Google Tag Manager integration guide (incl. dataLayer support) for Psychic Bazaar (www.psychicbazaar.com)

Specification document

Table of Contents

1 Introduction

This document specifies the requirements for integrating Google Tag Manager with www.psychicbazaar.com.

The main requirement, and the one that the majority of this document articulates, is the requirement that any data required for reporting via a web analytics tool (e.g. SnowPlow Analytics or Google Analytics) is passed into the dataLayer where it is accessible by Google Tag Manager to pass on to analytics programmes.

Identifying all the data elements that might be required in the future for reporting is not trivial. However, we have a straightforward methodology for doing exactly this:

1. Identify all the 'static' entities that make up the www.psychicbazaar.com web pages. This will include entities like products, listings, blog posts and accounts. These are core elements that make up the different web pages on the site. We can be exhaustive by combing through the website carefully.

2. Identify all the relevant data points for each entity e.g. each product has a name, SKU and price. A listing contains one or more products. A web page can contain one or more listings, as well as one or more blog posts. This enables us to develop a data model for each entity.

3. Identify all the relevant actions that a user can take on each page, and the 'dynamic' entities that these refer to. Actions include e.g *add to basket*, *zoom in to product*, *share blog post, place an order*. The dynamic entities they refer to include user accounts*,* baskets, orders.

4. Identify all the relevant data points for each action e.g. an *add-to-basket* must include a product. This enables us to develop a data model for each action (an **event model**).

5. Formalize the dataLayer requirements based on the data models developed in (2) and (4) above.

The structure of this document follows the structure of the methodology outlined above. In section 2, we document the static entities that make up www.psychicbazaar.com web pages, and specify a data model for those entities. In section 3, we document the different actions that a user can take on their customer journeys, and a data model for all those different actions / events.

In both cases, we are clear to specify the different fields and provide examples of Javascript snippets required to implement the specification.

2 Identifying and tracking the static entities that make up the elements on web pages

2.1 The structure of the Psychic Bazaar website

Psychic Bazaar is an online retail site:
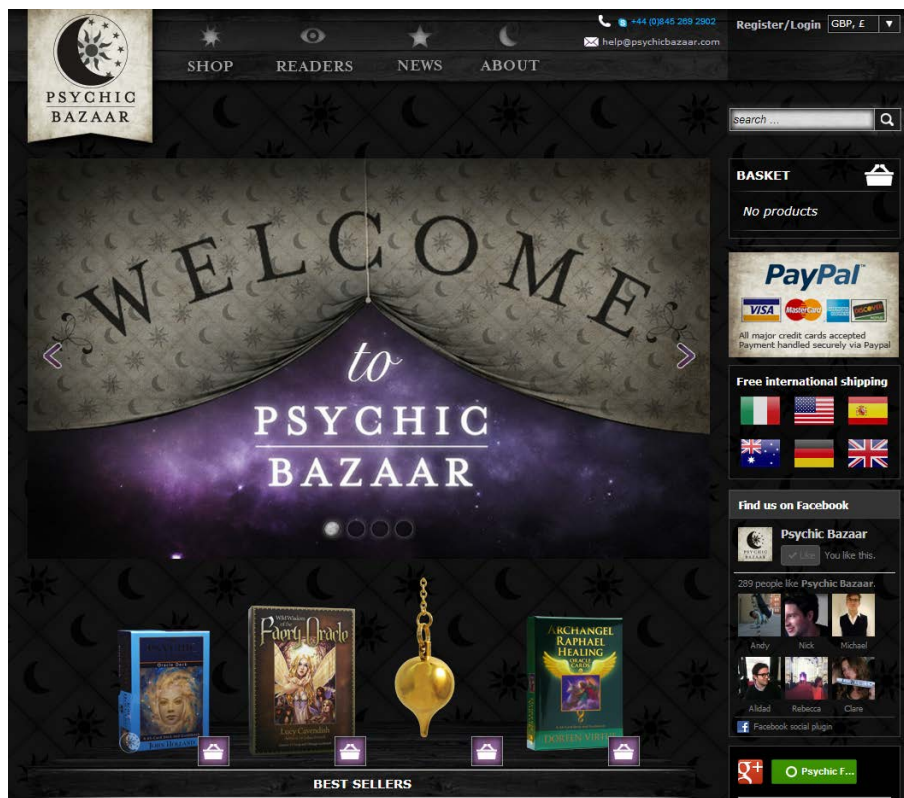
A 'classifieds' section, where Psychics can advertise their services:

And a blog / news feed:

We would therefore expect three groups of entities: those related to the online retail element, those related to the classified section, and those related to the blog.

