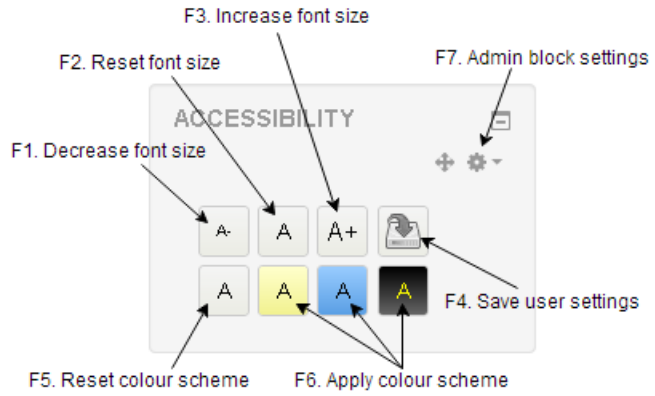
Moodle Accessibility Block
DEVELOPERS DOCUMENTATION

```
Accessibility Block for Moodle

Provides options for changing text size and colour scheme. Settings can be
saved to persist between sessions.

Also integrates ATbar from Southampton University ECS
http://www.atbar.org. Code used under BSD Licence.

To install, place all files in /blocks/accessibility and visit
/admin/index.php in your browser.

This block was written by Mark Johnson <mark@barrenfrozenwasteland.com>
It is copyright of Mark Johnson, Richard Taunton's Sixth Form College and
contributors.

Translations are copyright of their respective authors, as indicated in
each language file.

Current languages are English, German, Spanish, Croatian and French.  If
you're interested in adding a translation please raise a bug on Github
https://github.com/marxjohnson/moodle-block_accessibility/issues

Released Under the GNU General Public Licence
http://www.gnu.org/copyleft/gpl.html
```

Introduction

This documentation is targeted to further Accessibility Block developers. To find out more about the block, please refer to GitHub repository[1]. This documentation brings the basic working principles and program flows. To use or edit diagrams in this document, please refer to draw.io[2].

Here is a quick introduction view into the Accessibility Block plugin. Figure 1 shows the basic functionalities (action buttons F1, F2… F7) as follows:



F1. Decrease font size
F2. Reset font size and erase db record
F3. Increase font size
F4. Save user settings to db
F5. Reset scheme and erase db record
F6. Apply colour schemes
F7. Block settings (admins only)

Figure 1 Block functionalities

Using Accessibility Block, each user can set font size and colour scheme for their current session. Settings can persist between sessions if user saved it permanently using the "save" button (F4). These are all *per-user settings* that are stored in *$USER* variable or in the database in case the user saved it permanently. There are also *block instance settings* that can be set by administrator. *Block instance settings* are standard block settings achieved through edit_form.php script. It defines the colour profiles configuration for the block instance, ATbar appearance and auto-save option.

To achieve block appearance throughout entire Moodle site (so-called "Sticky block"), once the plugin is installed into Moodle, it's recommended to add its instance onto the **Moodle homepage**. Homepage instance of the block enables *"Page contexts*" option in *block instance settings* (F7) to be set to *"Display throughout entire site*".

If you are about to develop the code, you might want to make sure to have debugging enabled in *Site administration* > *Development* > *Debugging*. If you want to edit the code in Javascript, you might want to consider disabling caching in *Site administration* > *Appearance* > *AJAX and Javascript*. Read more about the *developer mode* in Moodle documentation[3].

[1] https://github.com/marxjohnson/moodle-block_accessibility
[2] https://drive.google.com/file/d/0BwrUG2o1JsD_cmt0X1JWWDN2dTg
[3] http://docs.moodle.org/dev/Developer_Mode

Basic working principle

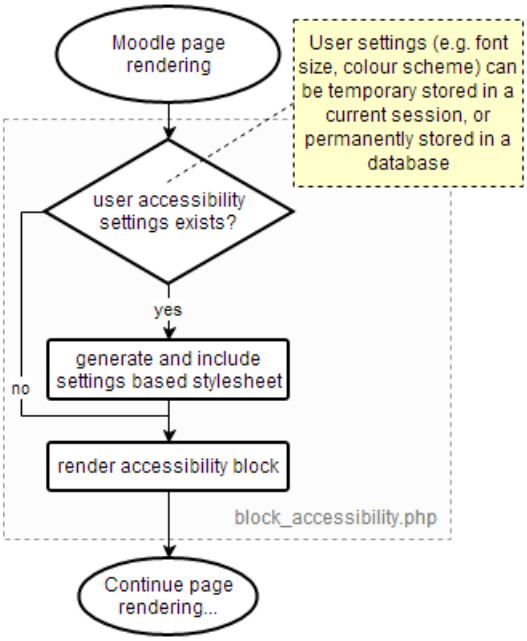


Figure 2 Basic working principle

Moodle system generates a page which is displayed to a user. It uses *accessibility_block.php* plugin script to render and include Accessibility Block component within it. The script determines if user settings (e.g. font size, colour scheme) already exist and applies it onto a page as shown by Figure 2. If no user settings exist, no custom styles are applied until user interacts to the block using action buttons (F1, F2… F6). As user interacts to the block, the block needs to interact to Moodle system to apply changes and save session settings. This can be achieved in two modes:

