# 第一阶段

标准SQL语句分类：

1.DDL：data define language 定义数据结构语言

CREATE /DROP /ALTER

2.DML: data manipulate language 操作数据

INSERT/DELETE/UPDATE

3.DQL:data query language 查询数据

SELECT

4.DCL:data control language 控制用户权限

GRANT/REVOKE

## MySQL

### day02

1. 常用SQL语句
2. 删除数据

DELETE FROM user WHERE

1. 修改数据

UPDATE user SET uid WHERE

1. 计算机存储字符
2. 如何存储英文字符

ASCII:128个 所有的英文字母和符号进行编码

latin-1:256个，兼容ASCII

1. 如何存储中文字符

GB231: 6千多汉字进行编码，兼容ASCII

GBK:两万多汉字进行编码，兼容ASCII

BIG5:台湾繁体字编码，兼容ASCII

Unicode：对世界上主流国家的主流语言进行编码，兼容ASCII，不兼容GBK,BIG5，具体分为UTF-8,UTF-16,UTF-32三中存储方法。

1. 解决mysql中文乱码

MySQL默认latin-1

脚本另存为的编码为UTF8

客户端连接服务器的编码（SET NAMES UTF8）

服务器端创建服务器的编码（CHARSET=UTF8）

1. MySQL中的列类型

创建表时，指定的列的数据类型

1. 数值型——引号可加可不加

BYTE BIT

TINYINT 微整型，占1字节 -128~127

SMALLINT 小整型，占2字节 -32768~32767

INT 整型，占4字节

BIGINT 大整型，占8字节

FLOAT单精度浮点型，占4字节，最大3.4e38，可能产生计算误差

DOUBLE 双精度浮点型，占8字节，可能产生误差

DECIMAL(M,D) 定点小数，小数点不会变化，几乎不会产生计算误差 M总的有效位，D小数点后的有效位数

BOOL 布尔型，只有两个值TRUE/FALSE 在存储时，变成TINYINT(1/0)

1. 时间日期型——引号必须加

DATE 日期型 ‘2019-11-04’

TIME 时间型 ‘14:40:30’

DATETIME 日期时间型 ‘2019-11-04 14:40:30’

1. 字符串型——引号必须加

VARCHAR(M) 变长字符串，不会产生空间浪费，操作速度比较慢，M的最大值65535

CHAR(M) 定长字符串，可能产生空间浪费，操作速度比较快，M的最大值255 手机号码，身份证号固定长度的数据

TEXT(M) 大型变长字符串，M最大为2个G

### day03

1. 列约束

可以对要插入的数据进行特定验证，只有满足条件才能插入，否则被认为非法的插入

1. 主键约束——PRIMARY KEY

声明了主键列上的值不予重复，一个表只能有一个主键，查询时按主键编号排序，加快查询速度

NULL 表示空，在插入的时候，无法确定的数据

主键约束中禁止插入null

1. 非空约束——NOT NULL

声明了非空约束的列上不能插入NULL

1. 唯一约束——UNIQUE

声明了唯一约束的列上不能插入重复的值，允许插入NULL，而且允许插入多个NULL

1. 检查约束——CHECK

可以对插入的数据进行自定义的验证

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

mysql不支持检查约束，会降低数据的插入速度，造成服务器端的压力

1. 默认值约束——DEFAULT

可以使用DEFAULT关键字声明默认值，

INSERT INTO laptop VALUES(,,,DEFAULT);

INSERT INTO laptop(lid) VALUES(10);

1. 外键约束——FOREIGN KEY

两者的数据类型一致

FOREIGN KEY(familyId) REFERENCES laptop\_family(fid)

1. 自增列

AUTO\_INCREMENT:自动增长，直接赋值为NULL，或获取当前的最大值

注意：

只适用于整数型的主键列上，允许手动赋值

1. 简单查询
2. 查询特定的列

查询所有员工的编号，姓名，性别，生日

SELECT eid,ename,sex,birthday FROM emp;

1. 查询所有的列

SELECT \* FROM emp;

1. 给列起别名

查询所有员工的姓名和工资，使用汉字别名

SELECT ename AS 姓名,salary AS 工资 FROM emp;

查询所有员工的编号，姓名，性别，生日，使用汉字别名

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

查询所有员工编号和姓名，使用1个字母别名

SELECT eid id,ename name FROM emp;

1. 显示不同的记录/合并相同的记录

查询出都有那些性别的员工

SELECT DISTINCT sex FROM emp;

查询员工都分布在那些部门

SELECT DISTINCT deptId FROM emp;

1. 在查询时执行计算

查询出所有员工的姓名及其年薪

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

假设每个员工的工资增加500，年中奖6000，查询出所有员工的姓名及其年薪

SELECT ename 姓名,(salary+500)\*12+6000 年薪 FROM emp;

1. 查询结果集排序

查询所有的部门，结果集按照部门编号升序排列

SELECT \* FROM dept ORDER BY did ASC;

查询所有的部门，结果集按照部门编号降序排列

SELECT \* FROM dept ORDER BY did DESC;

查询所有的员工，结果集按照工资降序排列

SELECT \* FROM emp ORDER BY salary DESC;

查询所有的员工，年龄从小到大排序

SELECT \* FROM emp ORDER BY birthday DESC;

查询所有的员工，结果集按照姓名升序排列

SELECT \* FROM emp ORDER BY ename;

查询所有的员工，结果集按照工资降序排列，如果工资相同，按姓名排序

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

查询所有的员工，结果集按照性别排序，如果性别相同，按照年龄从小到大排序

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

1. 条件查询

查询出编号为5的员工数据

SELECT \* FROM emp WHERE eid=5;

查询出姓名叫king的员工的编号，姓名，工资

SELECT eid,ename,salary FROM emp WHERE ename=’King’;

查询出20号部门所有的员工

SELECT \* FROM emp WHERE deptId=’20’;

查询出所有的女员工

SELECT \* FROM emp WHERE sex=0;

比较运算符 ：> < >= <= = !=

查询出1991-1-1后出生的员工

SELECT \* FROM emp WHERE birthday>’1991-1-1’;

查询出不在10号部门的员工

SELECT \* FROM emp WHERE deptId!=10;

查询出没有明确部门的员工

SELECT \* FROM emp WHERE deptId IS NULL;

查询出有明确的部门的员工

SELECT \* FROM emp WHERE deptId IS NOT NULL;

查询出工资6000以上的男员工

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

查询出工资在5000-7000的员工

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

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

查询出工资在5000以下和7000以上的员工

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

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

查询出1993年出生的员工

SELECT \* FROM emp WHERE birthday>=’1993-1-1’ AND birthday<=’1993-12-31’;

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

查询出20号部门和30号部门员工

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

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

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

1. 模糊条件查询

查询姓名中以e结尾的员工

SELECT \* FROM emp WHERE ename LIKE ‘%e’;

查询出倒数第2个字符为e的员工

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

% 匹配任意多个字符 >=0

-匹配一个字符 =1

必须合着LIKE 关键字使用

1. 分页查询

查询的结果集有太多的数据，一次显示不完，可以分页显示

每页的数据量

当前页码

SELECT \* FROM emp LIMIT start,count;

start:(当前页码-1)\*每页数据量

课后任务：

查询出工资在8000以上的女员工的姓名、性别、生日，工资，结果集按照工资降序排列，显示前两个人。

SELECT ename,sex,birthday,salary FROM emp

WHERE salary>8000 AND sex=0

ORDER BY salary DESC

LIMIT 0,2;

## JS

### day01

1. 复杂查询
2. 聚合查询/分组查询

|  |
| --- |
| 聚合函数  函数是一个功能体，提供若干个数据，产出特定结果  COUNT()/SUM()/AVG()/MAX()/MIN() |

分别使用部门编号和员工编号获取数量

SELECT COUNT(deptId) FROM emp;

SELECT COUNT(eid) FROM emp;

查询出所有男员工的数量

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

查询所有员工的工资总和

SELECT SUM(salary) FROM emp;

查询出男员工的平均工资

SELECT AVG(salary) FROM emp WHERE sex=1;

SELECT SUM(salary)/COUNT(\*) FROM emp WHERE sex=1;

查询出工资最高的员工

SELECT MAX(salary) FROM emp;

查询出年龄最大的员工

SELECT MIN(birthday) FROM emp;

|  |
| --- |
| 分组查询：只能查询分组条件和聚合函数 |

分别查询出男女员工的数量

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

查询出1991年出生的员工

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

查询出3月份生日的员工

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

1. 子查询

查询出比tom工资高的员工

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

查询出和同一年出生的员工

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

1. 多表查询

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

|  |
| --- |
| 上述语法无法查询出没有部门的员工，无法查询出没有员工的部门 |

1. 内连接

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

1. 左外连接

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

1. 右外连接——显示右侧表中所有记录

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

1. 全连接

显示左侧表和右侧表中所有记录——FULL JOIN

MySQL不支持全连接

UNION 合并相同记录

UNION ALL 不合并相同记录

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

UNION ALL

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

(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);

1. 学习编程语言的步骤：
2. 了解背景知识、历史、当前、特点、应用场景
3. 搭建开发环境，编写hello world
4. 常量和变量
5. 数据类型
6. 运算符
7. 逻辑结构
8. 通用的小程序
9. 函数和对象
10. 第三方库和框架
11. 个人项目

程序员必做50题

JAVASCRIPT高级程序设计第三版

JavaScript 学习指南

1. JS概述
2. 历史

1995年，JS最早出现在Netscape公司的浏览器中

1996年，IE3中也出现了JS，称为JScript

1997年，ECMA组织制定了JS的标准ECMAScript，简称EC

2009年，遵循了CommonJS规范，开始向服务器发展，出现了NodeJS

1. 现状

既可以运行在客户端浏览器，可以运行在服务器端

1. 特点

解释性语言，编译一行执行一行

弱类型语言

基于对象

跨平台

1. 应用场景

开发浏览器端的交互效果

创建WEB服务器，数据库操作等

1. JS开发环境
2. 浏览器自带的JS解释器

谷歌 火狐 欧朋 IE Safari

1. 服务器端使用NodeJS解释器 谷歌，v8引擎

<https://nodejs.org> 下载地址

node –v 查看当前nodejs的版本号

1. 运行JS代码

浏览器

创建01.js和01.html，把01.js嵌入到01.html中

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

NodeJS

node c:/xampp/.../01.js 回车

1. JS语法规范

区分大小写

每行代码结尾的分号可加可不加

分为单行（//）和多行（/\* \*/）注释

1. 变量

存储数据的容器

1. 声明变量——var
2. 变量的命名规则

字母、数字、下划线、美元符，其中不能以数字为开头，不能使用关键字和保留字

1. 变量赋值

var a;

