SERVER DAY01

学习目标：高级前端、大前端、前端全栈工程师

1.课程体系

阶段1：项目流程、数据库、服务器动态语言

阶段2：HTML基础、JS基础、AJAX、CSS

阶段3：JS高级、BOM&DOM、jQuery

阶段4：H5高级、Bootstrap、Node.js

阶段5：Vue.js、Angular、React

阶段6：Ionic、PhoneGap、微信

2.学前准备

(1)全部技术内容从零开始、专业没要求

(2)英语单词从零开始、只要下功夫

(3)对时间有要求！早上-白天-晚上-夜间

(4)对键速有要求！—— 350次/分钟 TopSpeeder

3.Windows常用快捷键

Win+e：打开我的电脑

Win+d：显示/隐藏桌面

Win+r：打开“运行”窗口 calc/mspaint/cmd

Alt+Tab：在多个窗口间切换(从左往右)

Alt+Shift+Tab：在多个窗口间切换(从右往左)

Alt+F4：关闭当前窗口

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

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

4.软件工程

Debug： Ada - Bug - Debug（调试程序）

IBM - DOS - Bill Gates - Microsoft - Windows

软件过程/生命周期——重点内容：

**(1)软件定义期**

1)可行性研究阶段 —— 《可行性研究报告》

技术、人员、资金、时间、设备、政策法规、风俗、回报率

2)需求分析阶段 —— 《软件需求说明书》

功能性需求点、非功能需求点

**(2)软件开发期**

3)概要设计阶段 —— 《概要设计说明书》—— 架构师

技术(软件硬件网络)选型、功能模块划分、数据结构设计

4)详细设计阶段 —— 《详细设计说明》——设计师

模块实现细节、对象、属性、方法、算法

5)编码实现阶段 —— 《项目开发进度报告》

UI/UE工程师、前端工程师、后台工程师

6)项目测试阶段 —— 《项目测试报告》

**(3)软件维护期**

7)项目部署阶段 —— 运维工程师&开发人员

开发模式(Develop Mode) => 生产模式(Production Mode)

8)项目维护阶段 —— 运维工程师&开发人员

5.“学子商城”功能性需求

www.codeboy.com

www.codeboy.com/admin/login.html

**前台子系统：**

产品模块：首页、搜索、列表、详情

用户模块：注册、登录、收藏夹、用户中心

购物车模块：添加购物车、修改购物车、下单

**后台子系统：**

产品模块：添加、修改、删除、列表

用户模块：列表、删除

订单模块：列表、修改

统计模块：各种统计信息

**App子系统：**

产品模块：首页、搜索、列表、详情

用户模块：注册、登录、收藏夹、用户中心

购物车模块：添加购物车、修改购物车、下单

6.搭建一个自己的网站服务器，让同桌来访问 —— 试一试

**(1)在自己的电脑上安装一款WEB服务器**

下载地址：xampp.org 默认安装路径： c:/xampp

**(2)启动自己的WEB服务器**

点击Apache后的Start按钮，看到80,443端口打开

**(3)编写网页保存在自己的WEB服务器中**

保存在c:/xampp/htdocs目录下——默认只有该目录下的网页才能被其他人访问到

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

**(4)让同桌用他的浏览器访问自己的WEB服务器中的网页**

127.0.0.1 访问自己的WEB服务器

172.163.X.X 访问指定的那台WEB服务器

|  |
| --- |
| 电脑小知识：网络上如何区分不同的计算机？  方式1：通过不同的域名，如jd.com  方式2：通过不同的IP地址，每台计算机都有 |
| 如何查看自己计算机的IP地址： Win+R => cmd => ipconfig |
| 172.163.100.171 这是教师机在当前网络中的地址  127.0.0.1 这个IP地址永远指向本机，即使没有联网 |

课后任务：

(1)在自己的电脑上安装XAMPP软件，搭建WEB服务器，自己访问自己的WEB服务器。

SERVER DAY02

复习：

软件过程：

软件定义期：可行性研究、需求分析

软件开发期：概要设计、详细设计、编码实现、测试

软件维护期：项目部署、项目维护

初试WEB服务器：

安装XAMPP，并启动，编写并包存在c:/xampp/htdocs

192.168.X.X 路由器分配给当前计算机的IP地址

127.0.0.1 指代当前计算机

今日目标：

(1)常用的MySQL命令 —— 难点(单词稍多),重点！

1.项目过程：概要设计阶段 —— 架构师

任务：技术选型(网络/语言/框架)、项目结构(子系统/模块)、数据结构(数据特点/内容)

项目中存储数据的方式：

(1)服务器内存：存取速度快；非永久存储、容量有限

(2)外存/文件系统：永久存储、格式随意(txt/xls/...)、容量大；存取速度慢

(3)云存储：托管给云服务提供商

(4)数据库服务器：可以永久存储/也可以非永久存储、存储速度快(存储优化)、容量大

|  |
| --- |
| KB/MB/GB/TB/PB/EB/ZB/YB/BB..... |

2.扩展知识：什么是服务器(Server)

硬件服务器：一台能给其他计算机提供特定数据的计算机

软件服务器：运行起来监听客户端请求，并提供响应内容的程序

个人操作系统：

Windows / Mac

企业应用操作系统：

Linux(RedHat/CentOS/Ubuntu)

Unix(Solaris/AIX/FreeBSD....)

|  |
| --- |
| data：数据 date：日期 |
| quit：退出 quite：很，非常 |
| exit：出口/退出 exist：存在 |

3.数据库服务器

Database：用于存储项目中的批量业务数据，如银行中储户信息、论坛中帖子信息、商城中的商品/用户/订单...实现高效的数据访问。

历史上出现过的数据库的种类：

(1)网状数据库

(2)层次型/树型数据库

(3)关系型数据库(1980s~至今的主流)

SQLite、MySQL、PostgreSQL、SQLServer、Oracle、DB2

(4)非关系型数据库(NoSQL，最新的)

4.最常用的关系型数据库——MySQL介绍

Oracle版本： www.mysql.com

Martin版本： www.mariadb.org

|  |
| --- |
| 开源最流行的工具栈(LAMP)：  Linux + Apache + MySQL + PHP |

安装MySQL：可以从上述两个官网下载并安装，或者直接下载安装一个服务器套装XAMPP = Apache + MySQL + PHP

(xampp.org)

MySQL系统部署结构：

(1)服务器端：负责存储和维护数据(人无法直接读取)——工行总行

(2)客户端：负责向服务器发起增删改查命令——工行遍布全国的ATM机

**如何使用MySQL客户端程序连接MySQL服务器：**

mysql.exe -h127.0.0.1 -uroot -p 不加分号！

可简写为

mysql -uroot 不加分号！

**如何退出与MySQL服务器的连接：**

quit

午间练习：故意将上述两个命令写错，看看错误提示是什么（自己翻译出英文错误消息）？

|  |
| --- |
| MySQL服务器中数据的逻辑结构：  **Server => Database => Table => Row => Column** |

5.MySQL服务器常用命令

SQL：Structured Query Language，结构化语言，可用于操作关系型数据库服务器中数据的语言——可用于增删改查数据。1980年由IBM提出，由Oracle产品化，后来提交给ISO成为行业标准语言，当前所有的关系型数据库产品都支持该语言。

注意：(1)SQL命令必须都以英文分号结尾！(2)SQL命令不区分大小写！(3)SQL语句中可以使用单行注释#...或者多行注释/\*...\*/

常用的SQL语句：

(1)show databases ; 显示当前已有的所有数据库

(2)quit ; 退出连接

(3)use 库名 ; 进入指定的数据库

(4)show tables ; 显示当前库中所有的表名

(5)desc 表名 ; 描述一下指定表的结构(有哪些列)

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

(6)drop database if exists 库名; 丢弃指定数据库(如果它存在的话)

(7)create database 库名 ; 创建一个新的数据库

(8)create table 表名(列, 列) ; 创建一个新的表

(9)insert into 表名 values(值, 值); 向表中插入一行数据

(10)

(11)

(12)select \* from 表名; 查询指定表中所有的数据

练习：连接到MySQL服务器，查看当前有哪些数据库，退出连接

练习：查询MySQL服务器默认自带的五个库中各有多少个表

练习：查询出MySQL服务器默认自带的phpmyadmin库中下面三个表中各有哪些列：

pma\_\_favorite、pma\_\_history、pma\_\_relation

练习：创建一个新的脚本文件4.sql，丢弃数据库dangdang，创建数据库dangdang，执行该文件。

|  |
| --- |
| 小知识：提交SQL语句给服务器执行的两种方式：  **(1)交互式执行：**输入一行命令，回车提交执行一行....适合于临时性查看数据  mysql -uroot  **(2)脚本式执行：**把要执行的多条命令编写在一个文本文件中，一次性的全部提交给服务器执行，需要的话以后可以反复执行；适合于批量的增删改数据  mysql -uroot < c:/xxx/2.sql |

课后练习：创建“学子商城”必需的笔记本表



创建脚本文件xuezi.sql，编写SQL语句：

(1)删除数据库xuezi，如果存在的话

(2)创建数据库xuezi

(3)进入数据库xuezi

(4)创建笔记本信息表laptop，包含如下几个列：

lid - 笔记本编号，如'1002'

pic - 笔记本的图片路径，如'img/2.jpg'

title - 笔记本标题，如'戴尔燃7000'

price - 笔记本价格，如'3500'

isOnsale - 是否特价，如'是'或者'否

(5)插入4行笔记本数据

(6)在交互模式下查询出所有的笔记本数据

SERVER DAY03

复习：

MySQL部署结构：

(1)服务器端：负责存储和维护数据(人无法直接看)

(2)客户端：用于向服务器发起增删改查命令

使用MySQL客户端连接服务器的两种方式：

(1)交互模式： ——查

mysql.exe -h127.0.0.1 -uroot -p

mysql -uroot

(2)脚本模式：——增删改

mysql -uroot < d:/xx/yy.sql

Server => Database => Table => Row => Column

常用的SQL命令：系统关键字都大写，非关键字都小写

SHOW DATABASES;

USE dangdang;

SHOW TABLES;

DESC book;

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

DROP DATABASE IF EXISTS dangdang;

CREATE DATABASE dangdang;

USE dangdang;

CREATE TABLE book(

name VARCHAR(128),

price INT

);

INSERT INTO book VALUES('Sanguo', '15');

INSERT INTO book VALUES('XiYou', '25');

SELECT \* FROM book;

练习：创建学子商城必需的笔记本信息表

今日目标：

(1)补充SQL语句：删除和修改语句

(2)列类型

(3)列约束

1. 补充SQL语句：

(10)delete from 表名 ; 删除指定表所有的数据

delete from 表名 where 条件; 删除满足指定条件的记录

(11)update 表名 set 列=值, 列=值 ; 修改指定表所有数据行

update 表名 set 列=值, 列=值 where 条件 ; 修改满足指定条件的记录

练习：创建一个2\_tedu.sql文件，编写如下SQL语句

丢弃然后重建数据库tedu，进入该库。

创建保存部门信息的表**dept**: did, dname, empCount

插入3行部门数据： 10 Develop 3

20 Market 1

40 Test 2

创建保存员工信息的包**emp**: 有如下的列：

#eid, ename, sex, salary, birthday, deptId

插入6行员工数据，分属于上述三个部门，例如

#5513 TOM M 8000 1990-1-1 20

删除最后一个员工 —— 需要执行几条语句？

将一个员工由10号部门转到20号部门 —— 需要执行几条语句？

#午间练习：删除上述代码，根据注释再现代码

|  |
| --- |
| 面试题：WEB项目中哪里可能导致乱码问题？  数据库乱码、后台语言乱码、网络传输、浏览器解析 |

2.数据库中的乱码问题

计算机只能处理数字，如何保存字符？——把字符转换为数字

A - 65 B - 66 a - 97 b - 98

AABBAaBb 可以编码(encode)为 6565666665976698

6565666665976698 可以解码(decode)为AABBAaBb

**ASCII字符集/编码方案**：包含128个字符，对英文常用字符/符号进行了编码。

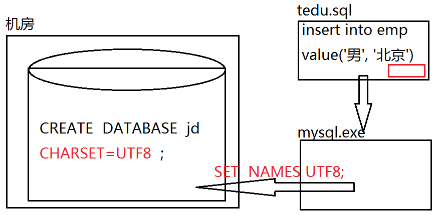
**GB2312/GBK字符集**：包含21003个字符，对英文/常用简体汉字都进行了编码，兼容ASCII编码。

**BIG5字符集**：包含1万3000多个字符，对英文/常用繁体汉字都进行了编码，兼容ASCII编码。

**LATIN-1字符集**：包含256个字符，对英文/西欧常用符号进行了编码，兼容ASCII编码。

**Unicode字符集**：包含6万多个字符，对世界上主流语言的常用符号都进行了编码，兼容ASCII编码。 编码方案又具体分为UTF-8、UTF-16、UTF-32等等多套方案。

数据库中乱码问题产生的根源：存储数据时(编码)使用某个字符集，显示时(解码)使用了另一套！



解决方法：——三处一致

(1)x.sql脚本文件另存为UTF-8

(2)提交SQL语句时设置语句所用的字符集为UTF8

(3)创建数据库时指定默认字符存储方案为UTF8

3.MySQL中的列类型

CREATE TABLE 表名 ( 列名 列类型 );

MySQL中常用的列类型 —— 在手册第11章——MySQL提供的列类型比其它所有数据库都要多：

**(1)数值类型 —— 数值数据可用''括起来/也可以不括**

TINYINT:微整数，占1字节，-128~127，如学生.年龄

SMALLINT:小整数，占2字节，-32768~32767，如部门.员工数量

INT:整数，占4字节，-2147483648~2147483647，如帖子.回复数量

BIGINT:大整数，占8字节，19位数字，如当前距离计算机元年多少毫秒

FLOAT(M,D):单精度浮点小数，占4字节，3.4E38

DOUBLE(M,D):双精度浮点小数，占8字节，1.79E308

DECIMAL(M,D):定点小数，存储时小数点不动，不会产生四舍五入，不会产生精度丢失，M表示总的有效位数，D表小数点后允许的位数，如笔记本.价格 DECIMAL(7, 2)；高考成绩 DECIAL(4, 1)

BOOL: 布尔类型，只有两个值 TRUE/1 和 FALSE/0，如商品.是否特价、员工.是否在岗、用户.是否在线、新闻.是否置顶显示

**(2)日期时间类型——数据必须用''括起来**

DATE:日期，如员工.生日

CREATE TABLE emp(birthday DATE)

INSERT INTO emp VALUES( '1990-1-1' )

TIME:时间，如学生.上课时间

INSERT INTO stu VALUES( '8:30:00' )

DATETIME:日期时间，如帖子.发表时间

CREATE TABLE article(pubTime DATETIME)

INSERT INTO article VALUES( '2017-10-5 22:18:35' )

**(3)字符串类型——数据必须用''括起来**

CHAR(M):定长字符串，长度不够\0来凑，操作效率远高与VARCHAR！如员工.身份证号，M不能超过255

VARCHAR(M):变长字符串，结尾最多只需要一个\0，可能更省空间！如员工.个人简历、帖子.内容，M不能超过65535

TEXT(M):大型变长字符串，M不能超过2G

|  |
| --- |
| 小知识：浮点小数  1234.5678 等于：  123.45678x10^1  12.345678x10^2  **1.2345678x10^3 科学计数法 1.2345678E3**  0.12345678x10^4  12345.678x10^-1  123456.78x10^-2  ...... |

|  |  |  |
| --- | --- | --- |
|  | CHAR(4) | VARCHAR(4) |
| 'a' | a\0\0\0 | a\0 |
| 'ab' | ab\0\0 | ab\0 |
| 'abc' | abc\0 | abc\0 |
| 'abcd' | abcd | abcd |
| 'abcde' | abcd | abcd |
| '一' | 一\0\0\0 | 一\0 |
| '一二' | 一二\0\0 | 一二\0 |
| '一二三' | 一二三\0 | 一二三\0 |
| '一二三四' | 一二三四 | 一二三四 |
| '一二三四五' | 一二三四 | 一二三四 |

4.MySQL中的列约束

Constraint：约束，指插入数据时需要对数据进行的检验，如：商城中用户名唯一、密码不能为空、学生年龄必须在18~60之间....只有符合规定的数据才允许插入，否则插入失败。

CREATE TABLE 表名 ( 列名 列类型 列约束 );

SQL标准共提供了六种列约束：

**(1)唯一约束：unique**

声明为唯一约束的列上不能出现重复值。

**(2)非空约束：not null**

声明了非空约束的列上不能出现null值

**(3)主键约束：primary key**

声明为主键的列上不能出现重复值，也不能出现null值，且数据库会根据主键列上值对整个表中的数据由小到大排序！

注意：一个表中最多只能有一个列声明为PRIMARY KEY！其它唯一且非空的列只能声明为 UNIQUE NOT NULL；

|  |
| --- |
| 小知识：计算机中NULL的含义  Null：空、空白、空缺，指应该有此数据，但暂时不知道确切的值，就可以先声明为null。员工.年终奖、新员工.部门编号、新部门.尚未确定的部门经理 |

课后练习：**创建“学子商城”必需的数据库结构，注意列类型和列约束**

编写SQL脚本文件，重新创建数据库xz，进入该数据库；

创建笔记本型号表：

xz\_laptop\_family(

fid-型号编号,

fname-型号名称,

laptopCount-属于该型号的笔记本数量

)

插入三种笔记本型号，如“联想E470”、“小米Air”、“MacBook”，每款型号对应的笔记本数量分别有3/2/2。

创建笔记本信息表：

xz\_laptop(

lid-笔记本编号

pic-图片

title-标题

price-价格

type-所属类别，如“轻薄本”/“游戏本”/“普通本”

marketDate-上市日期

isOnIndexTop-是否首页置顶显示

familyId-所属型号的编号

)

插入7条笔记本记录，分属于上述三个笔记本型号。

删除编号为5的笔记本，对应的型号下笔记本数量应该-1；

修改编号为2的笔记本所属型号为另一种型号，原型号下的笔记本数量-1，新型号下的笔记本数量+1.

SERVER DAY04

复习：

列类型：

数值类型： 20 '20'

tinyint / smallint / int / bigint

float / double / decimal(m,d)

bool (TRUE-1/FALSE-0)

日期时间类型： '1990-1-1'

date / time / datetime

字符串类型： 'TOM'

char(m) / varchar(m) / text(m)

列约束： CREATE TABLE stu( age TINYINT 约束)

(1)唯一约束： unique

(2)非空约束： not null

(3)主键约束： primary key = 唯一+非空+排序

练习： 注意使用合理的列类型和列约束

编写1\_xz.sql，丢弃并重建数据库xz，进入该库。

创建笔记本型号表laptop\_family：

fid, fname, laptopCount

10 联想E480 3

20 Mac Book 2

30 戴尔燃700 2

创建笔记本信息表，并插入7行数据laptop：

lid(编号), pic(图片路径), title(标题), price(价格), spec(规格), marketDate(上市日期), isRecommended(是否为首页推荐), familyId(所属型号的编号)

今日目标：

(1)补充：列约束

(2)SQL中的查询

1.补充：列约束

CREATE TABLE stu( age TINYINT 约束)

(1)唯一约束： unique

(2)非空约束： not null

**(3)主键约束：** primary key = 唯一+非空+排序

(4)默认值约束： default

可以为某个列声明一个默认值，若插入数据时未指定具体值，就可以使用默认值。

CREATE TABLE user(

avatar VARCHAR(128) DEFAULT 'img/logo.gif'

);

INSERT INTO user VALUES( 'img/lwh.jpg' ); #不使用默认值

INSERT INTO user VALUES( DEFAULT ); #使用默认值

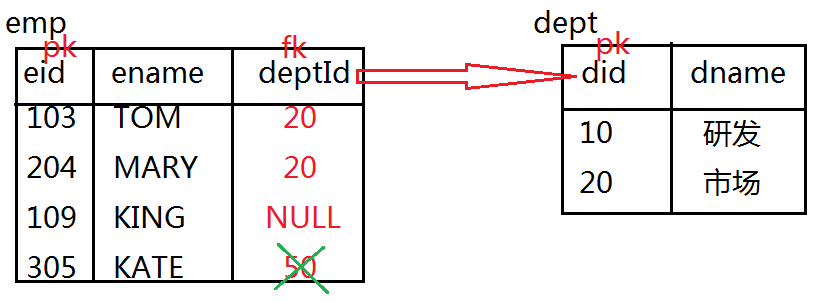
(5)检查约束：check

检查约束可以对新插入的数据值的范围进行检查，满足条件的允许插入，否则插入失败！ —— MySQL不支持，因为检查约束会降低所有插入和更新操作的执行效率。

CREATE TABLE stu ( age INT CHECK( age>=6 AND age<=15) )

(6)外键约束：foreign key

声明了外键约束的列，可以插入重复值或NULL值，但所有插入的值必须在另外一个表的主键列上出现过。因为检查约束会降低所有插入和更新操作的执行效率，慎重使用。



CREATE TABLE dept( did INT PRIMARY KEY, dname VARCHAR(32));

CREATE TABLE emp(

eid INT PRIMARY KEY,

ename VARCHAR(32),

deptId INT ,

**FOREIGN KEY(deptId) REFERENCES dept(did)**

);

2.项目中如何保存日期和时间

方式1：使用VARCHAR，如'1990-1-12'，不便于对齐，不便于比较大小

方式2：使用DATE，如'1990-1-1'，

好处：会自动补齐，便于比较大小

不足：不便于实现国际化应用，不同系统日期时间表示范围不同

方式3：使用BIGINT，如1520478184225，表示指定的时间距离计算机元年多少毫秒

|  |
| --- |
| i18n： internationalization，国际化，一个实现国际化的项目可以做到“对中国人现实汉语及对应的习惯，对美国人显示英语及对应的习惯”  中国人： 1997-7-25  美国人： 7-25-1997， Jul 25, 1997  欧洲人： 25/7/1997 |
| 计算机元年：1970-1-1 0:0:0 GMT |
| 计算机中如何存储系统时间？ —— 转换为一个大整数，表示指定的时间距离计算机元年经过了多少毫秒  0 1970-1-1 0:0:0  1000 1970-1-1 0:0:1  1000\*60 1970-1-1 0:1:0  1000\*3600 1970-1-1 1:0:0  1000\*3600\*24 1970-1-2 0:0:0  1000\*3600\*24\*365 1971-1-1 0:0:0  1000\*3600\*24\*365\*48 大约就是今天的时间 |

3.扩展小知识：MySQL中如何实现自增列

MySQL专门提供了一个 AUTO\_INCREMENT 关键字，只能用于INT 型的 PRIMARY KEY 列上，用于实现主键上整数的自动增长。

自增列上允许手工赋值，后续的插入只要插入NULL即可，就会在当前最大值基础上+1.

4.MySQL简单查询 —— 查询特定的列

示例：查询所有员工的姓名及其工资

SELECT ename, salary FROM emp ;

练习：查询出所有部门的名称及所在地

SELECT dname, location FROM dept;

练习：查询所有员工的姓名、月薪、生日、性别

SELECT ename, salary, birthday, sex

FROM emp; #列出现顺序随意

练习：查询所有员工的姓名、所在部门编号、员工编号、所在部门的编号、员工编号

SELECT ename, deptId, eid, deptId, eid

FROM emp ; #某列可以反复出现

练习：查询出员工的编号、姓名、性别、生日、工资、部门编号

SELECT eid, ename, sex, birthday, salary, deptId

FROM emp;

SELECT \* FROM emp ; #\*指代所有的列名

5.MySQL简单查询 —— 给列取别名

示例：查询所有员工的姓名和工资，列名用中文呈现

SELECT ename AS 姓名, salary 月薪

FROM emp; #AS用于给列取别名，可以省略

练习：查询所有员工的编号、姓名、所在部门编号，列名都用中文显示

SELECT eid AS 工号, ename AS 姓名, deptId 部门编号

FROM emp;

练习：查询所有员工的姓名(用n呈现)、生日(用b呈现)、工资(用s呈现)

SELECT ename AS n, birthday AS b, salary AS s

FROM emp;

6.MySQL简单查询 —— 只显示不同值

示例：查询出有员工的所有部门的编号

SELECT DISTINCT deptId

FROM emp; #distinct：不同的，相异的

练习：查询出公司有哪些性别的员工

SELECT DISTINCT sex

FROM emp;

7.MySQL简单查询 —— 查询时执行运算

示例：计算2+3的和

SELECT 2+3 ;

SELECT (123+256-378)\*119 / 110 AS 结果;

示例：查询出所有员工的姓名，月薪以及年薪

SELECT ename, salary, salary\*12

FROM emp;

练习：老板想看看如果每个员工每月加薪10%，年底再给一次性的20000年终奖，每人年底总收入是多少

SELECT ename, salary, salary\*(1+0.1)\*12 + 20000

FROM emp;

练习：假设每个员工每月需要缴纳社保比例占月薪的30%，但年终的20000年终奖无需缴税，计算每个员工每个月的平均到手收入

SELECT ename, (salary\*(1-0.3)\*12 + 20000) / 12

FROM emp;

8.MySQL简单查询 —— 单条件查询

MySQL提供比较运算符： = != > >= < <=

示例：查询出10号部门的所有信息

SELECT \* FROM dept

WHERE did=10; #where：满足...条件

示例：查询出编号大于10的部门的所有信息

SELECT \* FROM dept

WHERE did>10;

练习：查询出7788号员工的所有信息

SELECT \* FROM emp

WHERE eid=7788;

练习：查询出20号部门所有的员工信息

SELECT \* FROM emp

WHERE deptId=20;

练习：查询出所有的女员工信息

SELECT \* FROM emp

WHERE sex=0 ;

练习：查询出不在10号部门的员工信息

SELECT \* FROM emp

WHERE deptId != 10;

练习：查询出工资达到6000的所有信息

SELECT \* FROM emp

WHERE salary >= 6000;

练习：查询出不在任何部门的员工的信息

SELECT \* FROM emp

#WHERE deptId = NULL ; #NULL不等于任何数据，以及它自身

WHERE deptId IS NULL ; #NULL的判定不能用=或!=

练习：查询出在某个特定部门的员工的信息

SELECT \* FROM emp

WHERE deptId IS NOT NULL ; #此处不能用 != NULL

9.MySQL简单查询 —— 多条件查询

MySQL提供了两个逻辑运算符： AND(并且) OR(或者)

示例：查询出10号部门的女员工所有信息

SELECT \* FROM emp

WHERE deptId=10 AND sex=0 ;

练习：查询出工资在6000~8000之间的所有员工信息

SELECT \* FROM emp

#WHERE 6000<salary<8000; #错误写法！

WHERE salary>=6000 AND salary<=8000;

练习：查询出10号和30号部门的所有员工信息

SELECT \* FROM emp

WHERE deptId=10 OR deptId=30;

练习：查询出在1990年出生的所有员工信息(在1-1号和12-31间)

SELECT \* FROM emp

WHERE birthday>='1990-1-1' AND birthday<='1990-12-31';

练习：查询出工资小于5000和大于8000的员工所有信息

SELECT \* FROM emp

WHERE salary<5000 OR salary>8000;

10.MySQL简单查询 —— 模糊条件查询

MySQL中使用 **%** 代表“任意多个任意字符”； 用 **\_** 代表“任意一个字符”—— 这两个符号表示模糊查询必须与 **LIKE** 关键字配合！

示例：查询出姓名中包含字符E的所有员工信息

SELECT \*

FROM emp

#WHERE ename = 'E'; #错误

#WHERE ename = '%E%'; #错误

WHERE ename LIKE '%E%';

练习：查询出姓名中第二个字符是E的员工所有信息

SELECT \*

FROM emp

WHERE ename LIKE '\_E%';

课后练习：

(1)删除上面笔记中所有的SQL语句，根据提示写出所有语句 —— 经典练习！！

(2)根据“学子商城数据库结构设计说明书”编写出该项目所需要的.sql文件——创建出数据库，及所有的表，无需插入数据。

SERVER DAY05

复习：

day01：

软件过程：

软件定义期：可行性研究、需求分析

软件开发期：概要设计、详细设计、编码实现、测试

软件维护期：项目部署、项目维护

day02：

MySQL中常用的SQL语句：

show databases;

use 库名;

show tables;

desc 表名;

show warnings; #显示上一条语句产生的警告

show create table 表名; #显示创建表时所用的语句

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

drop database if exists 库名;

create database 库名 charset=utf8;

use 库名;

create table 表名( 列名 列类型 列约束 );

insert into 表名 values('值');

delete from 表名 where 条件;

update 表名 set 列名='值' where 条件;

select \* from 表名;

day03:

列类型：

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

日期时间类型：date/time/datetime

字符串类型：char/varchar/text

列约束：

唯一/非空/**主键**/默认值/检查/外键

day04：

简单查询：

查询特定的列 select ename,sex from emp

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

显示不同的值 select distinct sex from emp

执行运算 select salary\*12 from emp

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

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

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

复杂查询：

今日目标：

(1)MySQL中的查询

(2)WEB服务器

|  |
| --- |
| Order: 订单、排序 |
| Set：设置、集合 |
| Table：表、桌子 |

1.MySQL中的简单查询 —— 查询结果的排序

示例：查询出所有员工信息，要求按工资由小到大排序

SELECT \* FROM emp

ORDER BY salary ; #默认是由小到大

#ORDER BY salary ASC; #ascendant，升序

示例：查询出所有员工信息，要求按工资由大到小排序

SELECT \* FROM emp

ORDER BY salary DESC ; #descendant，降序

练习：查询出所有员工信息，按照姓名由小到大排序

SELECT \* FROM emp

ORDER BY ename ;

练习：查询出所有员工信息，按照姓名由大到小排序

SELECT \* FROM emp

ORDER BY ename DESC;

练习：查询出所有员工信息，按照年龄由大到小排序

SELECT \* FROM emp

ORDER BY birthday ASC ; #生日由小到大排列

练习：查询10号部门中员工的所有信息，按工资由大到小排列

SELECT \* FROM emp

WHERE deptId=10

ORDER BY salary DESC ;

#注意：where子句和order by子句的顺序问题

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

SELECT \* FROM emp

ORDER BY salary DESC, ename ASC ;

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

SELECT \* FROM emp

ORDER BY sex ASC , birthday DESC ;

2.MySQL中的简单查询 —— 分页查询 —— 重点&面试重点

分页查询：用于分批次的显示大量的数据。语法：

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

LIMIT start, count ;

(1)start：是一个数字，表示从哪一行开始获取数据

(2)count：是一个数字，表示一次最多获取的行数，即页面大小

假设每页显示5行记录，则

第1页： LIMIT 0, 5

第2页： LIMIT 5, 5

第3页： LIMIT 10, 5

第4页： LIMIT 15, 5

第5页： LIMIT 20, 5

....

第n页： LIMIT (n-1)\*5, 5

练习：分页查询出员工信息，页面大小为5，显示第1页

SELECT \* FROM emp LIMIT 0, 5 ;

练习：分页查询出员工信息，页面大小为5，显示第2页

SELECT \* FROM emp LIMIT 5, 5 ;

练习：分页查询出员工信息，页面大小为5，显示第3页

SELECT \* FROM emp LIMIT 10, 5 ;

练习：分页查询出员工信息，页面大小为5，显示第4页

SELECT \* FROM emp LIMIT 15, 5 ;

3.MySQL复杂查询 —— 子查询

示例：查询出“研发部”所有员工的信息

步骤1：查询研发部的部门编号 => 10

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

步骤2：查询部门编号为10的员工信息

SELECT \* FROM emp WHERE deptId = 10;

综合上述两步为一条语句：

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

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

) ;

练习：查询出工资比TOM高的所有员工信息

步骤1：查询TOM的工资 => 6000

SELECT salary FROM emp WHERE ename='TOM';

步骤2：查询工资大于6000的员工信息

SELECT \* FROM emp WHERE salary>6000;

综合上述两步：

SELECT \* FROM emp WHERE salary>(

SELECT salary FROM emp WHERE ename='TOM'

);

练习：查询出比TOM年长的所有员工信息

步骤1：查询TOM的生日 => 1990-1-1

SELECT birthday FROM emp WHERE ename='TOM';

步骤2：查询出生日小于1990-1-1的员工所有信息

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

综合上述两步：

SELECT \* FROM emp WHERE birthday<(

SELECT birthday FROM emp WHERE ename='TOM'

);

练习：查询出与TOM同年出生的所有员工信息

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

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

步骤2：查询出生日中年份等于1990的员工所有信息

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

综合上述两步：

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

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

);

4.MySQL复杂查询 —— 聚合/分组查询

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

MySQL提供了几个聚合函数：

COUNT()：计算若干数据的个数

SUM()：计算若干数据的总和

AVG()：计算若干数据的平均值

MAX()：计算若干数据的最大值

MIN()：计算若干数据的最小值

示例：查询出员工的总数

SELECT COUNT(eid) FROM emp; #14

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

SELECT COUNT(sex, deptId) FROM emp;

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

SELECT COUNT(\*) FROM emp; #推荐写法

示例：查询出所有员工工资的总和

SELECT SUM(salary) FROM emp;

练习：查询出所有员工工资的平均值（用两种方法）

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

SELECT AVG(salary) FROM emp;

练习：查询出所有员工工资的最大值和最小值

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

练习：查询出工资高于所有人平均工资的员工的数量 —— 难度系数：5星

步骤1：查询出所有员工的平均工资 => 6842

SELECT AVG(salary) FROM emp;

步骤2：查询出工资大于6842的员工的数量

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

综合上述两步：

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

SELECT AVG(salary) FROM emp

);

示例：查询出每个部门的编号及该部门员工的数量——先分组再聚合计算

SELECT deptId, COUNT(\*) FROM emp

GROUP BY deptId ; #按照部门编号分组

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

练习：按照性别分组，查询每组中工资的最大值、最小值、平均值

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

FROM emp

GROUP BY sex;

5.MySQL复杂查询 —— 跨表/多表查询 —— 了解

查询结果集中的数据来自于多个不同的表。

示例：查询每个员工的姓名及其所在部门的名称

SELECT ename, dname

FROM emp, dept ; #笛卡尔积！错误！

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

SELECT ename, deptId, did, dname

FROM emp, dept

WHERE deptId = did ; #跨表查询必须有连接条件

注意：上述写法是SQL-92版本中的写法。存在一定的缺陷：

(1)无法显示没有部门的员工 —— KING

(2)无法显示没有员工的部门 —— 测试部

SQL-99标准对此进行了改变，但是SQL变得很复杂，有四种写法：

**(1)内连接查询：inner join ... on ...**

SELECT ename, dname

FROM emp INNER JOIN dept

ON deptId=did ;

#内连接的效果与SQL-92标准完全一样

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

SELECT ename, dname

FROM emp LEFT OUTER JOIN dept

ON deptId=did ; #13+1

#可以显示出“左侧表”中所有的记录！即使右侧表中没有对应的记录

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

SELECT ename, dname

FROM emp RIGHT OUTER JOIN dept

ON deptId=did ; #13+1

#可以显示出“右侧表”中所有的记录！即使左侧表中没有对应的记录

**(4)全外连接查询：full join ... on ...**

SELECT ename, dname

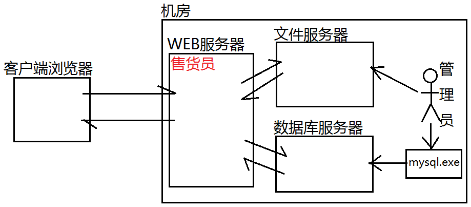
FROM emp FULL JOIN dept

ON deptId=did ; #13+1+1

#可以显示出“右侧表”和“右侧表”中所有的记录！—— MySQL不支持

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

6.WEB服务器



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

**(1)静态WEB服务器**：提供的内容任何时间任何人访问都是一样的

常见的静态WEB内容：HTML/CSS/JS/图片/视频/Flash...

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

常见的动态WEB技术：

JSP = HTML + JAVA

PHP = HTML + php

ASP.NET = HTML + C#

Node.js

Python

.....

7.PHP简介

1994年诞生，Personal Home Page语言，用于记录自己的网页的访问统计。后来开源给社区，逐渐流行；1999年，改名为 PHP is Hypertext Preprocessor。特点：开源、简单、跨平台、占用资源少。黄金组合：

LAMP = Linux + Apache + MySQL + PHP

8.搭建PHP开发和运行环境

(1)服务器端：下载并安装PHP服务器——XAMPP

c:/xampp/apache/bin/httpd.exe 静态WEB服务器

c:/xampp/php/php.exe PHP解释器

(2)服务器端：启动WEB服务器，保证端口被打开(默认是80)

(3)服务器端：编写网页，保存在WEB服务器的特定目录下

c:/xampp/htdocs/

记得删除此目录下的index.php文件！

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

(4)客户端：打开浏览器，访问远程的WEB服务器即可

127.0.0.1:80

课后练习：

(1)删除上面笔记中所有的SQL代码，根据注释重新写出示例代码——强化练习！

(2)创建3.php，在此网页中向客户端输出如下内容：

姓名：丁当

电话：13501234567

邮箱：dingdang@tedu.cn

————————————————————

版权所有® DINGDANG.COM

(3)创建4.php，在此网页中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\*1=4 4\*2=8 4\*3=12 4\*4=16

......

PHP DAY01

复习：

MySQL中的简单查询：

select ename, sex from emp;

select ename AS n, sex 性别 from emp;

select DISTINCT sex from emp;

select ename, salary\*12 from emp;

select salary from emp where deptId=10;

select \* from emp where deptId=10 AND sex=0;

select \* from emp where ename LIKE '%E\_';

select \* from emp order by salary;

select \* from emp limit 0, 10;

MySQL中的复杂查询：

子查询：

select \* from emp where salary>(

select salary from emp where ename='TOM'

);

聚合查询：

select COUNT(\*) from emp where deptId=10;

select SUM(salary) from emp where deptId=10;

select AVG(salary) from emp where deptId=10;

select MAX(salary) from emp where deptId=10;

select MIN(salary) from emp where deptId=10;

select deptId,COUNT(\*) from emp group by deptId;

多表查询：

select ename, dname from emp, dept

where deptId=did; #13 SQL-92

SQL-99标准：

select ename, dname

from emp inner join dept

on deptId=did; #13

select ename, dname

from emp left outer join dept

on deptId=did; #13+1 KING

select ename, dname

from emp right outer join dept

on deptId=did; #13+1 40-测试部

select ename, dname

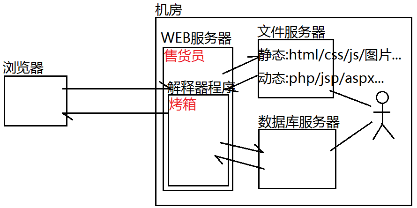
from emp full join dept

on deptId=did; #13+1+1

PHP语言与SQL语言的关系：

SQL语言用于操作数据库服务器

PHP语言运行于WEB服务器



今日目标：

(1)PHP中的数据类型 —— 重要&简单

(2)PHP中的运算符 —— 重要&零乱&复习后很简单

1.面试题：如何自学一门编程语言？

(1)了解背景：历史、特点、应用场合、趋势

(2)搭建环境：Apache + PHP解释器 => XAMPP，编写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

(8)函数和对象

(9)第三方库和框架

(10)实用小项目

2.PHP中的变量和常量

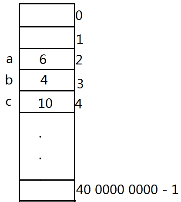
数学中：

a = 6

b = 4

c = a + b

a/b/c...称为变量，Variable——值可能发生改变的量。计算机中变量的本质：是内存中一个存储空间的别名。



PHP中声明一个变量：

$变量名 = 值 ; #值若是数字不加引号！字符串必须加引号！

输出一个变量的值：

echo $变量名 ;

|  |
| --- |
| 变量名的命名规则：  (1)变量名中可以有一个字符或多个字符，如$a、$empAge  (2)变量名中可以包含数字、字母(大小写)、下划线；但不能以数字开头！如： $age2、$my\_book、$\_newEmp；但不能写作：$2age、$emp\*age  (3)PHP是严格区分大小写的！ $emp、$Emp、$EMP是不同的变量名。 |

练习：创建一个变量，表示第一款笔记本的单价；再创建一个变量，表示第一款笔记本购买的数量；再创建一个变量，表示第一款笔记本金额的小计。输出该小计。

练习：创建一个变量，表示第二款笔记本的单价；再创建一个变量，表示第二款笔记本购买的数量；再创建一个变量，表示第二款笔记本金额的小计。输出该小计。

练习：创建一个变量，表示第三款笔记本的单价；再创建一个变量，表示第三款笔记本购买的数量；再创建一个变量，表示第三款笔记本金额的小计。输出该小计。

练习：创建一个变量，表示上述三款笔记的金额总和，输出该总和

常量：Constant，值永远不会发生改变的量

PHP中声明一个常量：

const 常量名 = 值 ;

输出一个常量：

echo 常量名 ;

|  |
| --- |
| 常量名中可以包含数字、字母、下划线，但不能以数字开头。  习惯上，为了和变量名区分，常量名用纯大写形式。 |

午间练习：

新建7.php页面，使用常量保持圆周率π。再使用程序分别计算30度角、60度角、70度角转换成弧度值各是多少？

角度转换为弧度公式： ? = deg \* PI / 180

|  |
| --- |
| MySQL中的列类型：  数值类型：  日期时间类型：  字符串类型： |

3.PHP中的数据类型 —— 共有三大类八小种

**(1)原生值类型**

int：范围-2147483648~2147483647，如果超出自动变为float。0开头的数字是八进制数；0x开头的是十六进制数。

float：也称为double，表示浮点小数 $score = 95.0;

string：字符串，必须使用单引号或双引号括起来

boolean：也称为bool，只能取值为true/false —— PHP中的true不等价于1，false也不等价于0！

**(2)复合类型**

array：

object：

**(3)其它类型**

null：

resource：

|  |
| --- |
| echo：只能输出变量的值  var\_dump()：可以输出变量的值和类型，变量检查 |
| PHP中单引号和双引号字符串有细微的差别：—— 重点掌握！  若字符串中只有普通的字符，则单引号和双引号无区别！  若字符串中有 $xxx，双引号会理解为一个变量名！ |
| bool中的true用echo输出为"1",  false用echo输出为"" —— 只能用var\_dump()才能看到真正的数据 |

练习：创建一个变量表示笔记本商品的名称，另一个变量表示笔记本的价格，使用一个echo 输出如下的内容：

笔记本名称：戴尔燃7000 笔记本价格：6500

练习：创建一个变量表示商城中的用户名，另一个变量表示用户的登录密码，一个变量表示用户的登录次数，一个变量表示用户当前是否在线。使用一个echo输出如下的内容：

用户名：dingding 登录密码：123456 登录次数：58 当前是否在线：1

4.PHP中的运算符 —— 稍微有点难度&零乱

运算符：可用于对变量和常量执行特定的运算

(1)算术运算符：

+ - \* / %(求模、取余) ++(自加) --(自减)

(2)比较运算符：

> >= < <= == !=

===(全等于，类型和值都相等，避免隐式类型转换问题)

!==(不全等，只要类型或值有一项不同，就判定为不全等)

注意：比较运算的结果是true或false

(3)逻辑运算符：

&&/AND(并且/与) ||/OR(或) !/NOT(非/取反)

&&: 两个式子都是true，总结果才是true；否则就是false

||：只要有一个式子是true，总结过就是true；否则就是false

!： 对true取反是false，对false取反是true

|  |  |
| --- | --- |
| 取余运算：用于判定一个数能否被另一个数整除(余数为0能整除，否则就不能整除)  echo 15/4 ; #3.75  echo 15%4; #除法，只要余数：3  echo 1500%45; #15  echo 9/4; #2.25  echo 9.1/4; #2.275  echo 9.9/4; #2.475  --------------------------------------------  echo 9%4; #1  echo 9.1%4; #1  echo 9.9%4; #1 #求余运算会直接忽略小数部分 | |
| $age = 20 ;  $age = $age + 1 ;  echo $age; #21 | $age = 20;  $age ++; #自加+1  echo $age; #21 |

练习：创建变量表示员工的月薪，每月底扣除个税30%，年底的年终奖20000，不交税，计算每个月平均到手的工资是多少

练习：

$i = 10; $j = $i++; echo $i, $j; #11, 10 #先取值再自加

$i = 10; $j = ++$i; echo $i, $j; #11, 11 #先自加再取值

练习：

$i = 10; $k = $i++ + ++$i + $i++; echo $i, $k;

$i = 10; $k = --$i + $i-- + --$i; echo $i, $k;

练习：判断下面两两大小比较的结果

123 > 3

'123' > '3' #隐式类型转换 string=>int

'abc' > 'ABC' #字符串比较的是字符的编码

true > false

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

123 == true #隐式类型转换 int=>bool

0 == false #隐式类型转换 int=>bool

'abc' == true #隐式类型转换 string=>bool

'' == false #隐式类型转换 string=>bool

null == false #隐式类型转换 null=> bool

课后练习：

(1)公园规定老人(满80岁)和儿童(不满6岁)可以免费入园。创建一个变量表示一位游客的年龄，根据此游客的年龄判断其是否可以免费入园。

(2)创建一个变量表示用户输入的登录名，一个变量表示用户的登录密码，如果用户名为root，且密码为123456，则登录成功；否则登录失败。判断此用户的输入能否成功登录。

(3)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：一个年份数字是否为闰年，只要满足下面两个条件之一即可：

1)该年份能被400整除；

2)该年份能被4整除，且不能被100整除。

PHP DAY02

复习：

自学一门新的编程语言：

(1)了解背景

(2)搭建环境

(3)变量和常量

$age = 20; echo $age;

const PI = 3.14; echo PI;

$uname = 'tom';

echo "用户名：$uname";

(4)数据类型

原生值类型： int、float、string、boolean

复合类型：array、object

其它类型：null、resource

(5)运算符

算术：+ - \* / % ++ --

比较：> >= < <= == != === !==

逻辑：&& || !

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方库、框架

(10)实用项目

|  |
| --- |
| 有基础学员的自学内容：  PHP官方手册——“语言参考”相关章节  PHP扩展内容：面向对象&第三方库  https://pan.baidu.com/share/link?shareid=3623474815&uk=171029850 |

今日目标：

(1)补充：运算符

(2)逻辑结构：选择结构 —— 重点

(3)逻辑结构：循环结构 —— 重点

1.PHP中的运算符

(1)算术运算符

(2)比较运算符

(3)逻辑运算符

(4)位运算符

<<(按位左移) >>(按位右移动)

(5)赋值运算符

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

(6)拼接运算符

. .=

$str= 'hello';

$str = $str . 'world';

$str .= 'world';

注意：PHP中+只用于算术运算，不能用于拼接！

(7)三目运算符

? :

语法：式1 ? 式2 : 式3

含义：如果式1为true，则返回式2结果，否则返回式3的结果（含义等同于if...else...）

|  |
| --- |
| 面试题：下面的四种方法，哪一种可以最快速的实现一个数字\*8的效果  a) $i = $i\*8 #可以实现，速度不是最快  b) $i \* 8; #不能实现  c) $i = $i<<8; #\*2^8即\*256  d) $i = $i<<3; #\*2^3即\*8  $i = 5; #int，占4个字节，计算机存储为：  00000000 00000000 00000000 00000101  把此数据按位左移1位，变为： 10 = 5\*2^1  0000000 00000000 00000000 000001010  把此数据按位左移2位，变为： 20 = 5\*2^2  000000 00000000 00000000 0000010100  把此数据按位左移3位，变为： 40 = 5\*2^3  00000 00000000 00000000 00000101000  把此数据按位左移4位，变为： 80 = 5\*2^4  0000 00000000 00000000 000001010000  ...  把此数据按位左移4位，变为： 5\*2^N |
| $age = 20 ;  $age = $age + 5 ;  $age += 5 ; #作用与上一句相同，速度更快 |
| 单目运算符： $a++  双目运算符： $i + $j  三目运算符： $a ? $b : $c |

练习：创建变量保存学生的语文和数学成绩，用一个echo输出该学生的总分，形如：

该学生总分为： 170

练习：创建变量保存商品单价和购买数量，用一个echo输出商品金额小计，形如：

商品金额小计：4500

练习：变量保存学生成绩，输出是否及格

练习：创建一个变量表示游客的年龄，公园门票价格2元，如果是老人和儿童票价打五折，否则打八折，输出该游客最终需要付的门票金额。

练习：创建一个变量表示论坛中用户的个人签名，若某用户签名为""，则自动赋值为“该用户很懒,什么都没留下”，输出该用户的签名。

练习：创建两个变量保存两个数字，使用三目运算符返回这两个数字中较大的那个值。

练习：变量保存学生成绩，输出：优秀、及格、不及格

练习：论坛中用户有积分，5000分评定为白金用户，5000~3000分评定为黄金用户，3000~1000评定为白银用户，否则评定为普通用户。创建变量表示用的积分，输出该用户的级别。

2.程序的逻辑结构

程序 = 数据 + 操作/算法

程序操作/算法由三种基本结构组成：

(1)顺序执行

(2)选择执行

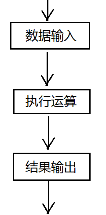
(3)循环执行

3.程序逻辑结构——顺序执行

顺序执行：1=>2=>3

示例：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。

程序执行步骤：顺序执行

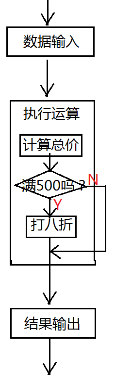


4.程序逻辑结构——选择执行（if...else...）

选择执行：有些代码会执行，有些则不会执行

示例：收银台程序V2，实现“满500打八折”功能

“如果.....就......”



PHP中的选择执行语法：

if( 条件判定 ){

#条件判定为true时执行的代码

}

|  |
| --- |
| 语句1;  if( $total >= 500 ){  语句2;  }  语句3;  如果条件为true： 语句1=>语句2=>语句3  如果条件为false： 语句1=>语句3 |

练习：公园规定：老人儿童可以半价票入园，其他人全票入园，票价原价为2元。创建变量表示人的年龄，输出其应付的票价。

练习：创建一个变量表示论坛中用户的个人签名，若某用户签名为""，则自动赋值为“该用户很懒,什么都没留下”，输出该用户的签名。

示例：收银台程序V3，实现“已付金额足够则输出找零；否则输出错误提示”功能

“如果......就.....； 否则就......”

if( 条件判定 ){

#条件为true时执行的代码

}else {

#条件为false时执行的代码

}

|  |
| --- |
| 语句1;  if( $total >= 500 ){  语句2;  }else {  语句3;  }  语句4;  如果条件为true： 语句1=>语句2=>语句4  如果条件为false： 语句1=>语句3=>语句4 |

练习：创建变量表示考试成绩，如果满60则输出及格，否则输出不及格。

练习：创建一个变量表示一个年份，如果是闰年，则输出“该年份有366天”，否则输出“该年份有365天”。

练习：创建变量表示用户输入的用户名和密码，如果用户名为root且密码为123456，输出“登录成功”，否则输出“用户名或密码错误”

练习:创建变量保存学生成绩，输出：优秀、及格、不及格

练习：学子商城中，订单状态可能取值和对应编码：

等待付款——10 备货中——20 运输中——30

派送中—— 40 已签收——50 未知状态—— 其它

在数据库中真实存储的是每个状态的编码，创建一个变量表示某个订单的状态编码，输出为对应的汉字

|  |
| --- |
| if选择结构可能的形式：  if( 条件 ){  } |
| if( 条件 ){  }else {  } |
| if( 条件 ){  }else {  if(){  }else{  }  } #情形较多时可能导致代码可读性差 |
| if( 条件1 ){  }else if( 条件2 ){  }else if( 条件3 ){  }else {  } #情形较多时推荐的写法 |

练习：创建变量保存学生成绩，输出：优秀、及格、不及格

练习：学子商城中，订单状态可能取值和对应编码：

等待付款——10 备货中——20 运输中——30

派送中—— 40 已签收——50 未知状态—— 其它

在数据库中真实存储的是每个状态的编码，创建一个变量表示某个订单的状态编码，输出为对应的汉字

5.程序逻辑结构——选择执行（switch...case...） —— 了解

switch：开关、切换

case：情形、场景

switch..case...：在多个不同的情形间切换，语法：

|  |  |
| --- | --- |
| $stat = 30;  switch( $stat ){  case 10: #stat==10  语句1;  **break; #打断，不再执行**  case 20: #stat==20  语句2;  **break;**  ....  default:  语句n;  } | $stat = 30;  if( $stat==10){  语句1;  }else if($stat==20){  语句2;  }else if(...){  .....  }else{  语句n;  } |
| switch..case...可以被if...else...代替，且只能进行相等判定，功能没有if...else...强大 | |

6.程序逻辑结构——循环结构 —— 重点&难点

循环结构：反复多次执行一样或类似的操作。如：打印100遍Hello、从Hello1打印到Hello100。

循环执行的二要素：

**(1)循环体：** 要反复执行的内容

**(2)循环条件：** 循环可以继续进行的条件

循环执行的第一种语法：

while( 判定条件 ){

#循环主体：要执行的语句

}

含义：当循环条件仍然满足时，就再次执行一次循环主体；直到循环条件为false，退出循环。

|  |  |
| --- | --- |
| $count = 0;  if( $count<10 ){  echo '需要再跑一圈';  }  echo '结束'; | $count = 0;  while( $count<10 ){  echo '需要再跑一圈';  }  echo '结束'; |
| 大括号最多执行一次 | 大括号可能执行很多遍 |

课后任务：—— 一起讨论完成！

(1)输出：1/2/3.../100

(2)输出：2/4/6.../98 —— 2星题

(3)输出：95/90/85/80..../5 —— 3星题

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

(4)输出：※

(5)输出：十个※+<br>

※※※※※※※※※※<br>

(6)输出：五行十列个※ —— 4星题

※※※※※※※※※※<br>

※※※※※※※※※※<br>

※※※※※※※※※※<br>

※※※※※※※※※※<br>

※※※※※※※※※※<br>

(7)输出： —— 5星题

※

※※

※※※

※※※※

※※※※※

PHP DAY03

复习：

自学新语言的基本步骤

(1)了解背景

(2)搭建环境

(3)变量和常量

(4)数据类型

原生值类型： int 、 float、 string、 bool

复合类型：array、object

其它类型：null、 resource

(5)运算符 —— 稍乱

算术：+ - \* / % ++ --

比较：> >= < <= == != === !==

逻辑：&& || !

位： << >>

赋值：= += -= \*= /= %=

拼接：. .=

三目：A?B:C

其它： , [] =>

(6)逻辑结构 —— 难点&重点

顺序执行： 步1=>步2=>步3

选择执行： if(){ }else{ } switch..case..break..

循环执行： while(){ }

(7)通用小程序 —— 有简单&有难

(8)函数和对象

(9)第三方库和框架

(10)实用小项目

|  |
| --- |
| 有基础学员扩展练习：  程序员必做的50道题 |

|  |  |
| --- | --- |
| if( 条件 ){  语句  } | while( 条件 ){  语句  } |
| 语句可能执行0/1次 | 语句可能执行0~N次 |
|  |  |

练习：

1.输出10个Hello

2.输出1/2/3/...100

3.输出2/4/6/...98

4.输出90/80/70/...10

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

5.使用循环计算出1+2+3+4+...100的累加和，输出该整数和

循环条件： $i——1~100

循环主体： $sum = $sum + $i ; $i++;

6.使用循环计算出1\*2\*3\*4\*...10的累乘积，输出该乘积

循环条件： $i——1~10

循环主体： $chengJi = $chengJi\*$i; $i++;

7.使用循环计算出90/5 + 80/10 + 70/15 + 60/20 + 50/25的和，输出该和

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

8.输出：

※

9.输出：

※※※※※※※※※※<br>

提示：把练习8循环执行10次

10.输出：

※※※※※※※※※※<br>

※※※※※※※※※※<br>

※※※※※※※※※※<br>

※※※※※※※※※※<br>

※※※※※※※※※※<br> 第i行有10列

提示：把练习9循环执行5次 —— 循环的嵌套

11.输出：

※ 第1行有1列

※※ 第2行有2列

※※※ 第3行有3列

※※※※ 第4行有4列

※※※※※ 第5行有5列 =>第i行有i列

12.输出：

1\*1=1 第i行有i列

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

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

4\*1=4 4\*2=8 4\*3=12 4\*4=16

今日目标：

(1)循环结构 —— do..while.. —— 掌握

(2)循环结构 —— for —— 重点

(3)数组 —— 重点

1.PHP中的循环结构 —— do..while...

do: 做，执行 while：当...的时候

do...while...： 执行XXXX操作当YYYY的时候。

|  |  |
| --- | --- |
| **while** | **do...while** |
| $i = 0;  while( $i<5 ){  echo '※';  $i++;  } | $i = 0;  do{  echo '※';  $i++;  }while( $i<5 ); |
| 先判定再执行 | 先执行再判定 |
| $i = 10;  while( $i<5 ){  echo '※';  $i++;  } | $i = 10;  do{  echo '※';  $i++;  }while( $i<5 ); |
| while的主体可能执行0~N次 | do...while的主体执行1~N次 |
| 结论：while和do...while一般情况下执行效果完全一样；仅在第一次都不能执行时有细微的差别——while至少执行一次 | |

练习：使用do..while输出10个Hello

练习：使用do..while输出1/2/3/...100

练习：使用do..while输出2/4/6/...98

练习：使用do..while输出90/80/70/...10

练习：使用do..while循环计算出1+2+3+4+...100的累加和，输出该整数和

练习：使用do..while循环计算出1\*2\*3\*4\*...10的累乘积，输出该乘积

.......

