

---

# LiftIgniter Integration Guide: Javascript

LiftIgniter • 2017

---

# The 5 Second Version

**WHO:** Our beacon tracks who a person is, and tells us what they're looking at

**WHAT:** Our JS SDK is a pre-defined set of instructions for what is going to happen on the page

**HOW:** A Mustache.JS template defines how the recommendations are displayed

**WHERE:** A particular `<div>` on the page where the recommendations will be displayed

---

# Overview

## Track Everything

- [Inventory](#)
- [Users](#)
- [Activity](#)
- [Single Page Applications\\*](#)

## Display Recommendations

- [Widgets](#)
- [A/B Testing](#)

## Post-Implementation

- Reporting
- Troubleshooting

## FAQ

---

**Track Everything**

**Inventory**

---

---

# Inventory

## What is Inventory?

- Any item that will be displayed as a recommendation
- Articles, videos, song tracks, ads, content snippets...

## How to track:

1. Add our JS beacon to all site pages
2. When a user visits a page, the beacon will automatically scrape the page for metadata
3. If successful, you will see `_inventory.gif` in the Network log

---

# Metadata

## What metadata is scraped?

- Custom LiftIgniter JSON

```
<script id="liftigniter-metadata"
type="application/json">
{ "tags" : ["apple", "nutrition"]}
</script>
```

- OpenGraph tags

```
<meta property="og:title" content="sample-title"/>
```

## Options

- [Hide pages](#)
  - [Parse data from the DOM](#)
  - [Include timestamp fields](#)
  - [Rules and exceptions](#)
  - Use an ID to identify unique pages instead of a URL ([contact Support](#))
-

---

# Considerations

## Inventory Sync

The Javascript must fire on a page at least once for it to be added to the Inventory. As a result, items may not be added as soon as they are published.

## Item Expiration

By default, an item added via the JS beacon will expire 30 days from when it was [last seen](#).

---

---

**Track Everything**

**Users**

---



---

# Users

## What are Users?

- Any unique visitor to the site

## How to track:

1. Add our JS beacon to all site pages
  2. When a user visits, the beacon will automatically set a [first-party cookie](#) to track activity
  3. As the user moves through the site, their activity allows us to build up a profile of their tastes and preferences
-

---

# Considerations

## Set Custom UserID

You can [set the userID](#) so that we associate their behavior to a known user across multiple devices

## Demographic Data

**LiftIgniter does NOT require PII or demographic data of any kind.**

Since the strongest signals for our algorithms are behavior based, other user data is generally unnecessary. If you have explicit user data that you think might help inform our recommendations, you can include it via the [/user API endpoint](#) or as [context information](#) in an Activity.

---

---

# Track Everything Activity

---

---

# Activity

## What Events are tracked?

- [Pageview + Heartbeat](#)
- [Widget Events](#)
  - widget\_shown
  - widget\_visible
  - widget\_click

For complete results and CTR comparison, it is critical that all widget events are tracked.

## How to track:

1. Pageview and Heartbeat events are tracked automatically by the beacon
  2. [Widget events are tracked](#) by calling the `$p("track")` function in the callback of `$p("register")`
  3. If successful, you will see `_activity.gif` in the Network log for each pageview, heartbeat, and [each widget registered](#)
-

---

# Considerations

## Other Events

We support several [standard event types](#), including conversion and engagement. These can be triggered at any time using `$p("send")`

## Multiple Events

You can send multiple events in a single call using `$p("sendRobust")`. This queues the event in a buffer, and sends it on the user's new pageview.

---

---

**Track Everything**

**Single Page Applications**

---

---

# Single Page Applications

## Tracking Pageviews

If you have a website that consists of a single page, and content is loaded using the pushState API of HTML5 or a simple hash routing, we provide [a setting to track the transition](#).

---

---

# Complex SPAs

## Not using PushState or hash routing?

These types of single page sites will want to leverage [\\$p\("setContext"\)](#) to tell LI what the user is currently viewing, and [\\$p\("resetPageview"\)](#) to tells us when a user transitions to a new “page”.

