HTML部分：

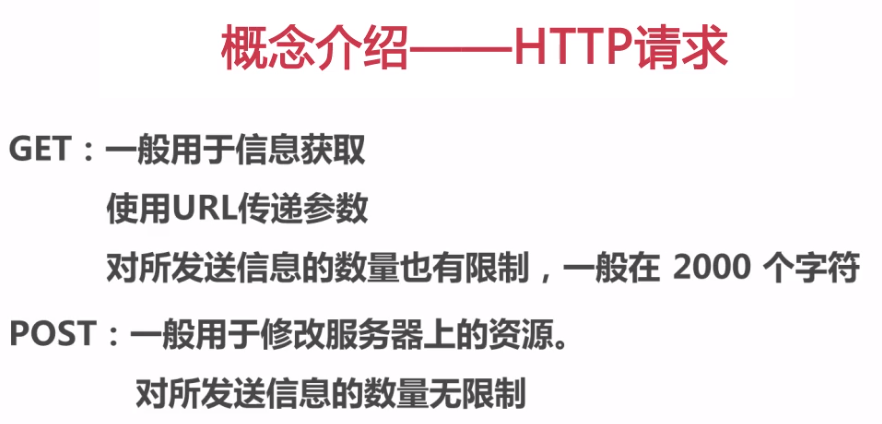
CSS部分：

**AJAX部分**：

1. 创建XMLHttpRequest对象：var request = new XMLHTTPRequest（）；



1. HTTP请求方式简介：

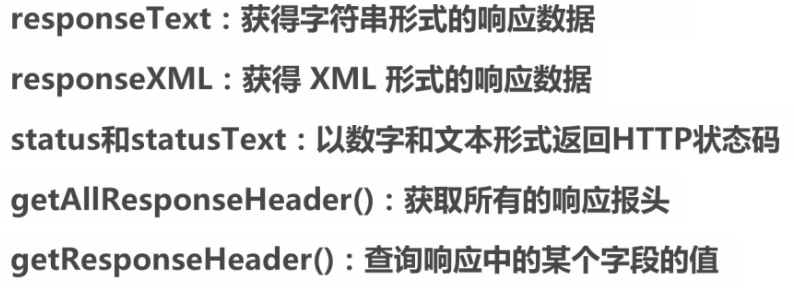


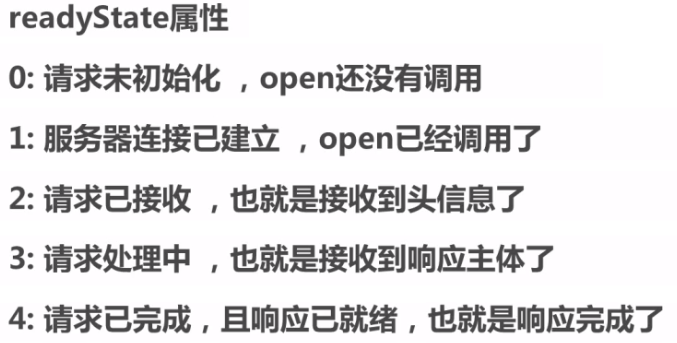
1. XMLHTTPRequest常用函数：



Send(string):发送数据

1. XMLHTTPRequest取得响应：





例子：



**JavaScript部分**：

Js窗口互动函数：

Document.write（）；//输出相应字符

Alert（）； //提示框，点击才可进行其他操作

Confirm（str）；//在消息对话框中显示文本返回值boolean类型

Prompt（str1,str2）；//显示一个提问窗口，有文本框显示内容，str1在对话框中的文本，str2在文本框中的文本

Window.open(URL，窗口名称，参数字符串)；//窗口名称\_blank新窗口显示页面；\_self当前窗口显示;\_top框架网页中在上部窗口中显示页面

Window.close（）；//关闭本窗口

Js原生操作：

Var something = document.getElementById（“id”）/ByTagName/ByClassName

Js中字符串操作函数substr和substring的区别：substr（index，length）//substring（start，stop）【substring中含头不含尾】

Js中math的常用函数：



Js中window.history.back()和window.history.go(-1)区别：back是返回上一页原页面的内容会保留；go是原页面表单中的内容会丢失。

