|  |
| --- |
| CHARMe Plugin |
| Integration Guide |

**v0.5.1**

**7th Oct 14**

Table of contents

[1 Overview 1](#_Toc382230001)

[1.1.1 CHARMe Plugin Bootstrapper (charme.js) 1](#_Toc382230002)

[1.1.2 CHARMe Plugin 1](#_Toc382230003)

[1.2 External Libraries 2](#_Toc382230004)

[2 Installation 3](#_Toc382230005)

[2.1 Setup 3](#_Toc382230006)

[2.2 Installing the Plugin 3](#_Toc382230007)

[3 Integration 4](#_Toc382230008)

[3.1 Including charme.js 4](#_Toc382230009)

[3.2 Marking up the datasets 4](#_Toc382230010)

# Change log

v0.5.1:

* Added new section 3.3 - ‘Making the CHARMe Plugin window moveable’
* Updated sections 4.2 - ‘User Issues’, and 4.3 - ‘Outstanding Bugs’
* Various minor corrections to text and layout

# Overview

The CHARMe (CHARacterization of Metadata) project aims to improve knowledge sharing in the climate sciences community and to enable users of climate data to judge their fitness for purpose. For more details see the CHARMe project website: <http://www.charme.org.uk/>.

The CHARMe system consists of several components, including the CHARMe Plugin and the CHARMe Node. This document focuses on the CHARMe Plugin and the means for integrating it with an existing site. The CHARMe Plugin provides a user interface to allow in-situ viewing and creation of CHARMe annotations, and is intended to be used alongside datasets etc. hosted by a data provider.

**Note that this version of the document applies to version 0.5.1 of the Plugin.**

As it is intended to be integrated with existing sites with minimal effort, the CHARMe Plugin is designed to be as agnostic as possible of the hosting environment. As such it is implemented purely in JavaScript, CSS, and HTML. It imposes no additional requirements on the data provider, and has no external dependencies. All that is required to integrate the CHARMe Plugin into an existing web page is to include one JavaScript file, and then markup the sections of HTML where the CHARMe Plugin should be inserted. This process is detailed in the Installation section.



Figure 1: The CHARMe Plugin scripts. Only charme.js needs to be included, the rest are included automatically.

The CHARMe Plugin has a number of dependencies on common libraries such as jQuery and Angular.js, however these are all distributed with the plugin, and resolved internally. The plugin runs within an iFrame which means that it is completely isolated from the hosting site, avoiding both conflicts with existing JavaScript frameworks, and global namespace pollution.

### CHARMe Plugin Bootstrapper (charme.js)

This is a lightweight JavaScript component that is the sole include required on the integrator’s site. This script bootstraps the rest of the plugin, which is loaded in a sandboxed iFrame environment. The CHARMe Bootstrapper has no dependencies on JavaScript libraries, and all dependencies required by the plugin are isolated inside the iFrame.

### CHARMe Plugin

Due to the necessary complexity of the JavaScript behind the CHARMe Plugin, it is organized into many sub-components. External libraries have been used where possible to minimize the amount of boilerplate code required, and to ensure cross-browser support. The most notable examples of this are the use of jQuery, which provides many convenience functions not natively provided by the JavaScript language; Angular, which provides a framework for building rich JavaScript applications; and json-ld, which provides basic but essential JSON linked-data support. All required libraries are packaged and distributed with the CHARMe Plugin. When deployed, these libraries will reside in a CHARMe directory to avoid conflict with other existing libraries that the integrator might already be using. This is especially important in the case of jQuery which is used extensively. Because the CHARMe Plugin operates within a completely isolated iFrame, there is no conflict with existing versions of the libraries on the integrator’s site.

## External Libraries

|  |  |
| --- | --- |
| Angular | Angular is a JavaScript library provided by Google. It is a framework within which rich JavaScript applications can be developed following a prescriptive Model View Controller style approach. This results in cleaner, more modular JavaScript and HTML, with enforced separation of concerns. |
| Bootstrap | Bootstrap provides a number of commonly used user interface elements via CSS and JavaScript. This allows for a simple, clean, and attractive interface without having to define boilerplate CSS styles and JavaScript behavior for common use cases. The Bootstrap framework was developed by Twitter and is the foundation of that site’s user interface. |
| jQuery | jQuery provides convenience methods that extend the default functionality of the JavaScript language. In particular it allows for powerful querying of the DOM in a cross-browser way, making progressive enhancement much simpler. |
| JSON-LD | The JSON-LD library provides functions for simple manipulation of linked data graphs. In particular, it allows for normalization of graphs into a flattened form, which makes processing graphs easier and more performant. |
| Promise.js | This library is a requirement of jsonld.js, and is an implementation of the ‘Promises’ software pattern. The promises pattern is a means of deferring execution of code until some asynchronous event has completed. This is often achieved via callbacks, but there has been a tendency toward the promises model of late, as it makes code clearer, and allows for chaining of events.  This particular library is Alex Russell’s implementation (<http://infrequently.org/2013/06/sfuturepromiseg/> ) of the A+- Promises standard - which is being adopted by browser vendors as we speak, so is preferable to jQuery’s deferred framework (a similar, but non-compatible concept) as it is standard compliant. |