变量声明未赋值，此时的值为undefined

可以为变量多次赋值，并且赋不同类型的值

1. 一次声明多个变量

var a=1,b=9,c;

1. 常量

一旦声明，不允许重新赋值，必须设置初始值

整理思维导图

声明变量保存圆的半径，声明常量保存圆周率，声明两个变量分别保存计算的面积和周长。

准备自己项目的数据库

### day02

1. 数据类型

原始类型和引用类型

原始类型：数值型、字符串型、布尔型、未定义型、空

引用类型：Object、Array、function、Date，包装类型（String、Number、Boolean）,内置对象（Global、Math）

值类型+引用类型(ES+DOM/BOM+自定义)

1. 数值型——number

整型和浮点型

8进制 以0开头的数字 011-9

16进制 以0x开头的数字 0XF-15

0XF0 0X3F 0X40 0XFF 0X100

|  |
| --- |
| typeof 检测数据类型（原始类型）number，string，undefined，boolean |

1. 字符串型——string

被引号所包含的内容都是字符串型，不区分单双引号

查看任意一个字符的unicode码：'h'.charCodeAt()

1. 布尔型——boolean

true/false

在程序中表示真或假

常用于一些是否的结果，是否登录，是否注册，一些运算符结果返回布尔型

1. 未定义型——undefined

声明了变量未赋值返回undefined

1. 空——object（引用类型）

只有一个值null

1. 数据类型转换
2. 隐式转换

在运算过程中自动发生的数据类型转换

* 数值+字符串 数值转为字符串 +表示拼接
* 数值+布尔型 布尔转为数值 true-1，false-0
* 字符串+布尔型 布尔转为字符串

|  |
| --- |
| JS中加号作用：  ①执行加法运算  ②执行字符串拼接，前提是加号的两端有一个数据是字符串 |

|  |
| --- |
| 减法、乘法、除法的隐式转换  如果运算符的两端不是数值，则自动调用Number函数转为数值 |

1. 强制转换

* 强制转为数值型

Number('2') //2

Number('2a') //NaN

Number(true) //1

Number(undefined) //NaN

Number(null) //0 '',' '

NaN:not a number 不是一个数字

* 强制转为整型

parseInt()

parseInt(3.14) //3

parseInt('3.14a') //3

parseInt(true) //NaN

parseInt('a3.14') //NaN

将小数和字符串转为整型，如果要转换的数据中含有非数字，则返回NaN

* 强制转为浮点型

parseFloat()

parseFloat('3.14a') //3.14

parseFloat('a3.14') //NaN

和parseInt()一样，只是转的数据为浮点型

* 将数值和布尔型强制转字符串

toString()

var num=3;

var str=num.toString(); //'3'

1. 运算符

表达式：由运算符连接的操作数据所组成的形式

算数运算符、比较运算符、逻辑运算符、位运算符、赋值运算符、三目元运算符

1. 算术运算符

+ - \* / % ++ --

% 取余

++ 自增

-- 自减

console.log(a++) 先打印a的值，然后自增

console.log(++a) 先自增，然后打印a的值

var num=3;

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

console.log(num); //1

1. 比较运算符

* < >= <= ==（等于） != ===(全等于) !==

== 比较两个值是否相同

=== 比较两个值和类型是否相同

3>'10' 字符串转为数值

'3'>'10a' 比较首个字符的unicode码

3>'10a' '10a' 转为NaN，和任何值比较返回NaN

NaN==NaN //NaN

1. 逻辑运算符

或者|| ：关联的一个条件为true结果就是true

并且&&：关联的两个条件为true结果就是true

非 ！：取反

**短路逻辑**

&& 当第一个条件为false的时候，就不在执行第二个条件

|| 当第一个条件为true的时候，就不在执行第二个条件

说明：逻辑短路无需关注最终的结果是true/false,重点是看你第二个条件是否会执行

### day03

1. 位运算符

1 10 11 100 101 110 111 1000 1001

100110 32+4+2=38 100000 100 10 100110

25=16+8+1 11001

按位与& 上下两位都为1，结果是1，否则是0

7&12 111 1100 0100-4

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

7^12 1011-11

按位右移 >> 从最后位删除 48=32+16 110000 48>>1-11000 24 大概是原来的1/2

按位左移 << 在二进制的最后补0 成倍增加

1. 赋值运算符

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

+= 初始值为字符串，拼接

1. 三目运算符

一目运算符：一个操作数据和

二目运算符：两个操作数据和一个运算符

三目运算符：三个操作数据和两个运算符

条件表达式？表达式1：表达式2

1. 浏览器函数

alert（） 弹出警示框

prompt() 弹出提示框，返回string类型 点取消返回null

1. 流程控制——选择执行
2. if语句

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

false:0 NaN '' null undefined

1. if else 语句

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

1. if else嵌套

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

1. switch case

一项特殊的多项分支语句，

|  |
| --- |
| switch(表达式)  {  case 值1:  语句1;  break;  ...  default:  语句 n+1;  } |

表达式和case值比较是全等于

### day04

1. 循环执行

一遍又一遍执行相同或相似的代码

两要素：

循环条件：控制循环次数

循环体：重复执行的相同或相似的代码

1. while

|  |
| --- |
| while(条件表达式)  {  循环体；  } |

1. break和continue

break：结束循环

continue：结束本次循环，继续下次循环

1. do while

|  |
| --- |
| do  {  循环体；  }while(条件表达式) |

1. for

|  |
| --- |
| for(初始值;条件;增量)  {  循环体;  } |

多个循环条件，最后一个起作用

### day05

1. 函数

系统函数与自定义函数

function：功能体，函数，方法。接收若干个数据，返回处理的结果。——用于封装多次执行的代码。

1. 创建普通函数

|  |
| --- |
| function 函数名称(){  函数体  } |

调用

函数名称();

1. 创建带有参数的函数

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

创建函数时参数是形参，调用函数时的参数是实参

如果形参未赋值，返回undefined

1. 带有返回值函数

|  |
| --- |
| function 函数名称(参数列表){  函数体;  return 值; //函数调用后产出的结果  } |

return 返回函数值，没有return或return没加值，返回undefined；结束函数，不会执行后续代码

1. 变量作用域

变量和函数的作用范围，分为

全局作用域：在函数外使用var声明的变量，可以在任何位置访问到

函数作用域：在函数内使用var声明的变量，只能在函数内访问到

变量声明提升：（JS）

var声明的变量提升到所在作用域的最前面，赋值还是原来的位置

1. 函数作用域

函数的可访问范围，分为全局函数和局部函数

和变量提升一样，使用function创建的函数会提升到所在作用域的最前边

1. 递归

在函数内部调用自身，本身也是一个无线循环

用法：要结合return使用

### day06

1. 匿名函数

没有名称的函数

function (){ }

1. 创建函数

|  |
| --- |
| 函数声明  function fn(){}  函数表达式  var fun=function (){}  变量名称就是函数名称 |

函数名称就是一个变量，保存一个函数的结构

|  |
| --- |
| 函数表达式创建的函数不存在变量提升，使用时需要创建在调用 |

1. 匿名函数自调用

目的：创建函数作用域，防止污染全局

|  |
| --- |
| function (形参){  }(实参); |

1. 回调函数

将匿名函数以实参的形式传递

|  |
| --- |
| function fn(a){  //var a;  a();  }  fn(function (){}) |

1. 全局函数

Number()/parseInt()/parseFloat()

encodeURI() //编码URI

decodeURI() //解码URI

isNaN() //检测一个数值是否为NaN

isFinite() //检测是否为有限值

eval() //执行字符串中的表达式

1. 对象

是一组属性和方法的集合

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划分的

自定义对象：用户自己创建的对象

1. 自定义对象

对象字面量

内置构造函数

自定义构造函数

1. 使用对象字面量创建对象

|  |
| --- |
| var person={  name:'tom',  'user@pwd':'123456'  } |

1. 访问对象中的属性

对象,属性名

对象.['对象名']

对象不存在，返回undefined

1. 内置构造函数创建对象

new Object()

需要访问对象添加属性

1. 遍历对象中的属性

for in

|  |
| --- |
| for(var key in d对象){  对象[key]  } |

1. 对象中的方法

|  |
| --- |
| var peron={  name:'tom',  say:function (){  this.name  }  } |

### day07

1. 检测属性是否存在

|  |
| --- |
| person.birthdar===undefined  对象.hasOwnProperty('属性名')  '属性名' in 对象 |

1. 引用类型存储

原始类型——栈内存 值拷贝

引用类型——数据堆内存，地址在栈内存 地址拷贝

变量地址赋为null，释放堆内存空间

1. 数组

多个元素组成的集合，每个元素就是一条数据

1. 创建数组

数组字面量

|  |
| --- |
| var arr=[空/元素]; |

内置构造函数

var arr=new Array(空/元素/初始化长度);

1. 访问数组中元素

数组[下标] 下标-0

1. 获取数组长度

数组.length

1. 数组分类

索引数组：以0及以上的整数做下标

关联数组：以字符串做下标，只能单独添加每个元素

1. 遍历数组元素

for-in

|  |
| --- |
| for(var key in 数组){  数组[key]  }  既可以遍历索引数组，也可以遍历关联数组 |

循环

|  |
| --- |
| for(var i=1;i<数组.length;i++){  数组[i]  }  只能遍历关联数组 |

1. 数组API

tostring/join/concat/slice

reverse() 翻转数组元素

sort() 排序，默认是按照unicode码从小到大排序

|  |
| --- |
| sotr(function (a,b){  return a-b; //小-大  return b-a; //大-小  }) |

splice(start,count,value) //删除数组中元素 start可为负数

push() //数组末尾添加元素，返回数组个数

pop() //数组末尾删除元素

unshift() //数组开头添加元素

shift() //数组开头删除元素

### day08

1. 二维数组

数组中每个元素还是数组

|  |
| --- |
| 数组[[],[],...] |

1. 字符串

包装对象：原始类型向对象一样，具有属性和方法

string number boolean

将任意数据转为字符串

new String() 返回对象

String() 返回字符串

1. 转义字符

\' 将引号转成普通字符

\n 换行符

\t 制表符

1. 字符串API

length

charAt() //获取下标对应的字符

charCodeAt() //某个字符的unicode码

indexOf() //查找某个字符的下标

lastIndexOf() //查找某个字符最后一次出现的下标 找不到返回-1

toUpperCase() //转大写

toLowerCase() //转小写

slice(start,end) //截取字符串 不包含end

substr(start,count) //截取字符串

substring(start,end) //截取字符串 start不能为负 end可以小于start

split(sep) //将字符串按照指定的字符分割数组

1. 匹配模式

search() 查找满足条件的第一个 返回下标，否则-1 i ignore/global

match() 查找满足条件所有的 返回数组 ig

replace() 查找并替换 返回数组 ig

1. Math对象——直接引用

PI 圆周率

abs() 绝对值

round() 四舍五入

ceil() 向上取整

floor() 向下取整

max() 最大值

min() 最小值

random() 随机数 0~1 不包含1

### day09

1. Date对象

用于日期时间的存储与计算

1. 创建date对象

new Date('2019/11/18 10:45:41')

new Date(2019,10,18) 月份0-11

new Date(6000) 距离计算机元年毫秒数对应的日期时间

new Date() 存储当前操作系统时间

1. 获取Date对象中存储的日期时间

getFullYear()

getMonth()

getDate()

getHours/Minutes/Seconds/Milliseconds

getDay() 星期 0-6

getTime()

