# Js

## this指向问题

在 JavaScript 中 this 不是固定不变的，它会随着执行环境的改变而改变。**this的最终指向的是那个调用它的对象**

* 在方法中，this 表示该方法所属的对象。
* 如果单独使用，this 表示全局对象。
* 在函数中，this 表示全局对象。
* 在函数中，在严格模式下，this 是未定义的(undefined)。
* 在事件中，this 表示接收事件的元素。
* 类似 call() 和 apply() 方法可以将 this 引用到任何对象。

阮一峰：<http://www.ruanyifeng.com/blog/2018/06/javascript-this.html>

讲解：<https://juejin.cn/post/6844903462984155143>

注意：在箭头函数中this的指向问题

<https://m.html.cn/qa/javascript/20201.html>

## call,bind,apply

理解：<https://www.cnblogs.com/pssp/p/5215621.html>

## Js 异步执行

理解：<https://www.jianshu.com/p/b8234b3314c8>

<https://www.imooc.com/article/20580?block_id=tuijian_wz>

## 数组去重

Indexof判断

双指针

## 闭包

**闭包是指有权访问另一个函数作用域中的变量的函数**

**闭包内部的变量是会一直保存着不被销毁的**

## 作用域和作用域链

<https://juejin.cn/post/6844904099012411399#heading-1>

## 堆栈类型释放

## 原型和原型链

## 多种继承方式

## 正则表达式

常用的一些语法：

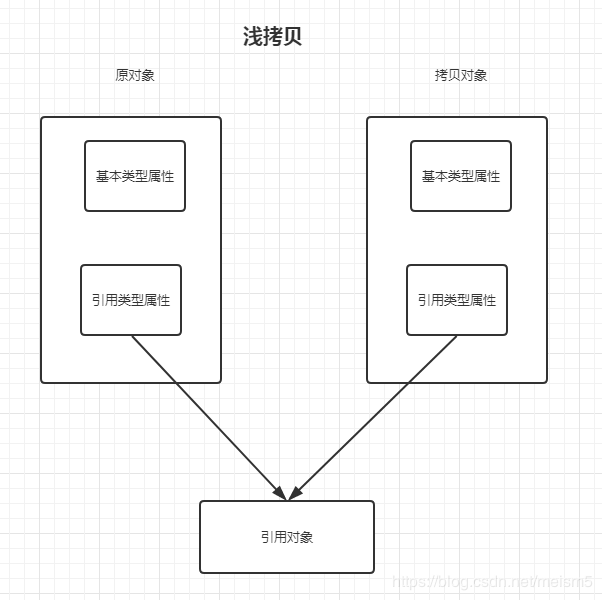
<http://c.biancheng.net/view/5632.html>

<https://www.runoob.com/regexp/regexp-syntax.html>

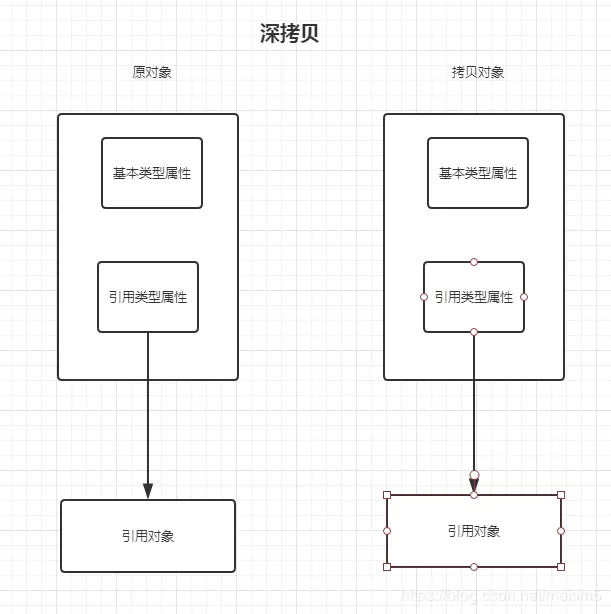
## 深浅拷贝

<https://juejin.cn/post/6844904004611211271>

1. 什么是深拷贝、浅拷贝



浅拷贝就是使用一个变量去拷贝一个引用类型在栈中的内存地址



深拷贝就是指在堆空间中，另外开辟出一个空间，并把原有的堆空间中保存的引用类型的值拷贝到新的空间中来

1. 手写深拷贝
2. JSON.parse(JSON.stringify((obj))

