#### **目标:高级前端、大前端、前端全栈工程师**

[一、课程体系 1](#_Toc524105575)

[阶段1: 项目流程、数据库、JS基础、Node.js 2](#_Toc524105576)

[1、项目流程与数据库 2](#_Toc524105577)

[DAY01 软件周期 2](#_Toc524105578)

[DAY02 服务器&数据库 4](#_Toc524105579)

[DAY03 语句操作&列约束 7](#_Toc524105580)

[DAY04 列约束以及各种查询 10](#_Toc524105581)

[复习以及练习 16](#_Toc524105582)

[2、JavaScript基础 22](#_Toc524105583)

[DAY01 JavaScript介绍 22](#_Toc524105584)

[DAY02类型 27](#_Toc524105585)

[DAY03 if 31](#_Toc524105586)

[DAY04 switch for do循环 34](#_Toc524105587)

[DAY05 for循环 37](#_Toc524105588)

[DAY06函数 对象 40](#_Toc524105589)

[DAY07 数组 43](#_Toc524105590)

[DAY08 字符串操作 48](#_Toc524105591)

[DAY09 Date Number Boolean对象 51](#_Toc524105592)

[3、Node.js基础-成亮 53](#_Toc524105593)

[DAY01 全局对象 模块系统 53](#_Toc524105594)

[DAY02 包和npm以及其他模块 58](#_Toc524105595)

[DAY03 Node服务器搭建 61](#_Toc524105596)

[DAY04 中间件MySQL模块 63](#_Toc524105597)

[DAY05-DAY06项目练习 66](#_Toc524105598)

# 一、课程体系

**学习目标：高级前端、大前端、前端全栈工程师**

阶段1：项目流程、数据库、JS基础、Node.js

阶段2：HTML基础、AJAX、CSS、Bootstrap

阶段3：JS高级、BOM&DOM、jQuery、Vue.js

阶段4：VueUI、H5高级、Node.js

阶段5：Angular、React

阶段6：Ionic、PhoneGap、微信

2.学前准备

(1)全部技术内容从零开始、专业没要求

(2)英语单词从零开始、只要下功夫

(3)对时间有要求！早上-白天-晚上-夜间

(4)对键速有要求！—— 350次/分钟 TopSpeeder

3.Windows常用快捷键

Win+e：打开我的电脑

Win+d：显示/隐藏桌面

Win+r：打开“运行”窗口 calc/mspaint/cmd

Alt+Tab：在多个窗口间切换(从左往右)

Alt+Shift+Tab：在多个窗口间切换(从右往左)

Alt+F4：关闭当前窗口

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

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

4.软件工程

Debug： Ada - Bug - Debug（调试程序）

IBM - DOS - Bill Gates - Microsoft – Windows

## 阶段1: 项目流程、数据库、JS基础、Node.js

### 1、项目流程与数据库

#### DAY01 软件周期

成亮

18514668981

chengliang@tedu.cn

1.软件工程

IBM -> DOS(Bill Gates) -> Microsoft -> Windows

软件工程学科

(1)软件过程

(2)软件测试

(3)软件度量

(4)软件质量保证

(5)软件标准

2.软件过程

软件的开发流程/软件生命周期

**(1)软件定义期**

①可行性研究阶段——《可行性研究报告》

技术、人力、设备、时间、资金、回报、政策、风俗

②需求分析阶段——《需求分析报告》

功能性需求、非功能性需求

**(2)软件开发期**

③概要设计阶段

子系统、模块、各自的功能

④详细设计阶段

主体内容、使用的程序、算法、数据库...

⑤编码实现阶段

设计师：出效果图

前端：把效果图转换成网页(html、css、js)

后端：为前端提供页面中所需要的数据

⑥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

|  |
| --- |
| 目标项目：学子商城(www.codeboy.com)  前台子系统：商品模块、用户模块、购物车模块  后台子系统：商品模块、用户模块、订单模块  移动端子系统：商品模块、用户模块、购物车模块 |

3.服务器

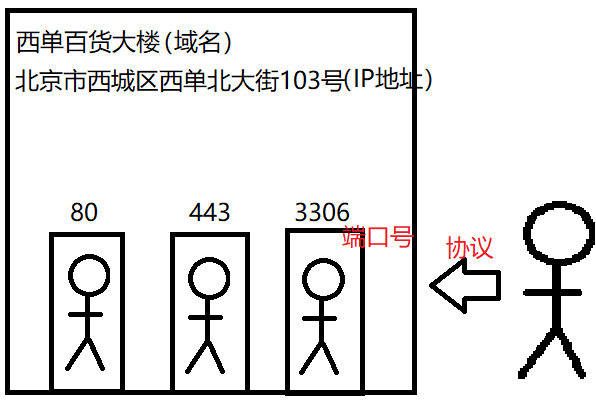
server：为客户端提供各种服务的功能强大的计算机。

访问服务器：

(1)服务的**域名/IP地址**

(2)每一个服务都有对应的**端口号**

(3)提供访问该服务所使用的**协议**



访问web服务器(提供网页服务)

http://www.codeboy.com:80

**访问自己电脑创建的web服务器**

启动web服务器 /webServer/start.cmd

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

练习：在public下创建4.html，随便输入内容，然后使用服务器来访问。

**获取自己电脑在局域网的IP地址**

在命令行输入 ipconfig

在浏览器中 http:// IP地址:80 回车

例如：http://172.163.100.163:80

练习：同桌之间尝试使用局域网IP地址访问对方服务器。

#### DAY02 服务器&数据库

复习

软件开发流程

软件定义期

可行性研究阶段

需求分析阶段

软件开发期

概要设计阶段

详细设计阶段

编码实现阶段

测试阶段

软件维护期

项目部署阶段

项目维护阶段

服务器

提供服务，功能强大的计算机

访问服务器下的某一个服务器

访问服务器 IP地址/域名 端口 协议

访问服务器下的web服务器

http://127.0.0.1:80

http://localhost:80

http://172.163.100.163:80

输入http://www.taobao.com 如何获取淘宝首页的。

源码下载地址

http://code.tarena.com.cn

用户名：tarenacode 密码：code\_2013

今日目标：

项目中如何保存数据——数据库 （难点：单词）

1.保存数据的几种形式

内存 临时保存数据

文件 不便于管理

第三方机构 腾讯云，百度云，阿里云

独立数据库服务器

2.数据库概述

Database：数据库，以特定的结构批量存储数据的软件。

数据库的发展史

网状数据库->层次型数据库->**关系型数据库**->非关系型数据库

|  |
| --- |
| 关系型数据库中数据的逻辑结构  Server->Database->Table->Row->Column  服务器->数据库->表->行->列 |

3.MySQL

MySQLDB分支：https://www.mysql.com/

MariaDB分支：https://mariadb.org/

xampp下载地址

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

|  |
| --- |
| xampp是一个服务器套装，包含多款服务器软件。 |
| d->Deamon 精灵，守护者  mysqld.exe 启动MySQL服务器 |

4.MySQL数据库系统：

(1)服务器端：

负责存储数据

C:/xampp/..../mysqld.exe

(2)客户端

负责向服务器发起增删改查的命令

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

(3)连接MySQL数据库服务器

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

简写： mysql -uroot

-h 服务器的IP地址/域名 可以使用localhost

-P 端口，MySQL服务使用3306端口

-u 用户

-p 密码

练习：尝试使用错误的信息连接数据库服务器。

5.MySQL管理命令

quit; 退出服务器连接

show databases; 显示数据库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; 描述表中有哪些列(表头)

练习：查看phpmyadmin数据库下以下表都有哪些列

pma\_\_favorite 2

pma\_\_history 6

pma\_\_recent 2

6.MySQL常用的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中的数据。——增删改查

SQL语言最早是由IBM提出，后提交给ISO，成为行业内标准语言。分为多个版本:SQL-87、SQL-92、SQL-99。当前标准的SQL语言被绝大多数的关系型数据库所支持。

|  |
| --- |
| SQL命令的两种执行方式  (1)交互模式：客户端连接MySQL，输入一行，回车，服务器执行一行。适用于临时性的查看数据  mysql -uroot 回车  (2)脚本模式：客户端把要执行的多行命令写在一个脚本文件中，一次性的提交给服务器执行。适用于批量的操作数据。  mysql -uroot < C:/xampp/...../02.sql |

练习：使用脚本模式向数据库服务器提交多行SQL命令，尝试写错其中一行，执行查看结果。

**SQL语法规范**

(1)每条SQL语句必须以英文分号作为结尾，一条语句可以跨越多行，见到分号认为结束。

(2)若第n条SQL语句有语法错误，则后续所有的语句不会再执行。

(3)SQL语句中不区分大小写。习惯上数据库关键字用大写，非关键字用小写。

(4)SQL语句还可以使用单行注释：#... 或者多行注释: /\*...\*/，注释的内容不会被服务器所执行。

**日常开发中常用的SQL命令**

(1)丢弃数据库，如果存在的话

DROP DATABASE IF EXISTS jd;

(2)创建新的数据库

CREATE DATABASE jd;

(3)进入刚刚创建的数据库

USE jd;

(4)创建数据表

CREATE TABLE student(

sid INT,

name VARCHAR(6),

sex VARCHAR(1),

score INT

);

(5)向数据表中插入记录

INSERT INTO student VALUES

('10','Tom','M','89'),('11','Kate','F','92').....;

(6)查询数据表所有的记录

SELECT \* FROM student;

练习：创建脚本文件05.sql；先丢弃数据库tedu，如果存在的话；创建数据库tedu;

练习：创建脚本文件06.sql; 先丢弃数据库xuezi，如果存在的话；创建数据表laptop，包含列有lid(编号)，title(标题)，price(价格)，number(数量)；插入3条记录，查询出所有记录。

课后任务：

(1)复习今天内容，保留注释，重新编写代码

(2)练习：

创建脚本文件xz.sql，重新创建数据库xuezi；丢弃数据库xuezi,如果存在。创建数据表user，包含列uid编号，uname用户名，upwd密码，email邮箱，phone电话，sex性别，user\_name真实姓名，registerTime注册时间。 插入3条记录，查询出所有记录。

(3)预习MySQL手册第11章列类型

#### DAY03 语句操作&列约束

复习

网状数据库，层次型数据库，**关系型数据库**，非关系数据库

MySQL

server->database->table->row->column

连接

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

mysql -uroot

脚本模式

mysql -uroot < C:/xampp/..../xz.sql

mysql管理命令

quit;/show databases;/use 数据库名;/show tables;/desc 表名;

SQL命令

CREATE DATABASE xz;

USE xz;

CREATE TABLE user(

uid INT,

...

);

INSERT INTO user VALUES(),()...;

SELECT \* FROM user;

1.MySQL中常用SQL命令

增删改查

(1)更改记录

UPDATE user SET

upwd='654321', email='wh@tedu.cn' WHERE uid='1';

练习：更改编号为2的数据，电话改为1318888888，性别改为'M',注册时间改为 '2017-3-1'

(2)删除记录

DELETE FROM user WHERE uid='3';

2.数据库中的中文乱码

**计算机中如何存储英文字符**

ASCII: 总共有128个，对所有的英文字符及符号进行了编码。

hello 72698977

Latin-1: 总共有256个，兼容ASCII，同时对欧洲符号进行了编码

**计算机中如何存储中文字符**

GB2312: 对6千多常用的汉字进行了编码，兼容ASCII码

GBK: 对2万多汉字进行了编码，兼容GB2312

BIG5: 台湾繁体字编码，不兼容GB2312和GBK；兼容ASCII码

Unicode：对世界上主要的语言的字符进行编码，兼容ASCII码，不兼容GBK，GB2312，BIG5等。具体分为UTF-8，UTF-16，UTF-32三种存储方案。

**中文乱码产生的根源**

MySQL默认使用Latin-1编码，而这个编码不能存储中文的。

**解决MySQL存储中文乱码**

(1)sql脚本文件另存为的编码

(2)客户端连接服务器使用的编码

(3)服务器端数据库存储数据使用的编码

都统一使用UTF8编码。

**设置editplus的默认编码：**

Tools->preferences->file->default encoding

练习：创建脚本文件01\_dangdang.sql；设置连接数据库编码utf8，创建数据库dangdang，进入该数据库，创建数据表book,包含bid编号，title标题，price价格，pubDate出版时间，isJia是否加价购(是->1，不是->0)；插入4本书的信息，修改第2条书的信息，删除第3条，查询结果。

解决存储中文

3.MySQL中列类型

列类型：创建表的时候，指定列可以存储的数据类型

CREATE TABLE book( bid 列类型 );

**(1)数值类型——可以不加引号**

TINYINT: 微整型，占用1个字节，-128~127

SMALLINT: 小整型，占用2个字节，-32768~32767

INT: 整型，占用4个字节，-2147483648~2147483647

BIGINT: 大整型，占用8个字节，范围很大

FLOAT: 单精度浮点型，占4个字节，3.4E38，范围比INT大的多，但是精度不如INT，可能产生误差。

DOUBLE: 双精度浮点型，占8个字节，1.79E308，范围比BIGINT大的多，但是精度不如BIGINT，可能产生误差。

DECIMAL(M,D): 定点小数，不会产生计算误差舍入；M代表总的有效位数(不包含小数点)，D代表小数点后有效位数。

BOOL: 布尔型，取值只能是TRUE/1、FALSE/0，数据库在存储的时候使用TINYINT(1或0); **TRUE和FALSE不能加引号**。

219.85

25000.32

123456.789\*10^-1

12345.6789

1234.56789\*10^1

123.456789\*10^2

12.3456789\*10^3

**(2)字符串类型——必须加引号**

VARCHAR(M): 变长字符串，不会产生空间浪费，操作速度相对慢，M不能超过65535。

CHAR(M): 定长字符串，可能产生空间浪费，操作速度相对快，M不能超过255。

TEXT(M): 大型变长字符串，M不超过2G

**(3)日期时间类型——必须加引号**

DATETIME: 日期时间类型 '2018-11-2 15:18:30'

DATE: 日期类型 '2018-11-2'

TIME: 时间类型 '15:18:30'

|  |  |  |
| --- | --- | --- |
|  | CHAR(5) | VARCHAR(5) |
| a | a\0\0\0\0 | a\0 |
| ab | ab\0\0\0 | ab\0 |
| abc | abc\0\0 | abc\0 |
| 达内 | 达内\0\0\0 | 达内\0 |

age TINYINT

empCount SMALLINT

commentCount Int

price DECIMAL(6,2) #9999.99

phone CHAR(11)

blog VARCHAR(8000)

isVip BOOL

练习：创建脚本文件02\_xuezi.sql；创建数据库xuezi，设置保存的字符编码为utf8；创建保存笔记本信息的表laptop，包含列lid编号，title标题，price价格，stockCount库存量，shelfTime上架时间，isIndex是否显示在首页。插入4条记录

练习：创建脚本文件03\_tedu.sql；创建数据库tedu，设置存储的编码；创建保存部门信息的表dept，包含(did，dname部门名称，empCount员工数量)，插入3条数据

10 研发部 3

20 市场部 2

30 运营部 1

创建保存员工信息的表emp，包含(eid，ename姓名，sex，birthday生日，salary工资，deptId所属部门编号)，插入对应的员工数据。

4.列约束

MySQL可以对插入的数据进行特定的检验，只有满足条件的数据才能插入成功，否则会提示“非法插入”，“禁止插入”。

例如：插入人的年龄范围是0~100之间，商品的价格大于0...

MySQL提供了六种列约束

(1)主键约束——PRIMARY KEY

声明了主键约束的列上不能插入重复的值，如果主键的数据是数值会自动按照由小到大的顺序排列——加快查找速度。注意：一个表中最多有一个主键。

|  |
| --- |
| MySQL中的NULL含义：空的，空缺的；  指应该有某个数据，但暂时还不能确定具体的指；  例如：还没到发年终奖，未决定员工的部门，未确定部门经理 |

课后任务：

(1)复习今天内容，重新编写代码

(2)练习：

创建脚本文件xuezi.sql，创建数据库xuezi，使用utf编码；创建保存笔记本分类表xz\_laptop\_family(fid，fname分类名称，laptopCount 笔记本数量)

10 戴尔 3

20 小米 1

30 苹果 2

创建笔记本信息表xz\_laptop，包含(lid，title，price，detail介绍，shelfTime上架时间，isOnsale是否在售，familyId所属分类)，插入6条记录。

#### DAY04 列约束以及各种查询

复习

更改记录

UPDATE emp SET salary=5000,sex=0 WHERE eid=5;

删除记录

DELETE FROM emp WHERE eid=6;

中文乱码

Latin-1

Unicode，utf8

sql脚本文件，客户端设置连接使用的编码，服务器存储数据使用的编码

列类型

数值型

TINYINT(-128~127) SMALLINT(-32768~32767) INT(-2147483648~2147483647) BIGINT FLOAT DOUBLE DECIMAL(6,2) 9998.99 BOOL

字符串型

VARCHAR(M) 65535

CHAR(M) 255

TEXT

日期时间型

DATE TIME DATETIME

列约束

主键约束 PRIMARY KEY

今日目标

列约束

简单查询

1.列约束

(1)主键约束——PRIMARY KEY

(2)唯一约束——UNIQUE

声明了唯一约束的列上不能插入重复的值，但可以插入NULL，而且可以插入多个NULL

(3)非空约束——NOT NULL

声明非空约束的列上不能插入NULL值，如果插入的是NULL也不会报错，会显示空内容。

(4)默认值约束——DEFAULT

可以使用DEFAULT关键字为列声明默认值；有以下两种用法：

INSERT INTO xz\_laptop VALUES(1,'小米air', DEFAULT...);

INSERT INTO xz\_laptop(lid,title) VALUES(2,'苹果pro');

(5)检查约束

检查约束可以对插入的数据范围进行验证

CREATE TABLE student(

age TINYINT CHECK(age>18 AND age<60)

);

MySQL不支持，会降低插入和更新等操作效率。

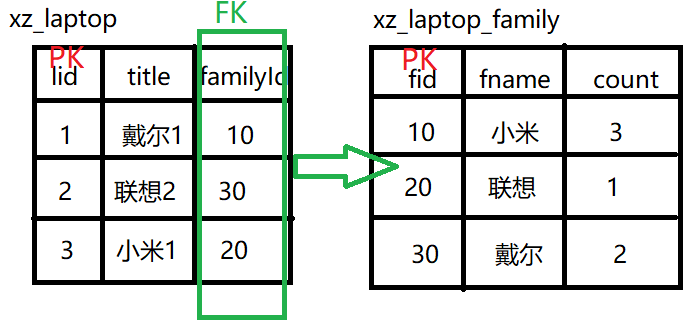
(6)外键约束——FOREIGN KEY

声明了外键约束的列，取值必须在另一个表的主键上出现过

取值可以是NULL;

**注意：**外键的列类型要和另外一个表主键的列类型一致。

FOREIGN KEY(familyId) REFERENCES xz\_laptop\_family(fid)



|  |
| --- |
| 所有的列约束中，主键约束推荐使用——可以提高查询的效率；其它的列约束是否使用根据项目而定，可以不用——会影响数据的插入和更新效率。 |

2.MySQL中的自增列

AUTO\_INCREMENT: 自动增加，加入一个列声明了自增列，无需手动赋值，直接指定为NULL，会自动获取当前的最大值，新纪录会执行+1然后插入。

注意：

允许手动赋值

只用于整数型的主键列上

练习：使用列约束；

创建脚本文件02\_tedu.sql；创建数据库tedu

创建部门表 dept，包含did，dname部门名称；

10 研发部 20 市场部 30 运营部 40 测试部

创建员工表 emp，包含eid，ename，sex，birthday生日，salary工资，deptId隶属的部门编号

插入14条记录，分属于10/20/30部门，其中最后一个员工不属于任何一个部门。

3.项目中如何存储日期时间数据

2018-11-25

2018年11月25日

11-25-2018

25/11/2018

**存储的是距离计算机元年的毫秒数，使用BIGINT类型。**

计算机元年 1970-1-1 0:0:0

1秒钟=1000毫秒

数据库存储1000 1970-1-1 0:0:1

1000\*60 1970-1-1 0:1:0

1000\*60\*60 1970-1-1 1:0:0

2018年距离计算机元年的毫秒：48\*365\*24\*60\*60\*1000

男 女

男孩 女孩

男人 女人

man woman

girl boy

male female

1 0

4.简单查询

**(1)查询所有的列**

SELECT \* FROM emp;

\*代表所有的列

**(2)查询特定的列**

示例：查询所有员工的姓名，生日，工资。

SELECT ename,birthday,salary FROM emp;

练习：查询所有员工的编号，姓名，性别，工资

SELECT eid,ename,sex,salary FROM emp;

**(3)给列取别名**

示例：查询所有员工的姓名和工资，列名使用汉字。

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；全部使用中文别名。

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

练习：查询所有员工的编号，姓名，性别，生日；全部使用1个英文字母作为别名。

SELECT eid a,ename b,sex c,birthday d FROM emp;

|  |
| --- |
| AS关键字可以省略 |

**(4)只显示不同的值/合并相同的项**

练习：查询公司都有哪些性别的员工

SELECT DISTINCT sex FROM emp;

练习：查询公司员工都分布在哪些部门

SELECT DISTINCT deptId FROM emp;

**(5)查询时执行运算**

示例：计算5+3-7/2\*9

SELECT 5+3-7/2\*9;

练习：查询所有员工的姓名及其年薪，并且起别名。

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

练习：假设所有员工工资加500，年终奖是5000，查询出姓名和年薪，并且起别名。

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

(6)查询的结果排序

示例：查询所有员工的信息，结果按照工资由低到高排序

SELECT \* FROM emp ORDER BY salary ASC;

示例：查询所有员工的信息，结果按照工资由高到低排序

SELECT \* FROM emp ORDER BY salary DESC;

练习：查询所有员工的信息，结果按照年龄由大到小。

SELECT \* FROM emp ORDER BY birthday ASC;

练习：查询所有员工的信息，结果按照姓名升序排序。

SELECT \* FROM emp ORDER BY ename ASC;

练习：查询所有员工的信息，按生日由大到小，若生日相同，再按照姓名升序排列。

SELECT \* FROM emp ORDER BY birthday DESC,ename ASC;

练习：查询所有员工信息，按照工资由大到小排序，要求女员工必须排在男员工的前边。

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

|  |
| --- |
| asc -> ascend 升序  desc -> descend 降序  如果不加排序规则，默认是按照升序排列（asc）。  ORDER BY可以按照**数值、日期、字符串(**字符的编码)来排序 |

(7)条件查询

示例：查询编号为5的员工工资

SELECT \* FROM emp WHERE eid=5;

|  |
| --- |
| SQL语句中支持的比较运算符：  = != > < >= <=  等于 不等于 大于 小于 大于等于 小于等于 |

练习：查询出工资大于等于6000的员工所有信息。

SELECT \* FROM emp WHERE salary>=6000;

练习：查询出所有女员工的信息；

SELECT \* FROM emp WHERE sex=0;

练习：查询出员工姓名为king的所有信息

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

练习：查询出不在10号部门的员工的信息。

SELECT \* FROM emp WHERE deptId!=10;

练习：查询出1990年之前出生的员工信息

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

练习：查询出1993年之后出生的员工信息

SELECT \* FROM emp WHERE birthday>'1993-12-31';

练习：查询出没有明确部门的员工信息

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

SELECT \* FROM emp WHERE deptId IS NULL;

|  |
| --- |
| 注意: NULL不能和任何值进行等于/不等于判定，包括NULL自己。可以使用 IS NULL 和 IS NOT NULL |

练习：查询所有有明确部门的员工信息

SELECT \* FROM emp WHERE deptId IS NOT NULL;

练习：查询出10号部门所有女员工的信息

SELECT \* FROM emp WHERE deptId=10 AND sex=0;

|  |
| --- |
| AND(并且) OR(或者) |

练习：查询出10号和30号部门所有员工信息

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

练习：查询出工资在6000~8000所有员工信息

SELECT \* FROM emp WHERE salary>=6000 AND salary<=8000;

SELECT \* FROM emp WHERE salary BETWEEN 6000 AND 8000;

|  |
| --- |
| between ... and ... >= 第1个值 <= 第2个值  not between ... and ... |

练习：查询出不在6000~8000所有员工信息。

SELECT \* FROM emp WHERE salary<6000 OR salary>8000;

SELECT \* FROM emp WHERE salary NOT BETWEEN 6000 AND 8000;

练习：查询出1990年出生的员工信息。

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

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

练习：查询10号和20号和30号部门的员工信息。

SELECT \* FROM emp WHERE deptId=10 OR deptId=20 OR deptId=30;

SELECT \* FROM emp WHERE deptId IN(10,20,30);

练习：查询出不在10号和20号部门的员工信息

SELECT \* FROM emp WHERE deptId NOT IN(10,20);

(8)模糊条件查询

示例：查询出姓名中含有字母e的员工信息

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

练习：查询出姓名以e结尾的员工信息

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

练习：查询出姓名中倒数第2个字符为e的员工信息

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

|  |
| --- |
| % 任意多个字符 >=0  \_ 任意一个字符 =1  上述两个符号不能和=连用，必须使用LIKE |

(9)分页查询

分页显示：假如查询的结果集中有太多的记录，一次显示不完，可以分多页显示。

|  |  |  |
| --- | --- | --- |
| 页码 | 开始 | 结束 |
| 第1页 | 0 | 4 |
| 第2页 | 5 | 9 |
| 第3页 | 10 | 14 |
| 第4页 | 15 | 19 |

开始 = (页码-1)\*每页数量

SELECT \* FROM emp LIMIT start,count;

start 开始的值

count 查询的数量

示例：每页大小是5，查询第1页的数据

SELECT \* FROM emp LIMIT 0,5;

练习：每页大小是5，查询第2页的数据

SELECT \* FROM emp LIMIT 5,5;

练习：每页大小是5，查询第3页的数据

SELECT \* FROM emp LIMIT 10,5;

练习：每页大小是5，查询第4页的数据

SELECT \* FROM emp LIMIT 15,5;

**注意：**LIMIT 后边的两个数值不能添加引号。

综合：

SELECT \* FROM emp

WHERE ...

ORDER BY...

LIMIT...

练习：查询出所有的男员工工资最高的前三个人。

课后任务：

(1)复习今天内容

(2)删除所有的代码，保留注释，重新编写SQL语句。

#### 复习以及练习

复习：

软件过程：

软件定义期：可行性研究、需求分析

软件开发期：概要设计、详细设计、编码实现、测试

软件维护期：项目部署、项目维护

MySQL中常用的SQL语句：

show databases;

use 库名;

show tables;

desc 表名;

show warnings; #显示上一条语句产生的警告

show create table 表名; #显示创建表时所用的语句

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

drop database if exists 库名;

create database 库名 charset=utf8;

use 库名;

create table 表名( 列名 列类型 列约束 );

insert into 表名 values('值');

delete from 表名 where 条件;

update 表名 set 列名='值' where 条件;

select \* from 表名;

列类型：

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

日期时间类型：date/time/datetime

字符串类型：char/varchar/text

列约束：

唯一/非空/**主键**/默认值/检查/外键

简单查询：

查询特定的列 select ename,sex from emp

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

显示不同的值 select distinct sex from emp

执行运算 select salary\*12 from emp

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

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

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

复杂查询：

1.MySQL中的简单查询 —— 查询结果的排序

示例：查询出所有员工信息，要求按工资由小到大排序

SELECT \* FROM emp

ORDER BY salary ; #默认是由小到大

#ORDER BY salary ASC; #ascendant，升序

示例：查询出所有员工信息，要求按工资由大到小排序

SELECT \* FROM emp

ORDER BY salary DESC ; #descendant，降序

练习：查询出所有员工信息，按照姓名由小到大排序

SELECT \* FROM emp

ORDER BY ename ;

练习：查询出所有员工信息，按照姓名由大到小排序

SELECT \* FROM emp

ORDER BY ename DESC;

练习：查询出所有员工信息，按照年龄由大到小排序

SELECT \* FROM emp

ORDER BY birthday ASC ; #生日由小到大排列

练习：查询10号部门中员工的所有信息，按工资由大到小排列

SELECT \* FROM emp

WHERE deptId=10

ORDER BY salary DESC ;

#注意：where子句和order by子句的顺序问题

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

SELECT \* FROM emp

ORDER BY salary DESC, ename ASC ;

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

SELECT \* FROM emp

ORDER BY sex ASC , birthday DESC ;

2.MySQL中的简单查询 —— 分页查询 —— 重点&面试重点

分页查询：用于分批次的显示大量的数据。语法：

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

LIMIT start, count ;

(1)start：是一个数字，表示从哪一行开始获取数据

(2)count：是一个数字，表示一次最多获取的行数，即页面大小

假设每页显示5行记录，则

第1页： LIMIT 0, 5

第2页： LIMIT 5, 5

第3页： LIMIT 10, 5

第4页： LIMIT 15, 5

第5页： LIMIT 20, 5

....

第n页： LIMIT (n-1)\*5, 5

练习：分页查询出员工信息，页面大小为5，显示第1页

SELECT \* FROM emp LIMIT 0, 5 ;

练习：分页查询出员工信息，页面大小为5，显示第2页

SELECT \* FROM emp LIMIT 5, 5 ;

练习：分页查询出员工信息，页面大小为5，显示第3页

SELECT \* FROM emp LIMIT 10, 5 ;

练习：分页查询出员工信息，页面大小为5，显示第4页

SELECT \* FROM emp LIMIT 15, 5 ;

3.MySQL复杂查询 —— 子查询

示例：查询出“研发部”所有员工的信息

步骤1：查询研发部的部门编号 => 10

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

步骤2：查询部门编号为10的员工信息

SELECT \* FROM emp WHERE deptId = 10;

综合上述两步为一条语句：

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

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

) ;

练习：查询出工资比TOM高的所有员工信息

步骤1：查询TOM的工资 => 6000

SELECT salary FROM emp WHERE ename='TOM';

步骤2：查询工资大于6000的员工信息

SELECT \* FROM emp WHERE salary>6000;

综合上述两步：

SELECT \* FROM emp WHERE salary>(

SELECT salary FROM emp WHERE ename='TOM'

);

练习：查询出比TOM年长的所有员工信息

步骤1：查询TOM的生日 => 1990-1-1

SELECT birthday FROM emp WHERE ename='TOM';

步骤2：查询出生日小于1990-1-1的员工所有信息

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

综合上述两步：

SELECT \* FROM emp WHERE birthday<(

SELECT birthday FROM emp WHERE ename='TOM'

);

练习：查询出与TOM同年出生的所有员工信息

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

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

步骤2：查询出生日中年份等于1990的员工所有信息

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

综合上述两步：

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

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

);

4.MySQL复杂查询 —— 聚合/分组查询

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

MySQL提供了几个聚合函数：

COUNT()：计算若干数据的个数

SUM()：计算若干数据的总和

AVG()：计算若干数据的平均值

MAX()：计算若干数据的最大值

MIN()：计算若干数据的最小值

示例：查询出员工的总数

SELECT COUNT(eid) FROM emp; #14

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

SELECT COUNT(sex, deptId) FROM emp;

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

SELECT COUNT(\*) FROM emp; #推荐写法

示例：查询出所有员工工资的总和

SELECT SUM(salary) FROM emp;

练习：查询出所有员工工资的平均值（用两种方法）

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

SELECT AVG(salary) FROM emp;

练习：查询出所有员工工资的最大值和最小值

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

练习：查询出工资高于所有人平均工资的员工的数量 —— 难度系数：5星

步骤1：查询出所有员工的平均工资 => 6842

SELECT AVG(salary) FROM emp;

步骤2：查询出工资大于6842的员工的数量

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

综合上述两步：

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

SELECT AVG(salary) FROM emp

);

示例：查询出每个部门的编号及该部门员工的数量——先分组再聚合计算

SELECT deptId, COUNT(\*) FROM emp

GROUP BY deptId ; #按照部门编号分组

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

练习：按照性别分组，查询每组中工资的最大值、最小值、平均值

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

FROM emp

GROUP BY sex;

5.MySQL复杂查询 —— 跨表/多表查询 —— 了解

查询结果集中的数据来自于多个不同的表。

示例：查询每个员工的姓名及其所在部门的名称

SELECT ename, dname

FROM emp, dept ; #笛卡尔积！错误！

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

SELECT ename, deptId, did, dname

FROM emp, dept

WHERE deptId = did ; #跨表查询必须有连接条件

注意：上述写法是SQL-92版本中的写法。存在一定的缺陷：

(1)无法显示没有部门的员工 —— KING

(2)无法显示没有员工的部门 —— 测试部

SQL-99标准对此进行了改变，但是SQL变得很复杂，有四种写法：

**(1)内连接查询：inner join ... on ...**

SELECT ename, dname

FROM emp INNER JOIN dept

ON deptId=did ;

#内连接的效果与SQL-92标准完全一样

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

SELECT ename, dname

FROM emp LEFT OUTER JOIN dept

ON deptId=did ; #13+1

#可以显示出“左侧表”中所有的记录！即使右侧表中没有对应的记录

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

SELECT ename, dname

FROM emp RIGHT OUTER JOIN dept

ON deptId=did ; #13+1

#可以显示出“右侧表”中所有的记录！即使左侧表中没有对应的记录

**(4)全外连接查询：full join ... on ...**

SELECT ename, dname

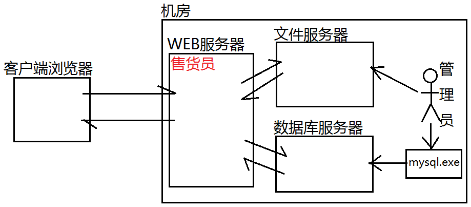
FROM emp FULL JOIN dept

ON deptId=did ; #13+1+1

#可以显示出“右侧表”和“右侧表”中所有的记录！—— MySQL不支持

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

6.WEB服务器



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

**(1)静态WEB服务器**：提供的内容任何时间任何人访问都是一样的

常见的静态WEB内容：HTML/CSS/JS/图片/视频/Flash...

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

常见的动态WEB技术：

JSP = HTML + JAVA

PHP = HTML + php

ASP.NET = HTML + C#

Node.js

Python

.....

### 2、JavaScript基础

#### DAY01 JavaScript介绍

复习

|  |
| --- |
| 标准SQL语句分类  **(1)DDL: Data Define Language 定义数据**  CREATE/DROP/ALTER(修改)  **(2)DML: Data Manipulate Language 操作数据**  INSERT/DELETE/UPDATE  **(3)DQL: Date Query Language 查询数据**  SELECT  **(4)DCL: Date Control Language 控制用户权限**  GRANT(授权)/REVOKE(收权) |

CREATE TABLE emp(

eid INT PRIMARY KEY AUTO\_INCREMENT

);

INSERT INTO emp VALUES(NULL....)

