHECK

(5)FOREIGN KEY(列名) REFERENCES 表名(列名)

(6)DEFAULT 值

今日目标：

(1)简单查询语句 —— 必须当日掌握！

(2)复杂查询语句 —— 难点！尽快掌握！

|  |
| --- |
| 高阶自学题： 数据库设计范式  根据设计范式自己设计“学子商城”需要的表结构 |

1.计算机中如何保存时间？

使用一个大的整数表示时间，如：1490123456789

值表示指定的日期时间距离“计算机元年”经过了多少毫秒。

如： 0代表1970-1-1 0:0:0

1000代表1970-1-1 0:0:1

3600\*1000代表1970-1-1 1:0:0

24\*3600\*1000代表1970-1-2 0:0:0

365\*24\*3600\*1000代表1971-1-1 0:0:0

项目中的日期时间数据都使用大整数（BIGINT）来表示日期时间。不推荐使用DATE/DATETIME格式，因为不同国家语言的人习惯不同：'2017-8-10'、'8-10-2017'、'10/8/2017'

2.简单查询 —— 查询特定的列

语法：

SELECT 列名, 列名, 列名

FROM 表名;

示例：查询出所有员工姓名及其工资

SELECT ename,salary FROM emp;

练习：查询出所有员工的生日、姓名、编号

SELECT birthday,ename,eid FROM emp;

3.简单查询 —— 查询所有的列

语法：

SELECT \* FROM 表名;

提示：该语句真实项目中较少使用

4.简单查询 —— 给列取别名

语法：

SELECT 列名 [AS] 别名, 列名 [AS] 别名

FROM 表名;

示例：查询所有员工姓名，表头显示为“姓名”

SELECT ename AS 姓名 FROM emp;

练习：查询出员工编号、姓名、工资，列名全部用中文呈现

SELECT eid AS 编号, ename n, salary '工 资' FROM emp;

5.简单查询 —— 只显示列上不同的值

语法：

SELECT DISTINCT 列名

FROM 表名;

示例：查询所有员工都处于哪些部门中

SELECT DISTINCT deptId FROM emp;

练习：查询出所有员工工资共有多少种

SELECT DISTINCT salary FROM emp;

6.简单查询 —— 查询时执行算术运算

示例：SELECT 1+2;

示例：查询出每个员工姓名及其年薪

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

练习：公司决定给每人月薪+200，每人再加5000年终奖，查询出每个员工的姓名及其年终所拿到的总薪水

SELECT ename, (salary+200)\*12+5000 AS 年薪 FROM emp;

7.简单查询 —— 查询结果的排序

语法：

SELECT ....

FROM 表名

ORDER BY 列名 [ASC|DESC], 列名 [ASC|DESC];

ascendant 升序 descendant 降序

示例：显示所有员工的编号、姓名，按编号降序显示

SELECT eid,ename FROM emp ORDER BY eid DESC;

练习：显示所有员工编号、姓名、工资，按工资由小到大排序

SELECT eid,ename,salary FROM emp ORDER BY salary;

练习：显示所有员工编号、姓名、工资，按姓名由大到小排序

SELECT eid,ename,salary FROM emp ORDER BY ename DESC;

练习：显示所有员工编号、姓名、工资，按工资由小到大排序、若工资相同，再按照姓名由大到小排序

SELECT eid,ename,salary FROM emp ORDER BY salary ASC, ename DESC;

8.简单查询 —— 对查询到的记录进行筛选/条件查询——小难点

语法：

SELECT ....

FROM 表名

WHERE 条件判定;

示例：查询出编号为8的员工的所有信息 —— 相等查询

SELECT \* FROM emp WHERE eid=8;

练习：查询出姓名为KING的员工所有信息

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

练习：查询出部门编号为10的员工所有信息

SELECT \* FROM emp WHERE deptId=10;

练习：查询出部门编号为10的员工所有信息，按工资降序排列

~~SELECT \* FROM emp ORDER BY salary DESC WHERE deptId=10;~~  #错误写法

SELECT \* FROM emp WHERE deptId=10 ORDER BY salary DESC;

示例：查询出工资大于等于6000的员工的所有信息 —— 不等查询

SELECT \* FROM emp WHERE salary >=6000;

练习：查询出1991年之前出生的员工的所有信息

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

示例：查询出不在10号部门的员工的所有信息 —— 不等查询

SELECT \* FROM emp WHERE deptId != 10;

SELECT \* FROM emp WHERE deptId <> 10;

9.简单查询 —— 多条件查询

(并且)语法：

SELECT ....

FROM ...

WHERE 条件1 **AND** 条件2;

示例：查询出工资在6000(包含)~7000(不包含)之间的员工所有信息

#SELECT \* FROM emp WHERE salary>=6000;

#SELECT \* FROM emp WHERE salary<7000;

SELECT \* FROM emp WHERE salary>=6000 AND salary<7000;

练习：查询出所有在1990年出生的员工所有信息

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

上述语句的变种： 列名 BETWEEN 值1 AND 值2

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

练习：查询出工资大于8000的10号部门的员工所有信息

SELECT \* FROM emp WHERE salary>8000 AND deptId=10 AND sex='男' AND addr='北京';

(或者)语法：

SELECT ....

FROM ...

WHERE 条件1 **OR** 条件2;

示例：查询出部门编号为10或30的员工所有信息 15:05

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

练习：查询出工资小于等于6000 **以及** 大于等于10000的员工信息

SELECT \* FROM emp

WHERE salary<=6000 OR salary>=10000;

练习：查询出工资大于8000 **以及** 在10号部门的所有员工信息

SELECT \* FROM emp

WHERE salary>8000 OR deptId=10;

练习：查询出所有1990年 **和**1992年出生的员工所有信息

SELECT \* FROM emp

WHERE

( birthday>='1990-1-1' AND birthday<='1990-12-31')

OR

( birthday BETWEEN '1992-1-1' AND '1992-12-31' );

练习：查询出10、30、40号部门的员工所有信息

SELECT \* FROM emp

WHERE (deptId=10) OR (deptId=30) OR (deptId=40);

上述语句等价于： WHERE 列名 IN (....)

SELECT \* FROM emp

WHERE deptId IN (10, 30, 40);

(取反/非)语法：

SELECT ....

FROM ...

WHERE NOT 条件1 ;

练习：查询出不在10、30、40号部门的员工所有信息

SELECT \* FROM emp

WHERE (deptId!=10) AND (deptId!=30) AND (deptId!=40);

等价于

SELECT \* FROM emp

WHERE deptId NOT IN (10, 30, 40);

练习：查询出所有不在1990和1992出生的员工的所有信息

SELECT \* FROM emp

WHERE

NOT

(

( birthday>='1990-1-1' AND birthday<='1990-12-31')

OR

( birthday BETWEEN '1992-1-1' AND '1992-12-31' )

);

示例：查询部门编号为NULL的员工所有信息

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

SELECT \* FROM emp WHERE deptId IS NULL;

10.简单查询 —— 条件查询之——模糊查询

语法：

SELECT ...

FROM ...

WHERE 列名 LIKE 模糊匹配表达式；

模糊表达式一般都是字符串类型：

WHERE title LIKE '戴尔'; #标题必须等于“戴尔”

WHERE title LIKE '%戴尔'; #%匹配任意多个任意字符

WHERE title LIKE '戴尔%'; #%匹配任意多个任意字符

WHERE title LIKE '%戴尔%'; #%匹配任意多个任意字符

WHERE title LIKE '\_戴尔'; #\_匹配任意一个任意字符

WHERE title LIKE '\_戴尔\_\_'; #\_匹配任意一个任意字符

练习：查询出所有姓名中带字符E的员工所有信息

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

练习：查询出所有姓名以字符E开头的员工所有信息

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

练习：查询出所有姓名倒数第二个字符是E的员工所有信息

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

练习：查询出姓名中包含字符E或A的员工所有信息

SELECT \* FROM emp

WHERE ename LIKE '%E%' OR ename LIKE '%A%';

11.简单查询 —— 分页查询 —— 重点

若要查询的满足条件的数据行太多，不方面一次性全部显示给用户，可以“一页一页”的显示。分页查询在不同的数据库中语法不同。

语法：

SELECT ...

FROM ...

WHERE ....

ORDER BY ....

LIMIT start, count ;

start是一个数字，表从哪一行开始获取数据

count是一个数字，表一次最多读取多少行记录

假设：每页最多只显示5条记录

第1页： ... LIMIT 0, 5;

第2页： ... LIMIT 5, 5;

第3页： ... LIMIT 10,5;

....

第n页： ... LIMIT (n-1)\*5, 5;

示例：查询所有员工，每页显示5条记录，要第1页

SELECT \* FROM emp LIMIT 0, 5;

示例：查询所有员工，每页显示5条记录，要第2页

SELECT \* FROM emp LIMIT 5, 5;

示例：查询所有员工，每页显示5条记录，要第3页

SELECT \* FROM emp LIMIT 10, 5;

示例：查询所有员工，每页显示5条记录，要第4页

SELECT \* FROM emp LIMIT 15, 5;

练习：查询出姓名中包含字符E的员工的信息，按姓名升序排列，每页显示3条记录，显示第1页

SELECT \* FROM emp WHERE ename LIKE '%E%' ORDER BY ename LIMIT 0,3;

练习：查询出姓名中包含字符E的员工的信息，按姓名升序排列，每页显示3条记录，显示第2页

SELECT \* FROM emp WHERE ename LIKE '%E%' ORDER BY ename LIMIT 3,3;

练习：查询出姓名中包含字符E的员工的信息，按姓名升序排列，每页显示3条记录，显示第3页

SELECT \* FROM emp WHERE ename LIKE '%E%' ORDER BY ename LIMIT 6,3;

12.复杂查询 —— 分组查询/聚合函数

语法：MySQL提供了五个聚合函数：

COUNT( ) MIN( ) MAX( ) SUM( ) AVG( )

示例：查询所有员工数量

SELECT COUNT(eid) FROM emp; #11

SELECT COUNT(ename) FROM emp; #11

SELECT COUNT(deptId) FROM emp; #10

SELECT COUNT(\*) FROM emp; #11

练习：查询出10号部门的员工数量

SELECT COUNT(\*) FROM emp WHERE deptId=10;

示例：查询所有员工工资的最大值

SELECT MAX(salary) FROM emp;

练习：查询出所有员工中年龄最大（生日最小）的员工的生日

SELECT MIN(birthday) FROM emp;

练习：查询出所有员工工资的总和

SELECT SUM(salary) FROM emp;

练习：查询出所有员工工资的平均值(用两种写法)

SELECT SUM(salary)/COUNT(\*) AS 平均工资 FROM emp;

SELECT AVG(salary) FROM emp;

语法：

SELECT COUNT(..),SUM(..), AVG(...), MAX(...), MIN(...)

FROM 表名

GROUP BY 列名;

示例：查询出每个部门的编号，及该部门下员工的数量

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

练习：查询出每个部门的编号、平均工资、最大工资、最小工资

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

FROM emp GROUP BY deptId;

13.复杂查询 —— 子查询 —— 难点！

示例：查询出‘Development’部门所有的员工信息

步骤1：查询Development部门的编号 =>10

SELECT did FROM dept WHERE dname='Development'; #10

步骤2：查询10号部门中员工的所有信息

SELECT \* FROM emp WHERE deptId=10;

整合步骤1和步骤2：

SELECT \* FROM emp WHERE deptId=(

SELECT did FROM dept WHERE dname='Development'

);

练习：查询出工资大于SMITH的工资的员工所有信息

练习：查询出比King年长的员工的所有信息

练习：查询出工资 比所有员工平均工资 大的员工的所有信息

课后练习：

(1)上述三个练习

(2)把笔记中所有的示例、练习中的SQL语句全部删除，根据注释重新写一遍。

## PHP

### Day01跨表查询,Web服务器,PHP入门

理解 => 仿写 => 默写

理解 => 仿写 => 默写

....

理解 => 仿写 => 默写

复习：

见思维导图

练习:

(1)查询出编号比 (KING的员工编号) 大的员工的所有信息

步骤1：查询KING的员工编号 =>7

SELECT eid FROM emp WHERE ename='KING';

步骤2：查询编号比7大的员工所有信息

SELECT \* FROM emp WHERE eid>7;

综合：

SELECT \* FROM emp WHERE eid>(

SELECT eid FROM emp WHERE ename='KING'

);

(2)查询出 与KING在同一部门 的员工所有信息

步1：查询KING的部门编号 =》10

SELECT deptId FROM emp WHERE ename='KING';

步2：查询部门编号为10的员工信息

SELECT \* FROM emp WHERE deptId=10;

综合：

SELECT \* FROM emp WHERE deptId=(

SELECT deptId FROM emp WHERE ename='KING'

);

(3)查询出 比KING年长 的员工所有信息

SELECT \* FROM emp WHERE birthday<(

SELECT birthday FROM emp WHERE ename='KING'

);

今日目标：

(1)补充：复杂查询——跨表查询 掌握

(2)WEB服务器概述 了解

(3)PHP语言入门 重点&难点

#### 1.复杂查询 —— 跨表查询/多表查询

语法SQL-92：——必须有WHERE相等判定，否则会产生笛卡尔积

SELECT ....

FROM 表1, 表2

WHERE 表1.列1 = 表2.列2;

**注意：SQL-92标准中的跨表查询语法可能产生"数据缺失"**

示例：查询所有员工姓名及其所在部门名称

~~SELECT ename,dname FROM dept, emp;~~ #产生笛卡尔积

|  |
| --- |
| 从两个集合中任意取出一个元素于另一个集合中的任一元素组合，方案数共有 M x N 个 —— 笛卡尔积 |

SELECT ename,deptId, did, dname

FROM dept, emp

WHERE deptId=did; #此处的判定条件必须有！

语法SQL-99： 内连接查询 INNER JOIN ... ON ...

SELECT ename, dname

FROM emp INNER JOIN dept

ON emp.deptId = dept.did;

注意：内连接查询与SQL-92标准中的查询效果一样

语法SQL-99： 左外连接查询 LEFT OUTER JOIN ... ON ...

SELECT ename, dname

FROM emp LEFT OUTER JOIN dept

ON emp.deptId = dept.did; #小手指向左边表

注意：左外连接会把左边表中所有记录都显示

语法SQL-99： 右外连接查询 RIGHT OUTER JOIN ... ON ...

SELECT ename, dname

FROM emp RIGHT OUTER JOIN dept

ON emp.deptId = dept.did; #小手指向右边表

注意：左外连接会把右边表中所有记录都显示

语法SQL-99： 全连接查询 FULL JOIN ... ON ...

SELECT ename, dname

FROM emp FULL JOIN dept

ON emp.deptId = dept.did; #小手指向左边和右边表

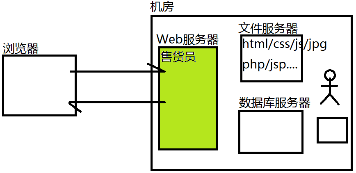
注意：全连接会把左边变/右边表中所有记录都显示——MySQL不支持全连接。可以使用左连接合右连接查询结果集的“**合并”**来代替实现。

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

UNION

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

#### 2.WEB服务器



WEB服务器的作用：等待着....接收客户端的请求，理解该请求查找必须的请求内容，返回给客户端。

**静态Web服务器：**

提供的内容任何人任何时间访问都是一样的

静态网页技术：HTML/CSS/JS/图片/Flash....

**动态Web服务器：**

提供的内容不同人不同时间访问都可能不同，一般都需要访问数据库、其他系统、进行复杂运算....

静态网页技术：JSP/ASP.NET/PHP/Node.js

#### 3.PHP Web服务器的搭建

(1)服务器端：下载并安装一款静态Web服务器，如ApacheHttpd

C:\xampp\apache\bin\httpd.exe

(2)服务器端：下载并安装PHP处理器

C:\xampp\php\php.exe

(3)服务器端：编写并保存PHP网页在服务器的特定目录下

C:\xampp\htdocs

(4)服务器端：启动Web服务器

============================

(5)客户端：使用浏览器访问服务器上的内容

http://localhost

|  |
| --- |
| 关于计算机的地址：  外网/公网地址：可以在百度中输入“IP地址”进行查看  内网/局域网地址：可以在cmd输入ipconfig命令进行查看  指向当前计算机/本机的地址： 127.0.0.1  指向当前计算机/本机的域名： localhost |

|  |
| --- |
| 一个计算机可以打开的端口范围：1~65535  1024以下一般是通用的协议所用端口 |

练习：编写一个3.php，向客户端浏览器输出自己的姓名(中文)、自己的生日、联系电话

提示：网页中的换行是 '<br>'，空格是 '&nbsp;'

练习：编写一个4.php，向客户端输出如下内容：

1\*1=1

2\*1=2 2\*2=4

3\*1=3 3\*2=6 3\*3=9

4.面试题：如何自学一门新的编程语言？

(1)背景知识：历史、特点、应用领域

(2)搭建环境，写出HelloWorld

(3)变量和常量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用小程序，练习语法

(8)函数和对象

(9)常用的第三方组件、工具、库、框架

(10)实用小项目

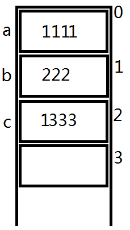
5.PHP中的变量和常量

变量：Variable，用于保存可变化的值的存储空间。本质就是计算机中一段内存的编号别名

a = 111;

b = 222;

c = a+b;



声明变量： $变量名 = 值 ;

输出变量的值： echo( $变量名 );

|  |
| --- |
| PHP中的变量名必须一直带着$  变量名中严格区分大小写，如 $age、$Age、$AGE是不一样的  变量名中可以由数字、字母、下划线组成，但不能以数字开头  $age、$name2、$\_pageCount3 合法的  $2name 非法的  变量名中可以包含多个单词，推荐使用下划线命名法或驼峰命名法  $userlastname 不推荐  $user\_last\_name 下划线命名法/匈牙利命名法  $userLastName 驼峰命名法 |

练习：编写6.php，创建一个变量保存笔记本的标题，赋值为'戴尔燃7000'；再创建一个变量保存笔记本的价格，赋值为7688；再创建一个变量保存笔记本的上架时间，赋值为'2010-5-10 22:18:33'，输出上述笔记本相关的三个变量的值。

练习：编写7.php，创建一个变量保存用户的登录名，赋值为'dingding'；再创建一个变量保存登录密码，赋值为'123456'；再创建一个变量保存用户头像图片路径，赋值为'img/u1.png'，输出上述用户相关的三个变量的值。 再次重新赋值上述三个变量，再次输出。

Constant：常量，一旦赋值就不能再改变的量。

定义常量： const 常量名 = 值 ;

输出常量的值： echo( 常量名 );

|  |
| --- |
| 为了和变量名相区分，习惯上常量名都用纯大写 |

练习：创建一个常量保存银行账户的日取款限额，值为20000，输出该常量；再试着修改该常量的值，再次输出。

6.PHP中的数据类型 —— 重点

PHP中的数据类型分为三大类八小种：

**(1)标量类型/值类型**

int/integer：整数

float/double: 小数类型，PHP中的float等同于double

string: 字符串，必须使用单引号或双引号括起来

bool/boolean: 布尔类型，取值为true/TRUE、false/FALSE

**(2)复合类型**

array

object

**(3)其它类型**

null/NULL

resource

7.PHP中的整数类型

$myInt = 2147483647; #int

$myInt = 2147483648; #float

由此可知，PHP中的int类型使用4个字节来存储，最大到2147483647。

$i1 = 123; #十进制整数 1/2/3/4/5/6/7/8/9/10/11...

$i2 = 0123; #八进制整数 1/2/3/4/5/6/7/10/11/12...

$i3 = 0x123; #十六进制整数1/2/3/4/5/6/7/8/9/a/b/c/d/e/f/10/11...

课后练习：

(1)创建多个变量保存学子商城中一个用户需要的所有数据，包括用户编号、登录名、登录密码、邮箱、手机号、性别、头像图片路径、当前是否在线、注册时间等数据，输出这些数据的值；

修改用户的登录密码、当前是否在线、注册时间等变量的值，重新输出所有变量的值。

(2)自学PPT中变量类型转换相关知识，完成下面的练习：

创建一个字符串变量，用于保存一个订单总金额；

将上述变量值+10，表示加上运费后的总金额，再显示计算后的结果；

注释上一行代码后，使用var\_dump函数检查总金额的数据类型；

尝试把金额变量解析为一个数值，再重复执行上一步程序，观察结果。

### Day02 PHP运算符,三目

复习：

|  |
| --- |
| 第一代语言：机器码01001  第二代语言：汇编语言  第三代语言：C、C++、C#、Java、PHP .......  第四代语言：SQL  第五代语言：人工智能 |

SQL：结构化查询语言，专用于操作关系型数据库

**(1)DDL：Data Define Language**，定义数据的结构-列

CREATE / DROP / ALTER / TRUNCATE

**(2)DML：Data Manipulate Language**，操作数据-行

INSERT / DELETE / UPDATE

**(3)DQL：Data Query Language**，查询数据-不影响数据

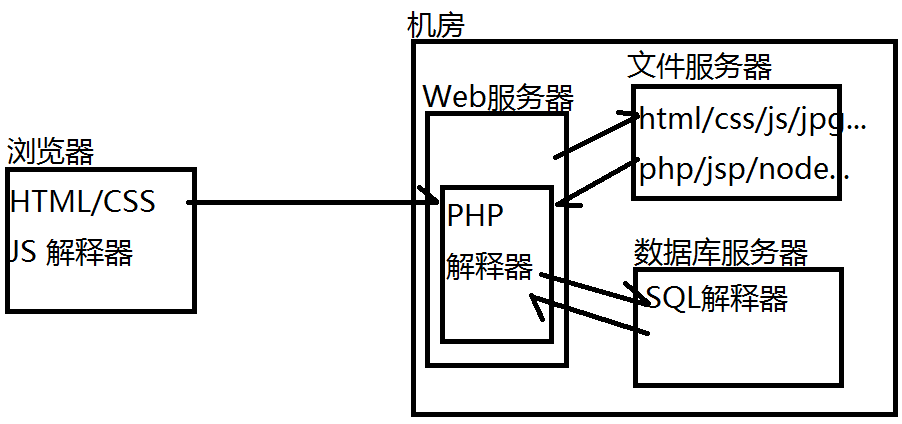
SELECT

**(4)DCL：Data Control Language**，控制用户的权限

GRANT(授权) / REVOKE(收权)

PHP：超文本预处理语言，常用于编写动态Web服务

|  |
| --- |
| 百度面试题：SQL中DELETE和TRUNCATE异同？ |
| 自学：如何给MySQL的root添加密码？如何创建一个新的用户？如何给授权和收权？ |
| 阿里面试题：各种浏览器的内核的名称 |



学习新语言的步骤：

(1)了解背景

(2)搭建环境

(3)变量常量

$userAge = 20;

echo $userAge;

const PI = 3.14;

echo PI ;

(4)数据类型

标量类型： int / float / string / bool

复合类型： array / object

其它类型： null / resource

(5)运算符

(6)逻辑结构

(7)通用小程序

(8)函数对象

(9)第三方组件、库、框架

(10)实用小项目

|  |
| --- |
| 提示：有基础学员可以参看“PHP手册”—— 语言参考章 |

今日目标：

(1)PHP中的运算符

1.PHP中的运算符 —— 算术运算

+ - \* / %(取余/求模) ++ --

%：表示求余运算

++： 表示自加1，运行效率快于普通的加法运算 $i=10; $i = $i+1; 等价于 $i++;

--：表示自减1

练习：创建一个变量，表示月薪6000；另一个变量表示每月奖金500；另一个变量表示年终奖15000，最后计算出一年的总薪资值，并输出。

练习：计算 下列表达式的值：

123+456

'ABC'+'XZY'

'ABC'+123

'ABC'+'123'

'5ABC' + 123

'5.5ABC6'+123

'5.5.5ABC'+'123a4'

'ABC'+true

'ABC'+false

结论：

(1)+只能用作算术的加法，不能用于字符串拼接

(2)算术运算作用于字符串会自动将字符串转换为数字——隐式转换

(3)算术运算作用于bool会自动将true转换为1；false转换为0——隐式转换

练习：创建一个字符串变量表示商品1的单价，一个字符串变量表示商品1的数量；创建一个字符串变量表示商品2的单价，一个字符串变量表示商品2的数量；计算出购物车的总金额，并输出

练习：创建一个变量1值为1234567，一个变量2值153，判定变量2能否整除变量1 —— 不能；

创建一个变量1值为1234557，一个变量2值153，判定变量2能否整除变量1 —— 能

练习：指出下列表达式的输出结果

$i = 10;

$i++;

echo $i; #11

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

$i = 10;

$j = $i++; //先取值再自加

echo $i, $j; //11, 10

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

$i = 10;

$j = ++$i; //先自加再取值

echo $i, $j; //11, 11

练习：口算下列代码的输出结果

$a = 10;

$b = $a++ + ++$a + $a++;

echo $a, $b;

练习：口算下列代码的输出结果

$a = 10;

$b = 10;

$c = $a++ + $b-- + ++$a + --$b ;

echo $a, $b, $c;

|  |
| --- |
| SQL中没有==，=既表示赋值又表示比较：  UPDATE emp SET age=20 WHERE eid=5; |
| PHP/JS/JAVA中，=表赋值，==表示比较  $age = 10 把10赋值给age  $eid==5 判定eid与5是否相等 |

2.PHP中的运算符——比较运算

> >= < <= != == ===(全等于) !==(不全等于)

比较运算的结果是：true 或 false

注意：==和!=会进行隐士类型转换，先统一类型再判定；===和!==不会进行类型转换，只要类型不同直接判定为不等！

$age = 22;

echo $age>18;

练习：创建变量，赋值为下列值，输出下列比较运算的结果

123 > 33

'123' > '33'

'123' == 123 string=>int隐式转换

1 == true bool=>int隐式转换

true != false

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

'abc' == true string=>bool，""和"0"转换为false，其它转换为true

0 == null PHP中0和null可以判定为相等

null == null PHP中null和null可以判定为相等

|  |
| --- |
| PHP: '123' > '33'  比较运算会进行隐式转换，string=>int |
| JS: '123' < '33'  比较运算会进行隐式转换，比较的字符的编码大小 |

3.PHP中的运算符 —— 逻辑运算

&& || !

&&：与/并且/并, 条件1 && 条件2

||：或，条件1 || 条件2

!：非/取反， !条件1

练习：创建一个变量保存员工的年龄，输出该员工是否适合于继续工作。年龄在18~65之间适合，否则不适合。

练习：创建一个变量表示年份，页面中输出这个年份是否为闰年。

注：闰年的判断公式满足下列两个条件之一即可

(1)年份能被4整除，且不能被100整除的是闰年。

(2)年份能被400整除的是闰年

( ()&&() ) || ()

练习：创建一个变量表示用户输入的用户名，一个变量表示用户输入的密码，如果用户名为dingding，密码为123456，输出该用户可以成功登录吗：1

|  |
| --- |
| 难理解的知识点：“短路”逻辑运算    短路：一个前置条件已经决定了最终的结果，导致后置的条件无需执行。  与运算： false && 条件2 => false  或运算： true || 条件2 => true  上述两种特殊情形下，条件2无需执行——条件2被短路  练习：  $i = 10;  $j = 10;  $k = $i -- < 0 && $j++>0; // i? j? k?  练习：  $i = 10;  $j = 10;  $k = $i -- > 0 || $j++>0; // i? j? k?  练习：—— 很难！使用短路逻辑实现if...else....  1、价格打折：创建变量保存商品总金额：如果金额>=100元，则享受八折。输出最终实际应收金额  2、创建变量保存用户的一段留言：如果有内容，变量值就是留言的内容；否则就赋值为“主人很懒，什么也没留下” |

4.PHP运算符 —— 位运算符 —— 了解

<<(左移运算) >>(右移运算)

面试题：如何计算一个数\*8最快？ D

A. $i = $i \* 8;

B. $i \*= 8;

C. $i << 8;

D. $i << 3;

结论：

若对一个数字的二进制表示左移1位，相当于x2^1

若对一个数字的二进制表示左移2位，相当于x2^2

若对一个数字的二进制表示左移3位，相当于x2^3

若对一个数字的二进制表示左移4位，相当于x2^4

若对一个数字的二进制表示左移5位，相当于x2^5

....

5.PHP运算符 —— 赋值运算符

= += -= \*= /=(除等) %=(模等)

示例：

$age = 10;

$age += 10; 等价于 $age = $age + 10;

$age -= 10; 等价于 $age = $age - 10;

$age \*= 10; 等价于 $age = $age \* 10;

$age /= 10; 等价于 $age = $age / 10;

$age %= 10; 等价于 $age = $age % 10;

6.PHP运算符 —— 拼接运算符

**. .=**

示例：

$msg = 'hello' . 123 ;

$msg .= 'world' ; 等价于 $msg = $msg . 'world';

|  |
| --- |
| 目：指一个运算符可以操作的变量的数量  单目运算： $i++  双目运算： $i + $j  三目运算： $a ? $b : $c |

7.PHP运算符 —— 三目运算符 —— 重点

语法： 表单式1 **?** 表达式2 **:** 表达式3

含义： **若表达式1为true，整个表达式的值为表达式2的值；否则整个表达式的值为表达式3的值。** if....else.....

示例：

$age = 30;

$result = $age>=18 ? "成年人" : "未成年人";

echo $result;

练习：创建两个变量，分别保存一个整数，使用三目运算输出这两个数中较大的那个的值。

练习：使用一个bool变量保存员工是否已婚，根据此值输出：该员工婚姻状况：已婚

课后练习：

(1)删除所有的示例代码，根据注释重写出来

(2)创建一个变量表示员工工资，按如下规则在页面中输出该工资金额的级别：

(1)大于等于20000，显示“高工资”

(2)小于20000大于等于8000，显示“中高工资”

(3)小于8000，显示“普通工资”

(3)体重健康指数（克莱托指数）计算公式如下：

体重(kg)÷身高²(m)

20-25正常，20以下偏瘦，25以上偏胖。

例如：某人是60kg，1.7m，那就是：

60÷1.7²=20.76，属于“正常”体重；

创建变量保存体重和身高，输出判定结果。

### Day03 PHP选择,循环

有基础学员： PHP访问MySQL

无基础学员： 理解 => 仿写 => 默写

复习：

自学新语言的基本步骤：

(1)了解背景

(2)搭建环境

(3)变量常量

变量： $变量名 = 值; 弱类型语言

常量： const 常量名 = 值;

(4)数据类型

标量类型： int float bool string

复合类型： array object

其他类型： null resource

(5)运算符

算术运算： + - \* / % ++ --

比较运算： > >= < <= == != === !==

逻辑运算： && || !

位运算： << >>

赋值运算： = += -= \*= /= %= .=

字符串拼接： . .=

三目运算： 表达式1 ? 表达式2 : 表达式3

(6)逻辑结构

(7)通用小程序

(8)函数对象

(9)第三方组件

(10)小项目

今日目标：

(1)选择结构：if...else... switch...case...

(2)循环结构：while.... do....while.... for...

|  |
| --- |
| 计算机程序的三种基本逻辑结构：  (1)顺序执行  (2)选择执行  (3)循环执行 |

1.选择执行 —— if

语法：

**语句1;**

**if( 真假判定 ){**

**语句2;**

**}**

**语句3;**

执行结果：

如果真假判定结果为true， 语句1=>语句2=>语句3

如果真假判定结果为false， 语句1=>语句3

练习：创建一个变量表示单价、表示数量、表示用户支付的金额；#1

计算商品总价，“如果总价>=500，则总价打八折”；#2

输出总价、找零金额 #3

练习：创建一个变量保存用户的签名，如果签名为空字符串("")，就赋值为“该用户很懒，什么都没写”，最后输出用户的签名

练习：创建一个变量，保存分页查询中用户想显示的页号，如果没有指定(null)或者指定了无效的值(小于等于0），就赋值为1，最后输出用户想显示的页号

2.选择执行 —— if...else....

语法：

**语句1;**

**if( 条件判定 ){**

**语句2;**

**}else{**

**语句3;**

**}**

**语句4;**

运行结果：

若条件判定为true，执行语句1=>语句2=>语句4

若条件判定为false，执行语句1=>语句3=>语句4

练习：创建一个变量保存学生的年龄，假设18岁以上且60岁以下的人才允许入学，输出该用户是否允许入学。

练习：创建一个变量保存用户登录名，用户登录密码；若用户名为'dingding'且密码为'123456'，提示“欢迎回来：xxx”,否则提示“用户名或密码有误”。

练习：创建变量保存用户的政治面貌($zzmm=1)：

1-党员、2-团员、3-群众、其它

午间练习：使用if...else...的嵌套实现：

学子商城中使用一个数字保存订单的状态：

1-等待付款、2-备货中、3-运输中、4-派送中、5-订单完成、其他状态

3.if...else...嵌套的两种形式

形式1：

if( ... ){

}else {

if( ... ){

}else{

}

}

形式2：

if( ... ){

}else if( ... ){

}else if( ... ){

}else{

}

练习：创建一个变量保存一个高考成绩，假设600分以上可以上一本、500分以上可以上二本、400分以上可以上三本、300分以上可以上专科、否则只能复读

4.选择执行 —— switch...case...

switch: 切换 case：情形

语法：

**switch( 变量名 ){**

**case 值1:**

**...;**

**break;**

**case 值2:**

**...;**

**break;**

**default:**

**...;**

**}**

在多种情形之间进行切换：

if($status===10){

}else if($status===20){

}else if($status===30){

}else {

}

上述代码还可以写作：

switch( $status ){

case 10: #if($status===10)...

....

case 20: #else if($status===20)...

.....

case 30: #else if($status===30)...

....

default: #else{ ... }

....

}

说明：if..else..比switch..case..更加的灵活，凡是switch..case..可以实现的效果，if..else...都可以实现。

**注意：switch..case..判定中默认若某个case匹配成功，则执行此case下的代码以及后续所有case下的代码，而不再执行后续的条件判定！！——使用break关键字打断后续的执行**

练习：创建一个变量保存一个订单的付款方式，可能值如下：

1-在线支付 2-货到付款 3-支付宝支付 4-京东白条 其它方式

根据变量的整数值输出对应的中文说明字符 (1)if..else (2)switch..case

5.循环执行 —— while —— 重点&小难点

while：当 ... 还成立的时候(就再次执行)

循环结果的二要素： (1)循环体 (2)循环条件

语法：

**while( 循环继续的判定条件 ){**

**循环体;**

**}**

含义：计算判定条件，为true则执行一遍循环体；再次计算判定条件，为true则再次执行一遍循环；再次计算判定条件.... —— while的循环主体代码可能执行0~N次。

示例：打印出5个※

$i = 0;

while( $i<5 ){

echo '※';

$i++;

}

练习：使用while循环打印出

0

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

1

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

2

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

...

9

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

练习：输出“■□”连续输出10次

练习：计算1+2+3...100的总和

循环条件： $i=1; $i<=100;

循环主体： $sum=0; $sum += $i;

练习：计算2+4+6+8...98的和

练习：计算 1/5 + 1/10 + 1/15 + 1/20 + ... 1/100的和

示例：从1+2+3....+N一直加下去，直到总和加到刚超过10000时，停止，那时的N是几？

循环主体： $sum += $i;

循环条件： $sum < 10000;

|  |
| --- |
| 小知识： 死循环 + break  while( true ){  .....;  if( 某个条件下 ){  break; #打断循环，退出当前循环  }  } |

6.循环执行 —— do...while —— 掌握

语法：

**do {**

**循环主体;**

**}while( 循环得以继续的条件判定 );**

示例：打印五个※

$i = 0;

do {

echo '※';

$i++;

}while( $i<5 );

练习：使用do...while...计算1\*3\*5\*7\*...15

|  |
| --- |
| while和do..while循环在很多时候是通用的。  细微差别：  while：先判定再执行，循环主体可能执行0~N  do..while：先执行再判定，循环主体可能执行1~N |

课后练习——通用小程序，五星题很难！！！必须一起讨论完成！！！

(1)\*\*\*\* 打印出所有的三位的水仙花数

153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3

提示：100~999

(2)\* 在一行中打印出10个※ 使用while和do..while...

(3)\*\*\* 打印出5行10列个※ —— 循环的嵌套

※※※※※※※※※※

※※※※※※※※※※

※※※※※※※※※※

※※※※※※※※※※

※※※※※※※※※※

(4)\*\*\*\*\* 打印出如下的※

※

※※

※※※

※※※※

※※※※※

(5)\*\*\*\*\* 打印出九九乘法表，本质和(4)相同

1\*1=1

2\*1=2 2\*2=4

3\*1=3 3\*2=6 3\*3=9

### Day04 for和foreach

复习：

掌握新语言的步骤：

(1)了解背景

(2)搭建环境

(3)变量常量

(4)数据类型

(5)运算符

算术： + - \* / % ++ --

比较： > >= < <= == != === !==

逻辑： && || !

位： << >>

赋值： = += -= \*= /= %=

拼接： . .=

三目： ? :

其它： , [] =>

(6)逻辑结构

顺序执行：

选择执行：if...else... switch...case...break

循环执行：while... do...while... for... foreach...

(7)通用小程序 —— 程序员必做的100道题

(8)函数和对象

(9)第三方组件

(10)小项目

1.了解：PHP中的“其它运算符”

逗号运算符： ,

示例： echo $uname, $age;

$age=10, $price=200;

取下标运算符： [ ]

箭头运算符： =>

2.Day01：PHP中的类型转换问题

隐式类型转换： 2 + '3A'

强制类型转换：

$i = 7.35; $j = (int)$i;

练习：由易到难：

(1)打印1个※

(2)打印1行10列※

(3)打印5行10列※

(4)打印5行10列※，呈现出三角形

(5)打印九九乘法表

今日目标：

(1)循环结构 —— for —— 重点

(2)数组和foreach循环 —— 重点

#### 3.循环执行 —— for

语法： —— 先判定再执行

**for( 表达式1; 表达式2; 表达式3){**

**循环主体;**

**}**

**表达式1：**执行且仅执行一次，在整个循环开始之前执行；一般习惯于执行变量的初始化语句。

**表达式2**：每次循环前都要执行的判定操作，值为true就执行循环主体一次；否则就退出循环；一般执行循环继续的判定语句。

**表达式3：**每次循环主体执行完后自动执行；一般执行循环变量的增减控制。

|  |  |
| --- | --- |
| $i = 0;  while( $i<5 ){  echo '※';  $i++;  } | for( $i=0; $i<5; $i++){  echo '※';  } |

练习：使用for循环输出0/1/2...99

练习：使用for循环输出99/98...0

练习：使用for循环输出0/2/4..98

练习：使用for循环输出95/90/85...5/0

练习：使用for循环计算：

1+3+5+...99的和

练习：使用for循环计算：

5/99 + 10/98 + 15/97 + 20/96 + 25/95 + 30/94

练习：使用for循环嵌套输出九九乘法表

练习：计算1\*2\*3\*4....得到一个乘积，第一次遇到乘积>=10000，立即退出；输出此时的乘积值

|  |
| --- |
| **break: 打断**  作用：用于终止当前正在执行的switch、for、while的执行主体 |
| **continue: 继续**  作用：用于跳过此次循环，继续变量改变，进行下一次循环 |

练习：计算1+2+3+4+6+7+8+9+10+11+12+13+14+16+...99

跳过所有以5结束的数字：5/15/25/35....

练习\*\*\*\*\*：输出2000年及以后的5个闰年年份

闰年：(1)能被4整除但不能被100整除 (2)能被400整除

$year记录年份，$count记录已经输出的年份的个数

练习\*\*\*\*\*\*：输出100以内的所有质数

质数：素数，只能被1和自身整除的数

外层循环$i：2/3/4......99

内层循环$j：N: 2/3/4....N-1

|  |
| --- |
| 标量类型：  复合类型： array、object  其它类型： |

4.PHP中的数组

创建一个变量，保存1个学生的年龄：

$age = 18;

创建一个变量，保存5个学生的年龄(一个变量中保存多个数据)：

$ageList = [ 20, 18, 23, 19, 22 ]; #数组

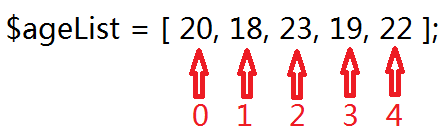
**数组：Array，是一种特殊的数据类型，一个数组变量中可以保存多个数据。**

声明一个数组变量：

$变量名 = [ 值1, 值2, 值3 ]; #新版本的PHP

$变量名 = array(值1, 值2, 值3); #老版本的PHP

数组中每个元素自动获得一个编号（元素的下标index）：0、 1、 2、 ....



输出某个元素的值： echo $数组名[下标];

修改某个元素的值： $数组名[下标] = 值;

输出数组中元素的总数： echo count($数组名);

向数组尾部添加新元素： $数组名[ ] = 值;

练习：创建一个数组，保存5个商品的价格，使用for循环输出每个价格； 再试着给下标为0、1、4的商品的价格+10，再次使用for循环输出每个价格；

练习：创建一个数组，保存4个商品的名称，使用for循环输出每个商品的名称

#### PHP数组分类:

|  |
| --- |
| PHP中数组的分类：  **(1)索引数组(Indexed Array)：**每个元素的下标是整数：0/1/2...  $arr1 = [10, 18, 21, 17];  $arr2 = ['联想', 7000, true, 9500, 5];  **(2)关联数组(Association Array)：**每个元素的下标是字符串  $arr3 = ['lname'=>'联想', 'lid'=>7000, 'isOnsale'=>true, 'price'=>9500, 'stockCount'=>5];  $arr4 = ['lname'=>'戴尔', 'lid'=>7000, 'isOnsale'=>true, 'price'=>9500, 'stockCount'=>5]; |

课后练习：

1.创建一个数组变量，保存学子商城中一个用户的注册信息，包括编号、登录名、密码、年龄、当前是否在线、头像图片路径；

2.创建一个数组变量，保存学子商城中又一个用户的注册信息，包括编号、登录名、密码、年龄、当前是否在线、头像图片路径；

3.创建一个数组变量，保存学子商城中第三个用户的注册信息，包括编号、登录名、密码、年龄、当前是否在线、头像图片路径；

4.创建一个数组变量，保存上述三个小数组。使用for循环输出这个大数组中的所有数据。

提示：只有索引数组可以使用for循环遍历；关联数组目前只能依次访问每个下标

### Day05 foreach和函数

复习：

(1)了解背景

(2)搭建环境

(3)常量和变量

(4)数据类型

标量类型：int float string bool

复合类型：array object

其它类型：null resource

(5)运算符

算术、比较、逻辑、位、赋值、拼接、三目、其它 , [] =>

(6)逻辑结构

顺序执行

选择执行 if..else.. switch..case..break..

循环执行 while.. do..while.. for.. foreach..

