Youneeq Setup Guide

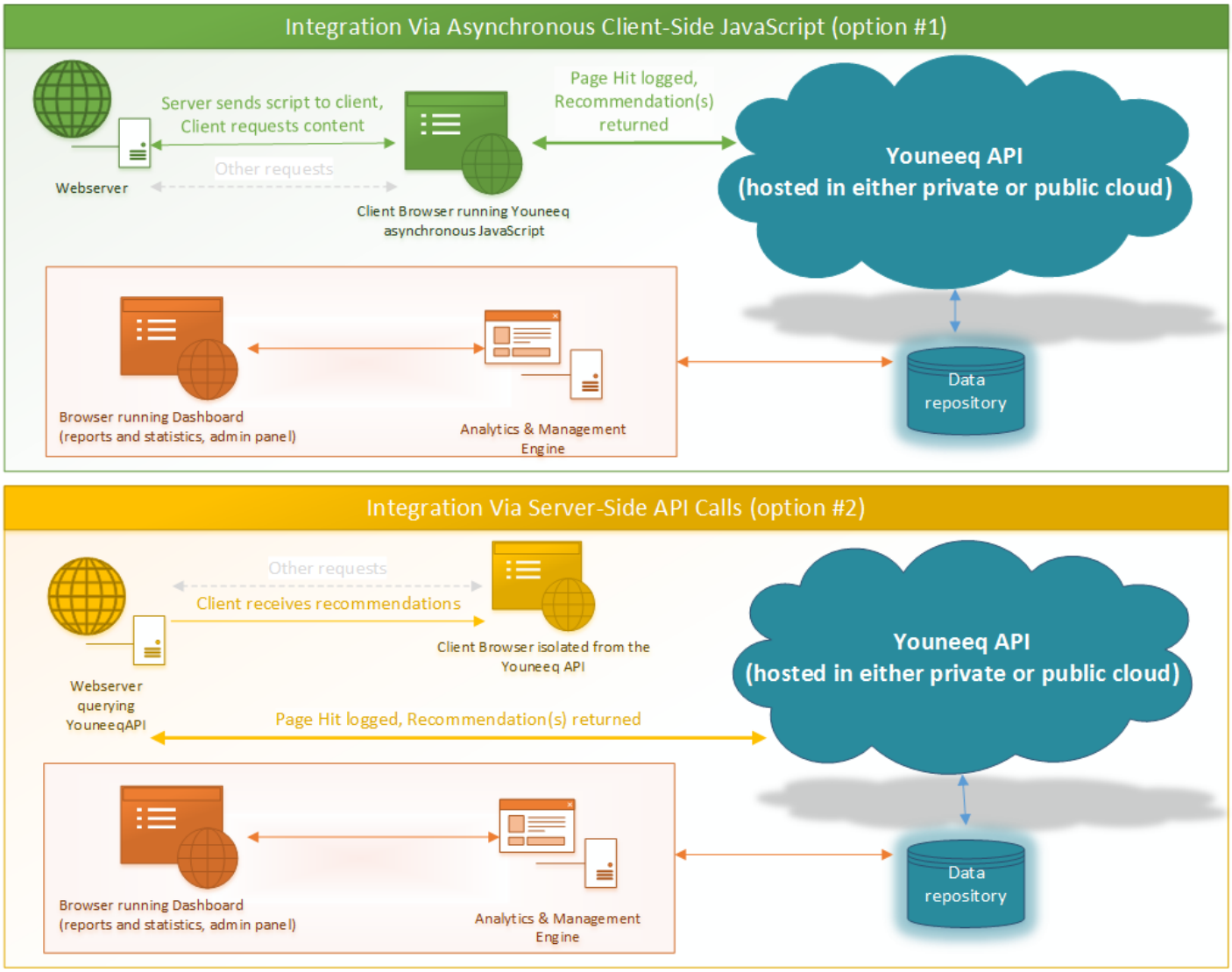
Last revised September 22, 2020

Overview

Description

Youneeq is a recommendation and content personalization system. Exposed as a web service, it can be integrated into any website that employs a data driven structure for managing content, or recommend content from external sources. Behaviors can be tracked from as specific as the individual level to as broad as site wide, and provides recommendations targeted to an individual and filtered as needed. Youneeq utilizes a RESTful API, and by default JSON (JavaScript Object Notation) is used to send and receive data, allowing requests to be passed to and from JavaScript without converting the data between formats (for more detail on REST and JSON see: http://en.wikipedia.org/wiki/Representational_state_transfer and http://en.wikipedia.org/wiki/JSON). If needed, we can easily extent our API without breaking existing functionality for customers. If our syntax, the use of JSON, and/or a RESTful approach to an API don't work for a client then a custom endpoint can be set up to support custom formatters (e.g. XML/SOAP, integration with a legacy service, etc…).

Youneeq Integration Diagram – a typical example of the request workflow



Overlays

An overlay solution available, contact support for more details. This is a popular feature with some early on in the process, as value demonstrated option is given to upgrade into full integration with a more complete suite of features.

Setup

Our integration specialists are always available to help get your site up-and-running. We typically handle building the integration package for you, so as to minimize the time and resources needed by your IT/IS team.

Plugins and pre-built integrations

We currently have plugins for the following CMS's:

- **WordPress**, **Drupal**, **DNN**

We have experience integrating with many custom CMS solutions, including:

- **Libercus, Adobe AEM, Polopoly**, **TN Blox**, **Clickability**

Integrating the recommendations manually

This approach can be used for integrating into a 3rd party tag, into an existing site template, or CMS integration

1. Script references

These need to load before the rest of the scripts. Often reference scripts will appear towards the top of a page but for ease of deployment there is no issue in pasting these just above the remainder of the script tags.

