# 前端优化：

1.减少http请求次数，适当设置缓存

2.压缩代码

3.部署前，压缩图片  
4.优化项目代码（合理利用选择器，合理的利用属性，合理的定义函数，减少href和src为空的情况，减少冗余代码）  
5.减少DOM操作

# SERVER-day01

1.前端课程体系（2019.4.1）

1. 服务器端开发：项目导入/数据库/JS基础/服务器开发
2. 前端核心技术：HTML/AJAX/CSS/bootstrap（响应式）
3. 前端进阶技术：JS高级/BOM&DOM/jquery/VueJS(框架)
4. H5移动端开发:H5/微信开发/HybridApp

框架技术：Angular/React(App)/Python语言

2．Windows 中常用的快捷键

Ctrl+S 保存

Ctrl+C 复制

Ctrl+V 粘贴

Ctrl+X 剪切

Ctrl+A 全选

Ctrl+Z 撤销（返回到上一步）

window+D 显示/隐藏桌面

Backspace 返回

window+E 打开资源管理器

Alt+Tab右切换工作窗口

Alt+Shit+Tab 左切换工作窗口

Alt+F4 关闭当前窗口

Win+R 运行 cmd（命令行）/mspaint （画图）/calc（计算器）

上下文键+对应项的字母

F2 重名名

F5 刷新

Ctrl+B 在浏览器中查看网页效果

3.软件工程：研究和应用如何系统得的、规范化的、可定量化的方法开发和维护软件。

目标是在给定成本、进度的前提下，开发出具有有效性、可维护性、可重用性、可以移植性的软件产品

4.软件过程/软件生命周期

1. **软件定义期**  
   1)研究阶段  
   技术、经济（资金）、时间、设备、政策法规、风俗、回报率

2)需求分析阶段  
功能性需求、非能性需求、

(2)**软件开发期  
 3**）概要设计阶段

子系统、模块、各自的功能、模块间的接口  
 4）详细设计阶段

模块实现细节、属性、方法、算法  
 5）编码实现阶段

后端工程师

前端工程师

UI设计师

6）项目测试阶段  
（3）**软件维护期** 7）项目部署阶段

部署到服务器

8)项目维护阶段

5.商城功能性需求

[www.codebody.com](http://www.codebody.com)

www.codebodyy/admin/login.html

前台：

产品模块：首页、列表、详情

用户模块：注册、登录、用户中心、收藏夹

购物车模块：添加购物车、修改、删除、下单

后台:

产品模块：添加、修改、删除、列表

用户模块：列表、修改、删除

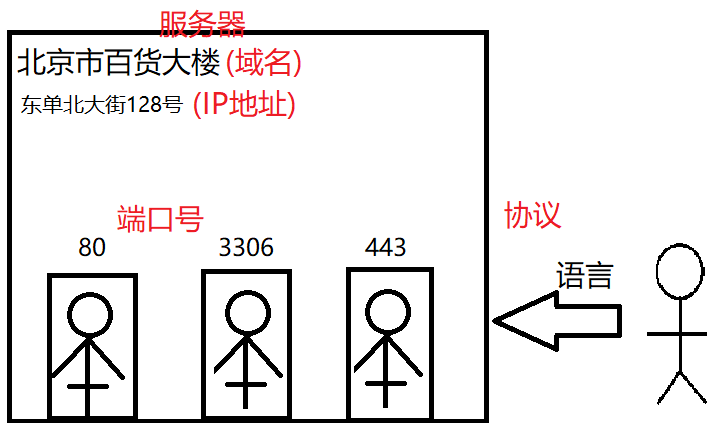
订单模块：列表、修改

6．Web项目中的服务器  
服务器（Server）:为客户端提供各种服务的功能强大的计算机。  
访问服务器：

（1）服务器的地址:域名/IP地址

（2）所对应服务的端口号

（3）提供该服务所使用的协议



7.访问自己电脑创建的Web服务

启动web服务器 、webserver/start.cmd

使用浏览器去访问该服务  
<http://127.0.0.1:80> 通过服务器的IP地址

<http://localhost:80> 通过服务器的域名

使用局域网的IP地址访问

Ipconfig 查看电脑在局域网的IP地址

以太网适配器 以太网 ipv4

172.163.6.254：80

课后任务

1. 开启自己电脑的web服务器，让同桌使用服务器来访问。
2. 复习今天内容
3. 预习，了解Mysql

http://Nodejs.org

# SQL-day02

1.项目中的数据(2019.4.2)

内存/特定的文件（excel、word）/第三方的服务器/数据库服务器

2.数据库

数据库是按照一定的形式来组织、存储数据，目的是为了对数据操作——增删改查

（1）数据库的发展历史

网状数据库->层次型数据库->关系型数据库->非关系型数据库（NoSql）

（2）关系型数据库逻辑结构

|  |
| --- |
| Server->Datebase->table->Row->Column |

(3)常见的关系型数据库

SQLite 微型数据库

SQL Server （只能运行Windows服务器）使用于Windows系统，收费，中型数据库

Orace 大型数据库，收费

MySQL 中小型数据库，免费，适用于各种操作系统

3.MySQL 数据库

MySQLAB->SUN->Oracle

oracle分支:mysql

martin分支:MariaDB

XAMPP软件(需要安装node.js)

|  |
| --- |
| 服务器套装，包含多款服务器软件mysql、Apache...  <https://www.apachefriends.org/download.html> |

(1)mysql的部署结构

服务器端;负责存储/维护数据——银行服务器机房

C：/xampp/mysql/bin/mysqld.exe 启动服务

确保3306端口不被占用

客户端:负责向服务器端发起增删改查——ATM机

C:/xampp/mysql/bin/mysql.exe 客户端工具

(2)使用客户端连服务器端

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

mysql.exe -hlocalhost -P3306 -uroot -p

-h host主机（IP地址/域名）

-P port 端口8-u user 用户]

-p password 密码

mysql -uroot 简写形式

注意事项：不能在连接时不能加分号

4.mysql常用管理应用命令

quit; 退出服务器连接

show databases ;显示服务器上当前所有的数据库

use 数据库名; 进入指定的数据库

show tables; 显示当前数据库中所有的数据表

desc 表名； 描述表中都有那些列（表头）

5.SQL命令  
SQL：Structured Query Language 、结构化查询语言，用于操作关系型数据库服务器，对数据执行增删改查等操作

练习: 查看root用户登录查看服务器上有哪些数据库。

练习: 查看phpmyadmin和test 分别查看都有哪些数据表。

练习

pma\_history

pma\_recent

pma\_favotie

SQL命令的两种执行方式

（1）交互模式：客户端输入一行，点击回车。服务器执行一行。适用于临时性的查询数据

（2）脚本模式：客户端吧要执行的多行命令卸载一个文本文件中，一次性的提交给服务器。适用于批量的操作数据

不能进入mysql -uroot< C:/xampp/../02.sql 回车

练习：查看phpmyadmin中以下都有那些列

SQL语法规范

（1）每条SQL命令可以跨越多行的，遇到英文分号作为结束

（2）假如某一条命令出现语法错误，则此条语句以及后边所有的语句不会再执行

（3）SQL命令不区分大小写，习惯上关键字用大写，非关键字用小写

（4）SQL 命令中可以使用单行注释（#..）和多行注释（/\*...\*/）,注释的内容不会被服务器所执行

常用的SQL命令

（1）#丢弃指定的数据库xuezi,如果存在

DROP DATABASE IF EXISTS xuezi;

（2）#创建新的数据库xuezi

CREATE DATABASE xuezi;

（3）#进入创建的数据库

use xuezi;

（4）创建保存的

CREAT TABLE(

sid INT,

name VARCHAR(8),

score VARCHAR(1)

);

练习;编写脚本文件04\_td.sql,先丢弃数据库td,如果存在；创建数据库td.进入到该数据库。

(5) 向数据表插入数据

INSERT INTO student VALUES('1','tom','m','90');

（6）查询所有的数据

SELECT \*FROM student;

（7）修改数据

UPDATE studen SET name='lucy', score='100' WHERE sid='2'

(8)删除数据

DELETE FROM student WHERE sid='3';

练习：编写脚本文件05\_dang.sql，先丢弃再创建数据库dang;进入该数据库，创建保存图书数据的表book，包含图书编号bid，书名title，价格price；插入两条数据，修改1条，删除1条，查询结果。

课后任务

(1)复习今天内容

(2)练习

编写脚本文件xz.sql，先丢弃在创建数据库xz，进入该数据库，创建保存用户数据的表user，包含(uid, uname, upwd, email, phone, sex, userName真实姓名, regTime 注册时间,isOnline 是否在线)，插入5条记录，删除1条，修改1条，查询结果。

(3)预习mysql中的列类型 手册第11章

# SQL-day03

标准SQL命令分类（2019.4.3）

|  |
| --- |
| DDL:Data Ddfine languange 定义数据结构  CREATE（创建）/DROP（丢弃）/ALTER(修改表的结构)  DML:Data Manipulate Language 操作数据库  INSERT(插入)/UPDATE（修改）/DELETE（删除）  DQL:Data Query language 查询数据  SELECT  DCL:Data Control Language 控制用户权限  GRANT(授权)/REVOKE(收权) |

**1.计算机存储字符**

（1）如何存储英文字符

ASCLL：总共有128个，对所有的英文字母和符号进行编码

ABC 656667

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

（2）如何存储中文字符

GB2312：对常用的6000多汉字进行编码，兼容ASCLL码

GBK：对2万多汉字进行了编码，兼容GB2312

GBK5:台湾繁体字编码，兼容ASCLL码

Unicode：对世界上主流的国家常用的语言进行了编码，兼容ASCLL码，不兼容GB2312,GBK,BIG5,具体分为UTF-8、UTF-16、UTF-32存储方案。

（3）解决mysql存储中文乱码

使用UTF-8编码形式

SQL脚本文件另存为的编码

客户端连接服务器的编码（SET NAMES UTF8）

服务器端创建数据库使用的编码（CHARSET=UTF8）

练习:编写脚本文件01\_dang.sql,创建dang,设置编码为utf8;创建保存图书的数据表book，包含bid，title标题，author作者，price价格，publish出版社，pubTime出版时间

插入4条记录，查询结果。SET NAMES UTF8;

DROP DATABASE IF EXISTS dang;

CREATE DATABASE dang CHARSET=UTF8;

USE dang;

CREATE TABLE book(

bid INT,

title VARCHAR(10),

author VARCHAR(8),

price INT,

publish VARCHAR(10),

pubTime VARCHAR(15)

);

INSERT INTO book VALUES

('1','格林童话','安徒生','20','北京大学出版社','2019-04-03'),

('2','西游记','吴承恩','20','北京大学出版社','2019-04-03'),

('3','红楼梦','曹雪芹','20','北京大学出版社','2019-04-03'),

('4','水浒传','施耐庵','20','北京大学出版社','2019-04-03'),

('5','三国演义','罗贯中','20','北京大学出版社','2019-04-03');

UPDATE book SET price='26',publish='户庙出版社' WHERE bid='3';

交互模式 select \*from dang;

**2.mysql中的列类型**

创建数据表的时候，指定的列可以存储的数据类型

CREATE TABLE book(bid 列类型)

（1）数值类型——引号可加可不加

TB->GB->MB->BYTE（字节）->BIT(位)

TINYINT 微整型（占1个字节）范围 -128~127

SMALLINT 小整型，(占2个字节)范围 -32768~32767

INT 整型，（占4个字节）- 2147483648~ 2147483647

BIGINT 大整型，（占8个字节）

FLOAT (M D)单精度浮点型，（占4个字节），范围。3.4E（38），范围比INT大的多，可能产生计算误差。M表示总的有效位数 D小数的有效位数

DOUBLE (M D) 双精度浮点型，（占8个字节），范围比BIGINT大的多。

DECIMAL（M D）定点小数，不会产生误差

BOOL 布尔型，只有两个结果 TRUE、FALSE（不能加引号）,真正存储数据的时候，会自动变成1和0；也可以直接使用1和0；数据中的列类型自动变成TINYINT。

（2）日期时间类型——必须加引号

DATE 日期型 '2019-12-31'

TIME 时间型 '14:34:31'

DATETIME 日期时间型 '2019-12-31 14:34:31'

（3）字符类型——必须加引号

VARCHAR(M) 变长字符串 优点：不会产生空间浪费，操作速度相对慢 M的最大值是 65535

CHAR(M) 定长字符串 优点：可能产生空间浪费，操作速度相对快，常用于存储手机号码、身份号码等固定长度字符。M的最大值是255

TEXT(M) 大型长字符串， M最多存2G

使用合理的列类型

练习：编写脚本文件02\_xuezi.sql,先丢弃再创建数据库xuezi，使用utf8编码，进入该数据库，创建保存笔记本数据的表laptop ,包含lid,title,price,stockCount库存量，shelfTime上架时间，isIndex是否为首页推荐

插入4条记录

SET NAMES UTF8;

DROP DATABASE IF EXISTS xuezi;

CREATE DATABASE xuezi CHARSET=UTF8;

USE xuezi;

CREATE TABLE laptop(

lid INT,

title VARCHAR(100),

price DECIMAL(6,2),

stockCount SMALLINT,

shelfTime DATE,

isIndex BOOL

);

INSERT INTO laptop VALUES

('1','小米','5999','1000000','2019-4-3',TRUE),

('2','联想','5999','1000000','2019-4-3','FALSE'),

('3','鸿基','5999','1000000','2019-4-3','1'),

('4','戴尔','5999','1000000','2019-4-3','1');

UPDATE laptop SET price='8999' WHERE title='小米';

#DELETE FROM laptop WHERE title='鸿基';

练习:编写脚本文件03\_tedu.sql,创建数据库tedu，设置编码为utf8,进入该数据库。

创建保存部门数据的表dept，包含did、dname部门名称、emCount部门员工数量，插入以下数据

10 研发部 3,20 市场部 2 ，30 运营部 2

创建员工数据的表emp,包含eid、 ename、sex、brithday生日、salary工资、deptld所属部门编号

插入若干条记录

SET NAMES UTF8;

DROP DATABASE IF EXISTS tedu;

CREATE DATABASE tedu CHARSET=UTF8;

USE tedu;

CREATE TABLE dept(

did SMALLINT,

dnaem CHAR(8),

emCount INT

);

INSERT INTO dept VALUES

(10,'研发部','3'),

(20,'市场部','2'),

(30,'运营部','2');

CREATE TABLE emp(

eid INT PRIMARY KEY,

ename CHAR(5),

sex BOOL,

brithday DATE,

salary DECIMAL(8,2),

deptld SMALLINT

);

INSERT INTO emp VALUES

('1','张三',1,'1996-10-28','5900',''),

('2','张三',1,'1996-10-28','5900',''),

('3','张三',1,'1996-10-28','5900','');

**3列约束**

mysq可以随插入的数据进行特定的验证，只有满足条件才允许插入到数据表中，否则认为是非法的插入。

例如：一个人的年龄只能是男或女，一个人的年龄0~100

（1）**主键约束——PRIMARY KEY**

声明了主键约束的列上不能出现重复的值，表中查询的记录按照主键从小到大排序——加速查找速度；

通常主键添加到编号列中。

注意：一个表中只能出现一个主键，主键列上不允许插入NULL.

|  |
| --- |
| NULL 表示空，在插入数据时，无法确定要保存的值；  例如：无法确定的员工的工资、姓名、声明 |

课后任务

(1)复习今天内容

(2)练习

编写脚本文件xz.sql，创建数据库xz，设置编码utf；

创建保存笔记本家族分类的表laptop\_family( fid, fname 分类名称, laptopCount笔记本数量 )

10 联想 2 20 戴尔 2 30 小米 3

创建笔记本数据表laptop(lid，title，price，spec规格，detail详情，shelfTime上架时间，isOnsale是否在售，familyId 所属分类编号)

插入若干条数据

(3)预习mysql中列约束

(1)SET NAMES UTF8;

DROP DATABASE IF EXISTS xz;

CREATE DATABASE xz CHARSET=UTF8;

USE xz;

CREATE TABLE laptop\_family(

fid SMALLINT PRIMARY KEY,

fname CHAR(5),

laptopCount int

);

INSERT INTO laptop\_family VALUES

(10,'联想',2),

(20,'戴尔',2),

(30,'小米',2);

CREATE TABLE laptop(

lid SMALLINT,

title CHAR(10),

price DECIMAL(8,2),

spec SMALLINT,

detail VARCHAR(300),

shelfTime DATE,

isOnsale BOOL,

familyId SMALLINT

);

INSERT INTO laptop VALUES('1','小米','3999.99','103','游戏本','2019-4-3','1','1');

**（2）非空约束 ——NOT NULL**

声明了非空约束的列上不允许插入NULL值

**（3）唯一约束——UNIQUE**

声明了唯一的列不能插入重复的值，允许插入NULL ,甚至多个NULL**;**一个表中可以出现多个唯一约束

|  |
| --- |
| 说明：NULL这个值比较特殊，它和任何值都不等，甚至和自身都不等 |

练习：给笔记本家族表中，分类名称中添加唯一约束，插入数据测试。

**（4）检测约束——CHECK**

检查约束可以对插入的数据可以进行自定义验证

CREATE TABLE student (

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

);

mysql 不支持检查约束，会降低数据的插入速度

（5）默认值约束——DEFAULT

可以使用DEFAULT 关键字生命默认值，有两种可以使用默认值

INSERT INTO laptop\_family VALUES(50,'华硕',DEFAULT);

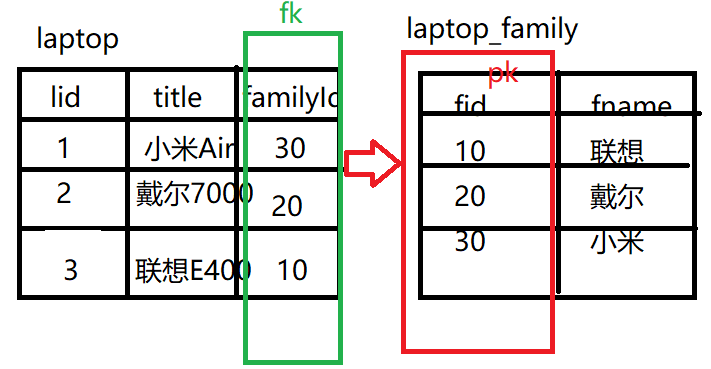
INSERT INTO laptop\_family (lid,fname)VALUES(60,'神州');

练习：设置笔记本表中价格默认为6999，分别使用两种方式调用

(6)外键约束——FOREIGN KEY

声明了外键约束的列上,取之必须有在另一个表中的主键列上出现过，两者的列类型要保持一致，允许使用NULL

FOREIGN KEY(外键列)REFERENCES（引用） 另一数据表（主键）



2.mysql中的自增列

AUTO\_INCREMENT:自动增长，假如一个列声明了自增列，无需手动赋值，直接赋值为NULL ,会获取当前的最大值，然后加1

注意：

只适用于整型的列上

自增列允许手动赋值

练习：编写脚本文件01\_tedu.sql，创建数据库tedu，设置编码为UTF8；

创建保存部门数据的表dept，包含(did, dname)

10 研发部 20市场部 30运营部 40 测试部

创建保存员工数据的表emp，包含(eid, ename, sex, birthday, salary, deptId)

使用恰当的列约束和自增；

插入15条记录。

#设置客户端连接的编码

SET NAMES UTF8;

#丢弃数据库

DROP DATABASE IF EXISTS tedu;

#创建数据库

CREATE DATABASE tedu CHARSET=UTF8;

#进入该数据库

USE tedu;

#创建保存部门数据的表

CREATE TABLE dept(

did SMALLINT PRIMARY KEY AUTO\_INCREMENT,

dname VARCHAR(8) UNIQUE

);

#插入数据

INSERT INTO dept VALUES(10,'研发部');

INSERT INTO dept VALUES(20,'市场部');

INSERT INTO dept VALUES(30,'运营部');

INSERT INTO dept VALUES(40,'测试部');

#创建保存员工数据的表

CREATE TABLE emp(

eid INT PRIMARY KEY AUTO\_INCREMENT,

ename VARCHAR(8) NOT NULL,

sex BOOL DEFAULT 0,

birthday DATE,

salary DECIMAL(8,2), #999999.99

deptId SMALLINT,

FOREIGN KEY(deptId) REFERENCES dept(did)

);

#插入数据

INSERT INTO emp VALUES(NULL,'tom',1,'1990-5-5',6000,20);

INSERT INTO emp VALUES(NULL,'jerry',DEFAULT,'1991-8-5',7000,10);

INSERT INTO emp VALUES(NULL,'david',1,'1995-10-20',3000,30);

INSERT INTO emp VALUES(NULL,'maria',0,'1993-12-5',8000,10);

INSERT INTO emp VALUES(NULL,'leo',1,'1996-5-5',4000,20);

INSERT INTO emp VALUES(NULL,'black',1,'1989-6-5',10000,30);

INSERT INTO emp VALUES(NULL,'peter',1,'1990-12-3',12000,20);

INSERT INTO emp VALUES(NULL,'franc',1,'1996-5-5',8000,10);

INSERT INTO emp VALUES(NULL,'lucy',0,'1986-5-5',5500,30);

INSERT INTO emp VALUES(NULL,'Tacky',1,'1984-5-5',8000,10);

INSERT INTO emp VALUES(NULL,'jone',1,'1993-5-5',6500,20);

INSERT INTO emp VALUES(NULL,'king',0,'1993-5-5',7500,10);

INSERT INTO emp VALUES(NULL,'richard',1,'1992-8-5',13500,20);

INSERT INTO emp VALUES(NULL,'lily',0,'1996-3-1',15500,30);

INSERT INTO emp VALUES(NULL,'brown',1,'1997-5-5',18500,NULL);

3.简单查询

（1）查询特定列

示例：查询所有员工的姓名、生日

SELECT ename,birthday FROM emp

练习：查询所有员工的编号，姓名，性别，薪资

SELECT eid AS 编号,ename ,sex,salary FROM emp;

（2）查询所有的列

SELECT \*FROM emp;

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

（3）给列起别名

示例：查询有员工的姓名和工资，使用中文别名

SELECT ename AS 姓名,salary AS 工资 FROM emp;

练习：查询所有员工的编号，姓名，性别，生日；使用中文别名

SELECT eid AS 编号,ename AS 姓名,sex AS 姓名,birthday AS 生日,salary AS 薪资 ,deptid AS 部门 FROM emp;

|  |
| --- |
| 在起别名的时候，AS 关键字可以省略 |

（4）显示不同的记录/合并相同的记录

练习：查询出都有哪些性别的员工

SELECT DISTINCT sex FROM emp;

练习：查询出员工都在哪些的部门

SELECT DISTINCT deptid FROM emp;

（5）查询时执行计算

示例：计算2+3-4+5\*6/3

SELECT 2+3-4+5\*6/3;

练习：查询所有员工的姓名及年薪

SELECT ename AS 姓名, salary\*12 AS 工资 FROM emp;

练习：查询员工资加500 年薪5000

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

(6)查询结果集排序

示例：查询所有部门表数据，结果集按照标号从小到大排序

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

示例：查询所有部门表数据，结果集按照标号从大到小排序

SELECT \* FROM dept ORDER BY did DESC;#descendant

练习：查询所有员工的工资从高到低

SELECT \* FROM emp ORDER BY salary DESC;

练习：查询员工生日降序

SELECT \*FROM emp ORDER BY birthday DESC;

练习：查询员工姓名升序

SELECT \* FROM emp ORDER BY ename ASC（升序排）;

练习：查询员工工资降序 工资相同按姓名排序

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

练习：查询员工是所有的列要求女员工显示在前，如果性别相同按照生日的降序排

SELECT \* FROM emp ORDER BY sex,birthday DESC;

|  |
| --- |
| ORDER BY 可以按照数值，字符串、日期时间排序、默认是按照升序排序（ASC） |

（7）条件查询

示例：查询出编号为5的员工所有列

SELECT \* FROM emp WHERE eid=5;

练习；查询出姓名为 king的员工的编号，工资，姓名、生日

SELECT eid,salary,ename,birthday FROM emp WHRER ename='king';

练习：查询出20部门下员工所有列。

SELECT \* FROM emp WHERE deptid='20';

练习：查询出工资在6000以上员工所有列

SELECT \*FROM emp WHERE salary >=6000;

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

练习：

SELECT \*FROM emp WHERE birthday>='1993-1-1';

练习：查询出不在10 号部门的员工

SELECT \* FROM emp WHERE deptid!=10;

练习：查询出没有明确部门的员工所有列

SELECT \* FROM emp WHERE deptid is Null;

练习：查询出有明确部门的员工所有列

SELECT \* FROM emp WHERE deptid IS NOT Null;

练习：查询出工资在6000以上的男员工所有列

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

练习：查询工资在7000到10000之间的员工

SELECT \* FROM emp WHERE salary >=7000 AND salary <=10000;

SELECT \* FROM emp WHERE salary BETWEEN 7000 AND 10000;

练习：查询不在工资在7000到10000之间的员工

SELECT \* FROM emp WHERE salary NOT BETWEEN 7000 AND 10000;

练习:查询员工生日在1990-01-01之前 和生日在1995-12-31

SELECT \* FROM emp WHERE birthday<='1990-01-01' OR birthday >='1995-12-31';

练习：查询出20号和30号员工所有列

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

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

查询出不在20号和30号员工所有列

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

|  |
| --- |
| IS NULL /IS NOT NULL  AND/OR  BETWEEN..AND/ |

（8）分页查询

假如查询的结果集有太多的数据，一次显示不完，可以使用分页显示。

需要两个条件：当前的页码，每页的数据量

每页的开始=（当前的页码-1）\*每页数据量

|  |
| --- |
| SELECT \* FROM emp LIMINT start,count;  start:每页的开始  count:每页数据量 |

假设每页显示5条数据

第1页：SELECT \* FROM emp LIMIT 0,5;

第2页：SELECT \* FROM emp LIMIT 5,5;

第3页：SELECT \* FROM emp LIMIT 10,5;

练习：

第一页SELECT \* FROM emp LIMIT0,6;

第二页SELECT \* FROM emp LIMIT6,6;

第三页 SELECT \* FROM emp LIMIT12,6;

课后任务

(1)复习今天内容

(2)练习：

删除代码，保留注释，重新编写代码。

(3)查看学子商城数据库资料，尝试创建

# JS-day01

1.简单查询——模糊条件查询

示例：查询出姓名中含有字母e的员工所有列

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

练习：查询出姓名中倒数第2个字符为e的员工所有列

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

练习;查询出姓名中以e结尾的员工所有列

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

|  |
| --- |
| SQL中提供了两个模糊查询的的字符%可以匹配任意个字符 >=0.  \_可以匹配任意1个字符 =1  注意：在模糊条件查询中必须使用关键字LIKE,不能使用= |

2.复杂查询

（1）聚合查询/分组查询

示例：查询出所有员工的数量

SELECT COUNT(eid) FROM emp;

SELECT COUNT(\*) FROM emp; #推荐用法

练习：使用员工的姓名计算员工数量

SELECT COUNT(ename) FROM emp;

练习：使用员工部门编号计算员工的数量

SELECT COUNT(deptid) FROM emp;

查询所有男员工的数量

SELECT COUNT(\*) FROM emp WHERE sex='1';

|  |
| --- |
| 聚合函数  函数就是一个功能体，需要提供若干个数据，产生某个结果。——饺子机  COUNT 总数量  SUM 总和  AVG() 平均  MAX 最大  MIN 最小 |

练习：查询出所有员工的工资总和

SELECT SUM(salary) FROM emp;

练习：查询出所有员工的平均工资

SELECT AVG(salary) FROM emp;

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

练习:查询出男员工工资最高的

SELECT MAX(salary) FROM emp;

练习：查询出年龄最大的员工

SELECT MIN(birthday) FROM emp;

|  |
| --- |
| 分组查询：只能查询分组条件和聚合函数 |

练习：查询出男女员工的平均工资，最高工资

SELECT sex, AVG(salary),MAX(salary) FROM emp GROUP BY sex;

练习：查询出每个分部门的员工数量，最高工资，最低工资

SELECT deptid COUNT(\*),MAX(salary),MIN(salary) FROM GROUP BY deptid;

|  |
| --- |
| 函数补充  YEAR() 获取日期中的年份  MONTH() 获取日期中的月份 |

练习：查询出1993年出生的员工所有列

SELECT \*FROM emp WHERE YEAR(birthday)=1993;

练习：查询出5月份出生的员工的所有列

SELECT \*FROM emp WHERE MONTH(birthday)=5;

(2)子查询

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

示例：查询出研发部员工的所有列

步骤1：查询出研发部的部门编号

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

步骤2：根据研发部部门编号查询员工

SELECT \*FROM emp WHERE deptid=10;

综合：

SELECT \*FROM emp WHERE deptid=( SELECT did FROM dept WHERE dname ='研发部'

);

练习：查询出比tom工资高的员工有哪些

SELECT salary FROM emp WHERE ename='tom';

SELECT \*FROM emp WHERE salary>'6000';

综合：

SELECT \* FROM emp WHERE salary>( SELECT salary FROM emp WHERE ename='tom');

练习：查询出和tom同一年出生的员工

步骤1：查询出tom出生的年份

SELECT birthday FROM emp WHERE ename='tom';

步骤2：查询出1990年出生的员工

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

综合

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

(3)多表查询

示例：查询出所有员工及其部门名称

SELECT ename,dname FROM emp,dept;

错误：笛卡尔积

多表查询如何避免笛卡尔积：添加查询条件

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

|  |
| --- |
| 上述多表查询语法是 SQL—92中的，无法查询出有部门的员工，也无法查询出没有员工的部门。  SQL—99中提出了新的多表查询语法。 |

（1）内连接 INNER JOIN ...ON 和SQL-结果一致

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

（2）左外连接 LEFT OUTER JOIN ..ON

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

查询结果是左侧所有的记录都显示；OUTER 可以省略

（3）右外连接 RIGHT OUTER JOIN ..ON

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

查询结果右侧表中所有记录都显示，OUTER 可以省略

（4）全连接 FULL JOIN ON

显示左侧和右侧所有的记录——mysql 不支持

UNION 合并相同的项

UNION ALL 不合并相同的项

(1)(SELECT ename,dname FROM emp LEFT OUTER JOIN dept ON deptid=did) UNION (SELECT ename,dname FROM emp RIGHT OUTER JOIN dept ON deptid=did);

(2)(SELECT ename,dname FROM emp LEFT OUTER JOIN dept ON deptid=did) UNION ALL (SELECT ename,dname FROM emp RIGHT OUTER JOIN dept ON deptid=did);

学习一门编程语言的基本步骤

（1）了解背景知识：历史、现状、特点、引用场景

（2）搭建开发环境，编写 hello world

（3）变量和常量

（4）数据类型

（5）运算符

（6）逻辑结构

（7）通用小程序

（8）函数和对象

（9）第三方的库、框架

（10）实用的项目

有基础的学员

程序员必做的50道题

JavaScript

1.JS的概述

（1）历史

1995年，JS出现在Netscape 的浏览器中

1997年，JS提交给了ECMA，出现了ECMAcript 标准规范

ActionScript(flash 语言)

2009年，JS遵循CommonJS规范，开始想向服务器端发展

（2）现状

即可运行在客户端浏览器，也可以运行在服务器端NodeJS

（3）特点

解释型语言， 编译一行执行一行

弱类型语言

基于对象

跨平台

（4）应用场景

制作浏览器的交互效果

创建一个APP

创建web服务器，操作数据库等服务器应用

2D绘图

2.JS的执行环境

（1）浏览器自带的JS解释器（引擎）

（2）NodeJS下的JS解释器（http://nodejs.org nodes 的下载地址）

在命令行下node -v 查看当前的NodeJS版本号

把js文件嵌入到html文件中

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

NodeJS下：

W+R cmd

node C:/xampp/.../01\_hello.js 回车

3.JS语法规范

（1）JS 脚本区分大小写

（2）每行结尾的分号可加可不加 习惯都加 最好加上

（3）单行注释(//) 和多行注释(/\*...\*/)

4.变量

变量是用于存储数据的容器

x=1

y=2

(1)声明变量

var x=1;在内存在创建空间，名称叫 x;把数据1存放到空间x中。

练习：声明多个变量，分别保存一个员工的编号，姓名、性别、生日、工资、部门编号 ，并打印出来。

//声明变量

var eid=1;

var ename='tom';

var sex=1;

var birthday='1993-5-5';

var salary=7000;

var deptid=10;

console.log(eid,ename,sex,birthday,salary,deptid);

（2）变量命名规则

变量的名称由字母、数字、美元符号，下划线组成的；不能以数字开头。

多个连词之间命名（驼峰命名、下滑线命名）

驼峰命名法：userName userAge userPassword

下滑线命名：user\_name user\_age

不能使用关键字和保留字作为变量名



（3）声明变量未赋值

var num；

变量声明后未赋值，结果为undefined

弱类型语言，声明变量的时候不要指定数据类型，后期可以存储任意的类型。

var num;

num =1;

console.log(num);

num=5;

console.log(num);

（4）一次性声明多个变量

var a=1,b=2,c;

console.log(a,b,c);

练习：声明多个变量，分别保存语文、数学、总成绩；最后打印总成绩。

var chesen,math,total;

chesen =101;

math=121;

total= chesen +math;

console.log(total);

（5）声明常量

一但声明不能重新赋值

例如：圆周率、春节的日期...

const PI=3.14;

console.log(PI);

课后任务

（1）复习今天内容，整理思维导图

（2）练习

声明变量保存圆周率的半径，声明常量保存圆周率；声明变量保存圆的面积和周长，并打印出来。

var r=3;

const PI=3.14;

var s=PI\*r\*r;

var c=PI\*r\*2;

console.log(s,c);

（3）预习js的数据类型和运算符。

# JS-day02

复习（2019.4.9）

复杂查询

聚合查询/分组查询/子查询/跨表查询

查询所有男员工的所有列，获取工资最高的前三个人。

SELECT \* FROM emp

WHERE sex=1

ORDER BY salary DESC

LIMIT 0,3;

开发环境

浏览器/NodeJS

变量 var a=1; 字母、数字、下划线、美元符号，不能以数字开头。

常量 const PI=3.14;

学习一门编程语言的基本步骤

(1)了解背景知识：历史、现状、特点、应用场景

(2)搭建开发环境，编写hello world

(3)变量和常量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方库、框架

(10)实用的项目

1.数据类型

数据分为原始类型和引用类型

原始类型分为数值型、字符串型、布尔型、未定义型、空

（1）数值型

分为整型和浮点型

整型在内存中占4个字节，浮点型占8个字节

八进制 以0开头， 例如 013 -> 11 console.log(013);

十六进制 以0X开头， 例如0XA ->10 console.log(0XA);

A~F 代表10~15 不区分大小写

0XFF ->255

浮点型分为普通小数和指数型小数

3.14E6 ->314000000

3.14E-6 ->0.00000314

typeof 检测数据类型

var PI=3.14E6

console.log(PI,typeof PI);

//打印 a 字符类型也可 typeof

console.log('a',typeof 'a');

（2）字符串型

数据被引号包含就是字符串类型，不区分单双引号

查看任意一个字符的Unicode码

'a'.charCodeAt() // 97 console.log('a'.charCodeAt());

（3）布尔型

true/false

在程序中表示真或者假

一般用于是否的结果，例如是否登录、是否注册、是否在售，是否在售，是否为会员.....

（4）未定义型

声明了变量未赋值，结果就是undefined

var a;

console.log(a);

（5）空——null

常和应用类型数据一起使用

2.数据类型转换

（1）隐式转换

①数字+字符串 数字转换成字符串

1+'2' //'12'

var str1=1+'2';

console.log(str1,typeof str1);

var str2='3'+'4';//'34'

console.log(str2,typeof str2);

②数字+布尔型 布尔型转成数字 true->1 false->0

3+true //4

3+false //3

③字符串+布尔型 布尔型转成字符串

'5' + true // '5true'

JS中加号（+）的作用

执行法运算

执行字符串拼接

减法（-）、乘法（\*）、除法（/）隐式换行

将运算符两端的数据转为数值型（自动调用 Number）,如果转换失败，返回NaN(Not a Number),不是数字。任何值和NaN执行减乘除结果返回NaN

练习：查看以下程序的运行结果

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

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

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

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

(2)强制转换

①将任意的数据强制转为数值型

Number()

|  |
| --- |
| Number('a'); //NaN  Number('1'); //1  Number('1a'); //NaN(not a Number) parseInt('1a'); //1  Number(undefined); //NaN  Number(null); //0  Number(true); //1 |

②将数据转为整型

parseInt()第一个是数字就转为第一个数

常用于将字符串或数字转为整型，其它的数据返回NaN；如果要转换的字符串以非数字开头，也返回NaN

③将数据转为浮点型parseFloat() 和parseInt的用法基本一致，只是转的数据是浮点型。

④数值型和布尔型转为字符串型

toString();

var num=10;

num.toString();//'10'

3.运算符

表达式：由运算符连接的操作数据，所组成的形式。

运算符分为算术运算符、比较运算符、逻辑运算符、位运算符、赋值运算符、三目运算符

（1）算术运算符

+ - \* / % ++ --

% 取余

++ 自增，在原来的基础上加1

- - 自减，在原来的基础上减1

console.log(num++);先打印num的值，再执行自增

console.log(++num);先执行自增，再打印num的值

练习：

var num=3;

console.log(num -- + --num);//4

先取出num的值3；然后执行自减，num的值变成2；先执行自减，num变成了2，然后再取出num的值，也就是1

（2）比较运算符

< > >= != == ===(全等于) ！===(完全不等于)

返回一个布尔型的值

= = 比较两个值是否相等

=== 不仅比较值，还会比较类型是否相等

|  |
| --- |
| 如果数据类型不同，会发生数据类型转换 |

3>'10' 字符串转成数字10

'3'>'0' ；两个字符串比较的是Unicode码

'3'>'10'比较是3的Unicode码和'10'的第一位 1的Unide码比较

3>'10a' //false 实际上是把'10a'进行隐式转化Number('10a')返回的是NaN，3和NaN比较是false

3<'10a'//false

3=='10a'//false

3！='10a' //true

'10a' -> NaN

NaN 和任何值比较（ > < >= <= == ===)都返回falseNaN==NaN 返回false

（3）逻辑运算符

或着(||) 关联的两个条件只需要满足其一结果是true,否则false

并且(&&) 关联的两个条件都是true,结果是true

非（反向）! !false ->true !true->false

**逻辑短路**

&& 当第一个条件为false的时候，就不需要再执行第二个条件

|| 当第一个条件为true的时候，需要执行第二个条件

逻辑短路无需关注最终结果是true还是false,重点在

练习：以下程序是否回报错

|  |
| --- |
| var num=3;  num>5 && console.log(a);  num<1 || console.log(a); |

练习：声明变量保存一个人的年龄，如果满18岁，打印'成年人'

var age='19';

age>=18 && console.log('成年人');

（4）位运算符（了解）

在执行运算的时候用，计算机会把数据转成二进制进行运算。

1 10 11 100 101 110 111 1000 1010

100110=32+4+1=38

按位与（&）上下两位都是1， 结果是1 ，否则是0；

按位或（|）上下两位含有1，结果是1，否则是0；

7|9 1111

0111

1001

13|20 11011

01011

10010

按位异或（^）上下两位不同为1，相同为0

4^7 11 3

100

111

5^8 1101 13

0101

1000

按位右移（>>）删除二进制的最后一位或者多位，缩小到原来的一半或者更多

按位左移（<<）在二进制的最后添加0，成倍增加

课后任务

(1)复习今天内容，整理思维导图

(2)练习：

声明变量保存一个年份，判断这个年份是否为闰年，结合逻辑短路，如果是闰年，打印"闰年".

闰年：4年一个闰年(能被4整除)，并且不能被100整除，或者能被400整除。

var year=2020;

//如果是闰年，打印'闰年'

//如果前边是true，执行后边

// &&

//先执行&& 再执行||

//(year%4==0 && year%100!=0 || year%400==0) && console.log('闰年');

(year%400==0 || year%4==0 && year%100!=0) && console.log('闰年');

(year%400==0 || year%4==0 && year%100!=0) || console.log('平年');

(3)预习逻辑结构 if if-else switch '

# JS-day03

复习（2019.4.10）

数据类型

原始类型和引用类型

数值型、字符串型、布尔型、未定义型、空

检测数据类型 typeof

数据类型转换

隐式转换

+ - \* / % ++ -- 3>'10'

强制转换

Number()/parseInt('3a')/parseFloat()/num.toString()

运算符

算术运算符 + - \* / % ++ --

比较运算符 > < >= <= == != === !==

逻辑运算符 || && ! 逻辑短路

位运算符 & | ^ >> <<

赋值运算符

三目运算符

学习一门编程语言的基本步骤

(1)了解背景知识：历史、现状、特点、应用场景

(2)搭建开发环境，编写hello world

(3)变量和常量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方库、框架

(10)实用的项目

1.赋值运算符

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

练习：声明变量保存商品的价格，价格在原来的基础之上打八折，然后打印价格

var price=100;

price\* = 0.8;

console.log(price);

2.三目运算符

由三个表达式或者数据组成的形式

条件表达式 ？ 表达式1：表达式2

如果条件表达式为true，执行表达式1

如果条件表达式为false，执行表达式2

总结运算符优先级

3.浏览器端函数

alert () 弹出警示框（消息框）

js脚本写入alert('hello web前端'); htm调用

prompt () 弹出提示框（输入框），需要使用变量接收输入的值；值得类型是字符串。

4.流程控制

程序=数据+算法；

程序分为顺序执行、选择执行、循环执行

（1）if语句

|  |
| --- |
| if(条件表达式){  语句1  } |

如果条件表达式为true，会执行后边的语句

在条件表达式中，有一些值默认转成false

0、NaN、'' 、undefined、null

（2）if-else语句

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

如果条件表达式为true，执行语句1，否则条件表达式2为false，执行语句2.

|  |
| --- |
| 如果if或者else后只有一行语句，否则后边的大括号可以省略的。 |

练习：使用弹出提示分别输入商品的单价和数量，计算出商品的总价，如果总价满500打八折；使用变量保存账户余额，如果余额足够支付，打印支付成功（pay success），否则打印余额不足(pay error)。

（3）if-else嵌套

用于判断多种情况

|  |
| --- |
| if（条件表达式1）  { 语句1;}elsei（表达式2）{  语句2;  }else ...if（表达式n）{  语句n;  } else{  语句n;  } |

（4）switch -case 语句

是一种特殊的分支语句，可以根据一个表达式的值，来选择执行不同的程序

|  |
| --- |
| switch(表达式){  case 值1：//如果表达式的值是值1  语句1；  break;  ...  case 值n:  语句n:  break;  default:  语句 n+1;//以上所有的表达式和值比较都是false  } |

注意：在case中，表达式和值比较实用的是完全的等于比较，要求值和类型都满组才是true

**对比if-else嵌套和switch-case语句**

相同点：两者都可以用于多项分支语句

不同点：if-else既可以判断相等，也可以判断不等的情况，使用范围更广；switch-case只能用于全等于的比较，结构上更为清晰，执行效率相对高。课后任务

课后任务

(1)复习今天内容，整理思维导图

(2)练习

使用switch-case语句完成成绩评判

原理：让分数除以10，然后取整，在使用语句。

var score=90;

score=parseInt(score/10);

switch (score){

case 10:

case 9:

console.log('优秀');

break;

case 8:

console.log('良好');

break;

case 7:

console.log('中等');

break;

case 6:

console.log('及格');

break;

default:

console.log('不及格');

}

(3)预习while，do-while，for循环，循环嵌套

# JS-day04

复习

运算符

赋值运算符 = += -= \*= /= %=

三目运算符 条件表达式 ? 语句1 : 语句2

流程控制

顺序执行、选择执行、循环执行

if(条件表达式){ 语句; }

if(条件表达式){ 语句1; }else{ 语句2; }

if(条件1){ 语句1; }else... if(条件n) { 语句n; }else{ }

false: 0 NaN '' undefined null

switch(表达式){

case 值1:

语句1;

break;

...

default:

语句n;

}

学习一门编程语言的基本步骤

(1)了解背景知识：历史、现状、特点、应用场景

(2)搭建开发环境，编写hello world

(3)变量和常量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方库、框架

(10)实用的项目

**1.循环**

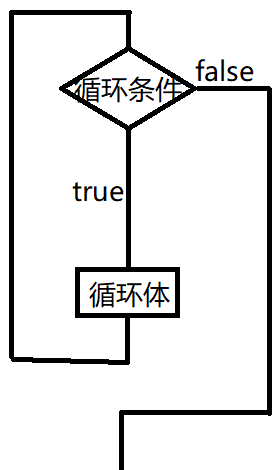
循环：就是一边又一遍执行相同或者相似的代码

循环的要素

循环条件：重复的次数

循环体：执行的代码

（1）while循环



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

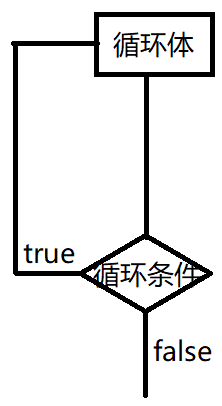
break 可以结束任何形式的循环，break执行后，后边所有的循环体和循环条件不再执行

练习：声明保存一个数字，循环循环提示框弹出提示框，输入数字；如果输入的比保存的大，警示框提示“big”,如果输入的比保存的小，警示框提示“small”，否则警示提示框提示'right'，结束循环。

03\_break.html

思路：循环前声明变量保存数字，无限循环的循环体中弹出提示框，获取输入的值，再和之前的变量比较(if-else嵌套)

（2）do-while 循环



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

不管循环条件是否为true,都会先执行一次循环体

练习：打印100~1之间的所有整数

var i=100;

do{

console.log(i);

i--;

}while (i>=1);

练习：70 75 80 85 90 95 100

var i=70;

do{

console.log(i);

i+=5;

}while(i<=100);

练习：计算1~100之间所有能被3整除的数字的和

var i=1;

var sum=0;

do{

//i就是1~100之间所有的整数

//所有能被3整除的数字

if(i%3==0){

sum+=i;

}

i++;

}while(i<=100);

console.log(sum);

练习：计算10的阶乘

var i=10;

var ride=1;

do{

ride\*=i

i--;

}while(i>=1)

console.log(ride);

练习：声明变量保存密码'123456'循环弹出提示框，输入密码，如果输入正确，警示框弹出'login success,并结束循环 break'

（4）for循环

|  |
| --- |
| for（初始值;循环条件；增量）{  循环体  } |
| ①执行初始值  ②判断循环条件  ③如果循环条件是true执行循环体，是false结束循环  ④如果执行完循环体，执行增量  ⑤重新执行第二步 |

2.break和continue

break:结束循环，后续不会在执行其他的循环

continue：跳出本次循环体，继续执行下一次循环

3.循环嵌套

while、do-while、for 循环三者之间可以互相嵌套

课后任务

(1)复习今天内容，整理思维导图

(2)练习

打印99乘法表

for(var j=1;j<=9;j++){

for(var i=1,str='';i<=j;i++){

str+=i+'\*'+j+'='+(i\*j)+' ';

}

console.log(str);

}

打印本世纪前10个闰年（当打印完第10个，提前结束循环break）

计算1~100之间所有整数和，当和大于4000的时候，提前结束循环（break），打印总和和当前i的值

for(var i=1, year=2000;year<=2100;year++){

if(year%4==0&&year%100!=0||year%400==0){

console.log(year);

i++;

if(i>10){

break;

}

}

}

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

sum+=i;

if(sum>4000){

console.log(sum,i);

break;

}

}

(3)预习js中的自定义函数和作用域

# JS-day05

复习

循环

两要素：循环条件，循环体

while(循环条件){ 循环体 }

do{ 循环体 }while(循环条件)

for(初始值;循环条件;增量){ 循环体 }

break和continue

循环嵌套

for while do-while

|  |
| --- |
| var i=0,sum=0;  do{  i++;  if(i%2==0){  continue;  }  if(i%5==0){  break;  }  sum+=i;  }while(i<10); |

学习一门编程语言的基本步骤

(1)了解背景知识：历史、现状、特点、应用场景

(2)搭建开发环境，编写hello world

(3)变量和常量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方库、框架

(10)实用的项目

**1.函数**

parseInt()/parseFloat()...

**函数分为**：系统函数和自定义函数

自定义函数：

function:功能体，函数，用于封装反复执行的代码，可以接收若干个数据，返回处理的结果。——饺子机

（1）创建普通的函数

|  |
| --- |
| function 函数名称（）{  函数体；//封装的的反复执行的代码  } |

调用

函数名称()

练习：使用函数封装1+2的计算结果，调用3次

function sum(){

console.log(1+2)

}

sum();

sum();

sum();

练习：使用函数封装计算1~100之间所有整数数的和，并调用三次

function getsum(){

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

sum+=i;

}

console.log(sum);

}

getsum();

getsum();

getsum();

（2）创建带有参数的函数

|  |
| --- |
| function 函数名称(参数列表){//形参->形式上的参数  函数体；  } |

调用

函数名称(参数的列表);//实参 ->传递的数据

参数列表：可以是0个或者多个，之间用逗号隔开；创建是的参数称为形惨，调用时的参数称为实参，实参会赋值给形参

形参可以理解为是一个变量，只是未赋值，默认是undefined

练习：创建函数，传递1个参数，计算1~任意数字之间的和

function getsum(n){

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

sum+=1;

}

console.log(sum);

}

getsum(5);

（3）带有返回值得函数

|  |
| --- |
| function 函数名称(参数列表){  return 值;  } |

调用

函数名称(参数名称);

return 表示函数执行后的，返回的结果；可以保存下来。

注意事项：

①如果没有return和return后没有值，结果都是undefined;

②return后的所有代码都不执行

③

练习：创建函数getMax，返回任意两个数字中的最大值

function getMax(a,b){

/\*if(a>b){

return a;

}else {

return b;

}

return a>b ? a :b;

}

var Max=getMax(2,3);

console.log(Max)

练习：创建函数get Max2 返回任意三个数的最大值

function getMax2(a,b,c){

var Max = a > b ? a : b;

return Max>c ? Max : c;

}

var big= getMax2(30,36,40);

console.log(big);

练习:创建函数 getStatus, 传递状态码；返回对应的中文

1-等待付款 2- 等带发货 3- 运输中 4- 已签收 5 -已取消

function getStatus(num){

switch (num)

{

case 1:

return '等待付款'

case 2:

return '等待发货'

case 3:

return '运输中'

case 4:

return '已签收'

case 5:

return '已取消'

default:

return '无法追踪'

}

}

var str= getStatus(2);

console.log(str);

**2.变量作用域**

（1）作用域

变量和函数的可访问范围，分为两种

函数作用域：在函数中使用var关键字声明的变量只能在内部访问。

全局作用域：在函数外部使用var声明的变量，可以在任意的位置访问

**注意：**在函数内使用var声明的变量是局部变量，不使用var声明的变量是全部变量

（2）变量的声明提升

JS程序执行前，会将使用var声明的变量提升到所在作用域的最前面

函数作用域

函数的可访问范围，也分为全局作用域和函数作用域

（1）函数声明提升

和鼻梁声明提升，JS程序在执行前，会将function创建的函数提升到所在作用域的最前边，调用可以在任意合法的位置。

4.递归调用

在函数的内部调用自身

递归要有一个结束的条件，结合return来使用。

练习：使用递归来计算1~任意数字之间的和。

斐波那契数列

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

课后任务

(1)复习今天内容，整理思维导图

(2)练习

使用递归和普通函数来计算斐波那契数列

function fib(n){

//第1项和第2项的值是1

if(n==1 || n==2){

return 1;

}

//其它的项的值=前1项的值+再前1项的值

return fib(n-1)+fib(n-2);

}

var res=fib(5);

/\*

fib(4)+fib(3)

fib(3)+fib(2)+fib(2)+fib(1)

fib(2)+fib(1)+fib(2)+fib(2)+fib(1)

console.log(res);\*/

function fib1(n){

var n1=1,n2=1;

for(var i=3;i<=n;i++){

var m=n2;

n2=n1+n2;

n1=m;

}

console.log(n2);

}

fib1(5);

(3)预习js中的自定义对象

# JS-day06

复习

学习一门编程语言的基本步骤

(1)了解背景知识：历史、现状、特点、应用场景

(2)搭建开发环境，编写hello world

(3)变量和常量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方库、框架

(10)实用的项目

**1.匿名函数**

没有名称的函数 function() { }

（1）创建函数

函数声明方式

|  |
| --- |
| function fu() { } |

函数表达式方式

|  |
| --- |
| var fun =function(参数列表){函数体  }  fun();此时的fun就是函数名称 |

对比fun和fun()的区别

fun() 是函数的调用，返回函数的调用结果

fun 是函数的名称，对应的是函数的整体结构

函数声明和函数表达式创函数的区别

函数声明存在函数提升，调用可以在函数的任意合法位置

函数表达式不存在函数的提升，必须先创建，再调用

练习：使用函数表达式创建函数，计算任意两个数之间所有证书的和，并返回结果

var getSum=function (n1,n2){

for (var sum=0;n1<=n2 ;n1++ ){

sum+=n1;

}

return sum;

}

console.log(getSum(50,100));

（2）匿名函输自调用

目的：创建局部作用域，防止污染全局

|  |
| --- |
| (function(形参列表){  函数体中的变量和创建的函数只能在内部使用  })(实参列表); |

（3）回调函数

将匿名函数以实参形式传递，这个匿名函数就称为“回调函数”；此时的形参就是函数名称，如果调用匿名函数，只需要 形参名称（）

|  |
| --- |
| function fn(a){  //a就是传递的匿名函数的名称  a()//执行传递的函数函数体的代码  }  fn(function(){....}); |

练习：创建函数，传递两个参数，实参都是以匿名函数的形式传递，在匿名函数中分别返回数字；计算两个数字相加的和并打印出来

**2.全局函数**

**parseInt() 将数据转为整型**

**parseFolat() 将数据转为浮点型**

**encndeURI()对一个URL中的中文进行编码**

**decndeURI()对已经编码的URL进行解码**

**isNaN()检测一个值是否为NaN是->true不是->false**

**isFinite() 检测一个值是否为有限值 是->true不是 ->false 1/0 INfinity 无穷 无限值**

**eval() 执行字符串表达式的值**

**练习：使用弹出提示框输入一组运算，使用eval来执行该运算，并打印出来**

**3.对象**

是引用类型数据

对象：是一组属性和方法（功能）的集合

哪些是对象

一个手机：属性颜色,品牌，尺寸；方法有打电话，发短信，看视频，玩游戏

一个人

**万物皆对象**

（1）JS中的对象

内置对象：JS提供的

宿主对象：根据JS不同的执行环境来划的

自定义对象：自己创建的对象

（2）自定义对象

对象字面量

内置构造函数

自定义构造函数

4.使用对象字面量创建对象

使用大括号{}创建空对象

属性名和属性值之间用冒号隔

多组属性之间用逗号隔开

属性名中引号可加，如果含有特殊字符必须加引号

|  |
| --- |
| {属性名：属性值，....} |

练习：创建一个对象，添加姓名、性别、年龄

var phone={

color:'black',

brand:'huawei',

size:6.5,

'madeIn':'china'

};

console.log(phone);

var people={

name:'张三',

sex:'男',

age:20

};

console.log(people);

练习：创建一个员工的对象，包含编号、姓名、性别、生日、工资、所属部门编号

var emp={

eid:011,

ename:'李四',

sex:0,

birthday:'1999-9-9',

salary:5000,

'dept-id':20

}

5.访问对象的属性

对象.属性名

对象[' 属性名']

可以添加之前不存在的属性；

如果属性不存在，属性值为undefined

练习：创建一个电脑对象，包含的属性有屏幕尺寸，品牌，颜色，内存大小；获取品牌的属性值，修改颜色的属性值，添加产地属性

6.使用内置函数构造函数创建对象

new Object();创建一个空的对象

需要单独添加每一个属性

对象['属性名']='属性值'

对象.属性名='属性值'

练习：使用内置的构造函数创建用户对象，包含编号，用户名，密码，邮箱，手机

（7）遍历对象的属性

访问所有的属性

|  |
| --- |
| for(var key in ){  key 对象中所有的属性名  对象[key]通过属性名查找对应的属性值  } |

练习：创建对象，包含若干个成绩属性，遍历对象中属性获取总成绩。

提升声明sum用于保存总和。

var scor={

chinese:85,

math:95,

enging:101,

wuli:70,

}

var sum=0;

for ( var key in scor) {

//cosole.log(key,scor[key]);

//把每一个属性值加到sum中

sum+=scor[key]

}

console.log(sum);

（8）对象中的方法

方法也称为成员方法，对应的是一个匿名函数

|  |
| --- |
| var person={  name:'tom',  say:function(){  this.name//this指代当前的方法  }  } |

课后任务

（1）复习今天内容，整理思维导图

（2）练习

创建一个圆的对象，包含半径、圆周率属性；添加获取面积和获取周长的方法。分别返回对应的值。

（3）预习JS中的数组（创建、遍历，API）

# JS-day07

复习

匿名函数 function(){ }

创建函数

var fn=function(){ }

function fn(){ } fn其实也是一个变量

函数自调用 防止污染全局

(function(){ 函数作用域; })();

回调函数 将匿名函数以参数传递

function fn(a){ a() }

fn( function(){ 函数体; } )

全局函数

parseInt/parseFloat/encodeURI/decodeURI

isNaN/isFinite/eval

对象

内置对象/宿主对象/自定义对象

对象字面量/内置构造函数/自定义构造函数

{属性名:属性值,...}

new Object();

访问对象中的属性

对象.属性名 / 对象['属性名']

遍历对象属性

for(var key in 对象){ 对象[key] }

对象中的方法

{ name:'tom', say:function(){ this.name } }

对象.say()

学习一门编程语言的基本步骤

(1)了解背景知识：历史、现状、特点、应用场景

(2)搭建开发环境，编写hello world

(3)变量和常量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方库、框架

(10)实用的项目

1.检测对象中是否含有某个属性

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

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

'属性名' in对象 true->存在 false->不存在

**数组**

**数组是由多个元素组成的集合，每个元素就是一个数据**

**2.创建数组**

（1）数组字面量

[元素1，元素2...]