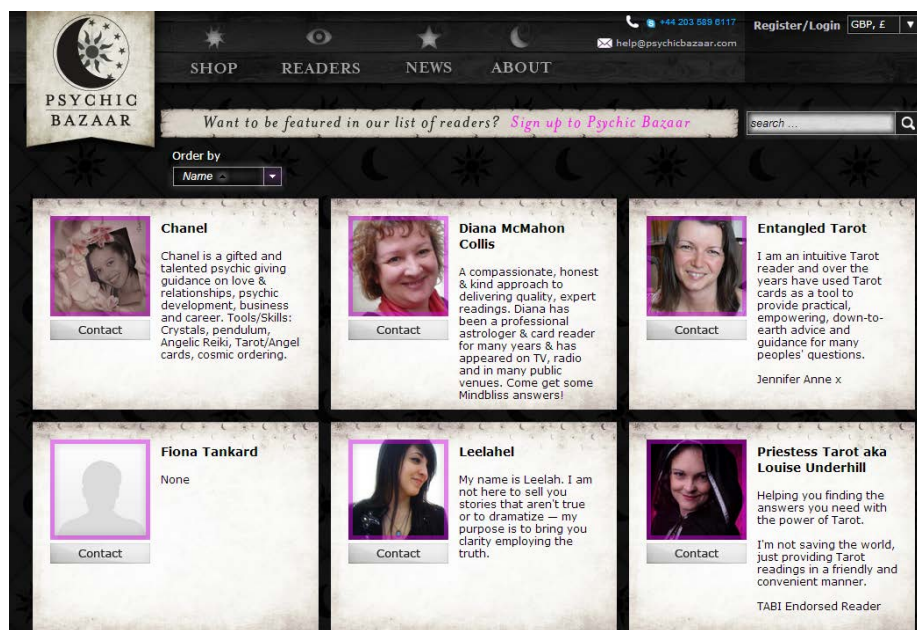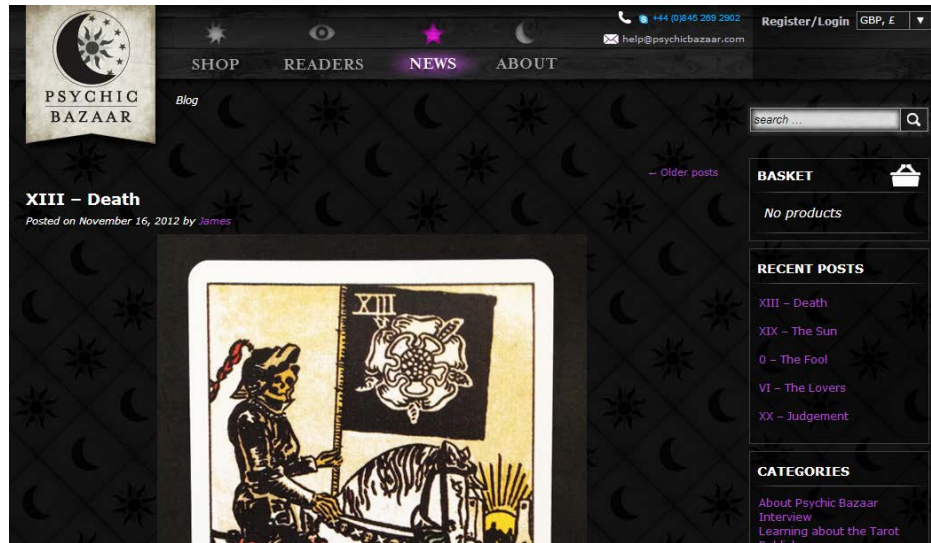## 2.2 Identifying the static entities that make up the web pages

2.2.1 ONLINE RETAIL

Our relevant entities for online retail specifically are:

- Products
- Listings
- Pages

It does **not** include

- Baskets
- Transactions

The reason is that baskets and transactions are dynamic entities that result from actions that a user takes (e.g. *adding a product to basket* and then *entering payment details on a checkout page*). We therefore include these entities as part of our **event model**, which we develop in section [XXX].

2.2.1.1 Products

Field	Example values	Description
product_sku	pbz00123	Unique product identifier
product_name	'Tarot of the Moon Garden'	Product name
product_tags	['tarot', 'contemporary', …]	Array of qualities / features / attributes associated with the product. For Psychic Bazaar, this should include the top level category the product belongs to (e.g. 'Tarot', 'Oracle', 'Jewellery', a list of all

		the '**genres**' a product is associated with, as well as '**publisher / manufacturer**' of the product and the product '**type**'. (There may be more than one genre and type for each product.)
product_currency	'gbp'	Currency code
product_unit_price	'19.99'	Product price

Example:

```
'product': {
        'product_sku': 'pbz00123',
        'product_name': 'Tarot of the Moon Garden'
        'product_tags': ['tarot', 'contemporary', …],
        'product_currency': 'gbp',
        'product_unit_price': '19.99'
}
```

2.2.1.2 Listing

Field	Example values	Description
type	full, box-summary, montage-summary	There are three types of listing on the PBZ site: full page listings, and summary listings (which can either be a single picture i.e. 'box' or a 'montage'
items	[product1, product2…]	An array of products that make up a listing

Example:

```
'listing': {
        'type': 'box-summary',
        items: [
                { 'product_sku': 'pbz00123', 'product_name': 'Tarot of the Moon
                Garden', 'product_tags': ['tarot', 'contemporary'],
                'product_currency': 'gbp', 'product_unit_price': 19.99 },
                { 'product_sku': 'pbz00044', 'product_name': 'Evil eye bracelet',
                'product_tags': ['jewellery', 'eastern'], 'product_currency': 'gbp',
                'product_unit_price': ' 9.99' },

                …

        ]
}
```

2.2.2 CLASSIFIED SECTION

Our relevant entities for the classified sections are:

- Readers
- Listings

2.2.2.1 Readers

There are only two data points we need to collect for each reader:

Field	Example values	Description
reader_id	44	Unique reader identifier
reader_name	"Chanel"	Reader name

2.2.2.2 Listing

Listings in the classified section is just an array of readers.