---



---

**Display Recommendations**

**Widgets**

---

# Widget Setup

1. Create Widget div
  2. Create Mustache.js Template
  3. Execute widget rendering and tracking functions
-

---

# Create Widget Div

```
<div id="li-recommendation-widget"></div>
```

## Notes

- The div ID and the widget name do not have to be identical, but it can help future troubleshooting if they are consistent.
  - Each widget on the page should have a separate ID
-

---

**Mustache.js** allows you to create “logic-less” templates, replacing sections of HTML with `{{tags}}` to be filled in later

[Tutorial: HTML Templates with Mustache.js](#)

---

# Create Mustache Template

```
<script type="application/mustache"
  id="li-recommendation-template">
  {{#items}}
  <div class='recommended_item'>
    <a href="{{url}}">
      
      <span class="title">{{title}}</span>
    </a>
  </div>
  {{/items}}
</script>
```

## Notes

- The parameter names in the template (e.g. “thumbnail”, “title”) must match the parameter name in your inventory metadata
  - You can request and populate [any field](#) present in your metadata, including [arrays](#)
-

---

# Execute render and track functions

```
// Callback renders and injects results into the widget div according to the Mustache template.
$.register({
  max: 5, // Number of items you want to show
  widget: 'default-widget',
  callback: function(resp) {
    // Query selector should match widget div ID
    var el = document.querySelector('# li-recommendation-widget ');
    // Template should match Mustache Template ID
    var template = document.querySelector('#li-recommendation-template ').innerHTML;
    // Render the inner HTML of the widget div using the Mustache template and the items in the
    response;
    el.innerHTML = $.render('render', template, resp);

    $.track({
      // Widget div name and recommendation template name
      elements: document.querySelectorAll('#li-recommendation-widget > div.recommended_item '),
      name: 'default-widget ',
      // Match widget name
      source: 'LI',
      // Source "LI" indicates recommendations are provided by LiftIgniter
    });
  }
});

// Executes the registered call.
$.fetch();
```

---

# Order of Execution

The tracking beacon calls our full [JS SDK](#) asynchronously

“[register](#)” tells us which widgets are on the page, how many items to return for each one, and other options

Call fetch **only once** to get items for [all registered widgets](#)



This initializes the SDK with your JS key, along with any [custom configuration settings](#)

Executed in callback of `$p("register")`.  
“[render](#)” fills out the widget div template with the items returned. “[track](#)” attaches an event listener and tags on the URLs in the widget.

# A/B Testing

1. Splitting Users
2. Implementing Test
3. Verifying Results

---

---

# Splitting Users

## Use LI Hash:

1. Call `$p("userHash")` to create a hash of the user cookie
2. Use `$p("abTestSlice")` to take the hash modulo 100. This creates 100 bins (0-99) and assigns the user to one of those bins.
3. You can then set which bins should be shown the LI slice of the test, and which should be shown the base

## Use Your Own:

- If you are using some other method for A|B testing, just [let us know!](#)
- Some tools, like Optimizely, require our JS beacon to be called from within their own scope, causing conflicts with the globally applied tracking beacon. This can be resolved by wrapping the beacon script with:

```
if (typeof $signiter_var === 'undefined') {  
<tracking beacon here> }
```



---

# Implementing Test

## Overwrite Existing

- Copy the HTML of the existing recommendation area to create your mustache template to ensure the LI recommendations look the same as the existing

## Add New Widget

- Construct your template according to the design you want
- Check out some of our [tips on design, number and placement](#) of recommendations and how they affect CTR

Detailed [instructions and code examples](#) are available in our documentation!