练习：创建数组，包含多个商品的名称；创建数组，包含多个城市的名称

（2）访问数组中的元素

数组[下标] 下标是从0开始，第一个元素的下标就是0

（3）内置构造函数

newArray(元素1,元素2...);

newArray(3) 初始化元素个数为3，也可以添加更多个元素

练习：创建数组，保存若干个课程名称；

练习：创建数组，初始化长度5，天剑篮球场上的五个位置

（4）数组的长度

数组.length 获取数组中元素的个数

作用：在数组的末尾添加新的元素

数组[数组.length]=值；

练习：创建一个空数组，使用数组长度添加若干个国家的名称

2.数组的分类

数组分为索引数组和关联数组

索引数组以0以上的整数作为下标

关联数组以字符作为下标，只能单独的添加元素

3.遍历数组

（1）for-in

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

练习：创建数组，包含有多个学生的成绩，获取总成绩。

（2）循环

|  |
| --- |
| for(var i=0;i<数组.length;i++){  i 代表下标 0~ 数组长度-1  数组[i] 代表下标对应的元素  }  只能遍历索引数组 |

练习：创建数，包含有多个姓名，把tom全部改为成"汤姆"

创建数组

遍历得到每个元素

判断每个元素是否为tom,如果是，重新赋值为"汤姆"在遍历之后，打印数组

练习：创建数组，包含多个姓名，查询tom出现的次数

练习：创建数组，包含多个数字，获取这组数字的最大值

创建数组 23 85 74 46 95 100

声明变量用于保存最大值，默认把第一个元素放进去

遍历数组

var num=[23,85,74,46,95,100];

for(var i=1,max=num[0];i<num.length;i++){

if(num[i]>max){

max=num[i];

}

}

console.log(max);

4.数组API( application programming interface(应用程序编程接口，预定义好的一些方法或者函数)方法)

toString(); 将数组中的元素按照逗号组合

join()将数组中的元素按照指定字符组合成字串，默认是逗号

concat(arr2,arr3) 拼接多个数组

slice(start,end) 截取数组中的元素，start开始的下标，

end结束的下标，不包含end本身，如果是负数表示倒数；

返回一个数组。

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

var arr=['a','b','c','d','e','f','g'];

var arr1=arr.slice(2,4);

var arr2=arr.slice(-2,-1);

var arr3=arr.slice(1,2);

console.log(arr1.concat(arr2,arr3));

splice(start,count,value1,value2...)删除数组中的元素，

start 开始的下标，count删除的数量，value删除后插入的元素；返回删除的元素，原数组会发生变化。

练习：创建数组a~h,每个字母是一个元素，删除d,e,替换f为m,在下标为1的位置插入字母z

var arr=['a','b','c','d','e','f','g','h'];

console.log(arr.splice(3,2));

console.log(arr.splice(3,1,'m'));

console.log(arr.splice(1,0,'z'))

console.log(arr);

revarse() 翻转数组中的元素

sort() 对数组中的元素进行排序，默认按照Unicode码从大到小排序

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

课后任务

（1）复习今天任务整理思维导图

（2）练习

使用遍历数组翻转数组中的元素

['a','b','c','d'] -> ['d','c','b','a']

使用冒泡排序将数组中的数字从小到大排序循环嵌套

（3）字符串中的API

# JS-day08

复习

检测对象中是否含有某个属性

对象.属性名 === undefined

对象.hasOwnProperty('属性名')

'属性名' in 对象

数组

数据的集合

创建数组

[ ] new Array();

访问数组元素

数组[下标]

数组的长度

数组.length

数组分类

关联数组/索引数组

数组的遍历

for(var key in 数组){ }

for(var i=0;i<数组.length;i++){ }

数组API

toString/join/concat/slice/splice/reverse/sort

学习一门编程语言的基本步骤

(1)了解背景知识：历史、现状、特点、应用场景

(2)搭建开发环境，编写hello world

(3)变量和常量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方库、框架

(10)实用的项目

1.数组API

push() 往数组的末尾添加元素，返回数组的长度

pop() 删除数组尾元素，返回删除的元素

unshift() 往数组的开头的添加元素，返回数组的长度

shift()删除数组开头的元素，返回删除的元素

2.二维数组

数组中的每个元素也是数组

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

访问二维数组中的元素 arr[下标][下标]

3.字符串

包装对象：目的是让原始类型数据行引用类型一样，具有属性和方法。

JS提供了3中对象：String Number Boolean

将任意数据转为字符串

new String() 返回对象，使用和字符没有区别

String() 返回字符串

（1）转义字符—— \

转换字符的意义

用法：在要转义的字符前加

\' 将引号转义普通的字符

\n 将字符 n转义成换行符

\t 将字符t转义成制表符（tab键 效果）

（2）字符串API

length 获取字符串的长度

charAt()获取下标对应的字符，也可以使用数组形式字符串[下标]

练习：遍历字符串，打印出每一个字符。

charCodeAt() 获取某个字符的Uncode码

indexOf(value,start) 查找某个字符串的下标，value要查找的字符串，start开始的下标，默认是0，如果找不到返回-1

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

练习：声明变量保存字符串，检测该字符串中是否含有@，如果有打印"合法的邮箱"，否则打印“不合法邮箱”

toUpperCase() 将英文字母转为大写

toLowerCase() 将英文字母转为小写

练习：声明变量保存4个英文字母，循环弹出提示框，输入验证码（不区分大小写），如果输入正确结束循环。

06\_code.js 06\_conde.html

slice(start,end)截取字符串，start开始的下标，end结束的下标，不包含end本身；如果为负数表示倒数；end为空截取到最后。

练习：声明变量保存邮箱，分别截取出邮箱的用户名和域名tom123@163.com

查找@的下标

0~下标 用户名

下标的下一位~最后一位 域名

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

练习：声明变量保存身份证号，截取出年月日和性别；打印“1998年06月20日 性别女”

110236199806202589

substing(start,end) 截取字符串，start开始的下标，end结束的下标，如果end为空截取到最后；如果下标为负数，自动转成0

对比slice和sustring

slice中下标允许使用负数，substring 会自动将负数转为0；slice中下标的顺序start小于end，substring中下标大小部分顺序

练习：将一个英文单词的首字母转大写，其余的字母转小写

heLL0 ->Hello

截取第一个字符 转大写

截取第二个字符 转小写

结果拼接起来

var str='heLLo';

console.log(str.substring(0,1).toUpperCase()+str.substring(1).toLowerCase());

split(sep) 将字符串按照指定字符分隔成数组，sep是指定的字符

tom123@163.com

练习：使用split分隔邮箱，获取用户明和域名

4.匹配模式（掌握）

作用：查找、替换字符串

match（value） 用于查找匹配的字符串，返回所有满足条件的元素，组成数组 /china/ig

i ->ignore（不区分大小写）

g ->global（全局查找）

search(value) 用于查找满足条件的第一个字符串的下标，如果找不到返回-1

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

5.Math 对象

不需要使用new创建，可以直接使用

PI 取圆周率

abs() 取绝对值

floor() 向下取整

ceil() 向上取整

round() 四舍五入取整

max() 取一组数字的最大值

min() 取一组数字的最小值

pow() 求x的y次幂

random() 取随机数 >=0 <1

练习：随机产生0~9之间的一个整数

console.log(Math.floor(Math.random()\*10));

课后任务

（1）复习今天内容，整理思维导图

（2）练习：

创建数组，包含有若干元素，每次随机取出1个元素；数组有10元素 随机取0~9，作为下标

创建数组，包含所有的字母（大写，小写）和数字（0~9），随机取4个元素（循环4次），放入一个新数组push。如何不重复？

（3）预习Date对象

# JS-day09

复习

push/pop/unshift/shift

字符串

包装对象 String

转义字符 \' \n \t

API

length/toUpperCase/toLowerCase

charAt()/charCodeAt/字符串[]/indexOf/lastIndexOf

slice/substr/substring/split

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

匹配模式

match(/china/ig)/search/replace

Math对象

PI/abs/floor/ceil/round/max/min/pow/random

学习一门编程语言的基本步骤

(1)了解背景知识：历史、现状、特点、应用场景

(2)搭建开发环境，编写hello world

(3)变量和常量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方库、框架

(10)实用的项目

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

获取每个单词，按空格将字符串分隔为数组

遍历数组，将每个单词首字母取出转大写，其它的字母取出转小写，拼接起来替换之前的字符串'hOw aRE yOu'->'How Are You'

获取每个单词，按空格将字符串分隔为数组

遍历数组，将每个单词首字母取出来转为大写，其他的字母取出转小写，拼接起来替换之前的字符串

var str='hOw aRE yOu';

//将英文分隔成多个单词

//字符串按照空格分隔为数组

var arr=str.split(' ');

//遍历数组，将每个自单词的首字母转大写，其余字母转小写

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

//取首字母

var first=arr[i].slice(0,1).toUpperCase();

//取取余字母

var last=arr[i].slice(1).toLowerCase();

//把转换的结果替换之前的

arr[i]=first+last;

}

//将数组arr转为字符串

console.log(arr.join(' '));

1.Date对象

（1）创建对象

用于对日期时间进行存储和计算

new Date ('2019/4/18 10:30:40')

new Date('2019,3,18,10,30,40');//第二参数月份范围 0~11

new Date() 存储当前的系统时间

new Date(15000000000)存储的是距离计算机元年的毫秒对应的日期时间

1970-06-23T14:40:00.000Z

（2）获取Date对象中的日期时间

getFullYear()/getMonth 范围0~11/getDate/getHours/getMinutes/getSeconds/getMilliseconds

getDay 范围0~6/getTime 距离计算机元年毫秒数

练习：创建Date对象，保存当前系统的日期时间；根据对象打印 2019年04月19日 星期四

var now=new Date();

var year=now.getFullYear();

var month=now.getMonth()+1;

var date=now.getDate();

var day=now.getDay();//0~6

//判断月份是否小于10

if(month<10){

//增加前导0

month='0'+month;

}

//创建数组，保存中文的星期

var week=['星期日','星期一','星期二','星期三','星期四','星期五','星期六'];

console.log( year+'年'+month+'月'+date+'日 '+week[day] );

（3）转为本地字符串

toLocaleDateString()//年-月-日

toLocaleTimeString()//时：分：秒

toLocaleString()//年-月-日 时：分：秒

（4）修改Date对象中的日期时间

setFullYear/setMonth/setDate/setHours/

setMinutes/setMillseconds/setTime

setTime修改后，可能影响其它的日期时间

练习：创建Date 对象，保存员工的入职时间'2019/4/18';

3年后合同到期,计算到期时间；合同到期前一个月续签合同，如果周末，提前到周五，计算续签时间。

入职时间：2019-4-18

到期时间：2022-4-18

续签时间：2022-3-18

2.Number 对象

new Number() 将数据转为数值型，返回对象

Number () 将数据转为数值型，返回数值

Number.MAX.VALUE 获取Number的最大值

Number.MIN\_VALUE 获取Number的最小值

toFixed(n) 保留小数点后n位

toString(n) 将数值转为字符串，n转换的进制

3.Boolean 对象

newBoolean() 将数据转为布尔型，返回对象

Boolean() 将数据转为布尔型，返回布尔型

!!值 隐式将数据转为布尔型

toString() 将布尔型数据转为字符串

4.错误处理

//var a=1；

Syntax Error :语法错误，错误的使用了中文，缺少括号等；

出现后所有代码都不执行

//console.log(b);,

ReferenceError：引用错误，使用了未声明的变量，属于运行时出现的错误，影响后续代码的执行

TypeError:类型错误，错误的使用了数据，例如把变量。数组当做函数使用，影响后续代码的执行

RangeError:范围错误，参数的使用超出了范围；属于运行时的错误，影响后续代码的执行

|  |
| --- |
| try{  尝试执行的代码，可能产生错误  }catch(err){  err捕获到的错误  具体处理错误的内容  } |

练习：声明一个变量add未赋值；尝试调用add函数；如果执行错误。给add赋值一个匿名函数，然后再调用add

5.ES6

ECMAScript6

ES6 标准入门

http://es6.ruanyifeng.com/

(1)块级作用域

使用let关键字声明变量，只能在块级作用域下使用，不能被外部访问，不存在变量提升。

块级作用域：{ } for、while、do-while、if...

防止污染全局

(2)箭头函数 =>

是回调函数的另一种写法，和匿名函数不完全一样。山西陕西

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

如果箭头函数的函数体中只有一行代码，并且是return形式的，可以简化为 sort( (a,b)=>a-b )

练习：创建函数add，传递两个参数，每个参数都是回调函数，在回调函数中返回一个数字；在函数add中计算两个数字相加的和。

function add(a,b){

console.log(a()+b());

}

add(function(){return 7;},function(){return 8;})

function add(a,b){

console.log(a()+b())

}

add(()=>7,()=>8);

课后任务

(1)复习今天内容，整理思维导图

(2)练习

计算2019-5-1 9:30:00 距离 2019-12-25日相差的天,小时，分钟，秒钟。

(3)复习服务器，预习nodejs中的模块概念

# NODEJS-day01

复习

学习一门编程语言的基本步骤

(1)了解背景知识：历史、现状、特点、应用场景

(2)搭建开发环境，编写hello world

(3)变量和常量

(4)数据类型

(5)运算符

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方库、框架

(10)实用的项目

1.ES6

（1）函数中的参数

ES6 允许为参数设置默认值，如果没有传递实参，自动调用形参的默认值

（2）模板字符串

|  |
| --- |
| `一组字符串你，可以直接写JS ${js 表达式}` |

2.nodejs 概述

nodejs基于谷歌V8引擎（速度效率非常高）运行在服务器端的语言 ，基于JS

对比JS和nodejs

（1）JS 运行在客户端浏览器，存在多个浏览器，容易产生兼容性的问题；nodejs在服务器端只有一个环境，不存在兼容性。

（2）两者都有内置（ES）对象、自定义对象、宿主对象（根据执行环境的不同划分）

（3）JS用于网页中的交互效果，nodejs用于服务器的操作，例如web服务器创建，数据库操作，文件操作等

nodejs.org 英文官网

nodejs.cn 中文网

nodejs的特点

单线程逻辑处理

非阻塞的异步I/O处理

支持数万个并发连接

nodejs的应用场景

基于社交网络的大规模web应用

nodejs不适合CPU密集型的应用

递归、数据加密解密、数据挖掘和分析数据

nodejs的执行方式

脚本模式 node C:/xampp/../1.js

交互模式

node 回车 进入交互模式

两次ctl +C 或者 输入 .exit 退出交互模式

3.全局对象

全局作用域下的变量就是全局对象下的属性，全局对象下的函数就是全局对象下的方法，可以使用全局对象来访问

nodejs:global

在交互模式下，声明的变量和创建的函数都属于是全局对象下的，可以使用global来访问；

例如：var a=1； 可以global.a访问

在脚本模式下，文件中的声明的变量和创建的函数都不是全局对象下的，一个文件默认会创建一个独立的作用域，叫做文件（模块）作用域，防止污染全局，不能使用global来访问

js:window

在浏览器下，文件中声明的变量或者创建的函数都属于是全局作用域下的，会污染全局

（1）console.log对象——控制台

global.console.log()打印日志

global.console.info()打印消息

global.console.warn()打印自定义的警告

global.console.error()打印自定义错误

global.console.log()打印自定义的错误

global.console.time('自定义的字符创')开始计时

global.console.timeEnd('自定义字符串')结束计时

自定义字符串前后保持一致

练习：计时查看for 、while、dowhile循环 10000的耗时

（2）process对象——进程

process.arch 查看当前CPU的架构

process.platform 查看当前的操作系统

process.env 查看当前系统的环境变量

process.version 查看当前nodejs的版本号

process.pid查看当前的进程号

process.kill()结束某个编号的进程

（3）Buffer对象———缓冲区

一块用于临时存储数据的内存区域，可以存储文件数据、网络上传输的资源（视频、在线直播......）

创建Buffer

Buffer.alloc(5,'abcde');

buffer 数据 <Buffer 61 62 63 64 65>

将buffer数据转为字符串

buf.toString()

4.模块

模块就是一个封装好的的功能体

在nodejs下模块分为自定义模块、核心模块（官方提供的）第三方模块

npmjs.com

（1）自定义模块

在nodejs下，任意一个js文件都是一块模块，文件中的代码默认被认为函数所包含

|  |
| --- |
| (function(exports,require,module,\_\_filename,\_\_dirname){  //程序员编写的代码  }) |
| \_\_filename 当前模块的完整路径和文件名称  \_\_dirname 当前模块的完整路径不含文件名称  require() 引入一个模块  module 指代当前的模块对象  module.exports 当前模块导出的对象，包含供其它模块使用的属性和方法  exports（导出） 等价于module.exports |

练习：在07\_2.js声明变量b，创建函数fn，导出变量和函数。

var b=2;

var c=4;

function fn(){

console.log(3)

}

module.exports.myb=b;

//导出函数：导出函数的结构，函数名称

module.exports.myfn=fn;

exports.c=c;

课后任务

（1）复习今天内容，整理思维导图

（2）练习

创建两个模块main.js（主模块），circle.js(功能模块)，

在功能模块中创建两个函数，传递1个参数半径，计算圆的周长和面积，导出这连个函数；在主模块中引入功能模块，并调用两个方法

（3）预习核心模块的querystring,url,fs

# NODEJS-day02

复习

1.模块分类

|  |  |  |
| --- | --- | --- |
|  | 以路径开头 | 不以路径开头 |
| 文件模块 | require('./circle.js')常用于用户自定义的模块，如果后缀名是js,则可以省略 | require('url')  用于引入官方提供核心模块 |
| 目录模块 | require('./02\_2')在02\_2目录下寻找  package.json;寻找main属性对应的文件，找不到会去引入index.js | require('04\_2')自动到node\_modules中寻找目录模块04\_2,如果找不到会到上一级目录下寻找，直到顶级（当前磁盘）目录，常用于第三方模块 |

练习：创建模块03\_1.js,引入当前目录下的 03\_2目录模块；在03\_2下创建test.js，导出一个函数fn(计算两个数字相加)，在03\_1.js调用

练习：在05目录下创建模块05\_1.js，引入不带路径模块05\_2,在05\_2下含有hello.js文件（打印一句话）

2.npm 和包

包（package）:是一个目录模块，里边包含多个文件，其中有一个文件命名为package.json ,是包说明文件，包含有包的信息

npm:包管理工具，安装nodejs的时候会附带安装，用于管理包，包含下载，上传，升级..

npm 官网：www.npmjs.com

（1）切换目录

cd 完整路径

如果要进入其它盘符

例如：d: 回车

进入指定的目录，按下shift键，在空白区域单击鼠标右键->在此处打开Powershell

（2）使用npm下载安装第三方包

npm install 包名称

（3）生成package.json

npm init -y 自动生成一个package.json文件，后期使用npm 安装的包都会记录到这个文件中

npm install 自动安装package.json 文件中记录的包

3.全局函数

parseInt/parseFoat/encodeURI/decodeURI/isNaN/isFinite/eval

（1）一次性定时器

|  |
| --- |
| var timer=setTime(回调函数，间隔的时间)；  当间隔的时间到了，执行回调函数，单位是毫秒  清除  clearTime(timer); |

（2）周期性定时器

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

（3）立即执行定时器

|  |
| --- |
| 开启  var timer=setImmediate(回调函数)  回调函数会放入到队列中  档主线程程序执行完，才会执行队列中的内容  清除  cleatImmediate(timer) |
| process.nextTick(回调函数)  在主线程的末尾执行 |

4.querystring 模块----查询字符

属于核心模块，nodejs 官方提供的，可以直接引入，不需要创建

www.codeboy.com/detail.html?lid=5&name=dell

浏览器向服务器发请求，传递数据的一种方式

parse()将查询字符串格式化为对象

stringify() 将对象转换成字符串

练习：获取百度查询中查询字符串中的关键字

e=utf-8&tn=baidu&wd=电脑

https://www.codeboy.com:80/product/detail.html?lid=5&name=dell

5.url模块

parse() 将url格式化为对象

protocal 协议

hostname 服务器域名/IP地址/

pathname 文件在服务器上的路径

query 查询字符串

forma()将对象转成url

query 属性对应的是对象

练习：获取浏览器中传递数据

https://www.tmooc.cn:3000/web/1903.html?

sid=10&course=javascript

获取url 中的查询字符串

将查询字符串格式化为是对象

09\_exercise.js

6.fs模块 ——文件系统模块

用于文件的操作，目录的创建，删除，读取；文件的创建、读取、写入、删除..

（1）查看文件的状态

fs.stat(path,callback)

path 要查看的文件路径

callback 回调函数，用于获取文件的状态

err 如果查看失败的错误信息

stats 文件的状态信息

isDirectory()是否为目录

isFile() 是否为文件

课后任务

（1）复习今天内容，整理思维导图

（2）练习

创建模块01\_1.js,引入02\_2不带路径的目录，在02\_2中创建文件file.js,在file.js中，导出一个查询自符串，在01\_1.js中获取该字符，并获取其中的数据。

（3）预习文件系统模块的mkdir, rmdir,readdir,writeFile,appendFile,readFile

复习

1.模块分类

|  |  |  |
| --- | --- | --- |
|  | 以路径开头 | 不以路径开头 |
| 文件模块 | require('./circle.js')常用于用户自定义的模块，如果后缀名是js,则可以省略 | require('url')  用于引入官方提供核心模块 |
| 目录模块 | require('./02\_2')在02\_2目录下寻找  package.json;寻找main属性对应的文件，找不到会去引入index.js | require('04\_2')自动到node\_modules中寻找目录模块04\_2,如果找不到会到上一级目录下寻找，直到顶级（当前磁盘）目录，常用于第三方模块 |

练习：创建模块03\_1.js,引入当前目录下的 03\_2目录模块；在03\_2下创建test.js，导出一个函数fn(计算两个数字相加)，在03\_1.js调用

练习：在05目录下创建模块05\_1.js，引入不带路径模块05\_2,在05\_2下含有hello.js文件（打印一句话）

2.npm 和包

包（package）:是一个目录模块，里边包含多个文件，其中有一个文件命名为package.json ,是包说明文件，包含有包的信息

npm:包管理工具，安装nodejs的时候会附带安装，用于管理包，包含下载，上传，升级..

npm 官网：www.npmjs.com

（1）切换目录

cd 完整路径

如果要进入其它盘符

例如：d: 回车

进入指定的目录，按下shift键，在空白区域单击鼠标右键->在此处打开Powershell

（2）使用npm下载安装第三方包

npm install 包名称

（3）生成package.json

npm init -y 自动生成一个package.json文件，后期使用npm 安装的包都会记录到这个文件中

npm install 自动安装package.json 文件中记录的包

3.全局函数

parseInt/parseFoat/encodeURI/decodeURI/isNaN/isFinite/eval

（1）一次性定时器

|  |
| --- |
| var timer=setTime(回调函数，间隔的时间)；  当间隔的时间到了，执行回调函数，单位是毫秒  清除  clearTime(timer); |

（2）周期性定时器

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

（3）立即执行定时器

|  |
| --- |
| 开启  var timer=setImmediate(回调函数)  回调函数会放入到队列中  档主线程程序执行完，才会执行队列中的内容  清除  cleatImmediate(timer) |
| process.nextTick(回调函数)  在主线程的末尾执行 |

4.querystring 模块----查询字符

属于核心模块，nodejs 官方提供的，可以直接引入，不需要创建

www.codeboy.com/detail.html?lid=5&name=dell

浏览器向服务器发请求，传递数据的一种方式

parse()将查询字符串格式化为对象

stringify() 将对象转换成字符串

练习：获取百度查询中查询字符串中的关键字

e=utf-8&tn=baidu&wd=电脑

https://www.codeboy.com:80/product/detail.html?lid=5&name=dell

5.url模块

parse() 将url格式化为对象

protocal 协议

hostname 服务器域名/IP地址/

pathname 文件在服务器上的路径

query 查询字符串

forma()将对象转成url

query 属性对应的是对象

练习：获取浏览器中传递数据

https://www.tmooc.cn:3000/web/1903.html?

sid=10&course=javascript

获取url 中的查询字符串

将查询字符串格式化为是对象

09\_exercise.js

6.fs模块 ——文件系统模块

用于文件的操作，目录的创建，删除，读取；文件的创建、读取、写入、删除..

（1）查看文件的状态

fs.stat(path,callback)

path 要查看的文件路径

callback 回调函数，用于获取文件的状态

err 如果查看失败的错误信息

stats 文件的状态信息

isDirectory()是否为目录

isFile() 是否为文件

课后任务

（1）复习今天内容，整理思维导图

（2）练习

创建模块01\_1.js,引入02\_2不带路径的目录，在02\_2中创建文件file.js,在file.js中，导出一个查询自符串，在01\_1.js中获取该字符，并获取其中的数据。

（3）预习文件系统模块的mkdir, rmdir,readdir,writeFile,appendFile,readFile

# NODEJS-day03

复习

1.模块分类

|  |  |  |
| --- | --- | --- |
|  | 以路径开头 | 不以路径开头 |
| 文件模块 | require('./circle.js')常用于用户自定义的模块，如果后缀名是js,则可以省略 | require('url')  用于引入官方提供核心模块 |
| 目录模块 | require('./02\_2')在02\_2目录下寻找  package.json;寻找main属性对应的文件，找不到会去引入index.js | require('04\_2')自动到node\_modules中寻找目录模块04\_2,如果找不到会到上一级目录下寻找，直到顶级（当前磁盘）目录，常用于第三方模块 |

练习：创建模块03\_1.js,引入当前目录下的 03\_2目录模块；在03\_2下创建test.js，导出一个函数fn(计算两个数字相加)，在03\_1.js调用

练习：在05目录下创建模块05\_1.js，引入不带路径模块05\_2,在05\_2下含有hello.js文件（打印一句话）

2.npm 和包

包（package）:是一个目录模块，里边包含多个文件，其中有一个文件命名为package.json ,是包说明文件，包含有包的信息

npm:包管理工具，安装nodejs的时候会附带安装，用于管理包，包含下载，上传，升级..

npm 官网：www.npmjs.com

（1）切换目录

cd 完整路径

如果要进入其它盘符

例如：d: 回车

进入指定的目录，按下shift键，在空白区域单击鼠标右键->在此处打开Powershell

（2）使用npm下载安装第三方包

npm install 包名称

（3）生成package.json

npm init -y 自动生成一个package.json文件，后期使用npm 安装的包都会记录到这个文件中

npm install 自动安装package.json 文件中记录的包

3.全局函数

parseInt/parseFoat/encodeURI/decodeURI/isNaN/isFinite/eval

（1）一次性定时器

|  |
| --- |
| var timer=setTime(回调函数，间隔的时间)；  当间隔的时间到了，执行回调函数，单位是毫秒  清除  clearTime(timer); |

（2）周期性定时器

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

（3）立即执行定时器

|  |
| --- |
| 开启  var timer=setImmediate(回调函数)  回调函数会放入到队列中  档主线程程序执行完，才会执行队列中的内容  清除  cleatImmediate(timer) |
| process.nextTick(回调函数)  在主线程的末尾执行 |

4.querystring 模块----查询字符

属于核心模块，nodejs 官方提供的，可以直接引入，不需要创建

www.codeboy.com/detail.html?lid=5&name=dell

浏览器向服务器发请求，传递数据的一种方式

parse()将查询字符串格式化为对象

stringify() 将对象转换成字符串

练习：获取百度查询中查询字符串中的关键字

e=utf-8&tn=baidu&wd=电脑

https://www.codeboy.com:80/product/detail.html?lid=5&name=dell

5.url模块

parse() 将url格式化为对象

protocal 协议

hostname 服务器域名/IP地址/

pathname 文件在服务器上的路径

query 查询字符串

forma()将对象转成url

query 属性对应的是对象

练习：获取浏览器中传递数据

https://www.tmooc.cn:3000/web/1903.html?

sid=10&course=javascript

获取url 中的查询字符串

将查询字符串格式化为是对象

09\_exercise.js

6.fs模块 ——文件系统模块

用于文件的操作，目录的创建，删除，读取；文件的创建、读取、写入、删除..

（1）查看文件的状态

fs.stat(path,callback)

path 要查看的文件路径

callback 回调函数，用于获取文件的状态

err 如果查看失败的错误信息

stats 文件的状态信息

isDirectory()是否为目录

isFile() 是否为文件

课后任务

（1）复习今天内容，整理思维导图

（2）练习

创建模块01\_1.js,引入02\_2不带路径的目录，在02\_2中创建文件file.js,在file.js中，导出一个查询自符串，在01\_1.js中获取该字符，并获取其中的数据。

（3）预习文件系统模块的mkdir, rmdir,readdir,writeFile,appendFile,readFile

# NODEJS-day04

复习

fs模块

fs.stat/statSync/mkdir/rmdir/readdir/writeFile

appendFile/readFile/unlink/existsSync

http协议

通用头信息 请求方法 请求URL 响应状态码

响应头信息

请求头信息

请求主体

http模块

模拟浏览器向服务器发请求

http.get(url,function(res){

res.statusCode

res.on('data',function(buf){ })

} )

创建web服务器

var server=http.createServer();

server.listen(8080);

server.on('request', function(req,res){

req.method/url

res.writeHead(200,{ })

res.write()

res.end()

})

1.express框架

基于nodejs的，快速、开放、极简的web开发框架

www.expressjs.com.cn

安装 nmp install express

|  |
| --- |
| const express=reqire('express');//引入express  var server=express();//创建web服务器  server.listen(8080);//设置端口 |

（1）路由

浏览器向web服务器发送请求，web服务器根据请求的方法和请求的URL来做出响应

三要素：请求方法、请求的URL、响应（回调函数）

响应的对象（res）

send() 响应内容，只能响应一次

sendFile（）响应文件，必须使用决定对路径(杠杠\_\_dirname)

redirect()响应的重定向

请求的对象(req)

req.method 请求的方法

req.url 请求的URL

req.headers 请求的头信息

req.query 获取请求时查询字符串传递的数据，并格式化对象

练习：创建文件03\_express.js,使用express 创建web服务器，添加以下路由

get /index '这是首页'

get /login 响应文件login.html

post /reg '欢迎注册'

get / 跳转到 /index

练习：创建文件04\_post.js,使用express创建web服务器，添加路由

get /reg 响应文件 reg.html

2.post和get传递数据

post请求是通过表单提交（暂时）传递数据，服务器端是通过事件的形式获取数据（后期会有简单的方法）

get请求以查询字符串形式传递数据，服务器端使用req.query获取数据，结果是对象。

查询字符串传递数据容易被浏览器所缓存，而post传递数据不会出现在地址栏

|  |
| --- |
| req.on('data',functon(buf){ buf 就是获取的数据，格式为buffer，转为字符串后格式为查询字符串，需要借助查询字符串模式格转化为对象 |

3.使用路由传递数据——路由传参

设置路由中接收的名称

|  |
| --- |
| server.get('/detail/:lid',function(req,res){  req.params //获取路由传递的数据，格式为对象  }); |

浏览器传递

http://127.0.0.1:8080/detail/5

说明：5就是传递的数据，使用lid来接收

练习：创建购物车的路由，请求的方法get,请求的URL:/shoppin,传递商品的名称（pname）和价格（price）

商品模块

商品列表/product/list 删除/product/delete 修改/update

用户模块

用户列表/user/list 删除/user/delete 修改/update

4.路由器

路由的使用过程中，不同的模块可能出现的相同的URL，把同一个模块下的所有路由放到一个容器，这个容器就是路由器，路由器最终要引入到web服务器下才能使用

创建路由器模块（自定义模块）

|  |
| --- |
| const express=require('express');  var router=express.Router();//创建路由器对象  router.get('/list',function(req,res){ });//添加路由  module.exports=router;//导出路由器对象 |

在web服务器下使用路由器

|  |
| --- |
| const productRouter=require('');  server.use('/product', productRouter);  //把路由器挂载到/product下，  访问形式 /product/list |

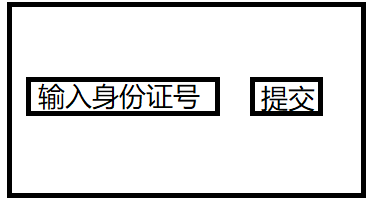
练习：创建用户模块路由器（user.js）,添加路由由用户列表（/list,get）、在web服务器下引入并使用，挂载到/user下

课后任务

(1)复习今天内容，整理思维导图

(2)课后练习

创建web服务器，添加路由(get /page)，响应一个html文件；输入身份证号，点击提交(post, /mypage)，响应出生的年月日以及性别。



(3)复习mysql中sql语句

# NODEJS-day05

复习

1.中间件

中间件理解为是一个过滤器，作用是为主要的业务逻辑服务

分为应用级中间件、路由级中间件、内置中间件、第三方中间件、错误中间件

（1）应用级（自定义）中间件

每个中间件都是一个函数

server.use(path,function(req,res,next){ })过滤有中url为path的路由。

server.use(function(req,res,next){ })过滤所有的路由

练习：给url为/reg的路由添加前置的中间件，验证用户名是否可用

练习：创建路由（get,/view）响应当前的浏览次数；在中间件的外部声明变量初始为0，每次浏览次数加1.在中间件中完成加1，在路由中把该变量响应到浏览器。

（2）路由级中间件

路由器的使用

servar.use(path,路由器);

（3）内置中间件

在express4下只保留了一个内置的中间件

express.static('目录')

把静态资源托管到指定的目录，如果浏览器请求静态资源，自动到该目录下查找。

静态资源：html、css、浏览器js、图像....

（4）第三方中间件

npm install 中间件

使用中间件

|  |
| --- |
| server.use(bodyParser.urlencode({  extended:false  }))  urlencoded:将post请求的数据格式化为对象  extended:不是用第三方qs模块，而是使用核心模块  querystring 将查询的字符串格式化为对象 |

在路由中获取post请求数据

req.body 格式为对象

练习：创建04\_post.js,创建web服务器，托管静态资源到public下，在public下，创建login.html,使用浏览器请求该文件，点击按钮，向服务器发请求。

2.mysql

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

mysql -uroot

增 INSERT INTO emp VALUES(NULL,'tom'...)

删 DELETE FROM emp WHERE uid=5;

改 UPDATE emp SET uname='tom',salary=8000 WHERE uid=5;

查 SELECT \*FROM emp;

（1）连接 mysql

var connection=mysql.createconnection();创建连接对象、传递连接数据库需要的服务器、端口、用户名、密码、要使用的数据库

connection.connect() 建立连接

connection.query(sql,callback) sql 是要执行的SQL 语句，callback回调函数，用于获取SQL语句的结果connection.end()

（2）连接池

var pool=mysql.createPool();创建连接池对象，传递连接需要的服务器地址、端口、密码、要使用数据库、设置连接池的大小connectionLimit,默认是15

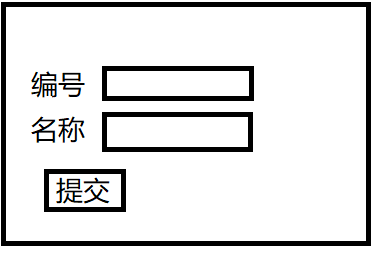
pool.qurey(sql,callback);执行SQL语句，sql要执行

课后任务

(1)复习今天内容，整理思维导图

(2)练习

创建web服务器，托管静态资源到public下，在该目录下创建add.html，点击提交按钮，向服务器发请求(get /add)，服务器获取数据，并将数据插入到tedu下的dept，如果插入成功，响应“部门添加成功”



# NODEJS-day06

复习

分页查询

页码

每页数据量 2

start=(页码-1)\*数据量

select \* from xz\_user limit 0,2;

select \* from xz\_user limit 2,2;

课后任务

完成商品模块 xz\_laptop

添加、修改、删除、检索、列表

创建商品路由器 product.js

# HTML-day01

day01

李然 QQ1535212067

一.课程安排

1.Html5 Basic 2天

|  |
| --- |
| 搭建网页结构 |

2.Ajax异步数据交换(3天)

|  |
| --- |
| 异步完成前后台数据的交互 |

3.项目1（2天）

|  |
| --- |
| MySQL nodejs html ajax  从前端页面实现对用户表中数据的增删改查 |

4.css3 （4+2天）

|  |
| --- |
| 美化页面 |

5.Bootstrap框架 （4天）

|  |
| --- |
| 响应式的框架，css的框架  简化开发 |

6.项目2 （3天）

|  |
| --- |
| boot完成商城的商品部分所有页面 |

7.这20天课程遇到的问题

|  |
| --- |
| 1.知识点细碎，没条理  2.单词量大，关键词比较长，初期没有提示  3.键速 200 |

二.web基础知识

|  |  |
| --- | --- |
| HTML:泛指前端网页技术  2014年9，html4.0升级html5.，简称h5  Html5 大前端技术   |  | | --- | | html4.01在1999.12月发布  xhtml1.0 在2000.1月发布，语法更严谨  <img/> <img> | |

1.web与internet的关系

|  |
| --- |
| internet 全球性计算机互联网络  俗称 互联网，因特网，交换网，交际网  www服务world wide web 万维网  ftp:文件的上传下载的服务  Email:电子邮件的服务  telnet：远程登陆的服务  BBS：电子公告板，俗称论坛  以上都是运行在internet应用程序 |

2. internet上的应用程序都会分成两类

|  |
| --- |
| 1. C/S client 客户端 server服务器  王者荣耀 QQ.exe |
| 2. B/S browser浏览器 server服务器  所有的网站 网页版微信 |
| 3.C/S结构和B/S结构的区别  C/S需要升级 B/S不需要升级 |

3.web运行原理

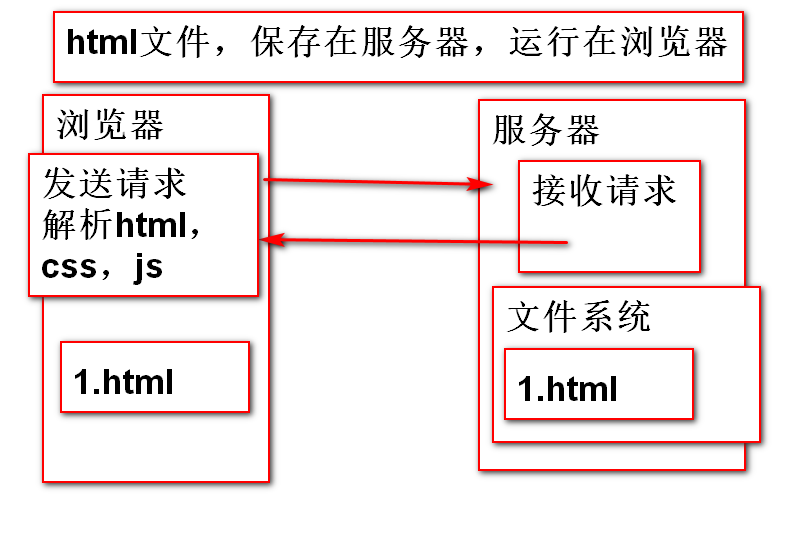
|  |
| --- |
| web：运行在internet上的一种B/S结构的应用程序  俗称网站  internet:为web运行提供了网络环境  web运行的原理：基于浏览器和服务器，以及通信协议来实现数据的传输和展示  通信协议：规范了数据是如何打包和传递的  网页的通信协议 http:// https:// |
| 服务器  1.功能  存储数据  接收用户的请求，并且给出响应  提供程序的运行环境  具备一定的安全功能  2.服务器产品  Apache--php  TOMCAT--java  IIS .net  3.服务器端技术  nodejs  java  php  .net  python |
| 3.浏览器  1.功能  代表用户发送请求  作为html，css和js的解析器  以图形化的界面展示给用户看  2.浏览器产品  safari  chrome  firefox  opera  ie  3.浏览器端的技术  html5 css3 javascript 11:04~~11:19 |

三.html快速入门

1.什么是html

|  |
| --- |
| HyperTexts Markup Language  超文本标记语言  标记：超文本组成形式，<关键字>，具体独特的功能 |

2.html的特点



|  |
| --- |
| 1.以.html或者.htm为后缀  2.由浏览器解析执行  3.用<>来定义标记  4.可以嵌套js脚本 |

3.html语法

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1.标记   |  |  |  | | --- | --- | --- | | 标记，元素，标签，节点  <关键字>  标签具备一些功能  学习html第一步，背关键字，熟悉对应的功能  标记分为两类   |  | | --- | | 1.双标记（封闭类型标记）  <关键字></关键字>  双标记成对出现，有开有关  ex:<a></a> | | 2.单标记(非封闭类型标记，空标记)  <关键字> 或者 <关键字/>  ex：<input/> <br/> <img/> |   练习：  新建一个text文档，改名为first.html  写一对html标记  一对head标记  一对title标记  一对body标记 |   2.标签的嵌套   |  | | --- | | 在一对标签内部，出现另一对或者另一个标签，组合形成功能的叠加  学习html第二步，背嵌套顺序  语法：缩进，嵌套顺序  <标签1>  <标签2>  <标签3></标签3>  </标签2>  </标签1> |   3.属性和值   |  | | --- | | 通过属性和值，对标签进行修饰  <标签 属性1="值1" 属性2="值2" ...></标签> | | 1.标准属性，通用属性。所有标签都有的属性  id:定义元素在页面中唯一的标识  title：鼠标悬停在元素生显示的文本  style：css中，定义内联样式  class：css中，引用类选择器时使用 | | 2.专有属性，只针对一个标签起作用的属性 |   学习html第三步，背专有属性，及效果  4.注释   |  | | --- | | 不会被浏览器解析运行的部分  一般编写代码说明  <!-- -->  注释不能嵌套注释  注释不能写在标签内部 | |
| 学习html就三步  1.学习固定标签关键字，及功能  2.学习固定的嵌套关系。  3.学习固定的属性及其值 |

四.HTML的文档结构

1.html文档的构成

|  |
| --- |
| 1.文档类型声明  <!doctype html>  告诉浏览器，解析运行本篇代码，使用h5的规则解析 |
| 2.网页的结构  <html></html>  表示网页的开始和结束。一个html文件中只能有一对html标签  <!doctype html>  <html>  <head></head> 网页头部，定义网页的信息  <body></body> 网页主体，展示内容  </html> |
| 3.head标签，定义网页的信息   |  | | --- | | 1.<meta> 元数据标签  <meta charset="utf-8">  <meta name="description" content="描述内容">  <meta name="Keywords" content="关键字"> | | 2.head中其它标签  <title></title> 网页标题  <script></script> 编写js或者引用js文件  <style></style> css中定义内部样式  <link>引入外部css样式 |   4.body元素   |  | | --- | | 指定网页的主体  <body></body>  属性：bgcolor body的背景颜色，取值颜色合法值  text body的文本颜色，取值颜色的合法值 |   练习：01\_ex.html  编写完整的html页面结构 |

五.文本标记(重点\*\*\*)

1.标题元素

|  |
| --- |
| 在页面中以醒目的方式显示文本  <h1></h1>  ....  <h6></h6>  特点：1.字体大小有变化 h1最大，h6最小  2.字体加粗  3.单独成行，上下有垂直间距 属性：align 设置标记内容水平对齐方式  取值：left/center/right |

2.段落标记

|  |
| --- |
| <p></p> 以突出的形式表示一段文字  特点：1单独成行  2.上下有垂直间距  属性：align:left/cener/right |

3.换行标记

|  |
| --- |
| 空格折叠现象：在html中，文本中不管有多少个空格和回车，都会被浏览器解析成一个空格显示  <br> 或者 <br/> |

4.特殊字符(实体)

|  |
| --- |
| &nbsp; 空格  &lt; <  &gt; >  &times； X  &reg; 商标注册 ®  &copy；版权 ©  &yen; |

5.分割线，水平线

|  |
| --- |
| <hr> 或者 <hr/>  属性  size="5px" 取值px为单位的数字，水平线的粗细  width="50%" 取值px或者%,水平线的宽度  align="left" left/center/right  color="blue" 取值为合法颜色值 |

6.预格式化

|  |
| --- |
| <pre></pre>  保留html代码中的回车和空格效果 |

7.文本样式标签

|  |
| --- |
| <em></em> <i></i> 斜体  <strong></strong> <b></b> 加粗  <del></del> <s></s> 删除线  <u></u> 下划线  <sup></sup> 上标  <sub></sub> 下标 |

8.分区元素--不写样式的分区元素，是看不见的。

|  |
| --- |
| 1.块分区<div></div>  用于页面布局 |
| 2.行分区<span></span>  为文本添加样式的时候用span |

9.元素的显示方式

|  |
| --- |
| 1.块级元素  元素独立成行，在页面中从上往下排列  ex:h1~h6 p div pre |
| 2.行内元素(行级元素)  多个行内元素在一行中显示，从左往右排列  ex:span i em strong b u sub sup del s |
| 3.行内块  表现方式是行内元素(多个行内块可以共用一行)  但是具有块级元素的其它特征(宽高，边距)还没学  ex:input |
| 4.table |

六.图形和链接

1.图形图像

|  |
| --- |
| <img/> 或者 <img>  属性：src 引用图片资源 url  \*.html和图片是兄弟关系，直接在src中写图片名称 |

练习



EditPlus快捷键设置总结

|  |
| --- |
| 加载1.key。1.key eclipse的开发快捷键  1.设置代码补全和lorem的补全  在zc下，把第二个和第三个勾勾上  2.设置tab空两个格    把默认的4变成2  3.导入1.key    选择1.key就可以使用了  4.常用快捷键  ctrl+d 删除当前行  alt+↓/↑ 移动当前行下移/上移  ctrl+alt+↓ 复制当前行  5.打开浏览器运行html的快捷方式  ctrl+b 打开浏览器    6.添加注释快捷方式  ctrl+/ |

作业:

1.完成nodejs6个接口的编写

2. web第二阶段练习效果图示\day01\_html\_01作业



# HTML-day02

day02

六.图像和链接

1.图像的使用

|  |
| --- |
| <img>  属性  src="url/图片资源路径"  ex：<img src="图片资源路径"> |

2.URL

|  |
| --- |
| 1.URL是什么  Uniform Resource Locator 统一资源定位器  就是路径 |
| 2.绝对路径  完整的路径  协议+主机名称+文件目录结构+文件具体的名称  使用场合：使用网络资源的时候，使用绝对路径  使用网络资源图片  优点：节省服务器本地存储空间  缺点：资源不稳定  绝对路径使用本地的资源，是从最高盘符开始查找  c:\img\08.png  但是本服务器资源，在项目中，不能用绝对路径 |
| 3.相对路径    使用本服务器资源，用相对路径  1.同级目录资源  直接使用资源名称 src="08.png"  2.兄弟文件夹的子元素  直接使用兄弟文件夹的名称，在用/调用资源名称  src="img/08.png"  3.父级目录中资源  先使用../返回上一级，在引用资源名称  src="../08.png" |

3.属性

|  |
| --- |
| src：source 源，资源路径  alt： 图片错误时，显示的文本  title：鼠标悬停时显示的文本  width：设置图片的宽度  height：设置图片的高度  如果设置宽高比，不符合图片原始宽高比。  图片会产生失真效果  解决图片失真效果：  一般情况下，width和height只设置一个，让另外一个自动适应 |

练习

|  |
| --- |
| 01\_ex.html  页面中引入6张图片  1.图片与页面是兄弟关系  2.图片是页面的兄弟的儿子  3.图片是页面的兄弟的儿子的儿子  4.图片是页面父亲的兄弟  5.图片是页面的父亲的兄弟的儿子的儿子  6.图片是页面父亲的父亲的兄弟 |

4.链接

|  |
| --- |
| 1.语法  <a href=""></a>  属性：href 链接路径  target 指定打开链接的方式  取值 \_self 默认值，在当前页面打开新网页  \_blank 在新的页面打开网页 |
| 2.a标签的其它表现形式  1.新建邮件，配合windows的邮件软件使用  <a href="mailto:cheng@t.com">发邮件</a>  2.执行js  <a href="javascript:show()">执行js</a>  3.下载  <a href="1.zip">下载</a>  4.返回页面顶部  <a href="#">返回页面顶部</a> |
| 3.锚点  什么是锚点  锚点是网页中的一个记号  可以通过超链接的方式，链接到锚点，让页面跳转到锚点位置显示  使用锚点   |  | | --- | | 1.定义锚点  第一种方式：任何标签的id中写锚点名称  <Any id="锚点名称"> </ Any >  第二种方式：a标签的name属性中写锚点名称  <a name="锚点名称"></a> | | 2.链接到锚点  <a href="#锚点名称"></a> | | 3.链接到其它页面的锚点  <a href="其它页面的url#锚点名称"></a>  <a href="02\_a.html#hl">葫芦娃</a> | |

七、表格

|  |
| --- |
| 早期table用于布局  由于效率低下，后来被淘汰。使用div+css的布局  表格现在就一个单纯的作用——显示数据  <table>表格  <tr> 行 row  <td></td> 列 data  .........  </tr>  </table> |
| 属性   |  | | --- | | table的属性  width="300px" 表格的宽度  height="300px" 表格的高度  border="1px" 表格的边框  align="center" 表格的水平对齐方式  bordercolor="red"表格边框的颜色  bgcolor="green"表格的背景颜色  cellpadding 单元格内边距，内容到边框的距离  cellspacing单元格外边距，边框到边框的距离（单元格之间的距离） | | tr的属性  bgcolor="green"背景颜色  align="center" 表格这一行的内容水平对齐方式  valign="top" 表格这一行的内容垂直对齐方式  取值：top上/middle中间/bottom底部 | | td的属性  列设置宽高  宽高 > 内容，就显示设置宽高  宽高 < 内容，按内容显示  width="200px"  height="200px"  align="right" 内容水平对齐方式  valign="bottom" 内容垂直对齐方式  bgcolor="red"背景颜色  colspan 跨列  rowspan 跨行 | |
| 不规则的表格   |  | | --- | | 1.跨列：colspan  从指定单元格位置处开始，横向向右合并n个单元格（n包括自己），被合并的单元格要删除 | | 2.跨行  从指定单元格位置处开始，纵向向下合并n个单元格 |   3.可选标记   |  | | --- | | 1.表格的标题  <caption></caption>  如果要设置标题，caption必须放在<table>之后  2.行/列的标题  <th></th>,td的属性，th都可以用  th的特点：文字加粗，水平居中 |   4.表格的复杂应用   |  | | --- | | 我们制作的表格，浏览器解析的时候会自动添加  <tbody></tbody>标签  行分组：可以将连续的几行，划分到同一个组中，进行统一的管理。  1.表头<thead></thead>  2.表主体<tbody></tbody>  3.表脚<tfoot></tfoot> | | 2.表格的嵌套  表格中所有的嵌套，都只能放在td中完成 | |

八、列表

1.列表的作用

|  |
| --- |
| 按照从上到下，或者从左到右来显示所有的数据并且可以在数据前增加标识 |

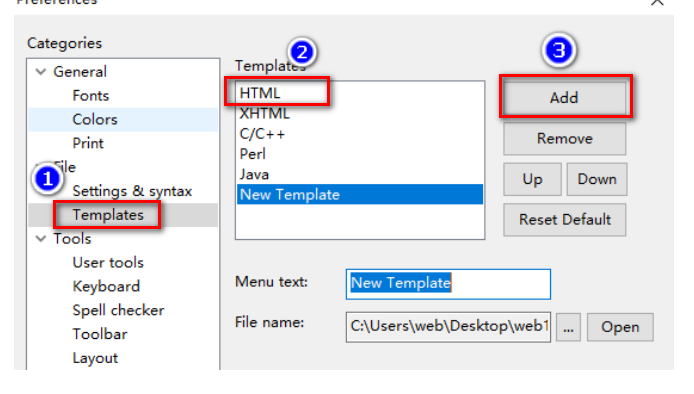
2.列表的组成

|  |
| --- |
| 列表都是有列表类型和列表项来组成的  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  <li></li> list item |

3.有序列表

|  |
| --- |
| <ol>  <li></li >  </ol>  属性：type |

设置html模板



# HTML-day03

day03

八.列表

1.有序

2.无序

3.列表的嵌套

|  |
| --- |
| 1.列表中嵌套其他元素 |
| 2.列表中嵌套列表 |
| 原则，嵌套只能写在li中。不管是ol还是ul，直接子元素只能是li. |

4.定义列表

|  |
| --- |
| h5新出的，一个名词，以及对这个名词的解释和定义  <dl> 定义列表  <dt> </dt> 要解释说明的名词  <dd></dd> 对名词解释的内容  </dl> |

九.结构标签

|  |
| --- |
| h5新出的标签，专门用于网页布局  为了让代码有可读性，h5推出一堆与div功能一模一样的标签，专门用作网页布局  1.<header></header>  定义网页的头部，或者某个区域的顶部  2.<footer></footer>  定义网页的脚部，或者某个区域的底部  3.<nav></nav>  定义网页的导航栏  4.<section></section>  定义网页的主体内容  5.<aside></aside>  定义网页的侧边栏  6.<article></article>  定义与文字相关的内容  比如，论坛，回帖，用户评论.... |

十.表单，\*\*\*\*重点\*\*\*\*\*

1.作用

|  |
| --- |
| 提供可视化的输入控件  收集用户输入的信息，并提交请求给服务器  总结：form自带提交请求的功能  ajax提交不需要form的支持  在同一个功能中，使用form就不用ajax,使用ajax就不用form |

2.form的组成

|  |
| --- |
| 前端部分  提供表单控件，与用户交互的可视化控件  后端部分  后台接口对提交的数据进行处理 nodejs+mysql |

3.form表单详解

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| <form></form>  属性：   |  | | --- | | 1.action="" 定义表单提交时，发送请求的动作(url)  如果不写值，默认提交给本页 | | 2.method="" 定义表单的提交方法  取值   |  | | --- | | 1.get 默认值  特点：明文提交(参数在地址栏中显示)，不安全  提交的数据有大小限制，最大为2kb  使用场合：向服务器要数据的时候，使用get | | 2.post  特点：隐式提交，参数不在地址栏中显示  提交数据没有大小限制  使用场合：要传递数据给服务器的时候，使用post | | 3.其它的提交方法  delete put option....... | | | 3.enctype="" 指定表单数据的编码方式  设置允许将什么类型的属性传递给服务器  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  (不能提交文件)  2.text/plain 允许提交普通字符给服务器，不能包含特殊符号  3.multipart/form-data 允许提交文件给服务器 |   <form action="url" method="get"  enctype="application/x-www-form-urlencoded">  </form>  收集的数据，发送给url这个接口，使用get提交  要求提交的数据是任意字符，不是文件 |

4.表单控件

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 在form标签中，能够与用户交互的可视化控件  1.表单控件的分类   |  | | --- | | input元素 基础9种  textarea 多行文本域  select+option下拉选择器  其它元素  h5新出的input元素 10种 |   2.input元素   |  | | --- | | <input type="">  在页面中，input提供了各式各样的输入控件  文本框，密码框，单选框，多选等....  共同的属性  1.type="" 指定input元素的类型  2.name="" 为控件定义名称，此名称是给服务器用的，如果想往服务器传递参数，此名称必须写  3.value 是控件的值，真正传给服务器的值  例外,在按钮控件中,value是设置按钮上显示的文字  4.disabled 禁用，不能修改值，也不能提交  取值：disabled="disabled"  disabled="true" --js使用  disabled  这种属性，称之为无值属性 |   3.input详解9种   |  | | --- | | 1.文本框和密码框  文本框 type="text" input默认是text  密码框 type="password"  属性：  maxlength 设置输入的最大值  readonly 无值属性，不能修改，允许提交  placeholder 占位提示符 | | 2.按钮  提交按钮 type="submit"  将表单中的数据收集整理，发送给服务器  普通按钮 type="button"  没有功能，配合事件，调用js代码  重置按钮 type="reset"  将当前表单内容恢复到初始化状态  按钮中value，设置按钮上的文本，此value不提交 | | 关于按钮的题外话  h5给我们推出了新的按钮标签  <button></button> 自带submit功能 | | 3.单选和复选按钮  单选按钮 type="radio"  多选按钮 type="checkbox"  属性name，除了定义控件名称之外，还起到分组的作用，分了组才会有单选效果  必须有value属性，不然传递的参数永远是on  checked 无值属性 设置默认被选中的项 | | 4.隐藏域  type="hidden"  想把数据提交给服务器，但是又不想用户看到  这种数据放到隐藏域中  用户看不见，但是可以提交 | | 5.文件选择框，上传文件  type="file"  前提：提交方法，使用post  enctype属性，设置为multipart/form-data  属性multiple，可以选择多个文件上传 |   4 textarea多行文本域   |  | | --- | | <textarea></textarea>  允许录入多行文字  cols 设置输入的列数  rows 设置输入的行数  根据计算硬件不同，不准确 |   5.下拉选   |  | | --- | | <select>  <option></option>  <option></option>  ....  </select>  注意：  1.如果option没有value，那么select的value是option的内容  2.如果option有value,那么select的value是选中option的value值  属性 name value  selected 默认选中  size 默认值为1，  值为1的时候，为下拉选择框  值>1的时候，为滚动选择框  multiple 无值属性，多选 |   6.其它元素   |  | | --- | | 1.label 关联文本与表单的控件  <label for=""></label>  属性for绑定要关联的表单控件的ID |   7.为控件分组   |  | | --- | | <fieldset> 分组  <legend>标题</legend>  form表单控件......  </fieldset> | |

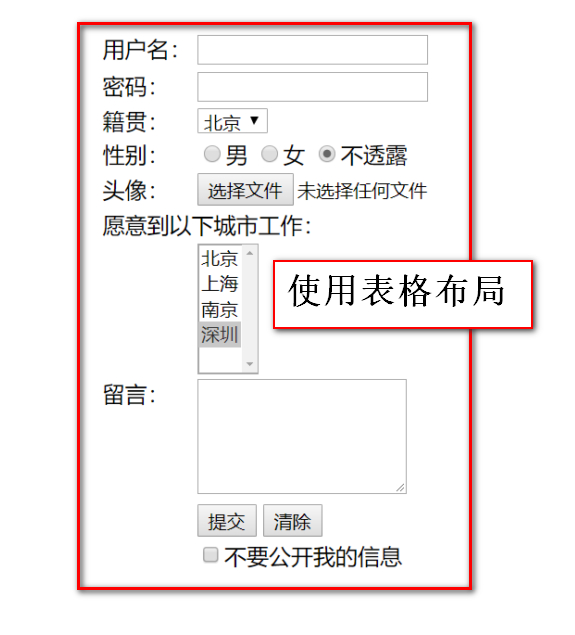
十一.浮动框架

|  |
| --- |
| <iframe style="width:100%" src="01\_ex.html"  frameborder="0" scrolling="no"></iframe>  语法 <iframe></iframe>  属性 src 引用的资源路径  frameborder="0" 清除边框  scrolling="no" 清除滚动条  width/height |

|  |
| --- |
| colspan 跨列，删除当前tr中td  rowspan 跨行，删除其他tr中td |

作业：

1.表单练习



2.完成下图布局



3.nodejs 6个接口，配合今天学习form。

# AJAX-day01

day04

一、for表单

1.h5新表单元素

①邮箱，提交的时候，验证@以及@前后至少有一个字符

|  |
| --- |
| <input type="email"> |

②搜索类型，自带快速清除功能

|  |
| --- |
| <input type="search" > |

③url类型，提交时验证http://开头

|  |
| --- |
| <input type="url" > |

④电话号码，移动端使用时会弹出虚拟键盘

|  |
| --- |
| <input type="tel" > |

⑤数字类型，

|  |
| --- |
| <input type="number" name="my\_number" max="10" min="2" step="2">  max:能接收的最大值  min：能接收的最小值  step:步长，按一次按钮，变化几个数字 |

⑥范围类型

|  |
| --- |
| <input type="range" name="my\_range" max="10" min="1" step="2">  max:能接收的最大值  min：能接收的最小值  step:步长，滑动一次按钮，变化几个数字 |

⑦颜色类型

|  |
| --- |
| <input type="color" name="my\_color"> |

⑧日期类型

|  |
| --- |
| <input type="date" name="my\_date"> |

⑨月份类型

|  |
| --- |
| <input type="month" name="my\_month"> |

⑩周类型

|  |
| --- |
| <input type="week" name="my\_week"> |

超纲

|  |
| --- |
| <button><button>  替代<input type="submit" >增加可读性 |

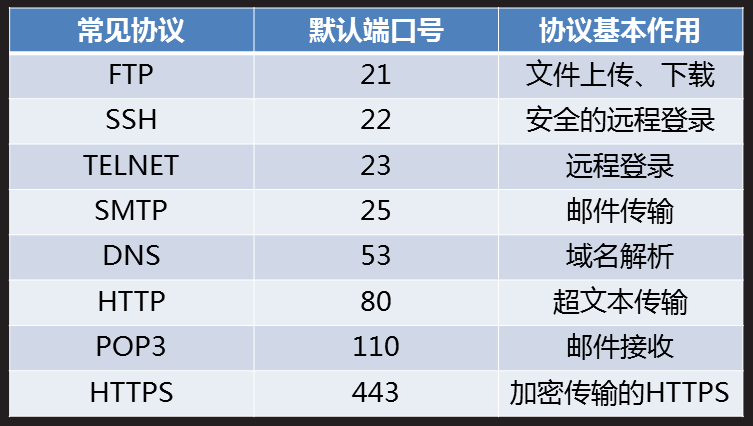
二、http协议

1.URL

|  |
| --- |
| 结构：协议+主机名称（127.0.0.1/localhost）+目录结构+文件名称  URL完整的结构  <scheme>://<user>:<pwd>@<host>:<port>/<path>;<params>?<query>#<frag> |

①scheme ：方案，协议。设置以哪种方式获取服务器资源

协议是不区分大小写的，常用的协议http://ftp:...



