Oboe.js

API

This page is also available as a PDF.

此页面还提供 PDF 格式。

The oboe function 双簧管功能

Oboe.js exposes only one function, `oboe`, which is used to instantiate a new Oboe instance. Calling this function starts a new HTTP request unless the caller is managing the stream themselves.

Oboe.js 只公开了一个函数， `oboe` ，用于实例化一个新的 Oboe 实例。除非调用者自己管理流，否则调用此函数会启动一个新的 HTTP 请求。

```
oboe( String url )

oboe({
   url: String,
   method: String,          // optional
   headers: Object,         // optional
   body: String|Object,     // optional
   cached: Boolean,         // optional
   withCredentials: Boolean // optional, browser only
})
```

▼ Deprecated API

```
// the doMethod style of calling is deprecated
// and will be removed in v2.0.0:
oboe.doGet(    url )
oboe.doDelete( url )
oboe.doPost(   url, body )
oboe.doPut(    url, body )
oboe.doPatch(  url, body )

oboe.doGet(    {url:String, headers:Object, cached:Boolean} )
oboe.doDelete( {url:String, headers:Object, cached:Boolean} )
oboe.doPost(   {url:String, headers:Object, cached:Boolean, body:String|Object} )
oboe.doPut(    {url:String, headers:Object, cached:Boolean, body:String|Object} )
oboe.doPatch(  {url:String, headers:Object, cached:Boolean, body:String|Object} )
```

The `method`, `headers`, `body`, `cached`, and `withCredentials` arguments are optional.

`method` 、 `headers` 、 `body` 、 `cached` 和 `withCredentials` 参数是可选的。

- If `method` is not given Oboe defaults to `GET`.

  如果没有给出 `method` 双簧管默认为 `GET` 。

- If `body` is given as an object it will be stringified using `JSON.stringify` prior to sending. The Content-Type request header will automatically be set to `text/json` unless a different value is explicitly given.

  如果 `body` 作为对象给出，它将在发送之前使用 `JSON.stringify` 进行字符串化。除非明确给出不同的值，否则 Content-Type 请求标头将自动设置为 `text/json` 。

- If the cached option is given as `false` cachebusting will be applied by appending `_={timestamp}` to the URL's query string. Any other value will be ignored.

  如果缓存选项以 `false` 形式给出，缓存清除将通过将 `_={timestamp}` 附加到URL 的查询字符串来应用。任何其他值都将被忽略。

Cross-domain requests 跨域请求

Oboe.js is able to make cross-domain requests so long as the server is set up for CORS, for example by setting the Access-Control-Allow-Origin response header.

只要服务器设置为 CORS，Oboe.js 就能够发出跨域请求，例如通过设置 Access-Control-Allow-Origin 响应标头。

- If `withCredentials` is given as `true`, cookies and other auth will be propagated to cross-domain requests as per the XHR spec. For this to work the server must be set up to send the Access-Control-Allow-Credentials response header.

  如果 `withCredentials` 被指定为 `true` ，则 cookie 和其他身份验证将根据 XHR 规范传播到跨域请求。为此，服务器必须设置为发送 Access-Control-Allow-Credentials 响应标头。

Note that Internet Explorer only fully supports CORS from version 10 onwards.

请注意，Internet Explorer 仅从版本 10 开始完全支持 CORS。

When making requests from Node the `withCredentials` option is never needed and is ignored. The cors middleware might be useful if you are serving cross-domain requests from Express/Connect.

从 Node 发出请求时，永远不需要 `withCredentials` 选项，它会被忽略。如果您正在为来自 Express/Connect 的跨域请求提供服务，则 cors 中间件可能会有用。

BYO stream

Under Node.js you may also pass `oboe` an arbitrary ReadableStream for it to read JSON from. It is your responsibility to initiate the stream and Oboe will not start a new HTTP request on your behalf.

