前端学习笔记

JSONP

跨域的原理：

利用<script>标签没有跨域限制的“**漏洞**”（历史遗迹啊）来达到与第三方通讯的目的。当需要通讯时，本站脚本创建一个<script>元素，地址指向第三方的 API 网址，形如：<script src="http://www.example.net/api?param1=1&param2=2"></script> 并提供一个回调函数来接收数据（函数名可约定，或通过地址参数传递）。 第三方产生的响应为 json 数据的包装（故称之为 jsonp，即 json padding），形如：callback({"name":"hax","gender":"Male"}) 这样浏览器会调用 callback 函数，并传递解析后 json 对象作为参数。本站脚本可在 callback 函数里处理所传入的数据。

补充：“历史遗迹”的意思就是，如果在今天重新设计的话，也许就不会允许这样简单的跨域了嘿，比如可能像 XHR 一样按照 CORS 规范要求服务器发送特定的 http 头。

HTML 行内元素、块状元素、行内块

HTML 可以将元素分类方式分为行内元素、块状元素和行内块状元素三种。首先需要说明的是，这三者是可以互相转换的，使用 display 属性能够将三者任意转换：

(1)display:inline;转换为行内元素
(2)display:block;转换为块状元素
(3)display:inline-block;转换为行内块状元素

行内元素

行内元素最常使用的就是 **span**，其他的只在特定功能下使用，修饰字体<b>和<i>标签，还有<sub>和<sup>这两个标签可以直接做出平方的效果，而不需要类似移动属性的帮助，很实用。

行内元素特征：(1)设置宽高无效
(2)对 margin 仅设置左右方向有效，上下无效；padding 设置上下左右都有效，即会撑大空间
(3)不会自动进行换行

1. **和其他元素都在一行上**，遇到父级元素边界会自动换行
2. 高、行高以及内外边距都不可以改变
3. 宽度与内容一样宽，且不可改变
4. 行内元素只能容纳文本或者其他行内元素
5. 对于行内元素，需要注意的是：设置宽度 width 无效，设置高度无效，可以通过设置 line-height 来设置，设置 margin 只有左右有效，上下无效，设置 padding 只有左右有效，上下无效

a - 锚点
abbr - 缩写
acronym - 首字
b - 粗体(不推荐)
h1-h6

块状元素

块状元素代表性的就是 div，其他如 p、nav、aside、header、footer、section、article、ul-li、address 等等，都可以用 div 来实现。不过为了可以方便程序员解读代码，一般都会使用特定的语义化标签，使得代码可读性强，且便于查错。

块状元素特征：(1)能够识别宽高
(2)margin 和 padding 的上下左右均对其有效
(3)可以自动换行
(4)多个块状元素标签写在一起，默认排列方式为从上至下