**SQL数据库部分**：

关系模型：是使用二维表实体信息之间关系的模型，包括一对一，一对多，多对多三种关系

E-R图设计：

要以最小模块设计分E-R图，然后消除冲突，包括属性，命名和结构冲突。

|  |  |  |
| --- | --- | --- |
| **名称** | **类型** | **数据范围** |
| Int | 整型 | 4字节int型，范围约2^31 |
| Bigint | 长整型 | 8字节整数类型，范围2^63 |
| Real（别名：float（24）） | 浮点型 | 4字节 |
| Float（m，n）m:个数n:小数位 | 浮点型 | 4字节 |
| Double | 浮点型 | 8字节 |
| Decimal | 高精度小数 |  |
| Char（n） | 定长字符串 | 储存指定长度字符串 |
| Varchar（n） | 变长字符串 | 储存0-n个字符 |
| Boolean | 布尔类型 | True或False |
| Enum(枚举值) | 枚举类型 |  |
| Date | 日期类型 |  |
| Time | 时间类型 |  |
| Datetime | 日期+时间类型 |  |

**SQL操作数据库的能力**：

1. **DDL**：创建，删除，修改表的能力（Data Definition Language）
2. **DML**：增删改的能力(Data Manipulation Language)
3. **DQL**：用户查询数据的能力(Data Query Language)

Tips：SQL语句不区分大小写

**主键**：不能重复且能唯一标识当前记录的字段（注：不能使用任何业务相关的字段作为主键，比如手机号，邮箱这类可变标识）通常采用自增整数类型和全局唯一GUID类型（通过MAC地址，时间戳等随机生成且唯一的标识）

**联合主键**：允许多个字段唯一标识一个记录，允许一列中有重复字符，只要不是所有的主键列都重复即可

**外键**：通过设置一个外键，连接两个数据表，以达到减少数据重复记录的情况。

例：添加外键约束 ALTER TABLE 表名

ADD CONSTRAINT 外键约束名

FOREIGN KEY 指定外键

REDERENCES 关联表 关联列名

删除外键约束 ALTER TABLE 表名

DROP FOREIGN KEY 删除外键名

**MySQL提示符修改**：prompt \D(完整日期)\d(当前数据库)\h(服务器名称)\u(当前用户)

**创建数据库**：CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] [db\_name];

**修改数据库**：ALTER {DATABASE | SCHEMA} [db\_name] [DEFAULT] CHARACTER SET [=] charset\_name(编码方式，默认utf-8);

**删除数据库**： DROP {DATABASE | SCHEMA} {IF EXISTS} db\_name;

**创建数据表**：CREATE TABLE [IF NOT EXISTS] table\_name（column\_name data\_type,…）;

**查看数据表列表**：SHOW TABLES [FROM db\_name];

查看数据表结构（列名定义）：SHOW COLUMNS FROM db\_name;

插入操作：INSERT tb\_name[(col\_name,…)]

记录查找：SELECT \* FROM tb\_name；

自动编号，必须和主键组合使用，起始为1，增量为1 => AUTO\_INCREMENT

主键约束：PRIMARY KEY（一表一个）

唯一约束：UNIQUE KEY（一表可多个）

设置默认值：例(ENUM(‘1’,’2’,’3’) DEFAULT ‘3’)（枚举值中默认值为3）；

外键约束：FOREIGN KEY (id) REFERENCES tb\_name (id);

**外键约束的参照操作**：

1. CASECADE：从父表中删除或更新且自动删除或更新子表中匹配的行；
2. SET NULL:从父表中删除或更新行，并设置子表中的外键列为null，此项必须保证字表列中没有指定not null；
3. RESTRICT：拒绝对父表的删除或更新操作；
4. NO ACTION：和RESTRICT相同（标准SQL关键字）。

**修改数据表**：(添加单列)ALTER TABLE tb\_name ADD [COLUMN] col\_name column\_definition [FIRST|AFTER col\_name]

(添加多列)ALTER TABLE tb\_name ADD [COLUMN] (col\_name column\_definition,…)

(删除列)ALTER TABLE tb\_name DROP [COLUMN] col\_name;(TIPS:多列删除同上)

(添加主键约束)ALTER TABLE tb\_name ADD [CONSTRAINT [symbol]] PRIMARY KEY [index\_type] (index\_col\_name,…)