Let obj={

reg : /^reg$/,

fun: function(){},

syb:Symbol(‘foo’),

undefined:undefined

};

Let copyObj=JSON.parse(JSON.stringify(obj));

Console.log(copyObj);//{reg:{}}

缺点：

1. 会忽略 undefined
2. 会忽略 symbol
3. 不能序列化函数正则对象等特殊对象
4. 不能处理指向相同引用的情况，相同的引用会被重复拷贝
5. 递归(遍历对象、数组直到里边都是基本数据类型，然后再去复制，就是深拷贝)

function deepClone(obj){

let cloneObj;

// 判断当输入的数据是简单数据类型时，直接复制

if(obj && typeof obj !== 'object'){

cloneObj = obj;

}

// 当输入的数据是对象或者数组时

else if(obj && typeof obj === 'object'){

// 检测输入的数据是数组还是对象

cloneObj = Array.isArray(obj) ? [] : {};

// 变量数据对象

for(let key in obj){

// 判断对象是否存在key属性

if(obj.hasOwnProperty(key)){

if(obj[key] && typeof obj[key] === 'object'){

// 若当前元素类型为对象时，递归调用

cloneObj[key] = deepClone(obj[key]);

}

// 若当前元素类型为基本数据类型

else{

cloneObj[key] = obj[key];

}

}

}

}

return cloneObj;

}

## evenLoop



## 垃圾回收机制

* 在Javascript中，如果一个对象不再被引用，那么这个对象就会被GC回收。
* 如果两个对象互相引用，而不再被第3者所引用，那么这两个互相引用的对象也会被回收。因为函数a被b引用，b又被a外的c引用，这就是为什么函数a执行后不会被回收的原因。

在 JS 中，闭包存在的意义就是让我们可以间接访问函数内部的变量。[[1]](#endnote-1)

## 跨域

<https://blog.csdn.net/qq_38128179/article/details/84956552>

JSONP

在HTML标签里。如script,img这种获取资源的标签时没有跨域限制的，利用这个点

CORS

Document.domain

postMessage

nginx代理跨域

WebSocket协议跨域

# Vue

## 组件间通信

<https://juejin.cn/post/6844904046285815816>

<https://segmentfault.com/a/1190000019208626>

### props/emit

父组件A 向 子组件B 传递数据 通过props的方法

子组件B 向 父组件A 发送数据 通过emit

* 父组件向子组件传递：



说明：

父组件<coms-son :sons=”sonlist”>//自定义名称sons便于子组件调用，引号里面内容为传递的值. :自定义事件名=“传递的值”

子组件遍历传递过来的值，props里内容sons为父组件中子标签自定义的名字

**注意：**

props 只可以从上一级组件传递到下一级组件，也就是父子组件，即这就是单向数据流

props是只读，不可以被修改，所有被修改都会失效和被警告

* 子组件向父组件传递(通过事件形式)：





说明：

子组件绑定一个点击事件，在方法中自定义点击事件onEmitIndex，并且传递值index.（“自定义事件”，传递的值）

父组件接受onEmitIndex事件，且声明函数接受传递过来的值。“与子组件自定义事件名称保持一致=“函数（$event）接受传递的值”