(7)通用小程序

(8)函数和对象

(9)第三方组件

(10)小项目

数组(Array)：是一个变量，其中可以保存多个数据

**索引数组(Indexed Array)：**适合于多个同类型的数据

$ageArr = [18, 15, 20];

echo $ageArr[0], $ageArr[1], $ageArr[2];

echo count($ageArr);

$ageArr[ ] = 19; //[3]

for($i=0; $i<count($ageArr); $i++){

echo $ageArr[$i];

}

**关联数组(Assocication Array)：**适合于多个不同类型数据

$row = [

'uname'=>'tom',

'upwd'=>'125',

'age'=>25,

'isOnsale'=>true

];

$row['email'] = 'tom@tedu.cn';

echo $row['age'];

不能使用for循环遍历关联数组！

练习：

创建一个数组，保存一个笔记本商品信息，包括编号、图片、标题、购买数量、是否特价；输出其中的每个元素值；

创建一个数组，保存一个笔记本商品信息，包括编号、图片、标题、购买数量、是否特价；输出其中的每个元素值；

创建一个数组，保存一个笔记本商品信息，包括编号、图片、标题、购买数量、是否特价；输出其中的每个元素值；

再创建一个大的数组，其中保存上述三个小数组；使用循环遍历出每个商品的信息。

练习：创建一个二维数组，保存3个员工的信息，每个员工都有编号、姓名、工资、生日、是否在岗等属性，输出所有数据

|  |
| --- |
| 注意：关联数组中每个元素的下标是个**“字符串”**！所以：  echo $arr['uname']; 此时下标必须加引号！  echo "用户名：$arr[uname]"; 此时下标不能加引号！ |

今日目标：

(1)补充：foreach循环 —— 必须掌握

(2)函数 —— 抽象&重点

(3)PHP中访问数据库的函数 —— 最重点

|  |
| --- |
| 循环： while( ){ } do{ }while( ) for(;;){ } foreach(){} |

#### 1.逻辑结构之循环结构 —— foreach

语法1：

foreach(数组名 as 变量名){

循环主体；

}

含义：对于数组中的每个元素，依次赋值给指定的变量名，执行一次循环体。

语法2：

foreach(数组名 as 下标变量名=>元素变量名){

循环主体；

}

含义：对于数组中的每个元素，依次把下标赋值给下标变量名，元素值赋值给元素变量名，执行一次循环体。

foreach循环专用于遍历数组中的所有元素；不用于普通的循环，如打印1/2/3...99。

**索引数组：**即可使用for遍历，也可使用foreach遍历(更简单不易出错)

$arr = [15, 18, 14, 22];

foreach( $arr as $v ){

echo $v;

}

foreach( $arr as $k=>$v){

echo $k . $v;

}

练习：创建一个索引数组，保存5个笔记本厂家名称，使用for循环打印出所有数据；再使用foreach循环打印数据。

#### 关联数组：无法使用for遍历，但可以使用foreach遍历！

练习：创建一个关联数组，保存一个用户的信息：编号、用户名、年龄、积分。使用foreach遍历每个数据。

练习：创建一个二维数组，保存购物车所有的商品数据（一个购物车可以保存多条商品信息），每个商品需要记录：图片、标题、单价、数量、是否勾选；使用foreach的嵌套输出所有数据

foreach( $cart as $laptop ){

foreach($laptop as $k=>$v){

}

}

午间练习：

1)创建一个索引数组，保存5个笔记本的价格，使用foreach计算这5个笔记本的总价格；

2)创建一个索引数组，保存5个笔记本的价格，使用foreach计算这5个笔记本的价格最大值；

3)创建一个索引数组，保存5个笔记本的价格，使用foreach计算这5个笔记本的价格最小值；

4)创建一个索引数组，保存5个笔记本的价格，使用foreach计算这5个笔记本的价格平均值；

2.了解：PHP中的预定义变量

预定义：由PHP解释器定义的变量，无需程序员声明可以直接使用。

$\_REQUEST：其中保存着客户端提交的请求数据 request：请求

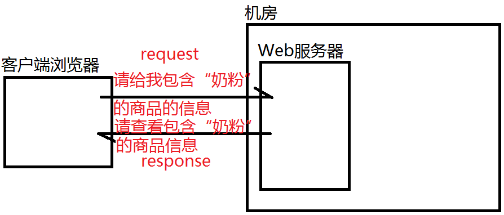
$\_GET

$\_POST

$\_FILES

$\_SESSION

上述PHP预定义变量全都是“关联数组”，初始时都是array(0)；



|  |
| --- |
| 浏览器如何给服务器传递请求数据：URL中的**查询字符串**  http://taobao.com/search.php?key1=value1&key2=value2&key3=value3... |
| 服务器如何接收客户端客户端发来的请求数据：  $value2 = $\_REQUEST['key2']; |

练习：客户端通过浏览器地址栏向服务器发送两个数据，num1和num2，PHP服务器接收到两个数据，在页面中输出它们的和

练习：客户端通过浏览器地址栏向服务器发送两个数据，uname和upwd，PHP服务器接收到两个数据，判断若用户名为dingding，密码为123456，就在页面中输出“登录成功”；否则就输出“用户名或密码错误”

#### 3.函数 —— 抽象&了解

Function：函数、功能体；封装着需要多次调用的复杂的代码。

声明函数：

function 函数名( ){

函数主体/要执行的代码

}

执行/调用函数：

函数名( );

练习：创建一个函数，printStar()，其中在一行中输出10个※，执行此函数3次。

声明带参数的函数 —— 指定**“形式参数/形参”**

function 函数名( 参数名1, 参数名2, 参数名3...){

函数主体

}

调用带参数的函数 —— 提供**“实际参数/实参”**

函数名( 值1, 值2, 值3... );

练习：创建一个函数，printRect()，用※打印出方块，需要接收两个参数：$rows和$cols，据这两个参数打印出$rows行$cols列的方块。调用此函数3次。

练习：创建一个函数，add()，可以接收三个数字做参数，函数体中计算出这三个数字的和，并打印出来。调用此函数2次。

练习：创建一个函数，maxNum()，可以接收5个数字做参数，函数体中获取这5个数字中的最大值，并输出。调用此函数2次。

声明有返回值的函数：

function 函数名( 形参列表 ){

函数主体;

return 值;

}

调用有返回值的函数：

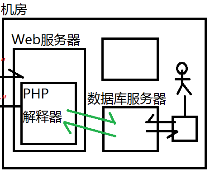
$变量名 = 函数名( 实参列表 );

练习：创建一个函数，cartCalc()，接收一个商品单价、购买数量、折扣金额做参数，返回该商品应支付的总金额。 调用此函数2次，分别输出其返回值

|  |
| --- |
| 函数的知识点：  (1)如何声明函数  (2)什么是形参和实参  (3)什么是返回值 |
| function add($n1, $n2){ #声明一个函数  $sum = $n1 + $n2;  return $sum;  }  $s = add(10, 20); #执行一个函数，获取其返回值 |

|  |
| --- |
| PHP提供了多套访问MySQL的函数：  mysql\_\* 如：mysql\_connect()  mysqli\_\* 如：mysqli\_connect() Improved 提高/改进 |

#### 4.使用PHP预定义的函数 —— MySQL操作函数



提示：使用PHP连接MySQL的过程与命令行客户端访问MySQL类似。

**(1)连接到MySQL服务器**  mysql -uroot

$conn = mysqli\_connect( ........ );

**(2)向服务器提交要执行的SQL语句** select \* from emp;

$sql = "DDL/DML/DQL";

$result = mysqli\_query( $conn, $sql );

**(3)查看执行结果** ......

if($result===false){ } else { }

**(4)断开到数据库的连接（可以省略）** quit

mysqli\_close( $conn );

课后练习：

(1)创建一个函数：arrMax()，接收一个数组做参数，返回该数组中所有数字的最大值；调用该函数。

(2)创建一个函数：arrMin()，接收一个数组做参数，返回该数组中所有数字的最小值；调用该函数。

(3)创建一个函数：arrAvg()，接收一个数组做参数，返回该数组中所有数字的平均值；调用该函数。

(4)使用PHP访问MySQL中的tedu数据库，向emp表中插入一行新的员工记录。

(5)使用PHP访问MySQL中的tedu数据库，从emp表中删除编号为1的员工记录；执行两次，看看结果是成功还是失败

### Day06 MySql 增删改查

复习：

见思维导图

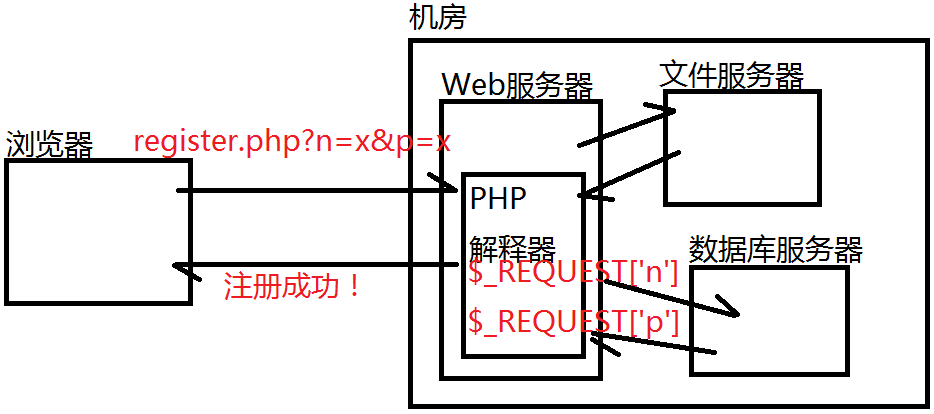
今日目标 CRUD操作

(1)使用PHP向数据库发送INSERT - Create

(2)使用PHP向数据库发送DELETE - Delete

(3)使用PHP向数据库发送UPDATE - Update

(4)使用PHP向数据库发送SELECT - Retrieve



1.使用PHP向MySQL发送INSERT —— 注册新用户

**客户端给服务器提交数据：**

http://127.0.0.1/xuezi/data/register.php?uname=tom&upwd=123&email=tom@tom.com&phone=135011111111

**服务器创建data/register.php：**

(1)接收客户端提交的上述四个数据

@$n = $\_REQUEST['uname'];

if($n===null || $n===""){ die(""); }

(2)连接到MySQL服务器

$conn = mysqli\_connect( ... );

(3)提交SQL语句给MySQL服务器

$result = mysqli\_query($conn, "......");

(4)输出执行结果

if($result===false){ }else { }

|  |
| --- |
| PHP小技巧：如何压制/屏蔽当前行代码中的警告/错误消息： @ |
| PHP小技巧：若页面中有严重的错误，可使用die()终止执行 |
| PHP小技巧：获取刚刚执行的INSERT语句产生的自增编号：  $id = mysqli\_insert\_id( $conn ); |

2.使用PHP向MySQL发送DELETE —— 用户删除

**客户端给服务器提交数据：**

http://127.0.0.1/xuezi/data/user\_delete.php?uid=12

**服务器创建data/user\_delete.php：**

(1)接收客户端提交的上述四个数据

@$u = $\_REQUEST['uid'];

if( $u===null || $u===""){ die(""); }

(2)连接到MySQL服务器

$conn = mysqli\_connect(....);

(3)提交SQL语句给MySQL服务器

$sql = "DELETE FROM xz\_user WHERE uid=$u";

$result = mysqli\_query($conn, $sql);

(4)输出执行结果

if($result===false){} else { }

|  |
| --- |
| PHP小知识：获取最新执行的一条DML语句影响的函数：  $c = mysqli\_affected\_rows( $conn ); |
| PHP小知识：在当前文件的当前位置引入/包含另一个文件的内容： require('被包含的文件.php'); |

|  |
| --- |
| DDL: CREATE DROP ALTER TRUNCATE  DML: INSERT DELETE UPDATE  DQL: SELECT  DCL: |

3.使用PHP向MySQL发送UPDATE —— 更改用户基本信息

**客户端给服务器提交数据：**

http://127.0.0.1/xuezi/data/uc\_basic.php?uid=8&user\_name=强东&gender=1&phone=18501234567&email=qd@tedu.cn

**服务器创建data/uc\_basic.php：**

(1)接收客户端提交的上述五个数据

(2)连接到MySQL服务器

(3)提交SQL语句给MySQL服务器

(4)输出执行结果

注意：SQL语句的写法

UPDATE xz\_user

SET user\_name='强东',gender=1,email='qd@tedu.cn',phone='135013423'

WHERE uid=8;

4.使用PHP向MySQL发送SELECT —— 用户登录(只查询一行记录)

**客户端给服务器提交数据：**

http://127.0.0.1/xuezi/data/login.php?uname=xx&upwd=yy

**服务器创建data/login.php：**

(1)接收客户端提交的上述二个数据

(2)连接到MySQL服务器

(3)提交SQL语句给MySQL服务器

$sql = "SELECT \* FROM xz\_user WHERE uname='' AND upwd=''"; //此语句查询到的结果集中最多只有一行记录

(4)输出执行结果

|  |
| --- |
| $result = mysqli\_query($conn, $sql);  函数的返回值：  DML: 失败-false 成功-true  DQL: 失败-false 成功-结果集的描述对象 |

|  |
| --- |
| $result = mysqli\_query($conn, 'SELECT ...');  处理方法：  (1) $row = mysqli\_fetch\_row($result);  返回值是一个NULL或者索引数组（表中的一行记录）  (2) $row = mysqli\_fetch\_assoc($result);  返回值是一个NULL或者关联数组（表中的一行记录）  (3) $rowList = mysqli\_fetch\_all($result)  返回值是一个二维数组，包含查询到的所有记录行 |

5.使用PHP向MySQL发送SELECT —— 用户列表(查询多行记录)

**客户端给服务器提交数据：**

http://127.0.0.1/xuezi/data/user\_list.php

**服务器创建data/user\_list.php：**

(1)连接到MySQL服务器

(2)提交SQL语句给MySQL服务器

$sql = "SELECT \* FROM xz\_user"; //此语句查询到的结果集中包含该多行记录

(3)输出执行结果

课后练习：

(1)整理今天所学的有关数据库操作的所有函数

(2)删除课堂中演示的“增删改查”相关页面，仿写出“用户注册”、“用户删除”、“修改用户注册信息”、“用户登录”、“用户列表”相关功能必需的服务器端页面

# 赵旭

## HTML Basic

### Day01 html 基础标签

HTML + JavaScript + Ajax + CSS

zhaoxu@tedu.cn

HTML5

1、WEB基础知识(了解)

1、Internet - 网

由若干台电脑、手机、平板 通过 网线(WIFI) 连接起来的结构

2、基于Internet上的程序

1、C / S 结构

Client / Server

特点：必须通过指定的客户端软件才能访问服务器端的一种程序 ：如 ：桌面版QQ，... ...

2、B / S 结构

Browser / Server

Browser ：浏览器

特点：通过 浏览器 就能访问服务器端的一种程序

如 ：网页版百度，网页版 QQ，网页版 京东，... ...

3、WEB

1、什么是WEB

WEB，是基于Internet上的一种应用程序(网页应用程序)

WEB页面，简称WEB页(网页)，就是保存在服务器端上的一个具体的页面( \*\*.html / \*\*.htm 作为结尾的文件)

2、WEB浏览器

1、功能

1、提交用户请求 (UA : User Agent)

2、作为HTML 以及 脚本执行的 解释器

3、以图形化的方式显示web页面

2、主要浏览器产品

1、Microsoft Internet Explorer (IE)

2、Mozilla Firefox

3、Google Chrome

4、Apple Safari

5、Opera Opera(欧朋)

3、主要技术

1、HTML

2、CSS

3、Javascript

2、HTML入门(重点)

1、什么是HTML

Hyper Text Markup Language

超级 文本 标记 语言

超文本 ：也是文本，但会具备特殊功能

普通文本 a : 普通字符 a

超文本 a : 表示超链接

普通文本 b : 普通字符 b

超文本 b : 加粗显示文本

标记 ：超文本的表现形式

普通文本 a : a

超文本 a : <a></a>

语言 ：具备一定的语法规范

HTML 也具备自己的语法规范

WEB页面(HTML页面)就是由 HTML 语言来进行开发的 ，以 .html 或 .htm 进行结尾的文件

开发 & 运行网页的工具：

1、记事本

2、浏览器

2、HTML 的基础语法

1、标记语法

HTML中用于描述功能的符号称为"标记"

标记在使用时，用尖括号 "<>",标记的分类

1、封闭类型的标记

也称为 "双标记" , 必须成对出现

语法：<标记>内容</标记>

Demo ：

1、创建 p 标记 --<p>...</p>

2、创建 div 标记 -- <div></div>

3、创建 header 标记 -- <header></header>

2、非封闭类型的标记

也称为 "单标记"

语法：<标记> 或 <标记/>

Demo ：

1、创建 br 标记 -- <br/>

2、创建 hr 标记 -- <hr/>

3、创建 img 标记 -- <img/>

4、创建 input 标记 -- <input/>

2、标记(元素)的嵌套

在一对标记中，允许出现另外一对(一个)标记

<标记1>

<标记2></标记2>

</标记1>

注意：嵌套标记的书写格式 -- 被嵌套的标记要通过一个 缩进键(Tab) 来表示层级关系

Demo :

1、编写一对 body 标记，在body标记中，嵌套一对 div标记，在 div 标记中 ，嵌套一对 a 标记，在 a标记中，嵌套一对 b 标记，b标记中，随意编写一些文本

<body>

<div>

<a>

<b>这是一段测试文本</b>

</a>

</div>

</body>

3、标记的属性

1、什么是属性

用来修饰标记的效果的内容，就是属性

2、语法

1、属性必须声明在开始标记中

2、属性与标记名称之间，用 空格 隔开

3、属性的值 与 属性之间 使用 "=" 连接

4、一个元素允许有多属性，多属性间，排名不分先后，中间用 空格 隔开即可

Demo :

1、创建一个 div 标记 ，增加属性 ，设置 align 属性的值为 center ，设置 id 属性的值为 container，设置 title 属性的值为 这是一个div

<div align="center" id="container" title="这是一个div"></div>

四大标准属性：

1、id ：定义元素在页面中独一无二的名称

2、title ：鼠标悬停在元素上时，体现的文字

3、class ：引用 类选择器时使用(CSS中)

4、style ：定义 内联方式方式使用(CSS中)

4、注释

语法：<!-- -->

注意：

1、注释不能嵌套

2、注释不能出现在标记中

3、HTML 文档结构

W3C ：Word Wide Web Consortium (万维网联盟)

1、HTML文档的组成

1、文档类型声明

告知浏览器当前的HTML文档用的是哪个版本

在网页的最顶端 <!doctype html>

2、由一对 html 根标记，来表示页面的开始与结束

Demo :

1、在 htdoc 中，创建一个网页 01-first.html 文件

创建一个 记事本，将 .txt 重命名为 .html

2、搭建HTML网页结构

1、添加 文档类型声明

2、添加 html 根标记

2、<html> 标记

包含两对子元素

1、<head></head>

包含的子元素(2对)

1、<title>网页的标题</title>

2、<meta charset="utf-8"/> -- 能正常显示中文

2、<body></body>

属性：

1、text ，取值是一个颜色值(red,green,blue...)

2、bgcolor ，取值也是一个颜色值

Demo :

在刚才的 Demo 基础上

1、在 <html>中 增加 <head> 和 <body>

2、为网页指定标题 - 我的第一个HTML文档

3、指定网页的字符编码格式为 utf-8

4、在 body 中 输出一句话 "我的第一HTML页面"

5、设置 body text为red，bgcolor为yellow

3、文本

1、特殊文本的实现

页面的空格以及一些特殊字符需要通过转义字符的方式体现

1、&nbsp; 表示一个空格

2、&lt; 表示 <

3、&gt; 表示 >

4、&copy; 表示©

Demo :

1、创建一个页面 02-text.html

2、在页面中 输出以下内容

The <p> element. ©2017 By Tarena

The &lt;p&gt; element.&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&copy; By Tarena

2、文本样式相关标记

<b></b> : 加粗

<i></i> : 斜体

<u></u> : 下划线

<s></s> : 删除线

<sup></sup> : 上标

<sub></sub> : 下标

Demo ：

这是一段包含 ，粗体，斜体，下划线，删除线，上标，下标的文本内容

将以上文本内，对应文字的特殊效果，用标记体现出来

3、标题元素

以 醒目 的方式表现出文本

语法：

<h1>一级标题</h1>

<h2>二级标题</h2>

<h3>三级标题</h3>

<h4>四级标题</h4>

<h5>五级标题</h5>

<h6>六级标题</h6>

属性：

1、align

文本的水平对齐方式

取值：left / center / right

4、段落元素

语法：<p></p>

属性：align : left / center / right

Demo :

1、在 02-text.html 中，增加以下内容

1、用 段落标记表示 :The first paragraph

2、用 段落标记表示 :The second paragraph ，文本表现为 右对齐

5、换行元素

<br>

6、分割线元素

作用：在页面中表现为一条直线

语法：<hr>

属性：

1、size

尺寸，以 px 为单位的数值

2、width

宽度，以 px 或 % 为单位的数值

3、align

水平对其方式

4、color

颜色

Demo ：

02-text.html 中

增加一根水平线，size为5px，宽度为50%，居中对齐，颜色为红色(red)

7、预格式化

作用：保留源文档中的回车 和 空格 的作用

<pre></pre>

8、分区元素

1、块分区元素

语法：<div></div>

作用：布局

2、行内分区元素

语法：<span></span>

作用：设置同一行文字内的不同样式

9、行内元素 与 块级元素

按照元素们的表现形式来分类，分为 行内元素 和 块级元素1、行内元素

在一行内允许显示多个元素的，称为 "行内元素"

span,i,b,s,u,sup,sub

作用：包裹文本，并处理文本的表现形式

2、块级元素

每个元素独占一行显示的，称为 "块级元素"

div,p,h1,h2,h3,h4,h5,h6

作用：布局

注意：

1、p标记不能嵌套其它的块级元素

2、行内元素中 最好 不要嵌套块级元素

4、图像 和 链接

1、URL

目录结构 ：

目录 ，保存文件的文件夹名称

多个文件夹之间的关系，就是目录结构

1、什么是 URL

URL (Uniform Resource Locator)，统一资源定位器。用来标识某资源文件的位置

2、URL 在 WEB 中的表现形式

共三种表现形式：

1、绝对路径

特点：从文件所在的最高级目录处开始查找资源文件所经过的路径，就是绝对路径

使用场合：当想访问互联网上的资源时，只能用绝对路径

完整的绝对路径分四部分：

1、协议名

http

2、域名(主机名，IP地址)

www.codeboy.com

3、目录路径

img/header

4、文件名

logo.png

http://www.codeboy.com/img/header/logo.png

img/header/logo.png

2、相对路径(重点)

场合：使用本机文件时使用

什么是相对路径：

从当前文件位置处开始查找资源文件所经过的路径，就是相对路径

1、同目录，直接用

2、子目录，先进入

3、父目录，先返回

3、根相对路径

路径形式是以 / 作为开始的。

/ : 表示的是服务器的根路径

2、图像

1、图像格式

WEB中支持的图像格式

1、\*.jpg

2、\*.gif(动图)

3、\*.png(透明)

2、图像的语法

标记：<img>

属性：

1、src ：指定要显示图像的 URL

2、width ：图像的宽度

3、height ：图像的高度

4、title ：鼠标悬停时，要显示的文本

Demo :

1、将 学子网的 logo 下载下来

2、显示在自己的网页中

3、超链接

1、语法

标记：<a>内容</a>

属性：

1、href : 要链接的HTML页面URL

2、target : 目标，指定新网页的打开形式

取值：

1、\_blank : 在新标签页中打开网页

2、\_self : 在自身标签页中打开新网页(默认值)

Demo

1、创建一个超链接，内容为 ：学子商城，点击时，在自身标签页中 打开 http://www.codeboy.com

2、创建一个超链接，内容为 ：学子商城的LOGO，点击时，在新标签页中打开 http://www.codeboy.com

2、链接的四种表现形式

1、点击操作时，完成资源下载的操作

链接的资源为 zip / rar 时则为下载操作

<a href="day01.rar">下载</a>

2、电子邮件链接

<a href="mailto:zhaoxu@tedu.cn">联系我们</a>

前提 ：

必须在计算机中安装并配置好至少一个邮件客户端的信息

3、返回页面顶部的空连接

<a href="#">返回顶部</a>

4、执行Javascript代码片段

<a href="javascript:JS脚本代码">执行JS</a>

3、锚点

1、作用

锚点用于在网页中的某个位置处做个记号,允许从其他位置处跳转到记号位置处

2、使用方式

1、定义锚点

1、使用 a 标记的 name 属性允许定义锚点

<a name="maodian1">xxxx</a>

2、任何标记的 id 属性定义锚点

2、链接到锚点

<a href="#锚点名称">链接到锚点</a>

<a href="页面URL#锚点名称">链接到锚点</a>

### Day02表格,列表,表单

1、表格

1、表格的作用

按照一定的格式来显示数据的

表格是由 单元格(列)，行 来组成的

2、表格的语法

1、标记

1、<table></table> : 表示一个表格

2、<tr></tr> : 表示表格中的一行

3、<td></td> : 表行中的一列(单元格)

<th></th> : 行/列 标题 (加粗，水平居中)

4、<caption></caption> : 表格的标题

该元素必须添加在 <table>之下，所有的tr之上

2、常用属性

1、table

1、width

2、height

3、align

设置表格的对齐方式 ， left / center / right

4、border

边框宽度，取值以 px 为单位的数值

5、bgcolor

6、cellspacing ：单元格外边距(单元格与单元格 或 单元格与表格之间的距离)

7、cellpadding : 单元格内边距(单元格与内容之间的距离)

2、tr

1、align

控制当前行内容的水平对齐方式

2、valign

控制当前行内容的垂直对齐方式

top / middle / bottom

3、bgcolor

3、td / th

1、align

2、valign

3、width

4、height

5、colspan ：跨列

6、rowspan ：跨行

Demo :

在 刚才的表格基础上，增加以下内容

1、为表格增加边框 1px

2、为表格增加尺寸 400 \* 400

3、为表格增加单元格内边距 5px

4、为第四行 增加属性 ，内容水平居中对齐

3、表格的复杂应用

1、行分组

允许将若干行划分到一个组中，以便实现统一管理

1、表头行分组

允许将第一行的内容单独分到表头行分组中

<thead></thead>

2、表尾行分组

允许将最后一行的内容单独分到表尾行分组中

<tfoot></tfoot>

3、表主体行分组

<tbody></tbody>

2、不规则表格创建

通过 td 的跨行 和 跨列来实现不规则的表格

1、跨列

<td colspan="">

从指定单元格位置处开始，横向向右，合并掉几个单元格(包含自己),被合并掉的单元格，要删除

2、跨行

<td rowspan="">

从指定单元格位置处开始，纵向向下，合并掉几个单元格(包含自己),被合并掉的单元格，要删除

3、表格的嵌套

被嵌套的所有的内容，只能放在 td 中

2、列表

1、作用

按照一定的格式显示数据

2、列表的分类 & 语法

1、列表的组成

由列表类型 以及 列表项 来组成

2、分类 & 语法

1、有序列表

<ol></ol> --> Order List

允许包含若干列表项：

<li></li> --> List Item

2、无序列表

<ul></ul> --> Unorder List

允许包含若干列表项：

<li></li>

3、属性

1、ol

1、type

1 、按数字方式排列,默认值

a 、按小写英文方式排列

A 、按大写英文方式排列

i 、按小写罗马数字排列

I 、按大写罗马数字排列

2、start

指定标识从 几 开始显示

2、ul

1、type

1、disc，实心圆,默认值

2、circle，空心圆

3、square，实心矩形

4、none

3、列表的嵌套

被嵌套的内容只能放在 li 中

Demo :

1、声明一个列表在 html 中(有序)，包含两个列表项，显示 两部小说的名字

2、在两个列表项中，再各嵌套一个无序列表，各写3-4个小说中的主角

3、定义列表

1、作用

以一种特殊的结构来排列数据

通常用语对一类事物/名词 的解释上面

2、语法

<dl></dl>：声明一个定义列表

<dt></dt>：声明要解释的事物名称 或 名词

<dd></dd>：对上述名词或事物解释的内容

3、常用场合

图文混排

3、结构标记

1、结构标记的作用

搭建网页的结构(布局),用于替代 div，最大的优点是提升了标记的"语义性"

2、结构标记详解

1、<header></header>

作用：表示网页 或 某块内容的头部

2、<nav></nav>

作用：表示网页中的 导航内容

3、<section></section>

作用：表示网页主体内容中的某一部分

4、<article></article>

作用：出现在文字描述性比较强的场合:一则新闻，论坛中的帖子，微博信息，博客的信息

5、<aside></aside>

作用：表示网页中，或某部分内容的 边栏信息

6、<footer></footer>

作用：表示网页中尾部的信息

4、表单(难点)

1、表单的作用

用于收集用户的信息并提交给服务器

表单主体是由两部分组成的

1、表单元素

收集信息，并提交给服务器

2、表单控件

用于与用户交互的一些元素：文本框，密码框

2、表单元素(难点)

标记：<form></form>

注意：只有放在 <form></form> 里面的表单控件的值，才能提交给服务器

属性：

1、action

提交给服务器处理程序的地址 (动作)

2、method

提交方式，以什么样的方式把数据交给服务器

1、get

特点：

1、以明文的方式将数据传递给服务器(提交的数据会显示在地址栏上)

2、安全性很低

3、提交数据最大限制为 2KB

4、向服务器要数据时可以使用

2、post

特点：

1、以隐式的方式将数据传递给服务器

2、安全性很高

3、不限制提交数据的大小

4、要提交数据给服务器处理时使用

3、... ...

3、enctype

作用：指定表单数据进行编码的方式

取值：

1、application/x-www-form-urlencoded

默认值，允许将任意类型的文本提交给服务器

2、multipart/form-data

允许将文件提交给服务器

3、text/plain

不对任何数据进行编码和传输

Demo :

1、创建 05-form.html

2、声明一个表单元素 form

3、指定提交地址为 login.php,提交方式为 get，编码方式为默认

3、表单控件

表单元素分类：

1、input元素

2、textarea 多行文本域

3、select 和 option 选项框

4、其它元素

5、新input元素

详解 ：

1、input元素

里面会包含若干个不同的表单控件

标记 ：<input>

属性 ：

1、type

根据不同的type属性值，可以创建不同的input元素

2、name

定义表单控件的名称，主要提交给服务器使用的

注意：如果不声明name属性的话，元素则无法提交给服务器

3、value

定义当前控件的值，主要提交给服务器使用的

4、disabled

禁用控件，无值的属性

<input disabled>

2、文本框 与 密码框

文本框：<input type="text">

密码框：<input type="password">

具备除以上四个属性外，还具备以下几个独立属性：

1、maxlength

限制输入的字符数，取值为数字

2、readonly

只读，无值属性

Q : readonly 和 disabled 的区别

1、readonly ，是只读的意思，是允许提交给服务器，不让用户改数据而已

2、disabled ， 是禁用的意思，不仅不让用户改数据，而且还不能提交给服务器

3、placeholder

占位符，即默认显示在文本框用于给用户提示的文字

3、按钮

1、提交按钮

作用：将表单提交给服务器

标记：<input type="submit" value="显示的文本">

2、重置按钮

作用：将表单恢复到初始化的状态

标记：<input type="reset" value="显示的文本">

3、普通按钮

作用：通过 JS 自定义功能

标记：<input type="button" value="显示的文本">

4、<button>显示的内容</button>

属性：

1、type ：submit / reset / button

4、单选按钮 和 复选框

单选按钮：<input type="radio">

复选框：<input type="checkbox">

属性：

1、name

设置名称，并用于分组，一组单选按钮 或 复选框的名称必须相同

2、value

必须设置

3、checked

默认被选中，无值属性

5、隐藏域 和 文件选择框

1、隐藏域

<input type="hidden" name="" value="">

想提交给服务器，但不想给用户看的数据，可以放在隐藏域中

2、文件选择框

<input type="file" name="">

注意：

1、form的 method 属性值必须为 post

2、form的 enctype属性值必须为 multipart/form-data

http://localhost/02-HTML/Day02/login.php?uname=wenhua.li&upwd=bingbing

http://localhost/02-HTML/Day02/login.php?uname=wenhua.li&upwd=fengjie&gender=1&question=0&question=1&question=2&question=3

<http://localhost/02-HTML/Day02/login.php?uname=wenhua.li&upwd=fdsafdsafdsa&gender=0&question=0&question=1&question=2&question=3&uid=10000567789>

## JavaScript Basic

### Day01 表单,JavaScript

1、表单

1、form ...

2、input ...

3、textarea 控件

标记：<textarea></textarea>

属性：

1、name

定义名称，提交给服务器使用

2、cols

指定文本域的列数，变相指定宽

3、rows

指定文本域的行数，变相指定高

4、readonly

只读

Demo :

创建 01-form.html 文件

创建一个 多行文本域，列数为 50，行数为 5，名称为 intro

4、选项框

1、标记

<select></select>

表示 下拉列表 或 滚动列表

<option></option>

列表中的内容，允许出现多个

2、属性

1、<select>

1、name ：控件的名称

2、size ：默认显示选项的数量，默认为1，为下拉列表，如果取值>1的话，则为滚动列表

3、multiple

设置多选，无值的属性

2、<option>

1、value ：选项的值，提交给服务器用

2、selected ：默认被选中

5、其它标记

1、label 元素

作用：关联文本与表单控件的，点击文本时就如同点击表单控件一样

标记：<label>文本</label>

属性：

1、for

指定要关联的表单控件的ID值

2、为控件分组

<fieldset>

<legend>分组标题</legend>

分组中的内容

</fieldset>

3、浮动框架

1、作用

将其他页面导入到当前页面中来

2、语法

标记：<iframe></iframe>

属性：

1、src

要引入的页面的url

2、frameborder

指定浮动框架的边框，默认为1，则显示边框

不需要边框则设置为 0

3、width

4、height

练习：

创建一个 02-iframe 的网页，将 01-form.html 页面引入进来 ，适当调整宽度，高度，边框

6、新input元素 (HTML5)

1、电子邮件类型

作用：提交时会验证数据是否符合Email的规范

标记：<input type="email">

2、搜索类型

作用：在文本框的基础上，提供了快速清除操作

标记：<input type="search">

3、URL类型

作用：提交时会验证数据是否符合Web站点的URL规范(绝对路径)

标记：<input type="url">

4、电话号码类型

作用：在移动端设备中，能展开 拨号键盘，在PC中无效

标记：<input type="tel">

5、数字类型

作用：只能让用户输入 或 选择数字

标记：<input type="number">

属性：

1、min ：当前控件接受的最小值

2、max ：当前控件接受的最大值

3、step ：微调数字时每次变化的长度,默认为1

6、范围类型

作用：提供一个滑块，让用户选择数字

标记：<input type="range">

属性：

1、min ：当前控件的最小值

2、max ：当前控件的最大值

3、value ：设定初始值

7、颜色类型

作用：提供一个颜色拾取控件

语法：<input type="color">

8、日期类型

作用：提供一个日期输入控件

标记：<input type="date">

9、周类型

作用：提供一个日期控件，用于选取周

标记：<input type="week">

10、月份类型

作用：选取月份控件

标记：<input type="month">

=============================

Javascript Basic

1、Javascript 概述(了解)

Javascript，简称为 JS，是一款能够运行在 JS解释器/引擎 中的脚本语言

JS解释器/引擎 是JS的运行环境：

1、独立安装的JS解释器 - NodeJS

2、嵌入在浏览器中的JS解释器

JS的发展史：

1、1992年 Nombas 开发了一款语言 ScriptEase

2、1995年 Netscape(网景) 开发了一款语言 LiveScript,更名为 Javascript

3、1996年 Microsoft(微软) 开发了一款语言 JScript

4、1997年 网景 将Javascript 1.1 提供给了ECMA(欧洲计算机制造商联合会),ECMA 获取了 JS 的核心，称之为 ECMA Script (ES)

完整的JS组成：

1、核心(ES)

2、文档对象模型(Document Object Model) - DOM

允许让 JS 与 HTML 文档打交道

3、浏览器对象模型(Browser Object Model) - BOM

允许让 JS 与 浏览器进行交互

JS是一款基于对象的编程语言

2、JS的基础语法

1、浏览器内核

内核负责页面内容的渲染，由以下两部分组成：

1、内容排版引擎 - 解析HTML/CSS

2、脚本解释引擎 - 解析Javascript

2、搭建JS运行环境(重点)

1、独立安装的JS解释器 - NodeJS

console.log("Hello World");

console.log('Hello World');

2、使用浏览器中的内核(JS解释引擎)

1、在浏览器的控制台(Console)中，输入脚本并执行

2、将JS脚本代码嵌入在HTML页面中执行

1、采用HTML元素事件执行JS代码

事件 ：

1、onclick

当元素被点击时执行的操作

ex: 当按钮被点击时，在控制台中输出 Hello World

<button type="button" onclick="console.log('Hello World');"></button>

2、将JS脚本编写在 <script></script> 并嵌入在HTML文档的任何位置

<script>

console.log("... ...");

document.write("<h1>Hello Wolrd</h1>");

</script>

3、将JS脚本编写在外部独立的JS脚本文件中(\*\*\*.js)

步骤：

1、编写JS脚本文件

2、在HTML中引入脚本文件

<script src=""></script>

练习：

1、先创建一个 base.js 的文件

2、在文件中执行以下代码

console.log(" .... ... ");

document.write(" ... ... ");

window.alert("这是在外部脚本文件中的内容");

3、在 html 文档中，引入 base.js 文件

3、JS调试

当代码编写出现错误时，在运行的时候，在错误位置会停止

碰到错误代码，会终止当前语句块的执行，但不影响后续块的执行

<script></script> 为一块

4、JS语法

1、语句 - 可以被JS引擎执行的最小单元

由表达式、关键字、运算符 来组成的

严格区分大小写 ：name 和 Name

所有的语句都是以 ; 来表示结束

所有的标点符号都是英文的

; 和 ；

. 和 。

: 和 ：

" 和 “

' 和 ‘

() 和 （）

[] 和 【】

{} 和 {}

2、注释

单行注释: //

多行注释: /\* \*/

3、变量 与 常量

1、变量

1、什么是变量

用来存储数据的一个容器

2、声明变量(重点)

1、声明变量

var 变量名;

2、为变量赋值

变量名=值;

3、声明变量并赋初始值

var 变量名=值;

注意：

1、变量在声明时没有赋值的话，那么值为 undefined

2、声明变量允许不使用var关键字，但并不推荐

练习：

1、创建一个网页 04-variable.html

2、声明一对 <script></script>,并完成以下变量的声明

1、声明一个变量用于保存用户的姓名，并赋值为 "张三丰";

2、声明一个变量用于保存用户的年龄，赋值 68

3、如何 将变量的数据 打印在控制台上？？

4、一条语句中声明多个变量

var 变量名1=值,变量名2=值,变量名3;

3、变量名命名规范

1、由字母，数字，下划线以及 $ 组成

var user\_name; 正确

var user-name; 错误

var $uname; 正确

2、不能以数字开头

var 1name;错误

3、不能使用JS中的关键字 和 保留关键字

4、变量名不能重复

5、可以采用"驼峰命名法",小驼峰命名法使用居多

6、最好见名知意

var a;

var uname;

4、变量的使用

1、为变量赋值 - SET操作

只要变量出现在 赋值符号(=)的左边一律是赋值操作

var uname="张三丰";

uname="张无忌";

2、获取变量的值 - GET操作

只要变量没有出现在赋值符号(=)的左边，一律是取值操作

var uname="wenhua.li";//赋值操作

console.log(uname);

var new\_name = uname;

new\_name 是赋值操作

uname 是取值操作

uname = uname + "bingbing.fan";

赋值符号出现的话，永远都是将右边的值，赋值给左边的变量(从右向左运算)

2、常量

1、什么是常量

在程序中，一旦声明好，就不允许被修改的数据

2、声明常量

const 常量名=值;

常量名在命名时采用全大写形式

作业：

1、声明一个变量 r ，来表示一个圆的半径，并赋值

2、声明一个常量PI ，来表示圆周率3.14

3、通过 r 和 PI 来计算 该圆的周长，保存在变量l中

周长 = 2 \* π \* 半径

4、通过 r 和 PI 来计算 该圆的面积，保存在变量s中

面积 = π \* r \* r;

5、在控制台中打印输出

半径为 \* 的圆的周长是 \*

半径为 \* 的圆的面积是 \*

笔记本名称:ThinkPad E460

笔记本价格:3000

笔记本库存:100台

### Day02js数据类型,运算符

1、数据类型

1、什么是数据类型

保存在变量中的数据的类型是什么

它决定了数据在内存中要占多大的空间

bit : 位

8bit = 1byte 字节

1024byte = 1kb 千字节

1024kb = 1MB 兆字节

1024MB = 1GB

1024GB = 1TB

2、数据类型分类

1、原始类型(基本类型/值类型)

1、number 类型

数字类型，表示32位(4字节)的整数以及64位(8字节)的浮点数

整数：

1、十进制 ：10,457,6989

var num = 457;

2、八进制

由 0-7 共8个数字组成，逢8进1

在JS中，数字以0开始的，都是8进制数字

var num = 010;//8进制数的10，表示的是十进制的8

3、十六进制

由 0-9，A-F组成，逢16进1

在JS中，数字以0x开始的，都是十六进制数字

var num = 0x10;//十六进制的10，表示的是十进制的16

小数：

使用小数点记录数据：123.456，78.5

使用指数记录数据：1.2e2 （1.2 \* 10的2次方）

2、string 类型

字符串类型

需要使用 "" 或 '' 括起来

由Unicode字符，数字，标点组成

每个字符都有一个Unicode码

将 字符 转换为 Unicode码:"张".charCodeAt().toString(16)

将 Unicode码 转换为 字符: var s="\u5f20"

中文的范围是从 "\u4e00" ~ "\u9fa5"

练习：

1、将各自的姓名中的每个字符，都转换为Unicode码

2、再将姓名的Unicode码转换为对应的字符

转义字符：

控制台中打印输出 "Hello World"

作用：在JS中，表示一些特殊的符号或功能

\n : 换行

\t : 一个制表符(缩进键)

\" : "

\' : '

\\ : \

var str = "Hello \n Word";

3、boolean 类型

仅有两个值 ：true 和 false

boolean 与 number 在进行运算时，true 当做1，false当做0

var num = 1;

var res = true;

console.log(num+res); // 2

4、null 空

让对象不再指向任何空间

5、undefined 未定义

出现场合：

1、声明变量但从未赋值时

2、试图访问对象中不存在的属性

2、引用类型

object array function

3、查看变量的数据类型

var result = typeof(变量);

练习：

1、声明一个变量，保存数据 "张三丰"，打印数据类型

2、声明一个变量，保存数据 68，打印数据类型

3、声明一个变量，保存数据 true，打印数据类型

2、数据类型转换

1、隐式转换

在某些操作中，变量的数据类型会自动的产生转换操作

数字 + 字符串 ：数字转换为字符串

数字 + 布尔值 ：将布尔值转换为数字

字符串 + 布尔值 ：将布尔值转换为字符串

布尔值 + 布尔值 ：将布尔值转换为数字

2、转换函数

1、toString()

作用：将任意类型的数据转换为字符串

语法：var result = 变量.toString()

2、parseInt()

作用：将任意类型的数据转换为"整数"

语法：var result = parseInt(转换的数据);

注意：

parseInt() 把能转的部分都转掉，实在转不了，则返回一个NaN(Not a Number)

var result = parseInt("35"); 结果：35

var result = parseInt("35.5"); 结果：35

var result = parseInt("35abc"); 结果：35

var result = parseInt("你好"); 结果：NaN

var result = parseInt("你好35"); 结果：NaN

3、parseFloat()

作用：将任意类型的数据转换为"小数"

var result = parseFloat("35.3abc"); 结果：35.3

var result = parseFloat("abc35.3"); 结果：NaN

4、Number()

作用：将任意类型的数据转换为number类型

注意：只要包含非法字符，结果就是 NaN

语法：var result = Number(变量);

var result = Number("35.3"); 结果：35.3

var result = Number("35abc"); 结果：NaN

var input = window.prompt("请输入一个数字");

3、运算符 和 表达式

1、表达式

每个表达式一定会有一个结果

2、运算符

1、算术运算符

+，-，\*，/，%，++，--

1、% ：取余数

5 % 2 ：结果 1

2 % 5 ：结果 2

练习：

声明一个变量保存数字：1234

将以上数字的每位数字取出来保存在独立的变量中

输出：

个位：4

十位：3

百位：2

千位：1

2、++ ，--

只在自己的基础上做加1或减1的操作

var num = 10;

console.log(num++); //输出 ：10 ,输出后num变为11

console.log(++num); //输出 ：12,先由11自增为12，再输出

console.log(num); //输出 ：12

++在前，先自增，再使用

++在后，先使用，再自增

var num = 10;

++num;

console.log(num); //11

var a = 10;

10 (11)11 11(12) 12(13) (14)14 14

var result = a + ++a + a++ + a++ + ++a + a;

结果：72

3、注意

1、"+"两边的操作数如果不是number类型的话，则做连接操作

var num = 23;

var str = "32";

var r = num + str; //2332

var num1 = 23;

var num2 = 32;

var str = "Hello";

var r = num1 + num2 + str; //55Hello

r = str + num1 + num2;//Hello2332

r = str + (num1 + num2);//Hello55

2、除 + 以外的其它算术运算符，如果两边的操作数不是 number 的话，会通过Number()函数，默认进行转换，如果转换不成功，最终的运算结果就是 NaN

var num1 = 10;

var str1 = "15";

var str2 = "18";

var str3 = "18abc";

var r = num1 - str1; // ? -5

r = str1 - str2; // ? -3

r = str2 - str3; // ? NaN "18"-"18abc"

2、关系(比较)运算符

1、作用

做两个数据之间的大小，相等 或 不等 关系判断的

关系运算符的运算结果，一定是 boolean 类型

>,<,>=,<=,==,===,!=,!==

2、注意

console.log(3 > 10); // false

运算符两端，如果有一个数number类型的话，那么另外一个会自动通过 Number() 进行转换，然后再做比较,如果转换失败，结果为 false

console.log("3" > 10); // false

console.log("3a" > 10); // false

如果运算符两端都是字符串的话，则比较每位字符的Unicode码，再决定大小

console.log("张三丰">"张无忌"); // false

张三(19977)丰

VS

张无(26080)忌

=======================

console.log("3" > "10");// true

3(51)

VS

1(49)0

console.log("78">"132");// true

var a = 10;

var b = "10";

console.log(a == b); // true

console.log(a === b); // false

3、isNaN()

var a = window.prompt("Input a number:");

NaN:

1、参与算术运算时，结果一定是 NaN

2、参与比较运算时，结果一定是 false

判断是否为数字，使用：

isNaN()

var result = isNaN(变量);

isNaN判断变量是否为非数字，如果值为true，则不是数字，如果值为false，则是数字

isNaN("123"); // false

isNaN(123); // false

isNaN("123a"); // true

请输入一个数字：58

是奇数吗?false

是偶数吗?true

3、逻辑运算符

1、作用

条件的关联 以及 条件的取反操作

Q : 判断 变量 num 是否在 18 到 70 之间

&& , || , !

条件1 ，条件2 && ||

true true true true

true false false true

false true false true

false false false false

2、练习1

1、闰年判断

1、从弹框中输入一个年份，判断是否为闰年，如果是闰年，则打印输出 true ， 否则输出 false

条件：能被4整除，但不能被100整除，或者能被400整除

2、判断字符

1、从弹框中输入1个字符

2、在控制台中输出

是一个英文字符吗? true / false

是一个数字吗? true / false

是一个汉字吗? true / false

3、短路逻辑

1、短路 &&

条件1 && 条件2 ，如果条件1为false的话，就不用继续判断条件2，整个表达式的结果就为 false

条件 && 操作 , 如果条件为true，则执行后面的操作，如果条件为false，则不执行后面的操作。

练习：

1、从弹框中输入两个数字，分别保存到 a 和 b 两个变量中

2、如果 a > b 的话，则输出 ，a 比 b 大

2、短路 ||

条件1 || 条件2 如果条件1为true，则不用再执行条件2

条件 || 操作，当条件不为true的时候，再执行操作

3、作业

工资计算

1、从弹框中输入一个数字表示工资

2、如果 工资>=20000 , 提示 "高工资"

3、如果 工资在 8000 ~ 20000 之间，提示"中高工资"

4、如果 工资<8000，则提示 "普通工资"

### Day03 运算符,函数,作用域

1、运算符

1、位运算符

1、作用

将数字转换为二进制后进行运算

只做整数运算，如果是小数的话，则去掉小数位再运算

2、位运算

1、按位 与 ：&

语法 ：a & b

特点 ：将 a 和 b 先转换为二进制，按位比较，对应位置的数字都为1的话，那么该位的整体结果为1，否则就为0

ex：5 & 3

5 ：101

3 ：011

===========

001

结果 ：1

使用场合：任意数字与1做按位与操作，可以判断奇偶性，结果为1，则为奇数，否则为偶数

0 ：0

1 ：1

2 ：10

3 ：11

4 ：100

5 ：101

5 & 1

101

001

==========

001

4 & 1

100

001

====

000

2、按位 或 ：|

语法 ：a | b

特点 ：将 a 和 b 转换为 二进制，按位比较，对应位置的数字，至少有一位为1的话，那么该为的整体结果就为1，否则为 0

ex :

5 | 3

101

011

========

111 结果为 ：7

适用场合：任何小数与0 做 按位或的操作，可以快速转换为整数(并非四舍五入)

5 | 0

101

000

====

101

3、按位 异或 ：^

语法：a ^ b

特点：将 a 和 b 先转换为二进制，按位操作，对应位置上的两个数字，相同时，该位整体结果为0，不同时，该位的整体结果为 1

使用场合：快速交换两个数字

5 ^ 3

101

011

==========

110 结果为 6

练习：

var a = 5;

var b = 3;

a = a ^ b;

b = b ^ a;

a = a ^ b;

console.log(a,b);

3、赋值 和 扩展赋值运算符

1、赋值运算符 ：=

2、扩展赋值运算符

+=，-=，\*=，/=，^=，... ...

a += b; ==> a = a + b;

a = a + 1; ==> a += 1; ==> a++ ; ==> ++a

3、练习

1、从弹框中，分两次输入两个数字，分别保存在 a 和 b中

2、如果 a 大于 b的话 ，则交换两个数字的位置

使用 短路&&,扩展赋值运算符,位运算

4、条件运算符(三目运算)

单目(一元)运算符 ：++,--,!

双目(二元)运算符 ：+，-，\*，/，%，>,<, ... ...

三目(三元)运算符 ：?:

1、语法

条件?表达式1:表达式2;

条件是一个boolean类型的数据，如果条件结果为true，则执行表达式1的内容，并将表达式1的结果作为整体表达式的结果。如果条件为false，则执行表达式2的内容，并将表达式2的结果作为整体表达式的结果

ex:

var age = 28;

var msg = age >= 18 ? "你已成年" : "你未成年";

练习：

从弹框中录入一个数字表示考试成绩(score)

如果 成绩为 100 分 ，提示 ：满分

如果 成绩 >= 90 分 ，提示 ：优

如果 成绩 >= 80 分 ，提示 ：良

如果 成绩 >= 60 分 ，提示 ：及格

否则 ：提示 不及格

#### 2、函数

1、什么是函数

函数(function),是一段预定义好，并且可以被反复使用的代码块

预定义好 ：事先声明，但不是马上执行

反复使用 ：可以被多次调用

代码块 ：包含多条可执行的语句

2、函数的声明与调用

1、普通函数

语法：

function 函数名(){

语句块;

}

调用：在JS中任何的合法位置处，都可以通过 函数名() 的方式进行调用

练习：

1、声明一个函数，名称为 change

2、在函数中

1、通过弹框,分两次，录入两个数字，保存在 a 和 b

2、先打印 a 和 b的值

3、如果 a > b 的话，则交换两个数字的位置

4、再打印 a 和 b的值

3、在网页中，创建一个按钮，点击按钮时，完成 change 函数的调用

2、带参函数

1、定义语法：

function 函数名(参数列表){

语句块;

}

参数列表：

可以由0或多个参数的名称来组成，多个参数的话中间用 , 隔开

定义函数时的参数列表，都称为 "形参(形式参数)"

2、调用语法

任意合法JS位置处

函数名(参数列表);

调用函数时，所传递的参数列表，称之为"实参(实际参数)"

3、练习

1、定义一个函数 change ，该函数中接收两个参数(a,b)

2、在函数体中，如果 a 大于 b的话，则交换两个数字的位置，再打印输出结果

3、通过一个按钮调用函数，并且将 两个数字传递给函数

4、常用带参函数

parseInt(变量);

parseFloat(变量);

Number(变量)

console.log("");

3、带返回值的函数

var result = parseInt("35.5");

1、什么是返回值

由函数体内 带到 函数体外的数据，叫做"返回值"

2、语法

function 函数名(参数列表){

语句块;

return 值;

}

允许通过一个变量来接收调用函数后的返回值

var 变量名 = 函数名(实参列表);

3、练习

定义一个可以接收三个Number参数的方法(名称为getMax)，在该方法中，计算并返回 最大的数值

#### 3、作用域

1、什么是作用域

指的是变量和函数的可访问范围，作用域分为以下两类

1、函数作用域

只在定义的函数内允许访问变量 和 函数

2、全局作用域

一经定义，在任何位置处都能访问

2、函数作用域中的变量

在某个函数中声明的变量，就是函数作用域中的变量，也可以称之为 "局部变量"。

function calSum(){

var a = 3;

var b = 5;

console.log(a,b); // 输出 3 5

}

console.log(a,b); //错误，因为 a ，b已经出了它的定义范围

练习：

1、定义一个 函数calSum，声明两个变量a，b，并赋值，并且在函数内打印两个变量

2、定义一个 函数calMax，直接打印输出 a，b，观察结果

3、全局作用域中的变量

一经声明了，任何位置都能使用。也称为 "全局变量"

声明方式：

1、将变量声明在最外层，所有的function之外

<script>

var a = 15;

var b = 18;

function showMsg(){

console.log(a);

console.log(b);

}

</script>

2、声明变量时，不使用var关键字，一律是全局变量，但有风险

建议：

全局变量 ，尽量都声明在 所有的 function 之外

声明变量时，也一律都使用 var 关键字

#### 4、声明提前

Q ：

function show(){

console.log(num);//报错

}

1、什么是声明提前

JS程序在正式执行前，会将所有var声明的变量以及function声明的函数，预读到所在作用域的顶端。但赋值还保留在原位置上

2、建议

最好将变量的声明 和 函数的声明都集中到所在作用域的最顶端

#### 5、按值传递

Q :

function change(a,b){

a ^= b;

b ^= a;

a ^= b;

console.log("在change函数中");

console.log(a,b);

}

function testChange(){

var a = 15;

var b = 18;

console.log("调用前：");

console.log(a,b);//15 18

change(a,b);

console.log("调用后：");

console.log(a,b);//15 18

}

1、什么是按值传递

基本数据类型的数字在做参数传递时，实际上时传递的实参的副本到函数中，原始数据并未发生过改变

2、建议

基本数据类型的数据在做参数传递时，尽量不要在函数中修改，因为即便修改，也不会影响原数据的

#### 6、局部函数

将函数 再声明在某个函数内，就是局部函数

<script>

function show(){

function inner(){

console.log("show 中的 inner 函数");

}

inner();//正确

}

inner(); // 错误，inner 是局部函数，只能在 show中被调用

</script>

#### 7、全局函数

ECMAScript预定义的全局函数,可供用户直接使用

1、paseInt();

2、parseFloat();

3、Number();

4、isNaN();

5、encodeURI()

URL:Uniform Resource Locator ：统一资源定位符/器

URI:Uniform Resource Identifier : 统一资源标识符

1、作用

对统一资源标识符格式的字符串进行编码，并返回编码后的字符串

编码：将多字节的字符，编译成多个单字节的字符

6、decodeURI()

1、作用

对已编码的URI进行解码(将已编码内容再转换回中文)

7、encodeURIComponent()

在 encodeURI 基础之上，允许将 特殊符号(:,/,...) 进行编码

8、decodeURIComponent()

9、eval()

作用：计算并执行以字符串方式表示的JS代码

var msg = "console.log('Hello World');";

eval(msg);//将 msg 作为 JS代码去执行

练习：

1、创建页面，通过弹框，输入一段JS代码

2、通过 eval() ，将输入的JS代码执行一次

#### 3、递归调用

1、什么是递归

递归指的是在一个函数中，又一次的调用了自己

2、递归的实现

1、边界条件

2、递归前进 - 继续调自己

3、递归返回 - 向上返回数据

3、解决问题

1、计算某个数字的阶乘

5的阶乘

5!=5\*4\*3\*2\*1

5! = 5 \* 4!