1. Dynamic AJAX mode (using Javascript)
2. Standard HTTP mode (using redirects)

AJAX mode is enabled by default and requires user to have Javascript enabled web browser. Otherwise, HTTP mode is also available (satisfying Moodle coding guidelines[4]) which requires page reload for any user interaction such as saving settings, applying custom styles and changing the font size.

When clicked, each of F1-F6 action buttons calls one of corresponding .php scripts (*database.php, changesize.php or changecolour.php*) that handle the request as shown by Figure 3:

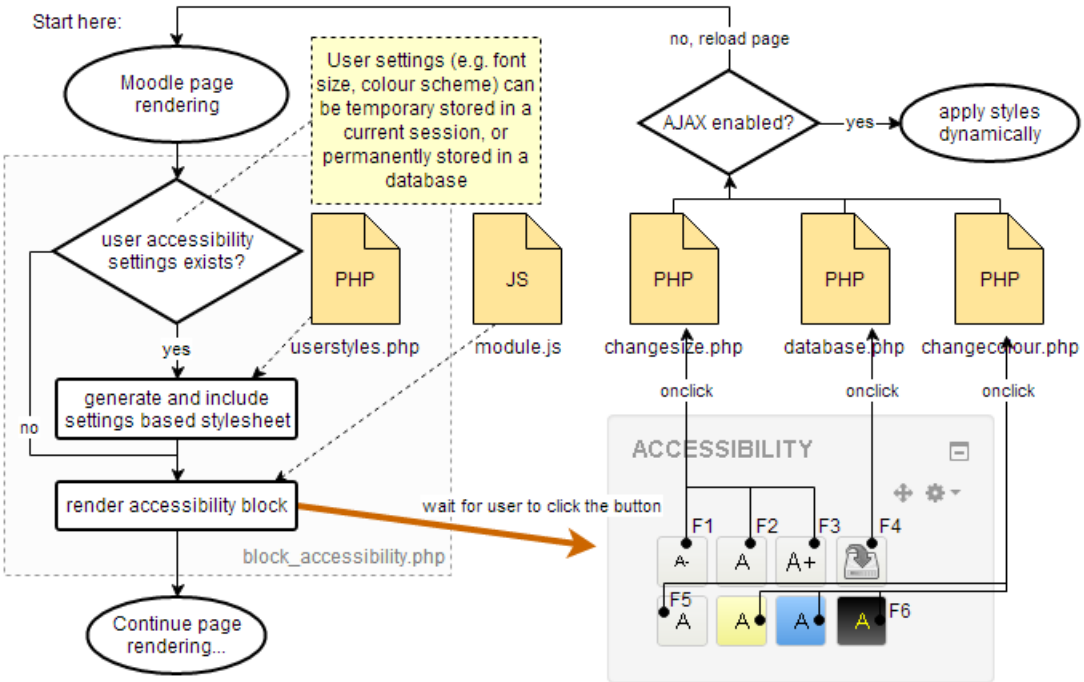


Figure 3 Accessibility block onclick actions

[4] http://docs.moodle.org/dev/JavaScript_guidelines

Userstyles.php script is used to read settings stored by user and generate the stylesheet based on those settings to be included onto a page. *Module.js* script accompanies Javascript functionalities used by block in AJAX mode. AJAX mode will enable browser to call corresponding scripts in the background and applying generated stylesheets with no need for page reload.

block_accessibility.php script

The *block_accessibility.php* script is a core of a plugin. It is used to setup and render a block action buttons and blocks layout to be displayed on a Moodle page. Immediately after block initialization, *userstyles.php* script is called to read if user settings are stored to apply styles onto a page. Figure 4 illustrates the code more in detail:

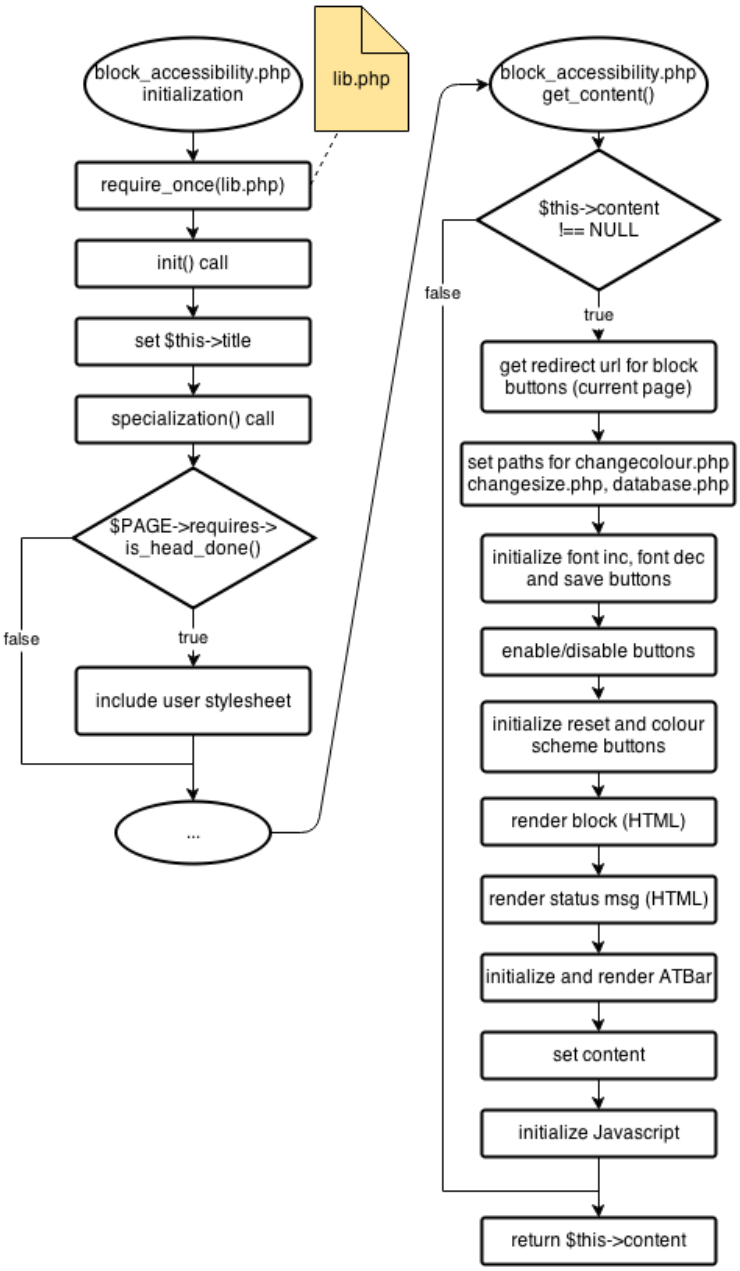


Figure 4 Moodle Accessibility Block initialization

userstyles.php script

The *userstyles.php* script is the cornerstone of the block. When the page loads, it checks if the user has a custom settings for the font size and colour scheme (either in the session or the database) and creates a stylesheet to override the standard styles with this setting. Figure 4 illustrates the code more in detail:

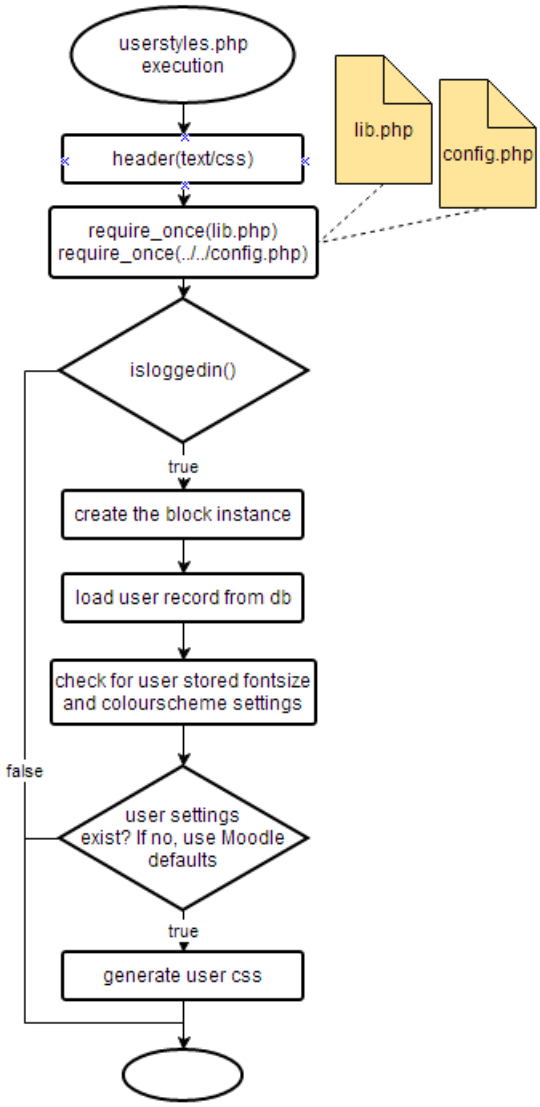


Figure 5 Userstyles.php script

changesize.php and changecolour.php scripts

The *changesize.php* and *changecolour.php* scripts are used to process onclick actions for F1, F2, F3 and F6 buttons of the block as shown in Figure 4. The scripts are used to save new user settings values to a current session so that settings persist even if page is reloaded again. *Changesize.php* receives argument through HTTP GET request so it can perform one of the following actions: increase font size, decrease font size or reset font size. *Changecolour.php* receives chosen scheme id through HTTP GET request (1, 2, 3 or 4). If received id is 1, the scheme reset will be performed, otherwise, chosen scheme will be stored. Figure 6 illustrates the code more in detail:

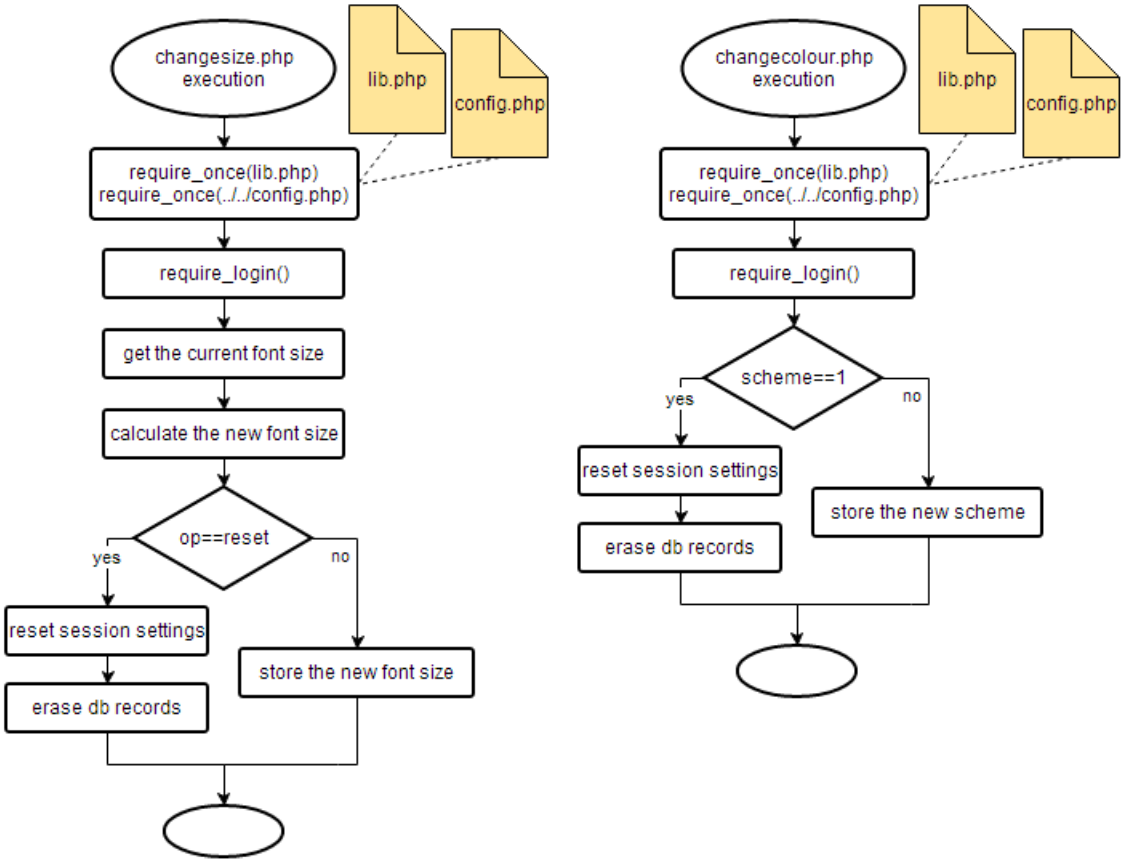


Figure 6 Changesize.php and Changecolour.php script

database.php script

The *database.php* script basically perform one of the two actions: save and reset. It communicates to the database and stores and/or updates per-user settings: ATbar appearance, colour scheme id or font size. Action save/reset and settings to perform action on are passed through HTTP GET arguments. Figure 7 illustrates the database table scheme for per-user settings.

1	**id**	bigint(10)
2	**userid**	bigint(10)
3	**fontsize**	decimal(4,1)
4	**colourscheme**	tinyint(1)
5	**autoload_atbar**	tinyint(1)

Figure 7 Per-user settings in the db

Document revisions			
Hrvoje Golčić hrvoje.golcic@gmail.com	1.0	02.05.2014.	Initial document written by Hrvoje Golčić