三、

在 js 中我们把一部分信息存储到到“当前浏览器”的“某一个域/源”中的某个位置，浏览器关闭后或者页面关闭，信息是不消失的，以后用该浏览器重新打开页面依然可以从上一次存储信息的位置把内容得到；

本地存储可以实习不同页面之间的通信，在 a 页面中操作的一些信息，我们将其存储到本地，必要的时候，我们可以从本地再把本地信息获取；（还有页面跳转的时候 url 地址栏的问号传参其实也是在实现页面和页面之间的传参）；

在当前市场主流存储方式：cookie、webStorage（localStorage/sessionStorage）==》你们之前听过的 session/数据库都是在服务器存储；

谷歌浏览器写的 cookie 火狐是得不到的，京东的 cookie 百度都不到；

四、cookie pk webstorage

cookie 兼容所有浏览器，webStorage 是 h5 提供方法不兼容 ie8-；

无论是 cookie 还是 webStorage 都可以在谷歌浏览器控制台的 Resources 选项中查看到；没有 resources 用 application 选项查看；

使用语法：

//：cookie：

1）通过 document.cookie="xx=8"设置，用 document.cookie 是获取

2）修改 document.cookie="xxx=sss"修改(获取的时候还要用正则匹配到我们需要的结果)；

3）删除的原理：设置 cookie 信息存储的时间是今天以前的一个时间，这样即可 cookie 就过期了，浏览器会自动删除

//webStorage：

localStorage：本地存储，不论浏览器关闭与否或关闭页面，本地存储都不会消失；只有 key 和 value

1）设置和修改，如果原有就是修改，没有就是设置；localStorage.setItem("name":"xxx");如果是做移动端开发，我们使用 cookie 或者 localStorage 都可以，移动端或者不需要考虑兼容的 pc 端，我们使用 localStorage 还可以做一个数据缓存的优化；

2）获取：localStorage.getItem([key],[value]);

3）移除 localStorage.remove("name")

4）localStorage.clear();把当前源/域下所有的使用 localeStorage 存储的内容都移除

5）localStorage.key([index])通过索引获取指定位置的 key（属性名！！不是属性值）；

sessionStorage：本地会话存储，只要页面关了，证明会话结束了，浏览器会把信息删除，游戏类，有网页聊天的常用

****

cookie 和 localStorage 是明文存储，所以用他们做存储的时候注意一下原则

1）不要存储过于重要的东西（用户名和账户，银行卡密码账户，手机号，身份证，）

->如必要时，我们需要给信息加密，一种是不可逆转加密（常用的是 MD5 加密库），例如密码，一种是可逆转加密：例如手机号，银行卡号，用户名等（自己扩展可逆转加密，每个公司都用自己的加密库）；

cookie PK localStorage

*cookie

->cookie 兼容所有浏览器;比如说购物车这种东西，你不能让 ie 低版本没有购物车啊

->cookie 存储的大小有限制，一般同源下最多存储 4kb

->cookie 有过期时间（过期时间可以自己设置，一般不要超过一个月）

->我们使用 360 安全卫士或者浏览器自带的清理垃圾的程序，很多时候会把 cookie 清理掉，用户可能出于安全的角度禁

止存储 cookie，无痕浏览，隐身模式

->cookie 不是单纯的本地存储，它的相关操作和服务器还有千丝万缕的联系；

->

->

*localStorage

->localStorage 不兼容 ie8-；

->localStorage 也会存在大小限制，一般同源下只能存储 5MB

->localStorage 不存在过期时间，只要我们不删除就会永久存在本地；

->但是安全卫士和浏览器清除功能不能清 localStorage，但是无痕浏览对 localStorage 没有任何影响

->localStorage 是正统的本地存储和服务器没有任何关系的；

->可以做性能优化，例如 css，js 源代码，以及一些不需要经常更新的数据(设置一个过期时间，以后每次刷新页面，看一下是不是超过过期时间了，如果超过了，请求最新的数据，如果没有超过，从本地读取)

//

项目实战思想

1）自动登陆记住用户名密码

第一次登陆的时候我们首先获取用户输入的用户名和密码，然后客户端想服务器发送请求，把用户名和密码(这时候的密码已经被加密)传递给服务器，为了保证安全，我们使用 post 请求；--》服务器接收到客户端传递过来的结果，到数据库中查看当前信息，如果存在就登陆成功，服务器把登陆成功或者失败的信息返回给客户端(如果登陆成功除返回状态，还需返回部分用户信息，如用户名、头像等)，客户端接收到这个响应，如果登陆成功，也获取到了部分信息；第一次登陆成功的话，我们需要把服务器返回的部分用户信息存储到本地，其中包含了用户名和密码(密码也是加密的)，第二次登陆的时候，从 cookie 获取到用户名和密码，然后把用户名和密码分别的显示到文本框中，但是密码是经过加密的，我们需要随便往密码框中输入一部分内容，而真正的密码保存在某一个变量中即可，接下来：如果用户没有修改密码，我们点击登陆的时候，我们获取页面中的用户名和之前保存的密码(不是文本框中的密码)，重新验证密码；如果用户把密码框中的内容修改了，我们需要获取文本框中最新的用户名和密码走第一次登陆的流程；本次用户名和密码如果登陆成功后，那么把之前的 cookie 信息修改；

2）注册