HTML5存储

# **关于浏览器端的存储**

## 关于存储

Cache 磁盘文件 数据库 内存

## H5的存储

### **目标**

1、解决4k的大小问题

2、解决请求头常带存储信息的问题

3、解决关系型存储的问题

4、跨浏览器

### H5的几种存储：本地存储(localstorage && sessionstorage)、离线缓存、indexDB 和Web SQL

1 本地存储：存储形式为key→value，永久存储，永不失效，除非手动删除，官方给出的文档，每个域名5M,

sessionStorage关闭窗口即过期。



使用方法：getItem setItem removeItem key clear，例如，localstorage.clear(),localstorage.key(0)

存储内容：数组、json数据、图片、脚本、样式文件、字体

使用注意事项：

1. 使用前要判断浏览器是否支持
2. 写数据时候，需要异常处理，避免超出容量抛错
3. 避免把敏感信息写入本地存储
4. Key唯一性

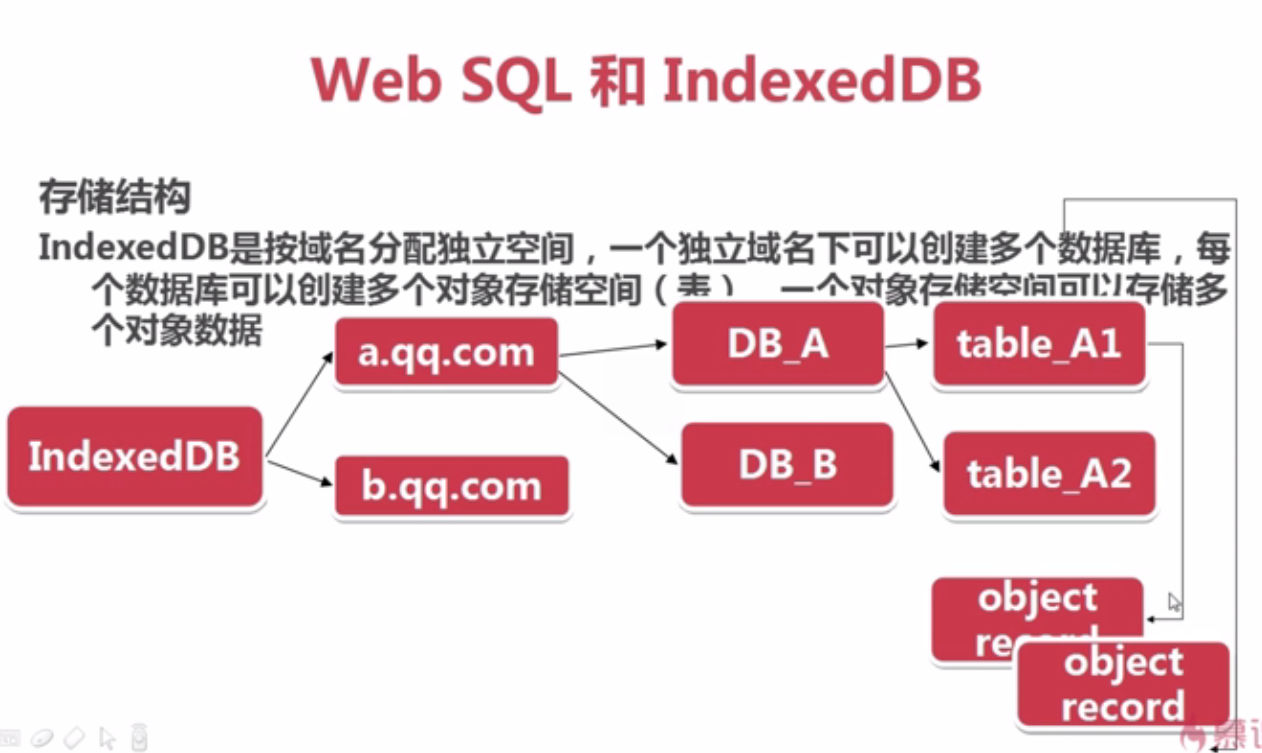
使用限制：

1. 存储更新策略，过期控制
2. 子域名之间不能共享存储数据
3. 超出存储大小之后如何存储（LRU，FIFO）
4. Server端如何取到

使用场景：

1. 利用本地数据，减少网络传输
2. 弱网络环境下，高延迟，低带宽，尽量把数据本地化

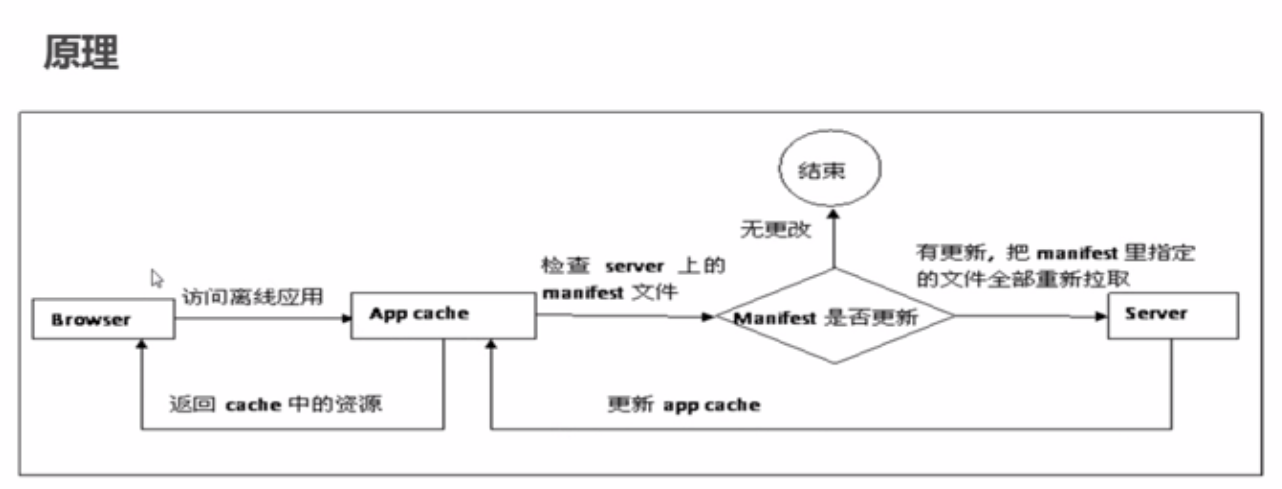