2.PHP中的循环结构——for循环 —— 最重点

for: “为了”，是最常用的循环结构

for( 语句1; 语句2; 语句3 ){

#循环主体

}

(1)语句1在整个循环开始之前执行，且仅执行一次——一般是循环变量的初始化语句

(2)语句2在每次主体执行之前执行，是一个判定语句：true时执行主体；false时退出循环——循环条件语句

(3)语句3在每次主体执行之后执行——循环变量的改变

(4)语句1和3中可以是用逗号分隔的多个语句

|  |  |
| --- | --- |
| $i = 0;  while( $i<5 ){  echo '※';  $i++;  } | for($i = 0; $i<5; $i++){  echo '※';  } |
| 结论：while和for的功能完全一样！只是for看起来更加紧凑。 | |

练习1)：输出10个Hello

练习2)：输出2000/2001/2002/....2999

练习3)：输出95/90/85/....5

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

练习4)：使用for计算出1+2+3+4+...100的累加和，输出该整数和

练习5)：使用for计算出1\*2\*3\*4\*...10的累乘积，输出该乘积

练习6)：使用for循环计算出90/5 + 80/10 + 70/15 + 60/20 + 50/25的和，输出该和

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

练习7)：输出1个※

练习8)：输出10个※，最后再加1个<br>

提示：把练习7循环执行10次

练习9)：输出5行10列个※

提示：把练习8循环执行5次 —— 循环的嵌套

练习10)：输出：

※

※※

※※※

※※※※

※※※※※

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

难题！练习11)输出本世纪(2000~2100)所有的闰年

2000 是吗？ 是就输出；

2001 是吗？ 是就输出；

...

2100 是吗？ 是就输出；

提示：在for循环中嵌入if判断

难题！！练习12)输出所有的“三位水仙花数”—— 一个三位数字，百位的立方 + 十位的立方 + 个位的立方 === 该数本身 ，如

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

思路：只能逐个去试：

100是吗？是就输出；

101是吗？是就输出；

...

999是吗？是就输出；

$i是水仙花数吗？先取出个位、十位、百位上的数字，再判定是否满足水仙花数条件，满足的就输出

难题！练习12)输出所有的“四位水仙花数”

难题！！！！！练习13)输出100以内所有的质数

for循环的变种：

|  |
| --- |
| for($i = 0; $i<5; $i++){  echo '※';  } |
| $i = 0;  for( ; $i<5; $i++){ #语句1可以为空白，分号不能少  echo '※';  } |
| $i = 0;  for( ; $i<5; ){ #语句3可以为空白  echo '※';  $i++ ;  } |
| $i = 0;  for( ; ; ){ #语句2可以为空白，等同于 for( ; true; ){ }  echo '※';  $i++ ;  } |

3.与循环相关的两个PHP关键字

continue：继续

如果在循环中出现continue，表示跳过此次循环后续的代码，继续执行下一次循环。

示例：输出1/2/3/4/6/7/8/9/11/12/13/14/16...99

for($i=1; $i<100; $i++){

if($i%5===0){

continue;

}

echo $i;

}

练习：输出1+2+5+7+10+11+13+14+17.......98

提示：计算累加和，跳过所有能被3或4整除的数

break：打断

break可用于switch..case..，表示打断不再执行switch中剩余的case代码；还能用于循环，表示打断整个循环，不再继续执行，调到循环外面。

示例：

for($i=1; $i<100; $i++){

if($i%5===0){

break;

}

echo $i;

} #只能输出： 1/2/3/4

练习：计算1\*2\*3\*4\*...\*100的累乘积，要求如果中间$chengJi超过21747483647，立即终止执行，输出此时的乘积值，以及$i的值

难题!!! 练习：输出本世纪所有的闰年，输出够10个即不再输出。

|  |
| --- |
| while( true ){  #让用户猜系统产生的随机数  if( 猜中 ){  break;  }  } |

4.PHP中的数据类型 —— 数组类型

PHP中如何保存1个员工的年龄： $age = 20;

PHP中如何保存100个员工的年龄：

$age1 = 20;

$age2 = 22;

$age3 = 31;

$age4 = 19;

.....

保存批量的数据不会创建批量的变量！而使用数组变量！

array：数组，数据的组合/集合

**创建一个数组：**

$数组变量名 = [ ]; #空数组

$数组变量名 = [ 值, 值, 值, 值 ];

|  |
| --- |
| 原生值类型：一个变量只能存一个数据  复合类型中的数组类型：一个变量可以存储任意多个数据 |

**查看数组中内容：**

echo $数组变量名; #错误！！

var\_dump($数组变量名); #正确！！

**获取数组中数据的个数(数组长度)：**

echo count($数组名);

**操作数组中的某个数据：**

echo $数组名[ 数据的下标 ] ; #输出某个数据

$数组名[ 数据的下标 ] = 值; #修改某个数据

练习：创建一个数组，保存5个员工的工资

练习：创建一个数组，保存4个用户的用户名

练习：创建一个数组，保存6个用户当前是否在线

结论：数组中可以存储任意类型的数据。

课后练习：

(1)创建一个数组，保存8个学生的考试成绩，使用for循环输出所有的数据元素。

(2)使用for循环为上述数组中的每个考试成绩+5分。

(3)再使用for循环输出修改后的所有数组元素。

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

难题！输出所有的“四位水仙花数”。

难题！！！！输出100以内所有的质数。

有余力的学员继续完成“程序员必做的50道题”以锻炼程序思维。

PHP DAY04

复习：

自学新语言的步骤：

(1)了解背景

(2)搭建环境

(3)变量和常量

$变量 = 值 ; echo $变量;

const 常量名 = 值 ; echo 常量名;

(4)数据类型

强类型语言： C/Java int age = 20;

弱类型语言： PHP/JS $age = 20;

值/标量类型： int float string boolean

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： A?B:C

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

打星星、九九乘法表、闰年、水仙花数、质数....

(8)函数和对象

(9)第三方库和框架

(10)实用小项目

今日目标：

(1)自定义数组 —— 项目中的重点

(2)PHP预定义数组 —— 重点&难点

(3)自定义函数 —— 了解

(4)PHP预定义函数 —— 项目中的重点

1.自定义数组

数组：array，一个可以保存多个数值的变量

**定义一个数组：**

$ageList = [ 20, 38, 25, 37 ];

$nameSet = [ 'Tom', 'Mary', 'King' ];

$isMarriedArr = [ true, false, false, true ];

**输出整个数组不能用echo，要用var\_dump();**

**获取数组中元素的个数(数组的长度)：**

echo count($ageList);

**操作某个元素：**

echo $nameSet[ 1 ];

$nameSet[ 1 ] = 'Maria';

**向一个数组中追加新元素：**

$nameSet[ ] = 'Scott'; #PHP会自动为新元素分配下标

|  |
| --- |
| **[ ]运算符的三种用法：**  (1)创建新数组： $arr = [ 10, 30, 50 ];  (2)访问某个已有元素： $arr[2] = 35;  (3)向数组中追加新元素： $arr[ ] = 75; |

练习：创建一个数组保存3个笔记本的标题，使用循环输出数组中的每个元素。再向数组中追加2个新的笔记本信息，使用循环输出数组中的每个元素。

PHP中数组的两种类型——重要知识点：

**(1)索引数组**：每个元素的下标是数字，形如0/1/2/3....

$arr = [10, 30, 50, 60];

$emp = [3051, 'Tom', 6000, true];

**(2)关联数组**：每个元素的下标是自定义的字符串

$emp = [ 'eid'=>3051, 'ename'=>'Tom', 'salary'=>6000, 'isMarried'=>true ];

结论：索引数组适合于存储多个相同含义的数据，如10个员工的姓名；关联数组适合于存储多个不同含义/类型的数据，如一个商品的多个信息。

练习：创建一个数组，保存学子商城中一个用户的信息，包含编号、用户名、性别、是否在线。

练习：创建一个数组，保存学子商城中一个用户的信息，包含编号、用户名、性别、是否在线。

练习：创建一个数组，保存学子商城中一个用户的信息，包含编号、用户名、性别、是否在线。

练习：创建一个数组，保存学子商城中一个用户的信息，包含编号、用户名、性别、是否在线。

练习：创建一个数组，保存学子商城中一个用户的信息，包含编号、用户名、性别、是否在线。

练习：创建一个数组，保存上述五个用户。

$userList = [ $user0, $user1, $user2, $user3, $user4 ];

$userList = [ [ ], [ ], [ ], [ ], [ ] ]; 二维数组

午间练习：循环执行把上述二维数组中的每个用户信息输出到一行中

2.PHP中的循环结构 —— foreach循环 —— 了解

foreach："为了每一个"，针对数组中的每一个元素，执行一次循环体——只能用于遍历数组。语法：

foreach( $数组名 as $变量名 ){

#循环主体

}

foreach( $数组名 as $下标变量名=>$元素变量名 ){

#循环主体

}

$arr = [ 10, 30, 50, 70 ];

foreach( $arr as $v ){

echo $v;

}

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

$v = 10; 执行一次循环主体;

$v = 30; 执行一次循环主体;

$v = 50; 执行一次循环主体;

$v = 70; 执行一次循环主体;

循环结束。

练习：创建一个数组，保存4个员工的姓名，使用foreach循环遍历出每个元素。 —— foreach可以遍历索引数组

练习：创建一个数组，保存一个笔记本信息，包括：编号、标题、价格，是否特价。使用foreach遍历输出每个元素的值。—— foreach可以遍历关联数组

|  |  |  |
| --- | --- | --- |
|  | **for循环** | **foreach循环** |
| 能遍历索引数组吗  $arr=[10, 30, 50]; | for($i=0; $i<count(...); $i++){  echo $arr[$i];  } | foreach($arr as $v){  echo $v;  } |
| 可以跳序遍历吗？ | for($i=0; $i<count(...); $i+=2){  echo $arr[$i];  } | 不能 |
| 可以倒序遍历吗？ | for($i=10; $i>=0; $i--){  echo $arr[$i];  } | 不能 |
| 能输出1~10吗？ | for($i=1; $i<=10; $i++){  echo $i;  } | 不能 |
| 能遍历关联数组吗  $u=['uid'=>10, 'uname'=>'tom'] | 不能 | foreach($u as $v){  echo $v;  }#项目中此用法很少！ |

3.PHP中的预定义数组变量

预定义变量：由PHP解释器自动为每个PHP文件创建的变量，程序员可以直接使用，无需自己手工声明。如下：

$\_REQUEST：请求( request )，用于封装客户端发给服务器请求数据

$\_GET：

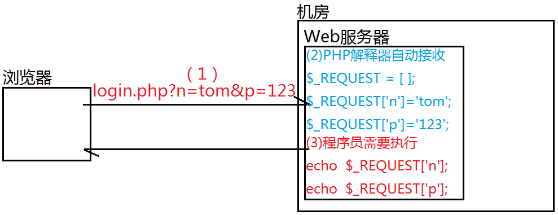
$\_POST：

$\_FILES：

$\_COOKIE：

$\_SESSION：

初始情况下，上述几个变量都是关联数组： array(0){ }



(1)客户端浏览器如何给服务器发送请求数据？——浏览器用户

http://127.0.0.1/login.php?n=tom&p=123

页面地址后的?k=v&k=v字符串称为“查询字符串”/“请求参数”

(2)WEB服务器会如何处理客户端提交的请求数据呢？——WEB服务器

$\_REQUEST = [ ];

$\_REQUEST['n'] = 'tom';

$\_REQUEST['p'] = '123';

(3)程序员编写的页面如何获取请求数据呢？——程序员

echo $\_REQUEST[ 'n' ];

echo $\_REQUEST[ 'p' ];

练习：创建add.php，客户端访问此页面时需要提交num1、num2和num3三个数据，服务器端页面接收这三个数据，向客户端输出三个请求数据的整数和。

练习：创建star.php，接收客户端提交的rows和cols两个请求数据，根据这两个数据输出指定行数和列数的※

练习：创建sum.php，接收客户端提交给服务器的start和end两个请求数据，服务器端页面输出从start到end的累加和

4.PHP中函数的基本概念

Function：函数，功能体，接收一些要处理的数据(称为参数)，加以执行处理，得到特定的结果 —— 并可以反复多次执行。

|  |
| --- |
| 程序员最怕的事情：Ctrl+C/V代码 —— 代码的可维护性差！  使用函数封装需要反复多次执行的代码！ |

**创建一个最简单函数：**

function 函数名( ){

#函数中要执行的所有代码

}

**运行/执行一次函数：**

函数名( ) ; #()表示“执行”

**创建一个可以接收运行参数的函数：**

function 函数名( $变量1, $变量2, .... ){ #形式参数

#函数中要执行的所有代码

}

**运行/执行一次函数：**

函数名( 值, 值, .... ) ; #实际参数

练习：创建一个函数getScore()，其中声明三个变量，分别表示语文、数学、英语的成绩，输出三科总分。调用此函数三次。

练习：创建一个函数printStar()，其中输出一个5行10列的※矩阵。调用此函数三次。

课后练习：

**函数练习：**

(1)创建一个函数getSum()，接收三个数字做参数，函数体中计算出这三个数字的代数和，并输出。调用此函数三次，每次调用传递不同的实参值。

(2)创建一个函数getAvg()，接收三个数字做参数，函数体中计算出这三个数字的平均值，并输出。调用此函数三次，每次调用传递不同的实参值。

**数组练习：**

(1)创建一个数组，保存学子商城中一个用户的信息，包含用户编号、用户名、密码、邮箱、手机、头像、性别；

(2)再创建一个数组，保存学子商城中一个用户的信息，包含用户编号、用户名、密码、邮箱、手机、头像、性别；

(3)再创建一个数组，保存学子商城中一个用户的信息，包含用户编号、用户名、密码、邮箱、手机、头像、性别；

(4)再创建一个数组，保存学子商城中一个用户的信息，包含用户编号、用户名、密码、邮箱、手机、头像、性别；

(5)创建一个空数组，将上述四个商品的数据保存到其中；输出这个大数组中的所有数据 —— 尽可能漂亮的输出。

PHP DAY05

复习：

**(1)数组——**一个变量中可以保存多个数据

索引数组： for foreach

$priceList = [0=>3500,1=>4800,2=>5300];

关联数组： foreach

$laptop =['id'=>1001, 'price'=>6500, 'isOnsale'=>true]

**(2)foreach**

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

echo $k . $v;

}

**(3)系统预定义数组变量：$\_REQUEST**

http://127.0.0.1/login.php?uname=ding&upwd=123

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

$p = $\_REQUEST['upwd'];

**(4)函数**

function: 函数，功能体，用于封装若干需要反复多次执行的语句。

function mysql\_connect( ) { .... }

mysql\_connect( );

function drawImage( ){ .... }

drawImage();

今日目标：

(1)复习：自定义函数

(2)调用PHP预定义的函数——访问MySQL数据库

1.函数的基础概念

**定义一个简单的函数：**

function 函数名( ){

#函数主体

}

**调用/运行一次函数：**

函数名( );

**定义带运行参数的函数： 形式参数**

function 函数名( $变量名1, $变量名2, ... ){

#函数主体

}

**调用/运行一次函数： 实际参数**

函数名( 值1, 值2, ... );

**定义有返回值的函数：**

function 函数名( $变量名1, $变量名2, ... ){

#函数主体

return 值 ; #返回函数的计算结果

}

**调用/运行一次函数：**

$变量 = 函数名( 值1, 值2, ... ); #接收函数的返回值

练习：创建函数 getSum($n1, $n2, $n3)，计算这三个数的和，返回计算结果。

练习：创建函数 getAvg($n1, $n2, $n3)，计算这三个数的平均值，返回计算结果。

试着调用上述函数

|  |
| --- |
| 函数总结：如何定义/调用？什么是参数？什么是返回值？ |

2.MySQL的客户端工具

(1)命令行工具： mysql.exe

(2)网页版本工具： phpMyAdmin

使用方法：启动Apache，启动MySQL，打开浏览器，在地址栏中输入http://127.0.0.1/phpmyadmin即可使用此客户端工具

3.使用PHP连接数据库服务器

提示：步骤与命令行工具连接MySQL服务器是一样的

(1)连接到MySQL服务器

mysqli\_connect( ... )

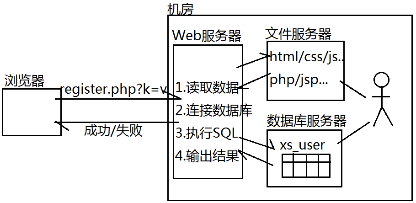
(2)提交SQL语句给MySQL服务器执行

mysqli\_query( ... )

(3)查看返回的执行结果

(4)关闭到MySQL服务器的连接

mysqli\_close( ... )



4.学子商城项目：用户模块 —— 基于单表的CRUD

CRUD: Create / Retrieve / Update / Delete 增删改查

用户模块需要的功能点：

(1)用户注册 —— INSERT

(2)修改用户信息 —— UPDATE

(3)用户删除 —— DELETE

(4)用户登录 —— SELECT(一条)

(5)用户列表 —— SELECT(多条)

5.完成学子商城用户模块 —— 用户登录功能点

(0)创建服务器端页面user\_register.php，用户访问此页面，提交注册信息

(1)编写PHP页面，读取客户端提交的请求数据：uname、upwd、email、phone。

(2)使用PHP连接MySQL服务器。

(3)向MySQL服务器提交一条INSERT语句。

(4)根据SQL语句执行结果，向客户端输出“注册成功/失败”。

|  |
| --- |
| PHP小知识：如何屏蔽一行代码产生的警告信息输出？  @代码内容; 在代码之前添加一个@符号 |
| PHP小技巧：如何终止页面的执行？  die( ) ;  die('终止执行的原因说明'); |
| PHP背景知识：  **mysql\_xxx()**：早期提供的一套函数  **mysqli\_xxx()**：Improved，提升版/改进版/性能优化版 |
| 小知识：获取刚执行的INSERT语句在数据库中产生的自增编号  $id = mysqli\_insert\_id( $conn ) ; |

6.完成学子商城用户模块 —— 用户删除功能点

(0)创建服务器端页面user\_delete.php，用户访问此页面，提交要删除的用户的编号

(1)编写PHP页面：读取客户端提交的请求数据：uid

(2)使用PHP连接MySQL服务器。

(3)向MySQL服务器提交一条DELETE语句。

(4)根据SQL语句执行结果，向客户端输出“删除成功/失败”。

|  |
| --- |
| 小知识：获取刚刚执行增删改语句影响的行数  $num = mysqli\_affected\_rows($conn); |

7.完成学子商城用户模块 —— 修改个人信息功能点

(0)创建服务器端页面user\_update.php，用户访问此页面，提交要修改的用户的编号、修改后的真实姓名、性别、邮箱、电话

(1)编写PHP页面：读取客户端提交的请求数据：uid、user\_name、gender、email、phone

(2)使用PHP连接MySQL服务器。

(3)向MySQL服务器提交一条UPDATE语句。

(4)根据SQL语句执行结果，向客户端输出“修改成功/失败”。

8.完成学子商城用户模块 —— 用户列表功能点

(0)创建服务器端页面user\_list.php，用户访问此页面

(1)无

(2)使用PHP连接MySQL服务器。

(3)向MySQL服务器提交一条SELECT语句。

(4)向客户端输出查询结果：一个二维数组。

|  |
| --- |
| 扩展小知识：所有的SQL语句分为如下四类：  **(1)DDL:** Data Define Language，数据定义语言——定义列  create drop alter(修改表的列) truncate(截断表)  **(2)DML:** Data Manipulate Language，数据操作语言——操作行  insert delete update  **(3)DQL:** Data Query Language，数据查询语言——对数据无影响  select  **(4)DCL:** Data Control Language，数据控制语言——与安全/权限  grant(授权) revoke(收权) |
| **$result = mysqli\_query($conn, $sql)的返回值说明：**  (1)若$sql是DML/DDL： 返回false-执行失败；返回true-执行成功  (2)若$sql是DQL：返回false-执行失败；返回“查询结果集的描述对象”-执行成功 |
| **重要知识点：如何从查询结果集描述对象中抓取查询数据**  (1) $rowList = mysqli\_fetch\_all( $result, 1 ): 抓取所有的数据行，得到一个二维数组  (2) $row = mysqli\_fetch\_row( $result ); 抓取一行数据，得到一个索引数组，或者null  (3) $row = mysqli\_fetch\_assoc( $result ); 抓取一行数据，得到一个关联数组(Association Array)，或者null |

9.完成学子商城用户模块 —— 用户登录功能点

(0)创建服务器端页面user\_login.php，用户访问此页面，提交登录名和登录密码

(1)编写PHP页面：读取客户端提交的请求数据：uname/upwd

(2)使用PHP连接MySQL服务器。

(3)向MySQL服务器提交一条SELECT语句。

(4)如果能查询到记录则向客户端输出“登录成功”，否则输出“用户名或密码错误”。

10.补充知识点：如何在一个PHP页面中使用其他页面中的语句

require('xx.php'): 在当前文件的当前位置引入另一个文件的全部内容。最典型的应用场景：

(1)创建数据库连接的公共配置文件

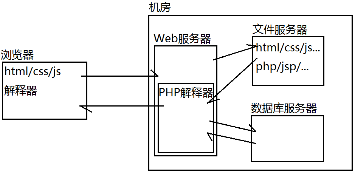
(2)全站中所有页面公用的页头和页尾

**好的学习方法是什么？——整理“前端100天”思维导图**

HTML 基础 DAY01

复习：

1.Web项目的部署结构



**静态Web技术(客户端技术)：**提供的内容任何人在任何时间访问都是一样的

HTML/CSS/JS/Flash....

**动态Web技术(服务器端技术)：**提供的内容不同人在不同时间访问都可能改变——一般需要访问数据库等

PHP/JSP/ASP.NET/Node/Python....

自学一门新语言的步骤：

(1)了解背景

(2)搭建环境

~~(3)变量和常量~~

~~(4)数据类型~~

~~(5)运算符~~

~~(6)逻辑结构~~

~~(7)通用小程序~~

~~(8)函数和对象~~

~~(9)第三方库和框架~~

(10)实用项目

结论：HTML编程语言非常简单！——只学2天！——在背单词，后续天天用肯定不会忘记！——记得整理思维导图！

今日目标：

(1)HTML语言

1.HTML语言概述

Hyper Text Markup Language：超文本标记语言，1991年产生，文件后缀为.html、.htm、.xhtml等。后来升级为HTML1.0 .... HTML4.0、XHTML、HTML5等多个版本。

世界上的第一个网页：

http://info.cern.ch/hypertext/WWW/TheProject.html

HTML语言的特点：简单、跨平台。

标记：标签、元素，指用<>括起来的特殊单词，能够被浏览器中的HTML解释器翻译为一种特殊的页面效果。

2.搭建HTML语言的运行环境

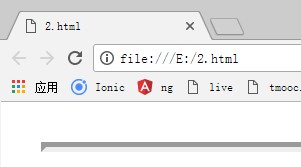
**标准情况下：**

服务器端：安装WEB服务器，编写并保存网页，启动WEB服务器

客户端：安装WEB浏览器，输入服务器的地址开始访问

**简单情况下：**

自己编写的HTML网页，可以直接拖拽到浏览器中显示——没有经过WEB服务器的处理——不推荐这样使用。



3.HTML基础语法

(1)HTML文档中可以包含任意字符，以及W3C/HTML标准预定义的标签。

(2)标签分为两类：

双标记标签： <标签名>内部内容</标签名>

单标记标签： <标签名/> 或简写为<标签名>

(3)开始标签上可以声明属性：

<标签名 属性名='值' 属性名="值">

如：<hr width='50%' color="red" **id**="shuipingxian2" **title**="" **class**="" **style**="">

(4)标签可以嵌套，但不能交叉

正确：<p>A<span>B</span>C</p>

错误：<p>A<span>B</p>C</span>

(5)HTML文档可以使用注释

<!-- ... -->

|  |
| --- |
| SQL中的注释： # /\*...\*/  PHP中的注释： #、// /\*...\*/  HTML中的注释： <!-- ... --> |

4.一篇HTML文档的标准结构

<! 文档类型声明 >

<html>

<head>...</head>

<body>...</body>

</html>

5.HTML标准中常用的标签 —— 重点！

|  |  |
| --- | --- |
| **标签名** | **含义** |
| <html></html> | 标识一篇HTML文档的开始和结束 |
| <head></head> | 定义文档的头部 |
| <body></body> | 定义文档的主体 |
| <title></title> | 只能声明在<head>标签内，标识文档的标题 |
| <meta> | 只能声明在<head>标签内，用于标识网页的“元数据” |

|  |
| --- |
| metadata：元数据，指描述数据的数据  {  empId: 101  empName: tom  isMarried: true  salary: 8000  } |

6. HTML标准中常用的标签 —— 文本相关标签——重点！

|  |  |
| --- | --- |
| **标签名** | **含义** |
| <b></b> | Bold，文本加粗显示 |
| <i></i> | Italic，文本斜体显示 |
| <u></u> | Underline，文本添加下划线 |
| <s></s> | Strike，文本添加删除线 |
| <sub></sub> | 文本显示为“下标字” |
| <sup></sup> | 文本显示为“上标字” |
| <h1></h1> | 一号标题字 |
| <h2></h2> | 二号标题字 |
| <h3></h3> | 三号标题字 |
| <h4></h4> | 四号标题字 |
| <h5></h5> | 五号标题字 |
| <h6></h6> | 六号标题字 |
| <p></p> | Paragraph，表示为文章中的一个段落，每段落必然重启一行 |
| &nbsp; 空格  &trade; 商标TM  &reg; 注册商标®  &lt; 小于号  &gt; 大于号  &times; ×号 | HTML转义字符 |
| <div></div> | 区块，最简单的区块元素，不附加任何样式 |
| <span></span> | 跨距，最简单的行内元素，不附带任何样式 |

练习：统计出来h1~h6标签默认字体大小是标准文本字体大小的几倍？

H1:2em、H2:1.5em、H3:1.17em、

H4:1em、H5:0.83em、H6:0.67em

小知识点：HTML块级标签和行内标签

所有的HTML标签可以分为两大类：

**块级元素(block)：**必须独立占据一整行，同一行中不允许存在其他内容

H1~H6、P、DIV

**行内元素(inline)：**可以与其他内容同处于同一行

B、I、S、U、SUB、SUP、SPAN

7. HTML标准中常用的标签 —— 图片和超链接——重点！

|  |  |
| --- | --- |
| **标签名** | **含义** |
| <img src="" alt="" width="" height="">单标记标签；IMG是一个行内元素！ | 在页面中呈现一幅图片  src：Source，指定图片的地址  alt：Alternative，指定图片无法加载时显示的提示文字  width：指定显示的宽度  height：不指定的话会与宽度等比例缩放，若指定了可以实现拉伸或压缩图片的效果 |
| <a href="" target=""></a> | Anchor：超链接、锚点，用于在不同的网页间跳转，或者是在一个网页的不同部分跳转  href：指定跳转的地址  target：指定如何打开目标地址，可取值为\_self(当前窗口)或\_blank(空白窗口) |

小知识：常见的图片格式

(1)jpg/jpeg：色彩细节丰富，适合于展示照片；只能是矩形。

(2)png：色彩比较丰富，支持透明。

(3)gif：色彩一般丰富，支持动画效果。

切记：网页中显示的图片一定不要太大！如1024\*768图片大约在xxKB或100+KB足够了！

小知识：图片地址的种类

URL：Unified Resource Locator，统一的资源定位符，互联网上所有的网页、图片、音视频都有一个唯一的地址，称为URL地址。

**绝对地址**：以协议名或//开头的URL地址

http://127.0.0.1:8080/HTML\_DAY01/img/1.jpg

https://www.baidu.com/img/bd\_logo1.png

http://tmooc.cn/script/img\_v2/logo\_v3.png

//127.0.0.1:8080/HTML\_DAY01/img/1.jpg

//www.baidu.com/img/bd\_logo1.png

//tmooc.cn/script/img\_v2/logo\_v3.png

**相对地址**：不以协议名或//开头，而是指定相对于当前资源的相对地址

img/1.jpg 在当前路径下寻找img目录

./img/1.jpg 在当前路径下寻找img目录

../img/1.jpg 在当前目录的上一级目录下寻找img目录

超链接A元素的两种用法：

(1)在不同的网页间跳转：

<a href="./2.html">跳转到2</a>

<a href="http://www.baidu.com">跳转到百度</a>

(2)在某个网页的不同部分间跳转：

<a href="#chapter3">跳转到第三章</a>

........

<h1 id="chapter3">第三章 大闹天宫</h1>

课后练习：

编写HTML网页，实现如下的页面效果：



HTML 基础 DAY02

1.表格

1.表格语法

1.标记

1.表格

<table></table>

2.行

<tr></tr>--table row

3.单元格/列

<td></td>--table data

练习：

1.创建一个4行4列的表格

2.在table标记中嵌套tr

3.在tr中嵌套td

2.属性

1.table属性

border:表格边框

width:宽度

height:高度

align:设置表格的水平对齐方式

left/center/right

bgcolor:设置表格的背景颜色

cellpadding:设置单元格的内边距（内容与td之间的间距）

cellspacing:设置单元格外边距（td边框外的间距）

练习：

在练习一的基础上将表格的样式设置如下:

1.设置表格尺寸为400\*400px

2.设置背景颜色为orange

3.设置表格水平居中对齐

4.设置内边距为5px

5.设置外边距为10px

2.tr属性

1.align 设置当前行内容的水平对齐方式

left/center/right

2.valign 设置当前行内容的垂直对齐方式

top/middle/bottom

3.bgcolor 设置当前行的背景颜色

练习：在练习二的基础上

1.设置第二行的数据靠右对齐

2.第二行背景颜色为紫色purple

3.设置第三行垂直方向靠上对齐

3.td属性

width:宽度

height:高度

align:水平对齐方式

valign:垂直对齐方式

bgcolor:背景颜色

colspan:跨列

rowspan:跨行

4.表格的复杂应用

1.行分组

可以将连续的几个行，划分到一个组中，并进行统一管理

1.表头行

表格中最上面一行进行分组的话，可以放在表头行中。

<thead></thead>

2.表主体行

表格中将若干行放在tbody中，进行统一管理

<tbody></tbody>

3.表尾行

将表格中最后一行进行分组管理的话，可以放在表尾行中。

<tfoot></tfoot>

2.不规则表格使用

1.跨列

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

2.跨行

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

注意：

如果表格进行了分组管理，不能将不同组设置跨行操作。

5.可选标记

1.表格标题

<caption></caption>

如果设置表格标题，则必须将<caption>放于<table>下的第一个子元素位置处。

2.行/列标题

<th></th>所有的td都可以用th取代

6.表格嵌套

表格的嵌套，在td中取嵌套另一个表格

2.列表

1.列表的作用

按照从上到下（从左到右）的方式来显示所有的数据，并且可以在数据前增加显示的标识。

2.列表组成

1.列表都是由"列表类型"和"列表项"来组成。

1.列表类型

有序列表：<ol></ol>--order list

无序列表：<ul></ul>--unorder list

2.列表项

用于展示数据（嵌套在列表类型中）

<li></li>--list item

练习：

创建有序列表，4个列表项，每个列表项中分别写上四大名著的名称

2.列表属性

1.有序列表

1.type 指定列表标识的类型。

取值：

1 数字，默认值

a 小写字母

A 大写字母

I 大写罗马数字

i 小写罗马数字

2.start 指定起始编号从第 几 位置开始排序。

2.无序列表

1.type 指定列表的标识类型

取值：

1.disc 实心圆点

2.circle 空心圆

3.square 实心方块

4.none 不显示标识

3.列表嵌套

在一个列表项（li）中去嵌套另一个列表.

语法：

<ol>

<li>

<ul>

<li>内容</li>

<li>内容</li>

</ul>

</li>

</ol>

3.定义列表

1.什么是定义列表

定义列表常用于给出一类事物或对名词的解释说明。

2.语法

1.<dl></dl> 表示定义列表

2.<dt></dt> 表示定义列表中要解释说明的名词

3.<dd></dd> 表示定义列表中对名词解释的内容

<dl>

<dt>名词</dt>

<dd>解释名词的具体内容</dd>

</dl>

使用场合：图文混排时使用

3.结构标记

1.结构标记的作用

用于描述整个网页的结构（取代div做布局）

提升标记的语义性和可读性。

2.常用的结构标记

1.<header></header>

作用：定义网页或某部分内容的头部。

2.<nav></nav>

作用：定义网页的导航链接部分。

3.<section><section>

作用：定义网页主体内容。

4.<aside></aside>

作用：定义页面中侧边栏的信息，靠近边缘。

5.<footer></footer>

作用：定义网页中偏底部的内容，比如：备案号，解释说明，联系方式等。

6.<article></article>

作用：定义与文字相关的内容，如：论坛帖子，微博条目，用户评论等。

4.表单

1.作用

1.提供可以与用户交互的可视化控件（用户名和密码输入框，登录按钮等）

2.收集用户信息并提交给服务器

2.表单的组成部分

1.前端部分

表单控件，与用户交互的可视化控件

2.服务器端部分

对收集的数据进行提交（提交数据的处理）

3.表单

标记：<form></form>

属性：

1.action

作用：定义表单被提交时发生的动作，通常定义的是服务器上处理程序的地址（url）,默认提交给本页

2.method

作用：指定表单数据的提交方式

取值：

1.get(默认值)

1.明文提交，待提交的数据会显示在浏览器的地址栏中。

2.安全性不高

3.提交数据有大小限制，限制2KB

4.向服务器要数据时使用get方式

2.post

1.隐式提交，提交的数据不会显示在地址栏中。

2.安全性较高

3.提交数据无大小限制

4.要传递数据给服务器时使用。

3.delete 一般禁用

4.put 一般禁用

3.enctype

作用：指定表单数据的编码方式，允许将什么样的数据提交给服务器。

取值：

1.application/x-www-form-urlencoded

默认值，允许将任意的字符提交给服务器（文件无法提交）

2.multipart/form-data

允许将文件提交给服务器

3.text/plain

只能将普通文本提交给服务器

4.表单控件

能够与用户进行交互的可视化控件。

1.分类

1.input 元素

2.textarea-多行文本域元素

3.select和option-选项框元素

4.其它元素

2.input元素

1.作用

在页面中提供各种各样的输入控件。如：文本框，密码框，单选按钮，复选框...

2.语法

1.标记<input>或<input/>

2.属性：

1.type 指定创建输入控件的类型

ex:

type="text"

type="password"

2.name 为控件定义名称，服务器端使用（必须值）

3.value 控件的值，服务器端使用

4.disabled 禁用控件，不能操作并且数据也不能提交给服务器。

无值属性，只要出现在标记中，就是禁用。

3.详解

1.文本框和密码框

文本框：<input type="text">

密码框：<input type="password">

属性：

1.maxlength

指定限制输入的字符数

2.readonly

只读，只能看，不能改，但允许被提交。

无值属性，直接将属性放在标记中。

3.placeholder

占位符，即默认显示在控件上的文本。

2.按钮

1.提交按钮

<input type="submit">

功能：将表单中的数据，提交给服务器程序使用

2.重置按钮

<input type="reset">

功能：将表单中的数据恢复到初试化状态。

3.普通按钮

<input type="button">

没有功能

注意：

属性：value 定义按钮上显示的文字。

3.单选按钮和复习框

单选按钮：<input type="radio">

复选框：<input type="checkbox">

属性：

name 除了定义控件的名称之外，还能起到分组的作用。

checked 设置默认被选中，无值属性。

HTML基础JavaScript DAY01

1.表单元素

1.input元素

1.隐藏域和文件选项框

1.隐藏域

<input type="hidden">

要提交给服务器的数据，但是不想展示给用户看可以放在隐藏域中。

2.文件选择框

<input type="file">

注意：

1.method属性的值必须为post

2.enctype属性的值必须为multipart/form-data

2.textarea元素

1.作用

允许输入多行数据的文本框。

2.语法

标记：<textarea></textarea>

属性：

name:定义控件名称，提供服务器使用

readonly:只读，无值的属性

cols:指定文本域的列数，即一行能显示多少英文字符（中文字符减半）

rows:指定文本域的行数，即默认显示多少行数据，超出rows的话会出现滚动条。

3.select元素

1.语法

1.<select></select>

作用：在页面中表示一个选项框

2.<option></option>

作用：表示选项

2.属性

1.select属性

1.name 定义选项框的名称

2.size 定义默认显示选项的数量，默认值为1

3.multiple 设置多选，无值属性

只有滚动列表支持多选。

2.option属性

1.value 定义选项的值

2.selected 设置默认选中项，无值属性

4.其它元素

1.label元素

作用：关联表单控件的文本

标记：<label></label>

属性：for

设置要与该元素关联的表单控件的id值

2.为控件分组

<fieldset></fieldset> 为控件分组

<legend></legend> 为分组指定标题

3.浮动框架

作用：允许在一个网页中，再引入其他的网页

语法：<iframe></iframe>

属性：

1.src 要引入的网页路径（url）

2.frameborder 浮动框架的边框，默认值1

3.width 宽度

4.height 高度

练习：

新建footer.html网页，在网页中写上版权说明，在index.html网页中通过iframe引入footer.html。

2.新表单元素

1.什么是新表单元素

在HTML5版本中新提出来的控件。

2.详解

1.电子邮件类型

<input type="email">

作用：表单在提交时，会验证数据是否符合email的规范(@)

2.搜索类型

<input type="search">

作用：提供了快速清除功能

3.url类型

<input type="url">

作用：提交时，验证数据是否符合url的规范（绝对路径的规范）

4.电话号码类型

<input type="tel">

作用：在移动端设置中，显示“拨号键盘”

5.数字类型

<input type="number">

作用：只能接收数字，并可以灵活调整数字的值

属性：

value:控制的值

min:能够接收的最小值

max:能够接收的最大值

step:每次调整数字时的变化范围

6.范围类型

<input type="range">

作用：提供一个滑块组件，允许用户选取指定范围的值。

属性：

value:控制的值，默认值。

min:能够接收的最小值

max:能够接收的最大值

step:范围的步长

7.颜色类型

<input type="color">

作用：提供一个颜色拾取控件

8.日期类型

<input type="date">

作用：提供一个日期选择控件

9.周类型

<input type="week">

10.月份类型

<input type="month">

3.JavaScript概述

1.什么是javascript?

JavaScript，简称js,是一种运行于js解释器/引擎中脚本语言。

运行环境：

1.独立安装的js解释器(node)

2.嵌入在浏览器内核中的js解释器

2.js的发展史

1.1992年Nombas公司为自己的软件开发了一款脚本语言ScriptEase,可以运行在浏览器中。

2.1995年网景开发了另一款脚本语言LiveScript,后期更名为JavaScript

3.1996年Microsoft在IE3.0版本克隆版本JSscript

4.1997年提交给ECMA(欧洲计算机制造商联合会)，定义了ECMAScript(ES5,ES6)

3.JS的组成

由以下三部分组成：

1.核心 (ECMAScript)

2.DOM(Document Object Model)

文档对象模型

提供了一组允许操作页面元素的函数和属性。

3.BOM(Browser Object Model)

浏览器对象模型

提供了一组允许操作浏览器的函数和属性。

4.js的特点

1.语法类似于c,java

2.无需编译，由js解释器直接运行

3.弱类型语言

4.面向对象的

4.JavaScript的基础语法

1.使用js

1.运行环境搭建

1.独立安装js解释器（node）

1.在命令行界面输入node 按回车键

2.输入console.log("打印输出的内容");

2.使用浏览器内核嵌入js的解释器

浏览器内核负责内容的渲染由两部分组成：

1.内容排版引擎-解析HTML和CSS

2.脚本解释引擎-解析javascript

1.直接在浏览器中输入js脚本

console.log();

2.将JS脚本嵌入在HTML页面中执行

1.HTML的元素的事件执行js脚本

事件：onclick-鼠标单击时要执行的操作

ex:

<input type="button" value="你点我试试？" onclick="console.log('点你咋地？！')">

2.在<script>标记中编写脚本并执行

网页的任意位置处，嵌入一对<script></script>标记，

在<script>中编写js代码。

js的输出方式：

控制台输出：console.log();

弹出框输出：alert();

网页中输出：document.write();

3.使用外部脚本文件(.js为后缀)

1.创建脚本文件

2.在文件中编写js脚本

3.在使用的网页中引入

<script src="js文件的路径">

//引入外部的js文件后，这对scirpt标记中不能再写js脚本

</script>

练习：

1.创建网页，在网页中添加一对<script>标记，以3级标题打印输出自己的姓名；

2.创建一个按钮，为按钮添加单击事件，在控制台打印输出自己的年龄；

3.使用外部js文件的方式，在网页中以2级标题输出笔记本名称和价格。

JavaScript DAY02

1.js的基础语法

2.js调试

1.F12调出控制台，查看提示错误及其位置。

2.出错时只影响当前代码块，不会影响其他代码块，后续代码块继续执行。

3.语法规范

1.js语句：可执行的最小单元

必须以 ; 结束

严格区分大小写

ex:console.log(); //正确

Console.log(); //错误

所有符号必须是英文标点符号

2.js注释

// :单行注释

/\*\*/ :多行注释

4.变量和常量

1.变量

1.声明变量

var 变量名;

ex:

var uname;

2.为变量赋值

变量名=值;

3.声明变量时直接赋值

var 变量名=值;

注意：

1.允许在一条语句中声明多个变量，用逗号隔开即可。

ex:

var user\_name="Hanmm",user\_age=18;

2.如果声明变量，但未赋值，则值默认是undefined。

3.声明变量时可以省略var,但不推荐。

2.变量名的命名规范

1.不允许使用关键字和保留关键词

var const if else for while int string...

2.允许包含字母，数字，下划线，$

3.不能以数字开头

4.最好见名知意

ex:

var a;

var uname;

5.尽量使用"小驼峰命名法"

var uname;

var userName;//小驼峰

var UserName;//大驼峰

var user\_name;//下划线

3.变量的使用

1.声明变量 未赋值 值默认是undefined

2.使用未声明的变量，则 报错。

3.取值操作-get

变量只要没出现在=的左边，一律是取值操作。

4.赋值操作-set

变量名出现在=的左边，一律赋值操作。

练习：

用三个变量来保存笔记本信息：

笔记本名称

笔记本价格

笔记本库存

并将笔记本信息打印在控制台：输出格式如下：

ex:

笔记本名称：ThinkPad E460

4.常用的使用

1.什么是常量？

在程序中，一经声明就不允许被修改的数据就是常量。

2.语法

const 常量名=值;

常量名在程序中，通常采用大写形式。

ex:

const PI=3.14;

练习：

1.创建一个常量PI,赋值3.14，并打印在控制台，

再将声明好的常量PI的值更改为31.4，尝试打印输出，观察结果。

5.数据类型

1.数据类型的作用

规定了数据在内存中所占的空间。

40：占4个字节 也就是32位

40.1

bit:位

8bit=1byte 字节

1024byte=1KB

1024kb=1MB

1024MB=1G

1024G=1T

2.数据类型详解

1.数据类型分类

1.原始类型（基本类型）

1.Number类型

数字类型

作用：可以用来表示32位的整数，也可以是64位的浮点数（小数）。

整数：

1.十进制

由0-9的数字组成，缝十进一

var age=25;

2.八进制

由0-7的数字组成，缝八进一

八进制中以0开头

3.十六进制

由0-9和A-F组成，缝十六进一

A：10

B：11

C：12

D：13

E：14

F：15

十六进制中以ox开头

浮点数（小数）：

小数点计数法：12.34

指数计数法：3.4e3（3.4乘以10的3次方）

2.String类型

字符串类型

作用：表示一系列的文本字符数据，如：姓名，性别，地址等。

每个字符在计算机中都有一个唯一的编码来表示该字符，该码称为Unicode码。

查找一个字符的unicode码：

"一".charCodeAt();//默认十进制

"一".charCodeAt().toString(16);//十六进制

674e-->unicode转换成汉字？

转义字符：\u

其它转义字符：

\n:换行

\t:制表符(缩进)

\":"

\':'

\\:\

字符串在使用过程中，需要使用''或""括起来。

3.Boolean类型

作用：在程序表示真或假的结果

取值：true或false

ex:

var isBig=true;

在参与到数字运算时，true可以当做1运算，false可以当做0运算。

ex:

var result=25+true;//结果：26

4.Undefined类型

作用：表示使用的数据不存在

undefined类型只有一个值，即undefined,当声明的变量未赋值时，该变量的默认值是undefined。

ex:

var num;

console.log(num);//结果undefined

5.Null类型

null用于表示尚不存在的对象。

null类型只有一个值即null,如果函数或方法返回的是对象，找不到该对象时，返回的通常是null。

ex:

document.getElementById("uname");

2.引用类型

2.数据类型转换

1.隐式（自动）转换

不同类型的数据在计算过程中自动进行转换。

1.数字+字符串：将数字转换为字符串

var num=15;

var str="Hello";

var result=num+str;//15Hello

2.数字+布尔：将布尔转换为数字

var num=15;

var isSun=true;

var result=num+isSun;//结果16

3.字符串+布尔：将布尔类型转换为字符串

var str="Hello";

var isBig=true;--》"true"

var result=str+isBig;//结果Hellotrue

4.布尔+布尔：将布尔类型转换数字

var isBig=true;

var isSun=false;

var result=isBig+isSun;//结果1

问题：

var num1=15;

var num2=17;

var str1="Hello";

1.str1+num1+num2

结果：Hello1517

2.num1+num2+str1

结果：32Hello

允许使用typeof()的函数检查变量的数据类型。

2.强制转换-转换函数

1.toString()

将任意类型数据转换为字符串

语法：var result=变量.toString();

ex:

var num=15;

var result=num.toString();

console.log(typeof(result));//string

2.parseInt()

将任意类型的数据转换为整数

如果转换不成功，结果为NaN(Not a Number)

语法：var result=parseInt(数据);

ex:

var num="123abc"; //结果123

var num1="abc123";//结果NaN

3.parseFloat()

将任意类型的数据转换为小数

如果转换不成功，结果为NaN。

语法：var result=parseFloat(数据);

4.Number()

将任意类型的数据转换为number

注意：如果包含非法字符，则返回NaN

语法：var result=Number(数据);

练习：

1.弹出一个输入提示框，输入金额，并保存在变量money中

ex:

var money=prompt("请输入金额：");

2.将money+10,再打印输出加10后的结果

3.注释掉第2步，使用typeof()检查money的类型。

4.将money转换为数字，再+10打印输出。

6.运算符和表达式

1.表达式：由运算符和操作数组成的式子就是表达式

任何一个表达式一定会有结果。

ex:

15+17

var x=y=10;

17<15

2.运算符

1.算术运算符

+,-,\*,/,%,++,--

5%2=1 5模2 余数是1

++:自增，只做+1操作

++在前，先自增，再运算

++在后，先运算，再自增

--:自减，只做-1操作

--在前，先自减，再运算

--在后，先运算，再自减

ex:

var num=5;

console.log(num++);//5

console.log(++num);//7

练习：

var num=5;

5 (6)6 6(7)

var result=num + ++num + num++ + ++num + num++ +num; 结果：42

(8)8 8(9) 9

2.关系运算符（比较运算符）

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

运算结果为：boolean类型（true,false）

问题：

1.5>"10" 结果：false

关系运算符两端，只要有一个是number的话，另外一个会隐式转换为number,再进行比较。

2."5">"10" 结果：true

关系运算符两端，如果string的话，则比较每位字符的unicode码，以unicode码不相等的两个字符比较结果作为整体结果。

"张三丰">"张无忌" 结果：false

"三".charCodeAt();

"无".charCodeAt();

3."3a">10 结果：false

Number("3a")--->NaN

NaN

注意：

NaN与任何一个数据做比较运算时，结果都是false。

NaN与任何数据做!=，结果永远都是true。

NaN与任何数据做==，结果永远都是false.

isNaN()函数:

作用：判断指定数据是否为非数字，如果不是数字，返回值为true，是数字的话返回值为false。

3.逻辑运算符

逻辑运算符：&&，||，!

!:取反

&&：并且，关联的两个条件都为true时，整个表达式的结果为true。

||：或者，关联的两个条件，只要有一个为true,整个表达式的结果为true。

短路逻辑与：&& 做条件判断

当第一个条件为false时，整体表达式的结果就是false,不再判断或执行第二个条件。

当第一个条件为true时，会继续判断或执行第二个条件。

短路逻辑或：||

当第一个条件true时，就不再执行后续表达式，整体结果为true。

当第一个条件false时，继续执行第二个条件或操作。

4.位运算符

<<,>>,&,|,^

右移是把数字变小，左移是把数字变大

按位与：& 判断奇偶性

任意数字与1做按位与 ，结果为1，则为奇数，结果为0，则为偶数。

var num=323;

var result=num & 1;

console.log(result);

按位或：| 对小数取整

将任意小数与0做按位或，结果则取整数部分。

var num=123.45;

var result=num | 0;

console.log(result);

按位异或：^ 用于交换两个数字

var a=3;

var b=5;

a=a^b;

b=b^a;

a=a^b;

console.log(a);

console.log(b);

二进制位数，逐位比较，不同则为1，相同则为0。

练习：

声明两个变量，分别保存两个整数。

用异或实现数字的换位。

输出方式如下：

console.log("换位前a:"+a);

console.log("换位后a:"+a);

课后作业：

1.从弹出框输入一个年份，保存在变量year中。

2.判断year年是否为闰年，并给出提示。

闰年：能被4整除但不能被100整除 或 能被400整除的年份

输出格式如下：

假设输入2000：提示"2000是闰年"

输入2001：提示"2001年不是闰年"

JavaScript DAY03

1.运算符

1.赋值运算符和扩展运算符

1.赋值运算符 =

2.扩展运算符

+=，-=，\*=，/=，%=，^=....

ex:

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

a=a+1;--> a++,a+=1,++a

实现两个数字的换位：

a^=b;

b^=a;

a^=b;

2.条件运算符

条件运算符又称三目运算符。

单目（一元）运算符，只需要一个操作数或表达式。

ex:++，--，！

双目（二元）运算符，需要两个操作或表达式

ex:+,-,\*,/,%,>,<,>=,<=,==,===,!=,!==,&,|,^,&&,||

三目（三元）运算符，需要三个表达式或操作数。

语法：

条件表达式？表达式1：表达式2；

先判断条件表达式的值，如果条件为true,则执行表达式1，如果条件为false,则执行表达式2。

ex:

从弹出框中输入成绩，如果成绩大于60，则输出及格，否则输出不及格。

练习：

从弹出框中输入成绩：

如果成绩等于100，则输出"奖励1000元"，

如果成绩>=90，则输出"奖励100元"，

如果成绩>=70，则输出"奖励10元素"，

否则，输出"请大家吃饭"。

2.函数

1.什么是函数

函数，function,也称为方法

函数-功能。

函数是一段预定义好，并可以被反复执行的代码块。

预定义：提前定义好，并非马上执行。

反复执行：允许被多次调用。

代码块：可以包含多条可执行的语句

使用过的函数：

parseInt();

parseFloat();

Number();

alert();

document.write();

2.定义和使用函数

1.普通函数的声明和使用

1.声明

function 函数名(){

//函数体-若干可执行的代码

}

2.调用函数

在任意javascript合法位置处通过

函数名(); 对函数进行调用。

3.要同时调用多个函数方式如下：

function test(){

calculate(); //调用函数1

print();//调用函数2

}

练习：

1.定义一个函数名称calaulate(计算)

在函数体内，定义两个变量,num1, num2并赋值为数字，计算两个数字之和，并打印在控制台。

2.在网页中添加一个按钮，单击按钮时调用函数。

2.带参函数的声明和使用

1.声明

function 函数名(参数列表){

//函数体-js语句

}

参数列表：可以声明1个或多个参数，多个参数间用逗号分隔。

声明函数时，声明的参数称为“形参”。

2.调用

函数名(参数值列表);

注意：尽量按照声明函数的格式进行调用

调用时，传递的参数值称为“实参”。

练习：

定义带参函数，函数体内计算两个参数的乘积，并打印输出。单击按钮时调用函数。

3.带返回值函数的声明和使用

1.声明

function 函数名(){

//函数体

return 值;

}

2.调用

允许使用一个变量接收函数的返回值

var result=函数名();

练习：

定义一个可以接收三个参数的函数，函数体内返回三个数字的和。并将结果打印在控制台。

3.函数的作用域

1.变量作用域

作用域表示的是变量或函数的可访问范围。

js中的作用域分为两种：

1.函数作用域：只在函数范围内有效

函数作用域又称为局部变量，只在声明的函数中有效

2.全局作用域：js代码的任何位置都有效

全局作用域又称全局变量，一经声明，任何位置都能使用。

1.不在function中声明的变量，为全局变量

2.声明变量不使用var ,无论任何位置声明，都是全局变量（不推荐）

2.注意

1.全局变量和局部变量冲突时，优先使用局部变量。

2.声明提前

js程序正式执行之前，function声明的函数，会将所有var声明的变量，都预读到所在作用域的顶部，但赋值还是保留在原位置。

函数同样也有声明提前的功能。

3.按值传递

1.什么是按值传递

原始数据类型，在做参数传递时，都是按照“值传递”的方式来传参。

值传递：真正传递参数时，实际上传递的是值的副本（复制出来一个值），而不是原始值。

\*以后不要把原始数据类型拿到函数中去修改值。

练习：

1.创建变量score,赋值为90；

2.再创建函数changeScore(num),在函数体内将num+5再输出。

3.调用changeScore函数,并将score以实参传入函数，再输出score的值。

3.函数作用域

分为两种：

1.局部函数

在某个function中声明的函数

2.全局函数

在最外层<script>定义的函数就是全局函数，全局函数一经定义，任何位置都能调用。

4.ECMAScript提供一组全局函数

1.parseInt();

2.parseFloat();

3.Number();

4.isNaN();

5.encodeURI();

URL:Uniform Resource Locator统一资源定位器，俗称：路径

URI:Uniform Resource Identifier(标识符)

作用：对统一资源标识符进行编码，并返回编码后的字符串。

所谓的编码，就是将地址中多字节的文字编码成单字节文字。

6.decodeURI();

作用：对已编码的URI进行解码，并返回解码后的字符串。

7.encodeURIComponent();

在encodeURI的基础上，允许对特殊字符进行编码。

8.decodeURIComponent();

允许对特殊字符解码。

9.eval();

作用：执行以字符串表示的js代码

练习：

定义一个函数，在函数体内接收弹出框输入的一段js代码（document.write("<h1>js全局函数</h1>")），并用eval函数来执行该段js代码。

5.递归调用

递归：在一个函数的内部再一次调用自己。

问题：想要求5的阶乘

5！ 5\*4\*3\*2\*1 -->5\*4！

4！ 4\*3\*2\*1 -->4\*3!

3！ 3\*2\*1 -->3\*2!

2！ 2\*1 -->2\*1!

