复习

MySQL中的复杂查询

聚合查询/分组查询

COUNT(\*) / SUM(salary) / AVG(salary) / MAX(salary) /MIN(salary)

SELECT sex,SUM(salary) FROM GROUP BY sex

子查询

多个SQL语句的嵌套，一个SQL的结果，作为另外一个SQL的条件

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

跨表查询/多表查询 —— 表之间要有关联(员工所在部门的编号，和部门表里边的部门编号两者关联)

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

SQL99

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

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

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

全连接 FULL JOIN —— MySQL不支持

UNION 合并相同记录

UNION ALL 不合并相同记录

JS执行环境

客户端：浏览器本身所带的js解释器 —— 兼容性问题

服务器端：nodejs提供的谷歌V8引擎 —— 不存在兼容性问题

JS中的变量

存放信息的容器

var x = 1; var y; //undefined

var x=1, y;

y=2;

变量命名：可以是字母，数字，下划线，美元符号，不能以数字开头

常量

存的值是不变的 const PI = 3.14;

数据类型

学习一门编程语言路线图

(1)了解语言的背景，历史，特点，应用领域，现状

(2)搭建开发环境，编写 hello world

(3)声明变量和常量

**(4)数据类型**

**(5)运算符**

(6)逻辑结构

(7)通用小程序

(8)函数和对象

(9)第三方的类库，插件，组件，框架

(10)开发项目

1.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)、null

(1)数值型

整型和浮点型

八进制(010)，十进制，十六进制(0XF)

3.14E8

(2)字符串型

常用于表示一些文本字符数据，例如：姓名、邮箱、电话、家庭住址

特点：使用引号(单引号或双引号)把数据包含

查看汉字的Unicode编码

'一'.charCodeAt(); //19968

查看汉字的Unicode编码十六进制形式

'一'.charCodeAt().toString(16); // 4e00

查看16进制Unicode编码对应汉字

console.log('\u4e00');

检验汉字范围 4e00 ~ 9fa5

(3)布尔型

在程序中表示真或者假的结果

取值 true/false

常用语表示是否的结果

例如：是否登录，是否注册，是否是会员...

(4)未定义型

只有一个值 undefined

当声明一个变量，未赋值，此时变量的值为undefined

(5)null

在js中，null表示空，常用于释放一个对象。

**检测数据类型**

typeof( 参数 ) 参数表示要检测的数据

结果有number(数值型)，string(字符串型)，boolean(布尔型)，undefined(未定义型)

2.数据类型转换

(1)隐式转换

①数字 + 字符串：数字会被转成字符串

10 + 'kb' //'10kb'

②数字 + 布尔值：布尔值会被转换数字 true->1/false->0

1 + true //2

③布尔值 + 布尔值：布尔值会被转成数字 true->1/false->0

true + false //1

④布尔值 + 字符串：布尔值会被转成字符串

true + 'abc' //'trueabc'

练习：查看输出结果

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

console.log(num1+num2+num3); //'3sttrue'

console.log(num2+num3+num1); //'sttrue3'

console.log(num3+num2+num1); //'truest3'

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

(2)强制转换

①强制转换成整型

parseInt()

parseInt('23.55t'); // 23

从第一位开始找数字，遇到非数字或者小数点，结束，如果第一位是非数字，返回NaN

NaN --> Not a Number 不是一个数字

练习： 转换15a.8 hello1 1hello

②强制转换成浮点型

parseFloat()

parseFloat('23.55t') //23.55

从第一位开始找数字，遇到数字继续往后找，遇到小数点继续往后找，知道遇到非数字就会终止。如果第一位是非数字返回NaN

③转换成数值型

Number()

Number('23.55t') //NaN

Number('23.55') //23.55

如果被转换的值中有非数字，则结果为NaN

④转换成字符串型

toString()

20.toString() // '20'

toString可以将数值型和布尔型的值转换为字符串类型。

3.两个运行在客户端的函数

alert() 弹出警示窗口

prompt() 弹出提示窗口

注意：prompt返回的结果是字符串类型。

练习：弹出两次提示窗口，分别数字，最后打印两个数字相加的结果。

4.运算符和表达式

由运算符连接的操作的数据，这种形式就成为表达式。

(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，然后再执行加1**

console.log(++num); //**先执行加1，然后再打印num的值**

|  |
| --- |
| 练习：以下程序的结果是？  var a = 1;  console.log(a++ + ++a); //a=3  console.log(a++ + ++a + a++) |

(2)比较(关系)运算符

> >= < <= == != ===(全等于) !==(不全等于)

产生的结果是一个布尔型的值(true / false)

等于(==)：比较值是否相同

全等于(===)：比较**值和类型**是否相同，两个都满足结果是true

'3' > 10 //false

比较运算符的两端，只有有一个是数值型，另外一个隐式转换为数值型

'3' > '10' //true

比较运算符的两端，如果都是字符串，比较的是Unicode码值

练习：比较两个中文的大小

'张三丰' > '张无忌'

//三 -> 19977 无 -> 26080

如果第一个字符相同，则继续比较第二个字符。

'3m' > 10

字符串3m会隐式转换为数值型，使用的是Number，结果NaN;

NaN和任何的值比较，结果都是false，即使和自身比较也是false。

(3)逻辑运算符

&& 并且 | | 或者 ! 非

&& 两个条件都满足结果是true

| | 两个条件至少满足其一结果是true

! 反向，逆向；非真为假，非假为真

逻辑中断(短路逻辑)

&& 如果第一个条件为false，整个的结果就是false，第二个条件也没有继续执行的必要了

| | 如果第一个条件为true，整个的结果就是true，第二个条件也没有继续执行的必要了

练习：

|  |
| --- |
| var a = 1,b=2;  a>3 && (a=5);  b<5 | | (b=4);  //a结果是1 b结果是2 |

(4)位运算符(了解)

1 10 11 100 101 110 111 1000 1001 1010 1011

1100 1101

在执行位运算的时候，十进制的值会转成二进制。

按位与(&) 上下两位都是1，结果才是1，否则是0

按位或( | ) 上下两位只要有一个1，结果就是1，否则是0

按位异或(^) 上下两位相同是0，不同是1

按位右移(>>) 删除最后的几位， 例如 3>>1 结果1，数字变小

按位左移(<<) 在末尾添加0，例如3<<1 结果6，数字变大

练习： 10>>2 ->2 5^8 ->13

(5)赋值运算符

= +=(在原来基础之上加) -= \*= /= %=

超过500打八折，600在原来基础之上打八折。

课后练习：

声明一个变量 var year = 2014;

判断年份是否为闰年？

4年一闰(能被4整除)，100年不闰(能被100整除)，400年再闰(又能被400整除)

&& | |