练习：

查询出所有男员工工资最高的前3个人。

SELECT \* FROM emp WHERE sex=1 ORDER BY salary DESC LIMIT 0,3;

1.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; #推荐写法

练习：使用员工的姓名这一列查询员工数量

SELECT COUNT(ename) FROM emp;

练习：使用员工的部门编号这一列查询员工数量

SELECT COUNT(deptId) FROM emp;

练习：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，提供数据，产出结果。——饺子机  COUNT(...)/SUM(...)/AVG(...)/MAX(...)/MIN(...) |

练习：查询出所有员工的工资总和是多少

SELECT SUM(salary) FROM emp;

练习：查询出所有员工的平均工资是多少

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

SELECT AVG(salary) FROM emp;

练习：查询出工资最高的员工工资是多少

SELECT MAX(salary) FROM emp;

练习：查询出工资最低的员工工资是多少

SELECT MIN(salary) FROM emp;

练习：查询出年龄最大的员工

SELECT MIN(birthday) FROM emp;

|  |
| --- |
| 分组查询: 只能查询分组的条件和聚合函数。 |

示例：查询出每个部门的员工数量是多少。

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

练习：查询出男女的员工的平均工资，最高工资，最低工资是多少

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

函数补充：

|  |
| --- |
| YEAR(...) 获取日期中的年份  MONTH(...) 获取日期中的月份  DAY(...) 获取日期中的日 |

示例：查询出1991年出生员工的所有信息

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

练习：查询出3月份出生的员工所有信息

SELECT \* FROM emp WHERE MONTH(birthday)=3;

练习：查询出5号出生的员工所有信息

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

(2)子查询

**把一个SQL语句的查询结果作为另外一个SQL语句的查询条件**

示例：查询出研发部所有的员工信息

步骤1：查询出研发部的部门编号——10

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

步骤2：根据研发部的部门编号10查询员工信息

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工资高的员工的所有信息

步骤1：查询出tom的工资——6000

SELECT salary FROM emp WHERE ename='tom';

步骤2：查询出工资比6000高的员工所有信息

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同一年出生的员工信息

步骤1：查询出tom的出生的年份是多少——1990

SELECT YEAR(birthday) FROM emp

WHERE ename='tom';

步骤2：查询出出生年份为1990年的员工信息

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：查询出所有员工的姓名及其部门名称。

SELECT ename,dname FROM emp,dept;

错误：产生笛卡尔积！

多表查询如何避免产生笛卡尔积，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，无法查询出没有部门的员工，也无法查询出没有员工的部门。  SQL-99中提出了新的多表查询方法。 |

**(1)内连接 INNER JOIN...ON...** 和SQL92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;

查询结果是右侧所有记录都显示；OUTER可以省略

**(4)全连接 FULL JOIN**

显示左侧和右侧所有记录——MySQL不支持

UNION 合并相同的项；

UNION ALL 不合并相同的项；

(SELECT ename FROM emp\_us)

UNION

(SELECT ename FROM emp\_cn);

(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.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

有基础的学员

程序员必做50题

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

3.JS概述

(1)历史

1995年，JS最早出现在Netscape的浏览器中

2009nian ,JS遵循CommonJS规范，开始向服务器端发展。

(2)现状

既可以运行在客户端浏览器，也可以运行在服务器端。

(3)特点

解释型语言，编译一行执行一行。

弱类型语言

基于对象

跨平台

3.JS的执行环境

(1)浏览器自带的JS解释器

(2)NODEJS下的JS解释器

https://nodejs.org nodejs下载地址

在命令行下 node -v 查看系统中nodejs版本号

(3)执行JS代码

**浏览器：**

创建01.js和01.html

在01.html中引入01.js

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

**NODEJS下：**

node C:/xampp/..../01.js

4.JS代码规范

(1)区分大小写

(2)每行代码结束的分号可加可不加，提倡都加

(3)分为单行注释(//...)和多行注释(/\*...\*/)

5.变量

变量就是用于存放数据的容器。

x=1 y=2

(1)声明变量

**var a=1;**

练习：声明多个变量，分别保存员工的编号，姓名，性别，生日，工资，部门。

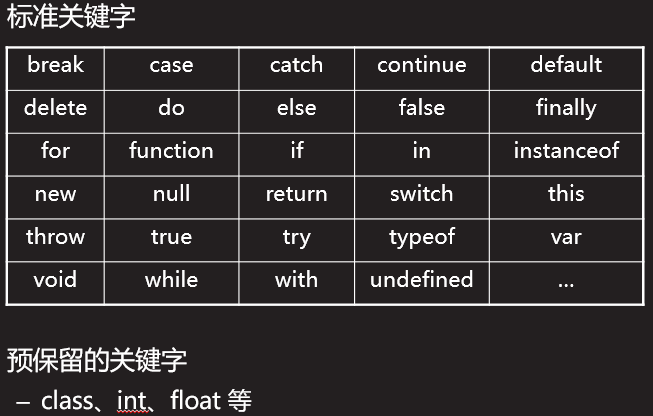
(2)变量的命名规则

变量名称是以字母、数字、美元符号($)、下划线组成的；不能以数字开头。

多个连词之间的命名方法(下划线命名法、驼峰命名法)

user\_name userName

不能使用关键字和保留字作为变量名



(3)变量的注意

var a;

变量声明后未赋值，此时的值是undefined

使用未声明的变量会报错

(4)一次性声明多个变量

var a=1,b=2,c;

多个变量之间用逗号隔开。

3.常量

一旦声明不能重新赋值。

例如：春节的日期、中秋节的日期、圆周率...

关键字： **const** PI=3.14;

4.数据类型

分为原始类型和引用类型

原始类型分为数值型、字符串型、布尔型、未定义型(undefined)、空(null)

(1)数值型

分为整型和浮点型

整型在内存中占4个字节，浮点型占8个字节

8进制，以0开头，例如012 -> 10

16进制，以0X开头，例如0XF -> 15

a~f 代表 10~15 不区分大小写

0XFF -> 255

课后任务：

(1)复习今天的内容，删除注释重新编写代码

(2)练习：

①使用常量保存圆周率，使用程序分别计算半径为5(变量)的30度/60度/90度的弧度是多长。

②使用变量分别保存商品的单价为12.5,数量为20；单价为30，数量为8.5；计算总和。

(3)预习JS的数据类型

#### DAY02 类型

复习

开发环境

浏览器 js

服务器 nodejs

变量声明

var $\_c1=3;

字母，数字，美元符号，下划线

var a; undefined

a=1;

var a,b=2,c=5;

常量声明

const PI=3.14;

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null

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

(1)数值型

分为整型和浮点型

8进制(012以0开头) 16进制(以0X开头) 10进制

1234.56 -> 1.23456\*10^3 -> 1.23456E3

**typeof(数据) 检测数据类型**

(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。

查看一个字符的Unicode编码

'a'.charCodeAt() //97

(3)布尔型

在程序中表示真或者假的结果

true/false

常用于一些是否的结果，是否登录，是否注册，是否是会员，是否在售

isLogin=true; isOnsale=false;

(4)未定义型

声明了变量未赋值，结果就是undefined

(5)空

用于释放(销毁)一个引用类型的地址，只有一个值null

2.数据类型转换

(1)隐式转换

①数字+字符串： 数字被转成字符串

1+'a' //'1a'

②数字+布尔型： 布尔型被转成数字 true->1 false->0

1+true //2

1+false //1

③布尔型+字符串： 布尔型转成了字符串

true+'hello' //'truehello'

**JS中加号(+)的作用**

执行加法运算

执行字符串的拼接

**使用 - \* /执行运算**

尝试将运算符的两端转成数值型，如果含有非数字则返回NaN(Not a Number)，不是一个数字。

NaN类型number

练习：查看以下程序的运行结果

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

num1+num2+num3 //'4tedu'

num2+num3+num1 //'truetedu3'

num3+num1+num2 //'tedu3true'

(2)强制转换

①将任意类型转为整型

parseInt( )

parseInt('1.5a') //1

从开头查找数字，遇到非数字或者小数点；返回前边的数字;如果开头是非数字，返回NaN。

②将任意类型转为浮点型

parseFloat()

parseFloat('1.5a') //1.5

转换规则和parseInt类似，区别在于遇到小数点继续往后查找数字。

③将任意类型转为数值型

Number()

Number('1.5a') // NaN

如果要转换的数据中含有非数字，返回NaN

true -> 1 false->0

④数值型和布尔型转为字符串型

toString()

var num=10;

num.toString(); //'10'

num.toString(16); //a

如果要转换的数据是数值，可以设置进制(8,16,2)

3.运算符

由运算符连接操作的数据，所组成的形式就是表达式

(1)算术运算符

+ - \* / % ++ --

% 取余

++ 自增，在原来的基础之上加1

-- 自减，在原来的基础之上减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

(2)比较运算符

> < >= <= == != ===(全等于) !==(不全等于)

返回一个布尔型的结果

== 只是比较两个值是否相同

=== 不仅比较值，还会比较类型是否相同

3>'10' //false

数字和字符串比较，字符串转成数字。

'3'>'10' //true

比较首个字符的Unicode码，如果首个字符，则比较第二个字符

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

3>'10a' //false

3<'10a' //false

3=='10a' //false

'10a' -> NaN

NaN和任何数比较(> >= < <= ==) 都返回false

NaN==NaN 返回false

(3)逻辑运算符

&& 并且 || 或者 ! 非

返回一个布尔型的结果

&& 关联的两个条件都满足，结果是true，否则false

|| 关联的两个条件只需满足其一，结果是true，否则false

! 取反 !false -> true !true -> false

练习：声明两个变量保存用户名和密码，如果用户名是'root'，并且密码是'123456'，打印true，否则打印false；

练习:声明一个变量保存年龄，如果年龄大于90岁，或者年龄小于3岁，打印true，否则打印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，就不需要再执行第二个条件

练习：以下两个语句是否会报错。

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

(4)位运算符(了解)

在执行运算符的时候，会把数字转成二进制进行运算

1 10 11 100 101 110 111 1000 1001 1010

2 4 8

110101=100000+10000+100+1

32 16 4 1

25 = 16+8+1

10000+1000+1

11001

按位与(&) 上下两位都是1，结果是1，否则是0

按位或(|) 上下两位含有1，结果是1，否则是0

按位异或(^) 上下两位不同为1，相同为0

按位右移(>>) 删除二进制的最后一位，大概变小到原来一半

按位左移(<<) 在二进制的最后添加一个0，增加一倍

(5)赋值运算符

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

练习：声明变量保存商品的价格，打五折，计算价格。

(6)三目运算符

单目运算符：只需要一个数据或者表达式

a++ a-- !false

双目运算符：需要两个数据或者表达式

+ - \* / % > < >= <= == != === !==

&& || & | ^ >> << = += -= \*= /= %=

三目运算符：需要三个数据或者表达式

条件表达式 ? 表达式1 : 表达式2

如果条件表达式为true，执行表达式1；

如果条件表达式为false，执行表达式2；

练习：声明两个变量，分别保存两个数字；比较两个数字的大小，打印最大值的变量名。

练习：声明两个变量，分别保存用户名和密码；如果用户名是root，并且密码是123456，打印“登录成功”，否则打印“登录失败”

课后任务：

(1)复习今天内容，整理当天的思维导图

(2)练习：

声明一个变量保存年份，判断这个年份是否为闰年，如果是打印“是闰年”，否则打印“不是闰年”

闰年：4年一个闰年(能被4整除，和4取余为0)，并且不能被100整除，或者能被400整除。

(3)预习逻辑结构(if if-else..)

#### DAY03 if

复习

原始类型数据

数值型 整型和浮点型 3.14 3.14e2

字符串型

'1' 'undefined'

'a'.charCodeAt()

布尔型

true/false

未定义型

undefined

空

null

typeof(2) number/string/boolean/undefined/object

数据类型转换

隐式转换

数值 + 字符串 = 字符串

数值 + 布尔型 = 数值

布尔型 + 字符串 = 字符串

\* / - 将数据转为数值型 NaN

强制转换

整型 parseInt('a1a') true/false -> NaN

浮点型 parseFloat('a1.5a') true/false -> NaN

数值型 Number('1.5a') true/false -> 1/0

转字符串(数值和布尔型) 15.toString(2)

运算符

算术运算符 + - \* / % ++ --

比较运算符 > < >= <= == != === !==

逻辑运算符 或且非 || && ! 逻辑短路

位运算符 & | ^ >> <<

赋值运算符 = += -= \*= /= %=

三目运算符 条件表达式 ? 表达式1 : 表达式2

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

练习：

声明一个变量保存年份，判断这个年份是否为闰年，如果是打印“是闰年”，否则打印“不是闰年”

闰年：4年一个闰年(能被4整除，和4取余为0)，并且不能被100整除，或者能被400整除。

4年一闰，100年不闰，400年再闰

1.浏览器端函数

alert() 弹出警示框(消息框)

prompt() 弹出提示框(输入框)，需要使用变量来接收输入的值；值的类型是字符串型。

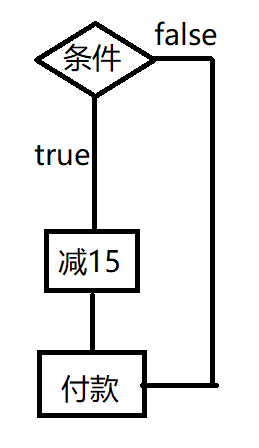
2.流程控制

程序 = 算法 + 数据

程序分为顺序执行、选择执行、循环执行

(1)if语句

满30减15



|  |
| --- |
| 语句1;  if(条件表达式){  语句2;  }  语句3; |

执行流程：

①执行语句1

②执行条件表达式，如果为true，执行语句2；false跳过

③执行语句3

注意：如果if后的大括号语句中只有一行，是可以省略大括号。

|  |
| --- |
| if(age>=18)  console.log('成年人'); |

在if语句的条件表达式中，有一些值默认就是false

0、NaN、''、undefined、null

(2)if-else语句

|  |
| --- |
| 语句1;  if(条件表达式){  语句2;  }else{  语句3;  } |

执行过程：

①执行语句1

②执行条件表达式，如果是true，执行语句2；如果是false，执行语句3

练习：使用弹出提示框分别输入商品的**价格**和**数量**，判断商品**总价**是否满500，如果满500打八折；使用变量保存当前**余额**为600，如果总价足够支付，则打印支付成功，否则打印余额不足。 05\_exercise.html 05\_exercise.js

(3)if-else的嵌套

用于判断多种情况

|  |
| --- |
| 语句0;  if(条件表达式1){  语句1;  }else if(条件表达式2){  语句2;  }else...if(条件表达式n){  语句n;  }else{  语句n+1; //以上所有的条件表达式都是false  } |

执行流程：

①执行语句0

②执行条件表达式1

如果条件表达式1是true，执行语句1

如果条件表达式1是false，执行条件表达式2

如果条件表达式2是true，执行语句2

如果条件表达式2是false，执行条件表达式n

③以上所有的条件表达式为false，执行语句n+1

练习：声明变量保存订单的状态码（1,2,3,4,5...）

1-等待付款 2-等待发货 3-运输中 4-已签收 5-已取消

根据状态码打印对应内容；不存在的状态码打印‘非法的状态’

2.switch-case语句

是一种特殊的分支语句，可以根据一个表达式的不同值，来选择执行不同的程序。

|  |
| --- |
| 语句0;  switch(表达式){  case 1: //如果表达式的值是1  语句1;  break; //终止，不会再往后执行其它语句  ....  case n:  语句n;  break;  default:  语句n+1;  } |

**注意：**在case中表达式和值的比较使用的是全等于比较，要求值和类型都满足结果才是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可以判断相等或者不等的情况，使用范围更广；switch-case只使用全等的情况，结构上更为清晰，执行效率相对高。

3.循环执行

循环：就是一遍又一遍执行**相同或者相似**的代码。

循环的两个要素

循环的条件：重复的次数

循环体：重复执行的相同或者相似代码

(1)while循环

|  |
| --- |
| while(循环的条件){ //是一个布尔型的值  循环体  } |

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

使用while循环打印 11~19之间所有的整数

使用while循环打印 1~100之间所有的奇数(使用if判断是否为奇数)

使用while循环打印 20 22 24 26 28 30

使用while循环计算1~100之间所有的整数和

(3)预习do-while和for循环

《JavaScript高级程序设计》第三版 红宝石书

#### DAY04 switch for do循环

复习

弹出警示框 alert()

弹出提示框 var a=prompt()

程序执行：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(条件表达式1){ 语句1; }else if(条件表达式2){ }.....else{ }

var type='群众'

switch(type){ case '团员': ... break; ... default: }

循环执行

一遍又一遍执行相同或者相似的代码

循环条件

循环体

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.break关键字

break可以结束任何形式的循环

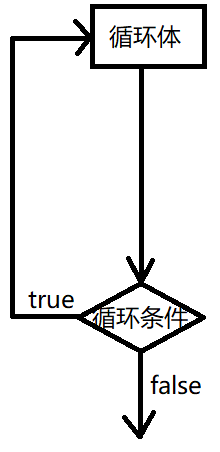
练习：使用无线循环形式来计算1~10所有数字相乘的结果。

练习：使用变量保存1个数字，循环弹出提示框的形式输入数字，如果数字猜大了，警示框提示'big'；如果数字猜小了，警示框提示'small'；否则提示'right'，结束循环。

03\_break.html 03\_break.js

说明： **''(空字符)在和数字比较的时候，转成0**

2.do-while循环



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

不管循环条件是否为true，都会执行一次循环体。

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'，结束循环。

05\_dowhile.js 05\_dowhile.html

3.for循环



1~10

|  |
| --- |
| for(初始值;循环条件;i的变化){  循环体  } |
| ①执行初始值  ②判断循环条件  ③如果循环体是true执行循环体，是false结束循环  ④如果执行了循环体，执行i的变化  ⑤重新执行第2步 |

4.break和continue

break: 结束循环，后续不会再执行其它的循环

continue: 跳过本次循环，继续下一次循环

练习：计算1~100之间所有偶数的和(遇到奇数跳过)

练习：打印1,2,5,7,10,11,13,14,17....98

打印1~100之间，去除所有能被3或4整除的数。

练习：计算新中国成立后一共有多少个闰年。

练习：打印本世纪的前10个闰年

练习：计算1~100之间的和，当和大于4000的时候，结束循环，打印总和以及当前i的值。

5.循环嵌套

while，do-while，for循环三者之间可以相互嵌套。

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

打印倒着的99乘法表

9\*9=81 8\*9=72.

8\*8=64

....

1\*1=1

(3)预习JS中自定义函数

#### DAY05 for循环

复习

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;i的变化){ 循环体 }

break 和 continue

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

循环嵌套

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

学习目标

函数相关(重点)

变量的作用域(重点)

函数的作用域(重点)

递归调用(掌握)

1.函数

parseInt()/parseFloat()/typeof()...

分为系统函数和自定义函数

自定义函数

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

(1)创建普通函数

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

调用

函数名称();

练习：使用函数封装10+20的结果，并打印出来，调用3次。

练习：使用函数封装计算1~100所有数字的和，打印结果，调用3次。

(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(参数列表); //实参->实际的参数

**参数列表**：可以是0个或者是多个数据，之间用逗号隔开；创建函数时的参数称为形参，调用函数时的参数成为实参，调用的时候，实参的值会赋给形参。

形参本质上就是一个声明了的变量，但未赋值。

练习：创建函数getSum，传递1个参数，计算1~任意数字之间的和。

练习：创建函数getRun，传递2个参数，打印任意两个年份之间的所有闰年个数。

(3)带有返回值的函数

|  |
| --- |
| function 函数名称(参数列表){  函数体  return 返回值;  } |

调用

函数名称(参数列表)

return表示函数执行后，所返回的结果。

注意：

①如果没有return或者return后没有返回值，结果都是undefined。

②return可以返回任意类型的数据

③return后的所有代码都不会被执行

练习：创建函数add，传递3个参数，返回3个数字的和。

练习：创建函数getMax，传递2个参数，返回最大值。

练习：创建函数getMax，传递3个参数，返回最大值。

练习：创建函数isRun，传递1个参数(年份)，是闰年返回true，不是闰年返回false。

练习：创建函数getStatus，传递1个参数(订单状态码)，返回对应的中文；

1-待付款 2-待发货 3-运输中 4-已签收 5-已取消 其它状态码-无法查询

**对比return、break和continue**

return 返回结果，用于终止函数的执行，常用于函数中。

break 用于结束循环，结束switch语句；

continue 跳过当前循环，继续下一次循环。

2.变量的作用域

(1)作用域

变量或者函数的可访问范围

分为两种：

**全局作用域**：在全局作用域下声明的变量可以在任意位置访问到。

**函数(局部)作用域**：函数作用域下声明的变量只能在函数内部访问。

注意：在函数内部使用var关键字声明的变量是局部变量，不适用var关键字声明的变量是全局变量。

(2)变量声明提升

JS程序执行前，使用var关键字声明的变量会提升到**所在作用域**的最前边；但赋值还是在原来的位置。

|  |
| --- |
| console.log(a);  var a=1; //var a;会提升到前边; 而 a=1还是在当前位置; |

3.函数的作用域

函数和变量类似，也分为全局作用域和函数(局部)作用域

全局作用域：在全局作用域下创建的函数可以在任意位置调用

函数(局部)作用域：在函数(局部)作用域下创建的函数只能在函数内部调用

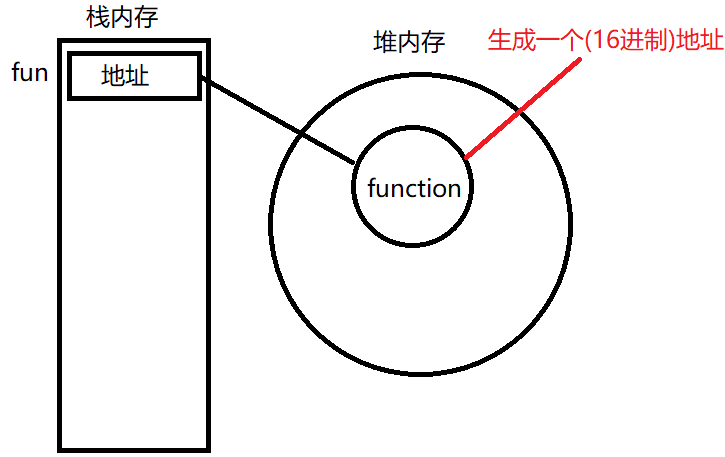
**函数声明提升**

和变量一样，JS在程序执行前，把使用function声明的函数提升到最前边。

|  |
| --- |
| function fun(){  return 1;  } |

函数调用 fun() 获取函数的返回值(return后的值)

函数名称 fun 保存的是函数在堆内存中的地址



4.递归的调用

在函数的内部调用自身

递归要有跳出条件，需要结合return使用。

斐波那契数列

1 1 2 3 5 8 13 21 34 55 89 144

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

创建函数，传递1个参数，使用递归计算斐波那契数列的第n项是多少

(3)预习JS的对象

#### DAY06 函数 对象

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

没有名字的函数 function(){ }

|  |
| --- |
| 创建函数——函数声明  function 函数名称(){} |

(1)创建函数——函数表达式

|  |
| --- |
| var 函数名称=function(形参列表){  函数体  return 返回值;  }  调用：函数名称(实参列表) |

**对比函数声明和函数表达式的区别**

函数声明存在函数提升，在任何的位置都可以调用

函数表达式不存在函数提升，必须先创建再调用

练习：使用函数表达式来计算1~100之间的和。

(2)匿名函数自调用

创建函数(局部)作用域，防止造成全局污染。

|  |
| --- |
| (function(){  函数体//就是局部作用域,创建的变量和函数都是不能被外部访问  })(); |

(3)作为回调函数

把匿名函数作为实参传递给形参，此时的形参就是函数名称

|  |
| --- |
| function fn(num){  //num就是函数的名称  num(); //调用传递的匿名函数  }  fn(function(){ ... }); |

练习：创建函数add，传递2个参数，2个参数都是以匿名函数的形式传递；在匿名函数中分别返回1个数字。

最后在add中计算两个数字相加。

2.系统函数(全局函数)

encodeURI 对一个url进行编码

decodeURI 对一个已经编码的url进行解码

parseInt 将数据转为整型

parseFloat 将数据转为浮点型

isNaN 检测一个数据是否为NaN 是->true 不是->false

isFinite 检测一个数据是否为有限值，是->true 不是->false 1/0 -> Infinity 无限值

eval 执行字符中的表达式 eval('1+2') -> 3

练习：使用弹出提示框输入一组运算，使用eval执行这组运算。 06\_eval.html 06\_eval.js

3.对象

是一种引用类型的数据，存储在堆内存中。

对象：是一组**属性**(property)和**方法/功能**(method)的集合

哪些是对象？

一个手机: 属性有颜色，品牌，尺寸，厚度... 功能有打电话、发短信、照相、聊天、游戏....

一辆汽车：属性有品牌、外观、车型... 功能有代步、撞人、拉货、取暖、乘凉...

(1)JS中的对象

内置对象：JS提供的

宿主对象：根据不同的执行环境来划分

自定义对象：自己创建的对象

(2)创建自定义对象

①对象字面量(直接量)

②内置构造函数

③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开

多组属性之间用逗号隔开

属性名中引号可加可不加，如果出现特殊字符，必须添加引号

|  |
| --- |
| var phone={ color:'red', 'made-in':'china' }; |

练习：创建一个部门对象，属性有部门编号，名称，员工数量

练习：创建一个员工对象，属性有编号，姓名，性别，生日，工资，所在部门

5.使用内置构造函数创建对象

|  |
| --- |
| var book=new Object(); //创建一个空对象  book.id=103; //添加属性，属性名不能添加引号  book['titlte']='三国演义'; //添加属性，属性名必须加引号；如果不加引号，会被认为是变量. |

练习：创建一个汽车对象，属性有编号，品牌，颜色，价格

练习：创建一个电脑对象，属性有型号，品牌，尺寸，产地

6.访问对象中的属性

(1)获取属性值

|  |
| --- |
| emp.eid  emp['ename']  如果要获取的属性名不存在，返回undefined |

(2)遍历对象中的属性(for-in)

获取对象中每一个属性名，进而获取属性值

|  |
| --- |
| for(var key in emp){  //emp 要遍历的对象  //key 要遍历的每一个属性名  //emp[key] 通过属性名获取对应的属性值  } |

**注意：**for-in不能遍历预定义(JS默认为每一个对象添加的)属性。

练习：创建一个商品对象，包含编号，标题，价格，是否在售(isOnsale)，商品库存量(stockCount)，使用for-in遍历所有的属性。

7.检测对象中是否含有某个属性

'属性名' in 对象 // true->存在 false->不存在

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

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

8.对象中的方法

|  |
| --- |
| var person={  name: 'tom',  say: function(){  this.name //this指代当前的对象  }  }  person.say(); //调用对象中的方法 |

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

创建一个圆对象，添加属性半径，圆周率；添加方法计算周长(getLength)和计算面积(getArea)；分别返回圆的周长和面积

(3)预习JS中的数组

#### DAY07 数组

学习目标：

数组(重点)

1.什么是数组？

程序 = 数据 + 算法

数组也是一种数据类型，是数据的集合，可以包含其它任意的数据类型。

数组是通过所在位置的顺序号来访问每一个元素的，顺序号是从0开始，最大序号是 元素个数-1。

**2.定义一维数组**

(1)数组字面量

var arr = [ 元素1, 元素2…. ];

(2)构造函数 new Array()

var arr = new Array('tom','jerry'….)

创建数组，初始化数组元素

var arr = new Array(3)

创建数组，初始化大小为3，值为空.

练习：

创建一个数组country, 在数组中添加5个国家；

(3)数组元素的访问

数组名称[ 元素顺序号 ]

注意：元素顺序号是从0开始，最大的元素个数-1

3.对比undefined和null的区别

哪些情况出现undefined？

①创建了变量，没有赋值

②访问对象中，不存在的成员属性

③访问数组中，没有值的数组元素

null的作用？

用于释放一个变量；例如之前变量中存储了一个对象，或者存储了一个数组，都可以赋值为null来释放这个变量。

练习：

信息录入系统，通过提示框每次输入一个员工的姓名，输入完之后，打印出所有的员工；直到输入exit，退出提示框。

需要使用死循环，每次都执行弹出提示框；输入的内容，插入到数组中；当提示框中的值是exit，执行break终止循环。

while(true){ }

var names=[];

var str = prompt()

**4.获取数组的长度(个数)**

数组名称.length

**5.遍历数组元素**

for(var i=0;i<10;i++){ }

练习：遍历一个数组，下标从大到小显示

①通过for循环遍历数组元素的下标可以实现遍历数组元素。

②通过for-in来遍历数组的属性来实现遍历数组元素，数组也是对象。

练习：

创建一个空数组(员工)，添加员工的编号，姓名，年龄，性别，电话，部门；最后通过for-in遍历数组

**数组的分类：**

数组分为索引数组(下标是顺序号)，关联数组(下标是自定义属性);

练习1：

声明函数，在函数中传递参数(实参是数组)，实现获取所传入数组中的最大值，并返回；

getMax([20,38,80,50,70]); -> 80

function getMax(arr){

var max=arr[0];

//比较

return max;

}

练习2:

定义函数，求数组中的最小值，并返回

练习3:

定义一个函数，查找数组中指定元素的位置，并范围

[26,38,72,97,85]

6.数组中的方法

toString( ) 返回数组转换的字符串

join([参数]) 返回字符串，参数是数组元素之间的分割符，默认是逗号(,)

concat(参数1,[参数2],[参数3]) 拼接多个数组，返回拼接后的数组，参数是所要拼接的数组

slice(参数1,[参数2]) 截取数组的元素，返回截取后的数组；参数1开始截取的位置，参数2结束的位置(为空，截取的最后)，(start ~ end-1)

splice(参数1,参数2,参数3…参数n) 删除数组的一部分；参数1是开始的位置，参数2是删除的长度；参数3以后是删除后，所要替换的新元素；

reverse() 翻转数组中的元素，会改变原来数组。

sort([参数]) 对数组进行排序，默认是按照Unicode码从小到大；参数是指定排序方式

数字[小-大] sort(function(a,b){ return a-b; });

数字[大-小] sort(function(a,b){ return b-a; });

push(参数) 往数组的最后添加元素，会改变原来数组。

pop() 从数组的最后删除一个元素，会改变原来数组。

unshift(参数) 往数组的最前边添加元素，会改变原来数组。

shift(参数) 从数组的第一个删除元素，会改变原来数组。

7.二维数组

二维数组就是数组的嵌套，数组中每一个元素也是数组。

课后练习：

创建省份信息大数组

每一个小数组都是一个省份的信息，第一项保存省份，后边的每一项是城市

最后遍历出整个数组

[

['山东',…..]

] 复习

匿名函数

function fn(){ } //函数声明

var fn=function(){ } //函数表达式

创建函数

匿名函数自调用

(function(a){ })(1);

回调函数

function add(num){ num(); }

add(function(){ var a=1; return a; });

系统函数

encodeURI/decodeURI/parseInt/parseFloat/isNaN/

isFinite/eval

对象

属性和方法的集合

字面量 内置构造函数 自定义构造函数

{name:'tom',age:18}

var person=new Object();

person.name='tom';

person['age']=18;

遍历对象属性

for(var key in person){ person[key] }

'name' in person

person.hasOwnProperty('age') //true false

person.name===undefined //false true

方法

var person={say:function(){},name:'tom',run:function(){} }

person.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

学习目标——数组

数组是由多个元素组成的集合。

每一个元素就代表一个数据。

1.创建数组

(1)数组字面量

[元素1,元素2...]

数组中可以存放任意类型的数据

(2)内置构造函数

|  |
| --- |
| new Array(5) //初始化数组的长度为5，可以继续添加第6个元素...  new Array(元素1,元素2....); |

访问数组中的元素

数组名称[下标]，下标是从0开始

练习：创建数组，初始化长度为5，添加5个课程。

练习：创建数组，直接添加5个员工的姓名。

2.获取数组元素的个数

数组名称.length

使用数组长度添加一个新的元素

数组[数组.length]=值

练习：创建一个空数组，使用数组长度添加3个国家名称。

3.数组的分类

数组分为**索引数组**和**关联数组**

索引数组：以整数作为下标

关联数组：以字符串作为下标

|  |
| --- |
| var arr=[]  arr['eid']=1;  arr['ename']='tom'; |

4.遍历数组中的元素

for循环，循环遍历数组元素的下标

|  |
| --- |
| var scrore=[85,79,93];  for(var i=0;i<score.length;i++){  i 代表数组元素的下标  score[i] 下标对应的元素  }  注意：只能遍历索引数组。 |

for-in

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

练习：创建数组，保存所有员工的工资，使用for循环遍历计算平均工资。

练习：创建数组，保存4个汽车的品牌名称，把品牌名称为'宝马'改为'BMW';

使用循环来找到宝马的下标，使用下标改变元素的值为'BMW'

练习：创建数组，包含元素推荐、热点、娱乐、财经；分别使用for循环和for-in遍历该数组中的元素。

练习：创建一个函数getIndex，传递2个参数(数组，字符串)，返回字符串的下标，如果找不到该字符串的下标，则返回-1；

5.数组中的方法(API)

API-应用程序编程接口，预先定义好的函数/方法

toString() 将数组中的元素按逗号分隔成字符串

join('|') 将数组中的元素按照指定的字符分隔成字符串

concat(arr1,arr2...) 拼接两个或者更多的数组

slice(start, end) 截取数组中的元素，start开始的下标，end结尾的下标，不包含end本身；负数表示倒数。

splice(start, count,value1,value2....) 删除数组中的元素；start开始的下标，count删除的个数，value1,value...删除后补充的元素。

reverse() 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由小到大

|  |
| --- |
| 对数字排序  sort(function(a,b){  return a-b; //由小到大  //return b-a; //由大到小  }) |

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

将数组中的元素进行翻转，不能使用reverse

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

了解冒泡排序

(3)预习字符串下的方法(api)

#### DAY08 字符串操作

复习

创建

字面量 [ ]

内置构造函数 new Array('a','b'); new Array(5)

访问数组元素

arr.length

arr[arr.length]

下标 0 最后一个 arr.length-1

数组的分类

索引数组 下标是整数

关联数组 下标是字符串

遍历数组元素

for循环 遍历下标 0~arr.length-1

for-in

api、方法

转为字符串 toString() join('-')

拼接数组 arr1.concat(arr2,...)

截取数组元素 slice(start,end)

删除数组元素 splice(start,count,value1,value2...)

翻转数组元素 reverse()

数组元素排序 sort(function(a,b){ return b-a; })

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(方法)

push() 往数组的末尾添加元素，返回数组的长度

pop() 删除数组末尾的元素，返回删除的元素

unshift() 往数组的开头添加元素，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

数组中的每一个元素也是数组

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

访问二维数组中的元素 arr[下标][下标]

3.字符串操作

包装对象：目的是让原始类型的数据也可以像引用类型一样，具有属性和方法。

JS提供了3种包装类型：String、Number、Boolean

将任意数据包装成字符串对象

new String( true ) 返回object类型

String(true) 返回字符串 'true'

(1)转义字符 —— \

转换字符的意义

\n 将字符n转义成换行

\t 将字符t转义成制表符(tab键)

\' 将引号转成普通的字符

...

练习：打印出现 welcome to chi\na

(2)字符串中常用方法

toUpperCase() 将英文字母转为大写

toLowerCase() 将英文字母转为小写

练习：初始化4个英文字母(有大小写)保存变量中，循环弹出提示框，输入4个字符(不区分大小写)，如果输入正确结束循环。 06\_code.html 06\_code.js

length 获取字符串的长度

charAt() 获取下标对应的字符

charCodeAt() 获取某个字符对应的Unicode码

练习：使用变量保存字符串“javascript”，获取字符“a”出现的个数。

indexOf(value, start) 查找某个字符串的下标，value是要查找的字符串，start开始查找的下标，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，检测该字符串是否为邮箱格式；如果是打印true，不是打印false；(查看是否含有@)

slice(start, end) 截取字符串，start开始的下标，end结束的下标，不包含end本身；如果end为空，截取到最后

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

substring() ?