1！ 1\*1

求n的阶乘，用一个函数表示？

function f(n){

/\*if(n==1){

return 1;

}else{

return n\*f(n-1)

}\*/

return n==1?1:n\*f(n-1)

}

效率：

在本次调用还未结束时，就开始下次的调用，本次调用就会被挂起，直到所有的调用都完成之后，才会依次返回，调用次数越多，效率越低。

练习：

以下数列（斐波拉契数列）

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

通过f(n)计算数列中第n个数字的值。

已知：

该数列中第一个数和第二个数都是1。

从第三个数开始，每个数字都是前两个数字的和。

f(1)=1

f(2)=1

f(3)=f(2)+f(1)

f(4)=f(3)+f(2)

...

f(n)=f(n-1)+f(n-2)

练习：

创建一个函数avg(),接收三个参数，返回这三个数的平均值。

要求：在avg()内部创建一个内部函数sum()用以计算所有数值的和。

3.分支结构

1.if结构

语法：

if(条件){

语句块;

}

注意：

条件尽量时boolean类型，如果不是的，以下条件值，会当做false

if(0){....}

if(0.0){...}

if(""){}

if(undefined){}

if(null){}

if(NaN){}

ex：

if(35){} 真

if后的{}可以省略

省略后，if只控制下面的第一条语句

建议：不要省略if后的{}

2.if...else...结构

语法：

if(条件){

语句块

}else{

语句块

}

3.if...else if...else

语法：

if(条件1){

语句块1;

}else if(条件2){

语句块2;

}else if(条件3){

语句块3;

}else{

语句块n;

}

练习：分三次从弹出框中输入年，月，日

判断该日期是该年的第？天 8分钟

ex:

输入年：2017

输入月：3

输入日：5

2017年3月5日是2017年的第64天

2:平年28天 闰年29天

1,3,5,7,8,10,12 各31天

4,6,9,11 各30天

ex:

2017年9月20 ？？

先累加1-8月 ，再加上20天

课后作业：

写完上面练习的同学用

switch...case 结构完成上面的练习（日期计算）

JavaScript DAY04

1.switch...case结构

1.作用（使用场合）

等值判断

2.语法

switch(值/表达式){

case 值1：

语句块1;

break;//结束switch结构，可选的

case 值2：

break;

....

default:

语句块n;

break;

}

注意：swith后的值或表达式，与case后的值进行比较的时候使用的===判断

ex：

弹出框中输入1-7之间的数字

输入1：今天吃红烧排骨

输入2：今天吃红烧丸子

输入3：今天吃红烧鱼

输入4：今天吃红烧猪蹄

输入5：今天吃红烧茄子

输入6：今天休息

输入7：今天休息

3.特殊用法

执行相同的操作时：

switch(值/表达式){

case 值1：

case 值2：

case 值3：

语句块1;

}

练习：

使用switch...case结构改版日期计算（昨天课后作业日期计算）5分钟

2.循环结构

1.特点

1.循环条件：循环的开始和结束

2.循环操作：要执行的相同或相似的语句

2.循环-while

1.语法

while(条件){

//循环体-循环操作

//更新循环条件

}

ex：打印输入1-100之间的数字

练习:打印1-100之间所有数的和 3分

2.循环的流程控制

1.break

作用：终止整个循环的运行

2.continue

作用：终止本次循环的运行，继续执行下一次循环

ex:循环从弹出框中输入信息，并且打印，直到输入exit为止。

3.循环-do...while

1.语法

do{

循环体

}while(条件);

执行流程：

1.先执行循环体

2.再判断循环条件

如果条件为真，则继续执行循环体

如果条件为假，则跳出循环体操作

练习1:

用do...while改版while循环输入信息操作3分

1.从弹出框中循环输入一段信息并打印在控制台

2.输入exit退出程序

练习2：猜数字游戏 8分钟

1.随机生成一个1-100之间的数字

Math.random(); 生成0-1之间的小数

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

2.让用户输入一个整数

如果输入的数比随机数大，提示“猜大了”

如果输入的数比随机数小，提示“猜小了”

否则：猜对了

允许循环猜数

3.输入exit也可以退出游戏

4.循环-for

1.ex

打印1-100之间的所有的数字

var i=1; //表达式1

while(i<=100){ //表达式2

console.log(i);

i++; //表达式3

}

2.for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

表达式3：更新循环变量

执行流程：

1.先执行表达式1

2.判断表达式2的结果（boolean类型）

3.如果条件为真，则执行循环体，否则退出

4.执行完循环体后，再执行表达式3

5.判断表达式2的结果（真继续，否则退出）

ex:

实现10！

10\*9\*8\*7\*6\*5\*4\*3\*2\*1

循环条件：

从1开始到10结束

循环操作：

计算每个数字累乘的结果

练习1：

定义一个函数，打印乘法口诀表中的任意一行

传参，传几，就打印第几行 8分

1\*1=1

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

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

1\*4=4 2\*4=8 3\*4=12 4\*4=16

...

每一行就是一个循环，乘号后面的数字就是你想打印的行数，行数其实就是你传的参数，乘号前面的数字永远都是从1开始到打印的行数（循环）。

练习2：

创建一个函数，计算打印以下数列的和

1/1+1/3+1/5+1/7.....+1/999的和

循环条件：从1开始，到999结束，分母以2递增

循环操作：累加 1/循环条件变量 的和

3.for三个表达式特殊用法

for(表达式1;表达式2;表达式3){}

1.省略表达式

三个表达式可以任意省略，分号不能省

但一定要在循环的内部或外部将表达式补充完整（一般省略表达式1）

2.表达式1和表达式3允许写多个表达式，用逗号隔开

ex:打印1-100的和

function printSum(){

for(var i=1,sum=0;i<=100;i++)

sum+=i;

}

3.循环嵌套

1.循环嵌套

在一个循环体的内部，又出现一个循环

for(var i=1;i<=100;i++){ //外层循环

for(var j=1;j<=100;j++){ //内层循环

}

}

外层循环走一次，内层循环走一轮

练习1：

在控制台中打印以下图像

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

//外层循环打印行数

//内层循环打印\*

练习2：打印乘法表

1\*1=1

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

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

1\*4=4 2\*4=8 3\*4=12 4\*4=16

......

4.数组

1.什么是数组

在一个变量中保存多个数据

数组是按照线性顺序来排列的-线性结构

数组中，除第一个元素外，每个元素都有一个直接的“前驱元素”

数组中，除最后一个元素外，每个元素都有一个直接的“后继元素”

2.声明数组

语法：

1.var 数组名=[];

ex： var names=[];

2.var 数组名=[元素1,元素2,元素3];

ex:var name=["孙悟空","猪八戒","沙和尚"]

3.var 数组名=new Array();

创建一个不包含任何元素的空数组。

4.var 数组名=new Array("孙悟空","猪八戒","沙和尚")

练习：

创建一个数组，该数组中保存3名学员的姓名

创建一个数组，该数组中保存3名学员的年龄

在控制台中，分两次打印两个数组的值

课后作业：

打印如图所示的图像

练习1：

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

练习2：

\*

\*\*\*

\*\*\*\*\*

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

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

JavaScript DAY05

1.数组的使用

取值和赋值操作都使用 数组名[下标]

1.为数组赋值

数组名[下标]=值;

取值：

数组名[下标]

ex:

var names=["小红","小明","小强"];

names[0]="铁扇公主";

console.log(names);

2.获取数组的长度

数组的长度：数组中元素的个数

属性：length

语法：数组名.length

3.配合循环，遍历数组中的每一个元素

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

i:表示每个元素的下标

names[i]:每个元素

}

4.length表示数组中即将要插入的元素的下标

var names=["小红","小明","小强"];

names[names.length]="范冰冰";

练习：不停的从弹出框中录入员工姓名，将所有录入的姓名保存在一个数组中（names）,直到输入exit为止,在控制台打印数组中所有的元素。

2.关联数组

php:$array=["names"=>"suwukong"]; 与js中不一致

以字符串作为元素下标的数组，就是关联数组

以数字作为下标的数组，就是索引数组

在js中的关联数组：

var array=[];

array["字符串下标"]=值;

ex：

array["西游记"]="古代神话故事";

array["红楼梦"]=["贾宝玉","林黛玉","刘姥姥"];

注意：

1.关联数组中，字符串下标的内容是不记录到length中

2.只能通过字符串做下标取数组中的值

3.for...in

遍历出任意数组中的字符串下标以及索引下标

语法：

for(var 变量 in 数组){

//变量：字符串下标 或 索引下标

//取值：数组名[下标]

}

练习：定义出一个关联数组，然后遍历出数组中的内容，并在控制台打印

4.冒泡排序

1.什么是冒泡排序

排序算法的一种，将一组数据进行排序，小的数字往前排，大的数字往后排。

var arr=[23,9,78,6,45];

往后越比次数越少

5个数字 比较4轮

6个数字 比较5轮

10个数字 比较9轮

...

特点：

1.n个数字，则比较n-1次(外层循环)

for(var i=1;i<arr.length;i++)

2.轮数增加，比较次数减少(内层循环)

for(var j=0;j<arr.length-i;j++){

if(arr[j]>arr[j+1])

//交换arr[j]和arr[j+1]的位置

}

3.嵌套循环解决冒泡排序的问题

5.数组的常用方法

1.toString()

作用：将一个数组转换为字符串，并返回转换后的结果

语法：var str=数组对象.toString();

2.join()

作用：将数组的元素通过指定的字符连接到一起，并返回连接后的字符串

语法：var res=数组对象.join("指定的字符");

ex:

var arr=["吕布","西施","东施","小乔"];

用#连接

var res=arr.join("#");

console.log(res);

练习：

1.使用数组实现由\*组成的三角形

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

3.concat()

作用：拼接两个或者更多的数组，并返回拼接后的结果

语法：var res=数组对象.concat(数组1,数组2,...)

注意：

1.concat不会改变原有数组，而是返回拼接以后的结果

2.要拼接的数据可以是数组，也可以是普通数据

练习：

声明两个数组分别存放一些国家的名字

数组1：大国

l=["中国","美国","俄罗斯"];

数组s：小国

s=["阿富汗","日本","韩国"];

将小国全部移入到大国的数组中，打印输出。

4.slice()

作用：截取子数组，从指定的数组中，截取几个链接的元素组成一个新的数组

语法：var arr=数组名.slice(start,[end]);

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

0 1 2 3

var arr=["中国","美国","俄罗斯","加拿大"];

-4 -3 -2 -1

end:指定结束位置处的下标（不包含）,该参数可以省略，如果省略的话，就是从start开始一直截取到尾

注意：

该函数不会影响现有数组，会返回一个全新的数组

ex:

var arr=["中国","美国","俄罗斯","加拿大"];

要求：

分别使用正数和负数两种方式

取出美国和俄罗斯组成新的数组并打印出来

5.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,e3...);

start:指定要添加或删除元素的起始下标位置

count:指定要删除的元素个数，取值为0表示不删除

e1:要增加的新元素

e2:要增加的新元素

e3:要增加的新元素

返回值：返回一个由删除元素所组成的新数组

注意：该函数会改变现有数组

6.reverse()

作用：将一个数组反转

语法：数组名.reverse();

注意：该函数会改变当前数组的内容

ex:

声明一个数组，内容随意

倒序打印出该数组的内容

7.sort()

作用：排序，默认情况下，按照数组元素的Unicode码进行升序排序

语法：数组名.sort();

注意：该函数会改变当前数组的内容

特殊:允许自定义排序函数，从而实现对数字的升序或降序的排序

语法：数组名.sort(排序函数);

ex:

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

//排序函数（升序）

function sortUp(a,b){

return a-b;

}

//排序函数（降序）

function sortDown(a,b){

return b-a;

}

//为数组指定排序函数

arr.sort(sortUp);

arr.sort(sortDown);

原理：

1.指定排序函数，sortUp,定义两个参数，表示数组中的相邻的两个数字。

2.将排序函数指定给数组的sort()方法，数组会自动传递数据到sortUp()中，如果返回值>0的话，则交换量个数字的位置，否则不变。

/\*

var arr=[12,23,17];

arr.sort(sortUp);

12 23

function sortUp(a,b){

return a-b;

}

12 17 23

\*/

使用匿名函数完成升序：

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

使用匿名函数完成降序：

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

匿名函数：是一个没有名称的函数

只用一次函数时，可用匿名函数。

练习：

声明一个整数数组，随意定义数字

1.在页面上添加两个按钮，一个为"升序"，一个为"降序"

2.点击升序按钮时，数组按升序排序，并打印

3.点击降序按钮时，数组按降序排序，并打印

AJAX DAY01

1.进出栈操作

js是按照标准的“栈式操作”访问数组（访问数组的一种方式）

所有的栈式操作的特点都是"后进先出"

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];

//向数组中增加新元素

var newArr=arr.push(40);

console.log(newArr)

2.pop()

出栈，删除（删除栈顶数据）并返回删除元素

ex:

var arr=[10,20,30];

var r1=arr.pop();

console.log(r1); //30

var r2=arr.pop();

console.log(r2); //20

3.shift()

删除数组头部的（第一个）元素并返回删除元素

var arr=[10,20,30];

var re=arr.shift();

4.unshift()

作用：在数组的头部（第一个）元素位置处，增加新元素，并返回新数组的长度

var arr=[10,20,30];

var res=arr.unshift(40); //4

结果：arr=[40,10,20,30]

2.二维数组

1.什么是二维数组

在一个数组中的元素又是一个数组，也可以称为数组的数组

2.二维数组的使用

var names=[

["孙悟空","猪八戒","沙和尚"],

["曹操","大乔","小乔"],

["贾宝玉","林黛玉","薛宝钗"]

];

//获取 "大乔"

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

练习：

根据省份信息，查询城市列表 8分

1.创建一个数组，保存省份名称

2.创建一个二维数组，保存上面省份中的城市信息（城市信息要与上面省份相对应）

3.创建一个函数，接收一个省份名称作为参数，打印该省份对应的所有城市列表

1.ajax

1.URL

1.URL的作用

用于表示任意一个资源的位置（互联网上）

2.详解

格式：

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

scheme:方案\协议，以哪种方式到服务获取资源，协议不区分大小写，常见的协议：http,https,ftp

ssh:安全的远程登录

SMTP:邮件传输

POP3：邮件接收

DNS：域名解析

TELNET:远程登录

host:主机名，服务器主机名或IP地址或域名（由DNS转换为IP地址） 127.0.0.1=localhost

port:端口号 80 443

user:用户名，访问某些特定资源时需要用到的信息

pwd:密码，访问某些特定资源时需要用到的密码

path:路径，资源在服务器上具体存放位置

params:参数（跟服务器有关）

query:查询字符串，要传递给服务器的数据

http://127.0.0.1/login.php?uname=dangdang&upwd=12345

frag:锚点

http://127.0.0.1/a.html#NO1

http://img12.360buyimg.com/babel/jfs/t13804/61/581101319/94660/bead4bc3/5a0eaa1aNa2c916a2.jpg

2.HTTP协议

1.什么是HTTP

Hyper Text Transfer Protocol:超文本传输协议

规范了数据如何打包以及传递

2.详解

1.请求（request）消息

客户端带给服务器的数据都有哪些，由三部分组成

1.请求的起始行

1.请求方法

1.GET

表示客户端向服务器获取资源时使用

特点：

1.无请求主体

2.靠地址栏传递查询字符串

2.POST

表示想传递数据给服务器时使用

特点：

1.有请求主体

3.PUT

表示客户端想放置文件到服务器（禁用）

4.DELETE

表示客户端要删除服务器端的数据（禁用）

5.HEAD

表示客户端只想获取指定的响应头

6.CONNECT

测试连接

7.TRACE

追踪请求路径

8.OPTIONS

选项，保留以后使用

2.请求URL

3.协议版本：HTTP/1.1

2.请求头

1.Host:localhost/127.0.0.1

作用：告诉浏览器请求哪一个主机

2.Connection:keep-alive

作用：告诉服务器要进行持久连接

3.User-Agent:

作用：告诉服务器自己（浏览器）的类型

4.Accept-Language:zh-cn

作用：告诉服务器自己能接纳的自然语言

5.Accept-Encoding:gzip

作用：告诉服务器自己能接收的数据压缩类型是什么

6.Referer:http:localhost/Day01/login.html

作用：告诉服务器请求来自哪个页面

3.请求主体

Form Data

2.响应（response）消息

1.响应起始行

1.协议版本号：HTTP/1.1

2.响应状态码

作用：告诉浏览器，服务器的响应状态是什么问题（有问题，没有问题，有问题\问题大概是什么）

1xx:100-199 提示信息

2xx:成功响应

200:ok

3xx：需要进行重定向

301：永久性重定向

302：临时重定向

304：Not Modified

4xx:客户端请求错误

404：Not Found 请求资源不存在

403：Forbidden 权限不够

405：Method Not Allowed 请求方法不被允许

5xx:服务器运行错误

500：服务器内部错误

3.原因短句

对状态码的简单解释

2.响应头

1.Date

作用：告诉浏览器，服务器的响应时间

格林尼治时间（+8h）

2.Connection

作用：告诉浏览器已经启动持久连接

3.Content-Type

作用：响应主体的类型是什么，告诉浏览器，用什么样的方式解析响应主体

1.text/html:响应回来的数据是html文本

2.text/plain:响应回来的数据是普通文本

3.text/css:响应回来的数据是css样式

4.application/xml:响应回来的数据是xml格式

5.application/javascript:响应回来的数据是js脚本代码

6.application/json:响应回来的数据是json格式的字符串

7.images/jegp:响应回来的数据是图片

3.响应主体

Response

3.缓存

1.什么是缓存 & 工作原理

客户端将服务器响应回来的数据进行自动保存，当再次访问时，直接使用保存的数据。

2.缓存的优点

1.减少了冗余数据的传输，节省客户端流量

2.可以节省服务器带宽

3.降低对服务器资源的消耗和运行要求

4.降低了由于远距离而造成的延时加载

3.与缓存相关的消息头

1.Cache-Control消息头

作用：从服务器将文档传到客户端时起，可以认为此文档处于新鲜的秒数

语法：

Cache-Control:max-age=处于新鲜的秒数

ex:

Cache-Control:max-age=3600;

Cache-Control:max-age=0;每次都从服务器下载新资源

2.Expires消息头

作用：指定缓存过期的确切时间（格林尼治时间）

语法：Expires:Thu,23 Nov 2017 GMT

如果希望客户端不缓存，可以给一个过期的时间

Expires:Wed,22 Nov 2017 00:00:00 GMT

Expires:0;---此方法不标准，执行性不是特别好

4.在网页上设置消息头

1.更改服务器配置

2.网页上增加消息

<meta http-equiv="消息头" content="值">

ex:

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

或

<meta http-equiv="Expires" content="0">

php:

header("Expires:0");

Response.AddHeader("Expires","0");

4.DOM操作

1.什么是DOM

1.DOM：Document Object Model

文档对象模型

让js动态的操作页面的元素

<input type="text">

<div>这是一个div</div>

2.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"></div>

<input type="text" id="uname">

2.在js中，允许通过元素的id来获取页面指定的元素

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

elem就是指定id元素在js中的表现形式（DOM对象）

练习：

1.创建一个网页

2.在网页中创建一个div，id为d1,内容自定义

3.在页面中创建一个p，id为test,内容自定义

4.在页面中创建一个按钮，点击按钮时，将id为d1的元素获取出来，同时将id为test的元素也获取出来，并打印在空控制台，观察结果。

3.获取/设置元素中的内容

<div id="d1">这是一个div元素</div>

DOM对象的一个属性：innerHTML

表示的就是标签内的文本

ex：获取div标签中的文字内容

1.获取div元素

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

2.获取div元素中的文本内容

console.log(d.innerHTML);

3.更改div元素内的文本内容为"这是更改后的文本"

d.innerHTML="这是更改后的文本";

练习：

1.在上面练习的基础上完成以下操作

1.添加一个按钮，点击时，获取div中的文本，并打印在控制台

2.再添加一个按钮，点击时，将div中的文本内容更改为“这是我的第一个DOM操作”

4.获取/设置表单控件的数据

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

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

<input type="radio" value=""> ...

<select>

<option value="0">北京</option>

<option value="1">天津</option>

</select>

在DOM中，表单控件们，是通过DOM对象的value属性来获取/设置表单控件的值

步骤：

1.通过id获取表单控件元素

2.通过DOM对象的 value 属性获取/设置控件的值

ex:<input type="text" value="请输入用户名">

练习：

1.页面中创建一个文本框id="uname",再创建一个div元素id="d1",再创建一个按钮

2.当点击按钮时，将文本框的数据，显示在div中

AJAX DAY02

1.简化document.getElementById()

该方法是根据指定的id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);

};

调用：$(id)

练习：

1.创建一个common.js的文件，封装$(),负责接收一个id值作为参数，并返回对应的DOM对象

2.在网页中创建一个文本框，一个div,一个按钮

3.点击按钮时，将文本框的数据以二级标题的形式显示在div中（$()）

2.HTML元素的事件

1.什么是事件

允许通过用户的行为来激发的操作就是事件

常见事件：onclick(单击事件)

2.其它事件

1.文本框与密码框-onblur(失去焦点)

onblur事件：失去焦点事件

2.文本框与密码框-onfocus

onfocus事件:获取焦点的事件

练习：

1.创建一个网页

2.页面中创建如下几个元素：

1.文本框-用户名称：uname

2.密码框-用户名称：upwd

3.密码框-确认密码：cpwd

以上三个元素，各创建一个span元素

3.添加以下事件

1.文本框获取焦点时，提示“用户名名称6-18位”

2.文本框失去焦点时，判断，用户名称是否为空并给出提示（用户名不能为空/通过）

3.upwd获取焦点时，提示“密码6-18位”

4.upwd失去焦点时，判断，密码框是否为空给出相应提示

5.cpwd失去焦点时，判断cpwd的值是否与upwd的值一致，并给出提示

3.body的onload事件

当网页加载时要执行的操作

方式1：<body onload="函数()">

方式2：window.onload=function()

4.select的onchange事件

当选项框中的内容发生改变时要触发的事件

3.AJAX

1.名词解释

1.同步（Synchronous）

在一个任务进行中时，不能开启其它的任务

同步访问：浏览器在向服务器发送请求的时候，浏览器只能等待服务器的响应，不能做其它的事情。

出现场合：

1.地址栏输入网址访问页面（www.baidu.com）

2.a标记默认跳转

3.submit按钮的表单提交

2.异步（Asynchronous）

在一个任务进行中时，可以开启其它的任务

异步访问：浏览器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

使用场合：

1.用户名的重复验证

2.聊天室

3.股票走势图

4.搜索框建议（百度，京东）

2.AJAX

1.Asynchronous Javascript And Xml

异步的 js 和 xml

2.本质:

使用js提供的异步对象（XMLHttpRequest）,异步的向服务器发送请求，并接受响应数据（数据格式是Xml）

AJAX请求中，服务器响应回来的数据部分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的局部内容。

3.创建XMLHttpRequest对象-异步对象（xhr）

标准创建：

var xhr=new XMLHttpRequest();

IE8以下：

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

允许通过window.XMLHttpRequest 来判断浏览器是否支持标准创建，如果浏览器不支持标准创建，那么window.XMLHttpRequest的值就是null

if(window.XMLHttpRequest){

var xhr=new XMLHttpRequest();

}else{

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

}

练习：

1.创建一个xhr对象，并且将创建好的对象打印在控制台上，观察结果 5分

2.创建一个函数，creatXhr()

3.在函数体内，判断浏览器是否支持标准创建，如果支持返回标准创建的对象，否则返回创建好的IE8以下的异步对象（返回值）

4.XHR常用方法和属性（重点）

1.open()

作用：创建请求

语法：xhr.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，网页中创建一个按钮，单击时，完成以下功能：

1.获取xhr对象

2.创建一个请求，采用异步的方式，get的请求方式，向response.php发送一个请求

3.创建response.php,响应输出一句话“我的第一个AJAX程序”

2.readyState 属性

作用：表示xhr对象的请求状态

值：0-4表示5个状态

0：请求尚未初始化

1：已经打开到服务器的链接，正在发送请求中

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，才表示所有的响应都已经接收完毕。

3.status 属性

作用：表示的是服务器的响应状态码

值：

记住一个值 ：200

当status的值是200的时候，表示服务器已经正确的处理请求以及给出响应。

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

(xhr对象的状态在做一些改变时，这个事件会一直监视着它)

语法：

onreadystatechange=function(){

//判断xhr的readyState为4并且xhr的status值为200，就可以获取/响应数据了

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

//接收响应回来的结果

var resText=xhr.responseText;

console.log(resText);

}

}

5.send()

作用：发送请求

语法：xhr.send(body)

body:请求主体

如果没有请求主体，body位置处为null(get)

如果有请求主体，则放请求主体数据到body位置(post)

5.发送异步请求的步骤

1.创建xhr对象

2.创建请求

3.设置xhr的onreadystatechange(回调函数)

判断状态，并接收响应回来的数据

4.发送请求

练习：

1.创建一个php文件，响应输出一句话“我的AJAX练习”

2.创建网页，在网页中增加一个按钮，一个div元素id为show,单击按钮时，异步向php文件发送一个请求，并将响应回来的数据以h1标题方式显示在div中。

6.使用get提交方式，发送请求数据（带参数）

http://127.0.0.1/login.php?uname=value1&upwd=value2

练习：

1.创建一个网页

添加一个文本框框（id=uname）

添加一个按钮

单击按钮时，将文本框中的数据作为参数，异步的提交给ajax-get.php页面，并且将响应回来的数据显示在一个div里面

url="ajax-get.php?uname="+value1;

2.创建ajax-get.php

1.接收异步请求提交过来的数据

2.将数据拼接成“欢迎XXX”,再响应给浏览器

AJAX DAY03

1.使用ajax发送post请求

注意两点：

1.post的请求将数据放在请求主体中

xhr.send(body);

ex:

xhr.send("uname=value1&upwd=value2");

2.在发送请求之前，需要手动修改请求消息头

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

练习：

使用ajax的post请求方式完成异步的注册-参考学子商城-注册页

1.创建一个register.html页面

包含以下表单控件

uname-<input type="text">

登录名重复性验证，失去焦点时验证数据库中是否存在该用户名

upwd-<input type="password">

email-<input type="email">

phone-<input type=text>

user\_name-<input type="text">

gender-<select>

注册按钮：<input type="button">

单击按钮时，使用post提交方式，异步发送register.php，并且完成注册给出响应（注册成功/注册失败）

2.js对象数据格式

1.js对象的数据格式

var lindaiyu=[];

lindaiyu[0]="林黛玉";

lindaiyu[1]="1990";

lindaiyu[2]="160cm";

lindaiyu[3]="50kg";

改版：

var lindaiyu=[];

lindaiyu["name"]="林黛玉";

lindaiyu["birth"]="1990";

lindaiyu["height"]="160cm";

lindaiyu["weight"]="50kg";

ex:取值

lindaiyu["name"]

数组描述的是林黛玉的个人信息，如果你把林黛玉看成是一个对象，那么她的个人信息就是她这个对象的属性。

js对象的语法：

var 对象名={

属性：值,

属性：值,

属性：值,

属性：值

};

ex:林黛玉这个对象所对应的属性如下

var lindaiyu={

name:"林黛玉",

birth:"1990",

height:"160cm",

weight:"50kg",

}

如果对象取值的时候，直接对象.属性 名称就可以

ex:lindaiyu.name --->林黛玉

练习：

1.创建一个对象，贾宝玉

2.该对象包含三个属性

name,值为 贾宝玉

height,值为 178cm

gender,值为 男

3.控制台上分别将贾宝玉的三个属性输出

2.JSON

1.什么是JSON

JavaScript Object Notation

js 对象 表现方式

js对象表示法，即以js对象的格式表现出来的字符串。

2.JSON语法

1.JSON对象

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（单引号不可以），值如果是字符串的话，必须也用""引起来。

ex:

var computer='{

"name":"电脑",

"price":5600

}'

2.JSON数组

1.普通数组

'["小乔","大乔","貂蝉"]'

2.对象数组

'[

{

"name":"小乔",

"height":"160cm",

"age":18

},

{

"name":"大乔",

"height":"163cm",

"age":20

},

{

"name":"貂蝉",

"height":"165cm",

"age":21

}

]'

3.JSON文件创建

以.json为后缀的文件，里面包含的是符合json格式的数据

4.将JSON字符串，转换成js对象/数组

var mperson='{"name":"TOM","age":18}';

//json对象

var arr='["小乔","大乔","貂蝉"]';

//json数组

var arr1='[

{"name":"Lucy","age":19},

{"name":"Lily","age":19}

]'; //json数组

如何把上面的数据转换成js对象/数组格式

1.使用eval()将数据转换成js对象数组（不推荐）

2.使用JSON.parse()来将JSON字符串解析为js对象

var obj=JSON.parse(mperson); //js对象

var obj=JSON.parse(arr);//js数组

var obj=JSON.parse(arr1);//js数组

练习：

1.创建一个user.json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,gengder属性，分别赋值

2.在网页中使用异步请求的方式向json文件发送请求，并按找以下格式输出

用户名称：XXX

用户密码：XXX

用户性别：X

用户名称：XXX

用户密码：XXX

用户性别：X

用户名称：XXX

用户密码：XXX

用户性别：X

PROJECT DAY01

复习：

1.使用AJAX发送POST请求

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

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

2.JS对象数据格式

var 对象名={

属性：值,

属性：值,

属性：值

}

取值：对象名称.属性

3.JSON数据格式

var person='{

"name":"TOM",

"age":12

}'

4.将JSON格式的数据转换js对象/数组

1.eval()

2.JSON.parse

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

1.在php中，可以直接将数组转换成json格式的字符串

语法：

通过json\_encode()将数组转换为JSON字符串，并返回转换后的结果

ex:

在php中

$array=["钉钉","当当","冰冰"];

$str=json\_encode($array);

练习：

1.php中读取学子用户表中的数据，并将结果转换为关联数组，输出给前端时注意转换为json格式

2.网页中用户表的数据展示

2.XML

AJAX:Asynchronous Javascript And 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过，需要自定义

XML的宗旨是做数据传递的，而非显示数据

2.XML的语法结构

XML可以独立保存为\*\*\*.xml的文件，也可以以字符串的形式出现

1.XML的最顶端是XML的声明

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

2.XML标记的语法

1.XML标记必须成对出现

<person> 错误

2.XML严格区分大小写，开始和结束必须一致

<person></person> 正确

<Person></person> 错误

3.XML的标记允许被嵌套，注意嵌套顺序

<person>

<name>

<FirstName></FirstName>

<LastName></LastName>

</name>

</person>

4.每个标记都允许自定义属性，格式与html一致，但属性值，必须用""括起来

<person no="1001"></person>

5.每个XML文档，必须有一个根元素

3.解析XML文档对象的内容

1.核心方法

elem.getElementsByTagName("标签名称");

返回值：返回一个包含指定元素们的“类数组” （用for循环遍历）

ex:var xmlDoc=xhr.responseXML;

xmlDoc.getElementsByTagName("Student");

4.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字符串，再响应给前端

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

$xml.="<StudentList>";

...

$xml.="</StudentList>";

echo $xml;

练习：完成用户登录

1.创建一个02-login.html(非ajax)

提供以下控件

1.登录名称-文本框

2.登录密码-密码框

3.登录按钮

2.创建02-login.php

接收前端提交过来的数据u，判断用户登录是否成功，并给出提示。

==============项目

1.完成用户注册

1.创建一个01-register.html（非ajax）

提供以下控件（表单）

//注意元素name属性

1.登录名称-文本框

2.登录密码-密码框

3.确认密码-密码框

4.用户邮箱-电子邮件

5.联系方式-文本框

6.用户姓名-文本框

7.用户性别-下拉框

8.注册按钮 ============8分钟

2.创建01-register.php

接收01-register.html提交过来的数据，并插入到数据库，再给提示（注册成功/注册失败）

3.完成用户名称的重复性验证（异步，检查数据库中是否已存在当前用户名）5分钟

4.两次密码的验证 4分钟

当重复密码框失去焦点时，验证两次输入的密码是否一致，并给出提示（通过/，密码不一致）

PROJECT DAY02

1.复习：

1.在php中，将数组转换成json字符串

$array=["鼠标","笔记本","钢笔"];

$str=json\_encode($array);

2.XML语法规范

xml未预定义，需要自己定义

是做数据传递，展示数据

1.规范

1.声明

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

2.语法结构

1.必须成对出现

2.严格区分大小写

3.可以嵌套，注意嵌套顺序

4.必须的根元素

5.也可以自定义属性（与HTML中写法一致，但属性值必须用“”括起来）

2.解析XML内容

getElementsByTagName("标签名");

返回类数组（类似于数组）

3.在php中返回xml格式字符串

在php中拼接xml字符串

1.必须设置响应的消息头

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

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

2.拼接字符串

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

$xml.="<StudentList>"

$xml.="<Student>"

...

$xml.="</Student>"

$xml.="</StudentList>"

echo $xml;

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

项目：

1.查询用户表，显示如图所示

1.创建03-user-list.php文件

1.获取用户表里的数据 5分钟

1.数据库连接

2.拼sql语句

3.执行查询，并得到结果

4.将查询结果转换为关联数组

5.将关联数组转换为JSON格式并响应

2.创建03-user-list.html 3分钟

1.创建一个表格的表头

<thead></thead>

2.创建表主体

<tbody id="mybody"></tbody>

2.实现删除

点击删除这条数据立马没了，删完当前这条数据后还得加载用户列表

//删除指定的用户

参数uid:要删除的用户ID

<a href="javascript:deteleUser(user.uid)"></a>

1.在php中根据前端传递过去的uid完成删除

2.在用户列表页03-user-list.html,完成异步删除

function deleteUser(uid){

//用户ID作为参数传递给后端

//异步删除

}

3.新建05-sel-update.php文件

注意：

1.首先新建一个html模版页，保存时将文件的后缀更为.php

2.在当前.php文件的顶部写php代码

2.1根据传递过来的uid去查询当前用户的信息

2.将查询的用户数据显示在表单控件中。

3.执行修改动作

创建一个05-update.php文件

在此文件中执行修改数据（根据传递过来的uid）

课后练习：

1.把今天讲的项目自己再写一遍

2.尝试做用户列表的分页

1.当前想看的页数时第几页 currentPage

2.每一页显示的条数 pageSize 10条

3.总条数 totalSize

一共显示几页:totalSize/pageSize 4页

CSS DAY01

项目：

1.分页查询

SELECT \* FROM xz\_user limit 0,5 第一页

SELECT \* FROM xz\_user limit 5,5 第二页

SELECT \* FROM xz\_user limit 10,5 第三页

1.当前页数：currentPage

2.每一页显示的条数：pageSize

每页首条记录下标=(currentPage-1)\*pageSize;

(1 -1)\*5

(3 -1)\*5

练习1：

创建06-list-page.php 4分钟

1.声明变量currentPage,表示当前的页数

如果前端没有传递过来currentPage的值，那么默认为1

2.声明变量pageSize,表示每页显示的条数

如果前端没有传递过来pageSize的值，那么默认为10

3.根据pageSize和currentPage,来计算当前页显示的首条记录的下标，将结果保存在$start中

练习2：

1.在06-list-page.php中，根据$pageSize以及$start到数据库中查询数据

练习3：

1.计算尾页，即总页数 6分钟

1.获取xz\_user的总记录数，保存在$total(总条数)

查询总条数：select count(\*) from xz\_user

2.根据$total,$pageSize,计算总页数保存在$totalPage中

$totalPage=ceil($total/$pageSize)

ex: 8.5-->9

练习4：

1.把昨天的用户列表页面（03-user-list.html）另存为06-list-page.html

2.在06-list-page.html这个网页中找到table标签，然后在table中添加一对tfoot标记，并在tfoot中添加一行一列（合并6六列colspan="6"）

<tfoot>

<tr>

<td colspan="6"></td>

</tr>

</tfoot>

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

HTML元素的属性，修饰元素时又弊端：

1.相同的样式，使用不同的属性来完成

<body text="red">

<font color="red">

2.相同的操作，要重复的写

<font color="red">

<font color="blue">

1.CSS的概述

1.什么是CSS?

CSS：Cascading Style Sheets层叠样式表，级联样式表（简称：样式表）

2.作用

设置HTML网页元素的样式

3.HTML与CSS的关系

HTML：负责内容的展示

CSS：负责内容（元素）的修饰

4.HTML与CSS之间的使用原则

W3C建议尽量使用CSS属性去取代HTML属性来修饰元素

2.CSS语法规范

1.使用CSS样式的方式(重点)

1.内联样式

又称为行内样式

特点：将CSS样式定义在HTML开始标记中

语法：

<ANY style="样式声明1；样式声明2"></ANY>

样式声明：

1.由样式属性和值来组成

2.属性名与值之间用 冒号 连接

3.多个样式声明之间用 分号 分割

常用的CSS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）

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为单位的数字

ex：font-size:30px;

练习：创建一个CSS-basic.html

1.创建一个div标记，内容随意

2.设置该标记的样式如下：

文本颜色为黄色

背景颜色为红色

文字大小为24px

2.内部样式

在网页的头元素中增加一对<style>标记，在<style>标记声明该网页用到的样式规则

语法： <head>

<style>

/\*注释\*/

样式规则1

样式规则2

...

</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定义好的样式（就是把声明好的样式匹配给页面中的元素）

元素选择器：由元素的名称作为选择器

div,p,h1,span,a,img

选择器{}

ex：div{}

p{}

样式规则：

选择器{

样式声明；

}

ex:

div{

color:red;

font-size:20px;

}

p{

color:blue;

}

h1{...}

练习：

1.创建02-css-basic.html

2.创建若干个p标记，内容随意

3.创建若干个h1标记，内容随意

4.使用内部样式完成以下操作：

1.页面所有p标记文本颜色为蓝色，字体大小为32px

\*\*\*用外部样式为p标记设置背景颜色为灰色。

2.页面所有h1标记文本颜色为灰色，背景颜色为橙色

3.外部样式

独立于任何网页的位置处，声明一个样式表文件（\*\*\*.css为后缀），在.css文件中保存样式规则，然后在网页中引入.css文件。

使用步骤：

1.创建样式表文件，并编写样式规则

2.在网页中引入样式表文件

<head>

<link rel="stylesheet" href="\*\*.css">

</head>

2.CSS样式特征

1.继承性

大部分样式可以被继承（子元素继承父元素的样式特征）

必须是有层级关系的嵌套

<div style="color:red;">

<p>p</p>

</div>

2.层叠性

可以为一个元素定义多个样式，当样式属性不冲突时，可以同时将这些样式应用到元素上

div{

color:red;

}

div{

font-size:20px;

}

div{

background:gray;

}

3.优先级

如果样式声明冲突时，会按照样式的优先级来应用定义的样式规则

由低到高：

浏览器默认设置 最低

内部样式和外部样式 中(就近原则)

内联样式 最高

4.调整显示的优先级

!important规则：

调整显示的优先级

将!important添加在属性值之后，与值之间用空格隔开，就能优先使用当前样式

ex:

color:red !important;

练习：

新建一个网页，在网页中创建一个div元素，内容随意

1.用内部样式设置div的文本颜色为绿色，字体大小为30px

2.用内联样式设置div的文本颜色为灰色，然后运行网页查看页面文本效果

3.在绿色值后面添加!important,再运行网页查看效果

3.CSS基础选择器（重点）

1.选择器的作用

规范页面中哪些元素能够使用定义好的样式

2.选择器详解

1.通用选择器

作用：可以修饰页面上的任何元素

语法：\*{样式声明}

效率较低，尽量少用

ex:

\*{

color:red;

font-size:40px;

}

2.元素选择器

作用：设置页面上某种（类）元素的样式

语法：标记名称{声明样式}

ex：

div{}

p{}

span{}

3.类选择器

作用：定义页面上某个或某些元素的样式(谁想用谁就可以引用)

特点：通过元素的class属性进行引用

语法：

1.声明

.类名{样式声明}

注意：

1.类名是自定义的，但是注意类名不能以数字开头

2.类名不能包含特殊符号（&,^,%,$,#,@）

3.可以包含（\_,-）

2.引用

<ANY class="类名">

特殊用法：

1.多类选择器

让一个元素同时引用多个类选择器

语法：

<ANY class="类名1 类名2 类名3 ...">

2.分类选择器

将元素选择器和类选择器联合使用

对同一类元素中某些特殊的内容进行修饰

语法：元素名称.类选择器{样式声明}

ex:div.text{color:red;}

<div class="text">dddd</div>

<div>d1d1d1</div>

练习：3分钟

1.新建一个网页，创建div元素和p元素，内容随意

2.用类选择为所有元素设置字体颜色为红色

3.用分类选择器为p元素设置背景颜色为黑色

课后作业：

1.创建网页，在网页中添加所有结构标记，内容随意

2.在外部样式文件中用通用选择器定义所有内容的文字颜色为红色

2.在外部样式中用类选择器定义header和section的背景颜色为灰色

3.用分类选择器定义section的字体大小为50px

CSS DAY02

1.选择器

1.通用选择器

2.元素选择器

3.类选择器

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

4.id选择器

作用：设置指定ID元素的样式（专属定制）

语法：#id值{样式声明}

ex:

<div id="one"></div>

#one{

color:red;

}

练习：

新建网页，并创建p元素，内容随意，id为main

用id选择器设置p元素的样式：

文字大小为36px；

背景颜色为紫色（purple）；

文本颜色为黄色。

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,...{样式声明}

ex:

div,#main,.mycolor,p.text{color:red;}

等同于：

div{color:red};

#main{color:red};

.mycolor{color:red};

p.text{color:red};

6.后代选择器

作用：通过元素的后代关系匹配元素（多级嵌套）

语法：选择器1 选择器2 选择器3{样式声明}

7.子代选择器

作用：通过元素的子代（一层层级关系）关系匹配元素

语法：选择器1>选择器2{样式声明}

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

选择器：伪类选择器{样式声明}

1.连接伪类

:link 匹配元素尚未访问的状态

:visited 匹配元素访问过的状态

2.动态伪类

:hover 匹配鼠标悬停在元素上时的状态

:active 匹配元素被激活时的状态（超链接，文本框，密码框点击的时候）

:focus 匹配元素获取焦点时的状态（文本框和密码框）

练习：页面中增加一个input文本框，默认字体效果为灰色，字体为斜体（font-style:italic;）

input{

color:gray;

font-style:italic;

}

1.当激活文本框时，字体为非斜体

font-style:normal;

2.当获取焦点时，文本颜色为红色

3.选择器的优先级

权值：标识当前选择器的重要程度，权值越大优先级越高。

元素选择器：1

类选择器： 10

伪类选择器：10

ID选择择器：100

内联样式： 1000

选择器的权值加到一起，大的优先

权值相同，以后定义的为主

2.尺寸与边框

1.单位

1.尺寸单位

1.px：像素

1024\*768

2.in:英寸

1in=2.54cm

3.pt：磅(1pt=1/72in)

多数用于表示文字的大小

4.cm:厘米

5.mm:毫米

6.em:相对于父元素乘以倍数(多个父元素2em)

7.rem:根相对（元素字体大小乘以倍数，html\body）

2.颜色单位（颜色取值）

1.英文单词

red,blue,gray,green,yellow,black....

2.rgb(r,g,b)

r:0-255

g:0-255

b:0-255

3.rgba(r,g,b,alpha)

alpha:透明度，取值为0-1之间的小数，值越大，不透明度越高

4.#rrggbb

由6位16进制的数字\字母表示一个颜色

0-9或A-f

#000000:黑色

#ffffff:白色

#eeeeee:灰色

#ff11aa

5.#rgb是上面的缩写形式

#000:黑色

#fff:白色

#f1a

2.尺寸属性

1.作用

设置元素的宽度和高度

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.页面中哪些元素允许设置尺寸属性

1.所有的块级元素都允许设置尺寸

div,p,h1,h2..h6,ul,ol,dl,结构标记

2.本身具备width和height属性的行内元素是可以设置的

img,table

3.行内块允许设置尺寸

大部分的表单控件（单选按钮，复选框）

4.大部分的行内元素是无法设置尺寸

a,span,b,i,u,s等

3.溢出处理

当内容多，元素区域小的时候，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，溢出内容不可见

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

练习：如果所示效果 6分钟

border:1px solid black;

4.边框

1.边框属性

1.简写方式

border:width style color;

width:边框的宽度，以px为单位的数值

style:边框的样式

取值：

solid:实线

dotted:虚线边框(点)

dashed:虚线边框(线)

color:边框的颜色

取值：合法的颜色值

transparent:透明色

注意：

取消边框：border:0;或border:none;

练习：创建一个元素，id为container,尺寸为200\*200像素，并且设置四个方向的边框为1px,边框为实线条颜色为橙色。

2.单边定义

只设置某一条边的边框

属性：border-方向:width style color;

方向：top/bottom/left/right

上 下 左 右

练习：在上面练习的基础上，设置上边框为2px,blue,dotted的样式。

3.单属性定义

只设置四条边框的一个属性

属性：border-width/style/color:值；

ex:border-width:3px;

border-style:dotted;

border-color:red;

练习：在上面练习的基础上边框全部修改为虚线（dashed）

4.单边单属性的定义

只设置某一个方向的某一个属性

属性：

border-方向-属性：值；

方向：top/bottom/left/right

属性：width/style/color

ex：

border-left-color:blue;

border-right-style:solid;

border-bottom-width:6px;

2.边框倒角

将元素的直角倒换成圆角

属性：border-radius

取值：

1.以px为单位的数值

2.百分比 % 设置圆形（50%）

练习：

创建一个网页，添加一个div元素，设置其尺寸为200\*200px

1.设置倒角为5px,运行网页观察效果

2.设置倒角为50%，再观察效果

3.设置元素的宽度为400px,观察结果

单角设置：

border-top-left-radius:左上角

border-top-right-radius:右上角

border-bottom-left-radius:左下角

border-bottom-right-radius:右下角

3.边框阴影

属性：box-shadow

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

h-shadow:阴影在水平方向的偏移距离，必须值

取值为正：阴影向右偏移

取值为负：阴影向左偏移

v-shadow:阴影在垂直方向的偏移距离，必须值

取值为正：阴影向下偏移

取值为负：阴影向上偏移

blur:阴影模糊距离，取值越大，模糊效果越明显，以px为单位的数值（可选值）

spread:阴影的大小，指定要在基础阴影上扩充出来的大小，取值以px为单位的数值（可选值）

color:阴影颜色（可选值）

inset:将默认的外阴影改为内阴影（可选值）

练习：

在溢出处理练习基础上完成以下操作：

1.每幅图，增加5px的圆角

2.每幅图在鼠标悬停时，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：

outline:width style color;

widht:轮廓的宽度

style:轮廓的样式

取值：solid/dotted/dashed

color:轮廓的颜色

取消轮廓：

outline:none/0;

练习：在页面上创建一个文本框，取消文本框的轮廓，观察效果（文本框默认有轮廓）

3.框模型

1.什么是框模型

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：元素与元素之间的空白间距

内边距：元素边框与元素内容之间的间距

框模型的计算模式：

元素的实际占地宽度=左右外边距+左右边框+左右内边距+width;

元素的实际占地高度=上下外边距+上下边框+上下的内边距+height

课后作业：

1.参考学子商城登录页面效果

1.完成登录界面的布局（table/input）

2.点击文本框时取消轮廓效果

3.登录名输入框失去焦点时，验证登录名不能为空

CSS DAY03

1.外边距

1.什么是外边距

围绕在元素边框外的空白距离，就是外边距

也能表示当前元素与其他元素之间的空白距离

2.语法

属性：

margin 定义某个元素四个方向的外边距

margin-top/bottom/left/right 定义单边的外边距

取值：

1.具体数值，以px为单位

2.取值为负

让元素向相反的方向移动

margin-left:

取值为正，让元素向右移动

取值为负，让元素向左移动

margin-top:

取值为正，让元素向下移动

取值为负，让元素向上移动

3.取值为%

外边距的值，是父元素的宽或高的占比（50%）

4.取值为auto

自动计算外边距的值（控制块级元素水平居中对齐）

练习：

页面中新建两div元素，设置第一个div的左外边距为40px,第二个div在页面中水平居中显示

简写方式：

1.margin:value

四个方向的外边距

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

3.margin:v1 v2 v3 v4;

上 右 下 左 （顺时针方向）

3.自带外边距的元素

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

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,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，他们将合并为一个，最终取决于两个外边距中距离较大的那个。

如果两个外边距相遇时值相等，那么取其中一个值。

2.外边距的溢出

在某些条件下，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.为子元素设置上外边距时

练习：

1.创建两个div，id为d1和d2,分别设置不同背景颜色，尺寸200\*200；

2.在d2中嵌套一个子元素div,id="d3",设置其尺寸为100\*100，并设置其背景颜色

3.设置d3的上外边距为50px,观察其结果。

\*解决溢出方案：

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)

\*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

2.内边距

1.什么是内边距

元素边框与内容之间的空白距离

内边距会扩大元素边框占地区域

2.语法

属性：

padding 四个方向的内边距

padding-top/bottom/left/right 设置单边内边距

取值：

以px为单位的数值

以%形式设置

简写方式：

1.padding:value; 四个方向的内边距

2.padding:v1 v2;

v1:上下内边距

v2:左右内边距

3.padding:v1 v2 v3;

v1:上

v2:左右

v3:下

4.padding:v1 v2 v3 v4;

上 右 下 左 （顺时针）

练习：

1.创建两个div元素，id分别为d1和d2,尺寸200\*200，并设置边框为2px solid #f00;随意编写内容

2.为d1元素增加上下左右四个方向的内边距各20px，观察效果

3.属性：box-sizing

作用：指定框模型的计算方式

取值：

1.content-box

默认值，采用默认的计算元素的占地区域

实际占地宽度=左右边框+左右外边距+左右内边距+width;

实际占地高度=上下边框+上下外边距+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)

练习：完成如果所示的效果

3.背景属性

背景：可以是单一颜色或图片填充元素

1.背景色

属性：background-color

取值：合法的颜色值

注意：背景颜色默认是从边框的位置处开始填充的

2.背景图片

属性：background-image

取值：url(图片的路径);

ex:background-image:url(a.jpg);

3.背景图片平铺

属性：background-repeat

取值：

repeat 默认值，横向纵向都平铺

no-repeat 不平铺（图片只显示一次）

repeat-x 只在水平方向平铺

repeat-y 只在垂直方向平铺

4.背景图片尺寸

属性：background-size

取值：

1.width/height (ex:200px 300px)

2.width%/height% (ex:50% 50% 是元素自己的高宽占比)

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

练习：

给网页的body设置背景图（小图），给body设置高度为2000px,把图片固定在网页中，拖动滚动条，观察页面效果。

6.背景图片定位

作用：改变背景图在元素中的位置

属性：background-position

取值：

1.x y 具体的数值（px）

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

50% 50% 背景图在中间位置

3.关键字

x:left/center/right

y:top/center/bottom

ex:background-position:right top;

练习：

只显示图片的中花朵

7.背景简写属性

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

ex：

background:gray url(a.jpg);

注意：

如果不设置其中某个属性值的话，该位置采用默认值。

课后作业：

(所需要的素材在css-Day02-图片素材.rar)

完成css-Day03文件夹中"课后作业.png"的效果

CSS DAY04

1.渐变

1.什么是渐变

渐变指定是多种颜色平缓变换的一种显示效果。

2.渐变的主要因素

1.色标：一种颜色及其出现的位置

2.一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效果

2.径向渐变

以圆形的方式来实现填充

3.重复渐变

将线性渐变或径向渐变 重复几次实现填充

4.渐变详解

1.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

to top 从下向上填充渐变色

to bottom 从上向下填充渐变色

to left 从右向左填充渐变色

to right 从左向右填充渐变色

2.角度值

0deg 从下向上填充，等同于to top

90deg 从左向右填充，等同于to right

180deg 从上到下填充，等同于to bottom

270deg 从右向左填充，等同于to left

2.color-point

色标：颜色 及其 位置

取值：颜色 以及 位置的组合，中间用空格分开

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

2.径向渐变

属性：

background-image:radial-gradient([size at position],color-point1,color-point2,...);

size at position:

size:半径，以px为单位的数值

position:圆心所在位置

1.x y 具体数值

2.x% y% 元素宽和高的占比

3.关键字

x:left,center,right

y:top,center,bottom

ex：

100px at right top

半径 右上角位置

3.重复渐变

1.重复线性渐变

background-image:repeating-linear-gradient(angle,color-point1,color-point2,...);

color-point:位置一定要给绝对数值（px）,不要用相对单位%

2.重复径向渐变

background-image:repeating-radial-gradient([size at position],color-point1,color-point2,...);

练习：

设置两个div元素高宽各300\*300，第一个div用重复线性渐变，第二个div用重复径向渐变填充

5.浏览器兼容性

各个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，让浏览器支持渐变属性

Firefox:-moz-

Chrome & Sagari:-webkit-

Opera:-o-

IE:-ms-

2.文本格式化属性

1.字体属性

1.指定字体

属性：font-family

取值：字体名称，名称之间用逗号隔开

ex：

font-family:"微软雅黑",Arial,"黑体";

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（b,hn）

2.normal 正常

3.value 无单位的数字（整百倍）

400-900

400：等同于normal

900：等同于bold

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

将小写字符变成大写，但文本的大小与小写字符一致

属性：font-variant

取值：

1.normal 正常

2.small-caps 小型的大写字符

6.字体属性简写

属性：font

取值：style variant weight size family;

注意：

如果用简写方式，必须设置family的值，否则无效。

font:12px; 错误

font:12px "黑体"; 正确

练习：

创建一个div，并设置id为d1,内容随意,设置文本样式如下：

1.字体：微软雅黑

2.文字大小为24px

3.加粗显示文本并且为斜体

4.将所有的小写字符变为小型大写字符

2.文本格式

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（线条）

属性：text-decoration

取值：

none:无任何线条修饰

underline:下划线修饰

overline:上划线修饰

line-through:删除线修饰

4.行高

作用:定义一行文本的高度

特点：如果行高的高度大于字体本身的大小，那么该行文本将在指定的行高内呈现垂直居中的效果。

属性:line-height

取值：以px为单位数值

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：

在文本属性的练习基础上，增加以下样式：

1.文本的排列方式为两端对齐

2.当鼠标悬停时，文字出现下划线

3.要求首行文本缩进32px

4.增加文本阴影：水平，垂直偏移为0，模糊距离为5px,颜色为蓝色

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：指定单元格数据垂直对齐方式

取值：

top:上对齐

middle:居中对齐

bottom:下对齐

练习：创建网页，并在网页中添加表格

1.表格尺寸为400\*400，4行4列;

2.每个单元格的尺寸为100\*100,内容随意;

3.设置表格和单位元格的边框为1px solid #000;

4.设置每个单元格的左内边距为20px;

5.尝试为每个单元格增加上外边距15px。

2.表格的特殊属性

1.边框合并

取值：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.给两个值：

第一个值 表示水平间距

第二个值 表示垂直间距

注意：只有在边框分离模式下，边框边距才有效果，即border-collapse:separate时，border-spacing才有效。

3.表格标题位置

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

练习：

在前面练习基础上，给表格添加一个标题“表-1”，显示在表格下方。

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，即自动布局表格，列的尺寸实际上由内容来决定的。

2.fixed

固定表格布局，列的尺寸由设置的值为准

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.任何情况下都会加载显示表格（优点）

3.适用于确定每列大小时使用

4.固定表格布局不够灵活（缺点）

4.定位-浮动定位

1.定位

定位：指的是改变元素在页面中的默认位置

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（默认的定位方式）

2.浮动定位

3.相对定位

4.绝对定位

5.固定定位

3.定位详解

1.普通流定位

又称为文档流定位，页面中元素的默认定位方式

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显示

3.页面中的块级元素都是从上往下排列，每个元素独占一行

4.页面中的行内元素以及行内块都是按照从左到右的顺序来排列的

要解决的问题：让多个块级元素在一行中显示

2.浮动定位

1.什么是浮动&特点

1.元素一旦浮动起来，将不占据页面空间（脱离了文档流），其它未浮动元素将上前补位。

2.浮动元素会停靠在父元素的左边或右边，或其它已经浮动的元素的边缘上。

3.浮动定位解决的问题：能够让多个块级元素在一行中显示。

2.语法

属性：float

取值：

1.left

左浮动，让元素停靠在父元素的左边或挨着左侧已经浮动的元素

2.right

右浮动，让元素停靠在父元素的右边或挨着右侧已经浮动的元素

3.none

默认值，即无任何浮动效果

练习：

1.创建三个div元素，id分别为d1,d2,d3,尺寸都为200\*200，分别设置背景颜色

2.依次让d1,d2,d3实现右浮动，观察效果

3.将步骤2注销，依次让d1,d2,d3,实现左浮动，观察效果

课后作业：完成css-Day04文件夹中“课后作业.png"头部效果

CSS DAY05

1.浮动

1.浮动引发的特殊效果

1.元素一旦浮动起来以后就会变成块级元素

允许修改尺寸

能正常处理垂直方向外边距

2.当父元素显示不下所有已浮动元素时，最后一个将换行，但是，有可能被卡住；

3.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）；

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

2.清除浮动带来的影响

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

属性：clear

取值：

1.left

清除当前元素前面的元素左浮动所带来的影响

2.right

清除当前元素前面的元素右浮动所带来的影响

3.both

清除当前元素前面元素任何一种浮动所带来的影响

4.none

默认值，不做任何的清除浮动操作

3.浮动元素对父元素高度的影响

1.元素的高度都是以未浮动元素的高度为准的，浮动元素是不占页面的高度的

解决父元素的高度方案如下：

1.直接设置父元素的高度

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(块级)，并设置其clear:both;

