> *Wiki* ‣ *[[API--中文手册]]* ‣ *[[核心函数]]* ‣ ***请求***

- 本文档是D3官方文档中文翻译，并保持与最新版同步。
- 如发现翻译不当或有其他问题可以通过以下方式联系译者:
- 邮箱：zhang_tianxu@sina.com
- QQ群：D3数据可视化205076374，大数据可视化436442115

如果你不访问数据那么你就不能把它可视化。幸运的是有很多的方法可以把数据放到浏览器中。对于小数据集，你可以硬编码到你的脚本里，或者使用数据属性(data attributes)嵌入到DOM中。对于大数据集，你可以引用外部脚本并定义你的数据为一个全局变量。(JSONP就是一个常见的例子)。最通用的方式是使用 XMLHttpRequest, 或说**XHR**来加载数据到浏览器。这允许*异步(asynchronously)*加载数据(在数据加载的同时, 页面的其余部分就可以展示了)，并且比JSONP更安全。D3的xhr模块可以简单的加载和解析数据。

当异步加载数据时，代码取决于加载的数据通常存在于回调函数之内。例如D3网站的案例calendar visualization. 不依赖于数据的代码可以在页面加载时立即运行。你也可以发现保存数据到全局命名空间(global namespace)是很方便的，这样你就可以在初始化渲染之后访问它，例如在过渡期间。你可以使用闭包(closures)实现它，或者简单地指定加载数据为全局变量。

```
var data; // a global

d3.json("path/to/file.json", function(error, json) {
  if (error) return console.warn(error);
  data = json;
  visualizeit();
});
```

默认情况下，大多数的浏览器不允许跨域访问。为了支持跨域访问(enable cross-domain requests)，服务器必须设置头(header)为Access-Control-Allow-Origin: 。更多信息参见W3C关于跨域资源分享(Cross-Origin Resource Sharing)的建议。对于IE9，d3.xhr使用非标准的XDomainRequest支持跨域访问。注意, 为了发送身份验证, 请求需要用 `.on("beforesend", function (request) {request.withCredentials = true;})` 而非 `.headers("withCredentials", "true")` 进行修饰.

XHR

d3.**xhr**(*url*[, *mimeType*][, *callback*])

为指定的url创建一个异步访问。参数*mimeType*可能被指定为第二个参数，例如"text/plain"。如果指定了回调函数，那么请求就会使用GET方法立即发出，并且当资源被加载或者请求失败之后回调函数就会被异步调用。如果有错的话XMLHttpRequest对象代表了回复(response)。当错误发生时，response是未定义的。如果response有一个不成功状态值，那么错误就是XMLHttpRequest对象。如果没有指定回调函数，返回的request可以被分发使用xhr.get，xhr.post 或相似的函数，并使用xhr.on处理。

xhr.**header**(*name*[, *value*])

如果指定了*value*，设置请求头指定的*name*参数为指定的*value*值。如果*value*参数是null值，就移除指定名称的请求头，如果没指定value值就会返回请求头中指定*name*的当前值，请求头名称是大小写敏感的。

请求头只可以在发送(sent)请求之前被修改。因此，如果你想指定一个请求头就不能传递回调函数给d3.xhr constructor。而应该使用xhr.get或者相似的函数。例如：

```
d3.csv("/path/to/file.csv")
    .header("header-name", "header-value")
```

```
    .get(function(error, data) {
      // callback
});
```

xhr.**mimeType**([*type*])

如果*type*参数被指定了，就会设置请求的mime类型为指定的值。如果*type*是null，就会清除当前的mime类型(有的话)。如果没有指定类型*type*，就返回当前的mime类型(默认是null)。Mime类型被用来设置"Accept"请求头("Accept" request header)和覆盖MimeType(overrideMimeType)。请求头只可以在发送请求之前被修改。

xhr.**responseType**(*type*)

如果*type*被指定了，设置response类型(response type)，例如："", "arraybuffer", "blob", "document"，或者"text"。如果*type*没有指定，就返回当前的response类型默认是空字符串""。

xhr.**response**(*value*)

如果指定了*value*参数，就设置response的值函数为指定的函数。如果*value*没有被指定，返回当前的response的值函数，默认就是验证函数。response的值函数用来映射返回XMLHttpRequest对象为相应的数据类型。例如，对于文本请求，你可以使用 `function(request) { return request.responseText; }`，而对于JSON请求你可能使用 `function(request) { return JSON.parse(request.responseText); }`。

xhr.**get**([*callback*])

使用GET方法分发请求。如果指定了回调函数*callback*，在发送请求或者出错的时候就会被异步调用。回调函数使用两个参数调用：error(有的话)和response值。错误发生时response值是未定义的。如果没有指定回调函数*callback*，则"load"和"error"监听器会通过xhr.on注册。这个方法是xhr.send的一个方便的包装(wrapper)。

xhr.**post**([*data*][, *callback*])

使用POST方法分发这个请求，在请求体中可选地发送指定的data。如果指定了回调函数*callback*，在发送请求或者出错的时候就会被异步调用。回调函数使用两个参数调用：error(有的话)和response值。错误发生时response值是未定义的。如果没有指定回调函数"load"和"error"监听器会通过xhr.on注册。这个方法是xhr.send的一个方便的包装。

使用URL编码的例子：

```
d3.csv("/path/to/file.csv")
    .header("Content-Type", "application/x-www-form-urlencoded")
    .post("a=2&b=3", function(error, data) {
        // callback
    });
```