Field	Example values	Description
Readers	[reader1, reader2,…]	Array of readers

```
{ 'readers' : [
        { 'reader_id': '44', 'reader_name': 'Entangled Tarot' },
        { 'reader_id': '73', 'reader_name': 'Leelahel' },
        ...
    ]};
```

2.2.3 BLOG / NEWS SECTION

Our relevant entities for the classified sections are:

- Posts
- Listings

2.2.3.1 Posts

The following data should be captured for each post:

Field	Example values	Description
title	"Re-discovering some old favorites"	Post title
author	"James"	Post author
posted_on	2012-10-31	Date the post was published

```
'post': {
        'title': 'Re-discovering some old favorites',
        'author': 'James',
        'posted_on': '2012-10-31'
}
```

2.2.3.2 Listing

An array of posts

Field	Example values	Description
Posts	[post1, post2,…]	Array of posts

```
'listing': {
        { 'title': 'Re-discovering some old favorites', 'author': 'James',
        'posted_on': '2012-10-31' },
        { 'title': 'The Black Tarot Uncovered', 'author': James, 'posted
        on': '2012-11-23' },
        {...}
}
```

2.2.4   OBJECTS THAT ARE RELEVANT ACROSS ALL THREE SECTIONS (RETAIL, CLASSIFIED AND BLOG/NEWSFEED): PAGES

As well as identifying the above entities on the different sections of the Psychic Bazaar website, we also want to record the type of every web page served. There are a finite number of possible values. We would want to capture page type as part of SnowPlow page tracking.

Field	Value	Description
pageCategory	'Home', 'Catalogue', 'Product', 'Checkout', 'Confirmation'	The type of page. (There are four categories of page in the retail section)
	Home	Homepage
	Catalogue	Part of the retail site: where product listings are given
	Product page	Web page devoted to a particular product
	Checkout	Checkout page
	Confirmation	Order confirmation page
	Readers	Page with a listing of readers
	Account	User accont page used by readers to modify their profiles
	Blogfeed	Web page with listing of blog posts
	Post	Web page with a single blog post
	Other	Static pages e.g. "about Psychic Bazaar", "Jobs" etc.

Note: other page-level data (e.g. URL, page title etc.) will be captured automatically by SnowPlow – none of it needs to be passed to the `dataLayer` explicitly.

2.3 Declaring the static entities in the in the `dataLayer` at pageload

The entities identified in the previous section comprise the static entities we want to identify on each web page, and pass into Google Tag Manager so that they can be passed on to SnowPlow.

Because these entities describe the contents of each web page, they should be declared to the dataLayer on page load as per the dataLayer specification e.g.

```
<script>
   dataLayer = [{
      'pageCategory' : 'catalogue',
      'listing': [
         {'product_sku' : 'pbz00123', 'product_title': 'Tarot of the old
         path', 'product_tags': ['tarot', 'contemporary'],
         'product_currency': 'gbp', 'product_unit_price': 19.99},

         {'product_sku' : 'pbz00044', 'product_title': 'Thoth Tarot deluxe
         edition', 'product_tags': ['tarot', 'thoth', 'set'],
         'product_currency': 'gbp', 'product_unit_price': 39.99},

         {...},

         {...},

         ...

      ]
   }]
</script>
```

3   Identifying the events that can occur on a user journey

3.1   Overview

As in the preceding section, we're going to comb the each section of the website individually to identify the relevant events and associated data points that we need to capture in each of the three sections of the website i.e. the retail, classified and blog/news sections.

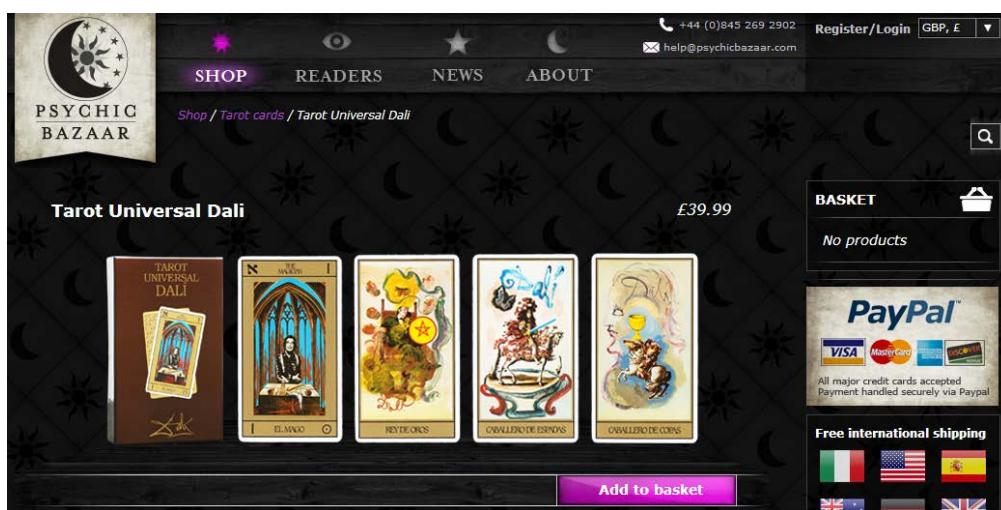## 3.2   Identifying and declaring the AJAX events that occur on a user journey