②<user>:<pwd> 用户名：密码 此种写法已过时

③<host> 主机名（域名、ip地址）

④<port> 端口号

⑤<path>路径，资源在服务器上具体存放的位置

⑥<params> 参数，保存跟踪状态的参数（session/cookie）

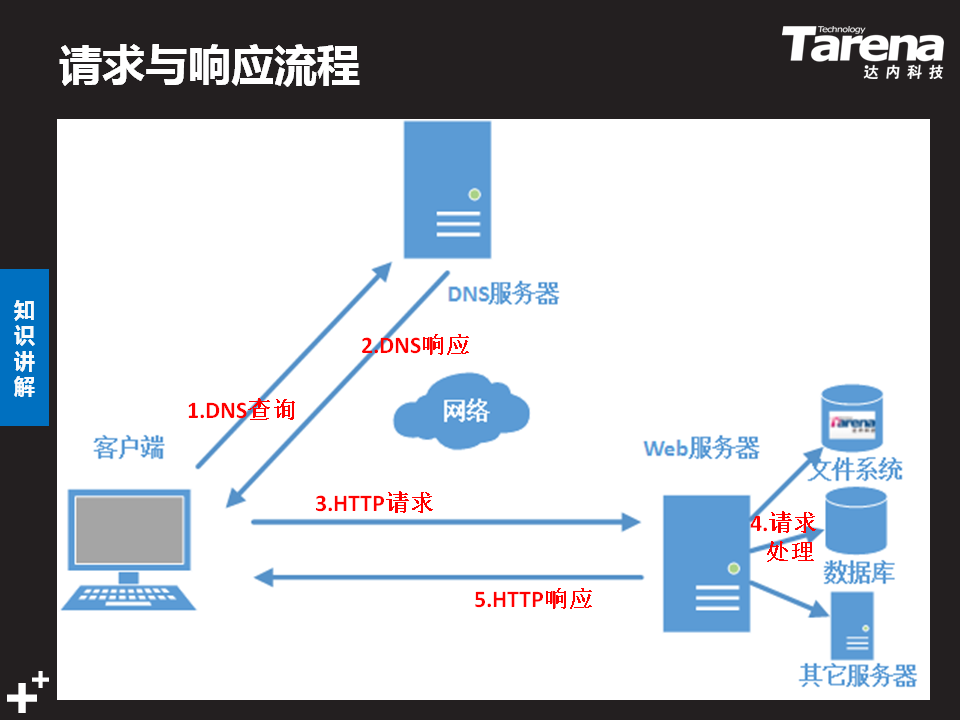
⑦<query> get方法提交请求时的查询字符串

⑧#<frag>锚点

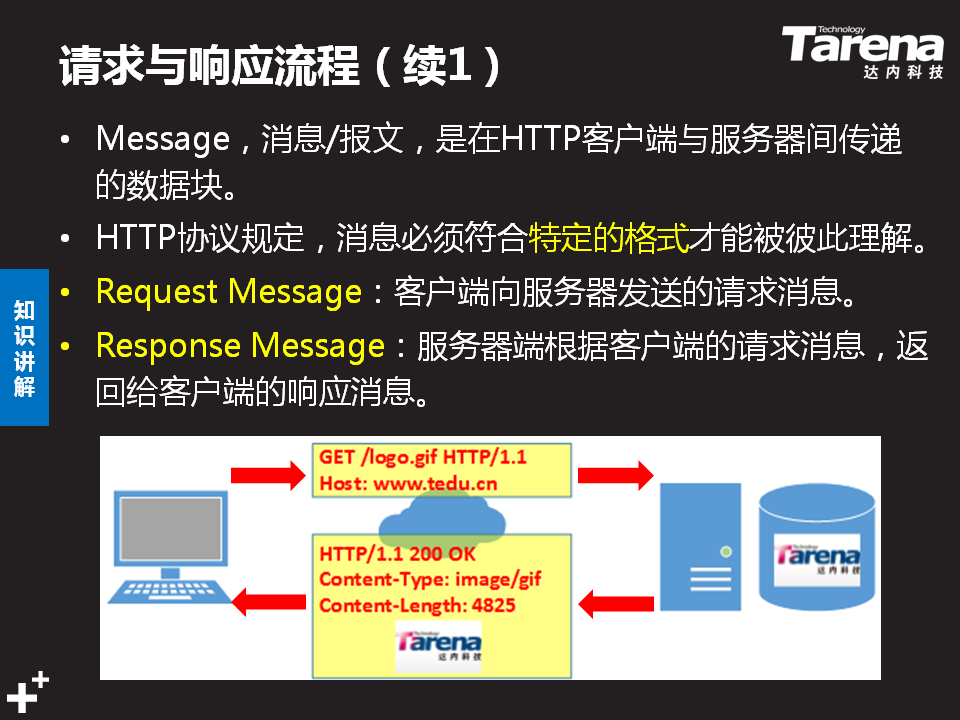
2.http协议

|  |
| --- |
| HTTP:HyperText Transfer Protocol 超文本传输协议  规范了数据是如何打包以及传递的  （专门用于传输html文件的协议，专门浏览网页的协议）  HTTP协议的历史  D:\GS\05_HTML\day01\web第二阶段练习效果图示\HTTP协议历史.png |

3.web请求原理详解



4.消息/报文Message



①请求消息 Request Message（请求起始行，请求头，请求主体）

②响应消息 ResponseMessage（响应起始行，响应头，响应主体）

5.Request Message

请求消息，客户端发送给服务器的数据块

由三部分组成，请求起始行，请求头，请求主体

①请求其始行

|  |
| --- |
| 1.请求方法  get:明文提交，不安全，有大小限值2kb  客户端向服务器要数据的时候使用get靠地址栏传递查询字符串的方式传参数  无请求主体  post:隐式提交，无大小限制  客户端要传递数据给服务器时使用。  delete:客户端想要删除服务器上的内容（一般禁用）  put:客户端想往服务器上放文件内容（一般禁用）  connect：测试连接  trace：追综请求路径  option:预请求  head：表示客户端只获取响应消息的头信息  有请求主体，  2.协议版本号  3.请求的url |

②请求头

|  |
| --- |
| 1.host:www.tmooc.cn  告诉服务器请求的是哪一个主机  2.Connection:keep-alive  告诉服务器开启持久连接  3.User-Agent：告诉服务器，浏览器的类型和浏览器的版本号  4.Accept-Encoding:gzip  告诉服务器，我这个浏览器可以接收的压缩文件的格式  5. **Accept-Language:** zh-CN  告诉服务器，我这个浏览器可以接收的自然语言的类型  6. **Referer:** http://www.tmooc.cn/  告诉服务器，当前请求来自于哪个网页 |

③请求主体

|  |
| --- |
| form data  get 请求没有请求主体  post有请求主体 |

6. ResponseMessage

响应消息，服务器发送给客户端的数据块

由三部分组成：响应起始行，响应头，响应主体

①响应起始行

|  |
| --- |
| 1.协议版本号 http/1.1  2.响应状态码  1XX正在请求，提示信息  2XX 200 响应成功  3XX 重定向 301 永久重定向  302 临时重定向  304 同一个请求，没有任何变，命中缓存  4XX 404 请求资源不存在  403 权限不够  405 请求方法不被允许  5XX 500服务器代码错误  3.原因短句（对响应状态码的简短解释） |

②响应头

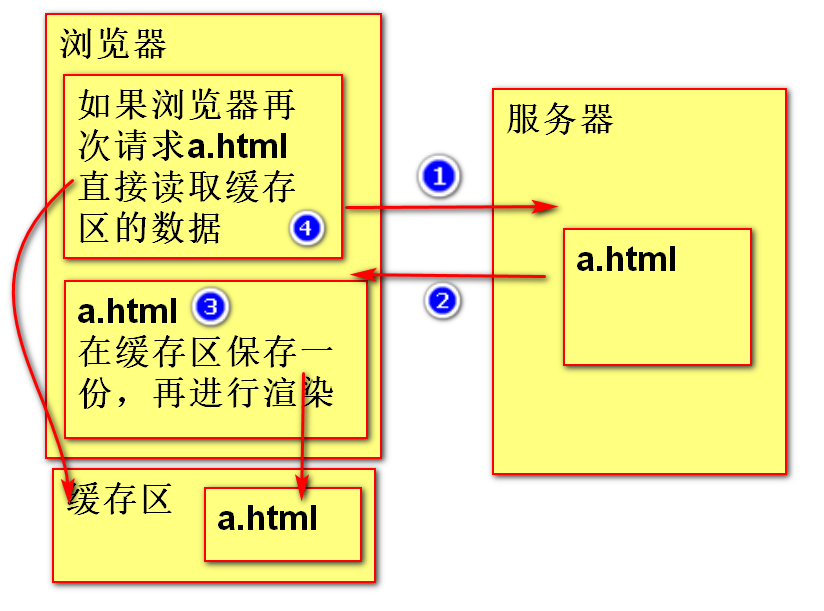
|  |
| --- |
| **1.Date:** Sun, 05 May 2019 06:23:42 GMT  告诉浏览器，服务器响应的时间，格林威治时间  2. **Connection:** keep-alive  告诉浏览器，服务器已经开启了持久连接  3.Conten-Type:  告诉浏览器，响应主体的类型是什么  取值 text/html 响应回来的数据是html文本  text/css响应回来的数据是css文本  application/javaScript 是js文本  image/jpg gif png 响应回来的是图片  text/plain 响应回来的是普通文本  application/json响应回来的是json字符串  application/xml 响应回来的是xml字符串 |

③响应主体

7.缓存

客户端经服务器响应回来的数据进行自动的保存

当再次访问的时候(请求不改变)，直接使用保存的数据



①缓存的优点：

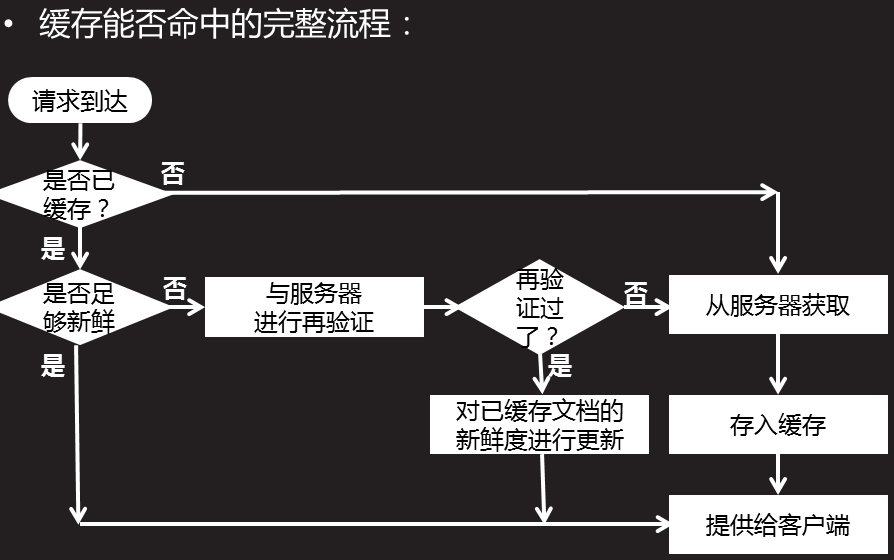
1.减少冗余的数据传输，节省客户端流量

2.节省服务器带宽

3.降低了对服务器资源的消耗和运行的要求

4.降低了由于远距离传输而造成的加载延迟

②缓存的新鲜度和过期



|  |
| --- |
| 1.请求—无缓存—连接服务器—存缓存—客户端得到数据  2.请求—有缓存—够新鲜—使用缓存—客户端得到  3.请求—有缓存—不新鲜—连接服务器确认是否过期—没有过期—更新缓存新鲜度—客户端得到的数据  4.请求—有缓存—不新鲜—连接服务器确认 |

②在网页添加

|  |
| --- |
| <meta http-3> |

8.HTTP 性能优化

①HTTP的连接过程

|  |
| --- |
| 发起请求-->建立连接-->服务器处理请求-->访问资源-->构建响应-->发送响应-->记录日志 |

②HTTP连接性能优化

|  |
| --- |
| 1.减少连接的创建次数(开启持久连接)  2.减少请求的次数（整合代码，不要让文件过多） 3.提高服务器端运行速度  4.尽可能的减少响应数据的长度 |

9.安全HTTP协议。https

|  |
| --- |
| HTTPS：安全版本的http协议  SSL：为数据通信特供安全支持  ①客户端发送请求--->ssl层加密-->服务器接收到加密文件-->ssl层解密，得到明文  ②服务器发送响应--->ssl层加密-->客户端得到加密文件-->ssl层解密，得到响应内容 |

ajax常见错误

|  |
| --- |
| 404 请求资源不存在  url是否写错  接口的路径是否写错  路由是否写错  文件位置是否放错 |
| 500服务器代码错误  一定是服务器中代码出错 |

作业

1.熟练完成nodejs接口，路由，静态资源

2.使用table完成下面布局



# AJAX-day02

day05 Ajax

一.Dom操作(简单dom操作)

|  |
| --- |
| ajax提交请求，不需要使用form表单  但是，form表单自带收集数据的功能  不使用form标签，就没有自动收集数据的功能了  我们需要自己手敲代码，收集数据  使用dom技术 |

1.完整的javascript的组成

①js核心，ECMA Script 6

②Dom:Document Object Model 文档对象模型

③Bom:Browser Object Model 浏览器对象模型

2.使用js的Dom获取页面某个元素

①获取元素对象

|  |
| --- |
| 1.使用元素的id获取对象  var elem=document.getElementById("id"); |

②通过对象，获取元素的内容/值

|  |
| --- |
| 双标签中间的东西，叫内容  单标签的value，叫值 |
| 获取内容，使用dom对象的innerHTML属性  获取：var res=elem.innerHTML;  设置：elem.innerHTML="abc"; |
| 获取值，使用dom对象的value属性 |

③使用dom操作的进阶

|  |
| --- |
| 获取对象的方案进阶  1.不使用id就能获取对象(后期xml中要使用)  通过标签名称获取当前页面中所有的该标签的元素，通过下标，获取某一个元素  var elem=document.getElementsByTagName("div")[0];  2.使用id获取对象的简写方式，ES6支持的写法  直接使用id代表对象  <div id="d1">测试文本1</div>  var res=d1.innerHTML; |

练习：01\_ex.html 一个div，一个input-text,button

需求，点击按钮，把input-text中填入的文本

显示到div上

|  |
| --- |
| 对于innerHTML的进阶  innerHTML是可以改变html的结构的  <div id="d1">  <a href="">123</a>  </div>  <script>  console.log(d1.innerHTML);  //innerHTML是可以改变html的结构的  d1.innerHTML+="<h1>abc</h1>"  </script> |

④事件

|  |
| --- |
| 通过用户的行为来激发的操作，就是事件  用户发生一个行为，这个行为绑定的方法就会被调用  button的事件  onclick 单击事件  文本框和密码框  onblur 焦点移除事件  onfocus 获取焦点事件onbluronfocus  body标签的事件  onload 页面加载完成后，马上调用js方法 |

练习

|  |
| --- |
| 03\_ex.html  1.页面中创建如下元素  文本框 uname  密码框 upwd  重复验证密码框 cpwd  在上面三个元素后面各创建一个span（3个span）  2.需求  1.文本框获取焦点时，span提示用户名6~10位  失去焦点时，非空验证，显示span中  2.密码框同上  3.cpwd失去焦点时，判断与upwd的值是否一致，提示写在span中 |

二.Ajax

1.同步Synchronous

|  |
| --- |
| 在一个任务进行的过程中，不能开启其它任务  同步访问：浏览器在向服务器发送请求时，浏览器只能等待服务器响应，不能做其他事  同步访问的出现场合：  1.地址栏输入url，是同步访问  2.a标签跳转  3.form提交 |

2.异步Asynchronous

|  |
| --- |
| 在一个任务开启时，可以进行其它的任务  异步访问：浏览器在向服务器发送请求时，用户可以在页面上做其它的操作  使用异步的场合：  1.用户名重复的验证  2.聊天室  3.百度搜索建议  4.股票走势图 |

3.什么是ajax

|  |
| --- |
| Asynchronous JavaScript And XML  异步的 js 和 xml  本质：使用js提供的异步对象，  异步的向服务器发送请求，  并接收响应回来的数据  异步对象 XMLHttpRequest |

4.使用ajax

|  |
| --- |
| 1.创建异步对象  2.创建请求  3.发送请求  4.接收响应数据 |

①创建异步对象

|  |
| --- |
| var xhr=new XMLHttpRequest();  这种创建方式，不兼容IE8以下的版本  下面是兼容ie8以下版本的创建方式  if(window.XMLHttpRequest){  //如果有这个属性，说明是IE8以上的浏览器  var xhr=new XMLHttpRequest();  }else{  //如果没有这个属性,说明是IE8以下的浏览器  var xhr=new  ActiveXObject("Microsoft.XMLHttp");  } |

②创建请求，打开连接

|  |
| --- |
| xhr.open(method,url,isAsyn);  method：string类型，请求的方法  url:string类型，请求的url  isAsyn:boolean，是不是要采用异步的方式提交请求 |

③发送请求

|  |
| --- |
| xhr.send(formdata);  注意：只有post请求，才有请求主体formdata  get方法不需要请求主体。  所以使用get请求的时候，有两种发送请求的写法  xhr.send(),xhr.send(null) |

④接收响应

|  |
| --- |
| 1.xhr.readyState属性  用于表示xhr对象的 请求状态  一共有5个状态  0：请求尚未初始化  1：已经打开连接，请求正在发送中  2：接收响应头  3：接收响应主体  4：接收响应数据成功 |
| 2.xhr.onreadystatechange监听  监听xhr.readyState值的改变，每改变一次，方法会调用一次，一共会调用4次。  注意：只有当xhr.readyState=4的时候，是我们要接收响应正确时刻 |
| 3.xhr.status响应状态码  只有响应状态码为200的，我们才接收响应  4.响应数据放在了xhr.responseText属性中 |
| xhr.onreadystatechange=function(){ if(xhr.readyState==4&&xhr.status==200){ var result=xhr.responseText;  console.log(result);  }  } |

5.带参数的get请求

xhr.open("get","http://127.0.0.1:8080/ajax/ajaxDemo1?uname="+u\_name+"&upwd="+u\_pwd,true);

注意：

|  |
| --- |
| 1.服务器中的get接口，可以使用浏览器地址栏验证 |
| 2.跨域错误    把ajax的html文件放入node服务器的静态资源中托管，使用访问服务器的方式打开页面 |
| 3.大括号缺失 |
| 4.没有错误信息的错误  使用console.log在不同位置打桩观察 |
| 5.错误的符号 |

作业1：

|  |
| --- |
| 使用get请求，完成完整的登录模块  input输入---ajax异步发送请求---服务器接收数据--查询数据库--发送响应--页面接收响应 |

作业2：

|  |
| --- |
| 使用get请求，获取商品列表，把响应得到的数据在div中显示 |

提高作业：

|  |
| --- |
| 注册验证，验证新用户名是否可用  模仿学子商城注册页面，使用onblur事件 |

# AJAX-day03

作业1：

|  |
| --- |
| 使用get请求，完成完整的登录模块  input输入---ajax异步发送请求---服务器接收数据--查询数据库--发送响应--页面接收响应 |
| 1.完成接口  2.完成ajax  C:\Users\web\Desktop\web1903\day06\day06_getlogin_html.pngC:\Users\web\Desktop\web1903\day06\day06_getlogin_node.png |

一.ajax的post请求

1.任务，使用get请求再完成一遍登录模块 10:45我写

接口名称 login\_post

2.使用post方法发送请求

①xhr.open，需要改成post方法

②xhr.open中的url，不需要？和参数了

③添加创建请求主体的代码，把数据放到请求主体中

④修改请求消息头，可以发送特殊符号

⑤xhr.send(请求主体)

|  |
| --- |
| //url不要?和后面参数  var url="http://127.0.0.1:8080/ajax/login\_post";  xhr.open("post",url,true);  //创建请求主体  var formdata="uname="+u\_name  +"&upwd="+u\_pwd;  //由于ajax默认传输是text/plain  //无法传递特殊符号，我们需要更改消息头  //让ajax请求可以传递特殊符号  xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  xhr.send(formdata); |

二.Json的解析

作业2：

|  |
| --- |
| 使用get请求，获取商品列表，把响应得到的数据在div中显示  1.创建一个接口userlist  2.在html使用ajax |
| 前台，ajax内部，xhr.responseText得到数据，是string类型。string中放着js对象组成的数组  1.js对象的数据格式  var obj={name:"abc",age:18}  语法：var 对象名称={属性1：值1，属性2：值2.....}  注意：js对象中，属性名没有双/单引号  2.JSON数据格式  json是一个string字符串，亲切的称之为json串  JavaScript object Notation  js 对象 表示法  以js对象的格式表现出来的字符串  3.json字符串的格式  ①用一对{}表示一个对象  ②json中对象的属性名，必须使用""引起来(使用单引号也是正确的，但是不推荐)  ③属性的值如果是字符串，也要带双引号  ④json的表现是一个字符串，所有最外层加引号(推荐使用单引号)  ⑤请求访问服务器，调用数据库代码，得到的结果，默认是一个json串  4.json解析，把json字符串转换成js对象数组  var arr=JSON.parse(result); |

练习

|  |
| --- |
| 04\_list.html 重写一遍get方法获取userlist01接口  要求，数据最后变成下图样式 15:05~15:20休息  C:\Users\web\Desktop\web1903\web第二阶段练习效果图示\ajax用户列表效果.png |

三.XML的解析

1.什么是XML

|  |
| --- |
| eXtensible Markup Language  可拓展 标记 语言  是html的一个变种，专门负责承载数据用的  xml中标签，关键字/属性都需要自己定义  语法  1.第一行写版本声明  <?xml version="1.0" encoding="utf-8"?>  2.xml的标记只有双标记，必须成对出现  3.xml的标签严格区分大小写，开始标签和结束标记必须一致  4.xml标记，允许嵌套，但是要注意嵌套顺序  5.每个xml文档，有且只有一个根元素 |

2.xml解析

|  |
| --- |
| 使用dom解析  ??.getElementsByTagName("student")  ①不能使用xhr.responseText获取xml。这样获取的xml不能解析。要使用var result=xhr.responseXML;  ② |

四.项目阶段

常见错误

|  |
| --- |
| 数据库出问题，有可能是sql语句错误 |
| 设置请求消息头，必须在xhr.opne和xhr.send之间写 |
| xml文件中，有结构错误 |

ajax项目

一.搭建项目结构

|  |
| --- |
| router文件夹中创建路由pro.js---------接口  挂载路由  创建静态文件夹mypro，与public平级-------html  静态文件夹资源的托管 |

1.登陆模块

|  |
| --- |
| 1.在pro.js中定义接口login  使用post方法，接收前端传递过来的用户名和密码  查询数据库，给出响应  2.在mypro文件夹下，新建网页01\_login.html  添加文本框和密码，点击按钮完成ajax异步请求  登陆响应结果使用弹窗显示17:13完成 |

2.显示用户列表的模块

|  |
| --- |
| 1.在pro.js中创建接口list get  查询用户所有数据，并响应给前端  2.在mypro文件夹中，创建网页02\_list.html  页面加载时，发起ajax异步请求。把数据格式显示成下图模式  C:\Users\web\Desktop\web1903\web第二阶段练习效果图示\ajax用户列表效果.png |

作业：

1.使用get方法获取userlist接口数据，显示成表格



2.删除

|  |
| --- |
| input-text输入uid,点击按钮，删除数据库中数据 |

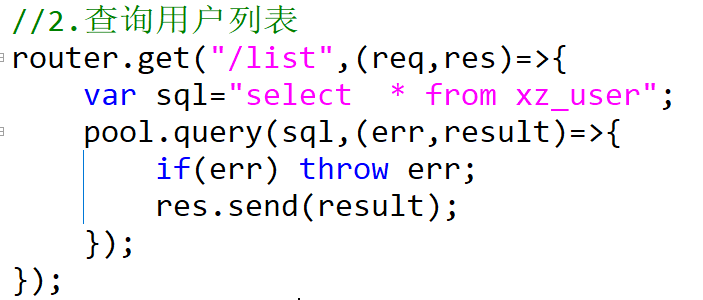
# AJAX-day04

day07

一.登录模块

二.查询所有的用户信息9:35我写

|  |
| --- |
| 1.在pro.js中创建接口list get  查询用户所有数据，并响应给前端  2.在mypro文件夹中，创建网页02\_list.html  页面加载时，发起ajax异步请求。把数据格式显示成下图模式  C:\Users\web\Desktop\web1903\web第二阶段练习效果图示\ajax用户列表效果.png |



三.删除

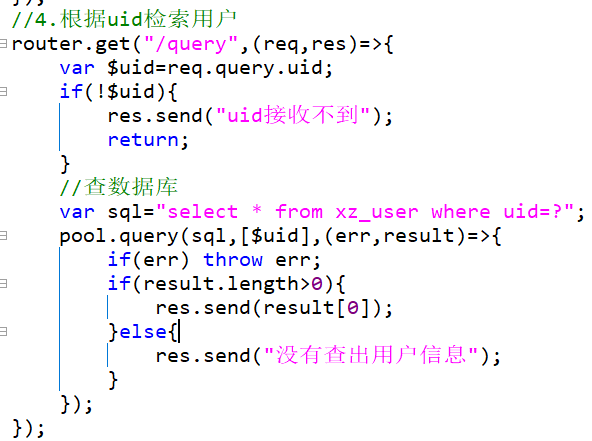
|  |
| --- |
| input-text输入uid,点击按钮，删除数据库中数据  get方法 |





四.修改模块

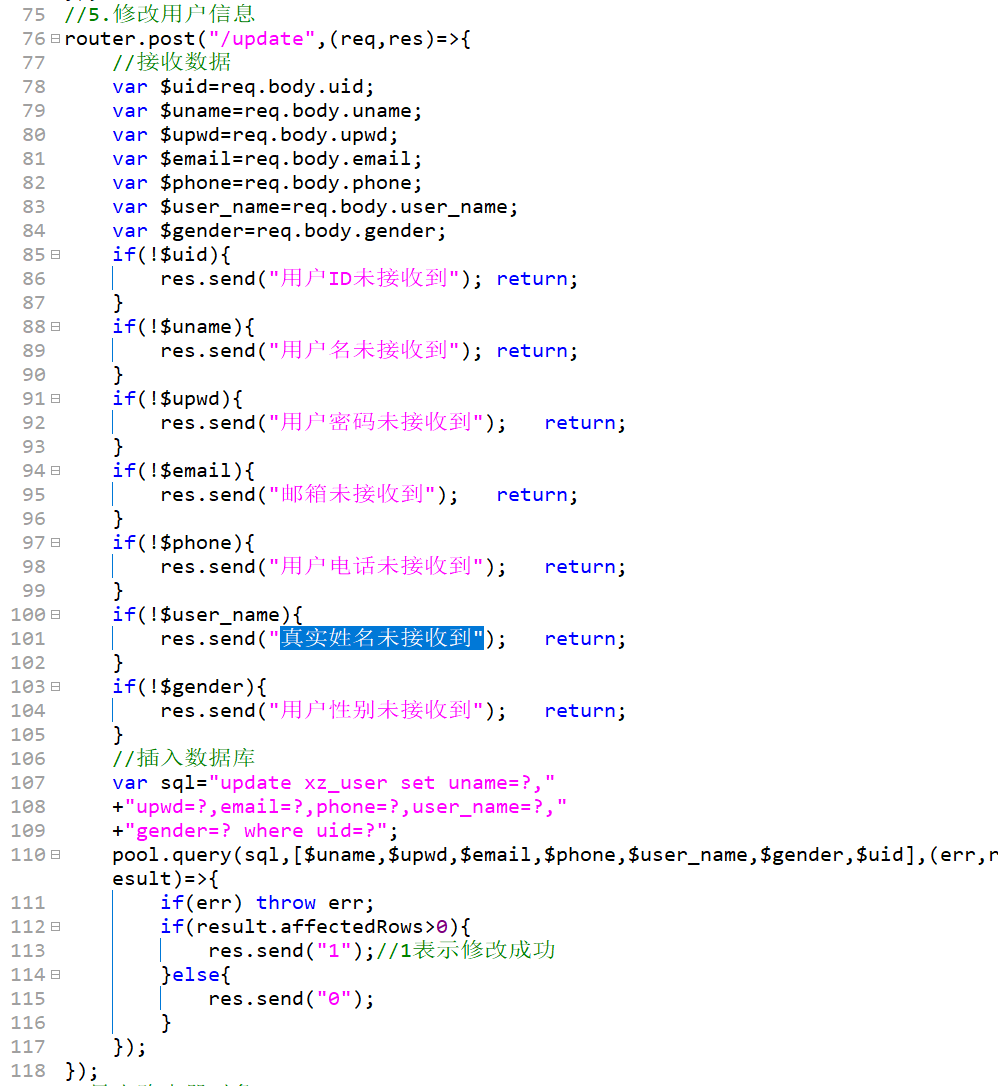
1.根据uid检索用户





2.修改模块

接口post方法 /update



作业：

1.注册

|  |
| --- |
| 1.注册名重复的验证  2.账户注册 |

常见错误

|  |
| --- |
| 服务器没有接受到参数，把提示当成响应传递给前台  前台把提示按照json格式解析了 |

# CSSDAY-01

day08

一.CSS概述

1.什么是css

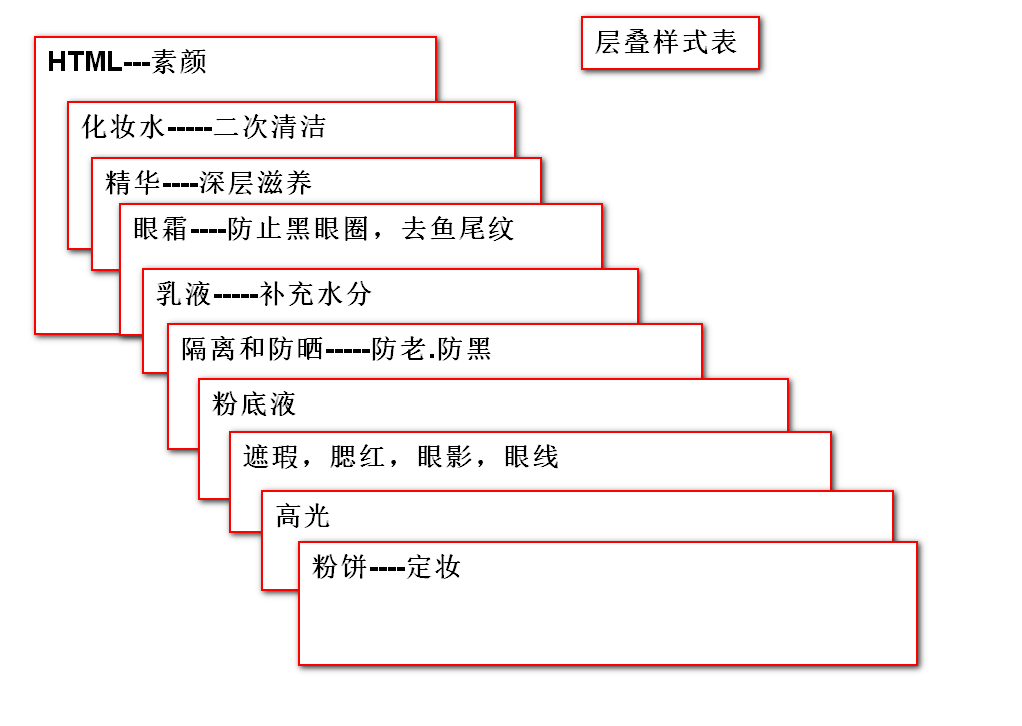
|  |
| --- |
| Cascading style sheets 层叠样式表，级联样式表  简称样式表 |

2.css的作用

|  |
| --- |
| 设置html网页中元素的样式，美化页面 |

3.HTML与CSS的关系

|  |
| --- |
| HTML：负责网页的搭建，内容的展示  CSS：负责网页的修饰，样式的构建 |



4.CSS与HTML的属性，使用原则

|  |
| --- |
| W3C规定我们尽量的使用css的方式来取代HTML属性  css样式  1.样式代码可以高度重用，html属性不能重用  2.方便后期维护，提高可维护性。 |

二.css的语法规范

1.使用css样式的方式

①行内样式(内联样式)

|  |
| --- |
| 在html标签的style属性中，编写样式  语法：<any style="样式声明"></any> any:任意标签  样式声明：样式属性 : 值  有多个样式声明：样式属性1：值1 ；样式属性2：值2  ex:  <div style="color:red;background-color:yellow">  内联样式在项目中，很少使用。  学习和测试的时候用  原因：1.内联样式不能重用  2.内联样式优先级最高 |

目前常用的样式

|  |
| --- |
| color:orange; 字体颜色  background-color:yellow; 背景颜色  font-size:36px； 字号大小 |

②内部样式

|  |
| --- |
| 在网页头部中，添加一对<style></style>标签  在style标签定义此网页所有的样式  <style>  选择器{样式属性：值；样式属性：值；........}  </style>  选择器：就是一个条件，符合这个条件元素，可以应用这个样式  项目中使用内部样式较少，学习和测试使用较多  内部样式的重用不能在其他html页面中生效 |

③外部样式

|  |
| --- |
| 单独创建一个css文件，编写样式  在html页面的head中使用link，引入这个外部样式  <link rel="stylesheet" href="css文件的url">  注意：rel="stylesheet"必须写，不然无效  此种方式，是开发中使用最多方式。 |

练习

|  |
| --- |
| 01\_ex.html  h1 内容随意，内联，背景为粉色pink，字体为yellow  h2 内容随意，内部，背景为粉色yellow，字体为red  h3 内容随意，外部，背景为粉色red，字体为blue |

2.CSS的特性

①继承性

|  |
| --- |
| 大部分的css效果，是可以被子元素继承的  必须是父子结构，子继承父 |

②层叠性

|  |
| --- |
| 可以为一个元素定义多个样式规则  多个规则中样式属性不冲突时，可以同时应用到当前元素上 |

③优先级

|  |
| --- |
| 如果对一个元素的多个样式声明，发生冲突时，按照样式规则的优先级去应用  默认优先级：由高到低  1.内联样式(行内样式)  2.内部样式和外部样式，就近原则  3.浏览器默认样式 |

④调整优先级

|  |
| --- |
| !important规则  放在属性值后面，与值之间有一个空格  作用，调整样式显示的优先级  内联样式不可以写!important  !important优先级，比内联样式高 |

练习

|  |
| --- |
| 02\_ex.html中  一个p标签，内容假文lorem  用内部样式设置文本颜色为蓝色，字号24px  用外部样式设置文本颜色为 红色，字号40px  让外部样式引入，F12查看页面效果  然后对调内部样式和link的位置，F12观察结果  尝试使用!important调整样式优先级 |

三.基础选择器(重点\*\*\*\*\*\*\*\*\*)

1.选择器的作用

|  |
| --- |
| 规范了页面中哪些元素能够使用定义好的样式  选择器就是为了匹配元素用  (选择器是一个条件，符合这个条件的元素就可以使用这个样式) |

2.选择器详解

①通用选择器

|  |
| --- |
| \*{样式声明}  由于\*的效率低下，项目中很少使用通用选择器  唯一使用的场合\*{margin:0;padding:0}所有元素内外边距清0 |

②元素选择器(标签选择器)

|  |
| --- |
| 页面中所有对应的元素，都应用这个样式  设置页面中某种元素的默认样式  元素名称{样式声明}  ex:p{} div{} |

③ID选择器，专属定制

|  |
| --- |
| <any id="id值"></any>  #id值{样式声明} 这种写法仅仅对页面上一个标签生效  一般id选择器在项目中很少单独使用。  通常会作为子代选择器或者后代选择器的一部分 |

练习

|  |
| --- |
| 03\_ex.html h2标签，内容随意(假文)，这个h2标签id为text1  使用id选择器设置文本为purple，背景yellow，  字体为斜体font-style:italic；查看页面观察效果  再使用元素选择器，设置文本为红色，背景pink。  这里有坑，会对就近原则产生怀疑，选择器权值问题 |

④类选择器

|  |
| --- |
| <any class="类名"></any>  .类名{样式声明}  作用：定义页面上某个或者某类元素的样式  (公共样式，谁想用，谁就可以用)  class属性来引用类名  类名的规则：1.类名之前的点不能省略，引用的时候没有点  2.类名不能以数字开头  3.类名只能包含 - \_ 符号  4.类名尽量的见名知意  类选择器的特殊用法  1.多类选择器  让元素引用多个类名，这些类的样式都会作用到当前元素上  <div class="text-danger bg-warning font-24">d</div>  2.分类选择器  元素选择器+类选择器{}  div.text-danger{ }div元素有text-danger类,才能应用这个样式  类选择器+类选择器{}  .font-24.text-danger{ }  元素，必须有font-24类和text-danger，才能应用此样式  作用：1.更精确的确定使用样式的元素  2.增加选择器的权值 |

练习

|  |
| --- |
| 04\_ex.html 页面中有div和p元素，内容假文随意  用类选择器设置字体颜色为红色，注意类名的命名规范  用分类选择器为p元素设置背景色为green |

⑤群组选择器

|  |
| --- |
| 将多个选择器放在一起定义公共的样式  语法：选择器1,选择器2,......{公共样式声明}  ex:  #content,div,.mycolor,p.text{color:red}  相当于  #content{color:red}  div {color:red}  p.text {color:red}  .mycolor {color:red} |

⑥后代选择器

|  |
| --- |
| 通过元素的后代关系匹配元素  后代：一级嵌套或者多级嵌套  语法：选择器1 选择器2 选择器3........{}  #content p span{color:red;}  id我content的元素，内部不管隔着多少代，有一个p元素  p元素内部，不管隔着多少代，有一个span.这个span就符合要求 |

⑦子代选择器

|  |
| --- |
| 通过元素的子代关系匹配元素  子代：一级嵌套，直接的儿子  选择器1>选择器2>....{}  子代选择器和后代选择器可以混写  #content p>span{background-color:yellow;} |

练习

|  |
| --- |
| 05\_ex.html  ul#content>li\*4>span>a  a标签字体36px，字体颜色red，背景颜色yellow  使用后代或者子代选择器，至少3种选择器写法 |

⑧伪类选择器

|  |
| --- |
| 匹配同一个元素，不同的状态下的样式  所有的伪类选择器都是这样开头的 选择器:  1.链接伪类  a标签，没有访问的状态  a:link{color:red;}  a标签，被访问过之后的状态  a:visited{color:black;}  2.动态伪类  :hover 鼠标悬停在元素上时，元素的样式  :active 匹配元素被激活时的状态  :focus 匹配元素获取焦点时的状态 |

练习

|  |
| --- |
| 1.06\_ex.html  一个a标签，内容随意，href随意-------------坑  1.访问后，文本颜色orange  2.被激活时，文本颜色绿色  3.鼠标悬停时，文本颜色红色  4.未被访问时，文本pink  2.页面在红添加一个input--text  默认字体为gray，font-style:italic；  1.被激活时，字体不斜：font-style:normal;  2.获取焦点时：文本为红色 |

⑨选择器的权值

|  |
| --- |
| 权值：标识当前选择器的重要程度，权值越大，优先级越高  !important >1000  内联样式 1000  id选择器 100  类选择器 10  元素选择器 1  \*通用选择器 0  继承的样式 无 |
| 权值的特点  1.当一个选择器中含有多个选择器时，需要将所有的选择器权值进行相加，然后比较，权值大的优先显示  2.权值相同，就近原则  3.群组选择器的权值，单独各算各的，不能相加  4.样式后面追加!important直接获取最高优先级  内联样式不能加!important  5.选择器的权值计算，结果不会超过自己的最大数量级  (1000个1相加，也不会大于10) |

作业1：css使用方法3个

css特性3个

选择器8个

权值

上述知识点的demo重做一遍

作业2：重新完成学子注册模块

# CSSDAY-02

day09

四.尺寸和边框

1.尺寸属性

|  |
| --- |
| 作用，设置元素的宽高  属性：  width: 设置宽  height:设置高  ----------------------  max-width 最大宽度  min-width 最小宽度  max-height 最大高度  min-height 最小高度  使用场合，响应式布局  取值：以px为单位的数字  父元素尺寸的%百分比 |

附加知识点：单位

|  |
| --- |
| 尺寸的单位  1.px像素  2.in英寸 1in=2.54cm  3.pt 磅值，多用于修饰字体大小粗细 1pt=1/72in  4.cm  5.mm  ---------相对单位-------------  6. % 父元素的百分之多少  7.em 是相对于父元素数值的单位  8.rem 是相对html标签数值的单位 |

2.页面中允许设置尺寸的元素

①块级元素

|  |
| --- |
| 所有的块级元素都可以设置尺寸  块级元素不设置宽，宽度占父元素100%。  块级元素不设置高，高度靠内容撑开，没有内容，就没有高 |

②行内元素

|  |
| --- |
| 行内元素设置宽高无效，  行内元素宽高，是靠内容撑开  但是，自带宽高属性的行内元素，可以设置尺寸 |

③table

|  |
| --- |
| table自带宽高属性，可以设置宽高 |

④行内块

|  |
| --- |
| input  行内块可以设置宽高 |

3.溢出处理

|  |
| --- |
| 当内容较大，元素区域较小，就会发生溢出效果  默认是纵向溢出  属性 overflow  取值 1.visible 默认值，溢出部分可见  2.hidden 溢出部分隐藏  3.scroll，不管是否溢出，都添加滚动条。  不溢出的时候，滚动条不能拖动  4.auto 自动，溢出的时候，溢出的方向有滚动条。  不溢出的时候，没有  控制滚动条的方向  overflow-x/overflow-y  如何让内容横向溢出  需要在宽度比较小的容器内部，添加一个宽度较大的元素盛放内容  在父容器上写overflow:auto。就可以做到横向溢出 |

附加知识点---颜色

|  |
| --- |
| 合法颜色值  1.颜色的英文单词(red,blue,yellow,pink,purple......)  2.#rrggbb  #000000---黑 #ffffff---白 #ff0000---红  #00ff00---绿 #0000ff---蓝  3.#aabbcc---->#abc  #000 #fff #f00 #0f0 #00f #666  4.rgb(0~255,0~255,0~255)  rgb(255,0,0)  5.rgba(r,g,b,alpha) alpha透明度 0~1 1不透明，0透明  #f00 #0f0 #00f #ff0 #0ff #f0f  #faa #afa #aaf #ffa #aff #faf |

4.边框

①边框属性---简写方式---4个方向边框一起设置

|  |
| --- |
| border:width style color;  width:边框的宽度，取值以px为单位的数字  style:边框的样式 取值：solid实线  dotted 点点虚线  dashed 线状虚线  double 双实线  color：边框的颜色 取值 合法的颜色值/transparent(透明)  最简方式 border:style;  取消边框 border:0; 或者 border:none; |

练习

|  |
| --- |
| 03\_ex.html  3个div,给每个div设置边框，每个div 10px\*10px  第一个 1px 实线 红色  第二个 5px 点虚线 蓝色  第三个 10px 线虚线 黄色 |

②单边定义边框

|  |
| --- |
| 只设置某一条边的3个属性  border-top：width style color;  border-right  border-bottom  border-left |

练习

|  |
| --- |
| 03\_ex中，div#d4 200px\*200px  上边框，3px实线 红色  右边框 5px 点 蓝色  下边框 10px 线虚线 黄色  左边框 15px 双实线 #f0f |

③单属性定义

|  |
| --- |
| border-width:  border-style:  border-color:  ex:  border-width:3px;  border-style:solid;  border-color:#f00;  border:3px solid #f00; |

④单边单属性

|  |
| --- |
| border-top-width:  border-top-style:  border-top-color:  border-right-width:  border-right-style:  border-right-color:  border-bottom-width:  border-bottom-style:  border-bottom-color:  border-left-width:  border-left-style:  border-left-color: |

5.边框的倒角(圆角)

|  |
| --- |
| 将直角设置成倒角，圆角  border-radius：  取值：以px为单位数字  %  50%就是个圆  单角设置  border-top-left-radius  border-top-right-radius  border-bottom-left-radius  border-bottom-right-radius |

练习

|  |
| --- |
| 画柠檬/芒果 |

6.边框阴影

|  |
| --- |
| box-shadow  取值：h-shadow v-shadow blur spread color inset;  h-shadow:水平方向的阴影偏移量  +：往右移动， -：往左移动  v-shadow:垂直方向的阴影偏移量  +：往下 -：往上  blur：阴影模糊距离，越大越模糊  无负值  spread：阴影尺寸，在基础阴影上扩出来的大小  负值，尺寸变小  color:阴影颜色  inset:向内扩撒阴影 |

练习

|  |
| --- |
| 圆形发光体 太阳，吸顶灯，日食，月食 |

7.轮廓

|  |
| --- |
| 轮廓指的是边框的边框，绘制于边框外的线条  outline:width style color;  outline:none;或者outline:0 去除 轮廓 |

五.框模型---盒子模型

|  |
| --- |
| 框模型--元素在页面上实际占地空间的计算方式  默认情况，一个元素在页面的实际占地宽度  左外边距+左边框宽度+左内边距+内容区域宽度+右内边距+右边框+右外边距  实际占地高度  上外边距+上边框宽度+上内边距+内容区域高度+下内边距+下边框+下外边距 |

1.外边距margin

|  |
| --- |
| 元素边框以外的距离，改变margin，元素有位移的效果 |

①语法

|  |
| --- |
| margin:v1; 设置4个方向的外边距  margin:v1 v2; v1上下 v2左右  margin:0 auto; 块级元素水平居中  auto对垂直外边距无效  margin:auto;  auto只对设置了宽的块级元素生效  margin:v1 v2 v3; v1上 v2左右 v3下  maring:v1 v2 v3 v4; 上右下左  单方向外边距设置  margin-top:  margin-right:  margin-bottom:  margin-left:  取值：1.以px为单位数字  2.%  3.+ margin-top ↓ margin-left →  - margin-top ↑ margin-left ←  4.auto 自动计算块级元素外边距，让块级元素水平居中  auto只对设置了宽度的块级生效  auto对下上外边距无效 |

练习

|  |
| --- |
| 05\_ex  两个div，尺寸300px\*300px  背景色随意。  设置两个div之间的间距是50px，  两个div都水平居中 |

2.外边距的特殊效果

①外边距合并

|  |
| --- |
| 两个垂直外边距相遇时，他们将合并成一个，值以大的为准  只能在布局设计的时候，尽量避免发生 |

作业

1：坑

|  |
| --- |
| 页面中两个div，宽高200px,分别设置背景颜色  为两个div设置4个方向外边距，观察效果  与div平级，添加两个span，内容随意，分别设置背景颜色  为两个span添加4个方向外边距，观察效果  与div平级，添加两个input-text，  为两个input添加4个方向外边距，观察效果 |

2.坑

|  |
| --- |
| div#d1>div#d2  #d1宽高300px  #d2宽高100px  分别设置背景颜色  给#d2设置上外边距，观察效果 |

3.今天所有知识点的demo

4.ajax 登录模块/注册模块/查询用户列表/删除/修改

# CSSDAY-03

day10

五.框模型

2.外边距的特殊效果

①外边距合并

|  |
| --- |
| 两个垂直外边距相遇时，他们将合并成一个。值以最大的为准。 |

练习

|  |
| --- |
| 01\_ex  1.两个div 200px\*200px 背景颜色不同  为两个div单独设置4个方向外边距 margin-方向  2.两个span，内容随意，背景颜色不同  为两个span单独设置4个方向外边距  3.两个文本框，  为两个input单独设置4个方向外边距 |

②关于块级元素，行内元素，行内块的外边距总结

|  |
| --- |
| 行内元素的特点：  设置宽高无效，宽高根据内容自动撑开  上下垂直外边距无效。可以与其他的行内元素和行内块元素共用一行 |
| 块级元素特点：  设置宽高有效。如果不设置宽高，高按内容撑开，宽占父元素宽度的100%。上下外边距有效，独占一行. |
| 行内块的特点：input  设置宽高有效，不设置宽高，自带默认宽高。  上下外边距有效，但是同一行修改一个行内块的垂直外边距，整行都会跟着一起改变位置。可以与其他行内元素和行内块共用一行 |

③自带外边距的元素

|  |
| --- |
| body h1~h6 p ol ul dl pre  由于不同浏览器对默认的外边距解析可能有差别  所以一般情况下，开发之前，需要把内外边距清空  \*{margin:0;padding:0}  或者通过群组选择器来写  blockquote,body,button,dd,dl,dt,fieldset,form,  h1,h2,h3,h4,h5,h6,hr,input,legend,li,ol,  p,pre,td,textarea,th,ul{margin:0;padding:0} |

练习

|  |
| --- |
| 01\_ex  div#d3 尺寸300px\*300px  #d3中有div#d4 100px\*100px  两个div不同背景颜色。  给#d4设置上外边距，在f12中改变上外边距的数值  观察效果 |

④外边距溢出

|  |
| --- |
| 在特殊情况下，为子元素设置外边距，会作用到父元素上  特殊情况：  1.父元素没有上边框  2.子元素内容区域上边与父元素内容区域上边重合的时候  (为父元素中第一个子元素设置上外边距时，这种说法不严谨)  解决方案：  1.给父元素添加上边框，弊端，影响父元素实际占地高度  2.给父元素添加上内边距，弊端，影响父元素实际占地高度  3.给父元素的一个子元素的位置添加空table元素 |

3.内边距

|  |
| --- |
| 内边距的改变效果，感觉是改变了元素的大小  不会影响其它元素，但是真正改变的是本元素的占地尺寸  内边距颜色和元素背景相同  语法：  padding:v1; 设置4个方向的内边距  padding:v1 v2; 上下 左右  padding无auto  padding:v1 v2 v3; 上 左右 下  padding:v1 v2 v3 v4;上右下左  单方向设置  padding-top  padding-right  padding-bottom  padding-left  取值，以px为单位的数字 %  padding无auto值 |

4.box-sizing属性(设置元素实际占地尺寸的公式)

|  |
| --- |
| box-sizing:  取值：1.content-box.默认值，按照之前的盒子模型计算元素占地大小(左外+左边+左内+内容区域宽度+右内+右边+右外)  2.border-box。设置的width和height是包含border+padding+内容区域的 整体宽高  元素实际占地大小  左外+设置的width+右外  上外+设置的height+下外  给复杂元素关系做布局时，经常使用border-box |

六.背景

1.背景颜色

|  |
| --- |
| background-color:合法颜色值 |

2.背景图片

|  |
| --- |
| background-image:url(08.png);  使用背景图片，可以让该元素的子元素，堆叠显示在背景图片上。  而使用img标签，默认不会有堆叠效果 |

3.背景图片的平铺

|  |
| --- |
| background-repeat:  取值：repeat 默认值，平铺  no-repeat 不平铺  repeat-x 水平方向平铺  repeat-y 垂直方向平铺 |

4.背景图片的定位

|  |
| --- |
| background-position:  取值：1. x y 以px为单位的数字  2.x% y% 百分比  3.关键字 x:left/center/right  y:top/center/bottom |

5.背景图片的尺寸

|  |
| --- |
| background-size:  取值：1. x y 以px为单位设置具体宽高  2.x% y% 按父元素宽高比设置  以元素的角度考虑这两个单词  3.cover 充满，只要求元素被背景图充满，背景图显示不全，也没关系.  4.contain 包含，只要求元素要把背景图整个包含住，元素有空白，也没关系 |

6.背景图片的固定

|  |
| --- |
| background-attachment  取值： scroll 默认值，背景图会随着窗口滚动条滚动  fixed 固定，背景图相对于窗口固定，窗口滚动时，背景图位置不变，但是只会在原容器内显示 |

7.背景的简写方式

|  |
| --- |
| background:  取值 color image repeat attachment position  最精简方式 background:color/image; |

七.渐变 gradient

1.什么是渐变

|  |
| --- |
| 渐变是指多种颜色，平缓变化的一种显示效果 |

2.渐变的主要因素

|  |
| --- |
| 色标：一种颜色，以及这种颜色出现的位置  一个渐变，最少两个色标 |

3.渐变的分类

|  |
| --- |
| 1.线性渐变，以直线的方式，来填充渐变色  2.径向渐变，以圆形的方式，来填充渐变色  3.重复渐变，将线性，径向渐变重复几次显示 |

4.线性渐变

|  |
| --- |
| background-image: linear-gradient(angle,color-point1, color-point2......);  angle:表示渐变的方向  取值 1.关键字 to top 从下往上  to right 从左往右  to bottom 从上往下  to left 从右往左  2.角度 0deg to top  90deg to right  180deg to bottom  270deg to left  可以取负值  color-point:色标 颜色 %/px |