(添加主键约束)ALTER TABLE tb\_name ADD [CONSTRAINT [symbol]] UNIQUE [INDEX | KEY] [index\_type] (index\_col\_name,…)

(添加外键约束)ALTER TABLE tb\_name ADD [CONSTRAINT [symbol]] FOREIGN KEY [index\_name](index\_col\_name,…) referencce\_definition

(添加/删除默认约束)ALTER TABLE tb\_nsme ALTER [COLUMN] col\_name {SET DEFAULT literal}DROP DEFAULT;

(修改列定义)ALTER TABLE tb\_name MODIFY [COLUMN] col\_name column\_definition [FIRST | AFTER col\_name]

(修改列名称)ALTER TABLE tb\_name CHANGE [COLUMN] old\_col\_name new\_col\_name column\_definition [FIRST | AFTER col\_name]；

(插入记录)INSERT [INTO] tb\_name [(col\_name,…)] {VALUES | VALUES} (可插入多条记录，用逗号隔开) || INSERT [INTO] tb\_name SET col\_name = ‘ ’;

(插入或者替换)是insert记录的升级版 REPLACE语句，记录存在则替换，记录不存在则添加

REPLACE INTO tb\_name(col\_name) VALUES (col\_value);

(更新记录)UPDATE [LOW\_PRIORITY] [IGNORE] tb\_reference SET col\_name={EXPR1 | default}[col\_name2 = ….]…[WHERE where\_condition];

(插入或更新)升级版如果记录存在就更新此记录INSERT INTO ... ON DUPLICATE KEY UPDATE ...

例：INSERT INTO students (id, class\_id, name, gender, score) VALUES (1, 1, '小明', 'F', 99) ON DUPLICATE KEY UPDATE name='小明', gender='F', score=99;

解析：若id=1的记录不存在，INSERT语句将插入新记录，否则，当前id=1的记录将被更新，更新的字段由UPDATE指定

(删除记录)DELETE FROM tb\_name [WHERE where\_condition]

(查找记录)SELECT select\_expr [,select\_expr…]

[

FROM tb\_reference

[WHERE where\_condition]

[GROUP BY {col\_name | position} [ASC | DESE],…]

[HAVING where\_condition]

[ORDER BY {col\_name | expr | position} [ASC | DESC],…]

[LIMIT {[offset,] row\_count | row\_count OFFSET offset}]

]

(为列名设置别名)SELECT name **AS** userId,userage AS usergroud FROM tb\_name;

**查询过滤条件**：根据条件/WHERE where\_condition

按照条件将查询结果分组/GROUP BY {col\_name | position} [ASC | DESC],..

分组条件HAVING where\_condition

对查询结果排序ORDER BY {col\_name} [ASC | DESC](升降序)

限制查询结果返回数量 LIMIT {[offset,] row\_count | row\_count OFFSET offset}

**子查询**：出现在其他SQL语句中的select语句，要用（）进行识别，且子查询和正常select语句一样可以进行distinct，group by，order by，limit等查询条件

**使用比较运算符**进行子查找，使用ANY，SOME或者ALL修饰比较运算符。



**使用[NOT] IN**的子查询：=ANY和NOT IN等效；！= ALL或者<>ALL与NOT IN等效；

**将查询结果写入数据表**：INSERT [INTO] tb\_name [(col\_name,..)] SELECT…

**多表更新**：UPDATE table\_references

SET col\_name = {expr1 | DEFAULT} (可多个col\_name)[WHERE where\_condition];

**多表连接类型**：INNER JOIN/CROSS JOIN/JOIN(内连接) || LEFT [OUTER] JOIN左外连接；RIGHT [OUTER] JOIN 右外连接；(添加连接条件是用on)

**创建数据表同时将查询结果写入到数据表**：CREATE TABLE [IF NOT EXISTS] tb\_name [create\_definition,…]

Select\_statement

**连接条件**：使用ON设定连接条件，使用where进行结果集的过滤。

**内连接**是指查找两个表的交集部分数据相关的记录，且仅显示连接条件相等的记录，多余记录不显示。

**左（右）外连接**显示左（右）表的全部记录以及右（左）表符合连接条件的记录。

Tips：多表连接通过多个inner join来实现

多表删除：DELETE tb\_name[.\*][,tb\_name[.\*]]…

FROM tb\_references

[WHERE where\_condition]