练习：使用变量存储身份证号，获取其中年月日和性别。打印1997年05月20日 男

110230199705203310倒数第2位 偶数->女 奇数->男

练习：变量保存邮箱地址，分别截取邮箱的用户名和域名

jerry1995@sina.com.cn

练习：将一个英文单词首字母转大写，其余转小写

'hELlo' -> 'Hello'

split(sep) 按照指定的字符分隔为数组，sep是分隔符

练习：使用split获取邮箱的用户名和域名(jerry@qq.com)

练习：将一句英文所有单词的首字母转大写，其余字母转小写； 'hOW arE yOu' -> 'how are you'

['hOW', 'arE', 'yOu'] -> ['How', 'Are', 'You']

4.匹配模式(掌握)

作用：用于查找、替换字符串。

replace(value1, values) 查找并替换。value1要查找的字符串，value2要替换的字符串；value1可以使用字符串形式，也可以使用正则表达式形式 /china/ig

i -> ignore 忽略大小写

g -> global 全局查找

match(value) 用于查找匹配的字符串，返回一个数组

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

5. Math对象

Math对象不需要使用new创建，可以直接使用。

PI 获取圆周率

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(x,y) 取x的y次幂

random() 取随机 >=0 <1

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

创建数组，存储若干个员工的姓名，每次运行随机取出1个员工的姓名打印出来

创建数组，包含a-z, A-Z, 0-9元素；在数组中随机取4个字符，放到一个新数组中。 push

(3)预习Date对象

#### DAY09 Date Number Boolean对象 ES6

复习

push()/unshift()/pop()/shift()

a-z A-Z

[ [],[],[]... ] 数组[1][3]

字符串

包装对象 new String() var str='123';

方法：toUpperCase()/toLowerCase()/charAt()/indexOf()/

lastIndexOf()/slice(start,end)/substr(start,count)

/substring()/split()

匹配模式：查找，替换

replace(/china/ig,'中国')

match(/china/ig) 数组

search(/china/i) 返回下标

Math对象

PI/abs()/ceil()/floor()/round()/max()/min()/pow(x,y)/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

1.字符串方法

String.fromCharCode(index) 获取任意Unicode码对应的字符

substring(start,end) 截取字符串，start开始的下标，end结尾的下标，不包含end本身；如果两个参数是负数，自动转成0.

2.Date对象

用于对日期和时间进行存储和计算

(1)创建Date对象

new Date('2018/11/11 10:20:30')

new Date(2018,10,11,10,20,30) //第2个参数月份的范围0~11

new Date() 存储当前的系统时间

new Date(1000\*60\*60\*24) 存储的是距离计算机元年的毫秒数对应的日期时间

(2)获取Date对象的信息

getFullYear/getMonth/getDate/getHours/getMinutes

/getSeconds/getMilliseconds/getDay(星期0~6)

/getTime(距离计算机元年毫秒数)

(3)转为本地字符串

toLocaleString() // 年-月-日 时:分:秒

toLocaleDateString() //年-月-日

toLocaleTimeString() //时:分:秒

(4)设置Date对象的信息

setFullYear/setMonth/setDate/setHours/setMinutes

/setSeconds/setMilliseconds/

setTime 设置距离计算机元年毫秒数，一旦使用后，年月日时分秒都会受到影响

练习：创建对象保存员工的**入职时间**'2018-11-16'，3年后到期计算**到期时间**，合同到期前1个月续签合同，假如**续签时间**是周末，提前到周五。提前一周通知人事准备续签(**提醒时间**)

3.Number对象

new Number(值) 创建对象

将一个数据转为Number对象，本质还是数字。

Number.MAX\_VALUE 获取计算机能存储的最大值

Number.MIN\_VALUE 获取计算机能存储的最小值

toFixed(n) 保留小数点后n位

toString() 将数字转为字符串类型

数字+'' //隐式转为字符串类型

4.Boolean对象

new Boolean(值) 创建布尔对象，本质上将数据转为布尔型

Boolean(值) 转为布尔型

!!值 隐式将数据转为布尔型

toString() 将布尔型数据转为字符串

**null的作用**：用于释放(销毁)一个引用类型的数据。

5.错误处理

SyntaxError 语法错误，错误的使用了中文，缺少括号...程序不会执行。

ReferenceError 引用错误，使用未声明的变量

TypeError 类型错误，错误的使用了数据的类型，错误的使用括号

RangeError 范围错误，参数的使用超出了范围

new Array(-1)

引用错误、类型错误、范围错误出现后，会影响后续代码的执行。

**异常处理**

|  |
| --- |
| try{  尝试执行的代码，可能出现错误  }catch(err){  err：捕获的错误信息  处理错误的方案  } |

6.ES6新特征

**E**CMA**S**cript6

ES2017 ES2018...

(1)块级作用域——局部作用域

使用let关键字声明的变量，只能在块级作用域下访问，不能被外部访问。

块级作用域： { }、 for、while、do-while都是块级作用域

(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致

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

如果箭头函数的函数体中只有一行代码，并且是以return形式，可以简化为

sort((a,b)=>a-b)

es6电子版

《ES6标准入门》第三版

http://es6.ruanyifeng.com

课后任务：

(1)复习今天内容，整理思维导图

(2)练习：

双色球：随机取1~33之间的6个数字不能重复，并且是从小到大排序，取1~16之间的1个数字（最后），组成一个新数组。

(3)预习服务器概念，NODEJS中的模块

### 3、Node.js基础

#### DAY01 全局对象 模块系统

学习目标：

(1)全局对象

(2)模块系统

1.全局对象

JS的全局对象 window

NodeJS的全局对象 global

**注意：**在nodejs交互模式下创建的变量或函数时全局的，而脚本模式下创建的属于局部的。

(1)console

console.time('LONG-LOOP');

检测的代码

console.timeEnd('LONG-LOOP');

作用：检测代码的运行时间。

练习：检测一组while的时间，对比for循环，哪个运行效率高。

(2)process

process.arch 查看CPU架构类型

process.platform 查看操作系统的类型

process.env 查看操作系统环境变量

process.cwd() 查看当前的工作路径

process.version/versions 查看当前NodeJS的版本信息

process.uptime() 查看nodejs运行时间

process.memoryUsage() 查看内存的使用情况

process.pid 获取当前进程的id

process.kill() 杀死某一个id对应的进程

(3)全局函数

定时器

①一次性定时器

setTimeout( func, delay ) 在一段时间后执行一次

func 要执行的任务，是一个回调函数，格式可以是箭头函数

delay 间隔的时间单位是毫秒

clearTimeout(timer) 清除一次性定时器

需要把开启的定时器放入到timer中。

②周期性定时器

setInterval(func, delay) 创建周期性定时器

func要执行的任务

delay间隔时间，单位毫秒

clearInterval(timer) 清除周期性定时器

需要把开启的定时器放入到timer中

练习：创建周期性定时器，打印10次'hello world'后，清除定时器。

③process.nextTick(()=>{ }) 开启定时器

④setImmediate(()=>{ });

clearImmediate(timer);

同步和异步

同步：程序按照顺序执行，先执行前边，后执行后边的。

异步：程序在执行的过程中，遇到异步函数等异步操作，会把这些操作放到主程序的最后边，主程序会跳过这个继续往后执行。

2.模块系统

|  |
| --- |
| 在nodejs中的文件中，任何一个js文件都是一个模块，在模块中都自动含有一个function  (function(exports,require,module,\_\_filename,\_\_dirname){  //\_\_filename 文件的完整路径和名称  //\_\_dirname 文件的完整路径  // require 用于引入其它的模块  //exports/module.exports 导出某些模块，是一个对象  **程序员自己编码的代码**  }) |

分为三种模块：核心模块、第三方模块、自定义模块；

(1)核心模块，直接引入 require('模块名称');

练习:

创建主模块09\_main.js

创建子模块09\_circle.js

在子模块中创建计算圆的周长和面积两个方法，并导出；

在主模块中引入子模块中的方法，并计算。

课后练习

创建主模块，引入子模块，调用方法计算周长和面积。

创建子模块，创建方法计算长方形的周长和面积，并导出

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新特征

(1)函数中的参数

ES6允许为形参设置默认值，如果没有传递实参，自动调用形参的默认值

(2)模板字符串

|  |
| --- |
| ` 在此之间可以写任意形式的代码 ${JS语法} ` |

练习：创建一个员工对象，包含姓名、性别、生日、工资；使用模板字符串打印员工的信息。

'姓名：\*\*，性别：\*\*，生日：\*\*，工资：\*\*'

2.nodejs概述

nodejs基于谷歌V8引擎(JS解释器)，运行在服务器端的语言，基于JS。

http://nodejs.cn 中文

http://nodejs.org 英文

**对比JS和NODEJS**

(1)JS运行在浏览器端，存在多个浏览器，容易产生兼容性的问题；而NODEJS在服务器端只有一个运行环境，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)

(3)JS用于网页中的交互效果，而NODEJS用于服务器的数据库操作、文件操作...

**NODEJS的执行方式**

脚本模式 node c:/xampp/..../1.js

交互模式

node

两次ctrl+c 或者输入 .exit

3.全局对象

**NODEJS: global**

在交互模式下，声明的变量或者创建的函数都属于全局对下的，可以使用global访问，例如 var a=1; global.a

在文件中声明的变量或者创建的函数都属于是局部作用域下的，不能使用global来访问。

**JS: window**

在浏览器下，文件中声名的变量或者创建的函数都属于是全局作用域下的，可以使用全局对象访问；

例如: var a=1; window.a

(1)console对象

global.console.log() 打印消息

global.console.info() 打印消息

global.console.warn() 打印警告消息

global.console.error() 打印错误消息

global.console.time('自定义字符串') 开始计时

global.console.timeEnd('自定义字符串') 结束计时

自定义字符串前后要保持一致。

练习：使用计时查看for，while，dowhile循环10000的耗时。

(2)process对象

查看当前计算机的进程

process.arch 查看当前CPU架构 X64

process.platform 查看当前的操作系统 win32

process.env 查看当前计算机的环境变量

process.version 查看当前nodejs的版本号

process.pid 查看当前的进程编号

process.kill( ) 杀死某一个编号的进程

(3)Buffer对象

缓冲区：在内存中存储数据区域，存储网络传输时的资源

创建buffer

var buf=Buffer.alloc(5, 'abcde');

将buffer存储的数据转为普通字符

buf.toString()

(4)全局函数

parseInt/parseFloat/encodeURI/decodeURI/isNaN

isFinite/eval

①一次性定时器

|  |
| --- |
| 开启  var timer=setTimeout(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  clearTimeout(timer); |

②周期性定时器

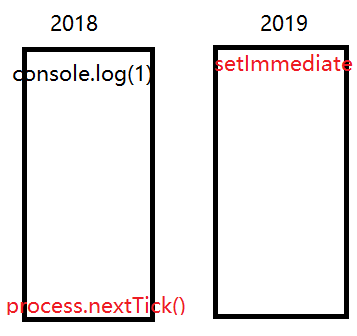
|  |
| --- |
| 开启  var timer=setInterval(回调函数，间隔的时候);  当间隔时间到了，执行回调函数；  清除  clearInterval(timer); |

练习：使用周期性定时器每隔3秒打印hello，打印三次后，清除定时器。

(3)立即执行(了解)

|  |
| --- |
| process.nextTick(回调函数)  在当前事件循环的结尾(2018年底)执行 |

|  |
| --- |
| var timer=setImmediate(回调函数)  clearImmediate(timer)  在下一个事件循环的开头(2019年初)执行 |



4.模块

模块可以理解为是一个功能体(积木块)

在NODEJS下模块分为自定义模块、核心模块(官方提供)、第三方模块

在NODEJS下，任意一个文件都是一个模块，文件中的代码默认是被一个构造函数所包含。

**以下代码中红色代码都是NODEJS自动为每一个文件添加的**

|  |
| --- |
| (function(exports,require,module,\_\_dirname,\_\_filename){  //程序员编码写的代码  }) |
| \_\_dirname 当前模块(文件)的完整路径  \_\_filename 当前模块(文件)的完整路径和文件名称  require() 引入一个模块  module 指代当前的模块对象  module.exports 当前模块的导出对象(公开)，可以供其它的模块使用的属性和方法  exports 等价于 module.exports |

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，分别获取圆的周长(getLength)和面积(getArea)，导出这两个函数；在主模块中引入功能模块，调用两个方法。

(3)预习 querystring、url、fs模块

#### DAY02 包和npm以及其他模块

复习

ES6新特征

函数的参数

function fn(a=1){ }

fn(2)

模板字符串

`${},' `

NODEJS

全局对象 global.fn() window

console log/info/warn/error/time('ab')/timeEnd('ab')

process arch/platform/env/version/pid/kill(1024)...

Buffer Buffer.alloc(5,'abcde')

setTimeout(回调函数, 间隔时间)/clearTimeout()

setInterval(回调函数, 间隔时间)/clearInterval()

process.nextTick(回调函数)

setImmediate(回调函数)/clearImmediate()

模块

自定义模块、核心模块、第三方模块

任意一个文件都是一个模块

exports require module.exports \_\_dirname \_\_filename

module.exports.a=1;

module.exports.fn=function(){ }

1.模块

|  |  |  |
| --- | --- | --- |
|  | 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的话，可以省略后缀名。 | require('querystring')  常用于引入官方提供的核心模块 |
| 目录模块 | require('./02\_2')  在02\_2目录下自动引入文件index.js；或者使用package.json文件声明main属性，来指定要引入的文件名称。 | require('04\_2')  要求引入的目录放在当前目录下node\_modules中。如果查找不到，则到上一级目录下查找，直到顶层目录。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(打印两个数字相加)，在03\_1.js中调用

练习：在05目录下创建模块05\_1.js，引入不带路径的目录模块05\_2，05\_2目录中含有hello.js文件(打印一句话)。

2.包和NPM

NPM: Node Package Manage

包(package): 就是一个目录模块，里边包含有多个文件，其中有一个文件命名为package.json的文件，是包说明文件。

自动下载 http://www.npmjs.com

**切换到下载的目录**

①cd 完整路径;

change directory

②在要下载的目录下，按住shift键，单击鼠标右键->在此处打开powershell窗口

**使用npm安装第三方包**

npm install 包的名称

3.核心模块

是NODEJS官方提供的模块，可以直接引入，不需要创建。

(1)查询字符串模块——querystring

浏览器向服务器发送请求，传递数据的一种方式

http://www.codeboy.com/product\_details.html?**lid=5&name=dell**

parse() 将查询字符串解析为对象

stringify() 将对象转换成查询字符串

练习：把百度搜索时的查询字符串解析为对象，获取关键词。

ie=utf-8&f=8&rsv\_bp=0&rsv\_idx=1&tn=baidu&wd=电脑

(2)URL模块

parse() 将url解析为对象

protocol 协议

hostname 主机(域名/ip地址)

port 端口

pathname 文件在服务器上的路径

query 查询字符串

format() 将对象转换成url

query属性对应的是对象

练习：浏览器请求的URL

https://www.tmooc.cn:3000/web/1810.html?sid=10&name=tom

获取URL中传递的sid和name的值

08\_exercise.js

(3)文件系统模块——fs

①fs.stat( path, callback )/fs.statSync(path) 查看文件的状态，通过回调函数来获取结果。

path 要查看的文件的路径

callback 回调函数，里边有两个参数

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

**对比同步和异步的区别？**

同步会阻止后续代码的执行，只有方法执行完，才能继续执行后边的代码；是通过返回值来获取结果。

异步不会阻止后续代码的执行，把执行的结果放到整个程序的最后；是通过回调函数来获取结果。

②fs.mkdir(path, callback)/fs.mkdirSync(path) 创建目录

path 要创建的目录的路径

callback 回调函数，只有一个参数

err 如果创建失败的错误信息

③fs.rmdir(path,callback)/fs.rmdirSync(path) 删除目录

path 要删除的目录的路径

callback 回调函数，获取删除的结果

err 如果删除失败的错误信息

④fs.readdir(path,callback) 读取目录中的文件

callback

files 读取的文件，返回数组

⑤fs.writeFile(path, data, callback) 写入文件/创建文件

data 要写入的数据

如果文件不存在，则创建文件，然后写入；如果文件已经存在，则清空文件中的内容，然后写入。

课后任务

(1)复习今天内容，整理思维导图

(2)练习

创建目录mydir，在该目录下创建文件1.txt，写入1；创建文件2.txt，写入2; 读取mydir下所有的文件；删除mydir（自学删除文件unlink）。

整个过程全部使用同步方法。

(3)预习http协议

#### DAY03 Node服务器搭建

复习

模块分类

|  |  |  |
| --- | --- | --- |
|  | 以路径开头 | 不以路径开头 |
| 文件模块 | require('./circle.js')  自定义模块 | require('url')  核心模块 |
| 目录模块 | require('./mydir')  自动寻找index.js文件  package.json文件寻找main属性对应的文件 | require('mysql');  到node\_modules中寻找目录名称为mysql的目录 |

npm install 包名称;

querystring parse() stringify()

url parse() format()

protocol hostname port pathname query

fs fs.stat/fs.statSync fs.mkdir/mkdirSync

rmdir/rmdirSync readdir/readdirSync

readdir(path, callback) var res=readdir(path)

writeFile(path,data,callback)

1.文件系统模块——fs

①fs.unlink(path,callback)/fs.unlinkSync(path) 删除文件

②fs.existsSync(path) 判断文件是否存储

存在true 不存在false

练习：判断文件num.txt是否存在，如果不存在则创建，并初始化数字0；在原来的数字上加1。

③fs.readFile(path,callback)/fs.readFileSync(path) 读取文件

返回的数据是buffer形式。

④fs.appendFile(path,data,callback)

fs.appendFileSync(path,data)

追加写入，如果文件不存在则创建文件，如果文件已经存在，则在末尾写入数据。

练习：使用文件操作来创建文件user.txt，每次写入一个对象,

{uid:1, uname:'tom', upwd: '123456'}

2.http协议

是浏览器和web服务器之间的通信协议。

(1)通用头信息

Request URL: 请求的URL，要向服务端请求哪个文件。

Request Method: 请求的方法 get/post

Status Code: 响应的状态码

2\*\* 服务器成功的响应

3\*\* 响应的重定向，跳转到另一个网址

4\*\* 客户端错误

5\*\* 服务器端错误

Romote Address: 请求的远程服务器的IP地址和端口

(2)响应头信息

Connection: keep-alive；连接的方式：持续连接

Content-Type: text/html; 响应的文件类型

Content-Encoding: 响应的文件压缩形式

Transfer-Encoding:响应时的传输方式，chunked(分段传输)

Location: 响应时跳转的URL，通常结合着300系列状态码。

(3)请求头信息

Accept: 客户端接受的文件类型有哪些

Accept-Encoding: 客户端接受的文件压缩形式

Accept-Language: 客户端接受的语言类型

Connection: 客户端和服务器的连接方式，持续连接

(4)请求主体

可有可无，客户端向服务器端传递数据

3.http模块

可以模拟浏览器向服务器端发请求，也可以创建web服务器

(1)模拟浏览器

http.get(url, callback)

get 请求的方法

url 请求的网址

callback 回调函数，用来获取服务器端的响应

res 响应的对象

res.statusCode 获取响应的状态码

res.on('data', (buf)=>{ })

使用事件来获取服务器端响应的数据

buf是服务器端响应的数据，格式为buffer数据。

(2)创建web服务器

var server=http.createServer() 创建web服务器

server.listen(3000) 分配端口，监听3000端口的变化

server.on('request', (req,res)=>{ });

//接收浏览器的请求，是一个事件，一旦有请求，自动执行

req 请求的对象

url 请求的路径，显示端口后的部分

method 请求的方法，直接通过地址栏默认使用get方法

headers 请求的头信息

练习：创建web服务器，监听3001端口，使用事件监听浏览器的请求，打印请求的方法、URL；

http://127.0.0.1:3001/admin/login.html

http://localhost:3001/member/shopping.html

05\_server.js

res 响应的对象

write() 响应的内容为文本形式，向浏览器中写入文本。

writeHead( 302,{ } ) 设置响应的状态码和响应的头信息；如果要跳转需要设置 Location属性。

end() 响应结束

练习：创建web服务器，监听3000端口；接收浏览器端的请求； 06\_server.js

/login 响应文本 this is login page

/member 响应文本 this is member page

/ 跳转到 /member

如果以上都没有匹配的， 响应文本 404 not found

3.express框架

基于NODEJS，用于构建web服务器的框架

官网：www.expressjs.com.cn

安装：npm install express

|  |
| --- |
| const express=require('express');  var server=express();  server.listen(3000); |

(1)路由

浏览器向web服务器发来请求，web服务器要根据请求的方法和请求的URL来作出响应。

路由三要素：请求的方法、请求的URL、响应的内容

响应的对象(res)

res.send() 响应文本，只能响应一次send；如果是数字认为是状态码。

res.sendFile() 响应文件，必须使用绝对路径(\_\_dirname)

res.redirect() 响应的重定向

课后任务

(1)复习今天内容，整理思维导图

(2)练习

使用express创建web服务器，创建以下路由

get '/index' 发送文本"这是首页"

get '/login' 发送文件 "login.html"

post '/register' 发送文本"注册成功"

get '/' 跳转到 /index

(3)预习express的中间件，浏览器向服务器传递数据方式有哪些(post，查询字符串，路由传递)

#### DAY04 中间件MySQL模块

复习

fs模块

stat/mkdir/rmdir/readdir/writeFile/appendFile/readFile/

unlink/existsSync

http协议

通用头信息 请求头信息 响应头信息 请求的主体

http模块

http.get(url,(res)=>{

statusCode

res.on('data', (buf)=>{ })

})

web服务器

var server=http.createServer()

server.listen(3000)

server.on('request', (req,res)=>{  
 req url/method/headers

res write/writeHead(302,{ Location:'/detail' })/end()

})

express框架

创建web服务器

var server=express()

server.listen(3000);

创建路由

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

res.send()/sendFile(\_\_dirname+'/login.html')/redirect('')

});

1.路由中的请求对象

req.method 获取请求的方法

req.url 获取请求的URL

req.headers 获取请求的头信息

req.query 获取请求时以查询字符串形式传递的数据，返回格式为对象。

练习：创建文件03\_post.js，创建web服务器，新建路由(get /reg)，响应一个注册文件(reg.html)

2.post和get请求

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

post请求是通过表单提交(现阶段)的方式传递数据，服务器端通过事件形式获取数据(后期会有简单的方法)

|  |
| --- |
| req.on('data', (buf)=>{  获取的结果是buffer数据，需要使用查询字符串解析为对象  }) |

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',(req,res)=>{  req.params //获取路由传递的数据，格式为对象  }) |

浏览器请求方式

http://127.0.0.1/detail/5

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URL：/shopping，请求的方法get，传递商品的价格(price)和名称(pname)。

用户模块

用户列表 /list 详情 /detail 删除 /delete

/user/list /user/detail /user/delete

商品模块

商品列表 /list 详情 /detail 删除 /delete

/product/list /product/detail /product/delete

4.路由器

路由在使用过程中，不同的模块可能出现相同的URL，把同一个模块下的路由挂载到特定的前缀。

例如：商品模块下的路由挂载到/product，访问形式/product/list，用户模块下的路由挂载到/user，访问形式/user/list

路由器就是一个js文件，把同一模块下的路由放到一起。

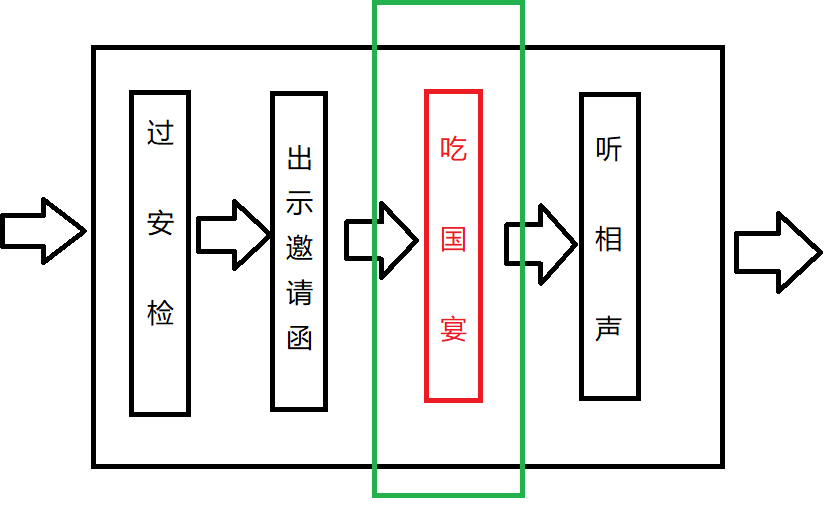
|  |
| --- |
| const express=require('express');  var router=express.Router(); //创建空的路由器对象  router.get('/list', (req,res)=>{ }); //往路由器中添加路由  module.exports=router; |

在web服务器下使用路由器

|  |
| --- |
| const userRouter=require('./user.js'); //引用路由器模块  server.use('/user', userRouter);//把路由器挂载到/user下，访问形式 /user/list |

练习：创建商品模块路由器(product.js)，添加路由商品列表(list)、商品删除(delete)、商品添加(add)，在web服务器引入，并挂载到/product

5.中间件



中间件的作用为主要的业务逻辑所服务。

**分为5个**

应用级中间件、路由级中间件、内置中间件、第三方中间件、错误级中间件

(1)应用级中间件

每一个中间件就是一个函数，需要配合其他的中间件或者路由使用。

server.use(回调函数) 拦截所有的路由

server.use('/detail', 回调函数) 拦截特定的路由

练习：创建路由(get，/view)响应当前的浏览次数，每次请求，响应的次数加1。

在函数外初始化一个变量，设置值为0; 在中间件中实现变量加1，在路由中响应变量。

(2)路由级中间件

用于在服务器中将路由器挂载到特定的URL

server.use('/user', userRouter);

(3)内置中间件

在express中只有一个内置的中间件

server.use( express.static('要托管的目录') )

托管静态资源到某一个目录，如果浏览器端要请求静态资源，则自动到这个目录下查找。

静态资源：html、css、客户端js、图像...

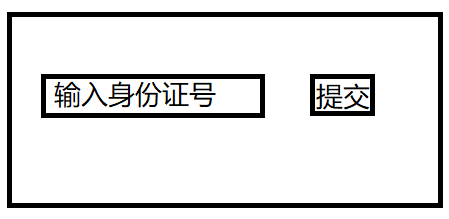
练习：将静态资源托管到files目录下，查看如果两个静态目录下有相同名称的文件，显示哪一个？

课后任务

(1)复习今天内容，复习MySQL中的SQL语句

(2)练习：

创建web服务器，托管静态文件（如下图），点击查询，服务器端获取输入的身份证号(中间件)，截取出生的年月日和性别；在路由中响应给浏览器。



(3)预习mysql模块的使用

https://www.npmjs.com/package/mysql

#### DAY05-DAY06项目练习

xz项目结构

node\_modules 项目所依赖的包

public 静态资源目录(html, css, img, 客户端js)

routes 项目中的路由器

app.js 项目入口文件

pool.js mysql连接池文件，在路由器中使用

练习：

用户检索，在 user\_query.html，输入框(用户编号)，点击提交，在客户端显示用户的这条数据。

select\*from xz\_user where uid=?;

课后练习：

创建修改用户信息(user\_update.html)，修改用户的user\_name,gender, phone, email四项，还需要提供用户的编号。检测输入的值，返回对应的状态码

复习

1.第三方中间件body-parser

将post请求的数据解析为对象

使用中间件

|  |
| --- |
| server.use(bodyParser.urlencoded({  extended:false  }))  urlencoded 可以将post请求的数据解析为对象  extended:false 不使用第三方中的qs模块，而是使用核心模块querystring将数据解析为对象 |

在路由中获取数据

req.body 返回一个数据的对象格式

**浏览器向服务器端传递数据的方式**

get req.query

post req.body

路由传参 req.params

2.MySQL模块

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

删 DELETE FROM emp WHERE uid=5;

改 UPDATE emp SET uname='jerry',sex=1 WHERE uid=5;

查 SELECT \* FROM emp;

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

(1)普通连接mysql

var connection=mysql.createConnection({ }); 创建连接对象，参数中传递mysql服务器的主机、端口、用户名、密码、选择使用的数据库。

connection.connect(); 执行连接

connection.query(sql,callback); sql表示要执行的SQL语句，callback回调函数，用于获取SQL语句的执行结果。

connection.end(); 执行完所有SQL语句，关闭连接。

(2)使用连接池

var pool=mysql.createPool({ });创建连接池对象，需要提供服务器的主机、端口、用户名、密码、使用的数据库，以及设置连接池的大小 connectionLimit

pool.query(sql, callback); 执行SQL语句；sql是执行的SQL语句，callback获取SQL语句的执行结果。

练习：删除编号为5的员工的信息，查看结果；修改编号为7的员工的姓名为 '张三'，生日为'1980-5-1'，查看结果。

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，向服务器端发送请求(/add，post)

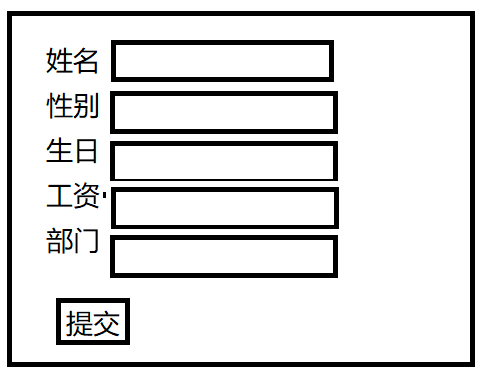
05\_add.js

课后任务

(1)复习今天内容，整理思维导图

(2)练习：

通过表单往数据库中插入员工信息 tedu下emp表



www.codeboy.com/pro

## 阶段2: HTML、AJAX、CSS、BOOT

### 1、HTML

#### DAY01 HTML

自我介绍

李然 QQ：1535212067

一.课程安排

|  |
| --- |
| 0.数据库+服务器(准备数据，准备服务器)  1.HTML5 Basic 2天+(搭建网页结构，可以显示数据，很丑)  2.Ajax异步数据交互 3天(页面和服务器的交互，为了得到数据在页面中显示)  3.Ajax项目 2天  4.CSS3 4+2天 （负责给页面化妆，让页面变得好看）  5.BootStrap框架 4天  6.Boot项目3天 |
| 面临的问题  1.知识点细碎，没有条理  2.单词量巨大，关键单词巨长  3.键速  4.笔记不要跟抄， |

二.Web基础知识

|  |
| --- |
| HTML5：HTML4.01升级版，大前端技术，泛指前端的所有技术 |
| HTML4.01在99年12发布  XHTML1.0在2000年1月发布，语法更严谨  <img> <img/> |

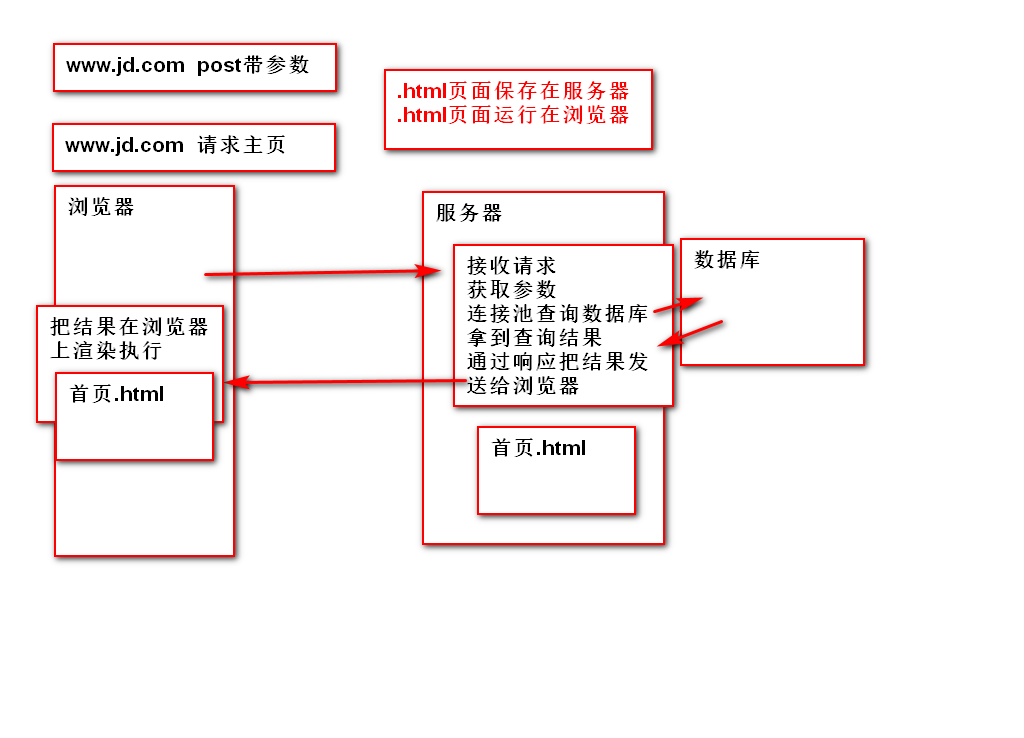
1.web与Internet

|  |
| --- |
| 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），访问网站的服务  BBS：电子公告版，俗称论坛  Email:电子邮件  telnet:远程登录  FTP:文件的上传下载 |

2.internet上的应用程序

|  |
| --- |
| 1.C/S client 客户端/ server 服务器  2.B/S Browser 浏览器 / server 服务器 |
| C/S和B/S的区别  C/S需要升级 B/S不需要升级  我们工作主要是B/S |

3.web运行原理



|  |
| --- |
| web:运行在internet上一种B/S结构的应用程序，俗称网站  Internet：为web提供网络环境  web的工作原理：  基于浏览器和服务器继续通信协议来实现数据的传输和展示 |
| 通信协议：规范了数据和打包和传送的 |
| 服务器：  1.功能  存储数据，接收用户的请求并给出响应、  提供了程序的运行环境  具备一定的安全功能  2.服务器产品  tomcat(java)  Apache(php)  IIS(.net)  3.服务器端的技术  PHP/JAVA/.NET/python/nodejs |
| 浏览器  1.功能  代表用户发送请求  作为HTML、css和js解析器  以图形化的解码展示给用户  2.浏览器产品  Chrome  FireFox  IE--edge  Oprea  Safari  3.浏览器支持的技术  H5,CSS,JS |

三.HTML快速入门

1.什么是HTML

|  |
| --- |
| HTML:HyperText Markup Language 超文本标记语言  a 普通文本  <a></a>超文本标记  标记：超文本的组成形式 <关键字>  具有自己的功能  语言：有语法结构 |

2.HTML的特点

|  |
| --- |
| 1.以.html或.htm为后缀  2.由浏览解析执行  3.用带有尖括号的"标记"来标识  4.可以在网页中执行js脚本 |

3.HTML的基础语法

|  |
| --- |
| 标记，标签，元素，节点，对象  主要是表示特殊的功能  标记在使用的时候，必须使用<>包裹起来 |
| 标签的分类  1.双标记(封闭类型)  <关键字></关键字>  双标记必须成对出现，有开始就有结束 <p></p>  2.单标记(非封闭类型)  <关键字>或者<关键字/> |

练习

|  |
| --- |
| 创建一个text文档，名称为first.html  在文档中写出标记  写一对html标记  一对head标记  一对title标记 |

2.标签的嵌套

|  |
| --- |
| 1.什么是嵌套  一对标记中出现另一个(对)标记，形成了层级。  <div>  <p></p>  </div>  2.语法  <关键字1>  <关键字2>  <关键字3></关键字3>  </关键字2>  </关键字1> |