3.2.1   RETAIL SECTION

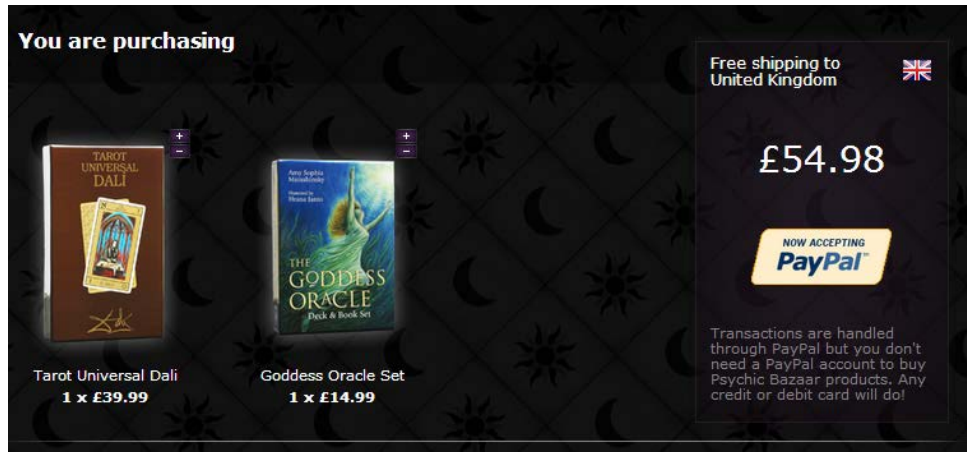#### 3.2.1.1   Add-to-baskets

Add-to-baskets are possible on catalogue pages:
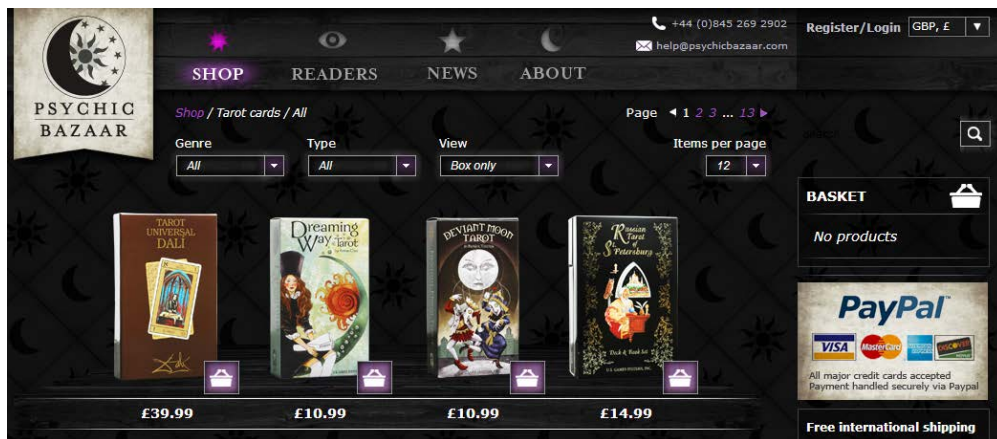
On product pages:

And on the checkout page (where the quantity of products that are already in the basket can be upped):

In each case, we need to pass to the dataLayer:

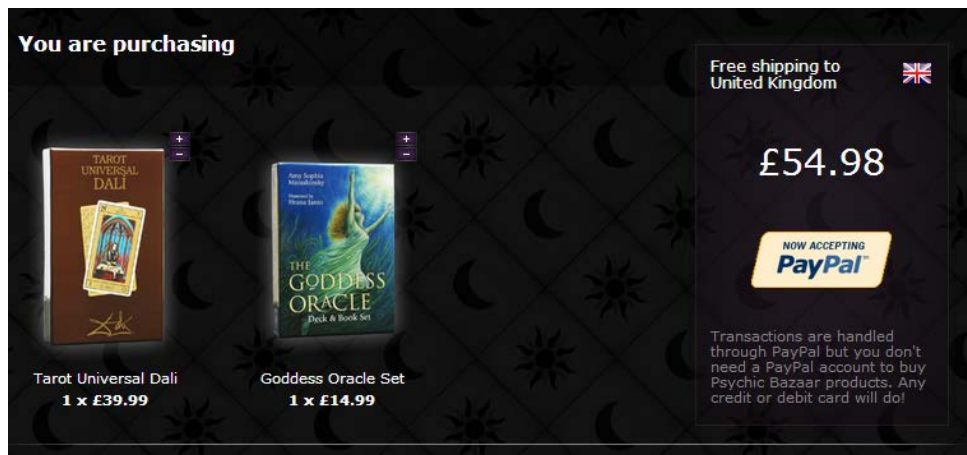| Field	Example value	Description
event	'add-to-basket'	Item added to basket
product	{'product_sku' : 'pbz00123', 'product_title': 'Tarot of the old path', 'product_tags': ['tarot', 'contemporary'], 'product_currency': 'gbp', 'product_unit_price': 19.99}	Product added to basket, formatted according to the product specification. Includes the SKU and the unit price and currency
quantity	1	Number of items added-to-basket. On the Psychic Bazaar website it is only possible currently to add one of each item to the basket at a time

```
dataLayer.push({
    'event' : 'add-to-basket',
    'product' : {
        'product_sku' : 'pbz00123',
        'product_title': 'Tarot of the old path',
        'product_tags': ['tarot', 'contemporary'],
        'product_currency': 'gbp',
        'product_unit_price': 19.99
        },
    'quantity': 1
    });
```