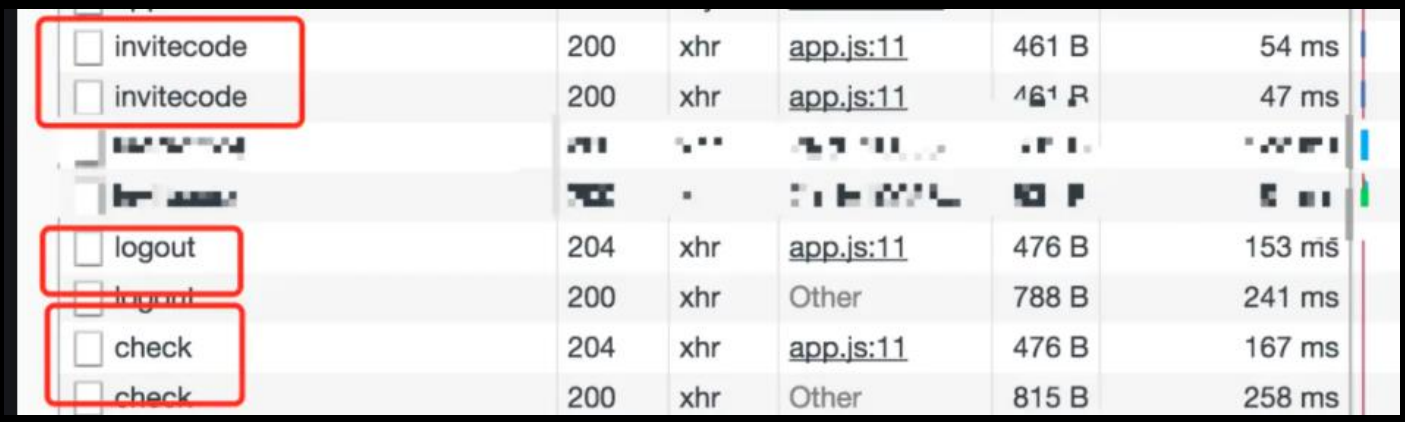
行内块状元素

行内块状元素综合了行内元素和块状元素的特性，但是各有取舍。因此行内块状元素在日常的使用中，由于其特性，使用的次数也比较多。

行内块状元素特征：(1)不自动换行
(2)能够识别宽高
(3)默认排列方式为从左到右

preflight request

有时候我们在调用后台接口的时候，会请求两次：



其实第一次发送的就是 **preflight request(预检请求)**

为什么要发预检请求？

我们都知道浏览器的同源策略，就是出于安全考虑，浏览器会限制从脚本发起的跨域 HTTP 请求，像 XMLHttpRequest 和 Fetch 都遵循同源策略。

浏览器限制跨域请求一般有两种方式：

1. **浏览器限制发起跨域请求**
2. **跨域请求可以正常发起，但是返回的结果(Response)被浏览器拦截了**

一般浏览器都是第二种方式限制跨域请求，那就是说**请求已到达服务器**，并有可能对数据库里的数据进行了操作，但是返回的结果被浏览器拦截了，那么我们就获取不到返回结果，这是一次失败的请求，但是可能对数据库里的数据产生了影响。

为了防止这种情况的发生，规范要求，对这种可能对服务器数据产生副作用的 HTTP 请求方法，**浏览器必须先使用 OPTIONS 方法发起一个预检请求**，从而获知服务器是否允许该跨域请求：如果允许，就发送带数据的真实请求；如果不允许，则阻止发送带数据的真实请求。

什么时候发预检请求？

HTTP 请求包括： **简单请求 和 需预检的请求**

1. 简单请求

简单请求不会触发 CORS 预检请求，“简属于单请求”术语并不属于 Fetch(其中定义了 CORS)规范。若满足所有下述条件，则该请求可视为“简单请求”：

使用下列方法之一：

- **GET**
- **HEAD**
- **POST**
    - Content-Type：(仅当 POST 方法的 Content-Type 值等于下列之一才算做简单需求)
        - **text/plain**
        - **multipart/form-data**
        - **application/x-www-form-urlencoded**

  POST 的 application/json 不是简单请求

2. 需预检的请求

“需预检的请求”要求必须首先使用 OPTIONS 方法发起一个预检请求到服务区，以获知服务器是否允许该实际请求。“预检请求”的使用，可以避免跨域请求对服务器的用户数据产生未预期的影响。

当请求满足下述任一条件时，即应首先发送预检请求：

- 使用了下面任一 HTTP 方法：
  a) DELETE
  b) CONNECT
  c) OPTIONS
  d) TRACE
  e) PATCH
- 人为设置了对 CORS 安全的首部字段集合之外的其他首部字段。该集合为：
  a) Accept
  b) Accept-Language
  c) Content-Language
  d) Content-Type
  e) DPR
  f) Downlink
  g) Save-Data
  h) Viewport-Width
  i) Width
  j) Content-Type 的值不属于下列之一：
  k) application/x-www-form-urlencoded
  l) multipart/form-data
  m) text/plain