4! = 4 \* 3!

3! = 3 \* 2!

2! = 2 \* 1!

1! = 1

/\*计算 n 的阶乘\*/

function f(n){

... ...

}

4、作业

已知下列数列 (菲波那切数列)

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

已知 第一个数为1，第二个数为1

从第三个数开始，每个数字等于前两个数之和

问题：通过一个函数，求某个位置上的数字（用递归实现）

https://www.baidu.com/s?wd=%E4%BD%B3%E8%83%BD

https://www.baidu.com/s?wd=尼康

### Day04函数,选择,循环

function test(){

console.log("Hello World");

}

var result = test();

console.log(result);

===========================

输出结果：

Hello World

undefined

<script>

var g\_uname = "sanfeng.zhang";

function showName(){

var uname = "wuji.zhang";

console.log(uname);//wuji.zhang

}

</script>

function test(){

console.log(num); // 打印 ：undefined

var num = 15;

}

演变成：

function test{

var num; // 先声明，但不赋值(自动完成)

console.log(num); // 打印 ：undefined

num=15;// 只赋值，不声明

}

推荐写法：

function test(){

var num = 15;

console.log(num); // 打印 ：15

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

作业：

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

已知：

该数列中前两个数都是 1

从第三个数开始，每个数字等于前两个数的和

/\*通过 f 函数，计算第 n 个数字是多少\*/

function f(n){

}

f(5) : 求第五个数字是多少

=================================================

#### 1、分支(选择)结构

1、流程控制结构

1、顺序结构 - 自顶向下的执行

2、分支结构 - 有条件的选择某一段代码去执行

3、循环结构 - 有条件的选择某一段代码反复执行

程序 = 数据结构 + 算法

2、练习

编写一个收银柜台收款程序，根据商品单价，购买数量以及收款金额计算并输出应收金额和找零

数据：商品单价，购买数量，收款金额，应收金额，找零

输入数据：商品单价(price),购买数量(count),收款金额(money)

输出数据：应收金额(total),找零(change)

Q : 当商品总价>=500时，享受8折优惠

3、if 结构

语法：

if(条件){

满足条件要执行的语句块

}

流程：

1、判断条件

2、如果条件为真，则执行 语句块 中的内容

3、如果条件为假，则跳过语句块去执行其他内容

注意：

1、if中的条件，最好是一个boolean的值，如果不是boolean类型，则会自动转换

以下情况，条件会自动转换为 false

if(0)

if(0.0)

if(undefined)

if(null)

if("")

if(NaN)

if(35.5){ //真 }

var num;

if(num){ //假 }

if("李文华真帅"){ //真 }

2、if 后的 { } 是可以被省略的，如果省略的话，只控制 if 下的第一条语句

问题：考虑异常情况，如果收款金额小于应收金额

如果 收款金额大于等于应收金额，则正常执行

否则 则给出异常提示

4、if ... else ... 结构

语法：

if(条件){

满足条件时，执行的语句块

}else{

不满足条件时，执行的语句块

}

练习：

1、从弹框中，输入一个年份信息

2、判断该年是否为闰年，并给出提示

输入年：2009

2009年不是闰年

输入年：2012

2012年是闰年

5、if ... else if ... 结构

语法：

if(条件1){

//满足条件1时，要执行的语句块

}else if(条件2){

//满足条件2时，要执行的语句块

}else if(条件n){

//满足条件n时，要执行的语句块

}else{

//以上条件都不满足时，要执行的语句块

}

练习：

从弹框中，分三次录入 年，月，日

判断该日是该年的第多少天

提示：

1,3,5,7,8,10,12 ：每月31天

4,6,9,11 ：每月30天

2 ：闰年29天，平年28天

2017年8月21日 ：

31+29+31+30+31+30+31+21

6、switch ... case

Q :

从弹框中录入 1-7 中的任一一个数字

录入 1 ：输出 ：今天吃红烧肉

录入 2 ：输出 ：今天吃红烧排骨

录入 3 ：输出 ：今天吃红烧丸子

录入 4 ：输出 ：今天红烧鱼

录入 5 ：输出 ：今天吃烤羊腿

录入 6 ：输出 ：今天休息

录入 7 ：输出 ：今天休息

场合：等值判断

语法：

switch(变量){

case 常量值1:

语句块1;

break;//通过 break 结束switch的执行(可选 )

case 常量值2:

语句块2;

break;//可选

default:

语句块n;

break;//可选

}

练习：

1、使用 特殊的 switch 结构完成下列要求

输入年 ，月

输出 该月有多少天

1,3,5,7,8,10,12 : 31天

4,6,9,11 : 30 天

2 : 平年28天，闰年29天

ex:

年：2017

月：5

输出：31天

1、使用 特殊的 switch 结构完成下列要求

2、改版日期计算器(难度)

输入年月日，判断该日是这一年的多少天

var month = Number(prompt());

var totalDays = 0;

switch(month-1){

case 11:

totalDays += 30;

case 10:

totalDays += 31;

case 9:

totalDays += 30;

... ...

}

#### 2、循环结构

1、问题

1、控制台上打印一句Hello World

2、控制台上打印十句Hello World

3、控制台上打印10000句 Hello World

4、改版第3步，增加

第 ? 句 Hello World

第 1 句 Hello World

第 2 句 Hello World

...

第 10000 句 Hello World

5、改版第4步

将 Hello Word 替换成 "你好 世界"

练习：

6、打印输出 1-100之间所有数字的和

2、什么是循环结构

循环，就是一遍又一遍的执行相同或相似的代码

两个重要的要素：

1、循环条件 ：循环执行的次数

2、循环操作 ：循环体，要执行相同 或 相似的代码是什么

3、while 循环

1、语法

while(条件){

//循环操作

}

流程：

1、判断条件

2、如果条件为真，则执行循环操作，然后再次判断条件。当条件为假时，则退出循环结构

4、循环的流程控制语句

1、break

破坏了整个循环的执行 - 结束循环

2、continue

结束本次循环，继续执行下次循环

练习：

1、使用 continue ，打印输出 1-100之间所有偶数的和

2、让用户循环从弹框录入信息，并将信息打印在控制台上，直到用户输入 exit 为止

输入数据：sanfeng.zhang

sanfeng.zhang

输入数据：wuji.zhang

wuji.zhang

输入数据：exit

3、猜数字游戏

1、随机生成一个 1-100 之间的数字

Math.random() 返回 0-1 之间的小数

var r = parseInt(Math.random()\*100)+1;

2、要求用户输入一个整数

若比生成的随机数大，则输出 “猜大了”

若比生成的随机数小，则输出 “猜小了”

直到用户 猜对为止

3、如果用户输入exit ,也可以退出游戏

3、作业

用户从弹框中输入年，月，日，判断该日是星期几?

1900年1月1日 是星期1

推荐思路：

从1900年1月1日 到 输入的年月日 共有多少天求出来，与 7 取余

1、通过一个循环计算从1900年 到输入年前一年有多少天?

2、再从1月，到输入月份的前一个月共有多少天?

3、再累加 输入的日

4、用整体结果 与7取余，给出提示

附加作业：

将 输入的月份的日历打印输出

2017年8月

日 一 二 三 四 五 六

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19、

20 21 22 23 24 25 26

27 28 29 30 31

### Day05循环,数组

while(i > 0){

switch(i){

case 1:

console.log();

break;

case 2:

console.log();

continue; // 作用在 while

}

}

switch(i){

case 1:

console.log();

continue; // 错误

}

===================================================

1、循环结构

1、do ... while()

do{

循环操作

}while(循环条件);

do...while VS while

1、while

先判断条件，再执行循环操作

如果条件不满足，循环一次都不执行

2、do ... while

先执行循环操作，再判断循环条件

即便条件不满足，也要执行一次循环操作

练习：

将猜数字的游戏，改版成 do ... while 循环版本

2、for() 循环

1、语法

for(表达式1;表达式2;表达式3){

//循环操作

}

表达式1：循环条件的声明

表达式2：循环条件的判断(boolean)

表达式3：更新循环条件(i++,... ...)

执行过程：

1、计算表达式1的值

2、计算表达式2的值，如果结果为true则执行循环体，否则退出

3、执行循环体

4、执行表达式3

5、再计算表达式2的值，为true执行循环体，否则退出

while do...while 和 for 的使用场合：

1、while 和 do...while

适合使用在不确定循环次数的循环下

2、for

适合使用在确定循环次数时使用

练习：

1、定义一个函数，打印乘法口诀表中的任一一行

输入要打印的行数：5

1\*5=5 2\*5=10 3\*5=15 4\*5=20 5\*5=25

2、定义一个函数，打印 1/1+1/3+1/5+...1/999的和

3、定义一个函数，判断指定的数字是否为素数

素数：只能被1 和 它本身整除的数字

输入一个数字：7

是素数

循环条件：从 2 开始 ，到输入的数字-1结束

循环操作：判断输入的数字能否被循环条件整除,能被整除，则不是素数，不能被整除，是素数

4、有一对兔子，从出生后的第3个月起都生一对兔子，小兔子长到第3个月后每个月又生一对兔子，假如兔子都不死的话，打印输出 前10个月，每个月有多少对兔子

n1 n2 n1 n2 n1 n2

1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 , 55

2、三个表达式的特殊用法

1、for(;;)

三个表达式可以任意的省略，不推荐省略

2、for 第一个 和 第三个表达式的多样化

for(var i=1,j=2;i<=10;i++,j++)

练习：

1 + 6 = 7

2 + 5 = 7

3 + 4 = 7

4 + 3 = 7

5 + 2 = 7

6 + 1 = 7

for(var i=1,j=6;j>=1;i++,j--){

console.log(i+"+"+j+"="+(i+j));

}

3、嵌套循环

在一个循环的内部又出现一个循环

for(var i=0;i<10;i++){ // 外层循环

for(var j=0;j<10;j++){ // 内层循环

}

}

外层循环执行一次，内层循环要执行一轮

练习：

1、控制台中输出下面的图形

\*\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*\*

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

2、数组

1、什么是数组

数组，即一组数据，使用一个变量来存放多个数据

数组中的元素是按照"线性"书序来排列的，所以也可以称之为是一个 "线性表"

2、声明数组

1、声明一个空数组变量

var 数组名 = [];

2、声明数组并赋初始值

var 数组名 = [数据1,数据2,数据3,... ...];

3、声明一个空数组变量

var 数组名 = new Array();

4、声明一个数组并赋初始值

var 数组名 = new Array(数据1,数据2,... ...);

练习：

1、创建一个函数，分别用两种不同的方式创建数组(arr1,arr2)

arr1保存 ：张三丰，张翠山，张无忌

arr2保存 ：金花婆婆，殷素素，赵敏

2、直接将数组打印在控制台上

3、数组是引用类型的对象

引用类型的对象，是将数据保存在 "堆" 中的

在"栈"中，会对 "堆" 中数据的地址进行引用

1、将数组赋值给其它变量时，实际赋的是数组的地址

练习：

1、创建一个数组 arr1 ，包含"张无忌","赵敏"

2、将 arr1 赋值给 arr2

3、将 arr2 的第一个元素更改为 "金花婆婆"，打印输出 arr1 和 arr2 中的所有元素

2、数组在做参数的时候，传递进去的实际上是地址(按引用传递 即 按地址传递)

练习：

1、创建一个数组array，保存"张无忌","赵敏"俩元素

2、创建一个函数change，接收一个数组作为参数，在函数体内，将第一个元素更改为"金花婆婆"

3、调用change函数，并将array数组作为参数，调用完成后，打印array的值，观察结果

4、修改change函数，在函数体内，将传递进来的数组重新new一个Array("孙悟空","猪八戒")

5、重复步骤三，观察结果

3、null

表示的是让引用类型的对象不再指向任何空间.一般用于主动释放对象的空间

//让 array 指向数组的地址

var array = ["wuji.zhang","zhaomin"];

//释放 array空间，不再指向任何地址

array = null;

4、访问数组的元素

赋值，取值，都是使用 下标 来完成的

1、为数组的元素赋值

数组名[下标] = 值;

var array = ["张无忌","赵敏"];

array[1] = "周芷若";

array[2] = "小昭";//增加一个新元素，在第3个位置处

array[5] = "灭绝师太";

2、获取数组中的值

数组名[下标];

3、length属性

作用：获取数组的长度(数组中元素的个数)

1、允许将数组的length属性值设置为0，来完成数组元素的清空操作

2、配合循环 做数组的循环遍历操作

var array = ["张无忌","赵敏"];

输出：

第1名 ：张无忌

第2名 ：赵敏

for(var i=0;i<array.length;i++){

console.log("第"+(i+1)+"名:"+array[i]);

}

3、配合着数组，找到最新元素要插入的位置

var array = ["张无忌","赵敏"];

array[array.length] = "金花婆婆";

作业：

1、定义一个函数，找出一个整数数组中的最大值

var nums = [38,5,62,789,12];

该数组中的最大值为：789

推荐步骤：

1、声明函数 findMax，在函数体中，声明数组

2、声明一个变量 max，保存该数组中的最大值，并且设置数组的第一个元素为 max 的默认值

3、从第二个元素开始，依次与 max 进行比较，只要比max大，就赋值给max

2、定义一个函数，查找指定元素的位置(下标)

var nums = [38,5,62,789,12];

用户输入：12

该元素的下标为：4

用户输入：6

该元素不存在

day18

var arr1 = [36,78];

var arr2 = arr1;

var num1 = 10;

var num2 = num1;

console.log();

arr.length;

1、练习

要求用户手动的录入整数数字，并且保存到数组中，输入 exit 为止，并且查找该数组中的最大值是多少

请输入数字(exit退出):78

请输入数字(exit退出):56

请输入数字(exit退出):126

请输入数字(exit退出):35

... ...

请输入数字(exit退出):exit

数组为：[78,56,126,35]

该数组中的最大值为:126

=================================================

1、数组

1、使用 for ... in 遍历数组元素

语法：

for(var 变量 in 数组){

//变量：表示的是数组中每个元素的下标

}

2、关联数组

php中 ：$array = ["name"=>"sanfeng.zhang"];

用字符串做下标，就是关联数组

var array = []; //声明一个空数组

array["西游记"] = "古代神话故事";

array["红楼梦"] = ["贾宝玉","林黛玉","刘姥姥"];

Q : array.length

注意：

1、关联数组是不算做数组内容的，不记录到 length 中

2、关联数组只能通过 字符串 做下标取值

3、允许通过 for...in 来循环遍历关联数组的字符串下标(也能遍历出内容的数字下标)

3、冒泡排序

[23,9,78,6,45] -> [6,9,23,45,78]

冒泡排序：车轮战，两两比较，小的靠前

特点：

1、轮数 ：共比较了 length - 1 轮

2、每轮中比较的次数 ：随着轮数的增加，次数反而减少

代码：

双层循环表示整个排序的过程

1、外层循环 ：控制比较的轮数，从1开始，到length-1(能取到)结束

2、内层循环 ：控制每轮中比较的次数，并且也要表示参与比较的元素的下标，从0开始，到 length-1-i(轮数变量)

[23,9,78,6,45]

共5个元素

外层：从 1 开始，到 4 结束

内层：

第一轮 第一次：从 0 开始 ，到 3 结束

第二轮 第一次：从 0 开始 ，到 2 结束

... ...

#### 2、数组的常用方法

1、toString()

将一个数组转换为字符串

语法：var str = 数组对象.toString();

2、join()

作用：将数组中的元素通过指定的分隔符连接到一起，并返回连接后的字符串

语法：var str = 数组对象.join("&");

练习：(10分钟)

1、使用数组实现由 \* 组成的三角形

直角三角形，等腰三角形

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

2、使用数组实现 九九乘法表

3、连接数组

函数：concat()

作用：拼接两个或更多的数组，并返回拼接后的结果

语法：var result=arr1.concat(arr2,arr3,arr4);

注意：

1、concat不会改变现有数组，而是返回拼接后的

2、每个参数不一定非是数组，也可以是普通的数据

练习：

1、声明两个全局数组，分别保存一些国家的名称

数组1：选中的国家

sel = ["中国","美国","俄罗斯","日本"];

数组2：备选国家

unsel=["朝鲜","越南","老挝","柬埔寨"];

2、将 备选国家 全部移入到 入选国家中，打印数据

结果：

sel = ["中国","美国","俄罗斯","日本","朝鲜","越南","老挝","柬埔寨"];

unsel=[];

4、获取子数组

函数：slice()

作用：从指定数组中，截取几个连续的元素组成的新数组

语法：arr.slice(start,[end]);

start:从哪个下标处开始截取，取值为正，从前向后取，取值为负，从后向前算位置。

0 1 2 3

var arr=["中国","美国","日本","英国"];

-4 -3 -2 -1

end : 指定结束处的下标(不包含)，该参数可以省略，如果省略的话，就是从start一直截取到结束

注意：

1、该函数不会影响现有数组，会返回全新的数组

练习：

var arr=["中国","美国","日本","英国"];

取出 美国 和 日本 组成的一个子数组

5、修改数组

函数：splice()

作用：允许从指定数组中，删除一部分元素，同时再添加另一部分元素

语法：

arr.splice(start,count,e1,e2,... ...);

start:指定添加或删除元素的起始位置/下标

count:要删除的元素个数，取值为0表示不删除

e1,e2,... ... : 要增加的新元素，允许多个

返回值：返回一个由删除元素所组成的数组

注意：splice 会改变现有数组

练习：

在 sel 和 unsel 的基础上完成下列操作

1、从备选国家中(unsel)，选择一个国家移入到 选中的国家(sel)

推荐步骤：

1、用户输入 ：越南

从 unsel 中将越南移除

再将 越南 sel 的末尾处

2、用户输入 ：墨西哥

提示：国家不存在

6、数组的反转

作用：颠倒数组中元素的顺序

语法：arr.reverse();

注意：该函数会改变当前数组的内容

练习：

1、声明一个整数数组

2、倒序打印输出内容(使用reverse())

7、数组排序

函数：sort();

语法：arr.sort(排序函数);

作用：默认情况下，按照元素的Unicode码大小按升序排列

特殊：允许自己指定排序函数，从而实现对数字的升序和降序的排列

语法：arr.sort(排序函数);

ex:

var arr = [12,6,4,72,115,89];

//排序函数(升序)

function sortAsc(a,b){

return a-b;

}

arr.sort(sortAsc);

原理：

1、指定排序函数 如 sortAsc,定义 两个参数，如 a 和 b。数组会自动传递数据到 sortAsc 里面去，如果返回值是>0的数，则交换两个数的位置，否则不变

使用匿名函数完成排序：

arr.sort(function(a,b){return a-b;});

练习：

1、声明一个整数数组，随意定义数字

2、页面上添加两个按钮，一个"升序"按钮，一个"降序"按钮

3、点击 升序 按钮时，数组按升序排序，并打印

4、点击 降序 按钮时，数组按降序排序，并打印

8、进出栈操作

JS是按照标准的"栈式操作"访问数组的

所有的"栈式操作"的特点都是"后进先出"

"栈式操作"讲究的"入栈"和"出栈"

1、push()

入栈，在栈顶(数组的尾部)添加指定的元素，并返回新数组的长度

var arr = [10,20,30];

//向栈顶增加新数据 40

var len = arr.push(40);

//len 保存的是 40 入栈后 arr 的长度，值是4

2、pop()

出栈，删除并返回栈顶的(数组尾部)元素

var arr = [10,20,30];

var r1 = arr.pop();//arr = [10,20]

var r2 = arr.pop();//arr = [10]

==============================

arr :

r1 : 30

r2 : 20

3、shift()

删除数组头部的(第一个)元素并返回

var arr = [10,20,30];

var r1 = arr.shift();

============================

arr : [20,30]

r1 : 10

4、unshift()

作用：在数组的头部(第一个)元素位置处，增加新元素

var arr = [10,20,30];

arr.unshift(40);

arr : [40,10,20,30]

3、二维数组

1、什么是二维数组

一个数组中的元素又是一个数组，也可以称之为：数组的数组

2、创建二维数组

var names=[

["孙悟空","猪八戒","沙悟净"],

["潘金莲","西门庆","武大郎"],

["贾宝玉","刘姥姥","林黛玉"],

["貂蝉","貂蝉","貂蝉"]

];

//获取猪八戒

console.log(names[0][1]);

//获取林黛玉

console.log(names[2][2]);

4、作业

1、通过 进出栈 操作，完成 十进制对二进制的转换

1、创建一个函数 decode2bin,接收一个十进制的数字做参数

2、函数体中，将参数 拆成二进制数字，保存在一个数组中

3、将数组的元素使用""连接，并返回

十进制 转 二进制 ：除2取余法

num : 5

num % 2 : 结果为 1

num = parseInt(num / 2) : 结果 2

num % 2 : 结果为 0

num = parseInt(num / 2) : 结果 1

num % 2 : 结果为 1

num = parseInt(num / 2) : 结果 0

结果为 0 时则不用再继续 %2

2、根据省份信息，查询城市列表

1、创建一个全局数组，保存几个省份名称

2、创建一个全局二维数组，保存不同省份中的城市信息(数据要与省份数组中的数据对应的上)

3、创建一个函数，接收一个省份名称做参数，打印该省份中对应的所有的城市信息

### Day06 HTTP协议

#### 1、HTTP 协议

1、作用

规范了数据是如何打包的 以及 数据时如何传递的

2、Message 消息 / 报文

Message 指的是在HTTP客户端与服务器间传递的数据块

分类：

1、Request Message : 客户端向服务器发送的请求消息

2、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

#### 3、请求消息(Request Message)

1、请求起始行

1、请求方法

1、GET

表示客户端向获取服务器上的资源(img/js/..)

特点：

1、无请求主体

2、依靠地址栏传递数据给服务器

2、POST

表示客户端想传递数据给服务器

特点：

1、有请求主体

3、PUT

表示客户端想把文件放到服务器上(禁用)

特点：

1、有请求主体

4、DELETE

表示客户端想删除服务器上指定的文件(禁用)

5、HEAD

表示只想获取指定的响应头

6、CONNECT

测试连接

7、TRACE

追踪请示路径

8、OPTIONS

选项，保留以后使用... ...

2、请求URL

3、协议 和 版本号

HTTP / 1.1

2、请求头

1、Host:localhost

告诉服务器请求哪一个虚拟主机

2、Connection:keep-alive

告诉服务器做持久连接

3、User-Agent

告诉服务器，客户端浏览器的类型

4、Cache-Control:max-age=0

告诉服务器缓存信息，max-age，即不缓存

5、Accept-Language : zh-CN

告诉服务器自己能接收的自然语言

6、Accept-Encoding:gzip

告诉服务器自己可以接收的压缩类型

7、Referer: url

告诉服务器请求来自于哪个页面

3、请求主体

Form Data : 是真正的请求主体

注意：查询字符串

Query String : 不算作请求主体的，是属于 URL 中的一个部分

#### 4、响应消息(Response Message)

1、响应起始行

1、协议版本号

HTTP / 1.1

#### 2、响应状态码

1、1XX

提示信息

2、2XX

200 ：服务器端成功响应所有的信息 OK

3、3XX

需要客户端进行重定向

301 ：永久性重定向

302 ：临时重定向

304 ：Not Modified

4、4XX

客户端请求错误

404 ：Not Found，请求资源不存在

403 ：Forbidden，没有访问权限

405 ：Method Not Allowed，请求方法不被允许

5、5XX

服务器运行错误

500 ：服务器内部错误

501 ：没有实现

3、原因短句

对状态码的简单解释

200 : OK

404 : Not Found

... ...

#### 2、响应消息头

1、Content-Type

响应的主体类型，告诉浏览器，响应的数据时什么格式以及什么类型的

取值：

1、text/plain

纯文本，告诉浏览器按纯文本的方式解析

2、text/html

文本与网页，告诉浏览器按html的格式解析内容

3、text/css

样式，告诉浏览器按 css 的方式解析

4、application/javascript

JS代码片段，按JS的方式运行数据

5、application/xml

按 xml 的方式解析

6、application/json

按 json 的方式解析

3、响应主体

由服务器端响应回来的数据

#### 2、缓存

1、缓存的工作原理

客户端可以自动保存已访问过得文档的副本，这些副本就是所谓的缓存

当客户端再向同一URL发送请求时，那么就直接从缓存中将文件取出来，而不用再重新发送请求

2、优点

1、减少了冗余的数据传输，节省流量

2、缓解服务器带宽瓶颈的问题，服务器可以节省更多的带宽

3、降低了对服务器的资源消耗和运行要求

4、降低了由于远距离而造成的加载延迟

3、与缓存相关的消息头

1、Cache-Control

作用：从服务器将文档传来之时起，认为新鲜的秒数

取值：秒数 60 / 3600 / ... ...

如果取值为 0 / no-cache，表示每次都要重新刷新网页

2、Expires

明确指定缓存的过期时间,取值是格林尼治标准时间(GMT)

Expires:Fri,25 Aug 2017,08:00:00 GMT

Expires:0 表示不需要缓存

4、如何在网页中设置消息头

在 meta 标记中完成消息头的设定：

1、http-equiv ：指定消息头的名称

2、content ：指定消息头对应的内容

练习：

设置网页的消息头Cache-Control,其值为 no-cache

<head>

<meta charset="utf-8">

<meta http-equiv="Content-Type" content="text/html">

</head>

#### 3、DOM操作

DOM:Document Object Model 文档对象模型

让JS又能力操作页面的元素

1、使用JS获取页面上的某个元素

1、为元素增加ID属性

<div id="d1">Hello Wolrd</div>

2、在JS中，允许通过元素的ID获取页面元素

var elem = document.getElementById("元素ID");

elem 就是指定ID值的元素在JS中的表现形式

2、修改/获取 标记内的内容

属性：innerHTML

赋值：为 某元素的 innerHTML 属性赋值

取值：获取某元素的innerHTML 属性

ex:

var elem=document.getElementById("d1");

elem.innerHTML = "你好中国!";

console.log(elem.innerHTML);

练习：(5分钟)

1、在页面中创建一个 div，设置其内容为 Hello World ，并设置ID为 d1

2、添加按钮，获取 d1 元素中的HTML内容，并且打印在控制台上

3、添加按钮，将 d1 的内容更改为 "这是我的第一个DOM操作"

3、获取/设置 表单控件的数据

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

在DOM中，文本框，密码框，... ...也都是通过value属性来获取或设置控件的值

步骤：

1、获取页面表单控件元素(通过id获取)

2、通过元素的 value 属性 获取/设置 控件的值

练习：

1、页面中创建一个文本框，一个div，一个按钮

2、点击按钮时，将文本框中的数据，以一级标题的形式显示在div中

4、简化 document.getElementById()

document.getElementById 的作用：

根据 "指定的ID值" 获取对应的 "HTML元素"

/\*\*

\* 根据指定的ID值，获取对应的HTML元素

\* 参数 id ：要获取的元素的ID值

\* 返回值 ：获取的对应的HTML元素

\*/

function $(id){

return document.getElementById(id);

}

练习：

1、创建一个 common.js 的文件，封装$函数，负责接收一个id值作为参数，并且返回对应的对象

2、将练习

1、页面中创建一个文本框，一个div，一个按钮

2、点击按钮时，将文本框中的数据，以一级标题的形式显示在div中

中所有的 document.getElementById 替换成 $()

#### 5、HTML 元素的事件

事件：在某些特殊的情况下，可以被激发的一个操作

常用事件：onclick,... ...

1、文本框 和 密码框 的事件 - onblur

onblur事件 ：鼠标失去焦点时的事件

2、文本框 和 密码框 的事件 - onfocus

onfocus事件 ：获取焦点时要执行的操作

练习：

1、页面中创建一个 文本框，后面再创建一个span

2、文本框失去焦点时，判断文本框中的数据是否为空，如果为空，在span中提示用户名不能为空，否则提示 通过

3、<body> 的事件 - onload

在网页加载的时候，要执行的操作，可以封装在 onload 中

## AJAX异步请求

### Day01 DOM,Ajax,GET

1、DOM

1、事件

onchange

练习：

1、创建一个二维数组

var city = [

["哈尔滨",""],

["长春",""],

["沈阳",""]

];

2、当 selProvince 发生改变时，将对应的 城市信息 显示在selCity中

如果选项是 ===请选择=== ，则将 请先选择省份，显示在 selCity 中

#### 2、AJAX

1、名词解释

1、同步

在一段时间内，只能做一个事情

同步访问：在访问服务器时，客户端只能等待服务器的响应，不能做其它事情。用户体验非常的差

代表场合：

1、输入网址访问页面

2、a标记的默认跳转

3、submit按钮的表单提交

2、异步

在一段时间内，可以同时干多个事情

异步访问：在向服务器发送请求时，不耽误用户在网页上做其它的操作

代表场合：

1、用户名的重复性验证

2、聊天室

3、股票走势图

4、搜索建议(百度，京东，淘宝 的搜索框)

2、什么是AJAX

Asynchronous Javascript And Xml

异步的 JS 和 Xml

本质：使用 JS 提供的 XMLHttpRequest 对象 异步的向服务器发送请求，并接受响应数据（格式是 xml(已过时)）

AJAX中，服务器响应回来的是部分的数据而不是完整的页面，并且可以以无刷新的效果来更改页面中的局部内容

3、获取 AJAX核心对象 - XMLHttpRequest

标准创建:var xhr=new XMLHttpRequest();

IE8以下 :

var xhr=new ActiveXObject("Microsoft.XMLHttp");

允许通过 window.XMLHttpRequest 判断浏览器是否支持 XMLHttpRequest()。如果 window.XMLHttpRequest的值是null的话，说明需要通过 ActiveXObject()创建，否则 需要通过 XMLHttpRequest() 来创建

var xhr;

//判断浏览器是否支持 XMLHttpRequest

if(window.XMLHttpRequest){

xhr = new XMLHttpRequest();

}else{

//浏览器不支持 XMLHttpRequest

xhr = new ActiveXObject("Microsoft.XMLHttp");

}

练习：

1、创建 xhr 对象，并且将创建好的对象打印在控制台上观察结果

2、将 创建 xhr 的步骤封装成一个函数

1、在 common.js创建一个函数

function createXhr(){}

2、函数体内

判断浏览器是否支持 XMLHttpRequest，根据结果创建 xhr 并且将 xhr对象 返回

3、页面中创建一个按钮，点击按钮时，获取 xhr 对象，并打印在控制台上

#### 4、XHR 的常用属性 和 方法(难点)

1、open() - 方法

作用：创建请求

语法：open(method,url,isAsyn)

1、method

string 类型

请求方式 ：get,post,... ...

2、url

string 类型

请求地址

3、isAsyn

boolean 类型

指定采用同步(false)还是异步(true)的方式发送请求

练习：

1、创建一个网页 getresponse.html ,在网页中创建一个按钮，点击按钮时，实现以下功能：

1、创建 / 获取 XHR 对象

2、创建一个请求，采用异步的方式，get的提交方法，向 response.php 发送一个请求

2、readyState - 属性

作用：表示 xhr 对象的 请求状态

值：由 0 - 4 表示 5个状态

0 : 请求尚未初始化

1 : 已经打开到WEB服务器的连接，正在向服务器发送请求

2 : 请求完成

3 : 正在接收服务器端的响应

4 : 接收响应数据成功

注意：当 readyState的值为4的时候，表示所有的响应都已经接收完毕。

3、status - 属性

作用：服务器的响应状态码

值：

只记住一个即可 200

当 status 的值是 200的时候，表示服务器已经正确的给出所有的响应

4、onreadystatechange - 事件

作用：当 xhr 的readyState 属性值发生改变的时候，要自动激发的操作。

语法：

xhr.onreadystatechange = function(){

//每当xhr.readyState的值，发生变化时，要执行的操作。

//判断 xhr.readyState 为4的时候 并且 xhr.status 为 200的时候，才能获取正常的响应数据

if(xhr.readyState == 4 && xhr.status == 200){

//可以接收响应回来的数据

//responseText 属性，表示服务器响应回来的文本

var resText=xhr.responseText;

}

}

5、send() - 方法

作用：发送\提交请求

语法：send(body);

body : 是请求主体

没有请求主体的提交方式，body位置处，要写 null

有请求主体的提交方式，body位置处，编写的就是请求主体的数据

5、发送异步请求的步骤

1、创建 / 获取 XHR 对象

2、创建请求 ：xhr.open()

3、设置 xhr 的 onreadystatechange 事件

判断readyState 以及 status 的值，并接收响应

xhr.onreadystatechange=function(){

if(xhr.readyState==4&&xhr.status==200){

xhr.responseText;

}

}

4、发送请求:xhr.send();

练习：

创建一个页面06-ajax-exercise.html，创建一个按钮，点击按钮时，向 06-ajax-exercise.php 异步的发送一个请求，并将 响应回来的数据(欢迎光临) 显示在页面的一个div中

6、使用GET提交方式，发送请求数据

在请求地址后，拼接请求参数(查询字符串)

...

xhr.open("get","xx.php?name=value&name1=value1",true);

...

ex:

xhr.open("get","check.php?uname=zs&upwd=123",true);

在 check.php 中，如何获取 uname 和 upwd 值?

$uname=$\_REQUEST["uname"];

$upwd=$\_REQUEST["upwd"];

作业：

1、创建 hm1.html 页面，页面上提供一个文本框，用于接收用户输入的姓名，当失去焦点时，异步提交给php，验证用户名称是否存在，并显示在页面中

2、创建 hm2.html 页面，页面上提供文本框 和 密码框 分别表示用户名 和 密码，再提供一个按钮，点击按钮时，验证用户名和密码是否正确

以上操作，均使用 xuezi 数据库 以及 xz\_user表

### Day02 POST和JSON

1、使用 POST 方式发送请求

1、提交的请求数据

提交的请求数据需要按照指定的格式拼好，放在 send() 中传递

xhr.send("name1=value1 & name2=value2");

2、设置一个请求消息头

POST 提交方式，必须将 Content-Type 的请求消息头更改为 application/x-www-form-urlencoded

在发送请求之前：

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

2、JSON

1、什么是JSON

JavaScript Object Notation

JS 对象表示法,即以JS对象的方式来表现出来的字符串

2、JS对象

var fbb = [];

fbb["yuwen"] = 121;

fbb["shuxue"] = 73;

fbb["yingyu"] = 116;

var fbb = {

yuwen:121,

shuxue:73,

yingyu:116

}

console.log(fbb.yunwen);

练习：(5分钟)

1、创建一个对象 ， yangmi

2、包含三个属性

height , 值为 165

weight , 值为 50

gender , 职位 女

3、在控制台上分别将以上三个属性打印输出

3、JSON

1、JSON对象

语法：

1、用一对 {} 来表示一个对象

2、对象的属性名称，必须用 "" 引起来，值如果也是字符串，也必须用 "" 引起来

3、属性 与 值，用 : 连接

4、多对属性 与 值之间，用 , 分开

ex:

var ym = '{"height":175,"weight":50}';

2、JSON数组

1、普通数组表现 ：'["杨幂","范冰冰","凤姐"]'

2、对象的数组 :

'[

{

"height":170,

"weight":55,

"gender":"女"

},

{

"height":180,

"weight":60,

"gender":"女"

}

]'

3、JSON文件

创建一个文件，以 \*\*\*.json 作为后缀

该文件中的数据必须是符合JSON格式的字符串

4、将JSON字符串 转成 JS对象/数组

var p = '{"name":"sf.z","age":25}';

1、使用 eval()

var obj = eval("("+p+")");

2、使用JSON.parse() 来解析JSON字符串得到JS对象

var obj = JSON.parse(p);

练习：(10分钟)

1、创建一个 users.json 的文件，里面包含一个数组，数组中有3个对象，每个对象包含 uname，upwd，gender 属性，分别赋值

2、创建一个 getusers.html，使用异步的方式向users.json 发送一个请求，并按以下格式打印输出数据

用户名称：xxxx

用户密码：xxxx

用户性别：男

用户名称：xxxx

用户密码：xxxx

用户性别：男

用户名称：xxxx

用户密码：xxxx

用户性别：男

5、在Php中，可以直接将数组(一维，二维，关联) 转换成JSON格式的字符串

php中通过 : json\_encode() 将数组转换为JSON字符串

语法：$str = json\_encode($array);

注意：如果服务器端响应回来的数据时JSON格式的，需要增加响应消息头

header("Content-Type:application/json");

练习：(10分钟)

1、改版省市级联的操作

2、将省份信息直接以JSON格式的字符串进行响应

3、在浏览器中得到JSON格式的字符串后，转行成JS对象/数组，进行解析，拼<option>,再将数据显示在 <select>

4、以相同的方式改版 城市信息

### Day03 XML

1、XML

1、什么是XML

eXtensible Markup Language

可扩展的 标记 语言

XML的标记没有被预定义，需要自行定义

XML的宗旨是做数据传递的，而非数据显示

2、XML的语法规范

XML可以独立的保存成 \*\*\*.xml 文件，也可以以字符串的形式出现(服务器端生成)

1、XML的最顶端是XML的声明

<?xml version="1.0" encoding="utf-8" ?>

练习：

创建一个 student.xml 文件，并添加xml声明

2、XML的标记的语法

1、XML标记都是成对出现，所有的标记必须有显示的结束

<person /> 错误

<person> 杨幂 错误

<person>杨幂</person> 正确

2、XML标记是严格区分大小写，开始和结束必须一直

<person></Person> 错误

<Person></Person> 正确

<PeRsOn></PeRsOn> 正确

3、标记允许嵌套，注意嵌套顺序即可(与HTML一致)

<person>

<name>范冰冰</name>

<age>40</age>

</person>

4、每个标记都允许自定义属性，格式等同于HTML的，但属性值，必须使用引号 括起来

<person id="1001" gender="male"></person>

5、每个XML文档中，有且只有一个根元素

练习：

在 student.xml 基础上完成

1、创建一对根元素 studentList

2、在 studentList 根元素中 ，使用一对 student 元素 来表示一名学员的信息

3、在 student 中，再分别使用三个元素来表示 姓名(name),性别(gender),年龄(age),并将值写在标记中

4、创建3个学员的信息

3、使用 AJAX 请求 XML 文档

1、要遵循AJAX的请求步骤

1、创建 / 获取 xhr

2、创建请求

3、设置回调

4、发送请求

2、创建请求时，需要将请求地址更改为 xml 文件的地址

xhr.open("get","student.xml",true);

3、在回调函数中，处理响应数据

使用 xhr.responseXML 来获取响应的数据，返回的是 XML文档对象

练习：

在 student.xml 基础上，使用AJAX向 student.xml发送请求，并将响应结果(responseXML)打印在控制台上

4、解析XML文档内容

1、核心

elem.getElementsByTagName("标签名");

elem : 表示的是获取范围

返回值：返回一个包含指定元素们的"类数组";

...

var xmlDoc = xhr.responseXML;

var students = xmlDoc.getElementsByTagName("student");

2、获取元素中的文本

elem.innerHTML

练习：

上个练习的基础上，将 student.xml 中的内容进行解析，格式如下：

姓名：范冰冰

年龄：40

性别：女

姓名：杨幂

年龄：31

性别：女

姓名：罗玉凤

年龄：40

性别：女

5、在PHP中返回xml格式的字符串

1、必须增加响应消息头

header("Content-Type:application/xml");

2、按照XML的语法结构，拼XML的字符串，再响应给客户端

$xml="<?xml version='1.0' encoding='utf-8' ?>";

$xml.= "<studentList>";

...

$xml.= "</studentList>";

echo $xml;

练习：

1、在 03-php-xml.php 响应回以下格式的 xml 字符串

<?xml version="1.0" encoding="utf-8">

<list>

<stu>

<name>卡卡西</name>

<gender>男</gender>

<age>30</age>

</stu>

</list>

2、在 03-php-xml.html中，使用AJAX请求响应数据

姓名：卡卡西

性别：男

年龄：30

### Day04

1、使用 POST 方式发送请求

1、提交的请求数据

提交的请求数据需要按照指定的格式拼好，放在 send() 中传递

xhr.send("name1=value1 & name2=value2");

2、设置一个请求消息头

POST 提交方式，必须将 Content-Type 的请求消息头更改为 application/x-www-form-urlencoded

在发送请求之前：

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

2、JSON

1、什么是JSON

JavaScript Object Notation

JS 对象表示法,即以JS对象的方式来表现出来的字符串

2、JS对象

var fbb = [];

fbb["yuwen"] = 121;

fbb["shuxue"] = 73;

fbb["yingyu"] = 116;

var fbb = {

yuwen:121,

shuxue:73,

yingyu:116

}

console.log(fbb.yunwen);

练习：(5分钟)

1、创建一个对象 ， yangmi

2、包含三个属性

height , 值为 165

weight , 值为 50

gender , 职位 女

3、在控制台上分别将以上三个属性打印输出

3、JSON

1、JSON对象

语法：

1、用一对 {} 来表示一个对象

2、对象的属性名称，必须用 "" 引起来，值如果也是字符串，也必须用 "" 引起来

3、属性 与 值，用 : 连接

4、多对属性 与 值之间，用 , 分开

ex:

var ym = '{"height":175,"weight":50}';

2、JSON数组

1、普通数组表现 ：'["杨幂","范冰冰","凤姐"]'

2、对象的数组 :

'[

{

"height":170,

"weight":55,

"gender":"女"

},

{

"height":180,

"weight":60,

"gender":"女"

}

]'

3、JSON文件

创建一个文件，以 \*\*\*.json 作为后缀

该文件中的数据必须是符合JSON格式的字符串

4、将JSON字符串 转成 JS对象/数组

var p = '{"name":"sf.z","age":25}';

1、使用 eval()

var obj = eval("("+p+")");

2、使用JSON.parse() 来解析JSON字符串得到JS对象

var obj = JSON.parse(p);

练习：(10分钟)

1、创建一个 users.json 的文件，里面包含一个数组，数组中有3个对象，每个对象包含 uname，upwd，gender 属性，分别赋值

2、创建一个 getusers.html，使用异步的方式向users.json 发送一个请求，并按以下格式打印输出数据

用户名称：xxxx

用户密码：xxxx

用户性别：男

用户名称：xxxx

用户密码：xxxx

用户性别：男

用户名称：xxxx

用户密码：xxxx

用户性别：男

5、在Php中，可以直接将数组(一维，二维，关联) 转换成JSON格式的字符串

php中通过 : json\_encode() 将数组转换为JSON字符串

语法：$str = json\_encode($array);

注意：如果服务器端响应回来的数据时JSON格式的，需要增加响应消息头

header("Content-Type:application/json");

练习：(10分钟)

1、改版省市级联的操作

2、将省份信息直接以JSON格式的字符串进行响应

3、在浏览器中得到JSON格式的字符串后，转行成JS对象/数组，进行解析，拼<option>,再将数据显示在 <select>

4、以相同的方式改版 城市信息

## CSS

### Day01

1、CSS概述

1、问题

1、设置整个页面中的文字颜色为红色

<body text="red">

2、设置页面中所有的div元素的文本颜色为蓝色

3、更改页面中所有的div元素的文本颜色为粉色

html的弊端：

1、相同的样式设置，使用的是不同的属性来完成的

<body text="red">

<font color="blue">

2、相同的操作，要重复的写

<div>

<font color="pink">春眠不觉晓</font>

</div>

<div>

<font color="pink">夜来风雨声</font>

</div>

可重用性 和 可维护性 较低

2、什么是CSS

Cascading Style Sheets

层叠样式表 -> 样式表

作用：设置HTML网页元素的样子

CSS 与 HTML 之间的关系：

HTML : 构建网页的结构

CSS : 构建网页元素的样式

HTML 与 CSS 之间的使用原则：

W3C建议尽量使用CSS来取代HTML属性

2、CSS的语法规范

1、使用CSS样式表的方式

1、内联方式

又称为 "行内样式","内联样式"

特点：将CSS内容定义在HTML标记中

语法：

<ANY style="样式声明1;样式声明2;... ...">

样式声明：

1、由样式属性名 以及 值来组成

2、属性名 与 值 之间使用 : 做连接

3、多个样式声明之间用 ; 做分隔

常用的样式属性 和 值：

1、设置文本颜色的属性和值

属性：color

值：合法的颜色值(英文)

ex: color:red

2、设置背景颜色的属性和值

属性：background

值：合法的颜色值(英文)

3、设置文字大小的属性和值

属性：font-size

值：以 px 或 pt 为单位的数值

ex: font-size:16px

练习：(5分钟)

1、创建一个页面 02-css-basic.html

2、创建一个div标记，内容随意

3、设置该标记内的内容样式如下(内联方式)

1、文本颜色(color) ：黄色(yellow)

2、背景颜色(background) ：红色(red)

3、文字大小(font-size) ：16px

2、内部样式表

在网页中的一个独立的地方声明该网页要用到的所有的样式

语法：

<head>

<style>

/\* 注释 \*/

样式规则1

样式规则2

...

样式规则n

</style>

</head>

样式规则 ：由 选择器 和 样式声明 组成

选择器：规范了页面中哪些元素能够使用声明好的样式,其实就是为了匹配页面中的元素

样式规则 ：

选择器{样式声明1;样式声明2;}

选择器{

样式声明1;

样式声明2;

}

div{

color:red;

background:yellow;

font-size:16px;

}

p{

... ...

}

练习：

在 02-css-basic.html 中

1、创建若干 p 标记，内容随意

2、创建若干 h3标记，内容随意

3、使用内部样式表完成下列操作

1、页面中所有的p标记

文本颜色 ：蓝色(blue)

文本大小 ：23px

2、页面中所有的 h3 标记

文本颜色 ：灰色(silver)

背景颜色 ：橙色(orange)

3、外部样式表

独立于任何网页的位置处，声明一个样式表文件(\*\*\*.css)，在文件中声明样式。在使用的网页中进行引入

使用步骤：

1、创建样式表文件，并编写样式

在 xxx.css 文件中编写样式规则

2、在网页上对样式表文件进行引入

<head>

<link rel="stylesheet" href="xxx.css">

</head>

#### 2、CSS样式表特征

1、继承性

必须是父子(有层级关系)结构

大部分的css效果是可以直接继承给子元素的

2、层叠性

可以为一个元素定义多个样式规则，规则中的属性不冲突时，可以同时都应用到当前元素上

div{

color:red;

}

div{

background:yellow;

}

div{

font-size:26px;

}

3、优先级

在层叠性基础上，如果样式声明冲突(重复)的话,则按照默认的优先级去应用样式

由低 到 高

1、浏览器默认设置

2、样式表(内外部)

4、!important 原则

显示调整样式属性优先级

样式属性:值 !important;

!important 加在谁的后面，就以谁为主

#### 3、CSS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声明好的样式

为了匹配页面的元素

2、选择器详解

1、通用选择器

作用：可以与页面上的任一元素相匹配

语法：\* { 样式声明}

ex:

\*{

font-size:12px;

color:red;

}

2、元素选择器

作用：设置页面上某种元素的默认样式

语法：标记{ 样式声明 }

ex:

h3{ color:red;}

div{font-size:12px;}

section{background:yellow;}

3、类选择器

作用：定义页面上某个或某些元素的样式

特点：通过元素的class属性进行引用的选择器

语法：

1、声明

.类名{ 样式声明 }

注意：

1、点 不能省略

2、类名不能以数字开头

3、由英文，-,\_ 来组成

2、引用

<ANY class="类名">

特殊用法：

1、分类选择器的定义方式

将 类选择器 和 元素选择器 结合使用

语法：元素选择器.类选择器{ }

ex:

1、div.text{}

定义class为text的div元素的样式

2、span.important{ }

定义class为important的span元素..