2.显示

1.显示方式

1.什么是显示方式

决定了元素在网页中的表现形式（块级，行内，行内块）

2.语法

属性：display

取值：

1.none 不显示元素-隐藏

特点：脱离文档流，不占据页面空间

2.block

让元素表现的和块级元素一致

特点：

独占一行，可是修改高宽

3.inline

让元素表现的和行内元素一致

特点：

不允许修改尺寸

多个元素在一行中显示

无法设置垂直外边距

4.inline-block

让元素表现的和行内块元素一致

特点：

多个元素在一行中显示，但是可以修改尺寸

5.table

让元素表现的与表格一致

特点：

尺寸以内容为准

每个元素独占一行

允许修改尺寸

练习：

1.网页中创建5个a元素

2.为每个a元素设置尺寸为100\*100

3.分别为a元素设置不同的显示方式（none,inline,block,inline-block,table）

2.显示效果

1.显示/隐藏

属性：visibility

取值：

1.visible:默认值，元素可见

2.hidden:元素不可见-隐藏

面试：display:none和visibility:hidden的区别

display:none 不占页面空间

visibility:hidden 占页面空间

2.透明度

属性：opacity

取值：0.0（完全透明）~1.0（完全不透明）之间的小数

3.垂直方向对齐方式

属性：vertical-align

场合：

1.表格中使用

取值：top/bottom/middle

2.图片（img）中使用

取值：

top：上

bottom：下

middle：中间

baseline:基线对齐，默认值

3.光标

1.作用

改变鼠标悬停在元素上时，鼠标的状态

2.语法

属性：cursor

取值：

1.default: 默认

2.pointer: 小手

3.crosshair: +

4.text: I

5.wait: 等待

6.help: 帮助

练习：

新建一个div元素，当鼠标移入到div上时，让光标变成小手状态。

4.列表

1.列表项标记

属性：list-style-type

取值：

1.none

2.disc

3.circle

4.square

2.列表项图像

作用：使用自定义图像作为列表项标识

属性：list-style-image

取值：url(图像路径);

3.列表项位置

作用：将默认的列表项标识的位置，放到li里面

属性：list-style-position

取值：

1.outside 默认值，将标识显示在li外面的

2.inside 将标识放于li里面

4.列表属性简写

属性：list-style

取值：type url() position;

常用方式：list-style:none;

列表的使用场合：

横向排列或纵向排列的内容，都可以使用列表来实现。

5.定位

相对定位，绝对定位，固定定位

1.定位相关属性

属性：position

取值：

1.static:静态，默认值

2.relative:相对定位

3.absolute:绝对定位

4.fixed:固定定位

2.偏移属性

top/bottom/left/right

以上四个属性的取值均为数字

ex:

top:150px 元素向下移动150px

left:20px 元素向右移动20px

right:-150px 元素向右移动150px

注意：

只有已定位元素才能使用偏移属性

3.定位详解

1.相对定位

1.什么是相对定位

元素相对于它原来的位置偏移某个距离

2.使用场合

做元素位置微调时使用

3.语法

position:relative;

配合偏移属性来实现位置的移动

left:10px;

top:10px;

2.绝对定位

1.什么是绝对定位&特点

1.绝对定位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他最近的，已定位的，祖先元素 去实现位置的初始化。

3.如果没有已定位的祖先元素，那么该元素就相对于body去实现位置的初始化

4.配合偏移属性 实现元素位置的修改

2.语法

position:absolute;

配合偏移属性实现位置的修改

3.使用场合

1.有堆叠效果的元素

2.弹出菜单

4.注意：

1.脱离文档流-不占据页面空间

2.绝对定位元素会变成块级元素

5.堆叠顺序

一旦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.只有已定位元素才能实现堆叠顺序的改变（z-index）

2.父子元素间，z-index无效，永远都是子元素压在父元素上方

3.固定定位

1.什么是固定定位

将元素固定在网页的某个位置处，位置不会随着滚动条而发生改变，固定在可视区域中。

2.语法

position:fixed;

配合偏移属性使用

注意：

1.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.固定定位的元素会变成块级元素

3.固定定位元素会脱离文档流-不占页面空间

练习：

设置当前网页的高度为2000px，在网页中增加一个div元素，设置其尺寸为50\*200，背景颜色为蓝色，并将该div固定在网页的右下方（right:50px,bottom:80px;）

练习：完成学子商城首页banner图效果（css-Day05文件夹中"作业1.png"）

课后作业：

完成学子商城头部效果（css-Day05文件夹中"作业2.png"）

CSS DAY06

CCS3 Core

1.复杂选择器

1.兄弟选择器

兄弟元素：具备相同父元素的平级元素之间称为兄弟元素。

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{}

注意：兄弟选择器，只能向后找，不能向前找

2.通用兄弟选择器

作用：获取某元素后所有满足条件的兄弟元素

语法：选择器1~选择器2{}

练习：

<h3>静夜思</h3>

<div>窗前明月光</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：

1.h3与div之间有30px的外边距

2.从第二div开始，每个div都有10px的上外边距

2.属性选择器

<input type="text" name="uname" value="" placeholder="">

1.作用

允许通过元素所附带的属性及其值来匹配元素

type="text"

2.语法

1.基本语法

[attr] //属性

作用：匹配页面中所有附带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：匹配页面中所有附带attr属性的elem元素

ex:

div[id] 匹配页面中所有带有id属性的div元素

input[type] 匹配页面中所有带有type属性的input元素

.text[placeholder] 匹配页面中所有带有placeholder属性，并有text类选择器的元素

3.[attr1][attr2]

作用：匹配同时附带多个属性的元素

ex:

div[id][class]

4.[attr=value]

作用：匹配attr属性值为value的元素

ex:

<input type="text">

<input type="password">

//匹配的是第一个input元素(以下三种写法均可)

input[type="text"]

input[type=text]

input[type='text']

3.伪类选择器

1.已学过的伪类

链接伪类：

:link(尚未访问) :visited(访问过的)

动态伪类：

:hover(鼠标悬停)

:active(激活状态)

:focus(获取焦点)

2.目标伪类

作用：突出显示活动的HTML锚点元素，匹配被激活的锚点。

语法：

选择器:target{...}

div:target{...}

#d1:target{...}

练习：

完成如果所示效果

3.结构伪类

作用：通过元素的结构关系来匹配元素（上下级嵌套结构）

1.:first-child

匹配的元素属于其父元素的第一个子元素

<div>

<p>p1</p>

<p>p2</p>

<p>p3</p>

</div>

ex：

p:first-child{...} //找父元素的第一个子p元素(p1)

2.:last-child

匹配的元素是属于其父元素的最后一个子元素

ex:

p:last-child{...}//找父元素的最后一个子元素（p3）

练习：

创建一个4\*4的表格，尺寸400\*100，并填充内容

1.设置第一行数据的背景颜色为黄色

2.设置最后一行数据的背景颜色为红色

3.:nth-child(n)

匹配的元素是属于其父元素中第n个子元素

table tr:nth-child(2) 匹配表格第二行数据

练习：

把第二行的第三列背景颜色改为blue

4.:empty

匹配没有子元素的元素

<div></div>

5.:only-child

匹配的元素是属于其父元素中的唯一子元素

<div>

<p id="p1">ppppp1</p>

</div>

<div>

<p id="p2">ppppppppp2</p>

<span>ssssssss</span>

</div>

p:only-child{color:red;}//改变的是p1的颜色

4.否定伪类

将满足指定选择器的元素给排除出去（不要了，排除）

语法：

:not(selector)

ex:

table tr:not(:first-child){

font-size:24px;

}

除第一行以外所有行的文字大小变为24px

练习：

设置每行中第一个列以外的所有列文字以加粗和斜体的方式显示。

4.伪元素选择器

1.:first-letter或::first-letter

作用：匹配某个元素的首字符

2.:first-line 或::first-line

作用：匹配某个元素的首行（第一行）

注意：首行与首字符冲突时，以首字符为准

3.::selection

匹配被用户选取的内容

注意：只能修改文本颜色或背景颜色

练习：

1.创建一个div元素，id为d1,内容随意

2.将d1中首字符设置为加粗，红色，大小为20pt

3.将d1中的首行文本设置为蓝色，斜体

4.将用户选取的内容设置为红色背景，黄色文字

2.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段内容

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)鹅鹅鹅</p>

2.:after或::after

作用：匹配到某元素的内容区域之后

<p>锄禾日当午(:after)</p>

3.属性

属性：content

作用：向匹配到的位置处增加内容

取值：

1.字符串：用""引起来（普通文本）

2.url()：生成一副图像

练习：

<p>学而时习之，不亦说乎</p>

<p>有朋自远方来，不亦乐乎</p>

输出：

子曰：学而时习之，不亦说乎

-摘自《论语》

子曰：有朋自远方来，不亦乐乎

-摘自《论语》

4.解决问题

1.解决浮动问题

div:after{

content:"";

display:block;

clear:both;

}

2.子元素上外边距溢出问题

div:before{

content:"";

display:table;

}

课后作业：

创建一个4\*4的表格，尺寸400\*200，并填充内容

1.设置第二行数据的背景颜色为黄色

2.设置第三行数据的背景颜色为红色

3.设置第四行第四列背景颜色为蓝色

4.设置每行除最后一列的所有文字大小变为24px并加粗

CSS PROJECT2 DAY01

1.弹性布局（Flexible Layout）

1.什么是弹性布局

弹性布局，是一种布局方式，注意解决某元素中“子元素”的布局方式，为布局提供最大的灵活性。

2.弹性布局的相关概念

1.弹性布局的容器

弹性布局的容器 简称为"容器"（子元素的父元素，称为容器）

2.弹性布局的项目

弹性布局的项目 简称为"项目"(要实现布局效果的子元素，称"项目")

3.主轴

项目们排列方向的一根轴，就称为主轴

如果项目们按x轴排列（横向排列），那么x轴就是主轴

如果项目们按y轴排列（纵向排列），那么y轴就是主轴

4.交叉轴

与主轴交叉的一根轴就是交叉轴

如果主轴是x轴，那么y轴就是交叉轴

如果主轴是y轴，那么x轴就是交叉轴

3.语法

1.flex容器

将元素变为flex容器后，那么所有的子元素将变为flex项目，都允许按照弹性布局的方式排列

如何将元素变为flex容器？

属性：display

取值：

1.flex:将快级元素变为容器

2.inline-flex:将行内元素变为容器

注意：

1.元素设置为flex容器之后，子元素的float,vertical-align,clear将失效

2.容器的text-align将失效

2.容器的属性

1.flex-direction

作用：指定容器的主轴及其排列方向

取值：

1.row 默认值，即主轴为x轴，起点在左端

2.row-reverse 主轴为x轴，起点在有端

3.column 主轴为y轴，起点在顶端

4.column-reverse 主轴为y轴,起点在底端

2.flex-wrap

作用：当一个主轴排列不下所有项目时，指定子项目如何换行

取值：

1.nowrap 默认值，即空间不够时，也不换行，项目会自动缩小

2.wrap 换行

3.wrap-reverse 换行反转

3.flex-flow

作用：是flex-direction和flex-wrap的缩写形式

取值：

1.row nowrap 默认值

2.direction wrap

4.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.space-between

两端对齐（常用）

4.center

在主轴上居中对齐

5.space-around

每个项目两端间距相同

5.align-items

作用：定义项目在交叉轴上的对齐方式

取值：

1.flex-start

交叉轴的起点对齐

2.flex-end

交叉轴的终点对齐

3.center

交叉轴居中对齐

4.baseline

交叉轴上基线对齐

5.stretch

如果项目未设置尺寸，在交叉轴上将占满所有空间

3.项目的属性

该组属性只能设置在某一项目元素上，只控制一个项目，是不影响容器以其其它项目的效果。

1.order

作用：定义项目的排列顺序，值越小，越靠近起点，默认值为0

取值：整数数字，无单位

2.flex-grow

作用：定义项目的放大比例，如果容器由足够的剩余空间，项目将放大

取值：

整数数字，无单位

默认值0，不放大

取值越大，占据的剩余空间就越多

3.flex-shrink

作用：定义项目的缩小比例，即容器空间不足时，项目该如何缩小

取值：

默认值为1，空间不足时，则等比缩小

取值为0，则不缩小（常用）

4.align-self

作用：定义当前项目在交叉轴上的对齐方式

取值：

1.flex-start

2.flex-end

3.center

4.baseline

5.stretch

6.auto (继承自父元素的align-items属性)

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轴和y轴上发生的转换效果

3D转换:增加在z轴的转换效果

2.转换属性

1.转换属性

属性：transform

取值：

1.none 默认值，无任何转换效果

2.transform-function

表示1个或多个转换函数

如果是多个转换函数的话，中间用空格隔开

2.转换原点

属性：transform-origin

取值：数字/百分比/关键字

2个值：表示原点在x轴和y轴上的位置

3个值：表示原点在x轴,y轴,z轴上的位置

默认的原点在 元素的中心位置处

(center center)

(50% 50%)

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轴上的位移距离

取值为正，元素向右移

取值为负，元素向左移

2.translate(x,y)

指定元素在x轴和y轴的位移距离

x:同上

y:取值为正，元素向下移

取值为负，元素向上移

3.translateX(x)

在x轴上的位移

4.translateY(y)

在y轴上的位移

2.缩放

1.什么是缩放

改变元素在页面中的尺寸

2.语法

属性：transform

取值：

1.scale(value)

value:横向或纵向的缩放比例

value:默认值为1

>1: 放大

<1: 缩小

负数：放大（水平和垂直都翻转180度）

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)

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

注意：

1.转换原点会影响最后的转换效果

2.旋转是连同坐标轴一起旋转的，会影响旋转后的位移效果

练习：

1.将转换原点更改为元素的左上角

2.再将元素旋转45度，观察效果

3.将d1元素先旋转45deg,再向右移动50px,观察效果

4.注释步骤3，再将d1先位移50px，再旋转45deg,观察效果

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，实际上改变的是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，实际上改变的x轴的倾斜角度值

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(xdeg)

4.skew(x,y)

练习：

1.创建200\*200的div，设置背景颜色

2.鼠标悬停时，该元素向右移动200px,向下移动200px,向x轴倾斜45度，同时放大1.5倍

4.3D转换

1.透视距离

模拟人的眼睛到3D转换元素之间的距离（z轴）

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)

以x轴为中心轴，旋转元素的角度

取值为正，顺时针旋转

2.rotateY(ydeg)

以y轴为中心，旋转元素的角度

3.rotateZ(zdeg)

以z轴为中心，旋转元素的角度

4.rotate3D(x,y,z,ndeg)

x,y,z取值大于0的数字时，表示该轴要参与旋转，取值为0则不参与旋转

rotate3D(1,0,0,45deg)只在x轴旋转45度

roatate3d(1,1,1,0deg)

3.过渡

1.什么是过渡

使得css属性值，在一段时间内平缓变化的效果

2.过渡语法

1.指定过渡属性

属性：transition-property

取值：

1.all 能使用过渡的属性，一律用过渡体现

2.具体属性名

ex:

transition-property:background

当背景的属性在发生变化时用过渡给体现出来

transition-property:border-radius;

当边框倒角在发生改变时用过渡体现出来

transition-property:all

允许设置过渡效果的属性：

1.颜色属性（背景,文字,边框颜色,阴影颜色）

2.取值为数字的属性（高宽，内外边距等）

3.转换属性（位移，旋转，缩放，倾斜）

4.阴影属性

5.渐变属性

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.指定过渡速度时间曲线函数

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

5.过渡属性的编写位置

1.将过渡放在元素声明的样式中

既管去，又管回

2.将过渡放在触发的操作中（hover）

只管去，不管回

6.过渡的简写形式

transition:property duration timing-function delay;

ex:

transition:background 2s ease,border-radius 2s;

transition:all 2s; //上面的简写

练习:

1.在页面中创建一个100\*100的div元素

2.鼠标悬停时，5秒中之内，向右移动500px的同时再旋转3圈

3.鼠标移出时，还能再滚回来

4.动画

1.什么是动画

使元素从一种样式逐渐变为另一种样式

即将多个过渡效果放下一起

动画是通过"关键帧"，来控制动画的每一步

关键帧：

1.定义动画执行的时间点

2.在该时间点上的样式是什么

2.动画的使用步骤

1.声明动画

为动画起名

定义关键帧

2.为元素调用动画

指定元素调用的动画名称以及各个参数属性

3.定义动画（声明动画）

@keyframes 动画名称{

/\*定义该动画中所有的关键帧\*/

0%{

/\*动画开始时元素时的样式\*/

}

25%{

/\*动画执行到1/4时的样式\*/

}

50%{

/\*动画执行到一半时的样式\*/

}

75%{

/\*动画执行到3/4时的样式\*/

}

100%{

/\*动画执行结束时的样式\*/

}

}

4.动画调用（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曲线函数

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

4.animation-dealy

作用：指定动画的延迟时间

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.具体数字

2.infinite 无限次数播放

6.animation-diretion

作用：指定动画的播放方向

取值：

1.normal

默认值，正常播放0%-100%

2.reverse

逆向播放 100%-0%

3.alternate

轮播播放

奇数次数播放时，正向播放

偶数次数播放时，逆向播放

7.动画的简写方式

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 默认值

2.forwards

动画播放完成后，将保持在最后一个帧状态

3.backwards

动画播放前，延迟时间内，动画保持在第一个帧的状态上

4.both

forwards与backwards的合体

9.animation-paly-state

作用：指定动画处于播放还是暂停的状态

取值：

1.paused 暂停

2.running 播放

课后作业：

完成6-project2-Day01\Day01文件夹中"课后作业.png"的效果

PROJECT2 DAY02

1.CSS Hack

1.解决问题

解决IE浏览器兼容性问题

解决办法：针对不同浏览器编写不同CSS代码

2.CSSHack 的原理

使用CSS样式的优先级解决兼容性问题

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，以便于识别不同浏览器

+:IE6,7的前缀

-:IE6的前缀

\0:IE8,9,10的后缀

\9\0:IE9,10的后缀

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

\*:识别IE6

\*+：识别IE7

div~div{}

\*div~div{IE6识别}

\*+div~div{IE7识别}

3.HTML头部引用Hack

使用IE条件注释来判断浏览器的版本，从而执行不同的代码

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代码

ex:

<!--[if gt IE 6]>

该段内容只显示在IE6以上版本的浏览器中

<![endif]-->

2.gte

大于等于

3.lte

小于等于

4.lt

小于

5.！

在指定条件以外版本的浏览器中执行操作

6.省略条件

只在指定版本的浏览器中执行操作

判断是否为IE浏览器或IE的指定版本

2.项目

1.首页

2.在首页中引入头部（header.html）

3.完成banner广告图

4.完成楼层1F（2F,3F与1F一样布局）

5.完成footer.html的布局

6.课后作业：（参照codeboy.com）

1.完成footer.html中底部版权显示

2.在首页中引入footer.html调整整体效果

3.完成楼层中其它小图的布局效果

JavaScript 进阶 DAY01

准备:

1. 犀牛书: JavaScript权威指南

2. 公众号: 前端大全!

3. 上届笔记 和 思维导图

谁抄的越勤，谁学的越烂！！！

正课:

String:

1. 什么是String:

2. API:

3. \*\*\*正则表达式:

1. 什么是String: 一串字符组成的字符数组

vs 数组:

相同: 1. 下标, 2. .length, 3. for, 4. slice

不同: 类型不同, API不通用!

字符串必须用' '——ESLint

2. API: 所有字符串API无权修改原字符串，只能返回新字符串

大小写转换:

str.toLowerCase() //转为小写

str.toUpperCase() //转为大写

何时: 不区分大小写时

获取指定位置的字符:

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

获取指定位置字符的unicode号:

str.charCodeAt(i) 字->号

将unicode号转回字: String.fromCharCode(号)

选取子字符串:

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

差别: substring不支持负数参数

变通: str.length-n

str.substr(starti,n) 从starti开始选取n个字符

\*\*\*查找关键词: 4种:

1. 查找一个固定的关键词的位置:

var i=str.indexOf("关键词",fromi)

在str中，从fromi位置开始，找下一个"关键词"出现的位置

返回值: 关键词第一个字的位置

找不到，返回-1

var i=str.lastIndexOf("关键词")

查找str中"关键词"最后出现的位置

优: 设置开始位置，找所有关键词的位置

缺: 只能查找一个固定的关键词

2. 判断是否包含符合正则条件的敏感词:

var i=str.search(/正则/i)

查找str中查找第一个符合正则要求的关键词的位置

返回值: 关键词位置

找不到返回-1

问题: 正则默认都区分大小写

解决: 在第二个/后加i ignore

问题: 只能找一个，且只返回位置，无法返回关键词内容

3. 获取所有关键词的内容:

var kwords=str.match(/正则/ig)

查找str中所有关键词的内容

返回值: 关键词数组

如果找不到, 返回null

强调: 如果一个函数可能返回null，就必须先验证不是null，再使用!

问题: 所有正则，默认只查找第一个关键词

解决: 在第二个斜线后加g global

问题: 只能找关键词的内容，无法返回每个关键词的位置

4. 即找所有关键词的内容，又找所有关键词的位置

reg.exec()

替换:

1. 简单替换: 将所有关键词，替换为统一的新内容

str=str.replace(/正则/ig,"替换值")

更简单: 如果关键词是固定的: str.replace("关键词","新值")

2. 高级替换: 根据每个关键词的不同，动态选择不同的新值替换

str=str.replace(

/正则/ig,

function(kword){//每找到一个关键词，就自动调用该function

//kword会自动获得本次找到的一个关键词

return 根据kword的不同，返回不同的新值

}

)

切割:

3. \*\*\*正则表达式:

什么是: 描述一个字符串中字符出现规律的规则!

为什么: 有些关键词会发生变化！

何时: 2种:

1. 查找可变的关键词

2. 验证格式

如何: 正则表达式的语法:

1. 一个固定的关键词原文，也是一种最简单的规则

2. 字符集: 规定一位字符上可用的备选字符列表

何时: 只要一位字符有多种备选字

如何: [备选字列表]

简写: 如果备选字符列表连续，可用-省略中间字符

比如: 一位数字: [0-9]

一位小写字母:[a-z]

一位大写字母:[A-Z]

一位字母: [A-Za-z]

一位字母或数字:[A-Za-z0-9]

一位汉字: [\u4e00-\u9fa5]

排除: [^xxx]

3. 预定义字符集:

对常用字符集的简化:

一位数字 \d

一位字母,数字,\_都行 \w [0-9A-Za-z\_]

一位空字符 \s 包括: 空格, tab, ...

通配符 .

问题: 字符集只能规定一位字符的备选内容

无法定义字符出现的个数

解决:

4. 量词: 专门规定一个字符集出现次数的规则

何时: 只要规定一个字符集出现次数时

如何: 字符集量词

2大类:

1. 有明确数量边界:

字符集{6,8} 至少6个，最多8个

字符集{6,} 至少6个，多了不限

字符集{6} 必须6个，不能多也不能少

2. 没有明确数量边界:

字符集? 可有可无，最多一个

字符集\* 可有可无，多了不限

字符集+ 至少一个，多了不限

5. 选择和分组:

选择: 或 |

规则1|规则2 在两个规则之间选其一匹配即可

何时: 只要在多个规则间选其一匹配时

分组: 将多个规则，用()包裹为一个整体

为什么: 量词默认仅修饰相邻的前一个字符集

何时: 只要一个量词，要修饰多个字符集时

如何: (规则1规则2...)量词

身份证号: 15位数字 2位数字 一位数字或X或x

后三位整体可有可无，最多一个

\d{15} ( \d\d [0-9Xx] )?

手机号: +86或0086 可有可无，最多1次

空字符 可有可无，多了不限

1

[34578]

9位数字

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

微信 wei xin w x

(微|w(ei)?)\s\*(信|x(in)?)

6. 匹配特殊位置:

三个位置:

1. 开头 ^ 比如: 开头的空字符: ^\s+

2. 结尾 $ 比如: 结尾的空字符: \s+$

何时: 如果只匹配紧挨开头或结尾的内容时

即匹配开头，又匹配结尾: ^\s+|\s+$  
 3. 单词边界 \b 包括: 开头, 结尾, 空格, 标点

JavaScript 进阶 DAY02

个人项目:

1. 静态界面

2. 收集业务数据

正课:

1. String

2. RegExp

3. Math

4. Date

1. String:

替换:

衍生: 删除: 将关键词替换为""

切割: 将一个字符串，按指定的字符切割为多段子字符串

如何: 2种:

1. 简单切割: 按一个固定的分隔符，分割字符串

var substrs=str.split("分隔符")

固定套路: 打散字符串为字符数组:

var chars=str.split("")

2. 复杂切割: 如果分隔符是变化的，可用正则切割

var substrs=str.split(/正则/)

2. RegExp:

什么是: 专门保存一条正则表达式，并提供用正则表达式执行验证和查找操作的API 的对象

何时: 只要在js中使用正则表达式，都要先创建RegExp对象

主要两个场景:

1. 验证字符串格式

2. 即查找所有关键词的内容，又查找每个关键词的位置

如何:

创建: 2种:

1. 直接量:

var reg=/正则/ig;

何时: 如果正则表达式不需要动态生成!

2. 用new:

var reg=new RegExp("正则","ig")

何时: 专门用于动态生成正则表达式

API:

1. 验证字符串格式:

var bool=reg.test(str)

用reg检查str是否符合正则表达式格式要求

问题: 正则，默认只要找到符合条件的内容，即使只是一部分，也返回true。

解决: 今后，凡是验证，必须前加^，后加$

表示从头到尾，完整匹配!

2. 即找所有关键词内容，又找所有关键词位置

var arr=reg.exec(str)

在str中找下一个关键词的内容和位置

返回值: 数组:[

"0": 本次找到的关键词内容,

"index": 本次找到的关键的位置

]

如果找不到，返回null

3. Math:

什么是: 专门保存算术计算的常量，并提供算术计算的API

何时: 今后，只要进行数学计算时

如何:

创建: 不能new!

所有API通过Math直接.调用

API:

1. 取整: 3种:

上取整: 只要小数部分超过，就取下一个整数

Math.ceil(num)

下取整: 舍弃小数部分

Math.floor(num)

vs parseInt(str) 可去掉数字后非数字字符(单位)

四舍五入取整: 整数后一位小数，够5进1，不够舍弃

Math.round(num);

vs .toFixed(2)

1. Math.round只能取整!

.toFixed(d) 按任意小数位数四舍五入

2. 返回值: Math.round() 返回数字类型

.toFixed() 返回字符串类型

2. 乘方和开平方:

Math.pow(底数,幂)

Math.sqrt(n)

3. 最大值和最小值

Math.max(值1,值2,.....)

Math.min(值1,值2,.....)

问题: max和min默认不支持在数组中找最大和最小

解决: Math.max(...arr)

4. 三角函数:

Math.sin Math.cos Math.tan

反三角函数:

Math.asin Math.acos Math.atan

5. 随机数:

0<=Math.random()<1

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

简写: 当最小值为0时，

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

4. Date:

什么是: 保存一个时间，提供操作时间的API

何时: 只要在程序中保存或操作时间，都要使用Date对象

如何:

创建: 4种:

1. 创建日期对象，并获得当前系统时间

var now=new Date();

强调:无法获得服务器时间，仅能获得客户端时间

2. 创建日期对象，封装自定义时间

var date=new Date("yyyy-MM-dd hh:mm:ss")

3. 可用一个毫秒数转化为时间

原理: 起始日期对象中保存的也是一个毫秒数(1970年1月1日0点至今的毫秒数)

为什么: 时间点受所在时区的影响，有所不同

时间段不受时区影响。

每个时区，将毫秒数时间段转为时间点时，各自的时间起点不一样，导致计算之后的终点时间各有不同。

何时: 只要将毫秒数转为日期对象执行操作时，都用new Date()

如何: var date=new Date(ms)

4. 复制一个日期对象:

var date2=new Date(date1);

JavaScript 进阶 DAY03

正课：

1. Date:

2. Error:

3. Function:

1. Date:

API: 3句话:

1. 单位:

FullYear Month Date Day

Hours Minutes Seconds Milliseconds

2. 每个单位上都有一对儿get/set方法

其中: getXXX() 专门获取单位的数值

setxxx() 专门设置单位的数值

比如: date.getDate() date.getFullYear()

date.setDate(20) date.setFullYear(2018)

特例: Day没有set, 不能修改

3. 取值范围:

Month: 0~11 计算机中的month比现实中小1

在取值和修改时，要修正

Date: 1~31

Day: 0~6

Hours: 0~23

Minutes/Seconds: 0~59

计算: 2种:

1. 两个日期对象可相减: 结果为毫秒差

2. 对任意单位做加减:

3步:

1. 获得单位上的当前值: 2018/2/13 getDate() 13

2. 做加减: 13-30 = -17

3. 将计算结果set回去  
 .setDate(-17);

setDate内置日期进制算法。

简化: date.setxxx(.getxxx()+n)

格式化:

.toLocaleString() 转为当地时区的简化版格式

.toLocaleDateString() 转为当地时区的简化版格式，仅保留日期

2. Error:

什么是: 程序执行过程中，导致无法继续正常执行的异常情况

只要程序发生错误，就会中断退出

错误处理: 即使程序发生错误，也保证不会中断退出的机制

如何错误处理:

try{

可能发生错误的正确代码

}catch(err){

一旦发生错误，就自动执行的错误处理代码

}

其中: err是错误对象: 错误发生时，自动创建的封装错误信息的对象。

效率: 放在try catch中的代码，效率略有降低

解决: 用if...else代替try...catch

鄙视: js中错误类型: 6种:

SyntaxError 语法错误

ReferenceError 引用错误 要用的变量没找到

RangeError 范围错误 参数超范围

TypeError 类型错误 错误的使用了(),[]等。

URIError URI错误

EvalError Eval错误

抛出自定义错误:

何时: 协作开发中，程序的作者，向程序的调用者，提示错误的使用程序。

3. Function

什么是: 保存一段可重用的代码段的对象

何时: 只要一段代码可能反复使用时，都要封装为函数，反复调用函数

如何:

创建: 3种:

1. 直接量: function 函数名(参数列表){

函数体;

return 返回值;

}

缺: 会被声明提前, 打乱程序的执行顺序

解决:

2. 赋值: var 函数名=function(参数列表){

函数体;

return 返回值;

}

优: 不会被声明提前

揭示: 函数其实也是一个引用类型的对象

函数名其实只是一个普通的变量

函数名通过地址引用函数对象

function其实是创建一个新函数对象的意思

3. 用new:

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

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，可根据传入的参数不同，自动选择匹配的函数执行。

为什么: 减少API的数量，减轻调用者的负担

何时: 只要一项任务，可能根据传入参数的不同，执行不同的流程时。

如何:

问题:js语法默认不支持重载，因为不允许多个同名函数同时存在！

解决: arguments

什么是: 函数调用时，自动创建的

保存所有传入函数的参数值的

类数组对象(长的像数组的对象)

vs 数组: 相同: 1. 下标; 2. .length; 3. 遍历

不同: 类型不同, API不通用

何时: 重载:

匿名函数:

什么是: 定义函数时，不指定函数名的函数

何时: 一个函数只使用一次时

为什么: 节约内存

如何: 2种:

1. 回调函数: 定义函数后，自己不执行，交给别的函数，由别的函数控制调用和执行过程。

2. 匿名函数自调: 函数定义后，自己调用自己!

为什么: 全局变量，用完，无法自动释放！污染全局！

解决: 今后禁止使用全局变量！

何时: 只有所有功能的代码，必须放在匿名函数自调中

如何: (function(){

//函数体

})();

鄙视:垃圾回收

什么是: 引擎会自动释放不再使用的对象的内存空间

为什么: 内存空间大小是有限的！系统都会释放不再被使用的对象，为后续新对象释放更多空间。

垃圾回收器: 专门统计并回收内存中不再被使用的对象的空间的 小程序

如何:

1. 垃圾回收器伴随主程序启动而启动，在后台伴随主程序运行而运行。

2. 垃圾回收器会记录每个对象被几个变量引用着

3. 只要一个对象不再被任何变量引用，则垃圾回收器自动释放该对象

习惯: 只要一个大对象，不再被使用，应该主动将变量赋值为null

JavaScript 进阶 DAY04

正课:

1. Function:

作用域和作用域链

\*\*\*\*\*闭包

2. \*\*\*OOP

1.Function  
 作用域(scope):

什么是: 变量的可用范围

为什么: 保证内部的变量不会影响/干扰外部的程序

包括: 2种:

1. 全局作用域: window

保存全局变量: 特点: 随处可用，可重复使用!

2. 函数作用域:

保存局部变量: 特点: 仅函数内可用，不可重用!

函数的生命周期:

1. 程序开始执行前:

1. 创建执行环境栈(ECS): 用于记录有多少函数正在被调用

2. 在ECS中记录首先调用的浏览器主程序(main)

3. 创建全局作用域对象window，保存main函数执行所需的所有全局变量

2. 定义函数时:

1. 用函数名定义变量

2. 创建函数对象保存函数的定义，函数名变量通过地址引用函数对象

3. 函数对象的scope属性，指回函数来自的作用域

3. 调用函数时:

1. 在ECS中加入本次函数调用的记录

2. 为本次函数调用，创建函数作用域对象(活动对象AO)

3. 在活动对象AO中保存所有局部变量

4. 让AO的parent属性，指向函数的scope引用的作用域对象

调用过程中，变量的使用顺序:

先在AO中，使用局部变量

局部没有，才去全局找

4. 函数调用后:

1. ECS中本次函数调用的记录，出栈

2. 导致函数作用域对象AO释放

3. 导致AO中局部变量释放

作用域链(scope chain):

什么是: 由多级作用域对象逐级引用形成的链式结构

保存着所有变量

控制着变量的使用顺序: 先局部，后全局

\*\*\*\*\*闭包

什么是: 即重用变量，又保护变量不被污染的一种机制

为什么: 全局变量和局部变量都有不可兼顾的优缺点

全局: 优: 可反复使用, 缺: 随处可用，易被污染

局部: 优: 仅函数内可用，不会被污染

缺: 不可重用!

何时: 今后只要希望一个变量，即可重用，又不会被污染

如何: 3步:

1. 用外层函数包裹受保护的变量和内层函数对象

2. 外层函数将内层函数对象返回到外部

3. 使用者调用外层函数，获得返回的内层函数

鄙视: 画简图:

1. 找受保护的变量

2. 找操作变量的内层函数

缺: 闭包比普通函数占用更多的内存空间

解决: 如果闭包不再使用，应尽快释放!

将引用闭包结构的变量赋值为null

2. \*\*\*OOP:

什么是对象: 程序中描述现实中一个具体事物的属性和功能的程序结构.

什么是面向对象: 程序中都是先用对象来描述现实中的具体事物，再按需调用对象的方法，访问对象的属性值。

为什么: 当要管理的数据非常多时，如果单独管理每项数据，极其不便于维护和使用。且不符合现实。

如何: 面向对象三大特点: 封装，继承，多态

封装:

什么是: 创建一个对象集中保存现实中一个事物的属性和功能

何时: 今后，只要使用面向对象的思想编程，都要先创建一个对象，来存储现实中一个事物的属性和功能

为什么: 便于维护和使用，且更符合现实的意义

如何: 3种:

1. 直接量: var obj={

属性:值,

... : ...,

方法:function(){

... this.属性名 ...

}

}

现实中: 具体事物的属性，会成为对象的属性

具体事物的功能，会成为对象的方法

问题: 对象自己的方法不能直接访问自己的属性

因为: 所有不加.直接访问的变量，只能在作用域链中查找，无法自动进入对象中查找属性。

错误解决: 在属性前加对象名.

问题: 对象名只是普通的变量名，可能发生改变。

正确解决: 在属性前加this.

this: 函数执行时，自动创建的，引用正在调用该函数的.前的对象本身。

何时: 只要对象自己的方法，想使用对象自己的成员时，都必须加this.

如何访问对象的成员:

成员=属性+方法

访问属性: obj.属性名

调用方法: obj.方法名()

2. 用new: 2步:

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

2. obj.属性名=值

obj.方法名=function(){ ... this.属性名 ... }

本质: 起始js对象底层，就是一个关联数组

问题: 每次只能创建一个对象，如果反复创建多个对象，重复的结构代码会非常多！——DRY

解决:

3. 用构造函数反复创建多个相同结构的对象:

构造函数(constructor): 描述一类对象，统一结构的函数

何时: 只要反复创建多个相同结构的对象前，都要先定义构造函数。

如何: 2步:

1. 定义构造函数，描述一类对象的统一结构:

function 类型名(属性参数列表){

this.属性名=属性参数;

this.方法名=function(){

... this.属性名 ...

}

}

2. 调用构造函数反复创建同一类型的对象:

var obj=new 类型名(属性值列表);

new做了4件事:

1. 创建一个新的空对象:

2. 设置新对象的\_\_proto\_\_属性，继承构造函数的原型对象

3. 调用构造函数，并将其中this自动指向新对象

构造函数用强行赋值的方式，将构造函数中规定的属性和方法，添加到新对象中

4. 返回新对象地址

总结: 构造函数两个作用:

1. 描述一类对象的统一结构

2. 将一个新的空对象，装修成规定的格式和内容

优: 代码重用!

问题: 在构造函数中添加的方法，每调用一次构造函数，都会创建该方法的副本。浪费内存！

解决: 继承:

继承:

什么是: 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: 只要多个子对象都拥有相同的成员时，应该仅在父对象中保存一份即可。所有子对象共用!

如何: js中都是继承原型对象

原型对象(prototype): 专门保存同一类型的所有子对象，共有成员的父对象。

何时: 只要多个子对象都拥有相同的成员时，应该仅在原型对象中保存一份即可。

如何:

创建: 不用自己创建，而是买一赠一:

每定义一个构造函数，都附赠一个空的原型对象

继承: 不用自己设置

每new一个子对象，自动设置子对象继承构造函数的原型对象！

共有属性和自有属性:

共有属性: 保存在原型对象中，所有子对象共有的属性

自有属性: 保存在子对象中，仅归当前子对象独有的属性

获取: obj.属性名

修改: 自有属性: 子对象.属性名=值;

共有属性: 只能用 原型对象.共有属性=值

内置对象的原型对象:

内置对象: ES标准中规定的，浏览器已经定义好的对象。

ES标准内置对象: 11个:

String Number Boolean —— 包装类型

Array Date Math RegExp

Function Object

Error

Global(在浏览器中被window代替)

鄙视: 包装类型(box)：

什么是: 专门封装一个原始类型的值，并提供操作原始类型值的API

为什么: 原始类型的值，仅是一个值而已，没有任何的方法。

何时: 只要试图对原始类型的值调用方法时，都会自动创建对应包装类型的对象。

一种类型: 都包含2部分:

1. 构造函数: 创建当前类型的子对象

2. 原型对象: 保存当前类型所有子对象共有成员

可解决浏览器兼容性问题:

问题: 旧浏览器，无法使用新的API

解决:

JavaScript 进阶 DAY05

正课:

1. OOP

继承

多态

2. ES5

1. OOP

继承:

原型链: 由多级父元素，逐级继承，形成的链式结构

保存了: 对象的属性和方法

控制着: 对象的属性和方法的使用顺序和共享范围

顺序: 先用自有，自己没有，才用共有

共享范围: 越靠原型链上层的共有成员，共享范围越广

vs 作用域链:

保存了所有不加.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

鄙视: 判断一个对象是不是数组类型, 共有几种方法

错误: typeof只能区分原始类型和function

无法进一步区分引用类型对象的具体类型

1. 判断原型对象:

var bool=Array.prototype.isPrototypeOf(obj)

等效于: obj.\_\_proto\_\_==Array.prototype

2. 判断构造函数:

var bool=obj instanceof Array;

等效于: obj.constructor==Array

问题: 不仅检查直接父对象，且检查整个原型链

解决:

3. 检查class属性:

每个对象内，都有一个class属性。

class属性记录了对象创建时的类型名称

class属性不随继承关系改变而改变

问题1: class属性是内部属性, 不允许用.直接访问!

解决: 唯一可查看class属性的办法:

只有Object.prototype.toString() 才可输出:

[object Class]

问题2: 多数内置类型的子对象，无法直接调用Object.prototype中的原始toString()

因为各自类型的原型对象，都纷纷重写了toString()

解决: call 抢:

想调用的函数.call(对象)

在执行时，相当于: 对象.想调用的函数()

Object.prototype.toString.call(obj)==="[object Array]"

4. 直接调用Array.isArray(obj)

强调: 原理同第三种方式，也是严格的检查

鄙视: 一个方法，何时定义在原型对象中，何时定义在构造函数对象上?

实例方法:保存在原型对象中，只能被当前类型的子对象调用的方法。

何时: 如果一个方法，只希望当前类型的子对象才能调用时

如何调用: 必须先创建该类型的子对象，再用子对象.调用实例方法。

比如: arr.sort() arr.indexOf() arr.push()

静态方法: 保存在构造函数对象上，不需要创建子对象，就可用构造函数.直接调用的方法

何时: 如果希望任何类型的对象，都可使用的方法，就可定义为静态方法

如何调用: 不需要创建子对象，只要用构造函数.就可直接调用。

比如: String.fromCharCode(unicode)

Array.isArray(obj)

多态: 同一个方法，在不同情况下表现出不同的状态

包括:

重载overload

重写 override:

什么是: 子对象觉得从父对象继承来的成员不好用时，可在子对象本地定义同名自有成员，覆盖父对象成员。

何时: 只要子对象觉得从父对象继承来的成员不好用！

如何:

自定义继承:

1. 修改一个对象的父对象:

Object.setPrototypeOf(child,father)

等效于: child.\_\_proto\_\_=father

2. 修改多个子对象的父对象:

其实就是修改构造函数的原型对象

时机: 还未开始创建子对象之前

定义完构造函数后，立刻更换

3. 两种类型间的继承:

何时: 只要定义多个类型时，发现拥有部分相同的属性结构和方法时

如何: 2步:

1. 定义抽象父类型:

什么是抽象父类型: 集中存储多个类型间相同的属性结构和方法 的类型

如何定义:

1. 抽象父类型的构造函数中保存相同的属性

2. 抽象父类型的原型对象中保存相同的方法

2. 让子类型继承父类型: 2步:

1. 在子类型构造函数中，先调用父类型构造函数

问题: 直接调用父类型构造函数, 则父类型构造函数中的this默认指window

解决: call 替换函数内不想要的this为指定的对象

2. 设置子类型原型对象继承父类型原型对象

2. ES5:

严格模式:

什么是: 比普通js运行机制，更严格的模式

何时: 今后，所有代码的开发，都必须在严格模式下！

如何: 2种:

1. 在整个<script>或js文件范围内启用严格模式:

在顶部添加: "use strict";

何时: 只要是新项目的开发，都必须整个script或js文件启用严格模式!

2. 仅在一个函数内启用严格模式:

在函数内的顶部加: "use strict";

何时: 旧项目向严格模式迁移时

规定:

1. 禁止给未声明的变量赋值

2. 匿名函数自调和普通函数调用中的this不再默认指向window，而是undefined

都是为了避免内存泄漏

3. 禁用arguments.callee

什么是: 在函数调用时，在函数内部引用当前函数本身

何时: 递归

为什么: 为了在函数内自动获得当前函数名,避免写死

递归的问题: 重复计算量极大！效率极低!

所以: 禁用arguments.callee，等于禁止使用递归！

解决: 所有递归，都可用循环代替!

4. 将静默失败升级为错误!

JavaScript 进阶 DAY06

正课:

1. ES5

2. ES6

1. ES5

保护对象:

保护属性:

属性分为2大类:

命名属性: 可用.直接访问到的属性

——服务器端语言 公有public

保护命名属性: 分为2大类:

数据属性: 实际保存属性值的属性

如何保护: 每个数据属性都有四大特性:

{

value: 实际存储属性值,

writable: true/false, 控制是否可修改

enumerable: true/false, 控制是否可被for in遍历到

问题: 隐藏并不彻底, 依然可用.直接访问！

configurable: true/false,

控制着: 1. 是否可删除该属性

2. 能否修改其它特性

强调: 一旦改为false，不可逆！

}

如何获取四大特性:

var attrs=Object.getOwnPropertyDescriptor(obj,"属性名")

如何修改四大特性:

Object.defineProperty(obj,"属性名",{

特性名: 值,

... : ...

})

问题: 每次只能修改一个属性的四大特性

解决: Object.defineProperties(obj,{

属性名:{

特性名:值,

... : ...

},

属性名:{

... : ...

}

})

问题: 保护是固定的，有限的几种

访问器属性: 不实际存储属性值，仅对其它数据属性提供保护！

何时: 1. 用自定义规则保护属性时

2. 计算属性(vue)/虚拟属性(mongodb)

包含: 四大特性: {

get(){ return this.受保护的数据属性},

set(val){

//val自动收到要赋的新值

if(val符合规则)

才this.受保护的数据属性=val

else

报错！

},

enumerable:true/false,

configurable:true/false

}

其中: 访问器属性的用法和普通属性完全一样

但是: 试图读取属性值时，自动调用get方法

试图为属性赋值时，自动调用set方法

set方法中的val会自动接到要赋的新值

如何定义访问器属性: 访问器属性只能通过Object.defineProperty或defineProperties创建

内部属性: 不可用.访问的属性

——服务器端语言 私有 private

比如: .class .\_\_proto\_\_

防篡改: 禁止删除/添加属性

3个级别：

1. 防扩展: 禁止给对象添加新属性

Object.preventExtensions(obj)

原理: 每个对象中都有一个内部属性extensible=true

控制着是否可为对象添加新属性

preventExtensions()其实是将extensible=false

2. 密封: 在兼具防扩展基础上，又禁止删除现有属性

Object.seal(obj)

原理: 不但将extensible=false

且自动修改所有属性的configurable为false

3. 冻结: 在兼具密封基础上，又禁止修改一切属性值

Object.freeze(obj)

原理: 在密封的基础上，进一步修改所有属性的writable为false!

Object.create:

什么是: 创建一个子对象，继承指定父对象，并为子对象扩展自有属性。

何时: 只有父对象，也想创建子对象时

如何: var child=Object.create(father,{

//Object.defineProperties

bao:{

value:"LV",

writable:true,

enumerable:true,

configurable:false

}

})

Object.assign() 鄙视: 复制或合并对象

复制: var clone=Object.assign({},obj)

将obj中的所以属性和属性值合并到空对象中,并返回

问题: 浅克隆: 仅复制数据属性的值

如果属性是引用类型对象，则仅复制对象地址

解决: 深克隆: var clone=JSON.parse(JSON.stringify(obj))

合并: 将多个原对象，合并为一个新对象，原对象保持不变

var newObj=Object.assign({},obj1,obj2,...)

何时: 如果页面需要的数据，来自于多个接口方法返回的多个对象。都要将多个对象，先合并为一个对象，用一次响应返回！

call apply bind:

如果函数中的this不是想要的，都可用call,apply,bind替换函数中的this为想要的任何对象

call apply:

什么是: 在调用函数时，临时替换函数中的this

何时: 如果调用函数时，临时替换this

如何:

fun.call(obj,参数值列表)

调用fun，临时替换fun中的this为obj，并传递参数给fun

fun.apply(obj,[参数值数组])

调用fun，临时替换fun中的this为obj，并传递参数给fun

强调: 唯一差别: 参数的形式上:

call,要求参数单独传入

apply,要求所有参数值必须放在一个数组中整体传入

bind:

什么是: 基于原函数，创建一个新函数，并永久绑定其中的this为指定对象

何时: 专门用于替换回调函数中的不想要的this

数组API:

判断: 判断数组中的元素是否符合条件

.every(): 判断是否数组中每个元素都符合要求

.some(): 判断数组中是否包含符合要求的元素

仅以.every()举例:

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

//elem当前元素值, i 当前位置, arr 当前数组对象

return 判断条件

})

用判断条件去检查每个元素，除非所有元素都满足要求，才返回true。

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,直接修改原数组

何时: 对原数组中每个元素执行相同操作时

如何: arr.forEach(function(elem,i,arr){

arr[i]=新值

})

.map() 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，原数组保持不变

何时: 希望原数组保持不变，返回新数组时

如何: arr.map(function(elem,i,arr){

return 新值

})

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，原数组保持不变。

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

return 判断条件

})

汇总: 将数组中每个元素的值汇总统计出一个最终的结论

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

//prev: 截止当前位置之前的临时汇总值

return prev+elem;

})

JavaScript 进阶 DAY07

正课:

1. ES6

let

参数增强

箭头函数

模板字符串

解构

for of

class

\*\*\*\*\*Promise

封装ajax函数

集合

1. let: 代替var

为什么: js中广受诟病的缺点: 声明提前! 破坏程序的正常执行顺序

特点:

1. 防止声明提前

2. 不允许重复声明同名变量

3. 添加块级作用域:

什么是块级作用域(block): 一个{}程序结构内，也是一个作用域。

比如: for while do...while if...else if...else

问题: js中没有块级作用域，块内的变量会影响外部！

为什么: 将块内的变量限制在内部，不要影响外部

如何: 只要用let声明的变量，仅在当前块内有效

原理: 在块内添加匿名函数自调，来划分临时作用域

4. 和for一起使用，可形成闭包!

原理: (function(i){...})(i)

2. 参数增强:

default: 在定义函数时，为函数参数添加默认值

有默认值的参数，在调用函数时，不必赋值

强调: 有默认值的参数必须在参数列表的结尾！

如何: fun(参数1,......,参数n=默认值)

rest: 代替arguments

为什么: arguments的问题:

1. 不是数组类型，无法使用数组的API!

2. 只能获得全部参数

何时: 只要不确定参数个数时，都用rest代替arguments

如何: 在定义函数时，固定参数之后，的剩余参数位置: ...数组名

spread: 代替apply, 将数组打散为单个元素，分别传入

如何: fn(..., ...数组)

3. 箭头函数: 省略function

何时: 所有匿名函数和回调函数，都可用箭头函数简写

如何: 去function，改=>

其实可以更简化:

如果只有一个参数，可省略()

如果函数体只有一句话，可省略{}

如果仅有的一句话还是return，可继续省略return

特点: 箭头函数内外，共用同一个this!

4. 模板字符串: 简化复杂字符串拼接:

今后，只要字符串拼接，都用模板字符串代替+

如何:

整个字符串用``包裹

``中支持换行，支持"", ''

``中要想执行表达式，只要放在${...}即可

5. 解构: 简化批量赋值:

何时: 只要从一个大的对象中，挑选部分属性值使用时

如何: 3种:

1. 数组解构: 下标对下标

var [变量1,变量2,...]=[值1,值2,...]

同时声明多个变量

从原数组中对应位置取出值，赋值给相同位置的变量

2. 对象解构: 属性对属性

var {属性1:变量1, 属性2:变量2,...}={属性1:值1, 属性2:值2,...}

3. 参数解构:

定义时: function ajax({type,url, data, dataType}){

type, url, data, dataType

}

调用时:

ajax({

dataType:"json"

url:"xxx.php",

})

何时: 今后，只要参数是否有值，或顺序都不确定

6. for of: 简化普通for循环

何时: 只要依次获取数组/类数组对象中每个元素时

如何: for(var 变量 of 数组/类数组对象){

变量会自动获得数组/类数组对象中每个元素

}

局限:

1. 只能获得元素内容，无法获得位置！

2. 只能完整遍历，无法选择

3. 只能遍历数字下标的数组或类数组对象, 无法遍历关联数组和对象（用for in遍历）

7. class:

封装:

class类型名{

constructor(){

this.xx=xx;

this.xx=xx;

}

方法名(){

}

方法名(){

}

}

强调: class中构造函数和各个方法之间不用逗号分隔

继承:

class Father{

constructor(){...}

}

class Child extends Father{

constructor(){

super(属性参数...)

}

}

静态: static 不需要创建子对象，就可用构造函数直接调用的方法。

如何:

class 类型名{

static 方法名(){...}

}

调用:

类型名.方法名()

访问器属性:

class 类型名{

constructor(参数){

this.访问器属性=参数

}

//prototype

get 访问器属性(){ return this.数据属性}

set 访问器属性(val){ if(判断条件) this.数据属性=val}

}

8. Promise: 解决: 回调地狱

什么是callback hell: 由于使用参数传递回调函数，导致步骤多时，参数的嵌套层级很深。

何时: 只要异步调用，可能发生延迟时，都要用Promise代替传统参数callback

如何: 定义时

function 第一件事(){

return new Promise(fn=>{

第一件事的内容

fn()

})

}

function 第二件事(){

return new Promise(fn=>{

第二件事的内容

fn()

})

}

function 第三件事(){

第三件事的内容

}

调用时:

第一件事()//return Promise(fn)

.then(第二件事)//return Promise(fn)

.then(第三件事)

DOM DAY01

复习:

1. 将类数组对象复制为数组:

var arr2=Array.prototype.slice.call(arguments)

将类数组对象复制为数组，并选取指定位置的剩余元素

var arr2= Array.prototype.slice.call(arguments,starti)

相当于arguments.slice(starti)

其实更简单的: var arr2= [].slice.call(arguments,starti)

2. promise中的错误处理:

其实: new Promise(可接收2件事)

.then( ) .catch( )

new Promise((正常函数,出错函数)=>{

如果顺利执行:

调用正常()

否则

调用出错()

})

3. 等待多个任务完成

前提: 每个任务都必须都返回Promise

如何: Promise.all([

task1(), task2(),...

]).then(()=>{所有任务完成后才执行的任务})

正课:

1. 什么是DOM

2. DOM Tree

3. 查找

1. 什么是DOM: Document Object Model

什么是: 专门操作网页内容的API标准——w3c

为什么: 统一不同浏览器操作网页内容的API标准

优点: 几乎所有浏览器100%兼容

2. DOM Tree:

什么是: 网页中所有内容在内存中都是保存在一棵树形结构中

网页中每项内容(元素,文本,属性,注释...)，都是树上的一个节点对象。

唯一的树根节点: document

为什么: 树形结构是最好的保存上下级包含关系的结构

节点对象: Node

网页中每项内容都是DOM树上的一个节点对象:

所有节点都有的三个属性:

nodeType: 节点类型  
 何时: 只要判断节点的类型时

包括:

document 9

element 1

attribute 2

text 3

问题: 无法进一步区分元素的标签名

nodeName: 节点名称

何时: 只要进一步判断元素的标签名时

——可代替nodeType

包括:

document #document

element 全大写标签名

attribute 属性名——不常用!

text #text

nodeValue: 节点值 ——不常用

document null

element null

attribute 属性值

text 文本内容

3. \*\*\*查找: 4种:

1. 不需要查找可直接获得节点:

document.documentElement html

document.head head

document.body body

document.forms[id/i] form

2. 按节点间关系查找:

何时: 如果已经获得一个节点了。想找周围节点时。

包括: 2种树:

1. 节点树: 包含网页中所有内容的完整树结构

2大类关系:

1. 父子: 4种:

elem.parentNode elem的父节点

elem.childNodes elem的直接子节点

elem.firstChild elem下的第一个直接子节点

elem.lastChild elem下的最后一个直接子节点

2. 兄弟: 2种:

elem.previousSibling elem的前一个兄弟元素

elem.nextSibling elem的后一个兄弟元素

问题: 受看不见的空字符的干扰!