Client sites load ***yq.js***, ***jquery.js***, ***json2.js***, and ***detect_timezone.js***. Unless the client web page loads them somewhere else, the page should include the JavaScript like so:

```
<script type="text/javascript" src="YOUNEEQ_API_HOST/scripts/jquery.js"></script>
<script type="text/javascript" src="YOUNEEQ_API_HOST/scripts/json2.js"></script>
<script type="text/javascript" src="YOUNEEQ_API_HOST/scripts/detect_timezone.js"></script>
<script type="text/javascript" src="YOUNEEQ_API_HOST/app/yqmin"></script>
```

2. yq.js calls Youneeq API to record page hit and get recommendations

Generic functions can be used on most any site with only a few options that will need to be configured before sending to the client, as in the example below:

Add a div where you would like recommendations to appear

```
// the named div is referenced in the on_yq_suggest function
<div id="youneeq "></div>
```

Insert a script tag with a jQuery(document).ready function

```
<script type="text/javascript">
    jQuery(document).ready(function ($) {
        // 3. called back by Youneeq API with suggestions
        on_yq_suggest function goes here
        // 2. Called when Yq is initialized
        on_yq_init function goes here
        // 1. Set up callback when Yq is initialized
        Yq.onready(my_yq_init);
    })</script>
```

Continued on next page…

Add the on_yq_suggest function

```
// 3. called back by Youneeq API with suggestions
function on_yq_suggest(suggestions) {
    //check if there are suggested results, if so populate 'youneeq' div
    if (suggestions && suggestions.suggest && suggestions.suggest.node) {
        var nodeids = suggestions.suggest.node;
        var stories = "";
        var recommend_header = '<div>News For You...</div>';
        for (var i = 0; i < nodeids.length; i++) {
            var articleTitle = nodeids[i].title;
            var articleLink = nodeids[i].url;
            stories += "<li><a href='" + articleLink + "'>" + articleTitle + "</a></li>";
        }
        stories = recommend_header + "<div style='column-count: 2;'><ul>" +
                  stories + "</ul></div>";
        //populate an element(div) of the id youneeq with recomendations
        $("#youneeq").append(stories);
    }
}
```