1. 本地日期——2019-11-18 14:36:22

toLocalString()

toLocalDateString()

toLocalTimeString()

1. 设置日期时间

setFullYear()

setMonth() 0~11

setDate()

setHours()/Minutes()/Sencons()/Milliseconds()

setTime() 所有时间都会发生比变化

1. 对象赋值

var d1=new Date();

var d2=new Date(d1);

1. Number对象

new Number()/Number()

toFixed(n) 获取小数点后n位

toString(n) 设置进制

1. Boolean 对象

new Boolean()

Boolean();

!值

1. 错误处理

语法错误

引用错误

类型错误

范围错误

自定义错误：throw '错误提示'

try{

//错误信息

}catch(err){

//处理错误的方案

}

1. ES6

ECMAScript

<http://es6.ruanyifeng.com/> 《ES6标准入门》

1. 块级作用域

使用let关键字声明的变量，不存在变量提升，不能多次赋值，只能在块级作用域使用

|  |
| --- |
| 使用大括号，例如 if else while for awitch ... |

可以给参数设置默认值

|  |
| --- |
| function add(a,b,c){  //b=b||0 ES6之前  return a+b+c  } |

1. 箭头函数

匿名函数的另一种写法，但不等于匿名函数

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

1. 模板字符串

解决了字符串拼接，如果需要使用JS表达式借助${}

|  |
| --- |
| `字符串 ${}` |

## Node.js

### day01

1. Node.js概述

基于谷歌V8 引擎，运行在服务器端的JS解释器

数据类型：值类型+引用类型(ES+扩展对象+自定义)

1. 对比JS和Node.js

JS运行在浏览器端，多个环境；一个环境、

内置对象，自定义对象、宿主对象（不同）

JS浏览器端的交互效果，Node.js服务器端操作

[www.nodejs.org](http://www.nodejs.org) [www.nodejs.cn](http://www.nodejs.cn)

交互模式

node 回车

退出：两次ctrl+c 或 .exit

适用于：交互、I/O密集型

1. 全局对象
2. node.js

交互模式下声明的变量和函数，都在global下；而脚本模式不是的

1. JS

浏览器下，都是全局的，使用window访问

1. console对象

console.log() 打印日志

console.info() 打印消息

console.warn() 打印警告

console.error() 打印错误

console.time() 开始计时

console.timeEnd() 结束计时

1. process对象

arch 当前cpu框架

platform 操作系统

version 版本

env 环境变量

kill() 结束进程

pid 查看进程

1. Buffer对象

var buf=Buffer.alloc(6,'abcdef')

一个汉子占三个字符

转为字符串：String()/toString()

1. 定时器——时间队列
2. 一次性定时器

let timer=setTimeout(回调函数，间隔时间)

clearTimeout(timer)

1. 周期性定时器

let timer=setInterval(回调函数，间隔时间)

clearInterval(timer)

1. 立即执行定时器

setImmediate(回调函数)

clearImmediate()

process.nextTick(回调函数)

1. 模块系统

分为三类模块：自定义模块、核心模块、第三方模块

一个文件就是一个模块

node.js自动为每个文件添加一个构造函数，所写的代码被构造函数包含

|  |
| --- |
| function (exports, require, module, \_\_filename, \_\_dirname){  //代码  } |

require() 引入一个模块

exports 导出对象的别名，等价于module.exports

module 当前模块

module.exports 当前模块导出的对象

\_\_filename 绝对路径和模块名称

\_\_dirname 绝对路径

|  |  |  |
| --- | --- | --- |
|  | 以路径开头 | 不以路径开头 |
| 文件模块 | require('./z.js')  自定义模块 | require('url')  常用于引入官方提供的核心模块 |
| 目录模块 | require('./02\_2')  package.json main对应的文件 | require('04\_2')  node\_modules文件查找  第三方模块 |

### day02

1. 包和npm

包：就是一个目录模块

npm：用于管理包的工具 npmjs.com

切换目录

cd 绝对路径

d: 切换盘符

shift+单机 powershell窗口

安装：npm install 包名

初始化package.json文件：npm init –y

自动安装package.json和package-lock记录的包：npm install

1. 查询字符串模块

查询字符串：浏览器向web服务器发送请求，传递数据的一种方式，位于URL中？后

parse() 解析为对象

1. URL模块

处理和解析url

stringify() 解析为字符串

parse() 解析url为对象

format() 将对象格式化为url

1. 文件系统模块

包括文件和目录形式

fs.stat(path,callback) callback——获取结果

stats.isFile()/isDirectory() 是否为文件/目录

1. 同步和异步

同步：阻止后续代码执行 返回值获取结果

异步：不会阻止后面代码执行 回调函数获取结果，进入线程池运行

1. 创建目录：

fs.mkdir(path,callback)/fs.mkdirSync(path)

1. 读取目录

fs.readdir(path,callback)/fs.readdirSync(path) 结果——数组

1. 删除目录

fs.rmdir(path,callback)/fs.rmdirSync(path)

1. 写入文件

fs.writeFile(path,data,callback)/fs.writeFileSync(path,data)

1. 追加写入

fs.appendFile(path,data,callback)/fs.appendFileSync(path,data)

1. 读取文件

fs.readFile(path,callback)/fs.readFileSync(path) 结果——buffer数据

1. 删除文件

fs.unlink(path,callback)/fs.unlinkSync(path)

1. 检测文件是否存在

fs.existsSync(path)

1. 文件中的流

fs.createReadStream(path)/fs.createWriteStream(path)

### day03

1. http协议

浏览器和web服务器端之间的通信协议

HTTP协议代理服务器常用端口号：80/8080/3128/8081/9098

HTTPS（securely transferring web pages）服务器，默认端口号为443/tcp 443/udp

FTP（文件传输）协议代理服务器常用端口号：21

Telnet（远程登录）协议代理服务器常用端口号：23

1. 通用头信息

Request URL:向服务器请求那些内容，对应浏览器网址

Request Method:请求方法 get/post/head put/delete/connect/options/trace



Status Code:响应的状态码

1\*\* 正在请求，还没结束

2\*\* 成功响应 OK

3\*\* 响应的重定向，跳转到另一个url Moved Permanently

4\*\* 客户端请求错误

5\*\* 服务器端错误 Service Unavailable

1. 响应头信息（response）

Accept-Ranges:响应的单位 byte 字节

**connection**:建立连接的方式 keep-alive 持久连接

Content-Type 响应的文件类型

Content-Length 长度

Content-Encoding 压缩方式

Location 跳转的url

**Cache-Control**: public, max-age=0 缓存控制

Expires: Fri, 22 Apr 2016 09:26:01 GMT 消息主体

1. 请求头信息（request）

Accept:接收的文件类型有哪些

Accept-Encoding 接收的压缩类型

User-Agent 客户端使用的浏览器

**Cache-Control**: public, max-age=0 缓存控制

**connection**:建立连接的方式 keep-alive 持久连接

Host: 127.0.0.1:8080 主机

1. 请求主体——可有可无

请求数据

1. http模块

创建web服务器，也可以模拟浏览器向另一台服务器发送请求

1. 模拟浏览器

http.get(url,callback)

1. 创建web服务器

|  |
| --- |
| let app=http.createServer() //创建web服务器  app.listen(8080) //设置端口  app.on('request',(req,res)=>{ //监听浏览器请求  req.url  req.method  res.writeHead(状态码,头信息对象) //设置状态码和头信息  res.write() //设置响应的内容  res.end() //结束并发送响应  }) |

### day04

1. 文件压缩

|  |
| --- |
| 引入zip模块  const zlib=require('zlib');  设置内容的压缩方式  Content-Encoding gzip  创建压缩  let gzip=zlib.createGzip()  要压缩的文件以流的方式读取并添加到压缩中，并响应到浏览器  fs.createReadStream('list.html').pipe(gzip).pipe(res) |

1. express框架

基于nodejs 平台 快速、开放、极简的web开发框架

|  |
| --- |
| const express=require('express')  let app=express()  app.listen(8080) |

1. 路由

浏览器向web服务器发请求，web服务器根据请求的url和请求的方法做出响应

**res 响应的对象**

send() 设置响应的内容并发送

redirect() 响应的重定向

sendFile() 响应文件 绝对路径\_\_dirname

|  |
| --- |
| 只能执行一次，表示响应结束 |

**req 请求的对象**

method 请求的方法

url 请求的url

query 查询字符串，格式为对象

params 路由传参数据，格式为对象

|  |
| --- |
| app.get('/package/:pname',(req,res)=>{  req.params  });  http://127.0.0.1:8080/package/mysql |

1. 获取post请求的数据

|  |
| --- |
| 事件：监听是否有数据传递  req.on('data',(chunk)=>{  chunk 传递的数据 buffer 查询字符串 对象  }) |

1. 路由器

管理所有的路由，将同一个功能模块下的所有路由放入同一个文件。为了代码维护和团队分工合作。

1. 创建路由器模块

|  |
| --- |
| 创建路由器对象  let router=express.Router()  挂载路由  router.get('/list',(req,res)=>{})  导出路由器对象  module.exports=router |

1. 在web服务器下引入并挂载

|  |
| --- |
| 引入路由器模块  把路由器挂载到web服务器，并添加前缀  app.use() |

### day05

1. 中间件

浏览器向服务器发请求，中间件可以拦截（过滤）请求，最终给路由提供服务

分为应用级（自定义）中间件、路由级中间件、内置中间件、第三方中间件、错误处理中间件

1. 应用级中间件

app.use(url,(req,res,next)=>{

url:路由的url

获取请求

做出响应

执行下一个中间件或路由

})

1. 路由级中间件

路由器的使用

1. 内置中间件

托管静态资源

把所有静态资源托管到指定的目录，如果浏览器请求文件，自动到该目录下查找，不需要在创建路由响应文件

app.use(express.static('目录'))

1. 第三方中间件

|  |
| --- |
| const bodyParser=require('body-parser'); //引入body-parser模块  app.use(bodyParser.urlencoded({  extended:false  })) //使用中间件，可以将post数据解析为对象  req.body //在路由中获取post数据 |

常用第三方中间件

http://www.expressjs.com.cn/

1. 传递及获取数据

|  |  |  |
| --- | --- | --- |
|  | 传递 | 获取 |
| 查询字符串（get） | uname=root&pwd=123456 | req.query |
| 路由传参 | /package/mysql | req.params |
| 请求主体  （post） | 只有表单 | req.body |

1. mysql模块

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

|  |
| --- |
| let connection=mysql.createConnection({}); //创建连接对象  connection.connect() //执行连接（可以省略）  connection.query(sql语句,callback); //执行SQL语句，会自动建立连接 |

查询 返回数组

插入 返回对象

修改 返回对象

删除 返回对象 affectedRows

解决SQL注入的问题：

|  |
| --- |
| 在SQL语句中使用占位符（？），对数据过滤 |

|  |
| --- |
| let pool=mysql.createPool({})  pool.query() |

## GIT

1. 查看安装的版本

$ git –version

1. 初始配置

git config --global user.name ""

git config --global user.email ""

1. 批量暂存

git add .

