vingd-popup Documentation

Release 0.8.5

Radomir Stevanovic, Vingd Inc.

December 12, 2013

CONTENTS

CHAPTER

ONE

OVERVIEW

The Vingd [1] popup library facilitates a nicer interaction of your users with Vingd. In a standard-sized popup window, it will happily handle user purchases, rewarding, and login (to your site) via Vingd.

Confirming a purchase, redeeming a voucher or logging in to your site via Vingd identity provider can always be accomplished by redirect her to the Vingd frontend. Although this is the most robust way to interact with Vingd in terms of cross-browser/environment compatibility, a more fluent user experience is achieved with Vingd popup.

The popup library can be seen in action on Vingd Newspapers demo site (reference implementation), and Vingd API demo site (full source available on both sites).

1.1 A quick `how-to` guide

The library provides an `onclick` link handler that opens the Vingd frontend in a popup window (for user to confirm purchase or redeem voucher) and notifies you afterwards of the result (user action) via event callbacks you provide.

For library to work, you have to ensure a working `popupURL` location where Vingd frontend shall be redirecting users when they are done. The library provides an out of-the-box handler, `popup.html` (see the *Download* page) and it is critical to place it in a web accessible location on your site (or, at least, on the same domain).

1.1.1 Purchase confirmation

For this example let's suppose you are running a site at `http://example.com/`; you already took care of enrolling objects in Vingd Registry, and you already implemented generating of Vingd Orders (if not, see Vingd API docs).

Let's say you hosted our `popup.html` file at `http://example.com/popup.html` (this is the `popupParams.popupURL`).

Now just add the following in the `<head>` section of your page:

```
<script type="text/javascript" src="http://apps.vingd.com/cdn/vingd-popup/v0.8/build/main.min.js"
<script type="text/javascript">
    var orderOpener = new vingd.popupOpener({
        popupURL: "http://example.com/popup.html",
        siteURL: "http://example.com",
        lang: "en",
        onSuccess: function(hwnd, args) {
            // TODO: verify purchase of 'oid' with 'token' (args.token),
            // probably via AJAX; notify user or update this page
        }
    });
</script>
```

[1] We're in process of re-branding as Vingd. Libraries may still have some internal references to the old name, "knopso", but "on the outside", towards the end-user, we are building the Vingd brand.

popupParams.siteURL can be omitted, in which case will default to parent page URL. Later on in your document, you can provide the purchase link (you should generate an Order before that to get the Order purchase URL):

```
<a href="https://www.vingd.com/orders/2598/add/" onclick="return orderOpener(this)" class="vingd-p
    Purchase X via Vingd!
</a>
```

or, if you are using jQuery:

```
jQuery(function($) { $(".vingd-purchase").click(orderOpener); });
```

By default, orderOpener will inspect href of the caller's source element and redirect the user there (in case of purchase this is the Order URL on Vingd frontend). The practice of setting href to Order URL provides a fail-back in cases of error (or disabled JavaScript); the popup will not open - user will instead be redirected to Vingd frontend and she will still be able to confirm the purchase [2].

Note, however, that you **do not** have to pre-generate all orders in advance. If you need to generate user/context-specific orders **on demand**, you can do that from the onInit() callback. In that case though, the functionality of your site depends on JavaScript, and there is no safe fail-back.

In the example above, what remains for you to implement is the verification of purchase in your backend and displaying the requested content to the user. Put it in (call it from) your onSuccess() callback.

For more advanced scenarios, bear in mind the library enables you to have different openers (orderOpeners) for different objects. That way you can handle individual purchases differently.

1.1.2 Redeeming vouchers

Using Vingd vouchers is somewhat simpler than confirming purchases, in a sense that handling onSuccess callback is not critical for user to receive vingds.

The minimal code for redeeming a voucher in popup is analogous to the one above (assuming we're still on the example.com site):

```
<script type="text/javascript" src="http://apps.vingd.com/cdn/vingd-popup/v0.8/build/main.min.js":
<script type="text/javascript">
    var voucherOpener = new vingd.popupOpener({
        popupURL: "http://example.com/popup.html",
        siteURL: "http://example.com",
        lang: "en"
    });
</script>
```

The callbacks like onSuccess, onCancel, etc. are optional. They can be added to the popupParams (after siteURL, for example), like this:

```
onSuccess: function() {
    alert("Voucher used successfully.");
},
onCancel: function() {
    alert("Voucher not used.");
},
onError: function(hwnd, args) {
    alert("Failed to cash-in the voucher: " + args.msg);
}
```

[2] In the **redirect mode** (contrast with *popup mode*), after a successful purchase on Vingd frontend, user is redirected to object_url, an URL given upon registering the object with Vingd Registry.