2. 元素树: 仅包含元素节点的树结构

2大类关系:

1. 父子: 4种:

elem.parentElement elem的父元素

elem.children elem的直接子元素

elem.firstElementChild elem下的第一个直接子元素

elem.lastElementChild elem下的最后一个直接子元素

2. 兄弟: 2种:

elem.previousElementSibling elem的前一个兄弟元素

elem.nextElementSibling elem的后一个兄弟元素

今后只要用DOM操作网页内容，都用元素树

说明: 元素树不是一棵新树，其实只是节点树的一个子集

childNodes和children: 动态集合(live collection)

什么是: 不实际存储属性值，每次访问集合都重新查找DOM树

优: 首次查找，效率高! 因为不用返回完整的属性。

缺: 每次访问集合，都会重新查找DOM树，降低效率

遍历:

不好: for(var i=0;i<children.length;i++){...}

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

遍历指定父元素下的所有后代元素: 2种:

1. 递归: 2步:

1. 定义函数仅遍历指定父元素下的所有直接子元素

2. 对每个直接子节点调用和父节点完全相同的操作

深度优先: 当一个节点同时拥有子节点和兄弟节点时，总是优先遍历子节点。所有子节点遍历完，才返回遍历兄弟节点。

2. 循环: 2步:

1. 定义迭代器:

迭代器: 可以依次获得每个后代元素节点的 专门对象

如何:

创建: var iterator=document.createNodeIterator(

parent, NodeFilter.SHOW\_ELEMENT, null, false

);

2. 循环调用迭代器，获得下一个节点对象:

var curr=iterator.nextNode()

内置深度优先遍历的算法

如果curr返回null，说明遍历结束

3. 按HTML查找: 4种:

1. 按id查找:

var elem=document.getElementById("id")

返回值: 一个元素

如果找不到返回null!

强调: 1. 只能在document上调用

2. 按标签名查找:

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

返回值: 多个元素的集合

如果找不到返回空集合

强调: 1. 可在任意父元素上调用

2. 不但找直接子元素，且在所有后代中查找

3. 按name查找:

var elems=document.getElementsByName("name")

返回值: 多个元素的集合

如果找不到返回空集合

强调: 只能在document上调用

4. 按class查找:

var elems=parent.getElementsByClassName("class")

返回值: 多个元素的集合

如果找不到返回空集合

强调: 1. 可在任意父元素上调用

2. 不但找直接子元素，且在所有后代中查找

3. 只要元素的一个class名匹配，就能找到该元素

强调: 返回的集合都是动态集合

问题: 每次只能按一个条件查找

当查找条件复杂时，步骤很繁琐

解决: 用选择器查找:

4. 用选择器查找:

1. 仅查找一个符合条件的:

var elem=parent.querySelector("选择器")

返回值: 一个元素

如果找不到,返回null

2. 查找多个符合条件的元素:

var elems=parent.querySelectorAll("选择器")

返回值: 多个元素的集合

如果找不到返回空集合

返回非动态集合: 实际存储属性值，即使反复访问集合，也不会导致反复查找DOM树

强调: 1. 可在任何父元素上调用

2. 选择器只要相对于当前父元素内部即可

3. 选择器的兼容性，受制于当前浏览器的兼容性

鄙视: 按HTML查找和按选择器查找的差别:

1. 返回值: 按HTML查找返回动态集合

按选择器查找返回非动态集合

2. 效率: 首次查找: 按HTML查找效率高

按选择器查找效率低

3. 易用性: 按HTML查找繁琐

按选择器查找简单

总结: 今后，只要一个条件即可找到想要的元素，首选按HTML查找

只要查找条件复杂，都选按选择器查找

jQuery中就是这么选择的

4. 修改:

内容: 3种:

1. html代码片段: elem.innerHTML 没有兼容性问题

2. 纯文本内容: elem.textContent 有兼容性问题

1. 去掉内嵌的标签

2. 翻译转义字符为正文

3. 表单元素的值: elem.value

属性:

样式:

DOM操作基本步骤:

0. 构建DOM树

1. 查找触发事件的元素

2. 绑定事件

3. 在事件处理函数中，查找要修改的元素

4. 修改元素(内容,属性,样式)

DOM DAY02

正课：

1. 修改:

2. 添加/删除:

1. 修改:

内容: .innerHTML .textContent .value

属性: 3种:

1. HTML标准属性: 2种:

1. 核心DOM: 最初的DOM API，要求支持所有结构化文档

优: 几乎万能 缺: 繁琐

获取属性节点: var attrNode=elem.attributes[i]

获取属性值: attrNode.value

其实可以一步: var value=elem.getAttribute("属性名")

修改属性值: elem.setAttribute("属性名","值")  
 移除属性: elem.removeAttribute("属性名")  
 判断是否包含指定属性: elem.hasAttribute("属性名")

2. HTML DOM: 专门操作HTML内容的API

对核心DOM 常用API的简化

优: 简单 缺: 不是万能

简化: HTML DOM提前将所有标准属性，封装在了元素对象上，可用.直接访问.

比如:

获取属性值: elem.属性名

修改属性值: elem.属性名="值"  
 移除属性: elem.属性名=""  
 判断是否包含指定属性: elem.属性名!==""

特例: class属性:

ES的对象中已经先入为主的包含了内部属性class，用来记录对象创建时的类型名。

HTML的class属性就无法同时存在

所以: DOM: html的class属性，更名为className

2. 状态属性: disabled selected checked

1. 不能用核心DOM修改: 因为值是bool类型

2. 只能用HTML DOM,打.修改

3. 自定义扩展属性:

何时: 2种:

1. 在元素上保存自定义数据

2. 代替其他选择器，用来查找元素绑定事件

如何: 2套:  
 不能用HTML DOM访问, 因为不是标准属性，未被提前封装在DOM元素对象中

1. 普通属性名+核心DOM

2. HTML5:   
 定义属性时: data-属性名="值"  
 获取或修改属性值: elem.dataset.属性名

固定用法: 用自定义扩展属性代替其它选择器，为元素绑定事件

为什么:

id: 唯一

class: 经常变化

elem: 做一种效果，可能使用任何元素

解决: 今后，只要给元素绑定事件时，都要先为元素添加自定义扩展属性，再用自定义扩展属性查找元素，绑定事件

优: 没有个数限制, 不会受样式影响而变化，不受元素限制

如何:

定义: data-属性名="值"

查找: 只能用属性选择器查找:[data-属性名=值]

样式: 2种:

1. 内联样式:

修改: elem.style.css属性名

强调: 1. css属性名都要去横线变驼峰

2.长度数值必须加px单位

获取:

问题: 不能用elem.style.css属性名

原因: elem.style仅包含内联样式! 无法获得内部或外部样式表中的样式

解决: 获取计算后的样式!

计算后的样式: 最终应用到元素上的所有样式的集合

且将相对值换算为绝对值

何时: 只要获取样式时，都要获取计算后的样式

如何: 2步:

1. 获得计算后的style对象:

var style=getComputedStyle(elem);

2. 从style中获得css属性值

var value=style.css属性值

强调: getComputedStyle获得样式都是只读！

2. 内部/外部: 3步:

1. 获得样式表对象:

var sheet=document.styleSheets[i];

2. 获得样式表对象中的cssRule

什么是cssRule: 样式表中每个{}就是一个cssRule

如何获得: var rule=sheet.cssRules[i]

3. 获得cssRule中的style对象的css属性

var value=rule.style.css属性

问题: 用elem.style.css属性一句话只能修改一个css属性

如果同时修改多个属性，代码会很繁琐

解决: 用class批量应用样式

如何: 2步:

1. 在css中准备好各种class

2. 用程序，在对应情况下，选择对应的class应用！

2. 添加/删除:

添加: 3步:

1. 创建空元素

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

<a></a>

2. 设置必要属性

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

a.innerHTML="go to tmooc";

3. 将新元素添加到DOM树:3种api:

parent.appendChild(a) 将a追加到指定父元素下末尾

parent.insertBefore(a, child) 将a插入到指定父元素下的现有子元素之前

parent.replaceChild(a, child) 用a替换指定父元素下的现有子元素

优化: 尽量减少操作DOM树的次数, 从而减少重排重绘

为什么:

HTML页面加载过程:

html -> DOM Tree

↓

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

↑

css -> cssRules

如何:

1. 如果同时添加父元素和子元素时，应先在内存中，将所有子元素添加到父元素中，再最后一次性将父元素添加到页面上

2. ?

DOM DAY03

正课:

1. 添加/删除:

2. HTML DOM 常用对象

3. BOM

window

打开关闭窗口

history

location

定时器

navigator

1. 添加删除:

优化: 尽量减少操作DOM树的次数

为什么: 减少重排重绘

如何: 2种:

1. 如果同时添加父元素和子元素，则应该现在内存中，将子元素添加到父元素，再最后将父元素一次性整体添加到DOM Tree

2. 如果父元素已经在页面上了，要添加多个平级子元素，则应该使用文档片段:

什么是文档片段: 内存中临时存储多个子元素的虚拟父元素

何时: 只要向网页中同时添加多个平级子元素时，都可用文档片段

如何: 3步:

1. 创建文档判断

2. 将子元素添加到文档片段

3. 将文档片段整体添加到DOM 树

删除: parent.removeChild(child)

通常: child.parentNode.removeChild(child)

2. HTML DOM常用对象:

Image: 创建: var img=new Image();

Select: 代表页面上一个select元素

属性:

.selectedIndex 获得当前select选中的option的下标位置

.value 获得select中选中的option的值

如果选中的option没有value属性，则用innerHTML代替

.options 获得当前select下所有option的集合

.options.length 获得select下所有option的个数

.options.length=0

.length => .options.length

清空所有option .length=0

方法: .add(option) 追加一个option

问题: .add不支持文档片段

.remove(i) 移除i位置的option

Option: 代表select下一个option元素

创建: var opt=new Option(text,value)

属性: .text .value .index

Table: 代表一个table元素

管着行分组:

创建行分组: var thead=table.createTHead()

var tbody=table.createTBody()  
 var tfoot=table.createTFoot()

删除行分组: table.deleteTHead()

table.deleteTFoot()

获取行分组: table.tHead table.tFoot table.tBodies[i]

行分组: 管着行:

添加行: var tr=行分组.insertRow(i)

在i位置插入一个新行

固定套路: 1. 在末尾追加新行: .insertRow()

2. 在开头插入新行: .insertRow(0)

删除行: 行分组.deleteRow(i) ——不常用!

问题: i要求是在当前行分组内的相对下标位置

无法自动获得

应换为: table.deleteRow(tr.rowIndex)

tr.rowIndex可自动获得当前行在整个table内的位置

获取行: 行分组.rows[i]

行: 管着格

添加格: var td=tr.insertCell(i)

固定用法: 行末尾追加一个格: tr.insertCell()

说明: 只能创建td，不能创建th

删除格: tr.deleteCell(i)

获取格: tr.cells[i]

Form: 代表一个form元素

获取: var form=document.forms[i/id]

属性: .elements 获得表单中所有表单元素的集合

.elements.length 获得表单中，表单元素的个数

.length => .elements.length

方法: .submit() 代替submit按钮，在程序中实现手动提交表单

表单元素:

获取: form.elements[i/id/name]

更简化: form.name

方法: .focus 让当前表单元素获得焦点

.blur

3. BOM: Browser Object Model

什么是: 专门操作浏览器窗口的API

没有标准, 导致浏览器兼容性问题

包括:

window: 2个角色:

1. 代替global充当全局作用域对象

2. 封装所有DOM和BOM的API

history:

location:

navigator:

document: 封装DOM树，并提供操作DOM的API

event

screen

window:

属性: .innerWidth, .innerHeight 浏览器窗口中，文档显示区的宽和高

方法: .open() .close()

.open("url","name")

三种: 1. 在当前窗口打开，可后退: .open("url","\_self")

2. 在新窗口打开，可打开多个: .open("url","\_blank")

3. 在新窗口打开，只能打开一个:

.open("url","自定义窗口名")

history: 保存当前窗口打开后，成功访问过的url的历史记录栈

在当前窗口中，每访问一个新url，都会将新url压入history

API: history.go(n)

3种: 前进: history.go(1) 后退: history.go(-1)

刷新: history.go(0)

location: 保存当前窗口正在打开的url的对象

属性:

.href 完整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

.hash #锚点地址

.search ?查询字符串

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(url)

3. 重新加载页面: 刷新: 2种:

1. 普通刷新:

优先从浏览器本地缓冲获取资源:

F5

history.go(0)

location.reload(/\*false\*/)

2. 强制刷新:

无论本地是否有缓存，总是强制从服务器获取资源

location.reload(true)

定时器: 2种:

1. 周期性定时器:

什么是: 让程序每隔指定的时间间隔，反复执行一项任务

何时: 只要让程序按照指定的时间间隔，自动执行一项任务

如何: 3件事:

1. 任务函数: 让定时器反复执行的任务

2. 启动定时器: timer=setInterval(task, interval)

让程序，每隔interval 毫秒自动执行一次task任务

3. 停止定时器: clearInterval(timer)

timer: 定时器的序号, 在内存中唯一标识定时器的整数

专门用于停止定时器

如何获得: 只能在启动定时器时获得。

何时: 只要一个定时器可能被停止，就要在启动时，先保存定时器序号

好的习惯: 在clearInterval之后，手动清除timer中残留的序号: timer=null;

停止定时器: 2种情况:

1. 用户手动停止:

2. 定时器可自动停止:

在任务函数中，设定临界值，如果没有达到临界值，则继续执行任务，否则，如果达到临界值，就自动调用clearInterval

2. 一次性定时器:

什么是: 让程序先等待一段时间，再执行一次任务。执行后，自动停止。

何时: 只要让程序延迟执行一件事时

如何: 3件事:

1. task

2. 启动定时器: timer=setTimeout(task,wait)

让程序等待wait毫秒后，自动执行一次task，执行后自动停止

3. 停止定时器: clearTimeout(timer)

DOM DAY04

正课:

1. 定时器原理:

2. navigator:

3. event:

4. 页面滚动:

附加: 对话框特效

1. 定时器原理:

定时器中的任务函数，必须等待主程序所有语句执行后，才能执行。

2. navigator:

什么是: 保存浏览器配置信息的对象

包括:

.cookieEnabled: 判断当前浏览器是否启用cookie

什么是cookie: 在客户端持久存储用户私密数据的小文件

为什么: 内存中所有数据都是临时的! 程序关闭，内存中一切变量都释放!

何时: 只要希望在客户端持久保存数据，都用cookie

.plugins: 包含浏览器所有插件信息的集合

什么是插件: 为浏览器添加新功能的小软件

如何判断是否安装指定插件:

.userAgent: 保存浏览器名称和版本号的字符串

何时: 只要判断浏览器名称和版本号

4. event:

什么是事件: 人为触发的，或浏览器自动触发的页面内容状态的改变。

什么是事件处理函数: 当事件发生时，自动执行的函数。

如何绑定: 3种:

1. 在HTML中绑定:

绑定: <ANY on事件名="js语句">

当事件发生时: 自动执行js语句

问题: 不符合内容与行为分离的原则，不便于维护和重用

但是: 在组件开发中，反而要求内容，行为和样式集中定义在一个小组件内，自成体系。

2. 在js中绑定, 每个事件只能绑定一个处理函数:

ANY.on事件名=function(){ ... }

当事件发生时: ANY.on事件名() //this->ANY

问题: 用赋值方式绑定事件处理函数

3. 在js中绑定，每个事件可绑定多个处理函数:

ANY.addEventListener('事件名',handler)

在浏览器中为ANY元素的指定事件，添加一个事件监听对象。将事件监听对象加入到浏览器的监听队列中。

触发事件时: 浏览器会遍历监听队列中的每个监听对象，找到触发事件元素上对应事件的监听对象，调用其处理函数

移除事件监听:

ANY.removeEventListener('事件名',handler)

说明: handler必须是绑定时使用的原函数对象

强调: 如果一个处理函数，有可能被移除，则不能使用匿名函数绑定。应使用有名的函数绑定

DOM事件模型:

什么是: 从事件触发到处理函数执行，所经过的过程

3个阶段:

1. 捕获capture: 由外向内，记录各级父元素上绑定的事件处理函数。——仅记录，不触发!

2. 目标触发: 优先触发实际点击的元素上绑定的处理函数

3. 冒泡执行: 由内向外，按捕获阶段记录的处理函数的倒序，依次执行父元素上的处理函数。

事件对象:

什么是: 当事件发生时，自动创建的，封装事件信息的对象

何时: 只要希望获得事件信息，或修改事件的默认行为时

如何获取: 事件对象默认总是以处理函数第一个参数，自动传入

如何使用:

1. 取消冒泡: e.stopPropagation()

2. 利用冒泡:

优化: 尽量减少事件监听的个数

为什么: 浏览器查找事件监听，采用的是遍历的方式

事件监听多，浏览器查找就慢

何时: 如果对多个子元素绑定相同事件时，都要利用冒泡

如何: 只要在父元素绑定一次，所有子元素共用即可!

2个难题:

1. 获得目标元素:

目标元素: 最初实际触发事件的当前元素

如何获得:

错误: this->父元素

正确: e.target

2. 筛选目标元素:

比如: 通过nodeName, class, 内容。。。

3. 阻止默认行为:

何时: 只要事件的默认行为不是想要的

如何: e.preventDefault();

何时:

1. 用a当按钮时，a会自动向地址栏中添加#锚点地址。

2. 提交表单时，如果验证没通过，可阻止提交

自定义表单提交:

input button + onclick + form.submit

input submit + form.onsubmit事件 + e.preventDefault()

3. HTML5中拖拽API: 首先要阻止浏览器默认的拖拽行为

4. 鼠标坐标: 3组:

1. 相对于屏幕左上角: e.screenX, e.screenY

2. 相对于文档显示区左上角: e.clientX, e.clientY

3. 相对于当前元素左上角: e.offsetX, e.offsetY

4. 页面滚动:

事件: window.onscroll

获得页面滚动过的高度: body顶部超出文档显示区顶部的距离

scrollTop=document.documentElement.scrollTop

||document.body.scrollTop;

滚动API:

window.scrollTo(left, top)

window.scrollBy(left的增量,top的增量)

JQUERY DAY01

正课:

1. 什么是jQuery

2. 如何使用jQuery

3. 查找

1. 什么是jQuery

第三方的 执行DOM操作的 极简化的 函数库

第三方: 下载

执行DOM操作: 学习jQuery还是在学DOM

极简化: 对DOM操作的终极简化

函数库: jQuery用函数解决一切问题，不再使用属性方式

为什么: 2个原因:

1. 是DOM操作的终极简化: 4个方面

1. 增删改查

2. 事件绑定

3. 动画

4. ajax

2. 解决了大部分浏览器兼容性问题:

凡是jQuery允许使用的，都没有浏览器兼容性问题

2. 如何使用:

3步:

1. 上官网，看是什么

2. 上官网，看快速入门案例

3. 看手册

下载: 版本:

1.x: 兼容旧版本浏览器，体积稍大

压缩: .min.js 删除所有注释和缩进，且极简化变量名

体积小, 可读性差 —— 生产环境中

未压缩: .js 具有完备的注释和缩进，且具有见名知义的变量名

体积大，可读性好 —— 开发环境中

2.x: 不再兼容旧浏览器，体积变小

3.x: 不再兼容旧浏览器，且功能更强大:

新特性:

1. for...of...代替each

2. requestAnimationFrame()

3. 支持Promise

4. 废弃bind, unbind, delegate, undelegate

5. 所有代码运行在严格模式下

在网页中加载jquery: 2种:

1. 将jQuery.js下载到网站本地文件夹，用相对路径加载

2. CDN网络:

内容分发网络: 根据客户端到网络中一台服务器的网络状况，自动选择最优的服务器下载资源。

何时: 生产环境中，共享的库/框架

如何: 不用下载资源到本地服务器

而是直接使用CDN地址，引用网络中的资源

强调: 都要先引入jQuery.js，再编写自定义js脚本

原理: 引入jQuery.js后，发生了什么:

jQuery.js向全局添加了一种新类型jQuery:

1. 构造函数: 用来创建jQuery类型的子对象

2. 原型对象: 用来保存所有jQuery类型的子对象共有的成员方法

要想使用jQuery简化版API，必须先创建jQuery类型的子对象。

为什么: 普通DOM对象不是jQuery类型的子对象，无法使用jQuery原型中的简化版API

如何创建: 3种:

1. 用选择器查找DOM元素，再将找到的DOM元素保存到新创建的jQuery对象中

var jQuery对象=jQuery("selector")

1. 用选择器查找符合条件的DOM元素

2. 将找到的DOM元素保存到新创建的jQuery对象中

2. 直接将已经到手的DOM元素封装到新建的jQuery对象中

var jQuery对象=$(DOM对象);

总结: 专门封装找到的DOM对象的类数组对象

jQuery API三大特性:

1. 一个函数两用: 没给新值，默认读取原值

给了新值，则变为修改操作

2. 自带遍历效果: 对整个jQuery类数组对象调用简化版API，等效于自动对jQuery对象中每个DOM对象分别调用简化版API

3. 查找:

1. 按选择器查找:

jQuery支持全部css的选择器, 且额外增加了个别选择器

基本选择器: 同CSS

层次选择器: 同CSS

子元素过滤选择器: 同CSS

基本过滤选择器(位置过滤): jQuery中新增

:first/last :eq/gt/lt(i) :even/odd

属性过滤: 同CSS

可见性: :hidden :visible

var bool=$(...).is("selector")

判断任意元素是否符合selector的要求！

JQUERY DAY02

正课:

1. 查找

2. 修改

3. 按节点间关系查找

4. 添加/删除/替换/克隆

5. 事件绑定

jQuery API三大特性:

1. 一个函数两用: 没给新值，默认读取原值

给了新值，则变为修改操作

2. 自带遍历效果: 对整个jQuery类数组对象调用简化版API，等效于自动对jQuery对象中每个DOM对象分别调用简化版API

3. 大多数API都返回正在操作的jQuery对象本身

1. 查找:

内容过滤: jQuery新增:

:contains(文本) 以元素的文本内容作为筛选条件

:has(选择器) 以元素的子元素特征作为筛选条件

:parent 非空的元素

:empty 空元素

可见性过滤:

:hidden 选择所有隐藏的元素

只能选择: display:none input type="hidden"

:visible

表单元素过滤:

:input 选择所有表单元素,包括:input select textarea button

:type: 每种type都有对应的选择器:

:text :password :checkbox :radio :submit ...

状态过滤: :disabled :checked :selected :enabled

2. 修改:

内容:

HTML代码片段: $(...).html() .innerHTML

纯文本内容: $(...).text() .textContent

表单元素的内容: $(...).val() .value

清空内容: $(...).empty()

属性:

HTML标准属性: $(...).attr("标准属性名"[,"值"])

$(...).attr({

标准属性:值,

标准属性:值,

})

状态属性: $(...).prop("状态属性"[,bool])

自定义扩展属性:

$(...).data("属性名",值) .dataset.属性名

样式:

1. 获取或修改内联样式: $(...).css("css属性名",值)

问题: 一次只能修改一个css属性的值

解决: 1. 链式操作:

2. 一个css同时修改多个css属性:

$(...).css({

css属性名:值,

css属性名:值,

})

强调: .css()在修改时，自动修改内联样式

在获取时，自动获取计算后的样式

2. 用class批量应用样式:

$(...).addClass("...")

$(...).removeClass("...")

$(...).hasClass("...")

$(...).toggleClass("...")

等效于:

if($btn.hasClass("down"))

$btn.removeClass("down")

else

$btn.addClass("down");

3. 按节点间关系查找:

1. 父子:

$(...).parent()

$(...).children(["selector"])

$(...).find("selector")

$(...).children().first() .firstElementChild

$(...).children().last() .lastElementChild

2. 兄弟:

$(...).next()

$(...).nextAll()

$(...).prev()

$(...).prevAll()

$(...).siblings(["selector"])

$(...).index() 查找任意元素在当前父元素下的下标位置

4. 添加/删除/替换/克隆

添加:

1. 用HTML代码片段创建新元素:

var $a=$("<a href='url'>text</a>");

2. 将新元素添加到DOM树

末尾追加

$parent.append($a) //return $parent

$a.appendTo($parent) //return $a

开头插入

$parent.prepend($a)

$a.prependTo($parent)

在当前元素前后插入

$child.after($a) 将a插入到child之后

$child.before($a) 将a插入到child之前

删除: $().remove()

替换:

$(旧).replaceWith($新) //return $旧

$新.replaceAll(旧) //return $新

克隆:

$().clone() 浅克隆: 仅克隆属性和样式，不克隆行为

$().clone(true) 深克隆: 即克隆属性和样式，又克隆行为

jQuery中$的原理: 举例说明，你阅读过的常用库/框架的代码部分

$是一个工厂函数, 内部自动new jQuery创建jQuery类型的子对象.

3种情况:

1. 参数为选择器时:

先查找DOM元素，再将找到的DOM元素保存在新创建的jQuery对象中

优化(Speed-up):

1. 如果选择器是id，则自动调用getElementById

2. 如果选择器是标签名,则自动调用getElementsByTagName

3. 如果选择器是class,则自动调用getElementsByClassName

4. 如果选择器复杂，自动调用querySelectorAll

2. 参数为DOM元素:

直接将DOM元素保存到新创建的jQuery对象中

3. 参数为HTML代码片段:

用html代码片段创建一个/多个新DOM元素，再将创建的DOM元素保存到新创建的jQuery对象中

5. 事件绑定:

jQuery中共有几种事件绑定方式，有什么不同:

1. bind/unbind() 代替addEventListener/removeEventListener

unbind() 三种重载:

unbind("事件名",处理函数): 移除当前元素指定事件上的一个指定的处理函数

unbind("事件名"): 移除当前元素指定事件上所有处理函数

unbind(): 移除当前元素上所有事件的处理函数

2. one() 只触发一次，触发后自动解绑

3. live/die() 已被废弃, 将所有元素的所有事件处理函数，都集中绑定在document根节点上

4. delegate/undelegate() 其实就是利用冒泡

$parent.delegate("selector","事件名",function(){

this->e.target->目标元素

执行操作!

})

鄙视: 简述jQuery中事件委托的原理

$parent.bind("事件名",e=>{

e.target->目标元素

if($(e.target).is(selector))

执行操作!

})

总结: bind vs delegate(利用冒泡)

1. 绑定位置: bind直接绑在目标元素(子元素上)

delegate(利用冒泡) 绑在父元素上

2. 监听个数: bind 监听个数多

delegate(利用冒泡) 监听个数少

3. 动态生成子元素:

delegate(利用冒泡) 动态生成的新子元素可自动获得父元素上的事件处理函数

bind 动态生成的新子元素必须反复绑定

5. on/off()

on("事件名",处理函数) 代替bind()

on("事件名","选择器",处理函数) 代替delegate()

6. 终极简化:

对常用的事件，进行了终极简化:

$(...).事件名(处理函数)

JQUERY DAY03

正课:

1. 事件:

2. 动画:

3. 类数组对象操作:

4. 添加自定义函数:

1. 事件:

1. 页面加载后自动执行: 2种:

1. DOMContentLoaded: 仅当dom内容(html和js)加载完就提前触发：

$(document).ready(function(){...})

简写: $().ready(function(){...})

更简化: $(function(){...}) => $(()=>{...})

2. 等待网页所有内容(html,css,js,图片)加载完成，才触发: 晚

window.onload=function(){...}

2. 鼠标事件:

mouseover mouseout : 进出子元素，同样反复触发父元素的处理函数

mouseenter mouseleave : 进出子元素，不再反复触发父元素的处理函数

hover等于同时绑定鼠标进入和移出事件

hover(鼠标进入, 鼠标移出)

强调: 代替的是: mouseenter和mouseleave

其实可以更简化:

如果鼠标进入或移出事件，可统一为一个函数，只需要传入一个函数即可。即响应鼠标进入，又响应鼠标移出事件

3. 模拟触发:

$(...).事件名() 调用该处理函数

$(...).trigger("事件名")

2. 动画:

简单动画: 3种:

1. 显示隐藏: .show() .hide() .toggle()

说明: 没有参数时，默认使用display控制显示隐藏，所以，不带动画效果

带动画效果: .show(ms)

问题:

1. 效率: 用定时器实现的动画效果，效率低

2. 效果是写死的，不便于维护

解决: 用transtion 代替 动画API

2. 上滑下滑: .slideUp() slideDown() slideToggle()

3. 淡入淡出: .fadeIn() fadeOut() fadeToggle()

万能动画:

$(...).animate({

目标样式

},ms)

强调: animate只支持单个数值的属性

问题:

1. 不支持颜色动画

2. 用定时器实现的动画效果

排队和并发:

并发: 多个css属性同时变化

放在一个animate内的多个css属性，是并发变化

排队: 多个动画效果，先后顺序执行

对同一个元素，先后调用多个动画API，则默认是排队执行

动画结束后自动执行:

$(...).动画API(......, callback)

callback会在动画结束后，自动执行

callback中的this，自动指当前正在播放动画的DOM对象

停止动画:

$(...).stop()

问题: 默认仅停止队列中当前正在播放的一个动画!

解决: 停止当前动画并清空队列: .stop(true)

选择/判断一个元素是否正在播放动画:

:animated

3. 类数组对象操作:

$(...).each( (i,elem)=>{ .forEach()

elem->当前正在遍历的元素->DOM

对elem执行任何操作

})

鄙视: $(...).each() vs $.each(obj, fn)

1. 存储位置:

$(...).each() 保存在jQuery类型的原型对象中

$.each() 定义在jQuery构造函数上

2. 调用方式:

$(...).each() 只能用jQuery类型的子对象调用

$.each() 让任何类数组对象都可享用each

$(...).index() indexOf()

$("选择器").index(DOM/jQ)

在左边的结果集合中，找右边的DOM或jQ元素的位置i

$("选择器").index("选择器")

在右边的结果集合中，找左边的选择器对应元素的位置i

$("选择器").index() 在当前元素的父元素内，找当前元素的位置i

4. 添加自定义函数:

如果jQuery的API不够用时:

如何添加:

向jQuery.fn中添加新函数=function(){  
 //this->将来调用该函数的jQuery类型的子对象  
 }

强调: 向jQuery原型对象中添加新函数，首先要验证是否已经引入了jQuery类型

JQUERY DAY04

正课:

1. 官方插件:jQuery UI

2. 第三方插件:

3. \*\*\*封装自定义插件:

1. 官方插件:jQuery UI

插件/组件: 具有独立样式和功能的小部件

为什么: 复用!

何时: 只要在项目中，发现反复使用的功能和样式，都可封装为插件, 反复使用

jQuery UI:

强调: jQuery UI依赖于jQuery，必须先引入jQuery，再引入jQuery UI

1. 效果:

1. 重写了animate方法，支持颜色动画

2. 为简单动画API，提供了更多的动画效果

3. 重写了addClass/removeClass/toggleClass，添加了动画效果

2. 交互

3. 部件:

用法: 3步:

1. 按部件的约定，定义HTML内容

2. 引入部件的css, js，应先引入jquery

3. 在自定义脚本中找到部件的父元素，调用部件API

有些API需要设置参数

原理: 侵入:

部件按自身的需要，隐式自动向HTML中添加class和行为

1. accordion

2. autocomplete

3. datepicker

4. dialog

2. 第三方插件:

validation:

fileUpload:

wysiwyg:

JQUERY DAY05

正课:

1. 第三方插件:

Fileupload

Wysiwyg

2. \*\*\*封装自定义插件

3. Ajax

4. \*\*\*\*\*跨域请求

1. 第三方插件:

fileupload:

1. 普通文件上传:

html: form method=“post” enctype=”multipart/form-data” action=”xxx.php”

单击提交按钮时，已经将文件上传到服务器

php: apache隐式自动将上传的文件，保存在tmp目录下

获得文件的基本信息($\_FILES[“name”])，验证或保存

将tmp目录中临时存储的文件，转储到目标目录下

2. ajax上传:

$.ajax({

type:”post”,  
 [url:”xxx.php](url:)”,

data: new FormData($(“#form”)[0]),

contentType:false,  
 processData:false

})

3. 上传插件:

Wysiwyg: 富文本编辑器: rich text editor

扩展: masonry

2. 封装自定义插件:

前提: 必须使用HTML，css和js的方式已经实现了插件的效果

如何: 2种:

1. jQuery侵入方式:

如何定义插件:

1. css文件封装插件样式:

2. 提前约定页面的HTML结构

3. 在jQuery的原型对象中添加插件函数

插件函数中2件事:

1. 侵入class和data-\*扩展属性

2. 为拥有data-\*扩展属性的元素添加事件

使用插件四步:

引入插件的css:

定义页面结构:

引入jquery和插件js

调用方式: $(父元素).插件函数()

2. bootstrap方式:

如何定义插件:

1. css文件封装插件样式:

2. 提前约定页面的HTML结构

3. 手动添加class和data-\*属性

4. 在插件的js文件中，查找data-\*属性的元素，绑定事件

使用插件四步:

引入插件的css:

定义页面结构, 选择要用的css样式类

为元素添加自定义扩展属性，从而添加行为

引入插件的js:

3. jquery中的ajax:

$.ajax({

type:“get/post”,

[url:”xxx.php](url:)”,

data: 查询字符串/对象/new FormData(),

dataType: “json”

}).then(data=>{

… …

})

其中:

data: 即支持查询字符串，又支持对象结构

dataType: 可省略

省略dataType, 则jquery ajax函数可根据服务端header的响应 头设置，自动决定是否调用JSON.parse

更简化:

$.get(“url”[,data]).then()

$.getJSON()

$.getScript()

$.post(“url”[,data]).then()

更简化: 获得一段HTML片段，并填充到指定父元素下

$(父元素).load(“url”)

$.get(“url”).then(html=>{ $(html).appendTo($父元素)})

4. \*\*\*\*\*跨域请求:

跨域:请求的发起网页地址origin和请求的目标地址不一致:

3种: 协议不同, 一级/二级域名不同, 端口号不同

问题: xhr对象禁止发起跨域请求的

解决: 不是所有元素/对象都禁止跨域

link script img iframe … 都支持跨域

方法一: script (可加载并执行js脚本)

请服务端返回一条可执行的js语句，填充要返回的数据

用script去请求服务端，获得可执行的js语句

专业名词: JSONP JSON with padding (填充式json)

实现一: 将处理数据的语句，直接定义到服务器端

缺点: 众口难调

实现二: 2步:

1. 在客户端提前定义处理函数，接收一个参数
2. 将调用函数的语句放在服务器端返回

缺点: 函数名写死！

实现三: 3步:

1. 在客户端提前定义处理函数，接收一个参数
2. 将函数名以查询字符串参数的形式传递给服务器端
3. 服务器端获得函数名，将其和要返回的数据拼接为一条js语句

特殊: 动态发起跨域请求:

1. 动态创建script并追加到body结尾, 并将script元素保存在公共的变量中
2. 在处理函数结尾，自动删除保存的script元素

jQuery中跨域:

$.ajax({type:”get”,url:”xxx.php”,datatype:”jsonp”}).then(处理函数！)

方法二: 服务器端设置允许跨域

其实: 服务器端一句话可解决跨域:

header(“Access-Control-Allow-Origin: \* ”)

HTML5 DAY01

今天学习的内容  
chengtao@tedu.cn

1:简介课程安排

(1)js/dom/jq 复习

(2)html5(十个新特性 canvas;视频;音频)

(3)bootstrap(响应式布网布框架)

(4)Nodejs(效率)

2:html5新特性(十)----(项目后台)

(1)新的语义标签

(2)增强型表单\*

(3)音频和视频

(4)Canvas绘图

(5)SVG绘图

(6)地理定位

(7)拖动API

(8)Web Worker

(9)Web Storage

(10)Web Socket

3增强型表单

3.1 新input type <input type=?>

H4:text;checkbox;password;radio;submit;reset;File;

H5:email;url;number;search;color;date;month;week

3.2 新的表单元素

H4:input;button;select;textarea

H5:datalist;progress;meter;output

3.3:html5新特性---datalist(数据列表)

<datalist id=”list3”> datalist本身不可见

<option>xxx</option>

<option>yyy</option>

</datalist>

<input type=”text” list=”list3”/>

datalist为input提供输入建议列表

3.4: html5新特性---progress(进度条)

<progress></progress> 左右晃动进度条

<progress value=”0.5” /> 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%停止.

3.4: html5新特性---meter(刻度尺)

Meter：用于标示一个值所值的范围:不可接受(红色),可以接受(黄色),非常优秀(绿)

薪水:

<meter min=”最小值” max=”最大值” low=”下限” high=”上限”

最佳值=”” value=”当前值”>

练习:使用定时器+meter实现可以动态变化的刻度尺，观察颜色变化..

3.5: html5新特性---output

output:输出，语义标签,没有任何外观样式，样式等同于span

商品单价: 3.50

购买数量: <input type=”number” value=”1”>

小计:<output>7.00</output>

3.6:html5新特性—(表单元素新属性)

H4:type;id;value;name;style;readyonly;disabled;checked

H5:

1. placeholder 占位符
2. autofocus 自动获取输入焦点
3. multiple 允许输入框中输入多个值，用逗号分值
4. form 用于把输入域放置在表单外部

<form id=”f3”></form>

<input type=”text” form=”f3” />

1. required 必填空
2. maxlength 字符串最大长度
3. minlength 字符串最小长度
4. min 指定数值最小值
5. max 指定数值最大值
6. pattern 指定输入内容符合正则表达式

login.html login.js

作业1: 搜索下载 图形验证码功能php程序

作业2：分析上传图片功能

HTML5 DAY02

今天学习的内容

1:复习昨天知识重点

2:作业1:

学子商城

\*\*Canvas项目(鱼妈妈鱼宝宝)

\*\*视频项目(回家)

3:今天的目标

3.1 html5新特性--视频与音频 (重点)

Flash被H5取代

Flash绘图(AS/Flex) 🡺 canvas/svg

Flash动画 -> canvas/svg 定时器

Flash视频和音频播放 -> video/audio

Flash 客户端存储 -> webstorage

3.2:H5新特性—视频播放

<video src=”x.mp4”></video>

<video>

<source src=”x.mp4”></source>

<source src=”x.ogg”></source>

<source src=”x.webm”></source>

您的浏览器版本太低，请升级

</video>

它本身是一个300\*150的inline-block元素

成员属性

autoplay 是否自动播放

controls 是否显示控件

muted 是否静音

loop 是否循环播放

poster 在播放第一帧画面之前广告(图片)

preload 视频加载策略

auto: 预加载一定时长视频和元数据

metadata: 仅预加载元素数(尺寸，时长，第一帧画面)

none: 不加载任何内容

##js对象属性

currentTime 当前播放时长

duration 总时长

paused: 当前视频是否处理暂停状态

volume（0~1） 音量

playbackRate: 回放速率: 大于1快播 小于1慢放

#js 成员方法

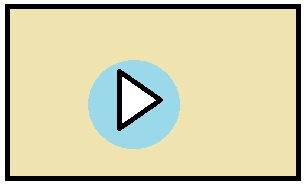
play() 播放视频

pause() 暂停视频播放

#js 事件

onplay 当视频开始播放触发事件

onpause 当视频暂停播放触发事件



练习1:不使用video自带控件,自定义播放暂停按钮

鼠标移出视频区域隐藏按钮，鼠标移入，显示按钮

练习2:视频暂停就显示广告，只要播放广告隐藏

3.3:H5新特性—音频

<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，但是没有controls属性，

则display:none;

成员属性

autoplay 是否自动播放

controls 是否显示控件

muted 是否静音

loop 是否循环播放

preload 视频加载策略

auto: 预加载一定时长视频和元数据

metadata: 仅预加载元素数(尺寸，时长，第一帧画面)

none: 不加载任何内容

##js对象属性

currentTime 当前播放时长

duration 总时长

paused: 当前视频是否处理暂停状态

volume（0~1） 音量

playbackRate: 回放速率: 大于1快播 小于1慢放

#js 成员方法

play() 播放视频

pause() 暂停视频播放

#js 事件

onplay 当视频开始播放触发事件

onpause 当视频暂停播放触发事件

3.4 html5 新特性—canvas绘图 (重点)

网页中的实时走势图,抢红包,网页游戏,地图应用..

(1)SVG 绘图 2D矢量绘图技术，2000年出现,后纳入h5

(2)Canvas绘图 2D位图绘图技术，H5提出

(3)WebGL绘图 3D绘图技术，尚未纳入H5标准

|  |
| --- |
| Canvas绘图难点所在:  (1)坐标系  (2)单词比较多 |

3.5 html5 新特性—canvas

Canvas画布:画布是H5提供的绘图基础

<canvas width=”500” height=”400”>

您的浏览器版本太低，请升级

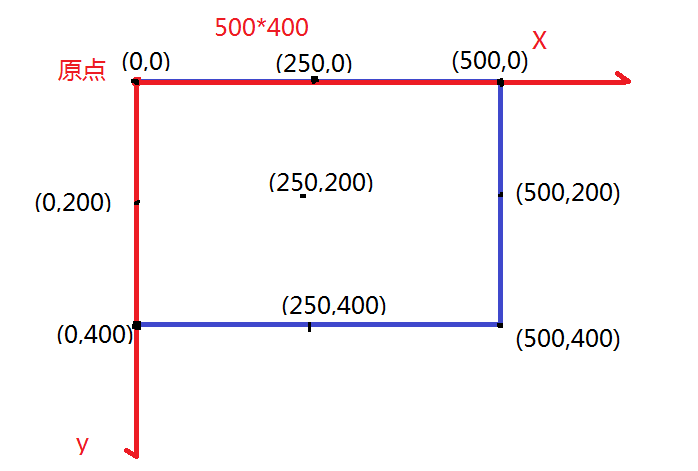
</canvas>

Canvas标签在浏览器中默认是300\*150的inine-block,画布宽度高度属性只能用js/属性来赋值.

不能用CSS样式赋值.

每个画布上有且只有一个”画笔”对象—使用该对象来绘图

var ctx = canvas.getContext(“2d”); 得到画布的画笔对象



(1)使用canvas绘制矩形(长方形)

矩形定位点在自己左上角

ctx.lineWidth = 1; 描边宽度(边线宽度)

ctx.fillStyle = “#999”; 填充样式

ctx.strokeStyle = “#000”; 描边样式

ctx.fillRect(x,y,w,h); 填充矩形

ctx.strokeRect(x,y,w,h); 描边矩形

ctx.clearRect(x,y,w,h); 清除矩形范围内所有图形

练习:左上角 右上角 左下角 右下角 居中

绘制5个矩形，大小100\*80 填充，颜色不同

练习2：在画布上描边一个可以左右移动100\*80矩形

提示:使用定时器，先清除画布上己有内容,重新绘制一个

上下移动 y不停修改

右下角45度 x,y不停修改

(2)使用canvas绘制文本

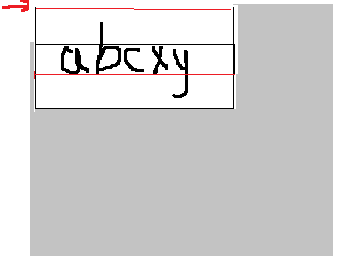
ctx.textBaseline = “alphabetic” 文本基线

ctx.font = “12px sans-serif”; 文本大小和字体

ctx.fillText(str,x,y); 填充一段文本

ctx.strokeText(str,x,y) 描边一段文本

ctx.measureText(str); 测量文本宽度



练习:左上角 右上角 左下角 右下角 居中

左右移动文字

作业1：商品分页显示 product\_list.html

HTML5 DAY03

做练习

HTML5 DAY04、

今天学习的内容

1:复习昨天知识重点

2:作业

2.1:学子商城分页

2.2:大鱼小鱼

2.3:第三方canvas库 two.js echart.js

<https://two.js.org>

http://echarts.baidu.com/index.html

3:html5新特性-SVG绘图

3.1:HTML5新特性—矩形

<rect width=”” height=”” x=”” y=”” fill=”” fill-opacity=””

stroke=”” storke-opacity=””></rect>

练习1:在SVG画布上正则央创建300\*30矩形柱子，

初始化淡红色(#faa)填充，和深以(#800)边框

都半透明,鼠标悬停，变为不透明

提示:修改属性setAttribute();

练习2：绘制两个柱子，初始高度1,使用定时器修改高度

一个高度 300停止，另一个变为150停止.

练习3:使用ajax从服务器端异步获取一段json数据

[{"label":"部门1","value":"190",,}]

获取四个部门数据，根据这些数据动态创建统计图

3.3:HTML5新特性—圆形

<circle r=”” cx=”” cy=”” ></circle>

练习1:左上角右上角 左下角右下角 居中绘制5个圆

练习2:在svg画布上随机绘制30个实心圆，大小随机，

位置随机，填充颜色随机，透明度随机,

点击某个圆之后，它慢慢变大，变淡,直至消失

从DOM树删除

作业1：学子商城分页

作业2: 仿网易音乐

作业3: 自学上传文件

HTML5 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:html5新技术--svg (重点)—椭圆

<ecllipse rx=”” ry=”” cx=”” cy=”” />

rx:水平半径

ry:垂直半径

3.2:html5新技术--svg (重点)—直线

<line x1=”” y1=”” x2=”” y2=”” stroke=”” stroke-width=””..></line>

3.3:html5新技术--svg (重点)—折线

<polyline points=”x,y x,y x,y ...” ></polyline>

3.4:html5新技术--svg (重点)—文本

<text text-size=”” fill=””...>文本内容</text>

3.5:html5新技术--svg (重点)—图像

<image xlink:href=”x.jpg” x y width height></image>

3.6:html5新技术--svg (重点)—渐变对象

<defs>

<linearGradient id=”r2” x1=”0%” y1=”0%” x2=”100%” y2=”0%”>

<stop offset=”0%” stop-color=”green”>

<stop offset=”50%” stop-color=”yellow”>

<stop offset=”100%” stop-color=”red”>

</defs>

<rect width=”400” height=”200” fill=”url(#r2)”

4.x:html5新特性--地理定位(实现需要联网)

Geolocation:地理定位,使用js获得当前浏览器所在地理坐标(经度，纬度，海拨，速度)数据，用于实现LBS应用(Location Base Service),如饿了么，高德导航...

获取地理定位类型

(1)浏览器自带对象geolocation[实现不了]

(2)百度地图,腾讯地图[准确度高]

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. 次选手机通信基站进行定位，定位精度在公里

PC浏览器如何获取定位信息

1. IP地理解析反向查找..

html5提供了一个新对象，用于获取当前浏览器定位信息

window.navigator.geolocation{};

getCurrentPosition:fn获取定位

百度地图:

(1)百度地图开发者 http://lbsyun.baidu.com/

(2)注册百度开发者帐户 手机

(3)创建一个网站:为网站申请访问密钥 AccessKey

网站名称[abc123]

[....]

(4) AccessKey

(5)开放示例

//加载百度API指定密钥

<script src="http://api.map.baidu.com/api?

v=2.0&ak=申请密钥">

</script>

var map = new BMap.Map("container");

// 创建地图实例

var point = new BMap.Point(116.404, 39.915);

// 创建点坐标

map.centerAndZoom(point, 15);

// 初始化地图，设置中心点坐标和地图级别

5.x:html5新特性—拖动API（拖动上传图片）

Drag & Drop :拖动和释入

HTML5为拖动行为提供7个事件，分为两组

拖动的源动(会动)可以触发3个事件

dragstart 拖动开始

drag 拖动中

dragend 拖动结束

拖动的目标对象(不动)可以触发4个事件

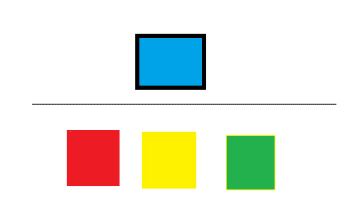
dragenter 拖动进入

dragover 拖动悬停在上方

dragleave 拖动离开

drop 拖动释放

练习:使用拖动事件的源对象和目标对象，完成**"拖动删除效果"**



提示:在刚开始拖动时(src.ondragstart) 记录被拖动源对id

释放(target.ondrop)根据此id找源对象。执行删除操作

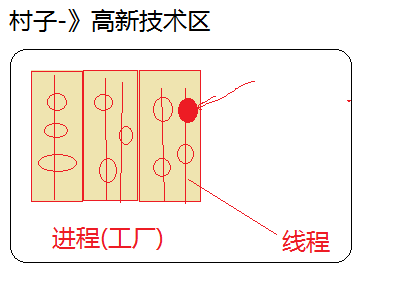
div.removeChild(c);

6.x:html5新特性之八—Web Worker 代码3行

程序:Program 可以被CPU执行的代码，存储在外存中

进程:Process 指程序被调用内存，分配执行空间，随时供CPU调用

线程:Thread 线程是进程内执行代码的基本单位



并发和并行:线程在中工作时并发状态运行.

并发:宏观上看同时执行，微观上看依次执行

并行:多个程序运行多核CPU

chrome浏览器中线程模型

一个chorme进程内部，多个线程可以“并发”向web服务器发起http请求，以获得所需资源

--资源请求线程

还有一个线程负责所有内容绘制和js代码执行

--UI主线程

作业1: 学子商城删除/更新/详细

HTML5 DAY06

今天学习的内容

1:复习昨天重点

2:作业

2.1:上传文件---(重点)

上传文件web项目通常使用功能:上传头像,邮件附件,上传商品图片,上传docx文档,喜玛拉雅，斗鱼...

上传原理:图

上传文件类型：指定任意类型/指定特定类型(jpg/png/gif)

上传文件方式:

(1)表单同步提交 --简单,用户感受差

(2)ajax上传 --复杂,用户感受好(拖动,预览图片)

(3)第三方js工具库—剪切

2.2:游戏(食物)

2.3:学子商城产品 删除 更新 详细

删除:有一定危险操作

1. DELETE FROM xz\_laptop WHERE lid = 3 极少
2. 给表添加一列 是否失效 expire ‘1’=>失效,’0’=>正常
3. 秘籍:软件项目有一个极特殊用户(超级管理员)

拥有软件项目中所有权利,

如果特殊的要求,超级管理员可以删除数据

如何操作

1. 添加列 xz\_laptop 产品表

expire 是否失效

类型 ENUM(‘1’,’0’)

ALTER TABLE xz\_laptop ADD expire ENUM('1','0') DEFAULT '0'

1. php product\_del.php

获取要删除产品编号

使用正则表达式验证

创建sql语句执行

操作系统日志添一条操作记录

2.4:商品更新

商品价格

<div>

apple mac air

[8300]

[确认]

</div>

3:今天的目标

3.1:web worker

HTML5 DAY07

今天学习的内容

1:复习昨天的项目

2:今天的目标:

学子商城 [更新/详细/搜索/添加]

作业1:学子商城 产品添加/用户分页

HTML5 DAY08

今天学习的内容

1:复习上周知识重点

2:今天的目标

2.1:项目[学子商城]

2.2:html5新特性

BOOT DAY01

今天学习内容

1:复习学子商城项目

2:今天学习的内容

2.1:html5 三个新特性—web Worker/web Storage/web Socket

2.2: html5 三个新特性—web Worker—理论多，代码少3行

程序:保存在外存中代码 1.html/1.css/1.js

进程:将程序调入内存中并且分配空间,在内存中运行程序称为 “进程”.

线程:处理进程内存，用于执行代码

并发:“宏观上看同时执行””微观上看依次交替执行”

并行:同时执行

chrom:中线程模型:请求资源线程—多个

运行js代码,渲染页面内容-1个

<button>按钮1</button>

<script src=”1.js”></script>

<button>按钮2</button>

解决方案:创建一个并发新线程，让它来执行耗时js任务

代码:

var w1 = new Worker(参数运行js地址);

#浏览器不允许Worker线程中操作(代码)不能有任何DOM和BOM对象.

原因:浏览器只允许UI线程操作DOM/BOM，若多个线程同时来操作DOM和BOM页面混乱.所以类似JQUERY也不能使用Worker中.

(\*)UI线程发送数据给Worker

woker(收)

onmessage = function(e){e.data}

ui(发)

var w6 = new Worker(“6.js”);

w6.postMessage(stringmsg);

(\*)Worker线程发送数据UI

UI(收数据)

var w6 = new Woker(“6.js”);

w6.onmessage = function(e){

e.data..

}

Woker(发数据)

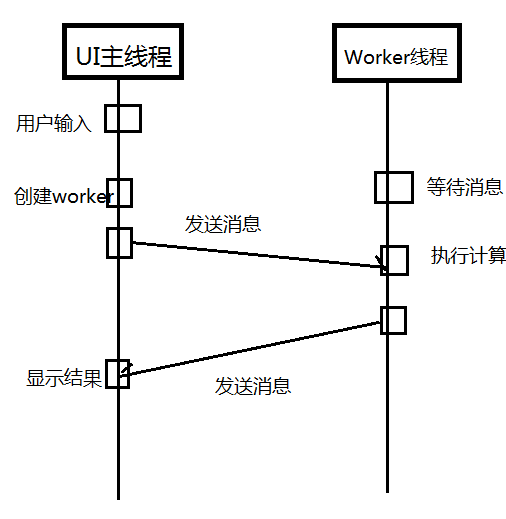
postMessage(执行结束);

练习:在HTML中有一个input,再添加按钮按钮"开始计算累加和"

,点击按钮时，创建一个Worker线程计算出用户输入数字

的累加和，下一个div显示计算结果input用户输入数字

button 开始计算累加和 div 显示计算结果



|  |
| --- |
| 项目中worker使用  1:只要js中有DOM和BOM代码不能使用Worker  2:Worker适合执行耗时JS任务, 复杂计算，加密和解密.. |

2.3: html5 三个新特性—web Storage

在浏览器中存储当前用户专用数据: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;操作复杂;不能超4KB

document.cookie="uid=10"; 保存一对数据cookie

document.cookie; 读取cookie数据

(2)Flash存储:依赖于Flash播放器

(3)H5 webstroage :不能超过8MB,操作简单

(4)IndexDB :可存大量数据，不是标准技术

|  |
| --- |
| Session:会话.“一个操作过程称为会话”  浏览器从打开某个网站的第一个页面开始（会话开始），中间可以打开多个页面(会话进行中)，直到关闭浏览器(会话结束),整个过程称为一次”会话”. |

webstorage 为浏览器提供二个对象

1. window.sessionStorage:类数组对象

作用:在同一个会话中所有页面之间共享数据,将数据保存sessionStorage中只限本次会话使用.

如:当前登录用户编号/权限列表/购物车信息/昵称

注意:安全性高数据不要存储sessionStorage

只要安全性高数据不要存储客户端

保存数据

sessionStorage[key] = val; //key数据名val 数据值

sessionStorage.setItem(key,val);

获取数据值

var val = sessionStorage[key];

var val = sessionStorage.getItem(key);

其它

sessionStorage.removeItem(key); 删除一对数据

sessionStorage.clear(); 清空所有数据

sessionStorage.length; 数据个数

sessionStorage.key(i); 获取第i个数据名

(2)window.localStorage

保存数据在磁盘上，可供此次会话以及及后续的会话中的页面共同使用,即使浏览器关闭也不会消失—永久存在：

作用:当前客户端对应所有会中共享数据,如:主机位置

保存数据

localStorage [key] = val; //key数据名val 数据值

localStorage.setItem(key,val);

获取数据值

var val = localStorage [key];

var val = localStorage.getItem(key);

其它

localStorage.removeItem(key); 删除一对数据

localStorage.clear(); 清空所有数据

localStorage.length; 数据个数

localStorage.key(i); 获取第i个数据名

练习:

创建index.html 首页

有一个下拉菜单,”请选择您喜欢的主题”,”蔚兰天空”,

”芭比公主”,”暗默主题”,各对应一个class名;

.blue{background:#ddf,color:#33a}

.pink{background:#fdf,color:#a3a}

.dark{background:#333,color:#eee}

option value=”pink”

document.body.className = pink

用户选中 select.onchange

创建 usercener.html 用户中心

打开此页面即可应用index.html中一样的主题样式,

启使重新启动浏览器，直接访问usercener.html仍是

之前选中的主题.

localStorage中若数据发生修改，会触发事件 onstorage事件，

可以监听此事件，实现监视localStorage数据修改.

2.4: html5 三个新特性—web Socket

HTTP协议:属于”请求-响应”，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，在些场景中不适合”实时走势图”

解决方案:”长轮询-心跳请求”—定时器+ajax

websocket协议:属于”广播-接收”模块，客户端连接到服务器不再断开，永久连接双方随时向对方发送消息.

ws:服务器端:php/java/node.js

ws:客户端:php/java/html5 提供对象

使用html5创建ws协议客户端

//连接ws服务器

var socket = new WebSocket(“ws://127.0.0.1:9001”);

//向服务器发送数据

socket.send(stringMes);

//接收服务器数据

socket.onmessage = function(e){e.data}

//断开到服务器连接

socket.close();

3.1:Bootstrap [css 与js框架 响应式布局]—(栅格布局系统)

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) 可以通话，短信，上网，视频 - HTML

4G:

5G:...