1. 日志的简洁显示

git log --pretty =oneline

1. 版本回退：移动HEAD指针

HEAD^^

HEAD~100 回退到上100个版本

$ git reset --hard "HEAD^^"

前往到指定版本

$ git reset --hard 唯一标识

查看唯一标识

$ git reflog

1. 从暂存区恢复

$ git checkout –- xyz.txt

1. 分支——主要出现在团队合作的项目中 相互不影响

master 主分支

1. 查看当前所有分支

$ git branch

1. 创建一个新分支

$ git branch 分支名

1. 切换到指定分支上

$ git checkout 分支名

1. 合并命令：合并指定分支内容 先切换到主分支上

$ git merge 分支名

1. 删除分支

$ git branch –D 分支名

注意：不能删除当前所在分支

1. 合并代码冲突

特殊格式代码：

<<<<<<<HEAD

当前分支的代码

==========

合并进入分支的代码

>>>>>>>分支名

解决办法：人工修改 再暂存+提交

1. git当网盘用

免费Git网站

1. GitHub

国外

1. 码云

中国 <https://gitee.com/>

201777187@qq.com/456123zxc

1. 把网上仓库克隆到本地

$ git clone 网址

1. 解决：重置保存的账号密码 必须管理员模式打开

$ git config --system –unset credential.helper

1. 上传到网上：

$ git push

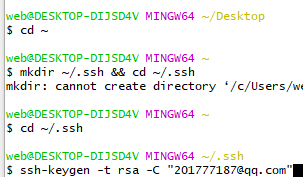
$ git pull

1. 制作钥匙

* 检查是否制作过钥匙

c:/用户/用户名/../.ssh/id\_rsa

* 在/.ssh创建钥匙



# 第二阶段

## THML5 Basic

### day01

1. web基础知识
2. web与internet

|  |
| --- |
| internet 全球性计算机互联网 互联网、因特网、交互网、交际网  www wide world 万维网，web |

1. Internet上的应用程序

|  |
| --- |
| C/S结构 客户端 需要下载安装客户端  B/S结构 浏览器 |

1. web运行原理

|  |
| --- |
| internet为web提供了网络环境  基于浏览器与服务器以及通信协议来实现数据的传输与展示 |

1. HTML快速入门
2. 什么是html

HypeText Markup Language 超文本标记语言

被标记括起来的文本，具备一定的功能

1. html的特点

* html或htm后缀
* 保存在服务器，运行在浏览器
* 标签语法，带有<>标识
* 在html中执行js脚本

1. html的基础语法

* 标记

标签，元素，节点，对象 <关键字>

* 标记分类

封闭类型标记 双标记

<关键字>内容</关键字>

非封闭类型标记 单标记

<关键字/> 或 <关键字>

ex: <input>

* 嵌套

嵌套有缩进

* 属性和值

修饰标签

<关键字 属性=值 属性=值></关键字>

值 支持“” ‘’无引号

属性分类：

1. 通用属性，标准属性

id :一个元素的唯一标识符

title:鼠标悬停在上面显示的文本

style:css 定义元素的行内样式

class：引入类选择器

1. 专有属性

* 注释

<!-- 注释 -->

注释不能嵌套 <>内不能添加注释

1. HTML文档结构
2. HTML构成

①文档类型的声明

<!doctype html> h5方式编码

②网页结构

代表网页的开始与结束，有且只有一套

<html></html>

<head></head> 头部

<body></body> 主体

1. head元素

<title></title> 页面标题

<meta/> 定义网页的全局信息

<meta charset="utf-8">

<meta name="description" content="">

<meta name="keywords" content="">

<script></script> 定义或引用js

<link/> css中引用外部样式

<style></style> css中定义内部样式

1. body

bgcolor="" 背景颜色

text="" 文本颜色

1. 文本标记
2. 标题

<hn></hn> n:1-6

属性 align left/center/right 文本水平对齐方式

1. 段落 paragraph

<p></p>

属性：align

1. 水平线

<hr>

属性：

align 水平对齐方式

color 分割线颜色

size 分割线尺寸

width/height px为单位

1. 换行标记

<br>

1. 预格式化标签

<pre></pre> 保留回车和空格

1. 特殊字符

&nbsp; 空格

&lt;< &gt;>

&reg; ® &copy;© ￥&yen; &times; ×

1. 文本样式标记

|  |
| --- |
| <b>加粗</b> <strong>bbb</strong><br>  <i>斜体</i> <em>iii</em><br>  <s>删除</s> <del>sss</del>  <u>下划线</u>  h5推荐使用有语义的标签  2<sup>3</sup>=8 上标  2<sub>3</sub> 下标 |

1. 分区元素

**块分区**

<div></div> 页面布局 单独成行

**行分区**

<span></span> 同一行文本，不同样式

1. 块级元素与行内元素

**块级元素**

在网页中独占一行，绝大多数是块级元素

从上往下排列

**行内元素**

在一行中显示

从左往右

**行内块** input

**table** <table></table>

### day02

1. 图像和链接

|  |
| --- |
| <img src="url" >  <a href="url"> |

1. url的表现形式

**绝对路径**

使用网络资源——不占用服务器本地存储空间；不稳定

完整路径：通信协议+域名+文件目录+文件

本地资源：绝对路径，从盘符开始找

**相对路径**

兄弟关系，直接写文件名

下级关系，先进入兄弟文件夹，在写文件名

上级关系，先进入上级文件夹 ../

1. img

<img/>

src="url"

alt="" 图片加载失败显示的文本

width="500" height="500" 设置图片宽高比和原始不同，会失真 解决：只设置一个

1. 超链接
2. 语法

<a></a>

属性

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

target="\_blank" 指定连接打开方式

\_self 默认当前

\_blank 新窗口

1. a标签的其他表现形式

<a href="a.zip">下载</a>

<a href="mailto:201777187@qq.com">发送邮件</a>

<a href="javascript:show()">调用js</a>

<a href="#">返回顶部</a>

1. 锚点

使用锚点——定义锚点

h4 <a name="锚点名称" ></a>

h5 <any id="锚点名称"></any>

使用锚点——跳转锚点

<a href="#锚点名称"></a>

1. 表格

渲染方式：表格内容一次性读取，之后一次性性渲染

1. 表格语法

<table>

<tr> ------table row

<td></td> ------table data

</tr>

</table>

1. 属性

**table**

border="1" 边框

width="400" 宽度

height="400" 高度

bordercolor="purple" 边框颜色

bgcolor="pink" 背景颜色

align="center" 表格本身水平方式

cellpadding="50" 单元格内边距

cellspacing ="50" 边框与边框的距离

**tr**

align="center" 当前行文本水平对齐方式

valign="bottom" 当前行文本垂直对齐方式

bgcolor="grey" 当前行背景色

**td/th**

width="150" height="150" 列宽度、高度

align="right" 内容的水平对齐

valign="top" 内容的垂直对齐

bgcolor="orange" 背景颜色

1. 不规则表格

**colspan=n 跨列**

指定单元格位置开始

横向向右合并n个单元格

被合并的删除

**rowspan=n 跨行**

指定单元格位置开始

纵向向下合并n个单元格

被合并的删除

1. 表格的可选标记

<th></th> 行列标题 可替代td

<caption></caption> 表标题，紧挨table

1. 表格复杂应用

表格行分组

<table>

<thead></thead>

<t><>

</table>

表格嵌套 在td中

table的特殊显示方式

尺寸：内容 尺寸 谁大决定最终

通过浏览器取色

<h1 style="color:red">qqq#66a5ff</h1>

<img src="h5.png" alt="">

### day03

1. 列表
2. 作用

有条理的显示数据，从上到下，在数据之前提供标识

1. 列表结构

有序列表 <ol></ol>

无序列表 <ul></ul>

列表项 <li></li> list item

1. 属性

ol

|  |
| --- |
| type="i" 标识项的类型 默认缺省值1 a A i  start="4" 起始编号 |

ul

|  |
| --- |
| type="none" 默认缺省值disc circle/square/none |

1. 列表嵌套

所有的嵌套必须要放在li中

1. 定义列表

给出一类事物，对这类名词的解释说明

|  |
| --- |
| <dl>  <dt>走路能减肥</dt> 解释的名词  <dd>走了1300公里，吃素</dd> 对名词的解释  </dl> |

1. 结构标记

用于描述整个网页的结构，取代div做布局的一套标记

增加了语义 有可读性

|  |
| --- |
| <header></header> 头部  <footer></footer> 脚步  <nav></nav> 导航  <aside></aside> 侧边栏  <section></section> 主体  <article></article> 文字相关的内容 |

1. 表单
2. 作用

提供可视化输入控件

自动收集整理用户输入的数据，并提交请求给服务器

注意：form表单自带收集数据的功能

ajax提交不需要form，但也失去自动收集数据的功能 使用js dom操作，手动收集数据

1. 表单组成

前端——表单控件

后端——后台接口

1. 表单

|  |
| --- |
| <form></form>  属性   1. action="" 提交表单时发生的动作，请求提交的url   默认——提交本页面   1. method="" 请求方法   默认——get  get：提交的数据最大2kb 明文提交  post：隐式提交 提交的数据大小没有限制 传递给服务器   1. enctype="" 编码方式   默认值 application/x-www-form-urlencoded  允许任意字符提交服务器（文件除外）  text/plain 允许提交普通字符（符号不是）  multipart/form-data 允许提交文件 |

1. 表单控件

能够与用户进行交互的可视化控件

**分类**

input元素 基础9中 h5——10

textarea 多行文本域

select/option 下拉选择

其他

**公有属性**

type 指定了input类型

name 为控件定义名称，提供给服务器使用

form表单提交必须写

ajax 不需写

value 控件的值 传递给服务器的值

注意：按钮不需要name value改变显示文本

disabled 禁用 不能修改，不能提交

无值属性

**input属性**

文本框和密码框 text、password

maxlength="3" 最大长度

placeholder="123" 默认值

readonly 无值属性，只读

按钮 submit

reset 让表单回到初始状态

button 普通按钮 调用js

H5新出的<button></button>效果同 submit

单选和复选框

单选按钮 type="radio"

