**ES6**

[一、 谈谈css3Flex布局 2](#_Toc8159)

[align-items 2](#_Toc32484)

[二、 浅谈session,cookie,sessionStorage,localStorage 2](#_Toc31096)

[1.1 Cookie 2](#_Toc25216)

[三、 px,em,rem区别 3](#_Toc32271)

[四、 animation和transiton 3](#_Toc17569)

[五、 Css3写一个三角箭头 3](#_Toc3485)

[六、 js闭包 3](#_Toc25410)

[1.1 执行的过程 4](#_Toc31861)

[七、 前端跨域 4](#_Toc7847)

[八、 节流防抖 5](#_Toc31261)

[九、 This的指向 6](#_Toc16200)

[十、 http三次握手协议 6](#_Toc20497)

[十一、 http常见状态吗 6](#_Toc2058)

[十二、 Vue生命周期 6](#_Toc16274)

[十三、 Route和router 8](#_Toc24405)

[十四、 hash和history模式区别 8](#_Toc16103)

[十五、 spa页面优缺点 8](#_Toc28107)

[十六、 spa页面直接给一个数组项目赋值，vue能检测到变化的流程吗 9](#_Toc6595)

[十七、 什么是MVVM 9](#_Toc18337)

[十八、 Vue-router常用hash和history路由模式实现原理 9](#_Toc21925)

# 谈谈css3Flex布局

Display:flex;

Flex-direction:column row column-reverse row-reverse

Flex-wrap:nowrap wrap wrap-reverse;

Flex-row:

Justify-content水平对齐的方式：flex-start flex-end center space-betWeen space-around

### align-items

# [浅谈session,cookie,sessionStorage,localStorage](https://www.cnblogs.com/jing-tian/p/10991440.html)

## Cookie

Cookie是被存储在浏览器客户端，有大小和个数限制生命周期随浏览器的关闭而结束。当然可以设置过期的时间！

session用于保存每个用户的专用信息，变量的值保存在服务器端,通过SessionID来区分不同的客户

WebStorage:localStorage（本地存储）和sessionStorage（会话存储）。

localStorage:localStorage的生命周期是永久的，关闭页面或浏览器之后localStorage中的数据也不会消失。localStorage除非主动删除数据，否则数据永远不会消失。

 sessionStorage的生命周期是在仅在当前会话下有效。浏览器关闭就没了

# px,em,rem区别

<https://www.cnblogs.com/leejersey/p/3662612.html>

px像素（Pixel）。相对长度单位。像素px是相对于显示器屏幕分辨率而言的

em是相对长度单位。相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸,通常要在body上设置大小，会继承父元素的大小

# animation和transiton

比如我的信息，一开始加载以后下一次进来就不需要额外重新请求服务器了

<https://www.w3school.com.cn/cssref/pr_animation.asp>

<https://www.w3school.com.cn/cssref/pr_transition.asp>

transition-property

transition-duration

transition-timing-function

transition-delay

animation-name

animation-duration

animation-timing-function

animation-delay

animation-iteration-count

animation-direction

# Css3写一个三角箭头

# js闭包

## 1.1 执行的过程

index.vue-beforeRouteLeave

router-beforeEach

proList.vue-beforeEnter

proList.vue-beforeRouteEnter

router-afterEach

App.vue-beforeUpDate

proList.vue-beforeCreate

proList.vue-created

proList.vue-beforeMount

index.vue-beforeDestroy

index.vue-destroyed

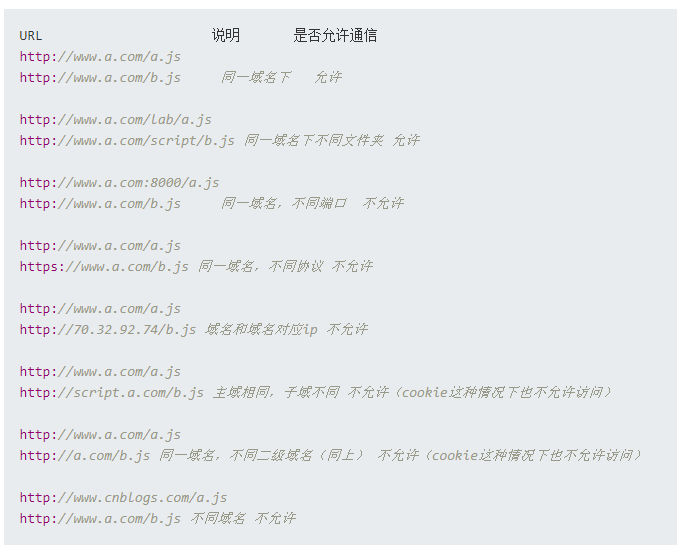
proList.vue-mounted

App.vue-updated

# 前端跨域

<https://segmentfault.com/a/1190000007326671>

这个时



# 节流防抖

function throttle(fn,time=250){

            let isRun=true;

            return function(){

                if(!isRun) return;

                isRun=false;

                setTimeout(()=>{

                    fn.apply(this,arguments);

                    isRun=true

                },time)

            }

        }

function debounce(fn, interval = 300) {

            let timeout = null;

            return function () {

                clearTimeout(timeout);

                timeout = setTimeout(() => {

                    fn.apply(this, arguments);

                }, interval);