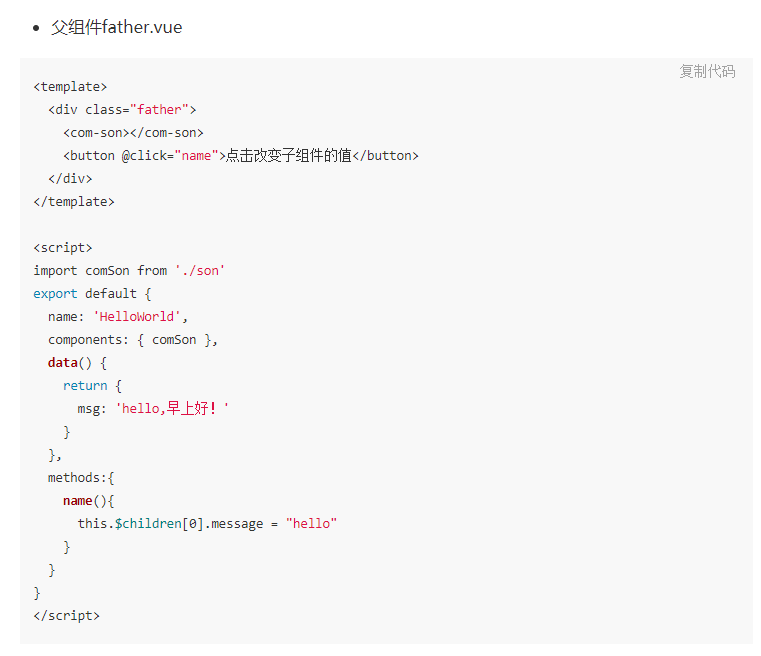
### parent/children与ref

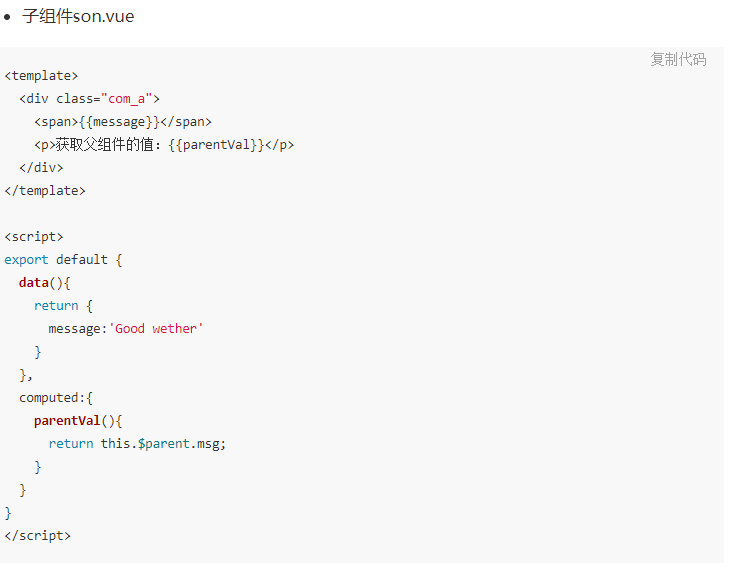
ref：如果在普通的 DOM 元素上使用，引用指向的就是 DOM 元素；如果用在子组件上，引用就指向组件实例

$parent / $children：访问父 / 子实例

需要注意的是：这两种都是直接得到组件实例，使用后可以直接调用组件的方法或访问数据。

* 子实例可以用this.$parent访问父实例
* 子实例被推入父实例的$children
* ref：如果在普通的 DOM 元素上使用，引用指向的就是 DOM 元素；如果用在子组件上，引用就指向组件实例



