

Twingly Widget



INTRODUCTION

Author: Björn Milton (bjorn.milton@twingly.com)
Rev: 2.6
2009-10-21

CONTENTS

Introduction	2
What is a widget?.....	2
Why use a widget?	2
Reliability.....	2
Moderation	2
Basic setup for all widgets.....	3
Common configuration	3
Styling the widgets.....	5
Known incompatibilities.....	5
Changelog.....	5
Version 2.0.1	5
version 2.0	5
version 1.4.2	6
version 1.4.1	6
Version 1.4	6
Version 1.3.2	6
Version 1.3.1	6
Version 1.3	6

INTRODUCTION

This paper is a short introduction to widgets in general and the widgets offered by Twingly in particular. This document describes the basic notion of a widget and why it is a good development paradigm. It also describes the technical platform and the setup needed for using any of the widgets. The specific details for the widgets are distributed in separate documents.

WHAT IS A WIDGET?

A widget consists of a piece of HTML code which, when inserted in a website, expands into a more complex entity. All logic takes place on the client side (the web browser), saving server resources. Behind the scenes, a client script and a stylesheet are downloaded to the browser. The client script provides the logic and generates all the necessary HTML code to let the user interact with the widget.

WHY USE A WIDGET?

A widget is very easy to integrate - just a short HTML snippet needs to be inserted into the existing code. Also, it is very easy to maintain: all changes, bug fixes and updates can be done transparently by the company providing the widget.

RELIABILITY

Widgets are very easy to integrate into any website with almost no work for the customer. Blog data and the widget scripts themselves are hosted on an extremely reliable and distributed server farm provided by Amazon Inc. This ensures full scalability and stability.

MODERATION

A blog or blog post that for some reason is deemed as inappropriate can be removed very easily. If a blog is removed, none of its posts will ever show up again.

All of the above is done through an easy to use interface, *Twingly Administration Center* (a document describing Twingly Administration Center can be found here:

http://static.twingly.com/docs/Twingly_Administration_Center_User_Guide.pdf).

User moderation has proven to be a very efficient and powerful tool for detecting and removing inappropriate content. With the moderation button a user can easily specify and submit the nature of a complaint for a specific post. The reports will be displayed in the *Abuse report* view in *Twingly Administration Center*.

BASIC SETUP FOR ALL WIDGETS

To run any of the widgets, you'll need to include a javascript file found at <http://eu.widgetdata.twingly.com/scripts/widget/twingly.widget.2.0.1.pack.js>.

To include this file, copy and paste the following html code into your page:

```
<script type="text/javascript"
src="http://eu.widgetdata.twingly.com/scripts/widget/twingly.widget.2.0.1
.pack.js"></script>
```

This code snippet should be included at the end of the html file just before the closing </body> tag.

```
<html>
  <head>
    <script type="text/javascript">
      //The configuration for the widgets should be put here
    </script>
    <!--Head section of your html file-->
  </head>
  <body>
    <!--Body section of your html file-->

    <!--Include the widget javascript file at the end of the body-->
    <script type="text/javascript"
src="http://eu.widgetdata.twingly.com/scripts/widget/
twingly.widget.2.0.1.pack.js"></script>
  </body>
</html>
```

Also note that there is a section in the head part of the html file that we will use for configuring the widget. Configuring the widgets is done by assigning values to different java script variables. Different variables apply to different widgets and they will be explained in detail in the widget specific sections. There are some variable that apply to all widgets. They are explained below.

COMMON CONFIGURATION

- `tw_language` - (string) Used to change the language in the widget interface. Default is English.
NOTE Please let us know if you have a request for a certain language that is not supported yet.

```
var tw_language = "swedish";
```

- `tw_exposejQuery` - (boolean) If true, the Javascript variable "jQuery" will be set to the widget's internal version of jQuery. Does not set the variable "\$". Use this feature carefully as the version of jQuery can be changed between two versions without notice. As of version 1.3.1 the old variable "tw_noConflict" is removed because of how we encapsulate jQuery in the widget. Default is false and jQuery will not be visible to any other script on the page.

```
var tw_exposejQuery = true;
```

- `tw_onComplete` – (function) Used as a hook for hooking up a function that will be executed when the widget is done executing. Can for example be used to display the number of links in another part of the page using `tw_numberOfPosts` (see the Twingly Blogstream document for details about `tw_numberOfPosts`).

```
var tw_onComplete = function () {
    alert('We got ' + tw_numberOfPosts +
        'posts linking to this page');
};
```

- `tw_onReportDone` – (function) Used as a hook for hooking up a function that will be executed when the user has made an abuse report. Can for example be used to display a confirmation that the abuse report has been received.

```
var tw_onReportDone = function () {
    alert('Report received!');
};
```

- `tw_onItemComplete` – (function) Used as a hook for hooking up a function that will be executed each time an item is done. The definition of an item is the repeating entities in the widget. In the case of the Blogstream widget it would be the html that represents on blogpost (marked by a red square in the picture).