3.属性和值

|  |
| --- |
| 允许通过属性和值对标记进行修饰  语法：  1.属性和值必须声明在开始标记中  <关键字 属性="值"></关键字>  2.多个属性之间用空格分开  <关键字 属性1="值1" 属性2="值2" ...></关键字> |
| 3.标准属性(通用属性)  所有的元素都支持的属性  id 定义此元素在页面中的唯一标识  title 鼠标悬停在元素上时，显示的文本  style css中，定义行内样式  class css 中，引用类选择器 |

学习HTML，就是学习固定关键字，学习此关键字作用

学习固定的标签属性，学习嵌套关系

4.注释

|  |
| --- |
| 语法：<!--注释内容-->  注意：注释不能嵌套注释，注释也不能出现在标记中 |

四.HTML的文档结构

1.HTML文档的构成

|  |
| --- |
| 1.文档类型的声明  <! html>告诉浏览器，我这篇html使用h5的规则解析 |
| 2.网页的结构  <!doctype html>  <html>  <head></head>  <body></body>  </html>  <html></html>表示网页的开头和结束  在html中包含两个标签  <head></head> 网页的头部，定义全局的信息  <body></body> 网页的主题，展示的内容 |
| 3.head元素是其他头元素的容器<head></head>  <title></title> 网页的标题  <meta/> 元数据，定义网页的全局信息    <meta charset="utf-8">  chrome 62以上的版本，自动识别中文  <meta name="Keywords" content="关键词的内容">  <meta name="description" content="描述的内容">  <style><style>定义网页的内部样式  <script></script>定义或者引用js文件  <link/> 引用外部样式文件 |
| 4.body元素  定义网页的主体  body的属性  bgcolor 背景颜色  text 字体颜色  取值：合法的颜色值---->颜色的应为单词 |

练习

|  |
| --- |
| 在01\_ex中，把所有代码删除  1.重新编写一遍，html的基本结构  2.在head中添加网页标题，添加utf-8  3.在body中添加任意文字，改变body背景颜色，和字体颜色  red green yellow black white pink blue |

五.文本标记

1.标题元素

|  |
| --- |
| 作用：在页面中以醒目的方式显示文本  语法 <h1></h1>一级标题 <hn></hn> n:1~6  ...  <h6></h6>六级标题  特点：  1.文本加粗  2.字号发生变化  3.独占一行，上下有垂直间距  属性align 设置标签内容水平方向对齐方式  取值：left/center/right |

2.段落标记paragraph

|  |
| --- |
| 语法：<p></p>  作用：突出的表示一段文字  特点：独立成行，上下有垂直的间距  属性：align |

练习：模拟个人简历

3.换行标记

|  |
| --- |
| <br>或者<br/> |

4.水平线(分割线)

|  |
| --- |
| <hr>或者<hr/>  width="50%" 表示水平线的宽度，取值为%或者px为单位的数字  size="5px" 表示水平线的尺寸(粗细)，取值为%或者px为单位的数字  align="left" 对齐方式  color="red" 设置水平线的颜色 |

5.预格式化标签

|  |
| --- |
| <pre></pre>  标签内部的文本，保留其格式，在页面上显示  空格折叠现象，一个或多个空格，会被浏览器渲染成一个空格显示  一个或多个回车，会被浏览器渲染成一个空格显示 |

6.特殊字符(实体)

|  |
| --- |
| 空格 &nbsp；  < &lt;  > &gt;  © &copy；  ® &reg;  ￥ &yen;  x &times; |

练习

|  |
| --- |
| Copyright © 2004 - 2018  <京东JD.com> 版权所有  p标签中输出上面这句话 |

7.文本样式

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

8.分区元素

|  |
| --- |
| 1.块分区，用于页面中布局  <div></div>  独立成行 |
| 2.行分区，处理同一行文本的不同样式  <span></span>  可以与其它的span和文字共用一行 |

9.行级元素和块级元素（元素的显示形式）

|  |
| --- |
| 1.块级元素  在网页中，单独成行，就是块级元素  从上往下排列  常见块级元素  h1~h6  p  div |
| 2.行内元素(行级元素)  在页面中，与其他行内元素/行内块共用一行  从左往右排列  页面中常见行内元素  i em b strong u sup sub span |
| 3.行内块元素  显示的方式与行内元素相同，但是具备块级元素的特征 |
| 4.table  表格的宽高是由表格的内容决定的  内容多，就宽一些，高一些，  表格中所的数据都预读到内存，一次性画到页面上 17：15回来 |

六.图像和链接

1.图像

|  |
| --- |
| 1.使用  <img>或者<img/>  属性  src="url" url--图片资源路径 |
| 2.URL  Uniform Resource Locator 统一资源定位符 简称：路径 |
| 3.URL路径的表现形式  1.绝对路径，一个写完整的路径  通信协议+主机地址+文件目录结构+文件的具体名称  合适使用：  1.使用网络资源的时候，使用绝对路径  优点：不占据自己服务器的存储空间  缺点：不稳定  2.使用本地资源，使用绝对路径(项目中100%用不到)  从最高盘符开始查找 C/D/E/F/G  项目中不用 |

练习：从tmooc再盗一张图片，放到img中显示

|  |
| --- |
| 打开tmooc，按f12---elements左边有个小箭头  鼠标去页面中选中元素，右侧会显示对应的代码  鼠标去src的图片路径上右键点击，open in new tab  把新页面中的地址栏所有内容复制到自己的img的src中 |
| 2.相对路径（参照物是html本身）  1.图片与html文件在同一个目录下，src直接引用图片名称  src="3.jpg" |

|  |
| --- |
| 开发小技巧  1.标签只写完整的第一个，按tab补全  2.属性如果拼写正确，会变色  3.属性值可以不添加"" 和 ''，但是不推荐裸写值  4.每次修改要记得保存 ctrl+s  5.页面刷新 F5  6.添加eclipse快捷方式  tools--首选项--keyboard--最右边load加载1.key  退回 ctrl+z  前进 ctrl+y  删除当前行 ctrl+d  整行复制 ctrl+alt+↓  整行位移 alt+↓  其他快捷方式  windows+d  alt+f4 |

作业：

1.Nodejs服务器代码

2.完成下图



#### DAY02 图像&列表&表格

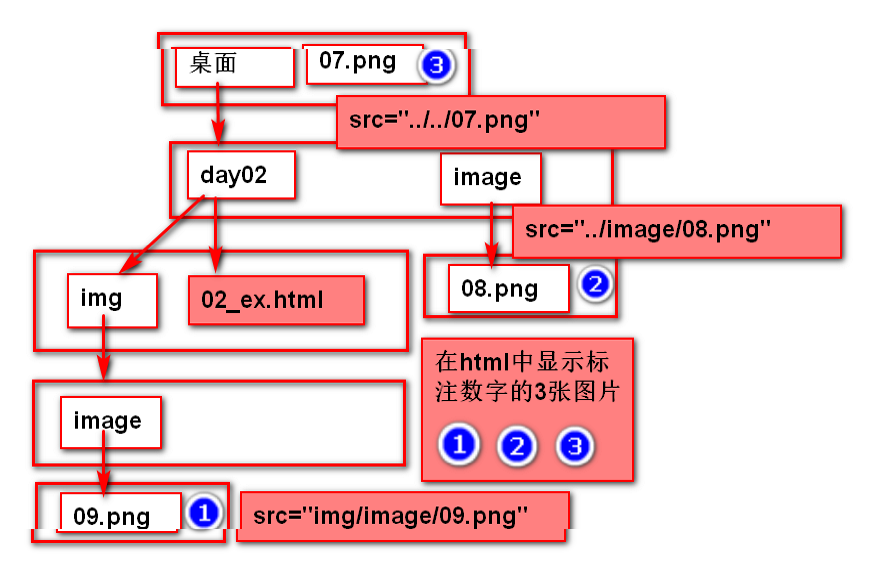
作业

一.图像和链接

1.URL

|  |
| --- |
| 1.绝对路径  1.1完整的路径  使用网络图片资源  通信协议+主机名称+文件结构目录+文件名称  http://www.codeboy.com/img/index/banner1.png  1.2本机资源  从最高盘符开始查找  c:\xxx\xxx\1.jpg  项目中不用 |
| 2.相对路径---参照物，本html文件    2.1在同一个目录下(兄弟关系)，直接引用兄弟文件  <img src="08.png" alt="">  2.2子集目录，文件在兄弟文件夹的内部，先找兄弟，在找兄弟的孩子  <img src="image/09.png" alt="">  2.3父级目录，文件是父级的兄弟，  先找父亲(../),再找父亲的兄弟  <img src="../07.png" alt=""> |

练习：



|  |
| --- |
| 3.属性  src:source 源，设置资源路径  alt:图片资源出错，显示的文字  width:设置图片的宽度，以px为单位的数字  height:设置图片的高度，以px为单位的数字  由于宽高比设置的不对，会发生图片失真的情况  解决：宽高只设置一个，另外一个自适应 |

2.链接

|  |
| --- |
| 1.语法 <a href="#"></a>  <a href="http://www.tmooc.cn/">Tmooc</a>  属性：  href:链接的路径  target:指定打开链接的方式  取值：\_self 默认值，在当前页面标签页红打开新链接  \_blank 在新的标签页打开链接 |
| 2.其它的表现形式  <!--下载 -->  <a href="1.zip">下载</a>  <!--新建邮件，前提：本机必须有邮件软件的客户端 -->  <a href="mailto:chengliang@tedu.cn">新建邮件</a>  <!--执行js脚本 -->  <a href="javascript:alert(1111111)">执行js</a>  <!--返回当前页面的顶部 -->  <a href="#">返回顶部</a> |

3.锚点

锚点就是网页中的一个记号

|  |  |
| --- | --- |
| 定义锚点  使用a标签的name属性定义锚点  <a name="锚点名称"></a> | 定义锚点  任意标签的id属性定义锚点  <any id="id"> </any> |
| 链接到锚点  使用a标签链接锚点，#锚点名称  <a href="#锚点名称">回到锚点</a> | 链接到锚点  使用a标签链接锚点， #id  <a href="#id">回到锚点</a> |

二.表格

1.表格语法

|  |
| --- |
| <table>  <tr>  <td></td>  <td></td>  <td></td>  </tr>  .....  </table> |

练习：

|  |
| --- |
| 04\_table.html创建一个4\*4的表格，内容1-16 |

2.属性

|  |
| --- |
| 1.table标签的属性  border:设置边框  width:设置表格的宽  height:设置表格的高  可以设置宽高  内容小于宽高的，按宽高尺寸  内容大于宽高，按内容改变尺寸  align:设置表格的对齐方式(table自己的水平位置)  取值 left/center/right  bgcolor:背景颜色  cellpadding:设置单元格内边距(边框到内容的距离)  cellspacing:设置单元格外边距（边框与边框之间的距离） |

练习

|  |
| --- |
| 04\_ex.html  设置表格400px\*400px，内容自拟  表格背景颜色 pink  表格边框1px  表格水平居中显示  表格内边距5px 表格外边距10px |
| 2.tr的属性  1.align设置当前行内容的水平对齐方式（设置表格内容的对齐方式）  2.valign设置当前行内容的垂直对齐方式  取值：top/middle/bottom  3.bgcolor 当前行的背景颜色 |
| 3.td的属性  width  height  align  valign  bgcolor  colspan  rowspan |

3.不规则表格的使用

|  |
| --- |
| 1.跨列  colspan=?  从指定的单元格位置处开始，横向向右合并?个单元格(包含自己)  被合并掉的单元格，要删除 |
| 2.跨行  rowspan=?  从指定的单元格位置处开始，纵向向下合并?个单元格(包含自己)  被合并掉的单元格，要删除 |

4.table中的可选标记

|  |
| --- |
| 1.表格的标题  <caption></caption>  如果设置标题，必须将<caption>放在<table>之后  2.行/列的标题  <th></th> 代替<td></td>  加粗，居中 |

5.表格的复杂应用

|  |
| --- |
| 1.行分组  可以将连续几个行，划分到一个组中，进行统一管理  表头行 <thead></thead> 表格最上面一行的标题放在thead中  表主体 <tbody></tbody>允许将若干放在tbody中进行统一管理  表尾行 <tfoot></tfoot> 表格中最后一行放在tfoot中 |
| 2.表格的嵌套  一张表格中，所有的嵌套的表格，只能放在td中 |

三.列表(重点\*\*\*)

|  |
| --- |
| 默认按照从上到下的顺序来显示数据，并且可以在数据前添加标识  使用列表布局(去掉标识，从上到下，从左到右排列) |

1.列表的组成

|  |
| --- |
| 1.有序列表<ol></ol> order list |
| 2.无序列表<ul></ul>unorder list |
| 列表项 <li></li> list item |

2.列表属性

|  |
| --- |
| 1.有序列表的属性 ol  type 设置标识符的类型  取值 ： 1 默认值，数字  a A 字母  i I 罗马数字  start 指定起始编号  取值：无单位数字 |
| 2.无序列表的属性 ul  type：设置标识  取值 disc 默认值 实心圆  circle 空心圆  square 实心方块  none 无 |

3.列表的嵌套

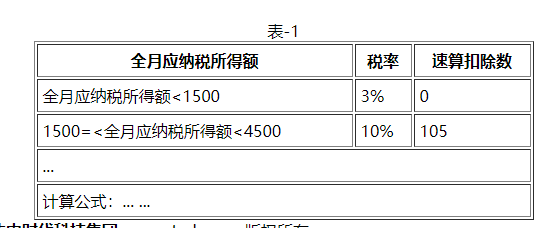
|  |
| --- |
| 所有的嵌套必须只能写在li中 |

练习：嵌套列表

作业：

1.nodejs服务器和连接数据的代码

2.完成表格



### 2、AJAX

#### DAY01 表单

一.列表

1.定义列表

|  |
| --- |
| 1.作用  做名词解释使用  长由于图文混排  <dl>  <dt>需要被解释说明的名字</dt>  <dd>对名词的解释内容</dd>  </dl> 10:17上课 |

二.结构标记

1.结构标签的作用

|  |
| --- |
| 代替div，用于描述整个网页的结构，提升标记的语义 |

2.常用的结构标记

|  |
| --- |
| <header></header>  定义网页或者某个区域的头部内容 |
| <nav></nav>  定义网页的导航链接部分 |
| <section></section>  定义网页的主体内容 |
| <aside></aside>  定义网页的侧边栏 |
| <footer></footer>  定义网页底部内容，多用于版权，解释说明，备案号 |
| <article></article>  定义与文字相关的内容  论坛，帖子，微博条目，用户评论 |

三.表单(重点\*\*\*\*\*\*\*\*&难点\*\*\*)

1.作用

|  |
| --- |
| 1.提供可视化的输入控件  2.收集用户填写的信息并提交给服务器 |

2.表单的组成部分

|  |
| --- |
| 1.前端部分  提供表单的控件，与用户做交互的可视化控件  2.后端部分  对提交的数据进行处理(接口) |

3.表单

|  |
| --- |
| <form></form> |

属性

|  |
| --- |
| 1.action  定义表单被提交时发送的动作  通常是定义服务器处理数据的地址，接口(url)  默认，提交给本页地址 |
| 2.method  指定表单数据提交的方式(方法)  1.get(默认值)  明文提交，提交时内容会显示在浏览器地址栏上  提交的数据有大小限制，2kb  向服务器要数据的时候使用get  2.post  提交数据无大小限制  隐式提交，提交的数据不会显示在地址栏中  把数据传递给服务器的时候，使用post  3.delete  4.put |
| 3.enctype 编码方式  设置表单允许将什么样的数据提交给服务器  1.提交普通字符，不能有特殊符号  enctype ="text/plain"  2.允许将任意字符提交给服务器  application/x-www-form-urlencoded  3.允许将文件提交给服务器  multipart/form-data |

4.表单控件

|  |
| --- |
| 能够与用户进行交互的可视化元素 |

4.1分类

|  |
| --- |
| 1.input元素  2.textarea 多行文本域  3.下拉选择框  4.其它元素 |

4.2input元素

|  |
| --- |
| 在页面中提供各种各样的输入控件，通过type属性来设置输入控件的类型  <input>或者<input/>  type 设置输入控件的类型  name 为控件定义名称，提供给服务端使用(必须值)  vaule 控件的值，提交给服务端使用  disabled 禁用控件，不能操作也不能提交  无值属性 |

type取值

|  |
| --- |
| 1.文本框和密码框  type="text" 默认值  type="password"  属性  1.maxlength 限制输入的最大字符数  2.readonly 只读，只能看不能写，但允许被提交，无值属性  3.placeholder 占位符，默认显示在控件上文本，只要得到焦点，文本就无效了，此文本不能提交 |
| 2.按钮  type="submit"提交按钮  将表单中的数据，提交给服务器  type="reset"重置按钮  让表单内容恢复到初始化状态  type="button" 无功能按钮，可以调用js脚本 |
| 3.单选按钮和复选按钮  单选按钮  type="radio"  多选按钮  type="checkbox"  name属性：除了定义控件名称，还起到分组的效果  必须有value，不然提交的值是on  checked：设置默认选中项，无值属性  checkbox的name要写成数组hobby[] |
| 4.隐藏域  type="hidden"  想提交给服务器，但是不想展示给用户看的数据，使用隐藏域 |
| 5.文件选择框  type="file"  注意：  method="post"  enctype="multipart/form-data"  multiple 无值属性，可以同时上传多个文件  需要name属性值为数组 |

4.3textarea文本域

|  |
| --- |
| 允许录入多行数据的文本框  <textarea></textarea>  name 控件的名称  readonly 只读  cols 指定文本域的列数，以英文单词为准，中文减半  rows指定文本域的行数， |

4.4下拉选择器框

|  |
| --- |
| <select> 表示一个下拉选  <option></option> 表示下拉选中的一个选项  .......  </select>  注意：  当option没有value属性时，提交的时候select的value为选中option的内容。  当option有value，提交时，select的value为选中option的value。  select属性  name选项框的名称  size 默认为1，定义显示选项的数量。  如果值大于1的数字。下拉选表现为滚动列表  multiple 无值属性，可以多选，但是name要用数组  设置了multiple之后，就算size="1" 也是滚动列表  option的属性  value  selected设置默认选中项 |

练习：



5.其它元素

|  |
| --- |
| 1.label  文本与表单控件的关联  <label></label>  属性  for:要与label中文字发生关联的form控件的id |
| 2.为控件分组  <fieldset></fieldset> 为控件分组  <legend></legend> 分组标题 |
| 3.浮动框架  允许在一个网页中，引入另外一个网页  <iframe></iframe>  属性  src="url"  width="100%"  height="200px" 高度存在问题，需要使用js.dom来解决  scrolling="no" 不显示滚动条  frameborder="0" 不显示边框 |

6.H5新表单元素

|  |
| --- |
| 什么是新表单元素  在HTML5版本中新提出来表单控件 |
| 1.电子邮件  <input type="email">  表单提交时，会验证数据是否符合email的规范(@以及@后面有内容) |
| 2.搜索类型  <input type="search">  提供快速清除文本框的功能 |
| 3.url类型  <input type="url">  提交时，验证数据是否符合url的规范,绝对路径的规范  验证http://以及后面必须有内容 |
| 4.电话号码  <input type="tel">  在移动设备中，显示拨号键的效果 |
| 5.数字类型  <input type="number">  只能接受数字,并且提供了箭头灵活的调整数字  属性  value 指定数字  min 能接收提交的最小值  max 能接收提交的最大值  step 步数，每次点击箭头，变化的数量 |
| 6.范围类型  <input type="range">  提供一个滑块组件，允许用户指定选取的范围  属性  value 指定初始数字  min 范围最小值  max 范围最大值  step 步数 |
| 7.颜色拾取器  <input type="color">  提供一个颜色的拾取器 |
| 8.日期类型  <input type="date">  提供了一个选择日期的控件，可以选择年/月/日 |
| 9.月份类型  <input type="month">  提供了一个选择月份的控件，可以选择年/月 |
| 10.周类型  <input type="week">  提供了一个选择周的控件，可以选择年/周 |

作业

1.nodejs服务器端代码

2.完成下图



#### DAY02 http

ajax的异步加载

一.HTTP协议

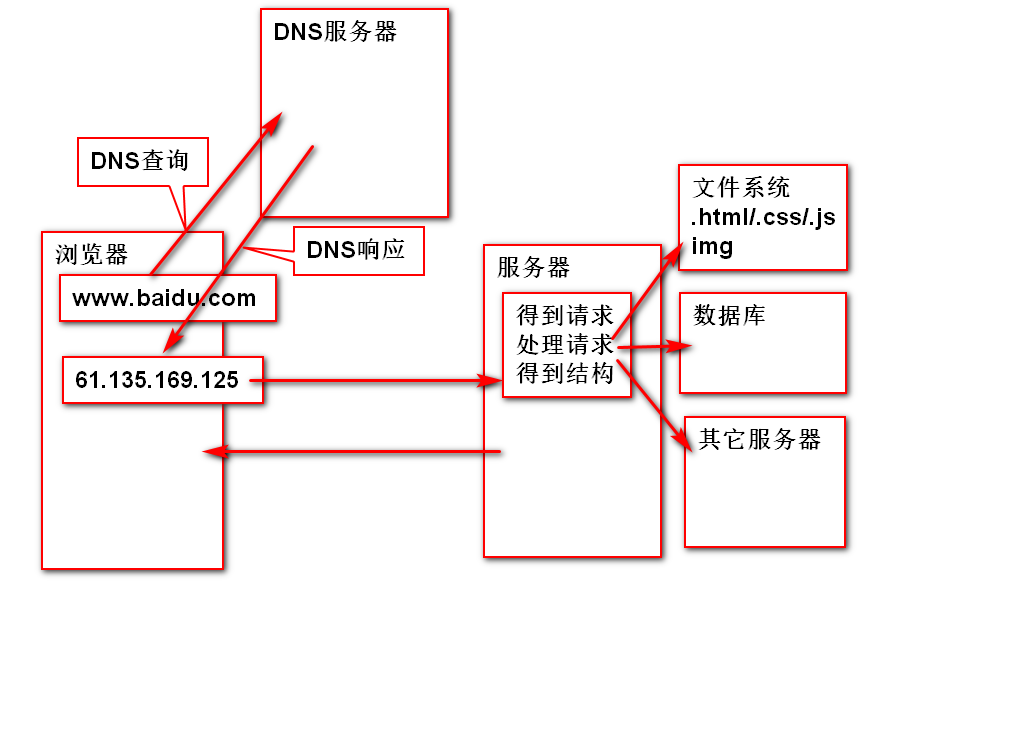
1.url

|  |  |  |
| --- | --- | --- |
| 结构：协议+主机名称+目录结构+文件名称  http://www.codeboy.com/admin/user\_list.html  url完整结构  <scheme>://<user>:<pwd>@<host>:<port>/<path>;<params>?<query>#<frag> | | |
| scheme 方案、协议，以哪种方式获取浏览器的资源  不区分大小写，常见协议，http/https/ftp | | |
| 常见的协议 | 默认端口号 | 基本作用 |
| ftp | 21 | 文件的上传下载 |
| SSH | 22 | 安全的远程登录 |
| TELNet | 23 | 远程登录 |
| SMTP | 25 | 邮件传输协议 |
| DNS | 53 | 域名解析 |
| HTTP | 80 | 超文本传输 |
| POP3 | 110 | 邮件接收 |
| HTTPS | 443 | 加密的传输http |
| <scheme>://<user>:<pwd>@<host>:<port>/<path>;<params>?<query>#<frag>  user：用户名  pwd：密码  host：主机名 localhost 127.0.0.1  port：端口号  path：资源在服务器上存放的位置  params: 参数，服务器端使用  query: 查询字符串 ?unmae=lilei&upw=1234567  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及传递的 |
| http的历史 现在使用的版本是http/1.1 不需要设置 |

3.详解



|  |
| --- |
| HTTP协议规定，客户端与服务器端传递的数据块Message（消息/报文）。必须符合特定的格式，这个Message才能被客户端和服务器彼此理解    1.请求消息(请求起始行，请求头，请求主体)  浏览器要发送给服务器的数据  2.响应消息(响应起始行，响应头，响应主体)  服务器发送给浏览器的数据 |

**Request请求消息**

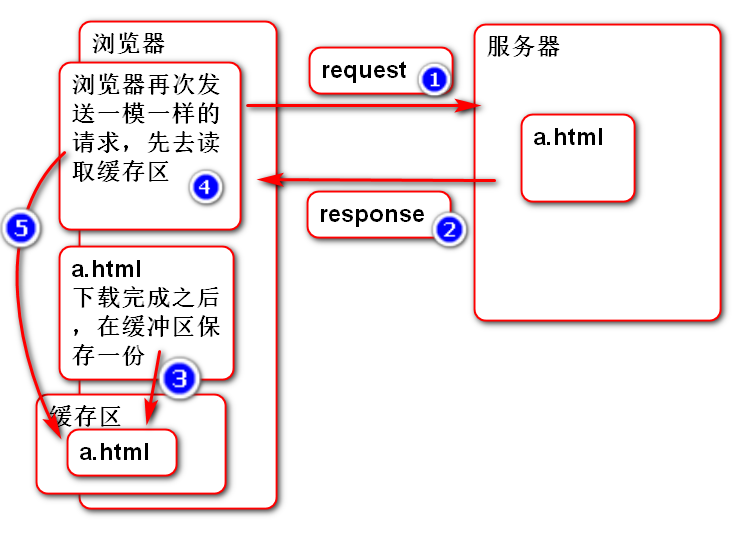
|  |
| --- |
| 客户端要发送给服务器的数据  由三部分组成 请求起始行，请求头，请求主体 |
| 1.请求起始行  -请求方法  -请求url  -协议版本 |
| 请求方法  get 客户端向服务器要数据的时候使用  靠地址栏传递查询字符串，无请求主体  post 客户端向服务器提交数据的时候使用  隐式发送。有请求主体(form data)  delete 表示客户端可以删除服务器上的内容(一般禁用)  put 表示客户端可以往服务器上放数据(一般禁用)  connect 测试连接  trace 追踪请求路径  option 选项  head 表示客户端只回去响应的消息头 |
| url 访问的地址链接  协议版本 http/1.1 |
| 2.请求头 Request Headers  host:告诉服务器，请求的是哪一个主机  Connection: keep-alive 告诉服务器进行持久连接  User-Agent:告诉服务器自己(浏览器)的类型  Accept-Encoding:gzip,deflate,告诉服务器，自己能接收的压缩文件类型  Accept-Language:告诉服务器自己能接收的自然语言的类型  Referer:引用/推荐人，告诉服务器请求来自哪个网页 |
| 3.请求主体 Form data |

**Response响应消息**

|  |
| --- |
| 服务器端发送给客户端的数据，  由三步组成，响应起始行，响应头，响应主体 |
| 1. 响应起始行  协议版本 http/1.1  响应状态码  原因短句，对状态码的解释说明 |
| 响应状态码  告诉客户端浏览器，服务器的响应状态是什么  取值：  1XX:100-199 提示信息  2XX:成功响应 200 ok  3xx：301 永久重定向 www.sun.com  302 临时重定向  304 请求未被修改，还是原来的请求，命中缓存  4XX：客户端请求的错误  404 NOT Found请求资源不存在  403 Forbidden 权限不够  405 Method not Allowed 请求方法不被允许  5XX：服务器运行错误 |
| 原因短句，就是对状态码的解释 |
| 2.响应头  Date 告诉浏览器，服务器的响应时间，格林威治时间 +8h是北京时间  Connection:keep-alive 告诉浏览器已经启动了持久连接  Content-Type:响应主体的类型  text/html 响应回来的数据是html文本  text/plain 普通文本  text/css样式文件  application/javascript js文件  image/png(jpeg.gif) 图片  application/json 响应回来的是json格式的字符串  application/xml 响应回来的是xml格式的字符串 |
| 3.响应主体 响应回来的数据  network中，在header的右边第二个 |

**缓存**

1.什么是缓存



2.缓存的优点

|  |
| --- |
| 1.减少冗余的数据传输，节省客户端的流量  2.节省服务器带宽  3.降低了对服务资源的消耗和运行需求  4.降低了远距离传输而造成的延迟加载 |
| 缓存的原理    1.请求--无缓存--访问服务器--存缓存--客户端得到数据  2.请求--有缓存--够新鲜--命中缓存--客户端得到数据  3.请求--有缓存--不新鲜--访问服务器看是否过期--没过期--更新缓存的新鲜度--客户端得到数据  4.请求--有缓存--不新鲜--访问服务器看是否过期--已过期--访问服务器--存缓存--客户端得到数据 |

3.与缓存相关的消息头

|  |
| --- |
| Cache-Control:http1.1的用法  从服务器将文档传到客户端之时起，  此文档处于新鲜的秒数，这个秒数是一个相对时间  语法：0 不缓存  max-age= 新鲜的秒数  Expires http1.0用法  指定过期的确切时间，是一个具体的时间点 |

4.网页如何添加缓存

|  |
| --- |
| <meta http-equiv="消息头" content="消息头对应的值">  ex  <meta http-equiv=" Cache-Control " content="max-age=3600"> |

5.Http性能优化

|  |
| --- |
| 1.http连接的过程  发起请求-->建立连接-->处理请求-->访问资源-->构建响应-->发送响应-->记录日志 |
| 2.http连接性能优化的思路  2.1减少请求次数  2.2减少连接创建次数(开启持久连接)  2.3提高服务器端的运行速度  2.4尽可能减少响应数据的长度 |
| 3.安全的HTTP协议  HTTPS，安全版本的http  SSL：为数据通信提供了安全支持  1.客户端发送请求消息时，在SSL层加密  服务器接收到加密文件，在SSL层解密  得到请求明文，对请求进行处理  2.服务器发送响应消息时，在SSL层加密  客户端接收到加密文件，在SSL层解密  得到响应明文，解析相应内容 |

二.DOM操作

1.完整的javaScript操作的组成

|  |
| --- |
| 1.js核心：ECMA Script ECMA--欧洲计算机制造联合会  2.DOM：Document Object Model 文档对象模型  让js动态的操作页面上的元素  3.BOM：Browser Object model 浏览器对象模型  让js动态的操作浏览器 |
| .DOM，这个阶段，获取页面上的元素，改变这个元素的内容 |

2.简单DOM操作

|  |
| --- |
| 1.要获取的元素一定要有id  <div id="d1">这是测试的文本！！</div> |
| 2.在js中，用id获取对象  document.getElementById("d1");  H5支持简写方式  d1就是div对象 |
| 3.获取/设置元素的内容 innerHTML  d1.innerHTML=...  var str=d1.innerHTML;  innerHTML指的是，双标签，开关标签之间的所有内容。  如果开关标签之间有其它元素,会得到该元素的文本字符串  如果设置innerHTML时，添加了新标签，在语法正确的情况下，新标签也会在页面上渲染 |
| 4.事件 用户与页面交互行为，调用什么js代码 |

作业：

复习nodejs接口写法

|  |
| --- |
| 在网页中创建一个div，内容随意  添加一个按钮，点击是获取div中文本并打印在控制台  再添加一个按钮，点击是将div中文本内容修改 |
| 在网页中创建一个div。内容随意  添加一个按钮，在div中动态生成一个3\*3的，内容随意的表格 |

ajax错误总结

|  |
| --- |
| 404：检查url是否正确，要访问的资源是不是在url指向的位置(html位置，和路由)  5XX: 检查接口中的代码是否正确 |

#### DAY03 ajax

一.DOM操作

1.获取元素的对象

|  |
| --- |
| 直接使用这个元素的id值 |

2.获取/修改元素的内容(只有双标签叫内容)

|  |
| --- |
| d1.innerHTML  可以获取/修改内容文本  还可以动态的添加元素 |

3.获取/设置表单控件的数据

|  |
| --- |
| 1.使用input控件的id代表控件对象  2.通过对象的value属性，获取/设置控件的值  <input id="uname" type="text">  uname.value="1000"; |

4.HTML元素的事件

|  |
| --- |
| 允许通过用户的行为来激发的操作就是事件  常用的事件 onclick 用户点击元素，激发某项操作 |
| 文本框/密码框，失去焦点事件 onblur  文本框/密码框,获得焦点事件 onfocus |
| body的事件 onload  当页面加载完成，马上要激发的操作 |

二.AJAX

|  |
| --- |
| 同步 Synchronous  在一个任务进行过程中，不能开启其它的任务。  同步访问：浏览器在向服务器发送请求时，浏览器只能等待服务器的响应，不能做其它事情。  出现场合：  1.地址栏输入网址并跳转  2.a标记跳转  3.submit表单的提交 |
| 异步Asynchronous  在一个任务开启的同时，可以开启其它任务。  异步的访问：浏览器在向服务器发送请求时，不耽误用户在网页上的其它操作  出现的场合  1.用户名的重复验证  2.聊天室  3.搜索建议(百度，京东，淘宝)  4.股票走势图 |

1.什么是AJAX

|  |
| --- |
| Asynchronous JavaScript And Xml  异步的 js 和 xml  本质：使用js提供的异步对象(XMLHttpRequest---xhr)  异步的向服务器发送请求，  并接收响应回来的数据  ajax中，可以无刷新效果，去更改页面的布局结构和内容 |

2.使用Ajax步骤

3.创建异步对象

|  |
| --- |
| 标准创建 var xhr=new XMLHttpRequest()  IE8以下的创建  var xhr=new ActiveXObject("Microsoft.XMLHttp"); |
| 如何判断用户的浏览器版本？  使用window.XMLHttpRequest属性  如果是高版本，这个属性有值  如果浏览器不支持标准创建，这个属性为null  if(window.XMLHttpRequest){  var xhr=new XMLHttpRequest();  }else{  var xhr=new ActiveXObject("Microsoft.XMLHttp");  } |

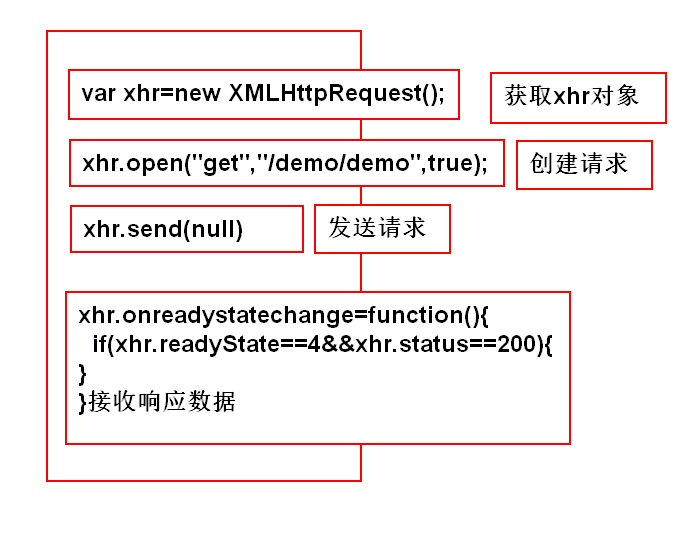
4.异步对象的常用属性和方法

|  |
| --- |
| readystate 表示xhr对象的请求状态  取值 0-4 表示着5个状态  0：请求尚未初始化  1.打开服务器连接，正在发送请求  2.接收响应头  3.接收响应主体  4.接收响应数据成功  当readystate的值为4的时候，才表示所有的数据接收完毕 |
| onreadystatechange 当xhr的readystate属性发生改变时触发 |
| status 表示服务器的响应状态码  只有在status=200，表示服务器已经正确处理了请求 |