3.2.1.2 Remove-from-basket

It is possible to remove items from the basket on any page where the basket is present. (Clicking 'X' by the item in the basket removes it.)
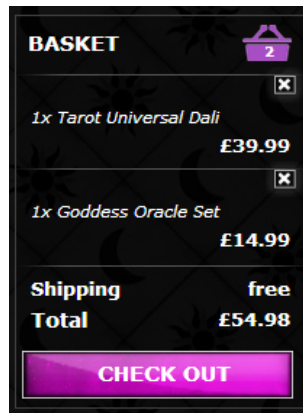
In addition, items can be removed from the basket on the checkout page:

The following data needs to be captured when *remove-from-basket*

Field	Example value	Description
event	'remove-from-basket'	Item added to basket
product	{'product_sku' : 'pbz00123', 'product_title': 'Tarot of the old path', 'product_tags': ['tarot', 'contemporary'], 'product_currency': 'gbp', 'product_unit_price': 19.99}	Product added to basket, formatted according to the product specification. Includes the SKU and the unit price and currency
quantity	1	Number of items added-to-basket. On the Psychic Bazaar website it is only possible currently to add one of each item to the basket at a time

```
dataLayer.push({
    'event' : 'remove-from-basket',
    'product' : {
        'product_sku' : 'pbz00123',
        'product_title': 'Tarot of the old path',
        'product_tags': ['tarot', 'contemporary'],
        'product_currency': 'gbp',
        'product_unit_price': 19.99
        },
    'quantity': 1
    });
```

3.2.1.3 Fill out the checkout form

The Psychic Bazaar checkout contains a number of fields. We want to track whenever a user enters something in one of those fields: including which field they edited, what they entered, and whether it was valid.

Field	Example value	Description
event	'checkout'	Checkout form filled in
field	'first name', 'last name'	Name of field edited
value	'John Doe'	Value entered
valid	true, false	Flag to indicate whether the value is 'accepted' by the website. Not set on fields where no validation is performed

```
dataLayer.push({
      'event': 'checkout',
      'field': 'first name',
      'value': 'John Doe'
});
```

3.2.1.4 Clicking on the Paypal button

We want to record the event that the user clicked on the Paypal checkout button. There is no data, however, that we need to capture with that event.

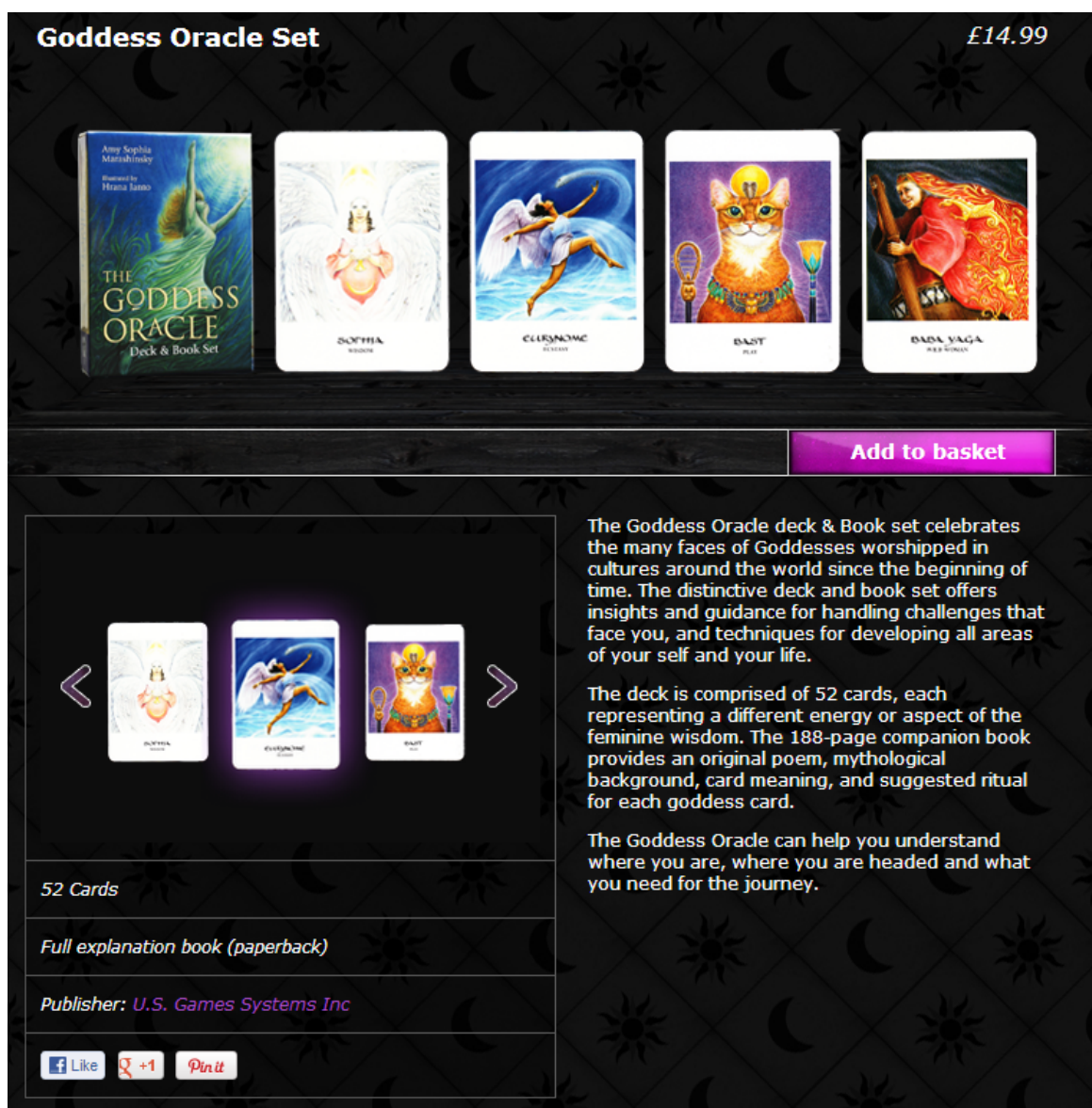| Field	Example value	Description
event	'paypal'	Click button to visit Paypal and arrange payment