This function hook is different since it needs to handle four arguments: `i`, `data`, `html` and `mode`. If the function returns something it will be treated as the html for the item.

`i` is the sequence number of the item. It will just increase by 1 for each item in the widget. `data` is the data structure for the item. `html` is the generated html for the item. `mode` indicates what kind of widget the call is coming from. If, for example, there are both a Blogstream widget and a Latest blogged widget on the same page, both of them will generate calls to `tw_onItemComplete`. To be able to know what widget made the call we have to check the mode parameter. The mode is a string and can have the following values: `links`, `toplist` or `latestblogged`.

The example below will put an image to the left of each item that links to the post belonging to the current item. This will be done for items contained in a Blogstream widget.

```
var tw_onItemComplete = function (i, data, html, mode) {
    if (mode == 'links') {
        var imageUrl = 'http://someurl.com/image.png';
        return '<a href="' + data.url + '" title="' +
            data.title + '">' +
            '</a>' + html +
            '<div style="clear:both;"></div>';
    }
};
```

STYLING THE WIDGETS

It is possible to change the look and feel of the widgets to make fit right into the design of the rest of the website.

Please note that we require that the logo and link to Twingly remains visible as stated in the user agreement.

The CSS file used by the widgets is located at

<http://eu.widgetdata.twingly.com/scripts/widget/css/widget2.0.css>

Pick the attributes you want to modify and put them into a file on your local servers. Use the following settings to be sure that your modified styles are used instead of the defaults:

```
//Note that tw_skipDefaultCss should only be set if you don't want
//any of the default styles applied. This option should be used with
//care since the default settings are well tested.
var tw_skipDefaultCss = true;

var tw_localCss = "http://your.domain.com/styles/mywidget.css";
```

And now you are almost done! After you followed the few steps in the product specific documentation (*Blogstream, Toplist, Latest Blogged or Related*) you should see the widget appear at the position in the page where you put the HTML snippet mentioned above.

KNOWN INCOMPATIBILITIES

- i. Older versions of script.aculo.us and Prototype

The widget does not work together with older versions of effects.js in the Javascript library script.aculo.us, because of a bug in that library. We recommend running the latest version, or at least version 1.8.1. Unfortunately we cannot fix this problem on our side.

CHANGELOG

Below is a brief description of new features and changes between versions, with most recent version first.

VERSION 2.0.1

- i. Fixed an internal bug that caused problems in some rare usages of the widget.

VERSION 2.0

- i. Added a new widget, Latest Microblogged, for displaying microblogs in temporal order that has linked to a given site.
- ii. Changed the default value for tw_useToolTip to false.

- iii. Changed so that the widget now use widget2.0.css

VERSION 1.4.2

- i. Fixed an URL encoding problem in the “show all” link in the Blogstream widget.

VERSION 1.4.1

- i. Added support for HTTPS. Please contact us if you need to include the widget on a HTTPS site.
- ii. Added Latest blogged translations for Latvian and Estonian.

VERSION 1.4

- i. Added the Blogwatch widget.
- ii. Fixed an issue with padding, which made the widget wider than an outer element with a specified width. The widget should now behave better. **This change will make the widget slightly narrower than before.**
- iii. Small change to the DOM structure: added a div with class “tw_showall” around the Show all link (Latest Blogged and Related).
- iv. Use of new css, widget1.4.css.

VERSION 1.3.2

- i. Added target=_blank to twingly.com links, which we didn't do in all cases.

VERSION 1.3.1

- i. Better encapsulation of jQuery in the widget. Removed the variable tw_noConflict and introduced a new variable, tw_exposejQuery. See the documentation for more information.
- ii. Fixed a bug that caused the pager to not work without setting tw_pagerLimit.
- iii. Added target=”_blank” to blog and blog post links in the Latest Blogged Widget, as they have in the other widgets.

VERSION 1.3

- i. Changed the default report button.
- ii. Use of new css, widget1.3.css.
- iii. tw_noConflict is now enabled by default.