---

---

**Post-Implementation**

**Reporting & Troubleshooting**

---

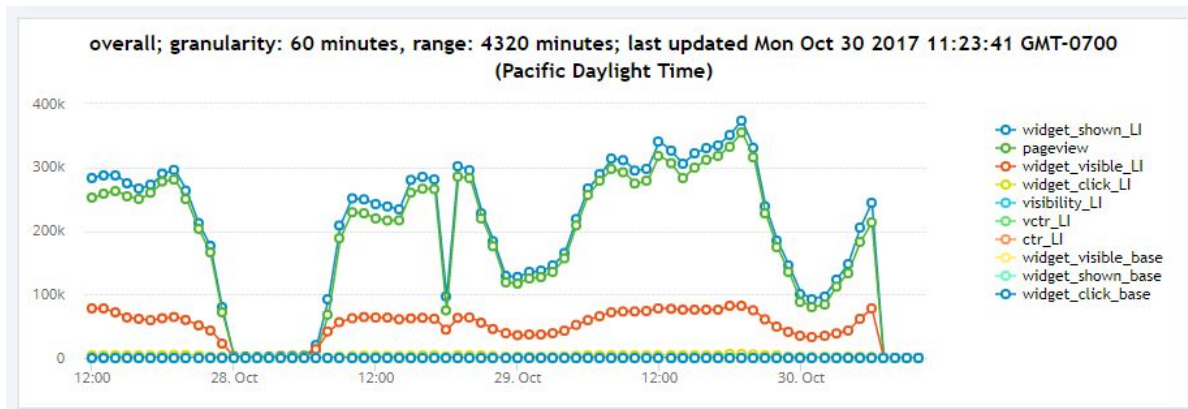
# Reporting - Daily Analytics



## LiftIgniter Lab

- If an A/B test is running, we'll automatically calculate the lift % over the baseline.
- Each day, we calculate daily and session-level data for a wide range of [metrics](#) to display in the [Analytics](#) page

# Reporting - Realtime



## Happening Now

- Raw counts are available in the [Realtime](#) page so you can see exactly what events have been reported in the last few minutes.
- You can also see a list of which items are appearing or being clicked most frequently in the “Top Clicked” and “Top Viewed” charts

---

# Troubleshooting - Inventory

## Health Check

### Configure Inventory

Setup URLs and OG tags to get the right stuff to be scraped.

If you include a field called "noShow" with value "true" in the LiftIgniter JSON object, then the item will be included in our inventory but will never be returned in recommendations.

Verifying...

URL (last 100 scraped)

Primary key

url

Metadata (OG tags)

Updated at

Scraped keys

List of keys scraped and amount of URLs found in.

Key	URLs
-----	------

- Check the last 100 URLs scraped from your site
- See the number of unique keys (metadata fields) across your entire inventory, and the percentage of URLs that have those keys
- See any recent errors we encountered when scraping your pages, such as JSON malformation

[Get Started > Inventory](#)

---

---

# Troubleshooting - Tracking

Render Track A/B test

[How to Render Widgets](#)  
Learn how to put in the code to render recommendations on your website.

[How to Track Widgets](#)  
Use JS code to track recommendations that are rendered on your website.

---

1,752,018	0	1,300,108	539,436	42,707
pageviews <sup>i</sup>	LI response count <sup>i</sup>	LI shown count <sup>i</sup>	LI visible count <sup>i</sup>	LI clicks <sup>i</sup>

[Show Widget Details](#)

There might be a few minutes latency after integrating correctly for stats to start growing.

(Stats in the last 24 hours. Last updated: Oct 30, 2017, 11:42:10 AM)

## Verify Events

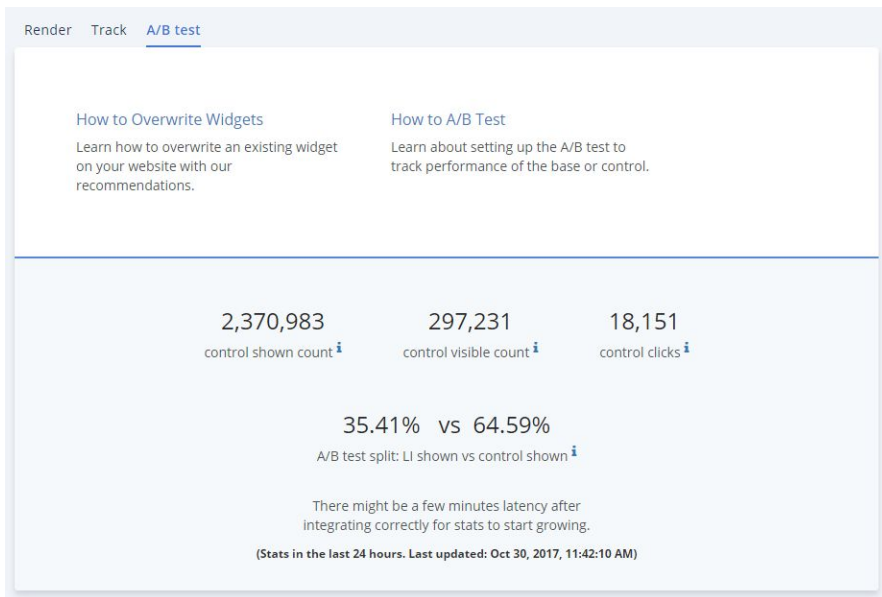
- Check counts for last 24 hours
- Counts update each time browser page refreshes