            }

        }

# This的指向

构造函数 this 指向实例

普通函数 this 指向window

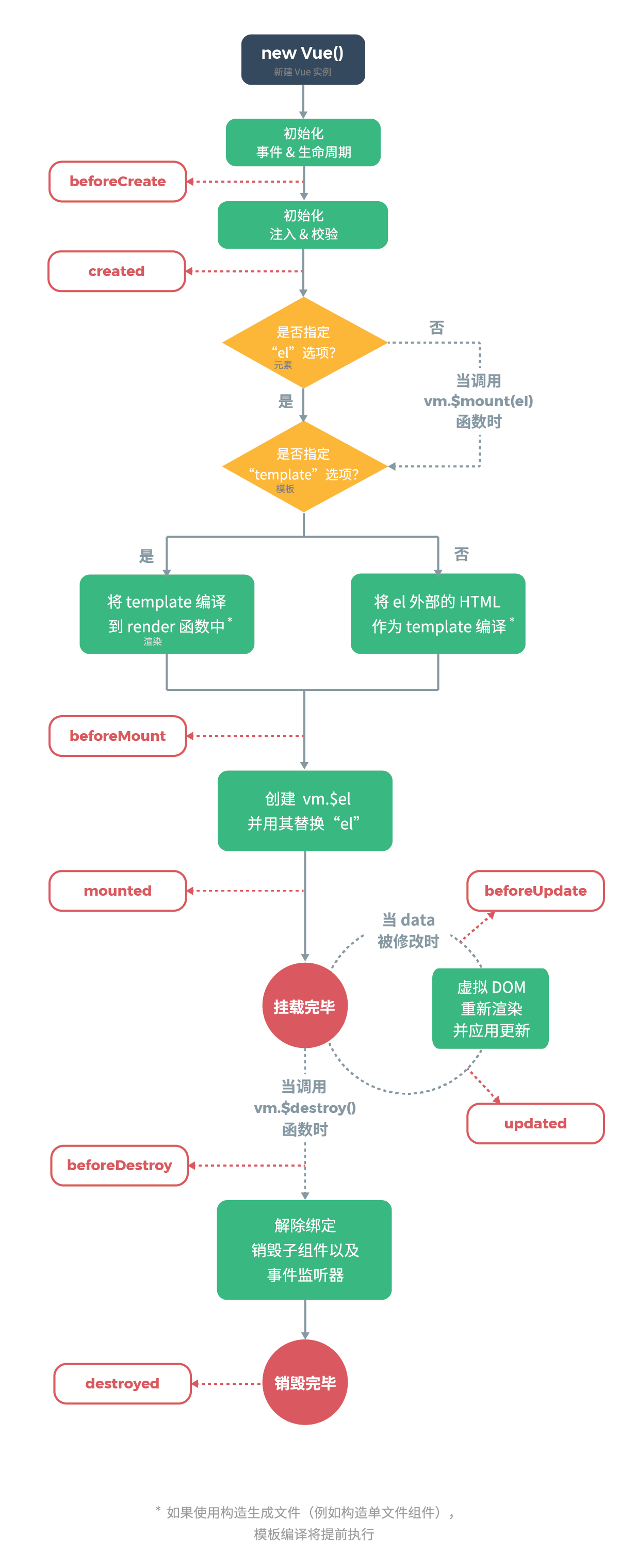
Apply call可以动态改变this指针

箭头函数

# http三次握手协议

# http常见状态吗

# Vue生命周期



# Route和router

Vue 路由中 hash 模式和 history 模式区别IMG_256  
router是“路由实例”对象包括了路由的跳转方法，钩子函数等

# hash和history模式区别

利用了 HTML5 History Interface

hash 虽然出现在 URL 中，但不会被包括在 HTTP 请求中，对后端完全没有影响，因此改变 hash 不会重新加载页面。

真要通过 URL 向后端发起 HTTP 请求时，两者的差异就来了。尤其在用户手动输入 URL 后回车，或者刷新（重启）浏览器的时候。

前端的 URL 必须和实际向后端发起请求的 URL 一致

# spa页面优缺点

**优点：**

* 用户体验好、快，内容的改变不需要重新加载整个页面，避免了不必要的跳转和重复渲染；
* 基于上面一点，SPA 相对对服务器压力小；
* 前后端职责分离，架构清晰，前端进行交互逻辑，后端负责数据处理；

**缺点：**

* 初次加载耗时多：为实现单页 Web 应用功能及显示效果，需要在加载页面的时候将 JavaScript、CSS 统一加载，部分页面按需加载；
* 前进后退路由管理：由于单页应用在一个页面中显示所有的内容，所以不能使用浏览器的前进后退功能，所有的页面切换需要自己建立堆栈管理；
* SEO 难度较大：由于所有的内容都在一个页面中动态替换显示，所以在 SEO 上其有着天然的弱势。

# spa页面直接给一个数组项目赋值，vue能检测到变化的流程吗

# 什么是MVVM

<https://juejin.im/post/5d59f2a451882549be53b170>

MVVM 是 Model-View-ViewModel 的缩写。

**Model**代表数据模型，也可以在Model中定义数据修改和操作的业务逻辑。  
**View** 代表UI 组件，它负责将数据模型转化成UI 展现出来。

**ViewModel** 监听模型数据的改变和控制视图行为、处理用户交互，简单理解就是一个同步View 和 Model的对象，连接Model和View。

# Vue-router常用hash和history路由模式实现原理

# Vue-router路由的钩子函数

# 