5.径向渐变

|  |
| --- |
| background-image:  radial-gradient(半径 at 圆心x 圆心y,color-point1,  color-point2.................)  半径 以px为单位的数字  圆心 1 x y 以px为单位的数字  2. x% y%  3关键字 x left/center/right  y top/center/bottom  color-point: 位置取值 1. % 半径的百分比%  2.px为单位的数字，半径失效 |

6.重复渐变

|  |
| --- |
| 重复的线性渐变  background-image:  repeating-linear-gradient(to left,#000 0px,  #ff0 25px,#000 50px);  重复的径向渐变  background-image:  repeating-radial-gradient(50px at center center,  #000 0%,#0ff 25%,#000 50%); 17:07~17:23休息 |

7.浏览器兼容问题

|  |
| --- |
| 如果有低版本(ie8.0以下)，想要使用渐变  chrome/safari -webkit-  firefox -moz-  opera -o-  IE -ms-  要低版本兼容渐变，需要在linear-gradient之前添加浏览器内核  background-image:  -webkit-linear-gradient()  background-image:  -moz-linear-gradient()  ......  兼容低版本浏览器，线性渐变的方向，要改变写法  不写to top/to left/to bottom/to right  改成初始点 top/left/bottom/right |

八.文本格式化(重点\*\*\*\*\*\*\*\*)

1.字体属性

①指定字号大小

|  |
| --- |
| font-size:  取值 px/pt/rem/em为单位的数字 |

②字体的类型

|  |
| --- |
| font-family |

一个页面中，编写css的思路

|  |
| --- |
| 从上往下，从左往右，从外往里  1.尺寸，大体位置  2.边框，背景相关  3.文本相关  4.微调 |

作业1：

margin:0 auto和text-align:center区别

作业2：完成登陆页的样式(table布局)



作业3.渐变的所有demo

# CSSDAY-04

day11

|  |
| --- |
| 1.尺寸，大体位置  2.边框，背景  3.文本  4.微调 |

三.文本格式化(重点\*\*\*\*\*\*\*\*)

1.字体属性

①字号大小

|  |
| --- |
| font-size:  取值：px/pt/em/rem |

②设置字体类型

|  |
| --- |
| font-family:"mv boli",华文彩云,黑体;  在当前设备的字体库，查找字体，如果有就使用，如果没有，就查找下一个。  如果字体名称中间有空格，必须加双引号 |

③粗体

|  |
| --- |
| font-weight:  取值 1.关键字 lighter normal bold bolder  2.无单位的100的整倍数 一般400~1000 |

④字体样式

|  |
| --- |
| font-style:normal/italic; 正常和斜体 |

⑤小型大写字母

|  |
| --- |
| font-variant:small-caps; |

⑥简写方式

|  |
| --- |
| font:style variant weight size family;  font:italic small-caps bolder 40px chiller;  最精简的方式 font:size family; |

2.文本属性

①字体颜色

|  |
| --- |
| color |

②文本水平对齐方式

|  |
| --- |
| text-align:  取值 left/center/right/justify两端对齐 |
| margin:0 auto和text-align:center的区别  margin:0 auto，控制的是当前元素本身在页面中居中(让自己居中)  text-align:center 控制的是当前元素内部的内容，在元素内部居中(让我的儿子居中) |

③行高

|  |
| --- |
| 定义一行数据的高度  特性：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，呈垂直居中方式显示  line-height  取值：1.以px为单位的数字，一般，行高的值与容器高度相同。  这样就可以让文字在容器垂直中间显示了  2.无单位的数字(整数小数都可以)，行高为字号大小的倍数  注意:文字如果有多行，不建议使用行高，文字会溢出，每一行的行高都是你设置的大小 |

④文本线条修饰

|  |
| --- |
| text-decoration  取值：overline 上划线  underline 下划线  line-through 删除线  none 去掉所有的修饰线条  a{text-decoration:none}去除a标签下划线 |

⑤首行缩进

|  |
| --- |
| text-indent:  取值：px为单位的数字 |

⑥文本阴影

|  |
| --- |
| text-shadow:h-shadow v-shadow blur color;  h-shadow:水平偏移  v-shadow:垂直偏移  blur：阴影模糊度  color：阴影颜色 |

练习

|  |
| --- |
| 01\_ex  定义div，内容随意，假文，有中文  1.设置字体系列  2.设置字号 大小  3.加粗并斜体  4.所有小写字母转为小型大写字母  5.两端对齐  6.文本有下划线  7.首行缩进30px  8.文本垂直居中 |

四.表格的样式

1.表格的常用样式属性table>tr>td

|  |
| --- |
| 1.table:之前学习过的样式，基本都可以使用  尺寸，边框，背景，字体，文本，内外边距  给table设置border,只设置最外面的大边框 |
| 2.td/th的样式  尺寸，边框，背景，字体，文本，内边距  外边距无效  vertical-align指定单元格数据的垂直对齐方式  取值：top/middle/bottom |

练习

|  |
| --- |
| 02\_ex  创建4\*4表格，宽高400px。内容随意  1.设置每个单元格的尺寸为100px\*100px  2.设置每个单元格边框1px solid #000  3.尝试给每个单元格设置上外边距20px.上内边距20px； |

表格是一种特殊的表现方式

|  |
| --- |
| 表格实际尺寸是根据内容数据的多少而决定的  单元格小，内容多，自动撑开  内容少，单元格就按照设置的尺寸来展示 |

2.表格特有的样式属性

①边框合并

|  |
| --- |
| border-collapse:  取值 ：1.separate 默认值，边框分离模式  2.collapse;边框合并模式 |

②边框边距

|  |
| --- |
| 向设置边框边距，必须保证边框是分离状态  border-collapse:separate ;  属性border-spacing:  取值：1.只有一个值 设置水平和垂直边框的 外边距  2.两个值 第一个值设置水平 第二个值设置垂直 |

③标题位置

|  |
| --- |
| caption-side:  取值：top 默认值  bottom |

④设置表格的显示规则

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 用来确定如何布局一张表格  table-layout  取值 1.auto 默认值 自动布局表格  列的尺寸实际使用内容决定的，内容比尺寸大，按照  内容大小显示。内容比尺寸小，按照尺寸显示  2.fixed 固定表格布局，列的尺寸以设置的为准  自动布局与固定布局   |  |  | | --- | --- | | 自动布局 | 固定布局 | | 单元格的大小自动适应内容 | 单元格大小取决于设置的值 | | 表格复杂时，加载速度较慢 | 任何情况下，都会加速加载表格(优点) | | 比较灵活(优点) | 固定布局不够灵活 | | 适用于不确定每列大小的，并且不复杂的表格 | 适用于能够确定每列尺寸大小的表格 | |

练习



五.定位(重点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

1.什么是定位

|  |
| --- |
| 改变元素在页面中的位置 |

2.定位的分类

|  |
| --- |
| 1.普通流定位  2.浮动定位  3.相对定位  4.绝对定位  5.固定定位 |

①普通流定位

|  |
| --- |
| 又称为默认文档流定位  1.每个元素在页面上都有自己的空间  2.每个元素都是从父元素的左上角开始渲染（显示）  3.块级元素按照从上到下逐个排列，每个元素单独成行  4.行内元素是多个元素在一行中显示，从左往右顺序排列，显示不下换行 |

②浮动定位

|  |
| --- |
| 让块级元素横向显示  float：  取值：left 让元素浮动后，在当前行停靠在父元素的左侧，或者挨着左侧已浮动元素  right让元素浮动后，在当前行停靠在父元素的右侧，或者挨着右侧已浮动元素  none 默认值，不浮动  浮动的特点  1.元素一旦浮动，脱离文档流(不占页面空间，后面未浮动元素会上前补位)  2.浮动元素会在当前行，停靠在父元素的左/右边，或者停靠在其他已浮动元素的边缘  3.当父元素横向显示不下所有浮动元素时，最后显示不下的，会自动换行  4.浮动用于解决，多个块级元素在同一行显示的问题 |

③浮动定位引发的特殊情况

|  |
| --- |
| 1.浮动元素占位的问题  当元素显示不下所有浮动元素时，最后显示不下的元素会换行  但是，已浮动元素会根据自己的浮动方向占据位置，  导致被挤下去的浮动元素，需要让开位置，在更下面的地方显示 |
| 2.元素一旦浮动，如果元素未定义宽度，那么元素浮动之后的宽度将以内容为准 |
| 3.元素一旦浮动，都会变成块级元素  允许设置尺寸，可以设置垂直外边距 |
| 4.文本，行内元素，是不会被浮动元素压在下面的。而是巧妙的避开，环绕着浮动元素显示 |

④清除浮动

|  |
| --- |
| 清除之前的浮动元素给自己带来的影响  由于元素浮动之后，会脱离文档流，会让后续不浮动的元素上前补位.  如果后续元素不想上前补位，需要对此元素设置清除浮动  clear:  取值：1.left 清除左浮动对我的影响  2.right 清除右浮动对我的影响  3.both 清除左右浮动对我的影响  4.none 不清除影响 |

⑤高度坍塌

|  |
| --- |
| 什么叫高度坍塌  1.父元素不写高，靠子元素撑起高度  2.所有子元素都浮动  那么所有子元素都脱离文档流，父元素认为自己内部没有元素了  所以父元素就没有高度了  解决方案：  1.父元素也浮动，弊端，影响父元素后面的非浮动元素  2.直接给父元素写高度 弊端，不是每次都能知道具体高是多少  3.overflow：hidden 弊端，会让真正要溢出不能显示  4.在父元素中追加一个块级元素，这个块级元素，没有内容，不写高。只写clear:both。就可以让父元素在文档流中找到内容的高度  解决，高度坍塌问题 |

作业

1.掌握css编写思路



|  |
| --- |
| 1.宽高，大体位置  2.边框，背景  3.文字所有  4.微调 |

2.今天关于table和浮动的所有demo

3.回家安装WebStorm10.zip，并且保证注册成功

# CSSDAY-05

day12

一.元素的显示方式

|  |
| --- |
| display  取值： 1.block 让元素的表现和块级一致  2.inline 与行内元素一致  3.inline-block 与行内块元素一致  4.table 与table一致  5.none 不显示元素，隐藏  块级：独占一行，可以设置尺寸，上下外边距有效  行内：共用一行，尺寸无效，上下外边距无效  行内块：共用一行，可以设置尺寸，上下外边距有效  table：独占一行，可以设置尺寸，尺寸以内容为主 |

二.显示效果

|  |
| --- |
| visibility  取值：1.visible 默认值，可见  2.hidden 隐藏不可见  问题：visibility:hidden和display:none区别  visibility:hidden；隐藏，元素不脱离文档流，在当前页面不可见，但是占据位置  display:none；隐藏，元素脱离文档流，隐藏后不占位置，后面元素上前补位 |

三.透明度

|  |
| --- |
| opacity  取值：0~1 值越小越透明  问题：opacity和rgba的区别  rgba只会改变当前颜色的透明度  opacity，元素内部只要元素相关的颜色都会跟着透明 |

四.垂直对齐方式

|  |
| --- |
| vertical-align  使用场合  1.表格中 td/th  取值 top/middle/bottom  2.img与文字的排版  改变的是img与前后文本的对齐方式  取值 top/middle/bottom/baseline基线  通常，会将所有的图片与文字的对齐方式，改为非基线对齐的方式 |

五.光标的设置

|  |
| --- |
| cursor  取值：1.default 箭头  2.pointer 小手  3.crosshair 十字  4.text 文本输入的I  5.wait 等待  6.help 帮助 |

六.列表的样式

|  |
| --- |
| ul的样式  1.列表项标识  list-style-type:  取值 none/disc/circle/square  2.列表项标识，设置为图片(图片要小)  list-style-image:url(路径);  3.列表项标识的位置  list-style-position:inside/outside(默认值)  设置列表项是在li的内部还是外部  4.简写方式  list-style:type image position;  项目中最常用的写法list-style:none;  去掉列表标识 |

七.定位---相对，绝对，固定定位（\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*）

|  |
| --- |
| position  取值：static 默认，静态(默认文档流定位)  relative 相对定位  absolute 绝对定位  fixed 固定定位  当一个元素，被position修饰，并且取值为relative/absolute/fixed其中一种时，这个元素被称为已定位元素  已定位元素，解锁了4个偏移属性  top + 往下 - 往上  right + 往左 -往右  bottom + 往上 -往下  left +往右 -往左  偏移属性，定义了元素距离某一个方向移动了多少距离 |

1.相对位移

|  |
| --- |
| 相对定位，相对自己原来位置偏移某个距离  position:relative;配合4个偏移属性使用  特点：1.不脱离文档流。后面元素不会上前补位  2.相对定位，如果不写偏移量，元素效果与没写定位是一样的.  使用场合  1.自身位置的微调  2.作为绝对定位的祖先级元素 |

2.绝对定位

|  |
| --- |
| position:absolute；配合偏移量使用  特点：  1.绝对定位，脱离文档流，元素不占页面空间，后面元素上前补位。  2.绝对定位的元素，会相对于  "离自己最近的""已定位的""祖先元素"实现位置的初始化  如果没有 "已定位的""祖先元素"，相对于body实现位置初始化  3.绝对定位元素会变成块级元素  4.绝对定位元素，如果不写宽，定义之后，宽靠内容撑开 |

3.固定定位

|  |
| --- |
| position:fixed;配合偏移量使用  将元素固定在页面的某个位置，不会随着滚动条发生位移变化  特点：1.脱离文档流  2.元素变为块级元素  3.不写宽的会被内容撑开  4.相对body做位置的初始化 |

4.堆叠顺序

|  |
| --- |
| 特点：  1.默认堆叠顺序，后定位的元素，堆叠顺序高  2.定位的脱离文档流，和浮动的脱离文档流，不是一个体系  3.使用z-index设置堆叠顺序  z-index:无单位数字 一般情况 1~1000  4.堆叠顺序，只对已定位元素有效  5.堆叠顺序，对父子级无效。子元素永远在父元素上面显示 |

CSS3 CORE

一.复杂选择器

1.兄弟选择器

|  |
| --- |
| 兄弟元素：具有相同父元素的平级元素  兄弟选择器，只能找弟弟，不能找哥哥 |

①相邻兄弟选择器

|  |
| --- |
| 获取紧紧挨在某元素后面的兄弟元素  选择器1+选择器2{} |

②通用兄弟选择器

|  |
| --- |
| 获取某元素后，所有符合要求的兄弟元素  选择器1~选择器2{} |

练习

|  |
| --- |
| <h3>小白兔</h3>  <div>小了白了兔</div>  <div>白了又了白</div>  <div>爱了吃了萝了卜了和了青了菜</div>  <div>蹦了蹦了跳了跳了真了可了爱</div>  标题与正文之间间距30px  正文每行之间间距10px |

2.属性选择器

|  |
| --- |
| id class style title name value width.......  允许通过元素所附带的属性，及其值来匹配页面元素，很精准 |

①[attr] attr代表任意属性

|  |
| --- |
| 匹配页面中所有带attr这个属性的元素  [id]{} [id][title]{} |

②elem[attr]

|  |
| --- |
| 匹配页面中所有带attr属性的elem元素  p[title]{} p[id][title][class].....{} |

③elem[attr=value]

|  |
| --- |
| 匹配页面中带attr属性，并且值为value的elem元素  div[title="woyouyeye"][class="c"]{} |

④模糊属性值匹配

|  |
| --- |
| [attr^=value]{} 匹配attr的值以value开头的元素  [attr$=value]{} 匹配attr的值以value结尾的元素  [attr\*=value]{} 匹配属性值中，有value的元素  [attr~=value]{} 匹配属性值中，有value这个独立单词的元素 |

3.伪类选择器

|  |
| --- |
| :link :visited :hover :active :focus 已经学过的伪类 |

①目标伪类(锚点)

|  |
| --- |
| 让被激活的锚点，应用样式  选择器：target{} |

练习

|  |
| --- |
| 09\_ex  点击菜名，显示图片  菜名放在ul>li>a  海鲜面 四季春饼 八大碗 |

作业：1.完成学子首页1楼



2.完成学子商城轮播图的静态页面



# CSSDAY-06

day13

一.复杂选择器

3.伪类选择器

①目标伪类target

②结构伪类

|  |
| --- |
| elem:first-child{} 代表两个条件  匹配elem的父元素的第一个儿子（elem的大哥）  这个大哥必须是elem元素  elem:last-child{}  匹配elem的父元素的最后一个儿子(elem的小弟)  这个最小的弟弟必须是elem元素  elem:nth-child(n)  匹配elem的父元素的第n个儿子(n从1开始)  这个儿子也必须是elem元素 |

练习

|  |
| --- |
| 01\_ex 4\*4的表格 200px\*200px. 有边框，内容随意  需求：表格第一行背景为红色  表格最后一行最后一列的背景为蓝色  表格第三行第二列背景为黄色 |

③匹配空元素

|  |
| --- |
| :empty{}  空元素  空元素：没有文本，没有空格，没有其他子元素的元素 |

④:only-child

|  |
| --- |
| 匹配，当前元素是其父元素的唯一的子元素 |

⑤否定伪类

|  |
| --- |
| :not(selector)  /\*除了第一个a标签，其它a的字体都变成黄色 \*/  a:not(:first-child){color:#ff0;}  /\*除了第三个a标签，其它a的字体都变成蓝\*/  a:not(:nth-child(3)){color:#f0f;} |

4.伪元素选择器

①内容伪元素

|  |
| --- |
| 1::first-letter{ } 或 :first-letter{}  匹配首字符的样式  2::first-line{ } 或 :first-line{}  匹配首行的样式  当首行和首字符冲突的时候，以首字符为准  3.::selection{} 必须是两个 ::  匹配选中部分的文字样式  注意：只能改变字体颜色和背景颜色 |

②伪元素选择器，内容生成

|  |
| --- |
| 使用css，添加httml元素，称之为伪元素内容生成  :before 或者 ::before  匹配到某个元素的内容区域的最前面，  添加一个内容，content：添加的文本或者图片  display：是设置这个添加的内容的显示规则  ::after 或者 :after  匹配到某个元素的内容区域的最后面，添加一个内容  注意：content中，只能添加文本或者图片  我们可以理解为在内容区域中，最前或者最后面，添加了一个元素。这个元素的显示方式，由display来设定 |

③伪元素内容添加，可以解决的问题

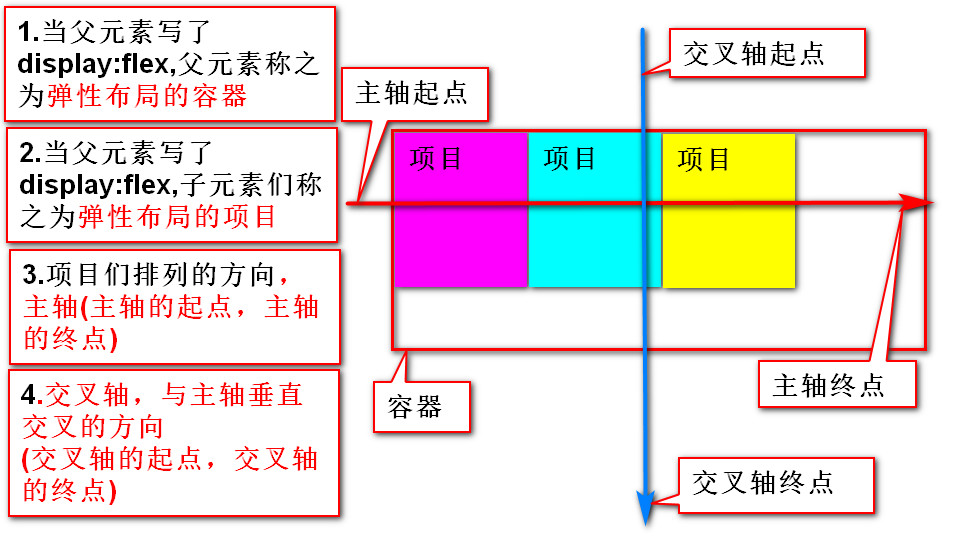
|  |
| --- |
| 1.外边距溢出  #parent::before{  content:"";  display:table;  }  2.思考，伪元素内容添加，解决高度坍塌？  #d1:after{  content:"";  display:block;  clear:both;  } |

二.弹性布局（重点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*）

1.什么是弹性布局

|  |
| --- |
| 是一种布局方式  主要解决某个元素中子元素的布局方式  为布局提供很大灵活性 |

2.弹性布局相关的概念和名词解释



|  |
| --- |
| 1.容器  要发生弹性布局的子元素，他们的父元素称之为容器  容器要设置属性display:flex;  2.项目  要发生弹性布局的子元素们，称之为项目  就是设置了display:flex那个元素的，子元素们  3.主轴  项目们在容器中排列的方向，就是主轴  如果项目横向排列，x轴就是主轴  如果项目纵向排列，y轴就是主轴  项目们的排列顺序，靠主轴的起点和主轴的终点来定义  4.交叉轴  与主轴垂直相交的一条轴，叫做交叉轴  项目们在交叉轴上的对齐方式，是交叉轴的起点和终点 |

3.语法

|  |
| --- |
| display，写在父元素中  取值：1.flex 将块级元素设置为容器  2.inline-flex 将行内元素设置为容器  特点：  1.弹性项目，默认x是主轴，主轴起点在左侧  (块级元素横向排列的第二个解决方案)  2.项目的float/clear/text-align/vertical-align属性失效  3.每个项目可以自由的设置尺寸 |

4.容器的属性

①主轴的方向

|  |
| --- |
| flex-direction:  取值： row默认值，主轴是x轴，主轴起点是左端  row-reverse,主轴是x轴，主轴起点是右端  column 主轴是y轴，主轴起点是顶端  column-reverse 主轴是y轴，主轴起点是底部 |

②设置项目换行

|  |
| --- |
| flex-wrap:  取值 nowrap 默认值，容器空间不够，也不换行，项目自动缩小  wrap 空间不够就换行  wrap-reverse 换行并反转 |

③主轴方向，项目换行的缩写

|  |
| --- |
| flex-flow:  取值 direction wrap |

④定义项目在主轴上的对齐方式

|  |
| --- |
| justify-content:  取值 1.flex-start 默认值，以主轴起点对齐  2.flex-end 以主轴终点对齐  3.center 在主轴上居中对齐  4.space-between 主轴两端对齐，两端无空白  5.space-around 每个间距大小相同 |

⑤项目在交叉轴上的对齐

|  |
| --- |
| align-items:  取值1.flex-start 交叉轴起点对齐  2.flex-end 交叉轴终点对齐  3.center 交叉轴居中对齐  4.baseline 交叉轴基线对齐  5.stretch 前提，项目不写高，占满交叉轴上所有的空间 |

练习

|  |
| --- |
| 06\_ex,ul>li 配合弹性布局完成 |

5.项目的属性

①order

|  |
| --- |
| 定义项目的排列顺序  取值，无单位整数，值越小，越靠近主轴起点 |

②flex-grow

|  |
| --- |
| 定义项目的放大比例  如果容器有足够大的剩余空间，项目将按比例方大  取值：无单位的数字 默认值0，不放大 |

③flex-shrink

|  |
| --- |
| 定义项目缩小的比例，容器空间不够时，项目该如何缩小  取值，无单位数字 默认值1. 取值越大，缩小比例越大。0，不缩 |

④align-self

|  |
| --- |
| 设置此项目在交叉轴上的对齐方式，不影响其它项目  取值：flex-start  flex-end  center  baseline  stretch  auto 使用容器定义的align-items的值 |

三.CSS hack

|  |
| --- |
| div{  background:-webkit-linear-gradient(....);  background:-o-linear-gradient(....);  background:-ms-linear-gradient(....);  background:-moz-linear-gradient(....);  }  由于不同的浏览器对css的解析认知不同，会导致同一份css在不同浏览器中生成页面效果不同  面对这种情况，开发人员需要针对不同浏览器写不同的css  这个行为，就叫做css hack  tmooc有css hack相关拓展视频 |

四.转换（重点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*）

1.什么是转换

|  |
| --- |
| 改变元素在页面中的位置，大小，角度，以及形状  2D转换。只在x轴和y轴发生的转换效果  3D转换。增加了z轴的转换效果。3D是模拟出来的 |

2.转换的属性

|  |
| --- |
| transform: 取值 1.none 默认值，无任何转换效果  2.transform-function 转换函数  表示1个或者多个转换函数  如果是多个转换函数，每个函数之间用空格分开  transform:translate(400px) rotate(90deg);  学习转换，就是学习转换函数 |

作业1.使用弹性布局完成学子商场首页1F



作业2：复杂选择器，弹性布局的所有属性，单词和效果

作业3：确保webstrom10 确保可以正确使用(注册码的问题)

# BootStrap-01

day15

准备webstrom10,见视频

一.响应式布局(css3)

1.什么是响应式网页

|  |
| --- |
| Responsive web page 响应式/自适应网页  可以根据浏览网页的设备不同(pc,pad,phone)  而自动改变布局，文字，图片效果。  不会影响用户体验 |

2.响应式网页必须保证几件事情

|  |
| --- |
| 1.布局，不能固定元素宽度.必须是流式布局(默认文档流+浮动)  2.文字和图片大小随着容器的大小而改变  3.媒体查询技术(css3的技术) |

3.如何测试响应式网页

|  |
| --- |
| 1.使用真实设备测试  好处：真是可靠  缺点：成本高，测试任务巨大  2.使用第三方测试软件  好处：不需要太多真实设备，测试方便  坏处：测试效果有限，有待进一步验证  3.使用chrome等浏览器自带的模拟软件  好处：简单方便  坏处：测试效果十分有限，需要进一步验证 |

4.编写响应式布局(重点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

①手机适配,设置视口

|  |
| --- |
| 只有移动端设备，需要设置视口  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0">  viewport 视口  width=device-width 视口宽度等于设备宽度  initial-scale=1.0 设置视口宽度能否缩放 1.0不能缩放  maximum-scale=1.0 允许视口缩放的最大倍率 1.0不缩放  user-scalable=0 是否允许用户手动缩放 0不能  最简洁的写法  <meta name="viewport" content="width=device-width, initial-scale=1"> |

②所有内容/文字/图片都尽量使用相对尺寸，不要使用绝对数字

③流式布局+弹性布局+媒体查询，组成响应式布局

④媒体查询 CSS3 Media Query

|  |
| --- |
| 媒体查询是写响应式必须具备的技术  Media：媒体，不同的用于浏览网页的设备  设备：screen（pc/pad/phone）  TV  print  Media Query:可以自动的根据当前浏览设备不同(尺寸，方向，解析度。。。)，有选择性的，执行一部分代码，忽略其他代码 |

二.BootStrap

|  |
| --- |
| boot封装了很多常用样式，以类名的形式让我们调用  但是，真正项目，很多样式还是需要我们手动开发和媒体查询  学会查boot文档的能力  boot内容分为5部分  1.如何使用bootstrap  2.全局样式  3.组件  4.js插件  5.sass  6.boot项目 |

三.如何使用bootstrap

|  |
| --- |
| 导入boot文件  *<!--设置视口-->* <**meta name="viewport" content="width=device-width,initial-scale=1"**/> *<!-- 导入css-->* <**link rel="stylesheet" href="../css/bootstrap.css"**/> *<!-- 导入3个js,有顺序要求-->* <**script src="../js/jquery.min.js"**></**script**> <**script src="../js/popper.min.js"**></**script**> <**script src="../js/bootstrap.min.js"**></**script**>  制作boot模板  file--->new--->edit Template file |

四.全局样式

|  |
| --- |
| .container 定宽容器，在每种尺寸下，都定义了最大宽度  .container-fluid 变宽容器，每种尺寸都是100%  boot支持4种屏幕尺寸 xl--lg--md--sm  boot中rem，默认一个rem是16px |

1.按钮相关的class

|  |
| --- |
| 基本类 .btn 所有的按钮，都需要添加这个类  按钮颜色  btn-success 绿色  btn-danger 红色  btn-warning 黄色  btn-info 青色  btn-primary 蓝色  btn-secondary 灰色  btn-dark 深色  btn-light 浅色  btn-link 链接  不同边框的按钮  btn-outline-danger/success/warning......  不同大小的按钮  btn-lg  btn-sm  btn-block |

2.图片相关

|  |
| --- |
| rounded 圆角 0.25rem  rounded-circle 圆形  img-thumbnail 缩略图，有一圈白色的边(padding)  img-fluid 响应式图片，图片可以缩放，但是不会超过原始大小 |

3.文本相关的class

|  |
| --- |
| .h1~.h6 字号大小，同h1标签到h6标签  text-danger/primary.......文本颜色  text-uppercase/lowercase/capitalize首字母大写  font-weight-bold/light/normal 字体 粗细正常  text-left/right/center/justify 文本对齐  text-\*-left/right/center 响应式 \*:sm/md/lg/xl |

4.列表相关class

|  |
| --- |
| list-unstyled 取点，左内边距清0  ul.list-group>li.list-group-item 创建列表组和列表项  激活项 .list-group-item.active 两个类都引用  禁用项 .list-group-item.disabled 两个类都引用  列表项颜色 list-group-item-danger/warning...... |

5.表格相关的样式

|  |
| --- |
| .table 基本量，对table本身和table的后代布局  .table-bordered 一个有边框的边框，后代td、th也添加边框  .table-danger/warning....表格颜色  .table-striped 隔行变色，奇数行变色  .table-hover 带鼠标悬停效果  .table-responsive-xl/lg/md/sm 响应式表格，写在table的父元素上 |

6.辅助类

①边框

|  |
| --- |
| 基本类  border/border-0/ 定义边框/取消边框  border-left/top/bottom/right 单边定义  border-left-0 单边的取消  border-danger/warning/info.... 边框颜色 |

②浮动

|  |
| --- |
| float-left/right/none  float-sm/md/lg/xl-left响应式的浮动  clearfix 解决高度坍塌，写在父元素 |

③显示

|  |
| --- |
| visible 显示  invisible 隐藏 |

④背景颜色

|  |
| --- |
| bg-danger/warning/success/primary.... |

⑤圆角

|  |
| --- |
| rounded/rounded-0 设置圆角/圆角清0  一个方向的圆角设置  rounded-top/right/bottom/left  rounded-circle 50% |

⑥内外边距

|  |
| --- |
| m-n mt/mr/mb/ml-n mx-n左右外边距 my-n上下  p-n pt/pr/pb/pl-n px-n 左右内边距 py-n上下  n:0---取消,  1--0.25rem,  2---0.5rem  3--1rem，  4---1.5rem,  5---3rem  上面所有的都有响应式  m-\*-n  m:m/mt/mb/ml/mr/mx/my \*:xl/lg/md/sm n:0~5  p-\*-n  p:p/pt/pb/pl/pr/px/py \*:xl/lg/md/sm n:0~5 |

五.组件

六.js插件

七.sass/scss

八.boot项目

作业

1.使用媒体查询完成布局



2.完成下图



# Bootstrap-02

day16

四.全局css

6.辅助类

⑦尺寸

|  |
| --- |
| w-25/50/75/100  h-25/50/75/100  如果需要其他尺寸，需要手写css  mw-100 mh-100 |

7.栅格布局 （重点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*）

①web页面中的布局方式，一般有三种

|  |  |  |
| --- | --- | --- |
| table布局 | div+css | Boot的栅格布局 |
| 简单，易控制 | 语义正确 渲染效率高 | 简单，易控制。语义正确，渲染效率高，支持响应式 |
| 语言错误 渲染效率低 | 控制起来很麻烦 | 不适用于复杂页面 |

②栅格的使用

|  |
| --- |
| 1.栅格必须在container或者container-fluid中  2.每一行都是一个row(弹性，主轴是x轴，可以换行，  左右-15px外边距)  3.每一个col自带左右15内边距 |

③响应式栅格布局

|  |
| --- |
| col-\*-n  \*:lg/md/sm/xl  n:1~12  在不同的屏幕下，执行不同的列数  <**div class="row"**>  <**div class="col-lg-2 col-md-3 col-sm-12"**>111</**div**>  <**div class="col-lg-8 col-md-9 col-sm-12"**>222</**div**>  <**div class="col-lg-2 d-lg-block d-md-none col-sm-12"**>333</**div**> </**div**> |

④不同屏幕尺寸，向上兼容

|  |
| --- |
| 只写小屏，在小屏，中屏，大屏，超大屏，都会应用  只写中屏，在中屏，大屏，超大屏，都会应用  只写大屏，大屏，超大屏，都会应用  只写超大屏，只在超大屏应用 |

⑤使用.col这个类

|  |
| --- |
| 使用.col,不添加数字，自动平局分配占地宽度  每一个col平局分配空间，col的个数可以超过12个  使用场合：  当一行内，元素需要横向现实，元素个数不能被12整除  每个元素占地空间一样 |

⑥列偏移

|  |
| --- |
| offset-\*-n 源码就是在操作左外边距 margin-left  \*:xl/lg/md/sm  n:0~11 |

⑦栅格嵌套

|  |
| --- |
| 在一个.col中再写一个row |

五.弹性布局

|  |
| --- |
| d-block/inline/inline-block/none display的取值  d-\*-block.....响应式 \*：xl/lg/md/sm  想使用弹性布局，需要在父元素上添加d-flex/d-inline-flex  创建弹性容器 |

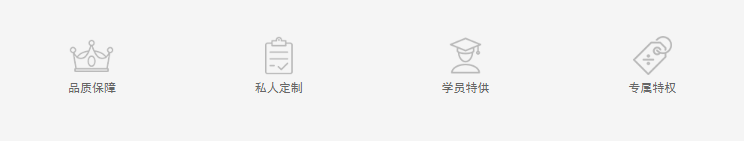
1.主轴方向

|  |
| --- |
| flex-row/column/row-reverse/column-reverse  响应式设置  flex-\*-row/column/row-reverse/column-reverse  \*:xl/lg/md/sm |

2.主轴对齐方式

|  |
| --- |
| justify-content-start/center/end/between/around  响应式设置  justify-content-\*-start/center/end/between/around  \*：xl/lg/md/sm |

练习 ul>li boot完成



六.表单相关的class

1.表单元素的排列放向

|  |
| --- |
| form-group 堆叠表单，内部控件垂直显示  form-inline 内联表单，内部控件横向显示(弹性，x轴) |

2.表单控件元素样式

|  |
| --- |
| form-control 对于text,password 基本类  块级，w100 字体 背景颜色 边框 圆角 过渡  col-form-label 设置文本框上下内边距  col-form-label-lg 上下内边距比较大  col-form-label-sm 上下内边距比较小  checkbox的基本类  form-check-input,绝对定位  要配合父元素的 form-check使用  form-text 块级，有上外边距 15:06~15:21休息 |

练习：



|  |
| --- |
| form-inline可以代替row作为栅格行  form-inline没有左右-15px的外边距 |

七.组件

|  |
| --- |
| 网页中有一些复杂的特殊效果，之前必须使用js,dom,jq  boot为我们封装好了很多常用的组件，我们可以直接使用。  方便，快捷。但是如果需要个性化的处理，比较繁琐。  boot中，大量使用元素的自定义属性，来调用方法 |

1.下拉菜单

|  |
| --- |
| 基本结构  <**div class="dropdown"**><**button class=" dropdown-toggle"**  **data-toggle="dropdown"**>下拉菜单</**button**>  <**ul class="dropdown-menu"**> </**ul**> </**div**>  div.dropdown 父级，相对定位  button.dropdown-toggle 画向下小三角  ul.dropdown-menu display:none  **data-toggle="dropdown"** 自定义属性的作用  1.当button被点击的时候激活事件  2.自定义属性的值，决定激活哪一个事件 |

2.按钮组

|  |
| --- |
| <div class="btn-group">  <button></button>  .....  </div>  外层div的class有以下几个值  btn-group 横向按钮组  btn-group-vertical 纵向按钮组  可以添加 btn-group-lg/sm 调整按钮组的大小 |

3.信息提示框

|  |
| --- |
| <div class="alert alert-danger alert-dismissible">  <span class="close" data-dismiss="alert">  &times;</span>  提示的信息  </div>  alert 基本类  alert-danger/warning/.....提示框颜色  alert-dismissible 为了修饰内部span.close  .close 右浮动  data-dismiss="alert" 自定义属性，事件使用alert的方式执行 |

4.导航

①水平导航

|  |
| --- |
| 基本结构 ul.nav>li.nav-item>a.nav-link  <ul class="nav nav-justified">  <li class="nav-item">  <a class="nav-link"></a>  </li>  </ul>  ul.nav 定义了弹性布局，主轴为x  那么之前学习的所有的弹性布局的样式都可以用  .nav-justified>.nav-item 要同时设置，让元素等宽显示  a.nav-link 块级，设置了hover，focus等样式 |

②选项卡导航

|  |
| --- |
| 基本结构  上层是选项卡  ul.nav.nav-tab>li.nav-item>a.nav-link  下层内容  div.tab-content>div.pane  ul.nav 变成选项卡添加类 nav-tab  nav-tab设置自己的孩子 nav-item nav-link  a标签添加自定义属性 data-toggle="tab"  以tab的方式激活事件  绑定目标元素 a的href属性指向下面内容的id值  div.tab-content>div.tab-pane配合使用  让div.tab-pane都是是display:none  然后div.tab-pane要写id，让a标签绑定 |

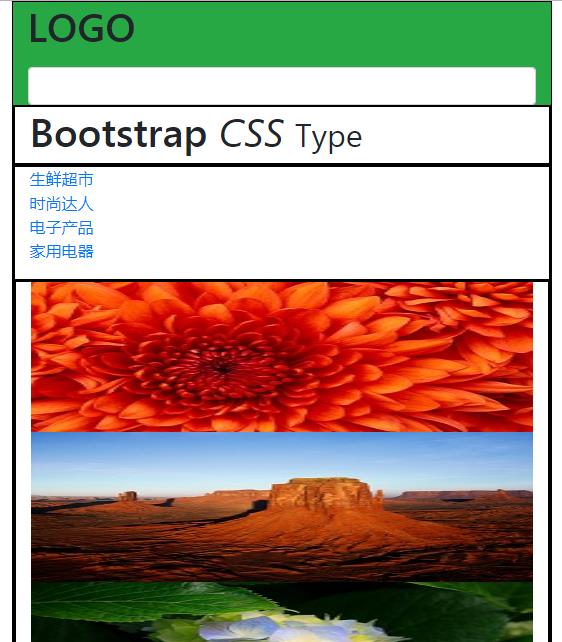
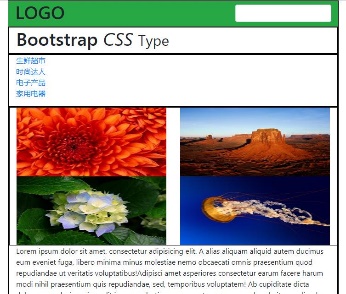
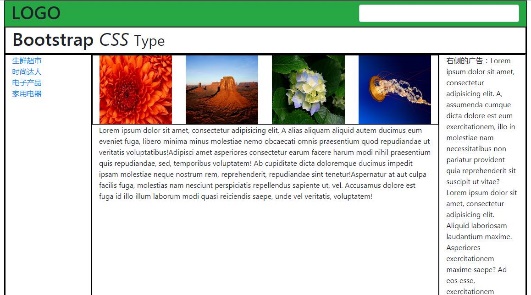
|  |
| --- |
| 一个事件激活，关注几件事情  1.激活这个事件的元素是啥 <a>  2.这个事件以什么方式激活 tab  3.事件激活后，改变的目标元素是谁 |

作业1.提高题，完成学子首页1L布局，使用栅格实现6个灰色框

允许手写宽高样式

坑：div灰块之间的空白怎么解决

作业2.使用栅格完成



3.组件所有demo都敲一遍

# Bootstrap-03

day17

七.组件

4.导航

③胶囊导航

|  |
| --- |
| <**ul class="nav nav-pills"**>  <**li class="nav-item"**>  <**a data-toggle="pill" href="#tab1" class="nav-link"**>口红</**a**>  </**li**>  </**ul**> <**div class="tab-content"**>  <**div id="tab1" class="tab-pane"**>  YSL,Dior,chanel,故宫  </**div**> </**div**>  ul的nav-tabs改成nav-pills  a的data-toggle="tab" 改成data-toggle="pill"  其他与选项卡导航一样 |

5.导航栏

|  |
| --- |
| <**h1**>导航栏</**h1**> <**div class="navbar navbar-expand-lg"**>  <**ul class="navbar-nav"**>  <**li class="nav-item"**><**a class="nav-link" href=""**>长的丑</**a**></**li**>  </**ul**> </**div**>  最外层div.navbar.navbar-expand-xl/lg/md/sm  .navbar-expand-xl/lg/md/sm作用，作为子元素ul.navbar-nav的选择器条件，让ul.navbar-nav的主轴方向从y轴变成x轴(ul.navbar-nav原本主轴方向是y轴)    响应式的导航栏，以.navbar-expand-lg为例  超大屏，大屏，li横向显示  中屏，小屏，li纵向显示  div.navbar-expand-\* 配合 ul.navbar-nav可以让导航栏在不同屏幕下，横向或者纵向显示 |

6.折叠效果

|  |
| --- |
| <**button data-target="#demo" data-toggle="collapse" class="btn btn-info"**>折叠</**button**> <**div id="demo" class="collapse"**>Lorem  折叠，需要折叠内容添加.collapse--->display:none  折叠事件 data-toggle="collapse"  折叠目标  如果使用a,就可以使用href=""  如果使用button,则使用data-target="" |

7.卡片

|  |
| --- |
| <**div class="card"**>  <**div class="card-header"**></**div**>  <**div class="card-body"**></**div**>  <**div class="card-footer"**></**div**> </**div**> |

8.卡片+折叠=手风琴

|  |
| --- |
| <div id="parent">  <div class="card">  <div class="card-header">  <a data-toggle="collapse"  href="#one" class="card-link">折叠</a>  </div>  <div id="one" data-parent="#parent"  class="collapse">  <div class="card-body">内容</div>  </div>  </div>  </div>  总结：  1. .collapse不能和.card-body在同一个标签上  不然会发生折叠/显示的卡顿  解决方案 div.collapse>div.card-body  2.多个折叠部分，可以同时打开，而不是开启一个，其他的关闭  解决方案，在所有的card外部，添加一个父元素div#parent  在所有的div.collapse上添加自定义属性  data-parent="#parent"  这样就可以保证，在外层div中，只存在一个折叠区域被打开 |

练习1：



练习2：响应式导航栏，中屏，小屏 垂直排列



9.折叠导航栏

|  |
| --- |
| 1.div.navbar-expand-sm和.anvabar-toggler的关系和作用    2.在外层div中.navbar-dark/light的作用  这个类对本div没有任何修饰  修饰的是后代元素的字体颜色  3..navbar-collapse的作用  .navbar-expand-sm .navbar-collapse配合  让ul的父级div在sm/md/lg/xl下显示  在xs下，ul的父级div执行.collapse的样式，隐藏  4.小总结 .navbar-expand-sm决定了几件事？  1.按钮在sm以上隐藏，在sm以下显示  2.ul在sm以上横向显示，在sm以下垂直显示  3.折叠的div在sm以上显示，在sm以下隐藏 |

10.媒体对象

|  |
| --- |
| boot提供了很多类，来处理媒体相关的对象  图片 影音  <div class="media">  <img>  <div class="media-body"><div>  </div> |

11.焦点轮播图

|  |
| --- |
| 1.轮播图结构  div.carousel data-ride="carousel" 相对定位  >div.carousel-inner w100 溢出隐藏  >div.carousel-item display:none  >img  2.左右箭头  a.carousel-control-prev/next 宽度，背景，位置  >span. carousel-control-prev/next-icon 左右箭头  需要重写.carousel-control-prev/next中的宽高，位置  .**carousel-control-prev**, .**carousel-control-next**{  **width**:4%;**height**: 20%;  **background**:**#aaa**;  **border-radius**: 0.25**rem**;  **top**:40%; }  a标签要添加自定义属性 data-slide="prev/next"  事件的目标是最外层div#demo href="#demo"  3.轮播指示器  ul.carousel-indicators 绝对定位，横向弹性  >li 宽高，背景色，外边距  我们需要重写li的样式  .**carousel-indicators li**{  **width**: 0.75**rem**;**height**:0.75**rem**;  **background-color**:**#fff**;  **border-radius**: 50%;  **margin**:0 4**px**; }  如果让li有被选中的状态，需要第一个li添加.active  我们需要重写.active .**carousel-indicators** .**active**{  **background-color**:**#0aa1ed**; }  点击li激活事件  data-slide-to="图片下标" 下标从0开始  事件改变的目标  data-target="#demo" 整个轮播最外层div的id |

12.模态框

|  |
| --- |
| 覆盖在父窗体上的一个子窗体，用于与用户做一些互动  div.modal>div.modal-dialog>div.modal-content  >div.modal-header+div.modal-body+div.modal-footer  关闭的事件  button data-dismiss="modal" |

八.其它组件

1.徽章

|  |
| --- |
| 跟小按钮十分类似  基本类 badge  颜色 badge-danger/warning/info...........  徽章样式 badge-pill |

2.巨幕

|  |
| --- |
| 巨大的内边距，有背景颜色，有圆角  div.jumbotron |

3.分页条

|  |
| --- |
| ul.pagination>li.page-item>a.page-link  li的修饰类 .active激活  .disabled禁用 |

作业

1.重新完成手风琴

2.重新完成焦点轮播图

3.折叠导航栏



作业：

1.完成学子一楼响应式布局，使用栅格

注意外边距，不同的对齐

# Bootstrap-04

day18

八.其他组件

4.面包屑导航

|  |
| --- |
| 常用于有路径关系的导航  ul.breadcrumb>li.breadcrumb-item  每个li中间连接的符号，需要重写  .**breadcrumb-item** + .**breadcrumb-item**::**before**{  **content**: **'>'**; } |

5.进度条

①普通进度条

|  |
| --- |
| div.progress>div.progress-bar.w-50.bg-danger  父div是进度条的槽  子div是进度条，子div的宽度，是进度条的进度  子div背景，是进度条的颜色 |

②带条纹的进度条

|  |
| --- |
| 在子div添加.**progress-bar-striped** |

③带动画的条纹进度条

|  |
| --- |
| <**div class="progress-bar w-100 bg-secondary progress-bar-striped progress-bar-animated"**></**div**> |

BOOT定制:scss

|  |
| --- |
| **bootStrap最重要：响应式+栅格布局+scss** |

1.CSS有很多缺点

|  |
| --- |
| 语法不够强大，没有变量，没有合理的样式复用机制  导致难以维护  我要使用动态样式语言，赋予css新的特性  提高样式语言的可维护性  常用的动态样式语言  1.SCSS/SASS （scss兼容sass,scss更接近css语法）  2.stylus  3.less |

十.SCSS详解

1.什么是scss

|  |
| --- |
| scss是一款强化css的辅助工具  它和css语法很像。  它在css的语法基础上，添加了变量，嵌套，混合，导入，函数等高级功能。  这些拓展命令让scss更加强大和优雅  浏览器不能直接解析scss文件，需要在想项目中把scss转义成css文件，让浏览器解析  scss可以让css开发更高效 |

2.scss的安装使用

|  |
| --- |
| scss在服务器端使用  nodejs v8.11以上，才可以使用scss  1.在线安装scss  在cmd中，输入命令npm install -g node-sass  2.无网络安装  找到sass的4个文件，  找到nodejs的安装路径,把sass4个文件放入node文件夹  在cmd中使用node-sass -v 验证版本 |

3.SCSS文件转换成css文件

①单文件的转换

|  |
| --- |
| 1.创建scss/01.scss文件  2.在项目路径下，打开黑窗口  3.输入下面命令 node-sass scss文件路径 css文件路径 |

②多文件转换 文件夹--->文件夹

|  |
| --- |
| node-sass scss文件夹 -o css文件夹 |

③单文件监听转换

|  |
| --- |
| node-sass -w scss/01.scss css/01.css |

④多文件监听，监听一个文件夹中所有文件

|  |
| --- |
| node-sass -w scss文件夹 -o css文件夹 |

十一.SCSS基本语法

1.变量

|  |
| --- |
| 使用$声明变量，变量名可以包含-，\_  命名规则基本与css选择器相同，尽量做到见名知意  变量声明在{}外，整个scss文件都可以使用。  变量声明在{}内，只有当前{}内可以使用  ！default规则，如果此变量在之前已经声明赋值了，那么使用之前的值。如果之前没有声明赋值，使用现在值 |

2.嵌套

|  |
| --- |
| 1.选择器的嵌套  **#content**{  **width**:**$my\_width**;  **div**.**top**{  **margin**:**$my\_width**;  **h1**{**font-size**:46**px**;}  **p**{**padding**:12**px**;}  }  **div**.**bottom**{  **border**: **$my\_border**;} }  自动转换成css的后代选择器  **#content** { **width**: 521**px**; } **#content div**.**top** { **margin**: 521**px**; } **#content div**.**top h1** {**font-size**: 46**px**; } **#content div**.**top p** {**padding**: 12**px**; } **#content div**.**bottom** { **border**: 1**px solid #00f**; } |
| 2.伪类的嵌套  需要在伪类选择器之前添加&，如果不添加，会生成一个空格导致伪类失效  a{width:21px;  &:hover{width:20px}  } |

练习 使用scss完成

|  |
| --- |
| $myblue:#0aa1ed;  $fontcolor:#fff;  $myheight:2.5rem;  $fontsize:0.8rem; |



|  |
| --- |
| 3.属性的嵌套  **div**{  **border**:{**style**:**solid**;**width**:1**px**;**color**:**#fff**;}; } |
| 4.群组选择器的嵌套  **nav**,**div**,**header**,**footer**{  **a**{**width**:100**px**;} } |

练习

|  |
| --- |
| 一个div#main.200px\*200px;红色背景  鼠标悬停，过渡到黄色背景，并且变为圆形  使用scss编写样式 |

3.导入

|  |
| --- |
| 在scss的语法中，如果一个scss文件以下划线开头，  那么这个scss文件就是一个局部scss文件  scss文件转换成css文件的时候，不会把局部scss文件进行转换  只转换，不以下划线开头的scss文件(全局scss文件)  导入的语法 @import "name";  真正导入的文件名称 \_name.scss 掐头去尾  导入时，不写\_,不写.scss后缀  局部文件被导入后，局部文件中的样式，会在全局文件转换的css中生成。  同时，局部文件中声明的变量，可以在全局文件中使用 |

4.混合器

|  |
| --- |
| 把多个选择器都会使用的样式，封装进一个混合器  需要使用的这些样式选择器，可以调用这个混合器  实现代码的重用。  关键字 声明混合器 @mixin 混合器名称(参数1，参数2){样式}  调用混合器 @include 混合器名称(参数1，参数2)  混合器使用场合---css hack |

5.继承

|  |
| --- |
| 一个选择器，可以使用另外一个选择器的所有样式  .my1{  width:100px;height:100px;  }  .my2{  background:#f00;  @extend .my1;  }  转换之后，继承的表现方式，是群组选择器  .my1, .my2 {  width: 100px;  height: 100px; }  .my2 {  background: #f00; } |

练习

|  |
| --- |
| 在选择器.my-border定义1px solid transparent  设置宽度400px 左外边距20px  在选择器.my-bottom-border中，继承.my-border  并设置底边框为红色，当鼠标悬停时，底边框为蓝色 |

6.运算

|  |
| --- |
| 加减乘除余  如果必要，会在不同单位间转换(前提是scss能转) |

①加法

|  |
| --- |
| + 除了做加法，还做字符串拼接  字符串拼接的时候  如果用有引号的字符串拼接无引号的，结果带引号的  如果用无引号的字符串拼接有引号的，结果不带引号 |

②减法

|  |
| --- |
| 由于变量声明的时候可以使用-  系统分不清楚 - 是不是属于变量名称  所以我们写减法的时候，要在 - 前后添加空格  width:$my-w - $my-h; |

③除法

|  |
| --- |
| 在scss中， / 的作用是分隔符  只有下面几种情况，我们判定为除法  1.运算式的两边，有变量，或者函数返回值的时候，是除法  width: $w/2;  2.运算式被()包裹的时候，是除法  height:(500px/2);  3.运算式是其他算术运算式的一部分的时候，是除法  margin-left:5px+8px/2px; |

④字符串的插值操作

|  |
| --- |
| 使用#{}在字符串中做插值  content: "liangliang ate #{50+32} baozis"; |

⑤颜色的运算

|  |
| --- |
| #112233+#445566=#557799  #rrggb rgb(r,g,b) 都是分段计算，红+红 绿+绿 蓝+蓝  rgba的元素，需要透明度相同，才允许计算  background:rgba(11,22,33,0.6)+rgba(22,33,44,0.6); |

十二.函数

1.scss预定义很多函数，有些函数直接可以在css中使用

|  |
| --- |
| rgba(r,g,b,alpha)  hsl(h,s,l)  hue:色调 取值0~360 3个色段 0~120 120~240 240~360  saturation：饱和度 0.0%~100%  lightness:亮度 0.0%~100% |

2.数学函数

|  |
| --- |
| round($v) 四舍五入  ceil($v) 向上取整  floor($v) 向下取整  min($v1,$v2....)  max($v1,$v2....)  random 随机数 |

3.字符串

|  |
| --- |
| unquote($v) 去掉双引号  quote($v) 加双引号  to\_upper\_case($str) 把$str转成大写  to\_lower\_case($str) 把$str转成小写 |

4.自定义函数

|  |
| --- |
| @function get\_msg($a,$b){  @return $a\*$b+($a/$b);  }  div{  width:get\_msg(3,2)+px;  } |

练习

|  |
| --- |
| 定义一个带参数的函数get-width($n)  函数内定义两个变量，分别是60px和70px  函数的返回值为，$n乘以两个数的最大值，加上$n乘以两个数的最小值。把返回值赋给div的width属性  min() max() |

十三.指令

|  |
| --- |
| @if(){}  @else if(){}  @else{}  bool的小括号可以去掉 |

项目阶段

|  |
| --- |
| 学子首页，产品列表页，产品详情页  注意：整个学子不是使用响应式布局设计，也没有使用boot  有一些细节，不去实现  在项目中创建一个文件夹，pro  在pro中，把css文件夹创建好，把图片资源导入 |

作业

1.使用面包屑导航完成



2.作业2，boot项目 头部



# JS高级-01

张东

微信:18301092802

准备:

1. 书: 犀牛书——JAVASCRIPT权威指南

2. 公众号: 前端大全

3. 上届的笔记，打印出来，上课跟着看

规矩: 不允许记笔记！

内容:

1. JavaScript高级: 闭包，面向对象，ES5,ES6

2. DOM/JQUERY: 做网页中所有的交互效果

3. VUE:

做案例的时间: 很长，要回答远程班同学的问题

基础好的同学: 打开你的个人项目，在上课做。

正课:

1. 正则表达式:

2. String类型中的正则函数:

1. 正则表达式: PPT

2. String类型中的正则函数:

在字符串类型中，支持3种正则表达式相关的操作:

1. 查找敏感词：4种场景:

1. 查找一个固定的敏感词出现的位置

比如: 查找一个字符串中是否包含"我草"

var i=str.indexOf("我草"[,starti]) //笔记中[]表示可选

位置 的

在str中从starti位置开始找下一个"我草"出现的位置

返回: 如果找到，就返回我草的中我字的下标位置i

如果找不到，返回-1

第二个形参starti，可省略。默认从0位置(开头)开始找

问题: 关键词是固定的！只要用户稍微换一个字，就无法匹配了！

2. 用正则表达式模糊查找一个关键词的位置

如何: var i=str.search(/正则/i)

在str中从头开始查找符合正则要求的敏感词的位置

返回值: 同indexOf的返回值完全相同！

问题: 正则表达式默认区分大小写的！

解决: 在第二个/后加i ignore 忽略(大小写)

问题: 只能返回敏感词出现的位置，不能返回敏感词的内容

3. 用正则查找敏感词的内容和位置:

如何: var arr=str.match(/正则/i)

返回值: 即返回找到的敏感词的内容，又返回敏感词出现的位置。

其实返回的是一个数组:

arr:[ 0:"敏感词内容", index:敏感词位置 ]

如果找不到，返回null

2. 替换

3. 切割

# JS高级-02

每个案例或小视频，至少写/看三遍

1. 代码+注释抄一遍

2. 保留注释，删掉代码，试着看着注释把代码填回来

3. 自己试着将注释+代码一起写回来——才是你自己会

肯定不能都记住，但是用到什么就复习什么，用到什么保证可以找到。——思维导图

作业:

1. 微信 正则表达式:

第一个字: (微|w(ei)?)

微 或 wei 其中ei可有可无

中间: \s\*

第二个字: (信|x(in)?)

信 或 xin 其中in可有可无

(微|w(ei)?) \s\*(信|x(in)?)