1.1.3 Dynamic voucher (order) fetch

Vouchers (as orders) can be generated on demand, when popup launches and calls the `onInit()` callback. You should amend `popupParams` above with:

```
onInit: function(hwnd, args, onDone) {
    $.get('ajax/voucher.php', function(data) {
        onDone(data.voucherURL);
    });
}
```

For further details, please see the documentation of `vingd()` class. The good starting point is `vingd.popupOpener()` function.

CHAPTER

TWO

DOWNLOAD

2.1 The prologue/epilogue page

A single static html file will serve as a popup prologue and epilogue page. Download it (save it from here) and host it in a public web location on the same domain as your site.

Just for a reference, `popup.html` file looks like this:

```
<!doctype html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta name="robots" content="noindex, nofollow" />
    <title>Vingd Popup Library Callback</title>
    <link rel="stylesheet" type="text/css" href="http://apps.vingd.com/cdn/vingd-design/v3.0/buil
    <script type="text/javascript" src="http://apps.vingd.com/cdn/vingd-popup/v0.8/build/popup.mi
</head>
<body>
    <noscript>JavaScript is required for Vingd Popup to function.</noscript>
</body>
</html>
```

2.2 Library code

Include the following on the page where you'll call popup from:

```
<script type="text/javascript" src="http://apps.vingd.com/cdn/vingd-popup/v0.8/build/main.min.js"
```

CHAPTER

THREE

THE VINGD OBJECT

3.1 Core functionality

class vingd()

A global `window.vingd` object is created and initialized upon script (`main.min.js`) load. To create a "popup opener" function object you should use `vingd.popupOpener()` function, with arguments describing details of a future popup open process.

For example, you can create `orderOpener` object once, like this:

```
var orderOpener = new vingd.popupOpener({
    popupURL: "http://example.com/popup.html",
    siteURL: "http://example.com",
    lang: "en",
    onSuccess: function(hwnd, args) {
        window.location = vingd.buildURL("http://example.com/verify.php", {token: args.token}
    }
});
```

And use it later in your document, while presenting your user with the "buy" option:

```
<a href="https://www.vingd.com/link/to/your/order" onclick="return orderOpener(this);"> buy <
```

When user clicks the link, a popup window will open (redirect user to order link on Vingd frontend), your page will darken, and user shall be presented with an option of buying your order. Upon purchase, popup window will automatically close and your `popupParams.onSuccess` handler will be called (in the context of your (parent) page - the page that hosted the link, and the `vingd` object).

`vingd.`**`popupOpener`**(*popupParams*)

The main library method. It constructs and returns a specialized click-on-link handler function that opens popup, interprets the result and calls your event callbacks.

Arguments

- **popupParams** (*object*) – Behaviour of the constructed handler is defined with the *popupParams* function argument.

  `popupParams.`**`popupURL`**
  Location of the `popup.html` file hosted on your server (see *Download*).

  `popupParams.`**`siteURL`**
  [optional]. Your site's URL. The user shall be redirected here when she closes the main window, and we don't know where else to redirect her. The default value is taken from the parent page URL (a page you are calling `vingd.popupOpener()` from).

  `popupParams.`**`lang`**
  [optional] The language of popup messages (supported values: `en`, `hr`). If omitted, the default value is used: `vingd.settings.lang` (i.e. `hr`).

`popupParams.`**`onOpen`**

[optional]. Callback function called **everytime** after a popup window is opened and focused. Default handler will call `vingd.darkenScreen()`. To prevent it, override it with `null`.

`onOpen``()`

`popupParams.`**`onClose`**

[optional]. Callback function called **always** as the last heart beat - after popup window is closed. Default handler will call `vingd.undarkenScreen()`. Override it by providing your function (you can also provide `null`).

`onClose``()`

`popupParams.`**`onInit`**

[optional]. Callback function called when popup first opened. If you don't specify your `onInit` callback, the default handler will redirect user to `href` property of a source link element, or `element.href` that you passed to the `opener()` function.

Note **You can generate your orders from this callback**. Or, more precisely, fetch them (on demand) via AJAX from your backend.

`onInit``(`*hwndPopup*, *args*, *onDone*`)`

Arguments

- **hwndPopup** (*object*) – reference to a popup window handle
- **args** (*object*) – query arguments
- **onDone(url)** (*function*) – callback function you should call at the end of your (asynchronous) initialization to redirect user to `url` (typically order purchase URL you fetched with AJAX from your backend).

`popupParams.`**`onSuccess`**

[optional]. Callback function called if popup operation (purchase confirmation, voucher redeeming, etc.) finished successfully.

`onSuccess``(`*hwndPopup*, *args*`)`

Arguments

- **hwndPopup** (*object*) – reference to a popup window handle
- **args** (*object*) – query arguments (including purchase verification token object: `args.token`)

`popupParams.`**`onError`**

[optional] Callback function called in the case of Vingd frontend error.

