前端通用部分

前端优化的方法

https://www.cnblogs.com/coober/p/8078847.html

XSS攻击与CSRF攻击

https://www.jianshu.com/p/a1e7f23189ab

网络请求部分

跨域

跨域的定义

> 跨域是指一个域下的文档或脚本试图去请求另一个域下的资源，广义的跨域请求：

1. 资源跳转: A链接、重定向、表单提交
2. 资源嵌入：<link> <script> <img> <frame>, 以及css中的url()、@font-face()等文件外链
3. 脚本请求：js发起的ajax请求、dom和js对象的跨域操作等

> 我们通常所说的跨域是狭义的，是由浏览器同源策略限制的一类请求场景

什么是同源策略

> 同源策略/SOP(same origin policy)是一种约定, 是浏览器的安全性能所需，所谓同源是指"协议+域名+端口"三者相同，即便两个不同的域名指向同一个ip地址，也非同源。 同源限制的行为：

1. Cookie、LocalStorage 和 IndexDB 无法读取
2. DOM 和 JS 对象无法获得
3. AJAX请求不能发送

跨域的方法

1. 通过jsonp跨域

> 通常为了减轻web服务器的负载，我们把js、css、img等静态资源分离到另一台独立域名的服务器上，在html中再通过相应的标签从不同域名下加载静态资源，基于这样的原理，我们可以通过动态创建script，再请求一个带参网址实现跨域通信。

1. 原生实现：动态创建script标签并设置回调函数

```
var script = document.createElement('script');
script.type = 'text/javascript';
// 传参并指定回调执行函数为onBack
script.src = 'http:/www.demo2.com:8080/loginuser=admin&callback=onBack';
document.head.appendChild(script;
// 回调执行函数
```

```
function onBack(res) {
    alert(JSON.stringify(res));
}
```

2. jquery封装实现

```
$.ajax({
    url: 'http://www.demo2.com:8080/login',
    type: 'get',
    dataType: 'jsonp',  // 请求方式为jsonp
    jsonpCallback: "onBack",     // 自定义回调函数名
    data: {}
});
```

> jsonp缺点：只能实现get一种请求

2. iframe + document.domain跨域

> 此方案仅限主域相同，子域不同的跨域场景; 两个页面通过js强制设置document.domain为基础主域，从而实现同域。

3. iframe + location.hash跨域

> 实现原理： a欲与b跨域相互通信，通过中间页c来实现。 三个页面，不同域之间利用iframe的location.hash传值，相同域之间直接js访问来通信。

4. iframe + window.name跨域

> window.name属性的独特之处：name值在不同的页面（甚至不同域名）加载后依旧存在，并且可以支持非常长的 name 值（2MB）。

5. postMessage跨域

> postMessage是HTML5 XMLHttpRequest Level 2中的API，且是为数不多可以跨域操作的window属性之一，它可用于解决以下方面的问题：
> a.） 页面和其打开的新窗口的数据传递
> b.） 多窗口之间消息传递
> c.） 页面与嵌套的iframe消息传递
> d.） 上面三个场景的跨域数据传递
> 用法：postMessage(data,origin)方法接受两个参数
> **data**： html5规范支持任意基本类型或可复制的对象，但部分浏览器只支持字符串，所以传参时最好用JSON.stringify()序列化。
> **origin**： 协议+主机+端口号，也可以设置为"*"，表示可以传递给任意窗口，如果要指定和当前窗口同源的话设置为"/"。

6. 跨域资源共享（CORS）

> CORS背后的基本思想是使用自定义的HTTP头部允许浏览器和服务器相互了解对方，从而决定请求或响应成功与否 服务器设置

```
// 设置允许的Origin
Access-Control-Allow-Origin: http://foo.example
// 设置允许的方法
Access-Control-Allow-Methods: POST, GET, OPTIONS
// 设置允许的头部信息
Access-Control-Allow-Headers: X-PINGOTHER, Content-Type
// 设置响应允许的时间
Access-Control-Max-Age: 86400
```

7. Nodejs中间件代理跨域

> node中间件实现跨域代理，原理大致与nginx相同，都是通过启一个代理服务器，实现数据的转发，也可以通过设置cookieDomainRewrite参数修改响应头中cookie中域名，实现当前域的cookie写入，方便接口登录认证。

- 利用node + express + http-proxy-middleware搭建一个proxy服务器。（2次跨域）
- 利用node + webpack + webpack-dev-server代理接口跨域（1次跨域）

8. nginx代理跨域

> 跨域原理： 同源策略是浏览器的安全策略，不是HTTP协议的一部分。服务器端调用HTTP接口只是使用HTTP协议，不会执行JS脚本，不需要同源策略，也就不存在跨越问题。

> 实现思路：通过nginx配置一个代理服务器（域名与demo1相同，端口不同）做跳板机，反向代理访问demo2接口，并且可以顺便修改cookie中demo信息，方便当前域cookie写入，实现跨域登录。

```
#proxy服务器
server {
    listen       81;
    server_name  www.demo1.com;

    location / {
        proxy_pass   http://www.demo2.com:8080;  #反向代理
        proxy_cookie_demo www.demo2.com www.demo1.com; #修改cookie里域名
        index  index.html index.htm;

        # 当用webpack-dev-server等中间件代理接口访问nignx时，此时无浏览器参与，故没有同源限制，下面的跨域配置可不启用
        add_header Access-Control-Allow-Origin http://www.demo1.com;  #当前端只跨域不带cookie时，可为*
        add_header Access-Control-Allow-Credentials true;
    }
}
```

9. WebSocket协议跨域

> WebSocket protocol是HTML5一种新的协议。它实现了浏览器与服务器全双工通信，同时允许跨域通讯，是server push技术的一种很好的实现。

http请求以及响应常见头部元素

https://blog.csdn.net/selinda001/article/details/79338766

http1 1.1 2.0的区别

https://www.jianshu.com/p/25b762d58e66

http和websocket的区别

https://www.cnblogs.com/goeasycloud/p/9355164.html

H5部分

block和inline元素举例及区别

https://www.cnblogs.com/ytshang/p/8931842.html

JS部分

Promise

> Promise有pending，fulfilled，rejected 三种状态，pending可以向其他两种状态转换，且转换一旦发生不可以再改变

Axios以及Axios拦截器

https://blog.csdn.net/qq_38145702/article/details/81558816

let const var的作用域 - 暂时性死区

https://www.cnblogs.com/ricoliu/p/6149912.html

JS常用的一些API

https://www.cnblogs.com/clairexia/p/6635029.html

阻止冒泡的方法

http://caibaojian.com/javascript-stoppropagation-preventdefault.html

闭包的产生以及使用场景

闭包的产生以及使用场景

原型链

https://www.cnblogs.com/DF-fzh/p/5619319.html

类的继承方法以及实例

https://www.cnblogs.com/humin/p/4556820.html

Window对象（BOM）的一些常见API history navigataor screen 以及其常用的功能函数

CSS部分

FLEX 和 GIRD 布局的应用场景

FLEX 以及 GRID 的一些详细属性

REM EM PX 的区别以及运用场景

描述实现一行内占用了一定空间 填充剩下所有空间的方法

盒模型

css不同类型的选择器及添加样式的内联方式 以及这些选择器的作用顺序以及！important的添加

React部分

虚拟DOM概念以及其优点

提高性能，Diff算法，减少不必要的重绘和重排版 https://blog.csdn.net/u010046318/article/details/77340188

React-router的原理

https://www.cnblogs.com/zd-aw123/p/9810897.html

Set-state的运作原理

https://www.jianshu.com/p/89a04c132270

正则表达式部分

不熟别说问题不大