复选按钮 type="checkbox"

name 分组使用，ajax也需要 必须写value值,没有是on

checked 无值属性，默认选中

隐藏域 数据提交给服务器，不给用户看

type="hidden"

文件选择框(上传文件) type="file"

method="post" enctype="multipart/form-data"

multiple 无值属性，可以选中多个文件

多行文本域 <textarea name="txt" id="" cols="30" rows="10"></textarea>

下拉选择框

<select>

<optin></option>

</select>

<select name="city" size="2" multiple> 下拉条，可以多选

其他元素

|  |
| --- |
| **label**  替代form中span  <label for="woman"></label>  **为控件分组**  <fieldset>  <legend>基本信息</legend>  </fieldset>  **浮动框架**  <iframe></iframe> 在一个html中引用其他的html  width="100%" height="800" 宽度、高度  src="../day03/03.html" 引用路径  frameborder="5" 边框  scrolling="no" 设置滚动条yes/no/auto |

**新表单元素**

邮箱：@前后有内容

<input type="email" name="em">

搜索类型 自带快速删除

<input type="search" name="s">

url 以http开头

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

电话号码 移动设备打开虚拟键盘

<input type="tel" name="tel">

数字类型

<input max="20" min="10" step="3" type="number" name="num">

max="20" min="10"

step="3" 步长

范围类型

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

颜色类型

<input type="color">

日期类型

<input type="date">

月份类型

<input type="month">

星期类型

<input type="week">

### day04

1. AJAX
2. URL

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

scheme 方案协议 协议



<user>:<pwd> 少量后台管理员使用

<host> localhost 127.0.0.1 域名/ip地址

<port> 端口号

<path> 目录结构 资源在服务器上存放的位置

<params> 参数 跟踪状态session/cookie

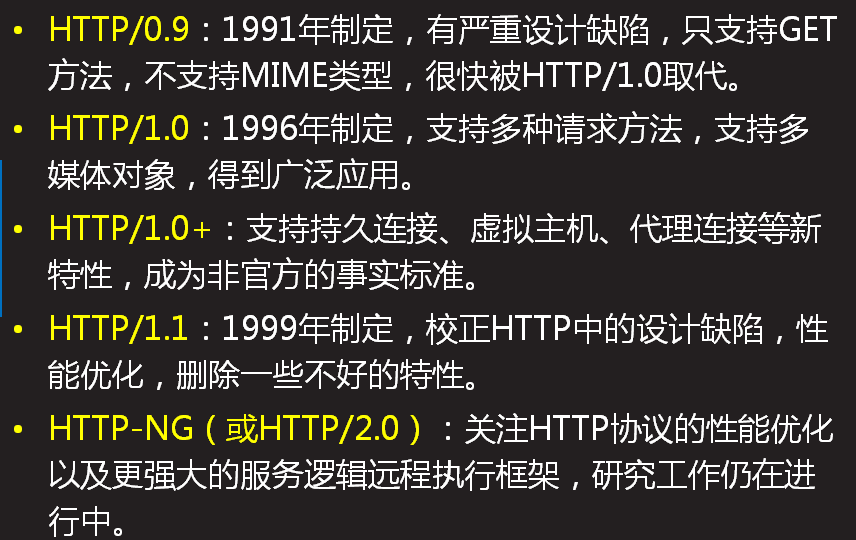
<query> 查询字符串

<flag> 锚点

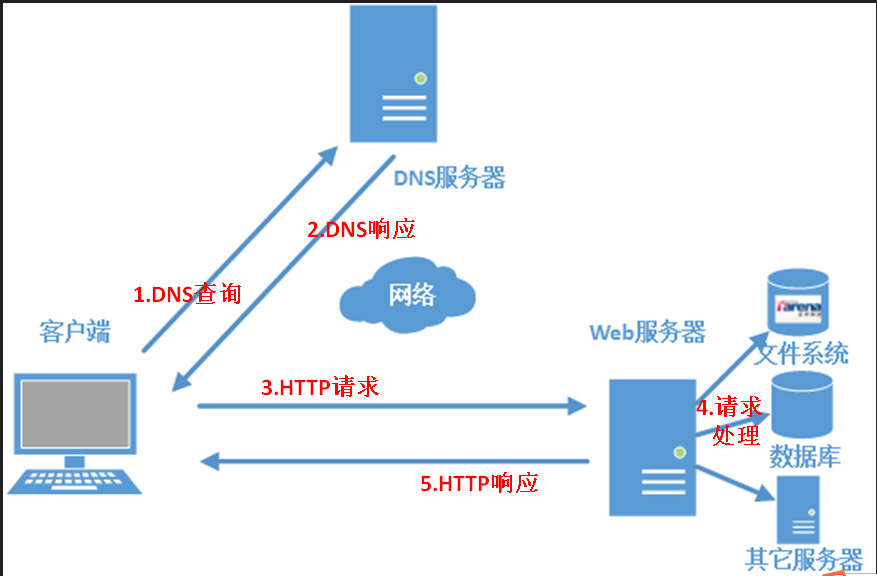
1. HTTP协议

hyper text transfer protocal 超文本传输协议

1. 历史与协议标准



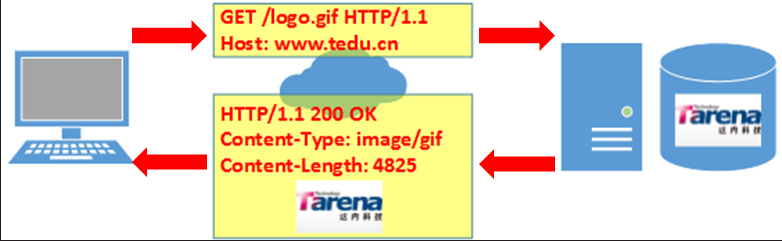
1. 请求响应流程



1. 消息/报文

request message

response message



**request message**

请求起始行

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GET 请求方法  HTTP/1.1 协议版本号  Host: [www.tmooc.cn](http://www.tmooc.cn) 请求的url  请求方法：原生的http的请求方法（get/post）   |  |  | | --- | --- | | 原生的http的请求方法 | resful API接口规则对请求方法的要求 | | get 明文传参，没有请求主体 | get 查询 无请求主体 | | post 隐式传参，通过请求主体传参 | post 创建 有请求主体 | | put 往服务器存放资源 | put 修改 有请求主体 | | delete 从服务器删除资源 | delete 删除 无请求主体 | | option 予请求 |  | |

请求头信息

|  |
| --- |
| Host: [www.tmooc.cn](http://www.tmooc.cn) 告诉服务器请求那个主机  Connection: keep-alive 告诉服务器开启持久连接  User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.139 Safari/537.36 告诉服务器浏览器基本信息  Accept-Encoding: gzip, deflate 告诉服务器 浏览器可接受的压缩文件格式  Accept-Language: zh-CN,zh;q=0.9 告诉服务器 接收的自然语言  Referer: <http://www.tmooc.cn/> 告诉服务器 这个请求来自于那个网页 |

请求主体

|  |
| --- |
| form data get delete  post put |

**response message**

响应起始行

|  |
| --- |
| HTTP/1.1 协议版本号  200 响应状态码  OK 原因短句 对响应状态码的简短说明  响应状态码，服务器告诉浏览器  1xx：正在请求  2xx：响应成功  3xx：重定向 301-永久重定向  302-临时重定向  304-命中缓存  4xx：404请求资源不存在  403 权限不够  405 请求方法不被允许  5xx：服务器代码错误 |

响应头信息

|  |
| --- |
| Date: Tue, 03 Dec 2019 03:38:00 GMT 服务器告诉浏览器响应时间  Connection: keep-alive 告诉浏览器已开启持久连接  Content-Type: text/html 响应主体类型  text/html html文本  text/css css文本  js文本 图片  text/plain 普通文本  application/xml xml字符串  application/json json字符串 |

响应主体

1. 缓存

客户端将服务器响应回来的数据进行自动的保存

1. 优点：

减少冗余的数据传输

节省服务器的带宽

1. 缓存的新鲜度与过期

请求-无缓存-连接服务器-存缓存-客户端得到

请求-有缓存-够新鲜-使用缓存-客户端得到

请求-有缓存-不新鲜-连接服务器确认是否过期-没过期-跟新新鲜度-客户端得到

请求-有缓存-不新鲜-连接服务器确人是否过期-已过期-连接服务器-存缓存-客户端得到

与缓存相关的的消息头

Cache-Control: max-age=0 此数据处理新鲜的秒数

修改缓存的消息头

<meta http-equive="消息头名称" content="值">

<meta http-equive=" Cache-Control " content=" max-age=67">

1. HTTP性能优化
2. http连接过程

发送请求-建立连接-服务器处理请求-访问资源-构建响应-发送响应-记录日志

1. http连接性能优化

减少连接次数

减少请求次数

尽可能的减少响应数据的长度

提高服务器端运行速度

1. 安全的http协议 HTTPS

|  |
| --- |
| SSL 为数据通信提供安全支持  客户端发送请求-》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  服务器发送响应-》ssl层解密-》客户端得到密文-》在ssl层解密-》得到响应文件 |

1. DOM操作（简易dom操作）

ajax不使用form 需要使用js的dom手动收集数据

1. javascript完整组成

js核心ECMA Script

DOM document object model 文档对象模型

js动态操作页面上标签

BOM browser object model 浏览器对象模型

js动态操作浏览器

1. 使用dom获取修改页面上的数据

得到标签对象

|  |
| --- |
| 通过事件调用方法  <button onclick="show()">显示</button>  获取元素对象  document.getElementById("uname") |

通过对象调用属性

|  |
| --- |
| 获取修改input的数据  obj.value="abc"  获取修改双标签数据<div></div>  obj.innerHTML="123"  注意：  es6：使用元素id当元素对象 |

1. innerHTML详解

可以在双标记中添加新的元素

动态改变页面结构，必须封装在字符串中

### day05

1. 事件

|  |
| --- |
| onload body 页面加载激活  文本框/密码框  获取焦点事件 onfocus  焦点移除事件 onblur |

1. AJAX
2. 同步 synchronous

同步访问：只能等待服务器的响应，不允许其他操作

1. 异步 asynchronous

异步访问：可以做其他操作

1. 什么是AJAX

AJAX asynchronous javascript and xml

本质：使用js提供的异步对象，异步的向服务器发送请求，并接受响应回来的数据

异步对象：XMLHttpRequest

1. 使用AJAX
2. 创建异步对象

new xhr=XMLHttpRequest();