正课:

1. String类型中的正则函数:

2. RegExp对象

1. String类型中的正则函数:

1. 查找敏感词: 4种:

1. 查找一个固定的敏感词出现的位置:

var i=str.indexOf("敏感词")

2. 用正则模糊查找一个敏感词的位置:

var i=str.search(/正则/i)

3. 要查找敏感词的内容: 2种:

1. 只查找第一个敏感词的内容和位置

var arr=str.match(/正则/i)

arr: [ "0": "我草", "index": 9 ]

问题: 只能找一个敏感词！

2. 查找字符串中所有符合条件的敏感词的内容:

var arr=str.match(/正则/ig) global

arr: [ "我艹", "我槽" ].length = 2

0 1

强调: 只包含关键词的内容，不再包含关键词的位置

如果找不到，返回null

强调: 如果一个函数可能返回null，就应该先验证返回值不是null，再使用！

4. 查找所有关键词的内容和位置:

reg.exec()

2. 替换: 2种:

1. 简单替换: 将所有关键词都替换为一致的新值

str=str.replace(/正则/ig, "新值");

将str中所有符合正则要求的关键词都替换为指定的"新值"

强调: 所有字符串都是不可变类型, 所有字符串的函数都无权直接修改原字符串，只能返回新字符串

所以，如果想获得新字符串，必须自己用变量接一下。

比如: str=str.replace(/小[\u4e00-\u9fa5]/,"\*\*")

//将str中 所有 小字开头的人名 都 替换成 \*\*，并将替换后的结果，手动保存回原变量中。

2. 高级替换: 根据搜到的不同敏感词替换成不同的新值

比如: 鄙视题: 将每个单词首字母改为大写！

//找到每个单词首字母: /\b[a-z]/g

//用replace将字符串中的所有首字母，替换为其对应的大写字母:

str="you can you up";

str=str.replace(//2个实参

/\b[a-z]/g, //第一个实参是要查找字符的正则表达式

//第二个实参是一个没有名字的函数

//每找到一个敏感词，就自动调用一次

//每次调用时都自动传入本次找到的敏感词给kword，比如: 本次找到“y”，那么kword=“y”

function(kword){

//1. 必须有一个形参

//2.根据kword接到的值不同，返回不同新的替换值

return y.toUpperCase() -> Y

//return出的新值，会被自动替换到当前找到的敏感词位置。

}

)

衍生: 删除关键词: 其实就是替换为空字符串

str=str.replace(/正则/ig, "");

作业: 鄙视: 删除字符串开头和结尾的空字符

3. 切割：将一个字符串按指定的分隔符，切割为多段子字符串

2种:

1. 分隔符是固定的:

var arr=str.split("分隔符")

将str按指定的分隔符切割为多段子字符串，保存在一个字符串数组中。

比如: var email="zhangdong@tedu.cn";

//将email按中间的@切割

var arr=email.split("@");

//["zhangdong","tedu.cn"]

// 0 1

var uname=arr[0];

var domain=arr[1];

2. 分隔符不确定，但有规律:

var arr=str.split(/正则/i)

比如: //再按</li>空字符<li>切割

var arr=html.split(/<\/li>\s\*<li>/i);

2. RegExp对象:

什么是: 保存一条正则表达式，并包含用正则表达式执行验证和查找功能 的对象

何时: 只要在js中使用正则表达式，都要先创建正则表达式对象，再使用正则。

如何:

1. 创建正则表达式对象:

2种:

1. 创建一个固定的正则表达式对象:

var reg= /正则/ig

问题: //之间不允许写js语句动态生成正则表达式

2. 正则表达式需要动态生成

var reg=new RegExp("正则","ig")

说明: 因为RegExp的第一个形参要求接受一个字符串格式的正则表达式。所以，我们有无数种办法，拼接出我们想要的任何字符串。

比如: var str=kwords.join("|");

"明月|白鹭"

var reg=new RegExp(str);

/明月|白鹭/

2. RegExp对象中包含两个函数:

1. 验证字符串是否符合格式要求

var bool=reg.test(str)

检查

用正则表达式reg，检查str字符串是否符合格式要求

返回值: true/false

坑: test()默认只要能找到符合条件的部分内容，就返回true

解决: 今后凡是验证，都必须前加^，同时后加$

比如: //定义手机号的规则表达式reg

var reg=/^1[3-9]\d{9}$/;

//用规则表达式去验证手机号是否符合格式要求

var result=reg.test(phone);

//如果符合要求

if(result==true){

... ...

2. 第四种查找: 查找所有敏感词的内容和位置

# JS高级-03

如果看不见函数的执行过程，该怎么办: 在函数内添加console.log()输出函数的执行和参数，以及返回值

比如:

str=str.replace(/\b[a-z]/,function(kw){

console.log("===内层函数被调用一次===");

console.log("=====自动传入kw=${kw}");

console.log("=====本次调用返回${kw.toUpperCase()}")

return kw.toUpperCase()   
});

再比如:

var arr=[12,3,123,2,1,23];

arr.sort(function(a,b){

console.log("\*\*\*内层函数被调用了一次\*\*\*");

console.log(`\*\*\*\*\*本次用${a}和${b}做比较`);

console.log(`\*\*\*\*\*本次调用返回${a-b}`)

return a-b

})

提示: 在浏览器控制台中，按上下键可找回曾经执行过的代码。

console.log的输出问题：

var arr=[12,3,123,2,1,23];

console.log(arr); //排序前输出数组的内容

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

console.log(arr); //排序后再输出数组的内容

执行结果:

▶ arr

1

2

3

12

23

123

▶ arr

1

2

3

12

23

123

问题: 两次输出的结果都是排序后的。

原因: console.log在首次输出对象时，仅是输出的对象引用地址。当我们点旁边的三角时，才自动去内存中找到对象现在的样子，展开对象。但此时，程序执行完了。

—— console.log()不是拍照！无法留住对象的瞬时内容

解决: 将对象的内容转为字符串才是拍照/打印

console.log(String(arr)) [ 1, 2, 3, 12, 23, 123 ]

|| "1,2,3,12,23,123"

arr.toString()

转 字符串

基础好的同学:

<http://code.tarena.com.cn/WEBCode/wfd1812/>

正课:

1. RegExp对象  
2. Function对象

1. RegExp对象

1. 验证: var bool=reg.test()

2. 查找关键词: 查找敏感词的内容和位置

var arr=reg.exec(str)

在字符串str中查找一个符合正则表达式reg要求的关键词

返回值: 跟str.match(reg)不加g，返回的值完全相同

arr [ 0: 关键词内容, index: 关键词的位置i ]

问题：每次只返回一个敏感词的内容和位置

解决：只要反复调用，reg.exec()会自动跳到下一个继续查找

找所有：用循环:

var reg=/小[\u4e00-\u9fa5]/g;

do{//反复do

//查找敏感词

var arr=reg.exec(str);

//如果找到敏感词，才输出

if(arr!=null){

console.log("找到敏感词!");

console.log(arr);

}else{//否则(找不到)，就退出循环

console.log("找不到了，就退出！")

break;

}

}while(true);

//true 不使用循环条件控制退出，因为我也不知道循环几次！

//而是在循环体内根据条件，用break随时可能退出循环。

2. \*\*\*\*\*Function对象:

什么是: 内存中保存一段可重用的代码段的对象，再起一个名字。

为什么: 重用！

何时: 如果一段代码段需要反复使用，就应该保存在一个函数中，反复调用函数。

如何:

创建: 3种:

1. 声明方式:

function 函数名(形参列表){

函数体;

return 返回值;

}

问题: 声明提前: 在程序开始执行前，会自动将所有var声明的变量和function声明的函数，都提前到当前作用域的顶部，集中创建。赋值留在原地

2. 用赋值的方式:

var函数名=function (形参列表){

函数体;

return 返回值;

}

优点: 仅函数名提前，函数体留在原地。

揭示了函数的重要秘密:

1. 函数名本质就是一个普通的变量

2. 而函数定义其实是一个对象(独立的存储空间)

3. 用new创建:

var fun=new Function("形参","形参",...,"...")

调用: var 返回值=函数名(实参值列表)

调用函数，等效于执行函数中的代码段

调用几次，就重复执行几次

如果不调用，是不会执行函数的内容的

重载:overload

什么是: 相同函数名不同形参列表的多个函数，在调用时，可根据传入实参的不同，自动选择匹配的函数调用。

为什么: 调用不方便！

何时: 同一件事，根据传入实参的不同，自动选择不同的执行过程执行时。

如何:

问题: js语言不支持多个同名函数同时存在，所以默认和不支持重载。

解决: 借助: arguments

什么是arguments: 每个函数中自动包含的可直接使用的接收所有传入函数实参值列表的类数组对象

类数组对象是: 长得像 数组的 对象

vs 数组: 相同: 1. 下标, 2. length, 3. for遍历

不同: 根本不同 类型不同 导致arguments不能使用数组家的任何函数。

匿名函数:

什么是: 创建函数时，不指定函数名

为什么: 2大作用:

1. 节约内存

2. 避免全局污染

何时: 2种:

1. 只要一个函数只使用一次

2. 划分临时作用域的时候，避免全局污染

如何: 2种:

1. 回调函数: 虽然我们自己定义的，但是不是我们自己调用执行的函数。

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

str.replace(/\b[a-z]/g, function(kw){ return ... })

都是为了节约内存！

2. 匿名函数自调: 创建函数后，立刻调用函数

(function(){

var t=new Date();

document.write("<h1>页面加载完成，at: "+t.toLocaleString()+"</h1>")

})();

创建函数后，自己调用自己

优点: 不但自己调用后释放，且函数内的局部变量调用后，也会自动释放。

总结: 1. 今后所有js代码都要写在匿名函数自调中

2. 今后禁止使用全局变量：

1. 浪费内存; 2. 全局污染

今日英语单词:

1. sort 排序 将多个数值，按从大到小或从小到大的顺序排列。

2. while 只要...的时候

while(条件) 只要 满足条件 的时候 就做什么事儿

3. break 中断 用于中途停止循环。

4. function 功能 程序中所有反复使用的功能都定义在function 功能名(参数){ —— 像极了数学里的函数

}

5. declare 声明 其实就是"创建"

var a 创建一个变量a

function fun() 创建一个fun函数对象

6. return 返回 用于从函数内，向函数外返回一个结果值

7. hoist 吊车、起重机 将物体从低处，提高到高处

8. argument 参数

parameter 也叫参数

调用函数时()中定义的和传入的，都成为参数变量

只不过，定义时的变量成为 形参

调用时的变量称为 实参

# JS高级-04

正课:

1. Function对象

2. 面向对象编程

1. Function对象：

1. 作用域(scope)和作用域链

什么是作用域:

用法: 一个变量的可用范围

本质: 保存多个变量的内存空间

为什么: 为了保护不同范围的变量间不互相干扰

包括: 2种:

1. 全局作用域: window对象

凡是不在函数内创建的变量和函数都自动加载全局作用域(window)中保存

2. 函数作用域: ?对象

函数作用域仅在调用函数时临时创建

所有函数内定义的变量和内层函数，都自动保存在临时的函数作用域中

一旦函数调用后，临时的函数作用域及其内部的局部变量一起释放！

vs Java:

有三级作用域:

全局

函数作用域

局部作用域:

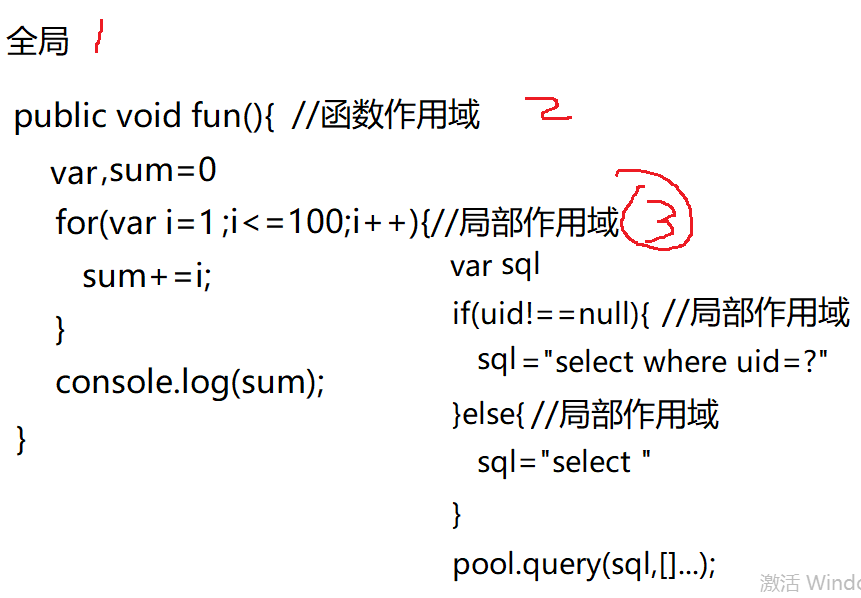
for() { ... }

if(){ ... } else { ... }

while(){ ... }

如果在大括号内声明的变量出了{}，都不能用。

比如:



但js中，出了{}还能用

作用域链: scope chain

什么是: 由多级作用域组成的链式内存结构

原理:

每个函数都有一个隐藏的属性scopes

不调用函数时scopes只包含window的引用，指向全局作用域。

调用函数时，创建临时作用域对象，并将临时函数作用域对象的引用地址添加到scopes中，window之前。

调用时优先使用临时函数作用域中的变量，如果函数作用域中没有，才去全局window中找。如果都找不到，才会报错。

调用后: 函数作用域会被释放，函数作用域中的局部变量也就释放！

2. 闭包:

什么是: 即重用一个变量，又保护变量不被篡改

为什么:

全局变量: 优: 可重复使用。缺: 易被污染。

局部变量: 优: 仅函数内可用。缺: 不可重用。

何时: 当一个函数需要拥有一个可重用的专属的变量时

如何: 给函数和变量找一个共同的父母函数

3步:

1. 定义外层函数包裹内层函数和要保护的变量

问题: 原来全局的函数，现在变成局部的函数了，外边用不了了

2. 外层函数将内层函数抛出到外部！

问题: 外层函数如果不调用，是不执行的

3. 使用者调用外层函数，获得外层函数返回的内层函数保存在变量中。

结果: 使用者可反复调用保存在变量中的内层函数

function parent(){

//包 红包

var total=1000;//要保护的变量

//2. 外层函数将内层函数返回到外部

//生 一个孩子

return function(money){//内层函数

total-=money;

document.write(`本次花了${money}元，还剩${total}元<br>`)

}

}

//3. 使用者调用外层函数，返回内层函数的对象，使用者用自定义的变量接住返回的内层函数对象。

var child=parent();

/\*child:function(money){

total-=money;

document.write(`本次花了${money}元，还剩${total}元<br>`)

}\*/

total=0;

child(100);//剩900

child(100);//剩800

问题: 全局没有total=1000，内层函数也没有total=1000，total保存在哪儿了？又是怎么被重复使用的？

鄙视: 闭包如何形成:

外层函数parent调用后，外层函数的作用域被内层函数引用着，无法释放。

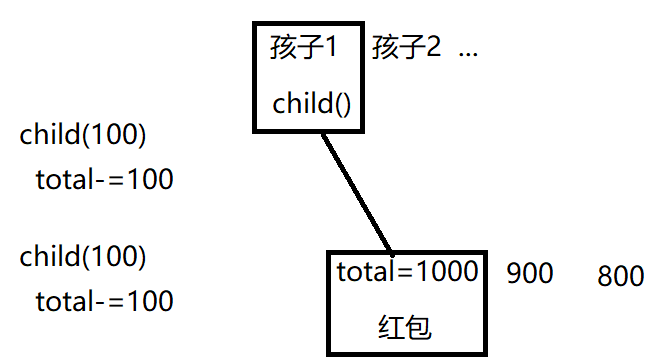
外层函数的作用域，因为是保存在内层函数的私密的scopes通讯录里，所以外人都不知道！且通过全局，绝对无法修改。

鄙视时: 画简图: 找两样东西:

1. 外层函数共向外返回了几个内层函数

父母 共 生了 孩子

2. 那个变量是受保护的变量（给孩子包的红包）



再比如:

function fun(){

//包 红包

var n=999;

//任何位置，给不存在的变量强行赋值，结果自动在全局创建！

//生 孩子1

nAdd=function(){n++};

//生 孩子2

return function(){

console.log(n)

}

}

var getN=fun();

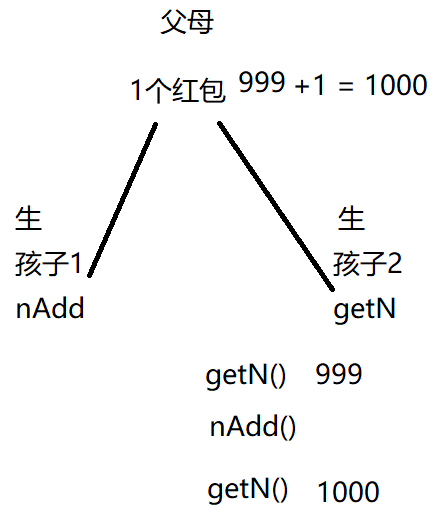
//生了两个孩子: getN和nAdd

//包了一个红包，两个孩子共用

getN();//获取红包金额 999

nAdd();//将红包+1

getN();//获取红包金额 1000



闭包的缺点: 闭包比普通的函数多占用一块内存！

多占用外层函数的作用域对象，不能释放

闭包是用外层函数包裹受保护的内层函数和变量，外层函数将内层函数和变量返回到外部，使用者调用内层函数

解决: 释放闭包: 2种:

1. 给child变量赋值为null

变量child依然在内存中

2. 删除child: delete window.child

删除 window中的child变量

前提: child变量使用window.添加的。

鄙视: var a=10; window.a=10; window["a"]=10;差别

相同: 都是向全局添加一个全局变量a=10

不同: var a不能用delete删除

window.a和window["a"]可用delete删除

总结: 将来就算被迫用全局变量，也必须用window.强行添加！

补: js中向数组中一个不存在的位置强行赋值:

不会报错，而是自动添加该新元素

比如: var arr=["亮","然"]

arr[6]="东" //不报错

结果: arr中6号位置，会自动添加东:

arr:[ 0:"亮",1:"然",6:"东"]

因为js中对象底层都是关联数组，所以，强行向对象中一个不存在的属性赋值，结果，会自动添加该属性。

比如: window.child=xxx;

结果: 会在window中自动添加一个变量child

且child可被delete删除

2. 面向对象:

什么是对象:

用法: 描述现实中一个具体事物的属性和功能的一种程序结构

本质: 内存中可存储多个数据或函数的的存储空间，再起一个名字。

为什么: 为了大量数据的维护和使用方便。

何时: 今后都是用面向对象方式开发程序

如何使用:

三步，也是面向对象三大特点: 封装 继承 多态

1. 封装:

什么是封装: 创建一个对象结构，集中保存一个事物的属性和功能。

为什么: 便于大量数据的维护和使用

何时: 只要用面向对象方式开发，都必须先封装对象。

如何: 3种:

1. 用{}创建:

var 对象名={

属性名 : 属性值 ,

... : ... ,

方法名 : function(){

可重用的功能

}

}

访问对象成员: 成员=属性+方法

访问属性: 对象名.属性名

属性其实就是保存在对象内部的变量而已

访问方法: 对象名.方法名()

方法其实就是保存在对象内部的函数而已

问题: 对象自己的方法，想使用自己的属性，也不能直接写属性名。

原因: 任何函数在执行时，只要没有加.前缀的变量，被限制只能在全局window中查找。不允许擅自进入对象。

不好的解决: 在属性名之前写死对象名:

console.log(`I'm ${lilei.sname},I'm ${lilei.sage}`)

问题: 紧耦合: 别人改，我被迫要跟着改

程序中追求的是松耦合: 别人变，我可不变，或自动跟着变。

好的解决方法: this.属性名

什么是this: 自动获得正在调用当前函数的.前的对象主语。——.前是谁？this自动就变成谁！

何时: 只要对象自己的方法，要使用自己的属性时，必须加this.前缀！

比如:

console.log(`I'm ${this.sname},I'm ${this.sage}`)

今日英文:

1. scope 范围

2. chain 锁链

# JS高级-05

正课:

1. 面向对象编程:

2. ES5

1. 面向对象编程:

面向对象三大特点: 封装 继承 多态

封装: 3种

1. 用{}

2. 用new: 2步:

1. 先创建一个空对象:

var obj=new Object();

{}

2. 用强行赋值的方式添加新属性:

obj.新属性=值

obj.新方法=function(){ ... }

揭示了一个js最底层的原理:

js中一切对象底层都是关联数组。

vs 关联数组:

1. 访问关联数组成员: 2种:

对象["属性名"] 比如: ym["math"]

↑翻译为

对象.属性名 比如: ym.math

固定用法: 如果属性名是动态获得，不是写死的，就必须用[]方式获得属性值，且不能加""

比如： for(var name in obj){

console.log(obj[name])

}

2. 访问不存在的成员: 不报错，而是返回undefined

3. 强行给不存在的成员赋值: 不报错，自动添加新成员

所以，今后向对象中，临时添加新成员，都用强行赋值的方式。

4. 遍历: 关联数组可用for in循环遍历

对象中的成员也可以用for in遍历

鄙视: 克隆对象:

什么是克隆对象: 照着原对象的成员结构，再创建一个新的一模一样的对象。

何时: 不希望别人直接使用原对象时，保护原对象

如何: 3步:

1. 先创建一个空对象

2. 遍历原对象中的每个属性

每遍历一个原属性，就向新对象中添加同名属性，值也和原属性值相同。

3. 返回新创建的对象.

前两种方式的问题: 一次只能创建一个对象。如果反复创建多个相同结构的对象时，代码会很繁琐

3. 用构造函数，反复创建多个相同结构的对象

什么是构造函数: 专门描述一类对象的统一成员结构的函数

为什么: 重用结构

何时: 当需要反复创建相同类型的多个对象时

如何: 2步:

1. 先定义构造函数，规定一类对象的统一结构

function 类型名(属性参数列表){

//有几个属性，就要同时定义几个形参

this.属性名=xx;

this.属性名=xx;

this.方法=function(){

... this.属性名

}

}

2. 用new调用构造函数，照图施工，创建新对象

var obj=new 类型名(属性值列表)

↓ ↓ ↓ 传入

function 类型名(属性形参列表)

原理: new做了4件事:

1. 创建一个新对象:

2. ?

3. 用新对象，调用构造函数

将构造函数中的this都换为新对象

通过给新对象强行赋值新属性的方式，添加新属性。

4. 返回新对象的结果

问题: 如果将方法的定义放在构造函数内，那么，每创建一个新对象，都会创建方法的副本——浪费内存

解决:

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: 节约内存，代码重用

何时: 只要多个子对象，拥有相同的属性值或方法定义时，都应该在父对象中定义一次。所有子对象共用！

如何: js中都是通过原型对象的机制来实现继承

什么是原型对象: 集中存储多个子对象，共有成员的父级对象。

何时: 只要为多个孩子定义统一的方法定义或属性值时，都要定义在原型对象中。

如何:

1. 原型对象不用自己创建: 买一赠一

每当创建一个构造函数时，都会附赠一个空的原型对象

可通过构造函数的prototype属性找到，当前构造函数对应的原型对象

比如: Student.prototype

2. 如何让孩子继承爹: 自动的

new的第二步是设置新对象的爹/原型继承构造函数的原型对象。

所有子对象都有一个\_ \_proto\_ \_属性指向构造函数的原型对象

凡是从\_ \_proto\_ \_指出的关系，都是继承关系。

比如: lilei.\_ \_proto\_ \_=Student.prototype

3. 如何利用继承关系，优化代码: 需要自己做

凡是多个子对象共用的成员方法或属性值，都要定义在构造函数的原型对象中——通过强行赋值的方式

比如: Student.prototype.intr=function(){ ... }

结果：所有孩子可以共用！

总结:

将来要定义构造函数时:

1. 所有的属性，都定义在构造函数中

所有定义在构造函数中的属性，将来都会创建到子对象中。

子对象中有什么属性，取决于构造函数中有什么属性

2. 所有的方法，都要强行添加到构造函数的原型对象中

构造函数中有多少个方法，子对象就能使用多少功能。

内置类型也都有构造函数和原型对象:

内置类型: Array Date RegExp Function ... ...

比如: Array家:

构造函数 + 原型对象

function Array(){ Array.prototype原型对象

... ... sort() join() ... ...

}

才能 new Array()

反例: Math不能new，因为不是构造函数

var arr1=[]; arr1.sort(); arr1.join()

var arr2=[]; arr2.sort(); arr2.join()

如果内置类型中的原型对象中的方法不够用，可自己向原型对象中添加共有方法

比如: 希望给所有数组都添加一个共有的方法sum()，可对当前数组中的所有元素求和:

Array.prototype.sum=function(){

... this->将来调用sum()的.前的数组对象 ...

}

比如，将来:

var arr1=[1,2,3];

var arr2=[1,2,3,4,5];

console.log(

arr1.sum(),//6

//this->.前的arr1

arr2.sum() //15

//this->.前的arr2

);

原型链:

什么是: 由多级原型对象组成的链式结构

原理:

1. 所有对象都有\_ \_proto\_ \_属性指向自己的爹

2. Object类型(function Object(){}和Object.prototype是所有对象的顶级父类型)

3. 只要在自己的原型链上的方法，自己都能用

只要不在自己的原型链上的方法，自己都不能用

强调: 因为爸爸和妈妈之间没有继承关系，所以子对象用不了姥爷中的方法

多态:

什么是多态: 同一个函数在不同情况下表现出不同的状态

包括2种情况:

1. 重载:

2. 重写(override)

什么是: 子对象中定义和父对象同名的成员

何时: 只要爹的不好用，就自己定义自己的

如何: 只要在子对象中，定义和父对象同名的成员

后果: 使用时，先用自己的。不再用父对象继承来的

建议: 所有自定义类型: Student或定义对象var lilei={}，都应该重写toString()方法便于调试

总结:

三大特点： 封装，继承，多态

封装: 创建对象: 2种

1. 如果只创建一个对象，不会反复创建:

首选: var obj={ ... ... }

2. 如果反复创建多个相同结构的对象时:

首选构造函数: 2步:

1. 定义构造函数:

function Student(sname,sage){

this.sname=sname;

this.sage=sage;

//不能将方法定义放在构造函数中

}

2. 用new 调用构造函数：

var lilei=new Student("Li Lei",11);

继承: 所有子对象共有的方法，应该添加到构造函数的原型对象中

用子对象调用方法时，先在子对象本地查找。如果自己没有，才延原型链向父级对象查找。直到找到为止。

强调: 原型对象中的方法中的this指将来调用该方法的.前的某个子对象。

多态: 重写: 如果从父对象继承来的方法不好用，可在子对象本地定义同名方法，覆盖父对象中的不好用的方法。

补: 按值传递，没有按~~引用传递~~

什么是: 当两个变量间赋值时，只是将原变量中的值赋值一个副本给对方。

两个后果:

1. 基本类型/原始类型: number string boolean ...

赋值是将原变量中的值本身复制一个副本给对方

修改新变量，原变量不受影响

2. 引用类型: 对象，数组，。。。

赋值只是将变量中保存的对象地址，复制一份给新变量。对象并没有创建一个新的。还是只有一个对象。只不过两个变量用相同的对象地址，指向同一个对象而已。任何一方修改对象，另一方，同时受影响。

今日英文:

1. DRY Don't repeat yourself

不要 重复 你自己的(代码)

2. constructor 构造函数

3. prototype 原型对象

\_ \_proto\_ \_ 原型

4. override: 重写 推翻/遮挡

# JS高级-06

总结: this

1. obj.fun() fun()中的this->.前的obj

2. fun() fun()中的this->默认指window

window.fun()

3. new fun() 中的this->新对象

4. Array.prototype.sum=function(){

this-> 将来调用sum的.前的任意数组对象！

}

正课:

1. 面向对象:

2. ES5

1. 面向对象:

自定义继承: 2种:

1. 只修改一个对象的父对象:

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

Object.setPrototypeOf(child,father)

设置 child 的 prototype为father

2. 批量修改多个对象的父对象:

构造函数.prototype=father

时机: 必须在创建子对象之前换，才能保证之后创建的所有孩子，自动继承新爹！

2. ES5

ECMAScript: javascript语言的国际标准

仅定义了js语言的基本语法

没有定义如何操作网页内容

问题: JavaScript的语言诞生极其草率，有很多广受诟病的缺陷。

ES5: 是ECMAScript制定第五版js语言的标准

1. 严格模式:

什么是: 比普通js运行机制要求更严格的模式

为什么: 为了改善js中很多广受诟病的缺陷

何时: 所有js程序都要运行在严格模式下

如何: 只要在当前代码的顶部添加: "use strict"

使用 严格 模式

新的要求:

1. 禁止给未声明的变量赋值:

旧js中: 给未声明的变量赋值，不会报错，而是在全局创建该变量——造成全局污染。

严格模式: 给未声明的变量赋值，报错！xxx is not defined!

2. 静默失败升级为错误:

什么是静默失败? 执行不成功还不报错！

ES5严格模式中: 将所有静默失败都变为报错！

3. 普通函数调用或匿名函数自调中的this，不再指向window，而是undefined。

4. 禁止使用arguments.callee

arguments.callee() 在当前函数执行内部，再次调用当前函数自己！——专门用于做递归调用。

递归2个问题:

1. 紧耦合: 在函数内写死函数名，将来函数名变化，函数内部被迫跟着修改。

解决: arguments.callee-> 自动获得当前函数本身，而不用管函数名

2. 致命问题: 重复计算量太大！递归效率极低！

严格模式禁用arguments.callee，就意味着，禁用了递归算法！

解决: 绝大多数递归，都可用循环代替！

2. 保护对象:

什么是: 防止对对象的成员或结构做一些无意义的篡改。

为什么: js中的对象，毫无自保能力

如何: 2大块内容:

1. 保护单个属性的值: 2种:

1. 保护普通数据属性的值:

数据属性: 值直接保存在属性的属性

如何: 其实ES5中，对象的每个数据属性，都变成了一个缩微的小对象，每个缩微的小属性对象中都包含一个属性值+3个开关属性: {

value: 实际存储属性值,

writable: 控制是否可修改当前属性值,

enumerable: 控制是否可被for in 遍历到

问题: 虽然可防住for in，但用对象.的方式依然可强行访问。

configurable: 控制是否可删除该属性

控制是否可修改前两个开关

一旦改为false，不可逆

所以当设置其他两个开关时，都带上configurable:false，作为双保险。

}

如果修改属性的开关:

1. 只修改一个属性的开关:

(重新)定义 obj的指定属性

Object.defineProperty(obj, "属性名", {

开关: true/false,

开关:true/false,

})

2. 同时修改多个属性的开关:

Object.defineProperties(obj,{

属性名: {

开关: true/false,

... : ... ,

},

属性名: {

开关: true/false,

... : ... ,

},

})

问题: 使用开关保护属性，只能提供基本的单调的保护。无法根据自定义规则保护属性。

比如: 年龄必须介于18~65之间

2. 用自定义规则保护属性值:

访问器属性:

什么是: 保镖，不实际存储属性值，专门提供对另一个数据属性的自定义保护。

何时: 只要使用灵活的自定义规则保护属性时，都要请保镖。

如何: 3步:

1. 前提: 先定义一个半隐藏的受保护的数据属性，实际存储属性值。且数据属性名不要用正式属性名，最好\_开头，以示区分。

Object.defineProperty(eric,"\_age",{

value: 26,

writable:true, //可以改

enumerable:false, //半隐藏

configurable:false //双保险

})

2. 请保镖: 一请，就是一对儿

一个保镖叫get函数, 专门负责从受保护的数据属性中取值

另一个保镖叫set函数，专门负责将要修改的值，保存回数据属性中。但是，set函数通常都会对传入的新值进行验证！只有验证通过，才保存回去。如果验证不通过，不保存，且报错！

Object.defineProperty(eric,"age",{

get:function(){ return this.\_age },

set:function(value){

//value会自动接住要赋的新值

if(value>=18&&value<=65){

this.\_age=value;

}else{

throw Error("年龄必须介于18~65之间")

}

}

})

3. 保镖何时发挥作用:

当试图用保镖获取属性值时，自动调用get()

比如: console.log(eric.age)

自动调用age:{ get:function(){ ... } }

当试图用保镖给属性赋值时，自动调用set()，且自动将要赋的新值，传给value形参，先验证  
 比如: eric.age=27 27

↓ 自动

会自动调用age:{ set:function(value){ ... } }

总结: 单从使用上来看:

访问器属性的用法和普通属性完全一样！

只不过，运行机制，不同。

2. 保护对象结构:

今日英语单词:

1. strict 严格

# JS高级-07

正课:

1. ES5

2. ES6

1. ES5  
 保护对象:

1. 保护属性:

2. 保护结构: 3个级别:

1. 防扩展: 禁止给对象添加新属性

Object.preventExtensions(obj)

阻止对obj对象扩展任何新属性

问题: 只能防止添加新属性

不能防止删除现有属性

2. 密封: 在兼具防扩展同时，禁止删除现有属性

Object.seal(obj)

1. 自动调用preventExtensions()阻止扩展

2. 自动将所有属性的configurable开关都改为false!

将来一个对象的结构，保护到seal密封程度就够了。

3. 冻结: 在兼具密封的基础上，又进一步禁止修改所有属性值。

Object.freeze(obj)

1. 自动调用seal()密封了obj

2. 自动将所有属性的writable：false

Object.create():

什么是: 基于一个现有的父对象，创建子对象，自动设置继承关系

何时: 没有构造函数，不能new时，也想创建子对象

如何:

var child=Object.create(father,{

属性1:{ value + 三个开关},

... : ...

})

1. 创建一个新的空对象child

2. 自动设置child继承father

3. 为child添加自有属性

call()/apply()/bind()

相同: 都是用来替换函数中不正确的this

何时: 只要函数中的this不是想要的，都可随意替换

如何: 2种:

1. 在调用函数时，临时换一次:

函数.call(替换this的对象, 实参值列表...)

比如: calc.call(lilei, 10000,1000,2000)

1. call是调用的意思，先调用calc函数

2. call将第一个参数对象替换calc中的this

3. call会将后续实参值列表注射入calc中传给形参

问题: 如果不确定要注入的实参值个数，可用数组向函数中注入实参值列表。

如果传入的实参值是放在一个数组中的，可将call()换成apply()，apply()可先打散数组为单个值，再分别传入要调用的函数中。

2. 创建一个和原函数一模一样的函数副本，并永久绑定this为指定的对象:

var newFun=原fun.bind(替换this的对象)

原理: 1. bind会创建一个和原函数一幕一样的新函数副本——只创建，并未调用

2. 将新函数中的this，永久绑定为指定的对象

结果: 从此，调用新函数时，不用再每次都临时替换this

总结: 只要函数调用时，其中的this不是想要的，就可用call/apply/bind替换

1. 如果只在本次调用时，临时替换一次，就用call()

如果给定的实参值是放在数组中的，则用apply()先打散，再调用函数，并传入。

2. 如果希望永久绑定this时，就用bind();

数组家的新函数:

1. 查找元素:

var i=arr.indexOf("元素值",starti)

在arr中，从starti位置开始，找下一个匹配的元素的位置i

如果找到，返回元素的位置

如果找不到，返回-1

2. 判断: 判断数组中的元素是否符合要求

1. 判断数组中的元素是否都符合要求:

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

//回调函数: 自动在每个元素上执行一次

//每次调用时，都会拿到三个值:

// elem 自动获得当前正在遍历的元素值

// i 自动获得当前正在遍历的元素位置

// arr 自动获得当前正在遍历的整个数组对象

return 判断条件

})

原理: 如果将回调函数去每个元素上都调用一次，都返回true，整个every才返回true

如果调用过程中，只要有一个元素返回false，则整个every返回false

2. 判断数组中是否包含符合要求的元素:

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

return 判断条件

})

原理: 只要有一个元素判断返回true，整个some()就返回true。

除非所有元素判断后都返回false，整个some()才返回false

3. 遍历: 对数组中每个元素执行相同的操作。

2种:

1. 对原数组中的每个元素执行相同的操作

arr.forEach(

//for(var i=0;i<arr.length;i++)

function(elem,i,arr){

//不需要返回值

}

)

2. 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

var newArr=arr.map(

//先创建一个新的空数组

//var newArr=[]

//再对每个元素调用回调函数

//for(var i=0;i<arr.length;i++)

function(elem,i,arr){

return 处理后的新值

}

//每次调用回调函数后，都会将本次调用的结果放入新数组中对应位置

//->newArr[i]

//返回新数组

//return newArr;

)

4. 过滤和汇总:

今日英文单词:

1. prevent: 阻止

2. Extension: 扩展

3. define: 定义 un define d

未 定义 过的

4. property: 内存中所有用对象.xxx都可访问的属性统称为property

5. seal 密封，打包

6. freeze 冻结，冷冻

7. call 打电话，程序中: 调用函数

8. every 每个

9. some 一些

10. element 元素: 2个意思:

1. 指数组中的每一项

2. 页面上一组<ANY></ANY>，称为一个页面元素

# JS高级-08

正课

1. ES5

2. ES6

1. ES5

过滤和汇总:

1. 过滤: 复制出数组中符合条件的元素，组成新数组

var newArr=arr.filter(

//var newArr=[];

//for(var i=0;i<arr.length;i++)

function(elem,i,arr){

//自动在每个元素上执行一次

return 判断条件

}//只有返回true的元素，才被复制到新数组中保存

//return newArr;

)

2. 汇总: 对数组内的元素进行求和汇总  
 var sum=arr.reduce(

//var prev=0;

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

function(prev,elem,i,arr){

//prev接住的是，截止到目前的临时汇总值

return prev+elem; //将当前元素值累加到临时汇总值

}//将本次累加的值再放回prev中

//return prev

)

2. ES6

1. let: 专门代替var，声明变量

为什么: var有两个广受诟病的缺陷:

1. 声明提前——打乱了正常的执行顺序

2. 没有块级作用域——块内和块外的变量就会互相干扰

let的特点:

1. 用let声明的变量不会被提前，也就不能提前使用

如果提前使用报错:

Cannot access 'a' before initialization

能 不 访问 a 之前 初始化

//不能在a初始化之前，访问a

在函数内:

let a=xx之前，到function(){内部，这篇区域是变量a的禁区。决不允许使用变量a。即使有全局变量a，也会报错！所以，如果局部变量和全局变量重名时，必须有一个变量要改名以示区分。

2. 添加了块级作用域:

块级作用域: if(){} while(){} for(){} ...

凡是这些块级作用域中的变量，都自成体系，不会影响外部！

原理: let相当于匿名函数自调！

2. 箭头函数: 代替一切函数定义：普通函数定义，匿名函数自调，回调函数

口诀：

1. 去function 变为 ()=>{ ... }

2. 如果只有一个形参，可省略()

3. 如果函数体只有一句话，则可省略{}

但如果仅有的一句话是return，则必须去掉return

箭头函数特点: 让内部this和外部this保持一致！

今后，只要希望回调函数内的this，和外部的this保持一致时，就可将回调函数改为箭头函数。让内部this与外部this保持一致！

但是，如果就是希望函数内的this与函数外的this不同时，就不能改成箭头函数！

比如: 对象中的方法不能改成箭头函数

3. for of: 专门简化for循环，遍历数组

遍历索引数组: 3种:

1. 普通for: for(var i=0;i<arr.length;i++)

场景: 什么都可以做:

顺序 倒叙

依次递增，也可以2,4,6 也可以5 10 15 20

2. forEach: arr.forEach((elem,i,arr){ ... })

场景: 即获得元素的内容，又需要使用元素的位置时

局限: 只能从头到尾，依次遍历！

3. for of:

场景: 仅关心每个元素的内容，不关心位置时

for(var elem of arr){

of会依次取出arr中每个元素的内容

局限: 无法获得当前正在遍历的位置

}

以上三个专门遍历索引数组

遍历关联数组或对象，必须用for in

for in只能遍历关联数组，不能遍历索引数组：

in的本意是找属性的意思，且不仅找自有属性，还找原型链上的共有属性。

4. 参数增强:

1. 默认值: default

ES6中可以给最后一个形参，定义默认值

如何调用函数时，没有给定最后一个实参值，则自动使用默认值代替

如何: 在函数定义时:

function fun(形参1,...,形参n=默认值){

}

调用时: fun(实参1,...缺实参n)

自动用默认值代替:

2. 剩余参数: rest

专门代替arguments

将来想用arguments的地方，都可用剩余参数代替

为什么: arguments的缺点：

1. 不是纯正的数组，是类数组对象

数组家好的函数，arguments一个都不能用

2. 只能获得全部，不能有选择的获得部分实参值

如何: 定义函数时:

function fun(固定形参1,固定形参2,...数组名){

//数组中会接住除前两个固定形参之外的所有剩余实参值。——是纯正的数组

}

优势: rest语法:

1. 是纯正的数组，数组家的函数，都可使用

2. 可有选择的获取部分想要的实参值。

3. 打散数组: spread

传统js中: 可用apply()打散数组为当值，再传入函数。

问题: apply()的本职工作是替换this，然后顺便打散数组。而实际需求中，多数情况仅打散数组，而与this无关。

比如: 本意从数组中找出最大值

Math.max.apply(null, arr)

//null可换成任何东西

//因为本需求和this无关

//但是必须写！

解决: 调用函数时:

Math.max(...arr)

//拆

5. 解构: destruct

什么是: 将一个大的对象/数组中的成员，提取出来，单独使用。

何时: 只要仅使用一个大的对象/数组中的一小部分成员时

如何: 3种:

1. 数组解构: 从数组中取出个别元素，单独使用

如何: 下标对下标

var date=[2019,6,7,8,30,0];

0

↓原数组中相同位置上的值被提取出阿里，赋值给了相同位置的变量

↓

0

var [y]=date;

y=2019;

var [ , , ,h,m,s]=date;

h=8, m=30, s=0

2. 对象解构: 从一个大的对象中提取出想要的个别成员单独使用:

如何: 属性对属性:

var 对象={

属性1: 值1,

... : ... ,

方法1: function(){ ... },

方法2: function(){ ... }

}

//比如只想用属性1的值和方法1函数

var {属性1: 变量1, 方法1: 变量2}=对象

结果: 变量1=值1,

变量2=方法1function(){ ... }

如果属性名和要用的变量名一致，可简写为:

var {属性1, 方法1}=对象:

结果: 属性名即当做配对，又作为提取出来之后的变量名使用。

3. 参数解构: 将所有参数放在一个对象中传入:

问题: 如果多个形参都不确定有没有时，既不能用默认值解决(只能用在最后一个参数)，又不能用rest解决(rest只能用在末尾的多个形参)

解决: 必须用参数解构： 2步

1. 定义函数时，就将所有的形参都以名值对的方式定义在一个对象中

2. 调用函数时，所有实参也必须以名值对的方式，放在一个对象中整体传入。

今日单词:

1. filter 过滤

2. reduce 统计、汇总

3. initialization 初始化 n. let a=10

initial() 初始化: v.

简写: init

4. default: 默认

5. rest 剩余

6. spread 散播

7. destruct 解构/拆解

# JS高级-09

正课:

1. ES6

1. ES6

面向对象:

1. 对{}中对象成员的简化:

1. 如果对象中的属性值来自于外部变量，且变量名刚好等于属性名: 可简写为只写一个名字:

var ip="127.0.0.1";

var port=3306;

var dbconfig={

ip, //ip: ip,

port, //port: port

}

2. 对象方法的简写:

var lilei={

sname:"Li Lei",

sage: 11,

intr(){ //intr: function(){

...this.sname...this.sage...

}

}

2. 对构造函数和原型对象的封装:

旧js: 构造函数和原型对象是分开定义的，不符合封装的要求。

ES6: 1. 用class{} 包裹构造函数和原型对象方法

2. 构造函数名提升为类型名，跟在class后

今后所有构造函数名统一称为constructor

3. 所有原型对象方法可:

去掉"类型名.prototype.前缀"

class认为，直接保存在class内的方法，默认就是保存在原型对象中的

去掉"=function"

比如: class Student{

//function Student(sname,sage){

constructor(sname,sage){

this.sname=sname;

this.sage=sage;

}

//Student.prototype.intr=function(){

intr(){

console.log(`I'm ${this.sname}, I'm ${this.sage}`)

}

}

Promise: 专门为了解决回调地狱的问题

场景: 多个异步函数，要求必须同步执行

回调地狱: 见视频

解决: Promise技术：

# DOM-01

正课:

1. 什么是DOM

2. DOM树

3. 查找元素

1. 什么是DOM: Document Object Model

什么是: 专门操作网页内容的一套对象和函数的集合

JS=ECMAScript + DOM + BOM

核心语法 操作网页 操作浏览器软件

何时: 今后只要操作网页的内容，都用DOM

DOM其实是一套对象和函数的标准——W3C制定

为什么: 统一所有浏览器开发网页内容的标准

几乎所有浏览器100%兼容DOM标准

如何: 增删改查元素 事件处理

2. DOM 树

什么是: 内存中，保存当前网页中所有内容的树形结构

为什么: 树形结构是最直观的存储上下级包含关系的结构。而网页中的所有元素内容，都是上下级包含关系的。

DOM树长什么样？

1. 当浏览器读到网页时，先在内存中创建网页的根 节点对象: documen

2. 浏览器按从上到下的顺序读取网页内容，每读到一项网页内容，就创建一个节点对象，并挂到指定父节点下。

3. 查找元素: 4种:

1. 不需要查找就可直接获得的元素

document 根节点对象

<html> 根元素 document.documentElement

<head>元素 document.head

<body>元素 document.body

2. 按节点间关系查找

何时: 如果已经获得一个节点对象，找周围附近的节点对象时

节点树: 包含所有网页内容的完整树结构

包括: 2大类关系

1. 父子关系: 4个属性

节点.parentNode

父 节点

节点.childNodes

子 节点们(直接)

返回多个直接子节点组成的类数组对象

可用下标访问指定位置的子节点

节点.firstChild

第一个子节点(直接)

节点.lastChild

最后一个子节点(直接)

2. 兄弟关系: 2个属性

节点.previousSibling

前一个 兄弟

节点.nextSibling

后一个兄弟

问题:节点树包含所有节点，包括看不见的空字符节点。给查找元素造成了极大的障碍。程序员只关心元素节点，不关心文字节点。

解决: 元素树: 仅包含元素节点的树结构

包括: 2大类关系

1. 父子关系: 4个属性

元素.parentElement

父 元素

元素.children

孩子们(直接)

返回多个直接子元素组成的类数组对象

可用下标访问指定位置的子元素

元素.firstElementChild

第一个 元素 子 (直接)

元素.lastElementChild

最后一个 元素 子

2. 兄弟关系: 2个属性

元素.previousElementSibling

前一个 元素 兄弟

元素.nextElementSibling

后一个 元素 兄弟

元素树的优点: 不包含看不见的空字符。不会受到空字符的干扰。

鄙视题: 遍历一个指定元素下的所有后代元素，找到符合条件的元素

递归遍历: 2步:

1. 仅遍历指定父元素下的直接子元素，执行替换操作

2. 对每个直接子元素，调用和父元素完全相同的操作

3. 按HTML特征查找

问题: 按节点间关系查找，必须先有一个节点，才能找周围的关系。

解决: 今后，在还未获得元素的情况下，执行首次查找，都用按HTML特征查找

如何: 4个函数：

1. 按id查找一个元素

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

强调: 1. 必须用document调用

2. 只能找到一个符合条件的元素

万一有两个id重了，只能找第一个

2. 按标签名查找多个元素  
 var elems=任意父元素.getElementsByTagName("标签名")

强调: 1. 可以用任意父元素调用

用哪个父元素调用，就只在哪个父元素下查找

2. 返回多个元素组成的类数组对象

3. 不仅查找直接子元素，且查找所有后代

3. 按class名查找多个元素

var elems=任意父元素.getElementsByClassName("class名")

强调: 1. 用任意父元素调用

2. 返回多个元素组成的类数组对象

3. 不仅查找直接子元素，且查找所有后代

4. 仅靠其中一个class，就可以找到元素

4. 按name属性值查找多个元素

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

专门用于找表单中的有name属性的表单元素

强调: 1. 只能用document调用

2. 返回多个元素组成的类数组对象

4. ?

# DOM-02

正课:

1. 查找

2. 修改

3. 添加/删除

1. 查找: 4种

1. 不需要查找就可直接获得  
 2. 按节点间关系查找

3. 按HTML特征查找: 4种特征:

1. 按id

2. 按标签名

3. 按class

4. 按name

问题: 一次只能按一个条件查找！当查找条件复杂（需要多个条件反复查找才能找到元素）时，代码会很繁琐，步骤会很多。

4. 按任意选择器查找元素:

2个函数:

1. 仅查找一个符合条件的元素:

var elem=任意父元素.querySelector("选择器")

强调: 1. 用任意父元素调用

强调:如果采用指定父元素，调用函数时必须是相对于父元素内的相对选择器

比如:

document.querySelector("table>tbody td:last-child ")

table.querySelector("tbody td:last-child")

~~table.querySelector("table>tbody td:last-child ")~~

2. 只能返回1个元素

3. 不仅查找直接子元素，且在所有后代中查找

2. 查找所有符合条件的元素:

var elems=任意父元素.querySelectorAll("任意选择器")

强调: 1. 用任意父元素调用，缩小范围

2. 可返回所有符合条件的元素的类数组对象

3. 不仅查找直接子元素，且在所有后代中查找

缺点: 用选择器查找，效率不如用HTML特征查找高

总结: 1. 如果只要一个条件，就能找到想要的元素时，首选按HTML特征查找

2. 如果查找条件复杂时，需要多个条件才能找到想要的元素时，就首选按选择器查找——简单！

2. 修改:

3种:

1. 内容:

1. 获取或设置元素开始标签到结束标签之间的原始HTML代码片段: 元素.innerHTML

2. 获取或设置元素开始标签到结束标签之间的纯文本内容: 元素.textContent

vs innerHTML

1. 去掉了内嵌的标签

2. 转义符号翻译为正文

3. 获取或设置表单元素的值:

表单元素.value

2. 属性:

3大类:

1. HTML标准属性: HTML标准中规定的属性

比如: title, class, href, src, target, id

2种:

1. 用最初的核心DOM函数: 4个

var 属性值=元素.getAttribute("属性名");

元素.setAttribute("属性名","属性值");

var bool=元素.hasAttribute("属性名")

元素.removeAttribute("属性名")

2. HTML DOM——核心DOM函数的简化版

将所有的HTML标准属性都提前定义在了元素对象上，属性值暂时都是""。

元素.属性

比如: 获得属性值: 元素.属性

修改属性值: 元素.属性=值

判断是否包含属性: 元素.属性!==""

移除属性: 元素.属性=""

总结: 今后只要是HTML标准属性，都可用.访问

特例: class属性，虽然是标准属性，但不能直接用.访问

class是js的关键词。

解决: 今后，凡是使用class属性，一律更名为className

使用className等于使用class

2. 状态属性: disabled checked selected

特殊: 他们的值都是bool类型

而核心DOM的4个函数只能操作字符串类型的标准属性

所以，状态属性只能用.访问！

3. 样式:

# DOM-03

正课:

1. 修改

2. 添加/删除

3. HTML DOM常用对象

1. 修改:

内容:

属性: 3种:

1. HTML标准属性: 用.

2. 三大状态属性: disabled checked selected

只能用.

3. 自定义扩展属性:

什么是: HTML标准中没有规定的，程序员自发添加的属性

何时: 2个典型场景:

1. 在客户端临时缓存数据，减少向服务器端发送请求的次数

2. 代替其它选择器，用于选择触发事件的元素，绑定事件处理函数

如何:

1. 添加自定义扩展属性:

在html页面中写死: <ANY data-自定义属性名="值"

比如: <img src="1.png" data-m="m.png" data-l="l.png">

用程序动态设置: 只能用setAttribute，不能用.

比如: img.setAttribute("data-m","m.png")

结果:<img data-m="m.png"

2. 修改自定义扩展属性: 也只能用setAttribute()

3. 读取自定义扩展属性的值: 只能用getAttibute()

比如: var m=img.getAttribute("data-m")

m="m.png"

强调: 因为自定义扩展属性，只出现在页面中，不在内存中，所以，只能用getAttribute() setAttribute()...来访问，不能用.访问.

样式:

1. 修改内联样式:

元素.style.css属性="值"

比如: div.style.display="block"

效果: <div style="display:block"

强调: 1. css属性名如果带-，要去横线变驼峰

比如: z-index -> zIndex

background-position -> backgroundPosition

2. 如果css属性值带单位，则必须加单位！

问题: 如果批量修改多个css属性时，使用style一次只能改一个css属性。代码会很繁琐。

解决: 今后，只要批量修改元素的样式都要先将多个css属性定义为一个class，再用.className="class名"一句话批量应用。

2. 添加/删除

添加一个新元素: 3步:

1. 先创建一个空元素:

//标签名

var a=document.createElement("a")

结果: <a></a>

2. 设置空元素的关键属性:

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

a.innerHTML="go to tmooc";

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

3. 将新元素挂载到DOM树上的指定位置上:

1. 在指定父元素末尾追加子元素

父元素.appendChild(新元素)

追加 孩子

比如: body.appendChild(a)

将a元素追加到body元素下子元素的末尾

2. 插入到现有一个元素之前

父元素.insertBefore(新元素, 现有元素)

比如: body.insertBefore(a, h1)

将a插入到h1之前

3. 替换现有的一个元素

父元素.replaceChild(新元素, 现有元素)

比如: body.replaceChild(a, h1)

用a替换body下的h1

优化: 尽量减少操作DOM树的次数

为什么: 每操作一次DOM树，浏览器都要重绘整个页面，效率很低！会造成闪屏

如何: 2种:

1. 如果同时添加父元素和子元素时，应该先在内存中将所有子元素添加到父元素中。最后再一次性将父元素整体添加到DOM树——只重绘一次

2. 如果父元素已经在页面上了，要添加多个平级子元素。

借助文档片段:

什么是: 内存中临时保存多个平级子元素的虚拟父元素

何时: 如果父元素已经在页面上了，要添加多个平级子元素

如何: 3步:

1. 临时创建文档片段托盘对象:

var frag=document.createDocumentFragment();

2. 将子元素临时添加到文档片段对象中

frag.appendChild(子元素)

3. 将文档片段添加到页面中指定的父元素下

父元素.appendChild(frag)

结果: frag将子元素都送到父元素指定位置后，就释放了！不占用页面空间。

# DOM-04

正课

1. 删除:

2. HTML DOM常用对象

3. BOM

4. 事件

1. 删除:

任意父元素.removeChild(孩子)

或者:

孩子.parentNode.removeChild(孩子)

2. HTML DOM常用对象

HTML DOM是对原有DOM函数和对象的简化

HTML DOM 对一些常用的元素对象也提供了简化的函数和属性

比如: <img> <select><option> <table>...</table> <form>...