# Installation

## Setup

The CHARMe Plugin is distributed as a zip file, or can be built from source. Currently the source and distributable zip are available in the CHARMe Plugin GitHub repository (https://github.com/charmeproject/charme\_plugin). The instructions below assume that you are installing from the prebuilt distributable that is available, along with the latest version of this file, from **https://github.com/charmeproject/charme\_plugin/tree/master/CHARMe\_plugin/dist**.

## Installing the Plugin

1. Unzip charme.plugin.zip to a temporary directory. All of the files will be unzipped into this base directory.
2. The CHARMe Plugin code now needs to be copied into the web application with which it will be integrated. This directory can be anywhere that is publicly accessible from the web server. For example, if your JavaScript is in a directory called ‘js’, then you might copy the CHARMe directory underneath it, e.g.

* index.html
* pages/
* js/**charme**
* styles/

Alternatively, you might give it its own top level directory, e.g.:

* Index.html
* pages/
* js/
* styles/
* **charme/**

1. The CHARMe Plugin uses **OAUTH 2.0** for authenticating with the CHARMe Node. This requires that each installation of the CHARMe Plugin has a unique **client ID**. Please contact STFC if this is your first time integrating with the CHARMe Node, as you will need to be provided with a client ID. Please note that when creating a new OAUTH client, you will be asked to provide a redirect URL. The URL used here should be the externally accessible location of the **redirect.html** file located under the **plugin/** directory, e.g.

http://<host name>/js/charme/plugin/redirect.html

1. It is now necessary to modify a few settings in the CHARMe Plugin, to customize it to the deployment environment. Under the root **charme/** directory, there is a JavaScript source file called **charme.settings.js**. Open this file, and locate the setting charme.settings.REMOTE\_BASE\_URL. Set the value of this to the URL at the root of the CHARMe Node web services. During the testing phase, this will be **https://charme-test.cems.rl.ac.uk/.** Next, locate charme.settings.AUTH\_CLIENT\_ID and set it to the client ID created for your CHARME Plugin installation, e.g.

charme.settings.REMOTE\_BASE\_URL='https://charme-test.cems.rl.ac.uk/';

charme.settings.AUTH\_CLIENT\_ID='12345';

# Integration

## Including charme.js

Once installed, integration of the CHARMe Plugin with an existing site is relatively straightforward. The CHARMe Plugin bootstrap file will need to be included from any page from which the plugin will be available. This simply requires charme.js to be included wherever other JavaScript includes are present in the HTML. This is typically in the <head> section of an HTML page, or toward the end of the <body> section. Although CHARMe is designed to be as efficient as possible, it is recommended that in order to minimize the performance impact on the existing site that the JavaScript is included at the end of HTML pages, e.g.

<!DOCTYPE HTML>

<html>

<head>

<title>CGI Earth Observation</title>

<!--

Scripts, styles, meta tags etc.

-->

</head>

<body>

<!--

Site content

-->

<script src="/js/charme/charme.js"></script>

</body>

</html>

charme.js included here

## Modifying your HTML to support the CHARMe Plugin

The CHARMe Plugin is used to view and create annotations associated with ‘targets’ such as datasets. Typically, the way this would work is that a small CHARMe icon C:\Users\henryad\Documents\Projects\CHARMe\http\DAV\js\charme\activebuttonsmall.pngappears next to the targets where they are published on a data provider’s website. A clear icon indicates that the CHARMe Plugin is available to create annotations, but that none currently exist. A blue filled icon indicates that annotations already exist for the target. As well as the CHARMe icon, a checkbox will also be inserted in order to allow multiple targets to be selected. To integrate with an existing site, the CHARMe Plugin needs to be told three things about each target:

1. Where in the page to insert a CHARMe button and checkbox
2. The URI of the target that this button will relate to
3. The type of the target

Targets are specified through the use of anchor tags. Wherever you wish a CHARMe icon to appear on the page, simply insert an anchor tag and set the “href” attribute to be the URI of the target. Finally, the type of the target should be specified using the appropriate type class. The supported type classes are enumerated in Table I. An example anchor tag:

<a href="http://localhost:8090/DAV/NASA/Chlorophyl/2002/MY1DMM\_CHLORA\_2002-07.JPEG" class="charme-dataset"></a>

These anchor tags can be inserted anywhere on the page that a CHARMe icon should appear. Ideally, this would be next to the target links themselves.