3.2.1.5 Completing the transaction

On the order confirmation page, we want to record the complete transaction including all the products bought.

Field	Example value	Description
event	'transaction'	
transactionId	'0123'	Unique order identifier
transactionTotal	39.99	Total order value
transactionCurrency	'gbp'	Currency
transactionShipping	3.00	Shipping cost
transactionTax	4.14	Total tax (VAT)
transactionProducts	[product 1, product 2]	Array of products that were part of the order
transactionCity	'London'	Delivery address, city
transactionCountry	'UK'	Deliver address, country
transactionPostcode	'N3 1TH'	Delivery address, postcode

```
dataLayer.push({
     'event': 'transaction',
     'transactionId': 'order-123',
     'transactionAffiliation': 'web',
     'transactionTotal': '127.93',
     'transactionShipping': '7.99',
     'transactionTax': '19.99',
     'transactionCurrency': 'GBP',
     'transactionProducts': [
          { 'product' : {
               'product_sku' : 'pbz00123',
               'product_title': 'Tarot of the old path',
               'product_tags': ['tarot', 'contemporary'],
               'product_currency': 'gbp',
               'product_unit_price': 19.99
               },
          'quantity': 5 }
     ],
     'transactionCity': 'London',
     'transactionCountry': 'United Kingdom',
     'transactionPostcode': 'A1 2BC'
});
```

3.2.1.6 'Liking', 'G+ing' or 'Pinning' a product

Each product page contains three buttons that enable users to share the product on Facebook, G+, or Pinterest:
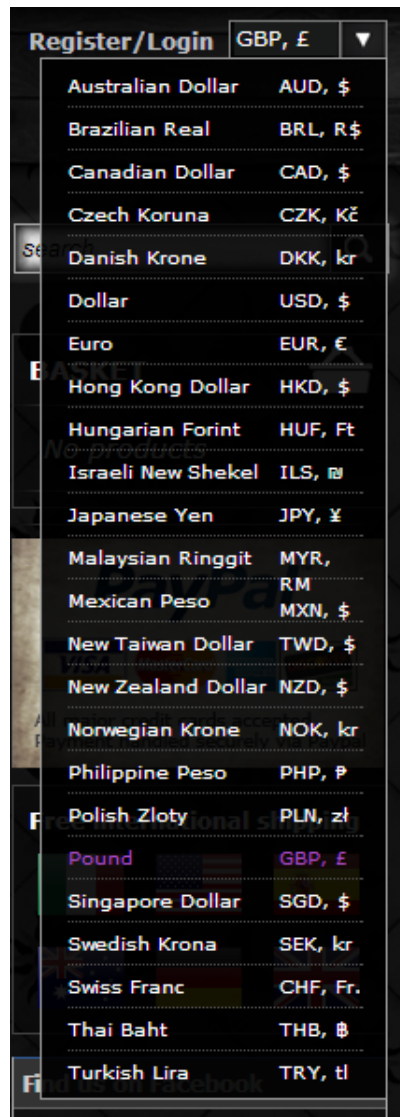

If any of the buttons are pressed, we want to capture the user action:

Field	Example value	Description
event	'like', 'g+', 'pin-it'	
object	'pbz00023'	The 'thing' shared i.e. the product SKU

```
dataLayer.push({
      'event': 'like',
      'readerId': 'pbz00023'
});
```

3.2.1.7 Changing the currency

Users can change via a dropdown on the top menu:

The following data should be captured:

Field	Example value	Description
event	'set-currency',	
object	'JPY'	The currency set (denoted by 3 digit ISO code)

```
dataLayer.push({
      'event': 'set-currency',
      'readerId': 'JPY'
});
```

3.2.2 CLASSIFIED SECTION

There are two sets of events that can occur to users in the classified section, which do not involve page loads (so need to be captured via event tracking):

