Scripts and AJAX

Scripts that is used within the plugin is located in the ResursBank\Module\Data class.

Variable	Script description
`$jsLoaders`	Default loaders that is usually added to the user sections for WordPress.
`$jsLoadersCheckout`	Scripts that is only loaded in the head, at checkout section.
`$jsLoadersAdmin`	Script loaded in the admin section.
`$jsDependencies`	Every script that has dependencies in other scripts should be added here. For example if some of the basic scripts needs jquery it is added like this: `private static $jsDependencies = ['resursbank' => ['jquery']];`
`$jsDependenciesAdmin`	As the regular dependency variable above, but for admin section.
`$styles`	Every css that needs to be loaded.
`$stylesAdmin`	Css that resides in the admin section.

Default scripts

The built in colleciton of scripts provides a few scripts for handling the basic calls of the plugins. The bundled scripts are listed below.

Script	Description
js/resursbank.js	Default front page script. Customer based non admin.
js/resursbank_admin.js	Admin (wp-admin) based scripts. Nothing peculiar actually, it's just code that doesn't have to be used in the front to save performance, etc.
js/resursbank_global.js	Global script. Loaded both with admin and front page. Contains amongst others an ajaxfier. We could create separate ajax calls for each thing we'd like to do, but that will in the end cost more space and the calls may look different each time they are used. I don't see why we would need that. See below in the AJAX calls section how this works.

Data and translations in frontend

To get urls, data or translations from the localization generator, use getResursLocalization('key-name'). If you are in admin, the frontend scripts will choose data from either the global set or the admin set. It is likewise for the public store front, where we instead fetch data from the global set or the "store front set".

AJAX calls

The action names of ajax calls are always prefixed with "resursbank_". This means, if you for example would like to make an ajax call that do maths in the backend, the action would look like *resursbank_do_maths*. To simplify life, this request in the frontend script goes through the ajaxifier that resides in resursbank_global.js, like this:

```
getResursAjaxify(
    'POST',
    'resursbank_do_maths',
    {
        'firstvalue':2,
        'secondvalue':2,
        'calctype': 'multiply'
    },
    function(actionResponse) {
        // My actions here.
    }
);
```

The ajax receiver for internally supported requests is very much automated from WordPress::setupAjaxActions. Each action name are added with the prefix rbwc_ as mentioned in the Actions and filters section. Each snake_case-action are from there translated to a camelCase action, so that the entire codebase looks the same. The default actions all resides in the PluginApi-class in the mentioned camelCase format. To add more, you could continue add methods like this, or hook up with your own ideas via external plugins.

Nonces in AJAX calls

Nonces are always added in the localizations and the ajaxifier itself so each call can be validated if necessary. For example, when we validate credentials from wp-admin, we always use the nonce to avoid problems.