`onError``(`*hwndPopup*, *args*`)`

Arguments

- **hwndPopup** (*object*) – reference to a popup window handle
- **args** (*string*) – query args including message string `args.msg`, the error message (as received from Vingd frontend)

`popupParams.`**`onCancel`**

[optional] Callback function called when user clicks the "Cancel" button on Vingd frontend.

Note If user simply closes the popup window, only the `popupParams.onClose()` will be called.

`onCancel``(`*hwndPopup*, *args*`)`

Arguments

- **hwndPopup** (*object*) – reference to a popup window handle
- **args** (*object*) – query arguments

`popupParams.`**`onExpand`**

[optional] Callback function called when user clicks the "Expand this popup" button on Vingd frontend.

Note Default handler opens `args.expand_url`, or if this URL is not not defined (by Vingd frontend), then to the `frontendURL` parameter.

onCancel (*hwndPopup*, *args*)

Arguments

- **hwndPopup** (*object*) – reference to a popup window handle
- **args** (*object*) – query arguments

`popupParams.`**`frontendURL`**

[optional]. The address of Vingd frontend used in a certain error scenarios (zombie popups, malformed extend actions, etc.). The default value is: `vingd.settings.frontendURL`. You never have to override it (except when you're hosting a local Vingd frontend).

`popupParams.`**`width`**

[optional] Width of popup window in pixels (as integer). If omitted, the default value is used: `vingd.settings.width`.

`popupParams.`**`height`**

[optional] Height of popup window in pixels (as integer). If omitted, the default value is used: `vingd.settings.height`.

Returns function

opener ([*element*])

You should use the returned function object to initiate purchase confirmation (or voucher redeeming or Vingd OAuth login) when user clicks your "purchase/redeem/login" link.

The only required propery of the *popupParams* object is the `popupParams.popupURL`. Note however, that your site will seem quite dysfunctional if you also don't handle at least the `onSuccess()` event.

The document at `popupParams.popupURL` location must handle the response from Vingd frontend (response is encoded in `GET` data). You should use the default handler supplied with the library, `popup.html` (see *Download*). The default handler will call your `on...` callbacks, it will handler error scenarios, zombie popup mode (when user closes the main/site/parent window/tab), and several other dirty details.

`vingd.`**`popupCloser`**(*window*)

Handles popup closing process. Depending on the result of popup operation, user-defined callbacks are executed (within a context of popup opener window).

Arguments

- **window** (*object*) – a handle to popup window which contains page loaded as a response from Vingd frontend (`popupParams.popupURL`).

Return values from Vingd frontend are used to call this popup's event callback functions: `onSuccess()`, `onCancel()`, `onError()` and `onExpand()`:

- On successful operation (purchase/voucher redeem/login): `popupParams.onSuccess` function is called. `args.token` can then be used to verify the purchase in backend via AJAX.
- Upon user clicking "Cancel" button while on Vingd frontend, the `popupParams.onCancel` function is executed.
- In case of an error, the `popupParams.onError` handler is called, receiving error description as an argument (`args.msg`).
- When user requests popup expansion, while on Vingd frontend, the `popupParams.onExpand` function is called (in a context of parent window) to be able to *close* the popup window and then open the requested page in parent window. The requested page URL is propagated thru `args.expand_url` GET parameter.

When appropriate handler is finished executing, this function closes the popup, which in turn triggers calling of `popupParams.onClose` handler from within `vingd.closeListener` internal method. (Note: problems have been reported for some Safari versions that do not handle delayed function execution properly.)

3.2 Simple utility functions

`vingd.`**`darkenScreen`**`()`

A utility method, usually used in onOpen handler, that darkens the screen by overlaying it with a semi-transparent black layer. To use, ensure that no screen element has a `z-index` at or above 10000. This layer can be suppressed using the `vingd.undarkenScreen()`.

`vingd.`**`undarkenScreen`**`()`

A utility method, usually used in onClose handler, that undarkens the screen (see `vingd.darkenScreen()`).

`vingd.`**`buildURL`**`(`*base*, *params*`)`

A utility function for building a valid URL given `base` URL and query parameters in a dictionary (JavaScript object).

`vingd.`**`getURLQueryArgs`**`(`*query*`)`

A utility function for parsing URL GET parameters given a `query` string (eg. `window.location.search`).

3.3 Callback handlers storage facility

When a new popup window is opened, its description is stored inside an object pushed into an array. This is done transparently and automatically with the ‘pushWindowData’ function (upon opening the popup).

Popup description includes all event callback function references.

To retrieve a popup data (for example to execute some of the previously defined event handlers), the `vingd.getWindowData()` function can be used.

