其他篇：

**Git**

git helper -a # 查看全部git子命令

git -version # 查看git版本

git checkout -b "新建分支名" # 创建一个分支并切换到新创建的分支

git branch "新建分支名" # 创建分支

git branch # 查看本地所有分支

git branch -r # 查看远程所有分支

git branch -a # 查看本地和远程分支

git branch -d "某分支名" # 删除某分支

git checkout "某分支名" # 切换到某分支

git switch "某分支名" # 切换到某分支

git status # 查看状态

git add "文件名" # 将某个文件存入暂存区

git add . # 将所有文件存入暂存区

git commit -m "备注信息" # 提交到仓库

git diff # 查看变更 工作区和暂存区的差别比对

git push <远程主机名><本地分支>:<远程分支> # 完整写法

git push origin master # 将本地的master分支推送到远程的master分支，如果master不存在 则会创建master分支

git push <新建分支名> # 将新建分支推送到远程分支

git push origin :master # 如果忽略本地分支，则推送了一个空分支，相当于删除了分支。

# 等同于git push origin --delete <要被删除的分支名>

git push # 将本地分支推送到远程分支，如果当前分支和远程分支之间存在追踪关系，则本地分支和远程分支都可以省略

git push origin --delete <要删除的分支名> # 删除远程分支

git fetch origin master # 将远程分支下拉到本地

git pull <远程分支地址> <远程分支名>:<本地分支名> # 完整写法

# 例如，git pull origin next : master .将远程分支与next分支合并。

git pull origin master # 获取远程分支,并于当前分支合并

git fetch origin master # 获取远程分支master到本地,不合并

git clone "远程地址" # 将远程分支克隆到本地

git merge "分支名" # 把现有分支合并到分支上

git reset HEAD file # 文件 add后,撤销修改

git remote add origin git项目地址 # 本地仓库和远程仓库建立连接

git remote -v # 查看远程关联的地址

git remote remove origin # 移除远程关联

**跨域：**

**1.**同源策略：

- 协议相同

- 域名相同

- 端口相同

**2.jsonP原理** ：就是利用 标签没有跨域限制的漏洞。通过 标签指向一个需要访问的地址并提供一个回调函数来接收数据当需要通讯时。

//JSONP 使用简单且兼容性不错，但是只限于 get 请求。

1. **CORS** ： 需要浏览器和后端同时支持。IE 8 和 9 需要通过 XDomainRequest 来实现。浏览器会自动进行 CORS 通信，实现 CORS 通信的关键是后端。只要后端实现了CORS，就实现了跨域。

XSS：跨站脚本攻击，指黑客往HTML文件或者DOM中注入恶意脚本，从而在用户浏览页面时利用注入的恶意脚本对用户实施攻击的一种手段

常见的注入方式：

一、存储型 XSS 攻击：

1.利用站点漏洞将一段恶意代码提交到 网站数据库中

2.用户向网站请求包含恶意代码的页面

3.当用户浏览该页面的时候，恶意脚本就会将用户的 Cookie 信息等数据上传到服务器

二、反射型 XSS 攻击：反射型 XSS 攻击过程中，恶意脚本属于用户发送给网站请求中的一部分，随后网站又把恶意脚本返回给用户，当恶意脚本在用户页面被执行时，就可以利用该脚本做一些恶意操作。

解决方法：

1.服务器对输入脚本进行过滤或转码

2.充分利用 CSP

限制加载其他域下的资源文件

禁止向第三方域提交数据

禁止执行行内脚本和未授权的脚本

提供上报机制，帮助尽快发现 XSS 攻击，以便尽快修复问题

3.使用 HttpOnly 属性来保护重要的 Cookie 信息（无法通过脚本读取 Cookie）

4.通过添加验证码防止脚本冒充用户提交危险操作，对于一些不受信任的输入，可以限制其输入长度，这样可以增大 XSS 攻击的难度。

//xss.js 插件

```

```

CSRF 攻击就是黑客利用用户登录状态，并通过第三方的站点来做一些坏事。

和 XSS 不同的是，CSRF 攻击不需要将恶意代码注入用户页面，仅仅利用服务器的漏洞和用户登录状态来实施攻击

解决方法：

主要的防护手段是提升服务器的安全性

加Token 验证

隐藏令牌：把 token 隐藏在 http 的 head头中

1. **请求码**

**- 200：服务器已经成功处理了请求。**

**- 3XX：表示重定向。**

**- 401：未授权，服务器请求身份验证。**

**- 404：服务器找不到请求的网页。**

**- 500：服务器内部错误，无法完成请求。**

**- 503：服务器目前无法使用。**

1. **哪些操作会造成内存泄露**（至少三个）

# Cooike，****sessionStorage，**localStorage**

1. 浏览器缓存 强缓存 协商缓存

从缓存位置上来说分为四种，并且各自有优先级，当依次查找缓存且都没有命中的时候，才会去请求网络。

Service Worker

Memory Cache

Disk Cache

Push Cache

缓存策略

强缓存：不会向服务器发送请求，直接从缓存中读取资源，

协商缓存：就是强制缓存失效后，浏览器携带缓存标识向服务器发起请求，由服务器根据缓存标识决定是否使用缓存的过程