Image对象: 就是一个<img>元素

唯一的简化，就是在创建<img>元素时:

var img=new Image();

强调: 通常只要两个元素可以用new简写: <img> <option>

其它元素依然需要document.createElement()创建

Select对象: 就是一个<select>元素

简化:

属性:

.selectedIndex 获得select下选中的option的下标位置

.options 获得select下所有option元素的集合

.options.length 获得select下所有option元素的个数

.length <==> .options.length

方法:

.add(option) <==> .appendChild(option) 追加一个option

问题: .add()不支持frag

.appendChild()支持frag的

.remove(i) 移除i位置的一个option

Option对象: 就是一个option元素

唯一的简化，就是在创建时:

var option=new Option(内容文本, value)

比如: var option=new Option("-请选择-",0)

结果: <option value="0">-请选择-</option>

vs 原DOM函数:

var option=document.createElement("option")

option.value=0;

option.innerHTML="-请选择-"

Table对象: 代表页面上一个<table>元素

管着行分组:

创建行分组: table.createTHead()

table.createTBody()

table.createTFoot()

删除行分组: table.deleteTHead()

table.deleteTFoot()

获得行分组: table.tHead

table.tFoot

table.tBodies[i]

行分组管着行:

创建行: var tr=tbody.insertRow(i)

在tbody中第i行位置插入一个新行

并返回刚添加的新行对象，便于后续对新行中添加格和数据。

强调: insertRow()不但创建新行，而且立刻将新行添加到tbody中。

vs 原DOM: 2步:

var tr=document.createElement("tr")

tbody.appendChild(tr)

固定套路: 在tbody开头插入一个新行

var tr=tbody.insertRow(0)

在tbody结尾追加一个新行

var tr=tbody.insertRow()

删除行: tbody.deleteRow(i)

删除tbody中i位置的行

问题: i 要求是行在tbody内部的相对下标位置

而一个tr对象的相对下标位置无法自动获得

可以自动获得的tr.rowIndex是相对于整个表中的下标位置。和tbody.deleteRow()的要求错位

解决: 今后只要删除行的标准写法都是:

table.deleteRow(tr.rowIndex)

因为.前的主语变成整个table，所以deleteRow()要求就变为使用整个表中的下标。就和tr.rowIndex的意义相符了。

获取行: tbody.rows[i]

行管着格:

添加格: var td=tr.insertCell(i)

通常都是在行末尾追加一格: var td=tr.insertCell()

vs 旧DOM:

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

tr.appendChild(td);

删除格: tr.deleteCell(i)

获取格: tr.cells[i]

Form对象: 代表一个<form>元素

获取表单对象: 其实也不需要查找，就可直接获得

因为表单对于所在的网页极其重要，所以网页就将本页面内的<form>元素都保存在一个forms集合中。可用[i]获取某一个表单对象

var form=document.forms[i]

比如: 如果一个页面中只有一个<form>

则 var form=document.forms[0]

属性: form.elements 获得表单中所有表单元素的集合

form.elements.length 获得表单中所有表单元素的个数

form.length <==> form.elements.length

表单元素对象: 比如: <input> <select> <textarea> <button>

获取表单中的表单元素:

普通方法: var 表单元素=form.elements[id|i|name]

比如: 获得form中name=username的input元素

var txtName=form.elements["username"]

获得form中的倒数第二个表单元素——input type=button保存按钮

var btnSubmit=form.elements[form.length-2];

简写: 如果一个表单元素有name属性，可直接用.name方法获得:

var txtName=form.elements["username"]

可简写为: form.username;

方法: 表单元素.focus() 让当前表单元素自动获得焦点

3. BOM: Browser Object Model

什么是: 操作浏览器窗口的对象和函数

JS=ECMAScript + DOM + BOM

核心语法 操作网页内容 操作浏览器窗口

ES标准 W3C标准 没有标准

问题: 没有标准——极大的兼容性问题，所以用的越来越少

包括: 对象体系

window: 3个角色:

1. 顶替了ES标准中的global，充当全局作用域对象

比如: 浏览器中: var a=10; window.a 输出10

再比如: window.parseInt() window.parseFloat()

window.Array window.Date window.RegExp

2. 包含所有ES,DOM,BOM的对象和函数

比如: window.document

window.document.getElementById()

3. 还代表当前正在打开的窗口

比如: window.close() 是关闭当前窗口的意思

新chrome浏览器禁止随意关闭窗口，弹出黄色警告

再比如: window.outerWidth 可获得当前窗口的总宽度

window.outerHeight 可获得当前窗口的总高度

打开窗口: 4种:

1. 在当前窗口打开新链接，可后退

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

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

2. 在当前窗口打开新链接，禁止后退

js: location.replace("新url")

原理: 用新的url地址，替换history中旧的地址，保证history中只有一条url，以此禁止后退

3. 在新窗口打开新链接，可同时打开多个

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

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

4. 在新窗口打开新链接，只能打开一个

html: <a href="url" target="自定义新窗口名">

js: window.open("url","自定义新窗口名")

原理: 其实，每个窗口在浏览器内存中都有一个唯一的名字，用来标识这个窗口。

浏览器规定，相同名称的窗口，只能打开一个。

新打开的同名窗口会覆盖现有的重名窗口

结果: 1. 如果打开链接时，使用自定义窗口名，则反复点击链接，只能打开一个

有两个预定义窗口名:

\_self: 自动获取当前窗口自己的名字给新窗口

结果: 新窗口会覆盖当前窗口

\_blank: 空白，就是不指定新窗口名，浏览器会自动在底层分配，保证不重复。

结果: 不限制打开的新窗口个数

window.history: 保存当前窗口打开后，成功访问过的url的历史记录数组。

当前窗口只要成功访问过一个url，url就会被push到history中保存。

能否前进后退，取决于现在正在看的url，在history前后是否有其他已经浏览过的url

只给我们开放了一个函数: history.go(i)

比如: 前进一步: history.go(1)

后退一步: history.go(-1)

刷新: history.go(0)

后退两步: history.go(-2)

window.location: 专门保存地址栏中url信息的对象，并提供了打开新链接的方法。

属性: 可分段获得url的各个组成部分:

完整url: location.href

<http://127.0.0.1:5500/BOM/10_location.html>?username=dingding&pwd=123456&favs=running#top

协议 主机/IP 端口 相对路径

协议: location.protocol

主机+端口: location.host

主机: location.hostname

端口: location.port

相对路径: location.pathname

查询字符串: location.search

?username=dingding&pwd=123456&favs=running

锚点地址: location.hash #top

方法:

1. 在当前窗口打开新链接，可后退: location.href="新url"

2. 在当前窗口打开新链接，禁止后退:

location.replace("新url")

3. 刷新: location.reload();

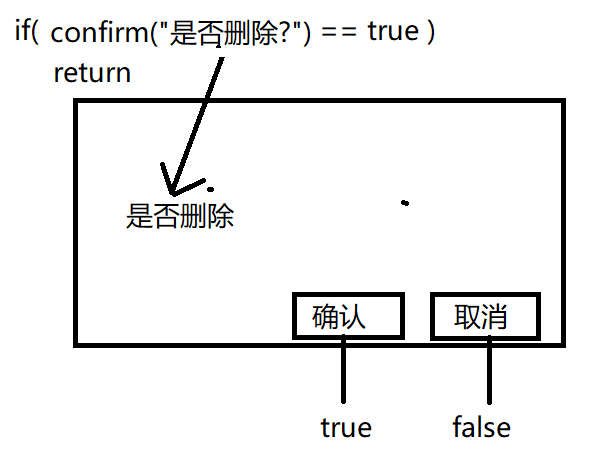
window.navigator: 保存浏览器配置信息的对象

鄙视题: 如何判断正在使用的浏览器的名称和版本号

userAgent: 保存浏览器名称和版本号的字符串

对话框: prompt()输入框 alert()警告框

confirm()确认框



# DOM-05

正课:

1. 事件:

1. 事件:

什么是: 浏览器自动触发的或用户手动触发的页面内容、状态的改变

什么是事件处理函数: 当事件发生时自动调用的函数

强调: 事件处理函数中的this永远指当前触发事件的元素

绑定事件处理函数:

每个元素对象上都有一批on开头的特殊属性，这些属性会在事件发生时自动触发对应的属性。

如果希望事件发生时，按照我们的意愿执行操作，就要提前将一个自定义函数赋值给on开头的对应属性保存起来

当事件发生时，会自动调用提前保存的函数。

如何绑定:

1. 在HTML中:

<ANY onclick="js语句/函数调用"

<script>

function 事件处理函数(){ ... }

问题: 不符合内容与行为分离的原则，不便于维护和重用

2. 在js中用赋值方式:

元素对象.onclick=function(){

... this 指当前触发事件的元素对象

}

问题: 一个事件只能绑定一个处理函数，不灵活！

3. 在js中用添加事件监听对象的方式:

元素对象.addEventListener("事件名",处理函数)

添加 事件 监听对象

强调: 真正的事件名是没有on前缀的:

比如: click change focus blur load

原理:

比如: btnShoot.addEventListener("click",function(){

发射普通子弹

})

btnShoot.addEventListener("click",function(){

发射跟踪导弹

})

结果: 单击btnShoot，会先后发射两种子弹

原理: 其实，浏览器中有一个巨大的事件监听队列

我们为每个元素绑定的事件监听对象都会被添加到监听队列中。

当事件发生时，浏览器采用遍历监听队列的方式，找到符合条件的事件监听对象，并执行其中的处理函数

找到几个符合条件的，就执行几个符合条件的

移除事件监听对象:

元素对象.removeEventListener("事件名", 处理函数)

移除 事件 监听对象

强调: 移除监听对象时，元素、事件名和处理函数都要和添加时完全相同！

坑: 如果一个处理函数可能被移除，则添加监听时，就要使用有名称的函数。不能使用匿名函数

事件模型:

什么是: 从触发事件开始，到所有事件处理函数执行完，所经历的过程

包括三个阶段:

1. 捕获: 由外(document)向内(实际触发事件的元素)

记录各级父元素上绑定的相同事件的处理函数

只记录，不执行

2. 目标触发: 优先触发目标元素上的处理函数

目标(target)元素: 最初实际触发事件的那个元素

3. 冒泡: 由内向外，依次触发各级父元素上收集的相同事件的处理函数

1事件绑定的位置不同，

2减少事件监听个数

3.动态生成元素

事件对象:

什么是: 在事件发生时，浏览器自动创建的，收集事件相关信息的对象。并提供了改造事件的函数

何时: 2种:

1. 想获得事件信息时: 点了那个元素，点在什么位置...

2. 想改造事件默认的行为: 取消冒泡！

如何:

1. 获取: 不用自己创建

事件对象总是作为处理函数的第一个实参默认传入！

我们只要在定义处理函数时，提前定义一个形参接住

比如: btn.onclick=function(e){

//当事件发生时: e自动收到浏览器给的event对象

}

btn.addEventListener("click",function(e){ ... })

2. 包括哪些功能?

取消冒泡: e.stopPropagation()

停止 蔓延

什么是: 当前元素处理函数执行完，停止继续执行父元素上的处理函数

何时: 只要不希望触发父元素上的处理函数时，都要取消冒泡。

事件委托/事件代理:

什么：将事件绑定在目标元素的父元素上，利用冒泡机制触发事件

优点：可以减少事件监听，节省大量内层的占用

可以将事件应用于动态添加的子元素上

缺点：可能造成事件在不应该的位置触发了

优化: 尽量减少事件监听的个数

为什么: 浏览器触发事件，是采用遍历监听数组中每个监听对象来匹配并触发处理函数的。数组中监听对象多，遍历速度慢，数组中监听对象少，遍历快！

何时: 多个平级子元素，要绑定相同的事件处理函数时，只要在父元素绑定一次即可

如何: 只要在父元素上统一绑定一次事件处理函数即可

点击子元素时，会自动冒泡到父元素上，执行提前委托好的处理函数。

坑1: 如何获得最初触发事件的目标元素？

错误: this 因为事件绑在父元素上，执行时，也是冒泡到父元素才执行的。所以，事件委托中的this一律指父元素。

正确: e.target 记录了事件最初触发的目标元素

目标

不随冒泡而改变

坑2: 所有点击的元素都是想要的吗？不是

在正式执行逻辑前，都要先判断目标元素是否是想要的。哪些属性可作为区分的条件: nodeName, className,...

节点名

标签名

阻止默认行为: 有些元素自带默认行为。

何时: 如果元素自带的默认行为不是我们想要的，就可用e.preventDefault() 来阻止默认行为的发生

阻止 默认

比如: bootstrap中

<a href="#content1" data-toggle="tab">

点a时，不但执行绑定的操作，还会附加修改地址栏中的url： 后加#content1锚点地址

所以，今后只要用a当做普通按钮用时，都要阻止默认行为。

鼠标坐标: 事件对象在事件发生时，就自动获得了鼠标在屏幕中的位置。

包括3组:

1. 相对于屏幕左上角:

e.screenX e.screenY

屏幕 屏幕

2. 相对于浏览器文档显示区左上角:

e.clientX e.clientY

客户端 客户端

3. 相对于当前元素左上角:

e.offsetX e.offsetY

偏移 偏移

# jQuery-01

正课:

1. 什么是jQuery

2. 如何使用jQuery

3. 查找元素

1. 什么是jQuery

jQuery是第三方开发的执行DOM操作的极简化的函数库

第三方: 由除了浏览器原生的以及咱们自己编写的之外，其它组织或个人开发的框架/库——要下载

执行DOM操作: jQuery还是在执行DOM的增删改查和事件绑定操作。——学jQuery，还是在学DOM

极简化: jQuery是对DOM操作的对象和函数进行的简化

函数库: jQuery中使用函数解决一切问题。jQuery中一切都是函数！

为什么:

1. 简单:

2. 兼容性: 凡是jQuery让用的，都没有兼容性问题:

比如: IE8不支持nth-child(i)

但是jQuery可随便使用nth-child(i)

因为jQuery内部其实调用的还是原生的DOM函数，且一旦发现不兼容，立刻用js程序模拟。

何时使用jQuery: 几乎所有大项目或框架的底层，都是用jQuery开发的。——仅适合于PC端项目

2. 如何使用:

下载: www.jquery.com

版本: 1.x 兼容旧的浏览器

jquery-1.11.3.js //未压缩

优点: 可读性好，适合学习和开发之用

缺点: 体积大，不便于下载和传输

jquery-1.11.3.min.js //压缩

优点: 体积小，便于快速下载和传输

缺点: 毫无可读型。

2.x 不再兼容旧浏览器

3.x 不再兼容旧浏览器，且添加了新功能

引入: 先引入jquery-1.11.3.js，再编写自定义js代码

因为DOM操作包含大量查找操作，所以js代码必须在body结尾引入，才能保证查找到任何想要的元素。

原理:

引入jquery-1.11.3.js时，其实是在内存中创建了一个新类型叫jQuery，包括

1. 构造函数: 负责创建jQuery类型的子对象

2. 原型对象: 负责保存所有jQuery的子对象共用的简化版函数，比如: .click() .html()都保存在jQuery.原型对象中

因为DOM元素不是jQuery家的孩子，所以，DOM元素不能直接使用jQuery家简化版函数。

只要想使用jQuery操作DOM，都要先创建jQuery类型的子对象。同时，使用选择器查找到DOM树上要操作的元素。将元素保存到jQuery子对象中。

var $btn=new jQuery("#btn1")

强调: 为了将jQuery家孩子和DOM家孩子区分，jQuery家孩子的变量都要以$开头——不是必须。

jQuery家的孩子，就可以调用原型对象中的简化版函数操作自己内部封装的找到的DOM元素了。

对jQuery家孩子调用简化版函数，等效于自动对内部封装的DOM元素调用原生的函数。

比如: $btn.click() =>

会被自动转换为 => btn.addEventListener("click",...)

强调: 原DOM中事件处理函数中的this，获得是DOM家的孩子。不能直接使用jQuery家的函数。所以:

必须先var $btn=$(this)，将this放入一个jQuery家的孩子中，才能使用jQuery家简化版函数。

强调: jQuery家的函数和对象，与DOM一定不能混用！

决定用jQuery，就要通篇都用jQuery。

jQuery函数的三个小脾气:

1. 几乎所有的函数都自带遍历:

因为jQuery子对象其实是一个可保存多个DOM元素的类数组对象。所以，对整个jQuery子对象调用一次简化版函数，相当于对内部的每个DOM元素都调用一次。不用自己写遍历！

2. 多数函数都是一个函数两用:

调用时如果没给新值，就获取原值。

调用时如果给了新值，就变为修改值

比如： $btn.html() 获取btn的内容

$btn.html("xxx") 修改btn的内容

3. 几乎每个函数都返回正在使用的jQuery对象，可继续使用。

强调: 因为$是即创建新对象，又查找DOM树，所以，应该尽量少用！

解决: 链式操作！

注意: 做事儿的步骤，上一步的返回值，刚好是下一步的主语

3. 查找:

jQuery只有两种查找:

1. 按选择器查找:

如何:

var $元素们=$("选择器")

$()里，可以写所有css选择器，都支持！

另外，jQuery也扩展了少量新的选择器——只能在jQuery的$()中使用

新选择包括:

1. 基本过滤/位置过滤:

什么是: 先将符合条件的元素，都取出来，放在一个集合中统一编号。且编号从0开始

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

vs 子元素过滤:

什么是: 按照元素在其父元素内的相对编号选取元素，下标从1开始

:first-child :last-child :nth-child(i)

2. 内容过滤:

1. 用元素内容中的文本作为筛选条件

:contains(文本)

2. 用元素内部的子元素拥有的特征来鉴别父元素

:has("选择器")

3. 可见性过滤: 根据元素是显示还是隐藏，来选取元素

:visible 选择未隐藏的元素

:hidden 选择隐藏的元素

强调: 只能找到display:none，和type="hidden"两种隐藏的元素

4. 表单元素过滤:

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

其实jQuery为每种type都提供了专门的选择器:

:text 选择普通文本框 <input type="text"

:password 选择密码框 <input type="password"

:radio 选择单选按钮 <input type="radio"

:checkbox 选择复选框 <input type="checkbox"

:button 选择按钮 <input type="button"

...

5. 状态过滤:

:enabled 选择启用的元素

:disabled 选择被禁用的元素

:checked 选择被选中的checkbox或radio元素

:selected 选择select中被选中的option

2. 按节点间关系查找:

# jQuery-02

正课:

1. 修改

2. 按节点间关系查找

3. 添加/删除/替换/克隆

4. 事件绑定

1. 修改: 同DOM

内容:

1. 原始HTML片段: .html(["新内容"]) <= .innerHTML

2. 纯文本内容: .text(["新内容"]) <= .textContent

3. 表单元素的值: .val(["新值"]) <= .value

属性:

1. HTML标准属性: 2种:

核心DOM的函数：a.getAttribute()/setAttribute()

HTML DOM ： a.href

2. 状态属性: 只能用.disabled .checked修改

3. 自定义扩展属性:

a.setAttribute()/getAttribute()

网页上可直接看到的属性: Attribute

HTML标准属性和自定义扩展属性

上边的都能用.getAttribute()/. setAttribute()

jquery，都改为.attr("属性名"[, "属性值"]) 一个函数两用

++++++++++++++++++++++++++++++++++++++++++++++++++++

内存中必须用console.log()才能看到的属性: Property

HTML标准属性和状态属性

下边的都能用.直接访问——jq中没有属性，只有函数

jquery中都改为.prop("属性名"[, "属性值"])

比如: 获取a的href属性: $a.attr("href") / $a.prop("href")

修改a的data-target属性值为"content1"

$a.attr("data-target","content1")

修改a的disabled为true

$a.prop("disabled", true )

简写: 同时修改一个元素的多个属性:

$元素.attr或prop({

属性名: "属性值",

... : ...

})

样式:

获取或修改一个css属性的值:

$元素.css("css属性名"[, "属性值"])

如果同时修改多个css属性:

$元素.css({

css属性名: "属性值",

... : ...

})

如果批量修改css属性，都首选class来操作:

添加class: $元素.addClass("class名")

$元素.removeClass("class名")

$元素.hasClass("class名")

=> $元素.is(".class名")

在有或者没有一个class之间来回切换:

$元素.toggleClass("class名")

比如: $元素.toggleClass("down");

在内部自动执行了:

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

$btn.addClass("down")

}else{//否则，就抬起

$btn.removeClass("down")

}

强调: .toggleClass()不能代替.addClass()或.removeClass()

如果只是想添加class，没想去掉class，首选addClass()

除非真的想在有和没有这个class之间来回切换时，才能选择toggleClass()

强调: 所有修改函数都是一个函数两用！

2. 按节点间关系查找:

父子关系

.parent() <= .parentNode

.children() 获得直接子元素

.children("选择器") 只选择符合选择器要求的直接子元素

何时: 如果只在直接子元素中查找

.find("选择器") 在所有后代元素中查找符合条件的元素

何时: 如果希望在所有后代中查找时

.children(":first-child") <= .firstElementChild

.children(":last-child") <= .lastElementChild

兄弟关系

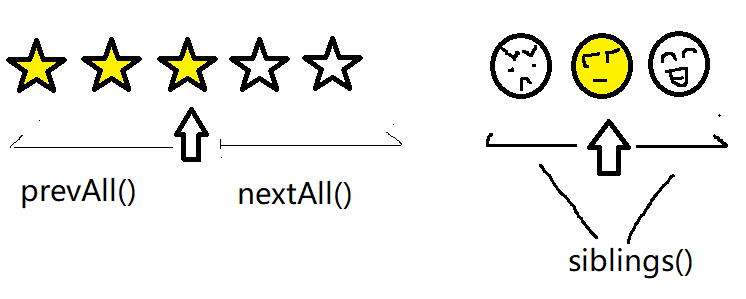
.prev() <= .previousElementSibling

.prevAll(["选择器"]) 在当前元素之前的所有兄弟中查找符合条件的

.next() <= .nextElementSibling

.nextAll(["选择器"]) 在当前元素之后的所有兄弟中查找符合条件的

.siblings(["选择器"]) 选择除自己之外的所有兄弟。不分前后。



3. 添加/删除/替换/克隆

1. 添加: 2步:

1. 用html片段创建DOM元素

var $新元素=$(`html片段`)

2. 将DOM元素添加到DOM树上

将新元素添加到DOM树上

1. 末尾追加: $父元素.append ($新元素) return $父元素

=> .appendChild()

简写: $父元素.append (`html片段`)

$新元素.appendTo($父元素) return $新元素

简写: $新元素.appendTo("父元素选择器")

比如:

1. 将新元素添加到父元素后，要修改父元素的宽度

$父元素.append ($新元素).css("width",500)

2. 将新元素添加到父元素后，要为新元素绑定事件

$新元素.appendTo($父元素).click(function(){ ... })

2. 开头插入: $父元素.prepend($新元素)

$新元素.prependTo($父元素)

3. 在现有元素前插入:

$现有元素.before($新元素) return $现有元素

=> .insertBefore()

$新元素.insertBefore($现有元素) return $新元素

4. 在现有元素后插入:

$现有元素.after($新元素) return $现有元素

=> .insertBefore()

$新元素.insertAfter($现有元素) return $新元素

5. 替换元素:

$现有元素.replaceWith($新元素,) return $现有元素

=> .replaceChild()

$新元素.replaceAll($现有元素) return $新元素

2. 删除: $元素.remove();

3. 克隆元素: $元素.clone();

4. 事件绑定:

DOM中: .addEventListener("事件名", 处理函数)

.removeEventListener("事件名", 处理函数)

坑: 如果一个处理函数可能被移除，则绑定时就要用有名称的函数。移除时，也要用函数名来移除。

jQ中: .on("事件名", 处理函数)

对21种常见的事件，提供了更简化的写法:

.事件名(处理函数)

比如: .click(function(){ ... }) ...

.off("事件名", 处理函数)

事件委托/事件代理:

$(父元素).on("事件名","选择器",function(~~e~~){

//on执行了: this=e.target;

var $tar=$(this); ~~//$(e.target);~~

~~if($tar.is("选择器")){~~//on把选择器拿走自动判断去了

//结果，凡是进入function中的，一定都是符合要求的

直接写正常的事件处理逻辑

~~}~~

})

两个福利: 1. on拿走了选择器，自动判断，咱们不用再写if

2. this->e.target，不用写e和e.target

总结: $有4种用途:

1. $("选择器") 创建jQuery对象，并查找元素，封装进jQuery对象中

2. $(DOM元素) 直接将获得的DOM元素，包装为一个jQuery对象，可调用jQuery家的简化版函数

比如： $(this) $(e.target) $(txt)

3. $(`html片段`) 用html判断创建新的元素

# jQuery-03

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:

页面加载后，自动执行:

问题1: 如何保证无论js代码写在网页的开头还是结尾，都能正常执行呢？

解决1: 将所有的事件绑定代码都写在window.onload=function(){ 中 }

比如://当窗口的内容都加载完之后

window.onload=function(){

//才查找按钮绑定事件

$("#btn1").click(function(){

alert("疼！");

})

}

问题2: window.onload=function(){}是用赋值方式绑定事件。整个窗口只能绑定一个处理函数。如果多个js文件中都包含window.onload=function(){}，结果只有最后一个会覆盖之前所有。如何让多个js文件中的window.onload=function(){}并存？

解决: $(window).on("load",function(){

//.addEventListener

//可进一步简写为:

$(window).load(function()

})

问题3: window.onload必须等待所有网页内容（HTML, JS, CSS, 图片）都加载完才能触发，才能绑定事件。用户有可能等不及要使用元素的功能！

如何让用户不必等待css和图片，也能提前用上网页功能？

解决: jQuery:

//当 DOM树 准备就绪 就提前触发

$(document).ready(function(){

$("#btn2").click(function(){

alert("疼！");

})

})

结论: 今后绝大多数页面初始化或事件绑定操作，都应该放在DOM树加载完就提前触发。

简写:

$(function(){

$("#btn2").click(function(){

alert("疼！");

})

})

总结: 今后所有jq代码，都应该放在$(function(){ ... })

今后见到$(function(){ ... })就是"DOM树加载后就提前执行"的意思

从此$(function(){ ... })代替了匿名函数自调成为所有程序的外层容器代码。

鼠标事件:

mouseover 鼠标进入 mouseout 鼠标移出

问题: mouseover 和mouseout 默认是冒泡的。

也就是说反复进出子元素同样会反复触发父元素上的事件

且不能用e.stopPropagation()解决，因为事件绑在最外层父元素上。不可能阻止所有子元素上来的冒泡。也不可能给每个子元素都绑定事件阻止冒泡。

解决: 就希望进出子元素，不再反复触发父元素上的事件

jQuery中都是用.mouseenter()和.mouseleave() 代替mouseover和mouseout

简写: 如果同时绑定鼠标进入和鼠标移出事件时:

jQuery中可只绑定一个: .hover()

绑定一个.hover()，等效于同时绑定了mouseenter()和mouseleave()

但是hover中需要2个函数。一个给mouseenter，另一个给mouseleave。

$("#target")

.hover(//=mouseenter+mouseleave

function(){//给mouseenter

$(this).addClass("hover")

},

function(){//给mouseleave

$(this).removeClass("hover")

}

)

极其特殊的简写: 如果两个处理函数刚好可以改为一致的，则只需要写一个处理函数即可。一个函数即给mouseenter，又给mouselieave

比如: $("#target")

.hover(//=mouseenter+mouseleave

function(){//即给mouseenter又给mouseleave

$(this).toggleClass("hover")

}

)

模拟触发: 即使没有点击按钮，也可执行按钮上的事件处理函数

如何: $(处理函数所在的元素).trigger("事件名")

触发

//可简写为: .事件名()

2. 动画: 2大类:

1. 简单动画: 固定的三种动画效果:

1. 显示隐藏: .show() .hide() .toggle()

默认: 代替display:block和display:none实现瞬间显示隐藏

所以，如果只是想显示隐藏，不带动画效果，show(), hide(), toggle()还是很好用的！

如果给函数添加ms数参数，就变成带动画的版本

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

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

简单动画函数致命的两个问题:

1. 在jQuery源代码中写死的动画效果，无法维护。

vs css动画: 极其便于维护

2. 都是用js定时器模拟的: 效率低！

vs css动画: 由专门的绘图引擎解析，效率高！

2. 万能动画: 可对任意css属性应用过渡的动画函数

$元素.animate({

css属性: 目标值,

... : ...

},动画持续时间)

执行效果: animate自动获得当前元素状态和css属性值，再与目标值比对，执行过渡动画效果。

问题: 只支持单个数值的css属性

不支持: 颜色过渡， CSS3变换

排队和并发:

排队: 多个css属性先后依次变化:

如何: 对同一个元素，先后调用多次animate，就实现排队执行。

原理:

其实，每个元素都有一个动画队列: [ ]

animate其实不是播放动画的意思

而是将现有动画定义假如到元素的动画队列中

动画何时执行取决于队列中前边有没有其他未执行完的动画。只有之前的动画都执行完，当前动画才自动开始

并发: 多个css属性同时变化

如何: 放在一个animate中的多个css属性默认并发变化

动画结束后自动执行:

其实每个动画函数都有最后一个形参，是一个回调函数。这个回调函数会在动画播放完成后自动被调用！

停止动画: .stop()

坑: 只能停止队列中当前正在播放的一个动画。

队列中后续动画依然继续执行。

解决: .stop(true) 清空队列

选择器: :animated 专门用于判断或选取正在播放动画的元素

3. 类数组对象操作:

问题: jQuery返回的都是类数组对象，经常需要查找/遍历。但是在原生js中，类数组对象很受歧视。因为不是纯正的数组类型，所以数组家的函数，类数组对象都不能用。

解决: jQuery模仿原生js中的indexOf和forEach，重新定义了两个index()和each()函数，专门给类数组对象用。

包括:

1. 遍历jQuery对象中封装的每个DOM元素:

$(...).each() 仿原生js中的arr.forEach()

何时: 只要遍历jQuery结果中的每个DOM元素时

如何:$(...).each(function(i, elem){

//i 正在遍历的当前位置

//elem 正在遍历的DOM元素

//想用elem调简化版函数，必须$(elem)一下

})

2. 查找: 在jQuery对象中查找一个DOM元素出现的位置

var i=$(全部元素).index(要找的元素)

比如: 在所有li中找一个li的位置

var i=$("ul>li").index($li)

简写: 如果在一个父元素内，找某个子元素的位置，可简写为: $子元素.index() ——默认就只在当前父元素内找

4. 添加自定义函数:

何时: jQuery提供的函数不够用时

如何: 其实就是在jQuery的原型对象中添加一个新函数。所有jQuery类型的子对象都可共用！

总结: $的原理: 共有四种用法:

1. $("选择器") 创建jQuery对象，并查找DOM元素

2. $(DOM元素) 将已经获得的DOM元素包装为jQuery对象

3. $("html片段") 创建jQuery对象，同时创建新的DOM元素

4. $(function(){ ... }) 绑定事件，在DOM内容加载后就自动触发

鄙视题: 你觉得我们公司的官网有哪些可以优化的地方:

F12->network->刷新页面

看请求次数: >50次，算多的

可以说，减少请求次数:

1. 用css精灵图，减少图片的张数

2. 尽量减少ajax请求的次数，尽量一次获得全部元素

看个别进度条比较长的文件

可以说，降低图片质量，减小图片体积

压缩/合并js和css代码，减少代码的文件个数和体积

常用事件列表:

click 单击

dblclick 双击

change 选中项改变 常用于select元素

blur 失去焦点

focus 获得焦点

keydown 键盘按下未抬起

keyup 键盘按键按下后抬起

mousedown 鼠标按键按下

mouseenter 鼠标进入元素 代替 DOM中 mouseover

mouseleave 鼠标移出元素 代替 DOM 中mouseout

mousemove 鼠标移动事件

mouseout DOM中的鼠标移出事件

mouseover DOM中的鼠标移入事件

mouseup 鼠标按键抬起

scroll 滚动条发生滚动时触发

input 文本框中输入文字改变时触发

# jQuery-04

正课:

1. \*\*\*封装自定义插件

2. ajax

3. \*\*\*\*\*跨域:

1. 封装自定义插件:

组件/插件: 拥有独立的HTML，css，js和数据的可重用的页面区域

为什么: 重用!

何时: 如果页面中一个功能区域可能被反复使用，都要将其封装为一个插件。

如何: 2种情况:

1. jQuery官方插件: jQueryUI

官网下载: jquery-ui.css

images/

jquery-ui.js

在页面中引入:

<link rel="stylesheet" href="css/jquery-ui.css"

<!--先引入jquery，再引入jquery-ui-->

<script src="js/jquery-1.11.3.js"

<script src="js/jquery-ui.js"

<script>自定义代码</script>

html中:

按照插件的要求，定义HTML内容结构

不需要加任何class

js: 查找插件的HTML父元素，对父元素调用一次插件函数即可自动添加样式和行为。

$("my-accordion").accordion();

原理:

侵入性: 插件函数根据自己的需要，自动为元素添加class和行为

好处: 简单

不好: 不可维护

2. 封装自定义插件:

前提: 已经用html，css和js实现了插件的样式和功能。

如何:

1. 将插件相关的css，剪切到一个独立的css文件中保存。

强调: 如何避免插件/组件间class冲突:

一个组件内的选择器，都要以插件的父元素的class作为查找的开头/起点。

2. 定义独立的js文件：

在jquery的原型对象中添加插件函数

1. 添加样式：侵入: 悄悄自动添加

2. 添加行为：

使用自定义插件: 同jQueryUI的用法。

2. ajax:

jQuery中已经封装好了现成的ajax函数，我们可直接调用

$.ajax({

url:"url路径",

type: "get/post",

data:"uname=dingding&upwd=123456",

//如果本次请求没有参数，可省略data

dataType: "json",

//按照国际标准所有服务端接口，都应该返回json格式的字符串。

//但是也不排除例外

//如果返回的不是json，就不要写dataType:"json"，会出错。

//因为dataType:"json"是自动执行JSON.parse()的意思。如果服务端返回的字符串不是json格式，则JSON.parse()会报错！

//success函数是一个回调函数，会在请求成功结束后，自动调用。参数result，会自动获得服务端返回的值。而且已经被JSON.parse()，范围为js对象了。

success:function(result){

... ...

}

})

//ES 6 promise中

.then(function(result){

//result 获得的就是服务端返回的数据

})

强调: success: 和 .then() 二选一即可！

3. \*\*\*\*\*跨域:

什么是: 一个域名下的网站，向另一个域名下的服务端发送请求。

比如:

网页 服务端

http://www.a.com -------> http://www.b.com

http://a.tedu.com --------> http://b.tedu.com

http://localhost:8080 --------> http://localhost:3000

http://www.12306.cn --------> https://www.12306.cn

(:80) (:443)

http://127.0.0.1:3000 --------> http://localhost:3000

IP 域名

http://tedu.cn/index.html--------> http://tedu.cn/index/login

相对路径 相对路径

不算跨域！

问题: 浏览器禁止ajax请求跨域获取数据

原理:

1. 浏览器其实是发出了请求的。

2. 请求也确实成功了，返回的结果，也拿回来了

3. 因为响应结果中带着响应头，记录着数据来源的IP地址。所以，当浏览器查验服务端返回的数据时，发现响应头中的IP不是当前网页的IP，不让用！

报错: Access to XMLHttpRequest at 'http://localhost:3000/' from origin 'http://127.0.0.1:5500' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

CORS: Cross Origin Resources Sharing

跨 源头 资源 共享 策略

浏览器禁止与当前网页不同来源的数据被使用

解决: 2种:

1. 简单: 单靠 服务端配置响应头

在服务端程序中，在发送消息之前:

手动修改响应头:

node.js ：

res.writeHead(200,{

"Access-Control-Allow-Origin":"http://客户端网页地址栏中的地址"

})

2. 公司: JSONP —— 同时修改客户端和服务端程序才能可支持。

前后端分离: 见视频。

# VUE-01

正课：

1. 什么是VUE:

2. 如何使用VUE:

3. MVVM设计模式

1. 什么是VUE:

VUE是渐进式的基于MVVM设计模式的纯前端JS框架

渐进式: 可以在项目中逐步使用VUE的功能，也可以和其它技术混搭。

vs 全家桶: 要使用，就必须使用全套所有技术。且不能喝其它技术混搭。

MVVM设计模式: ?

纯前端: nodejs学的不好和vue无关！

框架: 原生DOM vs jQuery函数库 vs VUE框架

原生DOM: 步骤多， 函数繁琐

jQuery函数库: 步骤没少，每个函数都变的很简单

框架: 彻底简化了步骤，无需人工干预。

什么是框架: 已经是一个半成品的项目，封装了大量重复性劳动。人只要提供个性化定制即可。

何时: VUE只适合于数据操作为主的项目

还是五件事: 增删改查+事件绑定.

2. 如何使用:

下载: 2种方式:

1. 下载并引入vue.js文件: 前2天

2. 用脚手架代码: 最后一天讲

版本: 2.5

开发版: 未压缩版本 , 有完备的错误提示

生产版: 压缩版本 , 删掉了错误提示

3. MVVM设计模式

传统前端划分:

HTML: 专门保存网页内容和结构的文档

CSS: 专门定义网页样式的文档

JS: 为网页添加交互行为

问题: HTML和CSS太蠢了！不会动态变化

一切交互都只能靠JS添加。

导致JS中大量重复的代码和重复的步骤。

MVVM模式:

1. 界面View: 增强版的HTML和CSS，可跟据数据自动变化

2. 模型数据Model:集中存储一个页面内的所有变化的数据。

3. 视图模型ViewModel: 将View界面和Model模型数据包裹起来，统一管理，自动同步修改。

ViewModel中(new Vue()对象)

包含两大子系统:

1. 响应系统: 将模型数据包裹起来，为每个模型变量自动添加get()和set()保镖。

今后只要修改任何模型变量，都自动经过set()，set()中会触发通知: xx变量变为xx新值

通知会发给虚拟DOM树

2. 虚拟DOM树

什么是: 创建new Vue时，通过扫描完整DOM树，仅提取可能变化的元素和属性，组成的一颗及精简的虚拟DOM树。

优点: 1. 查找元素快

2. 封装了重复性的增删改查DOM操作

虚拟DOM树接到通知，快速找到受影响的元素。调用已经封装好的DOM函数，仅更新受影响的元素的受影响的属性。

# VUE-02

正课：

1. 绑定语法:

2. 指令:

3. 双向绑定:

1. 绑定语法: 学名: 插值语法 Interpolation == {{}}

什么是: 让HTML可以自动找到程序中的变量的特殊语法

为什么: 因为传统的HTML是静态的，缺少动态变化的能力。导致js当中要想操作HTML，需要大量重复的代码。

何时: 只要HTML中某个位置的数据，需要根据程序中的一个变量动态变化！都要用绑定语法！

如何: 2步:

1. 先找页面中所有可能发生变化的地方有几处

2. 再在模型数据中定义相同数量的变量:

new Vue({

el:"#app",

data:{

变量名:值,

... : ...,

}

})

强调: HTML中有几处变化，data对象中就要有几个变量与之对应。

3. 在HTML中，可能发生变化的位置用绑定语法定义变量: {{变量或表达式}}

强调: 其实{{}}的用法和模板字符串中${}的用法完全一样！

能写：变量, 算术计算，关系/逻辑运算，函数调用, 访问数组元素, 三目——凡是有返回值的js表达式都能

不能: if else while for——都是程序结构，没有返回值！

结果: 运行时，HTML中的所有{{}}会自动去data中找同名的变量使用。且内存中的data中的变量值发生变化，HTML中的{{}}的值自动变化！用户最终看到的是{{}}中的变量或者表达式计算后的值！而看不见双花括号——节省了大量重复的查找和修改操作。——多亏了MVVM中的ViewModel中的两大子系统: 响应系统和虚拟DOM树。

2. 指令:

什么是: 为HTML添加更多新功能的Vue预置的自定义属性

为什么: 因为原来HTML缺少程序必须的功能: 判断/分支结构，循环。。。功能。所以只能依靠js中反复查找，反复修改来控制HTML元素的内容和状态。

如何: 13种:

1. 绑定属性:

什么是: v-bind属性专门动态绑定元素的属性值

为什么: 要绑定属性值，不能用{{}}，只能用v-bind或:简写

何时: 只要属性值需要根据变量动态变化时，就要用v-bind或:简写

如何: <img v-bind:src="pm25<100?'img/1.png':

pm25<200?'img/2.png':

pm25<300?'img/3.png':

'img/4.png'">

去{{}}换v-bind:

其实可简写: <img ~~v-bind~~:src="...

去{{}}换:

2. 控制元素的显示隐藏:

1. 控制一个元素的显示隐藏:

<ANY v-show="判断条件"

只要条件满足，就显示元素

一旦条件不再满足，就隐藏元素！

2. 控制多个元素，多选一显示:

<ANY v-if="判断条件" 单用时

v-if从现象上看，和v-show是完全一样的

但是原理不一样

鄙视: v-show vs v-if的差别:

v-show 用display:none 控制显示隐藏

v-if 用删除节点的方式，控制显示隐藏

如果一个元素频繁需要显示隐藏,v-show的效率高！

和v-else-if v-else 配合使用:

<元素1 v-if="条件1"  
 <元素2 v-else-if="条件2"

... ...

<元素n v-else

强调: v-if和v-else-if和v-else之间禁止插入一切其他元素。必须连续写！

原理: Vue会自动判断每个条件：

哪个条件符合，就只显示哪个条件的元素。其余元素都不显示。

如果前一个条件已经满足，则后续判断都不再执行。所以，不可能同时显示多个元素。

如果所有条件都不满足，会显示v-else的元素

3. 根据数组反复生成多个相同结构的HTML元素:

其实就是为HTML添加循环功能:

如何:

1. data中必须先定义一个可遍历的数组

2. 在HTML中使用v-for遍历数组，反复生成多个相同结构的元素，并动态绑定元素的内容。

语法:

<要反复生成的元素 v-for="(elem,i) of 数组" :key="i"

强调: 特例: v-for中的循环变量可被v-for自己或子元素用于绑定！

4. 事件绑定:

v-on:事件名="处理函数"

可简写为:

@事件名="处理函数"

强调:

1. 处理函数必须定义在:

new Vue({

el:"#app",

data:{ ... },

methods:{

处理函数(){

this.data中变量

}

}

})中的methods:{}结构中

2. 其实可以传参: @事件名="处理函数(实参值)"

3. 也可以用事件对象e：用法和DOM中完全一样

获得e: methods:{

事件处理函数(e){ ... }

}

后续:

获得目标元素，实现事件委托： e.target

取消冒泡: e.stopPropagation()

阻止默认行为: e.preventDefault();

键盘事件中获得键盘号: e.keyCode

获得鼠标坐标位置: e.screenX, e.screenY

e.clientX, e.clientY

e.offsetX, e.offsetY

5. 避免用户短暂看到{{}}

问题: 如果new Vue加载慢，则可能短暂看到{{}}

解决: v-cloak 可让元素在new Vue加载之前暂时隐藏。当new Vue加载完成后，会自动查找页面中的所有v-cloak斗篷删除该属性，让元素显示出来。

强调: v-cloak没有属性值！

问题: v-cloak空有属性名，没有配套的样式

解决: 只能自己手写！用属性选择器:

[v-cloak]{display:none}

强调: v-cloak不能改名！改名了，vue就找不到了！

其实，除了v-cloak外，还可用v-text代替{{}}绑定元素内容，避免短暂看到{{}}

如何:

<ANY v-text="`js模板字符串语法，用${变量}动态生成内容`"

比如:

<h1 v-text="`姓名: ${uname}`"></h1>

<h2 v-text="`性别: ${sex==1?'男':'女'}`"></h2>

原理: v-text在未绑定时，是一个浏览器不认识的属性，所以，显示不出来。直到new Vue加载完，认出v-text，才用V-text的内容代替元素的innerHTML内容。

v-cloak vs v-text:

v-cloak: 需要额外添加css样式，且需要选择放在哪个元素上，隐藏什么范围内的元素。整个网页都隐藏，用户体验还是不好的。

v-text: 仅对当前元素有效，不会影响其它元素的显示不显示。用户体验比v-cloak要好。但是使用反引号的语法，很晦涩。

6. 绑定HTML片段:

问题: 使用{{}}绑定HTML片段，VUE不会解析HTML片段为网页内容。而是原样显示HTML代码

解决: 今后，只要绑定一段HTML片段，必须用v-html。

如何: <ANY v-html="变量"></ANY>

7. 只在页面加载之初，绑定一次。之后及时数据变化，也不反复更改页面：v-once

强调: v-once没有任何属性值，写在元素中，就起作用

原理:

在构建虚拟DOM树时，会扫描到v-once的元素，所以首次绑定，能绑定v-once的元素的内容。

首次绑定后，v-once的元素会从虚拟DOM树中移除。从此，再出发变量改变时，不会再修改v-once的元素。

8. 万一内容的正文中，有{{}}，但是不是用于绑定的，会出错。

解决: v-pre 可阻止内容中的{{}}被编译，而是让{{}}原样显示。

总结:

1. 如果元素内容要变化:

用 {{变量或表达式}} 绑定

也可用v-text="`变量或表达式`"

2. 如果元素的属性值要变化: 用 :属性名="变量或表达式" 绑定

3. 一个元素控制显示隐藏: v-show="条件"

4. 多个元素选其一显示:

v-if="条件" v-else-if="条件" v-else

5. 反复生成多个相同结构的元素时:

v-for="(elem, i ) of 数组" :key="i"

6. 只要绑定事件处理函数都用: @事件名="处理函数"

7. 只要不希望用户短暂看到{{}}语法:

v-cloak 或 v-text

8. 只要绑定HTML片段: v-html

9. 只要希望只在页面加载之初绑定一次，之后不再变化，就用v-once.

10. 如果内容中包含不想被vue编译的{{}}正文，可用v-pre阻止VUE编译元素的内容。

# VUE-03

正课:

1. 双向绑定:

2. 计算属性:

3. 绑定样式:

4. 自定义指令:

5. 过滤器:

6. axios:

7. new Vue的生命周期

1. 双向绑定:

什么是单向绑定: 只能将程序中的修改更新到页面中。但是无法将页面中的更改，更新回程序中。

只能从Model -> View 不能从View -> Model

比如: {{}} v-text : 等等 都是单向绑定

比如: <input :value="变量"

结果: 变量中的值可正常显示在文本框中

但是文本框内容修改，变量的值不会改变

什么是双向绑定: 即能将程序中的修改更新到页面中，又能将界面中的修改，反向更新会程序中

何时: 只要希望通过表单元素的修改，来更改程序的内容时。——只要绑定表单元素，都用双向绑定。

如何：不要用: ， 要用v-model:属性名="变量"

原理: v-model其实就是自动为表单元素绑定了oninput或onchange事件。在这些内容更改事件中，自动修改当前Vue对象中的模型变量。

普通绑定为什么不能更新程序中的数据：因为没有自动添加事件，导致界面上更改，Vue框架不知道，自然就无法更近模型变量。

其它表单元素的双向绑定:

1. 单选按钮radio: 要绑定checked

不绑定value，因为radio中的value是写死的，固定不变的。等着被选择。

绑定checked: 因为是否选中radio，取决于checked属性的值。用户更改选中项，改的也是checked属性值。——radio只能绑定v-model:checked="变量"

绑定时:

<input type="radio" name="sex" value="1" v-model:checked="变量"

<input type="radio" name="sex" value="0" v-model:checked="变量" ~~"变量==值"~~

小脾气: v-model在radio中：

从程序绑定到页面时: 会自动用checked后绑定的变量值和当前radio写死的value值做比较，返回bool值决定当前radio是否选中。时不用自己在checked后写完整条件！只写一个变量名即可！

改变选中状态时，v-model会将选中的radio的写死的value值更新回程序中绑定的变量上

2. 单个checkbox绑定: 也绑定checked属性

<input type="checkbox" v-model:checked="isAgree"

3. select元素绑定: 绑定的是select元素整体的value属性

因为，无论选择了哪一个option，当前选中项的值都 变成select元素的value属性值。

如何: <select v-model:value="变量">  
 <option value="值1">

<option value="值2">

... ...

原理:

当从Model->View绑定时: 用变量的值和每个option的value做比较，按个相等，就选中哪一个  
 当修改select下的选中项后，将当前选中的option的value值更新回变量去

其实，以上绑定都有一个共同的简写: 只写v-model即可，可省略:value :checked等。v-model会个识别元素中起决定作用的可变的属性，自动绑定。

watch监控机制:

什么是: Vue中的watch监控机制可监控vue所有模型变量的变化。只要模型变量变化，就自动触发一个同名的函数。在函数中可调试变量的值或执行实时操作

何时: 希望只要变量的值变化，就立刻做一件事儿时

如何: new Vue({

el:"#app", //找要监控的整个页面空间

data:{

变量1:值1

}, //所有页面需要的模型变量

methods:{ ... }, //所有事件处理函数和一些自定义函数

created(){ ... }, //当new Vue()对象加载完自动执行

watch:{ //定义监控变量的函数

变量1(){//必须同名，只有同名的函数，才能监视同名的变量

执行一切js或vue的操作

}

}

})

2. 计算属性:

什么是: 不实际属性值，绑定时，动态通过其他现有属性，计算出自己的属性值。

何时: 页面上需要一个值，但是，这个值数据库中没有直接保存，而是需要动态计算出来。

如何: new Vue({

el:"#app",

data:{ },

methods:{ },

created(){ },

watch:{ },

computed:{

属性名(){

//通过其他变量计算获得最后结果

return 结果

}

}

})

使用计算属性：计算属性虽然定义成函数，但是，用法和普通模型变量完全一样！不要加()

特点:

1. 只要依赖的其它变量变化，计算属性都会自动重新计算

2. 一个属性如果多个位置使用，其实只计算一次。

原理: 计算属性的值，每计算一次，结果都会被Vue框架缓存起来，反复使用。除非依赖的变量发生改变时，才重新计算，然后再重新缓存，反复使用。

vs 函数: 也可以实现计算的效果

问题: 不会被缓存！每用一次就重复计算一次！

总结: 如果只关心计算结果时，首选计算属性

如果不关心结果，甚至没有返回值，只关心执行的过程时，首选函数。

3. 绑定样式:

2种:

1. 动态绑定单个css属性:

2种:

1. 用字符串方式，绑定整个style属性：

<ANY :style="变量"  
 data:{

变量:"top:50px; left:50px"

}

缺点: 在字符串中不便于修改某一个css属性

2. 都用对象方式，绑定style:

<ANY :style="变量"

data:{

变量:{

top:"50px", //必须px

left:"50px" //必须px

}

//绑定时，被自动翻译为字符串:

//"top:50px; left:50px"

}

强调: 固定的style和动态绑定的style可同时存在

比如：

<div style="position:fixed" :style="divStyle">

data:{

divStyle:{

top:"50px", //必须px

left:"50px" //必须px

}

}

运行时，结果: 动态绑定的style会和写死的style合并为一个style:

<div style="position:fixed; top:50px; left:50px"

2. 动态绑定class

2种:

1. 将整个class属性当做一个字符串绑定

问题: 不便于控制一个class的启用或禁用

2. 今后也要使用对象格式绑定:

<b class="固定不变的class" :class="变量名"

data:{

变量名:{

class1: true/false,

class2:true/false,

... ...

}

}

运行时:

1. 将变量中值为true的class转为字符串:

比如: { down:true, fade:true }

-> "down fade"

{down:false, fade:true}

-> "fade"

2. 用翻译好的class字符串和元素上规定的class=""，合并为一个class

如果想用程序切换使用/不使用某个样式类，只要修改类名对应的true/false就行

值为true，就是应用该样式类

值为false，就是不应用该样式类

4. 自定义指令:

何时: 当希望元素自动执行一件事，但是vue中的指令不够用，就可以自定义指令。

如何:

1. 定义一个新的自定义指令:

Vue.directive("指令名",{//指令名不应该带v-前缀

//会在当前元素加载后，自动执行

inserted(elem){

//elem: 接住的是当前指令所在的DOM元素

//可对elem执行任何原生DOM的函数

}

})

强调: 应该定义在不属于任何new Vue()的外部全局

将来所有new Vue()都可以共用这个自定义指令。

2. 使用指令: <ANY v-指令名

但是，使用指令时，必须加v-前缀

原理: 当<ANY>这个元素在HTML中加载完之后，自动去调用指令中携带的inserted()函数

5. 过滤器:

什么是: 将不能直接使用的变量值，过滤成可以给人看的新值的函数

何时: 只要变量的原始值不是想要的，都可用过滤器过滤成想要的值给人看。

如何:

1. 定义过滤器:

Vue.filter("过滤器名称",function(oldVal){

return 新值

})

2. 使用过滤器:

在绑定语法中都可用"|"将过滤器连接在变量之后

运行时: 原始变量值->过滤器->oldVal参数->return过滤后的新值->给人看

过滤器参数:

何时: 当根据不同值，选择过滤成不同的新结果时

如何: 2步:

1. 定义过滤器时添加形参:

function(oldVal,lang="cn"){

2. 使用过滤器时传入实参:

{{sex2|sexFilter("en")}}

强调: 定义时添加过滤器形参，从第二个形参开始添加。但是使用过滤器时，自定义实参值，却从第一个实参传入。但是还不会出错！因为过滤器函数在调用时，自动回将原始值作为第一个实参传入。我们添加的自定义实参，只能排在第二。

其实程序中的过滤器也可以用|串联多个:

{{变量|过滤器1|过滤器2|... }}

坑：定义后一个过滤器时，拿到的值不是原始值，而是上一个过滤过滤后的中间产物。

6. axios

什么是: 专门发送ajax请求的基于Promise的函数库

为什么:

1. 然哥四步/五步:

2. 可以自己封装

3. jQuery库中的$.ajax()

将来在Vue项目中，几乎不会用jQuery

如果仅仅为了发ajax请求而引入jQuery函数库，特别小题大做。

4. Vue官方有一个vue.resource组件

但已停止维护

5. axios: Vue官方推荐的最好的ajax函数库。

何时: 今后只要在任何位置想发ajax请求，都可用axios

如何:

1. 引入axios.js

2. 调用axios.get()函数发送请求

axios.get("url",{

params:{//相当于$.ajax中的data:{ ... }

参数1:值1,

... : ...

}

//运行时params中的参数会被翻译为字符串，拼接到url的结尾:

"url?参数1=值1&参数2=值2&..."

}).then( result=> {

为this保持和外部VUe一致，必须用箭头函数

result返回的不只是服务端响应结果

真正的响应结果包含在result.data中

})

7. new Vue的生命周期:

什么是: 一个new Vue()的加载过程:

何时: 如果希望在加载过程某个阶段，自动执行一项任务时。

如何: new Vue()的声明周期包括4个阶段，8个函数

钩子函数——其实就是在不同阶段自动执行的回调函数

前: beforeCreate(){ ... }

1. 创建(create)阶段: 创建new Vue()对象和data()对象

已经有data对象了，但是没有虚拟DOM树

可以发送ajax请求

因为没有虚拟DOM树，所以暂时不能用DOM操作

后: created(){ ... }

前: beforeMount() { ... }

2. 挂载(mount)阶段: 创建虚拟DOM树，将data中的变量值开始向DOM树上绑定

即有data对象，又有虚拟DOM树

即可发送ajax请求，又可执行DOM操作

后: mounted(){ ... }

//后两个阶段不是必须:

前: beforeUpdate(){ ... }

3. 更新(update)阶段: 当data中的变量值被改变时才触发

后: updated(){ ...}

前: beforeDestroy(){ ... }

4. 销毁(destroy)阶段: 当调用专门的$destroy函数销毁一个组件时，才触发

后: destroyed(){ ... }

# VUE-04

正课：

1. 组件:

2. 组件化开发:

3. SPA:

4. 脚手架:

1. 组件component:

什么是: 拥有专属的HTML,CSS,JS和数据的页面独立区域

为什么: 重用！

何时: 只要发现页面中有一个区域可能被反复使用，都要定义成组件。

如何:

1. 创建: Vue中的组件其实是一个包含html,数据和功能的对象

每个组件都是一个缩微的new Vue()

new Vue()中有什么组件中就有什么

Vue.component("组件名",{

~~el:"选择器",~~

template: //将来会反复复制多个和这里一模一样的html片段，所以将这一份标准的的片段称为模板

` <div> //组件模板必须用唯一的父元素包裹  
 <button>-</button> <span>1</span> <button>+</button>  
 </div>`,

~~data:{ ... },~~

data~~:function~~(){ //

return {

//模型变量

}

},

methods:{ ... }, //同new Vue()

created(){ ... }, //同new Vue()

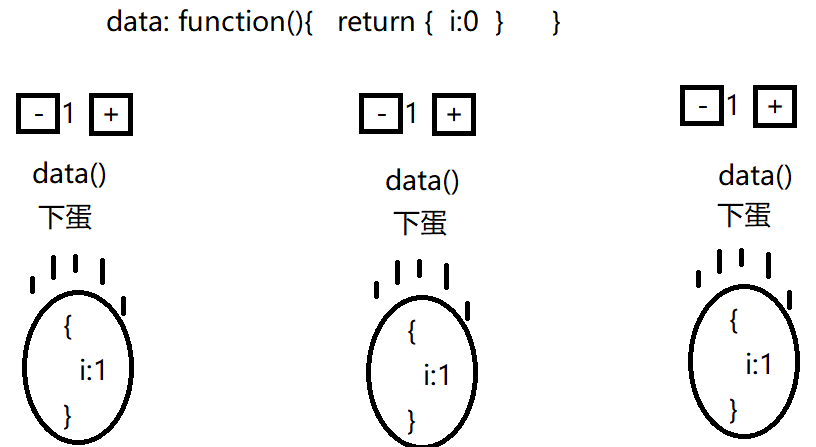
watch:{ ... }, //同new Vue()

computed:{ ... }, //同new Vue()

mounted(){ ... } //同new Vue()

})

为什么: data必须是function:



2. 使用组件: 组件在html中其实就是一个可重用的标签而已。组件名就是标签名。所以组件名起名不要用驼峰命名！因为HTML标签不区分大小写。可用-分隔多个单词

new Vue()要不要写？答：必须写！

因为是全靠new Vue()扫描页面时，发现不认识的标签，才会去加载自定义的组件。

加载时: 会用组件的template中的HTML模板片段，代替不认识的标签。

2. 组件化开发:

什么是: 将所有要做的HTML页面，切割为多个独立的组件，可以分工协作开发。

为什么: 1. 分工协作；2. 松耦合

何时: 今后做页面，都是先将页面划分成多个组件，再分别开发每个组件。

如何:

1. 将页面划分成多个功能区域-组件

2. 创建组件.js，其中定义组件对象：2种:

1. 全局组件: Vue.component()

什么是: 可以在页面任何位置使用的组件

何时: 只要页面中一个区域，可能在多个页面或者多个不同位置随意使用时

问题: 不受约束，随处可用

2. 子组件:

什么是: 必须在指定父组件内才能使用的组件

何时: 如果希望限制一个组件只能在其父组件内使用时，就必须定义子组件

如何:

1. 创建子组件:

var 组件对象名={ 和组件标准写法必须一致 }

强调:

1. {}中必须和Vue.component()中的{}内容格式一致！。

2. 变量名必须用驼峰命名

2. 在父组件中用components属性包裹所有子组件:

Vue.component("父组件",{

template:

data:

methods:

...

components:{ 子组件对象名,... }

})

3. 在主界面或父组件中，使用子组件标签，引入子组件的内容

结果: 浏览时，所有组件(父和子)，最终都合并为一个整体。不再包含不认识的标签。



# VUE-UI-01

今天学习的内容

程涛:taonetwork2018

1:第四个阶段课程介绍

(1)vue 移动端组件库和项目 6 day (mint-ui)

vuex 项目最后一天有个功能:购物车[数量vuex保存]

(2)html5新特性(视频/绘图) 7day

(3)微信公众平台--小程序(云开发) 6day

(4)混编 1day

2:vue

vue 当前很火前端框架，用户数众多

vue 开始设计时针对pc端用户而不针对移动端用户

国内团队:饿了么[MintUI/ElementUI],滴滴团队,MUI..

2.1:补充知识点:pc端项目和移动端项目

-区别:屏幕宽度

pc端: w > 992px

移动端: w< 992px

-区别:操作方式

pc端:鼠标和键盘

移动端:手指

-区别:事件

pc端:click/blur/keyup/keydown/..

移动端:touchstart/touchmove/touchend/tap/longtap

3:组件库--mintui

为了提高开发效率，为了统一开发风格可以使用组件库

来帮助我们完成项目.

组件通常包含:轮播图;导航条;按钮;进度条;面板;切换面板..

表单..