3.3:Responsive Web Page:响应式网页/自适应的网页2010年提出,

一个网页，会自动根据用户浏览设备不同，自动必变布局,

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;

(2)可以改变尺寸图片文字 font-size:1em;

响应式布局不足:代码更加复杂，不适合大型网页

3.5:如果测试网页

(1)使用真实物理设备

(2)使用第三方模拟器软件

(3)使用chrome 内置设备模拟器

3.5:如何编写响应式网页

(1)声明viewport元标签

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

|  |
| --- |
| viewport 元标签  早期3G手机为浏览器大尺寸网页，只能强行页面缩小,  导致图片/文字/链接  IOS提出”视口”概念:虚拟窗口，大小与设备相同，如果  浏览大尺雨网页，滑动.  Android 也借鉴该概念 |

#width=device-width 视口宽度=设备宽度

# initial-scale=1 表示初始化视频大小1原始大小

#user-scalable=no 用户是不顺可以调整视频大小

(2)使用流式布局

实现方法:float:left

(3)所有容器使用相对尺寸，不用绝对尺寸

.container{

/\*width:500px;\*/

width:50%;

}

(4)所有文字使用相对尺寸，不用绝对尺寸

body{

/\*font-size:12px;\*/

font-size:2em;

}

(5)所有图片使用相对尺寸，不用绝对尺寸

img{

width:120px;

width:25%;

}

(6)（最重要原则）使用CSS3 Media Query技术

3.6: CSS3 Media Query技术

Media:指浏览网页的设备,如screen(pc/pad/phone),print,tv,tty.

Media Query:查询当前浏览网页设备类型，以及特性(对比度，尺尺，手持方向等),而选择性执行某些css代码，而忽略另外一些.

媒体查询使用方法两种

1. 根据媒体查询结果执行不同的css文件

<link media=”screen” href=”1.css”/>

<link media=”print” href=”2.css”/>

1. 根据媒体查询结果执行css片段中不同部分

作业1：学子商城,产品添加

BOOT DAY02

今天学习的内容

1:复习昨天知识重点

2:游戏(食物/大鱼)

游戏玩法:大鱼后面跟小鱼在游动,大鱼负责吃食物,小鱼跟大鱼

大鱼吃过食物以后，喂小鱼

小鱼长时间不吃食物颜色红,黄,浅黄,白

游戏角色:

背景:大图片

海葵:canvas中路径实现

食物:二张图片,蓝色黄色

#如何创建食物

(1)创建文件 game/js/fruit.js

(2)修改html 加载fruit.js

(3)修改game/js/fruit.js 添加二个方法 类/init/draw

(4)修改game/js/main.js 添加食物对象并且调用对应方法

#通用小技巧

平滑由小变大过程,向上浮动平滑

deltaTime 二帧绘制时间间隔

3:今天新的目标

3.1:自定义响应式布局(重点)

(1)声明viewport 视频

(2)流式布局

(3)所有容器.文字.图片:相对尺寸不同绝对尺雨

(4)使用CSS3 Media Query

3.2: CSS3 Media Query查询方式

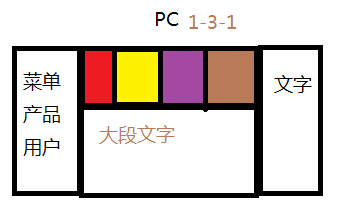
(1)根据媒体查询结果执行不同css文件(少)

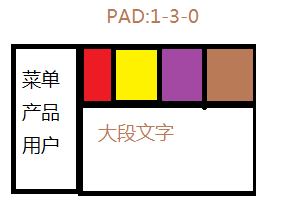
(2)根据媒体查询结果执行不同css片段(多)

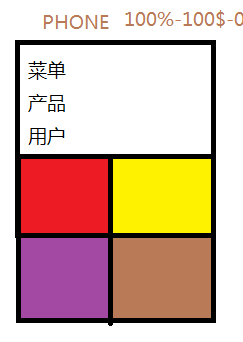
@media screen and (min-width:768px) and (max-width:991px){

选择器{样式设定}

}







3.4:Bootstrap 全局样式

|  |
| --- |
| bootstrap: 起步/引导程序 |

2.5:Twitter Bootstrap 框架

#官方网站 http://www.bootcss.com/

Bootstrap是一个HTML/CSS/JS 框架,用于开发响应式布局，移动设备优先项目.

(1)起步

(2)全局CSS样式(重要—{栅格布局})

(3)组件

(4)js插件(图片轮播)

(5)定制—自定义bootstarp 样式

提示:Bootstrap 知识点不难,1000+单词记忆

2.6:起步

下载 css fonts js

基本模板

|  |
| --- |
| <html lang="zh-CN">  language:指定当前网页所使用自然语言,如zh-CN zh-hk en-US ja  作用两个:为浏览器的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=edge">  扩展HTTP响应头消息  指定IE浏览器的兼容性-仅IE浏览器支持可以识别  网景<==>微软 IE  IE 6 内核6  IE 7 内核6/7  IE 8 内核6/7/8  IE 9 内核6/7/8/9  IE 10 内核6/7/8/9/10  IE 11内核6/7/8/9/10/11 |
| html5shiv.js  第三方编写的js文件，让老IE兼容html5新标签... header.. |
| respond.js  第三方编写的js文件,让老IE兼容 CSS3 Media Query |

2.7:Bootstrap 全局样式—按钮

.btn 按钮的基础样式

.btn-default 白底黑字的按钮

.btn-danger/warning/success/info/primary 五种颜色的按钮

.btn-lg/sm/xs 按钮四种大小

.btn-block 块级按钮

2.8:Bootstrap 全局样式—图片

.img-rounded 圆角图片

.img-circle 圆形图片

.img-thumbnail 缩略图

.img-responsive 响应式图片

2.9:Bootstrap 全局样式—文本

.text-danger/success/warning/info/primary 文字颜色

.bg-danger/success/warning/info/primary 背景颜色

.text-left/center/right/justify 对齐方式

.text-uppercase/lowercase/capitalize

2.10:Bootstrap 全局样式—列表

.list-unstyle 去除提示符号

.list-inline 行内列表

作业1:家里WIFI,使用手机访问自己PC电脑APACHE

192.168.10.10

作业2:依据图片响应式网页

BOOT DAY03

今天学习的内容

1:复习昨天知识重点(HTTP/HTTPS协议)

2:大鱼

大鱼组件部分:[眼睛/身体/尾巴]

眼睛:睁(时间长)闭(时间短)

身体:8张图片

尾巴:8张图片

小技巧：多长时间切换到下一张图片

var index = 0; 1 2 3 ... 7 图片数组下标

var start = 0; 计时开始

var end = 1000; 结束时间(切换下一个图片)

#驱向移动

#碰撞检测

3:今天的目标

3.1:Bootstrap 全局样式-(屏幕分类)

(1)大型PC屏幕 lg w>=1200px

(2)中等PC屏幕 md 1200px>w>992px

(3)小型PAD屏幕 sm 992px>w>768px

(4)超小PHONE屏幕 xs 768px>x

3.2:Bootstrap 全局样式-表格

.table

.table-bordered 带边框的表格

.table-striped 隔行变色

.table-hover 带悬停变色

.table-responsive 响应式表[添加特殊要求]

|  |
| --- |
| Bootlint 是Bootstrap官方提供HTML检测工具，检测出网页中常见的HTML/CSS错误，还能检测class层级结构错误! |

3.3:Bootstrap 全局样式-栅格(重点)

网页布局可用方法:

1. TABLE布局

好处：简单易控制

不足: 语义错误,页面渲染效率低

1. 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)Bootstrap 栅格

好处:相对简单易于控制，渲染效率高，支持响应式

不足:页面简单

使用方法:

1. 最外层必须使用容器

div.container和div.container-fluid

1. 容器里可以放置任何内容，若想使用栅格，必须div.row

.container > .row [行]

1. 一个.row中不能放置其他内容,只能放置 .col,列中可以放置其它内容

.container > .row > .col

1. Bootstrap中行默认12均等分，每个列必须指定行中占比
2. 栅格系统针对不同屏幕提供不同列

.col-lg-1/2/3.../12

.col-md-1/2/3../12

.col-sm-1/2./3../12

.col-xs-1/2./...12

1. 可以使用”列偏移”实现指定列及其后的列向右偏移的效果.

如果不足12列居右对齐

.col-lg-offset-1/2...12

.col-md-offset-1/2...12

.col-sm-offset-1/2...12

.col-xs-offset-1/2...12

1. 需要注意不同屏幕下列的适全性问题

.col-xs-\* 适用于xs/sm/md/lg 屏幕

.col-sm-\* 适用于 sm/md/lg 屏幕

.col-md-\* 适用于 md/lg 屏幕

.col-lg-\* 适用于 lg 屏幕

1. 一个列可以指定在不同屏幕下的宽度占比

div.col-xs-12/.col-sm-9/.col-md-6

div.col-xs-12/col-md-6

常见错误: div.col-xs-6/.col-md-6

1. 一个列可以指定特定的屏幕隐藏

.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

.hidden

作业1:栅格布局系统 上二节课四张图片

作业2:小鱼

BOOT DAY04

今天学习的内容

1:复习昨天重点内容

2:大鱼碰撞检测/小鱼/碰撞检测

2.1:大鱼吃食物(果实)

问题:碰撞检测

通常解决问题:计算二个元素之间距离

1大鱼 15食物[30其中15个食物活动]-->循环

0-14 <-->大鱼 <=30 个像素 吃了食物

问题：食物被大鱼吃食物

alive[i]=false;

2.2:小鱼

位置,角度,身体,尾巴,眼睛

3:今天目标

3.1:Bootstrap 栅格式系统

3.2:Bootstrap 表单

(1)默认表单

<form>

<div class=”form-group”>

<label class=”control-label”>用户名</label>

<input type=”text” class=”form-control”/>

<span class=”help-block”>用户名格式:6~12位</span>

</div>

</form>

(2)行内表单

<form>

<div class=”form-inline”>

<input type=”text” class=”form-control”/>

<span class=”sr-only”>用户名格式:6~12位</span>

</div>

</form>

(3)水平表单

水表表单=表单+栅格系统

|  |  |  |
| --- | --- | --- |
|  | 默认栅格系统 | 水平表单 |
| 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 class...

</div>

</div>

</form>

3.3:Bootstrap 组件—(下拉菜单:字体图标:导航条-重点)

下拉菜单三级结构

<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>

3.4: Bootstrap 组件—(字体图标)

web项目中常用图标字体

(1)FontAwesome – 675 个免费图标

(2)Glyphicons - 800 个收费图标

服务器字体使用

(1)web 服务器项目目录下必需有字体文件

(2)css中声明一个新的字体文件，并且指定下载位置

@font-face {

font-family: 'Glyphicons Halflings';

src: url('../fonts/glyphicons-halflings-regular.eot');

src: url('../fonts/glyphicons-halflings-regular.eot?#iefix')

}

(3)声明一个基础class 使用字体

.glyphicon{

font-family: 'Glyphicons Halflings';

}

(4)在html页面中输入图标对应编码

练习:

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,放大镜,删除(垃圾箱),加号,

减号,叉号,前进,后退,播放,暂停,停止,快进

3.5: Bootstrap 组件—(进度条)

.progress > .progress-bar

3.6: 面包屑 .bread-crum /徽章/巨幕/水井

4.1:Bootstrap第四部分:jquery插件---折叠(手风琴)

<a data-toggle=”collapse” href=”my-target1”>触发元素</a>

<button data-toggle=”collapse” data-target=”my-target1”>触发元素</button>

<div id=”my-target1” class=”collapse in”>展开收起元素</div>

4.2: Bootstrap第四部分:jquery插件---响应式导航条

作业1:参数intel 首页使用bootstrap实现

BOOT DAY05

练习

BOOT NodeJS DAY01

今天学习的内容

1:复习上周重点

2:弹幕

1:html

<canvas id=”can1”/> 显示弹幕文字

<canvas id=”can2”/> 视频画在画布上

(1)视频按照图片绘制画布上 ctx2.drawImage();

(2)创建定时器[]/循环绘制视频

(3)视频元素动态创建追加 body最后子元素并且隐藏

2:js

3:目录结构

index.html

js/ video.js 完成:视频显示

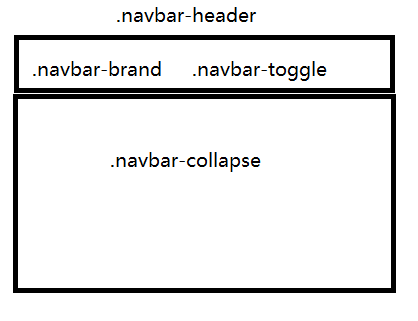
data.js 完成:弹幕

公共函数库文件

res/1.mp4 视频文件

3:今天的目标

3.1:Bootstrap js插件---(导航条)



.navbar-toggle 媒体查询:屏幕宽度大于768px 就隐藏

.navbar-collapse 媒体查询:屏幕宽度大于768px 绝对定位在

.navbar-header内;否则相对定位.navbar-header

#练习1:导航条

#练习2:大小小，小小大，页脚

3.2:Bootstrap 定制—less(重点)

3.3:样式语言分类

(1)静态样式语言

CSS可以直接被浏览器解析，但是作为一门语言，CSS并不完整，

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/SCSS/Styles/Less 在CSS的基础之上添加动态编程语言，必需特性，如：数据类型,变量,运算,函数,继承,提高了样式可维护性.

需要注意:浏览器只能识别CSS,所有动态样式语言代码必需转换CSS,转换过程(编译)

3.4:less样式语言

http://lesscss.cn

Less是一门CSS预处理语言—它扩展了CSS语言，增加变量，函数，等新特性.使CSS更易于维护和扩展.

3.5:less两种使用方法

(1)在客户端使用—学习(效率差)

编写一个x.less

在HTML中引入x.less文件，同时再引入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

less编译为css

会减慢客户端样式的呈现速度

(2)在服务器端使用-Less(必需掌握)

编写x.less

在开发电脑安装less编译环境(Nodejs/lessc),执行x.less编译x.css

再编写html文件，引入编译结师法x.css

客户端发出请求html/下载css

3.6:LESS语法

(1)Less支持所有CSS语法

(2)Less支持多行/单行注释

/\*\*/ 多行注释

// 单行注释—推荐使用(注释内容不会编译css文件)

(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 支持变量和常量算术运算

+ - \* / %

(5)less支持一个选择器中”混入”另一个选择器的样式

选择器1{...}

选择器2{.. 选择器1}

(6)less在样式混入时可以指定参数

选择器1(@参数1,@参数2..){}

选择器2{选择器1(值1,值2...)}

(7)less支持样式嵌套

选择器1{

选择器2{

}

}

上述代码会被编译为:

选择器1{}

选择器 2{}

NodeJS DAY02

今天学习的内容

1:复习昨天重点

2:视频项目

<video src=”” autoplay loop controls muted poster=””

preload=””></video>

#video 弱 [不能对视频加密/视频丢失]

#video 弱 直播不行

#强 微信 小程序 公众号 [pc/pad/ios/android]

2.1:同层播放 (苹果手机)

当点击视频播放,IOS手机默认行为,自动将视频弹出在一个独立，

窗口播放.(二个窗口)

***解决问题:同一层窗口播放视频***

在<video>标签中添加二个属性

webkit-playsinline="true" (IOS)让频视在小窗口内播放，

能不同跳出全屏播放

playsinline="true" (IOS)微信浏览器支持小窗口内播放

x5-video-player-type="h5" (Android)同层播放

示例:

<video webkit-playsinline="true" playsinline="true"

x5-video-player-type="h5" ></video>

***解决问题*: 全屏设置如果不设置如下属性，横屏会错位**

x5-video-player-fullscreen="true"

全屏设置对过监听resize事件实现 (Android )

x5-video-orientation="portraint">

2.2:video 专用样式

object-fit:选项

1. fill默认 失真
2. contain 包含(留白)
3. conver 覆盖(调整视频比例)

2.3:video 事件

(1)canplaythought 可以播放视频，视频全部加载完成

#iphone 点击播放

#android 可以加完成触发事件

(2)ended 播放结束

作业1:

项目: fly(h5视频项目(故事)) 微信公众号、小程序

玩:将fly保存你电脑->无线上网

你手机->无线上网

电脑->火狐浏览器-><http://192.160.10.1/fly/index.html>

火狐生成二维码

手机->微信扫描

3:今天的目标

3.1:Bootstrap 定制—(LESS 重点)

3.4:less样式语言

http://lesscss.cn

Less是一门CSS预处理语言—它扩展了CSS语言，增加变量，函数，等新特性.使CSS更易于维护和扩展.

3.5:less两种使用方法

(1)在客户端使用—学习(效率差)

编写一个x.less

在HTML中引入x.less文件，同时再引入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

less编译为css

会减慢客户端样式的呈现速度

(2)在服务器端使用-Less(必需掌握)

编写x.less

在开发电脑安装less编译环境(Nodejs/lessc),执行x.less编译x.css

再编写html文件，引入编译x.css

客户端发出请求html/下载css

3.6:LESS语法

(1)Less支持所有CSS语法

(2)Less支持多行/单行注释

/\*\*/ 多行注释

// 单行注释—推荐使用(注释内容不会编译css文件)

(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 支持变量和常量算术运算

+ - \* / %

(5)less支持一个选择器中”混入”另一个选择器的样式

选择器1{...}

选择器2{.. 选择器1}

(6)less在样式混入时可以指定参数

选择器1(@参数1,@参数2..){}

选择器2{选择器1(值1,值2...)}

(7)less支持样式嵌套(\*\*\*)

选择器1{

选择器2{

}

}

上述代码会被编译为:

选择器1{}

选择器 2{}

3.7:通过修改Bootstrap的less源文件实现定制

达到三个目标

(1)瘦身:删除不需要样式

(2)粗粒度定制

只需要修改variables.less中定的变量值即可

(3)细粒度定制 针对某个组件细节修改

修改组件对应less文件. dropdown.less

3.8:如何实现定制

(1)下载并且安装nodejs – (js解析器)

在命令行 node –v 可以看到版本信息

(2)下载less编译程序

e:\node\_modules\less\bin\lessc

示例:

e:\node\_modules\less\bin\lessc x.less x.css

3.9 定制Bootstrap

(1) 查找software/bootstrap/less

(2) e:\node\_modules\less\bin\lessc less/bootstrap.less bootstrap.css

4.0:Nodejs

面试题:用户在浏览器中输入 [www.taobao.com](http://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DNS服务器，把域名转换IP地址

(2)浏览器发起HTTP请求

(3)淘宝服务器接收请求消息，查找指定资源[默认index.html]

可能访问数据库，构建并且返回HTTP响消息

(4)浏览器接收响应

(5)浏览器缓存接收响应消息，并解析和渲染内容

4.1:静态网页和动态网页

静态:网页内容任何人在任何时间访问都是不变的

HTML/CSS/JS/视频/音频

动态:网页内容不同人在不同时间访问可能是不同的

JSP/PHP/ASP.NET/Node.JS

JSP = HTML+JAVA 功能强大可靠，适合大型企业级项目

PHP = HTML+PHP 简单易用,适合互联网项目

ASP.NET = HTML+C# 易用,适合windows 平台

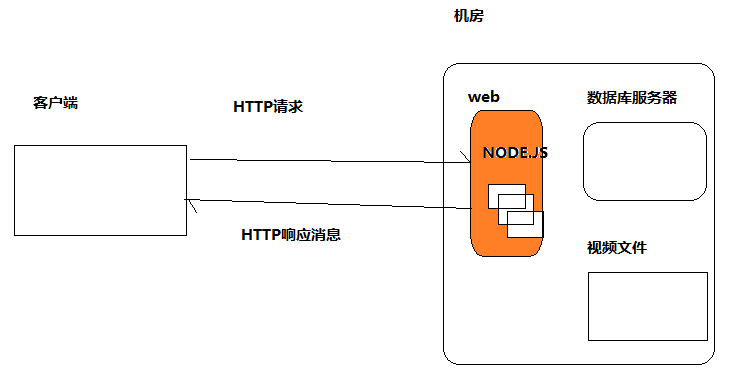
30w==2w+0.5w+20w+10w

Node.JS = HTML+”JS” 性能好,适合服务器端IO密集型项目,

不适合CPU密集型项目

CPU密集项目:滴滴打车,天气预报,大数据分析,大数据计算..

IO密集项目:查询,修改,删除,查询,查询,修改



4.2:Nodejs

Node.js不是JS,是一种软件开发平台，它的竞争对象JSP/PHP/ASP.NET,历史第一次有一种语言可以通吃前后端.

网站：阿里云镜像

https://npm.taobao.org/mirrors/node/

版本:0.12(16年初) 4.x(16年中) 6.x(16年底)

LTS:Long Term Support

Current:最新版本

4.3:Node运行的两种方式

(1)交互模块---简单了解(测试新功能)

REPL:输入一行代码执行一行

注意:交互模自带输出功能,不必一定要 console.log()

node 回车 进入交互模式

.exit 退出

(2)脚本模块—正式项目中使用

把要执行所有语句编写一个文本文件中(后缀任意，没有),一次性提交node解析器执行.

node 完整路径/x.js 回车

提交:只要安装成node.js,重启webstorm,ws可以自动发现node.exe解析，新的项目和文件编写一定UTF-8

练习:创建04\_node.js ,打印九九乘法表，以脚本模式运行在ws中

故障:运行 nodejs 进程不能正常工作

原因:创建文件夹 node。js/03\_node.js

建议:所有目录名称,不要有中文,不要有空格

4.4:面试题:如何自学一门新语言

(1)了解背景 – 百度百科（事件驱动/非阻塞IO）

(2)搭建开发环境,编写HelloWorld

(3)数据类型—重点

|  |
| --- |
| 前端JS中数据类型   1. 基本/原生/值类型   string/number/boolean/null/undefined  (2)引用类型  ES对象类型:Error/Data/Math/Array/RegExp/Object/....  BOM:window;docuemnt,location,event,screen,....  DOM:Element,Attribute,.....  用户自定义对象类型:{} |
| 前端NODE JS中数据类型  (1) 基本/原生/值类型  string/number/boolean/null/undefined  (2)引用类型  ES对象类型:Error/Data/Math/Array/RegExp/Object/....  用户自定义对象类型:{}  Node.JS 第三方对象:十万多 https://www.npmjs.com/ |

(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

位算运算符

赋值运算符

特殊运算符 . instanceof typeof

(6)逻辑结构

循环结构:while do ..shile for(;;) for(..in..) for(..of..)

练习:创建05.js 声明一个保存5个学生成绩的数组,

使用三种for循环依次打印出每个成绩

选择结构:if..else switch..case

练习:创建一个变量 var path = “/index”;

使用二种选择结构,判断path的值为哪一种

(/index;/search;/login) 调用不同执行函数

(7)通用小程序

练习:打印出100以内所有质数

(8)函数和对象

(9)常用的组件，第三方工具库,框架

(10)实际小项目

|  |
| --- |
| Modal: 模态框  Model: 模型  Module:模块 |

4.5:nodejs中特有概念—模块---(重点)

一个项目中多个”模块”,订单模块，用户模块，支付模块....

node.js按照功能不是，可以把函数，对象分别保存不同文件或目录下，这些目录和文件在node.js称为一个“模块”

04\_node.js 05.js

##注意:Node.JS每个一个模块都是一个独立构造函数，

Node.JS解析器自动为每一个.js文件添加如下代码.

(function(exports,require,module,\_\_filename,\_\_dirname){

exports:{} 用于声明向外部导出自己成员

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;

var j = 11;

console.log(i+j);

});

|  |
| --- |
| Node.js 模块中exports和module.exports对象区别是什么?  二者都可以用于向外导出自己内部成员，  但:实际向外导出成员 module.exports  exports 对应一个引用 |

每个模块都可以使用自己require();引入另一个模块—底层本质是创建指定模块一个对象实例.

08\_data.js 声明二个变量一个函数 ->导出一个变量一个函数

09\_app.js 引入

今日作业2:

仿写node.js提供一个模块:url

创建模块MyUrl.,向外导出一个方法

resolve(url);该方法接收参数如http://www.jd.com:80/admin

返回对象:

{

protocol:"http",

server:"www.jd.com",

port:80,

path:"/admin"

}

NodeJS day03

今天学习的内容

1:复习昨天重点

2:作业

3:今天的目标

练习1:创建文件模块 circle.js 其中常量PI,

声明二个方法 getSize(r) 求面积

getPerimiter(r); 求周长

创建主模块 02\_app.js 调用其公开二个方法

练习2:创建文件模块 arrayutils.js

包含一个sum(arr); 返回数组值和,

包含一个avg(arr); 返回数组值平均值

创建主模块 03\_app.js 调用其公开二个方法

3.1:nodejs 模块-目录模块

自定义模块的两种形式

(1)**文件模块**

创建一个JS文件,如m3.js,导出其公开数据，

其它模块可以require("./m3");引入

(2)**目录模块**

方式一:创建一个目录,假设m4,其中创建index.js文件，导

出需要公开数据。其它模块引入

var m = require(“./m4”);

方式二:创建一个目录,假设m5,其中创建5.js文件，导出需要

公开数据数据,创建package.json文件 main指定启动

文件 5.js 其它模块引入

var m = require(“./m5”);

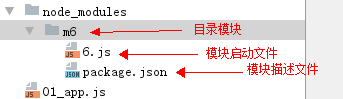
方式三:创建一个目录,必须名为 node\_modules

,其中再创建目录模块,假设 m6 其中创建 package.json

文件，其中声明 main属性指定默认执行启动js,如6.js，

其中导出需要公共数据，其它模块引入

require(“m6”);



|  |  |  |
| --- | --- | --- |
|  | 文件模块 | 目录模块(方式三) |
| 程序结构 | m1.js | node\_modules  m2  package.json  2.js |
| 模块名称 | m1 | m2 |
| 导出数据 | module.exports | module.exports |
| 导入模块 | var m = require(“./m1”); | var m = require(“m2”); |

练习1:创建二个目录模块,circle,rectangle都对外公开两个方法size(); perimeter();返回指定图形面积和周长

最后在最外层主模块中引入上述二个模块四个方法

circle.size(r); circle.perimeter(r);

rectangle.size(w,h);

rectangle.perimeter(w,h);

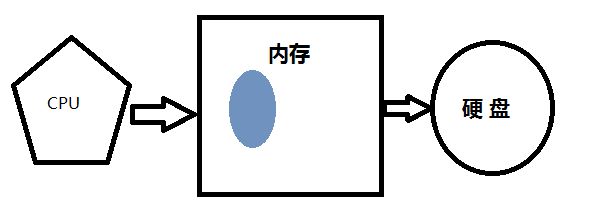
3.2:nodejs 原生模块-->(重点)

Node.js 解析引擎自带

3.3: Node.js 原生模块—Buffer(缓冲区)

Buffer:缓冲区,本质上是一块内存区域，用于暂存以后要用到数据

(数字,字符串,二进制图片/音频/视频),该区域称为“缓存”



#分配置一个指定大小缓冲区 1024字节

建议：缓冲区大小不超过 512KB 512\*1024

var buff1 = Buffer.alloc(1024);

#使用一个数组创建缓冲区

var buff2 = Buffer.from([1,2,3]);

#使用一个字符串创建缓冲区

var buff4 = Buffer.from(“abc”);

#将一个缓冲区内容转换字符串{如果缓冲区数据是字符串}

var str = buff4.toString();

3.4: Node.js 原生模块—QueryString

querystring模块用于处理HTTP请求URL中查询字符串

var obj = qs.parse(str); 把查询字符串解析js对象

var str = qs.stringify(obj) 把js对象转换为查询字符串

3.5: Node.js 原生模块—URL

url模块用于解析一个HTTP请求地址，获取其中各个部分

var obj = url.parse(str); 把一个URL地址为js对象

var obj = url.parse(str,true); 功能同上，并且把其中查询字符串转换

js obj

3.6: Node.js 原生模块—fs—(重点)

文件:在node.js 我们系统文件或者目录统称为文件

fs 模块提供对文件和目录进行，增删改查，读定文件内容

1. 同步读写文件(阻塞)

#同步读取文件内容 var data = fs.readFileSync(fileName);

#同步向文件写内容 fs.writeFileSync(fileName,data);

注意:(1)如果当前写入文件并不存在，自动创建一个

a.txt

(2)如果当前写入目录并不存在，报错

d:/abc/a.txt

(3)如果当前文件己经存在内容，清空

#同步向文件追加内容

fs.appendFileSync(fileName,data);

练习:使用上述方法，实现文件复制操作

./public/data.log-->backup\_data.log

1. 异步读写文件(非阻塞)

#异步读取数据 fs.readFile(fileName,function(err,data){ });

函数:文件读取完成后调用

err:读取文件不存在，权限不足,

自动创建错误对象

#异步写数据 fs.writeFile(fileName,data,function(err){})

函数:文件写结束调用

#异步追加数据 fs.appendFile(fileName,data,function(err){})

函数:文件追加结束

练习:使用非阻塞方法,实现public/data.txt文件,复制backup\_data.txt

3.7: Node.js 原生模块--http

HTTP 模块可用于编写基于HTTP协议客户端程序(即浏览器),也可以编写一个基于HTTP协议服务器(APACHE)

APACHE[1000并发量] nginx[28000] IIS[微软]

用http模块编写一个web服务器

(1)创建服务器对象 var server = http.createServer();

(2)绑定监听端口 1-65535 server.listen(3000);

(3)接收客户端请求， server.on(“request”,(req,res)=>{..})

(4)并且响应客户消息 res.end(str...);

req:请求对象:保存客户端请求消息

req.url 请求url地址

res:响应对象:服务器发送数据客户端,修改响应格式

作业1:使用node.js创建服务器，根据客户端的请求地址不同

输出不同的html页面内容

http://127.0.0.1:3000/login.html

http://127.0.0.1:3000/admin.html

若请求资源不存在，则返回 404.html

NodeJS DAY04

**今天学习内容**

**1:复习昨天的内容**

**模块:**

**A:自定义模块**

**文件模块:m.js app.js**

**目录模块:node\_modules/m6/**

**Package.json main:”./6.js”**

**Var m = require(“m6”);**

**B:原生:Buffer/url/querystring/fs/http**

**同步(阻塞)**

**var data = fs.readFileSync(fileName);**

**fs.writeFileSync(filename,str);**

**fs.appendFileSync(filename,str);**

**异步(非阻塞)**

**fs.readFile(filename,(err,data)=>{});**

**fs.writeFile(filename,str,(err)=>{});**

**fs.appendFile(filename,str,(err)=>{});**

**C:第三方**

**同步和异步:程序工作方式**

**同步：程序1另程序2,一个等待一个工作**

**异步: 程序1另程序2,一个工作一个工作**

**2:作业**

**3:今天的目标**

**3.1:node.js 原生模块 http**

**三个常用对象:**

1. **server 代表服务器对象**
2. **req (request)请求对象，保存客户端很多信息**

**req.url 客户请求地址**

**req.method 客户请求方式**

**req.httpVersion HTTP版本**

**req.headers 请求头**

1. **res (response)响应对象，保存服务器端向客户发送数据**

**res.statusCode = 200;**

**res.setHeader(); 设置响应头**

**res.write(); 向客户端输入响应消息主体**

**res.end(); 通知客户端，响应消息结束(只能一次)**

**常见错误:程序端口被占用**

**listen EADDRINUSE :::3000**

**解决：查找前面程序停止即可**

**练习1:创建动态web服务器，接收客户请求，若客户端请求地址为**

**/register.html 则向客户端输出 register.html网页**

**<form action="/register.do">**

**uname []**

**upwd []**

**[注册]**

**</form>**

**练习2:如果用户请求地址 /register.do 则解析url中查询字符串,**

**把客户端提交uname和upwd追加写入文件中 user.log ,并且向客户**

**端输出 "注册成功"**

**3.2:node.js 第三方模块 (mysql/express)**

**3.3:SQL语句的分类**

**DDL:数据定义语句 CREATE/DROP/ALTER…**

**DCL:数据控制语句 GRANT…**

**DML:操作操作语句 INSERT/UPDATE/DELETE**

**DQL:查询语句 SELECT**

**3.4:使用node.js访问mysql服务器**

**为了精简NODE.JS解析器，官方没有提供访问任何数据库相关模块,**

**必须使用工具 npm 下载第三方模块,在**[**www.npmjs.org**](http://www.npmjs.org) **搜索关键字**

**Mysql 可以得到相关模块，每个模块使用说明.**

**示例:下载**

**npm install mysql 简写 npm i mysql**

**3.5:mysql模块使用步骤**

**(1)复制模块 node\_modules day03根目录下**

**(2)加载mysql模块 const mysql = require(“mysql”);**

**(3)创建到数据库服务器连接**

**var conn = mysql.createConnection({})**

**(4)发送sql语句并且获取服务器返回结果**

**conn.query(sql,(err,result)=>{});**

**(5)关闭连接 conn.end();**

**练习1:删除xz库xz\_user表操作 uid=1**

**DELETE FROM xz\_user WHERE uid = ${uid}**

**练习2:使用http模块和mysql模块**

**创建动态web服务器，接收如下请求url**

**/login.html 服务器向客户端返回 login.html**

**/login.do 服务器接收客户提交 uname/upwd ?**

**在数据查询 xz\_user 返回 "登录成功"**

**"用户名或密码有误"**

**/userlist.do 服务器查询数据库所有用户信息,在一个**

**table中向客户输出**

**/register.html 用户注册表单[uid,uname,upwd,age]**

**/register.do 用户注册**

**作业1:学子商城:后台php🡪nodejs**

**后台管理员登录写完.[数据库表不动]**

NodeJS DAY05

今天学习的内容

1:复习昨天知识重点

node.js

什么是node.js:运行服务器端开发平台

模块:

1. 自定义模块

a:文件模块 m.js app.js var m = require(“./m”);

b:目录模块 node\_modules/m6/package.json/6.js

app.js var m = require(“m6”);

1. 原生模块

buffer/fs/http/url/querystring

var obj = url.parse(str,true);

fs.readFile(fileName,(err,data)=>{})

fs.writeFile(fileName,data,(err)=>{})

fs.appendFile(fileName,data,(err)=>{})

1. 第三方模块

mysql

var conn = mysql.createConntion({});

var sql = “SELECT id FROM xz WHERE uid = ?”;

占位符:安全【网络攻击手段-SQL注入】

conn.query(sql,[10],(err,result)=>{

if(err)throw err;

conn.end();

});

2:作业

3:今天的目标

3.1: nodejs第三方模块express (重点)

下载express 模块

npm i express http高级模块

npm i body-parser 处理post请求

npm i cookie-parser 处理cookie

npm i express-session 处理session

npm i cors 跨域

3.2:express模块

使用官方提供HTTP模块可以创建一个web服务器应用，但此模块

非常底层，要处理各种情形，比较繁琐。推荐使用HTTP高层模块，

express—第三方模块，是一个基于node.js http模块的高层模块，**简化**

服务器端开发。

1. 下载模块 npm i express
2. 将模块内容复制项目中 day04/node\_modules
3. 加载相应模块 http/express
4. 创建express对象，再创建服务器对象，并且绑定监听端口

var app = express();

var server = http.createServer(app);

server.listen(3000);

常规语法规则:

app.get(请求地址,回调函数);

#接收客户请求 GET /index.html

app.get(“/index.html”,(req,res)=>{});

#接收客户请求 POST /index.html

app.post(“/index.html”,(req,res)=>{});

示例:

以前:login.php 参数 数据库 json

$.get(“login.php”,…..);

现在:app.get(“/login.do”,(req,res)=>{

参数 数据库 json

});

$.get(“/login.do”…..);

3.1:express req和res对象

res.sendFile(\_\_dirname+文件绝对路径); =fs.read+res.write+res.end

res.send(向客户端返回html字符串响应);

res.json(obj); =setHead()+JSON.parse()+res.write()+res.end()

3.2:express GET 请求 接收参数二种方法

**#GET /login.do?uid=10&uname=tom**

app.get(“/login.do”,(req,res)=>{

//express为每一个req对象添加属性query

req.query.uid;

req.query.uname;

});

**#GET /login.do/12/tom**

app.get(“/login.do/:uid/:uname”,(req,res)=>{

//express 为每个req对象添加一个params 属性

req.params.uid

req.params.uname

});

3.4:为了项目提高效率🡪创建连接池

(1)创建连接池

var pool = mysql.createPool({…}); host;user;password;

connectionLimit:5 连接池中活动连接5个

建议范围：5~25

**1亿(PV/1day)= 8小时=28800秒=10-100=34/s**

**100000000/28800/100=**

(2)所有应用程序 租连接

pool.getConnection((err,conn)=>{

});

(3)归还连接 conn.release();

练习:get /del/301 删除指定用户编号记录

3.5:请求方式:HTTP协议

请求方法:用于标识此次请求的目的

GET 表示客户端想”获得”指定资源

POST 表示客户端想”上传/添加”指定数据给服务器，

请求数据在请求主体中

PUT 表示客户端想”放置”数据在服务器

DELETE 表示客户端想”删除”服务器上指定资源

3.6:发送请求

浏览器发送GET请求

地址栏输入url回车/表单GET/AJAX GET/超链接/js跳转/img src

浏览器发送POST请求

表单POST/AJAX POST

浏览器发送DELETE请求

AJAX-DELETE

浏览器发送PUT请求

AJAX-PUT

3.7:学子商城后台->登录/产品分页/删除/更新/详细

/用户分页/删除/更新/详细

3.8:目录结构

nodejsadmin

node\_modules/…放在第三方模块

app.js 程序入口

public 静态资源

product\_list.html 10

css 10

js 10

img….. 100

3.9: app.js 项目入口程序

1:加载对应模块

http;express;mysql;body-parser;cookie-parser;express-session;

2:先创建连接池

3:创建express 对象

4:配置第三方模块

5:指定静态资源目录 public

3.10:跨域请求

从一个域名下资源访问另一个域名下资源

示例:

www.tmooc.cn/index.js -->ajax-> [www.tts.cn/index.php](http://www.tts.cn/index.php)

跨域:

1:域名不同 tmooc.cn/index.js tts.com/index.php

2:协名不同 http://tmooc.cn/index.js https:/tmooc.cn/index.php

3:端口不同 http://tmooc.cn:80 http://tmooc.cn:8080

4:主机名与IP不同 http://127.0.0.1 http://localhost

跨域有允许访问程序

link->css script->src img->src …

跨域不允许访问程序

ajax->xhr 禁止跨域

学子商城后台跨域

apache/htdocs/nsadmin/**product\_list.html**

/css js img

用户访问地址:http://127.0.0.1/nsadmin/product\_list.html

ajax🡪nodejs/app.js

访问服务器:http://127.0.0.1:3000/product\_list

NodeJS DAY06

今天学习内容

1:总结node.js

1.1:node.js 运行于服务器端开发平台

1.2:node.js与前端js区别 数据类型

node.js第三方模块

前端js:DOM/BOM

1.3:模块:文件或者目录称为模块

文件模发起人node.js自动化

(function(exports,module,require,\_\_dirname,\_\_filename){

//自己代码

});

exports:向外导出成员

module:当前模块对象

require:fn 引入其它模块并创建指定模块对象

\_\_dirname:当前文件模块目录绝对路径

\_\_filename:

1.4:exports 和 module.exports

exports = modules.exports 底层操作

1.5:模块分类

1:自定义模块

#文件 m.js exports. app.js var m = require(“./m”);

#目录 node\_modules/m6/package.json

app.js require(“m6”);

2:原生模块

Buffer

url 解析url地址: url.parse(str,true)

http 创建http服务器/客户端

var server = http.createServer();

server.listen(3000);

server.on(“request”,(req,res)=>{})

fs

同步:阻塞

var data = fs.readFileSync(fileName);

fs.writeFileSync(fileName,data);

fs.appendFileSync(fileName,data);

异步:非阻塞

fs.readFile(fileName,(err,data)=>{});

fs.writeFile(fileName,data,(err)=>{});

fs.appendFile(fileName,data,(err)=>{})

querystring 解析查询字符串

3:第三方模块

mysql

1:创建连接池

var pool = mysql.createPool({connectionLimit:25});

2:pool.getConnection((err,conn)=>{});

3:conn.query(sql,[id],(err,result)=>{

if(err)throw err;

});

4:select result.length

express:

var app = express();

var server = http.createServer(app);

server.listen(3000);

app.get(请求地址”/list”,(req,res)=>{});

req.query.uid

req.body.uid 需要第三方插件

res.send(html);

res.sendFile(绝对路径+发送文件);

res.json({…});

2:学子商城->实现更多的功能[跨域]

模块一:后台管理员登陆

1. 客户端资源

所有静态资源放在 apache/htdocs/nodeadmin

请求地址 http://127.0.0.1/nodeadmin/index.html

1. 服务器端程序node.js

请求地址 http://127.0.0.1:3000/adminlogin

程序工作流程

1:启动node.js 服务器

2:启动apache 服务器(mysql启动)

3:客户请求 apache nodeadmin/index.html

4:发送登录请求 http://127.0.0.1:3000/adminlogin

注意事项:

如果客户端**跨域请求**服务器资源为了保存session工作正确

需要在ajax请求添加属性

xhrFields:{withCredentials:true}

#-----------------------------------------------------------------------

项目:{跨域}

两台web服务器 {nodejs 服务器业务功能}

{apaceh 服务器接收请求响应html静态资源}

一台数据库服务器 {数据 mysql服务器}

扩展思路与知识储备

(1)项目瓶颈:数据库

原理:数据库中数据保存在磁盘上的某一组文件

解决一:提供SQL查询效率

SELECT uid,uname,age FROM xz\_user;

数据库对象:索引 提高查询效率[select];

降低更新效率[update/delete/insert]

什么项目用:索引 查询多更新少

索引为什么快:工作原理

美团快递:送餐{死心眼} 数据库默认工作方式

美团快递:送餐{机灵}

ALTER TABLE xz\_user ADD INDEX (uname)

SELECT uid FROM xz\_user WHERE uname = “tom”;

400w-🡪没有加索引之前 查询一条记录 2s

400w->加索引之后 查询一条记录 0.00002s

解决二:主从数据库服务器{一个服务器负责写数据<主>}

{多个从服务器负责读数据<从>}

解决三:内存级数据库服务器{NOSQL} 极高数据读写 Redis

数据库读写速度太高->数据库压力太大{出问题} cpu 100%

社区:php/mysql 在线人数 t\_online 2 服务器down

$phpcount = 1; ->redis ++ -- 效率很好

tts:学生登录 8:45~9:30 ->mysql🡪Redis

微博/微信 微博[点赞] 120 Redis

(2)提高项目{PV Page View} 高性能服务器/页面静态化

*Nginx/*

(3)安全角度

SQL 注入 node.js ?

XSS 攻击 脚本 oa[报销/申请办公用品/审批] [确认]

document.all.readonly = true

过滤用户所有输入值/所有危险字符串判断

DDOS 攻击 向某个网站发送大量垃圾 防火墙

解决:网店->天猫->阿里云 6G/s

(4)数据分析数据挖掘(大数据/人工智能)

钱🡪一组新技术集合(hadoop/hbase/mapredure/hive…)

15T🡪分析用户行为->记录-(存储hadoop)

分析mapredure –》

….

人工智能:网易课堂:吴恩达

NodeJS DAY07

今天学习的内容

1:复习昨天内容

复习node.js

原理:node.js对文件模块自动代码添加

(function(){})

五个参数:

exports

module.exports

exports.age = 10; 向外导出变量

module.exports.age = 10;

module.exports = {name:”tom”}; ok

exports = {name:”tom”}; error

require();

2:今天的目标

2.1: 中间件-(路由级中间件)

Express 是一个自身功能极简，完全是中间件构成web框架

从本质上讲一个express应用就是由中间件组成系统.

2.2:中间件(Middleware)是一个函数,它可以访问请求对象和响应对象

(req request,res response),可以控制请求-响应流程,有一个参数

next变量 下一个中间件

2.3:中间件功能包括

执行任何代码

修改请求和响应对象

终结请求-响应循环 [拦载器]

调用下一个中间件

2.4:中间件分类

应用级中间件(\*\*\*)

路由级中间件(\*\*\*)

错误处理中间件

内置中间件

2.5:标准语法

app.use(url,(req,res,next)=>{

});

url:拦载地址(触发地址)

req:请求对象

res:响应对象

next:调用下一个中间件或路由

2.6:路由中间件

路由 = 请求方法+请求地址+处理函数

示例: app.get(“/list”,(req,res)=>{…})

语法

var router = express.Router(); //创建路由中间件对象

router.get(“/find”,(req,res)=>{});

router.post(“/list”,(req,res)=>{});

module.exports = router;

app.use(“/user”,router);

http://127.0.0.1:3000/user/find

http://127.0.0.1:3000/user/list

(1)创建三个路由文件

router/userRouter.js 保存所有用户功能

router/orderRouter.js 保存所有订单功能

router/productRouter.js 保存所有产品功能

app.js

var userRouter = require(“./router/userRouter”);

var orderRouter = require(“./router/orderRouter”);

var productRouter = require(“./router/productRouter”);

app.use(“/user”,userRouter);

app.use(“/order”, orderRouter);

app.use(“/product”, productRouter);

http://127.0.0.1:3000/user/list

http://127.0.0.1:3000/order/list

2.7: 项目

课程准备

知识点{思维导图/github}

JS 基础[原型/原型链;闭包;ES6;面试题]

dom/bom

JQuery

Canvas[小鱼大鱼]

Node.js

Vue[概念背]

Rect[概念背]

项目

参考www.codeboy.com/pro

最少二个{特效多,精彩效果} 原生版/vue版

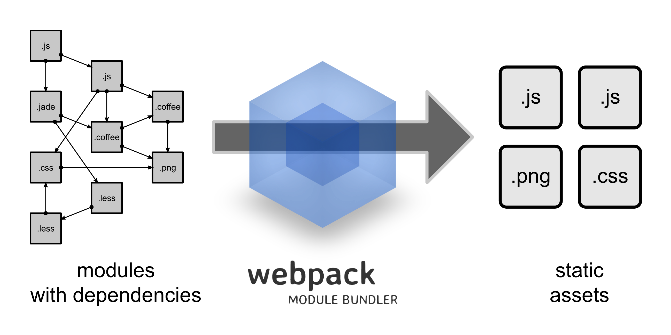
功能不用太多;细致

微信登录/微博登录

动画

商城:购物车/多选产品

2.8: 打包工具 webpack-----(演示)



webpack是一款流行打包工具，将各种资源

(.coffee/.less/.sass/图片)作为模块进行处理，生成一个或一组文件

安装配置使用

(1)创建目录 E:\>mkdir webpack-test

(2)进入指定目录 E:\>cd webpack-test

(3)初始化目录 E:\webpack-test>cnpm init

(4)询问一些问题:按回车选择默认值 自动生成文件

package.json

package name: (webpack-test) 包名

version: (1.0.0) 版本

description: 描述

entry point: (index.js) 入口程序

test command: 测试指令

git repository: git仓库地址

keywords: 关键字

author: 作者

license: (ISC) MIT MIT

Is this ok? (yes) yes

|  |
| --- |
| 如果下载npm第三方模块:  采用淘宝镜像  1:全局安装cnpm  npm install -g cnpm --registry=https://registry.npm.taobao.org  2:再安装其它工具 将npm 切换 cnpm |

(5)安装依赖工具

cnpm install webpack -save-dev

(6)全局安装webpack 指令直接操作

cnpm install webpack -g

1:创建test.js文件

webpack test.js test.bundle.js

webpack word.js word.bundle.js

2:创建1.css文件

webpack word.js word.bundle.js

3:安装二个css加载模块

cnpm install css-loader

cnpm install style-loader

webpack word.js word.bundle.js

4:监控项目,实时修改/过程/哪些模块

webpack word.js word.bundle.js --progress

webpack word.js word.bundle.js --watch

错误集锦:

1:node.js 404

原因1:node.js 服务器出错故障/没有启动

原因2:请求地址不正确

服务器: app.post(“/userupdate”,(req,res)=>);

客户端:ajax type:”POST”

url:”http://127.0.0.1:3000/userupdate”

vue DAY01

张众磊

bjzhangzl@tedu.cn

一、本阶段学习安排

①VueJS（3天+3天）

②Angular (4天+2天)

③react (4天+3天)

二、约定

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）

cn.vuejs.org

vuejs.org

框架：framework 有着自己的语法特点、都有对应的各个模块

库 library 专注于一点

框架的好处：

①提到代码的质量，开发速度

②提高代码的复用率

③降低模块之间的耦合度

（高内聚低耦合）

UI：user interface

GUI : graphical user interface

CLI : command line interface

API : application interface

在webStorm中创建一个模板的基本步骤：

①准备模板内容 全部选中 复制

②点击左上角的file -> new -> edit file templates

③点击弹窗中左上角+

④在输入框当中分别输入 模板名称、模板结尾的格式、模板内容

⑤点击ok ，保存，下次就可以使用

思维模式的转换：

从操作DOM的思维模式 切换到 以数据为主

一、Vue概述

1、what

是一个渐进式的构建用户界面的js框架

2、where

小到的简单的表单处理，大到复杂的数据操作比较频繁的单页面应用程序

3、why

①方便阅读的中文文档

②容易上手 （学习曲线比较缓和）

③体积小

④基于组件化的开发方式

⑤代码的可读性、可维护性得到了提高

4、how

工作方式：可以通过丰富的指令扩展模板，可以通过各种各样的插件来增强功能

搭建环境：

方式1

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

方式2:

直接引入对应的js文件

二、Vue中基础知识

1、双花括号

mustache(胡子)/interpolation（插值表达式）

语法：

<any>{{表达式}}</any>

作用：

将表达式执行的结果 输出当调用元素的innerHTML中；还可以将数据绑定到视图

练习：

demo03\_lianxi.html的文件

希望通过Vue去初始化一个学生对象(name/score)数组(5个)

将学生数组 渲染在table中

2、指令-循环指令

基本语法1：

<any v-for="tmp in array"></any>

基本语法2：

<any v-for="(value,index) in array"></any>

作用：

在遍历array这个集合时，将临时变量保存在tmp中，创建多个any标签

3、指令-选择指令

语法：

<any v-if="表达式"></any>

<any v-else-if="表达式"></any>

<any v-else="表达式"></any>

作用：

根据表达式执行结果的真假，来决定是否要将当前的这个元素 挂载到DOM树

练习：demo06\_lianxi.html

通过v-for v-if来重构demo03\_lianxi

要求：只显示成绩及格的同学

<tr v-for="stu in stuList" v-if="stu.score >= 60">

</tr>

4、指令-事件绑定

语法：

<any v-on:eventName="handleEvent"></any>

作用：

给指定的元素 将handleEvent的方法绑定给指定eventName事件

练习：

demo08\_lianxi.html

在data中初始化一个整形数组（3条数据：100,200,300）

视图：将数组展示在ul，在ul的下边添加上一个按钮：加载更多；

在点击按钮时，ul中的尾部追加一条新的随机数显示li

5、指令-属性绑定

基本语法：

<any v-bind:myProp="表达式"></any>

补充，支持简写：

<any :myProp="表达式"></any>

作用：

将表达式执行的结果 绑定 到当前元素的myProp属性

<img v-bind:src="'img/'+myImg" alt="">

动态样式绑定

<p :style="{backgroundColor:myBGColor}">动态样式绑定</p>

动态样式类绑定

<h1 :class="{myRed:false}">动态样式类的绑定</h1>

6、指令-双向数据绑定

方向1：数据绑定到视图

方向2：将视图中（表单元素）用户操作的结果绑定到数据

基本语法:

<表单元素 v-model="变量">

</表单元素>

练习： demo12\_lianxi.html(-:16:40)

在视图中：

checkbox button

实现的效果：

checkbox被选中，button可以被点击

checkbox取消选中，button不可操作

checkbox 可以通过双向数据绑定，将操作的结果绑定到某个变量；

变量绑定到button disabled

<button :disabled=''>

三、组件化

组件：组件就是可被反复使用的，带有特定功能的视图

所谓的组件化，就像玩积木一样，把封装的组件进行复用,把积木（组件）拼接在一起，构成一个复杂的页面应用程序。

组件树就是由各个组件构成的一种数据结构，它存在的意义是为了帮梳理应用程序

1、组件的创建

①全局组件

Vue.component('my-com',{

template:`

<h2>it is a header</h2>

`

})

②局部组件

new Vue({

components:{

'my-footer':{

template:''

}

}

})

2、组件使用

作为普通的标签去使用

<my-com></my-com>

3、注意事项

①组件的id和使用方式 遵循烤串式命名方式：a-b-c

②如果一个组件 要渲染多个元素，将多个元素放在一个顶层标签中，比如div、form

③全局组件可以用在id为example的范围内的任何一个组件内部，直接调用可以；但是局部组件只能在父模板中直接调用

每日一练：

分页功能

要求：

初始化一个数组pageList:[1,2,3,4,5],渲染一个按钮列表

nowPage:1

功能：

点击上一页、下一页，被选中的页码要高亮显示

（ <button :class="{}"></button> 条件是当前选中的页码和当前按钮的对应的页码一致）

vue DAY02

总结目前为止所掌握的指令：

v-for

v-if

v-else-if

v-else

v-on: @

v-bind: :

v-model

v-show

v-html

零、补充

v-model.number v-model.lazy v-model.trim

v-on.prevent

v-html

一、自定义指令

1、创建和使用

Vue.directive('change',{

bind:function(el,bindings){

//首次调用

},

update:function(el,bindings){

//只要是有数据变化，都会调用

},

unbind:function(){

//解绑

}

})

<any v-change="count"></any>

练习：

demo03\_lianxi.html

创建一个自定义的指令，允许接受参数，将接受到的参数用来修改调用当前元素的背景色

el.style.backgroundColor = bindings.value;

二、过滤器

过滤器是针对一些数据 进行筛选、过滤、格式化等相关的处理，变成我们想要的数据

过滤器的本质 就是一个带有参数带有返回值的方法

Vue1.\* 支持内置的过滤器，但是Vue2.\* 就不再内置过滤器，但是支持自定义过滤器。

1、过滤器的创建和使用

①创建

Vue.filter(

'myFilter',

function(myInput){

//myInput是在调用过滤器时，管道前表达式执行的结果

//针对myInput，按照业务需求做处理

//返回

return '处理后的结果'

})

②使用

<any>{{expression | myFilter}}</any>

2、如何在调用过滤器时，完成参数的发送和接受

①发送

<any>{{expression | myFilter(参数1，参数2)}}</any>

②接受

Vue.filter('myFilter',function(myInput，参数1，参数2){

return '处理后的结果'

})

(-:14:55)

练习：demo05\_lianxi.html

实现一个自定义的过滤器，在数据中有一个sex 0/1

h2 sex 0 | true/false--》女/girl

h2 sex 1 | --》男/boy

需要一个过滤器，根据管道传递来的数据，以及参数true/false来决定显示(true->中文 false->英文)

三、复合组件

知识回顾：

Vue.component('my-header',{

template:`<div></div>`

});

<my-header></my-header>

复合组件：并不是新的概念，就是一个组件，只不过这个组件中 可以调用其他的组件

注意事项：

①组件要渲染的内容 取决于在定义组件时template

<my-list>

<my-item></my-item>

</my-list>

效果是出不来的

②允许在一个组件中，直接来调用另外一个组件

练习：

创建一个登录窗口

my-login

my-input label/input

my-button button

四、生命周期

四个阶段：

create 准备工作 （数据的初始化。。。）

mount 挂载前后针对元素进行操作

update 数据发生变化，

destroy 清理工作 (关闭定时器、集合清空..)

beforeCreate/created

beforeMount/mounted

beforeUpdate/updated

beforeDestroy/destroyed

练习：( -：17:00)

demo09\_lianxi.html

创建一个组件（h1），在组件挂载完毕之后，启动一个周期性定时器，

实现的效果： h1 的透明度 0~1

①完成组件的创建和使用

②指定生命周期的处理函数

③动态样式绑定

④完成定时器的使用

五、常用属性

1、watch

① 表单元素的双向数据绑定

v-model="myValue"

②监听

watch:{

myValue:function(newValue,oldValue){

}

}

2、computed

计算属于是用于在模板中，搞定复杂的业务逻辑，因为有依赖缓存。

①指定计算属性

computed:{

myHandle:function(){

return 数据

}

}

②调用

<any>{{myHandle}}</any>

每日一练：

vue DAY03

一、组件间通信

1、父与子通信 （props down）

①发送

<son myName='zhangsan'>

</son>

②接受

到son组件：

Vue.component('son',{

props:['myName'],

template:`

<p>{{myName}}</p>

`

})

练习：demo02\_lianxi.html

(-: 10:40)

my-input

my-button

my-login

2、子与父通信 (events up)

①绑定

methods:{

handleEvent:function(msg){}

}

<son @customEvent="handleEvent"></son>

②触发

子组件内部：

this.$emit(‘customEvent’,100);

练习：综合练习

my-chatroom:一个无序列表 和 两个my-user构成

my-user：label/input/button

①完成组件的创建和使用

②调用my-user指定mike lucy (父与子)

③点击发送 （子与父）

3、ref(reference 引用/参考 外号)

帮助在父组件中 得到子组件中的数据、方法。

①指定ref属性

<son ref="mySon"></son>

②根据ref得到子组件实例

this.$refs.mySon

4、$parent

this.$parent得到父组件的实例

5、兄弟组件通信

①var bus = new Vue();

②接收方

bus.$on('eventName',function(msg){})

③发送方

bus.$emit('eventName',123);

练习：

在熊大-》熊二通信的基础之上；再来实现熊二与熊大的通信

在熊二：添加一个按钮，点击按钮，发送数据

(知道了)给熊大（在控制台打印出来即可）

熊二（触发） --》熊大（绑定）

二、补充组件创建的方式

1、直接在template属性中指定模板内容

①全局组件

Vue.component

②局部组件

{

components:{

'my-footer'：{template:``}

}

}

2、.vue结尾的文件

<template></template>

<script></script>

<style></style>

3、单独指定一个模板内容

<script

id='myContent'

type='text/x-template'>

</script>

Vue.component('',{

template:'#myContent'

})

综合练习：

完成ToDoBox的添加功能

①完成todoinput点击add，将数据发送todolist

todoinput（触发） --> todolist(绑定)

②todolist接受到数据，在控制台打印发送的数据

③在todolist中 保存每次添加进来的数据在数组中，根据数组元素的个数动态创建多个todoitem (v-for)

④在todolist中调用todoitem时，通过props down 来实现父与子的通信

can't read property push of undefined

每日一练：

实现删除功能（子-》父）

①要通过父-》子 告诉todoitem下标

②todoitem点击delete按钮时，将下标通知给todolist，

todolist接受到下标，从数组中移除指定位置的元素

vue DAY04

一、路由模块

路由模块的本质 就是建立起url和页面之间的映射关系

1、SPA的基本概念和工作原理

SPA：single page application 单一页面应用程序，只有一个完整的页面；它在加载页面时，不会加载整个页面，而是只更新某个指定的容器中内容。

比如Gmail、移动的webApp

工作原理：

①解析地址栏

完整的页面地址、路由地址

②根据路由地址 从路由词典中找到真正的要加载的页面

③发起ajax请求

请求要加载的页面

④像指定的容器中 插入加载来的页面

2、路由模块的基本使用

专业术语：

router路由器

route路由

routes 路由数组（路由词典）

①引入vue.js vue-router.js

②指定一个容器

<router-view></router-view>

③创建业务所需要用到的组件类

var MyLogin = Vue.component()

④配置路由词典

const myRoutes = [

{path:'',component:MyLogin},

{path:'/login',component:MyLogin}

];

const myRouter = new VueRouter({

routes:myRoutes

})

new Vue({

router:myRouter

})

⑤测试

修改地址栏中的路由地址，测试看加载的组件是否正确

注意事项：

①先引入vue，再引入插件

②一定要指定router-view

③route路由 {path:'',component:}

routes 路由数组 []

router 路由器:按照指定的路由规则去访问对应的组件 new VueRouter

练习：demo02\_lianxi.html

创建一个SPA，有两个组件类，分别是login,main.