1. 创建请求

xhr.open(method,url,isAsyn);

1. 发送请求

xhr.send(参数);

请求方法有请求主体，参数填请求主体

请求方法无请求主体，参数填null或不填

1. 接收响应

|  |
| --- |
| xhr.onreadystatechange=function()  {  if (xhr.readyState==4 && xhr.status==200)  {  let result=xhr.responseText;  console.log(result);  }  } |

使用

### day06

使用restful的get传参

|  |
| --- |
| 后台定义接口，需要把参数放入接口中，没有？添加：  router.get("/login\_restful/:uname&:upwd",  通过接口获取参数  let $uname=req.params.uname;  前台直接调用接口  xhr.open("get","/ajax/login\_restful/"+$uname+"&"+$upwd,true);  注意：多个参数之间可以用& - 但前后台一致  restfulget 传参不能在后台进行非空验证 报404 |

post 方法

|  |
| --- |
| xhr.open("post","/ajax/post\_login",true);  重新设置请求信息格式  xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  let formdata=`uname=${$uname}&upwd=${$upwd}`;  xhr.send(formdata);  有请求主体，send把请求主体当参数  后台的router：post-> router:put  前台的xhr.open("put",,) |

1. JSON数据格式
2. js对象的格式
3. json数据格式

json是一个字符串，json串

JavaScript object notation

* 一对{}表示一个对象
* 所有属性名称必须使用""包裹
* 属性值也是字符串，需要""包裹
* 由于json是字符串，外层需要''包裹

1. 把json解析为对象

arr=JSON.parse(json串)

1. xml解析

|  |
| --- |
| extensible markup language 模仿html  可扩展标记语言 |

1. 语法

|  |
| --- |
| 版本声明  <?xml version='1.0' encoding='utf-8'?>  标记必须成对出现  标签关键字区分大小写  一个xml有且只有一个根标签 |

restful接口文档

1. 登录 /pro/v1/login/:uname&:upwd

get 1-成功 0-失败

1. 获取所有用户 userlist

get 返回json

1. 删除用户 deluser/:uid

delete 1-删除成功

1. 查询用户 search/:uid

get 返回json

1. 修改用户 update/:uid

put 1-成功 0-失败

1. 注册 reg

post 1-成功 0-失败

1. 根据用户名称查询用户 search\_uname/:uname

get 1-存在 0-不存在

## CSS

### day01

1. css概述
2. 什么是css？

Cascading style sheets

层叠样式表,级联样式表,简称样式表

1. 作用

修饰页面中的元素

1. HTML与css的关系

html：页面的结构搭建，数据显示

css：页面的装饰

1. css和html属性的使用原则

|  |
| --- |
| w3c 建议尽量使用css的方式取代html属性  css：  样式代码可重用  提高代码可维护性5 |

1. css的语法规范
2. 使用css的方式

①行内样式，内联样式

|  |
| --- |
| 在元素的style属性中，做样式声明  <any style="样式声明">  样式声明 样式属性：样式值;  color:red; 字体颜色  background:yellow;背景颜色  font-size:36px; 字号大小  内联样式的特点：  不能复用  优先级最高  项目中一般极少使用内联样式 |

②内部样式

|  |
| --- |
| 在页面的<head></head>定义<style></style>  在style中编写样式 选择器{样式声明；}  选择器：就是一个条件 符合条件都会应用样式  div{color:purple;background:pink;}  内部样式特点：  可以重用，但只对当前页面生效  重用有局限性，项目使用较少 |

③外部样式

|  |
| --- |
| 单独创建一个css文件，在其中声明样式  在任何的html中  使用link标签引入  <link rel="stylesheet" href="my.css">  可以广泛使用 |

1. css的样式特征

①继承性

|  |
| --- |
| 大部分的css效果，是可以直接被后代继承的 |

②层叠性

|  |
| --- |
| 可以为一个元素设置多个样式规则  属性不冲突 |

③优先级

|  |
| --- |
| 属性冲突，按优先级规则  默认优先级 最高 内联样式  内部外部 就近原则（link编写位置）  最低 浏览器默认样式 |

④调整优先级

|  |
| --- |
| !important 规则  放在值与；之间 中间有空格  内联样式不能替代 内联样式不能添加 |

1. 基础选择器
2. 选择器作用

规范了页面中那些元素能够使用定义好的样式

就是为了匹配元素

1. 选择器详解

①通用选择器

|  |
| --- |
| \*{样式声明} \*匹配所有元素  \*{margin:0;padding:0;} 所有元素的内外边距清零——css reset |

②元素选择器，标签选择器

|  |
| --- |
| 设置页面中所有这个元素的公用样式  div{样式声明} |

③ID选择器（专属定制）

|  |
| --- |
| <any id="id值"></any>  #id值{样式声明}  一般不单独使用  一般用于后代或子代选择器 |

④类选择器

|  |
| --- |
| 定义： .类名{样式声明}  使用：<any class="类名"></any>  类名不能以数字开头 只能有两种符号- \_  类选择器的其他方式  多类选择器  <any class="类名1 类名2 ...">  分类选择器1  元素选择器.类名{样式声明}  分类选择器2  .类名1.类名2{样式声明}  分类选择器作用  清晰找到目标，  提升选择器权值 |

⑤群组选择器

|  |
| --- |
| 将多个选择器使用，连接 统一定义公共样式  li,ul{样式声明} |

⑥后代选择器

|  |
| --- |
| div p span{color:red;}  后代一级或多于一级的嵌套 |

⑦子代选择器

|  |
| --- |
| 通过元素的子代  div>span{color:blue;}  子代 一级嵌套 |

⑧伪类选择器

|  |
| --- |
| 匹配元素不同状态的选择器  link 匹配元素未访问状态  visited 访问后的状态  active 激活状态  hover 鼠标悬停的状态  focus 获取焦点  link》visited》hover》active |

⑨选择器的权值

|  |
| --- |
| 标识当前选择器的权值 权值越大 优先级越高  !important >1000  内联 1000  id 选择器 100  类选择器 10  伪类 10  元素选择器 1  通用选择器 0  继承的样式 无  权值特点：  所有元素权值相加 权值大的优先显示  权值相同，就近原则  群组选择器不相加，各算各的  内联样式不能添加important  选择器权值的计算，最终结果不会超过本身的最大数量级 |

1. 尺寸与边框
2. 尺寸属性

|  |
| --- |
| 宽度和高度  width 宽度 px单位数据，%父元素的  height 高度 px单位数据，%父元素的，父元素没定义失效  min-width  max-width  min-height  max-height |

相对单位

em 相对于父元素定义的数值的整倍数

rem 相对于html元素定义数值的整倍数

% 相对于父元素数值的百分比

1. 页面中允许设置尺寸的元素

|  |  |  |
| --- | --- | --- |
| 块级元素 | 行内元素 | 行内块 |
| 设置宽高有效 | 多数设置宽高无效 | 设置宽高有效 |
| 没设置宽，默认宽是父元素100%  没设置高，默认高靠内容撑开 | 实际宽高靠内容撑开 | 默认自带宽高  不同浏览器对input的默认样式差异比较大 |

1. 溢出的处理

|  |
| --- |
| 内容较大，元素区域较小  默认纵向溢出  overflow：visible 溢出部分显示  hidden 溢出影藏  scroll 添加滚动条 不管溢出都在  auto 自动添加滚动条，不溢出不添加  overflow-x：scroll 单独设置x轴  overflow-y：scroll 单独设置y轴  如何改成横向溢出：  在宽度较小的容器里添加宽度较大的元素 |

附加知识点——合法颜色

|  |
| --- |
| 颜色单词  #rrggbb 6个16进制  #ffffff 白色 #000000 黑色  #aabbcc-》#abc  rgb(r,g,b)  rgba(r,g,b,alpha) alpha 透明度0~1之间  rgb(a%,g%,b%)  hsl(300,50%,60%)  #006699--->rgb(0,102,153) |

### day02

1. 边框

①边框的简写方式

|  |
| --- |
| border:width style color;  width:边框的粗细 px  style：边框样式 dashed 断线虚线  solid 实线  dotted 点点虚线  double 双实线  color 边框颜色 transparent 等同于rgba(0,0,0,0)  最简方式 border: style  border:0 取消边框（css中只有0可省略单位） |

②单边定义

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

③单属性定义

|  |
| --- |
| border-style:solid;  border-width:30px;  border-color:yellow; |

④单边单属性

|  |
| --- |
| border-方向-属性 一共12个  ex：border-top-color:purple; |

1. 倒角，圆角

|  |
| --- |
| border-radius 以px为单位  % 50%圆  border-radius:0 取消圆角 |

1. 边框阴影

|  |
| --- |
| box-shadow:h v blur spread color inset  blur 模糊距离  spread 阴影大小  最简方式：水平、垂直偏移 |

1. 轮廓

|  |
| --- |
| 边框的边框，绘制与边框外面的线条，不占据元素空间  outline:width style color  一般情况清除轮廓：outline:0 |

1. 框模型——盒子模型

|  |
| --- |
| 元素在页面上实际占地空间的计算方式  实际占地宽高：  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  上外边距+上边框+上内边距+内容区域高度+下内边距+下边框+下外边距  外边距：margin 元素与元素  内边距：padding 边框与内容 |

1. 外边距

|  |
| --- |
| 改变外边距，元素有位移效果  元素位置微调  精确关注：上外边距和左外边距 |

①语法

|  |
| --- |
| margin:设置4个方向的外边距  margin-方向  % 上下左右 都是父元素宽度的百分比  auto 上下无效 只对设置宽度有效 水平居中 |

②简写方式

|  |
| --- |
| margin：v1; 4个生效  margin：v1 v2； v1上下 v2 左右  margin:0 auto;  margin :v1 v2 v3 上 左右 下  margin:v1 v2 v3 v4 上 右 下 左 |

1. 外边距特殊效果

①外边距合并

|  |
| --- |
| 两个垂直外边距相遇时，会合并成一个，值以最大为准  解决方案：规避  只写一个 |

②关于块级元素，行内元素，行内块

|  |  |  |
| --- | --- | --- |
| 块级元素 | 行内元素 | 行内块 |
| 设置宽高有效 | 多数设置宽高无效  自带宽高属性的,设置有效 | 设置宽高有效 |
| 没设置宽，默认宽是父元素100%  没设置高，默认高靠内容撑开 | 实际宽高靠内容撑开 | 默认自带宽高  不同浏览器对input的默认样式差异比较大 |
| 4个外边距都有效 | 只有左右有效 | 4个外边距都有效  上下边距改变，一行的元素上位置也会变 |