1. They can express ask for the contact details of one of the psychics listed, by clicking on the 'Contact' button on his / her listing

2. They can create create accounts and edit their account details (which is necessary to create and modify their classified ad)

3.2.2.1 Contacting a Psychic

If a user wants to get in touch with a psychic who's listed on the directory, he / she clicks on the contact button on the ad, which brings up a modal window:
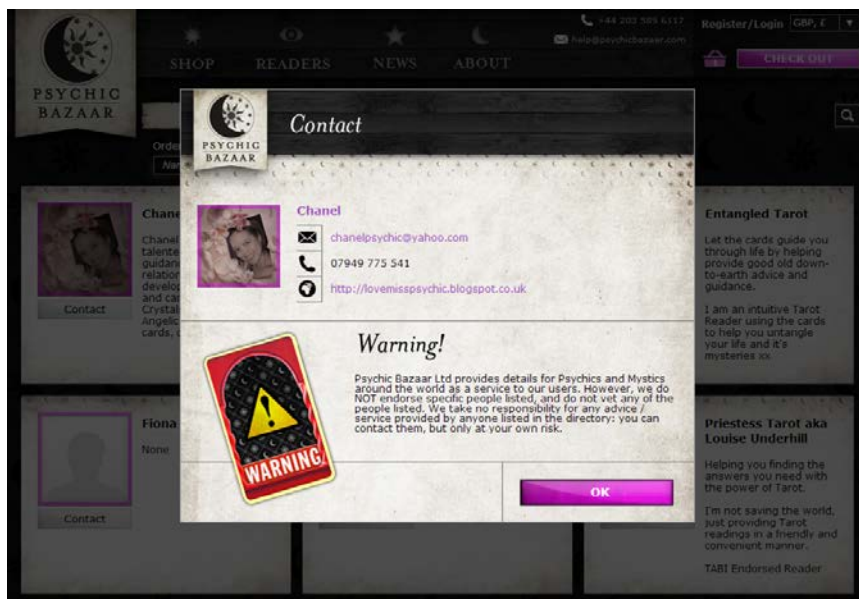

We need to capture the following data:

Field	Example value	Description
event	'contact-reader'	
reader_id	18	Unique identifier for each Psychic. We can use this to see analyse which listings drive the most interest

```
dataLayer.push({
     'event': 'contact-reader',
     'readerId': '18'
});
```

3.2.2.2 Create an account

Users can create an account by clicking the `Register/Login` link on the top right of every page:
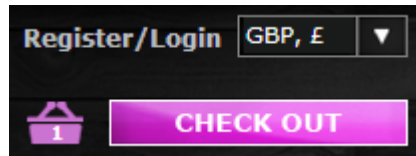
We want to capture this as an event:

Field	Example value	Description
event	'register/login'	

```
dataLayer.push({
        'event': 'register/login'
});
```

Users can also create an account by clicking on the 'Sign up to Psychic Bazaar' on the Readers page:
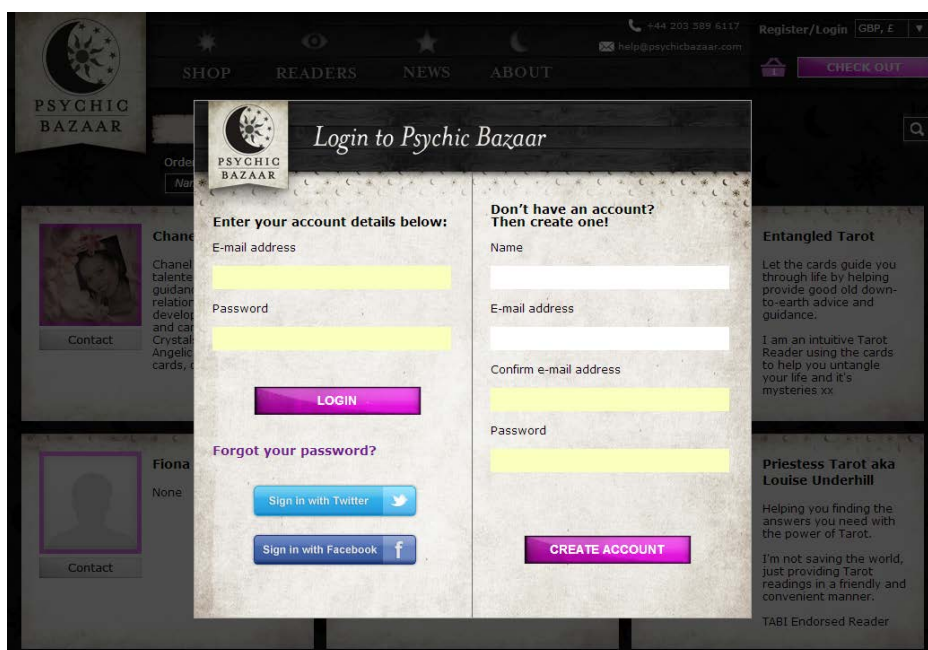

We want to capture this as a different event:

Field	Example value	Description
event	'sign-up-to-psychic-bazaar'	

```
dataLayer.push({
        'event': 'sign-up-to-psychic-bazaar'
});
```

By distinguishing the two ways of initiating account creation, we will be able to determine which drives most users.