**字符函数**：



比如保留小数位数



MySQL中字符起始位置是1，而不是0；

数值运算符与函数：



比较运算符与函数：

日期时间函数：



信息函数：

聚合函数：（聚合查询中只能加入被分组列）



自定义函数：有参数，有返回值；

创建自定义函数：CREATE FUNCTION function\_name

RETURNS {STRING |INTEGER |REAL |DECIMAL} routine\_body

函数体由合法的SQL语句构成；

函数体可以是简单的select或者insert语句；

**如果是复合结构就是用begin…end语句；**

复合结构可以包含声明，循环，控制结构。

*函数报错解决：创建函数时报错This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled (you \*might\* want to use the less safe log bin trust function creators variable*

*错误原因：这是我们开启了 bin-log, 我们就必须指定我们的函数是否是*

*1 DETERMINISTIC 不确定的*

*2 NO SQL 没有SQl语句，当然也不会修改数据*

*3 READS SQL DATA 只是读取数据，当然也不会修改数据*

*4 MODIFIES SQL DATA 要修改数据*

*5 CONTAINS SQL 包含了SQL语句*

*开启bin-log就一定要为函数传入一个参数*

*解决办法：set global log\_bin\_trust\_function\_creators = TRUE;*

**DELIMITER // :用//代替；称为语句结束符**

创建存储过程：

CREATE

[DEFINER = {user | CURRENT\_USER}]

PROCEDURE sp\_name([proc\_parameter[,…]])

[characteristic …] routinr\_body

**Tip:proc\_parameter:[IN | OUT | INOUT] param\_name type**

IN:表示该参数在调用存储过程时指定

OUT:表示该参数可以被存储过程改变，且可以返回

INOUT：表示该参数在调用时指定，并可以被改变和返回

调用存储过程：CALL 存储过程【没有参数可以不带小括号，有参数一定要带小括号】；

删除存储过程：DROP PROCUDURE [IF EXISTS] sp\_name；

存储过程和自定义函数的区别：前者单独调用，后者如同内置函数可以随SQL语句调用；

前者可以返回多个参数，后者只能返回一个参数

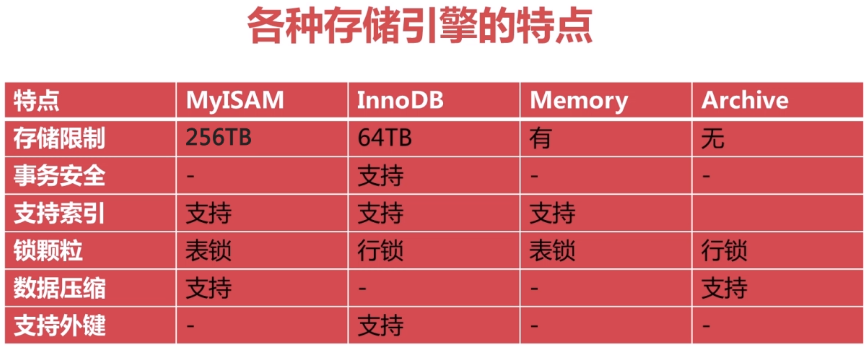
前者可以封装较为复杂的功能，方便调用（经常对表做操作，增删改查之类的），后者针对性更强一点（很少对表做操作，只是对表中数据做处理）。

Mysql支持的存储引擎：MyISAM，InnoDB，Memory，CSV，Archive

并发控制：多个连接对记录进行修改时保证数据的一致性和完整性

使用锁系统：共享锁（读锁）/排它锁（写锁）；

锁颗粒：表锁（开销最小的锁策略）/行锁（开销最大的锁策略）

分页显示查询结果：使用LIMIT <M> OFFSET <N>

其中LIMIT 总是设定pagesize，offset计算公式则为pagesize \* （pageindex - 1）

注意offset设置的值如果超过最大数量不会报错，但是会返回一个空集合。

数据库事务具有ACID四个特性：A（atomic）原子性；C(Consistent) 一致性；I(Isolation)隔离性；D(Duration)持久性

事务隔离级别：（innoDB中默认等级为Repeatable Read）

1.隔离级别最低（Read Uncommitted）：带来影响[一个事务会读到另一个事务更新后但未提交的数据，如果另一个事务回滚，那么当前事务读到的数据就是脏数据，这就是脏读（Dirty Read）]