ref 用于访问子组件实例或子元素：

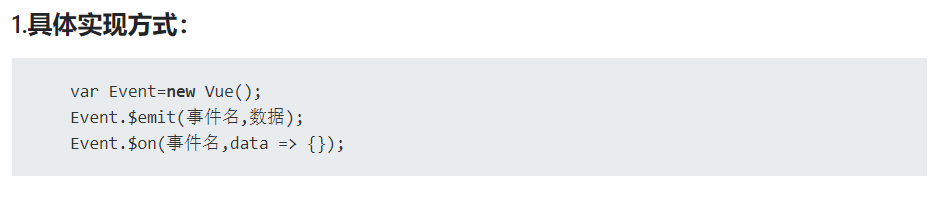


注意：

这两种方法的弊端，无法在跨域兄弟间通信

### $emit/$on

这种方法通过一个空的Vue实例作为中央事件总线（事件中心），用它来触发事件和监听事件,巧妙而轻量地实现了任何组件间的通信，包括父子、兄弟、跨级。



例子：







### Vuex



例子：

<https://juejin.cn/post/6844904046285815816#heading-4>

### $attrs/$listeners

用在父组件传递数据给子组件或者孙组件

注意：

如果仅仅是传递数据，就用$attrs/$listeners好点，如果不仅传递数据，还做中间处理，就用vuex好些

* $attrs：包含了父作用域中不被 prop 所识别 (且获取) 的特性绑定 (class 和 style 除外)。当一个组件没有声明任何 prop 时，这里会包含所有父作用域的绑定 (class 和 style 除外)，并且可以通过 v-bind="$attrs" 传入内部组件。通常配合 interitAttrs 选项一起使用。
* $listeners：包含了父作用域中的 (不含 .native 修饰器的) v-on 事件监听器。它可以通过 v-on="$listeners" 传入内部组件

### provide/inject

需要注意的是：provide 和 inject 绑定并不是可响应的。这是刻意为之的。然而，如果你传入了一个可监听的对象，那么其对象的属性还是可响应的----vue官方文档



### 总结





## Vue生命周期

<https://segmentfault.com/a/1190000011381906>

<https://segmentfault.com/a/1190000008010666?utm_source=sf-related>

vue.js的生命周期一共有10个：

beforeCreate：实例初始化之后，this指向创建实例，不能访问到data、computed、watch、method上订单方法和数据

created：实例创建完成，可访问data、computed、watch、method上的方法和数据，未挂载到DOM，不能访问到$el属性，$ref属性内容为空数组

beforeMount：在挂载开始之前被调用，beforeMount之前，会找到对应的template，并编译成render函数

mounted：实例挂载到DOM上，此时可以通过DOMAPi获取到DOM节点，$ref属性可以访问

beforeUpdate：响应式数据更新时调用，发生在虚拟DOM打补丁之前

updated：虚拟DOM重新渲染和打补丁之后调用，组件DOM已经更新，可执行依赖于DOM的操作

activated：keep-alive开启时调用

deactivated：keep-alive关闭时调用

beforeDestroy：实例销毁之前调用。实例仍然完全可用，this仍能获取到实例

destroy：实例销毁后调用，调用后，Vue实例指示的所有东西都会解绑定，所有的事件监听器会被移除，所有的子实例也会被销毁

# Http

<https://juejin.cn/post/6844904100035821575#heading-91>



## 用户打开网址过程

request请求阶段

1、URL 地址解析

2、DNS 域名解析

3、和服务器建立 TCP 连接

4、发送 HTTP 请求

response 响应阶段

5、服务器得到并处理请求

断开连接

6、和服务器断开 TCP 连接

渲染页面

7、客户端渲染服务器返回的内容

### UCL地址解析

1. 协议：（http://）

常用的传输协议：

* http 超文本传输协议：除了传递文本，还可以传递媒体资源文件（或者流文件）及XML格式数据
* https 更加安全的http：一般涉及支付的网站都要采用https协议（s:ssl 加密传输）
* ftp 文件传输协议：一般应用于把本地资源上传到服务器端