2 Web SQL 和 IndexedDB(Web SQL)



1. 离线缓存（offline application）

让web应用在离线的情况下继续使用，通过manifest文件指明需要缓存的资源

检测是否在线：navigator.onLine true → 在线





## H5离线缓存应用

1 在HTML页面中引用manifest文件

<html manifest=”sample.appcache”>

2 在服务器mime-type文件 添加 text/cache-manifest appcache（自行规定）

3 优势：完全离线 资源被缓存，加载更快 降低server负载

4 使用缺陷

1. 含有manifest属性的当前请求无论如何都会被缓存（html/php..页面）；
2. 更新需要建立在manifest文件的更新，文件更新后是需要页面再次刷新的（需要2次刷新才能获取新资源）；
3. 更新是全局性的，无法单独更新某个文件（无法单点更新）；
4. 对于链接的参数变化是敏感的，任何一个参数的修改都会被（master）重新缓存（重复缓存含参页面）index.html和index.html？renew=1会被认为是不同文件，分别缓存

5 使用场景：

1. 单地址的页面；
2. 对实时性要求不高的业务；
3. 离线webapp

## 其他客户端存储

UserData(IE) google Gears（Chrome）

## 总结

### H5存储优势

存储空间大 接口丰富 省流量 关系型 数据相对安全

### H5存储劣势

浏览器兼容 同源 脚本控制 更新策略