使用ajax的步骤

|  |
| --- |
| 1.创建xhr对象  var xhr=new XMLHttpRequest(); |
| 2.绑定监听  xhr.onreadystatechange=function(){  if(xhr.readyState==4 && xhr.status==200){  var result=xhr.responseText;  }  } |
| 3.打开链接，创建请求  xhr.open(method,url,true);  method必须是字符串 "get" "post"  url="127.0.0.1:3000/list?uname="+name+"&upwd="+pwd; |
| 4.xhr.send(); |

01\_ajax.html练习、



|  |
| --- |
| 服务器部分步骤  1.在xz文件夹中，把routes文件夹中的user.js复制。改名为demo.js.  2.创建接口ajaxDemo  //1.ajaxDemo  router.get("/ajaxDemo",(req,res)=>{  res.send("这是我的第一个ajax");  });  3.在app.js中把demo.js导入  const demo=require("./routes/demo.js");  server.use("/demo",demo);  4.启动服务器，在地址栏输入网址，验证  http://127.0.0.1:3000/demo/ajaxdemo  5.在xz中创建文件夹myex.并在app.js中托管静态资源  server.use(express.static('myex'));  6.在myex中创建文件01\_ajax.html中编写ajax代码。 |
| 前端代码  <input type="button" value="ajax" onclick="getMsg()">  <script>  function getMsg(){//尝试传递参数  //1.创建xhr对象  var xhr=new XMLHttpRequest();  //2.绑定监听器  xhr.onreadystatechange=function(){  if(xhr.readyState==4&&xhr.status==200){  var result=xhr.responseText;  console.log(result);  }  }  //3.打开链接，创建请求  xhr.open("get","/demo/ajaxdemo",true);  //4.发送请求  xhr.send();  }  </script> |

5.使用get发送带参数的请求

|  |
| --- |
| url="demo/login?uname=value1&upwd=value2" |

练习：

|  |
| --- |
| 把01\_ajax.html以及对应接口/ajaxDemo，删除，整体重新敲一遍  把02\_login.html以及对应接口/login，删除，整体重新敲一遍 |

作业

|  |
| --- |
| 1.在demo.js中，创建一个接口userlist  在接口中查询出用户表所有的数据，并响应给前端页面  select \* from xz\_user  创建一个网页03\_getList.html.  网页中有一个按钮，点击后，获取userlist接口的数据 |
| 2.在demo.js中，创建接口checkUname.模拟学子商城的注册需求。  文本框失去焦点时，使用ajax连接checkUname.查询数据库是否存在该用户名  如果存在，响应给页面"用户名被占用"  如果不存在，响应给页面"用户名可用"  创建网页04\_register.html。一个文本框，一个密码框，文本框失去焦点的时候调用连接服务器的ajax方法 |

ajax错误总结

|  |
| --- |
| 404：检查url是否正确，要访问的资源是不是在url指向的位置(html位置，和路由)  5XX: 检查接口中的代码是否正确 |
| show is not defined show()方法未定义 |
| Uncaught SyntaxError: Unexpected token { 标点符号，或者括号错误 |
| Cross origin requests are only supported for protocol schemes: http, data, chrome, chrome-extension, https.跨域错误，说明你的html是本地打开 |

#### DAY04 json&xml

一.ajax

1.post请求

|  |
| --- |
| 1.post请求的数据放在请求主体中  var formdata="uname="+uname.value+"&upwd="+upwd.value;  xhr.send(formdata);  2.发送请求之前，要手动修改请求消息头  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  默认的消息头 Content-Type:text/plain,传递的数据是普通的，无特殊符号的字符，主体中有特殊符号"&".必须要把值设置成可以有特殊符号的字符类型application/x-www-form-urlencoded  注意：设置请求消息头，放在xhr.open和xhr.send之间 |

二.JSON数据格式

1.js对象的数据格式

|  |
| --- |
| var stu={name:'lilei',age:20,gender:'男'} |

练习

|  |
| --- |
| 创建一个笔记本对象computer，包含以下三个属性  1.name 的值为MacBook Air  2.price 7800  3.left 200  在控制台按照以下格式打印出结果  笔记本名称 MacBook Air  笔记本的价格 7800元  var computer={  name:'MacBook Air',  price:7800,  left:200  } |

2.JSON的数据格式

|  |  |
| --- | --- |
| 1.什么是Json  JavaScript Object Notation  js 对象 表现方式  js对象表示法，以js对象的数据格式表现出来的字符串 | |
| 2.Json的语法  1.json中用一对{}来表示一个对象  2.对象中的属性名称必须用""引起来（不要用单引号）  如果值是字符串，也必须用""引起来  3.json表现出来是一个字符串，所以最外层添加单引号' ' | |
| js | json |
| var computer={  name:'MacBook Air',  price:7800,  left:200  } | var computer='{  "name":"MacBook Air",  "price":7800,  "left":200  }' |
| ["tom","lilei","hanmeimei"] | '[  {"name":"tom","age":18},  {"name":"lilei","age":20},  {"name":"hanmeimei","age":19}  ]' |

Json是一个以js对象为格式字符串，不能以 对象.属性 的方式获取值。需要先把json字符串转化成js对象，才能使用

|  |
| --- |
| 使用JSON.parse(json串)将json字符串解析为js对象/js数组。  可以在前端页面中，使用js语法操作对象/数组，再使用dom把内容动态的加载到html中 |

提高练习

|  |
| --- |
| 把数据库中所有的xz\_user中所有的数据，都动态放到html的table中 |

三.xml数据格式

|  |
| --- |
| 什么是XML  eXtensible Markup Language  可扩展 标记 语言  xml的标记是没有被预定义过的，需要自行定义  xml的宗旨，是做数据的传递，而不是数据展示 |

语法规范

|  |
| --- |
| 1.xml的最顶端做声明  <?xml version="1.0" encoding="utf-8"?>  2.XML标记的语法  必须成对出现  xml中，标签严格区分大小写  xml允许嵌套，但注意嵌套顺序  每个标记允许自定义属性，格式与html一致，但是值必须带""  每个xml文档，有且只有一个跟标签 |
| 练习：  创建一个student.xml的文件  在文件中声明xml的类型，然后 添加一对跟标记<studentlist>  在根标记中添加3个学生信息(student)  每个学生有name age gender 内容自拟 |
| ajax获取xml数据  使用 var result=xhr.responseXML;获取xml的对象  var students=result.getElementsByTagName("student");  var names=students[0].getElemetnsByTagName("name");  names[0].innerHTML  这种dom操作获取的数组，叫类数组  类数组可以使用下标查找元素，但是不能使用数组的API |

四.项目阶段

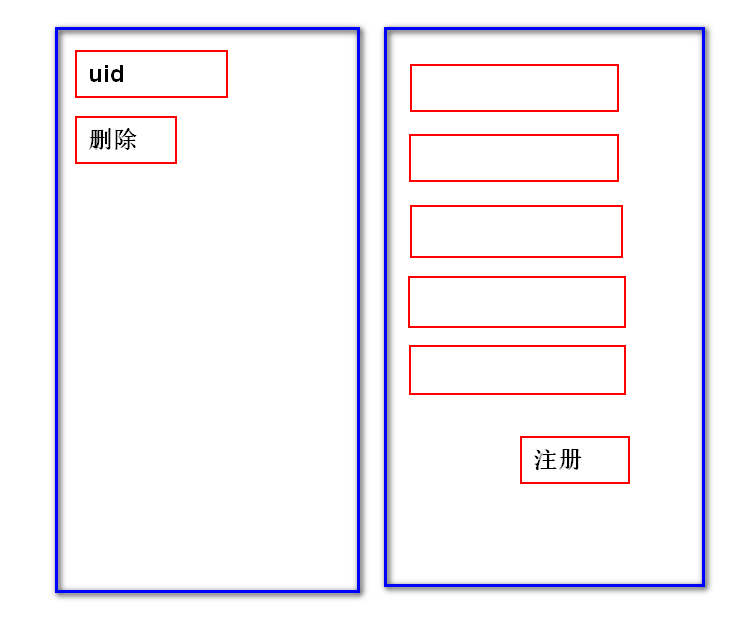
1.登录

|  |
| --- |
| 接收前端传递过来的用户名和密码  将用户名和密码取数据库查询  并给出提示 登录成功/用户名密码错误  1.myPro.js中写login接口  2.myPro文件夹中创建文件01\_login.html  在网页中添加用户名和密码输入框以及按钮。  点击按钮是异步的向后端发送请求，完成登录(post方法) |

2.用户列表

|  |
| --- |
| 在myPro.js定义一个接口userlist  查询所有的用户数据，并响应给前端  在myPro文件夹中创建网页02\_list.html  在body的onload事件中绑定方法get\_list() 16:50 <tbody id="content"> |

作业



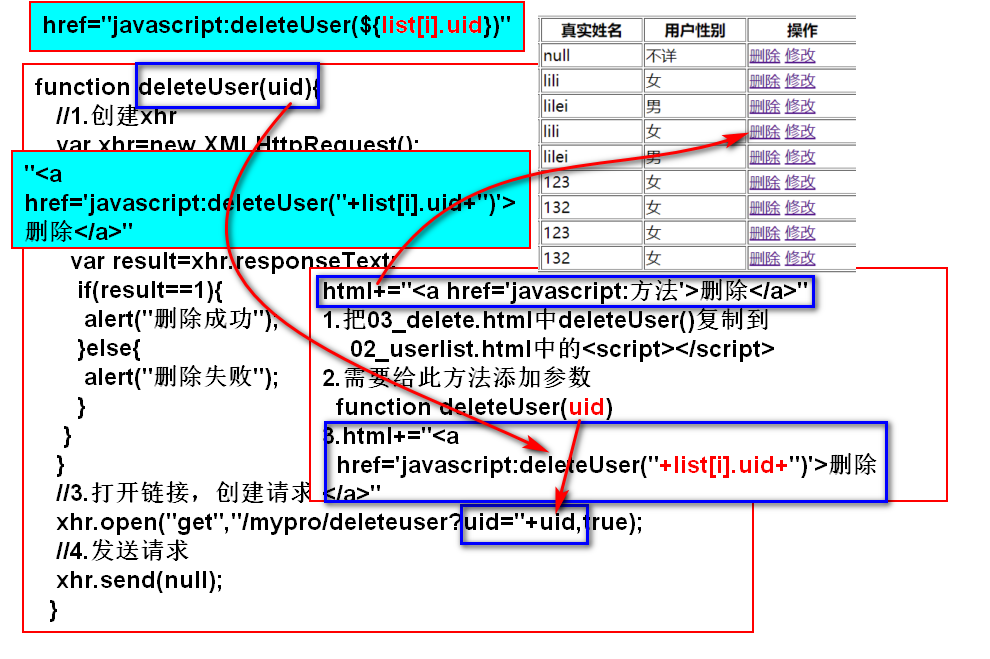
ajax错误总结

|  |
| --- |
| 404：检查url是否正确，要访问的资源是不是在url指向的位置(html位置，和路由)  5XX: 检查接口中的代码是否正确 |
| show is not defined show()方法未定义 |
| Uncaught SyntaxError: Unexpected token { 标点符号，或者括号错误 |
| Cross origin requests are only supported for protocol schemes: http, data, chrome, chrome-extension, https.跨域错误，说明你的html是本地打开 |
| Failed to execute 'setRequestHeader' on 'XMLHttpRequest': The object's state must be OPENED 修改请求头放在xhr.open之前或者xhr.send之后 |

#### DAY05 项目

ajax项目

1.删除用户的逻辑示意图



2.修改

3.注册

1.用户名框，失去焦点，验证数据库中名称是否存在

/checkUname

2.点击注册，把所有的信息都发送给服务器，插入数据库

/register

06\_register.html

作业：

使用ajax完成学子商城注册功能

周末作业：

注册

修改

登录

删除

列表

### 3、CSS

#### DAY01 样式&边框&尺寸

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.作用

|  |
| --- |
| 设置HTML网页中的元素样式 |

3.CSS和HTML的关系

|  |
| --- |
| HTML：负责网页结构的搭建，内容的展示  CSS：负责网页的修饰 |

4.CSS与HTML的属性使用原则

|  |
| --- |
| W3C建议我们尽量使用css的方式来取代html的属性  css样式：  1.样式代码高度的重用  2.提高可维护性 |

二.CSS的语法规范

1.使用CSS的方法

|  |
| --- |
| 1.内联样式  将css的样式定义在HTML标签的style属性中  <any style="样式声明; 样式声明"></any>  样式声明 样式属性:值 组成  多个样式声明之间用分号分隔 |
| 内联样式使用的非常少(禁止使用)  内联样式不能重用  内联样式优先级最高  项目中一般很少使用内联样式，只在学习和测试的时候使用 |

常用的css属性

|  |
| --- |
| color：设置文本的颜色  取值：合法颜色值(颜色的英文，#a~f\*3 #abc #aaf #afa # faa #aff #faf)  font-size：设置字体的大小  background:设置背景颜色 |
| 2.内部样式  在<head></head>标签中，添加<style><style>标签，在其内部定义样式规则  <style>  样式规则  </style>  样式规则：由选择器和样式声明组成  选择器{  样式声明；  样式声明；  }  span{  color:#afa;  background:#faf;  } |
| 内部样式在本页面可以 重用，但是其他页面不能重用  项目中使用不多，但是学习和测试中大量使用 |
| 3.外部样式  独立创建一个新文件\*.css。  可以在任何网页中引用  <head>  <link rel="stylesheet" href="css文件的url">  </head>  外部样式是项目中必须使用的样式写法。 |

练习

|  |
| --- |
| 01\_ex.html中，写3个标签  h1--->内联样式  h2--->内部样式  h3--->外部样式  字体颜色，字号大小，背景颜色 |

2.CSS样式的特性

|  |
| --- |
| 1.继承性  大部分的css效果是可以直接被子元素继承的  必须是层级结构 |
| 2.层叠性  可以为一个元素定义多个样式规则  规则中属性不冲突(不重复)，可以同时作用到当前元素上 |
| 3.优先级  如果样式声明冲突时，按照样式规则优先级去应用  默认的优先级，从高到低：  1.内联样式(行内样式)优先级最高  2.内部样式、外部样式，遵循就近原则  3.浏览器默认样式最低 |
| 4.调整优先级  ！important规则  放在属性值之后，与值之间用空格隔开  作用是调整优先级，让没有important属性不能覆盖他 |

练习

|  |
| --- |
| 02\_ex.html中添加一个p元素lorem，用内部样式设置文字颜色为蓝色  字体大小为24px,用外部样式设置字体大小为40px,字体颜色为红色，将外部样式引入html。  查看页面效果，然后改变内部样与外部样式引入的位置，再次查看页面效果、  使用!important调整样式的优先级 |

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  为了匹配元素 |

2.基础选择器详解

|  |
| --- |
| 1.通用选择器  \*{样式声明}  \*效率极低，尽量少用  \*{margin:0;padding:0}外边距，内边距清0 |
| 2.元素选择器(标签选择器)  设置页面中一类元素的默认样式  标签名称{样式声明}  p{color:red} |
| 3.类选择器  定义页面上某类元素的公共样式，谁想用我，就引用我的class名  元素通过class属性来引用该样式  .类名{样式的声明}  <any class="类名">  注意: "."不能省略  类名不能以数字开头  类名不能包含特殊字符(可以包含 - \_) |
| 类选择器的特殊用法  1.多类选择器  让一个元素引用多个类选择器  <p class="font\_blue font\_size24">烤生蚝</p>  2.分类选择器  将元素选择器和类型选择器联合使用  元素选择器.类选择器{}  可以匹配到，指向了某个类的，具体某个元素  <span class="font\_blue">水煮牛肉</span>  span.font\_blue{background:#faf;}  作用  1.指向更精确  2.优先级变高 |

练习

|  |
| --- |
| 03\_ex.html，添加div和p，内容随意。  用类选择器为所有的元素设置字体颜色为红色  用分类选择器设置p元素的背景为黄色 |
| 4.id选择器  设置指定元素的样式，专属定制  为元素设置id值，使用#id值作为选择器，编写专属样式 |
| 5.群组选择器  将多个选择器放在一起，一块定义某公共样式  选择器1，选择器2，选择器3......{公共样式声明}  #content,p.mycolor,span{color:red} |
| 6.后代选择器  通过元素的后代关系匹配元素  后代：一层或者多层的嵌套关系  语法：选择器1 选择器2 选择器3{样式声明} |
| 7.子代选择器  通过元素的子代关系匹配元素(一层嵌套)  选择器1>选择器2>选择器3{样式声明} |

练习

|  |
| --- |
| 使用子代和后代选择器，把三个a设置成三种不同字体颜色  <div>  <a></a>  <p>11111  <span>  <a></a>  </span>  </p>  <div><a></a></div>  </div> |
| 8.伪类选择器  匹配元素的不同状态下的样式  1.链接伪类  :link 元素尚未访问时的状态样式  :visited 元素访问过后的状态样式  #a1:link{color:red;}  #a1:visited{color:yellow;}  2.动态伪类  :acitve 匹配元素被激活时的状态  #btn:active{color:purple;}  :focus 匹配元素获取焦点时的状态  #txt:focus{background:#aff;}  :hover 鼠标悬停在元素时的状态  .img:hover{width:100px;} |

练习

|  |
| --- |
| 1.04\_ex 添加一个a标记，内容随意，设置以下样式  1.未被访问之前为黑色，没有下划线text-decoration:none;  2.鼠标悬停时，文本颜色为红色  3.被激活时，文本颜色为yellow  4.访问过后，文本颜色orange |
| 2.页面中添加一个input输入框，默认字体为灰色gray  字体斜体 font-style:italic;  1.被激活时，字体喝了脉动，非斜体（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  类和伪类选择器 10  元素选择器 1  \*选择器 0  继承的样式无权值 无  总结：  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的权值进行相加。  然后再进行比较，权值高的优先显示  注意：选择器的权值计算不会超过其最大的数量级  （1W个#id也不会到1000）  2.群组选择器的权值，单独计算，不会相加  3.样式后面加!important 该样式会获取最高优先级  内联样式不能添加!important  4.如果权值相同，使用就近原则 |

四.尺寸和边框

1.尺寸属性

|  |
| --- |
| 改变元素的宽高  width  最大宽度 max-width  最小宽度 min-width  height  最大高度 max-height  最小高度 min-height |

附加知识点：单位

|  |
| --- |
| 1.尺寸单位  px 像素  in 英寸 1in=2.54cm  pt 磅值，多数用它表示字体大小 1pt=1/72in  cm mm 厘米，毫米  em 相对单位，相对父元素的倍数  rem 相对单位，相对根元素的倍数(html)  % 相对单位，相对父元素的百分之多少 |
| 2颜色单位  1.颜色的英文单词 red blue yellow black orange pink purple  2.#rrggbb RGB光学三元色 6个16进制 #006699  3.#aabbcc--->#abc #f00 #0f0 #00f #fff #000 #333  4.rgb(r,g,b); r:0~255 g:0~255 b:0~255  rgb(255,255,255);白色  5.rgb(r%,g%,b%) 此变现形式使用极少  6.rgba(r,g,b,alpha) alpha透明度 |

周末作业：

1.ajax项目

登录

注册

删除

修改

列表显示

2.css的3 种使用方式，理论背

3.css的特点，理论

4.css的基础选择器 8种

5.尺寸 rem em 颜色

#### DAY02 盒子模型&边距

一.尺寸和边框

1.尺寸属性

2.溢出处理

|  |
| --- |
| overflow overflow-x overflow-y  当内容多，元素区域小的时候，就会产生溢出效果  默认是纵向溢出  取值：  1.visible 可见的，默认值，溢出可见  2.hidden 隐藏的，溢出的内容不可见  3.scroll 显示滚动条，不管是否溢出，都添加横向和纵向滚动条，(不溢出不可用)  4.auto 溢出的部分，显示滚动条，并可用 |

3.边框属性

|  |
| --- |
| border:width style color;  width:边框的宽度，以px为单位的数字  style:边框的样式  取值 solid实线  dotted虚线(点点)  dashed虚线(短线)  double 双实线  color:边框的颜色，合法的颜色值 (transparent 透明色)  这种写法，同时设置4个边框，是一种简写方式  取消边框 border:none/0; |

练习

|  |
| --- |
| 创建一个div，id=content 尺寸200\*200  设置4个方向的边框，2px 实线 橙色 |
| 单边设置  border-top/right/bottom/left  border-top:5px solid orange;  border-right:3px dotted #f00;  border-bottom:7px dashed #0ff;  border-left:11px double #f0f; |
| 单属性设置  border-color:#f00;  border-style:dotted;  border-width:10px; |
| 单边单属性  border-方向-属性(共12个)  border-top-style:double;  border-top-color:#44f; |
| 边框的倒角(圆角)  将直角倒成圆角  border-radius：  取值 以px为单位的数字  % 设置圆形(50%)  单角设置  使用两条边去确认一个角，先写上下后写左右  border-top-left-radius:左上  border-top-right-radius:右上  border-bottom-left-radius:左下  border-bottom-right-radius:右下 |
| 边框阴影  box-shadow: h-shadow v-shadow blur spread color  h-shadow 水平方向阴影偏移，必须值  v-shadow 垂直方向偏移量，必须值  以下可选参数  blur：阴影的模糊距离，可选值，越大模糊距离越明显  spread：阴影的尺寸，指定要在基础阴影上扩出的大小  color：阴影颜色  inset：把默认的外部阴影设置为内部阴影 |

练习：

|  |
| --- |
| 设置一个圆形发光的效果(太阳，日食) |

6.轮廓

|  |
| --- |
| 轮廓指的是边框的边框，绘制与边框外边的线条  outline:width style color; |

练习:尝试完成一个乒乓球拍

二.框模型（盒子模型）

|  |
| --- |
| 页面元素皆为框（盒子）  定义了元素框处理元素内容，内边距，外边距以及边框的一种计算模式  定义了一个元素占地大小的计算方式  盒子模型默认的计算方式：    元素实际占地宽度=左外边距+左边框的宽度+左内边距+内容区域的宽度+右内边距+右边框的宽度+右外边距  元素实际占地高度=上外+上边框+上内+内容区域高度+下内+下边框+下外  内边距：元素边框与内容之间的空白间距  外边距：围绕在元素边框外的空白区域(元素与元素之间的距离) |

1.外边距

|  |
| --- |
| margin 定义4个方向外边距  margin-top/right/bottom/left 定义某一个方向的外边距  取值  1.以px为单位的数字  为正数 top，元素往下移动  left, 元素往右移动  为负数 top,元素往上移动  left,元素往左移动  2.取值为%，相对于父元素尺寸的百分比  3.auto，自动计算块级元素的外边距，控制块级元素在水平方向居中对齐  margin的效果，改变元素在页面上的位置，多用于微调 |
| 简写方式  margin:value; 设置4个方向外边距  margin：v1 v2; v1:上下外边距 v2：左右外边距  margin：0 auto; 控制块级元素在水平方向居中对齐  margin:v1 v2 v3; v1:上外边距 v2:左右外边距 v3：下外边距  margin:v1 v2 v3 v4; 上 右 下 左(顺时针) |

练习

|  |
| --- |
| 05\_ex创建两个div 300\*300，随意设置两个不同背景颜色  两个div之间的间距为50px  第二个div水平居中显示 |

3.自带外边距的元素有哪些

|  |
| --- |
| h1~h6,p,body,ol,ul,dl,pre  通常要清除这些元素的自带外边距\*{margin:0;padding:0} |

4.外边距的特殊效果

|  |
| --- |
| 1.外边距的合并  当两个垂直外边距相遇时，他们将合并成一个外边距  最终的取值，取决于两个外边距中距离大的值 |
| 2.行内元素以及行内块元素的外边距  行内元素垂直外边距无效(img除外)，水平外边距相遇两个值会相加  行内块，设置垂直外边距，整行元素跟着发生变化 |

练习

|  |
| --- |
| 06\_ex创建两个div , #d1 #d2 200\*200 不同的背景颜色  在#d2中添加一个div#d3 100\*100 添加不同背景颜色  最后设置d3的上外边距为50px.查看页面效果 |

3.外边距溢出

|  |
| --- |
| 在特殊条件下，给子元素设置的上外边距，会作用到父元素上  特殊条件：1父元素没有上边框，2只有第一个子元素设置上边距时  解决方案：  1.为父元素增加上边框；弊端：元素实际占地高度变大  2.为父元素添加上内边距；弊端：元素实际占地高度变大  3.在父元素的第一个子元素位置添加一个空的<table></table> |

内边距

|  |
| --- |
| 不会影响其他元素，但是会改变元素自己的占地尺寸，会变大 |
| padding:value;设置4个方向的内边距  padding-top/right/bottom/left 定义单方向的内边距  取值  以px为单位的数字  %  简写方式  padding:value; 设置4个方向的内边距  padding:v1 v2; v1:上下 v2:左右 padding没有auto  padding:v1 v2 v3: v1:上 v2:左右 v3:下  padding:v1 v2 v3 v4;上 右 下 左 |

box-sizing属性

|  |
| --- |
| 设置盒子模型的计算方式  默认计算方式：  元素实际占地宽度=左外边距+左边框+左内边距+内容区域宽度+右内边距+右边框+右外边距  元素实际占地高度=上外边距+上边框+上内边距+内容区域高度+下内边距+下边框+下外边距 |
| box-sizing取值  content-box 默认值，会采用默认计算方式  border-box 元素的尺寸(定义宽高)，包含border,padding和内容区域宽高  元素实际占地宽度=左右外边距+设置的宽度width  元素实际占地高度=上下外边距+设置的高度height |

块级元素，行内元素，行内块

|  |
| --- |
| 块级元素：独立成行，可以设置宽高，margin上下有效 |
| 行内元素：与其他行内元素和行内块，共用一行  宽高无效，margin上下无效 |
| 行内块：与其他行内元素和行内块，共用一行  可以设置宽高，margin上下有效  一行行内元素，其中一个设置了上下margin,整行都跟着移动 |

作业：

1.实现外边距的合并效果

2.实现外边距溢出下过，及其解决方案

3.观察行内元素以及行内块元素的外边距效果

4.使用倒角，边框阴影，完成宇智波族徽

#### DAY03 文本&渐变&背景

一.背景

1.背景颜色

|  |
| --- |
| background-color:  取值  合法的颜色值  transparent;透明  注意：背景色的填充，默认从边框位置处开始 |

2.背景图片

|  |
| --- |
| background-image:url(资源路径) |

3.背景图片的平铺

|  |
| --- |
| background-repeat:  取值  1.repeat 默认值，平铺  2.repeat-x 水平方向平铺  3.repeat-y垂直方向平铺  4.no-repeat 不平铺，只显示一张 |

4.背景图的定位

|  |
| --- |
| background-position：x y  取值  1.以px为单位的数字  指的是，图片在水平和垂直方向的偏移量  2.x% y%  50% 50%中间  0% 0% 左上  100% 100% 右下  3.关键字  x：left/center/right  y: top/center/bottom |

5.背景图的尺寸

|  |
| --- |
| background-size: width height  取值：  1.以px为单位的数字  2.%  3.cover，填满容器，图片的宽高比不变，铺满整个容器的宽高，图片溢出的部分会被裁减掉  将背景图同比缩放，直到背景图完全覆盖容器，不留一丝空白，图片溢出的部分会被裁减掉  图片可能会显示不完整，但是容器完全被背景图充满  4.contain 图片自身宽高比不变，但是，要缩放到图片能够完全展示出来，所以会有空白区域 |

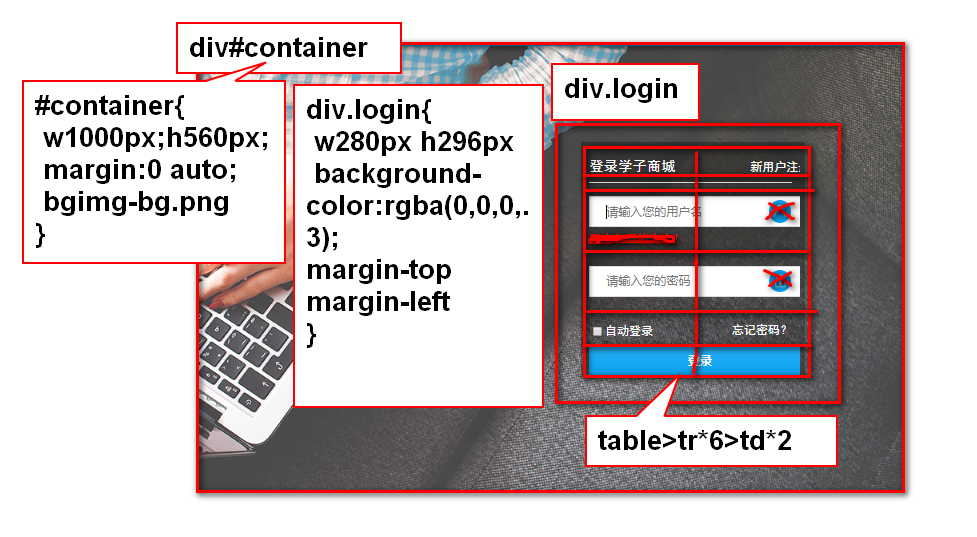
6.背景图的固定

|  |
| --- |
| background-attachment:  将背景图固定在网页某个位置，一直在可视的区域中显示，不会随着网页滚动条改变位置  取值：1.scroll 默认值，会随着网页滚动条改变位置  2.fixed 固定，不会随着网页滚动条改变位置 |

7.背景的简写方式

|  |
| --- |
| 在一个属性中，指定背景多个值  属性：background  取值：color url repeat attachment position  background:#faa url('08.png') no-repeat center center; |

练习



|  |
| --- |
| 1.先写结构布局  2.定义宽高，大体的位置  3.背景颜色图，边框所有  4.文字的所有  5.微调  从外往里写，从左往右，从上往下写 |

二.渐变gradient

1.什么是渐变

|  |
| --- |
| 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

|  |
| --- |
| 色标，一种颜色以及他出现的位置  一个渐变至少有两个色标 |

3.渐变的分类

|  |
| --- |
| 1.线性渐变，以直线的方向来填充渐变色  2.径向渐变，以圆形的方式实现渐变色 |
| 3.重复渐变，将线性渐变，或径向渐变 重复几次 |

4.线性渐变

|  |
| --- |
| background-image  取值：  新版本：linear-gradient(方向，色标1，色标2........)  色标的写法：颜色的合法值 位置的百分比  方向取值：to top/right/bottom/left  角度 按顺时针计算  0deg to top  90deg to right  180deg to bottom  270deg to left  旧版本写法：-浏览器内核-linear-gradient(方向,色标1,色标2....)  方向 top/right/bottom/left |

5.径向渐变

|  |
| --- |
| background-image:radial-gradient(半径 at 圆心,色标1,色标2......);  半径：以px为单位的数字  圆心：x y，以px为单位的具体数值  x% y% 元素的宽高占比  关键字 x:left/center/right  y:top/center/bottom |

6.重复线性渐变

|  |
| --- |
| repeating-linear-gradient(to right,#faa 0px,#ffa 50px);  注意在重复渐变中，色标的位置，要给绝对值数字，不要用% |

7.重复的径向渐变

|  |
| --- |
| repeating-radial-gradient(50px at center center,#060606 0px,#af8bd8 20px,#292727 35px); |

8.浏览器兼容性

|  |
| --- |
| 各个浏览器新版本都渐变  对于不支持的旧版本，可以添加浏览器内核前缀的方式，做兼容，让浏览器支持  chrome&safari: -webkit-  firefox:-moz-  ie:-ms-  opera: -o-  background: -webkit-linear-gradient(left, #006699 0%,#f00 25%, #990066 50%,#ff0 75%,#996600 100%); |

三.文本格式化（重点\*\*\*\*\*\*）

1.字体属性

|  |
| --- |
| 1.指定字体的类型  font-family:"类型1","类型2".... |
| 2.字体大小  font-size  以px、pt为单位的数字  em rem |
| 3.字体加粗  font-weight:  bold/normal/lighter/bolder  无单位的数字 100-1000整百数 常用400-900 |
| 4.字体样式  font-style  normal  italic |
| 5.小型大写字母  font-variant:  normal  small-caps; |
| 6.字体属性的简写方式  font:style variant weight size family  font:italic small-caps bold 30px "黑体";  简写方式最少要有size和family. |

练习：

|  |
| --- |
| 创建04\_ex，一个div 内容随意，最好中英结合，设置以下样式  1.字体为黑体  2.大小为32px  3.加粗并斜体显示  4.所有小写字母变成小型大写字母 |

2.文本格式化

|  |
| --- |
| 1.文本颜色  color:合法的颜色值 |
| 2.文本的水平对齐方式  text-align  left/center/right/justify(两端对齐) |
| 3.行高  定义一行数据的高度  如果行高大于字体本身的大小，该行文本在行高内成垂直居中的显示效果  line-height:  以px为单位的数字，一般与容器的高相同  无单位数字，line-height:2; 是字体大小的倍数 |
| 4.线条的修饰  text-decoration  1.none 没有线条，去掉a标签的下划线  2.overline 上划线  3.underline 下划线  4.line-through 删除线 |

5.首行缩进

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

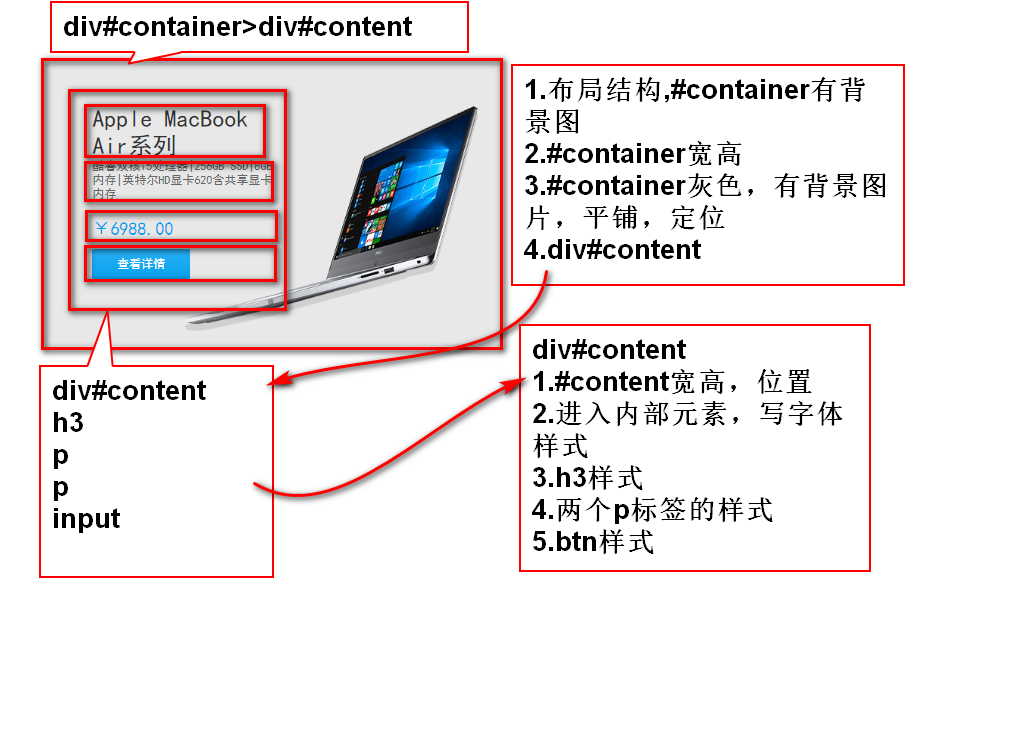
6.文本的阴影

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

练习

|  |
| --- |
| 在04\_ex中，创建div#d2,内容随意，添加以下样式  1.文本水平两端对齐  2.有下划线  3.行高是字体大小的1.5倍  4.首行缩进50px |