[Get Started > Render](#)

---

---

# Troubleshooting - A/B Testing



## Check A/B Splits

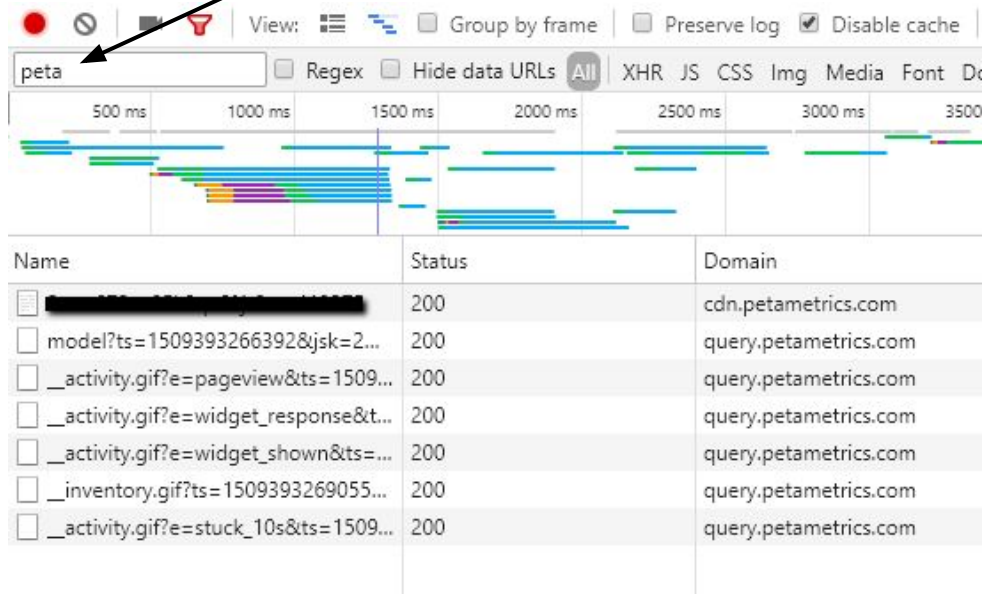
- Calculates the actual difference in events for base and LiftIgniter slices
- Verify against your test handler's settings to identify any discrepancies.

[Get Started > Render](#)

---

# Troubleshooting - Dev Tools

Use "petametrics"  
to filter logs



The screenshot shows the Chrome DevTools Network tab with a search filter set to "petametrics". The top toolbar includes icons for a red stop button, a refresh button, a filter icon, and a funnel icon. The search bar contains "petametrics" and is followed by checkboxes for "Regex", "Hide data URLs", and "All". Below the search bar are tabs for "XHR", "JS", "CSS", "Img", "Media", "Font", and "Doc". The main area displays a network waterfall chart with a time scale from 0 to 3500 ms. Below the chart is a table of network requests.

Name	Status	Domain
<input type="checkbox"/> [REDACTED]	200	cdn.petametrics.com
<input type="checkbox"/> model?ts=1509393266392&jsk=2...	200	query.petametrics.com
<input type="checkbox"/> _activity.gif?e=pageview&ts=1509...	200	query.petametrics.com
<input type="checkbox"/> _activity.gif?e=widget_response&t...	200	query.petametrics.com
<input type="checkbox"/> _activity.gif?e=widget_shown&ts=...	200	query.petametrics.com
<input type="checkbox"/> _inventory.gif?ts=1509393269055...	200	query.petametrics.com
<input type="checkbox"/> _activity.gif?e=stuck_10s&ts=1509...	200	query.petametrics.com

## Network

The following events should fire:

- Javascript snippet loads our SDK asynchronously
- Pageview
- Model = request for recommendations
- widget\_response = event tracking that recommendations were returned to the client
- Widget\_shown = recs loaded on page
- Widget\_click = recs scrolled into viewport
- Stuck\_10s = user was on page for at least 10 seconds
- Inventory.gif = the item metadata we collect for the current URL