Finally, add the my_yq_init function

```
// 2. Called when Yq is initialized
function my_yq_init() {
    var content_id = insert content id here;
    var categories = insert categories id here;
    var title = insert title here (optional);
    var image_url = insert image url here (optional);
    var description = insert description here (optional);
    //content or not content page logic using the opengraph type element
    if ($('meta[property="og:type"]').attr('content') === "article") {
        Yq.observe({
        'observe': [{
                'type': 'node',
                'name': content_id,
                'title': title, //required for 'is_panel_builder'
                'description': description,
                'categories': categories,
                'image': image_url
            } ],
            //fetches 10 recommendations
            'suggest': [{ 'type': 'node', 'count': 10, 'is_panel_builder': 'true',
                'isAllClientDomains':'true' }]
        },
        on_yq_suggest);
    }
    else { //if identifier not found (i.e. non-content page) do this...
        Yq.observe({
            //fetches 10 recommendations
            'suggest': [{ 'type': 'node', 'count': 10, 'is_panel_builder': 'true',
                'isAllClientDomains':'true' }]},
        on_yq_suggest);
    }
}
```

BASIC API Syntax

The observe, suggest and page_hit can be executed in a single request, independent requests, or a combination of any two together. This for instance allows page_hit to record analytics/reporting data on every page while only recording content details and providing recommendations on pages that contain articles/stories, or only providing recommendations on home pages while recording content details on article/story pages

Observe Request (* = required field to submit the request sub-object)

Field	Description	Type
observe	***record the details of the content being viewed***	
➢ **type***	content type being viewed	string
➢ **name***	name used to identify the content being viewed	string
➢ **categories***	categories associated with the content being viewed	string
➢ **title**	content title	
➢ **description**	content description	
➢ **image**	link to image appropriate for presentation in recommended content	
➢ **create_date**	creation date of content (e.g. publish date)	date string
➢ **expiry_date**	expiration date of content – only needed if overriding defaults	date string
suggest	***specifies format and filters of requested content recommendations***	
➢ **type***	content type	string
➢ **count***	number of results to return	integer
➢ **categories**	categories to include (leave blank if not filtering)	string
➢ **domains**	domains to search for recommendations(omit to search current domain)	string
➢ **date_start**	max content age to return from when first tracked or create_date value	date string
➢ **date_end**	most recent content to return if newest content is not wanted	date string
➢ **isUrlReturned**	boolean to identify whether to return content as a url	string/boolean
➢ **isAllClientDomains**	boolean that indicates if all domains for the client should be searched	string/boolean
➢ **is_panel_builder**	boolean that specifies outputting contentId, title, and url in place of choice	string/boolean
➢ **title**	content title	string
➢ **description**	content description	string
➢ **panel_custom**	custom choice for return info (title, url, image, description, date, domain, categories, like, read, domain_name)	string
➢**options**	options to specify for enabling additional features e.g. options:{…, …}	
strict_categories	*enforce category restriction even when insufficient content*	Array of string options
paging_enabled	*enable paging features to prevent in-page recommendation list duplicates*	
show_history	*do not filter out user history in recommendations*	
disable_history	*enables more stringent user history filters than the default setting*	
page_hit	***used for custom analytics/reporting, not required for recommendations***	
➢ **href***	url of requested/current page	string
➢ **referrer***	referring page	string
➢ **tz_off***	time zone offset	string
➢ **tz_name***	time zone name	string
bof_profile	user/profile identifier	string
alt_profile	over-rides bof_profile (prevents yq.js session Id from being used)	string
href	url of requested/current page	string