Finally, CHARMe requires a **div element with an id of ‘charme-placeholder’** to be inserted somewhere in the page. This div will be used as the location into which a ‘select all’ checkbox will be inserted. This step is important, as without this placeholder div, the CHARMe Plugin will not load. For example:

<div id=*"charme-placeholder"*></div>

|  |  |  |
| --- | --- | --- |
| Target Type | Target Class | Target Type URI |
| Dataset | **charme-dataset** | http://purl.org/dc/dcmitype/Dataset |
| Technical Report | **charme-technical-report** | http://purl.org/spar/fabio/TechnicalReport |
| Dataset Collection | **charme-metadata-document** | http://purl.org/spar/fabio/MetadataDocument |
| Discovery Metadata | **charme-metadata-document** | http://purl.org/spar/fabio/MetadataDocument |
| Browse Metadata | **charme-metadata-document** | http://purl.org/spar/fabio/MetadataDocument |

Table I: Supported CHARMe annotation target types. The 'Target Class' should be used as the class of the anchor tag in order to specify the type of the target.

## Making the CHARMe Plugin window moveable

It is possible for the iFrame in which the plugin is displayed to be freely dragged across the screen with a mouse, allowing the user to view the screen content underneath (the iFrame may also be minimized). However, as this is not a standard feature of iFrames, the CHARMe plugin includes additional code to enable it, and a third script – dragiframe.js (after charme.settings.js and charme.common.js, see Figure 1) – is automatically included in your web page. However, this will cause a conflict if the web page defines its own ‘onmousemove’ handler, e.g.

<div onmousemove="myFunction()"></div>

as this original function will by necessity be overwritten. Therefore, to avoid any such conflict, the dragging feature is not enabled by default. To enable it, you must simply include the **class ‘charme-draggable’** in the placeholder div element mentioned in the previous section, e.g.

<div id=*"charme-placeholder"* class=*"charme-draggable"*></div>

## Refreshing the CHARMe Plugin Icons

It is possible to programmatically refresh the CHARMe plugin icons on the Data Provider’s page without having to reload the entire webpage again from the browser. CHARMe provides a special function that can be invoked to trigger a rescan action on only the CHARMe elements on a page. This functionality would be useful if a data provider’s webpage was designed to dynamically acquire new target items, after the initial page load. Invoking the Rescan function will (a) Update the annotation count on each of the existing plugin access icons attached to all the targets, (b) create plugin icons for new targets that have appeared on the screen. Invoking the rescan function will temporarily set all the charme plugin icons to the “loading” icon [ ] whie the annotation count for each target is re-fetched.

To invoke the rescan, simply call the following function :-

charme.plugin.rescanPage()

An example code snippet is shown below, where the recan function has been attached to the onclick event of a button on the webpage.

<div>

<button onclick="charme.plugin.rescanPage()">Rescan</button>

</div>

## Browser Support

The CHARMe Plugin has been tested in the following browsers, and is known to work:

* Mozilla Firefox
* Google Chrome
* Internet Explorer 11

# Known Issues

## Integration Issues

### Overlapping of other elements

In order for the CHARMe Plugin to appear on top of the other contents of the screen it will need to have the highest ‘z-index’. This has been set to a very high value, however it’s possible that if the site you are integrating with has elements with the same or higher z-index that these may overlap and partly obscure the plugin. The only workaround for this is to decrease the z-index of the other elements on the screen.

### Viewing older annotations

Datasets annotated in previous versions of the plugin prior to the use of https will not be retrievable. Although the targets will appear to have annotations and will be searchable, attempting to view these annotations will result in an error. This will not be a problem with newer data, or any future annotations.

## User Issues

### Internet Explorer

To use the CHARMe plugin in Internet Explorer, each user must add your website to their list of ‘Trusted Sites’, in Windows 7 browser versions this is found in:

Tools > Internet Options > Security tab

(see e.g. [**http://windows.microsoft.com/en-gb/windows/security-zones-adding-removing-websites#1TC=windows-7**](http://windows.microsoft.com/en-gb/windows/security-zones-adding-removing-websites#1TC=windows-7) for more details).

They must then also select the ‘Enable Protected Mode’ option (in the same tab).

Furthermore, although the CHARMe Plugin works in Internet Explorer 11, this browser (and IE10) does not store a user’s browsing history within iFrames. Therefore the browser’s ‘back’ and ‘forward’ buttons will not work as the user might expect while using the plugin. This is a fundamental issue with IE, rather than with CHARMe.

## Outstanding Bugs

### Internet Explorer 8, 9, 10

The plugin does not currently work correctly in IE8 or IE9. Some as yet undiscovered problems may occur with IE10.