2、多类选择器的引用方式

让元素引用多个类选择器

语法：

<ANY class="c1 c2 c3">

4、id选择器

作用：与ID属性联用，为了设置 指定ID元素的样式(专属定制)

语法：

#ID值{ ... }

练习：

1、增加一个 div ，id为container

2、设置 container 的样式为

文字大小：36px

背景：紫色(purple)

斜体: font-style : italic;

5、群组选择器

作用：将多个选择器放在一起进行样式的声明定义

语法：选择器1,选择器2,选择器3,... { ... }

ex:

#container,div.text,.redColor,span{

color:red;

}

等同于 ：

#container{color:red;}

div.text{color:red;}

.redColor{color:red;}

span{color:red;}

6、后代选择器

作用：通过元素的后代关系匹配元素

语法：选择器1 选择器2{}

7、子代选择器

作用：通过元素的子代(一层层级)关系匹配元素

语法：选择器1>选择器2{}

#### 8、伪类选择器

作用：匹配元素不同的状态的选择器

语法：

所有的伪类都是以 : 作为开始的

选择器:伪类选择器{ ... }

1、链接伪类

1、:link

匹配元素尚未被访问时候的状态

ex:

a:link{ ... }

2、:visited

匹配元素访问过的状态

ex:

a:visited{ ... }

2、动态伪类

1、:hover

匹配鼠标悬停在元素上时的效果

2、:active

匹配元素被激活时的状态

超链接,文本框 和 密码框，点击的时候，就是 active

3、:focus

匹配元素获取焦点时的状态

文本框 和 密码框

练习：

1、页面增加一个 a 标记，增加以下效果

1、未被访问时，颜色为 黑色，没有下划线

text-decoration:none;

2、鼠标悬停的时候，增加下划线

text-decoration:underline;

3、被激活时(点击的一瞬间)，文本为orange色

4、访问后的时候，颜色变为 绿色(green)

2、页面增加一个 input 文本框，默认效果颜色为 灰色，字体为斜体(font-style:italic)

1、被激活时，字体为非斜体

font-style:normal;

2、获取焦点时，文本颜色为绿色(green)

#### 3、尺寸 与 边框

1、单位

1、尺寸单位

1、px ：像素

1980 \* 1024

1024 \* 768

2、in ：英寸

1in = 2.54cm

3、pt ：磅 (1pt = 1/72in)

多数表示文字大小

4、cm ：厘米

5、mm ：毫米

6、em ：

7、rem ：

8、%

2、颜色单位(取值)

1、rgb(r,g,b);

r : red

g : green

b : blue

r,g,b : 取值范围 0-255

rgb(32,78,125);

2、rgb(r%,g%,b%)

3、rgba(r,g,b,alpha)

alpha : 透明度，0-1之间的小数，越大，不透明度越高

4、#rrggbb

由6位十六进制的数字表示一个颜色

#112233

5、#rgb

#rrggbb 的缩写

#112233 -> #123;

#1a2233 -> 无缩写

6、表示颜色的英文单词

red,green,blue,yellow,... ...

2、尺寸属性

1、作用

改变元素的宽度 和 高度

2、语法

1、宽度

width : 宽度

min-width : 最小宽度

max-width : 最大宽度

2、高度

height : 高度

min-height : 最小高度

max-height : 最大高度

3、页面中允许设置尺寸的元素

1、所有的块级元素都允许设置尺寸

div,p,h1,h2,...,ul,ol,dl,dt,dd, ...

2、行内块元素允许设置尺寸

表单控件元素(单选按钮，复选框)

3、本身具备 width 和 height 的HTML元素

img , table

注意：大部分的行内元素是无法修改尺寸的

4、溢出处理

当内容多，元素小的时候，就会产生溢出的效果，默认都是纵向溢出

属性：

overflow,overflow-x,overflow-y

取值：

1、visible

可见的，默认值，溢出可见

2、hidden

隐藏的，溢出的内容全部隐藏，不可见

3、scroll

显示滚动条，溢出时，可用

4、auto

自动，溢出时才显示滚动条并可用

### Day02 边框

1、边框属性

1、边框属性

1、简写方式

border:width style color;

width : 以 px 为单位的数值

style : 边框样式

solid : 实线边框

dotted : 虚线边框(点)

dashed : 虚线边框(线)

color : 边框颜色，也可以取值为 transparent

注意： border:none; / border:0; 取消边框

练习：

1、创建一个元素，id为container，尺寸为200px\*200px，并且设置 四个方向的边框为 1px 的宽度，实线，orange的颜色

2、单边定义

只设置某一条边的三个属性

属性：

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

练习：

设置上边框的效果 2px ，green ，dotted的样式

3、单属性设置

设置四条边框的某一个属性

属性：

border-width/style/color : 值;

练习：

将四个方向的样式，全部修改为 线状虚线(dashed)

4、单边单属性设置

只设置某一方向的某一属性值

属性：

border-方向-属性:值;

方向：top / right / bottom / left

属性：width / style / color

练习：

将下边框的颜色更改为紫色

2、边框倒角

将元素的四个角，由直角倒换成圆角

属性：

border-radius:值;

取值：以px为单位的具体值 / %

练习：(2分钟)

1、创建一个网页 - 03-border-radius.html

2、创建一个元素 id 为container，设置尺寸 200\*200

3、设置 container 的倒角半径为 5px ， 观察效果

4、将 container 的倒角半径 更改为 50% ，观察效果

5、将container 的宽度更改为 400px ，观察效果

3、边框阴影

属性：box-shadow

取值：h-shadow v-shadow blur spread color inset

h-shadow :

阴影的水平偏移距离，必须的

取值为正，阴影右偏移

取值为负，阴影左偏移

v-shadow :

阴影的垂直偏移距离，必须的

取值为正，阴影下偏移

取值为负，阴影上偏移

blur :

模糊距离，取值越大，模糊效果越明显，以 px 为单位的数值

spread :

阴影的大小，指定要在基础阴影上扩充出来的大小距离，取值为 px 为单位的数值

color : 阴影颜色 ... ...

inset : 将默认的外阴影改为内阴影

练习 ：(3分钟)

在作业的基础上增加以下效果

1、每幅图，增加 5px 的倒角

2、每幅图在鼠标悬停的时候，增加阴影效果，阴影向右下偏移5px，颜色为 #ddd

4、轮廓

轮廓指的是边框的边框,绘制于边框外围的一条线

属性：

1、简写方式

outline:width style color;

outline:none / outline:0; 为取消轮廓

练习：

在页面上创建一个文本框，尝试着取消轮廓，再观察效果

#### 2、框模型

1、什么是框模型

框 ：页面元素皆为框

框模型 ：Box Model，定义了元素框处理元素的内容，内边距，外边距 以及 边框的 一种计算方式

外边距：元素与元素之间的距离

内边距：元素边框与内容之间的距离

当框模型介入到元素后，元素的占地尺寸会发生变化:

元素的宽度 = 左右外边距+左右边框+左右内边距+w;

元素的高度 = 上下外边距+上下边框+上下内边距+h

2、外边距

1、什么是外边距

围绕在元素边框周围的空白距离，就是外边距

也能表示当前元素与其他元素之间的空间量

2、语法

属性：

margin : 定义某元素四个方向的外边距

margin-top / right / bottom / left

取值：

1、具体的数值，以 px 为单位的数字

练习：

1、创建一个页面05-margin.html

2、创建两个div元素，id分别为 d1 和 d2设置尺寸为 200 \* 200 ，并且设置边框

3、设置 第一个元素，左外边距 20 px，上外边距 20px ，下外边距20px

2、取值为负数

让元素向反方向移动

margin-left

取值为正，让元素右移动

取值为负，让元素左移动

margin-top

取值为正，让元素下移动

取值为负，让元素上移动

3、取值为 %

外边距的值，占父元素宽或高的占比是多少

4、取值为 auto

自动计算外边距的值

注意：auto只对块级的左右外边距有效

使用方式：

1、为块级元素设置宽度

2、再设置其左右外边距为 auto

表现：块级元素在父元素中水平居中显示

练习：

为 d2 元素 增加左右外边距的值为 auto

简洁写法：

1、margin:value;

四个方向的外边距都相同

2、margin:v1 v2;

v1 : 上下外边距

v2 : 左右外边距

3、margin:v1 v2 v3;

v1 : 上外边距

v2 : 左右外边距

v3 : 下外边距

4、margin:v1 v2 v3 v4;

上 右 下 左

ex:

maring:0 auto;

3、自带外边距的元素

h1~h6,p,ul,ol,dl,dd,pre,body

通过 CSS Reset(CSS重写) 的手段，来重置具备外边距的元素

body,h1,h2,h3,h4,h5,h6,p,ul,ol,dl,dd,pre{

margin:0;

}

4、外边距的特殊效果

1、外边距合并

当两个垂直外边距相遇时，他们将合并成一个，最终的距离取决于两个外边距中距离较大的那个

2、外边距的溢出

在某些特殊的条件下，为子元素设置上外边距时，有可能会作用到父元素上

1、父元素没有上边框

2、为第一个子元素设置上外边距时

练习：

1、创建06-margin-question.html

2、创建两个div元素 d1 和 d2 ，分别设置不同的背景色，尺寸 200\*200

3、在d2中，再添加一个子元素 div ，d3，设置尺寸 100\*100，以及背景色

4、设置 d3 的上外边距为 50px，观察结果

解决方案：

1、为父元素增加上边框

弊端：对父元素的高度有影响

2、使用父元素的上内边距来取代子元素的上外边距

弊端：对父元素的高度有影响

3、在父元素的第一个子元素位置处，增加一个空 <table></table>

3、行内元素 以及 行内块元素的垂直外边距

1、行内元素垂直外边距无效(img除外)

2、行内块元素，设置垂直外边距时，整行元素都跟着发生改变

3、内边距

1、什么是内边距

元素边框与内容之间的距离就是内边距

内边距会扩大元素边框占地区域

2、语法

属性：

padding:value

padding-top / right / bottom / left:value;

取值：

1、以px为单位的数值

2、以 %为单位的数值

简介写法：

1、padding:value

四个方向内边距值

2、padding:v1 v2;

v1 : 上下

v2 : 左右

3、padding:v1 v2 v3;

v1 : 上

v2 : 左右

v3 : 下

4、padding:v1 v2 v3 v4;

上 右 下 左

练习：

1、创建08-padding.html

2、创建两个元素，d1 和 d2 ，尺寸 200\*200，并设置边框 2px solid #f00，随意编写文字

3、为d1元素增加 上下左右四个方向的内边距各20px，观察效果

#### 4、属性：box-sizing

作用：指定框模型的计算方式

取值：

1、content-box

默认值，采用默认的方式计算元素的占地区域

实际占地宽度=左右边框+左右内边距+width;

实际占地高度=上下边框+上下内边距+height

2、border-box

元素的尺寸，会包含border以及padding的值

实际占地宽度=width(包含了border和padding)

实际占地高度=height(包含了border和padding)

3、背景属性

1、背景色

属性：background-color

取值：合法的颜色值

注意：背景颜色是从边框的位置处开始绘制

2、背景图片

属性：background-image

取值：url(图片的地址);

3、背景平铺

属性：background-repeat

取值：

1、repeat

默认值，横向纵向都平铺

2、repeat-x

只在横向平铺

3、repeat-y

只在纵向平铺

4、no-repeat

不平铺

4、背景图片尺寸

属性：background-size

取值：

1、width height

2、width% height%

3、cover

将背景图等比放大，直到背景图完全覆盖到元素的所有区域为止

4、contain

将背景图等比放大，直到背景图碰到元素的某一个边缘为止

5、背景图片固定

作用：将背景图固定在网页的某个位置处，一直在可视化区域中，不会随着滚动条而发生位置的变化

属性：background-attachment

取值：

1、scroll

默认值，滚动

2、fixed

固定

注意：为body设置背景图时，允许设置背景图片的固定

6、背景定位

作用：改变背景图在元素中的位置

属性：background-position

取值：

1、x y

x:

背景图水平偏移距离

取值为正，向右移动

取值为负，向左移动

y:

背景图垂直偏移距离

取值为正，向下移动

取值为负，向上移动

2、x% y%

0% 0% : 背景图在左上角

100% 100% : 背景图在右下角

100% 0% : 背景图在右上角

3、关键字

x : left / center / right

y : top / center / bottom

background-position:top left;

background-position:center; /\*即水平又垂直的居中\*/

7、背景的简写属性

在一个属性中指定背景的多个属性值

属性：background

取值：color url() repeat attachment position;

注意：如果不设置其中某个值的话，该位置将采用默认值

background:red;

background:url(a.jpg) no-repeat;

### Day03渐变,文本,表格,浮动

#### 1、渐变

1、什么是渐变

渐变指的是多种颜色平缓变化的一种显示效果

2、渐变的主要因素

色标 ：一种颜色 及其 出现的位置

一个渐变是由多个色标组成(至少两个)

3、渐变分类

1、线性渐变

以直线的方向来填充渐变色

2、径向渐变

以圆形的方式实现的填充效果

3、重复渐变

将线性渐变 或 径向渐变 重复几次实现出来

4、线性渐变

属性：background-image

取值：linear-gradient(angle,color-point1,color-point2,...);

1、angle

表示渐变填充的方向或角度

取值：

1、关键字

1、to top

从下向上填充渐变色

2、to right

从左向右填充渐变色

3、to bottom

从上向下填充渐变色

4、to left

从右向左填充渐变色

2、角度值

1、0deg

从下向上填充渐变色等同于 to top

2、90deg

从左向右填充渐变色等同于 to right

3、180deg

从上向下填充渐变色等同于 to bottom

4、270deg

从右向左填充渐变色等用于 to left

2、color-point

色标 ：颜色 及其 位置

取值 ：颜色 以及 位置的组合，中间用 空格 隔开

ex:

1、red 0%

在填充方向的开始位置处颜色为 红色

2、green 50%

到填充方向一半的位置处，颜色变为绿色

3、blue 200px

到填充方向的200px的位置处，颜色变为蓝色

background:linear-gradient(to top,red 0%,green 100%);

background:linear-gradient(45deg,red 0px,green 50px,blue 100%);

#### 5、浏览器兼容性

各浏览器的新版本都支持渐变属性

对于不支持的浏览器版本，可以通过增加浏览器前缀的方式，让浏览器支持渐变

Firefox : -moz-

Chrome & Safari : -webkit-

Opera : -o-

IE : -ms-

#container .btn-submit{

background:linear-gradient(0deg,red,blue);

background:-moz-linear-gradient(0deg,red,blue);

background:-webkit-linear-gradient(0deg,red,blue);

background:-o-linear-gradient(0deg,red,blue);

background:-ms-linear-gradient(0deg,red,blue);

}

#### 2、文本格式化属性

1、字体属性

1、指定字体

属性：font-family

取值：用 , 隔开的字体名称列表

ex:

font-family:"微软雅黑",Arial;

font-family:"microsoft yahei",Arial,"simhei","simsun"

2、字体大小

属性:font-size

取值:以px 或 pt 为单位的数字

3、字体加粗

属性:font-weight

取值:

1、bold ：加粗

2、normal ：正常

3、value

无单位的数字

400 等同于 normal

900 等同于 bold

ex:

font-weight:700;

4、字体样式

属性：font-style

取值：

1、normal ：正常显示

2、italic ：斜体显示

5、小型大写字母

将小写字符变为大写，但大小与小写字符一致

属性：font-variant

取值：

1、normal ：正常

2、small-caps

小型大写字符

练习：

创建一个网页，定义一个div标记，并设置id为 container，随便编译内容，中英文 结合的

设置以上文本的样式如下：

1、字体 ：微软雅黑

2、文字大小 ：24px

3、加粗显示文字

4、斜体显示文字

5、将所有的小写字符变为小型大写字符

6、字体属性

简写方式

属性：font

取值：style variant weight size family;

注意：

使用简写属性时，必须要设置family的值，否则无效

font:12px;//不对

font:12px "微软雅黑";//正确

2、文本属性

1、文本颜色

属性：color

取值：合法的颜色值

2、文本排列

作用：指定文本，行内，行内块元素的水平对齐方式

属性：text-align

取值：left / center / right / justify(两端对齐)

3、文字修饰

属性：text-decoration

取值：

1、none : 无任何的线条修饰

2、underline : 下划线修饰

3、line-through : 删除线

4、overline : 上划线

4、行高

作用：一行数据所占的高度

属性：line-height

取值：以 px 为单位的数值

注意：如果行高设置的比文字的高度高的话，那么该行文本将在指定行高范围内垂直居中显示

ex:

line-height:30px;

line-height:1.5;/\*行高是font-size的1.5倍\*/

5、首行文本缩进

属性：text-indent

取值：以px为单位的数值

6、文本阴影

属性：text-shadow

取值：h-shadow v-shadow blur color

练习：

在 02-font.html 基础上，增加效果

1、文本的排列方式为 两端对齐

2、文字修饰 为下划线修饰

3、行高设置为 字体大小的1.5倍的高度

4、首行文本缩进24px

5、文本阴影，水平 垂直偏移为0，模糊距离为1px 或 2px ，颜色为 红色

3、表格

1、表格的常用属性

1、边距属性 ：padding

2、边框属性 ：border

3、尺寸属性 ：width ，height

4、文本格式化属性 ：font-\*，text-\*，line-height

5、背景属性：颜色，图片，渐变

6、vertical-align

作用：指定单元格数据的垂直对齐方式

取值：

1、top ：上对齐

2、middle ：居中对齐

3、bottom ：下对齐

练习：

创建 05-table.html

1、创建一个 400 \* 400 的表格，4行4列

2、创建每个单元格的尺寸为 100\*100

3、设置表格 和 单元格的边框 为 1

px solid #000

4、尝试着为每个td增加上外边距 10px

2、表格的特用属性

1、边框合并

属性：border-collapse

取值：

1、separate

默认值，即分离边框模式

2、collapse

边框合并

2、边框边距

作用：设置单元格之间或单元格与表格之间的距离

属性：border-spacing

取值：

1、给定一个值 ：水平和垂直的间距相同

2、给定两个值 ：

第一个值 表示的是水平间距

第二个值 表示的是垂直间距

注意：只有在分离边框模式下，边框边距才有效，即border-collapse:separate时，border-spacing才有效

3、标题位置

属性：caption-side

取值：

1、top ：默认值，标题在表格内容之上

2、bottom ：标题在表格内容之下

4、显示规则

作用：用来帮助浏览器指定如何布局一张表，也就是指定 td 尺寸的计算方式

属性：table-layout

取值：

1、auto

默认值，即自动表格布局，列的尺寸实际上是由内容来决定的

2、fixed

固定表格布局，列的尺寸由设定的值为准的

自动表格布局 VS 固定表格布局

1、自动表格布局

1、单元格的大小会适应内容

2、表格复杂时，加载速度较慢(缺点)

3、自动表格布局会比较灵活(优点)

4、适用于不确定每列大小时使用

2、固定表格布局

1、单元格的尺寸取决于设定的值

2、任何情况下都会加速显示表格(优点)

3、固定表格布局不够灵活(缺点)

4、适用于确定每列大小时使用

#### 4、定位 - 浮动定位

1、定位

定位，指的是改变元素在页面中默认的位置

2、定位的分类

按照定位的效果，可以分成以下几类：

1、普通流定位(默认定位方式)

2、浮动定位

3、相对定位

4、绝对定位

5、固定定位

3、定位 - 普通流定位

普通流定位，又称为"文档流定位",页面中元素们的默认定位方式

1、每个元素在页面中都有自己的空间

2、每个元素默认都是在其父元素的左上角开始出现

3、页面中的块级元素都是按照从上到下的方式逐个排列，每个元素独占一行

4、页面中的行内元素 以及 行内块元素都是按照从左到右的方式来排列的

问题：如何能够让多个块级元素在一行内显示呢???

4、浮动定位

1、什么是浮动 & 特点

如果将元素的定位方式设置为浮动定位，name元素将具备以下特征：

1、元素将被排除在文档流之外(脱离文档流)，不再占据页面的空间，其它未浮动元素要上前补位

2、浮动元素会停靠在其父元素的左边或右边，或其它已浮动元素的边缘上

3、浮动元素只能在当前行浮动

4、浮动定位解决的问题：让多个块级元素在一行内显示

2、语法

属性：float

取值：

1、none

默认值，即无任何浮动效果

2、left

左浮动，让元素停靠在父元素的左边，或挨着左侧已有的浮动元素

3、right

右浮动，让元素停靠在父元素的右边，或挨着右侧已有的浮动元素

练习：

1、创建 07-float.html

2、创建三个元素，id分别为 d1,d2,d3,尺寸200\*200 ,背景色分别为 red,green,blue

3、依次让 d1，d2，d3实现右浮动，观察效果

4、将步骤3注销，依次让d1，d2，d3实现左浮动，观察效果

3、浮动引发的特殊效果

1、当父元素显示不下所有已浮动子元素时，最后一个将换行，但是，有可能被卡住

2、元素一旦浮动起来后，宽度将以内容为主(未指定宽度情况下)

3、元素一旦浮动起来后，将变为块级元素

允许修改尺寸

能正常处理垂直方向外边距

4、文本，行内元素，行内块元素是采用环绕的方式来排列的，是不会被浮动元素压在底下的，而会巧妙的避开浮动元素

### Day04浮动,显示

1、浮动

#### 1、清除浮动所带来的影响

元素一旦浮动起来之后，就会对后续的元素带来一定的位置影响(后续元素要上前占位)，如果后续元素不想被影响的话(不想占位)，那么就可以通过清除浮动的方式来解决

属性：clear

取值：

1、none

默认值，不做任何的清除浮动操作

2、left

清除当前元素前面元素左浮动所带来的影响

3、right

清除当前元素前面元素右浮动所带来的影响

4、both

清除当前元素前面元素任何一种浮动所带来的影响

2、浮动元素对父元素高度带来的影响

元素的高度，都是以未浮动元素的高度为准的。浮动元素是不占高度的

解决父元素高度方案如下：

1、直接设置父元素高度

弊端：不是每次都知道父元素的高度

2、设置父元素也浮动

弊端：不是任何时候父元素都需要浮动，而且浮动会影响后续元素

3、为父元素设置 overflow

取值：hidden 或 auto

弊端：如果有内容需要溢出显示的话，也会一同被隐藏

4、在父元素中，追加空子元素，并设置其clear:both

5、... ...

#### 2、显示

1、显示方式

属性： display

取值：

1、none

让生成的元素不显示 - 隐藏

特点：

脱离文档流

2、block

让元素表现的和块级元素一样

特点：

1、独占一行

2、允许修改尺寸

3、允许正常处理垂直方向的外边距

3、inline

让元素表现的和行内元素一样

特点：

1、多个元素一行内显示

2、不允许修改尺寸

3、垂直外边距无效

4、inline-block

让元素表现的和行内块一样

特点：

1、多个元素能在一行内显示

2、允许修改尺寸

5、table

让元素表现的和表格元素一样

特点：

1、每行只显示一个

2、尺寸以内容为准

... ...

练习：

1、创建一个页面，创建5个超链接

2、分别先设置5个超链接的尺寸为100\*100

3、分别设置5个超链接的显示方式位

1、inline

2、block

3、inline-block

4、none

5、table

2、显示效果

1、显示 / 隐藏属性

属性：visibility

取值：

1、visible : 默认值，可见的

2、hidden : 隐藏,依然占据空间

2、透明度

属性：opacity

取值：0.0(完全透明) ~ 1.0(完全不透明) 之间的数字

练习：

1、在页面上创建一个div，在div中再创建一个h1 （2组）

2、在第一组的div 和 h1 中，设置 div的背景色为 rgba(255,0,0,0.3)

3、在第二组的div 和 h1 中，设置 透明度(opacity)为0.3

3、垂直对齐方式

属性：vertical-align

作用：设置 td 和 img 的垂直对齐方式

取值：

1、设置在 td 上的 值

1、top

2、middle

3、bottom

2、设置在img上的值

设置图片两边的文本，相对于图片的垂直对齐

1、top

2、middle

3、bottom

4、baseline ：基线对齐

常用操作：

在开发网页过程中，通常会先将所有的图片(img)的vertical-align属性修改为除baseline以外的其他值

3、光标

作用：当鼠标悬停在元素上时表现的形式

属性：cursor

取值：

1、default

2、pointer ：小手

3、crosshair ：+

4、text ： I

5、wait ：等待

6、help ：帮助

练习：

1、页面上创建一个div，尺寸100\*100

2、鼠标悬停上的时候，将光标该为 小手的形状

#### 3、列表

1、列表项标识

属性：list-style-type

取值：

1、none

2、disc

3、circle

4、square

... ...

2、列表项图片

作用：以自定义的图像来作为列表项的标识

属性：list-style-image

取值：url(图像地址);

3、列表项位置

属性：list-style-position

取值：

1、outside

将标识放置与li的外面

2、inside

将标识放置与li的里面

4、列表属性

属性：list-style

取值：style image position;

常用用法：

list-style:none;

#### 4、定位

1、定位属性

1、定位方式属性

属性：position

取值：static / relative / absolute / fixed

relative : 相对的

absolute : 绝对的

fixed : 固定的

注意：将 position修改为relative / absolute / fixed 中的任何一种的话，那么元素就被称之为"已定位元素"

2、偏移属性

属性 ：top / right / bottom / left

取值 ：以 px 为单位的数值

2、定位方式

#### 1、相对定位

1、什么是相对定位

元素会相对于它原来的位置偏移某个距离

多数会实现在位置微调时使用

2、语法

position:relative;

配合着top/right/bottom/left实现位置的微调

练习：

1、页面中创建一个div，尺寸为100\*30，设置边框或背景色

2、当鼠标悬停在元素上时，让元素向左上角移动5px(相对定位)

#### 2、绝对定位

1、什么是绝对定位 & 特点

1、绝对定位的的元素会脱离文档流，不占页面空间

2、绝对定位的元素会相对于离它最近的，已定位的，祖先元素 去实现位置的初始化和偏移

3、如果不存在已定位的祖先元素的话，那么就相对于body去实现位置的初始化和偏移

2、语法

position:absolute;

配合着偏移属性top/right/bottom/left实现位置的偏移

3、特点

1、绝对定位元素会脱离文档流，所以会压在其它元素之上

2、绝对定位的元素会变为块级元素

3、绝对定位的元素margin可以正常处理，但margin:0 auto 会失效

### Day05定位,选择器,弹性布局

1、定位

1、堆叠顺序

一旦将元素变为已定位元素的话，元素们则有可能出现堆叠的效果

属性：z-index

取值：无单位的数字，数字越大越靠上

注意：

1、只有已定位的元素才能使用z-index

2、父子元素间，z-index 无效，永远都是子压在父上

2、固定定位

1、什么是固定定位

将元素固定在页面的某个位置处，位置不会随着滚动条而发生改变。固定在可视化的区域中。

2、语法

position:fixed;

配合着 top / right / bottom / left 改变位置

3、注意

1、固定定位的元素会脱离文档流-不占页面空间

2、固定定位的元素会变成块级元素

3、固定定位的元素永远都是相对于body去实现位置的初始化和偏移

#### 2、复杂选择器

1、兄弟选择器

作用：获取某元素平级的后面的兄弟元素

<div>

<p id="p1"></p>

<p id="p2"></p>

<p id="p3"></p>

</div>

注意：兄弟选择器，只能向后找，不能向前找

1、相邻兄弟选择器

作用：获取紧紧挨在某元素后的兄弟元素

语法：选择器1+选择器2{ ... }

#p1+#p2{} //p2

#p1+p{}

2、通用兄弟选择器

作用：获取某元素后的兄弟元素

语法：选择器1~选择器2{ ... }

#p1~p{ ... }

练习：

<h3>静夜思</h3>

<div>床前明月光</div>

<div>疑是地上霜</div>

<div>举头望明月</div>

<div>低头思故乡</div>

通过 兄弟选择器设置如下操作：

1、h3 与 div之间有 30px 的外边距

2、从第2个div开始，每个div有10px的上外边距

2、属性选择器

<input id="uname" type="text" name="uname" placeholder="Plear input ...">

1、作用

通过元素所附带的属性及其值来匹配页面的元素

2、语法

1、基本语法：[attr]

作用：匹配附带 attr 属性的元素

ex:

1、[id] : 匹配页面中所有附带id属性的元素

2、[class]:匹配页面中所有附带class属性的元素

2、elem[attr]

elem : 表示任意元素

attr : 表示任意属性

作用：匹配页面中附带 attr 属性的 elem 元素

ex:

1、匹配页面中所有附带了id属性的div元素

div[id]

2、匹配页面中所有附带type属性的input元素

input[type]

3、.important[id]

3、[attr1][attr2]...

作用：匹配同时附带多个 属性的 元素

ex:

1、匹配页面中即有id又有class的div元素

div[id][class] { ... }

4、[attr=value]

作用：匹配 attr 属性的值为 value 的元素

ex:

1、匹配页面中所有的文本框

input[type=text]{ ... }

input[type='text']{ ... }

input[type="text"]{ ... }

#### 3、伪类选择器

1、目标伪类

作用：突出显示活动的HTML锚点元素，匹配到被激活的锚点

语法：

:target{ ... }

2、结构伪类

1、:first-child

匹配的元素是属于其父元素中的首个子元素

2、:last-child

匹配的元素是属于其父元素中的最后一个子元素

<style>

div:first-child{

...

}

div:last-child{

}

</style>

<header>

<div></div>

<div></div>

<div></div>

<div></div>

</header>

练习：

1、创建一个 4\*4 的表格，并填充内容

2、设置第一行的数据背景颜色为 黄色

3、设置最后一行的数据背景颜色为 红色

3、:nth-child(n)

匹配的元素是属于其父元素中的第n个子元素

练习：

将第二行中的第三列的背景颜色设置为 orange

4、:empty

匹配没有子元素的元素

5、:only-child

匹配的元素是属于其父元素中的唯一子元素

3、否定伪类

将满足指定选择器的元素给排除出去

:not(selector)

#tbl tr:not(:first-child){}

4、伪元素选择器

1、:first-letter 或 ::first-letter

作用：匹配某元素中的首字符

ex :

p:first-letter{

color:red;

font-weight:bold;

}

2、:first-line 或 ::first-line

作用：匹配某元素中的首行

ex:

p:first-line{

color:blue;

}

3、::selection

作用：匹配被用户选取的部分内容

注意：只能修改文本颜色 和 背景颜色

练习：

1、创建一个div元素，id为d1，并且通过lorem生成几段文本

2、将d1中的首字符样式设置为 加粗的字体，红色的文本颜色，文字大小为20pt，右外边距15px

3、将d1中的首行文本设置为，文字颜色为 orange色，斜体的字体样式

4、将用户选取的部分内容设置为 红色背景，黄色文字

3、内容生成

1、什么是内容生成

使用CSS动态的向某元素中插入一部分内容

2、伪元素选择器

1、:before 或 ::before

作用：匹配到某元素的内容区域之前

<p>(:before)举头望明月</p>

2、:after 或 ::after

作用：匹配到某元素的内容区域之后

<p>低头思故乡(:after)</p>

3、属性

属性：content

作用：向某位置处增加内容

取值：

1、字符串 ：用 "" 引起来的普通文本

2、url() ：要生成一副图像显示在指定的位置处

解决的问题：

1、浮动元素父元素高度的问题

.clear:after{

content:"";

display:block;

clear:both;

}

2、子元素外边距溢出的问题

#d2:before{

content:"";

display:table;

}

#### 4、弹性布局(Flexible Layout)

1、什么是弹性布局

弹性布局(Flexible Layout),是一种布局方式，是一种解决某元素中"子元素"的布局方式

2、flex的容器

将元素变为flex容器后，那么所有的子元素都将成为flex项目，都允许按照弹性布局的方式来排列

如何将元素变为 flex 的容器 ？？？

属性：display

取值：

1、flex:将块级元素变为弹性布局的容器

2、inline-flex:将行内元素变为弹性布局的容器

注意：

1、元素设置为 display:flex 之后，子元素的float,clear,vertical-align 将失效

2、子元素的尺寸是允许被修改的

3、弹性布局容器的text-align属性会失效

3、属性

1、容器的属性

1、flex-direction

作用：指定容器的主轴及其排列方向

主轴：项目排列的一根轴，如果横轴作为主轴的话，那么项目们则横向排列，如果纵轴作为主轴的话，那么项目们则纵向排列

交叉轴：与主轴交叉的轴,如果横轴为主轴，那么纵轴则为交叉轴，如果纵轴为主轴，那么横轴则为交叉轴

取值：

1、row

默认值，即主轴为 "横轴"，起点在左端

2、row-reverse

主轴为横轴，起点在右端

3、column

主轴为纵轴，起点在顶端

4、column-reverse

主轴为纵轴，起点在底端

2、flex-wrap

作用:一根主轴排列不下所有的项目时,如何换行

取值：

1、nowrap ：默认值，即不换行，项目缩小

2、wrap ：换行

3、wrap-reverse

3、justify-content

作用:定义项目在主轴上的对齐方式

取值：

1、flex-start : 在主轴的起点对齐

2、flex-end : 在主轴的终点对齐

3、center : 在主轴上居中对齐

4、space-between

两端对齐,第一个项目挨着起点，最后一个项目挨着终点，中间间隔平分

5、space-around

4、align-items

作用：项目在交叉轴上如何对齐

取值：

1、flex-start ：在交叉轴的起点对齐

2、flex-end ：在交叉轴的终点对齐

3、center ：在交叉轴上居中对齐

4、baseline ：在交叉轴上基线对齐

5、stretch ：如果项目未设置高度的话，将沾满容器的整个高度

2、项目的属性

该组属性只能设置在某一项目上，只控制某一项目，不影响容器以及其他项目

1、flex-grow

作用：定义项目的放大比例。如果容器有足够的多余空间的话，项目将如何放大

取值：

整数数字，默认为0，即不放大

取值越大，占据的剩余空间也就越多

### Day06转换,过渡,动画

#### 1、CSS Hack

1、解决问题

浏览器兼容性的问题

针对不同的浏览器编写不同的CSS代码

2、CSSHack的原理

使用CSS样式的优先级来解决兼容性问题的

3、CSSHack的实现方式

1、CSS类内部Hack

在样式属性名称前或属性值后增加前后缀，以便去识别不同的浏览器

+ ： IE6,7的前缀

- ： IE6的前缀

\0 ：IE8，9，10

\9\0：IE9，10

2、CSS选择器Hack

在选择器前增加前缀以便识别不同的浏览器

\*前缀：识别IE6

\*+前缀：识别IE7

\*div{ IE6中div显示效果 }

\*+div{ IE7中div显示效果 }

3、HTML头部引用Hack

使用IE条件注释来判断IE浏览器的版本，从而去执行不同的代码

语法：

<!--[if 条件 IE 版本号]>

满足条件要执行的HTML代码

<![endif]-->

版本号：6-10

条件：

1、gt

只有在大于指定版本的浏览器中执行指定代码

2、lt

只有在小于指定版本的浏览器中执行指定代码

3、gte

greater than and equal

只有在大于等于指定版本的浏览器中执行指定代码

4、lte

less than and equal

只有在小于等于指定版本的浏览器中执行指定代码

5、!

在条件版本以外的浏览器中执行指定的代码

#### 2、转换

1、什么是转换

改变元素在页面中的位置，大小，角度以及形状的一种方式

2D转换 ：只在x轴，y轴，平面上发生转换效果

3D转换 ：增加在 z轴上的转换效果

2、转换属性

转换属性：

属性：transform

取值：

1、none ：默认值，即无任何转换效果

2、transform-function

表示的是1个或多个转换函数，如果是多个转换函数(效果)的话，中间用 空格 隔开

转换原点：

属性：transform-origin

取值：数值 / 百分比 / 关键字

2个值：原点在x轴 和 y轴的位置

3个值：原点在x，y，z轴上的位置

3、2D转换

1、位移

1、什么是位移

改变元素在页面中的位置

2、语法

属性：transform

取值：

1、translate(x)

指定元素在x轴上的偏移距离

取值为正，元素右移

取值为负，元素左移

2、translate(x,y)

x ：同上

y ：指定元素在y轴上的偏移距离

取值为正，元素下移

取值为负，元素上移

3、translateX(x)

4、translateY(y)

2、缩放

1、什么是缩放

改变元素在页面中的尺寸(按比例)

2、语法

属性：transform

取值：

1、scale(value)

value : 横向 和 纵向的缩放比例

2、scale(x,y)

x : 横向的缩放比例

y : 纵向的缩放比例

3、单向缩放函数

scaleX(x)

scaleY(y)

缩放比例：

默认值为 1

缩小：0-1之间的数字

放大：大于1的数字

3、旋转

1、什么是旋转

改变元素在页面上的角度

2、语法

属性：transform

取值：

rotate(ndeg);

n取值为正，顺时针旋转

n取值为负，逆时针旋转

3、注意

1、转换原点会影响最后的转换效果

练习：

1、将转换原点更改为元素的左上角

2、再将元素旋转45deg，观看效果

2、旋转是连同坐标轴都一同旋转的，会影响旋转后的位移效果

练习：

1、将d2元素先旋转45deg，再向右位移50px

2、将d2元素先向右位移50px，再旋转45deg

4、倾斜

1、什么是倾斜

改变元素在页面中的形状

2、语法

属性：transform

取值：

1、skewX(xdeg)

让元素向着x轴的方向产生倾斜效果，实际上是改变y轴的倾斜角度值

x 取值为正，y轴逆时针倾斜

x 取值为负，y轴顺时针倾斜

2、skewY(ydeg)

让元素向着y轴的方向产生倾斜效果，实际上是改变x轴的倾斜角度制

y 取值为正，x轴顺时针倾斜

y 取值为负，x轴逆时针倾斜

3、skew(x)

等同于 skewX()

4、skew(x,y)

练习：

1、页面中创建一个元素200\*200的，增加边框或背景

2、鼠标悬停的时候，该元素向右偏移200px，向下偏移200px，旋转135deg，向x轴倾斜45deg，同时放大1.1倍

4、3D转换

1、透视距离

模拟人眼到3d转换元素之间的距离

属性：perspective

注意：该属性要加在3d转换元素的父元素上。

2、3D旋转

属性：transform

取值：

1、rotateX(xdeg)

沿x轴旋转的角度

2、rotateY(ydeg)

沿y轴旋转的角度

3、rotateZ(zdeg)

沿z轴旋转的角度

4、rotate3d(x,y,z,ndeg)

x,y,z 取值为大于0的数字时，则表示该轴要参与到旋转中来，如果取值为0，该轴则不参与旋转

练习：

使用rotate3d，x,y,z同时旋转

#### 3、过渡

1、什么是过渡

将CSS的属性值在一段时间内平缓变化的过程给体现出来

2、过渡的语法 - 4属性

1、指定过渡属性

作用：指定哪个属性值，在变化时需要使用过渡效果来体现

属性：transition-property

取值：使用过渡效果的属性名称

ex:

transition-property:background;

注意：

允许设置过渡效果的属性如下：

1、颜色属性

2、取值为数字的属性

3、转换属性 : transform

4、渐变属性

5、阴影属性

6、visibility属性

2、指定过渡时长

作用：指定在多长时间内完成过渡的操作

属性：transition-duration

取值：以 s 或 ms 为单位的数值

1000ms = 1s

3、指定过渡速度时间函数

属性：transition-timing-function

取值：

1、ease ：默认值，慢速开始，快速变快，慢速结束

2、linear ：匀速

3、ease-in ：慢速开始，加速结束

4、ease-out ：快速开始，慢速结束

5、ease-in-out : 慢速开始和结束，中间先加速后减速

4、指定过渡延迟时间

属性：transition-delay

取值：以 s 或 ms 为单位的数值

3、过渡的编写位置

1、允许将过渡属性编写在元素声明的样式中

2、允许将过渡属性编写在触发过渡效果样式中(hover)

4、过渡的简介写法

transition:property duration timing-function delay;

transition:all .5s linear;

练习：

1、在页面中创建1个 100\*100的元素

2、鼠标悬停时，5秒钟之内，向右偏移500px的同时再旋转3圈

3、鼠标移出去的时候，还能滚回来

#### 4、动画

1、什么是动画

使元素从一种样式逐渐变化为另一种样式

动画是复杂的过渡效果

动画是通过 "关键帧" ， 来控制动画的每一步

关键帧：

1、动画执行的时间点

2、在该时间点上元素的样式

2、动画的使用步骤

1、声明动画

2、为元素调用动画

3、声明动画

@keyframes 动画名称{

/\*定义该动画中所有的关键帧\*/

0%{

/\*动画开始时的样式\*/

}

25%{

/\*动画执行到 1/4 的时候的样式\*/

}

50%{

/\*动画执行到 1/2 的时候的样式\*/

}

100%{

/\*动画结束时的样式\*/

}

}

#### 兼容性

@-moz-keyframes 动画名称{

/\*定义该动画中所有的关键帧\*/

0%{

/\*动画开始时的样式\*/

}

25%{

/\*动画执行到 1/4 的时候的样式\*/

}

50%{

/\*动画执行到 1/2 的时候的样式\*/

}

100%{

/\*动画结束时的样式\*/

}

}

@-webkit-keyframes 动画名称{

/\*定义该动画中所有的关键帧\*/

0%{

/\*动画开始时的样式\*/

}

25%{

/\*动画执行到 1/4 的时候的样式\*/

}

50%{

/\*动画执行到 1/2 的时候的样式\*/

}

100%{

/\*动画结束时的样式\*/

}

}

@-o-keyframes 动画名称{

/\*定义该动画中所有的关键帧\*/

0%{

/\*动画开始时的样式\*/

}

25%{

/\*动画执行到 1/4 的时候的样式\*/

}

50%{

/\*动画执行到 1/2 的时候的样式\*/

}

100%{

/\*动画结束时的样式\*/

}

}

@-ms-keyframes 动画名称{

/\*定义该动画中所有的关键帧\*/

0%{

/\*动画开始时的样式\*/

}

25%{

/\*动画执行到 1/4 的时候的样式\*/

}

50%{

/\*动画执行到 1/2 的时候的样式\*/

}

100%{

/\*动画结束时的样式\*/

}

}

#### 4、动画属性

1、animation-name

作用：指定调用动画的名称

2、animation-duration

作用：指定动画执行一个周期的时长

取值：以 s 或 ms 为单位的数值

3、animation-timing-function

作用：指定动画的速度时间函数

取值：ease,linear,ease-in,ease-out,ease-in-out

4、animation-delay

作用：指定动画的延迟时长

取值：以 s 或 ms 为单位的数值

5、animation-iteration-count

作用：指定动画的播放次数

取值：

1、具体数值

2、infinite ：无限次

6、animation-direction

作用：指定动画的播放方向

取值：

1、normal ：正常播放

2、alternate ：轮流播放

奇数次数播放时，正向播放(0%~100%)

偶数次数播放时，逆向播放(100%~0%)

7、动画的简介写法

animation:name duration timing-function delay iteration-count direction;

8、animation-fill-mode

作用：规定动画在播放前或播放后的状态

取值：

1、none ：不改变默认行为

2、forwards ：当动画完成后，保持在最后一个帧的状态上

3、backwards ：在动画播放前，延迟时间内，动画将保持在第一个帧的状态上

4、both

forwards 和 backwards 的结合体

9、animation-play-state

作用：指定动画的播放状态

取值：

1、paused : 动画暂停

2、running : 动画播放

5、动画的调用时机

1、在 :hover 时调用动画

只有在鼠标悬停在元素上时，执行动画

2、在元素声明的样式中，直接调用动画

网页加载时，就执行动画

练习：

在网页加载时，调用 scroll 动画，执行一个周期的时长是3s，匀速执行，并且反复执行

# 张东

## JavaScript Core

### Day01 String,API

zhangdong@tedu.cn

三样东西:

书: 犀牛书——JavaScript权威指南

微信公众号: 前端大全

上届笔记和思维导图: 预习！禁止抄笔记! 上课时，打开跟着看！

正课:

String:

1. 什么是

#### 2. API:

大小写转换

获取指定位置的字符

选取子字符串

查找关键词

1. 什么是: 由一串字符组成的只读字符数组

vs 数组:

相同: 1. [i], 2. .length, 3. for循环, 4. slice()

不同: 类型不同！API不通用！

2. API:

公共特点: 所有API都无权修改原字符串，只能返回新字符串

大小写转换:

str.toUpperCase() 都转为大写字母

str.toLowerCase() 都转为小写字母

何时: 只要不区分大小写时，都要先转为统一的大小写，再比较和判断

比如: 验证码; 用户名; 邮箱;

php中大小写转换:

说明: php中字符串API的路径:

函数参考->文本处理->字符串->字符串函数->

strtolower(str)

strtoupper(str)

验证码:

1. 服务器端生成验证码,保存在session中一个变量里

2. 服务器端将验证码生成图片, 返回给客户端:

已经实现了: 01\_code.php

3. 客户端用img src="xxx.php":

4. 输入验证码, 提交到服务器端，

将输入的验证码和session中的验证码比对

验证是否正确

5. 服务器端返回验证结果:

### Day02 String API,正则

正课:

String

1. 获得指定位置的字符

2. 获取子字符串

3. \*\*\*查找关键词

4. \*替换

5. \*切割

\*\*\*\*正则表达式

#### String

1. 获得指定位置的字符: ——了解

str.charAt(i) => str[i]

php: substr($str, i, 1) 获得$str中i位置的1个字符

获得指定位置字符的unicode号:

str.charCodeAt(i)

将unicode号反向转回字:

String.fromCharCode(unicode号)

2. 获取子字符串:

str.substring(starti,endi+1) => str.slice(starti,endi+1)

substring vs slice

相同: 1. 含头不含尾

2. 省略第二个参数，表示一直选取到结尾

不同: 不支持负数参数！

变通: 用str.length-n 代替 -n

str.substr(starti,n) 根据个数，获取子字符串

php: substr($str, i, n) 获得$str中i位置开始的n个字符

简写: i支持负数参数

省略n表示到结尾

3. 查找关键词: 4种:

1. 查找一个固定的关键词出现的位置:

var i=str.indexOf("关键词",开始位置)

在str中，从“开始位置”开始，查找下一个"关键词"的位置

返回值: 下一个"关键词"的第一个字的下标位置

如果找不到了，返回-1

如果省略第二个参数-开始位置: 默认从0开始找

查找最后一个关键词的位置:

var i=str.lastIndexOf("关键词")

问题: 1. 区分大小写!

解决: 在indexOf之前，先将str转为小写，再将关键词也转为小写，再查找

问题: 2. 只能查找一个固定的关键词!

解决: 正则表达式

php: strpos -> indexOf

stripos 忽略大小写版本的indexOf

strrpos -> lastIndexOf

strripos 忽略大小写的lastIndexOf版本

$i=stripos(完整字符串,关键词[,开始位置])

2. str.search():

什么是: search是用正则表达式查找str中第一个匹配的关键词的位置

何时: 只要判断一句话中\*是否包含\*符合正则要求的敏感词时

如何: var i=str.search(/正则表达式/);

返回值: 返回第一个找到的匹配的关键词的位置

如果没找到返回-1

变通: 验证: 只要将正则表达式，前加^后加$，search也可做验证

问题: 1. 永远只能找第一个

2. 只能返回位置，无法返回关键词内容

3. str.match():

什么是: 查找str中所有和正则表达式匹配的关键词的内容

何时: 只要获得敏感词的\*内容\*时

如何: var arr=str.match(/正则表达式/)

返回值: 包含所有找到的敏感词内容的数组

如果找不到: 返回null!

强调: 如果一个函数有可能返回null，就必须先验证不是null，再使用!

问题: 1. 默认正则是区分大小写的！

解决: 在第二个/后加i, 意为忽略: ignore

2. 默认正则只匹配第一个敏感词！

解决: 在第二个/后加g, 意为全部: global

缺: 只能获得每个关键词的内容，无法获得每个关键词的位置

4. 即查找每个关键词的内容，又查找每个关键词的位置:

String做不了!

reg.exec()

4. 替换:

什么是: 将找到的敏感词替换为指定的内容

如何: 2种:

1. 简单替换: 将所有敏感词都替换为统一的替换值

str=str.replace(/正则表达式/,"替换值")

强调: replace无权直接修改原字符串，只能返回新字符串，所以，必须用变量接住替换后的新内容。

2. 高级替换: 根据每个敏感词的不同，动态选择对应的替换值

str=str.replace(/正则表达式/,function(kword){

//kword: 自动获得本次找到的敏感词

return 根据kword的不同，选择不同的替换值返回;

})

衍生: 删除: 将找到的敏感词，替换为“”

5. 切割:

什么是: 将字符串，按某种子字符串，分隔为多段

如何: 2种:

1. 简单切割: 切割符是一个固定不变的字符串

var substrs=str.split("切割符");

强调: 切割后的结果中，不包含切割符

2. 复杂切割: 切割符可能有多种情况

var substrs=str.split(/正则表达式/)

#### \*\*\*\*正则表达式: Regular Expression

什么是: 规定字符串中，字符出现规律的表达式

何时: 2种:

1. 用规则，模糊匹配多种敏感词时

2. 用规则验证字符串的格式: 密码强度，邮箱格式，日期。。。

如何: 语法:

1. 最简单的正则就是关键词原文本身

2. 字符集:

什么是: 1位字符上的备选字符的集合

何时: 只要一位字符上有多种可能的备选字时

如何: [备选字符列表]

强调: 1. 不能加逗号分隔

2. 每个[]，只能匹配一位字符

简写: 如果字符集中部分字符的unicode号是连续的，可用-省略中间字符

常用: [0-9] 一位数字

[a-z] 一位小写字母

[A-Z] 一位大写字母

[A-Za-z] 一位字母，大小写都行

[0-9A-Za-z] 一位字母或数字

[\u4e00-\u9fa5] 一位汉字

车牌号: [\u4e00-\u9fa5] [A-Z] [A-Z0-9] [A-Z0-9] [A-Z0-9] [A-Z0-9] [A-Z0-9]

除了: [^74]

匹配范围太大！

比如匹配HTML元素的开始标签: <a[^>]+>

3. 预定义字符集:

什么是: 对常用字符集的简写: 固定4种:

包括: \d 1位数字

\w 1位字母，数字或\_

\s 1位空字符: 空格，tab，...