Observe Response

Field	Description	Type
suggest	*the parent container for the recommendation list*	-
➢ ***type***	the name of the content type(s) returned	string
o ***choice***	the recommendation(s) returned for the content type	string
Using 'is_panel_builder' option:		
suggest	*the parent container for the recommendation list*	-
➢ ***type***	the name of the content type(s) returned	string
o ***id***	content identifier that was submitted in the 'name' field	string
o ***title***	content title	string
o ***url***	url to content	string

Additional API Syntax (examples provided in JavaScript)

Events

Both stateful/updatable events (e.g. like button click/unclick) and incremental events (e.g. tallying the positions on the page users are clicking) are supported. *Note - In most instances custom reporting will need to be set up in order to use this feature*

This example shows a JS function for supporting click tracking for a 'like' button using jQuery:

```
function like_click(content_id, domain_name) {
        var api_url = "http://api.youneeq.ca/api/eventaction";
        var like_data = {
                "is_state_change": "true",
                "event_name": "like",
                // using a Youneeq-generated user_id, for production a fallback for local_storage is recommended
                "user_id": localStorage.getItem("yq_session"),
                "domain": domain_name, // get the domain name
                "content_id": content_id,
                "value": true
        };
        var json_data = {'json': JSON.stringify(like_data)};
        var ajax_data = {: api_url,crossDomain:true, dataType:'jsonp',:json_data};
        $.ajax(ajax_data);
}
```

This following example shows a JS function that accepts stringified page positional data and records where a navigation event occurred on the page using jQuery:

```
function event_position_click(content_id, position_data) {
        var api_url = "http://api.youneeq.ca/api/eventaction";
        var event_data = {
                "is_state_change" : "false",
                "event_name" : "page_position",
                "user_id" : localStorage.getItem("yq_session"), // in this case the user_id is fetched from local_storage
                "domain" : document.location.hostname, // for local testing this can be replaced with a hard-coded domain
                "content_id" : content_id,
                "value" : position_data
        };
        var json_data = {'json' : JSON.stringify(event_data)};
```



```
        var ajax_data = {
                url : api_url,
                crossDomain : true,
                dataType : 'jsonp',
                data : json_data
        };
        $.ajax(ajax_data);
}
```

Category Filters Managed by Youneeq

Category filter settings can be stored and retrieved. This is generally implemented against each user's settings as an additional profile option.

The category tracking supports the following options:

add	**adds the list of categories included in the request to the users existing categories**
update	replaces the category filter settings with the categories included in the request
remove	removes the categories (if they exist) from the users existing categories
remove_all	removes all category filters, effectively enabling all categories unless overridden by site settings
send	just send the users current list of categories without making changes

A category filter request will always return the category filter settings after updates have been applied. For more information on using this feature please contact one of our integration specialists.