3.1:组件库--mintui组件库使用方法

下载安装二种方式

(1)方式一:(学习)

mint-ui官方网站下载css/js/fonts 加载当前html文件中

(2)方式二:

vue cli 脚手架工具下载配置

在脚手架当前目录下 npm i mint-ui -S

#以上指令下载安装

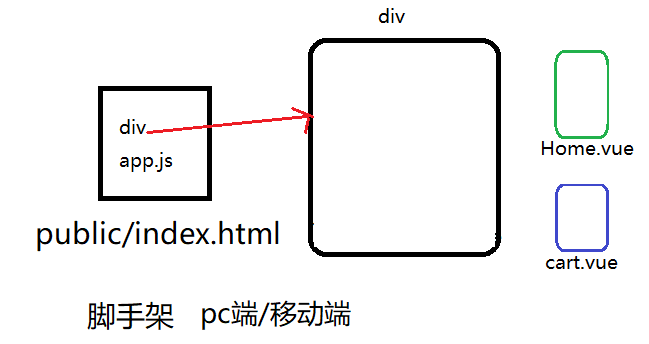
验证: node\_modules/mint-ui/lib 所有组件

package.json

启动脚手架指令

npm run serve

#其中 serve 可以配置package.json



3.2:组件库--引入mintui组件库

(1)引入方式一:按需引入

(2)引入方式二:完整引入(唯一正确)

-必须先下载 mint-ui库

-在main.js [引入其它库并且配置库]

import MintUI from "mint-ui" #完整引入mint-ui库

import "mint-ui/lib/style.css" #mint-ui 样式文件

Vue.use(MintUI);

#将mint-ui库所有组件注册vue实例中

#"mint-ui" node\_modules下目录名称

#MintUI 为组件起名

#只引mint-ui组件库库文件还有一个文件需要

#单独引入 mint-ui库有一个样式文件

3.3:组件库--调用mint-ui组件库第一个组件-Toast

短消息提供框:常用:删除成功;登录成功;用户名有误

基本语法: this.$toast({message:"",....});

参数

message:提示消息文本的内容

position:位置 top;bottom;middle

duration:持续时间(单位毫秒) 默认3000

className:类名为其添加样式

iconClass:图标类名

3.4:组件库--创建新组件

(1)创建空组件

src/components/exam/Exam01.vue

(2)为空组件指定路径

src/router.js path:'/Exam01'..

(3)运行组件:在浏览器地址栏中输入空组件地址回车

[http://127.0.0.1:8080/#](http://127.0.0.1:8080/#/)/Exam01

(4)添加内容再运行

3.5:组件库--图标--组件默认图标外观差-引入专业图标

https://www.iconfont.cn/

阿里旗下图标网站

(1)下载图标

-点击图片加入购物车

-点击购物车->点击[下载代码]按钮

-download.zip

#ttf 图标字体文件

(2)将图标应用当前项目

-在src目录下创建新目录font保存所有图标字体文件

-main.js 引入字体图标css文件

import "./font/iconfont.css"

-在组件中Exam01.vue 使用图标

this.$toast({

iconClass:"iconfont icon-food-sushi "

});

#注意:字体图标可以修改大小

#修改 src/font/iconfont.css

.iconfont {

font-family: "iconfont" !important;

font-size: 16px; #改变图标大小

}

3.6:组件库--交互式提示框

交互式提示框:是否删除数据

标准语法: this.$messagebox({})

title 标题

message 内容

showConfirmButton 是否显示确认按钮

showCancelButton 是否显示取消按钮

简写:

this.$messagebox("标题","内容");

this.$messagebox.alert("确认").then(action=>{})

this.$messagebox.confirm("确认执行操作").then(action=>{})

this.$messagebox.prompt("请输入年龄"

).then(({value,action})=>{})

示例:创建组件

-组件 /src/components/exam/Exam02.vue

-路径 /Exam02

-访问组件 http://127.0.0.1:8080/#/Exam02

3.7:组件库--表单输入域(编辑器)

<mt-field label="用户名" placehodler="请输入用户名"

v-model="uname"></mt-field>

type输入框类型 text/number/url/date/password

label 标签

placehodler 占位符

attr: 设置原生属性

...

官方网站: https://mint-ui.github.io/#!/zh-cn

示例:

(1)创建空组件src/components/exam/Exam03.vue

(2)配置路径 /Exam03

(3)http://127.0.0.1:8080/#/Exam03

常见错误

(1) Failed to compile 编译错误(拼写错误)

module was not found 组件没有找到

(2) tag <div> has no matching end tag.

div没有结束标签

(3) Uncaught TypeError: this.$toast is not a

main.js 写错了

作业:完成表单并且添加验证

# VUE-UI-02

今天学习的内容

1:复习昨天重点

2:今天的目标

2.1:mint-ui组件库-应用



2.2:mint-ui组件库-应用-分析功能

(1)顶部标题栏(完全自定义)

(2)面板(主体功能使用mint-ui面板组件)

(3)底部导航条(主体功能使用mint-ui tabbar组件)

2.3:mint-ui组件库-应用-目录



2.4:mint-ui组件库-应用-顶部标题栏



-创建子组件common/TitleBar.vue 完成以上功能

-左侧文字与右侧两张图片通过父组件 Home.vue 传入

-子组件负责接收父组件数据并且设计外观

-右侧两张图片触发点击事件处理函数也是由父元素传入

2.5:mint-ui组件库-应用-消息组件创建



-Message.vue

-分析结构:左侧图片,标题,子标题 右侧时间

2.6:mint-ui组件库-应用-消息列表组件

-MessageList.vue

-临时配置访问路径 /MessageList

-负责加载数据 weixin/json/messagelist.json

-依据数据创建循环

-调用Message.vue 子组件

解决小问题:

(1)如何动态读取文件中图片路径

(2)图片路径一定是在服务器端 示例: http://127.0.0.1:3000/1.jpg

<img :src="item.img" />

(3)本地图片路径报错

2.7:面板组件与底部导航条



作业1：将底部导航条中图片子组件

默认灰色图片,如果当前按钮被选中修改绿色

# VUE-UI-03

今天学习的内容

1:复习昨天的内容

常见错误:脚手架启动失败

npm run serve npm ERR! path D:\package.json -4058

原因:

(1):启动目录错误vue\_app\_00/npm run serve

(2):脚手架损坏现象 node\_modules

作业:

创建新组件

组位置:src/components/exam/Homework04.vue

路径: /Homework04

2:今天的目标

2.1:vue ui 组件库--表单

-开关组件

<mt-switch v-model="val1"></mt-switch>

val1:true 打开状态

val2:false 关闭状态

-单选框列表

<mt-radio

title="单选框列表"

v-model="val2"

:options="['选项a','选项b','选项c']"

>

</mt-radio>

options:列表项目内容

v-model:选中内容

-复选框列表

<mt-checklist

title="复选框列表"

v-model="val5"

:options="['选项a','选项b','选项c']"

>

</mt-checklist>

v-model="val5":数组

组位置:src/components/exam/Exam05.vue

路径: /Exam05

2.2:vue ui 组件库--面板-(重点)

父面板:最外层父面板

子面板:多个子面板中只有一个子面板显示状态

<mt-tab-container v-model="active"> 父面板

<mt-tab-container-item id="tab1"> 子面板

子面板1

</mt-tab-container-item>

<mt-tab-container-item id="tab2">

子面板2

</mt-tab-container-item>

<mt-tab-container-item id="tab3">

子面板3

</mt-tab-container-item>

</mt-tab-container>

注意:改变acitve值与子面板id一致则该子面板显示

组位置:src/components/exam/ExamContainer06.vue

路径: / ExamContainer06

注意事项:

(1)绑定事件:image;a;button;submit;img;video; 阻止默认行为

练习:

(1)在data保存数组商城三个选项["美食","购物","休闲"]

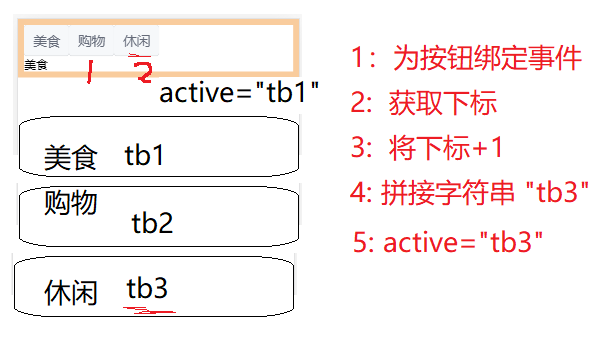
创建三个子面板对应三个选项

(2)依据数组创建三个按钮

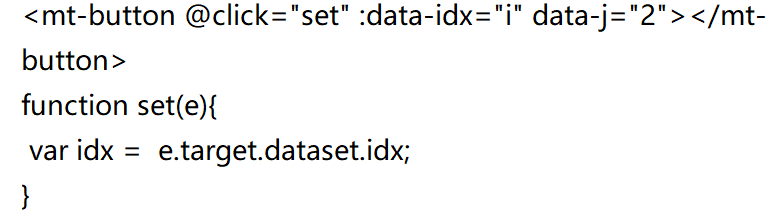
(3)点击按钮完成子面板切换

组位置:src/components/exam/ExamContainer07.vue

路径: /ExamContainer07



获取自定义属性



2.3:vue ui 组件库--tabbar(底部导航条)-面板-(重点)

|  |
| --- |
| <mt-tab-container v-model="selected">  <mt-tab-container-item id="外卖">  外卖面板1  </mt-tab-container-item>  <mt-tab-container-item id="订单">  订单面板2  </mt-tab-container-item>  </mt-tab-container> |

<mt-tabbar v-model="selected" fixed> 父元素

<mt-tab-item id="外卖"> 子按钮

<img slot="icon" src="../assets/1.jpg"></img>

外卖

</mt-tab-item>

<mt-tab-item id="订单">

<img slot="icon" src="../assets/2.jpg"></img>

订单

</mt-tab-item>

</mt-tabbar>

fixed:将导航导固定在底部

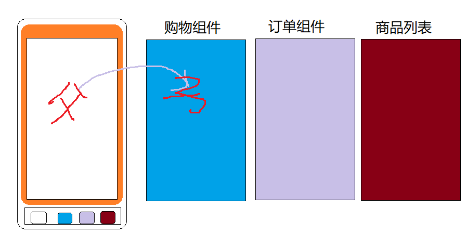
selected:当点按钮时将按钮id保存到此变量中

组位置:src/components/exam/ExamTabbar08.vue

路径: /ExamTabbar08.vue

复制图片:src/assets/....

2.3:父子组件传递



子组件:S09.vue

<h3>{{msg}}</h3> 父组件传 msg

<img :src="img\_url" /> 父组件传 img\_url

<button @click="add3"></button> 父组件传 add

<script>

export default {

props:{ //接收父组件数据

msg:{default:""}, //msg默认空字符串

img\_url:{default:""}, //img\_url默认空字符串

add3:{type:Function} //add 函数

}

}

</script>

父组件: F10.vue

<script>

//1:引入子组件文件

import S09 from "./S09.vue";

//2:注册子组件,起标签名

components:{

"s09": S09

}

myadd(){

}

</script>

3:调用子组件

<template>

<s09

msg="微信(11)"

:img\_url="require('../../asseets/01.png')"

:add3="myadd" >

</s09>

</template>

组位置:src/components/exam/S09.vue 子组件

组位置:src/components/exam/F10.vue 父组件

路径: /S09 不用添加

路径: /F10

作业1:复习vue 原理/组件

作业2:vue仿微信app 第一个按钮

# VUE-UI-04

今天学习的内容

1:复习昨天的重点

2:今天的目标

2.1:完成weixin--面板

<mt-tab-container v-model="active">

<mt-tab-container-item id="message">

<messagelist></messagelist>

</mt-tab-container-item>

</mt-tab-container>

2.2:完成weixin--底部导航条

<mt-tabbar v-model="active" fixed>

<mt-tab-item id="message">

图片:子组件

文字

</mt-tab-item>

</mt-tabbar>

2.3:完成weixin--底部导航条--按钮上显示图片子组件

TabBarIcon.vue

-focused:false 当前按钮是否是焦点

-selectedImage: 选中显示图片

-normalImage: 默认显示图片

2.4:修改tabbar默认样式

|  |
| --- |
| /\*修改tabbar 默认文字颜色\*/  .mint-tabbar>.mint-tab-item{  color:#999999;  }  /\*修改tabbar 选中文字颜色\*/  .mint-tabbar>.mint-tab-item.is-selected{  background-color:transparent;  color:#45c018;  } |

#注意：vue事件有一些修饰符 native:原生事件

如果自定义属性直使用事件绑定即可,如果调用组件库中组件

直接绑定失败情况,清加修饰符 native 启动原生事件绑定

父元素给子元素绑定事件 native

#注意:去除默认边框

-App.vue 根组件它的内补丁影响我们组件

-normalize.css->n.css 通用样式文件

public/index.html

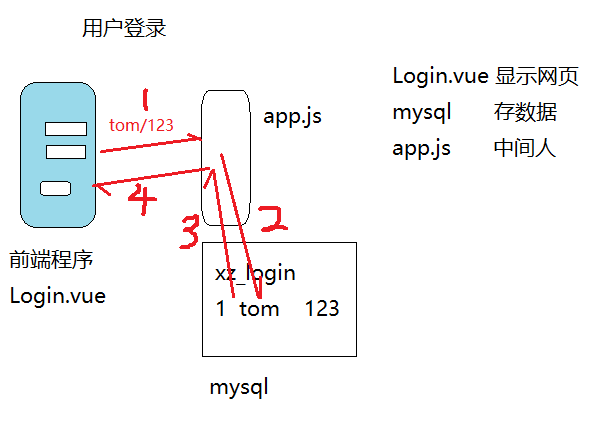
2.4:学子商城--登录--分析

用户打开登录网页输入用户名和密码点击提交按钮

-格式出错 提示:用户名格式不正确

-用户不存在 提示:用户名或密码不正确

-正确 跳转商品主页



2.5:学子商城--登录--[后端程序]服务器-数据库

-进入xz库 USE xz;

-创建表用户登录表

xz\_login [id 编号/uname 用户名/upwd 密码]

id INT PRIMARY KEY AUTO\_INCRMENET

uname VARCHAR(50),

upwd VARCHAR(32),

通用功能:密码内容需要加密后保存不是原文件保存

1 tom 123 原文

1 tom sd09234ksd90239epsd093kds0923 密文

为什么密码需要加密保存防止工作人员泄密

解决思路:将密码加密处理

解决方法:md5() 单向加密算法 加密结果32位密码

"123" => 202cb962ac59075b964b07152d234b70

***常规问题***:

-md5单向加密没有解密算法

-md5多次加密 md5(md5('123'))

-更有效方法强化用户密码难:

8位以上/数字/大写/小写/特殊符号

如何登录流程

(1)用户输入用户名和密码 tom/123将数据发app.js

(2)app.js 接收用户输入 tom/123

(3)app.js 将用户输入123加密与数据密码比较

SELECT id FROM xz\_login WHERE name = ?

AND upwd = md5(?);

数据库 xz\_login

1 tom 202cb962ac59075b964b07152d234b70

(4)pool.query(sql,[uname,upwd],(err,result)=>{

if(err)throw err;

if(result.length==0){

res.send({code:-1,msg:"用户名或密码错误"});

}else{

res.send({code:1,msg:"登录成功"});

}

})

-添加测试数据

2.6:学子商城--登录--[后端程序]服务器-app.js

(1)加载第三方模式

express web服务器

mysql mysql驱动模块

cors 跨域 脚手架8080 app.js 3000

express-session session

#下载 npm i mysql express express-session cors

#必须联网完成以上操作

(2)配置第三方模式

-mysql 创建连接池 [必须条件]

-配置跨域 cors 指定哪个程序可以跨域访问服务器

origin:["http://127.0.0.1:8080","http://localhost:8080"]

-配置session

secret:"安全字符串", #加密字符串

resave:true, #每次请求保存session数据

saveUninitialized:true #保存初始化数据

(3)处理用户登录任务

server.get("/login",(req,res)=>{

-参数 uname/upwd

-sql SELECT id FROM xz\_login WHERE uname = ?

AND upwd = md5(?)

-json {code:1,msg:"登录成功"}

});

测试:

(1)启动node app.js

(2)打开浏览器在地址栏输入

http://127.0.0.1:3000/login?uname=tom&upwd=123

按回车

http://127.0.0.1:3000/login?uname=tom&upwd=122

按回车

2.7:学子商城--登录--[前端程序]脚手架

-src/components/xz/Login.vue

-创建输入框 用户名 密码 提交按钮 (mint-ui)

-为提交按钮绑定点击事件

(1)获取用户名和密码

(2)正则表达式验证是否有误 显示短消息

(3)发送ajax请求

2.8:session-会话-(前端后端程序)

(1)session一个用户操作过程

-会话开始:当用户打开某个网站第一个网页

-会话中:中间用户可以打开此网站多个网页

-会话结束:当用户关闭浏览器

(2)session对象(在服务器为了保存此次会话专有数据对象)

session对象是保存当前会话中使用数据,在当前会话

中所有网页可以共享此数据，但是当会话结束session数据

失效

为什么使用session 对象,项目中有一些数据必须保存

session对象比如:当前登录用户编号 uid

解决问题:

作业1:完成学子商城用户登录组件

# VUE-UI-05

今天学习的内容

1:复习昨天的内容

2:作业

-session(会话) 会话指用户操作过程

(1)会话开始:用户打开某个网站第一个网页

(2)会话中:中间可以打开网站多个网页

(3)会话结束:当用户关闭浏览器

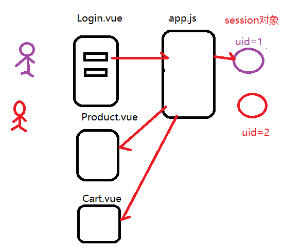
-session对象

session对象用来保存此次会话中所有网页共享数据

如果会话结束 session中保存数据丢失

将哪些数据保存session对象:uid 当前登录用户编号

#此编号是当前用户登录成功唯一凭证



-程序中如何使用session

(1)服务器node app.js

-引入express-session模块

const session = require("express-session")

-配置express-session模块

server.use(session({

secret:"128位字符串", #自定义字符串用于加密数据使用

resave:true, #每次请求更新数据

saveUninitialized:true #保存初始化数据

}))

-将数据保存session对象

req.session.uid = 2; #正确

req.session.id = 2; #错误

#为什么不能用id原因，session对象自己有一个属性id

-从session中获取数据

var uid = req.session.uid;

示例(1):

用户登录成功将uid保存session中,

用户获取购物中内容读取session中uid作为判断条件

- app.get("/login",(req,res)=>{

if(result.length > 0){

req.session.uid = 1;

}

})

-app.get("/cart",(req,res)=>{

var uid = req.session.uid;

if(!uid){

//没登录

}

})

(2)脚手架

main.js

1:配置axios发送ajax请求时保存session信息

axios.defaults.widthCredentials=true

2:配置axios基础路径

axios.defaults.baseURL = "http://127.0.0.1:3000/"

示例:登录

this.axios.get("login");

常见错误

-this.axios.get("http://127.0.0.1:3000/login");

-this.axios.get("http://localhost:3000/cart");

######如何调试ajax错误

(1)按F12->network 网络控制面板->

-登录操作

-出现新请求信息 login xhr 点击项目

-查询header 标签:检测出参数传递是否正确

-查询 response 标签:检测出返回结果是否正确

2.1：错误集锦

(1)脚手架启动错误 errno -4058

-启动目录没有脚手架程序

-node\_modules 出错造成启动

(2) http://127.0.0.1:8081/

原因:前一个脚手架没有停止 ctrl+c

(3) Uncaught (in promise) Error: Network Error

原因:node.js 出错停止工作 查询node.js控制台

(4) connect ECONNREFUSED 127.0.0.1:3306

原因:mysql停止工作

(5) Unknown column 'name' in 'where clause'

原因:未知列 name

sql语句中操作name列，但是表中没有此列

解决:sql语句错误/表没有不存在

(6) http://127.0... 404 (Not Found)

解决:请求程序地址错误 login

2.2:用户登录成功-跳转操作 /Home1

this.$router.push("组件路径");

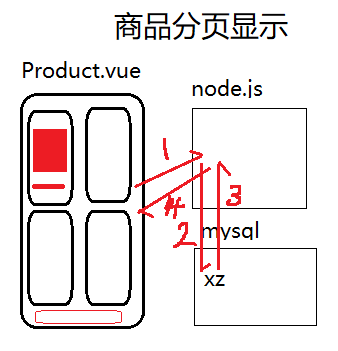
示例:

登录成功跳转 /Home1组件

this.$router.push("/Home1");

#创建空组件 /Home1

3.1:学子商城--商品列表-主页



3.2:学子商城--商品列表-数据库 xz\_laptop

(1)修改xz\_laptop 添加一列 img\_url

(2)更新 "01.jpg" "02.png"

(3)vue\_server\_00/public/img/01.jpg 02.png 图片保存目录

3.3:学子商城--商品列表-app.js

(1)指定静态目录 public

(2)分页显示商品信息

-参数 pno 页码 pageSize 页大小(每页几行)

默认值:pno:1 pageSize:4

-sql

(1)查询当前页内容

SELECT lid,price,title,img\_url FROM xz\_laptop

LIMIT ?,?

? 起始记录数 (pno-1)\*pageSize;

? 几行

示例:

第1页

SELECT lid,price,title,img\_url FROM xz\_laptop

LIMIT 0,4;

第2页

SELECT lid,price,title,img\_url FROM xz\_laptop

LIMIT 4,4;

(2)总页数

SELECT count(\*) as c FROM xz\_laptop;

Math.ceil(43/4)==11

-json

{code:1,msg:"查询成功",data:[],pageCount:11}

单页面/为什么嵌套/result.c

问题一：是单页面程序

脚手架打包命令: npm run build

生成单页面程序

生成目录 dist/

问题二:pool.query 为什么嵌套平行

var sql1

var sql2

pool.query(sql1);

pool.query(sql2);

http://127.0.0.1:3000/product

http://127.0.0.1:3000/product?pno=2

http://127.0.0.1:3000/product?pno=3&pageSize=5

3.4:学子商城--商品列表-Product.vue 显示商品列表

(1)创建Product.vue

(2)发送ajax请求获取分页数据 created()

/product

(3)获取服务器返回数据循环显示内容

(4)当用户点击 "加载更多" 显示下一页

作业1:商品列表页面创建

作业2:加载更多按钮

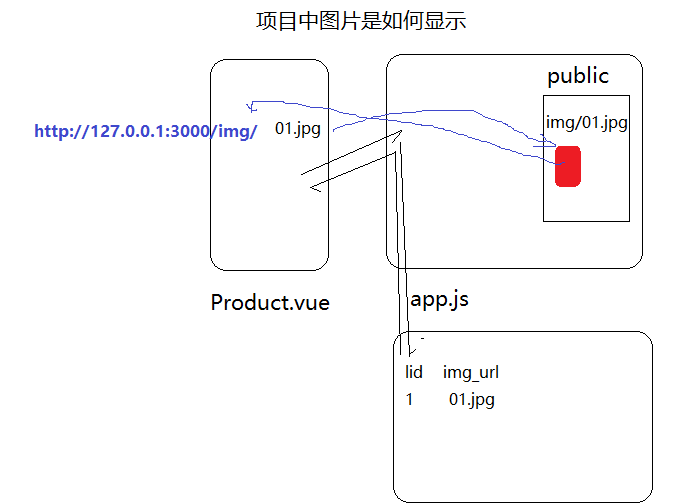
# VUE-UI-06

今天学习的内容

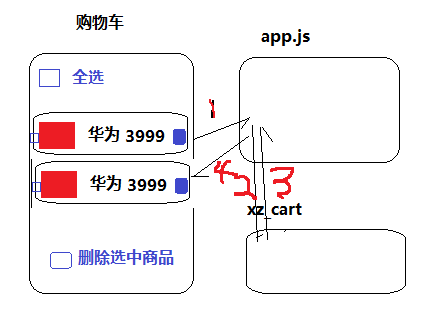
1:复习昨天的内容

2:今天的目标

2.1:昨天的项目完成 Product.vue



2.2:购物车--购物车--(重点)



2.3:购物车--购物车--数据库表

xz\_cart

id img\_url price count title uid

2.4:购物车--购物车--app.js

-查询指定用户购物车列表 tom/123 登录

(1)参数(不用脚手架传递参数) 1

var uid = req.session.uid;

(2)sql

SELECT id,img\_url,price,title FROM xz\_cart WHERE uid = ?

(3)json

{code:1,data:[..]}

示例:

(1)先登录

http://127.0.0.1:3000/login?uname=tom&upwd=123

别关闭浏览器

(2)查购物车

http://127.0.0.1:3000/cart

-删除指定购物车中一个商品

-app.js

参数:id

sql: DELETE FROM xz\_cart WHERE id = ?

json: {code:1,msg:"删除成功"}

测试：112~123

http://127.0.0.1:3000/delItem?id=4

-Cart.vue

-删除指定购物车中多个商品

-显示确认框 this.$messagebox("","");

作业1:购物车--购物车--Cart.vue-删除多个商品

分析:

(1)数据库 xz\_cart [id,img\_url,price,uid]

DELETE FROM xz\_cart WHERE id IN (6,7,8);

(2)app.js

参数: ids="5,6,7,9"

sql: DELETE FROM xz\_cart WHERE id IN (ids);

json {code:1,msg:"删除成功"}

(3)Cart.vue

<mt-button>删除选中商品</mt-button>

list:[{cb:true,id:1,..},{cb:false,}]

1,2,3

this.axios.get()

# HTML5-01

今天学习的内容

1:复习昨天的重点

2:今天的目标

2.1:购物车完成

-服务器端程序完成

测试

http://127.0.0.1:3000/delAll?ids=7,9

http://127.0.0.1:3000/delAll?ids=8

-<mt-button>删除指定数据</mt-button>

--判断选哪个商品list:[{cb:true,id:1},{cb:true,id:2},{cb:false,id:3}]

--拼接字符串1,2

--判断如果用户没有选中商品提示

2.2:vuex 作用

vuex是一个容器用于保存多个组件共享数据

当前组件data其中数据可以在当前组件中使用,其它组件

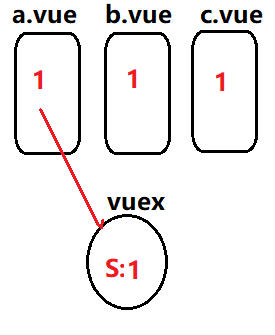
不能使用data中数据

解决问题:

(1)使用vuex多个组件共享使用数据

(2)当某个组件修改共享数据，

vuex通知所有使用此数据组件更新数据.保证所有组件使用数据状态统一.



2.3:vuex 使用方式

-脚手架下载vuex第三方模块 npm i vuex -S

结果:node\_modules/vuex/...

-main.js引入vuex

import Vuex from "vuex"

-main.js创建vuex实例对象

var store = new Vuex.Store({

state:{},

mutations:{},

getters:{}

})

state: 保存全局共享数据

mutations:修改全局共享数据函数

getters: 获取全局共享数据函数

-main.js将vuex注册vue实例中

Vue.use(vuex);

new Vue({strore})

-其它组件获取vuex数据 <template>

{{$store.getters.获取数据方法}}

-其它组件操作vuex数据 js

this.$store.commit("修改数据方法")

2.4:示例:将购物车中数量保存vuex

state:{cartCount:0} #购物车数量

mutations:{

increment(state){ #将购物车数量加1

state.cartCount++;

}

}

getters:{

getCartCount(state){

return state.cartCount; #获取购物车数量

}

}

<template>

<div>{{$store.getters. getCartCount }}</div>

</template>

2.5:html5新特性-(video/canvas绘图)-十个特性比散乱

2.6: html5新特性-表单新特性

(1)新input type <input type=? />

h4:text;password;checkbox;radio;submit;reset

h5:number;email;color;date;month;week

(2)新 element

h4:input;button;select;textarea;..

h5:datalist;progress;meter;output

-datalist 建议列表

<datalist id="list3">

<option value="xx">xx</option>

...

</datalist>

<input type="text" list="list3" />

#默认情况下datalist不可见

- progress 进度条

<progress></progress> 左右晃动进度条

<progress value="0.5"></progress> 具有指定值进度条

#练习:创建动态前进进度条

setInterval+progress

-meter 刻度尺(手机电池)

<meter min="最小值" max="最大值" value="当前"

low="下限" high="上限" optimum="最佳值">

</meter>

-output 输出

没有任何外观样式,外观 span

商品单价:￥3.5

商品数量:<input type="number" value="1" />

小计:<output>￥3.5</output>

(3)新 Attr <input ?>

h4:id;class;name;style;checked;readonly;

h5:

autofocus 自动获取焦点

placeholder占位符

form 用于把输入框放置在表单外部

<form id="f5"></form>

<input type="text" form="f5"/>

multiple 允许输入多值(用逗号分隔)

<input multiple type="email"

name="" value="a@a.com,b@a.com"/>

验证表单有关

maxlength;minlength;max;min;required;pattern;

-maxlength;minlength 字符串最大长度

-max;min 数字最大值与最小值

- required 不能为空

- pattern 正则表达式

2.7: html5新特性-视频--(重点)

基础知识

(1)视频文件格式 .mp4 .flv .webm .ogg ..

(2)解码器:如果浏览器播放指定视频，需要安装特定"解码器"

x.mp4 ->浏览器安装mp4解码器

x.flv ->浏览器没有安装flv解码器结果不能播放

解决:"格式工厂" x.flv->x.mp4->x.webm->x.ogg

标准语法

<video>

<source src="x.flv" />

<source src="x.mp4" />

<source src="x.webm" />

<source src="x.ogg" />

您的浏览器版本太低，请升级！！

</video>

简写:

<video src="x.mp4"></video>

2.8: html5新特性-视频--(重点)-常见方法

-常见属性

controls 显示播放控件 <video controls />

-方法:js 程序调用

-play(); 播放视频

-pause(); 暂停播放

-volume: 控制音量 (0~1)

-playbackRate: 播放速率 大于1快放 小于1慢播

-paused:true 当前视频是否处理暂停状态

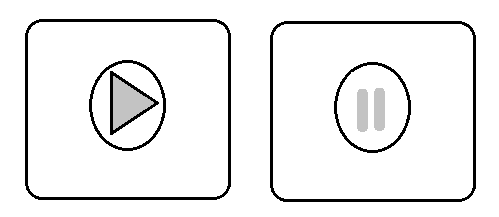
练习1:使用 video自带controls控制,自定义控件

完成视频播放和暂停功能 17:56

练习2:为视频绑定鼠标移入移出效果

鼠标移入视频区显示图片按钮 res/play.png

鼠标移出视频区隐藏图片按钮 res/pause.png



# HTML5-02

今天学习的内容

1:复习上周重点

2:今天的目标

2.1:html5新特性--视频音频

-常见属性

controls 是否显示播放控件 <video controls >

autoplay 是否自动播放 <video autoplay> 兼容性太差

loop 是否循环播放

muted 是否静态播放

poster 在播放视频之前是否显示广告图片

preload 预加载方式(策略)

none:不预加载任何数据

metadata:仅预加载元数据

(视频时长;视频第一个画面;视频高度宽度)

auto:预加载元数据，还加载一定时长视频

2.3:html5新特性--视频高级特性--事件

-canplaythrough 当前视频加载结束可以播放事件

duration 获取视频时长

-ended 视频播放结束触发事件

-timeupdate 视频正在播放

currentTime 当前播放时间点

2.4:html5新特性--视频高级特性--样式

video 样式 object-fit

fill: 默认 "填充" 将视频拉伸操作填满父元素(变形)

contain: "包含" 保持原有视频尺寸，父元素留有空白区域

conver: "覆盖" 保持原有视频尺寸, 宽度或高度至少有一

个与父元素一致(裁切)

2.5:html5新特性--音频

基础知识: x.mp3 x.wav ...

<audio src="x.mp3" id="a3"></audio>

特性1:默认audio不可见，如果添加controls属性可见

特性2:属性方法事件大多数与视频相同

练习:创建复选框实现控制背景音乐播放练习

当选中复选框播放背景音乐 a3.play()

当清空复选框暂停播放 a3.pause()

cb.change = function(){this.checked==true}

2.6:html5新特性--在网页上绘图--(重点)

-网页绘制三种技术

(1)svg: 2D矢量图 用线段组成图形

特点:可以无限放大或缩小不会失真,颜色不够丰富

(2)canvas:2D位图 用像素点组成图形

特点:不可以放大或缩小，颜色细腻

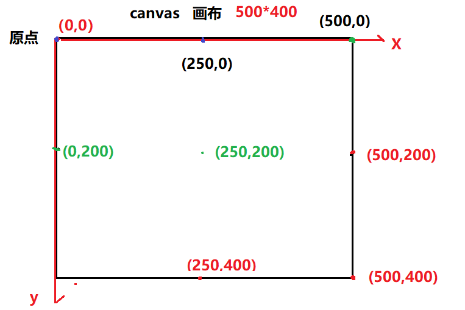
(3)webgl:3D位图

尚未纳入 H5标准

2.7:html5新特性--在网页上绘图--canvas

(重点-工作流程-坐标系-单词多)

-坐标系



-工作流程

(1)通过标签创建画布

<canvas id="c3" width="500" height="400"></canvas>

注意事项:画布宽度和高度只能通过js 或者标签属性来赋值

但是不能使用css赋值(出现变形)

(2)通过js程序获取画布

var c3 = document.getElementById("c3");

(3)通过画布元素获取画布中指定画笔对象[上下文对象]

var ctx = c3.getContext("2d");

#2d平面图形

-绘制矩形

ctx.strokeRect(x,y,w,h); 绘制空心矩形

x,y 矩形左上角位置

w,h 短形宽度和高度

ctx.strokeStyle = "#fff"; 设置空心边框样式

ctx.lineWidth = 1; 设置空心边框宽度

ctx.fillRect(x,y,w,h); 绘制实心矩形

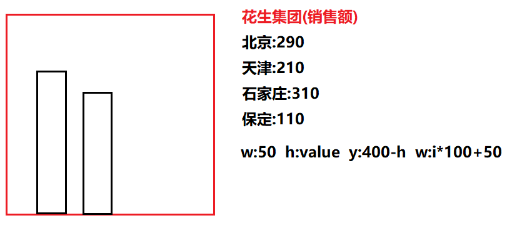
ctx.fillStyle = "#fff"; 设置实心填充样式

\*ctx.clearRect(x,y,w,h); 清除一个矩形范围内所有元素

练习1:绘制左右移动空心矩形

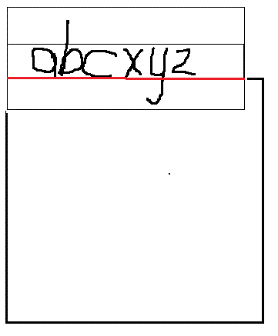
06\_rect\_exam.html

练习2:绘制柱状销售统计图



07\_rect\_exam.html

-绘制文字(字符串)



var str = "石家庄"

ctx.font = "19px SimHei"; 字体与大小

ctx.fillText(str,x,y); 绘制文本

ctx.textBaseline = ""; 指定文本基线

["alphabetic","top","bottom"]

str:绘制文本

x,y:文本左上角位置

作业1:薪水统计图[柱状统计图]

var sales = [

{m:"一月",s:3000},

{m:"二月",s:3200},

{m:"三月",s:3400},

...

{m:"十二月",s:5400}

];

# HTML5-03

今天学习的内容

1:复习昨天重点

2:今天的目标

2.1:html5新特性--canvas-路径(重点)

作用:路径作用绘制比较复杂图形

path:由任意多个点组成图形，图形本身不可见但是可以使用

"描边""填充"显示图形的效果

ctx.beginPath(); 开始一条新路径(上一条路径结束)

ctx.lineTo(x,y); 从当前点到指定点画一条直线

ctx.moveTo(x,y); 移动到指定点

ctx.stroke(); 描边

ctx.arc(cx,cy,r,start,end); 圆拱形

cx,cy 圆心

r 半径

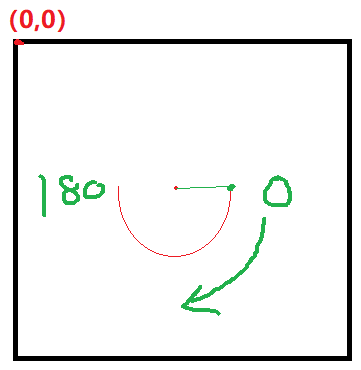
start 开始角度(弧度) 角度\*Math.PI/180=弧度

end 结束角度(弧度)

ctx.closePath(); 闭合一条路径

(从结束点到起始点画一条直线)

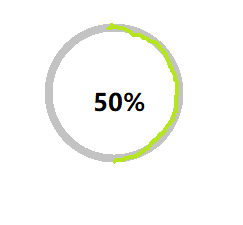
示例:半圆



练习1:使用圆环创建五环图形



练习2:使用圆环创建动态前进进度条



2.2:html5新特性--canvas-图片(重点)

图片在软件项目中放在哪里? 服务器指定目录中

原因

(1)图片有版权

(2)图片数量巨大

-开发流程序

(1)创建图片对象 var p3 = new Image();

(2)下载指定图片 p3.src = "x.png"

(3)为图片绑定下载完成事件 p3.onload = function(){}

(4)将事件绑定函数中绘制图片

ctx.drawImage(p3,x,y); x,y 左上角位置(原始大小图片)

ctx.drawImage(p3,x,y,w,h); w,h 宽和高(拉伸)

练习:绘制斜向小飞机

练习:绘制随鼠标移动而移动小飞机

2.3:html5新特性--canvas-变形(重点)

canvas绘图技术针对图像在绘制过程中进行变形(旋转)操作

-rotate(弧度)

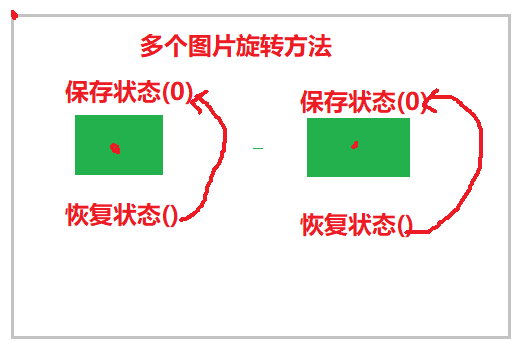
#旋转针对画笔对象,旋转轴心默认在画布原点上

#旋转角度会累加效果

练习:绘制不停旋转小飞机

定时器+旋转+绘制小飞机

-translate(x,y); 将画布轴点(原点)移到指定位置



-save() 保存状态

-restore() 恢复画笔状态

练习:创建二架飞机,飞机1飞机2

其中飞机2旋转速是是飞机1 ,2倍

手册:

https://developer.mozilla.org/zh-CN/docs/Web/API/Canvas\_API

2.4:html5新特性--canvas-变形(渐变对象)--了解

渐变对象:是一种图像效果从一种颜色慢慢过渡另一种颜色

(1)创建渐变对象

var g = ctx.createLinearGradient(x1,y1,x2,y2);

x1,y1 起点坐标

x2,y2 终点坐标

(2)添加颜色点

g.addColorStop(offset,color);

(3)将渐变对象赋值填充样式或者描边样式

ctx.fillStyle = g;

ctx.strokeStyle = g;

(4)绘图

ctx.fillRect(0,0,500,100);

作业1:

使用渐变对象完成销售统计图:

北京 上海

# HTML5-04

今天学习的内容

1:复习昨天的重点

2:作业

3:今天的目标

3.1:html5新特性--svg--(了解)

|  |  |  |
| --- | --- | --- |
|  | canvas | svg |
| 绘制图像类型 | 2D位图 | 2D矢量图 |
| 如何绘制 | 使用js代码绘图 | 标签绘图 |
| 事件绑定 | 只能将事件绑定画布 | 每个图形都可以绑定 |
| 应用场景 | 网页特效;游戏 | 地图 |

-开发流程

(1)创建画布

<svg id="s3" width="500" height="400">....</svg>

(2)向画布添加元素

<circle cx="" cy="" r="" fill="" stroke="" fill-opacity=""

stroke-opacity=""></circle>

#svg图形属性可以通过css设置

|  |
| --- |
| svg特性:  1:svg图形中属性是可以通过css设置  2:svg动态创建元素  (1)html字符串拼接  var html = "<circle></circle>";  svg.innerHTML = html;  (2)svg创建对象方法  var c = document.createElementNS("svg标准地址","标签");  #svg地址:http://www.w3.org/2000/svg  #标签:circle  c.setAttribute("r",100); //不能这样式 c.r = 100;  var r = c.getAttribute("r");  svg.appendChild(c); |

练习1:在画布上创建30个圆形要求大小，位置，颜色，透明度

随机 10"45

练习2:当点击某个圆时它会慢慢变大变淡最后消失,最后删除此

元素

3.2:html5新特性--svg--直线

<line x1 y1 x2 y2 stroke="" stroke-width="" stroke-linecap=""></line>

stroke 描边样式

stroke-width 线宽度

stroke-linecap 边线顶端样式 round

练习:汉堡包图标

3.3:html5新特性--svg-- 折线

一条折线上可以有任意多个连续点(x,y)

<polyline points="x,y x,y x,y ..." stroke=""></polyline>

练习:五角星

3.4:html5新特性--svg-- 文本/图像

<text font-size="" fill="" stroke="" x="" y="">文本内容</text>

<span></span> 不行

<image xlink:href="x.jpg" x="" y=""></image>

x:y 元素左上角

3.5:html5新特性--svg--渐变对象(一种特效)

<defs> 定义特效对象

<linearGradient id="g3" x1="" y1="" x2="" y2="">

<stop offset="偏移量" stop-color="" />

...

</linearGradient>

</defs>

#x1 y1 起点坐标 0% 100%

#x2 y2 终点坐标

#offset 颜色点位置 30%

<ANY fill="url(#g3)" stroke="url(#g3)"></ANY>

#ANY 任意标签 line;text;circle;...

3.6:html5新特性--svg--滤镜(一种特效)高斯模糊滤镜

<defs>

<filter id="f3">

<feGaussianBlur stdDeviation="数字" />

</filter>

</defs>

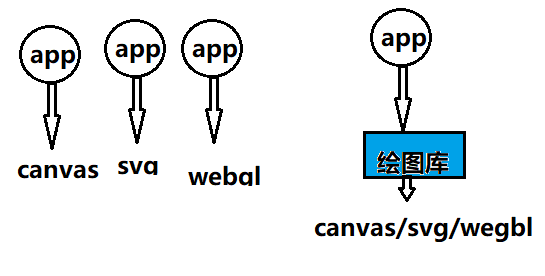
#数字:模糊度级别 0~10 7 看不清

<ANY filter="url(#g3)"></ANY>

3.7:html5新特性--echarts-百度第三方绘图库--(重点)

-d3.js 国外公司

-echarts 百度 https://www.echartsjs.com



#为什么要使用第三方绘图库

-绘图库提供非常复杂图形

-降低绘图难度

3.8:html5新特性--echarts-百度第三方绘图库--(重点)

(1)下载echarts库文件

https://www.echartsjs.com/download.html

(2)创建容器

<div id="main" style="width:500px;height:400px"></div>

(3)将echarts库引入网页中

<script src="echarts.js"></script>

(4)获取容器并且创建echarts对象 js

var main = document.getElementById("main");

var mychart = echarts.init(main);

(5)创建配置项(柱状统计图)

var option = {

title:{text:"echarts 入门示例"} #顶部标题

xAxis:{data:["衬衫","袜子","裤子","雪纺衫"]}, #x轴数据

yAxis:{}, #y轴数据(空)

series:[{type:"bar",data:[10,10,11,13]}] #数据源

}

type:"bar" 指绘图像类型 bar line pie .....

(6)将配置项添加 mychart对象即可 45

mychart.setOption(option);

练习:销售统计 08\_echart.html

{name:"冰箱",val:190},

{name:"电视",val:19},

{name:"洗衣机",val:390},

{name:"空调",val:90},

作业1:使用百度 echarts绘图库绘制饼状统计

# HTML5-05

今天学习的内容

1:复习昨天的内容

2:今天的目标

2.1:html5新特性--echarts-饼状图

{type:"pie",radius:"半径",center:["x","y"],data:[{value:12,name:"a"}]}

2.2:html5新特性--echarts-仪表图

var option = {

series:[{type:"gauge",

detail:{formatter:"{value}%"}, //显示进度 {32}%

data:[{value:32,name:"完成率"}]}]

}

2.3:html5新特性--拖放API

在网页实现拖放效果 Drag & Drop 拖动和释放

-拖动源对象(会动)-触发三个事件

dragstart 拖动开始

drag 拖动中

dragend 拖动结束

整个过程:dragstart\*1+drag\*n+dragend\*1

-拖动目标对象(不会动)-触发四个事件

dragenter 拖动进入

dragover 拖动悬停

dragleave 拖动离开

drop 释放

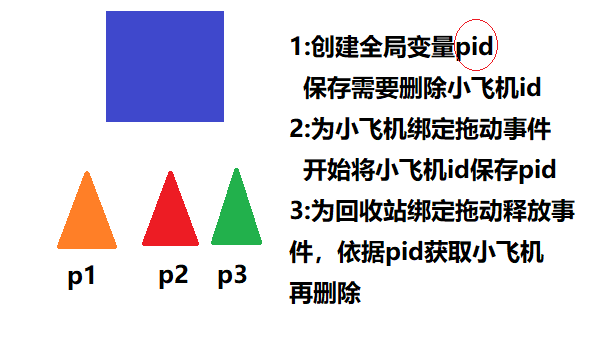
整个过程1:dragenter\*1+dragover\*n+dragleave\*1

整个过程2:dragenter\*1+dragover\*n+drop\*1

#dragover 默认行为:悬停结束后立即触发离开事件

#阻止事件默认行为

练习:使用拖动提供7个事件实现拖动删除效果



2.4:html5新特性--Web Worker(了解)-原理多代码3代码

程序:指可以被CPU执行代码，代码通常存储在磁盘上

1.html 1.css 1.js

进程:将程序(代码)调入内存中执行(在内存中代码称为进程)

线程:线程是进程内部组件部分

chrome浏览器线程模型

一个chrome进程内部至少7个线程:

-其中有一个线程负责所有资源绘制并且负责执行js(UI主线程)

-至少有6个线程负责向服务器发送请求获取资源(请求线程)

<button>按钮一</button>

<script src="1.js"></script>

<button>按钮二</button>

问题:如果1.js执行时间非常长，网页出现停止现象(空白)

解决方案:创建一个新线程(Worker)让它执行耗时js任务,

UI主线只负责绘制二个按钮

<button>按钮一</button>

<script>

new Worker("res/03.js");

#创建Worker对象

#创建新线程Worker执行03.js

</script>

<button>按钮二</button>

#注意事项:浏览器不允许Worker线程执行js程序中出现

任何DOM/BOM内容

#原因:浏览器开发者认为多个线程同时绘制网页容易出现

混乱现象.

总结:WebWorker 新线程用于执行耗时js任务

提高网页效率

js任务不能有DOM/BOM如果jquery.js不行

2.5:html5新特性--数据传递

Worker(03.js) 传递数据 UI(html) 线程

-Worker(03.js) 发送

postMessage(strmsg); //执行完毕

-UI(html) 接收

var w = new Worker("res/03.js");

w.onmessage = function(e){e.data}//执行完毕

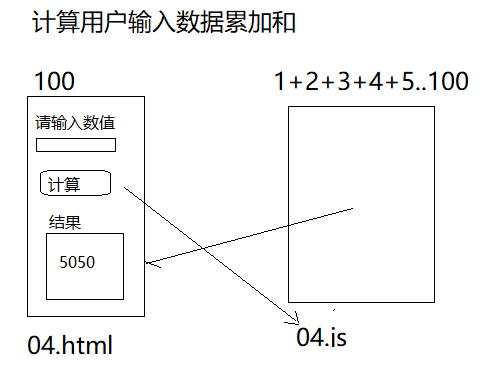
UI(html) 传递数据 Worker(03.js) 线程

-UI(html) 发送

w.postMessage(stringMsg)

-Worker 接收

onmessage = function(e){e.data}



创建04.html 04.js

2.6:html5新特性--Web Storage

在客户端浏览器中存储用户专用数据

示例:用户昵称;购物车

-客户端存储数据种类

(1)cookie 适用范围广 操作复杂 2KB

(2)webStorage 新 操作简单 8MB

(3)flash 依赖flash播放器

(4)indexDB 新 操作简单 没限

-webStorage两个对象

(1)sessionStorage 当前会话中多个网页可以共享数据

setItem(key,value); 保存数据 key 键 value 值

var value = getItem(key); 获取数据 value

removeItem(key); 删除一组数据

clear(); 清空所有数据

var key = key(i); 获取key

length 数据个数

(2)localStorage 保存数据永久

setItem(key,value); 保存数据 key 键 value 值

var value = getItem(key); 获取数据 value

removeItem(key); 删除一组数据

clear(); 清空所有数据

var key = key(i); 获取key

length 数据个数

#客户端保存数据要求:安全性差可以

作业2:登录与退出(模拟效果)

index.html 创建首页并且在右上角添加超链接 "请登录"

当用点击链接时跳转 login.html

login.html 创建表单请用户填写用户名和密码

点击提交:提示 3s秒后跳转首页

setTimeout+location.href

sessionStorage.setItem("uname","tom");

index.html 右上角 "欢迎回来:tom 退出"

点击 "退出"

logout.html 提示:您己退出 3s 后跳转 index.html

index.html 首页右上"登录"

作业1:销售统计图

(1)创建销售数据表

t\_sal[id:1,m:1,sal:100....] 12 行

(2)创建服务器程序 app.js

接收 get("/sal")将数据库销售表数据返回

(3)homework.html

1:发送ajax请求  原生ajax

2:依据返回数据创建销售统计图 echarts 柱状态

作业2：拖放上传图片功能(二天)-pc 端需要功能

# HTML5-06

今天学习的内容

1:复习昨天重点知识

2:作业

作业1:登录与退出

作业2:销售统计图[t\_sal/app/html]

3:今天目标

3.1:html5 webSocket 协议--原理(代码4行)

(1)http 协议

http协议:超文本传输协议

http协议:传输网页资源(html/css/img/avi/mp3/...)

网页内容搬运工

http://127.0.0.1:3000/index.html

http协议工作方式:请求和响应

#必须是客户端发起请求服务器端才会有响应数据

#而且一次请求只能获取一次响应数据

问题:有一些应用场景下此协议不适合

金股行业:[股票走势图] 多长时间发送一次请求

? 5分钟

? 10ms

解决问题:webSocket 协议

webSocket协议工作方式:广播和收听

客户端连接到服务器上就不再断开双方都可以随时向

对方发送数据

webSocket适用场景:股票走势图/群聊

3.2:html5 webSocket 服务器 提供现成

3.3:html5 webSocket 客户端 4行代码

-创建webSocket对象

var c = new WebSocket("ws://127.0.0.1:9001");

ws 协议名称

127.0.0.1 服务器在本地电脑上

9001 服务器默认工作端口

-接收服务器发送数据

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

-发送数据给服务器

c.send(stringMsg);

-关闭连续

c.close();

3.4:html5 游戏(大鱼和小鱼)--重点--功能分析

-蓝色大海

-紫色海葵左右摆动

-海葵头顶位置出生食物

-食物 蓝色(多)100 橙色(少) 200

-大鱼负责吃食物

-小鱼跟着大鱼

-大鱼负责喂小鱼

3.5:html5 游戏(大鱼和小鱼)--重点--目录

game 游戏目录

src 所有游戏中使用图片(复制图片)

js 所有js文件

index.html 唯一个网页

3.5:html5 游戏(大鱼和小鱼)功能一:背景 index.html

-创建画布标签

画布一:(前) 大鱼 小鱼 分数 z-index:1

画布二:(后) 背景 海葵 食物 z-index:0

为画布设置样式

游戏程序设计:一个角色一个文件

创建js game/js/background.js

创建drawBackground() 绘制背景图片

-main.js

作用:创建游戏所有全局变量

创建所有游戏中角色对象

绘制游戏角色

-requestAnimationFrame(gameloop);

以前定时器不适合创建动画效果智能定时器

定时器不停计算当前浏览器效率，得出最佳间隔时间

setInterval(fn,200); 固定时间200ms 20ms

固定时间 300ms 200ms

3.6:html5 游戏(大鱼和小鱼)功能二:海葵版本一(静态)

-创建文件 ane.js

-实现海葵依靠直线 路径 lineTo()

-一共50根海葵

-海葵间距 16 800/50

-海葵高度 200 50随机

3.7:html5 游戏(大鱼和小鱼)功能二:海葵版本二(动态)

3.x:html5 地理定位(小程序定位服务器 腾讯地图)

# HTML5-07

今天学习的内容

1:复习上周重点

2:今天的目标

2.1:大鱼和小鱼游戏

2.2:html5 游戏(大鱼和小鱼)功能二:海葵版本一(静态)

-创建文件 ane.js

-实现海葵依靠直线 路径 lineTo()

-一共50根海葵

-海葵间距 16 800/50

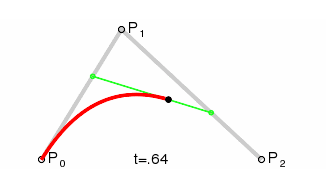
-海葵高度 200 50随机

2.3:html5 游戏(大鱼和小鱼)功能二:海葵版本二(动态)

功能分析:摆动海葵

解决方案:使用曲线替换直线完成海葵绘制

贝赛尔曲线:



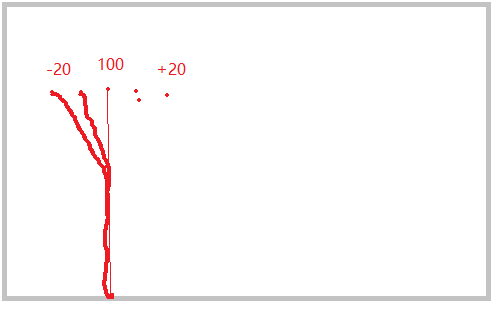
p0:起点坐标

p2:终点坐标

p1:控制点

曲线:从p0开始经过p1控制点最后绘制p2终点这样一条

曲线贝赛尔曲线



初始化:起点坐标 控制点坐标 终点坐标 一致

摆动:终点发生变化 -20 +20

贝赛尔曲线实现

moveTo(起点坐标x,起点坐标y)

quadraticCurveTo(控制点x,控制点y,终点x,终点y);

小知识: 正弦函数

-20 100 +20

右向100 +1 +2 +3 +4 +5 ..+20

左向 100 -1 -2 -3 -4 -5 ..-20

解决方案:正弦函数返回值 -1~1

var p = Math.sin(非常小小数 0.00032); p (-1~1)

2.4:html5 游戏(大鱼和小鱼)功能二:食物

(1)30个食物其中蓝色居多橙色很少 fruit.png blue.png

(2)同一个时刻15个食物显示15个隐藏

(3)食物行为:

出生:在海葵终点坐标位置出生

宽度+高度: 0~14

向上漂浮: y- spd

(4)食物漂浮出屏幕 隐藏

(5)食物被大鱼吃了 隐藏

2.5:html5 游戏(大鱼和小鱼)功能三:大鱼

(1)画布

can1:大鱼/分数/..

can2:背景/海葵/食物

(2)大鱼多张图片组件

眼睛:bigEye0.png ... bigEye1.png 二张眼睛图片

身体:bigSwim0.png .. bigSwim7.png 八张身体图片

尾巴:bigTail0.png .. bigTail7.png 八张尾巴图片

(3)大鱼行为

大鱼跟鼠标游动

大鱼吃食物

大鱼吃食物得分 蓝色食物100 橙色200分

作业1：

大鱼绘制操作

# WX-01

今天学习的内容

1:复习昨天重点

2:今天的目标

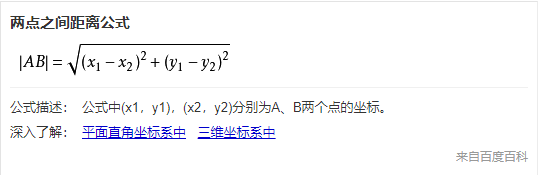
2.1:大鱼/大鱼吃食物/分数

2.2:大鱼吃食物

如果大鱼位置与食物位置非常接近表示大鱼吃到食物

如果食物消失

问题:大鱼与食物距离



collsion.js 完成大鱼与食物[碰撞检测]

2.3:分数

分数字符串我们将分数画在画布底端

1:吃蓝色食物100分

2:吃橙色食物200分

食物类型fruitType "blue","orange"

2.4:微信公众平台--小程序(重点)

开发界高手:张小龙

微信通讯产品:腾讯公司(超过10亿用户)

BATJ(百度/阿里/腾讯/京东)

微信公众平台(订阅号/小程序/小游戏)

订阅号:个人媒体平台(文章/音频/视频)

微信小程序:中小企业和个人(轻量级app)

2.5:微信公众平台--小程序(重点)-注册帐户

(1)微信公众平台

https://mp.weixin.qq.com/

(2)立即注册

订阅号 服务器 (\*)小程序 企业微信

(3)邮箱(没有在微信平台使用 订阅号 小程序 ...)

(4)注册主体类型

-企业和机构

(1)上传企业基本信息加盖公章图片清晰

(2)主体类型不能修改

(3)微信支付/卡包/附近小程序--面向企业

-个人

2.6:微信公众平台--小程序(重点)-登录

https://mp.weixin.qq.com/

(1)邮箱/密码

(2)左侧菜单:开发选项

(3)开发设置

(4)appid:小程序密钥(小程序身体证编号)

[wx2ffd16b52bc1d0db]

(5)左侧菜单:版本管理

开发版本-->审核版本->线上版本

2.7:微信公众平台--小程序(重点)-下载开发者工具

(1) 微信开发工具下载地址

https://developers.weixin.qq.com/miniprogram/dev/devtools/download.html

(2)安装下一步

(3)用手机微信扫二维进入工具

(4)创建项目



2.8:微信公众平台--小程序(重点)-工具

云开发:开通云开发->打开云开发管理界面

编译:将所有代码重新编译执行最新结果

预览:将代码上传微信服务器,使用手机微信扫描看到程序

结果

上传:将项目上传微信服务器[开发版本]

详情:项目详细

(1)调试基础库 2.2.5

注意事项:每一个小程序项目不能超过 2MB

2.9:微信公众平台--小程序(重点)-目录结构



2.10:微信公众平台--小程序(重点)-如何创建页面

exam01/

exam01.wxml 类似html

exam01.wxss 类似css

exam01.json 配置文件

exam01.js 类似js

2.11:微信公众平台--小程序(重点)-wxml

小程序wxml 微信xml

#严格匹配置开始结束标签

常见标签

view 相当于<div> #块级元素

text 相当于<span> #行内元素

image 相当于<img src="" />

form 相当于<form></form>

...