作业： transition: 1s;



#### DAY04 显示&浮动&表格

一.表格的样式

1.表格的常用属性

|  |
| --- |
| 之前学习的样式，在表格中，基本都可以使用  1.边框属性 table和td  2.边距属性 table都能使用，td没有margin  3.尺寸属性  4.文本格式属性 font-\* text-\* line-height  5.背景属性 颜色 图片 渐变  6.vertical-align:top/middle/bottom 要写在td才生效 |

练习

|  |
| --- |
| 01\_ex 创建4\*4的表格。400\*400 内容随意  1.设置每个单元格尺寸100\*100  2.设置表格和单元格的边框  3.尝试给每个td设置上外边距 |

2.表格特有的属性

|  |  |
| --- | --- |
| 1.边框合并  border-collapse  1.separate 默认，边框分离的模式  2.collapse 边框合并 | |
| 2.边框的边距  设置单元格与单元格之间的距离,必须保证border-collapse:separate;  border-spacing  1.取值一个value,垂直和水平距离相同  2.取值两个value1 value2. value1:水平距离，value2:垂直距离 | |
| 3.标题的位置  caption-side  top/bottom | |
| 4.表格的显示规则,告诉浏览器，我这张表格如果渲染，td尺寸的计算方式  table-layout:  1.auto 默认值，自动表格布局，列的尺寸实际是由内容决定的  2.fixed 固定表格的布局，列的尺寸由设置的值决定  需要table有尺寸，td有尺寸 | |
| 自动布局auto | 固定布局fixed |
| 单元格的大小会适应内容 | 单元格尺寸取决于设定的值 |
| 表格复杂时，加载的速度慢(缺点) | 任何情况下会加速加载表格(优点) |
| 布局会比较灵活(优点) | 布局会比较死板(缺点) |
| 适用于不确定每列大小，并且表格不太繁琐的时候 | 当确定每列尺寸的时候使用 |

二.定位(重点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

1.什么是定位

|  |
| --- |
| 元素在页面中的位置 |

2.分类

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

3.普通流定位

|  |
| --- |
| 又称文档流定位，页面中所有元素默认显示方式  1.每个元素在页面中都有自己的空间(盒子模型)  2.每个元素都是从父元素的左上角开始显示的  3.块级元素默认按照从上往下的方式逐个排列，每个元素独占一样  4.行内元素是多个元素在一行中显示，从左往右排列 |

4.浮动

|  |
| --- |
| 浮动特点  1.元素一旦浮动起了，脱离文档流  不占页面空间，其它未浮动元素会上前补位  2.浮动元素会停靠在父元素的左边或者右边  或其它已经浮动元素的左/右的边缘  3.浮动解决多个块级元素在同一行内显示的问题 |
| 语法  属性 float  取值  left：左浮动，元素浮动起来后，停靠在父元素的左侧，或者往左挨着已浮动元素  right：右浮动，元素浮动起来后，停靠在父元素的右侧，或者往右挨着已浮动元素 |

浮动元素引发的特殊情况

|  |
| --- |
| 1.当父容器横向显示不下所有浮动元素的时候，最后一个元素将换行显示  1.1默认情况下，最后一个元素会优先显示在最高的位置，再往左排列  1.2会发生浮动元素占位的情况，浮动元素根据浮动方向，占据方向之上的位置，不允许最后一个元素占用 |
| 2.浮动对默认宽度的影响  块级元素不写宽度，宽度默认是父级元素的100%  元素一旦浮动起来，宽度以内容为准  前题，不设置width |
| 3.元素一旦浮动起来，就变为块级元素，允许修改尺寸，设置垂直外边距 |
| 4.文本，行内元素，行内块，是不会被浮动元素压在下方的，二会巧妙避开，环绕着浮动元素显示 |

清除浮动

|  |
| --- |
| 元素一旦浮动之后，会对后续的元素带来一定的影响，后续元素会上前补位，如果不希望后续上前补位，可以给后续元素添加清除浮动的属性  clear:  取值  left:清除左浮动元素对我带来的影响  right:清除右浮动元素对我带来的影响  both: 清除所有浮动元素对我带来的影响 |

浮动元素对父元素高度带来的影响(高度坍塌)

|  |
| --- |
| 块级元素不写高度，他的高度是根据内容判定的  如果内部元素都浮动了，内部元素都脱了默认文档流  父级元素就失去了高度------高度坍塌  解决方案：  1.给父级元素添加高度 弊端：不是每次都知道具体的高度是多少  2.设置父级元素也浮动 弊端：会影响父元素后面的其它元素  3.给父元素设置overflow:hidden/auto 弊端：如果内容想要溢出显示，显示不了。  4.在父元素中追加一个空的块级元素，只设置clear:both |

三.显示

1.显示方式

|  |
| --- |
| 决定了元素在网页中表现形式(块级，行内，行内块，table)  语法 display  取值：  1.none 不显示元素，隐藏  2.block 让元素表现为块级  块级元素特征，独占一行，可以设置尺寸以及上下外边距  3.inline 让元素表现为行内  行内元素特征，多个元素一行显示，不能设置宽高以下上下外边距  4.inline-block 让元素表现为行内块  行内块特征，多个元素一行显示，可以设置尺寸以及上下外边距  5.table 让元素表现为table  table的特征：尺寸以内容为准，独占一行，允许修改尺寸 |

2.显示的效果

|  |
| --- |
| 属性：visibility  取值 visible/hidden |
| visibility:hidden和display:none的区别  visibility:hidden隐藏，不脱离默认文档流，依然占位  display:none 隐藏，脱离了默认文档，不占页面位置 |

3.透明度

|  |
| --- |
| opacity:0~1 1是不透明，0是全透明 |
| opacity和rgba的区别  opacity作用与元素，当一个元素设置了opacity之后，  这个元素本身，以及所有子元素，与颜色相关的属性，都会受到影响  rgba(0~255,0~255,0~255,0.5)只会改变你设定的这个颜色透明度 |

4.垂直对齐

|  |
| --- |
| vertical-align  使用在table中,取值top/middle/bottom  使用在img中，取值 top/middle/bottom/baseline  控制图片与两边文字垂直对齐方式 |

5.光标

|  |
| --- |
| 改变鼠标悬停在元素时的样式  cursor:  取值  1.default 箭头  2.pointer 小手  3.crosshair +  4.text I  5.wait 等待  6.help 帮助 |

四.列表

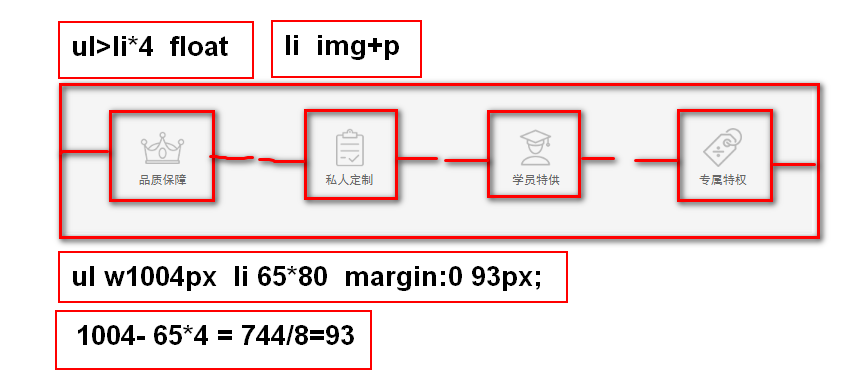
|  |
| --- |
| 列表标识  list-style-type  none  disc  circle  square |
| 列表标识的位置  list-style-position:outside/inside |
| 列表项引用图片  list-style-image:url() |
| 简写方式  list-style:type url() position;  常用的方式 list-style:none; |

作业

1.继续完成学子商城1F



2.完成学子商城首页的底部



#### DAY05 定位&复杂选择器

一.相对/绝对/固定定位 position（重点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*）

|  |
| --- |
| position  取值  static 静态，默认值。文档流定位  relative 相对定位  absolute 绝对定位  fixed 固定定位  注意:将元素的position设置为relative/absolute/fixed中的任意一个  此元素被称为**已定位元素** |

偏移属性

|  |
| --- |
| top/right/bottom/left 距离哪个方向多远  取值是以px为单位的数字 |

1.相对定位

|  |
| --- |
| 元素相对于它原来的位置偏移  在元素位置做微调的时候使用/**配合绝对定位使用**  position:relative;配合偏移属性使用  相对定位不脱离文档流，不会让后面元素补位 |

练习：

|  |
| --- |
| ul中5个li，横向显示，每个li中有图片，鼠标放到img上，图片往左移动8像素，往上移动8个像素 |

2.绝对定位

|  |
| --- |
| position:absolute;配合偏移属性使用  特点  脱离文档流，不占页面空间，后面元素要补位  绝对定位的元素，会相对于"最近的""已定位的""祖先元素"去做偏移量  绝对定位的元素，没有"已定位的""祖先元素"，相对body偏移  绝对定位会让元素变为块级  出现堆叠效果，就使用绝对定位 |

3.固定定位

|  |
| --- |
| 将元素固定在页面的某个位置，位置不会随着滚动条发生变化，一直固定在可视区域  固定定位始终相当于body去实现位置偏移  固定定位脱离文档流，不占页面空间，定位的元素会变成块级  position:fixed;配合偏移属性一起使用 |

4.堆叠顺序

|  |
| --- |
| 默认堆叠顺序，后发生定位的元素，堆叠层级高  默认堆叠顺序，不会大于设置z-index的顺序  z-index:无单位数字，理论最大值 2^31-1  父子元素之间，子元素永远在父元素上面  只有已定位元素，才能使用z-index |

综合练习



二.css3 core

1.复杂选择器

|  |
| --- |
| 1.兄弟选择器  作用于兄弟元素，只能往后找，不能往前找  兄弟元素：具备相同父级元素的平级元素之间，成为兄弟元素  1.1相邻兄弟选择器  选择器1+选择器2{}  在选择器1的后面，紧跟着一个元素，这个元素还得匹配到选择2  1.2通用兄弟选择器  获取某元素后面所有满足条件的兄弟元素  选择器1~选择器2{} |
| 2.属性选择器  id class name type value style titile ....  允许通过元素所带的属性以及值来匹配元素  2.1基本语法  [attr]{} ex:[id]{}匹配有id属性的标签  2.2elem[attr]  div[title]{} 匹配有title属性的div标签  2.3[attr1][attr2]  [class][title]{color:#f00;} 匹配同时具有class属性和title属性  2.4[attr=value]{}  [id=p2]{} 匹配id值为p2的元素  [attr\*=value]属性值中，包含指定值的元素  [attr~=value]属性值中，包含指定单词的元素  [attr^=value] 属性值中,以value开头的元素  [attr$=value] 属性值中,以value结尾的元素 |
| 3.伪类选择器  3.1目标伪类  突出显示获得的HTML锚点元素，匹配被激活的锚点  选择器:target{} div:target{} |
| 3.2结构伪类  1.:first-child{} 匹配的元素是其父元素的第一个子元素  2.:last-child{} 匹配的元素是其父元素的最后一个子元素  2.:nth-child(n){} 匹配的元素是其父元素的第n个子元素，n从1开始 |

练习

|  |
| --- |
| 07\_ex 创建4\*4的表格，400px\*400px 内容随意  第一行背景颜色为#aaf  最后一行背景颜色为#ffa  第三行第二列的背景颜色为#faa |
| 4 :empty  匹配没有子元素的元素  div:empty{} 匹配没有子元素的div元素 |
| 5.:only-child  p:only-child{} 匹配属于其父元素的唯一子元素 |
| 6：否定伪类 :not(selector)  div:not(:empty){} 匹配非空的div |

练习

|  |
| --- |
| 在上面表格基础上，  设置除了第一列以外的其他所有单元格字体都24px，加粗，倾斜 |

3.伪元素选择器

|  |
| --- |
| h2::first-letter{color:#f00;}  匹配某个元素的首字字符  h2:first-line{color:#aaf}  匹配某个元素的首行字符，首行与首字符冲突，优先首字符  h2::selection{color:#ffa}  匹配用户选取的内容，只能修改文本颜色和背景颜色 |

练习

|  |
| --- |
| 08\_ex创建一个div#d1 内容随意  将d1中的首字符设置为加粗红色，大小为20px  右边距15px，首行文本设置为橙色并倾斜。  用户选取内容是将文字颜色设置为黄色文字，红色背景 |

4.内容生成

|  |
| --- |
| 使用css动态的向元素中插入一段内容  1.:before或者::before  匹配元素的内容区域之前，插入的伪元素  2.:after或者::after  匹配元素的内容区域之后，插入的伪元素  <div id="d1">  (::before)  我是要成为海贼王的男人  (::after)  </div> |

练习

|  |
| --- |
| 子曾经曰过：  学而时习之，不亦说乎!  ----《摘自论语》 |

5.内容生成可以解决的问题

|  |
| --- |
| 1.外边距溢出  父元素没有上边框，给第一个子元素设置上外边距，父元素也会发生作用  解决方案，在第一个元素之前添加一个<table></table> |

作业：

1.使用定位完成



2.完成学子底部



3.使用内容生成，解决高度坍塌

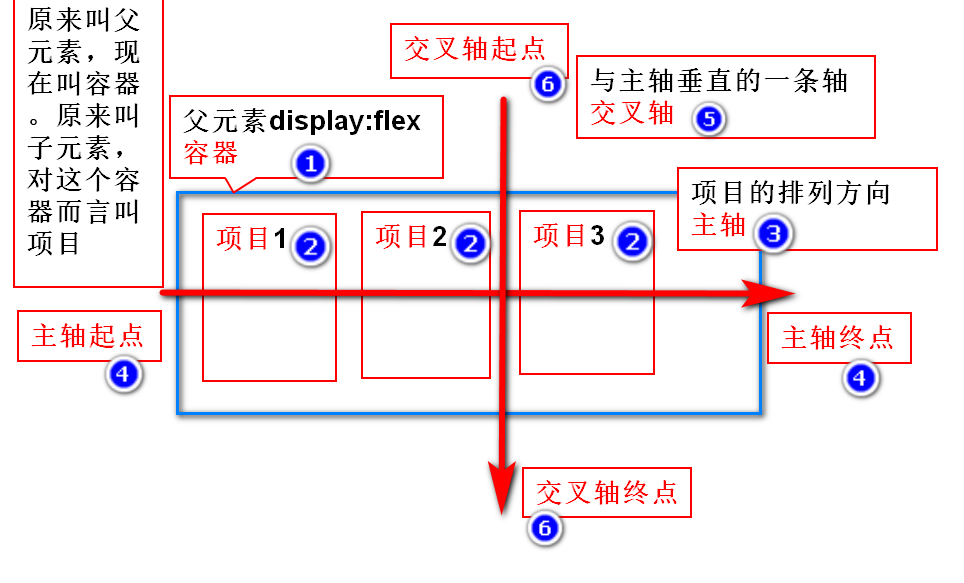
#### DAY06 弹性布局和转换

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

1.什么是弹性布局

|  |
| --- |
| 弹性布局就是一种布局方式  主要解决某元素中的子元素的布局方式，为布局提供最大的灵活性 |

2.弹性布局的相关概念和专业术语



|  |
| --- |
| 1.容器  要布局的子元素的父元素称之为容器，容器中写display:flex  2.项目  要布局的子元素称之为项目  3.主轴  项目们排列的方向，称之为主轴（水平和垂直）  如果项目们是按照横向排列，x轴就是主轴  如果项目们是按照纵向排列，y轴就是主轴  4.交叉轴  与主轴垂直相交的方向轴叫做交叉轴 |

语法

|  |
| --- |
| 将元素变为弹性容器，他所有的子元素将变成弹性项目，按照弹性布局的方式去排列显示  display  取值：flex, 将块级元素变为容器  inline-flex,将行内元素变为容器 |
| 元素设置为flex容器之后，子元素一些样式属性会失效  float/clear/vertical-align失效  子元素允许修改尺寸(项目是行内元素也可以修改尺寸)  容器的对齐方式text-align失效 |

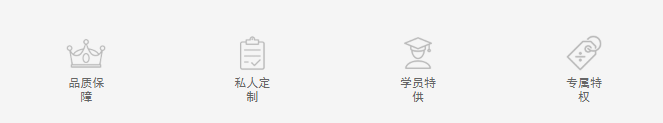
3.弹性容器的样式属性

|  |
| --- |
| 1.flex-direction 主轴方向  取值  1.row 默认值，主轴是x轴，主轴起点在左端  2.row-reverse, 主轴是x轴，主轴起点在右端  3.column 主轴是y轴，主轴起点在顶端  4.column-reverse主轴是y轴，主轴起点在底部 |
| 2.flex-wrap  当一个主轴排列不下所有项目时,项目的显示方式  取值  1.nowrap 默认值，空间不够，不换行，项目会自动压缩  2.wrap 空间不够，就换行，项目不压缩  3.wrap-reverse 换行，反转 |
| 3.flex-flow  是flex-direction和flex-wrap的缩写  取值 direction wrap;  ex:flex-flow:row wrap; |
| 4.justify-content  定义项目在主轴上的对齐方式  取值  1. space-between 两端对齐  2.space-around 每个间距相同  3.flex-start 默认值 在主轴起点对齐  4.flex-end 在主轴的终点对齐  5.center 在主轴上居中对齐 |
| 5.align-items (容器高度大于项目高度)  项目们在交叉轴上的对齐方式  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 如果项目未设置高度，在交叉轴上充满容器 |

4.项目的属性

|  |
| --- |
| 是单独设置给一个项目的，不影响容器和其他项目 |
| 1.order  取值为无单位的整数，定义项目的排列顺序，值越小，离起点越近，默认值为0； |
| 2.flex-grow  定义项目的放大比例  如果容器有足够大的剩余空间，项目将按照比例放大  取值为无单位整数，  默认值为0.不放大。取值越大，占据的剩余空间越多 |
| 3.flex-shrink  定义项目的缩小比例，空间不足时，项目该如何缩小  取值为无单位整数  默认值为1，空间不足，等比缩小  0，不缩小  取值越大，占据的空间越小 |
| 4.align-self  控制当前项目在交叉轴上对齐方式，与其他项目无关  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 如果项目未设置高度，在交叉轴上充满容器  6.auto 继承容器的align-items的效果 |

练习，使用弹性布局，完成下图



CSS hack

|  |
| --- |
| 由于不同的浏览器，比如IE6 IE7 firefox  对css的解析认识不同，会导致生成的页面效果不同，无法得到我们预期的效果  这个时候我们需要针对不同浏览器去写不同的css  让代码同时兼容不同的浏览器。  这个针对不同浏览器写不同css代码的过程，叫做CSS hack  课上不讲解，tmooc有对应的视频 |

二.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 形状等的一种方式  2D转换，在x轴和y轴上，发生转换效果  3D转换，添加了z轴 |

2.转换属性

|  |
| --- |
| transform:  取值 transform-function  none  如果有多个转换函数，用空格隔开 |
| 转换原点  transform-origin  取值：px为单位的数字/%/关键字  2个值，表示原点在x轴和y轴上的位置  3个值，x,y,z轴 |

3.2D转换函数

|  |
| --- |
| 1.位移，改变元素在页面中的位置  transform:translate(x)指定元素在x轴上的位移距离  + 元素往右  - 元素往左  transform:translate(x，y)指定元素在x和y轴上的位移距离  x取值同上  y取值 + 元素往下 - 元素往上  transform:translateX(x)指定元素在x轴上的位移距离  transform:translateY(y)指定元素在y轴上的位移距离 |
| 2.缩放  改变元素在页面中的尺寸  transform:scale(n)  取值，无单位的数字  n>1 根据转换原点放大  0< n <1 根据转换原点缩小  -1< n <0 翻转缩小 （水平和垂直方向都翻转了180度）  n<-1 翻转放大  transform:scale(x,y) x和y缩放尺寸  transform:scaleX(x) 单独设置x的缩放  transform:scaleY(y) 单独设置y的缩放 |
| 3.旋转  改变元素在页面中的角度  transform:rotate(ndeg);  n为正 顺时针  n为负 逆时针  注意：  转换原点会影响最后的旋转效果、  旋转是连同坐标轴一起旋转的，会影响旋转后的位移效果 |

练习

|  |
| --- |
| d1先旋转45度，然后位移200px  d2先位移200px，再旋转45度 |

|  |
| --- |
| 4.倾斜  改变元素在页面中的形状  transform:skew(x),等同于skewX(x)  让元素向着x轴发生倾斜，实际上改变的y轴的角度  + 逆时针  - 顺时针  transform:skewY(y)  让元素向着y轴发生倾斜，实际上改变的时候x轴的角度  + 顺时针  - 逆时针  transform:skew(x,y) |

练习

|  |
| --- |
| 06\_ex创建200\*200的div  设置背景颜色，鼠标悬停时，该元素向右偏移200px;  向下偏移200px；旋转135deg，向着x轴倾斜45度，放大1.5倍 |

4.3D转换

|  |
| --- |
| 浏览器不支持3D的位移，z轴看不到，只能模拟  1.透视距离  模拟人的眼睛到3D转换元素之间的距离  perspective  该属性要加载转换元素的父元素上 |
| 2.3D旋转  transform:rotate3D(0,0,1,20deg);  取值  1.rotateX(xdeg)  以x轴为中心轴，旋转元素的角度(老式爆米花机，烤羊腿)  2.rotateY(ydeg)  以y轴为中心轴，旋转元素的角度(旋转门)  3.rotateZ(zdeg)  以z轴为中心轴，旋转元素的角度(风车，摩天轮)  4. rotate3D(x,y,z,20deg);  x,y,z取值大于0的数字时，表示该轴参与旋转  取值为0，表示不参与旋转 |

三.过渡

|  |
| --- |
| CSS属性值，在一段时间内容平缓的变化 |
| transition-property:background;  border-radius;  all;  能够使用过渡效果的属性  1.颜色  2.取值为数字的属性  3.转换  4.阴影  5.渐变  6.visibility |
| 指定过渡持续时间  transition-duration:1s/1000ms |
| 简写方式  transition:all 1s;  transition:1s;----不要写在hover中，写在元素本身的样式里 |

作业：

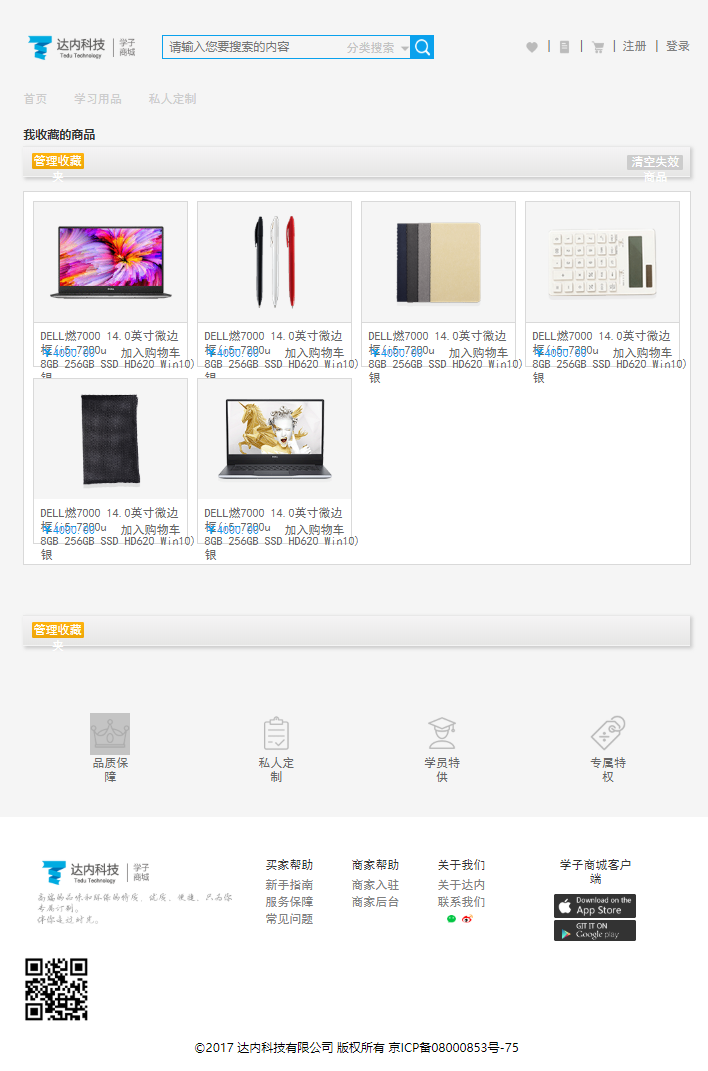
1.把转换的脑图自己总结

2.预习一下boot

3.完成学子首页，（尽量使用弹性布局）



4.



### 4、BOOT

#### DAY01 过度&动画&媒体查询

一.过渡（过度\*\*\*\*\*\*\*\*\*）

|  |
| --- |
| 某元素的css属性值在一段时间内，平滑改变到另外一个值  过渡主要观察的是过程和结果 |

设置能够过渡的属性

|  |
| --- |
| 支持过渡效果的样式属性  颜色的属性  取值为数值  transform  渐变  visibility  阴影 |
| 指定本次过度生效的属性  transition-poperty:上面的css属性/all  指定过度的时长  transition-duration:过度的时长/s/ms  指定过度时间曲线函数  transition-timing-function:  1.ease 默认值，慢-->快-->慢 慢速开始，快速变快，慢速结束  2.linear 匀速  3.ease-in 慢-->快 慢速开始，快速结束4.ease-out 快速开始，慢速结束  5.ease-in-out 慢速开始，先加速再减速，慢速结束 |
| 指定延迟执行过度的时间  transition-delay: s/ms |
| 过度属性的编写位置  1.将过度放在元素声明的样式中(元素自己的样式里)，过度效果有去有回  2.将过度放在元素的触发操作中(hover),过度效果有去无回 |
| 过度的简写  transition:property duration timing-function delay;  最少的方式 transition: duration； |

练习，翻滚吧，牛宝宝

二.动画

|  |
| --- |
| 使元素从一种样式，改变到另外一种，再改变到其他样式.......  相当于将很多个过渡效果放到一起使用 |
| 关键帧  1.动画的执行时间点  2.该时间点上的样式 |

动画的实现步骤

|  |
| --- |
| 1.声明动画及动画关键帧  @keyframes 动画名称{  //定义关键帧  0%{动画开始时的样式}  .........  100%{动画结束时的样式}  }  2.调用动画  animation-name: 动画名称;  animation-duration: 动画播放一个周期的时间  3.动画的其他属性  animation-delay  4.动画的速度时间曲线函数  animation-timing-function  ease/linear/ease-in/ease-out/ease-in-out |
| 5.animation-iteration-count  指定动画的播放次数  取值，具体的数字/infinite无限次 |
| 6.animation-direction:  动画的播放方向  取值 normal 正常 0%--100%  reverse 逆向播放 100%-- 0%  alternate 轮流播放  奇数次正向播放  偶数次逆向播放 |
| 7.简写方式  animation：name duration timing-function delay iteration-count direction; |
| 8. animation-fill-mode  指定动画播放前后的显示状态  1.none 默认值  2.forwards 动画完成后，保持在最后一个关键帧上  3.backwards (需要有delay) 动画开始之前，保持在第一个关键帧上  4.both,同时设置forwards和backwards |

动画的兼容性

|  |
| --- |
| 如果要兼容低版本浏览器，需要在声明动画的时候加前缀  @keyframes 动画名称{}  @-webkit-keyframes  @-moz-keyframes  @-o-keyframes |

三.CSS优化

|  |
| --- |
| 目的：减少服务器压力，提升用户体验 |
| 1.优化原则  尽量减少HTTP请求的个数  页面顶部引入css文件  将css和js放到外部独立的文件夹中 |
| 2.CSS代码优化  缩小样式文件  减少样式的重写  避免出现空的src和href  选择更优的样式属性值（能使用复合，简写的写法，就不要单独定义）  代码压缩 |

二.BootStrap

|  |
| --- |
| www.bootcss.com |

1.响应式布局(重点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

|  |
| --- |
| Responsive web page 响应式/自适应的网页  可以根据浏览器的设备不同(pc,pad,phone)  自动调用对应的布局，图片，文字效果，从而不会降低用户体验 |

2.响应式网页必须做到的前提

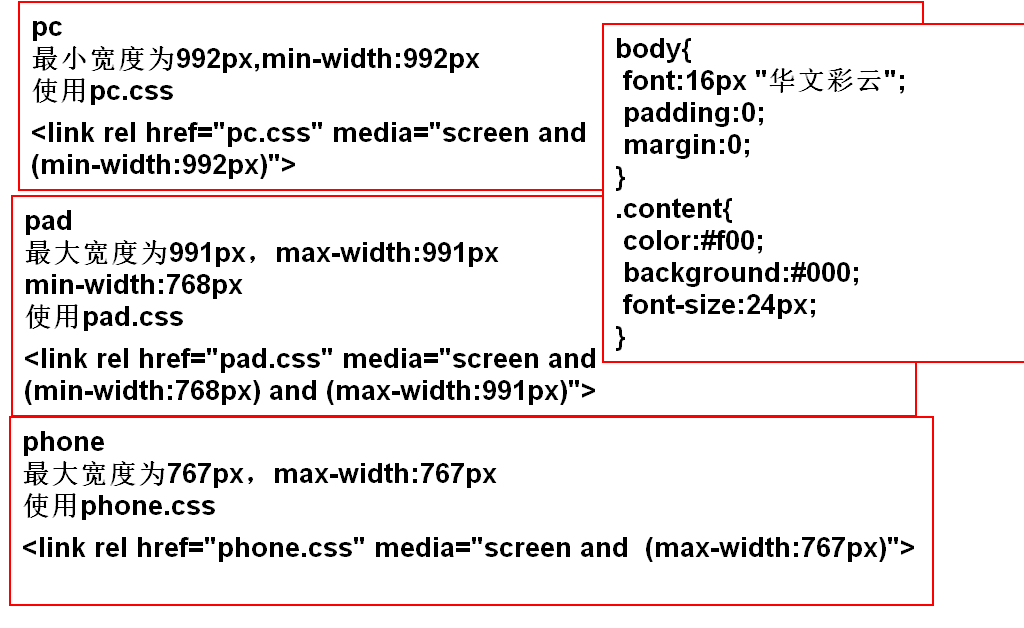
|  |
| --- |
| 1.布局：不能固定宽度，必须是流式布局(尽量少用定位，可以浮动)  2.文字和图片，大小随着容器大小而改变 em rem  **3.媒体查询技术** |
| 响应式网页存在的问题：  页面的复杂度极大的增加  只适用于内容不太多的页面(企业的官网，门户网站)  媒体查询技术属于h5/c3的技术，  boot把媒体查询这件事，封装了，不需要我们自己写了 |

3.如何测试响应式网页

|  |
| --- |
| 1.使用真实设备测试  好处：真实，可靠  缺点：测试任务量巨大 |
| 2.使用的三方的模拟测试软件  好处：无需添置太多真实设备，测试方便  缺点：测试效果有限，有待进一步验证 |
| 3.使用浏览器自带的模拟器测试  好处，简单方便  缺点：测试效果有**限，需要进一步验证** |

4.如何编写响应式布局

|  |
| --- |
| 1.在元数据标签中定义viewport---视口  name="viewport"  content设置能够允许网页进行操作  width=device-width 表示视口宽度就是设备宽度  initial-scale=1.0 表示视口宽度是否可以缩放 1.0不能缩放  maximum-scale=1.0 允许缩放的最大倍率  user-scalable=0 是否允许用户手动缩放 yes/no/1/0  一般的设置  <meta name="viewport" content="width=device-width, initial-scale=1"> |
| 2.所有内容/文字/图片，使用相对尺寸，不能使用 10px这种绝对的值 |
| 3.流式布局+弹性布局，在搭配媒体查询技术来完成响应式布局  float flex |
| 4.使用css3 Media Query技术做响应式网页  Media：媒体，只浏览网页的设备.如：screen(pc/pad/phone) tv print  Media Query:媒体查询，可以自动根据当前浏览器设备的不同（尺寸，解析度，方向不同），有选择执行一部分CSS而忽略其他部分的css |



|  |
| --- |
| 注意，此方案使用较少  所有的css文件都会被加载，每个css文件中，存在很多重复代码 |

根据媒体查询的结果，执行同一个css文件下的不同代码块

|  |
| --- |
| @media screen and (min-width:768px) and (max-width:991px){  选择器{样式}  } |

作业：

1.使用媒体查询完成响应式布局



2.提高题

使用媒体查询完成



#### DAY02 boot样式&栅格

一.Bootstrap

1.调试，使用webstorm

|  |
| --- |
| 1.注册码生成器  2.安装包  3.注册---help-->register |
| 创建项目  在xampp--htdocs--创建文件夹(LR)---创建项目(boot)  使用ws中的open打开文件夹 |
| 在boot项目中。创建文件夹css和js  把boot的4个文件复制进去 |
| 设置字体大小和颜色  在file--->setting-->输入font可以改变字号  color & fonts---->html 的选项，可以更改html文件的字体颜色 |
| 修改tab  general-->editor tab ---->修改成2 |
| 修改快捷键  keymap中选择eclipse |
| 所有使用boot的代码都必须放在container类的包裹中 |

二.全局css样式

1.按钮相关的class

|  |
| --- |
| btn 基本按钮  颜色按钮  btn-danger 危险  btn-success 成功  btn-warning 警告  btn-info 信息  btn-primary 主要  btn-secondary 次要  btn-dark 黑色  btn-light 浅色  btn-link 链接  不同的边框按钮  btn-outline-颜色  按钮大小  btn-lg 大按钮  btn-sm 小按钮  btn-block 块级按钮 |

2.图片相关的class

|  |
| --- |
| .rounded 倒角  .rounded-circle 圆  .img-thumbnail 缩略图  .img-fluid 响应式布局，图片会缩放，但不会超过原始大小 |

3.文字相关

|  |
| --- |
| .h1~.h6 字体大小  字体颜色  text-danger/success/info/primary/warning....  文字对齐  text-\*-left/right/center/justify  \*代表：lg/ sm 在大屏幕下对齐，在小屏幕下对齐  大小写  text-uppercase/lowercase/capitalize 大写，小写，首字母大写  粗体  font-weight-bold  斜体  font-italic |

4.列表相关的class

|  |
| --- |
| ul  list-unstyled 去除标识符  list-group 创建列表组  li  list-group-item 创建列表项  active 激活状态  disabled 禁用状态  li的颜色  list-group-item-danger/success/........ |

5.table相关class

|  |
| --- |
| table的class  table 基本类  table-bordered 带边框的表格  table-striped 隔行变色  table-hover 鼠标悬停效果  table-danger/warning.....  响应式表格  需要在table的父元素上，添加类table-responsive |

6.辅助类

|  |
| --- |
| 边框  border 基础类  border-top/right/bottom/left 设置4个方向的边框  border-0/ border-top-0/border-right-0....... 取消边框  border-danger/warning..... 边框颜色  rounded/rounded-0 设置/取消圆角  rounded-top/right/bottom/left |
| 浮动  float-\*-left/right \*：xl/lg/md/sm  clearfix |
| 显示  visible/invisible 显示/隐藏 |
| 背景  bg-danger/warning..... |
| 居中对齐  m-auto |
| margin  m-\*(0/1/2/3/4/5) (0/0.25/0.5/1/1.5/3rem)  mt-\*/mr-\*/mb-\*/ml-\* |
| padding  p-\*(0/1/2/3/4/5) (0/0.25/0.5/1/1.5/3rem)  pt-\*/pr-\*/pb-\*/pl-\* |
| 宽度/高度  w-\*(25/50/75/100)  h-\*同上 |