3、使用路由模块来实现页面跳转的方式

方式1：直接修改地址栏

方式2：js

this.$router.push(‘路由地址’);

方式3：

<router-link

to="路由地址"></router-link>

4、完成参数的传递

在页面之间跳转的时候，在有些场景下，需要同时指定参数

①明确发送方和接收方

list --20--> detail

②配置接收方的路由地址

/detail --》 /detail/:index

this.$route.params.index

③发送

routerLink to="/detail/20"

this.$router.push('/detail/20')

练习：demo05\_lianxi.html

SPA，有3个组件：

check/pay/send

check:button 点击跳转到pay,同时传递商品价格100给pay

pay: h1 显示商品价格信息 button 点击跳转send

send: routerLink-->check

步骤1:搭建架子，要求通过不同的url能够访问3个组件

步骤2：搞定跳转

步骤3：搞定传参

①明确发送方和接收方

check（发送） pay(接受)

②配置pay的路由地址

/pay --> /pay/:price

this.$route.params.price

③跳转的同时发送

5、路由嵌套

在一个路由中，path对应一个component，如果这个component需要根据不同的url再加载其他的component，称之为路由的嵌套

举例：比如A组件现在需要根据不同的url，加载B组件或者C组件

①给A组件指定一个容器

<router-view></router-view>

②配置路由词典

{

path:'/a',

component:A,

children:[

{path:'/b',component:B}

]

}

需求：现在有两个组件，分别是login/mail,建立SPA。

在此基础上，希望mail组件 嵌套inbox/outbox/draft

补充：在设置子路由，路由匹配规则依然是适用的，只不过路由地址为空和异常，要携带父组件的路由地址

/mail /mail/draft

二、与远程服务器端的通信

①引入vue-resouce.js

②发请求

this.$http.get(url).then((result)=>{

//result.data

})

三、搭建基于CLI开发环境的方式

①指定一个文件夹

C:\xampp\htdocs\framework\vue\project

②将tpls.zip拷贝到project中

③右键单击压缩包，解压缩到当前文件夹

④进入到tpls

⑤同时按下shift和鼠标右键，选择在此位置打开命令行串口

⑥执行npm install

⑦执行npm start

vue DAY05

1、针对路由模块综合练习

(:- 10:00)

练习：实现一个SPA，在这个SPA有两个组件类，cart和orderConfirm；

①在cart有一个按钮，点击跳转到orderConfirm，在跳转的同时发送价格为200

②orderConfirm接受到价格信息，显示在h2标签

③在orderConfirm中嵌套stepOne stepTwo，同时在orderConfirm中希望默认加载stepOne

在orderCOnfirm，指定容器 router-view

在orderConfirm的路由对象指定children

2、熟悉基于命令行的开发方式

2.1 认识工程目录结构

assets 资源文件(img/css/js/fonts/..)

components 组件

router/index.js 完成路由词典的设置

App.vue 根组件

main.js 创建一个Vue的实例，同时指定根组件

2.2 启动的方式

npm start之后做的事情：

①通过node执行build/dev-service.js 指定开发服务器并端口8080

②加载index.html和main.js

(在main.js 通过new Vue指定router/根组件)

③根据路由匹配规则，从router/index.js加载路由地址为空 显示Hell.vue

2.3 如何在模板中调用创建好的.vue结尾的文件？

2.3.1 全局组件

main.js

import MyHeader from '@/components/myheadder'

Vue.component('my-header',MyHeader)

2.3.2 局部组件

比如myList.vue需要myHeader

只需要在myList.vue的script中引入对应的组件类

import MyHeader from '@/components/myheadder'

export default{

components:{

my-header:MyHeader

}

}

3、练习 (:-15:30)

基于命令行的方式来实现todolist

①完成组件的创建

todobox.vue todolist.vue todoitem.vue todoinput.vue

②配置todolist todoitem todoinput作为全局组件

找main.js Vue.component

③给todobox指定路由地址

④实现add按钮的功能

步骤1：先实现数据 从todoinput到todolist

步骤2：在todolist中 动态创建多个todoitem,传递todoitem要显示的内容

项目大致流程：

①需求分析

②可行性分析

③概要设计、详细设计

④编码 ui test

⑤联合调试 test

⑥上线

⑦迭代、维护

4、搭建项目的脚手架

①启动apache和mysql

②初始化数据库

admin/xz.sql

③需求分析

login index

index进行路由嵌套（charts/productList/userList）

④修改项目的src目录

删除项目的src目录，从ftp中的vue/day05/src\_new.rar下载到本地，拷贝粘贴到项目的根目录，解压缩到当前文件夹

⑤准备需要用到的文件和文件夹

在assets目录下创建css文件夹、js文件夹，img文件夹

将bootstrap.css、jquery.js、jquery.flot.js、图片拷贝到对应的文件夹中

⑥创建spa中所需要用到login/index/charts/productList/userList组件类

⑦配置路由词典

login/index

--charts

--productList

--userList

每个组件就先显示一个h1标签。

要求：默认加载login，在index嵌套组件，默认加载charts组件到index

vue DAY06

综合练习：

demo02.html

开发方式：直接引入对应的js文件

实现一个SPA，两个组件类，分别是cart和orderConfirm，希望SPA默认显示cart；

在cart中有一个按钮（去结算）--》跳转到订单确认页面，并将价格123，传递给orderConfirm

orderConfirm接受传递来的参数渲染在h1标签中

传参步骤：

①明确发送方和接收方

cart--》orderConfirm

②配置接收方的路由地址

/oc --> /oc/:price

this.$route.params.price

③发送

/oc/123

ANGULAR DAY01

项目总结：

技术：vue vue-router vue-resource

bootstrap/fonts jquery jquery-flot 跨域

vue {{}} v-model @click lifecycle

vue-router spa 跳转 嵌套

vue-resource

一、Angular概述

what

是由Google和开源社区共同维护的一个js的框架

where

网页版的微信 。。

数据操作为主的SPA，也包括使用ionic来实现移动端的app，网页应用程序。。。

why

09年angular就诞生了 angularJS

angular采用强类型检查机制的ts

angular采用了很多优秀的工具，提高开发速度 网页性能 (rxjs/zone.js/immutable.js)

angular采用组件化的开发方式

how？

工作方式

通过组件来构造模板内容

通过服务来封装常用的功能

通过注入器来注入需要用到的模块或者服务

搭建环境

方式1：

npm install -g @angular/cli （全局安装angular的cli）

ng new my-app

cd my-app

ng serve --open

方式2：

下载模板项目

https://github.com/angular/quickstart

解压缩quickstart-master.zip

cd quickstart-master

npm install

npm start

vscode的基本使用

npm start开发服务器启动流程

（index.html-> my-app --> src/app/app.component.ts --> template）

工程的目录结构：

src

--app

--app.component.ts 根组件

--app.module.ts 根模块

main.ts 指定要启动哪个模块

index.html 首页

styles.css 全局样式类

二、组件的创建和使用

组件的本质是可被反复使用的，带有特定功能的视图

Vue:

Vue.component('my-header',{

template:``

})

<my-header></my-header>

.vue

Angular

1、组件的创建

①在src/app创建一个文件夹demo01

②在demo01文件夹中创建demo01.component.ts

③创建组件类

import {Component} from '@angular/core'

@Component({

//指定在模板中调用组件的方式

selector:'demo01'，

template:``

})

export class Demo01Component{}

2、组件的使用

①在模块中声明组件类

找到app.module.ts

import {Demo01Component} from './demo01/demo01.component'

declarations:[Demo01Component]

②使用

<demo01></demo01>

装饰器： decorator

是用来指定当前的类是什么类型

比如 @Component 组件类 @NgModule 模块类

元数据：metaData

是用来告诉angular如何来处理当前的类

比如 selector 是用来告诉angular当前这个类在模板中要怎么调用

练习：(:- 17:30)

在app/demo02文件夹中demo02.component.ts

要求demo02渲染一个无序列表（3个列表项 test01 test02 test03）

在根组件中调用demo02

三、angular指令的使用

Vue中指令：

v-for v-if v-else-if v-else

v-bind v-on v-model

v-show v-html....

Angular中的指令：

1、循环指令 ngFor

<any v-for="tmp in list"></any>

<any v-for="(tmp,index) in list">

</any>

<any

\*ngFor="let tmp of list;let myIndex=index">

</any>

2、选择指令

<any v-if="expression"></any>

根据表达式执行结果的真假来决定是否要挂载到DOM

<any \*ngIf="expression"></any>

每日练习：

app/demo03

demo03.component.ts

在这个组件类中，创建一个对象数组 stuList(有5个对象),每一个对象都包含score/name/age

将stuList中的每一个对象的数据 渲染在table中。

ANGULAR DAY02

一、ng中常见的指令

指令扮演的角色是 为了让模板内容具有更多的可能性（添加额外的功能）

注意事项：不能在一个元素上 同时调用多个结构型指令

解决思路：Angular的<ng-container>是一个分组元素，但它不会污染样式或元素布局，因为 Angular 压根不会把它放进 DOM 中

1、循环指令

\*ngFor

2、选择指令

\*ngIf

<ng-container></ng-container>可以解决在一个标签中使用多个指令的问题。

多重分支判断

<div [ngSwitch]="answer">

<p \*ngSwitchCase="'a'">A</p>

<p \*ngSwitchCase="'b'">B</p>

<p \*ngSwitchDefault></p>

</div>

3、属性绑定

vue: v-bind:src :src

<any [prop]="expression"></any>

将表达式执行的结果 绑定给 指定的元素的属性

动态样式绑定：

<p [ngStyle]="{color:myColor}">

it is a paragraph

</p>

动态样式类绑定：

<span [ngClass]="{myRed:false}">

it is a span element

</span>

4、事件绑定

vue: @click v-on:click

<any (EventName)="handleEvent()">

</any>

5、双向数据绑定

方向1：从数据到视图

方向2：将表单元素操作的结果 绑定指定的数据

Vue:v-model

Angular:

准备工作：

①app.module.ts

②import {FormsModule} from '@angular/forms'

③ @NgModule({

imports:[FormsModule]

})

<input [(ngModel)]="uname"/>

（:- 14:35）

练习：app/demo06 demo06.component.ts

渲染的模板内容：button/p

功能：在点击button时，启动一个定时器，让p的透明度0~0.1~0.2~..1

①组件的创建和使用

②ngStyle 事件绑定

监听模型数据的变化：

<!--扩充：如何在angular中监听模型数据的变化-->

<input type="text"

[(ngModel)]='uAddress'

(ngModelChange)="handleChange()"

/>

总结：

\*ngFor

\*ngIf \*ngSwitchCase \*ngSwitchDefault

ngStyle ngClass

()

[]

[(ngModel)]

二、ng中自定义指令

Vue自定义指令

Vue.directive('change',{

bind:function(el,bindings){

//el指调用该指令的元素

//bindings.value

},

update:function(){},

unbind:function(){}

})

v-change

Angular中自定义指令

1、创建一个指令类

import {Directive} from '@angular/core'

@Directive({selector:'[test]'})

export class TestDirective{

...

}

2、使用

①先到模块中声明

import {TestDirective} from '\*\*\*'

@NgModule({

delcarations:[TestDirective]

})

②使用

<any test></any>

补充：组件和指令的关系

组件就是一个带有模板的指令

进阶：如何在自定义指令类中 获取调用指令的元素？

import {ElementRef} from '@angular/core'

constructor(myEL:ElementRef){

//myEL.nativeElement

}

三、ng中常见的过滤器(管道)

Vue

<any>{{expression | myFilter(1,2)}}</any>

Angular管道使用语法：

<any>{{expression | myPipe:''}}</any>

<any>{{expression | myPipe1:'' | myPipe2:'':''}}</any>

uppercase lowercase date number slice

四、ng中自定义过滤器(管道)

declarations: 指令、管道、组件

Vue创建一个自定义过滤器：

Vue.filter('myCurrency',function(value,arg1){

return '操作后的结果'

})

{{expression | myCurrency(1)}}

管道（过滤器）的本质：带有参数带有返回值的方法

1、创建一个管道类

①test.pipe.ts

②

import {Pipe,PipeTransform} from '@angular/core'

③指定装饰器和元数据

@Pipe({

name:'myUppercase'

})

④实现管道类

export class TestPipe implments PipeTransform{

transform(value:any,...args:[]):any{

return '处理后的结果'

}

}

2、使用

① 到模块中声明

app.module.ts

import {TestPipe} from '...'

@NgModule({

declarations:[TestPipe]

})

② 调用

和内置的管道是一样的用法

angular开发整体框架涉及到8个部分：

模块：angular开发基本单位

组件：带有模板的指令/可被反复使用的带有特定功能的视图

模板：增强后的html标签 h1 \*ngFor \*ngIf {{}}

数据绑定: [] () [(ngModel)] \*ngFor..

元数据: metaData 告诉angular如何来处理当前这个类

指令：是为了扩展html的功能

依赖注入：将当前所需要用到的东西 拿到当前这个类中去使用的过程，为了降低耦合度

服务

每日一练：

自定义管道

练习：app/demo09\_lianxi

实现一个自定义的管道，在数据中有一个sex 0/1

h2 sex 0 | true/false--》女/girl

h2 sex 1 | --》男/boy

需要一个过滤器，根据管道传递来的数据，以及参数true/false来决定显示(true->中文 false->英文)

ANGULAR DAY03

一、service

服务是为了封装应用中逻辑，为了在组件中去复用

1、创建

log.service.ts

import {Injectable} from '@angular/core'

@Injectable()

export class LogService{

test(){}

}

2、使用

2.1 到模块中指定提供商

app.module.ts

import {LogService} from '...'

@NgModule({

providers:[LogService]

})

2.2 到组件中去调用

//引入

import {LogService} from '...'

//到构造函数中去实例化

constructor(private myLS:LogService){}

//使用

this.myLS.test()

3、练习：实现一个心跳服务，在心跳服务中封装两个方法：

start 启动周期性定时器，每隔0.5s '在心跳'

stop 关闭定时器

demo11\_lianxi/demo11.component.ts

两个按钮，开始心跳 结束心跳

①完成组件的创建、声明、调用

②完成服务的创建、提供商的指定、调用

4、网络请求服务

4.1 创建

myhttp.service.ts

constructor(private http:Http){}

sendRequest(myUrl:string){

//a-http-get

return this.http.get(myUrl)

.map((response:Response)=>response.json())

}

4.2 使用

4.2.1 到模块中指定Providers

app.module.ts

import {MyHttpService} from '..'

@NgModule({

providers:[MyHttpService]

})

4.2.2 指定依赖于HttpModule

app.module.ts

import {HttpModule} from '@angular/http'

@NgModule({

imports:[HttpModule]

})

4.2.3 到组件去调用服务

import {MyHttpService} from '..'

constrcutor(private myHS:MyHttpService){}

this.myHS.sendRequest('').subscribe((result)=>{})

4.3 进阶

处理通信过程中涉及到session的问题？

4.3.1 客户端

this.http.get(myUrl,{withCredentials:true})

4.3.2 服务器

header('Access-Control-Allow-Origin:http://localhost:3000');

header('Access-Control-Allow-Credentials:true');

4.4 练习(:- 15:55)

创建一个app/demo13.component.ts

store.http.service

服务需求：

发起网络请求

组件类：

一个无序列表 一个按钮

点击按钮时，通过服务中的方法，获取商品列表，将商品的标题显示在无序列表中

二、组件之间的通信的方法

学习一个框架：基本语法+通信的方式(内部、服务器)+router

1、复习 Vue中组件间通信：

①props down (father->son)

发送

Vue.component('father',{

template:`<son sonName='zhangsan'></son>`

})

接受

Vue.component('son',{

props:['sonName']

// this.sonName

})

②events up (son->father)

绑定(father)

Vue.component('father',{

template:`<son @customEvent="handleEvent"></son>`

})

触发(son)

this.emit('customEvent',123)

③ref

④$parent

⑤bus

2、Angular组件间通信：

1、props down(父与子的通信)

1.1 发送

<demo14-son

msg='hello world'>

</demo14-son>

1.2 接受

import {Input} from '@angular/core'

@Input() msg="";

// this.msg

2、events up (子与父的通信)

2.1 绑定

<demo14-son (customEvent)="handleEvent()">

</demo14-son>

2.1.1 定义一个方法

rcvMsg(msg:string){}

2.1.2 绑定

<demo14-son (toFatherEv ent)="rcvMsg($event)"> </demo14-son>

2.2 触发

import {Output,EventEmitter} from '@angular/core'

@Output() toFatherEvent = new EventEmitter();

this.toFatherEvent.emit('123');

三、自定义指令如何来实现数据的传递

1、发送

<any test="123" ></any>

2、接受

到指令类中接受传递来的数据

import {Input} from '@angular/core'

@Input() test = "";

//this.test

每日一练：

父子组件相互通信

demo15/my-chatroom.component.ts

my-user.component.ts

my-chatroom:一个无序列表 和 两个my-user构成

my-user：label/input/button

①完成组件的创建和使用

②调用my-user指定mike lucy (父与子)

③点击发送 （子与父）

ANGULAR DAY04

https://router.vuejs.org/zh-cn/advanced/navigation-guards.html

const router = new VueRouter({ ... })

router.beforeEach((to, from, next) => {

// ...

})

一、组件间通信的补充

1、ViewChild

①父组件中调用子组件指定引用的名称

<son #mySon></son>

②调用

import {ViewChild} from '@angular/core'

@ViewChild('mySon') mySon = null;

this.mySon//son组件的实例对象

2、本地变量#

①父组件中调用子组件指定引用的名称

<son #mySon></son>

②直接在父组件的模板当中操作son实例

mySon.\*\*

<button (click)="handleClick(mySon.sonName)"></button>

二、路由模块的使用

路由模块的定位：建立url和组件映射关系

1、复习Vue中路由模块

基本使用

引入vue-router.js

<router-view></router-view>

创建需要用到的组件

路由词典： new Vue({

router:new VueRouter({

routes:[

{path:'/',component:\*\*\*},

{path:'/login',component:\*\*\*},

{path:'\*',component:NotFound}

]

})

})

跳转方式

①直接修改地址栏

② this.$router.push()

③ <router-link to=""></router-link>

this.$router.go(-1);

传参

①明确发送方和接收方

②配置接收方的路由地址

{path:'/detail'}

-->

{path:'/detail/:id'}

this.$route.params.id

③发送

this.$router.push('/detail/2')

<router-link to="/detail/2">

</router-link>

嵌套路由:

在一个路由对象 支持嵌套其他的路由对象（在一个组件 可以嵌套其他的组件)

比如A组件 需要嵌套B组件 C组件

①给A组件指定个容器

<router-view></router-view>

②给B、C配置路由

{

path:'/a',

component:A,

children:[

{path:'/',component:B},

....

]

}

2、Angular路由模块的基本使用

①指定容器

<router-outlet></router-outlet>

②创建需要用到的组件类

③自定义的路由模块

app.router.ts

配置路由词典

注意事项:forChild->forRoot

④根模块中去指定依赖自定义路由模块

app.module.ts

import {AppRoutingModule} from './app.router'

@NgModule({

imports:{AppRoutingModule}

})

3、Angular页面之间的导航

①直接修改地址栏

②js

import {Router} from '@angular/router'

constructor(private myRouter:Router){}

this.myRouter.navigateByUrl('register')

③a

<a routerLink="register"></a>

④返回上一页

import {Location} from '@angular/common'

constructor(private myLocation:Location){

}

this.myLocation.back();

练习： (:- 14:40)

demo17\_lianxi

一个spa，有两个组件，

demo17\_product.component.ts

(购买按钮，点击跳转cart)

demo17\_cart.component.ts

(查看详情 a，点击跳转product)

要求默认显示 demo17ProductComponent

4、Angular路由传参

①明确发送方和接收方

②配置接收方的路由地址

/detail --> /detail/:id

//vue:this.$route.params.id

import {ActivatedRoute} from '@angular/router'

constructor(private myAR:ActivatedRoute){}

this.myAR.params.subscribe((result)=>{

//result.id

})

③发送

routerLink="detail/2"

[routerLink]="'detail/'+myIndex"

this.myRouter.navigateByUrl('detail/2')

this.myRouter.navigateByUrl('detail/'+myIndex)

5、Angular路由嵌套

login

mail (inbox outbox)

①完成4个组件的创建和声明

②给mail指定容器

router-outlet

③配置路由

{

path:'mail',

component:Demo19MailComponent,

children:[

]

}

注意事项：跳转到嵌套的路由: /mail/inbox

6、Angular路由守卫

①service创建

import {Injectable} from '@angular/core'

import {CanActivate} from '@angular/router'

@Injectable()

export class Demo20Guard implements CanActivate{

//返回的值 决定了保护的路由对象中的组件类是否能够被访问

canActivate(){

//处理

return true/false

}

}

②到模块中的providers指定service

app.module.ts

import {Demo20Guard} from '...'

@NgModule({

providers:[Demo20Guard]

})

③配置路由词典指定路由守卫

app.router.ts

import {Demo20Guard} from ''

{

path:'demo20',

component:Demo20Component,

canActivate:[Demo20Guard]

}

每日一练：

练习：demo21/

SPA有3个组件：

Demo21CheckComponent/Demo21PayComponent/Demo21SendComponent

check:button 点击跳转到pay,同时传递商品价格100给pay

pay: h1 显示商品价格信息 button 点击跳转send

send: routerLink-->check

步骤1:搭建架子，要求通过不同的url能够访问3个组件

步骤2：搞定跳转

步骤3：搞定传参

①明确发送方和接收方

②配置pay的路由地址

③跳转的同时发送

ANGULAR DAY05

练习

ANGULAR DAY06

练习

REACT DAY01

React（ReactJS ReactNative）

一、ReactJS概述

what

是facebook一个构建用户界面的js库

where

用在数据操作比较频繁（视图更新频繁--》DOM操作-》浏览器的性能瓶颈）

why

①浏览器的性能瓶颈

虚拟DOM 最大限度减少DOM的操作

②代码的开发和维护成本很高

单向数据流 减少业务的复杂度

how

方式1：cli

https://github.com/facebook/create-react-app

方式2：直接引入对应的js文件

二、ReactJS的核心概念

0、基础知识

①方法

ReactDOM.render(要渲染的元素，渲染的容器)

②jsx

ReactDOM.render(

<h1>hello world</h1>

，渲染的容器)

③babel

js编译器

1、JSX(推荐使用)

JSX(javaScript+xml)不是一种新的语言,只不过是允许在js中编写标签

注意事项：不允许直接渲染多个元素

jsx语法：

①遇到<,如果首字母是小写，就会用html去解析；否则就会用react按照组件去处理

②遇到{,就会用js来执行代码块

练习:

demo03\_lianxi.html

希望将h2的标签渲染，h2要执行个3目运算

2、Component

Everything is Component!

组件是可被反复使用的，带有特定功能的视图

2.1 review:

Vue.component('my-header',{

template:``

})

.vue

<template></template>

<script></script>

<style></style>

demo01.component.ts

import {Component} from '@angular/core'

@Component({

selector:'',

template/templateUrl:'',

styleUrls:[]

})

export class Demo01Component{}

2.2 React:

①创建组件类：

var MyComponent = React.createClass({

render:function(){

return <h2>...</h2>

}

})

②调用组件类：

<MyComponent></MyComponent>

③注意事项

组件类的名称必须是全驼峰

组件类要渲染的内容 不能直接返回多个元素

组件类在指定要渲染的元素时，第一个标签不能直接换行

2.3 复合组件

在一个组件中 可以调用其他的组件

练习：

demo06\_lianxi.html

创建一个复合组件，显示一个登录窗口

3、Props

在ng和vue中 是可以通过属性进行传值的。

3.1 基本用法

vue:

① <son sonName='zhangsan'></son>

② Vue.component('',{

props:['sonName'],

//this.sonName

})

angular:

①<son sonName='zhangsan'></son>

② import {Input} from '@angular/core'

@Input() sonName="";

this.sonName

react:

①<son sonName='zhangsan'></son>

②this.props.sonName

demo08\_lianxi.html (:- 15:40)

通过props来重构登录窗口

添加功能：在input显示占位符 请输入用户名/请输入密码

3.2 this.props.children

调用组件时所指定的属性 和组件内this.props对应的中的属性是一一对应的，其中有一个是比较特殊的 this.props.children

方法：

React.Children.map(

this.props.children,

function(child){

//

}

)

3.3 子与父的通信

可以成功的实现在子组件中调用父组件的方法

如果父组件的方法是支持传参的，

子组件--》父组件

基本步骤：

①在父组件中 定义一个带有参数的方法

②调用子组件时，将方法通过属性传递子组件

③子组件调用

this.props.func(123);

4、Ref

Vue:

<son ref='mySon'></son>

this.$refs.mySon

React:

<son ref='mySon'></son>

this.refs.mySon

创建AComponent BComponent，在MainComponent调用A和B

A：input

B: button

在点击B组件的按钮 获取A组件中input中的值

4.1组件的创建和调用

4.2完成A组件中输入框的值 传递父组件Main

什么时候传？触发onChange

怎么传？ ①父 定义 方法 ②方法给子 ③儿子 调用

4.3。。

5、State

REACT DAY02

一、state

把组件比作是一个状态机，组件中的数据 通过状态进行管理

3个基本操作

1、初始化

getInitialState:function(){

return {

myState1:true,

myState2:false}

}

2、获取

this.state.myState1

3、修改

this.setState(

{myState1:false},

function(){

//状态设置成功之后的回调函数

}

)

状态的功能：

①管理数据 （3个基本操作）

②将数据绑定到视图

练习：(:- 11:20)

将综合练习中Main组件的数据的处理 按照状态来进行处理

React的5大核心:

①jsx 可以帮助更好的封装组件类

②component 构建ui

③props 完成父子之间相互的通信

④ref 通过引用找到组件实例或者元素本身

⑤state 管理数据、完成数据到视图的绑定

二、lifecycle

Vue:

create/mount/update/destroy

beforeCreate

created

Angular:

OnInit OnDestroy

React:

mount/update/unmount

①挂载

componentWillMount

componentDidMount

②更新

componentWillUpdate(nextProps,nextState)

componentDidUpdate(prevProps,prevState)

componentWillReceiveProps

componentDidReceiveProps

③移除

componentWillUnmount

注意事项：

①this.refs.\*\*获取元素或者组件实例，必须等到componentDidMount之后

②调用组件时通过属性传来的值发生变化时，或者组件内部的状态发生变化时，才会执行和update相关的处理函数

③和update相关的处理函数 是有参数的！

练习：创建一个组件，该组件渲染一个H1标签（:- 14:30）

要求在组件挂载完毕之后，启动定时器来修改透明度 (0 0.1 0.2 --1 0 0.1)

分析：

①componentDidMount

②状态的初始化、读、写

<h1 style={{opacity:this.state.\*\*}}>

</h1>

注意：

jsx在编写html标签时候，并不是支持html所有的属性

style {{}}

class --> className

for --> htmlFor

三、表单

给表单元素 指定了value/checked/selected等相关属性，表单元素称之为受控表达元素。

反之，就是非受控表单元素

解决方案：

方案1： value --> defaultValue

方案2：

①初始化状态

②将状态的值绑定value属性

③onChange事件处理函数中 修改状态

jsx/component/props/ref/state/lifecycle

注意事项：通过属性传递给子组件的数据，在子组件中是只读的，是不允许修改的。

四、综合练习

1、三连棋

GameInit.html

①一共有3个组件

Game

Board 调用了Square9次

Square(就是一个按钮)

②点击Square组件需要操作

切换用户 X->O->X

保存棋盘信息 数组

(:- 16:20)

③在Board组件中定义一个方法，来修改状态中数据(xIsNext 取反/squares 在指定下标保存x或者o)；

然后将方法传递给Square组件，当按钮按下时来调用该方法

④判断输赢

在哪个组件中判断输赢

Board/Square

this.calculateWinner()

if( this.state.squares.indexOf(null) == -1)

2、ToDoBox

组件树:

ToDoBox = ToDoInput + ToDoList(多个ToDoItem)

步骤1：完成组件的创建和调用

步骤2：实现add

①到ToDoBox组件完成状态的初始化：数组

②子与父的通信(todoinput-->todobox)

第一步：在父组件中 定义一个带有参数的方法

第二步：把这个方法通过属性传递给子组件

第三步:子组件中点击add，调用通过属性传递来的方法同时完成参数的传递

③todobox将状态传递给todolist，动态创建多个todoitem

步骤3：实现delete

REACT DAY03

一、ReactNative概述

1、移动端开发的常见的3种方式

①WebApp

使用前端技术来编写的，可以在移动端设备的浏览器中运行的app，提供类似原生app体验

跨平台性 性能不如原生

②NativeApp

(android/ios/wp/bb/..)

采用的SDK（software development kit软件开发包）和java、oc，编写可以安装并运行在移动端设备的app，称之为原生的应用程序

性能好 跨平台没有优势

③HyBridApp

混合编程：使用前端技术结合着原生技术 来生成的可以安装在手机操作系统的app

2、概述

what?

通过js和react，构建移动端的原生的应用程序

why?

①开发成本很低，即使不了解真正的原生开发，也不影响构建nativeApp

②free

③原生应用程序性能很好

where？

前端人员可以使用RN来实现原生移动端的app开发

how?

方案1：

npm install -g create-react-native-app

create-react-native-app my-app

cd my-app/

npm start

方案2：

react-native-cli

3、搭建环境

①完成模拟器的安装和启动

②安装一个app-debug.apk到模拟器

app-debug.apk的作用：

①配置要连接的开发服务器的ip地址和端口号

windows+R -> cmd -> ipconfig

172.163.100.139

②设置手工刷新、热加载

③可以允许在浏览器console中去查看控制台的输出信息

③启动pc端的开发服务器

①准备一个工程模板

②将reactNativeForStu.zip统一放到c/xampp/htdocs/framework/react

解压缩到reactNativeForStu文件夹

③打开vscode，选择打开上述工程

c/xampp/htdocs/framework/react/reactNativeForStu/reactNative/myapp

④打开集成终端，执行npm start

二、ReactNative使用

ReactJS用到了React开发时的五大核心概念，

ReactNative也是使用五大核心概念，结合着js调用由RN所封装的原生组件进行编程，而生成的原生的app。

RN的工作方式：

在RN中所调用的任何一个组件，在对应平台运行时，都会变成该平台下的原生的视图控件。

不论在什么平台上，View都会直接对应一个平台的原生视图，无论它是UIView、<div>还是android.view.View

注意事项：

①如果导出的类，没有加default,引入类加上花括号；否则是无需加上花括号

②在RN中 不允许出现任何的html标签

1、组件的创建和使用

①app/components/demo01/demo01.js

②创建组件

import React,{Component} from 'react';

import {Text} from 'react-native';

export default class Demo01Component extends Component{

render(){

return <Text>Hello RN</Text>

}

}

③使用

找到根组件 index.android.js

import Demo01Component from './app/component/demo01/demo01'

<Demo01Component></Demo01Component>

2、Text

是用来显示一段文本内容

import {Text} from 'react-native'

<Text>hello</Text>

3、View

是用来指定一个容器

import {View} from 'react-native''

<View>

<Text></Text>

<Text></Text>

</View>

4、StyleSheet

import {StyleSheet} from 'react-native'

//创建样式类

const myStyles = StyleSheet.create({

myText:{

}

})

//使用

<Text style={myStyles.myText}></Text>

可以给一个组件类 同时指定多个样式类:

<Text style={[styles.bigblue, styles.red]}></Text>

5、Image

import <Image> from 'react-native'

//加载本地的图片

<Image source={require('imgPath')}></Image>

//加载网络图片

<Image source={{uri:'imgUrl'}}></Image>

注意事项：

通过require加载本地资源图片时，是不允许在路径存在任何的运算！

6、state在RN的使用

读和写和之前是一致

初始化的方式：把数据定义在构造函数中

constructor(){

super();//实现继承的类当中的构造函数

this.state = {

imgName:'1.jpg'

}

}

7、Button

import {Button} from 'react-native';

<Button title='clickMe' onPress={this.handlePress}></Button>

练习：demo05\_lianxi/demo05.js

在组件类中 渲染一个Text、Button，

Text默认显示0，点击Button，数字自增

组件的创建和使用

Text/Button

state

每日一练：

demo06\_lianxi/demo06.js

组件类：

渲染一个Text标签，在组件渲染完毕之后，启动一个定时器，修改透明度（0 0.1 0.2 ~~ 1 0..）

REACT DAY04

https://github.com/enaqx/awesome-react

https://github.com/jondot/awesome-react-native

一、FlatList

import {FlatList} from 'react-native';

//data指定数据源

//renderItem指定渲染列表项的方法，返回要渲染元素内容

<FlatList data={} renderItem={}></FlatList>

注意事项：

1、如果在渲染列表项目，有警告提示key的问题，解决方案：在数据源每一个数据添加上一个key属性，而且key保证唯一性

2、renderItem对应的处理函数是有参数的，这个参数是一个对象，对象中包含两个属性：item(对应的是遍历的集合中临时变量)\index(下标),还必须有返回值

二、TextInput

import {TextInput} from 'react-native'

<TextInput

secureTextEntry={true}

placeholder='plz input'

onChangeText={(msg)=>{}}

keyboardType=''

></TextInput>

练习demo09\_lianxi/demo09.js

创建一个组件类，组件类中有一个TextInput

要求：默认显示的值是zhangsan，按照受控表单元素的方式来处理

TextInput基本用法

状态的基本操作

三、TouchableOpacity

功能：将组件中调用的元素 添加上透明度渐变的效果

import {TouchableOpacity} from 'react-native'

<TouchalbelOpacity>

<Text></Text>

</TouchableOpacity>

四、switch

Android：Google所提倡Material Design (MD)

iOS: Apple 扁平化设计风格

受控表单元素的解决方案：

①初始化状态

②将状态绑定到指定的表单元素

③给表单元素绑定事件，在事件处理函数中完成状态的修改

五、fetch

Angular发起网络请求：

①准备工作

app.module.ts

import {HttpModule} from '@angular/http'

@NgModule({

imports:[HttpModule]

})

②服务

import {Injectable} from '@angular/core'

import {Http,Response} from '@angular/http'

@Injectable

export class MyHttpService{

constructor(private http:Http){}

sendRequest(url:string){

return this.http.get(url).map((response:Response)=>{

return response.json();

})

}

}

③指定providers

app.module.ts

import {MyHttpService} from '..'

@NgModule({

providers:[MyHttpService]

})

④component中调用服务

import {MyHttpService} from '\*\*'

constructor(myHS:MyHttpService){}

this.myHS.sendRequest(url).subscribe((result)=>{})

Vue发起网络请求：

①import VueResource from 'vue-resource'

②Vue.use(VueResource);

③this.$http.get().then(()=>{})

React中fetch

json格式的序列化:变成json格式的字符串

json格式的反序列化：将json格式的字符串变成对象或者数组

fetch("serverurl")

.then((response)=>{return response.json()})

.then((result)=>{

//result就是服务器端返回的真正的结果

})

练习：demo13\_lianxi/demo13.js

(:- 15:35)

在demo13.js中 创建一个组件，在组件挂载完毕3s之后，向服务器端发请求(http://172.163.100.139/ajia\_code/data/product/list.php)，将服务器端返回的数据显示在FlatList（在列表项中显示title到Text中）

fetch FlatList lifeCycle state

六、ScrollView

功能：让组件支持滚动效果

用法:

import {ScrollView} from 'react-native';

<ScrollView>

...

...

</ScrollView>

Text View Image TextInput Button Switch

FlatList ScrollView TouchableOpacity

fetch

StyleSheet

七、Flexbox

flexDirection 主轴是row/column 默认是column

alignItems 沿着交叉轴的对齐方式

justifyContent 沿着主轴的对齐方式

八、Navigation

路由模块的本质 是建立url和组件映射关系

Angular

1、基本用法

准备工作

①创建自定义路由模块app.router.ts

import {Routes,RouterModule} from '@angular/router'

const myRoutes:Routes = [

{path:'',component:\*\*}

]

@NgModule({

imports:[RouterModule.forRoot()],

exports:[]

})

export class AppRoutingModule{}

②到跟模块中指定依赖于路由模块app.module.ts

import {AppRoutingModule} from './app.router'

@NgModule({

imports:[AppRoutingModule]

})

指定容器 router-outlet

2、跳转方式

①js

import {Router} from '@angular/router'

constuctor(private myRouter:Router){}

this.myRouter.navigateByUrl('/login')

②routerLink

<any routerLink='/login'></any>

3、路由传参方式

①明确发送方和接收方

②配置接收方的路由地址、接受参数

path:'detail'

path:'detail/:id'

接受参数:

import {ActivatedRoute} from '@angular/router'

constructor(private myAR:ActivatedRoute){}

this.myAR.params.subscribe((result)=>{})

③发送

4、路由嵌套

①指定容器

router-outlet

②配置子路由

{

path:'mail',

component:MailComponent,

children:[

{},{}]

}

Vue

1、基本用法

<script src='js/vue-router.js'>

指定容器 router-view

new Vue({

router:new VueRouter({

routes:[

{path:'',component:\*\*}

]

})

})

2、跳转方式

①js

this.$router.push('/login')

②routerLink

<routerLink to='/login'></routerLink>

3、路由传参方式

①明确发送方和接收方

②配置接收方的路由地址、接受参数

path:'/detail'

path:'/detail/:id'

this.$route.params

③发送

4、路由嵌套

①指定容器

router-view

②配置子路由

{

path:'mail',

component:MailComponent,

children:[

{},{}]

}

ReactNative实现导航，第三方的项目ReactNavigation

基本使用：

①安装

npm install react-navigation

②创建要用到的组件

③配置路由

import {StackNavigator} from 'react-navigation'

import Demo16CartComponent from ''

const myNavigator = StackNavigator({

myCart:{

screen:Demo16Component

}

})

AppRegistry.registerComponent('myapp', () => myNavigator);

注意事项：

①引入组件类 花括号

②保证js调试正常打开

③重启开发服务器

④测试

REACT DAY05

练习：

demo17\_lianxi/

有一个RN项目，在项目中有两个组件，分别是

myLogin.js(MyLoginComponent)

myMain.js(MyMainComponent)

默认访问登录页

每一个组件就显示一个Text

一、跳转和传参

angular

import {Router} from '@angular/router'

constrcutor(private myRouter:Router){}

this.myRouter.navigateByUrl();

vue

this.$router.push('/myLogin');

React-Navigation:

//基本用法

①安装

npm install react-navigation

②创建要用到的组件类

③配置路由信息

import {StackNavigator} from 'react-navigation'

var myNavigator = StackNavigator({

login:{

screen:LoginComponent

}

})

AppRegistry.registerComponent('myapp',()=>myNavigator)

//跳转方式

this.props.navigation

.navigate('routeName')

//跳转并传递参数

①明确发送方和接收方

②发送

this.props.navigation

.navigate(

'routeName',{price:100})

③接受

this.props.navigation.state.params.id

练习(:- 11:30)：

demo19\_lianxi/

demo19\_check.js/demo19\_pay.js/demo19\_send.js

check:button 点击跳转到pay,同时传递商品价格100给pay

pay: Text 显示商品价格信息

button 点击跳转send

send: button 点击跳转到check

步骤1:搭建架子，要求通过不同的url能够访问3个组件

步骤2：搞定跳转

步骤3：搞定传参

①明确发送方和接收方

check（发送） pay(接受)

②pay接受数据

③跳转的同时发送

二、ActivityIndicator

基本用法:

import {ActivityIndicator} from 'react-native'

<ActivityIndicator></ActivityIndicator>

练习：

demo21\_lianxi/demo21.js

在这个组件类Demo21Component,渲染一个Button,ActivityIndicator

要求：默认并不显示ActivityIndicator,点击按钮，才显示。

&&

扩充：

实现选择功能

{

this.state.isLoading

&&

<Any></Any>

}

实现循环功能

{

this.state.list

.map(function(child){

return <Text>{child}</Text>

})

}

三、FlatList 进阶

onEndReached={()=>{}}

onEndReachedThreshold={0.5}

Text View Image StyleSheet FlatList

TextInput Button Switch

TouchableOpacity

ScrollView..

fetch

flexbox

demo24\_lianxi/todobox.js todolist.js todoinput.js todoitem.js

通过ReactNative所提供的组件来实现 ToDoBox、ToDoList、ToDoInput、ToDoItem

1、完成4大组件的创建和调用

2、添加 （:- 17\_20）

①将todoinput中输入框的数据 点击add，发送给todobox

父组件中定义一个带有参数的方法

方法传递给子

子调用方法传值

②todobox将todoinput传来的数据保存在状态中（数组)

③todobox调用todolist将状态中数据传递给todolist，todolist接受到之后 渲染多个todoitem

3、删除

①todolist在调用todoitem要指定要显示的内容

②实现删除功能

REACT DAY06

REACT DAY07

项目

1、准备工作

启动apache和mysql

提醒：将react-navigation安装好之后要记得做两件事：①重启开发服务器 ②保证模拟器myapp应用程序已经开启远程js调试

2、搭建项目架子

①在app目录下创建components目录，在components目录中，创建project目录

②在project目录中创建login.js main.js productList.js三个文件

③在index.android.js中配置路由词典，让默认访问login.js

3、实现具体页面需求

login

http://172.163.100.139/admin/data/user/login.php

touchableOpacity

跳转

productList:

http://172.163.100.139/admin/data/product\_list.php

①进到该页面，就能够显示商品列表信息

②商品列表项支持点击行为：弹窗显示该商品已售卖的个数

③解决警告问题

④实现商品列表 上拉加载更多的效果！

（:- 10:30）

onEndReached

⑤点击列表项，跳转detail，同时加载该商品的详情信息

①创建detail.js ，配置路由

②跳转完成参数传递

明确发送方(list)和接收方(detail)

发送 this.props.navigation

.navigate('detail',{id:2})

接受 this.props.navigation.state

.params.id

③在detail.js中 根据传递来的参数来获取该商品的详情信息

http://172.163.100.139/admin/data/product/details.php?lid=3

商品的图片（每隔1s，实现轮询播放）

标题

副标题

五大核心概念：

1、jsx:帮助更好来封装组件

2、component:①自己创建的组件 ②ReactNative所封装的组件

3、props

父与子之间的组件间通信

4、ref

帮助得到子组件类的实例

<Son ref='mySon'></Son>

this.refs.mySon

5、state

数据的管理

将数据绑定视图

learn once,write anywhere

如果要将一个组件固定在底部:

<View style={{flex:1}}>

<ScrollView></ScrollView>

<Button></Button>

</View>

将行和列的信息 直接封装对应的组件类:

row.js -> Row

col.js -> Col

//this.props.children

<Row>

<Button></Button>

</Row>

//将main.js所有的行和列 替换row/col

<Text style={

[styles.red, styles.bigblue]}>

</Text>

搭建路由:

使用reactNavigation所提供的stackNavigator来实现导航

①安装

②配置

import {StackNavigator} from 'react-native';

const myNavigator = StackNavigator({

login:{

screen:LoginComponent

}

})

③让路由生效

//ReactDOM.render

AppRegistry.registerComponent('myapp',()=>myNavigator)

Login:

Image 圆形? borderRadius

TextInput onChangeText

State:

①初始化

constructor(){

super();

this.state = {};

}

②读写

this.state.isLoading

this.setState();

Button onPress

fetch().then((response)=>response.json()).then((result)=>{})

跳转:

this.props.navigation.navigate();

选择判断功能的实现:

{

this.state.isLoading

&&

<ActivityIndicator> </ActivityIndicator>

}

循环功能：

this.state.list.map(function(msg){

return <View>

...

</View>

})

Main:

flexbox常见3属性:

flexDirection:'row/column'

justifyContent 主轴

alignItems 交叉轴

StyleSheet:

①创建

import {StyleSheet} from 'react-native'

const myStyles = StyleSheet.create({

myText:{fontSize:20},

myRed:{color:'red'}

})

②使用

②使用

<Any style={myStyles.myText}></Any>

<Any style={[myStyles.myText,myStyles.myRed]}></Any>

Props:

//遍历组件调用时内部子元素

React.Children.map(this.props.children,function(child){

return child

})

ProductList:

TouchableOpacity:

作为一个容器来使用，容器中元素在被点击会有透明度变化的效果 onPress

FlatList:

data 指定数据源

renderItem={(info)=>{

return <Text>{info.item}</Text>

}}

onEndReached={this.loadMore}

onEndReachedThreshold={0.5}

extraData={this.state}

Image:

加载本地图片

<Image source={require('../img/1.jpg')}></Image>

加载网络资源图片

<Image source={{uri:''}}></Image>

Detail:

将元素固定在页面底部:

<View style={{flex:1}}>

<ScrollView></ScrollView>

<Button></Button>

</View>

组件销毁时执行清理工作:

组件的生命周期处理函数:

componentWillMount

componentDidMount

componentWillUpdate

componentDidUpdate

componentWillReceiveProps

componentDidReceiveProps

componentWillUnmount

WEBAPP DAY01

WEBAPP

Ionic

一、基本介绍

1. WEBAPP = WEB + APP (Application 应用程序)

使用 WEB 技术开发移动设备的应用程序

2. 开发移动设备的应用程序

2.1 Native 原生开发

Android Java Android studio

iOS Objective-C Xcode

2.2 移动适配网站，m 站

2.3 React Native

2.4 Ionic

3. WEB 技术

3.1 HTML

3.2 CSS

3.3 JavaScript

3.4 Angular

3.5 TypeScript

3.6 Ionic API

4. Ionic

4.1 https://ionicframework.com/

5. Build amazing apps in one codebase, for any platform, with the web.

5.1 WEB 技术

5.2 天生跨平台

5.3 Components 组件，样式风格贴近于原生

5.4 社区支持好

5.5 强大的 CLI 命令行工具

6. 环境配置

6.1 升级 Node.js 为最新版本 (V8.10.0)

卸载 Node.js

安装 Node.js

验证 node -v

npm -v

6.2 Node.js 安装的第三方全局模块

npm i -g <第三方模块>

位置：C:/Users/用户名/AppData/Roaming/npm/node\_module

注意：AppData 是隐藏目录

6.3 有网络：cmd 执行命令：

npm i -g ionic cordova

6.4 无网络：

a. 删除原有的 ionic cordova

b. 解压缩 npm.zip

c. 复制 ionic ionic.cmd cordova cordova.cmd 到 .../npm/ 下

d. 复制 ionic 和 cordova 目录到 .../node\_modules/ 下

e. 验证：

新开 cmd：

ionic -v 3.19.1

cordova -v 8.0.0

7. 命令行程序的执行：

7.1 命令的作用是什么

7.2 命名在哪里执行

7.3 命令的具体写法

7.4 命令的结果

8. Ionic Hello world

8.1 创建项目 Ionic-1710N

8.2 在 Ionic-1710N/ 下执行 cmd：ionic start app-components blank

8.3 无网络：解压缩 app-components.zip 到 Ionic-1710N/ 下

8.4 运行，app-components/ 执行命令: ionic serve -l

9. Ionic 在代码更新后，自动编译，刷新浏览器页面（注意：这个特性不稳定）

需要重启：app-components/ 下

命令行执行 Ctrl + C 关闭当前服务

ionic serve -l 启动 (l:lab)

二、Ionic 的基本 UI 组件

10. Button 按钮

10.1 <button ion-button>default button</button>

10.2 按钮背景

10.3 外边框

10.4 无边框

10.5 圆角

10.6 有圆角块级

10.7 无圆角块级

10.8 按钮大小

10.10 图标按钮

11. Ionic 创建新的页面，在 app-components/ 下执行命令：

ionic g page button

12. Ionic 的页面跳转

12.1 修改 src/app/app.module.ts 文件，注册新页面

a. 引入新创建的页面

import {ButtonPage} from '../pages/button/button';

b. 在 declaration 数组里，添加元素 ButtonPage

c. 在 entryComponents 数组里，添加元素 ButtonPage

12.2 修改 src/pages/home/home.ts 文件

a. 引入新创建的页面

import {ButtonPage} from '../button/button'

b. 定义类属性：buttonPage;

c. 在构造器中初始化 buttonPage

this.buttonPage = ButtonPage;

12.3 修改 src/pages/home/home.html 文件

a. 创建一个 button

b. 为 button 做属性绑定

[navPush]="buttonPage"

13. ionic serve -l 启动后报错，网络问题

解决方式：ionic serve

14. List 列表

14.1 创建了 list 新页面，并且从 HomePage 跳向 ListPage

14.2 基本列表

14.3 按钮作为列表项

14.4 列表边距

14.5 无分割线列表

14.6 列表分隔项

14.7 列表头部

14.8 图标作为列表项

14.9 头像作为列表项

14.10 滑动列表项

14.11 列表的数据来自于控制器

a. 在 list.ts 文件中，添加一些样本数据

b. 在列表中做循环 \*ngFor="let course of courses"

c. 插值 {{course}}

d. 事件响应，事件绑定，定义函数

15. 修改 package.json 文件，添加 Ionic 的启动命令

"scripts": {

...,

"start": "ionic serve -l"

}

npm [run] start

16. Card 卡片

16.1 创建新的页面 card，从 HomePage 跳转到 CardPage

16.2 基本卡片

16.3 卡片内容为列表

16.4 图片卡片

17. Gesture 手势动作

17.1 创建新的页面 gesture，从 HomePage 跳转到 GesturePage

17.2 tap

17.3 press

17.4 pan

17.5 swipe

练习：页面 A 内部实现一个滑动列表，点击滑动列表中 item-options 跳向页面 B，

B 页面内实现一个图片卡片

WEBAPP DAY02

WEBAPP

Ionic Day02

1. 页面 A 内部实现一个滑动列表，点击滑动列表中 item-options 跳向页面 B，

B 页面内实现一个图片卡片

1.1 创建页面 A 和 B

1.2 从 HomePage 跳向 APage

1.3 APage 中实现滑动列表

1.4 BPage 中实现图片卡片

1.5 从 APage 跳向 BPage

2. FAB 浮动操作按钮 Floating Action Button

2.1 创建 fabPage，从 HomePage 跳向 FabPage

2.2 基本

2.3 居中

3. grid 栅格系统

3.1 创建 grid，从 HomePage 跳向 GridPage

3.2 基础等分

3.3 指定宽度

3.4 自动宽度

3.5 列偏移

3.6 垂直对齐

4. form 表单

4.1 创建 form，从 HomePage 跳向 FormPage

4.2 文本输入框

4.3 密码输入框

4.4 复选框

4.5 日期输入框

4.6 单选按钮

4.7 下拉列表

4.8 滑块

4.9 开关

4.10 提交按钮

5. Toast 提示 吐司

5.1 创建 toast，从 HomePage 跳向 ToastPage

5.2 在 toast.html 中，添加按钮，绑定 tap 事件

5.3 在 toast.ts 中，定义函数 basicToast()

5.4 import ToastController 控制器

5.5 在构造器 constructor 中，定义参数 toastCtrl: ToastController

5.6 在 basitToast 函数中，创建一个 toast 实例

5.7 调用 toast.present()，显示消息提示

6. 练习：创建一个用户登录页面 signIn

页面内实现一个登录表单，Email 和 Password

点击提交按钮显示消息提示：Invalid Email or password.

7. alert 提示框

7.1 创建 alert，从 HomePage 跳向 AlertPage

7.2 在 alert.html 中，创建按钮，绑定 tap 事件

7.3 在 alert.ts 中，定义函数 basicAlert();

7.4 引入 AlertController

7.5 DI 依赖注入 AlertController

7.6 在 basicAlert 中，创建 alert 实例

7.7 alert.present()，显示警告框

8. icon 字体图标

8.1 创建 icon，从 HomePage 跳向 IconPage

8.2 字体属性

8.3 ios md

8.4 属性绑定

9. Ionic 3 新特性：lazy loading 懒加载

9.1 创建 lazy 页面

9.2 修改 home.ts 文件，定义函数:

toLazyPage():void {

this.navCtrl.push('LazyPage');

}

9.3 修改 home.html 文件，添加按钮，绑定 tap 事件

10. 练习：把之前的所有页面改写为 lazy loading

11. 解压 app-components.zip，改目录名为 app-demo，放在 Ionic-1710N/ 下

12. 解压 server-demo.zip，放在 Ionic-1710N/ 下

13. Ionic 配合 Node.js 实现用户注册、登录、注销简单的系统

14. 先实现客户端首页 HomePage，添加登录表单和注册按钮

15. 创建 signUp 页面，点 Homepage 的注册按钮实现跳转

16. 在 signUp 页面中，添加注册表单

17. 创建文件 server-demo/sql/db.sql，建库建表

18. 使用 Node.js 和 Express 实现一个简单的服务器端

19. 创建目录 server-demo

20. 在 server-demo/ 执行 cmd：npm init 生成 package.json

21. 在 server-demo/ 执行 cmd：npm i -S express mysql body-parser

22. 在 server-demo/ 创建 app.js

23. 练习：实现客户端页面，建库建表，实现简单的服务器端

WEBAPP DAY03

WEBAPP

Ionic Day03

1. 在客户端获取用户填写的注册表单值

2. 修改 sign-up.ts 文件，声明属性 user (JS 对象)

user = {

email:'',

password: ''

};

3. 修改 sign-up.html 文件，做表单数据的双向绑定 two way binding

[(ngModel)] = 'user.email'

4. 修改 sign-up.html 文件，为提交按钮绑定 tap 事件

5. 修改 sign-up.ts 文件，定义函数 signUp()，验证表单值

6. 从 Ionic 客户端发送注册用户的 HTTP 请求

7. 修改 app.module.ts 文件

7.1 加载 HttpClientModule 模块

import {HttpClientModule} from '@angular/common/http';

7.2 把 HttpClientModule 加入数组 imports

8. 修改 sign-up.ts 文件

8.1 加载 HttpClient

import {HttpClient} from '@angular/common/http';

8.2 依赖注入 HttpClient

constructor(..., private httpClient: HttpClient){}

8.3 使用 this.httpClient 向服务器端发送各种请求

this.httpClient.get('/signUp')

.subscribe(

res => {

console.log(res);

},

err => {}

);

9. 修改 app.js 文件

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

// ...

res.send({"status": "ok"});

});

10. Ionic 跨域问题

修改 app-demo/ionic.config.json 文件

添加代理 proxy

,

"proxies": [

{

"path": "/signUp",

"proxyUrl": "http://127.0.0.1:3000/signUp"

}

]

重启 Ionic 客户端

11. 以 get 方式传递参数到服务器端

在 sign-up.ts 中，引入 HttpParams

import {HttpParams} from '@angular/common/http';

let params = new HttpParams().set('key',value);

this.httpClient.get(url, {params})

.subscribe(

// ...