`vingd.`**`pushWindowData`**`(`*hWnd*, *params*, *callbacks*`)`

Stores event handlers (*callbacks*) and Vingd frontend parameters for a popup referenced by a window handle *hWnd*.

`vingd.`**`getWindowData`**`(`*hWnd*[, *dequeue*]`)`

Retrieves popup data (including callbacks) for a window referenced by *hWnd*. To delete popup data, set *dequeue* to true.

CHAPTER

FOUR

CHANGE LOG

4.1 v0.8.5 (2013-02-06)

- IE bug fix: asynchronous window.open() - popup JavaScript starts before parent JavaScript block finishes. Now popup loaded routine is queued in parent's event queue (using `setTimeout()`).

4.2 v0.8.4 (2013-02-04)

- Templated messages refactored, generalized and simplified.
- Better handling of error states in prologue phase (error messages with call to action parent window, or <parent>.vingd object unavailable).

4.3 v0.8.3 (2013-01-31)

- A complete switch to the Vingd brand. The popup interface object but `knopso()` renamed to `vingd()`, but `knopso()` reference remained for backward compatibility.

4.4 v0.8.2 (2012-05-31)

- A complete switch to the new Vingd design (design branch v3.0). The popup interaction code is fully compatible with previous versions in v0.8 branch, but `popup.html` must be *Download*-ed and updated!

4.5 v0.8.1 (2012-05-12)

- Internationalization (*popupParams.lang*), support for English (`lang: en`) and Croatian (`lang: hr`, the default).

4.6 v0.8 (2012-02-27)

- Knopso re-branded as Vingd.

4.7 v0.7.2 (2012-02-16)

- Prologue page redesign (loading animation, instead of the title).

4.8 v0.7.1 (2012-02-01)

- Better support for login mode (zombie messages added).
- When in zombie mode, user auto-redirected to the recommended url if `display=page`; otherwise button-redirect offered.
- `siteURL` now has a default value: parent page URL.

4.9 v0.7 (2012-01-27)

- Support for vouchers and OAuth login via Knopso.
- Prologue and epilogue page merged to a single unified html stub which is now capable of handling purchase/voucher/login/etc. scenarios in a normal and zombie mode (see popup.html). This static html file is the only thing you'll have to host on your servers.
- Library now hosted on our servers (minimized, uncompressed). This will enable us to continuosly upgrade it and fix bugs, while not requiring from you to periodically pull the newest version.
- Much better error handling in `zombie` mode (ie. when user closes the site/parent page, but continues to interact with the popup). User is informed and the best available action is offered.
- All user messages localized (`lang` parameter).
- HTML templates for prologue/epilogue. DOM utility functions.
- Added `onInit` callback which is run on prologue opened. You can generate your orders here (fetch them via AJAX). The default implementation will only redirect user to `href` argument (this can be the `href` attribute of a source A/link element, or a query argument in the `popupURL`).
- Added `onResult` callback which is called from epilogue code after the specific result callback (`onSuccess`, `onCancel`, `onError`, `onExpand`).
- Default `onExpand` callback now more generic; it opens `expand_url` the epilogue receives.
- Added parameters: `cdnURL`, `siteURL` to better handle `zombie` scenarios.
- Automated build (main.js and loader/popup.js).
- Bug fixes (Safari undarken on close).

4.10 v0.6.2 (2011-12-07)

- Epilogue extensions (covering behaviour in cases parent is closed before popup).
- Shifting the increasing number of parameters in prologue/epilogue query (GET arguments) improves the robustness and decreases dependency on JavaScript cross-window communication (and different policies across browsers).
- Added `frontendURL` parameter for local testing.
- Knopso frontend URL change (to 'www.knopso.com' from 'my.knopso.com').
- Bug fixes (IE opacity, scrollbars visible).

4.11 v0.5 (2011-10-27) since v0.1 (2010-09-09)

- Support for Knopso purchase in popup.
- The `onSuccess` callback verifies token (received in JSON).

- Prologue page hosted on the site to prevent blocking of popup opening a site on different domain.
- Epilogue page hosted on the site (the same domain) to enable communication with parent-defined callbacks.
- `onExpand` callback for expanding Knopso frontend in a new browser window.
- Screen darkener enhanced.
- Utility functions: URL parse/unparse, browser window size/position.
- Popup parameters storage handles multiple popup windows per page.
- Documentation separated from code.

CHAPTER

FIVE

LICENCE

Vingd client popup library is licensed under the MIT License:

Parts of Vingd Popup Library (dealing with querying of browser window dimensions in a cross-browser compatible manner) are borrowed from Google's step2 PopupManager, which itself is covered by the Apache License below:

CHAPTER

SIX

INDICES AND TABLES

- *genindex*
- *search*