③默认自带外边距

|  |
| --- |
| body 8px  h1~h6 p ol pre  ul 上下外16px 左右内40px |

④外边距溢出

|  |
| --- |
| 为子元素添加上边距，会作用在父元素上  特殊：父元素不许有上边框  子元素内容区域上沿与父元素内容区域上沿一致  解决：父元素添加边框，增加了父元素的实际占地高度  父元素设置内边距 增加了父元素的实际占地高度  父元素添加overflow:auto/hidden 父元素不能溢出显示  给父元素的第一个子元素添加<table></table> |

⑤外边距颜色

|  |
| --- |
| 外边距是透明的  元素位置微调  一组元素之前有空隙，可以使用外边距 |

### day03

1. 内边距——padding

|  |
| --- |
| 边框到内容之间的距离  实际宽高没变，实际占地空间变大 |

①语法

|  |
| --- |
| padding:v1;  padding:v1 v2  paddding:v1 v2 v3  paddding:v1 v2 v3 v4  取值px为单位的数字  % 父元素宽度百分比  没auto  padding-top  padding-right  padding-bottom  padding-left |

什么时候使用

|  |
| --- |
| 把元素撑开 |

1. box-sizing

|  |
| --- |
| box-sizing:改变盒子模型计算公式  content-box 盒子模型默认计算方式 默认  border-box 左右外边距+width（padding+border+内容） |

1. 背景
2. 背景色

|  |
| --- |
| background-color 合法的颜色值 |

1. 背景图片

|  |
| --- |
| background-image:url(01.png) |

1. 背景图平铺

|  |
| --- |
| background-repeat repeat 默认 平铺  no-repeat 不平铺  repeat-x/y x y轴平铺 |

1. 背景图片的定位

|  |
| --- |
| background-position  取值 1个以px为单位的数字 控制x轴，y轴居中  2个以px为单位的数字 控制x轴y轴  % 1个 控制x轴，y轴居中  % 2个 控制x轴y轴  关键字 x:left center right y:top center bottom |

1. 背景图片的尺寸

|  |
| --- |
| background-size  取值 1个以px为单位的的数字 同时设置宽高  2个以px为单位的数字 分别设置宽高  % 同上  cove 元素被背景图充满 图片不一定显示全  contain 元素内背景图片是完整的 元素会有空白区域 |

1. 背景图片的固定

|  |
| --- |
| background-attachment  默认 scroll 滚动条  fixed 固定  background-position 是相对body定位 |

1. 背景简写

|  |
| --- |
| background:color image repeat attachment positin  注意：size单独写  最简color 或 image  重写不使用简写 不然会覆盖其他值 |

1. 渐变 gradient

|  |
| --- |
| 控制渐变的主要因素：色标 |

1. 渐变分类

|  |
| --- |
| 线性渐变 直线的方式  径向渐变 圆形填充  重复渐变 直线和圆形重复填充 |

1. 线性渐变

|  |
| --- |
| background-image:linear-gradient(方向，色标1 ...)  方向：写终点 to top/right/bottom/left  角度 0deg  色标 颜色+位置%  颜色 平均分配 |

1. 径向渐变

|  |
| --- |
| background-image:radial-gradient(半径 at 圆心 x 圆心 y，色标);  色标使用%表示位置 半径才有效 |

1. 重复渐变

|  |
| --- |
| background-image:repeating-linear-gradient(to right,#000 0px,#ffa 1px,#000 2px)  background-image:repeating-radial-gradient(10px at center center,#f00 -2%,#ff0 18%,#0f0 33%,#011 52%,#0ff 76%,#00f 91%); |

1. 浏览器兼容问题

|  |
| --- |
| 编写兼容 ie8.0以下版本的浏览器的行为——css hack  浏览器内核 -webikt- chrome/safari  -moz- firefox  -ms- ie  -o- opera  background-image:-webkit-linear-gradient(left,#000 0px,#ffa 1px,#000 2px)  注意：内核写渐变 方向从起点开始 |

1. 文本格式化
2. 字体属性
3. 字号大小

|  |
| --- |
| font-size: 单位：px em rem pt |

②字体类型

|  |
| --- |
| font-family  取值：pc库中  font-family:"Gigi",黑体,幼圆; |

③字体加粗

|  |
| --- |
| font-weight:bolder normal lighter bold  无单位数字 100的整倍数100~1000 |

④字体样式

|  |
| --- |
| font-style:normal/italic |

⑤小型大写字母

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

⑥简写方式

|  |
| --- |
| font: style variant weight size family  最简：font:size family |

### day04

1. 文本属性

①文本颜色 color

②文本对齐

text-align:justify/left/center/right

margin:0 auto 让块级元素居中

text-align:center 让元素内部的孩子（非块级）居中

③行高

|  |
| --- |
| 定义一行文本的高度  行高特点：行高大于文本字号时，文本将在行高的垂直中间显示  文本想相对容器垂直居中，行高高度设置容器高度  line-height:200px  取值：px单位  无单位数字(整数小数都行) 行高值字号整倍数  文本多行不建议行高显示 |

④文本线条的修饰

|  |
| --- |
| text-decoration: underline 下划线  overline 上划线  line-through 删除线  none 去掉所有线条  text-decoration:none a标签去除下划线 |

⑤首行缩进

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

⑥文本阴影

|  |
| --- |
| text-shadow:h-shadow x-shadow blur color |

1. 表格相关样式
2. 表格常用属性

|  |
| --- |
| table  尺寸 边框 背景 字体 文本 内外边距  th/td  尺寸 边框 背景 字体 文本 内边距 外边距失效  vertical-align:bottom |

1. 表格特殊的表现方式

|  |
| --- |
| table设置的尺寸值大，td的值小，  总结： |

1. 表格特有的属性

①边框合并

|  |
| --- |
| border-collapse:默认值 seperate分离  collapse 合并 |

②边框边距

|  |
| --- |
| border-spacing:10px 50px 水平，垂直 |

③标题位置

|  |
| --- |
| caption-side:bottom/top; |

④设置表格显示规则

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| table-layout:auto 默认自动布局  表格数据预先读取进内存，一次渲染  fixed 固定布局   |  |  | | --- | --- | | **table自动布局的特点** | **table固定布局特点** | | 单元格大小自动适应内容 | 单元格大小，取决于设置值 | | 表格复杂时，加载慢 | 任何情况下，加载都很快 | | 表格比较灵活 | 固定表格不灵活 | | 适用于不确定每列大小，并且不复杂的表格 | 适用于确定每列尺寸的表格 | |

1. 定位
2. 什么是定位

|  |
| --- |
| 设置页面中元素的位置 |

1. 分类

|  |
| --- |
| 普通流定位  浮动定位  相对定位  绝对定位  固定定位 |

1. 普通流定位（默认文档流定位）

|  |
| --- |
|  |

1. 浮动定位

|  |
| --- |
| 让块级元素横向显示  float:  取值：left 停靠父元素左边或者挨着其他浮动元素  right 停靠父元素右边或者挨着其他浮动元素  none 默认  特点：  脱离文档流 不占页面空间 后面元素上前补位  停靠父元素左右边或者挨着其他浮动元素  父元素横向显示不下的子元素，会自动换行  主要解决块级元素在同一行显示 |

1. 浮动引发的特殊情况

①元素一旦浮动，若元素没定义宽度，以内容为主

②元素一旦浮动，变成块级元素，宽高垂直外边距有效

③浮动元素占位

|  |
| --- |
| 浮动元素会在浮动方向上占位，导致被挤下去的元素不能在之前元素的位置上浮动 |

④天生不被浮动元素压着

|  |
| --- |
| 行内元素、行内块元素、文本不被浮动元素压着  环绕着浮动元素显示 |

脱离文档流意味着什么

|  |
| --- |
| 元素不在文档流占位  后续元素上位补齐  若没设置宽高以内容为主  元素变成=块级元素（设置宽高，上下外边距有效） |

1. 清除浮动（清除浮动元素带来的影响）

|  |
| --- |
| clear：left  right  both  清除之前浮动元素 |

1. 高度坍塌

|  |
| --- |
| 父元素没设置高度，内部所有子元素都脱离文档流，导致父元素认为在文档流中没子元素高度，父元素就失去了高度  解决：  父元素设置高度 高度不确定  父元素浮动 会影响父元素的同级元素  overflow：hidden  在父元素中添加一个空的块级元素，设置clear:both; |

### day05

显示方式

|  |
| --- |
| 决定元素在页面中的表现方式  display：inline 行内  block 块级  table table  inline-block 行内块  none 隐藏 |

显示效果

|  |
| --- |
| visibiity 默认值visible  hidden 隐藏  visibility:hidden; 占位置隐藏  display:none 不占位置隐藏 |

透明度

|  |
| --- |
| opacity: 0~1  rgba() 只会改变当前的透明度  opacity 元素内部只要颜色相关的透明度都会改变  rgba transparent |

光标的设置

|  |
| --- |
| 箭头 default  小手 pointer  文本 text  等待 wait  帮助 help  十字 crosshair |

垂直对齐方式

|  |
| --- |
| ①table/td/th  vertical-align:top/midle/bottom  ②img使用img前后的行内（行内块）  vertical-align:默认值 baseline基线  top/midle/bottom |

列表样式

|  |
| --- |
| list-style-type:none;  square circle  list-style-image:url(sinablog.png)  list-style-position:outside 默认 在ul内边距中  inside ul内容中  简写：  list-style:type/image position  list-style:none 去点 |

1. 定位（相对，绝对，固定）

|  |
| --- |
| postion  取值：static 默认静态  relative 相对定位  absolute 绝对  fixed 固定  注意：当元素被position修饰  解锁偏移属性  top/right/bottom/left  已定位元素 |

1. 相对定位

|  |
| --- |
| position：relative  top  right  left  bottom  特点：相对定位不脱离文档流  没偏移属性或偏移量为0 效果没做定位一样  相对自己原来的位置偏移  偏移属性的特点：  以左上为准  使用场合：  自身元素位置微调，类似margin  作为绝对定位元素的祖先级定位 |

1. 绝对定位

|  |
| --- |
| position：absolute  脱离文档流  没有祖先级已定位元素 相对body左上角定位  绝对定位 会相对于自己最近的 已定位 祖先级元素的左上角取偏移 |

1. 固定定位

|  |
| --- |
| position：fixed 配合偏移量  脱离文档流 |

1. 堆叠顺序

|  |
| --- |
| 只能设置在已定位元素上  z-index:整数  默认后写的元素堆叠顺序高 但是值不大于1  一般情况下1000以内  定位的脱离文档流和浮动的脱离文档流不是一个体系  堆叠顺序对父子关系无效 儿子永远在父上 |

### day06

1. 复杂选择器
2. 兄弟选择器

|  |
| --- |
| 兄弟元素:具备相同父元素的平级元素  兄弟选择器,只能往后找,不能往前找  ①相邻兄弟选择器  选择器1+选择器2{}  获取紧紧挨在选择器1元素后面的选择器2的元素  ②通用兄弟选择器  选择器1~选择器2{}  获取选择器1后面所有符合选择器2的弟弟元素  多用于一组元素,设置除了老大以外,其它元素的公共样式 |