在 Node.js 下，您还可以向 `oboe` 传递一个任意的 ReadableStream，以便它从中读取 JSON。启动流是您的责任，Oboe 不会代表您启动新的 HTTP 请求。

```
oboe( stream )   // Node.js only
```

node event

The methods `.node()` and `.on()` are used to register interest in particular nodes by providing JSONPath patterns. As the JSON stream is parsed the Oboe instance checks for matches against these patterns and when a matching node is found it emits a `node` event.

`.node()` 和 `.on()` 方法用于通过提供 JSONPath 模式来注册对特定节点的兴趣。在解析 JSON 流时，Oboe 实例会检查这些模式的匹配项，并在找到匹配节点时发出 `node` 事件。

```
.on('node', pattern, callback)

// 2-argument style .on() ala Node.js EventEmitter#on
.on('node:{pattern}', callback)

.node(pattern, callback)

// register several listeners at once
.node({
   pattern1 : callback1,
   pattern2 : callback2
});
```

When the callback is notified, the context, `this`, is the Oboe instance, unless it is bound otherwise. The callback receives three parameters:

通知回调时，上下文 `this` 是 Oboe 实例，除非它以其他方式绑定。回调接收三个参数：

`node`	The node that was found in the JSON stream. This can be any valid JSON type - `Array`, `Object`, `String`, `Boolean` or `null` 在 JSON 流中找到的节点。这可以是任何有效的 JSON 类型 - `Array` 、 `Object` 、 `String` 、 `Boolean` 或 `null`
`path`	An array of strings describing the path from the root of the JSON to the matching item. For example, if the match is at `(root).foo.bar` this array will equal `['foo', 'bar']`. Example usage. 描述从 JSON 根到匹配项的路径的字符串数组。例如，如果匹配项位于 `(root).foo.bar` ，则此数组将等于 `['foo', 'bar']` 。用法示例。
`ancestors`	An array of the found item's ancestors such that `ancestors[0]` is the JSON root, `ancestors[ancestors.length-1]` is the parent object, and `ancestors[ancestors.length-2]` is the grandparent. These ancestors will be as complete as possible given the data which has so far been read from the stream but because Oboe.js is a streaming parser they may not yet have all properties 找到的项的祖先数组，其中 `ancestors[0]` 是 JSON 根， `ancestors[ancestors.length-1]` 是父对象， `ancestors[ancestors.length-2]` 是祖父母。考虑到到目前为止从流中读取的数据，这些祖先将尽可能完整，但因为 Oboe.js 是一个流式解析器，它们可能还没有所有属性

```
oboe('friends.json')
   .node('name', function(name){
      console.log('You have a friend called', name);
   });
```

transforming the JSON 转换 JSON

If the callback returns any value, the returned value will be used to replace the parsed JSON node.

如果回调返回任何值，则返回值将用于替换已解析的 JSON 节点。

See demarshalling to an OOP model and Transforming JSON while it is streaming.

请参阅解组为 OOP 模型和在流式传输时转换 JSON。

```
// Replace any object with a name, date of birth, and address
// in the JSON with a Person instance

.on('node', '{name dob address}', function(personJson){
   return new Person(personJson.name, personJson.dob, personJson.address);
})
```

oboe.drop

If a node listener returns the special value `oboe.drop`, the detected node is dropped entirely from the tree.

如果节点侦听器返回特殊值 `oboe.drop` ，则检测到的节点将从树中完全删除。

This can be used to ignore fields that you don't care about, or to load large JSON without running out of memory.