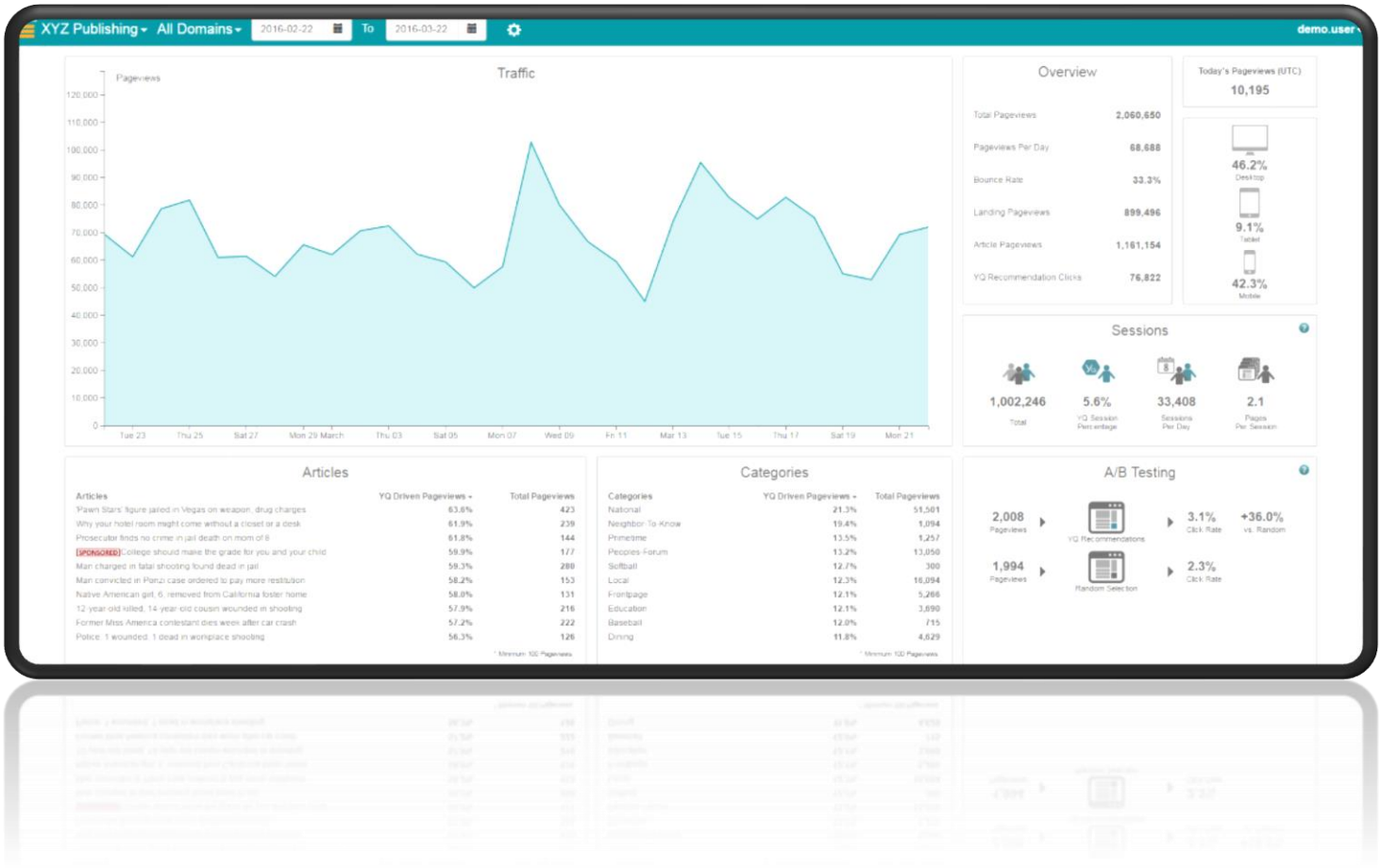
Special Content Types

In addition to article recommendations the service contains several extensions to handle storing and serving specialized content; this includes: '**local sponsored content**', '**classifieds'**, '**identity management metadata**'. If any of the aforementioned content types are of interest, or if another type of content delivery is desired then please contact one of our sales associates or integration specialists, as we will need to assess each request on a case-by-case basis to determine implementation feasibility.

Analytics

Youneeq Dashboard

All customers are given access to the Youneeq Dashboard. The dashboard is meant to address features our clients find absent or incomplete in other analytics packages, and as such should be used in conjunction with other robust analytics solutions.



Features Include

- A/B Testing
- Standard Traffic Statistics
- Article Traffic Statistics for All Stories
    - Breakdowns by Source (Direct, Social, Search, and more)
    - Top referral breakdown for every story/article
    - Article search
    - Find out more about your top referring sites
- Category Traffic
- Realtime Monitoring

If there's a feature you would like to see us implement, then please let us know.

Additional Analytics Support

Most web analytics packages work without issue against our service, and we routinely review the documentation and best practices for integration with leading solutions, such as Google Analytics, to make certain our service can be tracked with as little hassle as possible. If you have questions about a specific Analytics Package then let us know and a member of our staff will be glad to review your needs.