. 一切字符

强调: 只有规则和预定义字符集的规则完全匹配时，才能使用预定义字符集

4. 量词:

什么是: 专门规定一位字符集出现的次数

何时: 今后只要规定字符集的出现次数时，必须用量词

如何: 字符集量词

强调: 量词默认只修饰相邻的前一个字符集

包括: 2大类:

1. 有明确数量:

{n,m} 至少n个，最多m个

{n,} 至少n个，多了不限

{n} 必须n个，不能多也不能少

2. 没有明确数量:

? 可有可无，最多1个

\* 可有可无，多了不限

+ 至少1个，多了不限

5. 选择和分组:

选择: 等效于程序中的或

何时: 只要在两组规则间任意匹配其一即可

如何: 规则1|规则2

分组: 将多个子规则用()，包裹为一组

何时: 2种:

1. 只要一个量词需要同时修饰多个子规则，必须将多个子规则包裹为一组

2. 需要获得关键词中的部分子内容时

如何: (规则1...)

比如:

身份证号:

15位数字: \d{15}

2位数字: \d{2}

1位数字或X,x : [0-9Xx]

后三位整体: 可有可无，最多一次

手机号:

+86或0086 (\+86|0086)

至少一个空字符 \s+

地区代码和空字符整体 可有可无，最多1次

1

必须在3,4,5,7,8之间选一个 [34578]

9位数字 \d{9}

((\+86|0086)\s+)?1[34578]\d{9}

"微信" "微 信" "weixin" "wei xin" "w x" "wechat"

(微|w(ei)?)\s\*(信|x(in)?)

6. 匹配特殊位置: 3个位置:

开头 ^

结尾 $

何时: 只要要求匹配的关键词必须出现在开头或结尾时

比如: 只匹配开头的空字符: ^\s+

只匹配结尾的空字符: \s+$

即匹配开头，又匹配结尾: ^\s+|\s+$

正则表达式的问题:

如果验证时，只要部分匹配，就会返回true！

解决:

只要验证，必须同时前加^后加$，表示必须从头到尾完整匹配。

单词边界\b:

何时: 专门用于匹配一个独立的单词

包含: 空格，开头，结尾，标点符号

### Day03正则,Math

补充:

1. 正则:

分组编号: 正则表达式中每个()，都会自动获得一个分组编号，从1开始，连续递增，不重复。

何时: 只要希望获得关键词的局部子字符串时

如何使用:

1. 在正则表达式内使用:

\n 匹配和第n个分组相同的内容

2. 在js函数中使用:

$n可自动获得第n个()分组匹配的子内容

正课:

1. 内置对象和包装类型

2. \*\*\*RegExp

3. Math

4. Date

#### 1. 内置对象/类型:

什么是: ES标准中规定的，浏览器厂商已经实现的对象

为什么: 简化开发！

鄙视: 包括: 11个

String Number Boolean —— 包装类型

Array RegExp Math Date

Error

Function Object

Global: 是全局作用域对象: 在浏览器中被window代替

鄙视: 谈谈对包装类型的理解

什么是包装类型: 保存一个原始类型的值

提供操作原始类型值的API

为什么: 原始类型的值本身不具有任何功能

比如: var price=345.678; price.toFixed(2);

var unicode="张".charCodeAt();

何时: 只要试图对原始类型的值调用函数时，都会自动创建包装类型对象

如何: 自动使用:

过程:

试图对原始类型的值调用任何函数时:

1. 判断原始类型的类型名

2. 自动创建对应的包装类型对象,其中保存要操作的原始类型值

3. 调用包装类型对象，提起定义好的API

4. API调用后，包装类型对象自动释放！

鄙视: var str="hello";

str.len=10; //向str上添加一个新的属性，值为10

console.log(str.len);//输出str上的len属性值:?

原理: str.len=10; -> new String(str).len=10;

//执行后，new String()没人要，释放了！

console.log(str.len); -> console.log(new String(str).len);

另一种情况:

var str=new String("Hello");

str.money=10;

//因为new String()始终被str变量引用，不释放！

console.log(str.money);//10

#### 2. \*\*\*RegExp:

什么是: 保存一个正则表达式

提供使用正则表达式执行验证和查找的API

何时: 2种:

1. 验证:

2. 查找: 第四种查找方式: 即查找内容，又查找位置

如何:

创建: 2种:

1. 创建一个固定不变的正则表达式:

var reg=/正则/ig;

何时: 如果正则表达式是固定不变的

2. 动态生成正则表达式:

var reg=new RegExp("正则","ig");

何时: 如果正则表达式不是固定，需要动态生成，就必须用new RegExp创建

API: 2个:

1. 验证:

var bool=reg.test(str) 验证str的格式是否符合reg的要求

问题: 默认只要部分匹配就返回true

解决: 只要验证，必须前加^，后加$

php中: preg\_match()

2. 查找: 补齐第四种查找方式：即查找内容，又查找位置

var arr=reg.exec(str);

在str中查找下\*一个\*符合reg要求的关键词的\*内容和位置\*

返回值: [

0: 关键词完整内容,

1: 第1个分组的子内容,

index:关键词的位置

]

如果找不到返回null

强调: 不用手动指定fromi，也可自动找下一个

因为: 每次都自动修改reg.lastIndex=index+关键词.length

php中: preg\_match\_all(

"正则",

"完整字符串",

保存查找结果的数组,

PREG\_OFFSET\_CAPTURE //包含关键词位置

)

#### 3. Math:

什么是: 保存数学计算的常量，并提供数学计算常用API的对象

何时: 只要进行专门的数学计算时

如何:

创建: 不能new！

所有属性和API，直接通过大写Math调用！

包括:

（PHP中: 数学扩展->Math）

1. 取整: 3种:

上取整: 只要超过，就取下一个整数

Math.ceil(num)

下取整: 舍弃小数部分

Math.floor(num) 不能去单位

vs parseInt(str) 比floor聪明，能去单位——首选

四舍五入取整:

Math.round(num) vs num.toFixed(2)

缺: 只能取整 优: 可指定小数位数

优: 返回number 缺: 返回字符串

可直接用于计算 直接计算可能出错！

自定义round: 即可定义小数位数，又返回number

2. 乘方和开平方:

乘方: Math.pow(底数,幂)

比如: 10的2次方: Math.pow(10,2)

开平方: Math.sqrt(num)

3. 最大值，最小值:

Math.max(值1,值2,...) 获得多个值中的最大值

Math.min(值1,值2,...) 获得多个值中的最小值

问题: Math.max和min不支持从数组中获得最大值和最小值。

解决: Math.max/min(...arr)

...可打散数组为单个元素

4. 随机数:

Math.random() 随机生成0~1之间的一个随机小数

公式: 在任意min~max之间生成一个随机整数

parseInt(Math.random()\*(max-min+1)+min)

### day04日期Date,Err,function

正课:

1. Date

2. Error

3. \*\*\*\*Function

创建; 重载; 匿名函数;

#### 1. Date

什么是: 保存一个日期或时间，提供操作日期和时间的API

何时: 只要存储或操作日期和时间

如何: （PHP中:日期与时间相关扩展->日期和时间->Date/Time函数）

创建: 4种:

1. 获得系统当前日期和时间:

var now=new Date();

强调: 只能获得客户端本地时间，无法获得服务器时间

2. 自定义日期和时间:

var now=new Date("yyyy/MM/dd hh:mm:ss");

问题1: 不同编程语言中存储日期的格式不一致!

问题2: 字符串不便于计算和大小比较

问题3: 文字会受时区影响

解决: 不用字，也不用数组/对象来表示时间

用毫秒数

Date对象的本质: 其中保存了1970年1月1日至今的毫秒数

为什么: 1. 便于计算和比较大小

2. 数字在所有编程语言中都统一支持

3. 数字不受时区影响: 因为数字是一个时间段

3. 将ms转化为客户端日期对象:

var date=new Date(ms);

\*\*\*总结: 今后，在网络间传输时间或在服务器存储时间，都用整数（毫秒数/秒数）

4. 复制一个日期对象:

为什么复制: 日期计算默认都直接修改原日期对象

何时复制: 如果希望同时保留计算前后的开始时间和结束时间时，就要先复制开始时间，再用副本计算结束时间

如何: var date2=new Date(date1);

API: 3句话

1. 单位: FullYear Month Date Day ——都没有s结尾

Hours Minutes Seconds Milliseconds ——都有s结尾

2. 每个单位都有一对儿getXXX()/setXXX()方法

其中:

getXXX()负责获取单位的数值

比如: now.getFullYear() 2017

now.getHours() 11

setXXX(n) 负责修改指定单位的数值

比如: now.setFullYear(2018)

now.setDate(30)

福利: setXXX(n)

其中:n可以是任意值!

setXXX(n)可自动调整时间进制!

特例: day没有setDay()! 因为day是推算出来的

3. 取值范围:

Month: 0~11

强调: js中的月份比现实中小1

读取或修改时，可能需要+1或-1修正

Date: 1~31

Day: 0~6 日~六

Hours: 0~23

Minutes/Seconds: 0~59

日期计算: 2种情况:

1. 两日期对象可直接相减: 结果是毫秒差!

2. 对任意单位做加减:

3步:

1. 先取值: var d=now.getDate();

2. 做加减: d+=30;

3. 放回去: now.setDate(d); //setXXX()可自动调整时间进制

问题: setXXX() 直接修改原日期对象

解决: 先复制副本，再用副本计算

简写: now.setDate(now.getDate()+30);

将日期转为字符串:

date.toString(); 转化为当地时间的完整格式

date.toLocaleStirng(); 转化为当地时间的简化版格式

date.toLocaleDateString(); 简化版格式，仅保留日期部分

date.toLocaleTimeString(); 简化版格式，仅保留时间部分

date.toGMTString(); 转化为国际标准时间

自定义格式化日期函数

#### 2. Error:

什么是错误(bug): 程序执行过程中，导致程序无法正常执行的异常情况。

错误处理: 即使程序出错，也保证程序不会强行退出的机制

为什么: 只要程序出错，都会强行中断退出

何时: 只要希望程序即使出错，也不中断退出

如何:

try{

可能出错的正确代码

}catch(err){//err: 错误对象

只有发生错误时，才执行的错误处理代码:

3件事: 1. 提示用户, 2. 保存进度, 3. 记录日志

}

错误对象: 发生错误时，自动创建的，保存错误信息的对象

问题: 放在try catch中的代码，执行效率可能降低，且生成错误对象，占用内存空间

解决: 如果可以提前预知错误，则首选if...else...来提前预防错误。

错误类型: 6种

SyntaxError: 语法错误

ReferenceError: 引用错误, 要用的变量没找到

TypeError: 类型错误: 错误的使用了对象的方法或()或[]

RangeError: 范围错误, 参数超范围

URIError: URI错误, 如果请求所使用的url拼接错误

EvalError: Eval错误

抛出错误:

何时: 函数的作者向函数的调用者抛出自定义错误对象，告知调用者错误的使用了函数。

如何: throw new Error("自定义错误提示信息")

#### 3. \*\*\*\*Function

创建: 3种:

1. 声明: function 函数名(参数列表){ 函数体; return 返回值}

问题: 会被声明提前！

解决:

2. 赋值:

var函数名=function (参数列表){ 函数体; return 返回值}

优势: 不会被声明提前

揭示: 函数名其实仅是一个普通变量

函数其实是一个保存代码段的引用类型对象

函数名通过地址引用函数对象

3. 用new:

var 函数名=new Function("参数1","参数2",...,"函数体;...")

强烈不推荐使用:

鄙视: 以下创建函数正确的是:

function cmp(a,b){return a-b;}

var cmp=function(a,b){return a-b;}

var cmp=new Function(a,b,"return a-b");

var cmp=new Function("a","b","return a-b");

重载(overload)

什么是: 相同函数名，不同参数列表的多个函数，在调用时，可根据传入参数的不同，自动选择对应的函数执行

为什么: 减少API的数量，减轻调用者的负担

何时: 一件事，只是根据参数的不同，执行不同的逻辑时

如何: js语法默认不支持重载

为什么: js不允许多个同名函数同时存储在，最后创建的会覆盖之前创建的所有

变通: arguments

arguments:

什么是: 函数调用时自动创建的

接收所有传入函数的参数值的

类数组对象

类数组对象: 长的像数组的对象

vs 数组: 相同: 1. 下标, 2. .length, 3. for遍历

不同: 类型不同! API不通用!

何时: 只要传入函数的参数不确定时

是否还需要参数? 需要：

1. 参数名简单易用，且见名知义

2. 参数可以指导调用者正确使用函数

匿名函数:

什么是: 创建函数时，不指定函数名的函数

为什么:

1. 节约内存

2. 划分临时作用域

何时: 只要一个函数使用后，希望立刻释放时

如何:

1. 回调(callback): 将一个函数，交给另一个函数去自动调用。

比如: arr.sort(function(a,b){return a-b;})

xhr.onreadystatechange=function(resText){}

btn.onclick=function(){}

str.replace(/正则/,function(kw){return xxx});

2. 自调: 定义函数后，立刻调用自己

何时: 几乎所有自定义的脚本，都要放在匿名函数中

为什么: 避免使用全局变量，避免全局污染

如何:

1. (function(...){...})()

2. +function(...){...}()

### Day05func作用域,链,闭包

正课:

1. \*\*\*\*Function

\*\*\*\*作用域链

\*\*\*\*\*闭包

2. \*\*\*面向对象OOP

封装，继承，多态

1. \*\*\*\*Function

#### 作用域(scope)和作用域链:

什么是: 用途: 一个变量的可用范围

本质: 其实是一个保存变量的对象

为什么: 避免不同范围的变量间互相干扰

包括: 2级:

1. 全局作用域: window

保存全局变量: 优: 可重复使用，随处可用

缺: 全局污染

2. 函数作用域: 临时创建的活动对象AO

保存局部变量: 优: 不污染全局

缺: 不可重用，仅函数内可用

强调: 局部变量包含2中:

1. 在函数内var的

2. 参数变量也是局部变量

原理: 四个阶段:

1. 程序开始执行前:

在内存中创建执行环境栈(数组):

执行环境栈: 依次保存正在调用的函数的数组

函数调用后，会出栈

在栈中首先添加浏览器主函数的调用

创建全局作用域对象window

主函数引用全局作用域对象window

2. 定义函数时:

在window中用函数名创建变量

在window外创建函数对象保存函数的内容

函数名变量通过地址引用函数对象

函数对象用隐藏的scope属性，引用回自己诞生的作用域对象。——万一运行时缺东西，可以找诞生的作用域要！

3. 函数调用时:

在执行环境栈中添加当前函数调用

为本次函数调用创建活动对象AO:

活动对象: 专门保存本次函数调用所需的局部变量的对象——就是函数作用域对象

在活动对象中保存函数的局部变量

活动对象通过隐藏的parent属性引用window

变量的使用顺序: 先用AO中的局部变量，局部没有，才去window找。

4. 函数调用后:

函数的调用从ECS中出栈

导致: 函数作用域对象AO释放

导致: 局部变量一同释放！

作用域链: 由多级作用域对象，逐级引用形成的链式结构

作用: 存储所有变量(局部变量和全局变量)

控制着变量的使用顺序，先局部再全局

#### \*\*\*\*\*闭包:

什么是: 即重用一个变量，又保护变量不被篡改的一种机制

为什么:

全局变量: 优: 可重用

缺: 造成全局污染，易被篡改

局部变量:

优: 仅函数内可用，不会污染全局，也不会被篡改

缺: 不可重用

何时: 只要希望重用一个变量，又保护变量不被污染时

如何: 3步: 鉴别是否形成闭包的特征

1. 用外层函数包裹，要保护的变量和使用变量的内层函数

2. 外层函数返回内层函数

3. 调用者，调用外层函数，获得内层函数对象

缺: 闭包比普通函数占用更多内存！

解决: 用完后立刻释放！

getNum=null 导致: 内层小函数对象释放

导致: outer的AO释放

鄙视:

1. 先找受保护的变量和内层函数，画简图:

2. 同一次外层函数调用，返回的多个内层函数，共享同一个闭包变量

3. 多次外层函数调用，会创建多个闭包。多个闭包之间彼此毫无关系！

#### 2. \*\*\*面向对象OOP

什么是对象: 描述现实中一个事物的属性和功能的程序结构

同时存储一个事物的多个属性和功能的存储空间

什么是面向对象: 程序中，都是用对象来描述现实中一个具体事物

为什么: 现实中每个属性和功能，必须依附在一个具体对象上才有意义！

何时: 只要管理多个事物,都必须用对象管理！

如何: 三大特点: 封装，继承，多态

#### 封装:

什么是: 将一个事物的属性和功能集中定义在一个对象中

为什么: 1. 符合现实的习惯

2. 便于维护每个对象各自的属性和功能

何时: 只要使用面向对象的思想，第一步，都是封装对象

如何: 如何创建对象: 3种

1. 用{}直接量:

var obj={

属性名:值,

属性名:值,

... : ...

方法名(){//ES6

... ...

}

}

问题: 对象自己的方法如何动态访问自己的属性值？

解决:

错误1: 直接使用属性名!

报错: 找不到属性名变量

原因: 所有不加点的变量，只能在作用域链中找。无法自动进入对象中。

错误2: 用对象名.属性名

问题: 不便于维护! 对象名仅是普通变量，随时可能改名。

正确: \*\*\*\* this.属性名

什么是this: AO中自动定义的一个关键词

自动指向正在调用函数的.前的对象

为什么: 不受对象名的影响

何时: 只要对象自己的方法，要访问对象自己的属性，必须用this.属性名

补充: 事件处理函数中，也可用this自动获得触发事件的.前的元素对象本身。

访问对象中的成员: 属性和方法

属性: 其实就是保存在对象中的普通变量

比如: obj.属性名

方法: 其实就是保存在对象中的普通函数

比如: obj.方法名()

2. 用new创建: 2步:

var obj=new Object();//{}

obj.属性名=值;

obj.方法名=function(){ ... this ... };

揭示本质: js中一切对象底层，其实都是关联数组

对象其实就是关联数组的简化用法！

问题: 如果反复创建多个相同结构的对象时，代码繁琐，不便于维护！

解决: 构造函数

3. 用构造函数反复创建多个相同结构的对象:

什么是构造函数:

1. 定义同一类型的所有对象的统一结构

2. 将对象构造为想要的结构！

何时: 只要反复创建同一类型的多个对象时，都用构造函数

如何: 2步:

1. 定义构造函数:

function 类型名(属性参数列表){

this.属性名=属性参数;

this.方法名=function(){

... this ...

}

}

2. 用构造函数反复创建多个对象:

var obj=new 类型名(属性值列表)

new: 4件事:

1. 创建新的空对象:

2. ?

3. 调用构造函数，向对象中添加新属性

new自动修改this为正在创建的新对象

4. 返回新对象地址，保存到变量中

问题: 每个对象都创建一个方法的副本，浪费内存！

解决:

### Day06面向对象,ES5

正课:

1. \*\*\*面向对象:

继承, 多态

2. \*\*\*\*ES5

1. \*\*\*面向对象:

问题: 放在构造函数中的方法定义，会被反复创建副本，浪费内存！

解决: 继承

#### 继承

继承: 父对象的成员，子对象无需重复创建，就可直接使用

为什么: 代码重用, 节约内存！

何时: 只要多个子对象，需要相同的属性值或功能时

如何: js中的继承都是自动继承原型对象

原型对象:

什么是: 存储同一类型的所有子对象共有成员的父对象

何时: 只要多个子对象需要相同的属性值或功能时，都应该集中定义在原型对象中

如何使用: 2步:

1. 创建: 自动创建！买一赠一！

只要创建一个构造函数，都附赠一个空的原型对象

2. 继承: 自动继承！

只要使用构造函数创建一个子对象时，都会让子对象自动继承构造函数的原型对象——new的第2 步

如何向原型对象中添加共有成员:

类型名.prototype.新成员=值/function(){...}

成员的访问顺序:

优先使用自有成员

自己没有，才去父级原型对象中查找

自有属性和共有属性:

自有属性: 保存在对象本地，归当前对象独有的属性

共有属性: 保存在公共的父级原型对象中，所有子对象共享的属性

获取时: 没有差别: 子对象.属性名

修改时: 自有属性: 子对象.属性名=值

共有属性必须通过原型对象修改:

类型名.prototype.共有属性名=值

判断是否自有属性:

var bool=obj.hasOwnProperty("属性名")

内置类型的原型对象:

其实每种类型，都有两部分组成:

构造函数: 专门负责创建子对象

构造函数的原型对象: 专门负责保存所有子对象共有的属性和API

问题: 旧浏览器不支持新的API或浏览器中的API不够用！

解决: 为浏览器添加想要的API

如何: 2步:

1. 先判断当前浏览器是否支持该API:

其实就是判断当前浏览器的指定类型的原型对象中是否包含想用的API

如何: if(typeof 类型名.prototype.API !="function")

2. 向指定类型的原型对象中添加API

类型名.prototype.API=function(参数列表){

... ...

}

原型链:

什么是: 由多级父元素逐级继承，形成的链式结构

作用: 存储必须用.才能访问的所有对象可用的成员

控制成员的使用顺序和使用范围

顺序: 先自有，再共有

使用范围: 原型链中越靠上的成员，共享范围越大

顶级父类型: Object, 原型对象: Object.prototype

放在Object.prototype中的成员，所有对象都可使用

比如: toString() getOwnProperty()

vs 作用域链: 存储所有不用.就可直接访问的变量

控制着变量的使用顺序: 先局部，后全局

顶级作用域是: window——全局作用域对象

鄙视: 判断一个对象是不是数组类型！共几种方式!

错误: typeof: 只能区分原始类型和函数

无法进一步细致区分对象的类型名

正确:

1. 判断爹(原型对象)

var bool=父对象.isPrototypeOf(子对象)

2. 判断妈(构造函数)

obj.constructor===构造函数

问题: constructor是隐藏属性，不推荐使用:

变通: obj instanceof 构造函数

强调: 以上两种方式都不仅检查直接父类型，且检查整个原型链。

3. 验DNA:

其实,每个对象内都隐藏着一个class属性

class属性保存了对象创建时的最初类型

不随继承关系改变而改变

错误1: obj.class 拿不到，返回undefined

唯一正确办法: 只有Object.prototype中最原始toString() 才能输出class

返回值: [object class属性值]

Object

Array

Date

错误2: 使用子对象.toString()，容易被父对象中的toString()重写

解决: 用call强行调用！

Object.prototype.toString.call(obj)

//在运行时临时相当于obj.toString()

4. isArray()

专门判断任何一个对象是不是Array类型

如何: var bool=Array.isArray(obj)

其实内部使用的就是第三种方式——严格的验证！

鄙视: 何时将函数定义在原型对象中，何时将函数直接定义在构造函数上

答: 如果只允许指定类型的子对象才能使用的函数，必须放原型对象中，继承使用

如果希望不限制类型，所有对象都能使用的函数，可以直接放在构造函数上

比如: sort() push() isArray()

Array.prototype.sort()

Array.prototpye.push()

Array.isArray()

#### 多态: 同一个函数在不同情况下，表现出不同状态

2种:

1. 重载overload:

2. 重写override:

什么是: 如果子对象觉得父对象成员不好用！可在子对象本地定义同名成员，覆盖父对象成员。

为什么: 因为从父对象继承来的成员不一定都是好用的！

何时: 只要从父对象继承来的成员不好用，就可以重写自己的!

如何: 只要在子对象本地定义同名成员

原理: 成员的使用顺序: 先自有，再共有

自定义继承关系:

1. 只修改一个对象的父对象:

child.\_\_proto\_\_=father;

问题: \_\_proto\_\_是内部隐藏属性

解决: Object.setPrototypeOf(child,father)

2. 修改所有子对象的父对象:

其实就是修改构造函数的原型对象:

构造函数.prototype=father;

3. 两种类型间的继承:

问题: 两种类型间拥有部分相同的属性结构和方法定义

解决: 定义抽象父类型

如何: 3步:

1. 定义抽象父类型:

父类型构造函数: 保存所有子类型中共有的属性结构

父类型原型对象: 保存所有子类型中共有的方法定义

2. 在子类型原型对象中借用父类型构造函数

错误: 直接调用!

原因: 如果不用new , 直接调用构造函数, 其中this默认都指window！！！

正确: 只要函数调用时，this不是想要的，都用call替换！

父类型构造函数.call(正确的this,参数......)

3. 让子类型原型对象继承父类型原型对象

### Day07 ES5

正课:

ES5:

1. 严格模式:

2. \*\*\*对对象的保护

数据属性, 访问器属性, 防篡改

3. Object.create()

4. \*\*call/apply/bind

5. 数组API

#### 1. 严格模式:

什么是: 比传统js运行机制要求更严格的模式

为什么: js本身有很多广受诟病的缺陷

何时: 今后所有项目的开发，都必须在严格模式下进行

如何: 2个范围:

1. 在整个js文件或script标签内启用严格模式:

在js文件或script标签内第一行: "use strict";

何时: 新项目，都要启用严格模式

2. 仅在单个函数内启用严格模式:

在函数内的第一行:"use strict";

何时: 旧项目改造时，要逐个功能向严格模式迁移

要求:

1. 禁止给未声明的变量赋值:

2. 将静默失败升级为错误:

静默失败: 即修改不成功，又不报错！

3. 不建议使用arguments, arguments.callee

4. 普通函数调用和匿名函数自调中的this不再指向window！而是undefined

2. 对对象的保护:

为什么: js中普通的对象对自己的属性和结构，没有任何验证和保护的办法。

何时: 今后所有对象都要对自己的属性提供一定保护

如何:

1. 保护属性

ES5规定: 对象的属性分为2大类:

命名属性: 可用.直接访问到的属性

又细分为2类:

1. 数据属性: 直接存储属性值的属性

如何保护: 每个数据属性不再是一个简单的变量

而是一个拥有四大特定的小对象。

查看一个属性的四大特性:

Object.getOwnPropertyDescriptor(obj,"属性名")

返回值: {

value: 实际存储属性值,

writable: true/false, 控制只读

enumerable: true/false, 控制能否被for in遍历到

但是，用.依然可以访问到！

configurable: true/false, 控制:

1. 控制能否删除该属性

2. 控制能否修改其它特性

一旦该为false，不可逆！

总是伴随其它特性的修改，充当双保险

}

如何修改四大特性保护数据属性:

Object.defineProperty(obj,"属性名",{

要修改的特性:值,

... : ...,

})

问题: 一次只能修改一个属性的四大特性

解决: 同时修改多个属性的四大特性:

Object.defineProperties(obj, {

属性名:{

四大特性:值,

... : ...

},

属性名:{

四大特性:值,

... : ...

},

...

})

说明: 如果要修改的属性不存在，则自动创建

但是, 自动创建的属性，四大特性默认值都为false

问题: 只能用固定的三种特性保护属性，无法用自定义规则灵活保护属性

2. 访问器属性:

什么是: 不直接存储属性值，仅提供对其它数据属性的保护。

何时: 只要用自定义规则保护属性时

如何定义: 2步:

1. 定义隐藏的数据属性,起别名，实际存储数据

问题: 使用enumerable隐藏的属性，防for in，但防不住. 别人可用eric.\_age绕过访问器属性，直接篡改受保护的属性。

解决: 使用局部变量，代替隐藏的数据属性!

2. 定义访问器属性，从受保护的数据属性中读取或修改数据

Object.defineProperties(eric,{

age:{//访问器属性

get(){return \_age},

set(val){

if(val>=18&&val<=65)

\_age=val;

else

throw new RangeError("年龄必须介于18~65之间");

},

enumberable:true,

configurable:true

},

})

如何使用: 访问器属性的用法和普通属性完全一样！

只不过: 在试图获取属性值时，自动调用get

在试图修改属性值时，自动调用set

参数val自动获得等号右边的新值

内部属性: 不能用.访问的属性: class \_ \_proto\_ \_

#### 2. 防篡改:(保护结构)

3个级别:

1. 防扩展: 禁止向对象中添加新属性:

如何: Object.preventExtensions(obj)

原理: 每个对象内都有一个隐藏的内部属性extensible

默认true

preventExtensions将extensible改为false

2. 密封: 即禁止扩展，又禁止删除！

如何: Object.seal(obj)

原理: 即修改对象的extensible为false

又自动修改每个属性的configurable为false！

3.冻结：

如何：Object.freeze(obj)

原理：在密封的基础上，又将所有属性的writable：false；

#### 3. Object.create()

什么是: 仅基于一个现有父对象，创建他的子对象，并为子对象扩展新的自有属性

何时: 今后，如果只基于父对象，也想创建子对象时

如何: var child=Object.create(father,{//Object.defineProperties

属性名:{四大特性}

... : ...

});

#### 4. call/apply/bind:

相同: 如果函数中的this不是想要的，都可替换

不同:

call/apply: 强行\*调用\*一个函数，并\*临时\*替换函数中的this为指定对象

call: 要求传入bi函数的参数必须单独传入

apply: 要求传入函数的参数必须放在数组中整体传入

apply可先打散数组参数为单个元素，再传参

bind: 基于一个现有函数，\*创建\*一个一模一样的新函数，并\*永久\*绑定this为指定对象

总结: 如果是临时\*调用\*一个函数,用call/apply

如果"创建"一个新函数，永久绑定this时

所有回调函数中的this，要想替换都用bind

因为回调函数不是立刻执行，且不止执行一次！

#### 5. 数组API:

indexOf()已学

1. 判断: ——了解

every(): 判断数组中是否所有元素\*都\*符合要求

some(): 判断数组中是否包含符合要求的元素

如何: 仅以every()

var bool=arr.every(function(elem,i,arr){

//elem自动获得当前元素值

//i 自动获得当前位置

//arr 自动获得当前数组->this

return 针对当前元素执行的判断条件

})

执行过程: every会遍历每个元素，反复调用回调函数，只有所有函数调用都返回true，结果才返回true

2. 遍历API: 对数组中每个元素执行相同的操作

forEach(): 对原数组中每个元素执行相同操作

map(): 依次取出原数组中每个元素，执行相同操作后，放入新数组中返回。

如何:

arr.forEach(function(elem,i,arr){

arr[i]=新值;

});

arr.map(function(elem,i,arr){

return 新值;

})//return的新值会放入新数组中相同位置

3. 过滤和汇总:

过滤: 选取出原数组中符合条件的元素，组成新数组返回

var subs=arr.filter(function(elem,i,arr){

return 判断条件

}); //如果当前判断为true，则放入新数组返回

汇总: 将数组中每个元素统计汇总出一个最终的结果

var result=arr.reduce(function(prev,elem,i,arr){

//prev: 截止目前的临时汇总值

return prev+elem;

},base)

以base值为基数，累加arr中每个元素值

### Day08 ES6 特性

正课:

ES6

块作用域

箭头函数

参数增强

模板字符串

结构

OOP

for of

promise

#### 1. 块作用域:

(1)let: 声明一个不会被提前的变量

为什么: 传统js中声明提前破坏了程序正常的执行顺序

何时: 所有的var，都要用let代替

如何: 代替var

原理: let a之前不允许提前使用a，也不允许重复声明a

(2)let为js添加了块级作用域:

什么是块级作用域: 代码中任何一对{}之间的语句称为一个代码块，代码块中的变量，只能在块内使用

比如: if(){...}else{...} for(){...} while(){...}

问题: 没有块级作用域，块内的变量即使不执行，也会被提前到块外，污染块外部！

为什么: 为了避免块内的变量提前到块外，污染外部的变量

何时: 只要希望代码块内的变量不要影响外部的变量时

原理: 自动使用匿名函数自调方式，包括块内的代码

#### 2. 箭头函数: 简化回调函数

何时: 今后所有回调函数都可用箭头函数简化

如何:

1. 去function加=>

2. 更简化: 如果只有一个参数，可省略()

3. 更简化: 如果回调函数中只有一句话，可省略{}

强调: 如果只有一句话，则结尾;不要！

更简化: 如果仅有一句return可继续省略return

特点: 箭头内外共用同一个this

特例: 如果希望内外this不通用时，就不能用箭头函数

#### 3. 参数增强: 3种:

默认值default: ES6中可对参数指定默认值。

调用时，如果没有传入参数值，则默认使用提前指定的默认值

比如: Array.prototype.indexOf=function(elem,fromi=0){

//fromi=fromi||0;

}

强调: 有默认值的参数，必须放在参数列表的最后

剩余rest: 当传入函数的参数个数不确定时，可用...接住剩余的一切参数——代替arguments

何时: 只要参数个数不确定

为什么: arguments的3大问题:

1. 总是获得全部参数，不能有选择的获得部分

2. 是类数组对象，不是纯正的数组，不能使用数组API

3. 不能改名

如何: 在函数定义时的参数列表中: (其它固定参数, ...数组名)

优: 1. 可有选择的获得部分参数

2. 是纯正的数组

3. 可自定义名称

总结: 今后都用rest代替arguments

散播spread: 在函数调用时，用...打散数组类型的参数为单个元素，再传入函数——代替apply

何时: 如果函数本身需要多个参数单独传入，而需要传入的确是数组时

如何: 在调用时，传入参数时用"...数组"打散数组参数，单个元素分别传入

#### 4. 模板字符串:

什么是: 支持内置表达式的字符串拼接

——简化了字符串拼接

何时: 今后，只要拼接的字符串中包含要计算的表达式，就用模板字符串简化

如何:

1. 模板字符串必须用` `包裹

2. 如果包含需要动态计算的表达式，则用${}包裹

3. 反引号内默认支持换行

### Day09 ES6 特性

正课:

1. 解构

2. for of

3. class

4. \*\*\*Promise

#### 1. 解构:

什么是: 将一个对象或数组中的成员，分别赋值给多个单独的变量——简化批量赋值

何时: 只要将一个对象或数组中的成员，分别赋值给多个单独变量时

如何: 2种:

1. 数组解构:

var arr=[1,2,3];

var [a,b,c]=arr;

console.log(a,b,c);//1 2 3

强调: 数组解构靠下标对应

2. 对象解构:

var date={y:2017,m:9,d:21};

var {y:year,m:month,d:date}=date;

console.log(year,month,date);

强调: 对象解构靠属性名对应

函数调用时结构:

定义函数时: function fun({y:y,m:m,d:d}){...}

调用时: var date={y:2017,m:9,d:21};

fun(date)

#### 2. for of: 简化版的for循环

如何: for(var elem of arr)

elem可直接获得arr中每个元素值

何时: 代替for循环，遍历下标为数字的数组或类数组对象

强调: 不能遍历关联数组和对象，因为下标不是数字

缺点:

1. 无法获得下标——要用下标时，不能用for of

2. 只能遍历所有，不能有选择的遍历

3. 只能从前向后

#### 3. class类型:

什么是: 封装一类对象统一结构和API的 程序结构

——简化js中的面向对象

为什么: 传统的js类型定义方式不符合封装的特点

如何:

简化类型定义——封装:

1. 用class 类型名{}包括之前的构造函数和原型对象方法

2. 构造函数名升级为class名, 构造函数更名为constructor关键词

3. 原型对象方法不用写类型.prototype和function

直接写在class中的函数，自动添加prototype中

简化继承:

1. class 子类型 extends 父类型

类似于:

Object.setPrototypeOf(

子类型.prototype, 父类型.prototype

);

2. 借用父类型构造函数: super(参数...)

不用call，不用this！

其中: super特指父类型构造函数,且自动用当前正确this替换父类型构造函数中的this

super.父类型原型方法()

静态方法:

什么是: 不需要实例化对象，就可直接访问的成员

何时: 只要一个方法，不需要实例化也能使用时

原理: 相当于直接定义在构造函数对象上的方法

如何: static 方法(){...}

访问器属性:

get 属性名(){ return this.\_属性名;}

set 属性名(val){

if(判断)

this.\_属性名=val;

否则

报错

}}

#### 4. \*\*\*Promise

什么是: 回调函数的一种更直观的写法

——规范回调函数的使用

何时: 如果回调函数，必须等待主函数执行完，才能执行时

为什么: callback hell

如何: 2步:

0. 前一个函数定义不要在参数上写callback

1. 前一个函数定义中: 用一个巨大的Promise对象包裹函数的内容，并将Promise对象返回

Promise的参数是一个巨大的回调函数:

1. 参数: callback: 用于接收将来承诺要做的后续函数

2. 函数体: 包括原正常执行的逻辑，并在逻辑中决定何时调用callback

2. 调用前一个函数时，可用.then传入下一项任务的函数对象作为回调函数

错误处理: 2步

1. 其实: promise的函数参数，+还有第二个回调函数参数，用于在出错时调用 then catch

return new Promise(function(resolve,reject){

正常逻辑中:

如果正确执行, 调用resolve()

如果出错,调用reject("错误信息"/错误对象)

})

强调: 如果前一个函数调用了reject相当于抛出一个错误，不处理，程序就闪退

2. 错误处理:

在调用时，最后一个then的结尾追加

.catch(function(err){//err接住reject的参数

错误处理代码

})

等待多个任务完成才执行:

如何: Promise.all([ 函数1(), 函数2(), .......]).then(end)

1. 数组中的多个函数，并行！

2. 数组中最晚的任务执行后，自动调用end

## DOM

### Day01DOM,DOM树,查询

正课:

1. 什么是DOM

2. \*DOM Tree

3. \*查询: 4种

1. 什么是DOM: Document Object Model

什么是: 专门操作网页内容的统一的API标准——W3C

为什么: 为了统一操作网页内容的API标准

用DOM操作网页内容，几乎兼容所有浏览器

何时: 只要操作网页内容，都用DOM

如何:

#### 2. DOM树:

什么是: 内存中，存储网页中所有内容的树型结构

为什么: 树型结构最适合保存上下级关系，且可无限向下延伸。

何时: 只要存储不确定层级深度的上下级关系，都用树型结构

如何: 自动创建，自动维护

1. 当浏览器读取到HTM

document对象是整棵DOM L文档时，开始创建

2. 首先创建根节点document

树的树根

所有网页内容，都是document节点的后代节点

3. 依次读取网页中每个元素，属性，文本

网页中每项内容(元素，属性，文本)都是DOM树上的一个节点对象。

节点对象: Node

三大属性:

nodeType: 表示节点的类型

值是一个整数,包括:

document 9

elem 1

attr 2

text 3

何时: 只要鉴别节点的类型时

问题: 无法细致区分元素的名称

nodeName: 表示节点名称

何时: 只要进一步区分元素的名称时

包括:

document #document

elem 全大写的标签名

attr 属性名

text #text

nodeValue: 表示节点的值——了解

document null

elem null

attr 属性值

text 文本内容

DOM: \*查询，修改，添加，删除，事件绑定

#### 3. 查找: 4种:

0. 不需要查找就可直接获得元素:

html document.documentElement

head document.head

body document.body

1. 按节点间关系查找:

什么是: DOM树中任何节点都不是孤立的。

一个节点和父级，子级，兄弟之间都建立了联系

何时: 如果已经获得一个节点，找周围附近的有关系的节点时

如何: 2大类关系:

节点树: 包含所有网页内容的完整树结构

1. 父子关系:

elem.parentNode 获得elem的父节点

返回值: 唯一的一个父节点对象

elem.childNodes 获得elem的所有\*直接\*子节点

返回值: 所有直接子节点组成的类数组对象

elem.firstChild 获得elem下的第一个子节点

elem.lastChild 获得elem下的最后一个子节点

2. 兄弟关系:

elem.previousSibling 获得elem的前一个兄弟节点

elem.nextSibling 获得elem的后一个兄弟节点

问题: 一切文本都是节点对象，包括看不见的空字符，也是节点对象(tab，空格，换行)

解决: 元素树

元素树: 仅包含元素节点的树结构

优: 不受看不见的空字符的干扰

1. 父子关系:

elem.parentElement 获得elem的父元素

elem.children 获得elem的所有\*直接\*子元素

返回值: 所有直接子元素组成的类数组对象

elem.firstElementCh ild 获得elem下的第一个子元素

elem.lastElementChild 获得elem下的最后一个子元素

2. 兄弟关系:

elem.previousElementSibling 获得elem的前一个兄弟元素

elem.nextElementSibling 获得elem的后一个兄弟元素

元素树不是一颗新树，只是节点树的一个子级

问题: 1. IE9+

2. 遍历指定父节点下的所有后代节点: ——鄙视

问题: children和childNodes只能查找直接子节点，无法查找更深层次!

解决: 递归遍历:

如何: 2步:

1. 先定义函数，仅遍历直接子节点

2. 对每个子元素，调用和父元素完全相同的方法

算法: 深度优先遍历:

什么是: 每次同时碰到子元素和兄弟元素时，总是优先遍历子元素。所有子元素遍历完，才返回遍历兄弟。

问题: children和childNodes返回动态集合

什么是: 不实际存储数据，每次访问集合，都重新查找DOM树.

优: 首次查找，不需要返回完整数据，效率高!

缺: 反复访问集合，导致反复查找DOM树，效率低!

错误: for(var i=0;i<children.length;i++)

解决: 遍历时，提前缓存length

正确: for(var i=0,len=children.length;i<len;i++)

问题: 递归效率低，避免使用

解决: 用循环代替:

节点迭代器: NodeIterator:

什么是: 封装深度优先遍历算法，可以依次获取每个后代节点的对象。

如何: 2步:

1. 用父元素创建节点迭代器对象:

var iterator=document.createNodeIterator(

parent, NodeFilter.SHOW\_ALL , null, false

.SHOW\_ELEMENT

);

2. 反复调用迭代器的nextNode()方法获取下一个节点对象

nextNode()2件事:

1. 返回当前节点，2. 跳到下一个节点

如果没有后续节点，返回null

do{

var node=iterator.nextNode()

if(node!=null)

console.log(node.nodeName);

else break;

}while(true)

3. 按HTML查找: 4种:

1. 按id查找:

var elem=document.getElementById("id");

强调: 只能用document调用！

2. 按标签名查找:

var elems=parent.getElementsByTagName("标签名");

返回动态集合:

parent:父元素的id

强调: 1. 可在任意父元素上调用！仅在当前父元素下查找

2. 不仅找直接子元素，且查找所有后代中的元素

3．一律返回集合

3. 按name属性查找:

var elems=document.getElementsByName("name");

（当选按钮和复选框会找到多个）

返回动态集合:

强调: 只能用document调用

何时：仅用于查找表单中有name属性的元素

4. 按class属性查找:

var elems=parent.getElementsByClassName("class");

返回动态集合:

强调: 1. 可在任意父元素上调用

2. 不仅找直接子元素，且在所有后代中查找

3. 如果一个元素同时被多个class修饰时，只要按其中一个class，就可找到该元素

问题: 每次只能按一种条件查找，当查找条件复杂时，步骤就非常繁琐。

解决: 用选择器查找

4. 用选择器查找:

1. 仅查找一个元素:

var elem=parent.querySelector("selector");

2. 查找多个元素:

var elems=parent.querySelectorAll("selector");

强调: 1. 可在任何父元素上调用

2. 返回非动态集合:

非动态集合: 实际存储完整数据，即使反复访问集合，也不会反复查找DOM树

3. selector受制于浏览器的兼容性限 制

### Day02DOM改,增,删

正课:

1. 修改:

2. 添加和删除:

#### 1. 修改: 3种:

1. 内容: 3种:

1. 获取或修改元素的HTML代码片段内容:

elem.innerHTML

2. 获取或修改元素的纯文本内容

elem.textContent

vs innerHTML: 1. 去掉内嵌标签

2. 将转移字符翻译为原文

3. 获取或修改表单元素的值

elem.value

2. 属性:

1. HTML标准属性: HTML中规定的值，值为字符串类型的

2种:

1. 核心DOM:

操作一切结构化文档(html和xml)的统一API

特点: 繁琐！万能！

属性节点都保存在elem的attributes集合中

var node=elem.attributes["属性名"];

var value=node.nodeValue;

获取: var value=elem.getAttribute("属性名")

修改: elem.setAttribute("属性名","值")

判断有没有: var bool=elem.hasAttribute("属性名")

移除: elem.removeAttribute("属性名")

2. HTML DOM: 专门操作HTML文档的简化版DOM API

特点: 简洁, 不是万能！

其实: HTML DOM将所有HTML标准属性，已经提前封装在了元素对象中，默认值为""

如何:

获取: elem.属性名

修改: elem.属性名=值

判断有没有: elem.属性名!==""

移除: elem.属性名=""

特例: class属性:

核心DOM: 可直接使用class

HTML DOM: 必须换为.className => 就是HTML中的class

因为js的对象中已经提前有一个内部属性class，用来记录对象创建时的类型名

2. 状态属性: disabled selected checked

特点: 值是bool类型

不能用核心DOM操作

只能用HTML DOM操作

3. 自定义扩展属性:

什么是: 自定义的，不在HTML标准范围内的属性

为什么:

id的问题: 唯一！

class的问题: 不稳定，可能随时人为或被程序修改

何时: 只要标识多个元素，且不希望受个数和样式修改影响

总结: 今后，只要给元素添加行为时，查找元素都用自定义扩展属性

如何:

1. 定义: data-属性名="值"

2. 获取或设置: 2种:

1. 核心DOM

2. HTML5: elem.dataset.属性名

3. 样式: 2种:

1. 内联样式:

特点: 优先级最高, 当前元素独有

修改:

1. 仅修改一个内联样式

elem.style.css属性=值

强调: 所有css属性要去横线变驼峰

比如: z-index => zIndex

font-size=> fontSize

background-position=>backgroundPosition

2. 批量替换内联样式:

elem.style.cssText="...";

获取:

错误: elem.style.css属性

style仅表示内联样式，无法获得外部样式，丢样式

解决: 获得计算后的样式:

什么是: 最终应用到当前元素上的所有样式的合集

为什么: 一个元素的完整样式，可能来源自多个地方

何时: 只要获取样式，都要获得计算后的样式！

如何: 2步:

1. 获得计算后的style对象

var style=getComputedStyle(elem);

2. 获得style中的css属性值;

var value=style.css属性;

强调: 计算后的样式style是只读的，不能修改！

2. 修改样式表中的样式: ——了解

1. 获得样式表对象:sheet

var sheet=document.styleSheets[i];

2. 获得样式表对象中的一套规则: （一对儿大括号中的内容）

var rule=sheet.cssRules[i];

说明: keyframes: 还要继续: var sub\_rule=rule.cssRules[i]

3. 修改样式:

var style=rule.style;

style.css属性=值

问题: 一句话只能修改一个css属性值

解决: 今后都是用class来批量修改元素的样式

#### 2. 添加和删除:

3步:

1. 创建空元素:

var a=document.createElement("a");

<a></a>

2. 设置关键属性:

a.href="http://tmooc.cn";

a.innerHTML="go to tmooc";

<a href=" http://tmooc.cn "> go to tmooc </a>

3. 将新元素添加到DOM树

3种:

末尾追加: parent.appendChild(a)

中间插入: parent.insertBefore(a, child) 将a插入到child之前

替换: parent.replaceChild(a, child) 用a替换child

优化: 尽量减少操作DOM树的次数

为什么: 每次操作DOM树都会导致重新layout

什么是layout?