这可用于忽略您不关心的字段，或者加载大型 JSON 而不会耗尽内存。

```
.on('node', 'person.address', function(address){
   console.log("I don't care what's at", address);
   return oboe.drop;
})
```

`oboe.drop` can also be used in a shorthand form:

`oboe.drop` 也可以简写形式使用：

```
.on('node', 'person.address', oboe.drop)
```

Dropping from an *array* will result in an array with holes in it. This is intentional so that the array indices from the original JSON are preserved:

从数组中删除将导致数组中有空洞。这是有意的，以便保留原始 JSON 中的数组索引：

```
// json from service is an array of names:
['john', 'wendy', 'frank', 'victoria', 'harry']

oboe('names.json')
   .on('2', oboe.drop)
   .done(console.log)

// this logs:
//    ['john', 'wendy', , 'victoria', 'harry']
//    note the array hole
```

Meanwhile, dropping from an *object* leaves no trace of the key/value pair:

同时，从对象中删除不会留下键/值对的痕迹：

```
// json from service is a hash from names to numbers:
{'john':4, 'wendy': 2, 'frank': 0, 'victoria': 9, 'harry': 2}

oboe('gameScores.json')
   .on('wendy', oboe.drop)
   .done(console.log)
```

```
// this logs:
//    {'john':4, 'frank': 0, 'victoria': 9, 'harry': 2}
```

path event

Path events are identical to node events except that they are emitted as soon as matching paths are found, without waiting for the thing at the path to be revealed.

路径事件与节点事件相同，只是它们会在找到匹配路径后立即发出，而无需等待路径上的事物被揭示。

```
.on('path', pattern, callback)

// 2-argument style .on() ala Node.js EventEmitter#on
.on('path:{pattern}', callback)

.path(pattern, callback)

// register several listeners at once
.path({
   pattern1 : callback1,
   pattern2 : callback2
});
```

```
oboe('friends.json')
   .path('friend', function(name){
      friendCount++;
   });
```

One use of path events is to start adding elements to an interface before they are complete.

路径事件的一种用途是在元素完成之前开始向界面添加元素。

done event

```
.done(callback)

.on('done', callback)
```

Done events are fired when the response is complete. The callback is passed the entire parsed JSON.

响应完成时会触发 Done 事件。回调传递整个已解析的 JSON。

In most cases it is faster to read the JSON in small parts by listening to `node` events (see above) than waiting for it to be completely download.

在大多数情况下，通过监听 `node` 事件（见上文）来读取小部分的 JSON 比等待它完全下载更快。

```
oboe('resource.json')
   .on('done', function(parsedJson){
      console.log('Request complete', parsedJson);
   });
```

start event

```
.start(callback)

.on('start', callback)
```

Start events are fired when Oboe has parsed the status code and the response headers but has not yet received any content from the response body.

当 Oboe 已解析状态代码和响应标头但尚未从响应正文中接收到任何内容时，将触发启动事件。

The callback receives two parameters:

回调接收两个参数：

`status`	`Number`	HTTP status code HTTP 状态码
`headers`	`Object`	Object of response headers 响应标头的对象

```
oboe('resource.json')
   .on('start', function(status, headers){
      console.log('Resource cached for', headers.Age, 'secs');
   });
```

Under Node.js this event is never fired for BYO streaming.

在 Node.js 下，永远不会为 BYO streaming 触发此事件。

fail event

```
   .fail(callback)

   .on('fail', callback)
```

Fetching a resource could fail for several reasons:

获取资源可能因多种原因而失败：

- Non-2xx status code 非 2xx 状态码
- Connection lost
- Invalid JSON from the server 来自服务器的无效 JSON
- Error thrown by an event listener

  事件侦听器抛出的错误

The fail callback receives an object with four fields:

失败回调接收一个包含四个字段的对象：

`thrown`	The error, if one was thrown 错误，如果一个被抛出
`statusCode`	The status code, if the request got that far 状态代码，如果请求到达那么远
`body`	The response body for the error, if any 错误的响应正文（如果有）

`jsonBody`	If the server's error response was JSON, the parsed body 如果服务器的错误响应是 JSON，则解析的正文

```
oboe('/content')
   .fail( function( errorReport ){
      if( 404 == errorReport.statusCode ){
         console.error('no such content');
      }
   });
```

.source

Since v.1.15.0

Gives the URL (or stream) that the Oboe instance is fetching from. This is sometimes useful if one handler is being used for several streams.