1. 域名：（[www.baidu.cn](http://www.baidu.cn)）
2. 端口号：（:80）

用端口号来区分同一台服务器上的不同项目（端口号的取值范围0~65535）

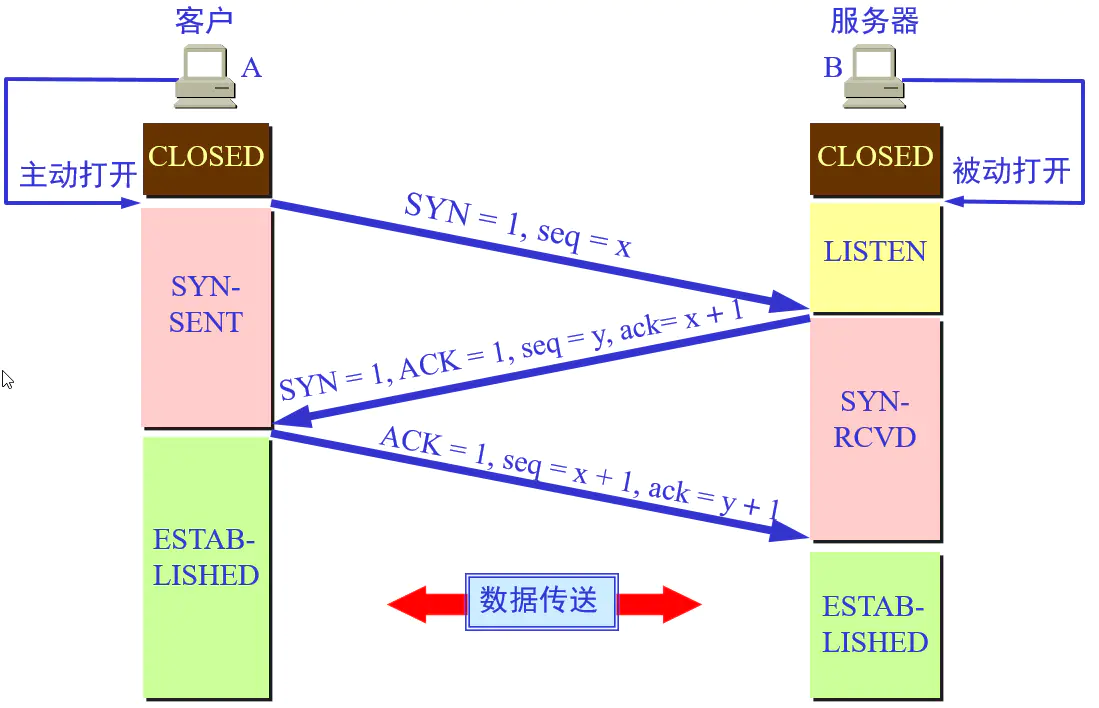
1. 请求资源路径名称：（/stu/index.html）
2. 问号传参信息：（?from=wx&lx=1）
3. HASH值：（#zhenyu）

### DNS 域名解析

DNS优化：

* DNS缓存：一般浏览器会在第一次解析后，默认建立缓存，时间很短，只有一分钟左右
* 减少DNS解析次数：一个网站中我们需要发送请求的域名和服务器尽可能少即可
* DNS预获取(dns-prefetch)：在页面加载开始的时候，就把当前页面中需要访问其他域名(服务器)的信息进行提前DNS解析，以后加载到具体内容部分可以不用解析了

