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

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)

# Overview

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 hosted by a data provider.

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.



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 conflicts with existing javascript frameworks, and global namespace pollution.

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

### CHARMe Plugin Bootstrapper (charme.js)

This is a lightweight javascript component that is the sole include required on the integrators 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, the required 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 it’s preferable to JQuery’s deferred framework (a similar, but non-compatible concept) as it’s standard compliant. |

# Installation

## Setup

The CHARMe plugin is distributed as a zip file, or can be built from source. Currently the source is available on the CHARMe Plugin github repository. The distributable zip file will also be made available from github in the future. The instructions below assume that you have the distributable zip file already via other means.

## Installing the Plugin

1. Unzip charme.plugin.zip to a temporary directory. This should produce a single directory called **charme** with multiple subdirectories
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, eg.

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

Alternatively, you might give it its own top level directory, eg:

* 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**. The process of creating a new OAUTH client is outside of the scope of this document. Please note that when creating a new OAUTH client entity, you will be asked to provide a redirect URL. The URL used here should point to the **redirect.html** file located under the **plugin/** directory. Eg.

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 located the setting charme.settings.REMOTE\_BASE\_URL. Set the value of this to the URL at the root of the CHARMe node web services. Next, locate charme.settings.AUTH\_CLIENT\_ID and set it to the client ID created for this CHARME Plugin installation. Eg.

charme.settings.REMOTE\_BASE\_URL='http://charme-dev.cems.rl.ac.uk:8027/';

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 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 possibly, 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.

Eg.

<!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

## Marking up the datasets

The CHARMe plugin is used to view and create annotations associated with datasets. Typically, the way that this would work is that a small charme icon C:\Users\henryad\Documents\Projects\CHARMe\http\DAV\js\charme\activebuttonsmall.pngappears next to the datasets 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 dataset. To integrate with an existing site, the CHARMe plugin needs to be told two things:

1. Where in the page to insert a charme button
2. The URI of the dataset that this button will relate to.

This is achieved through the use of anchor tags. Wherever you wish a CHARMe icon to appear on the page, simply insert an anchor tag with a class attribute that includes the class “charme-dataset”. Set the “href” attribute to be the URI of the dataset. Eg.

<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 dataset links themselves.

# Known 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 that 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.