提供 Oboe 实例从中获取的 URL（或流）。如果一个处理程序用于多个流，这有时很有用。

```
oboe('http://example.com/names.json')
   .node('name', handleName);

oboe('http://example.com/more_names.json')
   .node('name', handleName);

function handleName(name){
   console.log('got name', name, 'from', this.source);
}
```

.header([name])

```
.header()

.header(name)
```

`.header()` returns one or more HTTP response headers. If the name parameter is given that named header will be returned as a String, otherwise all headers are returned as an Object.

`.header()` 返回一个或多个 HTTP 响应标头。如果给定名称参数，则命名标头将作为字符串返回，否则所有标头都作为对象返回。

`undefined` wil be returned if the headers have not yet been received. The headers are available anytime after the `start` event has been emitted. They will always be available from inside a `node`, `path`, `start` or `done` callback.

如果尚未收到标头，将返回 `undefined` 。标头在 `start` 事件发出后随时可用。它们将始终在 `node` 、 `path` 、 `start` 或 `done` 回调中可用。

`.header()` always returns undefined for non-HTTP streams.

`.header()` 对于非 HTTP 流总是返回 undefined。

```
oboe('data.json')
   .node('id', function(id){
      console.log(   'Server has id', id,
                     'as of', this.headers('Date'));
   });
```

.root()

At any time, call `.root()` on the oboe instance to get the JSON parsed so far. If nothing has yet been received this will return `undefined`.

在任何时候，在双簧管实例上调用 `.root()` 以获得到目前为止解析的 JSON。如果还没有收到任何东西，这将返回 `undefined` 。

```
var interval;

oboe('resourceUrl')
   .start(function(){
      interval = window.setInterval(function(){
         console.log('downloaded so far:', this.root());
      }.bind(this), 10);
   })
   .done(function(completeJson){
      console.log('download finished:', completeJson);
      window.clearInterval(interval);
   });
```

.forget()

```
.node('*', function(){
   this.forget();
})
```

`.forget()` is a shortcut for .removeListener() in the case where the listener to be removed is currently executing. Calling `.forget()` on the Oboe instance from inside a `node` or `path` callback de-registers that callback.

`.forget()` 是 .removeListener() 的快捷方式，在当前正在执行要删除的侦听器的情况下。从 `node` 或 `path` 回调中对 Oboe 实例调用 `.forget()` 会注销该回调。

```
// Display only the first ten downloaded items
// but place all in the model

oboe('/content')
   .node('!.*', function(item, path){
      if( path[0] == 9 )
         this.forget();

      displayItem(item);
   })
   .node('!.*', function(item){
      addToModel(item);
   });
```

.removeListener()

```
.removeListener('node', pattern, callback)
.removeListener('node:{pattern}', pattern, callback)

.removeListener('path', pattern, callback)
.removeListener('path:{pattern}', pattern, callback)

.removeListener('start', callback)
.removeListener('done', callback)
.removeListener('fail', callback)
```

Remove any listener on the Oboe instance.

删除 Oboe 实例上的所有侦听器。

From inside node and path callbacks .forget() is usually more convenient because it does not require that the programmer stores a reference to the callback function. However, `.removeListener()` has the advantage that it may be called from anywhere.

从内部节点和路径回调 .forget() 通常更方便，因为它不需要程序员存储对回调函数的引用。但是， `.removeListener()` 的优点是可以从任何地方调用它。

.abort()

Calling `.abort()` stops an ongoing HTTP call at any time. You are guaranteed not to get any further `path` or `node` callbacks, even if the underlying transport has unparsed buffered content. After calling `.abort()` the `done` event will not fire.