---

# Troubleshooting - Console

```
> $p("runDiagnostics")
Diagnostics for Javascript key ██████████ run at Mon Oct 30 2017 13:27:47 GMT-0700 (Pacific Daylight Time) (20026
load). Get full details of requests to LiftIgniter by going to the Network Panel and filtering to petametrics. Messages
marked [WARNING] deserve serious attention, and messages marked [INFO] deserve consideration. Messages marked [DEBUG] a
you have a specific reason to look at them. Keep in mind that if you have selected 'Disable Cache' in your Network Pane
visitors; if you have not, then load times are comparable to those of repeat visitors

No errors were thrown on this page!

[INFO] You have rendered recommendation widgets on the page but no visibility events have fired for any widget. Scroll
rerun runDiagnostics to check that the visibility event fires (which will make this INFO message go away)

[DEBUG] fetchData ▶ {finalized: true, startTime: 1509393266391, timeout: 10000, fetchQueue: Array(0), dedupQueue: Array
[DEBUG] Startup times in milliseconds from start of page load ▶ {initStartTime: 1766, initEndTime: 1770}
[DEBUG] Event counts sent on this page, grouped by event type ▶ {pageview: 1, widget_response: 1, widget_shown: 1, stuc
[DEBUG] Time in milliseconds since page load on this page, for the first event of each type
▶ {pageview: 1770, widget_response: 2183, widget_shown: 2385, stuck_10s: 11770, stuck_3m: 192179}
[DEBUG] Number of widget_shown events by source: ▶ {LI: 1}
[DEBUG] Response widget map (number of widget_response events by widget name): ▶ {default-widget: 1}
[DEBUG] Shown widget map (number of widget_shown events by widget name): ▶ {default-widget: 1}
[DEBUG] Visible widget map (number of widget_visible events by widget name): ▶ {}
[DEBUG] Inventory data: ▶ {scrapingFinished: true, inventorySent: true, timing: {...}}
[DEBUG] Performance data from browser:
▶ Performance {onresourcetimingbufferfull: null, timing: PerformanceTiming, navigation: PerformanceNavigation, memory: !
0 errors, 0 warnings, 1 info messages printed
If including this data in email to support@liftigniter.com, please include everything that was printed in the console
```

## \$p("runDiagnostics")

The runDiagnostics function can be run in the console to print out diagnostics for the JavaScript integration. The output includes messages marked:

- [ERROR] - action required, may prevent recommendations from displaying or being tracked
- [WARNING] - attention required, may seriously hamper recommendation quality
- [INFO] - deserves consideration, performance could be improved.
- [DEBUG] - safe to ignore, usually used by LI support for deeper integration quality checks

---

# FAQ

---

---

# What's the difference between `widget_shown` and `widget_visible`? Why use both?

## `widget_shown`

- This is sent [when the widget is loaded anywhere on the page](#). However, it may not be seen by the user, especially if they are on mobile or the design means it is below the fold (e.g. at the bottom of an article)

## `widget_visible`

- This is sent [when the widget is scrolled into the user's viewport](#). As a result, we know the user probably saw the recommendations

It's important to send both so that we can calculate the *visibility* of an item (how often a widget is seen by the user out of all the times it was loaded - are they scrolling all the way to the bottom of the article and seeing the widget, or no?).

Knowing the visibility lets us get a more accurate reading of the CTR by calculating the [visible CTR](#) - the rate at which users click when they do see the recommendations in the widget.

---

---

# How do I prevent some items from being shown as recommendations?

## Lots of options!

- By default items expire from the inventory 30 days after it was last seen by our beacon (using the [API](#) allows you to set your own TTL value)
  - [NoShow](#): Add to inventory, but never show it (Recommended)
  - [NoIndex](#): Don't add to inventory at all
  - [Max Age](#): Don't show items older than X time
  - [Exclude a specific list of items for a particular query](#)
  - Boost new items, then gradually [decay over time](#)
-

---

# How do I boost specific items?

## To boost or not to boost...

In general, the algorithm will give preference to the right content at the right time all on its own. It's what we do best! However, we know that sometimes certain content needs a boost in order to meet impression quotas or other obligations.

- [Boost new items](#), then decay over time
  - Default [flat boost](#) - multiple items tagged with this will all have the same amount of boost applied
  - Custom rules - if you have a particular use case, don't hesitate to [reach out to Support!](#) We'll be happy to help implement rules specifically for your content to give automatic preference to sponsored stories, specific categories or tags, or other indicators. Just let us know how we can help!
-

---

**Check out our**  
**Documentation!**

---