常见错误:

(1)expect end-tag `text` 缺少结束标签

(2)模拟器没有任何内容 关闭网络防火墙

(3)输入中文不成功 关闭开发工具再启动

(1)文本标签

<text selectable="true"></text>

selectable="true" 文本的内容可以复制

(2)图片

<image src="1.jpg" lazy-load='true'></image>

lazy-load='true' 懒加载:等待图片下载成功后再显示

图片内容

图片默认320\*240 图片

2.12:微信公众平台--小程序(重点)-wxss

wxss 是一套用于小程序的样式语言,功能描述组件

(1)单位:rpx(responsive pixel)响应式像素,可以根据屏幕

宽度自适应调用图片文字容器大小

(2)引入外部样式文件 @import "./base.wxss"

2.13:微信公众平台--小程序(重点)-rpx

问题:移动端开发你使用哪个像素

-前端设计师:设计稿图片 物理像素

-前端开发:css 逻辑像素

示例:iphone6(2)/iphone plus(3)

逻辑像素 \* dpr = 物理像素

iphon6 375px\*2=750

iphon6plus ?

小程序解决方案: rpx 响应式像素

(iphone6) 750rpx 逻辑像素 === 750 物理像素

2.14:微信公众平台--小程序(重点)-引入其它样式文件

#将项目公众样式内容统一保存在一个文件中

style/common.wxss

exam01.wxss

@import "../../style/common/wxss"

#引入其它样式文件一定用相对路径

#小程序开发文档

2.15:微信公众平台--小程序(重点)-第三方组件库(样式库)

-小程序开发文档

https://developers.weixin.qq.com/miniprogram/dev/framework/

-Vant Weapp -有赞团队开发样式(项目)

https://youzan.github.io/vant-weapp/#/ 小程序

-WeUI

-iView Weapp

2.16:微信公众平台--小程序(重点)-配置文件

project.config.json 项目配置文件

app.json 项目全局配置文件

exam01.json 组件配置文件

--app.json 全局配置文件

#如果一个选项添加app.json 所有组件立即生效

"pages":[] 组件列表

[

"pages/exam01/exam01", #默认显示第一个组件

"pages/exam02/exam02"

]

"window":{} 全局窗口配置

"navigationBarBackgroundColor": "#F00", 导航条背景

"navigationBarTitleText": "学子商城", 导航条文本

"navigationBarTextStyle": "black" 导航条文本颜色

"tabbar":{} 底部导航栏

"tabBar": {

"list": [ 按钮列表

{ 一个按钮

"pagePath": "pagePath", 点击按钮跳转组件

"text": "text", 按钮文本

"iconPath": "iconPath", 按钮默认图标

"selectedIconPath": "selectedIconPath"按钮选中图标

}]

}

#注意事项:图标不能使用网络路径只有本地路径

#注意事项:按钮最少二个

作业1:

创建1个组件 pages/cloud/cloud 云服务

# WX-02

今天学习的内容

1:复习昨天重点

2:今天的目标

3.1:微信公众平台--小程序--js/事件

-小程序中js与网页程序中js有区别

(1)ECMA 基本类型:string;number;boolean;undefined;null

(2)ECMA 对象:Date;Math;RegExp;Array;Function;...

(3)DOM/BOM 不能使用

(4)小程序顶级对象 wx 相当于网页js中 window

#通过小程序文档 [API]查看所有组件

-js特殊文件

(1) 全局js app.js 全局共享数据

(2) 当前组件js base.js 当前组件使用

Page({}) 当前组件对象包括[事件/方法/data]

如何显示data中数据 {{msg}}

3.2:微信公众平台--小程序--js/事件-数据显示控制

(1)循环

<view wx:for="{{数组名称}}" wx:key="index">

{{item}} 当前对象

{{index}} 当前元素下标

</view>

#wx:for="{{循环数组名称}}"

#wx:key="index" 数组元素下标(排序规则) 快速排序

(2)if

<view wx:if="{{condition}}">xxx</view>

condition:表达式 true false

true 当前元素正常显示

false 当前元素删除

(3)if else

<view wx:if="{{condition}}">111</view>

<view wx:elif="{{condition}}">222</view>

<view wx:else>333</view>

(4)hidden

<view hidden="{{condition}}">内容</view>

condition:表达式 true false

true隐藏 false 显示

问题:什么时候使用if 什么时候使用hidden

(1)如果此元素需要频繁切换使用hidden

(2)如果此元素在运行中不大可能改变则 wx:if 较好

3.3:微信公众平台--小程序--js/事件-事件

pc端项目 屏幕宽度 > 970px 操作:鼠标键盘

移动端项目 屏幕宽度 < 970px 操作:手指

移动端项目:

-touchstart 当手指碰屏幕时发生事件不管几个手指

-touchmove 当手指在屏幕上滑动边续触发

-touchend 当手指离开屏幕时触发

zepto.js - 相当于 移动端 jquery

tap 触碰一次

longtap 长按一次 超过350ms

swipe 滑动

swipeLeft 左滑

..

3.4:微信公众平台--小程序--js/事件-事件

小程序事件分为二种

(1)冒泡事件:当一个组件上事件被触发后，该事件向父元素传递

(2)非冒泡事件:当一个组件上事件被触发后，不向父元素传递

小程序绑定事件方式

-支持事件冒泡

<view bind事件名="事件处理函数"></view>

-不支持事件冒泡

<view catch事件名="事件处理函数"></view>

3.5:微信公众平台--小程序--生命周期(组件)

组件生命周期:一个组件从创建开始到使用最后销毁过程

onLoad :组件创建成功后触发一次

(1)发送ajax请求

(2)获取传递参数 onLoad(options)

onReady :组件渲染成功后触发一次

onShow :显示组件 多次

onHide :隐藏组件 多次

onUnload :销毁 一次

练习1:创建组件 pages/admin/admin

2:admin默认显示

3:admin添加按钮 <button></button>

3.6:微信公众平台--小程序--特殊事件

(1) onPullDownRefresh 用户下拉操作[刷新操作]

(2) onReachBottom 用户上拉触顶[下一页]

#默认小程序禁止用户下拉操作

#修改默认行为

-全局修改: app.json

window:{

"enablePullDownRefresh": false #禁止用户下拉操作

}

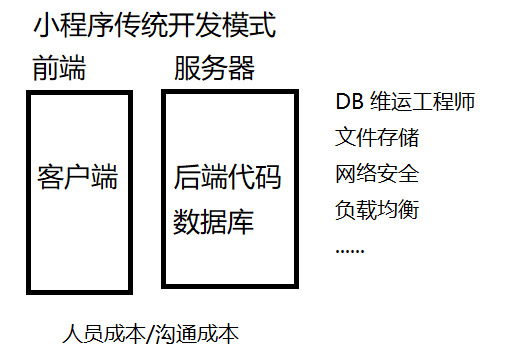
-当前组件修改: base.json

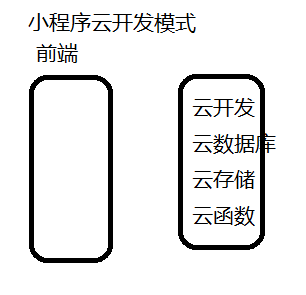
"enablePullDownRefresh": true

4:微信公众平台--小程序--云开发

小程序与腾讯云合作新方案[云开发]

云开发==[云函数+云数据库+云存储]





4.1:微信公众平台--小程序--云开发三个基础内容

-云数据库:数据增加/删除/修改/查询....

-云存储:上传文件/下载文件/分享文件/管理文件....

-云函数:获取appid/调用高级权限操作

4.2:微信公众平台--小程序--开通云开发

小程序开发工具-->"云开发" 开通

环境名称:[web-test-01]英文数字-

环境id:[] 自动生成环境复制下来

#免费环境

-5GB 云数据库

-2GB 云存储

-CDN 流程 5GB/月

#数据加载失败...

#退出云开发工具，再进一次

4.3:微信公众平台--小程序--云数据库(mongodb nosql数据库)

云开发提供一个JSON数据库提供2GB免费空间

|  |  |
| --- | --- |
| 关系型数据库 | 文档数据库 |
| 数据库 database | 数据库 database |
| 表 table | 集合 collection |
| 行 row | 记录 record/doc |
| 列 column | 字段 field |

#关系型数据库:适合复杂数据关系 企业内部软件系统

#文档型数据库:数据关系简单 频繁查询,更新 app

|  |  |
| --- | --- |
| 数据类型 |  |
| String 字符串 | Number数字 |
| Object 对象 | Array数组 |
| Bool 布尔 | GeoPoint 地理位置点 |
| Date 时间(客户端) | NULL |

4.4:微信公众平台--小程序--云数据库操作

(1)控制台: web1903

(2)云函数:

(3)小程序操作:

4.5:微信公众平台--小程序操作云数据库--添加

(1)通过控制台:创建集合 web1903

(2)初始化默认云数据库

const db = wx.cloud.database();

(3)向集合中添加新记录

db.collection("集合名称").add(

{

data:{name:"文华",age:60},

success:function(res){},

fail:function(err){}

}

)

add() 向集合中添加数据

data:{} 数据

success 添加成功回调函数

fail 添加失败回调函数

练习:创建新组件 pages/db/db

常见错误:

(1) -502005 database collection not exists

#集合不存在 web1903

原因1:拼写错误db.collection("拼写错误")

原因2:没有集合

(2)如果创建二个环境

const db = wx.cloud.database({

env:"环境id"

});

env:环境

(3) errCode: -501005 invalid env

原因:env:"环境id错误"

练习1:控制台创建集合web1903user

添加pages/db/db/ button

完成向web1903user集合添加一条记录

4.6:微信公众平台--小程序操作云数据库--更新

db.collection("集合名称").doc("当前记录id").update({

data:{

属性名:值

}

}).then(res=>{

console.log(res);

}).catch(err=>{

console.log(err);

})

doc() #当前记录id

update #更新数据

练习1:更新web1903集合数据 文华 age

4.7:微信公众平台--小程序操作云数据库--删除

db.collection("集合名称").doc(记录id)

.remove().then(res=>{}).catch(err=>{})

#注意事项:通过小程序一次只能删除一条数据

作业:

(1)创建集合 web1903emp [员工集合] 控制面板

(2)通过小程序

2.1:向集合添加员工信息

{no:1,name:"tom",sal:3000}

{no:2,name:"jerry",sal:7000}

{no:3,name:"kaka",sal:6000}

2.2:更新第一条记录sal 3500

2.3:删除最后一条记录

# WX-03

今天学习的内容

1:复习昨天的内容

2:今天的目标

2.1:作业

(1)创建集合 web1903emp [员工集合] 控制面板

(2)通过小程序

-:向集合添加员工信息

{no:1,name:"tom",sal:3000}

{no:2,name:"jerry",sal:7000}

{no:3,name:"kaka",sal:6000}

-:更新第一条记录sal 3500

-:删除最后一条记录

2.2:小程序云开发-云数据库--查询

(1)查询所有数据

db.collection("集合名称").get().then(res=>{}).catch(err=>{})

get:获取查询

then:查询成功后获取返回内容

res 查询结果

catch:查询失败

err 失败原因

(2)查询指定数据

db.collection("集合名称").where({name:"jerry"})

.get().then(res=>{}).catch(err=>{})

where查询条件

get 获取查询

then 查询成功后获取返回内容

res 查询结果

练习:实现web1903emp集合查询与删除操作

1:当组件创建成功后查询web1903emp集合中所有记录

2:将记录显示网页模板

data:{list:[]}

在小程序中将数据保存data有一个专用方法

this.setData({

属性名:新值

});

3:在网页模块每一条记录加一个按钮 <button></button>

4:为button绑定点击事件/删除指定记录

2.2:小程序云开发-云函数

云函数:特殊函数将其保存小程序云上(腾讯云)

云函数从功能很多优点:权限高完成复杂操作

演示三个示例

(1)求和云函数 sum()

(2)获取当前登录用户 openid

(3)批量删除云数据库中数据(ES7 语法)

2.3:小程序云开发-云函数-求和函数 sum

#注意事项:开发云函数要求本地下载安装nodejs v8.0 以上

(1)开发工具 coludfunctions

鼠标右键->新建node.js云函数

(2)云函数名称 sumc

(3)打开index.js

(4)添加云函数代码

exports.main = async (event,context)=>{

return {

sumc:event.i + event.j

}

}

exports 导出 event 事件对象(获取用户参数)

main 主函数 context 上下文对象(获取用户信息)

async 导步(ES7语法)

(5)上传部署

创建并部署,云端安装依赖

(6)测试云函数

(7)在小程序中调用开发云函数

wx.cloud.callFunction({ //调用云函数

name:"云函数名", //云函数名

data:{i:1,j:2} //参数

}).then(res=>{ //调用成功

console.log(res);

}).catch(err=>{ //调用失败

console.log(err);

})

#注意事项:如果你旧版本小程序工具没有server-sdk报错

#右击 countFunctions 在终端中打开

#npm install --save wx-server-sdk@latest

常见错误

(1) Error: errCode: -404011 cloud function execution error

原因:

-调用云函数不存在

-调用去函数没有上传

-云函数名子拼写错误

2.4:小程序云开发-云函数-login 登录

调用此云函数返回很多登录用户信息

openid

头像

性别

....

2.5:小程序云开发-云函数-批量删除 batchDelete03

云数据库web1903emp 同名记录 name:"kaka"

云函数目的:删除name:"kaka"

async:异步ES7 异步执行函数

await:等待如果执行耗时任务启动关键字等待任务

执行完毕

exports.main = async (event,context)=>{

try{

return await db.collection("web1903emp").where({

name:"kaka"

}).remove();

}catch(e){

console.log(e);

}

}

2.6:小程序云开发-云存储-5GB

(1)用户上传文件流程

-用户选择相册或拍照

-小程序上传所选中图片

-云存储返回图片 fileID //在云存储图片地址

(2)开发流程

-选择图片

wx.chooseImage({

count:1 //一次选择几张图片默认 9

sizeType:["original","compressed"]

//选中图片类型 (原图/压缩)

sourceType:["album","camera"]

//图片来源 (相册/相机)

success:function(res){ //选中图片成功回调

res.tempFilePaths //选中图片地址

}

});

-上传图片

wx.cloud.uploadFile();

cloudPath: 上传成功后新文件名

filePath: 选中图片名

success:res=>{ 上传成功

console.log(res.fileID);

上传成功后返回指定文件路径

}

作业1:文件上传与显示功能

\*-云数据库创建集合 myphoto 目标:保存上传文件fileID

\*-创建组件 pages/myphoto/myphoto

\*-添加按钮 "上传图片"

\* #将上传文件fileID保存myphoto中

\*-创建按钮 "显示图片"

?-获取集合中myphoto 创建循环遍历 fileID

<image src="fileID"></image>

# WX-04

今天学习的内容

1:复习昨天重点

2:今天的目标

2.1:作业:文件上传与显示功能

\*云数据库创建集合 myphoto 目标:保存上传文件fileID

\*创建组件 pages/myphoto/myphoto

\*添加按钮 "上传图片"

\*将上传文件fileID保存myphoto中

\*创建按钮 "显示图片"

\*获取集合中myphoto 创建循环遍历 fileID

<image src="{{item.fileID}} "></image>

2.2:小程序--学子商城项目-下载安装vant 第三方库

有赞团为vant网站

https://youzan.github.io/vant-weapp

(1)创建项目描述文件 package.json

右击 miniprogram 目录->选中在终端中打开

输入命令 npm init 回车 回车 ..

#npm init 功能生成项目描述文件 package.json

#执行成功后生成 package.json文件

(2)通过npm 安装vant 第三方组件库

右击 miniprograme 目录->选中在终端中打开

输入命令 npm i vant-weapp -S --production 回车

#安装成功看到文件夹 node\_modules

#右击miniprograme 在硬盘打开

(3)点击工具菜单-->构建npm

#将node\_modues内容复制一份 miniprogram\_npm

#常见错误 module .. not defined

#请重新登录

解决:点击工具菜单->构建npm

(4)点击工具菜单->详情

[\*] 使用npm 模块

(5)在组件中配置要使用哪个第三方组件

"usingComponents": {

"van-button": "vant-weapp/button/index"

}

#注意:组件路径修改与miniprogram\_npm

下目录结构相同

(6)在当前组件中使用button按钮

<van-button type="default">默认按钮</van-button>

2.3:小程序--学子商城项目-娱乐

(1)电影列表从"豆瓣网"下载最新电影列表

豆瓣网热映电影列表

http://api.douban.com/v2/movie/in\_theaters....

第一个部分:接口地址

第二个部分:apikey=密钥(别人开发者密钥)

第三个部分:start 第几条记录开始

count 本次查询几条记录

(2)如何获取"豆瓣网"热映电影列表-发送ajax请求

2.1:小程序有二种请求数据方式

|  |  |  |
| --- | --- | --- |
|  | 小程序端 | 云函数 |
| 发送方法 | wx.request({}) | 下载第三方ajax 库 (request) |
| 协议支持 | 只支持https | 根据第三方库决定 |
| 是否备案 | 经过ICP备案 | 可以不备案 |
| 域名个数 | 20个 | 无限制 |
|  |  |  |

2.2:开发云函数获取"豆瓣"电影列表

-创建云函数 movielist3

-点击movielist3 鼠标右键->终端打开

#注意下面命令有顺序

npm install --save request

npm install --save request-promise

#此次请求使用ajax库 request-promise

#上面库是request-promise 依赖库

2.3:开发云函数

(1)引入 request-promise 库

var rp = require("request-promise");

(2)导出创建main函数

exports.main = async (event,context)=>{}

event:事件对象保存请求参数

context:上下文对象 用户openid appid..

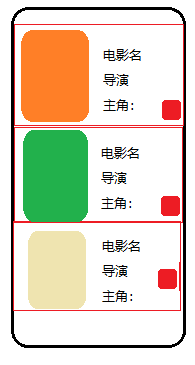
(3)创建url 请求地址

var url = `http://api..&start=${event.start}

&count=${event.count}`;

(4)rp(url) 返回查询结果

2.4:学子商城--娱乐首页 15:50



首页功能：

(1)打开首页组件时查看第一页电影

(2)向上滑动屏幕查看下一页

(3)点击详细按钮跳转电影详细组件显示内容

(4)电影图片/电影名称/电影导演/电影主角...

2.5:学子商城--娱乐首页-设计首页布局

2.6:学子商城--娱乐首页-添加样式

2.7:学子商城--娱乐首页-js实现以上功能

(1)功能一:打开首页组件时查看第一页电影

-onLoad

-loadMore(){} 加载更多

在此函数发送请求获取第一页数据

--调用云函数movielist3

--调用云函数时传参数 start:10 count:10

--保存data list

# WX-05

WX-05  
今天学习的内容

1:复习上周重点

2:今天的目标

2.1:学子商城--娱乐首页完成

-打开首页组件时查看第一页电影

-电影图片/电影名称/电影评分/电影主角... OK

-向上滑动屏幕查看下一页

-

向上滑动 onReachBottom上拉触底

常见错误:

(1) -404011 cloud function execution error

原因:云函数调用失败

a云函数上传失败

b 依赖库request下载

c 切换云环境

解决:

a,b检查你本地云函数拼写正确,重新下载request

打开云管理面板删除原先云函数再上传

c app.js 配置云环境使用哪个

-点击详细按钮跳转电影详细组件显示内容

目标:从home组件(关闭并且跳转) comment 评论组件

wx.redirectTo({

url:"/pages/comment/comment"

})

目标:从home组件(保留并且跳转) comment 评论组件

wx.navigateTo({

url:"/pages/comment/comment"

})

2.2:学子商城--娱乐详情完成

功能列表:

(1)电影图片

(2)评论功能

(2.1)--评论内容[vant]

(2.2)--打分 [vant]

(2.3)--上传图片[上传多张图片]

2.3:学子商城--娱乐详情完成-准备

(1)创建组件 comment

(2)创建云函数getDetail3 获取某部电影详细信息

(3)下载安装两个依据ajax

npm install --save request #依赖库

npm install --save request-promise #请求库

#每个云函安装自己所需依赖

(4)检查url 豆瓣电影详情 url

http://api.douban.com/v2/movie/subject/26794435?apikey=0df993c66c0c636e29ecbb5344252a4a

(5)上传部署云函数

(6)云测试

2.4:学子商城--娱乐详情完成-开发(第三方组件库 vant)

https://youzan.github.io/vant-weapp/ 文档手册

(1)输入框

(2)打分

(3)上传按钮

2.5:学子商城--娱乐详情完成-输入框

(1)查看手机

(2)打开配置文件comment.json

#删除多余目录 path/to/dist

{

"usingComponents": {

"van-field": "vant-weapp/field/index",

"van-button": "vant-weapp/button/index",

"van-rate": "vant-weapp/rate/index"

}

}

(3)在模板中使用组件 comment.wxml

<van-field

value="{{ value }}" //用户输入内容

placeholder="请输入用户名" //占位符

bind:change="onChange" //输入事件

/>

(4)在模板中使用组件 comment.js

Page({

data: {

value: ''

},

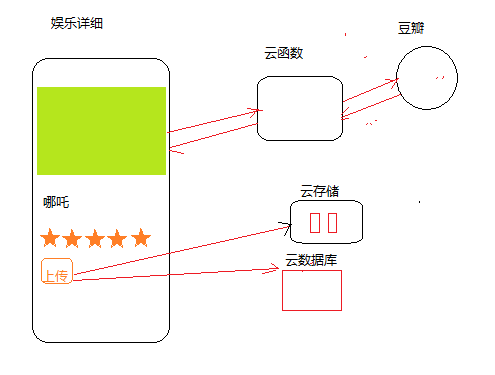
onChange(event) {

console.log(event.detail);

}

});

2.6:学子商城--娱乐详情完成-设计外观--comment.wxml



2.7:学子商城--娱乐详情完成-

功能1: 当组件加载成功显示电影详细信息

1.1:data添加属性 movieid:0 保存电影id

1.2:data 添加属性detail:{} 保存电影信息

1.3:添加方法 loadMore 组件创建成功调用

1.4:loadMore 调用云函数 getDetail4 参数id:电影

将云函数返回保存detail

1.5:在模板中显示detail对象中数据

作业1:loadMore

作业2:上传多张图片

2.1:上传图片 异步 9 先完成

2.2:将上传图片fileId 保存云数据库 异步 1 等待

# WX-06

今天学习的内容

1:复习昨天重点

2.今天的目标

2.1:学子商城评论(重点/难点)

常见错误

(1)JSON 0 位置错误

原因:服务器返回数据不正确不是严格JSON字符串

解决:(1)网络太差中间丢数据现象

(2)云函数写错误

"rating": {"max": 10, "average": 8.7 正确

appid: event 云函数错误

2.2:学子商城评论-上传图片[选择图片](重点/难点)

功能:一次请用户选择9张图片,

用户将图片保存data:{images:[]}

在按钮下预览选中9张图片

wx.chooseImage({})

count:9,

sizeType:[],

sourceType:[],

success:(res=>{});

2.3:学子商城评论-上传图片[上传图片](重点/难点)

-任务1：将data中选中图片上传云存储中

-任务2: 将图片fileID 与用户评论评分保存云数据库

问题:完成任务1与任务2遇到异步操作

[异步操作:任务1与任务2交替无序执行]

解决:ES6 Promise

(1)创建数组中元素Promise对象,每对象完成上传图片操作

Promise(reslove,reject)=>{

上传图片(新文件名,上传,将上传成功后fileID保存数组)

reslove();

}

(2)等待数组中所有Promise对象执行完毕

#等待list数组中所有Promise 对象执行完毕后，执行回调函数

Promise.all(list).then(res=>{

一次性将fileID保存云数据库集中

})

(3)将上传所有图片fileID一次保存云存储中

2.4:学子商城-商城地址--地理定位(腾讯地图)

html5 新特性有一个对象 geolocation 通过浏览器js程序

获取当前用户位置信息

位置信息(经度/纬度/海拨/速度)... 用于实现LBS

(location Base Service)基于位置服务

饿了么,滴滴打车..

手机浏览器定位

(1)智能手机GPS芯片与网络定位精度在米范围

(2)PC定位IP地址解析

html5中提供一个geolocation

widow.navigator.geolocation({

getCurrentPostion:fn

});

2.5:学子商城-商城地址--小程序(腾讯地图)

<map></map>

longitude 中心位置经度位置

latitude 中心位置纬度位置

scale 缩放级别 (3~20) 16

markes 标记点

polyline 路线 (其它经度和纬度)

show-location 显示位置

style="width:100%;height:600rpx"

#如何获取指定位置经度纬度

#http://api.map.baidu.com/lbsapi/getpoint/

#116.300901,39.916085

#创建组件 mymap 我的地图

#注意事项:在手机定位非常准确

2.6:学子商城-个人中心/我的电影(我喜欢电影)

功能1：添加自己喜欢电影[电影描述文字/电影图片]

a:创建表单 文字 上传图片

b:将图片上传云存储并且图片fileID保存云数据库

c:文件添加云数据库

功能2: 将添加成功后电影分页显示[左侧图片/文字]

a:查询云数据库喜欢电影信息分页

2.7:学子商城-个人中心/我的电影(我喜欢电影)

-功能

(1)在云数据库中创建集合 mymovie [在云控制面板中]

(2)创建组件pages/mymovie/mymovie

(3)将 home;mymap;mymovie 添加tabbar

(4)在组件mymovie(输入框/上传图片按钮/添加按钮)

https://youzan.github.io/vant-weapp/

4.1:mymovie.json 引入二个组件 输入框/按钮

4.2:添加输入框在模板中

4.3:添加二个按钮 [上传图片 提交] 在模板中

4.4:添加图片显示view

作业1:完成mymovie

作业2:下载明天使用软件

-(1)phonegap 服务器

PhoneGapSetup-win32.exe

-(2)phonegap 模板程序

-(3)蓝碟(自行下载)或夜神手机模拟器

BluestacksCnSetup\_6191\_legacy.exe

-(4)phonegap 手机apk 安装包

PhoneGap\_Develop.apk

# WX-07-PHONEGAP

今天学习的内容

1:复习昨天的内容

2:今天的目标

2.0:检查

注册订阅号帐户(新邮箱不能使用小程序注册时邮箱)

检查软件安装

phonegap desktop

夜神或蓝碟 https://www.yeshen.com

2.1:完成我喜欢的电影

2.2:混编 phonegap (了解)

#安装混编软件环境一定关闭防火墙

混编一种新技术:将web项目直接打包成功手机原生app安装包

原生:android java 3

原生:苹果 object c 3

h5 项目 3

微信小程序 3

支付宝小程序 3

2.3:混编 phonegap (了解)

国外:phonegap 混编平台

国内:uniapp https://uniapp.dcloud.io/

提供二个方案

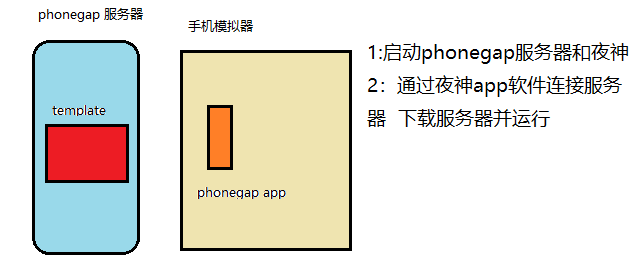
(1)一个程序打包成多种平台app

(2)支持手机底层硬件开发(课程内容)

2.4:混编 phonegap (了解)--创建开发环境

(1)4个程序支持 phonegap服务器/软件模板

手机模拟器/phonegap app



2.5:混编 phonegap (了解)--手机硬件

-device 设备

device.platform 获取手机操作系统 IOS Android

device.uuid 获取手机硬件编码 48

device.version 获取手机版本

-notification 提示功能

navigator.notification.alert("") 提示信息

navigator.notification.confirm(msg,fn) 确认消息

navigator.notification.beep(3) 蜂鸣器

navgator.notification.vibrate(1000) 震动1s

-camera 相机

navigator.camera.getPicture(fn1,fn2,{参数});

fn1 拍照成功

fn2 拍照失败

{quality:50} 拍照延迟时间

-capture 录音

navigator.device.capture.captureAudio(fn1,fn2,参数);

fn1: 录音成功回调函数

fn2: 录音失败回调函数

{limit:3} 录音几次

-accelerometer 加速度传感器

(1)获取用户当前位置信息

getCurrentAcceleration(fn1,fn2);

fn1:获取位置成功 返回对象 x y z

fn2:获取位置失败

(2)创建监听用于监听用户位置变化

var 变量 = watchAcceleration(fn1,fn2,参数);

fn1 获取位置成功 返回对象 x y z

fn2 获取位置失败

{frequency:1000} 1s监听一次

(3)取消监听

clearWatch(变量)

-补充环境启动过程

(1)启动phonegap 服务器

(2)启动手机模拟器(蓝碟/夜神)

(3)点击模拟器 phonegap app 连接服务器

(4)template/www/index.html

2.5:微信公众平台--订阅号-聊天机器人

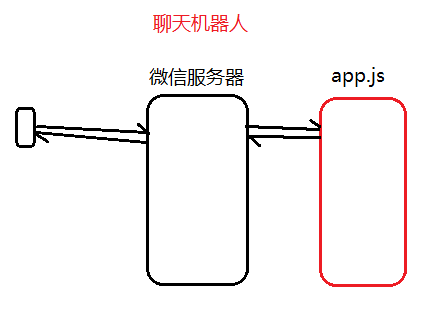
-微信公众平台(小程序-订阅号(个人媒体平台))

2013 产品

-专业(美食/技术/旅游/...)

-编造(历史/明星/国际/...)

2.6:微信公众平台--订阅号-聊天机器人-开发聊天机器人



任务一:复制微信订阅号中测试帐appid (订阅号钥匙)

wxa56801014e697901

任务二:申请合法域名并且绑定4000 端口上

解决方案:内网穿透工具,此工具临时生合法域名并且与4000

程序绑定一起

使用方法: ngroke.exe 协议名 端口

示例: .\ngrok.exe http 4000 回车

合法域名: https://d03b5142.ngrok.io

任务三:开发聊天机器人 app1903.js

(1)加载二个模块 express/wechat (聊天模块)

(2)创建配置对象

config = {appid:"复制", //订阅号钥匙

token="weixin"} //暗号

server.post("/",wechat(config,(req,res)=>{

req.weixin.Content //手机用户发送你信息

res.reply(回复); //回复

}))

(3)node app1903.js

任务四:配置去微信公众平台测试帐户里

接口配置:

url[https://d03b5142.ngrok.io]

token[weixin]

js接口安全域名

域名[https://d03b5142.ngrok.io]

任务五:测试

二维扫描

常见错误

(1)乱码:程序gbk编码,微信要求utf-8

(2)配置

作业1:

请MIS协助开外网完成下面三个课前任务：

(1)最新版本的Angular需要Node.js版本10.9.0或更高版本；

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

node-v11.0.0-x64.msi

node-v11.0.0-x86.msi

可以在命令行输入 node -v 查看已安装的Node.js的最新版本；

如果需要更新Node.js，请到官网nodejs.org或者中文网nodejs.cn下载最新LTS版本并安装。

(2)(可选)Angular项目脚手架工具可用于创建空白的Angular项目，请提前安装：

npm i -g @angular/cli

(3)(可选)运行上面安装的Angular脚手架工具，创建一个测试项目并运行：

ng new myproject01

进入项目目录：

cd myproject01

运行该Angular项目：

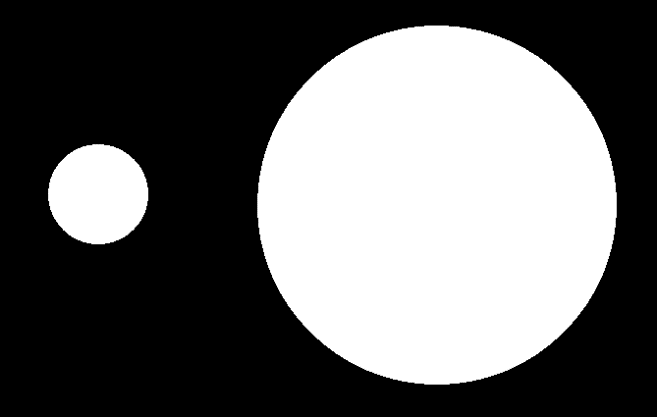
npm start

启动成功后，可以打开浏览器访问该Angular项目:

http://127.0.0.1:4200

提示：如果2/3总是失败，等待明天课上解决。

# ANGLUAR



CSS => Less/SCSS

JS => ES => TS(TypeScript)

DOM => jQuery => MVVM

CSS => SCSS(编译器)

JS => TS(编译器)，是JS的超集

**如何自学 = 基础知识 + 官网手册(零散时间)**

如何自学一门新语言？

如何自学一门新框架？

<https://cloud.tencent.com/developer/devdocs>

1.设计原则

(1)YAGNI

You Aren't Gonna Need It，你不会需要它；不写不需要的功能

(2)KISS

Keep It Simple and Stupid，让你的代码越简单/傻瓜越好

(3)OCP

Open Close Principle，开闭原则，对外界修改封闭(不允许修改已有代码)，对外界的扩展开放

(4)High Cohesion，Low Coupling

高聚合，低耦合；功能相关代码紧密在一起；功能不相干代码拆分越明确越好

(5)迪米特法则/最少知识原则

一个对象/组件，数据/操作越少越好

2.Angular

Angular，由Google在2009年创建的MVVM框架，适用于中大型的企业级SPA应用。

V1.x官网：https://angularjs.org/ Angular.js用JS编写

V2.x~8.x官网：https://angular.io/ Angular用TS编写

V2.x~8.x中文网：https://angular.cn/

提示：V1到V2的升级变化非常大！ V1支持SCRIPT直接引入和脚手架方式；V2开始只支持脚手架方式。

注意：Angular 8.x要求Node.js版本必须是10.9以上！

|  |
| --- |
| 提示：如果使用NPM从官方网站下载NPM包总是失败，可以把默认下载地址改为国内淘宝网镜像：  查看当前下载地址： npm config get registry  默认值为： https://registry.npmjs.org/  修改为淘宝网NPM镜像： npm config set registry=https://registry.npm.taobao.org/ |

3.创建第一个Angular项目

①安装Node.js和NPM

Node.js版本必须>=10.9

②安装全局的Angular脚手架工具

npm i -g @angular/cli

默认会在C:\Users\web\AppData\Roaming\npm目录下安装ng.cmd可执行文件及其相关文件

③运行全局脚手架，创建一个Angular空白项目

ng new myngapp01

此步需要联网从NPM官网下载第三方模块，如果总是失败可以拷贝他人下载好的空白项目

④进入项目根目录，运行该项目（Node.js项目）

npm start 或者 ng serve

|  |
| --- |
| Vue.js项目的主配置文件： vue.config.js  Angular项目的主配置文件： angular.json |
| ES6中有“模块”的概念： export/export default、 import..from..  Vue.js中无自己的“模块”概念——项目是由自定义组件构成；  Angular中有自己的“模块”概念——项目是由自定义组件构成，每个组件都要放在一个特定的Module中 |

4.Angular项目引导启动流程

①根据angular.json查找系统运行入口文件：main.ts

②main.ts启动主模块： AppModule

③在主模块中引导启动主组件：AppComponent

④主组件中声明了模板和样式，最终渲染在index.html中 <app-root/>

5.Angular核心概念之一 —— 组件

Component：组件，就是一段可以复用的页面片段，如轮播广告、手风琴、下拉菜单.......；每个组件内有自己专用的HTML片段、CSS样式、JS数据和操作。

Component = Template(母版) + Style + Script

提示：Angular中每个组件(Component)必须处于某个模块(NgModule)中！！

自定义组件的步骤：

①创建一个ts文件

例如： src/app/myc01.ts

②ts文件中声明并导出一个class

@Component({

selector: 'app-myc01', //标签名

template: '<h1>我的组件01</h1>' //组件模板

})

export class MyComponent01 { }

③在某个模块中声明该组件，否则无法使用该组件

@NgModule({ declarations: [ MyComponent01 ] })

④在当前模块内的其它组件的模板中使用该组件

//app.component.html

<app-myc01></app-myc01>

练习：创建一个自定义组件MyComponent02，其对应的元素为app-myc02；模板内容随意；试着在前面两个组件中使用这第三个组件

练习：创建一个自定义组件MyComponent03，其对应的元素为app-myc03；模板内容随意；试着在前面三个组件中使用这第四个组件

练习：创建一个文件夹myc05，其中自定义组件MyComponent05，其对应的元素为app-myc05；模板/样式独立出来都放在此文件夹下；试着在主组件中使用这个组件

**提示：可以使用Angular脚手架提供的 ng g component 组件名 来自动的创建一个组件所需要的文件；若没有成功安装全局命令ng，也可以使用 npx ng g component 组件名 。 （NPX是Node.js安装包提供的一个命令，可用于执行node\_modules下的可执行文件）**

6.Angular核心概念之二 —— 数据绑定

Data Bingding：把脚本中的数据(称为Model模型)，绑定到HTML(称为View视图)，未来只要Model发生改变，则View自动更新。

Angular中数据绑定的呈现方式：

①插值语法/大胡子语法，绑定到innerHTML：{{ }}

<any>{{expression}}</any>

②属性绑定：[ ]

<any [attr]="expression">

<any attr="{{expression}}">

③事件绑定：( )

<any (event)="fn( )">

注意：事件绑定中函数后的( )不能省略！

说明：(1)插值语法中可以使用算术运算吗？比较运算符？逻辑运算符？三目运算符？调用对象的方法吗？自增/自运算符？创建新对象吗——{{}}中不能出现new？

练习：使用数据绑定实现如下组件：myc11，购买数量增减



数据绑定相关的指令：

①循环绑定：\*ngFor

<li \*ngFor="let tmp of scoreList; let i=index;">

{{i}} - {{tmp}}

</li>

②选择绑定：\*ngIf

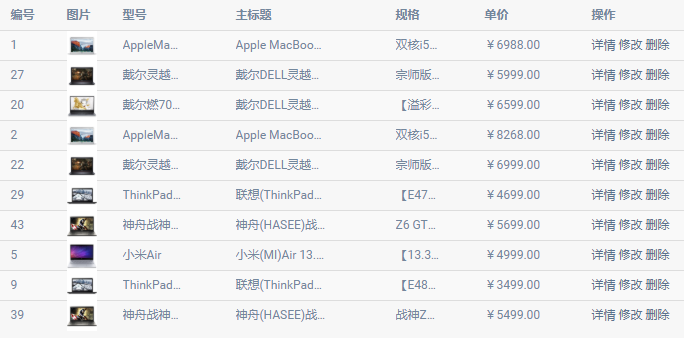
③样式绑定：[ngStyle]、[ngClass]

课后任务：

(1)比较Vue.js和Angular的不同

|  |  |  |
| --- | --- | --- |
|  | Vue.js | Angular |
| 插值表达式 |  |  |
| 属性绑定 |  |  |
| 事件绑定 |  |  |
| 循环指令 |  |  |
| 选择指令 |  |  |
| 样式指令 |  |  |

(2)创建组件，包含Model数据：productList: [ (),(),()...() ]，绑定到如下的表格中，要求：点击“删除”可以实现模型数据的删除功能



# WX-07-PHONEGA

复习：

(1)环境搭建

①升级Node.js到10.9+ node/npm/npx

②全局下载Angular脚手架工具

npm i -g @angular/cli

③运行脚手架创建空白项目

ng new myngapp02

④进入空白项目，运行当前项目：package.json/scripts

cd myngapp02

npm start

⑤使用客户端浏览器访问

http://127.0.0.1:4200

(2)Angular核心概念之一——组件

可以复用的页面片段，

Component = Script + Template + Style

Angular中的组件是多文件组件： .ts + .html + .css/.scss

注意：组件必须在某个模块(AppModule)中声明！

使用快捷命令创建： ng g component 组件名

(3)Angular核心概念之二——数据绑定

①innerHTML绑定：{{ }}

②属性绑定：[ attr ] = ""

③事件绑定：( click ) = "fn( )"

④循环绑定指令：\*ngFor

⑤选择绑定指令：\*ngIf

⑥选择绑定指令：[ngSwitch] \*ngSwitchCase \*ngSwitchDefault

⑦样式绑定指令：[ngStyle]

⑧样式绑定指令：[ngClass]

⑨双向数据绑定指令：

|  |
| --- |
| 面试题：Angular中指令分为哪几类：  **(1)组件指令**  <myc01></myc01>  Component extends Directive，组件继承自指令，组件是有模板的指令  **(2)结构型指令**  可以影响当前的DOM结构的指令；所有结构型指令都以 \* 开头  **(3)属性型指令**  不会影响DOM结构，只会影响当前元素的特征，如样式；所有的属性型指令都用 [ ] 括起来！ |

1.Angular核心概念之三——指令(Directive)

指令：是一种特殊的模板页面内容，可以对页面执行特殊的处理；

例如： <any \*ngFor="">

Angular中常见指令：

①循环绑定指令： \*ngFor

<any \*ngFor="let tmp of 集合对象">

②选择绑定指令： \*ngIf

<any \*ngIf="expression">

提示：\*ngIf会影响DOM结构

③选择绑定指令： [ngSwitch] \*ngSwitchCase \*ngSwitchDefault

<any [ngSwitch]="变量名">

<any \*ngSwitchCase="值">....</any>

<any \*ngSwitchCase="值">....</any>

...

<any \*ngSwitchDefault>....</any>

</any>

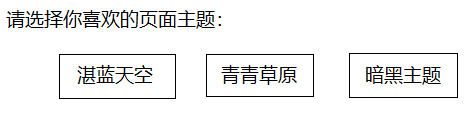
④样式绑定指令：[ngStyle]

<any [ngStyle]="obj">

⑤样式绑定指令：[ngClass]

<any [ngClass]="obj">

练习：创建新的组件myc02-directive，实现某块区域样式切换效果：



2.(了解)如何自定义指令

提示：可使用工具 ng g directive 指令名 快速的创建一个指令

<div appNeedStrong>....</div>

@Directive({ selector: 'appNeedStrong' })

export class MyDirective {

contructor( el: ElementRef){

el.nativeElement.xxx....

}

}

3.Angular中的双向数据绑定

**方向1：Model => View**

**方向2：View => Model**

Angular中实现双向数据绑定的方法：

<input [(ngModel)]="userName">

**提示：ngModel指令处于FormsModule，必须在当前模块中声明导入才能使用：**

import { FormsModule } from '@angular/forms';

@NgModule({

imports: [ FormsModule ]

})

如果想监视模型数据的改变（就像Vue.js中的watch函数），可以使用 ngModelChange事件：

<input [(ngModel)]="userName" (ngModelChange)="fn( )">

练习：创建密码输入框，随着用户的输入，后面的提示信息不停的在改变：

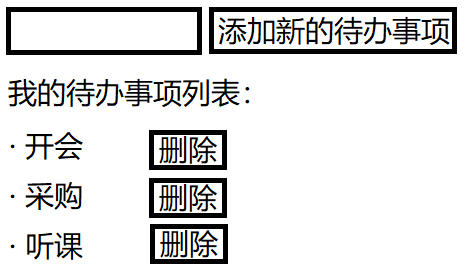
"密码不能为空"

"密码长度不能少于6位"

"密码长度合法"

"密码长度不能超过12位"

练习：使用Angular实现 TodoList



4.Angular核心概念之四——管道

Vue.js中的过滤器(filter)： <p>{{ 1 | sex('zh') }}</p>

Angular中的类似的概念称为管道(pipe)：<p>{{ 1 | sex:'zh' }}</p>

Vue.js没有内置任何过滤器；但Angular内置了很多好用的管道：

①lowercase：把数据转换为小写形式

②uppercase：把数据转换为大写形式

③slice：获取字符串或数组中的一部分

④json：把对象转换为JSON字符串

⑤date：把日期/数字转换为特定格式的日期字符串

⑥number：数字格式化(每三位加逗号，并指定小数位数)

⑦currency：把数字以货币形式显示

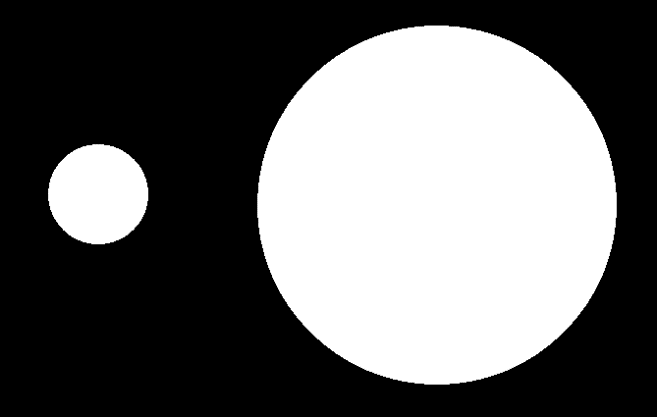
课后任务：

(1)对比Vue.js和Angular中数据绑定方法

(2)对比Vue.js和Angular中所有的指令

(2)创建组件 xz-register，实现学子商城注册页面中所有的输入校验和注册提交功能（需要借助XHR访问后台数据API）

# ANGULAR-01



CSS => Less/SCSS

JS => ES => TS(TypeScript)

DOM => jQuery => MVVM

CSS => SCSS(编译器)

JS => TS(编译器)，是JS的超集

**如何自学 = 基础知识 + 官网手册(零散时间)**

如何自学一门新语言？

如何自学一门新框架？

<https://cloud.tencent.com/developer/devdocs>

1.设计原则

(1)YAGNI

You Aren't Gonna Need It，你不会需要它；不写不需要的功能

(2)KISS

Keep It Simple and Stupid，让你的代码越简单/傻瓜越好

(3)OCP

Open Close Principle，开闭原则，对外界修改封闭(不允许修改已有代码)，对外界的扩展开放

(4)High Cohesion，Low Coupling

高聚合，低耦合；功能相关代码紧密在一起；功能不相干代码拆分越明确越好

(5)迪米特法则/最少知识原则

一个对象/组件，数据/操作越少越好

2.Angular

Angular，由Google在2009年创建的MVVM框架，适用于中大型的企业级SPA应用。

V1.x官网：https://angularjs.org/ Angular.js用JS编写

V2.x~8.x官网：https://angular.io/ Angular用TS编写

V2.x~8.x中文网：https://angular.cn/

提示：V1到V2的升级变化非常大！ V1支持SCRIPT直接引入和脚手架方式；V2开始只支持脚手架方式。

注意：Angular 8.x要求Node.js版本必须是10.9以上！

|  |
| --- |
| 提示：如果使用NPM从官方网站下载NPM包总是失败，可以把默认下载地址改为国内淘宝网镜像：  查看当前下载地址： npm config get registry  默认值为： https://registry.npmjs.org/  修改为淘宝网NPM镜像： npm config set registry=https://registry.npm.taobao.org/ |

3.创建第一个Angular项目

①安装Node.js和NPM

Node.js版本必须>=10.9

②安装全局的Angular脚手架工具

npm i -g @angular/cli

默认会在C:\Users\web\AppData\Roaming\npm目录下安装ng.cmd可执行文件及其相关文件

③运行全局脚手架，创建一个Angular空白项目

ng new myngapp01

此步需要联网从NPM官网下载第三方模块，如果总是失败可以拷贝他人下载好的空白项目

④进入项目根目录，运行该项目（Node.js项目）

npm start 或者 ng serve

|  |
| --- |
| Vue.js项目的主配置文件： vue.config.js  Angular项目的主配置文件： angular.json |
| ES6中有“模块”的概念： export/export default、 import..from..  Vue.js中无自己的“模块”概念——项目是由自定义组件构成；  Angular中有自己的“模块”概念——项目是由自定义组件构成，每个组件都要放在一个特定的Module中 |

4.Angular项目引导启动流程

①根据angular.json查找系统运行入口文件：main.ts

②main.ts启动主模块： AppModule

③在主模块中引导启动主组件：AppComponent

④主组件中声明了模板和样式，最终渲染在index.html中 <app-root/>

5.Angular核心概念之一 —— 组件

Component：组件，就是一段可以复用的页面片段，如轮播广告、手风琴、下拉菜单.......；每个组件内有自己专用的HTML片段、CSS样式、JS数据和操作。

Component = Template(母版) + Style + Script

提示：Angular中每个组件(Component)必须处于某个模块(NgModule)中！！

自定义组件的步骤：

①创建一个ts文件

例如： src/app/myc01.ts

②ts文件中声明并导出一个class

@Component({

selector: 'app-myc01', //标签名

template: '<h1>我的组件01</h1>' //组件模板

})

export class MyComponent01 { }

③在某个模块中声明该组件，否则无法使用该组件

@NgModule({ declarations: [ MyComponent01 ] })

④在当前模块内的其它组件的模板中使用该组件

//app.component.html

<app-myc01></app-myc01>

练习：创建一个自定义组件MyComponent02，其对应的元素为app-myc02；模板内容随意；试着在前面两个组件中使用这第三个组件

练习：创建一个自定义组件MyComponent03，其对应的元素为app-myc03；模板内容随意；试着在前面三个组件中使用这第四个组件

练习：创建一个文件夹myc05，其中自定义组件MyComponent05，其对应的元素为app-myc05；模板/样式独立出来都放在此文件夹下；试着在主组件中使用这个组件

**提示：可以使用Angular脚手架提供的 ng g component 组件名 来自动的创建一个组件所需要的文件；若没有成功安装全局命令ng，也可以使用 npx ng g component 组件名 。 （NPX是Node.js安装包提供的一个命令，可用于执行node\_modules下的可执行文件）**

6.Angular核心概念之二 —— 数据绑定

Data Bingding：把脚本中的数据(称为Model模型)，绑定到HTML(称为View视图)，未来只要Model发生改变，则View自动更新。

Angular中数据绑定的呈现方式：

①插值语法/大胡子语法，绑定到innerHTML：{{ }}

<any>{{expression}}</any>

②属性绑定：[ ]

<any [attr]="expression">

<any attr="{{expression}}">

③事件绑定：( )

<any (event)="fn( )">

注意：事件绑定中函数后的( )不能省略！

说明：(1)插值语法中可以使用算术运算吗？比较运算符？逻辑运算符？三目运算符？调用对象的方法吗？自增/自运算符？创建新对象吗——{{}}中不能出现new？

练习：使用数据绑定实现如下组件：myc11，购买数量增减



数据绑定相关的指令：

①循环绑定：\*ngFor

<li \*ngFor="let tmp of scoreList; let i=index;">

{{i}} - {{tmp}}

</li>

②选择绑定：\*ngIf

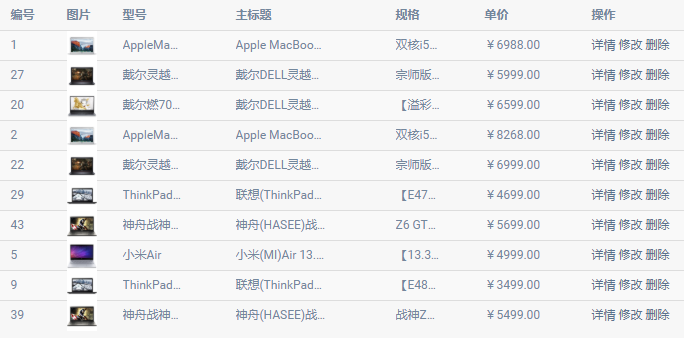
③样式绑定：[ngStyle]、[ngClass]

课后任务：

(1)比较Vue.js和Angular的不同

|  |  |  |
| --- | --- | --- |
|  | Vue.js | Angular |
| 插值表达式 |  |  |
| 属性绑定 |  |  |
| 事件绑定 |  |  |
| 循环指令 |  |  |
| 选择指令 |  |  |
| 样式指令 |  |  |

(2)创建组件，包含Model数据：productList: [ (),(),()...() ]，绑定到如下的表格中，要求：点击“删除”可以实现模型数据的删除功能



# ANGULAR-02

复习：

(1)环境搭建

①升级Node.js到10.9+ node/npm/npx

②全局下载Angular脚手架工具

npm i -g @angular/cli

③运行脚手架创建空白项目

ng new myngapp02

④进入空白项目，运行当前项目：package.json/scripts

cd myngapp02

npm start

⑤使用客户端浏览器访问

http://127.0.0.1:4200

(2)Angular核心概念之一——组件

可以复用的页面片段，

Component = Script + Template + Style

Angular中的组件是多文件组件： .ts + .html + .css/.scss

注意：组件必须在某个模块(AppModule)中声明！

使用快捷命令创建： ng g component 组件名

(3)Angular核心概念之二——数据绑定

①innerHTML绑定：{{ }}

②属性绑定：[ attr ] = ""

③事件绑定：( click ) = "fn( )"

④循环绑定指令：\*ngFor

⑤选择绑定指令：\*ngIf

⑥选择绑定指令：[ngSwitch] \*ngSwitchCase \*ngSwitchDefault

⑦样式绑定指令：[ngStyle]

⑧样式绑定指令：[ngClass]

⑨双向数据绑定指令：

|  |
| --- |
| 面试题：Angular中指令分为哪几类：  **(1)组件指令**  <myc01></myc01>  Component extends Directive，组件继承自指令，组件是有模板的指令  **(2)结构型指令**  可以影响当前的DOM结构的指令；所有结构型指令都以 \* 开头  **(3)属性型指令**  不会影响DOM结构，只会影响当前元素的特征，如样式；所有的属性型指令都用 [ ] 括起来！ |

1.Angular核心概念之三——指令(Directive)

指令：是一种特殊的模板页面内容，可以对页面执行特殊的处理；

例如： <any \*ngFor="">

Angular中常见指令：

①循环绑定指令： \*ngFor

<any \*ngFor="let tmp of 集合对象">

②选择绑定指令： \*ngIf

<any \*ngIf="expression">

提示：\*ngIf会影响DOM结构

③选择绑定指令： [ngSwitch] \*ngSwitchCase \*ngSwitchDefault

<any [ngSwitch]="变量名">

<any \*ngSwitchCase="值">....</any>

<any \*ngSwitchCase="值">....</any>

...

<any \*ngSwitchDefault>....</any>

</any>

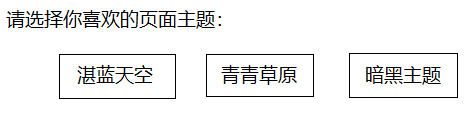
④样式绑定指令：[ngStyle]

<any [ngStyle]="obj">

⑤样式绑定指令：[ngClass]

<any [ngClass]="obj">

练习：创建新的组件myc02-directive，实现某块区域样式切换效果：



2.(了解)如何自定义指令

提示：可使用工具 ng g directive 指令名 快速的创建一个指令

<div appNeedStrong>....</div>

@Directive({ selector: 'appNeedStrong' })

export class MyDirective {

contructor( el: ElementRef){

el.nativeElement.xxx....

}

}

3.Angular中的双向数据绑定

**方向1：Model => View**

**方向2：View => Model**

Angular中实现双向数据绑定的方法：

<input [(ngModel)]="userName">

**提示：ngModel指令处于FormsModule，必须在当前模块中声明导入才能使用：**

import { FormsModule } from '@angular/forms';

@NgModule({

imports: [ FormsModule ]

})

如果想监视模型数据的改变（就像Vue.js中的watch函数），可以使用 ngModelChange事件：

<input [(ngModel)]="userName" (ngModelChange)="fn( )">

练习：创建密码输入框，随着用户的输入，后面的提示信息不停的在改变：

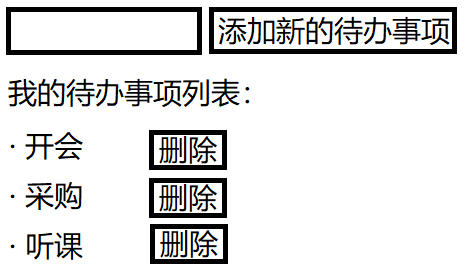
"密码不能为空"

"密码长度不能少于6位"

"密码长度合法"

"密码长度不能超过12位"

练习：使用Angular实现 TodoList



4.Angular核心概念之四——管道

Vue.js中的过滤器(filter)： <p>{{ 1 | sex('zh') }}</p>

Angular中的类似的概念称为管道(pipe)：<p>{{ 1 | sex:'zh' }}</p>

Vue.js没有内置任何过滤器；但Angular内置了很多好用的管道：

①lowercase：把数据转换为小写形式

②uppercase：把数据转换为大写形式

③slice：获取字符串或数组中的一部分

④json：把对象转换为JSON字符串

⑤date：把日期/数字转换为特定格式的日期字符串

⑥number：数字格式化(每三位加逗号，并指定小数位数)

⑦currency：把数字以货币形式显示

课后任务：

(1)对比Vue.js和Angular中数据绑定方法

(2)对比Vue.js和Angular中所有的指令

(2)创建组件 xz-register，实现学子商城注册页面中所有的输入校验和注册提交功能（需要借助XHR访问后台数据API）

# ANGULAR-03

# ANGULAR-04

# ANGULAR-05