### 三次握手(建立 TCP 连接)



### 发送Http请求

#### http报文（请求报文+响应报文）

 请求报文：

* 定义：所有经过传输协议，客户端传递给服务器的内容，都被成为请求报文
* 包含：
  + 起始行
  + 请求头（请求首部）
  + 请求主体

 响应报文：

* 定义：所有经过传输协议，服务器返回给客户端的内容，都被成为响应报文
* 包含：
  + HTTP状态码
  + 响应头
  + 响应主体

#### http状态码

 1 开头的代表处理中，一般见不到

 2开头：都是成功

200：OK：成功

201：CREATED：一般应用于告诉服务器创建一个新文件，最后服务器创建成功后返回的状态码

204：NO CONTENT：对于某些请求（例如：PUT或者DELETE），服务器不想处理，可以返回空内容，并且用204状态码告知

 3开头：代表成功，只不过中间需要中转一下

301：Moved Permanently：永久重定向（永久转移）

302：Moved Temporarily：临时转移，很早以前基本上用302来做，但是现在主要用307来处理这个事情，

307的意思就是临时重定向Temporary Redirect =>主要用于：服务器的负载均衡等

304：Not Modified：设置HTTP的协商缓存

 4开头：都是失败

400：Bad Request：传递给服务器的参数错误

401：Unauthorized：无权限访问

404：Not Found：请求地址错误

 5开头：都是失败

500：Internal Server Error：未知服务器错误

503：Service Unavailable：服务器超负荷

#### Http请求方式（get和post）

Get 和post 的区别

## Ajax

<https://zhuanlan.zhihu.com/p/98288927>

<https://juejin.cn/post/6844903618764603399>

### 定义

Ajax是一种异步请求数据的web开发技术，对于改善用户的体验和页面性能很有帮助。简单地说，在不需要重新刷新页面的情况下，Ajax 通过异步请求加载后台数据，并在网页上呈现出来。

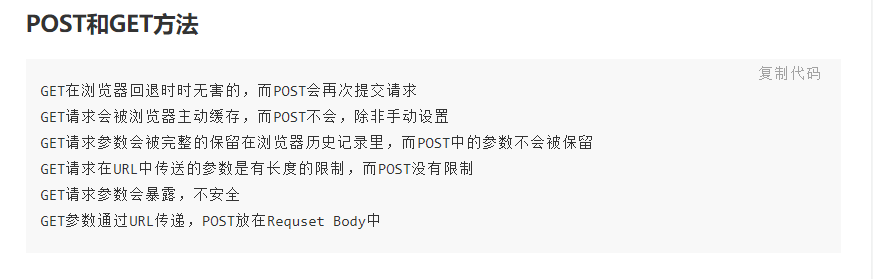
## Tcp和Udp区别



## http和https区别



## Get和post



## 常见状态码



# Es6

## Let 和const

## 箭头函数



## Class类

## 解构赋值



## 对象属性简写

## Promise异步解决方案





## For .. of



## Set/WeakSet

Set 对象允许你存储任何类型的唯一值，无论是原始值或者是对象引用。

## Map

<https://juejin.cn/post/6844903790898839559>

# 前端安全性问题

## CSRF

:cross-site request forgery 跨站请求伪造。攻击者盗用身份，以用户名义发送恶意请求。

用户登录受信任网站A后，用户登录信息保存在cookie中返回给浏览器。如果此时用户没有退出登录的情况下打开了危险网站B，B接受到用户请求返回攻击性代码，并发出一个请求要求访问第三方站点A，浏览器接受到请求后，携带Cookie信息访问网站A，但是A并不知道请求来自于哪里，导致B的恶意代码被执行。

