Jax JavaScript HTTP Library

Release 5.5.0

February 28, 2017

Contents

Welcome to the Jax JavaScript HTTP Library documentation. Here, you should find what you need to get familiar with the library and start writing code with it. Follow the links below to get started.

CHAPTER 1

Overview

Introduction

The Jax JavaScript HTTP Library is an alternative lightweight JavaScript HTTP library. It can be installed next to and work alongside other JavaScript libraries, such as jQuery.

About the Library

Jax is an open source and is available on Github or as a stand-alone download from the website.

Community & Support

Being an open source project, Jax welcomes input and collaboration from the community and maintains an open dialogue with the community. Issues can be submitted to the appropriate repository on Github.

CHAPTER 2

User Guide

This section of the documentation will dive a little deeper and explore common concepts and ways to use the Jax JavaScript HTTP Library. The topics covered here are some of the more common programming topics that utilize a set of the more common components from the library.

HTTP

The Jax JavaScript HTTP Library has its own built-in component to handle HTTP requests and responses. The main `send()` method is:

- `jax.http.send(url, options)`

And the alias methods to send specific request methods are respectively:

- `jax.http.get(url, options)`
- `jax.http.head(url, options)`
- `jax.http.options(url, options)`
- `jax.http.post(url, options)`
- `jax.http.put(url, options)`
- `jax.http.delete(url, options)`

The options object can accept the the following properties:

- `method`
  - The method to use, GET, POST, etc. (defaults to GET)
- `headers`
  - Additional request headers to send
- `username`
  - A username, if required
- `password`
  - A password, if required
- `async`
  - Boolean flag to allow asynchronous operation
- `data`

  - Array or object of values to send over in the request
- `type`
  - Force a specific content-type from the response (will auto-detect otherwise)
- `status`
  - An object of status-based function handlers that will trigger based on the status of the return response
- `progress`
  - An function to trigger on the download (GET) or upload (POST) progress of a request
- `trace`
  - Push the response into a trace/debug function
- `fields`
  - Boolean flag to manage and set CSV field names from the first row of data

Here's a basic GET call to a URL:

```
var text = jax.ajax('/test.txt', {
    "status" : {
        "200" : function(response) {
            console.log(response.text);
        }
    }
});
```

If you wanted to control what happened on an error, you could set the status to:

```
var text = jax.ajax('/test.txt', {
    "status" : {
        "200" : function(response) {
            console.log(response.text);
        },
        "404" : function(response) {
            console.log('Whoops, something bad happened.');
        }
    }
});
```

Auto-Detect Content

Using the built-in auto-detect content feature, you can easily get a data object parsed from a URL with a specific content-type:

JSON content:

```
var json = jax.get('/test.json');
```

CSV content:

```
var csv = jax.get('/test.csv', {"fields" : true});
```

XML content:

```
var xml = jax.get('/test.xml');
```

Check HTTP Status Check

You can check a URL and get back only status information on it by using the HTTP status methods:

- `jax.http.status(url)`
- `jax.http.isSuccess(url)`
- `jax.http.isRedirect(url)`
- `jax.http.isError(url)`

```
if (jax.http.getStatus('http://www.mydomain.com/') == 200) {
    console.log('The URL is OK');
}
```

Browser

The Jax JavaScript HTTP Library has a `browser` object that helps to get browser-specific information from the user client and handle specific browser-related functions.

Open a New Window

You can open a new browser window using the method below. Accepted option object properties are:

- `width`
- `height`
- `scroll`
- `resize`
- `status`
- `location`
- `menu`
- `tool`
- `x`
- `y`

```
jax.browser.open('http://www.domain.com/', 'my-popup', {"with" : 640, "height" : 480});
```

Routing Based on Device

You can route to a specific URL based on the device detected. You can set the `desktop` URL property, the `mobile` URL property for a all-inclusive mobile URL, or you can set a device-specific URL property

```
var options = {
    "desktop" : 'http://www.mydomain.com/',
    "ipad"    : 'http://www.mydomain.com/ipad',
    "iphone"  : 'http://www.mydomain.com/iphone',
    "android" : 'http://www.mydomain.com/android',
}
jax.browser.route(options);
```

Cookie

The Jax JavaScript HTTP Library has a `cookie` object that helps to manage cookie values for the user client.

Storing

You can store a cookie using the method below. Accepted option object properties are `expire`, `domain` and `path`.

```
jax.cookie.save('foo', 'bar', {"expire" : 300});
```

Retrieving

Then you can access the cookie value like this:

```
var foo = jax.cookie.load('foo');
```

Deleting

And remove the cookie value like this:

```
jax.cookie.remove('foo');
```

Storage

The Jax JavaScript HTTP Library has a `storage` object that helps to manage local storage values for the user client.

Storing

You can store a value in local stroage using the method below:

```
jax.storage.save('foo', 'bar');
```

Retrieving

Then you can access the value in storage like this:

```
var foo = jax.storage.load('foo');
```

Deleting

And remove the value in storage like this:

```
jax.storage.remove('foo');
```