调用 `.abort()` 可随时停止正在进行的 HTTP 调用。你保证不会得到任何进一步的 `path` 或 `node` 回调，即使底层传输有未解析的缓冲内容。调用 `.abort()` 后， `done` 事件将不会触发。

Under Node.js, if the Oboe instance is reading from a stream that it did not create this method deregisters all listeners but it is the caller's responsibility to actually terminate the streaming.

在 Node.js 下，如果 Oboe 实例正在从它没有创建的流中读取，则此方法会注销所有侦听器，但调用者有责任实际终止流式传输。

```
// Display the first nine nodes, then hang up
oboe('/content')
   .node('!.*', function(item){
      display(item);
   })
   .node('![9]', function(){
      this.abort();
   });
```

Pattern matching

Oboe's pattern matching is a variation on JSONPath. It supports these clauses:

Oboe 的模式匹配是 JSONPath 的变体。它支持这些条款：

`!`	Root object
`.`	Path separator
`person`	An element under the key 'person' 关键"人"下的元素
`{name email}`	An element with attributes name and email 具有属性名称和电子邮件的元素
`*`	Any element at any name 任何名称的任何元素
`[2]`	The second element (of an array) 第二个元素（数组的）
`['foo']`	Equivalent to .foo 相当于.foo
`[*]`	Equivalent to .* 相当于 。*
`..`	Any number of intermediate nodes (non-greedy) 任意数量的中间节点（非贪婪）

`$`	Explicitly specify an intermediate clause in the jsonpath spec the callback should be applied to 在应该应用回调的 jsonpath 规范中明确指定一个中间子句

The pattern engine supports CSS-4 style node selection using the dollar, `$`, symbol. See also the example patterns.

模式引擎支持使用美元、 `$` 符号的 CSS-4 样式节点选择。另请参阅示例模式。

Browser support

These browsers have full support:

这些浏览器完全支持:

- Recent Chrome
- Recent Firefox
- Recent Safari
- Internet Explorer 10 浏览器 10

These browsers will run Oboe but not stream:

这些浏览器将运行 Oboe 但不流式传输:

- Internet explorer 8 and 9, given appropriate shims for ECMAScript 5

  Internet Explorer 8 和 9，为 ECMAScript 5 提供了适当的填充程序

Unfortunately, IE before version 10 doesn't provide any convenient way to read an http request while it is in progress. It may be possible to add support for limited streaming in Internet Explorer 8 and 9 using the proprietary XDomainRequest but this object is has a reduced API and is buggy - only GET and POST are supported and it does not allow HTTP headers to be set. It also requires that responses set the `Access-Control-Allow-Origin` header and fails in IE8 for users browsing using InPrivate mode.

不幸的是，版本 10 之前的 IE 不提供任何方便的方式来读取正在进行的 http 请求。可以使用专有的 XDomainRequest 在 Internet Explorer 8 和 9 中添加对有限流的支持，但此对象具有简化的 API 并且存在错误 - 仅支持 GET 和 POST，并且不允许设置 HTTP 标头。它还要求响应设置 `Access-Control-Allow-Origin` 标头，并且在 IE8 中对于使用 InPrivate 模式浏览的用户失败。

The good news is that in older versions of IE Oboe gracefully degrades. It falls back to waiting for the whole response to return, then fires all the events instantaneously. You don't get streaming but the responsiveness is no worse than if you had used a non-streaming AJAX download.

好消息是，在旧版本的 IE 中，双簧管会优雅地降级。它退回到等待整个响应返回，然后立即触发所有事件。您没有获得流式传输，但响应速度并不比使用非流式 AJAX 下载差。

Improve this page 改进此页面

Is this page clear enough? Typo-free? Could you improve it?

这个页面够清楚吗? 无错字? 你能改进一下吗?

Please fork the Oboe.js website repo and make a pull request. The markdown source is here.

请分叉 Oboe.js 网站 repo 并提出拉取请求。降价源在这里。

Contact the author