1. 属性选择器

|  |
| --- |
| 1.[attr]{} attr表示属性名称  匹配页面中所有带有attr属性的元素[class] {color:#f00;}  2.[attr1][attr2]....{}  匹配页面中同时带有attr1,attr2,.....属性的元素[class] [title] { }  3.elem[attr1][attr2]..{}  匹配页面中同时带有attr1,attr2,.....属性的elem元素  span[title]{color:#f00;}  4.elem[attr1=v1][attr2=v2]....{}  匹配页面中同时带有attr1并且值为v1,attr2并且值为v2,.....属性的elem元素  p[title="千里眼顺风耳"]{color:#f00;}  5.属性值的模糊查询  [attr^=value]{} 匹配attr的值以value开头的元素  [attr$=value]{} 匹配attr的值以value结尾的元素  [attr\*=value]{} 匹配attr的值中有value的元素  [attr~=value]{} 匹配attr的值中有value这个独立的单词的元素(value前后有空格) |

1. 伪类选择器

|  |
| --- |
| :link :visited :hover :active :focus |

①目标伪类

|  |
| --- |
| 让被激活的锚点,应用的样式  :target{} |

②结构伪类

|  |
| --- |
| selector:first-child{}  找的是selector的父元素的第一个儿子  而且第一个儿子还得符合selector  selector:last-child{}  找的是selector的父元素的最后一个儿子  而且第一个儿子还得符合selector  selector:nth-child(n){} n从1开始 |

③:empty

|  |
| --- |
| 匹配内部没有任何元素的标签  内部不许有空格,文字,元素,回车 |

④:only-child

|  |
| --- |
| 匹配属于其父元素唯一子元素  唯一:仅限于元素,可以添加文本,空格,回车 |

⑤否定伪类

|  |
| --- |
| :not(selector) 符合selector的都不要 |

1. 伪元素选择器

|  |
| --- |
| :first-letter 或者 : :first-letter  匹配第一个字符  :first-line 或者 ::first-line  匹配第一行文本(如果与首字母冲突,听首字母的样式)  ::selection  匹配用户鼠标选中的文本  必须是双::,首字母无效,只能设置背景色和字体颜色 |

1. 伪元素选择器----内容生成

|  |
| --- |
| 使用css代码,动态的添加html元素  ::before 或者 :before  是在当前元素的内容区域的最前方(还是在内容区中),添加一个假的元素  这个元素默认是行内元素  使用content添加内容 (content只能设置文本和图片)  可以设置其他任意样式  可以设置display改变元素的显示方式  ::after或者 : after  是在当前元素的内容区域的最后面(还是在内容区中),添加一个假的元素  伪元素内容生成解决的问题：  外边距溢出：before{content:'';  display:table;}  高度坍塌： :after{  content:'';  display:block;  clear:both;  } |

1. 弹性布局(重要\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)
2. 什么是弹性布局

|  |
| --- |
| 就是一种布局方式,  主要解决某个元素的子元素的布局方式(横向排列,或者纵向排列)  为布局提供很大的灵活性 |

1. 弹性布局的相关概念

|  |
| --- |
| 容器:  要发生弹性布局的子元素们的父元素,称之为容器  也就是设置了display:flex的元素.  这个元素本身不是弹性布局,他的儿子们是弹性布局  项目:  要做弹性布局的子元素们,称之为项目  就是设置了display:flex的元素的子元素们  主轴:(4根)  项目们的排列的方向的一根轴,称之为主轴  如果项目们横向排列,x轴就是主轴  如果项目们纵向排列,y轴就是主轴  项目们在主轴上的排列顺序,就是主轴的起点和终点  交叉轴(2根)  与垂直相交的一根轴,叫做交叉轴  项目们在交叉轴上的对齐方式,就是交叉轴的起点和终点 |

1. 语法

|  |
| --- |
| 在项目中,设置display:flex 将块级元素设置为弹性容器  inline-flex 将行内元素设置弹性容器  特点:容器的text-align vertical-align 失效  项目的float clear失效 |

1. 容器的属性

①主轴方向：

|  |
| --- |
| flex-direction:  row 默认值,主轴是x轴,主轴起点在左边  row-reverse 主轴是x轴,主轴起点在右边  column 主轴是y轴,主轴起点在顶端  column-reverse 主轴是y轴,主轴起点在底部 |

②设置项目是否换行

|  |
| --- |
| flex-wrap:  nowrap 默认值,不换行  wrap 换行  wrap-reverse 换行,并反转 |

③主轴方向和换行的缩写

|  |
| --- |
| flex-flow:row wrap;  方向 换行 |

④项目在主轴上的对齐方式

|  |
| --- |
| justify-content:  flex-start 默认值,主轴起点对齐  flex-end 主轴终点对齐  center 主轴中间对齐  space-between 两端对齐,两端无空白  space-around 每个间距大小相同,两端有空白 |

⑤项目在交叉轴上的对齐

|  |  |
| --- | --- |
| align-items:  flex-start 默认值,在交叉轴起点对齐  flex-end 在交叉轴终点  center 在交叉轴中间对齐  stretch 前提 项目不定义高度,设置stretch,项目充满整个交叉轴 |  |

1. 项目的属性 只能设置在某一个项目上,不影响其他项目的效果

①项目排列属性

|  |
| --- |
| order:无单位整数;  定义项目排列顺序,值越小,越靠近主轴起点,默认值为0 |

②flex-grow

|  |
| --- |
| 定义项目的放大比例  如果容器有足够大的剩余空间,项目将变大  默认值 0 不放大,取值越大,项目放大的越快 |

③flex-shrink

|  |
| --- |
| 如果容器空间不足,设置项目该如何缩小  默认值 1. 设置为0不缩小,取值越大,项目缩小越快 |

④align-self

|  |  |
| --- | --- |
| 单独设置每一个项目交叉轴的对齐  flex-start 默认值,在交叉轴起点对齐  flex-end 在交叉轴终点  center 在交叉轴中间对齐  stretch 前提 项目不定义高度,设置stretch,项目充满整个交叉轴  auto 使用给容器定义的align-items的值 | 单独设置交叉轴的对齐 |

1. css hack

|  |
| --- |
| 由于不同浏览器,对css的解析认知不同,会导致同一份代码,在不同浏览器下,页面显示效果不同  1.开发人员要针对不同的浏览器写不同的样式.这个行为就叫写css hack  -o-  -webkit-  -ms-  -moz-  2.css reset |

### day07

1. 转换(重点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)
2. 什么是转换

|  |
| --- |
| 改变元素在页面中的位置,大小,角度,以及形状  2D转换 x轴 和 y轴上的转换  3D转换,模拟的,添加z轴 |

1. 转换的属性

|  |
| --- |
| 属性:transform:transform-function  转换函数  转换函数:  2D:4个  3D:1个 |

1. 2D转换

①位移

|  |
| --- |
| transform:  取值 translate(x) 设置在x轴位移  translate(x,y) 设置在x和y轴的位移  translateX(x) 设置在x轴位移  translateY(y) 设置在y轴位移 |

②缩放，改变元素大小

|  |
| --- |
| transform:  取值 scale(v1) 同时设置x轴y轴的大小  scale(v1,v2) 分别设置x轴y轴的大小  scalex(x)  scaley(y)  取值:1. v1>1 放大  2. 0> v1 <1 缩小  3. -1<v1<0 缩小,翻转  4. v1<-1 放大,翻转 |

③旋转

|  |
| --- |
| transform:  取值 rotate(ndeg)  n + 顺时针 - 逆时针  转换原点,会影响旋转效果  transform-origin:  取值: 以px为单位的数字  %  关键字 x:left/center/right y:top/center/bottom |
| 旋转会带着坐标轴一起旋转  所以旋转之后的位移方向,会发生变化 |

④倾斜

|  |
| --- |
| transform  skew(ndeg)和skewx(ndeg)  y轴向着x轴倾斜ndeg,  skewy(ndeg)  让x轴向着y轴倾斜ndeg  n + 顺时针 – 逆时针 |

1. 3D转换

|  |
| --- |
| 3D转换都是模拟 |

①透视距离

|  |
| --- |
| 模拟人的眼睛到3d转换元素之间的距离,距离不同,看到的效果就不同  perspective: 距离,这个属性要设置在转换元素的父元素上 |

②3D旋转

|  |
| --- |
| 1.rotatex(xdeg) 以x轴为中心,旋转元素 老式爆米花机 烤羊腿  2.rotatey(ydeg) 以y轴为中心,旋转元素 土耳其烤肉 旋转门  3.rotatez(xdeg) 以z轴为中心,旋转元素 摩天轮 电风扇\  4.rotate(x,y,z,ndeg)  x,y,z取值为0代表不参与旋转  >0代表参与 |

1. 过度
2. 什么是过度

|  |
| --- |
| 让css的值,在一段时间内平缓的变化  两个状态之间的变化可以使用过渡,可以使用动画  但是超过两个状态,只能使用动画 |

1. 过渡语法

①过度属性

|  |
| --- |
| transition-property:  取值,参与过渡的所有属性  transform不添加进过渡属性,也自动参与过渡  all,所有支持过渡的属性,都参与过渡  支持过渡的属性有哪些  1.颜色  2.大多数的取值为具体数字的属性  3.阴影  4.转换  5.visibility |

②过渡时长

|  |
| --- |
| transition-duration: 指定多长时间完成此过渡效果 取值 s/ms为单位的数字 |

③速度时间函数

|  |
| --- |
| transition-timing-function:  ease 默认值,慢开,中间加速,慢关  linear 匀速  ease-in 慢速开始,一直加速  ease-out 快速开始,慢速结束  ease-in-out 慢速开始,中间先加速再加速,慢速结束  <https://cubic-bezier.com/#.25,.1,.25,1> 贝塞尔曲线测试网页 |

过度的延迟时间

transition-delay:

过度的简写方式

transition:property duration timing-function delay;

最简方式

transition:duration;

过度代码编写位置

|  |
| --- |
| 原本样式  hover |

1. 动画

|  |
| --- |
| 关键帧 |

|  |  |
| --- | --- |
| 过度 | 动画 |
|  |  |
|  |  |
|  |  |

动画的使用步骤

声明动画

|  |
| --- |
| @keyframes 动画名称{  0%{动画样式}  ....  } |

使用动画

|  |
| --- |
| 1. 选中要使用的动画名称   animation-name  animation-duration  animation-timing-function |

动画的其他属性

动画的兼容性

|  |
| --- |
| 低版本的浏览器，定义动画时添加前缀  @-webkit-keyframes |