三.栅格布局(重要\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

1.web页面布局的方法

|  |  |  |  |
| --- | --- | --- | --- |
|  | table布局 | div+css布局 | BootStrap中的栅格布局 |
| 优点 | 简单，容易控制 | 语义正确，渲染效率高 | 简单，容易控制，语义正确，渲染效率高，支持响应式 |
| 缺点 | 语义错误  渲染效率低(最后一个数据加载完成后，才能统一开始渲染) | 控制比较麻烦  媒体查询写起来更繁琐 | 页面如果太复杂，不适合使用 |

总结：

|  |
| --- |
| 1.最外层需要使用容器  .container 定宽容器 最大宽度是写死的  .container-fluid 变宽容器 宽度是父元素的100%  2.容器中声明.row 弹性布局(一行内等分为12个单元格)  3.行中声明列 col-n(每个列都需要制定宽度占比n/12) |

作业：

使用栅格布局完成下图





#### DAY03 栅格&boot弹性&表单

一.栅格

|  |
| --- |
| col-xl-1/2/3/4..... xl:Extra Large w>=1200px  col-lg-1/2/3/4..... lg:large w>=992px  col-md-1/2/3/4.... md: medium 768<=w<=991  col-sm-1/2/3/4... sm:small 576<=w<=767  xs:Extra small w<=575  col 只写col，每一列平局分配 |
| 不同屏幕下，列的适应性问题  col-xl-\* 只对超大屏生效  col-lg-\* 对xl/lg生效  col-md-\* 对xl/lg/md生效  col-sm-\* 对xl/lg/md/sm生效 |
| 列偏移  可以通过列偏移类实现指定列和后面列的偏移。  offset-\* (\*:0~11)  offset-xl/lg/md/sm-\* |
| 栅格嵌套 |

二.弹性布局

|  |
| --- |
| d-flex或者d-inline-flex 创建一个弹性容器  flex-\*-row 不同屏幕下主轴方向x  flex-\*-column 不同屏幕下主轴方向y  flex-row-reverse  flex-column-reverse  justify-content-end/center/between/around  响应式flex类  d-\*-flex  flex-\*-row  flex-\*-row-reverse  flex-\*-column  flex-\*-column-reverse  justify-content-\*-start/end/center/between/around |

三.表单

|  |  |  |
| --- | --- | --- |
| 在form表中写的类  方向的排列  form-group+form-control 堆叠，垂直方向排列,宽度100%  form-inline 内联表单 水平排列 | | |
| form-control  form-check写在单选按钮/多选按钮的父级容器中，有相对定位  form-check-inline 写在单选按钮/多选按钮的父级容器中，自带弹性布局，交叉轴垂直居中  form-check-input 自带绝对定位，需要与上面form-check/ form-check-inline配合使用  form-check-label 底外边距为0  form-text 上外边距为0.25rem | | |
|  | 默认的栅格布局 | 水平表单 |
| 行 | div.row | div.form-inline |
| 列 | div.col | div.col |

四.组件

|  |
| --- |
| boot把一些功能性的模块，进行了封装  我们只需要按照boot要求的层次结构，以及类名套用。  就可以完成 |

1.下拉菜单

|  |
| --- |
| 1.外部大包裹.dropdown  <div class="dropdown">  2.内部菜单按钮  class="dropdown-toggle" 作用，画向下的小箭头  **事件的激活 自定义属性 data-toggle="dropdown"**  <button class="btn btn-primary  dropdown-toggle" data-toggle="dropdown">  3.菜单栏，div.dropdown-menu  <div class="dropdown-menu">  4.菜单栏中有菜单项 a.dropdown-item  <a href="#" class="dropdown-item">  5.菜单项有几个可选的类名  dropdown-header <h5 class="dropdown-header">  active <a href="#" class="dropdown-item active">  disabled <a href="#" class="dropdown-item disabled">  在每个item之间添加分割线 <**div class="dropdown-divider"**></**div**> |

2.按钮组

|  |
| --- |
| 外包裹添加class="btn-group"  把内部的显示成一组  使用 btn-group-lg/sm 设置按钮组大小  使用 btn-group-vertical 设置垂直按钮组 |

3.信息提示框

|  |
| --- |
| 1.外包裹 class="alert"  可以修改颜色 alert-danger/warning/info......  如果内部的x想使用类.close，父级要添加 alert-dismissible  <**div class="alert alert-danger w-50 alert-dismissible"**>  2.内部有文本提示  3.内部有关闭提示框的X. data-dismiss="alert" class="close"  <**span class="close" data-dismiss="alert"**> |

4.导航

|  |
| --- |
| (1)水平导航  1.ul必须有类名 .nav .nav-justified  .nav-justified本身没有任何样式，为了给子元素item引路  2.li class="nav-item"  3.a class="nav-link"  4.其它样式  ul中，如果有.nav-justified，意味着导航项等宽显示  如果想使用justify-content-\*做主轴的对齐，需要把.nav-justified删掉  flex-row/column/reverse都可以生效 |
| (2)选项卡导航  1.外包裹 ul class="nav nav-tabs"  <**ul class="nav nav-tabs"**  2.ul内部需要有两套  2.1选项卡  li class="nav-item" <**li class="nav-item"**>  a class="nav-link" <**a class="nav-link"**  2.2对应内容  div class="tab-content" tab-content对自己没有任何修饰，为了给内部子元素 .tab-pane引路<**div class="tab-content"**>  .tab-content> elem class="tab-pane" 组合之后，elem就隐藏了  如果想显示 .tab-content>.active  <**div class="tab-pane active"**>  3.点击选项卡的动作  a.nav-link有事件的激活，使用自定义属性data-toggle="tab"实现切换  <**a href="#apple" class="nav-link active" data-toggle="tab"**>  4.把选项卡的href和内容id进行绑定  内容的id <**div id="apple" class="tab-pane active"**>  与选显卡a标签的href绑定  <**a href="#apple" class="nav-link " data-toggle="tab"**>  5.页面刷新，默认显示的选项和内容没有设置  选项卡需要active  <**a href="#apple" class="nav-link active" data-toggle="tab"**>  对应的内容也需要active  <**div id="apple" class="tab-pane active"**> |

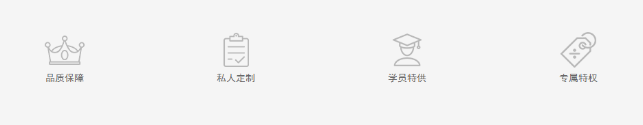
作业：

1.完成选项卡导航

2.完成boot中form操作



3.完成boot中弹性布局



#### DAY04 各种组件

一.组件

1.导航

|  |
| --- |
| (1)胶囊导航  .nav-pills替代选项卡导航的nav-tabs  data-toggle="pill"  其它部分同选项卡导航 |

2.导航栏

|  |
| --- |
| 在ul外层包裹一个div，就把导航变为了导航栏  1.外层包裹，div.navbar 弹性+相对定位  后面紧跟 navbar-expand-lg/md/sm (配合的类ul.navbar-nav)  此处的响应式，是设置导航在某一个屏幕下，横向显示，  其它屏幕纵向显示  2.导航可以使用ul。添加.navbar-nav(默认弹性，主轴为column)  配合之前的navbar-expand-lg/md/sm，可以使主轴变为row  3.li.nav-item>a.nav-link  注意：在外包裹中，可以使bg-color，设置导航栏颜色  整个导航栏横向显示，还是纵向显示，是靠navbar-expand-lg/md/sm控制的 |

3.折叠

|  |
| --- |
| 控制内容的隐藏与显示  在button中添加data-toggle="collapse" 和折叠目标 data-target="#内容id" #内容id对应折叠内容的id  如果使用a标签，直接在href中填写对应折叠内容的id  在内容中添加id和class="collapse"实现折叠 |

4.折叠导航栏

|  |
| --- |
| 1.外部包裹div .navbar .navbar-expand-\* .bg-dark .navbar-dark  .bg-dark .navbar-dark保持一致，都是dark/light  .navbar-dark本身没有样式，为子元素选择器引路  影响的子元素有.navbar-brand .nav-link .navbar-toggler .navbar-toggler-icon  2.外包裹内部  2.1 不折叠的选项 a. navbar-brand  2.2 折叠按钮 button.navbar-toggler  .navbar-expand-\* .navbar-toggler{设置什么时候显示/隐藏}  btn需要写自定义属性 data-toggle="collapse" data-target="#折叠内容id"  2.3内容div.collapse .navbar-collapse id="折叠内容的id"  .navbar-collapse 设置折叠导航打开的位置  2.3.1 导航的内容  ul.navbar-nav>li.nav-item>a.nav-link |

5.卡片

|  |
| --- |
| 通过.card创建一个卡片  .card-header 卡片的头部  .card-body 卡片的内容  .card-footer 卡片的底部  用于文字或者图片的展示 |

6.手风琴(卡片和折叠)

|  |
| --- |
| 1.最外层div#parent 这里的id是为了做data-parent="#parent"  目的是，一个折叠打开，其它在这个包裹的折叠都自己关闭  2.div.card>div.card-header>a.card-link  a标签中写折叠事件和折叠对象  data-toggle="collapse" href="#p1"  3.折叠内容  div.collapse#p1>div.card-body>p  4.注意，.card-body和.collapse不能在同一个div上,会冲突，会卡顿  5.如果希望一个卡片打开，其它卡片折叠  需要在折叠元素上添加事件data-parent="#parent"外层大包裹的ID |

7.媒体对象

|  |
| --- |
| 容器添加.media(flex)  项目有img  div.media-body  由于img和里面的div都是弹性项目，可以使用弹性布局的一切类  比如 img.align-self-center |

8.轮播

|  |
| --- |
| .carousel 旋转木马  最外层大包裹负责让图片轮播动起来  div.carousel data-ride="carousel"  1.图片的轮播  添加图片的包裹 .carousel-inner  每一张图片的包裹 .carousel-item  显示的图片 .active  可选择的样式类名  在图片上添加文字和标题 .carousel-caption |
| 2.导航指示符  ul.carousel-indicators  li data-slide-to="0" 滑动至哪一张图片，从0开始  data-target="最外层包裹.carousel的id"  重写boot提供的样式，让指示器变为圆点，颜色为#0aa1ed  <**style**>  .**carousel-indicators li**{  **width**:0.7**rem**;**height**:0.7**rem**;  **border-radius**: 50%;  **background**:**#fff**;  }  .**carousel-indicators** .**active**{  **background**:**#0aa1ed**;  } </**style**> |
| 3.左右箭头  a.carousel-control-prev>span.carousel-control-prev-icon  a.carousel-control-next>span.carousel-control-next-icon  a标签中有事件 data-slide="prev/next"  事件的目标href="#demo"  .**carousel-control-prev**, .**carousel-control-next**{  **background**:**rgba**(0,0,0,0.4);  **width**:3%;  **height**:20%;  **top**:40%; } |

9.模态框

|  |
| --- |
| modal模态框是覆盖在父窗体上的子窗体  模态框可以在不离开父窗体的情况下，与用户互动，提交一些交互信息  1.外包裹div.modal  2. div.modal内部一个div.modal-dialog  3. div.modal-dialog内部一个div.modal-content  4. div.modal-content中有modal-header modal-body modal-footer |

二.其它组件

1.徽章

|  |
| --- |
| 就是添加颜色样式和圆角  .badge .badge-pill .badge-danger/warning...... |

2.巨幕

|  |
| --- |
| 巨大的内边距  div.jumbotron |

3.分页

|  |
| --- |
| 外包裹ul.pagination创建分页  使用pagination-lg/sm 设置分页条的大小  内部li.page-item>a.page-link  使用active和disabled修饰li |

4.面包屑导航

|  |
| --- |
| 面包屑导航用于有层次关系的导航  ul.breadcrumb>li.breadcrumb-item>a  改变item的连接符号  .**breadcrumb-item** + .**breadcrumb-item**::**before**{  **content**:**">"**; } |

5.进度条

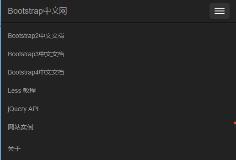
|  |
| --- |
| 进度条槽 div.progress  进度条在槽里写 div.progress-bar  w-\*是进度 bg-\* 进度条颜色  progress-bar-striped设置条纹进度条  progress-bar-animated 为进度条设置动画  可以在同一个槽上，添加多个进度 |

boot中重点

|  |
| --- |
| 媒体查询+栅格布局+scss |

作业：

1.完成boot中文主页，折叠导航栏



2.完成学子首页，轮播图

#### DAY05 Scss

一.Scss

1.CSS有几个缺点

|  |
| --- |
| 语法不够强大，没有变量和合理的样式复用机制  使得逻辑上相关的属性值必须以字面的形式重复输出  难以维护  动态的样式语言为css富裕了动态语言的特性  极大的提高了样式语言的可维护性 |
| 常见的动态样式语言：  1.scss/sass(scss兼容sass,scss更接近css的语法格式)  2.stylus  3.less |
| 动态语言---css的预处理----->\*.css (浏览器才可以解析) |

2.什么是SCSS

|  |
| --- |
| 是一款强化css的辅助工具  在css的语法上，增加了变量，嵌套，混合，导入，函数等高级功能。这些拓展令css更加强大与优雅 |

3.scss的作用，有助于更好的管理样式文件，以及更高效的开发项目

4.scss的使用

|  |
| --- |
| 1.在服务器端使用 10:21上课  （1）安装nodejs解释器  （2）安装scss的编译程序  在线安装npm install -g node-sass  注意：要求nodejs版本在8.11以上 |
| 2.把scss包中的4个文件拷贝进nodejs文件夹  在黑窗口或者ws的控制台(alt+f12)输入  node-sass -v 检测sass版本  如果输入版本，说明sass安装成功 |
| 3.编写01.scss文件  把.scss转化成.css文件  在正确路径下，打开黑窗口，输入  node-sass scss/01.scss css/01.css  如果生成01.css则正确 |
| 批量把scss转换成css，多文件转换命令  node-sass scss文件夹 -o css文件夹 |
| 单文件监听命令，scss文件一旦保存，自动转换为css文件  node-sass -w scss/01.scss css/01.css |
| 多文件监听  node-sass -w scss -o css |

二.SCSS的基础语法

1.变量

|  |
| --- |
| 使用$表示变量  变量的命名规范，遵循css中选择器的命名规范，  可以包含\_ -,不能以数字开头，见名知意 |
| $jd-red:#f10125;颜色变量  $w:100px;数值变量  $before-content:"子曾经曰过"字符串变量  $border-style:solid;样式变量 |
| 注意：  1.声明变量时，变量值可以应用其它的变量  2.变量定义在{}规则块外边，整个样式文件中都可以使用  如果定义在{}规则块内部，只能在当前规则块中使用  3.声明重复变量，后声明的变量会覆盖前面的变量  4.!default规则,如果变量已经声明赋值了，那就用它之前声明的值  否则使用现在声明的值 |
| 样式的嵌套规则  **#content**{  **color**:**#f00**;  **div**.**top**{  **margin**:0 **auto**;  **h1**{**font-weight**: **normal**}  **p**{**font-size**:20**px**;}  }  **div**.**middle**{**background**:**#00f**;} } **a**{  **color**:**#f00**;  &:**hover**{  **color**:**#00f**;  } } |
| 群组选择器的嵌套  **nav**,**div**,**footer**{  **a**{**color**:**#000**;  &:**hover**{**color**:**#f00**;}  } } |
| 属性嵌套  **div**{  **border**:{**style**:**solid**;**width**:1**px**;**color**:**#f00**;} } |

练习

|  |
| --- |
| 一个200\*200的div.红色背景，鼠标悬停过渡到黄色背景，并且变为圆形，历时2秒。使用scss |

2.导入scss文件

|  |
| --- |
| 在SCSS中，局部文件名以下划线开头  这样做，sass在编译时就不会编译以下划线开头的文件，而只是把这个文件用作与导入。  引用局部文件时，是关键字@import "局部文件名"，局部文件省略了下划线和后缀。  并且一个局部文件可以被多个SCSS文件引用 |

3.混合器

|  |
| --- |
| 把需要在多个样式中出现的，相同的部分提取出来，封装到混合器中  关键字 @mixin 混合器名称{重用的样式}  使用，关键字 @include 混合器名称。就可以在很多的样式中，使用封装好的样式了 |

练习:

|  |
| --- |
| a标签文字为灰色无下划线，用列表布局。ul>li\*3>a  在scss中编写样式，边框圆角使用混合器方式编写。  鼠标悬停在链接上，文字颜色变红 |
| 带参数的混合器（类似带参数的function）  定义混合器的时候，在（）添加参数  调用混合器的时候，在（）把参数补上  **@mixin lin-colors**(**$normal**,**$hover**,**$visited**){  **color**:**$normal**;  &:**hover**{**color**:**$hover**}  &:**visited**{**color**:**$visited**} } **ul li a**{  **@include lin-colors**(**#f00**,**#0f0**,**#00f**); } **#content a**{  **@include lin-colors**(**#faa**,**#afa**,**#aaf**); } |

4.继承

|  |
| --- |
| 继承就是说一个选择器可以继承另一个选择器定义的所有样式  在css中的表现形式是两个选择器共有的部分，变成了群组选择器 |

练习

|  |
| --- |
| 在选择器.my-border中定义1px solid transparent  宽度400，距离左边有20px。  在选择器.my-bottom-border中继承.my-border  设置底边框为红色，鼠标悬停时底边框为蓝色 |

三.运算

1.数字

|  |
| --- |
| 加减乘除，求模取整  会在不同单位间转换值  **width**:1**in**+8**pt**;  scss不能转换相对单位  **height**:1**rem**+1**em**;转换失败 |

(1)注意：加法

|  |
| --- |
| **p**::**before**{  **content**:**"Microsoft"**+**yahei**;  **font-family**: **A**+**"rial"**; }  结果  **content**: **"Microsoftyahei"**; **font-family**: **Arial**;  + 可以用于连接字符串  如果用引号去连接无引号的字符串，结果是有引号的  如果用无引号去连接有引号的字符串，结果是无引号 |

(2)减法

|  |
| --- |
| - 会被优先解析为变量名，所以使用变量和减法，需要-前后添加空格  **width**: **$size** - **$my-width** |

(3)除法

|  |
| --- |
| 在scss中，除号经常起到分隔的用途 /  **p**{  **font**:10**px**/5**px**;  **$width**:1000**px**;  **width**: **$width**/2;  **height**: (500**px**/2);  **margin**:5**px**+8**px**/2**px**; }  在以下情况视为除法运算  1.如果值，或者值的一部分，是变量或者函数的返回值  2.如果值被小括号包裹，视为除法  3.如果值是算术表达式的一部分，视为除法 |

(4)运算表达式与其他值连用时，用空格做连接

|  |
| --- |
| margin:4px+5px auto; |

(5)在有引号的字符串中，使用#{}插值语句可以添加动态的值

|  |
| --- |
| **content**: **"I ate** #{16+23} **baozis"**; |

2.颜色的运算

|  |
| --- |
| 颜色值是分段计算的，红+红 绿+绿 蓝+蓝  rgb(23,32,45)+rgb(11,23,33)  rgba(11,22,33,0.2)+rgba(22,33,44,0.1)  两个rgba相加，alpha的值，必须相等才可以计算，因为算术运算符不会作用到alpha |

四.函数

1.scss定义了多种函数，有些函数甚至直接在css中调用

|  |
| --- |
| 1.颜色函数  rgba(red,green,blue,alpha)  hsl(hue,saturation,lightness)  hue:色调 取值0~360 3个色段 每120一个色段  saturation：饱和度0.0%~100.0%  lightness:亮度 0.0%~100.0% |
| 2.数字函数  round($value) 四舍五入  ceil($value)向上取整  floor($value)  max($v1,$v2,......)  min($v1,$v2,......)  random() |
| 3.字符串函数  unquote($string) 删除字符串的引号  quote($string) 给字符串添加引号  To-upper-case()  To-lower-case() |

2.自定义函数

|  |
| --- |
| @function 函数名($n){  函数体；  @return 结果；  } |

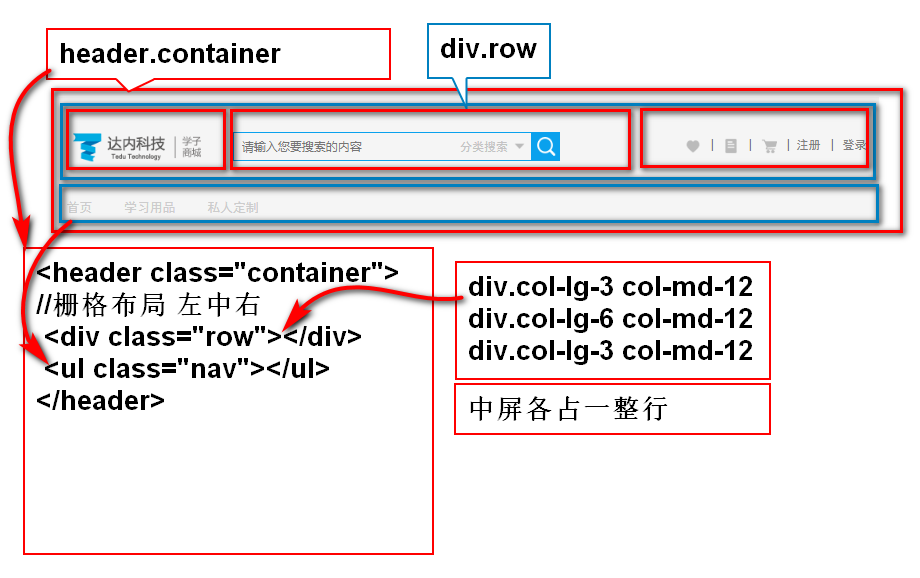
四.控制指令

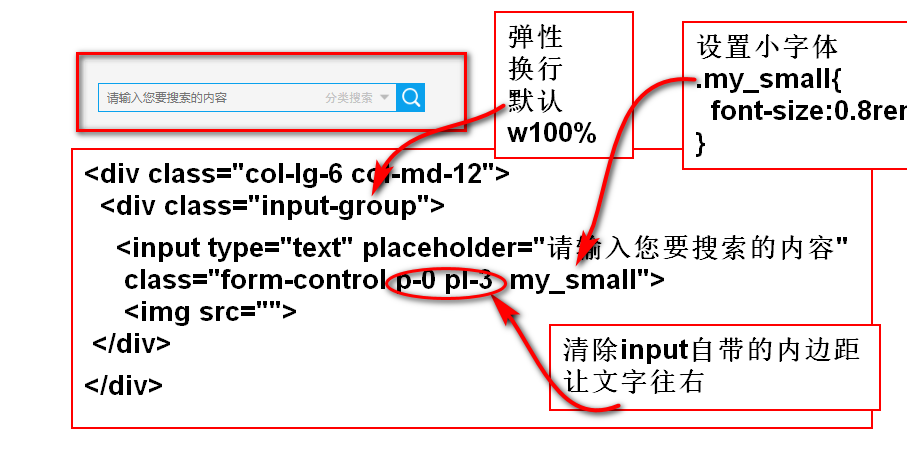
|  |
| --- |
| **@if** 1+1==4{**border-radius**:50%} **@else if**(1-1==-1){**border-radius**:30%} **@else** {**border-radius**:10%}  @if , @else if , @else  boolean表达式，可以添加括号，也可以不加 |

项目

一.学子商城首页

1.写头和底部





作业：

1.完成首页的底部



2.完成1楼



## 阶段2: JS、DOM&BOM、JQUERY、VUE

### 1、JS

#### DAY01 正则

张东

微信: 18301092802

准备:

1. 书: 犀牛书 —— JavaScript权威指南

2. 微信公众号: 前端大全

3. 上届笔记: 打印成纸质

不允许记笔记！

正课:

正则表达式:

1. 正则表达式: Regular Expression

什么是: 规定一个字符串中内容出现规律的规则

何时: 2种:

1. 格式验证:

2. 模糊查找关键词:

如何:

1. 最简单的正则表达式，是一个关键词的原文

2. 字符集:

什么是: 规定一位字符上多种备选字列表的集合

何时: 只要一位字符上有多种备选字时

如何: [备选字列表]

强调: 一个字符集只能匹配一位字符

简写: 如果备选字符是连续的，可用-省略中间字符

比如: 1位数字: [0-9]

1位小写字母: [a-z]

1位大写字母: [A-Z]

1位字母: [A-Za-z]

1位字母或数字: [A-Za-z0-9]

1位汉字: [\u4e00-\u9fa5]

需求: 车牌号:

1位汉字 1位大写字母 中点 五位大写字母或数字

[\u4e00-\u9fa5][A-Z]·[A-Z0-9] [A-Z0-9] [A-Z0-9] [A-Z0-9] [A-Z0-9]

排除: [^47] 除了4和7之外都行

强调: 范围太大！

3. 预定义字符集: 4种:

1位数字: \d

1位字母数字或下划线: \w 等效于[A-Za-z0-9\_]

1位空字符: \s 包括: 空格，tab… …

通配符: .

问题: 单靠字符集，无法灵活定义字符的个数

解决: 用量词

4. 量词:

什么是: 规定一位字符出现次数的规则

何时: 只要定义一位字符出现的次数

如何: 字符集量词

放在字符集之后，修饰相邻的前一个字符集

包含: 2大类:

1. 有明确数量边界: 3种:

{n,m} 最少n个，最多m个

{n,} 最少n个，多了不限！

{n} 必须n个，不能多也不能少

2. 没有明确数量边界: 3种:

? 可有可无，最多一个

\* 可有可无，多了不限

+ 至少一个，多了不限

问题: 量词默认仅修饰相邻的前一个字符集

解决: 分组

5. 分组:

什么是: 用()将多个字符集包裹为一组

何时: 只要希望一个量词能修饰多个字符集时

身份证号:

15位数字 2位数字 1位数字或Xx

后三位整体可有可无，最多1次

\d{15}(\d{2}[0-9Xx])?

6. 选择: 或

什么是: 在多个条件中任选其一匹配

何时: 只要在多个条件中任选其一匹配即可

如何: 规则1|规则2

作业: 编写正则，检查: 微 信 wei xin w x

7. 特殊位置: 3个:

字符串开头 ^

字符串结尾 $

何时: 仅匹配开头或结尾的关键词时

比如: 仅匹配开头的空字符: ^\s+

仅匹配结尾的空字符: \s+$

同时匹配开头或结尾空字符: ^\s+|\s+$

单词边界 \b

凡是能分割两个单词的字符

包括: 开头, 结尾，空格，标点…

比如: 匹配每个单词首字母: \b[a-z]

8. 匹配相同内容: 2步

1. 给第一个关键词加()

2. \1 代表和()中相同的内容

比如: 匹配连续的多个相同字母/字符

([a-z])\1\*

2. String的正则API:

1. 正则表达式: 见day01 am.pdf 用浏览器打开

2. String 的正则API:

查找关键词: 4种:

1. 查找一个固定的关键词出现的位置:

var i=str.indexOf("关键词",starti)

从str中starti位置开始，找下一个"关键词"的位置i

如果找到，返回关键词第一个字所在的位置i

如果找不到，返回-1

问题: 不支持正则，只能查找一个固定的关键词

2. 模糊查找一个关键词的位置:

var i=str.search(/正则/)

在str中从0位置开始找下一个符合"正则"要求的敏感词的位置i

如果找到，返回关键词第一个字的位置

如果没找到，返回-1

问题: 正则表达式默认区分大小写

解决: 在第二个/后加后缀i,表示ignore忽略大小写

更高要求: 既返回关键词内容，又返回关键词位置

3. 模糊查找关键词，并返回关键词内容: 2种:

1. 只返回一个关键词的内容:

var arr=str.match(/正则/i)

在str中查找符合/正则/要求的关键词，并同时返回关键词的内容和位置

返回值: arr[ "0": 关键词, "index": i ]

数组本质: 所有数组底层其实都是关联数组

获取关键词内容: arr[0] ~~arr.0~~

获取关键词位置: arr["index"] <==> arr.index

如果找不到，返回null

正则表达式第2个问题: 只找一个就结束了！

解决: 在第2个/后加g，global 全部

2. 查找所有关键词的内容(不关心位置):

var arr=str.match(/正则/g)

查找str中所有符合/正则/要求的关键词保存到arr中

返回值: 所有关键词组成的数组

找不到: 返回null

4. 既查找每个关键词的内容，又查找每个关键词的位置:

match()不能实现:

match()不加g时，只能找1个关键词

match()加g后，不能获得位置

解决: RegExp对象的exec()方法

替换关键词: 2种:

1. 简单替换: 所有关键词都替换为统一的新值

str=str.replace(/正则/ig,"新值")

回顾: 所有字符串API都无权修改原字符串，只能返回新字符串！所以必须用变量才能借助修改后的新值。

将str中所有符合/正则/要求的关键词都替换为统一的"新值"

2. 高级替换: 根据找到的每个关键词的不同，动态返回对应的新值替换。

str=str.replace(

/正则/ig, function(kword){ return 新值; }

)

原理: 1. replace查找str中符合条件的每个关键词

2. 每找到一个关键词，就自动调用回调函数

3. 每次调用函数都自动传入本次找到的关键词

4. 回调函数根据本次传入的关键词，加工后生成新替换值返回

5. replace将本地回调函数返回的值，替换到正在遍历的位置。

切割字符串:

#### DAY02 API&FegExp

正课:

1. String的正则API
2. RegExp
3. Function

\*\*\*\*\*闭包

作业:

(微|w(ei)?)\s\*(信|x(in)?)

1. String的正则API

替换: 衍生: 删除关键字: 替换为空字符串

切割: 将一个字符串，按照切割符，切割成多段子字符串

1. 简单切割: 切割符是固定的

比如: [zhangdong@tedu.cn](mailto:zhangdong@tedu.cn)

no zuo no die

var arr=str.split(“固定的切割符”)

返回多段子字符串组成的数组

固定套路: 打散一个字符串为字符数组

var chars=str.split(“”)

1. 复杂切割: 切割符不是固定的

var arr=str.split(/正则/);

2. RegExp对象:

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

何时: 只要在程序中保存一条正则表达式或使用正则表达式执行验证和查找时

如何:

创建: 2种:

1. 用//: var reg=/正则/ig

问题: 正则是写死的，无法再运行时动态生成正则表达式

1. 用new: var reg=new RegExp(“正则”,”ig”)

好处: 参数是一个字符串格式的正则表达式，所以可以用任何程序的手段先动态生成字符串，再传给new RegExp变为正则表达式对象——动态生成正则

API:

1. 验证: 验证一个字符串的格式是否符合正则表达式的要求。

var bool=reg.test(str)

验证str是否符合reg的格式要求

问题: test其实是只要找到符合规则的，就返回true，不要求从头到尾完整匹配

比如: /\d{6}/.test(“123456a”) -> true

解决: 今后只要验证，必须前加^，后加$，表示必须从头到尾完全匹配才行

比如: /^\d{6}$/.test(“123456a”) -> false

其实: reg.test()也可判断一句话中是否包含敏感词，且更简单。

比如: if(reg.test(msg)) 如果msg中包含符合reg的敏感词

1. 查找: 既查找每个关键词的内容，又查找每个关键词的位置

var arr=reg.exec(str)

让reg去str中找下一个符合规则的关键词的内容和位置

返回值: 和str.match()不加g的情况是一样的

arr[“0”: 关键词, “index”: i ]

如果找不到了，返回null

vs str.match() :

str.match()始终只能找第一个关键词

reg.exec()反复调用时，可自动向后找下一个

总结: 查找关键词: 5种

1. 只判断是否包含: var bool=reg.test(str)
2. 查找一个固定关键词的位置: var i=str.indexOf(“关键词”)
3. 查找一个可变关键词的位置: var i=str.search(/正则/i)
4. 查找第一个关键词的内容和位置:

var arr=str.match(/正则/i)

arr[“0”:关键词, “index”: i ]

1. 查找所有关键词的内容:

var arr=str.match(/正则/ig)

arr[关键词1, 关键词2, … ]

1. 查找所有关键词的内容和位置:

do{

var arr=reg.exec(str);

if(arr!=null)

arr[“0”:关键词, “index”: i ]

else

break;

}while(true);

3. Function

什么是: 保存一段可重用的代码段的程序结构，再起一个名字

何时: 如果一段代码可能被反复使用

程序的重要设计原则之一: DRY Don’t Repeat Yourself

如何:

创建: 3种:

1. 用声明方式:

function 函数名(形参列表){ 函数体; return 返回值 }

问题: 会被声明提前(hoist)

回顾: 声明提前: 当程序开始执行前，先将var声明的变量和function声明的函数提前到当前作用域的顶部，集中创建。赋值留在原地。

习惯: 所有声明都集中放在当前作用域的顶部。

1. 用赋值方式:

var函数名=function (形参列表){ 函数体; return 返回值 }

揭露了js中函数的本质: 函数其实是一个保存代码片段的对象，函数名其实就是一个普通的变量。

不会被声明提前！

1. 用new:

var 函数名=

new Function(“形参1”,…,”函数体; return 返回值”)

重载(overload):

什么是: 多个相同名称不同参数列表的函数，调用时可自动根据传入参数的不同，选择对应的函数执行。

为什么: 减少API的个数，减轻调用者的负担

何时: 1件事，可能根据传入参数不同，执行不同的处理过程时

如何:

问题: js语法默认不支持重载效果

为什么: js中不允许多个同名函数同时存在

如果同时存在，最后一个同名函数会覆盖之前所有

解决: 变通实现

1. 只定义一个函数，不定义参数列表
2. 调用函数时，按需传入参数
3. 所有参数都会被函数中的内置对象arguments自动接收。

arguments: 每个函数中自动创建的可接受所有实参列表的类数组对象(长的像数组的对象)

vs 数组: 相同: 1. 下标, 2. .length, 3. 遍历

不同: 不是数组类型，API不通用！

1. 函数中根据参数列表的个数和内容，动态选择执行一种处理逻辑

匿名函数:

什么是: 定义函数时不指定函数名的函数

为什么: 节约内存

匿名函数用完后，不再被任何变量引用着，就被垃圾回收！

何时: 如果一个函数只用一次时

如何: 2个场景:

1. 回调函数:

什么是: 自己定义函数，却给别人去调用

1. 自调函数:

作业:

function add(){

}

console.log(

add(1,2,3), //6

add(1,2,3,4,5) //15

);

#### DAY03 Function

正课:

1. Function
2. 面向对象
3. Function

匿名函数:

如何:

1. 回调
2. 自调：函数定义后，立刻自己调用自己执行一次

为什么: 避免使用全局变量，避免全局污染

何时: 用匿名函数划分临时作用域，避免使用全局变量

如何: (function(){ … })()

//function是创建一个函数的意思——动作

//返回函数地址，比如0x1234

//0x1234() 调用匿名函数

总结: 今后，禁止使用全局变量！

今后，所有js代码一律放在一个大的匿名函数自调内！

作用域和作用域链: PPT

闭包: PPT

2. 面向对象:

什么是: 程序都是用对象结构来描述现实中一个具体事物的属性和功能

为什么: 便于大量数据的管理和维护

何时: 今后，所有程序都要用面向对象思想实现

如何: 三大特点: 封装，继承，多态

1. 封装:

什么是: 用对象结构集中存储一个事物的所有属性和功能

为什么: 便于大量数据的管理和维护

何时: 今后，只要使用面向对象思想，都要先将零散的数据封装为对象

如何: 3种:

1. 用{}创建对象:

var obj={

属性名:值,

… : …,

方法名:function(){

… this.属性名 …

}

}

访问对象成员(属性+方法): 对象.成员名

问题: 对象自己的方法内，要使用当前对象自己的属性，不能直接写属性名——报错！未定义

原因: 对象不是作用域！所以不在方法的作用域链上，所以，即使自己的方法，也不能直接访问自己对象的属性

解决:

不好的解决: 写死“对象名.属性名”

问题: 如果对象名发生改变，被迫同时修改方法内的语句

好的解决: “this.属性名”

this: 自动引用正在调用当前方法的.前的对象

何时: 只要对象自己的方法，要访问自己的属性时，必须加this.