防御：

1. Cookie hashing (所有表单都包含同一个伪随机值)
2. 验证码
3. 在请求地址中添加 token 并验证

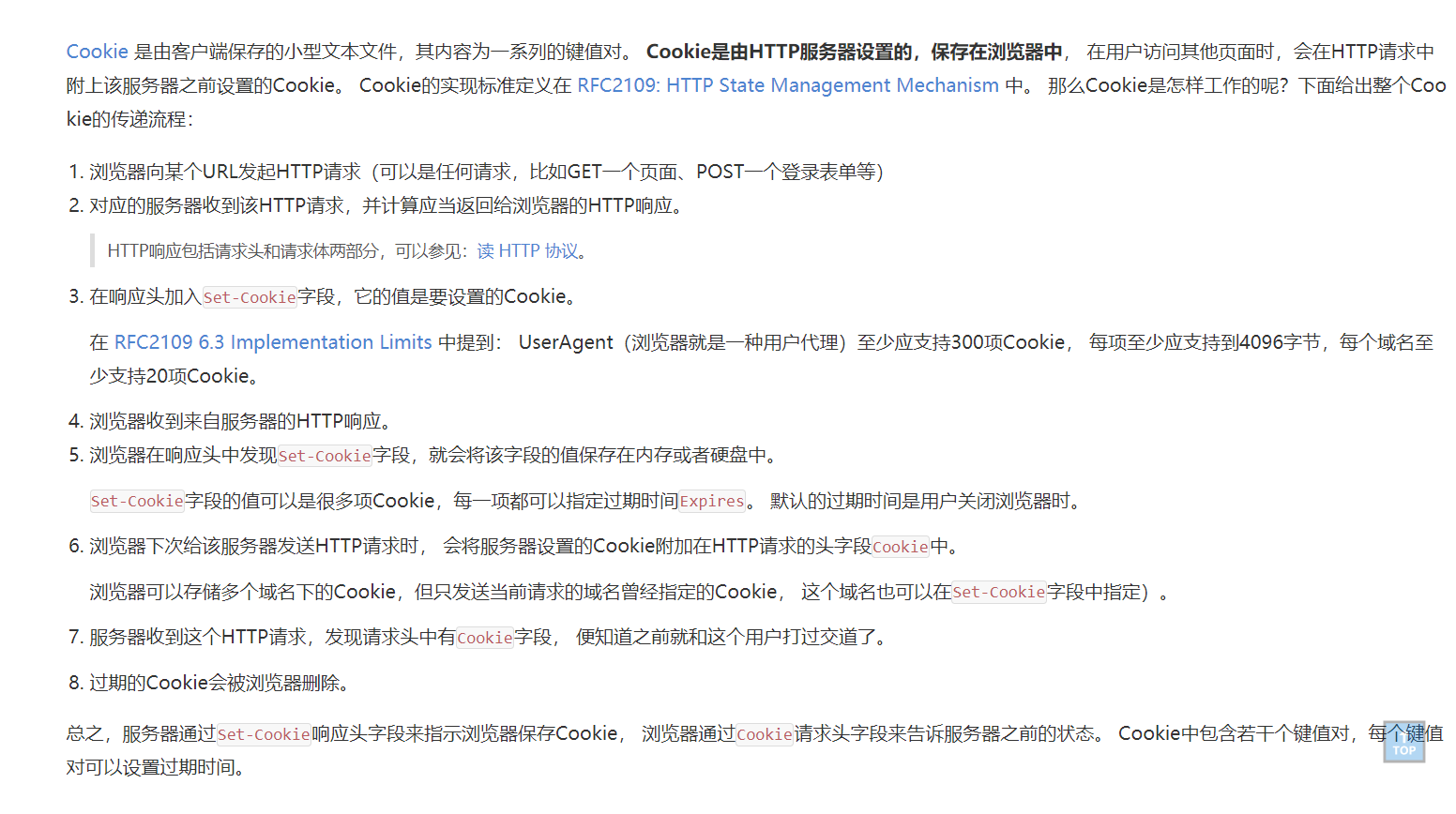
## XSS

cross-site scripting 跨站脚本攻击

指攻击者在网页中嵌入客户端脚本(例如JavaScript), 当用户浏览此网页时，脚本就会在用户的浏览器上执行，从而达到攻击者的目的.