网页的加载过程:

html -> DOM Tree

↓

Render Tree-> \*\*\*layout\*\*\* -> paint

↑ 计算每个元素的

css -> COM 绝对布局位置

只要修改DOM树, 包括: 修改样式,修改位置,添加删除元素

都会导致重新layout——效率低

解决:

1. 如果同时添加父元素和子元素时，都要先在内存中将子元素，添加到父元素中，最后，再将父元素，一次性添加到DOM树。

2.如果父元素已经在树上了，要同时添加多个平级子元素时：//提高性能

文档片段：

什么是：内存中，临时存储一颗dom子树片段的存储空间

如何：3步

1. 创建文档片段对象

var frag=document.createDocumentFragment();

1. 先将子元素添加到文档片段中

frag.appendChild(child);

将文档片段对象追加到父元素中

### Day03DOM,BOM

正课:

1. 添加/删除

2. HTML DOM常用对象

BOM

3. window

4. \*\*\*定时器

#### 1. 添加/删除:

优化: 尽量减少操作DOM树的次数

如何: 2种:

1. 同时添加父元素和子元素时，要先在内存中将子元素添加到父元素中，最后再将父元素一次性添加到DOM树

2. 如果父元素已经在树上，要同时添加多个平级子元素时:

文档片段:

什么是: 内存中，临时存储一棵dom子树片段的存储空间

何时: 只要同时添加多个平级子元素时

如何: 3步:

1. 创建文档片段对象

var frag=document.createDocumentFragment();

2. 先将子元素添加到文档片段中

frag.appendChild(child)

3. 将文档片段一次性添加到DOM树指定父元素下

parent.appendChild(frag)

强调: frag将子元素送到dom树后，自动释放

#### 2. HTML DOM常用对象:

什么是: 对常用的元素，提供了简化版的API

优: 简化

缺: 不是万能

Image: 创建: var img=new Image();

Select:

属性: .selectedIndex 快速获得当前选中项的下标位置

.value 当选中项有value属性时，会返回option的value

如果选中项没有value属性，则用内容代替

.options: 获得select下所有option元素对象的集合

options.length 获得选项的个数

.length => .options.length

固定套路: 清空所有option

sel.innerHTML="";

sel.length=0; =>sel.options.length=0;

方法: add(option) 代替 appendChild(option)

问题: 不支持frag

remove(i) 移除i位置的option

Option:

创建: var opt=new Option(text,value);

属性: .index .text .value

table:管着行分组:

创建: var thead=table.createTHead()

var tbody=table.createTBody();

var tfoot=table.createTFoot();

删除: table.deleteTHead()

table.deleteTFoot()

获取: tabel.tHead table.tFoot table.tBodies[i]

行分组:管着行:

创建: var tr=行分组.insertRow(i)

固定套路: 1. 在结尾追加一行: 行分组.insertRow()

2. 在开头插入一行: 行分组.insertRow(0)

删除: 行分组.deleteRow(i)

强调: 主语是行分组时，i要求是在行分组内的相对下标位置

获取: 行分组.rows 获得行分组内所有行的集合

行:管着格:

创建: var td=tr.insertCell(i)

固定套路: 末尾追加: tr.insertCell()

删除: tr.deleteCell(i)

获取: tr.cells

form:

获取: var form=document.forms[i/id/name];

属性: form.elements 获得表单中所有表单元素的值

强调: 表单元素包括: input select textarea button

form.elements.length 获得表单中表单元素的个数

form.length==> form.elements.length

固定套路: 获得结尾的按钮:

var btn=form.elements[form.length-n]

方法: form.submit() //在程序中手动提交表单

表单元素:

获取: var 表单元素= form.elements[i/id/name]

如果表单元素有name属性，则: form.name属性值

方法: elem.focus() 让当前表单获得焦点

elem.blur()

#### BOM: Browser Object Model

什么是: 专门操作浏览器窗口或软件的API

没有标准!

window: 2个角色:

1. 代替ES中的GLOBAL充当全局作用域对象

2. 保存所有内置，全局的API和对象

属性: 文档显示区大小: 浏览器窗口中专门显示网页的区域

/\*window.\*/innerWidth,

/\*window.\*/innerHeight

功能: 打开和关闭窗口:

打开窗口: /\*window.\*/open("url","target")

4种:

1. 在当前窗口打开，可后退

HTML: <a href="url" target="\_self">

js: open("url","\_self ")

2. 在当前窗口打开，禁止后退

js: location.replace("新url")

用新的url代替history中当前地址

3. 在新窗口打开，可打开多个

HTML: <a href="url" target="\_blank">

js: open("url","\_blank ")

4. 在新窗口打开，只能打开一个

HTML: <a href="url" target="自定义的窗口名">

js:open("url","自定义的窗口名")

原理: target属性其实是为新窗口指定内存中唯一的窗口名

浏览器规定，同一时刻，同名窗口只能打开一个

后打开的会覆盖先打开的

预定义name:

\_self 自动使用当前窗口的name

\_blank 不指定窗口名, 每打开一个窗口，浏览器会自动随机生成内部窗口名

关闭窗口: /\*window.\*/close();

history: 保存当前窗口打开后，成功访问过的url的历史记录栈

不允许修改history内容!

只能三个操作:

前进 history.go(1)，

后退 history.go(-1), history.go(-2)，

刷新 history.go(0)

location:

什么是: 保存当前窗口正在打开的url地址的对象

属性:

.href: 获取或设置完整的url地址

.protocol: 协议

.host: 主机名+端口号

.hostname: 主机名

.port: 端口号

.pathname: 相对路径

.search: 查询字符串参数

.hash: 锚点地址

鄙视: 将search转化为对象形式:

方法:

1. 在当前窗口打开，可后退:

location.assign("url") => location.href="url"=>location="url"

2. 在当前窗口打开，禁止后退:

location.replace("新url")

3. 刷新:

普通刷新: 优先从缓存中获取资源，缓存没有或过期，才去服务器找。

f5

history.go(0)

location.reload()

强制刷新: 无论有没有缓存，都强制从服务器获取新资源！

location.reload(true)

#### 4. \*\*\*定时器: 2种:

1. 周期性定时器:

什么是: 让程序每隔一个时间间隔，自动反复执行一项任务

何时: 只要让程序按照指定的时间间隔反复执行一项任务

——动画！

如何: 3步:

1. 定义任务函数: 变化一次的函数

function task(){...}

2. 将任务函数放入定时器中反复执行:

var timer=setInterval(task,ms)

启动定时器，让定时器每隔ms毫秒，自动反复执行task函数

其中: timer 指当前定时器唯一的序号

专门用于停止定时器之值

3. 停止定时器: clearInterval(timer)

2种: 1. 定时器自动停止:

在任务函数中，设定临界值

一旦达到临界值，就自动调用clearInterval

2. 手动停止定时器

2. 一次性定时器:

### Day04BOM定时器,navigator,event

正课:

1. 定时器:

一次性:

2. \*navigator:

3. \*\*\*event

#### 1. 定时器:

一次性: 让程序先等待一段时间，再延迟执行一项任务

执行后，自动停止

何时: 只要让程序延迟执行一项任务，且不需要反复执行时

如何: 3步:

1. 任务函数:

2. 启动定时器: timer=setTimeout(任务函数,ms)

其中: ms指延迟的毫秒数

3. 停止定时器: clearTimeout(timer)

在执行前，停止等待，不再执行任务

鄙视: 定时器中的任务函数，必须等到主程序所有语句执行完，才能执行！

var a=10;

function fun(){

a++;

}

setTimeout(fun,0);

//fun无论等待多长时间都必须到最后才执行

console.log(a);//10

#### 2. navigator:

什么是: 封装浏览器配置信息的对象

何时: 只要读取浏览器配置信息时

如何:

1. 判断是否启用cookie

什么是: 客户端持久存储用户私密信息的小文件

为什么: 内存中的数据都是临时的

何时: 只要在客户端持久保存数据时

如何判断是否启用cookie:

var bool=navigator.cookieEnabled

设置: 设置->高级->隐私->内容设置->查看和禁用cookie

2. 判断是否安装插件:

什么是: 为浏览器添加新功能的小软件

如何判断插件是否安装:

navigator.plugins["插件名"]!==undefined

3. 判断当前浏览器名称和版本号——鄙视

navigator.userAgent: 保存浏览器名称，版本，内核信息的字符串

何时: 只要判断浏览器名称和版本时

#### 3. \*\*\*event: ——DOM

什么是: 用户手动触发的，或浏览器自动触发的页面状态的改变。

事件处理函数: 当事件发生时，自动执行的函数

绑定事件处理函数: 3种:

1. 在HTML中绑定事件处理函数:

<ANY ... on事件名="js语句" ...>

问题: 1. 不便于集中管理事件

2. 不便于灵活重用

总之: 不符合内容与行为分离的原则

2. 在js中，用赋值方式绑定:

ANY.on事件名=function(){

this->ANY 当前触发事件的.前的元素

}

问题: 赋值是替换原函数。每个事件只能绑定一个处理函数

3. 在js中，为元素添加事件监听对象:

ANY.addEventListener("事件名",handler)

优: 一个事件，可同时添加多个处理函数

可随时添加和移除

如何移除:

ANY.removeEventListener("事件名",原handler);

问题: 如果一个处理函数，可能被移除，则绑定时，就必须用有名的函数。不能用匿名函数。

事件模型: 当事件发生时，浏览器触发事件的过程——鄙视

DOM标准认为: 点在内层元素上，也等效于点在外层元素上了

3个阶段:

1. 捕获: 由外向内记录各级父元素绑定的处理函数

捕获阶段只记录处理函数，不执行

2. 目标触发:

目标元素: 最初实际触发事件的元素

优先触发目标元素上的处理函数

3. 冒泡: 由内向外，按捕获阶段顺序的反向，依次触发父元素上的处理函数

事件对象:

什么是: 事件发生时，自动创建的记录事信息的对象

何时: 只要获得事件的信息，或修改事件的默认行为

如何:

创建: 自动创建

获取: 事件对象e总是作为处理函数的第一个参数，自动传入。

API:

取消冒泡: e.stopPropagation();

利用冒泡:

优化: 减少事件监听的个数

为什么: 浏览器触发事件处理函数，是用遍历方式找打处理函数并执行。

何时: 只要多个平级子元素，要绑定相同事件时

如何: 只要在父元素上绑定依次处理函数，所有子元素自动共用！

2大难题:

1. 获取目标元素:

错误: this->父元素

正确: e.target->记录了最初触发事件的元素

且不随冒泡而改变！

2. 鉴别e.target是否是想要的:

元素名, class属性

阻止默认行为:

何时: 只要一个元素的事件中，带有默认行为，且默认行为不是想要的，就要阻止

如何: e.preventDefault();

三个典型:

1. <a href="#xxx"

默认: 跳到锚点，在url结尾增加#xxx

2. 阻止表单自动提交！

自定义表单提交: 2种:

1. <input type=button js: form.submit()

2. <input type=submit

=>form.onsubmit(): e.preventDefault()

3. HTML5中做拖拽效果时

必须阻止浏览器默认的拖拽行为

### Day05BOM Event坐标

正课:

#### 1. \*\*\*event

鼠标坐标

页面滚动

项目:

详情页: 放大镜效果

首页楼层滚动:

1. \*\*\*event

鼠标坐标: 3对儿:

相对于屏幕左上角: e.screenX, e.screenY;

相对于文档显示区左上角: e.clientX,e.clientY

相对于当前元素左上角: e.offsetX, e.offsetY

如何选择: 和主角的活动范围保持一致！

页面滚动:

事件: window.onscroll

滚动距离: document.body.scrollTop

自定义控制滚动:

window.scrollTo(x方向的位置,y方向的位置)

window.scrollBy(x方向的位置增量,y方向的位置增量)

## JQuery

### Day01 jQuery类型,查找

正课：

1. 什么是jQuery

2. 如何使用jQuery

3. 查询

4. 修改

1. 什么是jQuery:

什么是: 第三方开发的 执行DOM操作的 极简化的函数库

第三方: 下载

执行DOM操作: 学jQuery还是在学DOM,增删改查,事件

极简化: 是DOM操作的终极简化

函数库: jQuery一切都用函数解决

为什么: 2个原因:

1. 终极简化

2. 屏蔽了大量兼容性问题: 只要jQuery让用的，都没有兼容性问题

何时: 今后几乎所有大型项目和框架的开发，都是用jQuery完成的。

2. 如何:

鄙视: 如何学习一项新技术

1. 找官网: 是什么 http://jquery.com/

2. 看DEMO: 快速入门

3. 下手册: 备查

下载: 2种版本:

1.x 支持旧浏览器

2.x 3.x 不再支持旧浏览器

压缩和未压缩版:

jquery-xxx.js 未压缩版

包括完整的注释，代码格式，见名知意的变量名

体积大，可读性好——学习或开发时

jquery-xxx.min.js 压缩版——专门工具生成

去掉所有注释，空字符，极简化变量名

体积小，可读性差——生产环境中

部署: 2种:

1. 将js部署在服务器本地:

2. 利用CDN网络:

什么是: 网络中的多态服务器，智能判断客户端到某台服务器的网络状况，选择最优服务器下载资源

为什么: 避免距离带来的网络延迟

何时: 在项目中用到了公共的库或框架

如何: 官网->下载->CDN地址

引入网页:

先引入jquery.js，再引入自定义脚本

<script src="本地相对路径/CDN网址">

#### \*\*\*jQuery类型:

引入jquery.js时: 其实是向全局作用域中，添加了一个新的类型——jQuery:

构造函数: 负责创建jQuery类型的对象

必须先创建jQuery类型的对象，才能用jQuery

原型对象: 保存jQuery对象可用的所有简化版API

只有jQuery类型的子对象，才能用简化版API

创建jQuery对象:

什么是jQuery对象: 封装DOM元素，并提供操作DOM元素的简化版API的类数组对象

为什么: DOM元素不是jQuery类型子对象，无法直接使用jQuery的简化版API

何时: 只要使用jQuery，必须先创建jQuery对象，封装DOM对象。才能使用jQuery简化版API

如何:

1. 通过选择器，从无到有的查找DOM元素，创建jQuery对象

jQuery("selector") 可进一步简写为: $("selector")

先查找符合"selector"的DOM元素

将DOM元素装进新建的jQuery对象中

2. 将到手的DOM元素封装进jQuery对象:

$(DOM元素) 创建一个jQuery对象，保存DOM元素

jQuery API的三大特性:

1. 一个函数两用:

没给新值，默认执行获取操作

给了新值，变为执行修改操作

比如: .html() 获取内容

.html(xxx) 修改内容为xxx

2. 自带遍历:

对jQuery对象调用一次简化版API，等效于对类数组对象中每个元素都调用一次API.

3. 几乎每个jQueryAPI都返回正在使用的jQuery对象本身

链式操作

#### 3. 查找:

1. 按选择器:

$(...)支持所有css选择器！

css中没有的，jquery添加的:

基本过滤:

先将所有元素收集到一个结果集合中统一编号

再选择集合中指定位置的元素

和元素在其父元素下的位置无关

见first vs first-child.png

内容过滤: 3种:

1. 按包含的文本内容过滤:

:contains(text) 查找内容中包含指定text文本的元素

2. 按子元素特征过滤:

:has(selector) 查找内容中包含符合selector要求的子元素的父元素

3. 按是否包含内容来过滤:

:empty 空元素

:parent 非空的元素

可见性过滤:

:hidden: 只能选择两种情况的隐藏:

display:none input type=hidden

[:visible](mk:@MSITStore:C:\xampp\htdocs\11_JQUERY\jQuery1.11.3_20150618.chm::/visible.html): 就是:hidden的反义

状态过滤: 三大状态: :disabled :checked :selected

表单元素过滤:

:input 选择所有四大类表单元素:

input button textarea selected

:[type] 每种input，都对应一种专门的选择器

比如: :text :password :checkbox :radio ...

css选择器 vs jquery选择器:

优先选css，因为效率高

jquery选择器都是用js程序模拟出来

2. 按节点间关系:

父元素: .parent() =>.parentNode

直接子元素: .children() => .children

查找父元素下符合条件的子元素:

.children(["selector"]) 只能找直接子元素

.find("selector") 在所有后代中查找

第一个子元素: .children().first() => .firstElementChild

最后一个子元素: .children().last() => .lastElementChild

前一个兄弟: .prev() => .previousElementSibling

.prevAll()

后一个兄弟: .next() => .nextElementSibling

.nextAll()

除我之外，所有兄弟: .siblings();

4. 修改: 一个函数两用！

内容: .html() => .innerHTML

.text() => .textContent

.val() => .value

属性:

标准属性: .attr("属性名"[,值])

状态属性: .prop("状态属性"[,bool])

自定义扩展属性: .data("属性名"[,值])

样式:

修改内联样式:

1. 仅修改一个css属性:

.css("css属性名",值) => .style.css属性=值

2. 批量修改多个css属性:

.css({

css属性名:值,

... : ...

})

获取值:

.css("css属性名") => getComputedStyle()获得计算后的样式

### Day02改,增,删,替,克,绑定

正课:

1. 修改:

用class批量修改样式

2. 添加，删除，替换，克隆

3. 事件绑定

#### 1. 修改:

用class批量修改样式:

1. 为元素追加一个class: $(...).addClass("class名")

2. 为元素移除一个class: $(...).removeClass("class名")

3. 判断是否包含一个class: $(...).hasClass("class名")

4. 为元素切换一个class: $(...).toggleClass("class名")

if($(...).hasClass("class名"))

$(...).removeClass("class名")

else

$(...).addClass("class名")

补: .index()

2种:

1. var i=$("selector").index(jq对象/DOM对象)

查找右边的jq对象或DOM对象，在左边的结果集合中的下标位置

2. 如果在同一个父元素下找某个子元素的位置

var i=$("child").index();

#### 2. 添加,删除,替换,克隆:

添加: 2步:

1. 用$()创建一个新元素: var $新元素=$("html片段")

2. 将新元素添加到dom树:

$("parent").append($新元素)

.prepend($新元素)

$("child").before($新元素)

.after($新元素)

可以更简化: $("parent").append/prepend("html片段")

$("child").before/after("HTML片段")

删除: $(...).remove();

补: .is("selector") 判断当前元素是否符合selector的条件

替换: $("selector").replaceWith(jq对象|DOM对象)

$(old).replaceWith($new); =>返回$olds

$(new).replaceAll($old); =>返回$new;

克隆: var $clone\_elem=$(...).clone()

强调: 默认浅克隆: 仅克隆样式和属性, 不可隆行为

深克隆: 即克隆样式和属性，又克隆行为

$(...).clone(true)

#### 3. 事件绑定:

鄙视: jQuery中共有几种事件绑定方式，区别:

DOM: .addEventListener("事件名",handler)

.removeEventListener(...)

jq:

1. $("target").bind/unbind("事件名",handler)

同addEventListener()

.unbind三种重载:

.unbind("事件名",handler) 移除当前元素上，指定事件上的名为handler的处理函数。

.unbind("事件名") 移除当前元素上，指定事件上的所有处理函数

.unbind() 移除当前元素上，所有事件的监听

2. $("target").one("事件名",handle) 同bind

区别: 只触发一次，触发后，自动解绑

3. .live/die("事件名",handle)——已废弃

原理: 将所有事件集中绑定在顶级document上

4. $("parent").delegate("selector","事件名",handler)

原理: 简化利用冒泡:

1. 获得目标元素: this->e.target

2. 筛选目标元素: 第一个参数: "selector"

只有满足"selector"要求的元素，才能触发事件

鄙视: .bind vs .delegate: 3点

1. .bind直接帮在目标元素上

.delegate 帮在父元素上

2. 监听个数: .bind 监听个数多——每个目标元素都添加

.delegate 监听个数少——只给父元素添加一个

3. 新增子元素自动获得事件处理函数:

.bind 只能对现有元素添加事件监听

新增元素无法自动获得监听

.delegate 只要父元素下的元素，无论现有，还是新增，都能自动获得父元素上统一的事件监听

5. .on/off:

1. 代替bind: .on("事件名",handler) 同bind

2. 代替delegate: .on("事件名","selector",handler)同delegate

6. .事件名:

强调: 仅对常用的事件提供了终极简化

页面加载后执行: 2种:

1. DOMContentLoaded: 仅DOM内容加载完，就可提前执行

DOM内容仅包括: html和js

提前触发

何时: 只要不依赖于css和图片的所有操作都可在DOM内容加载后，提前触发

比如: 事件绑定

jq: $(document).ready(()=>{

//DOM内容加载后，就可提前执行的操作

//比如: 事件绑定

})

简化: $().ready(()=>{

更简化: $(()=>{...})

其实: 写在body结尾的script中的代码默认就是DOM内容加载后自动执行

2. window.onload 在所有页面内容加载完成后自动触发

包括: html,css,js,图片

何时: 如果js代码必须依赖css或图片才能执行时

鼠标事件:

mouseover mouseout 进出子元素，会频繁触发父元素的处理函数

mouseenter mouseleave 进入子元素，不再频繁触发父元素的处理函数

简写: 如果同时绑定鼠标进入和移出事件时，可简写为hover

$(...).hover(

e=>{...} //给mouseenter

,

e=>{...} //给mouseleave

)

更简化: 如果两个处理函数，可用toggle统一为一个处理函数，则只需要传一个参数即可

模拟触发:

虽然没有触发事件，但是依然可用程序模拟执行元素的事件处理函数

如何: $(...).trigger("事件名")

其实可以更简单: $(...).事件名()

总结: jQuery简化了DOM五大操作:

查找, 修改, 添加, 删除, 事件绑定

### Day03动画,类数组,插件

正课:

1. 动画

2. 类数组对象操作

3. 插件:

官方插件: jquery ui

第三方插件:

\*\*\*自定义插件:

#### 1. 动画: 2种:

1. 简单动画: 3种固定效果

1. 显示隐藏: $(...).show|hide|toggle()

默认: 不带动画, 单纯为了简化display:block|none

加参数: 2个:

动画持续时间ms,

动画结束后自动执行的回调函数

2. 上滑下滑: .slideUp|slideDown|slideToggle(ms,callback)

3. 淡入淡出: .fadeIn|fadeOut|fadeToggle(ms,callback)

2大缺点: 1. 效果是写死的, 不便于维护和定制

2. 用js的定时器实现动画，效率比transition低

解决: 今后凡是网页中有规律的过渡效果，都要用:

起始class+结束class+transition

2. 万能动画:

$(...).animate({

要修改的css属性: 目标值,

... : ... ,

},ms,callback)

强调: 只支持单个数值的css属性

遗憾: 不支持颜色动画！

vs transition:

1. 效率: transition高

2. 支持属性种类: transition支持颜色过渡，animate不支持

3. 灵活性: animate可随意停止

transition 不可中断

排队和并发:

1. 并发: 多个css属性同时改变

如何: 默认同一个animate内的多个css属性并发变化

2. 排队: 同一个元素上，反复调用多个animate，每个animate必须等待前一个动画执行完，才执行。

如何: 对同一个元素反复调用多个animate

停止动画: $(...).stop()

强调: 默认只停止队列中当前正在播放的动画，队列中后续动画，依然执行

解决: 停止当前动画，并清空队列: .stop(true)

选择或判断一个元素是否正在播放动画: :animate选择器

延迟执行: $(...).delay(ms).animate(...)

优先用setTimeout代替

#### 2. 类数组对象操作: 4个

1. get([index]) 等效于[index]

返回值: index位置的dom元素

强调: get() 将jQuery结果集合转化为标准的数组对象

2. size() 等效于.length

3. .each(callback) 等效于数组中的forEach

鄙视:

$(...).each(callback) vs $.each(obj,callback)

new jQuery(...).each(callback) vs jQuery.each(obj,callback)

1. 存储位置: $(...).each()保存在原型对象中

$.each() 保存在构造函数上

2. 调用的方式:

$(...).each()只允许jQuery类型的子对象调用

$.each() 所有对象都可调用！

$.each()即可遍历索引数组，又可遍历关联数组和对象

4. .index() 等效于数组中的indexOf

2种用法:

var i=$("selector").index(jq对象|DOM对象)

查找右边的对象在左边的结果集合中的下标位置

何时: 跨多个父元素查找元素在结果集合中的位置时

var i= $(当前元素).index();

查找当前元素在其父元素下的下标位置

何时: 如果在同一个父元素下查找元素的位置

#### 3. 插件:

什么是: 独立的拥有完整功能的函数或程序

为什么: 功能重用！

何时: 只要使用函数库或框架时，发现部分功能会被反复使用，或者要使用的功能函数或插件没有提供时，都可使用插件来定义并重用新功能。

包括:

1. jquery官方插件:jQuery UI

交互行为:

效果: 重写了jQuery中的部分函数:

1. 让addClass也可添加动画持续时间和回调函数

问题: 依然使用定时器模拟过渡效果

2. 让animate支持颜色动画

3. 为show/hide/toggle添加了更多的动画效果

小部件: 拥有完整样式和行为的页面小功能

包含:3部分:

1. 部件的样式: .css文件

2. 部件的行为: .js文件

3. 部件的内容和格式: 自定义

如何使用: 3步:

1. 下载并在页面中引入部件的.css文件和.js文件

强调1: jquery ui是基于jQuery开发的，所以必须先引入jquery.js，再引入jquery ui.js

强调2: jquery-ui.css必须和jqueryui的images文件夹放在同一个目录下

2. 按照部件的要求定义HTML内容和格式

3. 在自定义脚本中: 找到要应用部件的HTML元素，调用部件的函数。

2. 第三方插件:

3. 自定义插件:

### Day04 jQuery UI和Ajax

正课:

1. 插件:

官方插件: jQuery UI

\*\*\*封装自定义插件:

第三方插件:

2. jQuery的Ajax封装

\*\*\*\*\*跨域请求

#### 1. 插件:

官方插件: jQuery UI

.autocomplete();

HTML: <input ... />

JS: $(文本框).autocomplete({

source:客户端数组|"远程.php路径"

})

PHP: 2种:

1. 返回只有一个键的关联数组, 只要键名为label，客户端autocomplete可自动识别

2. 返回包含多个键的关联数组: 2步:

1. 再次调用:

.autocomplete("instance").\_renderItem=function($ul,item){

//$ul: 自动获得当前下拉列表的ul元素

//item: 获得结果结合中，当前正在加载的元素对象

//操作: 3件事:

//1. 创建一个li>div:

//从item中获取键的值，拼接到div中

//2. 将新的li追加到$ul中

//3. 返回新的li的对象

}

2. 在autocomplete中定义select处理函数:

在单击每个列表项时自动触发:

select:function(e,obj){

//obj.item是当前li对应的原集合中的对象

//常用操作: 2个

//1. 将item中的主要内容，显示在文本框中

//2. 用当前选中项作为参数，跳转到新的url

return false;//必须

}

2. 日期选择: datepicker

HTML: <input />

CSS:

JS: $(文本框).datepicker()

3. 对话款: 表单提交

HTML:

<div id="xxx" title="标题"

<form>

...

//不用写提交和重置按钮

</form>

</div>

css:

JS: $(div).dialog(); //仅将div及其内容，变为对话框的样式

总结:

问题: jquery ui采用侵入的方式，自动加载样式和行为

优: 极大的减少了开发人员的代码量

缺: 侵入的class和行为都是写死的，不便于维护

解决: bootstrap vs jquery ui

bootstrap不采用侵入的方式，隐式添加任何代码

而是仅提供样式类库

由开发人员，自主的选择应用何种class

#### 自定义插件:

何时: 只要在项目中发现频繁重用的功能，都要封装为自定义插件

如何提取:

前提: 必须使用HTML,css,js实现了插件的完整样式和功能

Step1: 将当前功能的css，提取到一个专门的css文件中

强调: css中尽量少的使用id,元素选择器

尽量一切都用class去实现！

Step2: 将当前功能的js行为，提取到一个专门的js文件中

在插件的js文件中查找自定义扩展属性的元素

强调: 将来插件都是通过查找自定义扩展属性来为元素添加行为的

今后，只要出发事件的元素，都要标记自定义扩展属性

如何使用:

1. 按插件要求，编写HTML内容结构

2. 引入插件的css, 在HTML中对应元素上，手动添加class

第三方插件:

1. jquery validate

#### 2. jQuery的Ajax封装:

$.ajax({

url:"xxx.php",

type:"get|post",

data:"参数数据"|{变量:值,...}| $(form).serialize(),

//jquery-1.11.3.js

//success:data=>{...}

})//jquery-3.2.1.js

.then(data=>{...})

补充: jquery表单操作:

$(form).serialize()

### Day05 插件,Ajax,跨域

正课:

1. 插件:

自定义插件:

2. jQuery的Ajax封装

\*\*\*\*\*跨域

#### 1. 插件:

自定义插件:

1. jQuery validation:

2. 文件上传: 3种: ——鄙视

1. 普通上传:

<form

method="post"

enctype="multipart/form-data"

action="upload\_file.php"

2. 通过ajax上传:

$.ajax({

url:"upload\_file.php",

type:"post",

data:new FormData($("#form1")[0]),

contentType:false,

processData:false

});

3. 通过插件上传:

3. 富文本编辑器: wysiwyg

必须的文件:

css: jquery.wysiwyg.css

图片: jquery.wysiwyg.gb.png

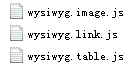
jquery.wysiwyg.gif

必须和css同目录

js: 5个:

wysiwyg 只支持旧版jquery: jquery.js 1.6.1

jquery.wysiwyg.js



html: <textarea>

#### 2. jQuery的ajax封装:

$.ajax({

url:"xxx.php",

type:"get|post|delete...",

data:"变量=值&变量=值&...",

或 {变量:值,变量:值,...},

或 $(form).serialize(),

dataType:"服务器响应回消息的类型",

text|json|xml|script|jsonp

success(data){//旧版

//在成功返回结果响应结构后自动触发

},

error(){//请求出错时自动触发},

complete(){//只要请求完成就自动触发},

beforeSend(){//在发送请求前自动触发}

})

xhr.onreadystatechange=function(){

if(xhr.readyState==4){

if(xhr.status==200){

success()

}else{//404,500,...

error()

}

complete()

}

}

xhr.open();

beforeSend()

xhr.send();

简写:

$.get("url",[data],fn)

分支: 了解:

$.getJSON("url",[data],fn)

$.getScript("url",[data],fn)

总结: 只要php中定义了header content-type，则get可完全代替getJSON和getScript

$.post("url",[data],fn)

特殊: $(parent).load("xxx.html或xxx.php")

专门以get方式请求一段HTML代码片段，并加载进当前父元素parent中

#### \*\*\*\*\*跨域:

什么是: 一个域名下的网页，想访问另一个域名下的资源

哪些情况: 4种:

1. 域名/IP不同

2. 协议不同

3. 端口号不同

4. 二级域名不同

问题: xhr禁止发送跨域的ajax请求

解决: 不是所有请求都不能跨域

比如: <link href=

<script src=

<img src=

<iframe

4种:

1. 2. 3. ——了解

4. JSONP: JSON With padding 填充式json

强调: 必须同时修改客户端和服务器端

如何: 3步:

1. 客户端: 定义js函数:

function handler (data){

//data准备接受服务器端返回的数据

//使用data

}

2. 服务器端: php:

不要只返回数据，返回一段js代码包裹数据:

首先: 获得请求中携带的客户端定义的函数名

$fun=$\_REQUEST["callback"];

echo "$fun ('$json')";

3. 动态创建一个script元素,设置src为.php路径

携带客户端处理函数名: .php?handler=函数名

jQuery中如何跨域:

$.ajax({

type:"GET",

url:"远程.php",

data:xxx,

success:fn

dataType:"jsonp"

})

# CMD导入SQL文件

开始=>Cmd=>mysql -uroot -p =>welcome>SET NAMES(可小写) utf8=>SOURCE +文件位置(可拖动进入) =>调试(有错)=>全部query OK

# 程涛

## HTML5

### Day01新语义标签,增强型表单

今天学习的内容

程涛 chengtao@tedu.cn

1:本阶段课程介绍

(1)html5新特性 8day

1.1 canvas 绘图(\*\*\*\*\*)

1.2 学子商城后台管理(\*\*\*\*\*\*)

基础好:听思路,听技术

基础弱:上课做项目/下课再做一项目

(2)Bootstrap 5day

(3)Vue 6day

3.1: 2014年推js框架

3.2:学子商城 vue 重写(\*\*\*\*\*)

#### 2:HTML5新特性

(1)新的语义标签

(2)增强型的表单

(3)音频和视频

(4) Canvas绘图

(5)SVG绘图

(6)地理定位(开网)

(7)拖放API

(8)Web Worker

(9)Web Storage

(10)Web Socket

#### 3: 增强型的表单

3.1新的input type <input type="?">

H4的input type:

text;password;submit;button;radio;select/option

checkbox;image;reset;file;hidden;

H5的input type:

email;url;number;tel;search;range;color;date;month;week

注意事项:h5新input type 存在兼容性问题(谨慎使用)

3.2新的元素

H4的元素:input;select/option;label;textarea;button

H5的元素:datalist;progress;meter;output

3.3 datalist --数据列表 datalist本身不可见

<datalist id="list3">

<option>xx</option>

<option>yy</option>

</datalist>

<input type="text" list="list3"/>

3.4 progress---进度条

<progress></progress> 左右晃动进度条

<progress value="0.7"></progress>

范围 0~1具有指定进度值进度条

练习:使用定时器+进度条实现一个可以动态

前进的进度条,到100%时弹出"下载完成"提示.

3.5: meter---刻度尺

用于标示一个值所在处的范围:不可接受(红色)可以接受(黄色)非常优秀(绿色)

<meter min="最低" max="最高" value="当前值" optimum="最佳值" low="下限" high="上限"></meter>

示例:

薪水<meter min="0" max="100000"

value="1" optimum="99999" low="20000" high="80000"></meter>

练习:使用定时器+meter 实现可以动态变化刻度尺

3.6: output--输出,语义标标签，没有任何外观样式，样式等同

span

商品单价: 3.5

购买数量:<input type="number" value="2">

小计:<output>3.5</output>

3.7新的属性

H4中表单元素属性: <input ?>

type;name;value;id;class;style;readonly;disabled;checked

H5中表单元素属性: <input ?>

placeholder;autofocus;multiple;form;required;maxlength;

minlength;max;min;pattern

(a)placeholder 占位符

<input type="text" placeholder="请输入用户名" />

(b)autofocus 自动获取输入焦点

<input type="text" autofocus/>

(c) multiple 允许输入框中出现多个输入(用逗号分隔)

(d)form 用于把输入域放置在form外部

<form id="f5"></form>

<input form="f5" />

===========输入验证相关=========

(e)required 必填项，内容不能为空

(f)maxlength 指定字符串最大长度

(g)minlength 指定字符串最小长度

(h)min 指定数字最小值

(i)max 指定数字最大值

(p)pattern: 指定正则表达式

4:学子商城后台设计与数据设计

作业1:完成学子商城登录操作

### Day02视频音频, Canvas绘图

今天学习的内容

1:复习昨天的内容

2:作业

3:今天的目标

3.1:html5新特性视频与音频

3.2:Flash被H5取代

Flash绘图(AS/Flex) ==> Canvas/SVG

Flash动画 ==> Canvas

Flash视频和音频播放 ==> video/audio

Flash 客户端存储 ==> WebStorage

#### 3.3:H5新特性--视频播放

H5提供了一个新的标签用于播放视频

<video src="res/x.mp4"></video>

<video>

<source src="res/x.mp4">

<source src="res/x.ogg">

<source src="res/x.avi">

您的浏览器版本过低，请升级

</video>

它本身是一个300\*150的inline-block元素

(1)成员属性

autoplay: false 是否自动播放 <video autoplay>

controls:false 是否显示播放控件

loop:false 是否循环播放

muted:false 是否静音

poster:"" 在播放第一帧之前显示海报

preload: 视频预加载策略

auto:预加载视频元数据及缓冲一定时长

metadata:仅加载元素数(时长,第一帧内容,尺寸)

none:不预加任何数据

(2)JS属性

currentTime 当前播放的时长

duration 总时长

paused:true 当前视频是否处理暂停状态

volume:1 当前音量 0~1

playbackRate 播放速率 大于1快播 小于1慢播

(3)成员方法

play(); 播放视频

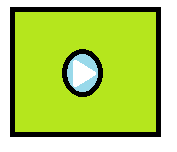
pause(); 暂停播放

(4)成员事件

onplay 当视频开始播放时触发事件

onpause 当视频暂停时时触发事件

练习1:不使用video自带controls自定义播放/暂停按钮



练习2:不论何时只要视频暂停显示一副广告,只要播放，就隐藏广告



#### 3.4:H5新特性--音频播放

<audio src="x.mp3"></audio>

<audio>

<source src="x.mp3" />

<source src="x.wav" />

</audio>

它默认300\*30的inline-block元素,但若没有controls属性不可见

audio标签/对象/方法/属性

(1)成员属性

autoplay: false 是否自动播放 <video autoplay>

controls:false 是否显示播放控件

loop:false 是否循环播放

muted:false 是否静音

preload: 视频预加载策略

auto:预加载视频元数据及缓冲一定时长

metadata:仅加载元素数(时长,第一帧内容,尺寸)

none:不预加任何数据

(2)JS属性

currentTime 当前播放的时长

duration 总时长

paused:true 当前视频是否处理暂停状态

volume:1 当前音量 0~1

playbackRate 播放速率 大于1快播 小于1慢播

(3)成员方法

play(); 播放视频

pause(); 暂停播放

(4)成员事件

onplay 当视频开始播放时触发事件

onpause 当视频暂停时时触发事件

练习:使用复选框控制网页的背景音乐

[\*]是否播放背景音乐

cb.onchange = function(){

this.checked

}

#### 3.5:html5 绘图技术--Canvas(重点&难点)

网页中的可用绘图技术

网络中统计图，网页游戏(第三方游戏框架)，3D应用，酷弦的特效，地图应用(SVG)都要使用绘图技术。

常用绘图技术有三种

(1)SVG绘图:2D矢量图绘图技术，2000年出现,后纳入H5标准

(2)Canvas绘图:2D位图绘图技术,H5提出的绘图技术

(3)WebGL绘图:3D绘图技术，未纳入H5标准

第三方游戏开发引擎

(1)cocos2d-js http://www.cocos.com/docs/js/index.html

(2)egret

(3)lufylegend http://www.lufylegend.com/

第三方3D 引擎

three.js

|  |
| --- |
| Canvas绘图难点所在  (1)坐标系  (2)单词比较多 |

3.6:Canvas 绘画

Canvas:画布

<canvas id="c2" width="500" height="400">

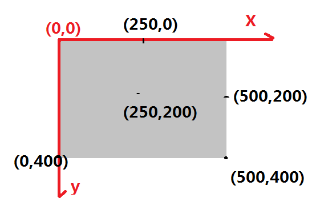
</canvas>

注意事项:画布高度与宽度只能使用html/js属性来赋值

不能使用css

每个画布上有且只有一个画笔,进行绘图

var ctx = c2.getContext("2d"); //获取画布上的画笔



3.7:Canvas 绘画--矩形

矩形的定位点在自己的左上角

ctx.lineWidth = 1; 描边宽度(边线宽度)

ctx.fillStyle = "#000" 填充的样式

ctx.strokeStyle="#aaa"描边样式

ctx.fillRect(x,y,w,h); 填充一个矩形

ctx.strokeRect(x,y,w,h)描边一个矩形

ctx.clearRect(x,y,w,h) 清除一个矩形范围内的绘图

练习1:在画布左上角，左下右上角，右下角，居中画矩形

练习2:在画布上描边一个可以左右移动矩形

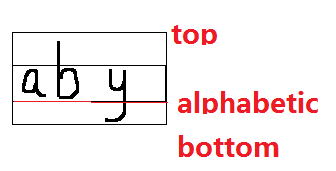
提示:使用定时器+先清除画布+再重新绘制矩形(x不停增加)

今天作业:学子商城商品列表分页查询

练习3:在画布上描边一个可以上下移动 17:28--17:33

练习4:在画布上描边一个可以45度

3.7:Canvas 绘制文本



ctx.textBaseline = "alphabetic"; //文本基线??

ctx.font = "19px sans-serif"; //文本大小和字体

ctx.fillText(str,x,y); //填充一段文本

ctx.strokText(str,x,y); //描边一段文本

ctx.measureText(str); //检测文本宽度 {width:x}

今天作业1:

创建datas.php程序接收请求返回如下数据

[

{"label":"部门1","value",150},

{"label":"部门2","value",190},

{"label":"部门3","value",180},

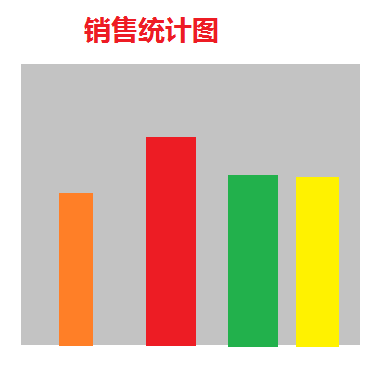
{"label":"部门4","value",180},

]

创建salesdata.html 显示画布并且发送ajax请求

datas.php,接收返回数据，绘制如下图形

柱子高度 value值



今天作业2:学子商城商品列表分页查询

### Day03 Canvas路径,图像

今天学习的内容

1:复习昨天HTML5重点知识

2:作业(验证码/分页)

3:今天的目标

3.1: HTML5新特性--Canvas

#### 3.2: HTML5新特性--Canvas---路径

path：由多个坐标点组成任意形状,路径不可见，

可用"描边","填充","裁剪'

ctx.beginPath(); 开始一条新路径

ctx.closePath(); 闭合当前路径

ctx.moveTo(x,y); 移动到指定点

ctx.lineTo(x,y); 从当前点到指定点画直线

ctx.arc(cx,cy,r,start,end); 绘制圆拱路径

cx,cy 圆心

r 半径

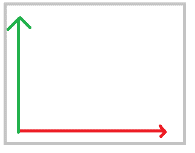
start 启始角度 角度 0~360 弧度 0~2\*PI

end 结束角度 角度 90\*Math.PI/180

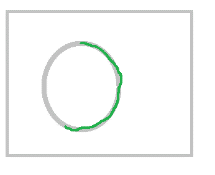
ctx.stroke(); 描边

ctx.fill(); 填充

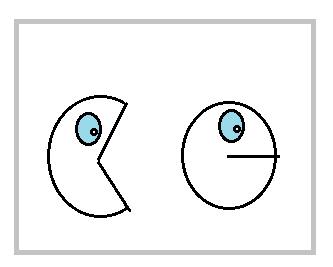
练习1:使用路径描边绘制一个坐标轴



练习2:使用圆拱形绘制可以前进的圆环进度条



练习3:创建二个函数 openMouth() closeMouth();



#### 3.3: HTML5新特性--Canvas---图像

canvas属于客户端技术，图片在服务器中，所以浏览器必须先下载

要绘制图片，且等待图片异步加载完成。

var p3 = new Image();

p3.src = "img/p3.png"; //向图片发送异请求

p3.onload = function(){ //图片下载完成自动触发事件

console.log(p3.width);

ctx.drawImage(p3,x,y); //原始大小绘图

ctx.drawImage(p3,x,y,w,h); //拉伸绘图

}

练习:右上角 左下角 右下角居中 400\*200

练习:左右移动飞机 16:22-16:27

#### 3.4: HTML5新特性--Canvas---变形操作

canvas绘图中也变形技术，可以针对某一个图像/图形绘制过程中进行变形:

ctx.rotate(弧度); 旋转画笔，轴点画布的原点

ctx.translate(x,y); 将整个画布的原点平移到指定点

ctx.save(); 保存画笔所有变形状态值 (原点,旋转角度);

ctx.restore(); 恢复画笔所有变形状态值

17:10-17:15

练习:在画布左上角画一个绕自己为中心旋转飞机1

练习:在同一个画布上右下角画旋转速度是飞机1

2倍飞机2

今天作业1:完成分页所有功能

今天作业2:删除(expire='1')/更新(价格)/详细()

(1)将商品添加一列 expire enum('0','1');

(2)更新所有商品添 expire '0' 正常

### Day04 Canvas,Svg绘图

今天学习的内容

1:复习上周知识重点

2:作业(分页)

3:今天的目标

3.1: h5新特性---canvas 绘图

3.2: h5新特性---canvas 第三方绘图工具库

(1)Two.js

(2)ECharts: 百度提供免费绘图工具,与地图整合的很好

(3)d3.js 绘图工具

#### 3.3: h5新特性---Svg 绘图

位图:由一个一个像素点且体制改革，每个点各有自己的颜色，

色彩细腻，但放大会失真.

矢量图:由一个一个线条组成，每个线条可以指定颜色，方向,

可以无限缩放，但细节不够丰富.

|  |  |  |
| --- | --- | --- |
|  | canvas绘图 | svg绘图 |
| 类型 | 2d位图 | 2d矢量图 |
| 如何绘图 | 使用js代码绘图 | 使用标签绘图 |
| 事件绑下 | 每个图形不是元素，无法直接给绑定事件 | 每个图形都是元素，可以直接绑定事件 |
| 应用场合 | 统计图，游戏，特效 | 统计图,图标,地图 |

svg技术诞生于2000年,早期作业XML的扩展，H5标准把常用SVG标签采纳为标准，但有些被废弃.

**svg技术在html5出现之前使用方法**

(1)在一个xml文档中声明绘制图形

(2)再编写html文档，使用 img/object加载xml文档

**svg技术在html5出现之后使用方法**

直接创建html5文档,在其中书写svg标签即可

<svg></svg>

<rect></rect> 绘制矩形

<circle></ circle > 绘制圆形

<ecllipse></ ecllipse > 绘制椭圆形

<polyline></polyline> 绘制折线

练习1:在svg画布正中央创建30\*300矩形柱子，初始时淡红色

(#faa)填充和深红色(#800)边框都是半透明的

鼠标悬停时，变为不透明。

提示:修改html元素属性用setAttribute();

练习2:绘制两个柱子，初始高度0,使用定时器，不断改它们高度,

到300停止

练习3:使用ajax从服务器获取一段数据

[{"label":"部门一",value":180}....]

根据这些数据绘图统计图

#### 3.4:使用svg 进制绘图--圆形

<circle r="" cx="' cy="" fill="" fill-opacity="" stroke=""

stroke-opacity></circle>

练习1:在画布左上角，右上角 左下角，右下角四个圆

练习2:在画布上随机绘制30个实心圆形，大小随机，

位置随机，颜色随机，透明度随机,

点击某个圆形之后，慢慢变大变淡，直到消失，从DOM删除

#### 3.5:使用svg 进制绘图--椭圆形

<ellipse cx="' cy="" rx="" ry=""></ecllipse>

rx:水平半径

ry:垂直半径

3.6:使用svg 进制绘图--直线

<line x1="" y1="" x2="" y2="" stroke="" stroke-width=""

stroke-linecap="butt/round"></line>



3.7:使用svg 进制绘图--折线

<polyline points="50,50 100,50 .." fill="transparent" stroke="">

作业1:删除/更新(价格)/详细信息

作业2:了解第三方绘图工具库 two.js echarts.js

### Day05Svg绘图,地理定位

今天学习的内容

1:复习昨天的内容

2:作业（重点&难点）

3:今天的目标

3.1: svg 绘图

3.2: 使用SVG进制绘图--文本

<text alignment-baseline="before-edge"

font-size="" fill="" stroke="" x= y>文本内容</text>

3.3: 使用SVG进制绘图--图片

<image xlink:href="x.jpg" x="" y="" width="" height="">

3.4: 使用SVG进制绘图--渐变对象

<defs> 定义特效对象:渐变对象属于一种特效对象

<linearGradient id="g3" x1="" y1="" x2="" y2="">

<stop offset="0%" stop-color="red" />

<stop offset="50%" stop-color="yellow" />

<stop offset="100%" stop-color="green" />

</ linearGradient >

</defs>

<ANY fill="url(#g3)" stroke="url(#g3)"></ANY>

练习:使用渐变对象，创建三个柱子，分配不同渐变色

3.5: 使用SVG进制绘图--滤镜

<defs>

<filter id="f2">

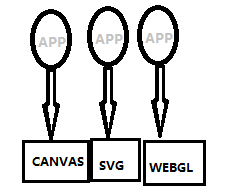
<feGaussianBlur stdDeviation="3" />

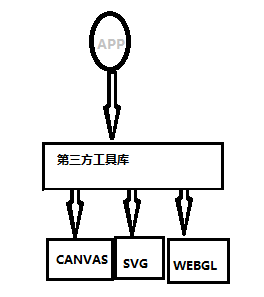
</filter>

<defs>

<ANY filter="url(#f2)">

3.6: 第三方绘图工具库





3.7 two.js

http://two.js.org

是一个2D绘图函数库,提供一套API，可用于不同技术下的绘图

SVG/CANVAS/WEBGL

使用方法

(1)创建html引用js文件

(2)创建对象

var two = new Two({}).appendTo(box);

(3)绘制图形

var c = two.makeCircle(200,200,100);

var r = two.makeRectangle(x,y,w,h);

(4)把绘图内容绘制DOM树

two.update();

https://two.js.org/

3.8 echarts.js

3.9:html5新特性---地理定位(开网)

Geololcation:地理定位,使用js获取当前浏览器所在地理坐标

(经度,维度,海拨,速度)数据，用于实应用:饿了么,高德导航...

手机浏览器如何获取定位信息

(1)首选手机中GPS芯片与卫星通信，定位精度在米

(2)次选手机通信基站定位获取，定位精度在公里

PC浏览器如何获取定位信息

通过ip地址进行反向解析,定位精度于ip地址库大小

PC浏览器如何获取定位信息

HTML5中提供新的对象: geolocation 用于获取浏览器

定位信息.