常犯错误: 以定义方法的位置来判断this的指向。

正确: this与定义的位置无关，只与调用时.前的对象有关。2种:

1. obj.fun() this->obj
2. fun() this默认->window

2. 用new:

var obj=new Object();

obj.属性名=值;

obj.方法名=function(){ … }

js中对象的本质: js中一切对象底层都是关联数组

访问对象的成员: 2种:

obj[“属性名”]=obj.属性名

如果属性名是动态变化的，必须用obj[表达式]

访问对象中不存在的属性: 不报错！返回undefined

随时可给对象添加任何新属性: 不报错！而是自动创建

前两种方式: 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，代码大量冗余——不便于维护

3. 用构造函数创建对象:

什么是构造函数: 描述一类对象相同结构的函数

为什么: 代码重用

何时: 反复创建多个相同结构的对象时

如何: 2步:

1. 定义构造函数:

function 类型名(){

this.属性名=值;

… = …;

this.方法名=function(){

… this.属性名 …

}

}

1. 用构造函数反复创建多个相同结构的对象

var obj=new 类型名();

new: 4件事:

1. 创建一个空对象
2. ？
3. 调用构造函数，并将构造函数中的所有this，自动替换为新的空对象

通过强行给新对象赋值的方式，向新对象中添加新属性

1. 返回新对象地址

#### DAY04 OOP&ES5

正课

1. OOP
2. ES5
3. OOP

封装:

构造函数的问题：如果将方法定义放在构造函数中，会导致每次创建新对象时，都重复创建方法的副本——浪费内存

解决: 保存在一处，共用！

继承:

什么是: 父对象的成员，子对象无需重复创建，就可直接使用

为什么: 代码重用，节约内存

何时: 所有子对象，需要相同的方法或属性时，只要在父对象中添加一次，所有子对象就可共用

如何: js中继承，都是继承原型对象(prototype)

创建原型对象: 不用创建，买一增一:

每创建一个构造函数，都附赠一个空的原型对象

构造函数.prototype=自己的原型对象

何时继承:

new的第2步，自动设置继承关系

子对象.\_\_proto\_\_=构造函数.prototytpe

如何向原型对象中添加共有成员: 强行赋值:

构造函数.prototype.共有成员=值

11种内置类型，也都有原型对象:

将来只要能new的都是一种类型

只要是一种类型，都包含2部分组成:

1. 构造函数: 专门用于创建该类型的子对象
2. 原型对象:专门用于保存该类型的子对象共有的成员

比如: Array, Date, RegExp, Function, …

原型链:

什么是: 由多级父元素逐级继承形成的链式结构

保存着: 所有对象可访问的属性和方法

控制着属性和方法的使用顺序: 先用自己的，自己没有才延原型链向上找

多态:

什么是: 一个函数在不同情况下表现出不同的状态

包括: 2种:

1. 重载:
2. 重写(override)

什么是重写: 子对象觉得父对象的成员不好用，可在本地定义自有同名成员，覆盖父对象的成员

何时: 只要觉得父对象的成员不好用，都可重写一个新的

如何: 只要在子对象本地定义同名成员即可

静态方法与实例方法:

什么是实例方法: 给某个类型的子对象调用的方法

何时: 如果要求，方法执行时，必须先有一个具体的子对象，才能用。

如何: 要么直接定义在对象上，要么定义在对象的原型对象上

什么是静态方法: 不需要具体对象，也可直接执行的方法

何时: 如果不需要任何具体对象，也想调用函数时

如何: 将函数定义在构造函数对象上

构造函数.静态方法=function(){ … }

调用时: 构造函数.静态方法();

自定义继承关系:

1. 只修改一个对象的父对象:

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

Object.setPrototypeOf(child,father)

设置set child 的of 爹prototype 为 father

问题: 一次只能改一个子对象的爹

1. 同时修改所有子对象的原型对象

其实就是修改构造函数的prototype属性

构造函数.prototype=father

时机: 必须在定义完构造函数后，开始创建子对象之前，就要更换。

1. 正式的两种类型间的继承:

问题: 两种类型间包含了部分相同的属性结构和方法定义

解决: 抽象父类型:

2步:

1. 定义抽象父类型:
2. 在父类型构造函数中包含相同的属性定义
3. 在父类型的原型对象中包含相同的方法定义
4. 删除子类型中相同的内容
5. 让子类型继承抽象父类型:
6. 让子类型构造函数继承父类型构造函数

Object.setPrototypeOf(

子类型.prototype, 父类型.prototype

)

1. 让子类型构造函数借用父类型构造函数

错误: 直接调用父类型构造:

父类型(参数值)

后果: 父类型构造函数中的this->window

解决: 函数内的this不是想要的，都可用.call()的方式，将正确的this注射进函数内，替换其中错误的this

正确的方法:

父类型构造.call(正确的this,参数值..)

结果: 执行时，父类型构造中的this，被外部正确的this给替换了！

2. ES5:

严格模式: 比普通js运行机制要求跟严格的执行模式

何时: 今后，所有项目代码，都必须运行在严格模式下！

如何: 在当前作用域顶部: “use strict”;

如果加载<script>元素或js文件顶部，则整段js代码都启用严格模式

如果只加在函数内的顶部，则仅函数内启用严格模式

要求:

1. 禁止给未声明的变量赋值:

普通js中，可以给未声明的变量赋值

结果: 会自动在全局创建该变量

并且造成误会！

1. 静默失败升级为错误:

静默失败: 执行不成功，也不报错！

1. 普通函数调用/匿名函数自调中的this，不再指向window，而是undefined
2. 屏蔽了arguments.callee

什么是arguments.callee: 函数内自动创建的，引用当前函数本身的关键字。

何时: 在使用递归算法时，可在函数内避免写死当前函数名。

屏蔽arguments.callee是在屏蔽递归算法

为什么: 递归算法效率极低，重复计算量太大！

解决: 几乎所有递归都可用循环代替！

保护对象:

保护单个属性:

ES5将对象属性划分为两大类:

命名属性: 凡是可用.直接访问到的属性

又被分为两大类:

数据属性: 直接存储属性值的属性

访问器属性: 不直接存储属性值，专门提供对另一个数据属性的保护

内部属性: 不能用.直接访问的属性

保护数据属性:

ES5将每个数据属性，都变成一个缩微的小对象：

{ //四大特性:

value: 实际存储属性值,

writable: true/false, //控制是否可修改

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

但是，用.依然可强行访问该属性

configurable: true/false, //控制是否可删除该属性

//控制是否可修改另外两个特性

}

保护对象结构:

#### DAY05 ES5&ES6

正课:

1. ES5
2. ES6
3. ES5

保护对象

保护属性

数据属性:

四大特性: value, writable, enumerable, configurable

获取: var 属性对象=

Object.getOwnPropertyDescriptor(obj,”属性名”)

修改四大特性:

Object.defineProperty(obj,”属性名”,{

开关: true/false,

… : …

})

问题: 一次只能修改一个属性的四大特性

Object.defineProperties(obj,{

属性名:{

开关: true/false,

… : …

},

属性名:{

开关: true/false,

… : …

}

})

问题: 无法使用自定义的规则保护属性

访问器属性: 访问器属性是不保存属性值，仅提供对另一个数据属性的保护——保镖

何时: 使用自定义规则保护属性时

如何:

前提: 先有一个实际存储数据的隐藏的数据属性

定义访问器:

Object.defineProperties(obj,{

\_数据属性:{ enumerable:false,configurable:false},

访问器属性:{

get:function(){ return this.\_数据属性},

set:function(value){

if(value符合规则)

this.\_数据属性=value;

else throw new Error(“自定义错误信息”)

},

enumerable:true,

configurable:false

}

})

保护对象的结构: 3个层次:

1. 防扩展: 禁止给对象添加新属性

Object.preventExtensions(obj)

阻止 扩展

设置obj对象，禁止扩展新属性

1. 密封: 在防扩展基础上，再禁止删除所有属性

本质: 在设置防扩展同时，自动设置所有属性的configurable为false

如何: Object.seal(obj)

1. 冻结: 在密封基础上禁止修改属性值

本质: 在密封基础上，自动将所有属性的writable设置为false！

如何: Object.freeze(obj)

Object.create(): 没有构造函数，也想创建子对象

如何: var child=Object.create(father,{

//defineProperties的语法

属性:{ 四大特性 }

})

创建新对象child，然后让child继承father，并为新对象添加自有属性

替换this:

.call/apply/bind

1. 调用一次函数，并临时替换函数中的this为指定对象：.call/apply

fun.call(obj,实参…)

2件事: 1. 调用一次函数执行

2. 在执行时临时替换其中的this为obj

.apply可打散数组为单个参数值后，分别传入函数:

如果函数需要多个参数值，但给的参数值却集中保存在一个数组中。

1. 基于原函数创建一个完全相同的新函数，并永久绑定this为指定对象

var newFun=fun.bind(obj,实参…)

创建一个和fun完全相同的新函数newFun

永久将newFun中的this绑定为obj

永久将newFun中的部分参数绑定为实参…

固定用法: 如果替换回调函数中的this，必须用.bind()

因为回调函数不是立刻执行！且不止执行一次！

数组API: 7个:

1. 查找: 查找一个指定的元素在数组中的位置

同字符串操作: var i=arr.indexOf(元素, starti)

1. 判断: 判断数组中的元素是否符合要求

2种:

1. 判断是否所有元素都符合要求：

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

//elem: 当前元素值

//i: 当前位置

//arr: 当前数组

return 判断条件;

})

every会拿着回调函数去每个元素上执行一次

每次执行时，都会将当前元素值，当前所在位置，当前数组对象，传入函数中

函数内用提供的参数值，判断当前元素是否符合条件，并返回判断结果

必须回调函数在每个元素上执行时，都返回true，整体判断结果才返回true！

1. 判断是否包含符合要求的元素:

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

//elem: 当前元素值

//i: 当前位置

//arr: 当前数组

return 判断条件;

})

1. 遍历:
2. 过滤:
3. 汇总:

#### DAY06 ES5&ES6

正课:

1. ES5
2. ES6
3. ES5

数组API:

遍历: 对每个元素都执行相同的操作

1. 直接修改原数组

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

elem…

arr[i]…

})

forEach会拿着回调函数去每个元素上执行一次

每次执行时，回调函数内都可用arr[i]修改原数组中的每个元素值。

1. 不直接修改原数组，而是返回新数组

var newArr=arr.map(function(elem,i,arr){

return 新值

})

map先创建一个新的空数组

然后拿着回调函数去每个元素上执行一次

每次执行时，都将回调函数返回的新值，放入新数组中相同位置

最后，将保存新值的新数组返回，原数组保持不变

过滤: 复制出原数组中符合条件的元素，组成新数组

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

return 条件

})

filter先创建一个新的空数组

会拿着回调函数去每个元素上执行一次

每次执行时，都获得当前元素的判断结果

只有返回结果为true的元素，才被放入新数组中返回

汇总: 对数组中的元素进行各种统计，得出最终结论

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

//prev: 截至到当前元素之前的临时汇总值

return prev+elem;

},base)

2. ES6: 不改变原理的基础上，尽量简化了API

let/const: 代替var/const来声明变量和常量的

为什么:

1. 因为传统的var和const会被声明提前
2. 没有块级作用域: for if while都不是作用域，其中的变量会被提前到外部，影响外部的代码。

使用后:

1. 不会被提前，不允许提前使用未声明的变量
2. 让for if while等程序块，也变为作用域

原理: let自动添加匿名函数自调来划分临时作用域，并将变量名前自动加\_，与其他变量区分。

参数增强:

1. 默认值(default): 即使用户不传入参数，参数也有备用的默认值代替

如何: 定义函数时: function fun(形参,…,形参=默认值)

强调: 有默认值得参数，必须在形参列表的末尾

1. 剩余参数(rest): 代替arguments接收所有不确定个数的参数

arguments: 2个问题:

1. 不是数组类型，不能使用数组API
2. 只能获得全部参数，无法有选择的获得部分

如何: 定义函数时: function fun(形参,…, …数组名)

其中: 数组可收集除前几个确定参数之外的剩余参数

1. 散播(spread): 代替apply专门用于打散数组为单个元素，再传入函数。

apply的问题: 主要功能是替换this，顺便可以打散数组为单个值。

如何: 调用函数时: fun(…数组)

箭头函数: 对一切回调函数或匿名函数自调的简写

如何: 回调函数: function (形参,…){ … }

可简化为: (形参,…)=>{ … }

如果只有一个形参，可省略()

如果函数体中只有一句话，可省略{}

如果这仅有的一句话还是return xxx，则必须省略return

特点: 箭头函数内外的this是同一个/共通的

总结: 如果希望内外this相同时，应该简化

如果反而希望内外this不同时，不能简化！

解构: 从一个大的对象中抽取想要的部分成员，单独使用

3种:

1. 数组解构: 从数组中抽取想要的元素出来，单独使用

如何: 下标对下标:

var arr=[1,2,3];

↓

var [x,y,z]=arr

var x,y,z

结果: x=1, y=2, z=3

arr[0]//麻烦，且没有意义

1. 对象解构: 从对象中抽取想要的成员出来，单独使用

如何: 属性名对属性名

var obj={x:1, y:2, z:3}

↓ ↓ ↓

var {x:a, y:b, z:c}=obj;

obj.x obj.y obj.z //麻烦

结果: a=1, b=2, c=3

简写: var {x:x, y:y, z:z}=obj;

可简写为var {x,y,z}=obj;

#### DAY07 ES6

正课:

1. ES6
2. ES6

解构:

参数解构:

何时: 当参数列表中有多个参数值不确定时

问题: 形参列表中有多个参数不确定是否有值

解决: 参数解构:

如何: 2步:

1. 定义函数时: 形参列表使用对象语法

2. 调用函数时：也使用对象语法传参

for of: 最简化的遍历索引数组或类数组对象的方式：

总结: 遍历索引数组:

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

var elem=arr[i];

万能

1. arr.forEach((elem,i,arr)=>{

})

无法控制遍历顺序和步调

1. for(var elem of arr){ … }

of会依次取出arr中每个元素值，保存在of前的变量中

无法获得位置i

无法修改原数组中的值

for of vs for in

for of 专门遍历下标为数字的索引数组/集合/字符串

of取得是元素值

for in 专门遍历下标为自定义名称的关联数组/对象

in取得是属性名

class:

什么是: 集中描述一类对象统一属性结构和行为的程序结构

何时: 今后只要创建一种类型，必须用class

如何:

1. 用class{}包裹原来的构造函数和原型对象方法
2. 构造函数名提升为class名，构造函数要更名为constructor
3. 直接定义在class内的方法，默认保存在原型对象中，且不用加Xxx.prototype前缀和”=function”

继承: 2步

1. class child extends father{}

不再需要Object.setPrototypeOf(…)

1. 子类型构造函数中: super(参数值)//不需要传入this

super自动指向extends后的父类型

super()调用父类型构造函数的意思

访问器属性:

1. 依然要在构造函数内定义受保护的隐藏的数据属性
2. 在class内:

get 访问器属性名(){ return this. …}

set 访问器属性名(value){ 验证value并给this.xxx赋值}

静态方法:

class Xxx{

static 方法名(){ }

}

Promise: 代替回调函数，实现多个异步调用，顺序执行

问题: 使用回调函数方式实现多个异步调用顺序执行会导致回调地狱(callback hell)

根源: 所有回调函数规定，在调用函数前，就要提前传入到函数中

解决: 让回调函数在函数后传入

如何:

前提: 不要在参数列表里传递回调函数了！

1. 定义函数支持Promise

在原函数内，用new Promise()包裹所有原代码

再在源代码外层套一层function(){}

function()中必须接收Promise附赠的open开关

在当前函数异步任务调用后，自动打开开关open();

function fun(){

return new Promise(function(open){

异步任务

异步任务执行完:open()

})

}

1. 将多个任务串联起来:

第一个函数().then(第二个函数).then(…)

强调: 中间的then中的函数，不要加()，因为不是立刻执行，且中间的函数必须支持Promise

错误处理: 2步:

1. new Promise(function(open,err){

//如果出错:

err(“错误消息”) //通向最后的.catch()

})

1. 在函数1().then().then()…..catch(function(errMsg){ … })

无论中间哪个then出错，都会执行最后的catch，并将then中err(“错误消息”)传给errMsg。

### 2、DOM&BOM

#### DAY01 DOM

正课:

1. 什么是DOM
2. DOM Tree
3. 查找:
4. 什么是DOM: Document Object Model

DOM: 专门操作网页内容的API标准——W3C

为什么: 统一所有浏览器操作网页内容的API

几乎所有浏览器100%兼容DOM API

包括: 查找网页上的元素

修改元素

添加新元素

删除现有元素

事件绑定

何时: 只要操作网页内容，只能用DOM API

1. DOM Tree

什么是: 内存中保存一个网页中所有内容的树形结构

为什么: 因为网页中的内容，都是有明显的上下级包含关系的。

何时: 只有理解了网页的树型结构，才能顺利找到要操作的DOM元素

如何:

1. 当浏览器读到一个HTML文件时，会自动在内存中创建一个树根节点：document
2. 浏览器解析HTML中的内容，每读到一项内容，就创建一个节点对象，然后将节点对象添加DOM树上

节点对象:

网页中每一项内容(元素，属性，文本)都是一个节点对象

节点对象的三个公共属性：——了解

node.nodeType: 判断该节点对象的类型:

值: 一个数字:

9 document;

1. Element; <h1> <span> <button>
2. Attribute; id title class
3. Text; 元素中或元素之间的内容

node.nodeName: 判断节点的名称:

document #document

Element 全大写的标签名: BUTTON SPAN …

Attribute 获得属性名

Text #Text

node.nodeValue: 获得节点的值

document null

Element null

Attribute 属性值

Text 文本内容

3. 查找:

1. 不需要查找，就可直接获得的元素:

<html> document.documentElement

<head> document.head

<body> document.body

2. 按节点间关系查找:

节点树: 包含所有网页内容的最完整的树结构

两大类关系:

1. 父子关系:

node.parentNode 获得node节点的父节点

node.childNodes 获得node下所有直接子节点的集合

node.firstChild 获得node下第一个直接子节点

node.lastChild 获得node下最后一个直接子节点

1. 兄弟关系:

node.previousSibling 获得node节点的前一个兄弟

node.nextSibling 获得node节点的后一个兄弟

问题: 节点树会受到看不见的回车，换行，空字符的影响

程序员通常只关心元素节点

解决: 元素树: 仅包含元素节点的树结构

好处: 不包含看不见的空字符

强调: 元素树不是一棵新树，只是节点树的子集。

两大类关系:

1. 父子关系:

node.parentElement 获得node节点的父元素

.parentNode

node.children 获得node下所有直接子元素的集合

node.firstElementChild 获得node下第一个直接子元素

node.lastElementChild 获得node下最后一个直接子元素

1. 兄弟关系:

node.previousElementSibling 获得node节点的前一个兄弟元素

node.nextElementSibling 获得node节点的后一个兄弟元素

遍历: 指定父元素下所有后代元素

2步:

1. 先定义函数，仅遍历指定父元素的直接子元素

2. 对每个直接子元素，调用和父元素完全相同的操作

#### DAY02 查找修改

正课:

1. 查找:
2. 修改:
3. 查找:

3. 按HTML特征查找: 4种：id tag name class

1. 按id查找: var elem=document.getElementById(“id”);

返回值: 返回一个元素对象，找不到，返回null

强调: 只能用document调用

2. 按标签名查找:

var elems=parent.getElementsByTagName(“tag”)

返回值: 返回所有符合条件的元素对象的集合

如果找不到，返回空集合: [].length=0

强调: 1. 可在任意父元素上调用

2. 不仅查找直接子元素，且在所有后代中查找

3. 按name属性查找:

var elems=document.getElementsByName(“name”)

何时: 专门查找表单中收集数据的表单元素

返回值: 多个表单元素的集合，如果找不到返回[]

如果明知道只找到一个元素: elems[0]

强调: 只能用document调用

4. 按class属性查找:

var elems=parent.getElementsByClassName(“class”)

返回值: 多个元素的集合，如果找不到返回[]

强调: 1. 可在任意父元素上调用，控制查找范围

2. 不仅查找直接子元素，且在所有后代中查找

3. 如果一个元素有多个class修饰，则只需要其中一个class就可找到该元素。

问题: 一次只能用一个条件查找

如果查找条件复杂，代码会很繁琐！

1. 用选择器查找元素: 2个:

何时: 当查找条件复杂时，首选按选择器查找

1. 仅查找一个元素:

var elem=parent.querySelector(“选择器”)

1. 查找多个元素:

var elems=parent.querySelectorAll(“选择器”)

按HTML查找 vs 按选择器查找:

1. 返回值:

按HTML查找返回动态集合: 不实际存储属性值，每次访问集合都重新查找DOM树

按选择器查找返回非动态集合: 实际存储完整属性值，每次访问集合不需要重复查找DOM树

1. 首次查找效率:

按HTML查找，效率高！

按选择器查找，效率低

1. 易用性:

按HTML查找功能单一

按选择器查找，功能强大，代码简洁

总结: 查找优化

1. 如果用一个条件就可找到元素时，首选按HTML查找
2. 如果查找条件复杂时，首选按选择器查找

2. 修改:

内容: 3种:

1. 原始HTML片段:

elem.innerHTML 可获取或设置元素开始标签到结束标签之间的HTML片段

1. 纯文本内容:

elem.textContent 可获取或设置元素开始标签到结束标签之间的纯文本内容

比innerHTML多做两件事:

1. 去掉内嵌的标签
2. 特殊符号翻译为正文
3. 表单元素的值: elem.value

属性: 3种:

1. HTML标准属性: HTML标准中规定的属性

2种:

1. 核心DOM API: DOM最初指定的可操作一切结构化文档(HTML和XML)的API: 可操作任何出现在元素开始标签中的属性

4个API:

elem.getAttribute(“属性名”)

elem.setAttribute(“属性名”,”值”)

elem.removeAttribute(“属性名”,”值”)

elem.hasAttribute(“属性名”)

2. HTML DOM: 专门针对HTML中固定的属性，提供的简化版API

HTML DOM提前将HTML规定的标准属性都定义在了元素对象上，可用.直接访问

elem.属性

冲突: class属性: elem.className

1. 状态属性: 4个:

enabled disabled selected checked 值都是bool

不能用核心DOM 4个API

只能用HTML DOM打.访问

1. 自定义扩展属性:

什么是: HTML标准中没有规定的，程序员自己添加的属性

何时: 2种:

1. 可用于在客户端临时缓存业务数据:
2. 代替id,class,元素等选择器用于选择元素并绑定事件

样式:

#### DAY03 修改对象

正课:

1. 修改
2. 添加/删除
3. HTML DOM对象
4. 修改:

属性:

自定义扩展属性:

如何: 2步:

1. 定义自定义扩展属性:

<ANY data-属性名=”值” … >

1. 获取或修改自定义扩展属性: 2种:
2. 用核心DOM修改

elem.getAttribute(“data-属性名”)

elem.setAttribute(“data-属性名”,”值”)

elem.removeAttribute(“data-属性名”)

elem.hasAttribute(“data-属性名”)

不能用HTML DOM访问

1. HTML5: 自动将所有data-开头的自定义扩展属性，收集到elem.dataset属性中

elem.dataset.属性名 属性名不要加data-前缀

用定义扩展属性查找元素: 属性选择器:[data-属性名=值]

样式:

修改: elem.style.css属性=”值”

强调: 相当于<ANY style=”css属性:值”

只能修改内联样式属性

问题: 一次只能修改一个css，代码繁琐

解决: 尽量用class，批量修改元素的样式

elem.className=”class名”

elem.className=””

获取: 错误: elem.style.css属性 仅能获得内联样式

正确: 获得计算后的样式: 2步:

1. 获得计算后的所有样式的对象:

var style=getComputedStyle(elem)

1. 获得style中的属性: style.css属性

强调: 计算后的样式，是只读的，不能修改！

因为来源不确定

2. 添加/删除元素:

添加元素: 3步:

1. 创建新的空元素:

var a=document.createElement(“a”)

<a></a>

1. 设置关键属性:

a.innerHTML=”go to tmooc”

a.href=<http://tmooc.cn>

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

1. 将新元素挂到DOM树上指定父元素下:

3种:

parent.appendChild(a) 将a追加到parent的所有直接子元素之后

parent.insertBefore(a,child) 将a插入到parent的子元素child之前

parent.replaceBefore(a,child) 用a代替parent的子元素child的位置

优化: 尽量减少操作DOM树的次数

为什么:每修改一次DOM树，都会导致重排重绘

HTML加载过程:

html -> DOM Tree

↓

RenderTree -> layout -> paint

↑

css -> COM

解决: 2种:

1. 如果同时添加父元素和多个子元素，应该先在内存中，将所有子元素添加到父元素后，再最后一次性将父元素添加到DOM树
2. 如果父元素已经在页面上了，要添加多个平级子元素时。应该使用文档片段：

文档片段: 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. 创建文档片段对象:

var frag=document.createDocumentFragment();

1. 将多个子元素临时加入到文档片段中:

frag.appendChild(child)

1. 将frag整体加入到DOM树的父元素下

parent.appendChild(frag)

删除: parent.removeChild(child)

让父元素，删除其下的一个子元素child

3. HTML DOM常用对象:

Image : 代表页面上一个<img>元素

简化: 创建: var img=new Image();

强调: 特例！其它元素，不能new!

Select: 代表页面上一个<select>元素:

属性: value 可获得当前选中项的值

如果选中的option，没有value属性，则用内容代替

selectedIndex 可获得选中项在select中的下标位置

options 可获得select下所有option对象

options.length 可获得select下所有option的个数

length = options.length

方法: .add(option) 代替: .appendChild()

问题: .add一次只能添加一个option，不支持frag

.remove(i) 移除i位置的option

Option: 代表页面上一个<option>元素

创建: var opt=new Option(text,value)

代替: .createElement(“option”)

.innerHTML=text;

.value=value;

Table:代表着页面上一个<table>元素

直接管着行分组:

添加行分组: var thead=table.createTHead()

相当于: .createElement(“thead”)

.appendChild(thead)

var tbody=table.createTBody();

var tfoot=table.createTFoot();

删除行分组: table.deleteTHead()

table.deleteTFoot()

获取行分组: table.tHead

table.tFoot

table.tBodies[i]

行分组对象: 代表页面上table下一个行分组元素

管着行:

添加新行: var tr=行分组.insertRow(i)

固定套路: tr.insertRow() 末尾追加！

tr.insertRow(0) 开头插入!

删除行:

获取行:

#### DAY04 bom

正课:

1. HTML DOM常用对象
2. 什么是BOM
3. Window对象
4. 打开和关闭窗口
5. History
6. Location
7. Navigator
8. 定时器
9. HTML DOM常用对象:

Table

行分组管着行:

添加行:

删除行: 行分组.deleteRow(i)

强调: i 是行在行分组内的相对下标位置

矛盾:   
行分组.deleteRow(i) i 要求是行分组内的相对下标位置

每个行上都有一个rowIndex属性，记录着行在整个表中的下标位置

所以: 行分组.deleteRow(tr.rowIndex)

结果会删除tr的下一行

解决: 今后删除行，一定不要用行分组作为.前的主语

应该用table.deleteRow(tr.rowIndex)

获取行: 行分组.rows[i]

tr管着格:

添加格: var td=tr.insertCell(i)

通常都是末尾追加: tr.insertCell()

删除格: tr.deleteCell(i)

获取格: tr.cells[i]

Form: 代表页面上一个<form>元素

获得表单对象: var form=document.forms[i]

属性: .elements 获得表单中所有表单元素的集合

.elements.length 获得表单中表单元素的个数

.length = .elements.length

方法: form.submit() 手动提交表单

表单元素:

获取: form.elements[i/name]

更简化: 如果表单元素有name属性，则可直接写:

form.name

方法: elem.focus() 自动获得焦点

elem.blur() 自动失去焦点

1. 什么是BOM: Browser Object Model

专门操作浏览器软件的API

问题: 没有标准！很多API已经废弃

包括:

window: 是包含所有ES,DOM,BOM原生API的全局对象

history: 掌管历史记录

location: 掌管地址栏

document: 掌管所有网页内容: DOM

navigator: 掌管浏览器的配置信息

screen: 掌管显示设备的信息

event: 掌管着事件相关操作

1. Window:

2个角色:

1. 代替ES中的global充当全局作用域对象
2. 封装所有ES,DOM和BOM的原生API

属性: 获得浏览器窗口大小: 2组:

1. 完整窗口大小: window.outerWidth/outerHeight

用途: outerHeight<指定值, 可判断窗口是否最小化了

1. 文档显示区大小: window.innerWidth/innerHeight

文档显示区：浏览器窗口中，专门用于显示网页内容的区域

4. 打开和关闭窗口:

API: window.open()

window.close()

打开新链接: 4种:

1. 在当前窗口打开，可后退

Html: <a href=”url” target=”\_self”>

Js: open(“url”,”\_self”)

1. 在当前窗口打开，禁止后退

Js: location.replace(“新url”)

原理: 用新url代替history中现有的url

1. 在新窗口打开，可打开多个

Html: <a href=”url” target=”\_blank”>

Js: open(”url”, ”\_blank”)

1. 在新窗口打开，只能打开一个

Html: <a href=”url” target=”自定义窗口名”>

Js: open(“url”,”自定义窗口名”)

其实, <a>的target属性是新窗口的名字(name)

name是在内存中唯一标示一个窗口的名称

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，会覆盖旧的窗口

预定义窗口名:

\_self: 自动获得当前窗口自己的名字给新窗口

\_blank: 不指定窗口名，让浏览器随机生成窗口名

5. history:

什么是: 保存当前窗口打开后，成功访问过的url的历史记录栈(数组)

控制着: 前进后退

如何: history.go(n)

history.go(1) 前进一步

history.go(-1) 后退一步

history.go(-2) 后退2步

history.go(0) 刷新

6. location:

什么是: 专门保存当前窗口正在打开的url

何时:1. 分段获得url中部分内容时

2. 页面跳转/刷新时

如何:

属性: .href 完整的url

.protocol 协议 http: https: ftp: …

.host 主机名+端口号

.hostname 主机名

.port 端口号

.pathname 相对路径

.hash #锚点地址

.search ?查询字符串

方法:

1. 在当前窗口打开，可后退:

location.href=”新url”

其实可简写为: location=”新url”

1. 在当前窗口打开，禁止后退:

location.replace(“新url”)

1. 刷新页面:
2. 普通刷新: 优先从浏览器本地缓存中获取资源，本地缓存中没有或过期，才被迫重新从服务器下载。

history.go(0)

location.reload()

F5

1. 强制刷新:

location.reload(true) 强迫: 始终从服务器下载新资源

7. navigator:

什么是: 专门保存浏览器配置信息的对象

包括:

1. .cookieEnabled: 判断是否启用了cookie

什么是cookie: 客户端本次持久保存用户私密信息的小文件

为什么: 程序内存中的数据，都是临时的。

何时: 只要希望持久保存的数据，都应该放在cookie中

典型: 记住密码

1. .plugins: 包含所有插件的集合

何时: 判断是否安装某个插件:

如何: if(navigator.plugins[“插件名”]===undefined)

说明没装

else

说明装了

1. .userAgent: 保存浏览器名称和版本号的字符串

何时: 专门用于判断浏览器的名称和版本号

8. 定时器:

2种:

1. 周期性定时器: setInterval()

什么是: 让程序每隔一段时间间隔，反复执行一项任务

如何:

启动: var n=setInterval(function(){

任务

}, 间隔ms)

停止: clearInterval(n) //n是内存中定时器的唯一序号

1. 一次性定时器: setTimeout()

什么是: 让程序延迟执行一项任务，执行后自动停止定时器。

如何:

启动: var n=setTimeout(function(){

任务

}, 延迟ms)

停止: clearTimeout (n)

#### DAY05 事件

正课:

1. 事件:
2. 事件:

什么是: 浏览器自动触发的，或者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: 在事件发生时，自动指定的回调函数

何时: 如果希望在发生事件时，执行一项任务

如何: 事件处理函数都要先绑定到元素上：3种:

1. 在HTML中: <ANY on事件名=”js语句”

问题: 不符合内容与行为分离的原则，不便于维护

1. 在js中,用赋值方式: elem.on事件名=function(){

this->当前触发事件的.前的元素

}

当事件发生时: elem.onclick()

问题: 一个事件只能绑定一个处理函数

1. 在js中，添加事件监听对象:

elem.addEventListener(“事件名”,function(){

… …

})

elem.removeEventListener(“事件名”,原处理函数)

强调: 1. 如果一个处理函数可能被移除，绑定时就不能用匿名函数，应该用具名函数

2. 移除时，必须通过函数名找到原函数对象

事件模型: DOM规定: 3个阶段

1. 捕获: 由外向内，依次记录各级父元素上绑定的处理函数
2. 目标触发: 优先触发目标元素上的处理函数

目标元素: 最初实际触发事件的那一个元素

1. 冒泡: 由内向外，按捕获的记录，反向执行各级父元素上绑定的处理函数

所有浏览器的开发者一致认为，点在内层元素上，等效于也点在外层元素上

事件对象:

什么是: 事件发生时，自动创建的保存事件信息的对象

何时: 1. 想获得事件信息时

2. 想改变事件的默认行为时

如何:

获取事件对象: 事件对象总是作为处理函数的第一个参数自动传入处理函数.

elem.onclick = function(e) { e->事件对象}

↑

当事件发生时: elem.onclick(新建event对象)

API:

1. 取消冒泡: e.stopPropagation()
2. 利用冒泡:

优化: 尽量减少事件监听对象的个数！

因为: 浏览器触发事件处理函数是采用遍历的方式查找监听对象。如果监听对象多，遍历慢

解决: 当多个子元素，需要绑定相同的事件时，其实只在父元素绑定一次即可！所有子元素通过冒泡共用！

2个难题:

1. 获得目标元素，而不是父元素

错误: this->父元素

正确: e.target

1. 判断目标元素是否想要的

在处理函数中先用if判断e.target是否想要的

只有是想要的时，才执行操作

3. 阻止默认行为:

何时: 只要元素自带的默认行为不是想要的，就可阻止

如何: e.preventDefault();

常用:

1. 用a当按钮时: 阻止a自动添加#xxx到地址栏
2. 其实submit按钮也能阻止提交！

input type=submit 触发 form.onsubmit=function(e){

如果验证不通过

e.preventDefault()

}

4. 鼠标位置: 3组:

1. 相对于屏幕左上角的坐标: e.screenX e.screenY

2. 相对于文档显示区左上角的坐标: e.clientX e.clientY

3. 相对于当前元素左上角的坐标: e.offsetX e.offsetY

滚动事件:

window.onscroll=function(){

var scrollTop=document.body.scrollTop

||document.documentElement.scrollTop

}