<https://segmentfault.com/a/1190000016551188>

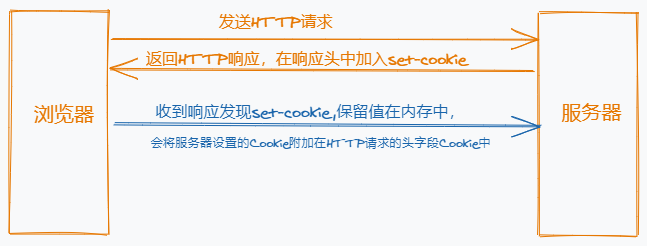
# Cookie



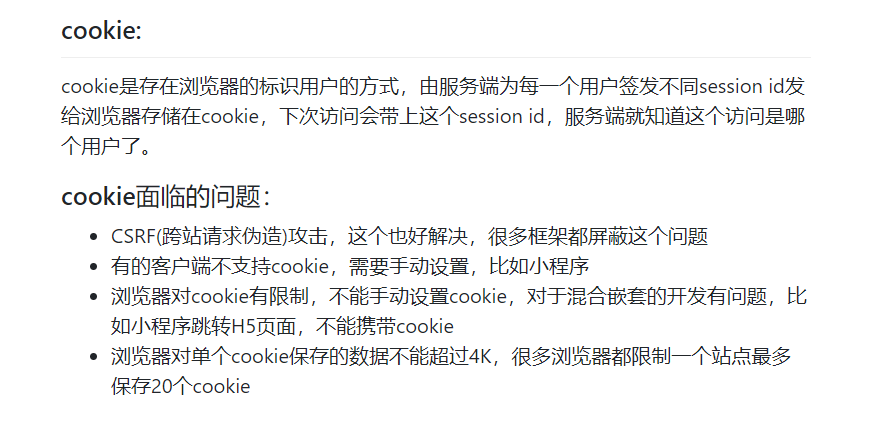
<https://harttle.land/2015/08/10/cookie-session.html>

Cookie是客户端保存的小型文本文件。

是由HTTP服务器设置的保存在浏览器中。

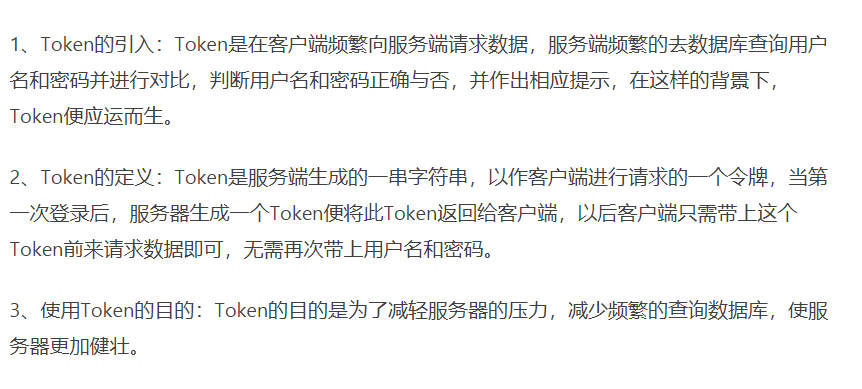


服务器通过Set-Cookie响应头字段来指示浏览器保存Cookie， 浏览器通过Cookie请求头字段来告诉服务器之前的状态。 Cookie中包含若干个键值对，每个键值对可以设置过期时间。



## Token:

<https://junyiseo.com/php/757.html>



## Cookie.Token,session

* session存储于服务器，可以理解为一个状态列表，拥有一个唯一识别符号sessionId，通常存放于cookie中。服务器收到cookie后解析出sessionId，再去session列表中查找，才能找到相应session。依赖cookie
* cookie类似一个令牌，装有sessionId，存储在客户端，浏览器通常会自动添加。
* token也类似一个令牌，无状态，用户信息都被加密到token中，服务器收到token后解密就可知道是哪个用户。需要开发者手动添加。
* jwt只是一个跨域认证的方案

# Css

## BFC

<https://www.cnblogs.com/qs-cnblogs/p/12349887.html>

1. 版权声明：本文为CSDN博主「胡歌的小奶音」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

   原文链接：https://blog.csdn.net/weixin\_44157964/article/details/108110673 [↑](#endnote-ref-1)