3.10:html5新特性---百度地图

(1)通过手机注册百度开发者帐户

http://lbsyun.baidu.com/

(2)创建一个网站:登录百度地图，为你的网站申请AccessKey

创建应用[aaa] 生成AK

(3)在自己网页中嵌入百度地图

作业1:php上传文件

http://www.w3school.com.cn w3c

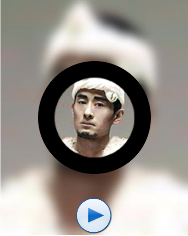
http://www.runoob.com 菜鸟

http://php.net/manual/zh/features.file-upload.post-method.php

https://www.cnblogs.com 博客园

作业2:音乐播放器、验证码(canvas)

注意事项:验证码一定后台程序输出(php/java/nodejs)





作业3：更新-->用户管理(分页/删除/更新/详细)

### Day06拖放API,WebWork

今天学习的内容

1:复习昨天知识重点

2:作业(项目;商品更新;用户管理;canvas;上传图片)

3:今天的目标

3.1:HTML5新特性:拖放API --- 重点&难点

Drag&Drop: 拖动和释放

HTML5为拖放源对象可以触发事件

dragstart 拖动开始

drag 拖动中

dragend 拖动结束

整个拖动过程: dragstart1+drag\*n+dragend\*1

HTML5为拖放目标对象可以触发事件

dragenter 拖动着进入

dragover 拖动着悬停在上方

dragleave 拖动着离开

drop 拖动释放

整个过程1:dragenter\*1+dragover\*n+dragleave\*1

整个过程2: dragenter\*1+dragover\*n+drop\*1

注意事项:

必须阻止dragover的默认行为,drop才可能触发

练习:使用拖动事件，实现拖动删除效果

上方垃圾箱，下方三架飞机

提示:在刚开始拖动 src.ondragstart 记录被拖动源

对象id,释放时(target.drop)根据id找到飞机并且删除

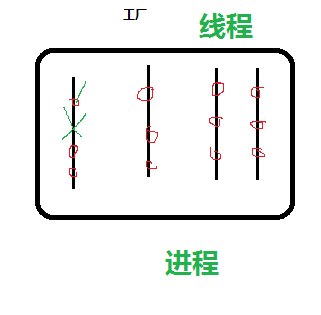
div.removeChild(c);

3.2: HTML5新特性--WebWorker (代码3行)

程序:program 可被 cpu执行代码，存储在外存中.

进程:process 程序OS调用内存,分配执行空间，等待CPU执行.

线程:Thread 线程是进程内执行代码基本单位



进程和线程

(1)进程是操作系统分配内存基本单位

(2)线程是CPU执行代码的基本单位

(3)线程必处于某个进程内部

(4)一个进程内必须至少有一个线程，也可以有多个

(5)一个操作系统可能同时存在几千个线程,它们是"并发执行"

宏观上看同时执行，微观上依次循环执行.

chrome浏览器的线程模型

一个chrome进程内部(旧版本6个线程),可以同时/并发向web服务器请求资源， ---请求资源线程

只有一个线程负责将内容绘制浏览器页中内容和解析js.--UI主线程

3.3:WebStorage

作业1：音乐播放器

作业2: 学子商城用户管理分页

作业3：跑步兔子

### Day07WebWork,Web Storage

今天学习的内容

1:复习昨天重点

2:作业(商品搜索分页/音乐播放器)

3:今天的目标

3.1:Web Worker

<button>按钮1</button>

<script src="x.js"></script>

<button>按钮2</button>

上述代码中若x.js很耗时，按钮1无法点击，按钮2在运行x.js

不可见---所有代码(html/css/js);

解决方案:创建一个并发执行新线程，让它来执行耗时任务.

#Worker线程有弱点

浏览器不允许不Worker线程操作任何的DOM/BOM对象

原因:浏览器只允许UI主线程操作DOM/BOM

所以:类以jQuery不能使用Worker执行.

**Worker线程可以给UI主线程发送消息**

UI主线程(收)

var s3 = new Worker("x.js");

s3.onmessage = function(e){e.data}

Worker线程(发)

postMessage(string)

**UI主线程发送消息给Worker线程**

UI主线程(发)

var w3 = new Worker("x.js");

w3.postMesssage("hello worker");

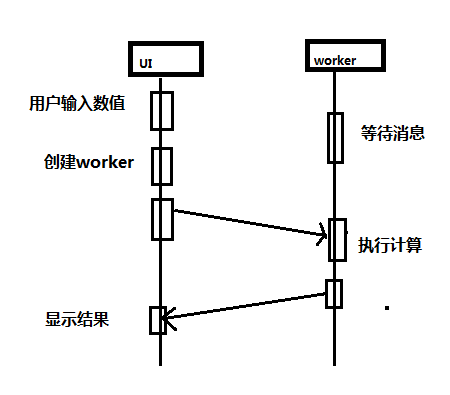
Worker线程(收)

onmessage = function(e){ e.data}

练习:在HTML中有一input,再有按钮“开始计算累加和”

点击此按钮，创建Worker线程计算出用户输入数字累加和

在下方有一个div显示计算结果.



项目中的worker的使用场景

1:只要js中有DOM/BOM就不能用worker

2:worker适合于耗时JS任务,复杂计算,加密,解密,大数据统计

3.2:Web Storage---->(重点&难点)

在浏览器中存储当前用专有数据;访问历史;内容定制;样式定制

在客户端存储数据可以使用技术

(1)Cookie技术:浏览器兼容性好;不能超过4KB,操作复杂

(2)Flash存储:依据Flash播放器

(3)**H5 WebStorage** :不能超8MB，操作简单

(4)IndexDB:可存储大量数据,还不是标准技术

|  |
| --- |
| Session会话:浏览器从打开某个网站的一个页面开始，中间可能打开很多个页面，直到关闭浏览器，整个过程称为"一次会话" |

WebStorage技术中，浏览器为用户提供两个对象

(1)window.sessionStorage类数组对象,会话级数据存储

此对象可供此会话中所有页面共同使用，浏览器一旦关闭就数据消失了，作用:登录用户名;登录用户id;该用户权限

sessionStorage[key]=value; #保存数据

sessionStorage.setItem(key,value); #保存数据

sessionStorage[key]; #获取数据

sessionStorage.getItem(key); #获取数据

sessionStorage.removeItem(key); #删除数据

sessionStorage.clear(); #清除数据

sessionStorage.length; #数据个数

sessionStorage.key(i); #获取第i个key

练习:创建 index.html login.html logout.html

[首页]index.html 右上角提示"请登录"超链接

[登录]login.html 输入用户名，密码，提交按钮 (sessionStorage)

提示"登录成功,3s后跳转 index.html"

setTimeout+location.href=index.html;

[首页]index.html 右上角提示"欢迎回来 ：xxx 退出登录"

[退出]logout.html 删除 sessionStorage name pwd

提示"您己退出,3s后自动跳转首页"

[首页]index.html 右上角提示"请登录"超链接

(2) localStorage类数组对象，本地存储(跨会话)

在浏览器所能管理的外存(**硬盘**)中存储着用户浏览数据，

可供此次会话以及后续会话的页面共同使用

浏览器关闭也不会消失--永久存在

localStorage [key]=value; #保存数据

localStorage.setItem(key,value); #保存数据

localStorage [key]; #获取数据

localStorage.getItem(key); #获取数据

localStorage.removeItem(key); #删除数据

localStorage.clear(); #清除数据

localStorage.length; #数据个数

localStorage.key(i); #获取第i个key

17:53--17:58

练习:创建 index.html,有一个下拉菜单,"请选择你喜欢的主题"

"蔚兰天空""芭比公主""暗黑主题" 各对应一个class

.blue{background:#ddf;color:#33a}

.pink{background:#fdf;color:#a3a}

.dark{background:#333;color:#eee}

用户选中某个选项，(select.onchange)为当前页面(body)

使用指定样式;

作业1：拖动图片上传文件

作业2: 商品搜索

### Day08WebStorage,WebSocket

今天学习的内容

1:复习昨天知识重点

2:作业(商品搜索;上传文件;商品添加)

3:今天的目标

3.1: Web Storage

3.2: Web Socket --(重点)

HTML5新特性--WebSocket--代码少-原理需要理解

HTTP协议:属于"请示-响应"模型，只有客户端发起请求消息，服务器才会返回响应消息，没有请求就没有响应，一个请求，只能得到一个响应，在些场景就力不从心：股票实时走势应用

解决方案：长轮询(Long-Poling)/心跳请求

--定时器/ajax--请求过于频繁，服务器压力过大，不够频繁，客户数据延迟较大.

WebSocket协议:属于"广播-收听"模型，客户端连接服务器就不再断开，永久的连接，双方就随时向对方发送消息，且是全双工不等待对方，WS协议在实时走势应用中有特别优势.

WS协议的应用程序也分为客户端程序和服务器程序

**ws 服务器应用**

监听指定端口，接收客户请求，向对方发消息，并接收消息

java/php/nodejs

**ws 客户端应用**

主动发起连接请求，保持永久连接，向对方发消息并接消息，可以

php/java/nodej/html5

掌握:

(1)连接到ws服务器 var ws = new WebSocket("ws://127.0.0.1:9001");

(2)向服务器发消息 ws.send(string)

(3)接收服务器发来消息 ws.onmessage = function(e){e.data}

(4)断开WS服务器的连接 ws.close();

练习:单词测试

(1)用户在save.html中不停录入新单词

(2)进入test.html开始测试，需要对之前录入的所有单词测试

(3)提交答案后，在result.html显示测试成绩

提示:录入的单词需要永久保存，而此次测试结果只需要在当

前会话保存. 17:40--17:50



//test.html 显示所有英文

for(var i=0;i<localStorage.length;i++){

var key = localStorage.key(i); //英文

var val = localStoage.getItem(key); //中文

}

作业1:商品添加;2:商品图片上传(原大小/md/sm/xs)

123123123.jpg

md\_123123123.jpg 320\*120

sm\_123123123.jpg 120\*80

xs\_123123123.jpg 80\*60

作业2:用户管理(分页/搜索/删除/更新/详细/添加)

作业3:用户管理(权限管理 RBAC[基于角色访问控制])

## Bootstrap

### Day01

今天学习的内容

1:复习html5新特性

2:作业

3:今天的目标

3.1 Bootstrap 响应式布局框架

3.2:什么是响应式网页

1G:只能通话

2G:可以通话，短信

2.5G：GPRS:可以通话，短信，上网--WML

3G:IOS/Android(Linux)可以通信,短信,视频,上网--HTML

京东: www.jd.com m.jd.com

3.3:Responsive Web Page:响应式网页/自适应的网页,2010提出，

一个网页，会自动根据用户浏览设备不同，自动改变布局,

可以被PC/PAD/PHONE正常浏览.

3.4:响应式网页必备

(1)流式布局 float:left;display:inline-block

(2)可以改变尺寸的图片

(3)可以改变尺寸的文字

响应式网页的不足

代码更加复杂,不适合于大型页面

3.5:如何测试响应式网页

(1)使用真实物理设备

(2)使用第三方测试软件

(3)使用chrome内置设备模拟器

3.6 自定义的响应式布局(重点&难点)

(1)声明viewport(视口)元素标签

|  |
| --- |
| viewport元标签  早期3G手机为浏览大尺寸网页，只能强行把网页缩小,  导致网页中图片,文字，超链接很少.  IOS提出一个"视口"的概念,视口用于盛放网页的内容，尺寸可以指定.  Android也借鉴该概念 |

width=device-width 表示视口宽度是设备屏幕宽度

initial-scale=1.0 表示初始的缩放比例

use-scalable=no 表示用户是否可以调整缩放比例

maximum-scale=1.0 最大缩放比例

minimum-scale=1.0 最小缩放比例

(2)使用流式布局

实现方法:float:left

(3)所有容器使用相对尺寸大小，不使有绝对

.content{

/\*width:500px;\*/

width:100%;

}

(4)所有文字使用相对尺寸大小，不使有绝对

body{

/\*font-size:12px;\*/

font-size:1.2em;

}

(6){最重要原则}使用CSS3 Media Query技术

Media:指浏览网页的设备,如 screen(pc/pad/phone);print;tv

Media Query:查询出当前浏览网页设备的类型和特性

(对比度,尺寸,手持方向)

3.7:媒体查询二种方式

(1)根据媒体查询的结果执行不同外部CSS文件

<link media="screen and (min-width:768px) and (max-width:991px)" rel="stylesheet" href="css/pad.css">

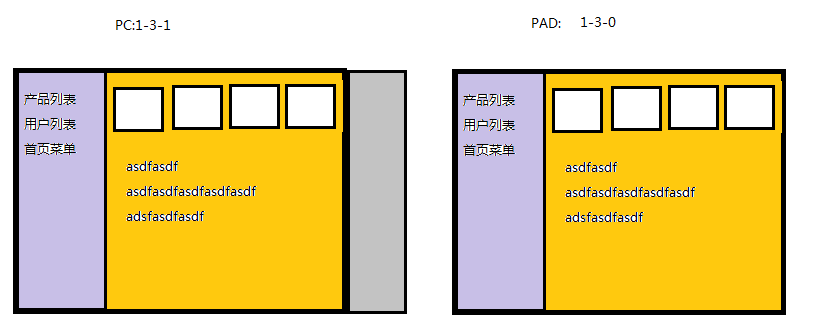
(2)根据媒体查询的结果执行CSS片段中不同部分

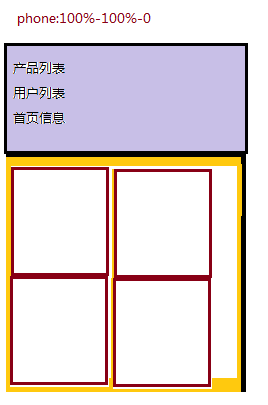
@media screen and (min-width:768px) and (max-width:991px){

选择器(样式设定)

}

练习:使用CSS3 Media Query 编写响应式网页





3.8:Bootstrap(起步,引导程序):

3.9:Twiter Bootstrap

中文网站: http://www.bootcss.com/

Bootstrap是一个HTML/CSS/JS框架,用于开发响应式布局，

移动设备优先WEB项目.

内容分为五部分:

(1)起步

(2)全局CSS样式

(3)组件

(4)JS插件

(5)定制(less)

提示:Bootstrap知识点不难，但难点1000+单词记忆

3.10: 起步下载配置Bootstrap

3.3.5 js/css/fonts

3.11: 基本模板

|  |
| --- |
| <html lang="zh-CN">  Language:指定当前网页所用自然语言,zh zh-CN,zh-hk  作用有两个:为浏览器翻译功能确定基础语言,为读屏软件确定基础发音. |
| <meta http-equiv="X-UA-Compatible" content="IE=6">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  IE=edge 告诉IE使用最新的引擎渲染网页 |
| html5shiv.js  第三方编写JS文件，是一个自调函数，让老浏览器兼容HTML5标签，如 header/aside/footer... |
| respond.js  第三方编写js文件，是一个自调函数，让老浏览器支持响应式网页。----CSS3 Media Query |
| Bootstap.css 提供默认样式重置(CSS Reset) |

作业1:自定义响应布局

作业2:Bootstrap 全局CSS样式--表格

作业3:商品图片上传->原始大小->大图->中图->小图

--选作

### Day02

今天学习的内容

1:复习昨天的知识重点

2:作业

3:今天的目标

3.1:Bootstrap 全局样式 (重点&难点 栅格系统)

3.2:Bootstrap全局样式 按钮

.btn 按钮基础样式

.btn-default 白底黑色按钮

.btn-danger/warning/success/info/primary 五种颜色

.btn-lg/sm/xs 按钮四种大小

.btn-block 声级按钮

3.3:Bootstrap全局样式 图片

.img-rounded 圆角图片

.img-circle 圆形图片

.img-thumbnail 缩略图片

.img-responsive 响应式图片

3.4:Bootstrap全局样式 文本

文本颜色 .text-danger/warning/successd/info/primary

背景颜色 .bg-danger/ warning/successd/info/primary

对齐方式 .text-left/center/right/justify

特效 .text-uppercase/lowercase/capitalize

3.5:Bootstrap全局样式 列表

.list-unstyled 去除提示符号

.list-inline 行内列表

3.6:Bootstrap中屏幕的分类

(1)大型PC屏幕(Large-lg) w>=1200px

(2)中型PC屏幕(Medium-md) 1200px>=w>=992px

(3)小型PAD屏幕(Small-sm) 991px>=w>=768px

(4)超小PHONE屏幕(ExtraSmall-xs) 767px>=w

3.7:Bootstrap全局样式--表格

.table

.table-bordered 带边框的表格

.table-striped 隔行变色的表格

.table-hover 带悬停变色的表格

.table-responsive 响应式表格(父元素)

查看手册"表格",实现表格

|  |
| --- |
| Bootlint是Bootstrap官方提供HTML检测工具，  它能自动检查某些常见html/css错误,  检测组件是否符合Bootstrap结构规则 |

3.8:Bootstrap全局样式--栅格布局系统(Grid System)

网页布局可用的方法

(1)table布局 好处:简单易控制,不足:语义错误,页面渲染效率低

(2)div+css布局 好处:语义正确,页面渲染效率高,不足:不容易控制

(3)栅格 好处:简单易控制，语义正确，渲染效率高,

3.9:Bootstrap全局样式--栅格使用方法

(1)最外层使用容器

div.container或 div.container\_fluid

(2)容器里可以放置任何内容，若想使用栅格系统.必须用 div.row

div.container>.row

(3)一个.row中不能放其它内容，只能放置.col列中可以放置任何内容

div.container>.row>.col

(4)bootstrap中行默认均分为12等份,每个列必须指定在行中占比

(5)栅格系统针对不同屏幕提供不同列

.col-lg-1/2/3/4../12

.col-md-1/2/3.../12

.col-sm-1/2..

.col-xs-1/2....

(6)可以使用"偏移"实现指定列及其后的列向右偏移的效果，如果不

足12列居中对齐

.col-lg-offset-1/2/3/....

.col-md-offset-1/2/3/....

.col-sm-offset-1/2/3/....

.col-xs-offset-1/2/3/....

(7)需要不同屏幕下列的适用

.col-xs-\* 适用xs/sm/md/lg 屏幕

.col-sm-\* 适用sm/md/lg 屏幕

.col-md-\* 适用md/lg 屏幕

.col-lg-\* 适用lg 屏幕

(8)一个列可以指定在特定屏幕下隐藏

.hidden-xs 仅在xs屏幕下隐藏

.hidden-sm 仅在sm屏幕下隐藏

.hidden-md 仅在md屏幕下隐藏

.hidden-lg 仅在lg屏幕下隐藏

(9)栅格系统可以嵌套，规则:在列中再嵌入行，行中再有列

.container >

.row >

.col-\*-\* >

.row > ...

3.10:Bootstrap全局样式--表单(小难点)

(1)默认表单

<form>

<div class="form-group">

<label class="control-label">用户名</label>

<input type="text" class="form-control" for=""/>

<span class="help-block"></span>

</div>

</form>



(2)行内表单

<form class="form-inline">

<div class="form-group">

<label class="control-label" class="sr-only">用户名</label>

<input type="text" class="form-control"/>

</div>

</form>



(3)水平表单

|  |  |  |
| --- | --- | --- |
|  | 栅格系统 | 水平表单中的栅格 |
| 最外层元素 | div.container | form.form-horizontal |
| 行 | div.row | div.form-group |
| 列 | div.col-\*-\* | div.col-\*-\* |

<form class="form-horizontal">

<div class="form-group">

<div class="col-\*-\*">

<label>用户....</label>

</div>

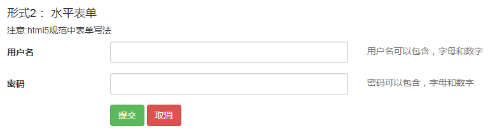
<div class="col-\*-\*">

<input type="text"...

</div>

</div>

</form>



3.11:Bootstrap组件----下拉菜单

下拉菜单必须三级结构

<div class="dropdown">

<a href="#" data-toggle="dropdown">触发元素</a>

<div/ul class="dropdown-menu">隐藏元素</div/ul>

</div>

作业1:学子商城图片上传

作业2:学子商城bootstrap

### Day03

今天学习的内容

1:复习昨天重点

2:作业-----(重点&难点)

3:今天新的目标

3.1: Bootstrap 组件----图标字体



web项目中常见图标字体

(1)FontAwesome---675个免费的图标

(2)Glyphicons----800个收费的图标

服务器端字体的使用方法:

(1)web项目目录必需有字体文件

(2)css声明一个新的字体，并指定字体件下位置

@font-face{

font-family:"Group...";

src:url("../fonts/gly.....eot");

}

(3)<span class="glyphicon glyphicon-save"></span>

练习:在按钮上添加如下小图标 14:50--14:55

主页(小房子),用户,配置,汉堡包,刷新,打分(星号),我喜欢(心形),发邮件,拍照定位,购物车,删除(垃圾箱),加号,减号对勾,前进,后退,前一张，后一张,播放暂停,停止, 快进,快退

3.2: Bootstrap 组件--警告框

<div class="alert alert-四种颜色">

<span class="close" data-dismiss="alert">

<p>......</p>

</div>

3.3:Bootstrap 组件--进度条

.progress > .progress-bar

3.4: Bootstrap 组件

分页条/徽章/标签/巨幕/水井

3.5: Bootstrap 组件--缩略图

<div class="thumbnail">

<img src="x.jpg">

<div class="caption"></div>

</div>

3.6: Bootstrap 组件--媒体对象

以"左中/中右/左中右"横向三部分展示一个商品/贴子

<div class="media">

<div class="media-left">img</div>

<div class="media-body">p/div</div>

<div class="meda-right">img</div>

</div>

3.7: Bootstrap 组件--面板

以"中/上中下/中下/上中" 纵向三部分展示数据

<div class="panel panel-颜色">

<div class="panel-heading"></div>

<div class="panel-body"></div>

<div class="panel-footer"></div>

</div>

面板一个重要用途:实现手风琴组件

3.X:Bootstrap JS插件

作业1：第一种授权方式自己完整实现

作业2：第二种授权方式创建表

**作业3：**实现手风琴组件

### Day04

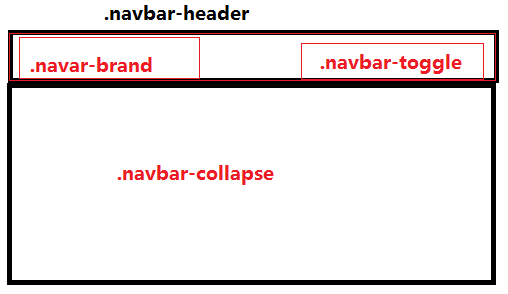
今天学习的内容

1:复习的昨天学习的内容

2:作业

3:今天的目标

3.1:Bootstrap 组件--导航栏--(难点)



响应式导航栏层次结构

(1)最外层 .navbar (.navbar-default 背景色)

(2)里面 .container

(3).navbar-header

.navbar-brand logo

.navbar-toggle 汉堡包

(4).navbar-collapse 折叠

.navbar-toggle媒体查询:屏幕宽度大于768px.就隐藏

.navbar-collapse媒体查询:幕宽度大于768px,

导航条颜色

浅色底深色字 .navbar-default

深色底浅色字 .navbar-inverse

导航条定位方式

相对定位:默认

固定定位:.navbar-fixed-

3.3:定制

3.4:样式语言分类

(1)静态样式语言

CSS可以直接被浏览器解析,但作用为一门语言，css并不完整,

缺少编程序语言基本条件：(数据类型;变量;运算符;函数;继承)

导致样式的**可维护性**差.

(2)动态样式语言

第三种:Sass/Less/Stylus,在CSS的基础上添加态编程序所必须的特性，如数据类型;变量;运算;函数;提高可维护性.

需求注意:浏览器只能识别CSS，所有动态样式语言必须转换CSS，

该过程称为"编译(Compile)"

3.4:Less样式语言

官网: http://lesscss.org

中文:http://lesscss.cn

Less是一门CSS预处理语言--它扩展CSS语言，增加，变量;函数;等性.使用CSS更易维护和扩展.

LESS两种用法

(1)在客户端使用LESS---学习(生产环境不推荐)

编写一个 x.less

在html中引入x.less文件,同时再引入一个less编译程序

less.js

客户端请求html,下载x.less以及less.js,在客户端运行

js文件把less编译 css

(2)在服务器端Less--必需掌握

编写x.less

安装less编译环境，执行x.less文件编译 x.css

再编写html文件,引入编译后x.css

客户端请求html, 下载 css文件

3.5:Less语法学习

(1)Less支持所有CSS语法

(2)Less支持多行/单行注释,但只有多行注释编译css文件中,

---推荐使用单行注释

(3)Less支持 变量(Variable)概念

定义:@变量名:值;

使用:background-color:@变量值

(4)Less支持变量和常量运术运符

+ - \* / %

3.x:Bootstrap 组件(插件)--图片轮播

作业 1：学子商城首页bootstrap

作业 2: 创建权限管理表

### Day05

今天学习的内容

1:复习昨天知识重点

2:作业

3:今天的目标

3.1: BootStrap js 插件

3.2:Less--(重点)

样式语言

(1)静态样式语言:CSS

(2)动态样式语言:less;sass;stylus;

less使用方法

(1)在客户端使用less (学习)

x.less x.html less.js(编译程序:x.less->x.css)

(2)在服务器端使用less (必需掌握)

x.less x.html node.js->x.less->x.css

3.3:less语法

(1)less支持所有css语法

(2)less 多行/单行: 只有多行注释会编译css文件中--推荐单行注释

(3)less支持变量

定义: @变量名:值;

使用:color:@变量名;

变量可以取值为任何合法样式值

(4)less支持运算符

+ - \* / %

(5)less支持选择器混入

选择器1{...}

选择器 2{选择器1}

(6)less支持参数

选择器1(@ 参数名,@参数名){.....}

选择器2{..选择器(111)}

(7)less提供一些函数

ceil()

floor();

percentage(num) 把小数转换百分比 0.5 50%

darken(color,percentage); 把指定颜色变暗

lighten(color,percentage); 把指定颜色变亮

img-width(url) 返回图片宽度

img-height(url); 返回图片高度

(9)

less中可以使用@import 实现文件包含--推荐使用

3.4:通过修改Bootstarp的less源文件实现定制

(1)瘦身: 删除不需要样式

只需要注释掉bootstrap.less 不需 @import

(2)粗粒度定制

只需要修改variables.less 中定义变量

(3)细粒度定制

修改件对应的.less文件 dropdown.less

3.5:定制编译bootstrap.less文件

(1)修改 variables.less 按钮危险颜色

(2)修改 bootstrap.less 注释不需组件 //@import

(3)copy 编译程序 lessc e:/npm/

(4)修改

4.1: 按钮颜色 btn-info-color 绿色 variables.less

4.2: 导航栏 default-背景 红色 navbar.less

(5)通过指令将 bootstrap.less编译为 bootstrap.css

a:开始->搜索程序和文件->cmd 回车

b:转换e盘 e: 回车

c:进入编译工具目录 cd npm 回车

d: lessc e:/src/bootstrap.less e:/src/bootstrap.css

(6)在html中加载新的bootstrap.css 文件

注意事项:如果出错，重新下载安装包即可

注意事项:安装环境

(1)下载并且安装 Nodejs 6.11-->8.9.0 正确

(2)下载less编译程序 lessc

3.6: Bootstrap js 插件(模态对话);

modal: 模态对话框

model: 模型

module: 模块

modal:模态:一个父窗口中打开了一个子窗口，若子窗口不关闭

父窗口无法获得输入焦点，该子窗口称为模态对话框:

window.alert()/confirm()/prompt();

<div class="modal fade" id="modal-login">

<div class="modal-dialog"> 尺寸定位

<div class="modal-content"> 背景/边框/阴影

<div class="modal-header">

<span class="close" data-dismiss="modal">X</span>

<h3 class="modal-title">登录</h3>

</div>

<div class="modal-body">主体</div>

<div class="modal-footer">尾部</div>

</div>

</div>

</div>

</div>

3.7: Bootstrap js 插件(图片轮播);

提示：轮播广告无需记忆,使用Bootlint提示功能

<div class="carousel" data-ride="carousel">

<div class="carousel-inner">

<div class="item active">

img+.carousel-captioin..

</div>

</div>

</div>

练习:

3.x:Bootstrap 写一个学子商城首页

作业1:复习nodejs

## NodeJs

### Day01

今天学习的内容

1:复习bootstrap重点知识

2:作业

3:今天的目标

3.1:(准备工作)NodeJS 下载并且安装

3.2:一道面试题 用户在浏览器中输入 www.taobao.com 直到看到页面之间发生了什么?

(1)操作系统访问网络上的DNS服务器，把域名转换IP地址

(2)浏览器获取返回ip地址并且向ip地址发送HTTP请求

(3)web服务器接收并解析请求消息,查找指定资源,可能访问数据库,构造并返回http响应消息

(4)浏览器接收并解析响应消息

(5)浏览器缓存接收响应消息，并解析和渲染响应的内容

3.3:静态网页和动态网页

静态网页:网页内容任何人在任何时间访问都是不变的.

HTML/CSS/JS/视频...

动态网页:网页内容不同人在不同时间访问可能不同的

JSP/PHP/ASP.NET/Node.JS

3.4:NodeJS 概述(重点&难点)

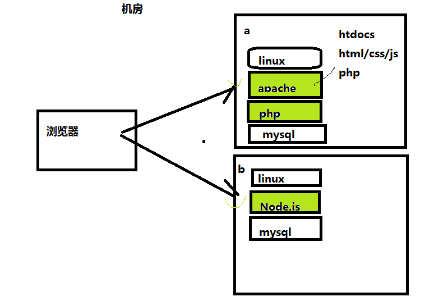
JSP=HTML+java :功能强可靠，适合大型企业级项目

PHP=HTML+PHP :简单易于，适合互联网项目

ASP.NET=HTML+C# :易用适合windows 平台

Node.JS=HTML+"JS" :性能好,适合于服务器端IO密集型项目，不适合CPU密集项目.

|  |  |
| --- | --- |
| CPU密集型项目 | 滴滴打车,天气预报 |
| IO密集型项目 | 输入输出(学子商城) |



概念:Node.JS不是JS,是一种服务器开发平台,它的竞争对手是PHP/JSP/ASP. 历史上第一次有一种语言可以通吃前后台.

官网:www.nodejs.org

版本:(16初)0.12 (16年中)4.x (16底)6.x

LTS:Long Term Support 长期稳定支持版本

Current:最新版本

安装时注意事项:安装目录(不要有中文,不要有空格)

3.5:NodeJS 运行模式

(1)交互模式--一般用于临时测试

REPL:Read Evaluate Print loop,输入一行代码执行一行

注意:交互模式自带输出功能,不必写 console.log();

node 回车 #进入交互模式

.exit #退出交互模式

(2)脚本模式--正式项目中使用的方式

把要执行的所有语句，编写在一个文本文件中(后缀任意，没有也行),一次性提交给node解释器.

node 完整路径名/x.js 回车

提示:只要安装完成Node.js重启一个WebStorm,ws可以自动检测node.js ,右键运行

练习:创建01.js文件，打印九九乘法表，以脚本模式运行,

ws中运行

3.6:如何自学一门语言 Node.JS

(1)了解背景

---百度百科 Node.JS (IO非阻塞/事件驱动)

(2)搭建开发环境

下载并安装nodejs.msi

(3)数据类型--(多)

|  |
| --- |
| 前端JS数据类型  (1)基本/原生/值类型  number/string/boolean/null/undefined  (2)引用类型  ES对象;String;Math;Date;Object;Function;RegExp;Error  *BOM:window;document;history;screen;location;event*  *DOM:Node/Element/Attr/*  用户自定义:{} |
| 后端Node.JS数据类型  (1)基本/原生/值类型  string/number/boolean/null/undefined  (2)引用/对象类型  ES:String/Number/Boolean/Math/Data/RegExp/Object..  用户自定义对象:{}  NodeJS原生第三方对象库:几百 十多万 |

(4)变量和常量

var age = 20;

const PI = 3.14;

(5)运算符

算术运算符 + - \* / % ++ --

比较运算符 < > <= >= == === !==

逻辑运算符 && || !

位运算符 << >> ^= !

三目运算符 ? :;

赋值运算符 += -= \*= /+ %=

特殊运算符 typeof .

(6)逻辑结构

循环: while do..while for(;;); for(..in..) for(..of..)

练习:03.js 声明一个保存5个学生成绩的数组,使用三种

循环依次打印出每个成绩值

选择结构:if..else switch..case

练习:创建一个变量 var path = "/path";使用二种选择结构,判断path的值为哪种(/index,/search/,/login)，调用不同的函数

(7)通用小程序

九九乘法表,100以内质数,数组排序...

练习:打印出100以内质数 14:34--14:36

(8)函数和对象

(9)常用组件,第三方工具

(10)实际小项目

3.7:Nodejs中特有的概念--模块(重点&难点)

|  |
| --- |
| Modal:模态框 Model:模型 Module:模块 |

一个web项目功能可以分为很不同的"模块":如：商品管理模块，用户管理模块，支付模块，促销模块...

Node.JS按照不同的功能，可以把函数，对象分别保存不同文件或目录下“文件或目录称为模块”

Node.js为每个模块创建独立构造函数，解释会(自动)添加到每个.js文件中.

05.js

(function (exports, require, module, \_\_filename, \_\_dirname){

exports:{} 用于声明向外部出自己成员

require:fn 用于导入其它的模块，并且创建指定模块对象

module:当前模块对象

\_\_filename

\_\_dirname

var i = 10;

console.log(document);

//自己编写程序内容

});

每个模块都可以使用自己的require()函数引入另一个模块--底层本质就是创建指定模块的一个对象实例.

var m = require("./模块文件名");

每个模块可以使用exports对象向外导出/公开一些自己的成员供其它模块使用

exports.成员名 = 成员值.

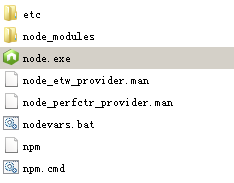
|  |
| --- |
| Node.JS中有二个对象可以向外公共数据;  exports和module.exports  Node.js底层代码: exports = module.exports;  所以：  若只是给exports对象添加新成员，则等于给module.exports添加新成员,但是若改了exports指向，则不同产生实质作用.  exports.age = 19;  module.exports.age = 19;  但是如果:  exports = {name:"tom",age:19} |

练习:创建一个文件模块:08\_circle.js 其中定义常量PI，声明一个方法getSize(r),根据传入圆形的半径，返回其面积，以及一个方法 getPerimiter(r);根据传入圆形半径，返回周长，再创建一个应用程序 09\_app.js 引入08\_circle.js模块，调用其公开的两个成员方法.

练习:创建一个模块:10\_ArrayUtil.js包含一个方法sum(arr),返回指定数组中所有数值的和，一个方法avg(arr)返回指定数组中的所有数组的平均值，使用module.exports导出上述两个成员

再写11\_app.js 模块，调用其公开的方法

3.8:Node.js模块分类



(1)Node.JS官方提供的模块--安装在解释器内器

require("模块名");

(2)第三方编写模块

(3)用户自定义编写模块

exports.x = require("./模块文件名");

3.9:Node.js 预定义模块---Global

该模块提供可以直接使用，而无需require(global);

console

setInterval();

setTimeout();

setImmediate(fb); 等价于setTimeout(0,fn);

3.10:练习:仿写Node.js提供一模块:URL

创建模块文件 MyUrl,向外导出一个方法 resolve(url);

该方法接收参数:

形如: http://www.jd.com:8080/ad/index?uname=tom&pno=3

返回对象:形如

{

protocol:"http",

server:"www.jd.com",

path:"/ad/index",

query:"uname=tom&pno=3"

}

再编写一个主模块:引入上述模块并调用

运算出错: nodejs路径不正确

(1)File->Setting->Lanaguages->Node.js and NPM

(2)Node interpreter:[c:\program Files/nodejs/node.exe][...]

作业1:授权己有拥角色/未拥有角色

作业2:url模块: query {uname:"tom",pno:3}

### Day02

今天学习的内容

1:复习昨天知识重点

2:作业

3:今天的目标

3.1:目录模块(重点与难点)

3.2:自定义模块两种形式

(1)文件模块

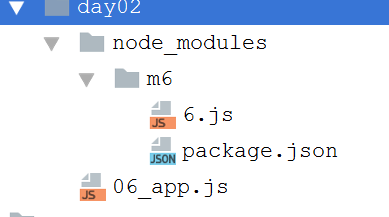
创建js文件，如m3.js 导出需要公开数据，其它的模块

require("./m3");模块

(2)目录模块

方式一:创建一个目录,必须名为node\_modules,其中再创建目录模块，假设目录名m6,其中创建描述文件(名称固定)package.json,其中声明main属性指定默认执行启动js文件,

假:6.js 其中导出需要公共的数据。其它的模块 require("m6");



注意结构

node\_modules 所有目录必须放在此目录下

m6 目录模块

6.js 启动js文件(名称可修改)

package.json 目录模块描述文件(名称不可修改)

{

"name": "m6", m6 模块名称

"version": "0.0.1", 目录模块版本

"author":"tao", 作者

"main":"./6.js" 启动程序

}

练习:使用以上的方式:创建两个模块,

circle/rectangle都对外公开两个方法

size()/perimeter()返回指定图形的面积和周长

最后在最外层的主模块中引入上述两个模块

circle.size(r);

circle.permeter(r);

rectangle.size(w,h);

rectangle.permeter(w,h);

方式二:创建一个目录/假设名为 m4,其中创建名为index.js(固定不变)文件，导出需要公共的数据，其它的模块可以 require("./m4");

方式三:创建一个目录，假设名为m5,其中创建package.json文件，其中声明main属性指定默认启动js文件,如:5.js,其中导出公开数据，

其它的模块可以require("./m5");

|  |
| --- |
| 自己尝试:若有个文件名为m7.js，还有目录  m7/index.js,还有 node\_modules/m7/index.js  最后的app.js  require("./m7"); 引入的哪个模块  require("m7"); 引入的哪个模块 |

小结:nodejs目录模块

(1)node\_modules/m5/package.json/index.js+app require("m5");

(2)m6/index.js + app require("./m6");

(3)m7/index.js/package.json + app require("./m7");

3.3:NPM 包管理器

Node Package Manager;Node.js 管理第三方模块工具,可用于下载,

更新,删除,维护包.

npm 工具默认 www.npmjs.org 网站下载第三方模块

npm install 包名; //安装

npm uninstall 包名;//删除

3.4:原生的模块([http/fs]重点)

3.5:Node.JS官方提供原生模块-- querystring

querystring模块用于处理HTTP请求URL中的查询字符串

var obj = qs.parse(str) ;;把查询字符串解析为JS对象

var str = qs.stringify(obj) ;;把js对象转换为查询字符串

3.6:Node.JS官方提供原生模块-- url

url模块用于解析一个URL请求地址，获取其中不同的部分

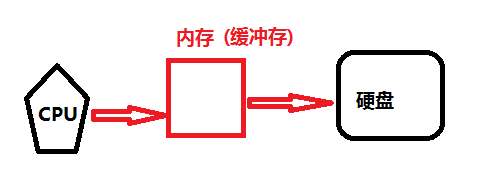
var obj = url.parse(str); 把一个URL地址解析为一个对象

var obj = url.parse(str,true); 把一个URL地址解析为一个对象,并把其中查询字符串也解析为对象

3.7: Node.js官方提供原生模块--Buffer

Buffer:缓冲区，本质是一块内存区域，用于暂存以后要用的数据

(可能是数字，字符串，二进制图片/音频视频),该区域称为缓存.toString()



(1)创建一个指定大小缓冲区

var buf1 = Buffer.alloc(缓冲区大小(字节));

示例:

var buf1 = Buffer.alloc(1024\*2);

(2)使用数字数组创建一个缓冲区

var buf2 = Buffer.from([1,2,3,4]);

(3)使用一个字符串创建一个缓冲区

var buf3 = Buffer.from("abc");

(4)把一个缓冲区的数据转换字符串

var str = buf3.toString();

3.8: Node.js官方提供原生模块--fs---重点

fs模块提供了对文件系统中的文件/目录进行增删改查，读写的功能

同步:二个程序一个等待一个工作

异步:二个程序二个都在工作

同步操作:

var data = fs.readFileSync(file); //同步读取数据

fs.writeFileSync(file,str/buf); //同步写数据

#写操作:默认操作先清除原来的数据/添加

fs.appendFileSync(file,str/buff); //同步追加

#"\r\n" 换行

练习一:使用同步的方法实现 3.txt 文件复制为3.backup.txt

练习二:使用同步的方法实现 p3.png 文件复制为p3.backup.png

练习三:使用同步的方法实现 am.docx 文件复制为am.backup.docx

练习四:使用同步的方法实现birds.mp4 文件复制为birds.backup.mp4

异步操作:

fs.readFile(file,function(err,data){ data}); //异步读取

fs.writeFile(file,str/buf,function(err){err}); //异步写数据

fs.appendFile(file,str/buff,function(err){err});//异步追加

练习一:使用异步的方法实现 3.txt 文件复制为31.backup.txt

练习二:使用异步的方法实现 p3.png 文件复制为p31.backup.png

练习三:使用异步的方法实现 am.docx 文件复制为am1.backup.docx

练习四:使用异步的方法实现birds.mp4 文件复制为birds1.backup.mp4

3.9: Node.js官方提供原生模块--http---重点

http模块可用于编写基于HTTP协议客户端或服务器端程序(浏览器-apache)

用http模块编写一个web服务器

(1)接收客户端http请求

(2)解析客户请求消息

(3)读取客户请请求资源

(4)向客户端发送http响应消息，主体就是客户端请求资源

示例:

var server = http.createServer(); //创建服务器对象

server.listen(8080); //绑定监听端口

端口:1-65535 建议 1024以后

server.on("request",function(req,res){

req:请求对象{请求浏览器数据}

res:响应对象{响应并且发送给浏览器数据}

});

作业1：点击操作

作业2：nodejs http模块创建服务器,

如果客户请求 /index.html

读取当前目录下index.html 返回客户端浏览器

### Day03

今天学习的内容

1:复习昨天知识重点

2:作业

3:今天的目标

3.1:第三方的模块(第二节课开网|重点)

安装5个第三方模块{e:/}

npm i express

npm i body-parser

npm i mysql

npm i cookie-parser

npm i express-session

npm i cors

3.2:nodejs 官方模块http

练习：静态web服务器:创建web服务器，接收客户端请求，

一律向客户端输出一句 <h3>hello nodejs</h3>

练习:动态web服务器:创建web服务器，接收客户端请求,

若客户端请求 /index.html则向客户输出 index.html内容

若客户端请求 /其它 则向客户输出

<h1>404 您查找的数据不存在</h1>

|  |
| --- |
| Error: listen EADDRINUSE :::8080  错误:原因 端口冲突(二个程序在工作时使用同一个端口)  解决:停一个程序或者修改程序端口 |

3.3:nodejs 第三方模块 mysql(此模块专用于访问与操作mysql数据库服务器)

为了精简node.js解释器、官方没有提供任何数据库相关模块,必须使用npm工具下载第三方模块，在www.npmjs.org 上搜索关键字 mysql ,可以得到很多相关模块。

使用工具下载安装

npm i mysql

mysql数据库:查询速度比较快更新慢下

数据库存储三种方案

(1)大量数据保存 mysql数据库中

(2)NOSQL解析方案:Redis(内存级别数据库)

(3)文件存储:试卷

3.4:nodejs 第三方模块 mysql-使用步骤

(1)创建到数据库的连接(保证mysql启动成功/第三方模块加载成功)

const mysql = require("mysql");

var conn = mysql.createConnection({...})

(2)创建一条SQL语句并且发送SQL

var sql = "...";

conn.query(sql,function(err,result){});

(3)关闭连接

conn.end();

3.5:nodejs 第三方模块 mysql-网络攻击手段

(1)SQL注入

利用:sql语句中弱点，添危害数据放在sql语句中.

var fname = "';drop table mysql.user ";

var fname = "';select into c:/aa.dat ";

var uname = "' OR 1=1 #";

var upwd = "";

解决方案一:占位符

var sql = "INSERT INTO user VALUES(null,?,?)";["tom","123"]

解决方案二:项目工程分为二个组成部分:

前端业务<----->核心功能(mysql);

(2)XSS 脚本攻击

var name = "<script>document.all.disabled=''</script>";

用户名 [jerry]

解决方案:

所有用户输入的数据过滤:drop table;

练习:仿写上述代码，使用node.js向 web1708acl02/t\_user

[uid/uname/upwd] 表添加一行新的记录.

练习:仿写上述代码，使用node.js向 web1708acl02/t\_user

uid=3 upwd=123456

练习:使用http和mysql模块

\*创建动态web服务器,接收如下请求URL

\*/register.html 服务器向客户端返回register.html内容

/register.do 接收客户端提交uname/upwd

GET req.query.uname

保存mysql t\_user

/login.html 服务器向客户端返回 login.html 内容

/login.do 接收客户端提交 unname/upwd

/userlist.html

/userlist.do 接收客户请求，发送用户列表

404.html

作业1:/userlist.do

作业2:授权

### Day04

今天学习的内容

1:复习昨天的内容

2:作业

3:今天的目标

3.1:第三方模块 express (重点&难点)

使用官方提的http模块可以创建一个web服务器应用，但是此模非常底层，要处理各种情形,推荐使用http模块进一步封装模块--express -第三方模块,V4版本中定义:express是一个请求处理工具，用于接收客户的请求消息，返回响应消息.

可以处npmjs.org 下载

npm i express

http://www.expressjs.com.cn

使用方式:

(1)在模块中引入express模块

const express = requre("express");

(2)创建 express对象

var app = express();

(3)创建服务器对象，绑定监听端口

var server = http.createServer(app);

server.listen(8080);

3.2:第三方模块 express "处理GET请求

app.get(url,(req,res)=>{

//昨天:200;text/html;write;end

res.send("向客户端返回html字符串");

//昨天:200;text/html;fs.readFile,write.end

res.sendFile(\_\_dirname+"文件名");

res.json(obj); //将obj转换json字符串输出客户端，并修改head

});

3.3:第三方模块 express "处理GET请求接收参数

(1)GET /user?uid=10&loc=bj

app.get("/user",(req,res)=>{

//express为每一个 req对象添加query属性

req.query.uid;

req.query.loc;

});

(2)GET /user/jsj/61

app.get("/user/:type/:id",(req,res)=>{

//express为req添加属性params属性

console.log(req.params.type);

console.log(req.params.id);

});

练习: 创建express服务器,配置 mysql 模块

GET /role/3 向客户端输出3号角色信息.json

(3)POST /user

app.post("url",(req,res)=>{

});

如何接收参数

a:req.on("data",(buf)=>{

var obj = qs.parse(buf.toString());

});

b:单独加载一个(第三方)模块(解析post传递参数)

const bodyParser = require("body-parser");

将模块添加中间件列表

app.use(bodyParser.urlencoded({extended:false}));

在post中获取参数

app.post("/user",(req,res)=>{

var uname = req.body.uname;

});

(3)app.put(url,(req,res)=>{})

(4)app.delete(url,(req,res)=>{})

练习:

(1)编写 SQL：创建数据库

dangdang book(bid,title,price,pubDate)

user(uid,uname,upwd)

添加4条记录

(2)创建express服务器,接收如下请求

/booklist.html

/login.html

/register.html

/jquery.js

public

(3) /book 获取图片列表

(4)/login 完成登录

(5)/reg 完成注册

(6)/del 完成删除图书

作业1:node js session cookie

作业2:授权

### Day05

今天学习的内容

1:复习昨天的内容

2:作业

3:今天的目标 [taonetwork]

3.1:图书管理(练习)

a:添加三个中间件(session;cookie;cors)

session(会话):过程

session对象:保存数据[在当前会话需使用数据,uid,权限列表]

b:nodejs使用session

(1)express-session 第三方模块(中间件)

(2)cookie-parser 第三方模块(中间件)

c:配置

cookie: app.use(cookieParser());

session:app.use(session({

resave:false, #每次请求是否需要重新设置cookie;

saveUninitialized:true #无论是否有cookie,设置标记

connect.sid

}));

d:使用

req.session.uid=10; 向session对象添加属性

req.session.uid; 获取

完成:(坑)

1:如果用户登录成功将uid保存session.uid对象

2:获取图片列表 session

3.2: cors 跨域访问

概念:从一个域名下程序访问另一个域名程序

示例: www.a.com/1.js ---> www.b.com/1.php

跨域:

(1)域名不同

(2)端口号不同

http://127.0.0.1:80/1.js -> http://127.0.0.1:8080/login

(3)协议不同;

http://127.0.0.1/1.js -> https://127.0.0.1/login

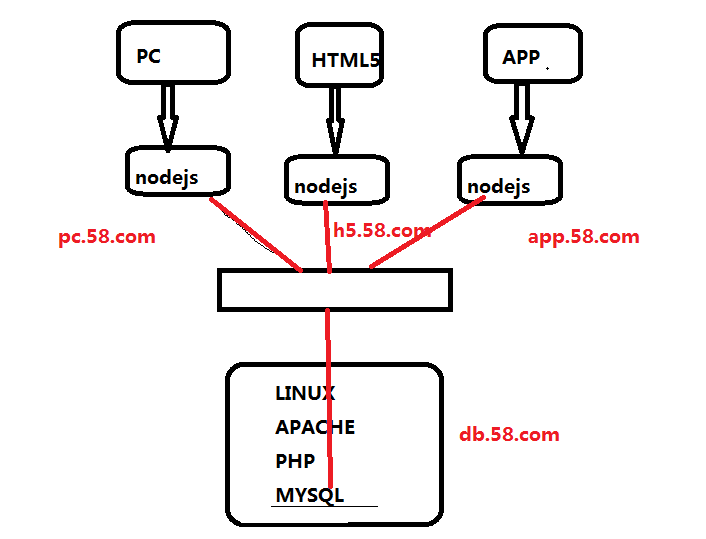
(4)ip与主机名

http://localhost/1.js -> http://127.0.0.1/login

跨域访问:

允许:js/css/img

不允许:xhr(ajax)



项目升级:[改动大]

1:将node.js/public/所有html/js/css 移动位置

apache/htdocs/book

2:nodejs /login /register /booklist

3:访问方式:apache<->node.js

http://127.0.0.1/book/login.html

-->ajax->

http://127.0.0.1:8080/login

4:重点改动程序

4.1:加载对应模块 cors

4.2:配置模块:只允许哪个域名下的资源来访问我

你访问我程序请带cookie

5:补坑:

原因:跨域cookie中保存数据丢失

添加如下的选项:指定ajax发起请求一定要带上cookie值

解决:apache/html/

$.ajax({

xhrFields:{withCredentials:true}

});

3.3:nodejs 项目(学子商城一部分功重写)

3.4:nodejs express中间件与路由(重点&难点)

### Day06

今天学习的内容

1:复习近几个阶段的知识重点

2:项目

3:今天的目标

3.1:nodejs 中间件 --(重点&难点)

Express;中间件(static);

中间件(Middleware)是一个函数(session;cors;cookie)

作用:它可以访问请示对象(request object) 也可以用于访问响应对象(resonse object)和web应用请求-响应中对象.