An example using JSON encoding:

```
d3.csv("/path/to/file.csv")
    .header("Content-Type", "application/json")
    .post(JSON.stringify({a: 2, b: 3}), function(error, data) {
        // callback
    });
```

xhr.**send**(*method*[, *data*][, *callback*])

使用指定的*方法(method)分发这个请求，在请求体中可选地发送指定的数据(data)。如果指定了回调函数callback*，在发送请求或者出错的时候就会被异步调用。回调函数使用两个参数调用：error(有的话)和response值。错误发生时

response值是未定义的。如果没有指定回调函数*callback*, 则"load"和"error"监听器会通过xhr.on注册。这个方法是xhr.send的一个方便的包装。

xhr.**abort**()

中止正在发送的请求。参见XMLHttpRequest的中止： XMLHttpRequest's abort.

xhr.**on**(*type*[, *listener*])

对指定的类型添加或者移除事件监听器到这个请求。类型必须是以下类型之一：

- *beforesend* – 在请求发送之前，允许自定义标题等来进行设定。
- *progress* – 用来监听请求的过程(progress of the request)。
- *load* – 当请求成功的完成之后。
- *error* –当请求不成功之后；此类型包含4xx和5xx返回值。

如果一个相同的类型监听器已经被注册，已存在的监听器就会在新监听器添加之前被移除。为了给同一个事件类型注册多个监听器，那么类型将遵循可选的命名空间，例如 `load.foo` 和 `load.bar` 。为了移除监听器，传递null值为监听器。

如果没有指定监听器，为指定的类型(有的话)返回当前分配的监听器。

简便方法

通常，d3.xhr不会直接使用。取而代之的是使用类型特定的方法，例如： d3.text 加载简单文本, d3.json 加载JSON，d3.xml 加载XML, d3.html 加载HTML, d3.csv加载逗号分隔值文件, d3.tsv加载制表符分隔文件。

d3.**text**(*url*[, *mimeType*][, *callback*])

指定的*url*创建一个文本文件请求。选项*mimeType*可以指定为第二参数，例如"text/plain"。如果指定了回调函数*callback*，请求将通过GET方法立即分发，当文件被加载或者请求失败之后回调函数将被异步调用。回调函数的调用使用两个参数：error(有的话)和response文本。错误发生时response文本是未定义的(undefined)。如果没有指定回调函数，返回的请求可能使用xhr.get或近似的方法分发，并使用xhr.on处理。

d3.**json**(*url*[, *callback*])

指定的*url*创建一个JSON文件请求其mime type为"application/json"。如果指定了回调函数*callback*，请求将通过GET方法立即分发，当文件被加载或者请求失败之后回调函数将被异步调用。回调函数的调用使用两个参数：error(有的话)和解析过的JSON。错误发生时解析过的JSON是未定义的。如果没有指定回调函数，返回的请求可能使用xhr.get或近似的方法分发，并使用xhr.on处理。

d3.**xml**(*url*[, *mimeType*][, *callback*])

指定的*url*创建一个XML文件请求。选项*mimeType*可以指定为第二参数，例如"application/xml"。如果指定了回调函数，请求将通过GET方法立即分发，当文件被加载或者请求失败之后回调函数将被异步调用。回调函数的调用使用两个参数：error(有的话)和解析为文档(document)的XML。错误发生时解析的XML是未定义的。如果没有指定回调函数，返回的请求可能使用xhr.get或近似的方法分发，并使用xhr.on处理。

d3.**html**(*url*[, *callback*])

指定的*url*创建一个文本文件请求。选项*mimeType*可以指定为第二参数，例如"text/html"。如果指定了回调函数，请求将通过GET方法立即分发，当文件被加载或者请求失败之后回调函数将被异步调用。回调函数的调用使用两个参数：error(有的话)和解析为文档碎片(document fragment)的HTML。错误发生时解析后的HTML是未定义的。如果没有指定回调函数，返回的请求可能使用xhr.get或近似的方法分发，并使用xhr.on处理。

d3.**csv**(*url*[, *accessor*][, *callback*])

指定的*url*创建一个[[CSV格式化]]文件请求, 其mime type为"text/csv"。如果指定了回调函数*callback*，请求将通过GET方法立即分发，当文件被加载或者请求失败之后回调函数将被异步调用。回调函数的调用使用两个参数：error(有的话)和每个RFC 4180的解析行(parsed rows)的数组。错误发生时行数组是未定义的(undefined)。如果没有指定回调函数，返回的请求可能使用xhr.get或近似的方法分发，并使用xhr.on处理。

d3.**tsv**(*url*[, *accessor*][, *callback*])

指定的*url*创建一个文本文件请求, 其mime type为"text/tab-separated-values"。如果指定了回调函数，请求将通过GET方法立即分发，当文件被加载或者请求失败之后回调函数将被异步调用。回调函数的调用使用两个参数：error(有的话)和每个RFC 4180的解析行(parsed rows)的数组。错误发生时行数组是未定义的(undefined)。如果没有指定回调函数，返回的请求可能使用xhr.get或近似的方法分发，并使用xhr.on处理。

2014年11月22日 00:11:22 gulu翻译

Howard L.: parsed rows 译作 "解析行"; 一些函数的*mimeType*参数被替换为固定值, 故翻译上有所调整. (2016年1月5日)