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

2.在Read Committed隔离级别下，一个事务可能会遇到不可重复读（Non Repeatable Read）的问题，不可重复读是指，在一个事务内，多次读同一数据，在这个事务还没有结束时，如果另一个事务恰好修改了这个数据，那么，在第一个事务中，两次读取的数据就可能不一致。

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

3.在Repeatable Read隔离级别下，一个事务可能会遇到幻读（Phantom Read）的问题。幻读是指，在一个事务中，第一次查询某条记录，发现没有，但是，当试图更新这条不存在的记录时，竟然能成功，并且，再次读取同一条记录，它就神奇地出现了。

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

4.Serializable是最严格的隔离级别。在Serializable隔离级别下，所有事务按照次序依次执行，因此，脏读、不可重复读、幻读都不会出现。虽然Serializable隔离级别下的事务具有最高的安全性，但是，由于事务是串行执行，所以效率会大大下降，应用程序的性能会急剧降低。如果没有特别重要的情景，一般都不会使用Serializable隔离级别。

**VUE.JS部分：**

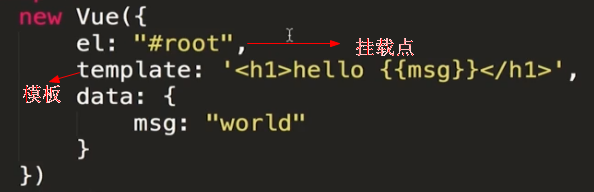
v-html：输出html代码

v-bind：用于绑定数据和元素属性；v-bind:属性名=””;简写成 **:**属性名=“”;

v-model:指令用来在 input、select、textarea、checkbox、radio 等表单控件元素上创建双向数据绑定，根据表单上的值，自动更新绑定的元素的值。

v-on：向指定元素中绑定一个事件

挂载点，模板，实例之间的关系：



实例就是指vue会根据挂载点，和给出的模板和数据自动生成实例显示在页面中

{{message}}：插值表达式 // v-text=“”：是一个指令，后面跟冒号中的数据内容 // v-html=“”：也是一个指令，后面会跟html语句中的内容

例如：v-text=“content” 🡺 data：{<h1>hello</h1>}

v-html=“content” 🡺 data：{<h1>hello</h1>}

输出结果将不尽相同

GIT版本管理：

1. git add:把文件加入仓库
2. git commit（-m “somestring”）：把文件提交到仓库
3. git status：掌握仓库的状态
4. git diff 文件名：查看文件修改记录
5. git log：查看文件提交历史记录（git log –pretty=oneline）简化显示
6. git reset –hard HEAD^（^多少表示回滚的次数（版本数））
7. git reset –hard 版本号（commit id）：可以回到之前提交过的当前版本之后的版本（可以用git reflog查看命令记录）
8. rm files：删除一个文件
9. git rm files:删除版本区文件
10. git checkout – files:回退到删除的文件（已经提交到版本区才可以回退）
11. git push origin master:向远程仓库推送本地仓库信息
12. git checkout -b dev（新版 git switch -c dev）：创建分支；git branch：查看分支；git checkout master（回到某一个分支）//git switch master（直接切换到已有的分支）
13. git merge dev（合并某个分支）//git branch -d dev（删除某个已经合并过的分支）
14. 查看远程库信息，使用git remote -v；
15. 本地新建的分支如果不推送到远程，对其他人就是不可见的；
16. 从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；
17. 在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地和远程分支的名称最好一致；
18. 建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；
19. 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。
20. 命令git tag <tagname>用于新建一个标签，默认为HEAD，也可以指定一个commit id；
21. 命令git tag -a <tagname> -m "blablabla..."可以指定标签信息；
22. 命令git tag可以查看所有标签。
23. 命令git push origin <tagname>可以推送一个本地标签；
24. 命令git push origin --tags可以推送全部未推送过的本地标签；
25. 命令git tag -d <tagname>可以删除一个本地标签；
26. 命令git push origin :refs/tags/<tagname>可以删除一个远程标签。