);

12. 在 app.js 中，接收表单数据

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

let email = req.query.email;

let password = req.query.password;

console.log(`email: ${email}; password: ${password}`);

res.send({"status": "ok"});

});

13. 以 post 方式发送注册请求

this.httpClient.post(url, {email: this.user.email, password: this.user.password})

.subscribe(

// ...

);

14. 在服务器中获取 post 请求参数时，需要配置 body-parser 中间件

app.js

a. 引入 body-parser

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

b. 配置中间件

app.use(bodyParser.json());

c. let value = req.body.key;

15. 实现服务器端热部署

cmd：npm i -g supervisor

关闭以前的服务

再次启动服务：

app-server/ 下，cmd：supervisor app

16. 在 app.js 中处理用户注册请求

16.1 const mysql = require('mysql');

16.2 let pool = mysql.createPool({user: 'root'});

16.3 let sql = 'INSERT INTO db.user VALUE(NULL, ?, ?)';

16.4 pool.query(sql, [email, password], (err, results) => {

if(err) throw err;

if(results.affectedRows === 1) {

res.send({"status": "ok"});

} else{

res.send({"status": "err"});

}

});

17. 在 sign-up.ts 中，对响应结果进行解析

17.1 let status = res['status'];

17.2 status === 'ok',页面跳回到 HomePage

this.navCtrl.push('HomePage'); // 参见 18 步操作

17.3 status === 'err'，使用 toast 提示服务器错误

18. 把 HomePage 改为 lazy loading

18.1 创建 home/home.module.ts

import { NgModule } from '@angular/core';

import { IonicPageModule } from 'ionic-angular';

import { HomePage } from './home';

@NgModule({

declarations: [

HomePage,

],

imports: [

IonicPageModule.forChild(HomePage),

],

})

export class HomePageModule {}

18.2 修改 home.ts

import {IonicPage, NavController } from 'ionic-angular';

@IonicPage()

18.3 修改 app.module.ts

删除 HomePage 的引入

以及两个数组的 HomePage 元素

18.4 修改 app.component.ts

删除 HomePage 的引入

rootPage:any = 'HomePage';

19. 在 app.js 中，用户注册时，先做验证，验证用户的邮箱是否存在

如果存在，返回客户端消息，alert 提示用户邮箱存在，不能使用

如果不存在，继续注册，成功后页面跳转回 HomePage

20. 用户登录

20.1 home.ts 定义 user = {email:'',password:''};

20.2 home.html 数据双向绑定

20.3 为登录按钮添加 tap 事件响应

20.4 home.ts 中定义函数 signIn();

20.5 home.ts 引入 HttpClient

20.6 home.ts 依赖注入 HttpClient

20.7 ionic.config.json，设置代理

20.8 home.ts signIn 中，发送登录请求

20.9 app.js 中，接受登录请求

20.10 app.js 中，获取表单数据

20.11 app.js 中，处理用户登录请求，返回对应的响应

20.12 home.ts 中，接收响应结果，根据不同的结果进行处理

20.13 创建页面 index，用户登录成功时，跳向 IndexPage

21. 练习：实现用户的注册和登录，注销？

FTP Day03/

app-project.zip

server-project.zip

WEBAPP DAY04

WEBAPP

Ionic Day04

1. 解压 app-project.zip，放在 Ionic-1710N/ 下

app-project 是基于 Ionic tabs 模板创建的空项目

cmd：ionic start app-project tabs

cmd: cd app-project

cmd: npm i

cmd: ionic serve -l

2. 为 app-project 添加一个新的 tab 页

2.1 创建一个新的页面 test

app-project/ cmd: ionic g page test

2.2 tabs.ts

a. import {TestPage} from '../test/test';

b. tab4Root = TestPage;

2.3 tabs.html

<ion-tab [root]="tab4Root" title="test page" tabIcon="heart"></ion-tab>

2.4 app.module.ts

a. import {TestPage} from '../pages/test/test';

b. declarations 添加 TestPage

c. entryComponents 添加 TestPage

3. test 的 tab 页实现 lazy loading

3.1 app.module.ts

删除 import 和两个数组的元素

3.2 tabs.ts

a. 删除 import

b. tab4Root = 'TestPage';

4. 删除 pages/ home contact about test 目录

5. tabs.ts

a. 删除 import home contact about

b. 删除 tab1Root - tab4Root 属性

6. tabs.html 删除所有的 ion-tab 标记

7. app.module.ts

删除 home contact about 的 import 和 元素

8. 创建五个页面

home(首页)

find(发现)

cat(分类)

cart(购物车)

user(个人)

app-project/ 下 cmd：ionic g page

9. 把五个页面设置为 tab 页

9.1 tabs.ts

声明属性

homePage = 'HomePage';

findPage = 'FindPage';

catPage = 'CatPage';

cartPage = 'CartPage';

userPage = 'UserPage';

9.2 tabs.html

<ion-tab [root]="homePage" tabTitle="首页" tabIcon="home"></ion-tab>

<ion-tab [root]="findPage" tabTitle="发现" tabIcon="search"></ion-tab>

<ion-tab [root]="catPage" tabTitle="分类" tabIcon="list"></ion-tab>

<ion-tab [root]="cartPage" tabTitle="购物车" tabIcon="cart"></ion-tab>

<ion-tab [root]="userPage" tabTitle="个人" tabIcon="contact"></ion-tab>

10. https://www.iconfont.cn

11. 解压 server-project.zip，放在 Ionic-1710N/ 下

12. server-project/sql/db.sql，建库建表

13. server-project/ 下创建文件 app.js，设置简单的服务器端

const express = require('express');

let app = new express();

app.listen(3000);

14. server-project/ cmd: node app | supervisor app

15. 下载 default.png，放在 src/assets/avatars/ 下

16. user.html，创建用户登录表单，和一个注册按钮，绑定 tap 事件

17. app-project/ cmd: ionic g page signUp

18. user.ts 中定义函数 toSignUpPage，跳向 SignUpPage

this.navCtrl.push('SignUpPage');

19. sign-up.html 实现注册表单，为注册提交按钮绑定 tap 事件

20. sign-up.ts 定义 user = {email: '', password: ''};

21. sign-up.html 表单数据双向绑定

22. sign-up.ts 定义注册提交函数 signUp()

23. app.module.ts 引入 HttpClientModule，加入 imports

24. sign-up.ts 引入 HttpClient

25. Constructor 中 DI HttpClient

26. ionic.config.json 中，设置注册请求的代理

27. sign-up.ts signUp() 中，发送注册请求，接收，输出响应

28. app.js，接受用户注册请求

29. app.js，配置 body-parser 中间件，解析请求的表单数据

30. app.js，返回一个响应

31. app.js 中，处理注册请求，返回对应的响应

32. 对注册用户的密码进行加密，bcrpytjs

32.1 app.js 引入 bcryptjs

32.2 对密码进行加密

bcryptjs.hashSync(plainPassword,

bcryptjs.genSaltSync(10));

33. sign-up.ts 中，解析返回的响应信息，根据不同的结果做处理

34. 如果用户注册成功了，视为登录成功，此时在 Ionic 客户端保存用户信息

34.1 app.module.ts 中，引入 IonicStorageModule

import {IonicStorageModule} from '@ionic/storage';

加入 imports 数组

IonicStorageModule.forRoot()

34.2 sign-up.ts 中，引入 Storage

import {Storage} from '@ionic/storage';

34.3 Storage 依赖注入构造器

34.4 使用 this.storage 保存用户信息

this.storage.set('key', value);

35. 创建页面 signIn，放置登录表单

36. tabs.ts 做判断：用户是否已经登录了

a. 登录了，user tab 页中显示登录用户的信息

b. 未登录, 跳向登录页面

37. 练习：基于 tabs 的用户注册登录系统

WEBAPP DAY05

WEBAPP

Ionic Day05

1. tabs.ts 做判断：用户是否已经登录了

a. 登录了，user tab 页中显示登录用户的信息

b. 未登录, 跳向登录页面

2. tabs.ts 中，引入 Storage，依赖注入 constructor

a. import {Storage} from '@ionic/storage';

b. private storage: Storage

3. tabs.ts constructor 中，从 storage 中取 user 键对应的值

this.storage.get('user').then(value => {

if(!value) {

this.userPage = 'SignInPage';

}

});

4. user.ts 中

a. 定义 user 属性

b. ionViewDidLoad 中，从 storage 取出 user，赋值给 this.user

c. ionViewDidLoad，Ionic 页面生命周期函数之一

5. user.html 中，显示用户的基本信息：avatar / email

6. user.html 中，为注销绑定事件函数 (tap)="signOut()"

7. user.ts 中，定义 signOut()

this.storage.clear();

this.navCtrl.push('SignInPage');

8. 用户登录

9. sign-in.ts 定义 user

10. sign-in.html 表单数据双向绑定，为登录提交绑定事件

11. sign-in.ts 定义 signIn 函数

12. sign-in.ts 提交登录请求

12.1 ionic.config.json，添加登录代理

12.2 引入 HttpClient，依赖注入

12.3 this.httpClient 发送登录 post 请求，做响应接收

13. app.js 中，添加登录请求的接收

14. 登录请求的处理

15. 返回合适的响应

16. sign-in.ts 中，解析响应结果，做对应的处理

a. 登录成功

b. 登录失败，alert 提示信息

17. 登录成功

17.1 引入 Storage，DI

17.2 app.js 中，用户登录成功时，返回数据库查询到的 user

res.send({

"status": "ok",

"user": results[0]

});

17.3 sign-in.ts 中，在 storage 中设置用户信息

this.storage.set('user', res['user']);

18. 创建用户个人信息页面

app-project/ 下，cmd: ionic g page userInfo

19. user.html 中，为设置按钮添加点击事件

(tap)="toUserInfoPage()";

20. user.ts 中，定义 toUserInfoPage

this.navCtrl.push('UserInfoPage');

21. user-info.html，添加个人信息表单

22. user.ts 中，跳向 UserInfoPage 时传递参数 this.user

this.navCtrl.push('UserInfoPage', {

user: this.uer

});

23. user-info.ts 中，接收参数

23.1 user:any;

23.2 constructor

this.user = navParams.get('user');

24. user-info.html 中，数据双向绑定

25. 练习：实现用户个人信息的设置 // todo

26. Ionic 组件：infinite-scroll 无限滚动

27. home.html 中，添加列表

28. home.ts 中，定义属性 itmes = [];

29. home.ts 中，constructor 中初始化 items

27. home.html 中，添加 ion-infinite-scroll 组件

28. home.ts 中，定义函数 doInfinite()

29. 创建商品表，为商品表添加样本数据

30. 一个商品的视图

31. 在首页里显示商品数据，无限列表展示

32. 练习

32.1 实现用户个人信息的设置

DB：update

32.2 基本的 ion-infinite-scroll

32.3 实现商品展示视图

32.4 实现商品的无限滚动

DB: 分页查询

WEBAPP DAY06

WEBAPP

Ionic Day06

1. 实现用户个人信息的设置

2. user-info.html，为保存按钮添加 tap 事件响应

3. user-info.ts，定义函数 saveUserInfo()

4. user-info.ts，测试表单数据接收

console.log(this.user);

5. user-info.ts，发送保存个人信息的 HTTP 请求

5.1 引入 HttpClient，DI HttpClient

5.2 this.httpClient.post();

6. ionic.config.json，代理设置的简化：

"proxies": [

{

"path": "/",

"proxyUrl": "http://127.0.0.1:3000/"

}

]

7. app.js，用户保存个人信息请求

7.1 接收请求

7.2 处理请求

7.3 返回响应

8.

9. sign-in.ts，去掉 user 的参数传递

this.navCtrl.push('UserPage');

10. user-info.ts，从 storage 中获取用户信息

constructor(..., private storage: Storage) {

this.storage.get('user').then(value => {

this.user = value;

});

}

11. app.js，用户注册成功时，再次查询当前注册用户的信息，返回 user

12. sign-up.ts，把服务器返回的 user 放入 storage

13. 实现商品展示视图

home.html，ion-list>ion-item>(ion-thumbnail>img)+h2+p+button

14. home.html 展示来自于数据库的商品信息，前 20 条记录

sql = 'SELECT \* FROM db.product LIMIT 20 OFFSET 20 \* (page - 1)';

15. home.ts

15.1 声明属性 products

15.2 ionViewDidLoad 发出 HTTP 请求商品表前 20 条记录

16. app.js

16.1 接收 get 请求

16.2 处理请求

16.3 返回响应

17. home.ts

this.products = res;

18. home.html

展示数据

19. 实现商品的无限滚动

20. home.html

添加 ion-infinite-scroll>ion-infinite-scroll-content

为 ion-infinite-scroll 绑定事件 ionInfinite

21. home.ts

21.1 定义函数 loadMoreData

21.2 发送 HTTP 请求商品的第 2 页数据

22. 请求路径参数的设置和获取

22.1 this.httpClient.get('/products/1', ...)...

22.2 app.get('/products/:page', (req, res) => {

const pageSize = 20;

let page = req.params.page;

let sql = `SELECT \*

FROM db.product

LIMIT ${pageSize}

OFFSET ?`;

pool.query(sql, [pageSize \* (page - 1), ...]

});

23. home.ts

定义属性 page:number = 1;

loadMoreData():void {

let url = `/products/${++this.page}`;

}

24. 解决数据全部加载完成后，还继续发送请求的问题

24.1 home.ts

定义属性 hasMoreData: boolean = true;

25. home.html

ion-infinite-scroll 添加属性

\*ngIf="hasMoreData"

26. 练习：实现商品详情页面

PHONEGAP DAY01

Wechat 开发

Day01 公众号开发

1. 官网：https://mp.weixin.qq.com (mp = media platform)

2. 账号分类

2.1 服务号

2.2 订阅号（个人）

2.3 小程序（个人）

2.4 企业号

3. 订阅号

4. 新媒体：博客，微博，微信，音频，视频

传统媒体：报纸，杂志，广播，电视

5. 自媒体：个人身份创办的一种媒体

6. 目的

7. 自媒体：

7.1 选择一个方向

7.2 选择一个平台

8. 注册微信个人订阅号

8.1 点击注册

8.2 选择订阅号

8.3 填写注册表单，输入验证码，提交

8.4 确认选择订阅号

8.5 主题信息登记

8.6 填写账号相关信息

8.7 进入公众号管理页面

9. 登录

10. 设置 - 公众号设置

11. https://iconfinder.com

12. 功能 - 自动回复

12.1 被关注回复

12.2 收到消息回复

12.3 关键词回复

13. 管理 - 素材管理

14. https://unsplash.com

15. 管理 - 用户管理

16. 功能 - 自定义菜单

17. 功能 - 投票管理

18. 统计

19. 以上属于微信公众号的编辑模式

20. 微信公众号的第二种模式 - 开发者模式

21. 编辑模式和开发者模式只能开启某一种

22. 开发者模式可以实现所有编辑模式的功能，同时具有更强大的功能、更灵活的设置

FTP：22-wechat/day01/wechat-1710N.zip

WECHAT DAY01

Wechat 开发

Day01 公众号开发

1. 官网：https://mp.weixin.qq.com (mp = media platform)

2. 账号分类

2.1 服务号

2.2 订阅号（个人）

2.3 小程序（个人）

2.4 企业号

3. 订阅号

4. 新媒体：博客，微博，微信，音频，视频

传统媒体：报纸，杂志，广播，电视

5. 自媒体：个人身份创办的一种媒体

6. 目的

7. 自媒体：

7.1 选择一个方向

7.2 选择一个平台

8. 注册微信个人订阅号

8.1 点击注册

8.2 选择订阅号

8.3 填写注册表单，输入验证码，提交

8.4 确认选择订阅号

8.5 主题信息登记

8.6 填写账号相关信息

8.7 进入公众号管理页面

9. 登录

10. 设置 - 公众号设置

11. https://iconfinder.com

12. 功能 - 自动回复

12.1 被关注回复

12.2 收到消息回复

12.3 关键词回复

13. 管理 - 素材管理

14. https://unsplash.com

15. 管理 - 用户管理

16. 功能 - 自定义菜单

17. 功能 - 投票管理

18. 统计

19. 以上属于微信公众号的编辑模式

20. 微信公众号的第二种模式 - 开发者模式

21. 编辑模式和开发者模式只能开启某一种

22. 开发者模式可以实现所有编辑模式的功能，同时具有更强大的功能、更灵活的设置

FTP：22-wechat/day01/wechat-1710N.zip

WECHAT DAY02

Wechat 开发

Day02 公众号开发

1. 开发者模式

2. 使用 Node.js 和 Express 搭建一个简单的服务器

2.1 创建目录 Wechat-1710N

2.2 Wechat-1710N/ 创建目录

server-official-account

2.2 server-official-account/

cmd: npm init

3. server-official-account/

cmd: npm i -S express

—— 以上操作如无网络，下载解压：FTP/22-wechat/day01/wechat-1710N.zip

4. server-official-account/

创建 app.js，搭建一个简单的 web 服务器

const express = require('express');

let app = new express();

app.listen(3000);

5. server-official-account/

cmd: node app | supervisor app

supervisor: npm i -g supervisor

6. 隧道服务

cmd: npm i -g ngrok

ngrok http 3000

https://ngrok.com/download

C:/ngrok.exe

C:/ cmd: ngrok.exe http 3000

MAC

https://ngrok.com/download

ngrok.zip

ngrok

~/Download/

./ngrok http 3000

7. server-official-account/

cmd: npm i -S wechat

8. app.js

const wechat = require('wechat');

let config = {

appid: '',

token: '',

encodingAESKey: ''

};

9. 微信公众平台 - 登录个人订阅号账号

开发 - 基本配置

复制 AppId，粘贴为 config 的 appid 值

10. 开发 - 基本配置

服务器配置(未启用) - 修改配置

10.1 token: weixin

设置 config 的 token 值为 weixin

10.2 生成 EncodingAESKey

设置 config 的 encodingAESKey 值为生成值

11. 复制 ngrok 生成的隧道地址，填写到公众平台的 URL 处

12. app.js

app.post('/', wechat(config, (req, res) => {

let message = req.weixin;

console.log(message);

}));

13. 开发 - 基本配置

服务器配置(未启用) - 修改配置

提交

14. 启用开发者模式

服务器配置(已启用)

15. 手机微信里，向自己的订阅号发送任意消息，观察服务器输出

16. 如果订阅号服务器在接收到消息后，没有返回响应，微信服务器会再重发两次消息，最后向用户发送消息：该公众号提供的服务出现故障，请稍后再试

17. 返回响应

res.reply('text');

18. 模拟编辑模式的关键词半匹配回复？

-> JavaScript | HTML | CSS

-> http://...

19. 聊天机器人

用户向订阅号发送消息

服务器端接受消息

服务器根据接受的消息返回响应

20. server-official-account/

cmd: npm i -S mysql

21. server-official-account/

创建文件 sql/db.sql

22. app.js

const mysql = require('mysql');

let pool = mysql.createPool({user: 'root'});

23. let sql = 'SELECT \* FROM db.chat WHERE ? like concat('%', question, '%')';

pool.query(sql, [content], (err, results)=>{// ...});

WECHAT DAY03

Wechat 开发

Day03 小程序开发

1. 微信小程序

1.1 是嵌入在手机微信应用中的小应用

1.2 不需要独立安装

1.3 天生跨平台

1.4 用户量 9 亿

2. 官网：https://mp.weixin.qq.com

注册小程序开发者账号

2.1 首页 - 注册

2.2 选择小程序

2.3 邮箱

2.4 进入邮箱，激活小程序开发账号

2.5 主体信息登记

3. 设置 - 开发设置

获取开发者ID - AppID

4. 首页 - 普通小程序开发者工具 - 微信开发者工具 - windows 64 、 windows 32 、 mac

5. 设置小程序基本信息

6. 安装小程序开发者工具

7. 创建小程序项目

7.1 创建目录 Wechat-1710N/wechat-app

7.2 指定项目目录

7.3 填写 AppID

7.4 填写项目名

7.5 选择建立普通快速启动模板

8. 小程序的项目结构

\* pages/ 目录：小程序的所有页面

index/ 目录：index 页面的所有文件

\* index.js: index 页面的业务逻辑（JS）

\* index.wxml: index 页面的内容 （html）

index.wxss: index 页面的样式 （css）

index.json: index 页面的设置

\* app.js 文件：小程序的业务逻辑

\* app.json 文件：小程序的公共设置

app.wxss 文件：小程序公共样式

project.config.json 文件：项目的开发设置

9. 练习：创建一个新的页面 test

9.1 pages 右键-新建-目录：test

9.2 test/ 右键-新建：test.js \*

9.3 test/ 右键-新建：test.wxml \*

9.4 test/ 右键-新建：test.wxss

9.5 test/ 右键-新建：test.json

9.6 app.json

"pages": [

...,

"pages/test/test"

]

10. 项目的首页是 app.json 中，pages 数组的第一个元素

11. 快速创建页面的方法（推荐）

app.json

添加 pages 数组的元素

"pages": [

...,

"pages/demo/demo"

]

保存 app.json

12. format: Shift + Alt + F

13. <text></text>

HTML 中没有这个标记

微信小程序新的标记

\*.wxml

\*.wxss

14. 提示：编译 .wxss 文件错误 ...

解决：

a. 登录用户目录/AppData/Local/微信开发者工具/User Data/WeappVendor

b. 删除 wcsc.exe

c. 重启开发者工具

15. 小程序页面之间的链接跳转

15.1 index.wxml

<navigator>to demo page</navigator>

15.2 为 navigator 添加属性 url='../demo/demo'

15.3 为 navigator 添加属性 open-type='redirect' 重定向

16. 数据绑定

16.1 demo.wxml

Hello，{{name}}！

16.2 demo.js

Page({

data: {

name: 'weapp'

}

});

17. 事件处理

17.1 demo.wxml

<button type='primary'>button</button>

17.2 添加属性 bindtap='testTap'

17.3 demo.js

testTap: function() {

console.log('test tap...');

}

18. 点击按钮，改变 text 中的文本内容

function() {

this.setData({

key: 'new value'

});

}

19. 获取 data 值

this.data.key

20. 新建页面 weather，设置为首页

21. weather.wxml

<text>{{weather}}</text>

<input placeholder="input city name"></input>

<button bindtap='query'>query weather</button>

22. https://heweather.com/

key: fa66a4c393974639bcb6f2c06bf504b8

https://free-api.heweather.com/s6/weather/now?location=beijing&key=fa66a4c393974639bcb6f2c06bf504b8

23. input 添加属性 bindinput='getCity'

24. weather.js

...,

getCity: function() {

console.log('input...');

},

25. getCity: function(event) {

console.log(event.detail.value);

}

26. data: {

...,

city: ''

}

getCity: function(event) {

this.setData({

city: event.detail.value;

});

}

27. query: function() {

let city = this.data.city;

console.log(city);

}

28. let url = `http://...location=${this.data.city}...`;

29. wx.request({

url: '',

success: function(res) {

console.log(res);

}

});

30. 错误：https://free-api.heweather.com 不在以下 request 合法域名列表中，...

30.1 上线时：设置 - 开发设置 - 服务器域名

30.2 开发时：project.config.json

"urlCheck": false

31. let weather = res.data.HeWeather6[0].now.cond\_txt;

32. let page = this;

wx.request({

// ...

page.setData({

weather: weather

});

// ...

});

33. 练习：http://cz88.net 纯真 IP 地址数据库

实现小程序，根据用户输入的 IP，返回地理位置

文件中的数据存入 MySQL 数据库，软件界面点击“解压”，生成文本文件

起始IP 终止IP 地理位置

WECHAT DAY04

Wechat 开发

Day04 小程序开发

1. 下载 http://cz88.net 纯真 IP 地址数据库

2. 实现小程序，根据用户输入的 IP，返回地理位置

3. 软件界面点击“解压”，生成文本文件

起始IP 终止IP 地理位置

转码 Notepad++ 菜单 - 编码 - 转为 UTF-8 编码

4. IP = Internet Protocol

A.B.C.D

[0, 255]

IPv4

IPv6

5. server-ip/sql/db.sql 建库建表

6. server-ip/

cmd: npm init

7. server-ip/

cmd: npm i -S mysql express

8. server-ip/data/

ip.data

dumpData.js

9. 实现数据的快速导出

10. MySQL 函数

INET\_ATON

INET\_NTOA

INET internet

A address

TO to

N number

SELECT \*

FROM db.ip

WHERE INET\_ATON('用户输入的 IP')

BETWEEN INET\_ATON(min) AND INET\_ATON(max);

11. 小程序中创建新的页面 ip，设为首页

<text>{{loc}}<text>

<input></input>

<button bindtap='query'></button>

12. <input bindinput='getIp'></input>

13. ip.js

getIp: function(event) {

this.setData({

ip: event.detail.value

});

}

query: function () {

let ip = this.data.ip;

let url = '';

wx.request({

url: url,

success: function (res) {

// ...

}

});

}

14. server-ip/

app.js

15. 创建新的页面：map

<map></map>

height: 100vh

16. longitude: 经度

latitude: 纬度

scale: 缩放级别

longitude='{{longitude}}'

17. markers: [

{

longitude: ,

latitude: ,

iconPath: '/resources/img/...',

width: ,

height: ,

}

]

18. polyline: [ // 多个折线

{ // 一条折线

points: [ // 折线里的多个点

{ // 第一个点

longitude: ,

latitude: ,

}

{}, // 其他的点

...

]

color: '',

width:

}，

{} // 其他折线

]

19. 练习：点击定位控件，使地图的中心是当前用户所在位置

WECHAT DAY05

Wechat 开发

Day05 小程序开发

1. controls，地图上可以触发事件的图片

id：标志

2. wx.getLocation();

3. wx.getSystemInfo({success: function(res){ // ...}})；

4. 点击定位控件，使地图的中心是当前用户所在位置

5. 点击天气控件，在当前位置的 marker 上显示实时天气信息

markers: [

{

...,

label: {

content: '天气信息'

}

}

]

6. 添加一个 control，id: 2

7. 点击天气图标

7.1 wx.getLocation

longitude = res.longitude;

latitude = res.latitude;

7.2 let url = `...location=${longitude},${latitude}...`;

wx.request({

url: url,

success: function(res) {

let cond = res...;

let temp = res...;

let content = `cond:${cond},temp:${temp}`;

page.setData({

longitude: longitude,

latitude: latitude,

markers: [

{

...,

label: {

content: content

}

}

]

});

}

});

8. 创建一个新的页面：game，设为首页

9. game.wxml 画一个正方形，填充某种颜色

10. 把 9 中的行内样式，设置到 game.wxss 中

11. game.wxml 画 2\*2 个正方形，填充某种颜色

占据屏幕的宽度

12. game.wxml 画 n\*n 个正方形，填充某种颜色

13. 具体的 2\*2 boxes 数据来自于 game.js

game.wxml

<view wx:for='{{list}}'>{{item}}</view>

注意：item 默认值，是 list 循环的当前项

game.js

data: {

list: [1, 2, 3, ...]

}

14. 同时指定每个 box 的颜色

game.js

data: {

boxes: [

{style: 'box2', color: 'red'},

...,

...,

...

]

}

game.wxml

<view

wx:for='{{boxes}}'

class='{{item.style}}'

style='background: {{item.color}};'></view>

15. 用户点击 box 时，添加事件响应 bindtap='select'

16. game.js 定义 select 函数

17. game.js

boxes: [

{..., test: 1},

...,

]

18. game.wxml

<view ...,

data-test='{{item.test}}'></view>

19. select: function(event) {

console.log(event.target.dataset.test); // 1, 2, ...

}

20. 点击任意 box，在 select 函数中生成新的 2\*2 boxes

每个 box 的颜色是随机的

this.setData({

boxes: boxes

});

21. 随机选择 2\*2 中的一个 box，为这个 box 设置不同的颜色

22. 这个 box 中不同的颜色应该尽量贴近于其他 boxes 的颜色

23. 在页面上显示分数，每次选对加 1 分

24. 练习：完善游戏的功能

Git

今日目标：

Git -> GitHub.com -> Node.js/Vue项目新浪云部署

|  |
| --- |
| Version：软件的发行版本号 —— 给最终用户使用  Revision：软件的开发版本号 —— 开发者每次修改保存都对应一个版本号 |

1.VCS/SCM系统介绍

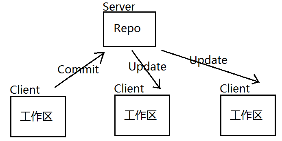
VCS：Version Control System，版本控制系统

SCM：Source Code Management，源代码管理系统

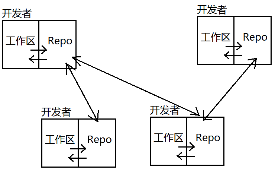
作用：用于在企业级的多人协作项目中实现项目文件的保存、共享、修改记录、历史回退、代码追踪等问题。

常见的VCS软件：

1. 集中式VCS软件：有中央服务器的概念，如SVN



1. 分布式VCS软件：没有中央服务器的概念，每一个开发者计算机中既有工作区，也有代码仓库。每次提交/更新都无需联网。如Git



|  |
| --- |
| VCS系统常用概念：  (1)Working Directory：工作区，用来保存/修改项目文件的目录  (2)Repository：仓库，用于永久的存储项目文件以及每个文件的每次修改记录 Repo = File + History  (3)Commit：提交，把工作区的文件修改“上传”到仓库，生成响应的版本号(Revision)  (4)Update：更新，把仓库中的最新内容“下载”到本地的工作区 |

Linus为了维护Linux源代码而创建的分布式系统。

2.使用Git的步骤

(0)卸载已有的Git软件，并删除之前保存的用户凭证

控制面板 -> 用户账户 -> 凭据管理器 -> 删除所有git相关的凭证

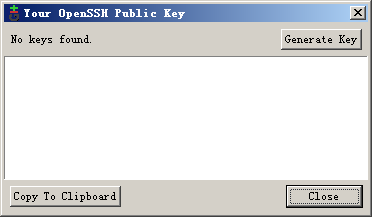
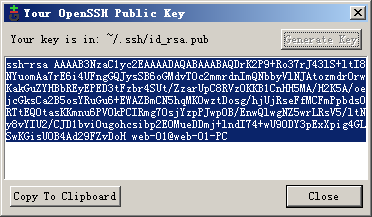
(1)到Git官网下载git的安装文件

官网：https://git-scm.com/

(2)安装Git程序，一路Next即可

---------------使用前的设置---------------------

(3)生成传输文件加密和解密需要的“私钥”和“公钥”

(4)告诉Git当前系统的Git用户名和邮箱

git config --global user.name "LiWenhua"

git config --global user.email "liwenhua@tedu.cn"

上述两个命令的本质是在修改C:\Users\用户名\.gitconfig这个文件

|  |
| --- |
| Windows系统中如何手工创建.开头的文件：  右键点击 -> Git Bash Here -> 输入 touch .文件名  touch命令来自于Linux系统，用于创建空白的文件 |

---------------开始使用Git操作本地的仓库---------------

(5)创建一个本地的代码仓库(Repo=File+History)

git init

|  |
| --- |
| 单机版的Git文件目录结构： |

(6)查看当前工作区和暂存区文件的状态

git status

(7)把工作区中的修改添加到暂存区

git add 文件名

git add . 点指代所有发生了修改的文件

(8)把暂存区中的修改永久提交到仓库

git commit -m "提交的说明原因"

注意：不添加-m参数会进入VIM编辑器！强制输入说明原因！

(9)查看完整的提交历史记录

git log

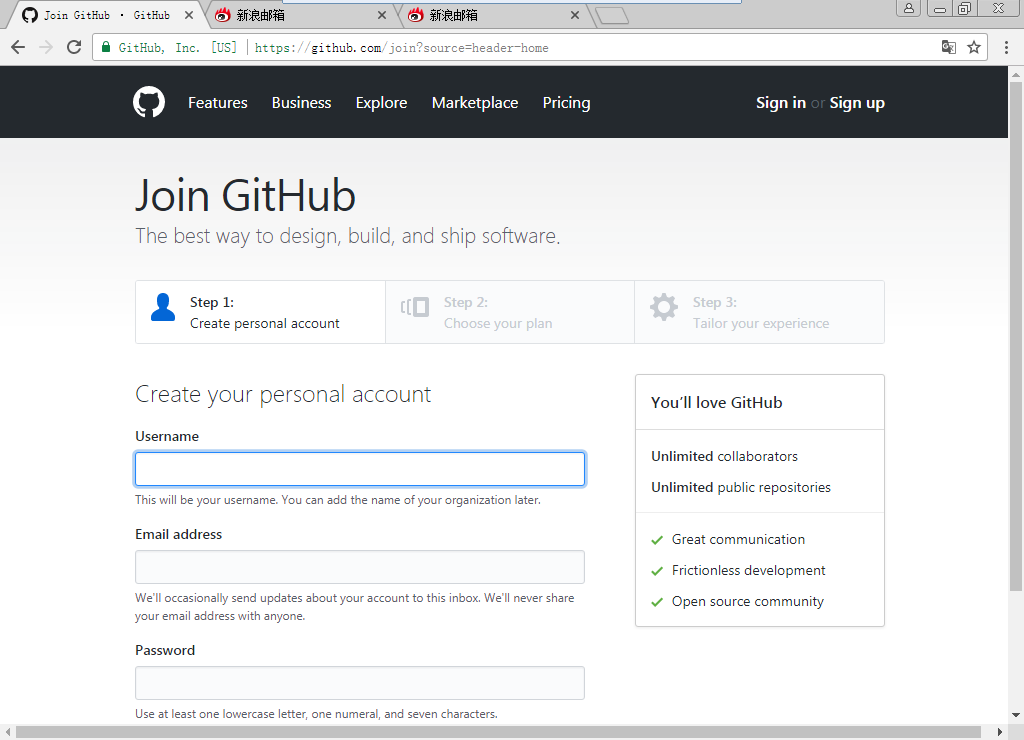
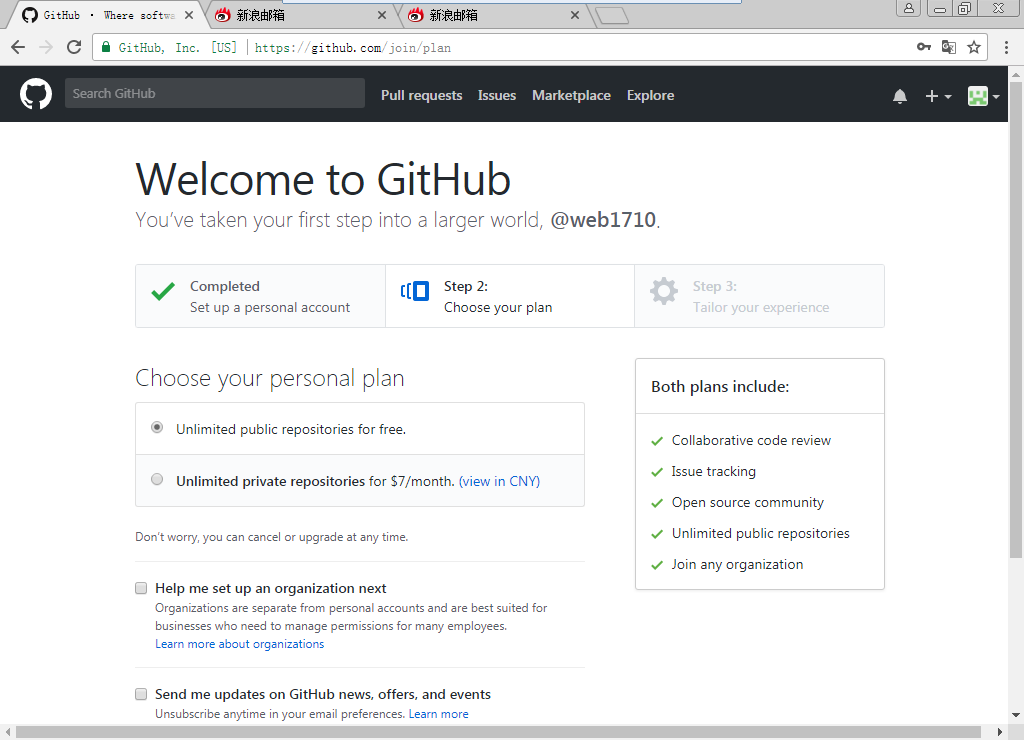
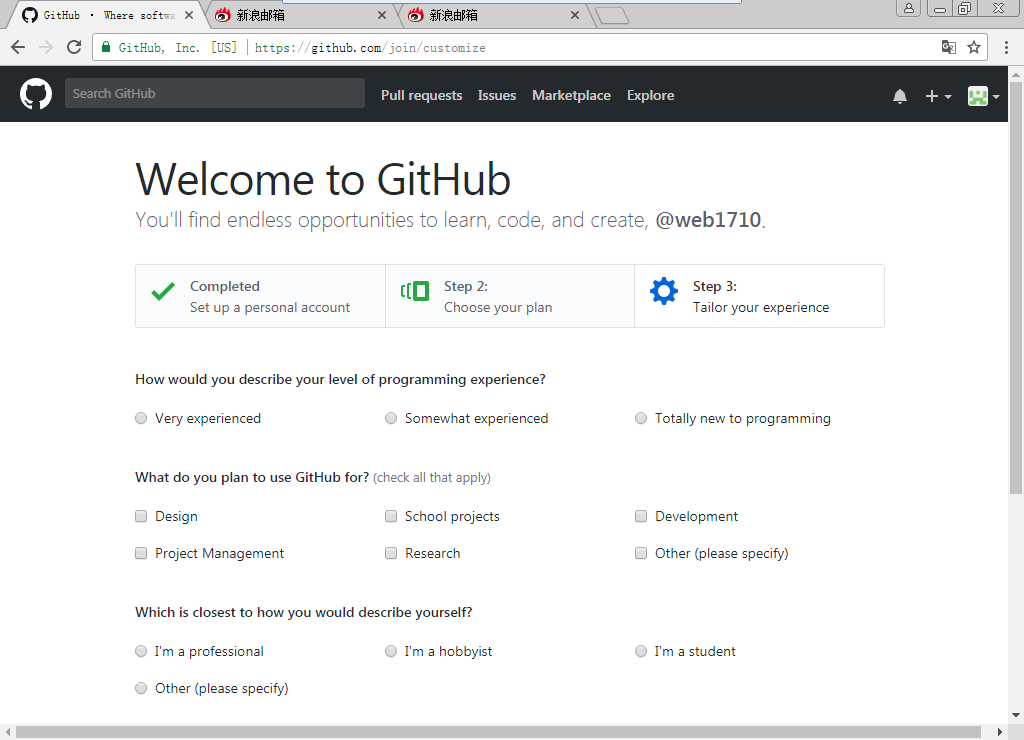
(10)从历史仓库中检出指定的文件内容

git checkout 版本号 文件名

---------------开始使用Git操作远程的仓库(Github)---------------

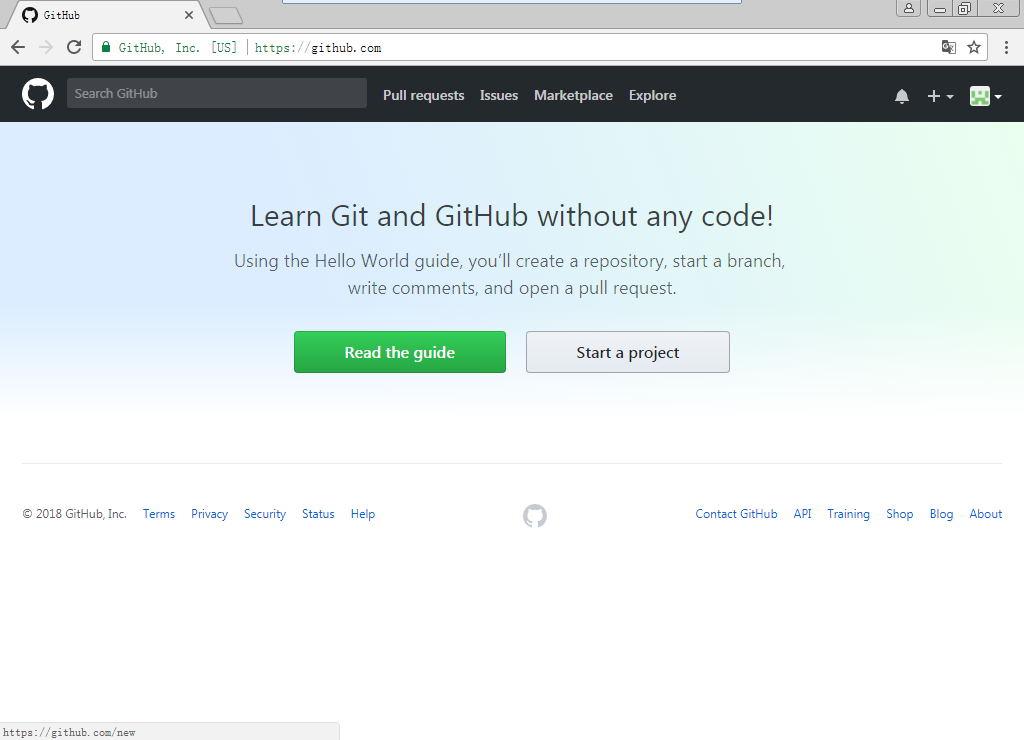
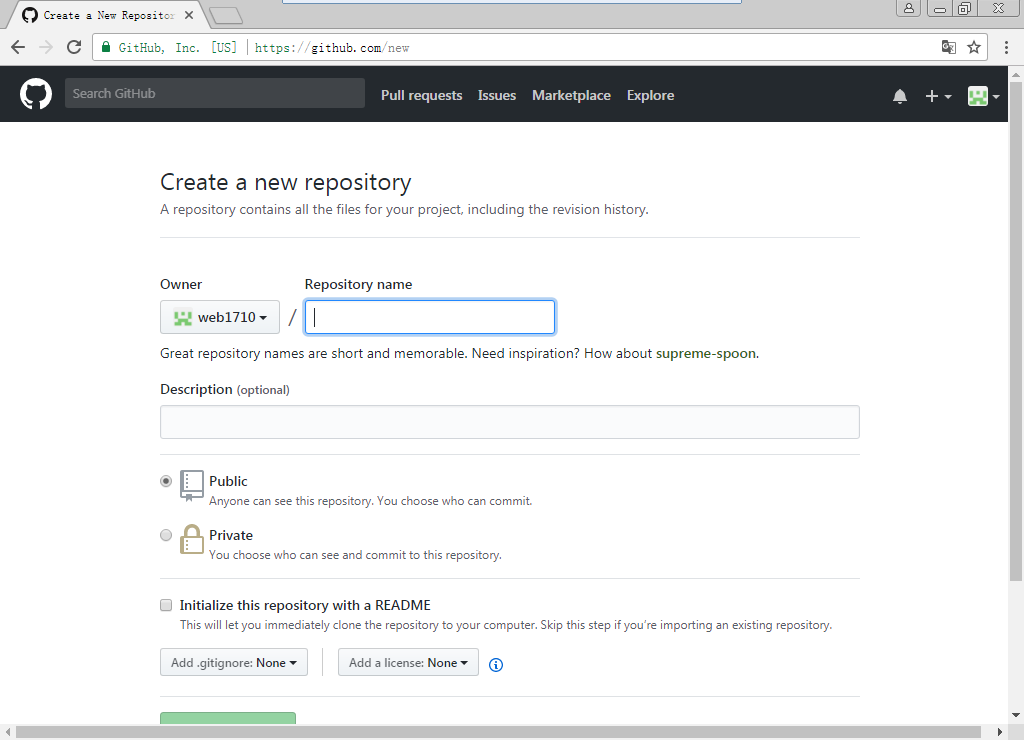
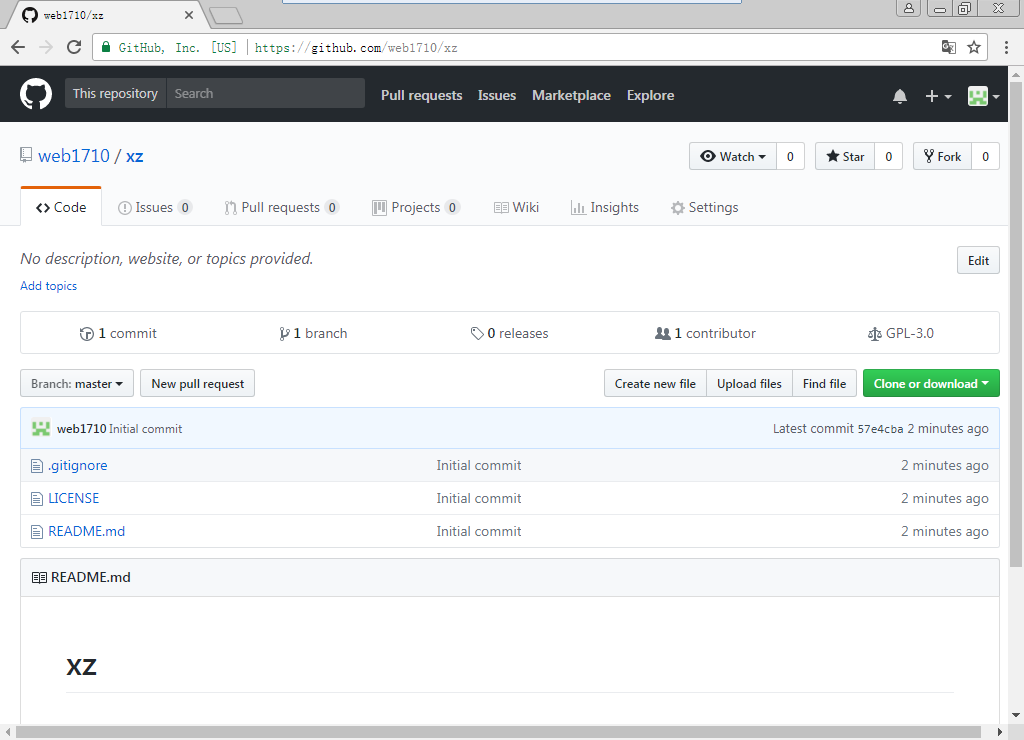
|  |
| --- |
| Git和Github的关系？  Git是SCM软件，每个安装该软件的计算机都有工作区+仓库，都是平等的；  Github是一个网站(github.com)，提供了仓库的托管服务，可以保证提供7x24全球可以访问的Git仓库。 |

(11)在github.com注册一个账户

登录邮箱，点击账户激活邮件。

(12)使用自己的账户登录github，为某个项目创建一个远程仓库

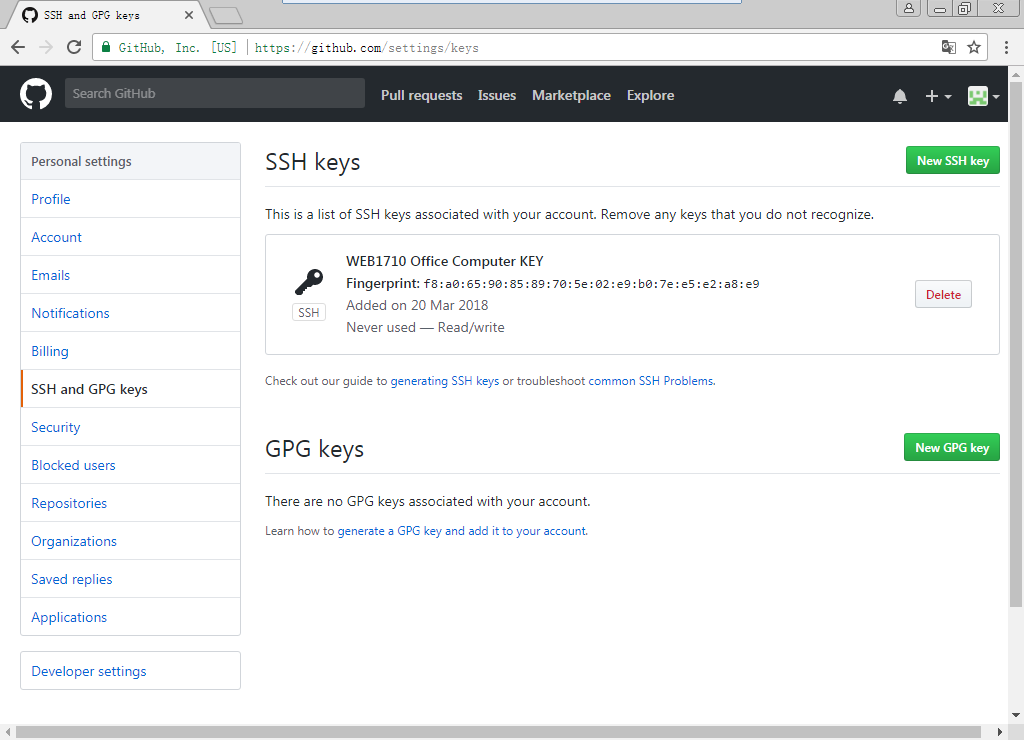
每个项目都有两个不同形式的访问地址：

https://github.com/web1710/xz.git (可以在浏览器地址栏中访问)

git@github.com:web1710/xz.git (使用Git客户端push/pull速度更快)

(13)把远程仓库内容克隆到本地，形成一个本地仓库

首先在github.com的用户设置添加当前电脑的Git公钥：



git clone 远程仓库地址

(14)在本地工作区中添加/提交修改内容到本地仓库，推送(PUSH)到远程仓库

修改

git add . 添加到本地暂存区

git commit -m "修改的原因说明" 提交到本地仓库

git push 远程仓库地址 将本地仓库推送到远程

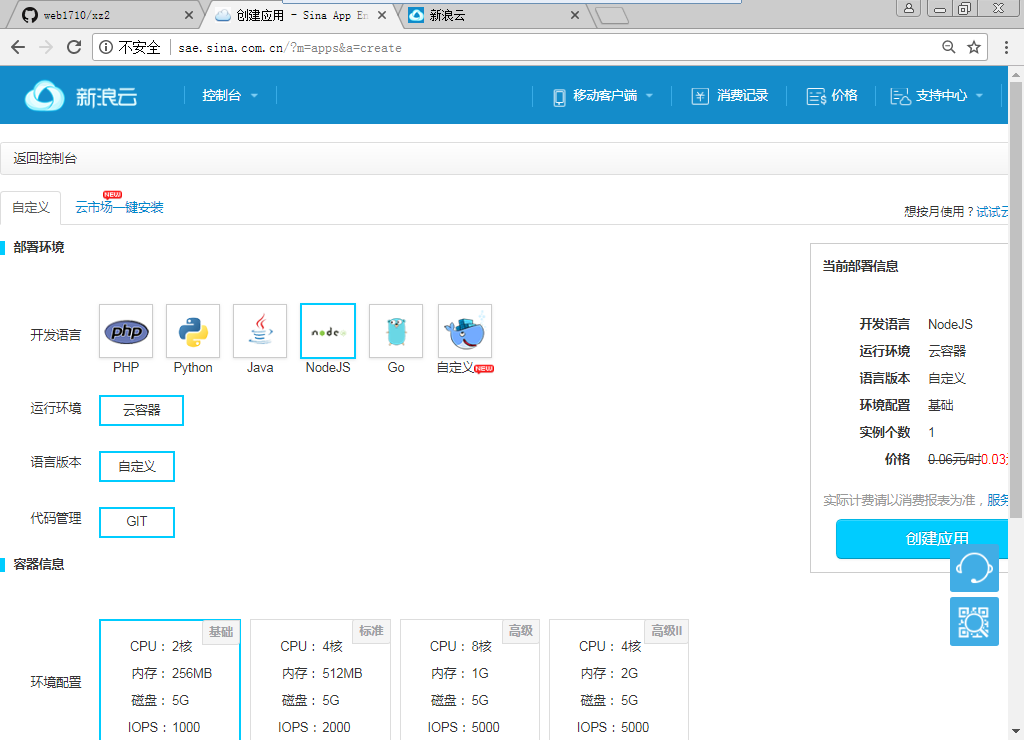
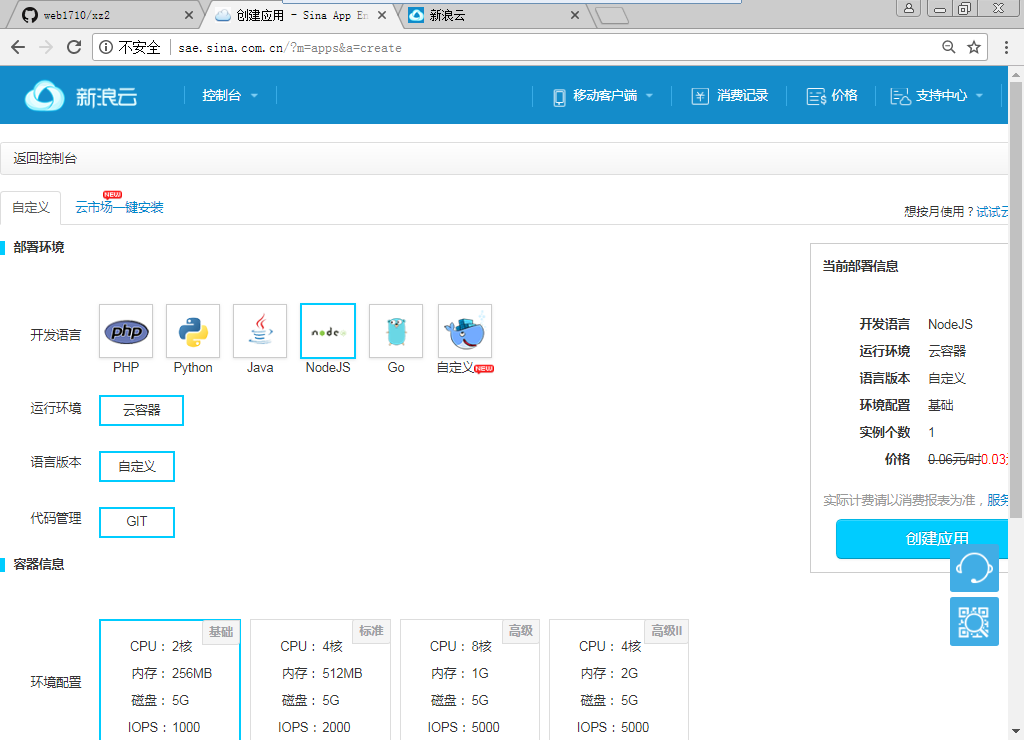
(15)其它同事可以拉取(PULL)远程仓库中的最新内容到各自的本地仓库

git pull 远程仓库地址 master

-------------如何使用新浪云中的Node/Vue服务器---------------

新浪云中的PHP服务器可以使用FTP、SVN或Git进行代码管理；但Node服务器只能用Git来管理。官方文档：

http://www.sinacloud.com/doc/sae/nodejs.html

每个新浪云项目都被自动分配了一个Git仓库。根据页面下方的提示，接下来开始克隆这个仓库到自己的计算机，添加文件并提交到本地仓库，再将本地仓库推送到远程仓库。

|  |
| --- |
| git clone 新浪云Git仓库地址 |
| 注意：  (1)Node.js默认监听端口必须为5050.  (2)package.json文件中必须声明  "scripts": {  "start": "node index.js"  } |
| git add .  git commit -m "提交说明"  git push 新浪云Git仓库地址 |