中间件的功能包括

(1)执行任何代码

(2)修改请求和响应对象

(3)终结请求-响应循环

(4)调用下一个中间件

中间件分类

(1)应用级中间件(拦载器)

(2)路由级中间件 路由=请求路径+处理方法+处理函数

app.get("/user",(req,res)=>{});

(3)错误处理中间件

(4)内置中间件(express static)

(5)第三方中间件(session;cookie; cors)

中间件语法:

var app = express();

#req 请求对象

#res 响应对象

#next 调用下一个中间件

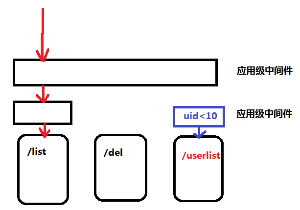
app.use(function(req,res,next){

console.log("中间任何代码");

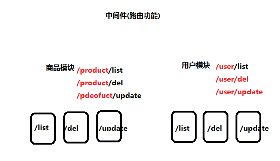
next();

});

应用级中间件



路由级中间件



//统一定义产品路由中间件

var router = express.Router();

router.get("/list",(req,res,next));

router.get("/del",(req,res,next));

router.get("/update",(req,res,next));

//统一定义用户路由中间件

var router1 = express.Router();

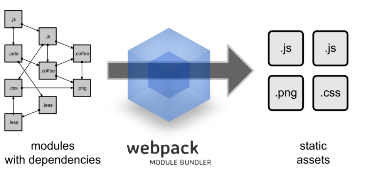
router1.get("/list",(req,res,next));

//加载应用

app.use("/product",router);

app.use("/user",router1);

3.2:webpack



webpack是一款加载器和打包工具:它能把各种资源(js/less/coffect/图片)等作为模块来使用处理并且打包.

安装和配置

1:创建测试项目 e:/webpack-test

2:进入项目目录中 e:

mkdir webpack-test

cd webpack-test

3:初始化项目 npm init

4:全局安装 npm install webpack -g

5:本地安装 npm install webpack --save-dev

示例:

1:创建js webpack-test/test.js

2:添加内容 hello

3:webpack test.js test.bundle.js

示例:

1:创建js webpack-test/word.js

2:添加

3:test.js 中添加文件 requre("./word");

4:webpack test.js test.bundle.js

示例

1:创建css 1.css

2:test.js 引入 1.css

3: webpack test.js test.bundle.js

错误:安装依装第三方模块(loader)

npm install css-loader style-loader --save-dev

并且修改过代码

示例:

(1)监控项止，内容实时修改

webpack test.js test.bundle.js --watch

(2)看打包过程

webpack test.js test.bundle.js --progress

(3)使用哪些模块

webpack test.js test.bundle.js --display-modules

示例:

(1)index.html 引入打包后文件

示例:

创建目录 e:/webpack-demo

dist #生成静态资源文件

node\_modules

src #源程序

script

styles

package.json 描述文件

webpack.config.js webpack配置文件

1:添加src/script/main.js

2:添加webpack.config.js

3:打包 webpack

示例:修改描述文件 package.json

添加入口

"webpack":" webpack --progress --colors "

再打包

npm run webpack

# 张众磊

## 学习计划

张众磊

bjzhangzl@tedu.cn

一、本阶段学习安排

①VueJS（3天+3天）

②Angular (4天+2天)

③webApp（3天+3天）

④react (4天+3天)

⑤PhoneGap（1天）

⑥微信 (3天)

二、约定

框架？jQuery 简化DOM操作 bootStrap 响应式的站点；框架就是一些优秀的代码，可被反复使用。

使用框架的好处：

①提高编码效率（开发速度）

②提高代码质量

③降低代码之间的耦合度。。。。

学习框架的阶段 重点不是跟着代码抄，而是：

①通过学习框架，提高工程的开发速度、代码质量

②掌握看文档、看手册的技巧

③架构把控的基本能力

要慢慢的把思维模式切换从学徒到开发者，必须要做到手勤（练习）脑勤（思维导图），否则一个月下来就是一场空。

注意：

学习框架时，代码很多，需要跟写和练习时，老师会提出来；

否则就看老师是怎么写的，不要盲目的跟着抄。

怎么才算学会一个框架的使用？

①明确框架的使用场景

②框架如何去使用（都有哪些核心的特点）

③如何自己去学习其他的框架

④源码如果有精力 可以多去花时间去了解（课堂上会讲关键点）

## Vue 项目创建流程

脚手架安装与部署项目流程

1:依赖软件 nodejs 6.xx

1.1： 下载nodejs

http://nodejs.cn/download/

https://npm.taobao.org/mirrors/node/

X86 适用 32 windows

X64 适用 64 windows

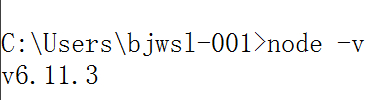
1.2: 下一步下一步

安装目录: 不要中文不要有空格

c:/program files/nodejs

1.3: 检测(windows->开始->运行->cmd)

node -v



2:安装cnpm 工具(nodejs工具 npm 安装软件)

npm install -g cnpm --registry=https://registry.npm.taobao.org

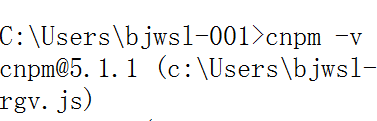
#npm 安装工具

#install 安装 uninstall 删除

#-g 全局

#ttps://registry.npm.taobao.org 淘宝镜像站点

#检测



3:安装 vue-cli 脚手架

cnpm install --global vue-cli

4:通过(脚手架)初始化项目(自动项目完整目录与配置文件)(***选做***)

#上课没初始化

vue init webpack 项目名称

示例

vue init webpack my-project

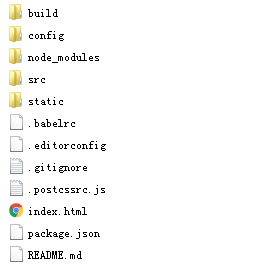
#提问 项目名称 :my-project

# 安装:vue-router? Yes/No Yes

# 安装:EsLint ? No

# No

检测：自动生成目录



#build 脚本目录

#config 配置目录(..)

#node\_modules 所有nodejs依赖工具包目录(**空**)

#src 代码目录

assets 资源目录(img/css/js)

components 组件目录

router 路由配置

App.vue 最上层组件

main.js 程序入口js

#package.json 项目描述文件

注意:tpls.zip

5:安装依赖工具包

e:

cd tpls

e:/tpls>cnpm install

注意:红色命令你输入，黑色自动出现

6:启动项目

cnpm run dev

注意：启动服务器:端口 8080

注意: 不需要apache

检测: http://127.0.0.1:8080/#/

停止服务器: ctrl+c

7:代码开发

src

8:vue 项目启动流程

在执行cnpm run dev 的时候 ，会在当前目录下找一个文件

package.json文件，启动服务器默认端口8080

找到src目录文件main.js ,创建项目唯一 new Vue();

要加载模板内容 App, App是src目录 App.vue结尾文件,

有一个 <router-view></router-view>

在src目录有一个router文件夹,该文件夹下有一个 index.js

文件:该文件是配置路由词典,指定路由地址

## VueJs

### Day01概述,双花括号,指令

一、VueJS的概述

GUI： graphical user interface 图形化用户界面

CLI： command line interface 命令行交互界面

做用户界面的框架或者库，并不一定非得是像bootStrap有着丰富的样式类，VueJS是通过组件来构造ui(user interface)的。

what？

VueJS是一个只关注视图层的渐进式的js框架。

Vue就像是React和AngularJS结婚时生的小宝宝，继承了React和Angular的一些优点

Vue.js 不支持 IE8 及其以下版本

所谓的渐进式，指的是Vue可以实现一些非常基础的操作，还可以实现一个大型的商业项目，从简单的功能，逐渐到复杂的功能，Vue都能胜任

why?

①是国人开发的，非常优秀的框架

②非常方便的中文手册

③Vue的内部实现让编码、运行更具有效率

when?

实现UI层，也可以实现整个项目

how?

方式1：

直接在工程中 引入vue.js文件

方式2：CLI

//全局安装 vue-cli

npm install --global vue-cli

//创建一个基于 webpack 模板的新项目

vue init webpack my-project

//安装依赖，走你

cd my-project

npm install

npm run dev

扩展：

SPA(single page application) 单页面应用程序，在一个完整的工程中 只有一个完整的页面，其余的都是一些代码片段。

二、VueJS的使用

1、双花括号

双花括号，叫做mustache语法（胡子语法）、interpolation语法（插值表达式）

基本语法：

<any>{{expression}}</any>

表达式是由变量和运算符（比如算术、比较运算符、逻辑运算。。。）构成。

作用：执行表达式，并将表达式的结果输出到当前元素的innerHTML中

练习：

新建一个demo03\_lianxi.html,去通过vue的语法来实现，直接在html中执行逻辑运算、三目运算，并将结果分别显示在h2\h3

2、选择功能（判断功能）

写代码有3种基本结构，分别是顺序结构、选择结构、循环结构

语法:

<any v-if="expression"></any>

作用：

根据表达式执行的结果的真、假来决定是否要将当前元素挂载到DOM

多重判断：

<h2 v-if="rightAnswer == 'A'">A</h2>

<h2 v-else-if="rightAnswer == 'B'">B</h2>

<h2 v-else-if="rightAnswer == 'C'">C</h2>

<h2 v-else>D</h2>

if(rightAnswer == 'A')

{

}

else if(rightAnswer == 'B'){}

else if(rightAnswer == 'C'){}

else{}

3、循环指令

语法1：

<any v-for="tmp in list"></any>

语法2：

<any v-for="(value,index) in list"></any>

语法3：用在遍历对象时候，需要知道是第几个属性

<any v-for="(value,key,index) in list"></any>

作用：根据集合中元素的个数 创建多个标签；让html扩展了循环的功能

综合练习：

在Vue中定义一个布尔类型的数据 isDataReady（初始化为false）,一个数组 myList(初始化一个空的数组)

在Vue的实例创建完毕，2s之后，将isDataReady修改为true，并向数组中插入3条随机的数据。

将myList中的数据 显示在ul中--》v-for

根据isDataReady来决定h2标签中的内容(值为false，显示正在加载数据，值为true，从DOM中移除调用)--》v-if

数据绑定到视图：

可以使用{{}}、指令，将数据绑定到视图，那么当数据发生变化的时候，视图会自动的更新。

思维模式1：js/jQuery DOM驱动的开发方式

思维模式2：数据驱动 ①简单 ②解耦（view和data）

扩展：

1、高内聚低耦合

耦合度：藕断丝连，不同的组件或者模块之间的关系，将降低耦合度的过程称之为解耦

聚合度：一个组件内部之间的关系

2、API

application interface 接口文档

4、常用指令

学到的指令：

v-if v-else-if v-else v-for v-on

4.1 事件绑定

<any v-on:eventName="事件处理函数"></any>

举例:

<button v-on:click="handleClick"></button>

简写：

<any v-on:eventName="事件处理函数"></any>

<any @eventName="事件处理函数"></any>

支持修饰符：

<form @submit.prevent="submitOrder"></form>

4.2 属性绑定

将变量中的值 绑定到 特定的属性上边

语法：

<any v-bind:attr="expression"></any>

简写方式：

<any :attr="expression"></any>

举例：

<img v-bind:src="imgUrl"/>

<a :href="linkUrl"></a>

<any v-bind:class="{myRed:expression}"></any>

根据表达式执行的结果的真假 来决定是否要将myRed的样式类 添加到该元素上

<any v-bind:style="{backgroundColor:myColor}"></any>

4.3 v-show v-html

<any v-show="expression"/> 根据表达式结果的真假来决定是否要显示

<any v-html="expression"/> 根据表达式中字符串所对应的标签构成的代码片段 直接替换当前元素的innerHTML

4.4 v-model

双向数据绑定:

方向1：将数据 绑定 到视图

方向2：将视图中用户操作的结果 绑定 到数据

（多数情况下 都是指的表单的操作input/select/textarea/...）

<formElement v-model="isUserAgree"/>

实现一个双向数据绑定

综合练习：

实现一个简单的分页功能

demo13\_lianxi.html

创建一个Vue的实例，初始化一个数组[1,2,3,4,5],

将数组显示如下效果：

上一页按钮 1 2 3 4 5 下一页按钮

功能如下：

高亮：蓝色的背景色

①默认第1页高亮显示

②点击上一页或者下一页 要求对应的页码高亮显示，如果已经到了边界值，禁用对应的翻页按钮

③点击某一页，该页码高亮显示

提示：

需要有个变量来记录当前选中的第几页

v-for v-bind v-on {{}}

### Day02指令,过滤器,组件

自定义指令

自定义过滤器

组件的创建和使用

复合组件的基础知识

一、自定义指令

内置指令：

选择 v-if/v-else-if/v-else/

循环v-for/

绑定 v-on/v-bind/v-model

显示隐藏v-show

更新HTML v-html

自定义指令？

内置的指令在很多时候 比较偏向于一些工具性的指令，比较通用一点；但是如果一旦需要和具体的业务相关，有可能就无法满足需求，就得学会如何自定义指令

1、自定义指令的创建

new Vue({

directives:{

change:{

bind:function(){

//当指令成功调用后 会执行的一个方法

}

}

}

})

2、自定义指令的使用

<any v-change></any>

demo:当指令被调用时候，想要修改调用指令的元素 中的innerHTML为 hello world

3、自定义指令的进阶知识

自定义指令的方法：

bind:首次绑定

update:当通过指令传递的参数有变化时候

unbind：解除绑定

方法是有参数的：

el:调用指令的元素

bindings:是一个对象，rawName/value/oldValue

规范：指令的命名建议遵循小驼峰式的命名方式，指令在调用时通过烤串式写法（要记得前面加上v-）

练习：

创建一个demo02\_lianxi.html

要求通过Vue去创建一个自定义指令 用来修改背景色为所传递的参数

二、过滤器

过滤器：实现数据的筛选、过滤、格式化

在VueJS的1.\*都是有内置的过滤器的，但是2.0之后只保留了一个自定义过滤器的方式。

1、过滤器的创建：

过滤器的本质：是一个有参数，有返回值的方法

new Vue({

filters:{

//value是通过管道传递给过滤器的数据

customFilter:function(value){

return 处理后的结果

}

}

})

2、过滤器的使用：

<any>{{ expression | customFilter}}</any>

3、过滤器中如何实现参数的传递和接收

接收:

new Vue({

filters:{

//value是通过管道传递给过滤器的数据

customFilter:function(value,arg1,arg2){

return 处理后的结果

}

}

})

传递：

<any>{{ expression | customFilter(arg1,arg2)}}</any>

扩展：

1、过滤器支持多重过滤

<any>{{expression | filter1 | filter2}}</any>

2、一个优秀的开源项目

https://github.com/wy-ei/vue-filter

练习：

创建一个demo05\_lianxi.html

要求通过vue创建一个过滤器，用来实现性别的转化

1 男 boy

0 女 girl

{{ 1 | sexChange(true)}} -->男

{{ 1 | sexChange(false)}} -->boy

三、组件

让webStorm支持es6的高亮显示：

file-->settings --> languages & framework --> javaScript -->ecma6

当掌握了Vue中的基础语法（双花括号、指令、过滤器），具备了写项目的基础技能，如果项目中需要反复用到一些内容，可以使用组件来完成内容的封装和复用

1、组件：可被反复使用的，带有特定功能的视图

可以将基于组件化的开发方式，理解为拼积木，每一个组件都是一个小积木，拼在一样构成整个应用程序

2、组件创建和使用

全局组件：

2.1 创建

Vue.component('my-component',{

template:'<h1>hello</h1>'

})

2.2 调用

直接将组件的名称作为普通的html标签一样去使用

<my-component></my-component>

局部组件：

2.1 创建

new Vue({

components:{

'my-component':{

template:'<h1>it is a header</h1>'

}

}

})

2.2 使用

<my-component></my-component>

注意事项：

1、组件不能直接返回多个要渲染的元素

将要渲染的元素放在同一个容器中，比如div

2、组件允许嵌套另外一个组件，换句话说，就是允许在一个组件中 调用另外一个组件，比如：

<my-main>

template:`<div>

<my-header></my-header>

<my-content></my-content>

<my-footer></my-footer>

</div>`

3、全局组件和局部组件的区别

通过Vue.component创建的组件是全局组件，它可以直接在任何一个组件或者模板内容中去使用

通过new Vue({components:{}})创建组件的组件是局部组件，它只能直接用在父模板中

4、提到组件，想到封装和重用，其次想到组件树（帮助理清楚一个应用程序的组件的基本架构）

练习：组件的创建和使用

demo09\_lianxi.html

创建全局组件的方式来完成一个登录页面：

my-login

my-login-button

my-register-button

my-username-input

my-userpwd-input

3、扩展

Vue.directive('myDirective',{

bind:function(){},

update:function(){},

unbind:function(){}

})

Vue.filter('myFilter',function(value){

return '处理后的结果'

})

创建组件的其它方式：

①指定一个script标签，并指定type='text/x-template',并指定一个id="myTemplate"

②在script标签中，指定模板内容

③在组件指定模板内容，可以直接：

Vue.component('my-component',{

template:'#myTemplate'

})

四、组件的生命周期

分为4个阶段：

create / mount / update /destroy

将这个生命周期中的方法 称之为钩子函数。

create:

初始化数据，准备工作

beforeCreate

created

mount:

和挂载到DOM树相关

beforeMount

mounted

update:

当数据发生变化，可以根据业务逻辑实施一些处理

beforeUpdate

updated

destroy:

清理工作

beforeDestroy

destroyed

知识点总结：

自定义指令

自定义过滤器

组件的创建和使用

组件的生命周期（create/mount/update/destroy）

每日一练：

创建一个demo13.html，

在该文件中，创建一个全局组件，在全局组件中渲染一个h1的标签

要求：在组件挂载完毕之后，启动一个定时器，修改数据，这个数据可以改变h1标签的透明度（0~1 0~1）

①组件创建和使用

②生命周期的钩子函数

③v-bind:style="{opacity:opacityValue}"

### Day03组件的计算属性,监听,通信

Vue组件中计算属性和监听属性

组件间通信

（基本语法+组件间通信+router+http）

一、Vue中常用属性

1、计算属性

①创建一个计算属性

new Vue({

computed:{

getNowCount:function(){

return \*\*

}

}

})

②调用计算属性

<any>{{getNowCount}}</any>

③计算属性与普通的方法的区别

计算属性是有依赖缓存的，如果依赖的数据没有发生变化,即使调用了计算属性，也不会执行方法

而普通的方法就不同了，只要是调用，肯定会执行

2、监听属性

如果希望在用户进行表单操作时候，可以设置一个监听属性

①在表单元素中指定v-model

<input type='text' v-model="myValue"/>

②在input所属的组件中 指定一个watch属性

Vue.component('',{

watch:{

myValue:function(){

//当数据中的myValue发生变化时，将会自动的执行该方法

}

}

})

二、Vue中组件间通信

一个基于Vue的项目，有可能会很多的组件，组件之间难免需要进行数据的传递，比如父组件 传数据 给子组件；子组件传数据给父组件等，需要用到组件之间的通信处理方式

1、props down

通过属性传值

①配置子组件的props

Vue.component('son',{

props:['userName']

})

②父组件中调用子组件的时候，通过属性把值传递过去

Vue.component('father',{

data:function(){

return {

uName:'zhangsan'

}

}

template:'<son :userName='uName'></son>'

})

练习：实现一个登陆

创建的组件有3个：my-button/my-input/my-login

2、events up

事件的绑定

事件的触发

通过事件方式去传值的基本步骤：

①在父组件中 定义一个带有参数的方法

Vue.component('father',{

methods:{

rcvMsg:function(msg){

//msg就是希望接收到的数据

}

}

})

②在父组件中 调用子组件的时候 ，绑定一个自定义事件处理函数

Vue.component('father',{

methods:{

rcvMsg:function(msg){

//msg就是希望接收到的数据

}

}，

template:'<son @customEvent="rcvMsg"></son>'

})

③在子组件中，触发对应的事件

Vue.component('son',{

//触发事件

//this.$emit('customEvent','发送的数据');

})

练习：

创建一个新的文件demo06\_lianxi.html

main-component：是由header-component和一个列表构成的;

header-component ：是由一个按钮构成的

要求：

在header-component中有一个数据:isUserLogin（布尔类型的数据）

在点击按钮时，会修改数据为相反的状态

main-component希望能够根据header-component中的isUserLogin的真假来决定列表是否要显示

3、ref

ref(reference)可以给一个组件指定一个引用的名称，父组件可以根据该引用的名称得到它的实例对象，进而进行操作。

基本步骤：

①在父组件中调用子组件时 指定ref属性

Vue.component('father',{

template:'<son ref="myRef"></son>'

})

②根据应用的名称得到子组件的实例

this.$refs.myRef.属性/方法

4、$parent

可以直接在子组件中，通过this.$parent得到父组件的实例对象，从而进行具体的操作

$refs和$parent可以直接打通父子组件的通信，但是提高了耦合度，对于组件本身的封装并不是一个好事。

建议使用props down,events up

5、兄弟组件通信

三个步骤：

①创建

var bus = new Vue();

②在需要接受数据的地方，绑定事件

//通过bus给绑定一个事件处理函数

bus.$on('toXiongEr',function (msg) {

console.log('熊二收到的数据是'+msg);

});

③在需要发送数据的地方，触发事件

//触发事件并传值

bus.$emit('toXiongEr','光头强又来偷木头了');

综合练习：

demo09\_lianxi.html

需要2个组件类: chat-room, my-input

my-input：label+input+button

chat-room: ul+my-input+my-input

功能：在点击my-input中的button时，将数据发送给chat-room,chat-room接受数据并显示在无序列表中

props down ,events up

### Day04 SPA, 路由(VueRouter)

（基本语法+组件+）

练习：

①先去创建各个组件

②实现添加功能

③实现删除功能

点击todoitem中的delete按钮时候，从todolist中删除指定下标的元素

todoitem（触发） --index-> todolist（绑定）

events up 绑定和触发

组件间通信：

兄弟组件间通信 bus

props down

events up

SPA(single page application)

http(Vue如何实现与远程服务器端的交互)

一、Vue中如何来实现一个SPA应用程序

1、SPA的基础知识

SPA：Gmail邮箱

在整个应用程序有一个完整的页面，这个完整的页面可以根据地址栏中url的不同，加载代码片段进来，从而显示给用户

SPA的工作原理：

①解析地址栏中的页面地址和路由地址

②加载基础页面地址

③根据路由地址，从路由词典中找要加载的模板页面地址

④发起ajax请求，将加载来的模板页面 插入到 指定的容器中

2、VueRouter

Vue中VueRouter是用来实现SPA

基础知识：

路由模块用来建立起url地址和模板页面的映射关系，支持路由之间的导航、参数传递、嵌套、守卫。。

SPA：有容器 ，有组件 ，配置路由词典

2.1 通过VueRouter来实现一个SPA的基本步骤：

①引入对应的js文件

script src=js/vue-router.js

②指定容器

<router-view></router-view>

③创建业务需要用到的各个组件

var LoginComponent = Vue.component('',{})

④配置路由词典

path/component/routes/router

//创建路由词典

const myRoutes = [

{

path:'/myLogin',

component:LoginComponent

},

{

path:'/myRegister',

component:RegisterComponent

}

]

//配置路由模块中的路由词典

const myRouter = new VueRouter({

routes:myRoutes

})

//Vue的实例去使用配置过的路由模块

new Vue({

router:myRouter

})

⑤测试路由地址是否能够正确访问

修改地址栏中的路由地址，测试 是否能够按照配置 正确的加载对应的组件

2.2 VueRouter实现路由导航的常见方式

①直接修改地址栏

②js

this.$router.push('路由地址')

③自定义标签

<router-link to="路由地址"></router-link>

练习：

创建一个demo02\_lianxi.html

要求创建一个SPA，有3个页面（每一个页面有一个组件）构成：main-component/person-component/setting-component

在main-component有一个按钮(个人中心)，点击时 跳转到person-component；

在person-component有一个超链接(设置头像)，点击，跳转到setting-component;

在setting-component有一个按钮(查看主页面),点击-》main-component

2.3 VueRouter实现参数传递的基本步骤

①确定发送方、接收方

发送方：login

接收方：main

②发送

http://\*\*\*.html#/myMain/zhangsan

this.$router.push('/myMain/'+this.userName)

<router-link :to="'/myMain/'+userName"></router-link>

③接收

3.1 允许接收

path:'/myMain'

-->

path:'/myMain/:uName'

3.2 接收数据

this.$route.params.uName

补充：

this.$router.push('路由地址');

//可以使用go方法在历史记录中，前进或者后退

this.$router.go(-1);

练习:

新建demo04\_lianxi.html

有两个组件 list-component,detail-component

list-component:

有一个列表（准备一个数组），点击列表项，跳转到detail组件,并将列表项背后数组的下标 传递给detail组件

detail-component:接收到传递来的下标，并显示在页面中

SPA实现的基本步骤

跳转

传参

①明确发送方 接收方

发送list

接收detail

②配置接受方的路由地址

path:'/detail'

-->

path:'/detail/:itemIndex'

2.4 路由嵌套

如果一个SPA中的某个页面，根据不同的地址，嵌套其它的组件，称之为路由嵌套，而VueRouter是支持路由嵌套。

①在准备嵌套其它组件的组件中，指定一个容器

router-view

举例：比如order组件中准备嵌套Step1\Step2\Step3，在order组件的模板中 指定一个router-view

②设置准备嵌套其它组件的路由地址

order组件准备嵌套，就需要在order组件的路由地址中添加一个children的属性

{

path:'/order',

component:OrderConfirmComponent,

children:[

{path:'',component:\*\*},

{path:'step1',\*\*\*\*\*\*\*\*}

]

}

每日一练：

spa基本、跳转、传参、嵌套

要求搭建新建一个demo06\_lianxi.html。

创建一个SPA程序，在SPA中有两个基础组件

login-component/mail-component（默认加载login）;

要求在mail-component能够嵌套另外两个组件

inbox-component/outbox-component

具体要求：

login:输入框，按钮，点击按钮，将输入框中的内容传递给mail

mail：接收login传来的用户名，显示：欢迎\*\*登录

还有两个按钮，收件箱、发件箱（默认加载收件箱），在点击收件箱和发件箱时 去 加载inbox\outbox

### Day05远程跨域,项目

Vue中处理与远程服务器端的方式：

①引入vue-resource.js

②直接发起网络请求

this.$http

.get(url)

.then(function(response){

//response.data

})

练习：

准备工作：打开Apache

创建一个php文件，在访问该文件的时候，返回一个json格式的对象数组(3个对象)，对象包含name/sex/age

创建一个Vue的组件，在点击按钮时，发起网络请求，

将请求到的数据显示在无序列表

在一个使用npm创建的项目中，执行npm install会干什么事？

会去查找当前目录下的package.json,解析该json文件找到dependencies 和 devDepenedices 对应的依赖列表，安装所需要依赖的所有的包。

扩展：什么是源代码管理（版本控制工具）？

git 分布式

svn 集中式

控制源代码的版本

（14:45 - 15:10）

1、如何根据模板去启动开发服务器

npm install

npm run dev

成功打开了浏览器，

http://localhost:8080/#/

2、结束开发服务器的运行：

ctr+c

3、创建的模板项目中 如何来使用vueResource呢？

①找到src目录下的main.js

②

import VueResource from 'vue-resource'

Vue.use(VueResource);

③使用$http

this.$http.get().then()

（16：10 - 16:35）

在当前案例的基础之上，支持点击列表项，将当前列表项的index传递给detail，detail接收并显示

①明确发送(list)、接收(detail)

②配置接收方的路由地址

③准备发送

<router-link to="/myDetail/30">

{{tmp.name}}

</router-link>

④准备接收

4、Vue-cli所创建的模板项目中关键文件

src/router/index.js 路由词典的设置

src/components/\*\*\*.vue 组件

src/App.vue 模板应用中根组件

src/main.js 是当前模板应用的主要设置的js文件，配置Vue的实例，配置VueResource，引入外部的css/js

package.json 配置了当前项目所依赖的包，通过npm执行时支持的指令

npm是一个包管理器

nodejs是支持服务器的语言

5、单文件组件嵌套其它组件的方式

举例：比如 在MyList组件中嵌套MyHeader

①找到列表的组件类

在script标签中，引入要嵌套的组件类

import MyHeader from '@/components/demos/header'

②在组件类中，把要嵌套的组件类设置为局部组件

components:{

customHeader:MyHeader

}

③调用组件

<customHeader></customHeader>

or

<customHeader/>

练习：ToDoBox

ToDoInput/ToDoList/ToDoItem/ToDoBox

①在src的components去创建一个文件夹 lianxi

②在lianxi文件夹中创建4个.vue结尾的文件

ToDoInput.vue

ToDoList.vue

...

③给ToDoBox设置一个路由地址 /todobox

④实现添加功能

⑤实现删除功能

### Day06 Vue重构项目

目标：使用vueJS来重构学子商城的后台管理界面

功能列表：login/main/chart/userList/productList

步骤：

0、准备工作

将admin.zip放到c/xampp/htdocs中，解压缩到当前文件夹

1、jquery\bootStrap的使用的方式

在src目录中的assets目录下，创建css/img/js文件夹，把文件放到对应的文件夹中

在main.js中，引入，比如：import './assets/css/bootstrap.css'

注意事项：要使用bootStrap记得字体图标文件

2、创建各个组件并配置路由词典

Login.vue

Main.vue(路由嵌套以下三个组件)

Chart.vue

ProductList.vue

UserList.vue

3、Login.vue （2:15 ~ 2:50）

重构

登录页面：点击登录按钮时，将用户名和密码发给服务器，根据服务器端所返回的结果，决定是否跳转到main

与服务器端通信：

①apache mysql 已经在运行

②通信不符合预期

第一：确定要请求的url是正确的

第二: 确定该url所需要的参数个数、参数类型

第三：通过network 确定header和response是正确的

4、Main.vue （15:30 - 16:20）

setContentHeight

和元素相关的变量，需要在mounted之后进行初始化

步骤：

①将admin/index.html中的标签拷贝粘贴到Main.vue的模板中

②解决图片404的问题

③在Main.vue中去初始化数据，在mouted之后，对数据进行赋值

④在mounted之后，给每一个a，绑定单击事件处理函数

5、Chart.vue

①jquery.flot.js引入到main.js

②将之前page-content注释中 必要的标签拷贝粘贴到Chart.vue组件的template中

③定义一个用来获取随机数据的方法

④找到id为placeholder的容器，设置为画板

⑤绘制折线图

启动定时器（500ms），绘制折线图

plot.setData([getRandomData()]);

plot.draw();

## ANGULAR (NG)

### Day01

### Day02(指令，自定义指令)

VisualStudioCode是支持插件的，可以在左侧的最后一个按钮，点击弹出的一个插件的列表，

如果需要安装某个插件，在搜索框中搜索，直接点击安装，可以重新加载或者关掉vscode再打开。

如果vsCode没有安装angular插件：

①找到johnpapa.angular2-2.3.3文件夹，拷贝文件夹

②找到C:\Users\web-01\.vscode\extensions

③粘贴到当前目录

④到vscode中重新加载

Angular整体开发的思路：

用 Angular 扩展语法编写 HTML 模板， 用组件类管理这些模板，用服务添加应用逻辑， 用模块打包发布组件与服务

Angular基本语法

一、指令

Angular中指定组件类的元数据时：

selector/template/templateUrl

Vue中的指令：

循环 v-for="(tmp,index) in 集合"

选择 v-if="expression" v-else-if v-else

绑定 v-bind:src :src/ v-on:click @click / v-model

v-show v-html...

1、Angular基本语法：

双花括号

<any>{{expression}}</any>

一、Angular中的内置指令

指令一般分为两种：

结构型指令 ：修改DOM结构

属性（行为）型指令： 修改元素的外观和行为

1、循环指令

<any \*ngFor="let tmp of list;let myIndex=index">

</any>

在遍历集合的同时，创建多个调用该指令的标签

2、选择指令

<any \*ngIf="expression"></any>

将根据表达式执行结果的真假 来决定当前元素是否要挂载到DOM

3、多重分支判断

<div [ngSwitch]="answer">

<p \*ngSwitchCase="'a'">A</p>

<p \*ngSwitchCase="'b'">B</p>

<p \*ngSwitchCase="'c'">C</p>

<p \*ngSwitchCase="'d'">D</p>

<p \*ngSwitchDefault>请输入正确的答案</p>

</div>

4、属性绑定

将一个表达式执行的结果或者变量中的值 绑定到 某个属性

语法：

<any [attr]="expression/value"></any>

练习：（2:07 - 2:17）

新建demo06\_lianxi的文件夹，demo06.component.ts

有一个对象数组（5个），每个对象都包含：age sex score，要求将对象数组中的内容显示在table中，只显示成绩在80分以上的行

注意事项：

多个结构型指令不能同时用在同一个标签上边，是会报错。

解决方案：借助于ngContainer，不会影响DOM树的结构

5、事件绑定

<any (eventName)="handleEvent()"></any>

<button (click)="handleClick()"></button>

6、双向数据绑定

前提：

必须声明当前的模块 依赖于 表单模块

①找到app.module.ts

②import {FormsModule} from '@angular/forms'

③@NgModule({

imports:[FormsModule]

})

<input [(ngModel)]="myName"/>

方向1：将变量中的值 绑定到 视图

方向2：将视图中用户操作的结果 绑定到 数据

扩充：

复习Vue：

v-bind:class="{myRed:true}"

v-bind:style="{color:myColor}"

1、ng动态样式

<any [ngClass]="{myRed:expression}"></any>

<any [ngStyle]="{backgroundColor:myColor}"></any>

2、如何在双向数据绑定的变量发生变化时去执行指定的操作呢？（监听）

<input [(ngModel)]="myName" (ngModelChange)="handleChange($event)"/>

练习：（16:10 - 16:25）

demo09\_lianxi demo09.component.ts demo09.component.html

视图中有一个按钮 有一个p

功能：点击按钮时，启动定时器，

修改p的透明度（0~1 0~1）

二、自定义指令类

复习Vue：

<any v-my-dir="123"></any>

new Vue({

directives:{

my-dir:{

bind:function(el,bindings){

//bindings.value

},

update:function(){},

unbind:function(){}

}

}

})

Vue.directive('my-dir',{

bind:function(){},

update:function(){},

unbind:function(){}

})

ng:

创建自定义指令类：

import {Directive,ElementRef} from '@angular/core'

@Directive({

selector:'[test]'

})

export class TestDirective{

constructor(private myER:ElementRef){

//构造函数

//得到调用指令的标签

this.myER.nativeElement

}

}

使用：

<any test></any>

自定义指令参数的传递和接收???

Angular的八大组成部分：

①Module

是Angular开发的基本单位，程序在启动时 首先先去启动根模块(AppModule),模块又会指定启动哪个组件(AppComponent)

模块可以包含组件、指令、服务。。。

②Component

组件是一个可被反复使用的，带有特定功能的视图

（本质是一个带有模板内容的指令类）

帮助我们封装可重用的视图

③Directive

指令是一个类(test.directive.ts),得到当前调用指令的元素，根据业务需求，在类中去完成功能，比如修改元素的内容，元素的透明度...

结构型 \*ngFor \*ngIf \*ngSwitchCase

属性型 [ngClass] ngStyle () [(ngModel)]

自定义

④metaData

元数据：元数据告诉 Angular 如何处理一个类。

要配合这装饰器(decorator)来使用

@NgModule({

imports:[BrowserModule,FormsModule,RouterModule],

declarations:[Demo01Component, AppComponent,HelloComponent],

bootStrap:[AppComponent]

})

⑤模板 template

我们通过组件的自带的模板来定义组件视图。模板以 HTML 形式存在，告诉 Angular 如何渲染组件。

有了自定义的标签

扩展了html（循环、选择、绑定。。。）

⑥数据绑定

插值表达式 {{}}

事件绑定 ()

属性绑定 []

双向数据绑定 [(ngModel)]

总结：

1、内置指令的使用

\*ngFor \*ngIf [ngSwitch] \*ngSwitchCase \*ngSwitchDefault

[src]="imgUlr"

(click)="handleClick()"

[(ngModel)]="userName" --> 指定当前模块依赖于表单模块

(ngModelChange)="handleChange($event)"

[ngClass]="{className:expression}"

[ngStyle]="{opacity:myColor}"

2、自定义指令

ElementRef

自定义指令在创建完之后，要到模块中声明！

每日一练：

demo11\_lianxi

demo11.component.ts

功能1：

一个复选框一个按钮，

复选框选中的结果要影响按钮是否禁用

功能2：

有一个输入框，一个无序列表

在无序列表中 显示用户输入过的所有的数据

### Day03通信,管道,自定义指令传参

组件之间的通信

管道

页面之间参数的传递

与服务器端的通信

一、组件间的通信

VueJS组件间通信：

①props down

<son myName="zhangsan"></son>

Vue.component('son',{

props:['myName'],

template:'<h1>{{myName}}</h1>'

})

②events up （子组件 与 父组件进行通信）

绑定事件处理函数（接收方）

触发事件（发送方）

步骤1：在父组件中定义一个带有参数的方法

rcvMsg:function(msg){}

步骤2：在调用子组件时 绑定

<son @customEvent="rcvMsg">

步骤3：触发事件

this.$emit(customEvent,'123');

③ref

<son ref="mySon"></son>

this.refs.mySon

④$parent

⑤bus

Angular组件间通信：

1、props down

①在调用子组件时 通过属性传值

<son myAge="20" mySex="boy"></son>

②在子组件，指定可以有一个输入属性为myAge

import {Input} from '@angular/core'

export class SonComponent{

@Input() myAge=“”；

//支持给属性起一个别的名字

//举例：mySex属性对应的值可以保存在userSex

@Input('mySex') userSex="";

}

③在子组件中 调用属性出来的值

this.myAge

this.userSex

2、events up

绑定

(customEvent)=""

触发

Output EventEmitter

步骤1：在父组件中定义一个带有参数的方法

rcvMsg(msg){

//....

}

步骤2：将方法绑定给子组件的自定义事件

<son (myEvent)="rcvMsg($event)">

步骤3：在子组件中触发指定的事件，并且捎带参数

import {Output,EventEmitter} from '@angular/core'

@Output() myEvent = new EventEmitter();

this.myEvent.emit('123');

练习时间：

demo13\_lianxi

main.component.ts 选择器 test-main

header.component.ts 选择器 test-header

功能：

在main组件调用header组件时，传字符串 '这是主页'给header，header解析并显示在h1

header组件在初始化3s之后，将isUserLogin的值 传递给main,main组件解析传来的值，来决定是否要显示列表

3、本地变量

本地变量 可以帮助在父组件的模板中，得到子组件的实例，读取子组件中定义的数据和方法， 去调用

步骤1：在父组件中调用子组件时，指定本地变量

<son #sonName></son>

步骤2：在父组件的模板中 通过sonName这个对象去调用属性或者方法

<son #sonName></son>

<h1>{{sonName.age}}</h1>

注意事项：不直接允许在js中，得到本地变量所指定的子组件实例

4、ViewChild得到本地变量所指定的子组件实例

步骤1：在调用子组件时 指定本地变量

<son #mySon></son>

步骤2：父组件引入 ViewChild

import {ViewChild} from '@angular/core'

步骤3：在父组件的类中 通过装饰器来定义需要接受的子组件实例的变量

@ViewChild('mySon') mySonInstance:any

this.mySonInstance.age/score/result()

练习：

demo15\_lianxi

demo15-parent.component.ts demo15-parent

demo15-son.component.ts demo15-son

在子组件中 定义两个方法，实现的功能分别是：

在控制台输出‘开始’，‘结束’

在父组件中 ，放上两个按钮，希望能够通过本地变量的方式，在点击按钮时，分别调用子组件中的方法

二、管道 pipe (过滤器)

Vue 2.0之后 没有内置的过滤器，只能自定义过滤器

过滤器的本质 是一个带有参数的，有返回值的方法

Vue.component('my-component',{

filters:{

myCurrency:function(value,arg1,arg2){

//value是通过管道所传递的数据

//arg1/arg2 调用过滤器本身时所传递的参数

}

}

})

<any>{{expression | myCurrency(1,2)}}</any>

Angular是支持内置的管道，也支持管道的自定义

基本用法：支持传参、支持多重过滤

<any>{{expression | myPipe:arg1:arg2 | myPipe2}}</any>

1、内置的管道

uppercase/lowercase

date

number

<any>{{123.45 | number:'4.1-2'}</any>

[整数位保留最小位数].[小数部分保留最小位数]-[小数部分最多保留位数]

小数点前边显示4位

小数点后边最少1位，最多2位

currency

<p>{{price | currency:'USD':true:'4.1'}}</p>

目标货币格式 是否显示货币符号 小数点的格式化信息

percent

<any>{{expression | percent }}</any>

<any>{{expression | percent:'4.2' }}</any>

slice

<any>{{expression | slice:start:end }}</any>

2、自定义管道

管道类

步骤1：创建文件myCurrency.pipe.ts

指定管道名称name、管道的类名、在方法中如何处理数据并返回处理后的结果

步骤2：

到模块中 声明

import {MyCurrencyPipe} from '\*\*\*'

@NgModule({

declarations:[

MyCurrencyPipe

]

})

步骤3：

<any>{{expression | myCurrency }}</any>

练习：自定义管道

创建一个demo17\_lianxi文件夹，

demo17.component.ts

sex-change.pipe.ts

要求通过angular创建一个管道(sexChange)，用来实现性别的转化

要求：管道支持传递参数，决定显示中文/英文

1 男 boy

0 女 girl

{{ 1 | sexChange(true)}} -->男

{{ 1 | sexChange(false)}} -->boy

三、自定义指令传参

①创建指令类 hello.directive.ts

selector:'[hello]'

②声明

app.module.ts去引入指令类，并进行声明

③调用

<any hello="myValue"></any>

④接收调用指令时 传来的参数

在指令类中，引入Input (输入属性)

@Input() hello：any

//解析调用指令时传来的参数

this.hello

每日一练：

demo19\_lianxi

demo19-parent.component.ts

demo19-son.component.ts

change-bg.directive.ts 将调用该指令的元素的背景色设置为一个随机的颜色

父组件：

定义一个数据，count=0;

在调用子组件时，将count传递给子组件

在组件初始化3s之后，将count赋值为5。

子组件：

子组件解析到父组件传递来的数据，显示一个包含指定数量列表项的列表，每一个列表项的背景色都是随机的（调用指令）

### Day04服务,请求,路由

ng组件的生命周期

服务

网络请求

路由模块

一、组件的生命周期

Vue中组件的生命周期：

create/mount/update/destroy

beforeCreate created

beforeMount mounted

... ..

Angular中组件的生命周期

1、ngOnInit

初始化

2、ngOnChanges 当有输入属性的时候，会在组件创建中ngOnInit之前调用；当输入属性发生变化时，也会调用

ngDoCheck 当输入属性对应的值是个集合，集合中的数据发生变化时 会调用的方法

3、ngOnDestroy

组件销毁

4、视图相关的处理函数

ngAfterViewInit

ngAfterViewChecked

ngAfterContentInit

ngAfterContentChecked

二、服务

服务:封装可重用的数据和方法

常用的服务：

网络请求服务

日志类服务

。。

1、创建

import {Injectable} from '@angular/core'

@Injectable()

export class TestService {

//数据

//方法

}

2、使用

①到模块或者组件指定providers

//该模块所有的组件都无需再指定提供商

import {TestService} from '\*\*\*'

@NgModule({

providers:[TestService]

})

//该组件以及子组件无需再指定提供商

import {TestService} from '\*\*\*'

@Component({

providers:[TestService]

})

②在组件中调用

import {TestService} from '\*\*'

constructor(private myTestService:TestService){

}

this.myTestService.属性/方法

练习：

创建一个心跳服务，封装两个方法，分别start和stop，分别输出开始心跳（1s，输出一次）、结束心跳

创建一个文件夹demo22\_lianxi

demo22.component.ts （两个按钮，在分别点击时，调用服务中的start\stop方法）

heart-beat.service.ts

3、封装网络请求服务

3.1 RXJS

Rx (Reactive Extensions） 响应式编程，是异步通信的新的方式

异步请求的处理：

①回调函数

<button onClick="handleClick()">

②promise

then

③ajax

new open send

④RXJS

Observable

subscribe

3.2 具体实现方式

① 指定当前模块依赖于网络请求模块

import {HttpModule} from '@angular/http'

@NgModule({

imports:[HttpModule]

})

②会将网络请求封装成一个服务

import {Injectable} from '@angular/core'

import {Http,Response } from '@angular/http'

import { Observable } from 'rxjs/Observable';

import 'rxjs/add/operator/map';

import 'rxjs/add/operator/catch';

export class MyHttpService{

constructor(private http:Http ){}

sendRequest(url:string){

this.http.get(url)

.map((response: Response) => response.json());

}

}

③ 将网络请求服务在模块中指定一个提供商

import {MyHttpService} from '\*\*\*'

@NgModule({

providers:[MyHttpService]

})

④调用服务

import {MyHttpService} from '\*\*'

constructor(private myService:MyHttpService){

}

this.myService

.sendRequest('http://8888')

.subscribe((result)=>{

console.log(this.result);

})

3.3 允许通信数据中携带和session相关的凭证信息

①客户端

在服务中的get

this.http.get(url,{withCredentials:true})

②服务器端

header('Access-Control-Allow-Origin:http://localhost:3000')

header('Access-Controll-Allow-Credentials:true')

练习：

创建文件夹demo24\_lianxi，创建

demo24.component.ts

store-http.service.ts

要求：

在服务中，封装一个方法,sendHttp,在方法中允许携带参数（发起请求的url）

session\_data.php

在组件创建完毕之后，向服务器端发起请求，验证用户是否登录，如果已经登录，提示'欢迎dingding';如果没有登录，提示“请去登录”

三、路由模块

VueJS路由模块：

建立url和组件映射关系

①基本步骤

vue-router.js

<router-view></router-view>

创建需要用到的各个组件

new Vue({

router:new VueRouter({

routes:[

{path:'/',component:LoginComponent},

{path:'/login',component:LoginComponent},

{path:'\*',component:NotFoundComponent}

]

})

})

②跳转的方式

<router-link to="/login"></router-link>

this.$router.push('路由地址')

③传参的基本步骤

明确发送、接收

配置接收方的路由地址 /login/:id

接收参数 this.$route.params.id

发送

<router-link :to="'/login/'+myId">

</router-link>

this.$router.push('/login/20')

④路由嵌套

准备嵌套其它组件 指定个容器 <route-view></router-view>

其它要嵌套进去的组件配置路由词典

{

path:'/mail',

component:LoginComponent,

children:[

{path:'/inbox',com\*\*\*\*\*}

]

}

Angular路由模块：

1、使用路由模块创建SPA的基本步骤

①指定容器

<router-outlet></router-outlet>

②创建要用到的组件 并声明

③创建app.router.ts

路由词典的配置

④在app.module.ts

import {AppRoutingModule} from './app.router'

@NgModule({

imports:[AppRoutingModule]

})