In both cases, the links cause a modal window to pop up:

We want to capture the loading of the modal window as an event. If the user clicks on the 'login / signup'

Field	Example value	Description
event	'login' or 'signup'	If users enter values for fields on the left, the event should be set to 'login'. If they modify fields on the right, it should be set to 'signup'
field	'email-address', 'password', login' confirm-email', 'sign-in-with-twitter', 'sign-in-with-facebook', 'login', 'create-account'	The name of the field the user enters a value for. (Or clicks a button in the case of the Twitter / Facebook buttons.)
value	'contact@snowplowanalytics.com'	Value entered in the form. For fields which correspond to buttons rather than text boxes (e.g. 'sign in with Twitter' of 'CREATE ACCOUNT', this should be blank
success	true or false	This is only set on the 'LOGIN' or 'CREATE ACCOUNT' buttons. If the action is successful (i.e. user logs in or creates an account), it should be set to 1. This is so we can identify if a user has difficulty performing either action

```
dataLayer.push({
     'event': 'login',
     'name': 'John Doe'
});
```

3.2.2.3 Edit your account page

In order to edit his / her classified ad, a psychic updates the information stored in his / her profile page:

Each update is made via AJAX, so we need to capture them using event tracking. We want to capture the following fields:

Field	Example value	Description
event	'update-account'	
field	'include-me-in-reader-directory', 'photo', 'change-profile-image', 'few-words-about-yourself', 'location', 'reading-languages', 'years-experience', 'reads-with', 'email', 'web', 'tel', 'password', 'currency', 'twitter', 'facebook', 'email-updates-for-me', 'email-news-from-pbz', 'email-other'	The different fields on the account page that a use can edit
value	true, false, 'I am an accomplished Psychic…'	The actual values entered into the field. Note: for password field this will always be blank. For the 'change-profile-image' field it will also be blank. For checkbox fields it will be 'true' or 'false'

```
dataLayer.push({
      'event': 'update-account',
      'field': 'include-me-in-the-reader-directory',
      'value': true
});
```

3.2.3   BLOG / NEWS SECTION

Currently there are no events that can occur to a user in the blog / news section that are not captured by page views. There are therefore no events to capture.

We understand that there is a plan to enable social sharing of blog posts (in the same way as users can share products): when that is implemented, we recommend capturing those events in the same way as sharing of products is captured as documented in this specification.

3.2.4   EVENTS THAT ARE NOT TIED TO ANY SECTION IN PARTICULAR

There are a handful of events that not tied to one of the 3 sections that make up Psychic Bazaar in particular.

3.2.4.1 Liking or 'G+ing' Psychic Bazaar as a whole

Earlier we documented capturing events where a user shares a specific product via a social network. It is also possible, on the homepage, to share Psychic Bazaar as an entity, on Facebook on G+:

Field	Example value	Description
event	'like', 'g+',	
object	'psychicbazaar'	The 'thing' shared i.e. Psychic Bazaar itself

```
dataLayer.push({
      'event': 'g+',
      'readerId': 'psychicbazaar'
});
```

4   Appendix: a methodology for working out what data to pass the `dataLayer`, and how to structure it

SnowPlow is unique amongst web analytics platforms in that we believe in tracking *everything*, and working out afterwards how to use that data to drive business value. In a traditional business analytics (and web analytics) approaches, uses need to be able to demonstrate the value of particular data points, before money is invested in collecting, warehousing and processing that data.

As a consequence of this difference in approach, SnowPlow requires that much more data is passed to your tag manager (be it Google Tag Manager or any other tag management system) that other web analytics platforms require. That puts us in a unique position to develop an overarching methodology for deciding what data should be passed to the tag manager, and define the steps to identify that data.

Broadly speaking, there are two types of data that we are interested in processing in SnowPlow: event data and page-level data.

4.1.1   EVENT DATA

At its heart, SnowPlow is a tool for capturing, storing and analysing event-stream data, with a focus on web event data. We aim to capture all the events that occur on an individual customer's journey, so we can do interesting things including:

1. Identify `sliding doors` moments that have a big impact on the future behaviour and value of a customer

2. Spot bits of the customer journey that do not work and improve them, for example through product development

3. Segment users by how they behave

To do this, it is critical that we capture *all* the events on a user journey. This is what stages (3) and (4) in the methodology are all about.

4.1.2   WEB PAGE ENTITY DATA

 As the web evolves, websites look less-and-less like document stores and more-and-more like interactive applications. A larger and larger fraction of interesting events in a user journey become AJAX events that happen between page loads. These are all things that are captured using SnowPlow event tracking tags.

Even with this evolution in the web, pageviews are still a large and very significant component of a user's web journey. Which pages a user has visited, and what information on each they have digested, is therefore key. Unfortunately, unpicking that from a log of URLs, long after the event, is not straight forward.

A better approach is to track now just what web pages were viewed, but what content was on those pages, and how much of them the user engaged with. SnowPlow enables the capture of this type of data, so that you can analyse later things like:

1. Conversion rates by product. (What fraction of the user's who viewed a particular product went on to add it to basket, or to buy it?)

2. Compare how far down different articles users read

In order to do this, however, we need to pass the relevant entity data to our tag management programme, so it can pass it on to SnowPlow. That is what steps (1) and (2) in the methodology are all about.