# GTM Recipes – Integrating Smartech JS

[GTM Recipes – Integrating Smartech JS 1](#_Toc502320762)

[Introduction 3](#_Toc502320763)

[Usage the recipes 4](#_Toc502320764)

## Introduction

Google Tag Manager recipe is a ready-made and reusable GTM settings consisting of predefined tags, triggers and variables, which can be easily imported into the GTM container.

Importing can be done inside the GTM container under Admin section.

The settings in the recipes includes variable definitions, triggers and also tags against Smartech JS (Custom Activities).

We have set up the recipes for different industry verticals, given below:

1.E-commerce

2. Travel

3. BSFI

Whenever we set Smartech JS on a clients GTM container we tend to start from scratch / base i.e creating variables to be passed in the JS payload, define required triggers and then set smartech JS in tags.

These recipes will help avoid many of the steps involved in the setting up GTM container.

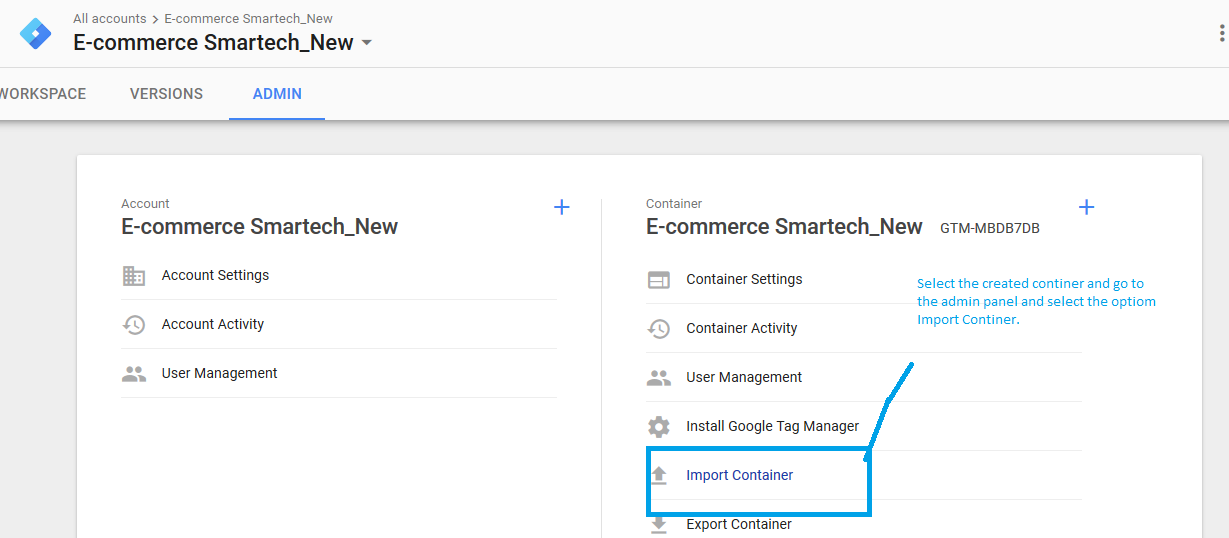
With these recipes our focus will be majorly on the variables and triggers. Tags are taken care on its own in large number of cases except few.

## 

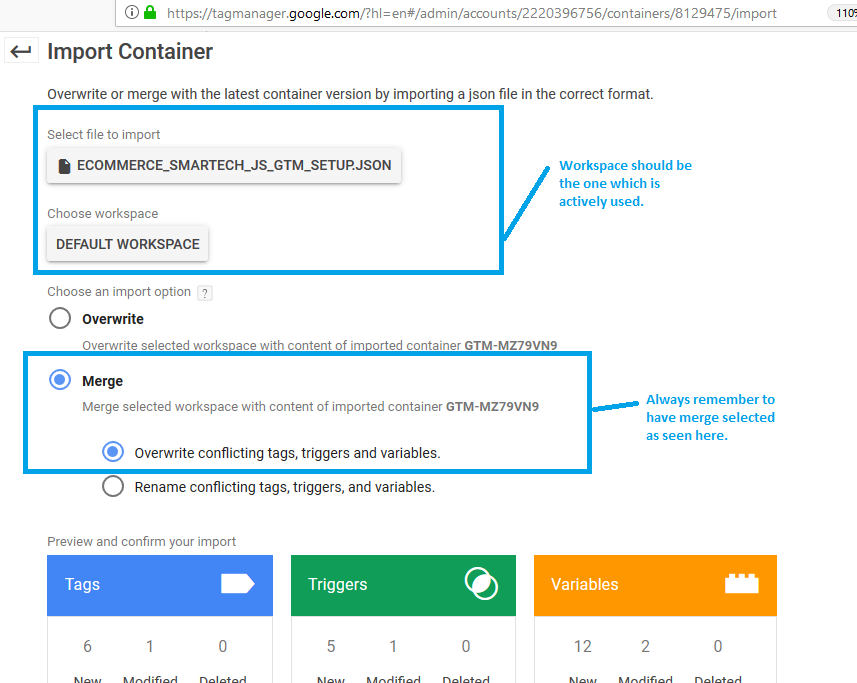
## Usage the recipes

1. **Import the recipe (JSON file) in the GTM container.**

In the admin section of the container there is an import container option, we can use this option and select the JSON file to import the pre-defined settings in the container.

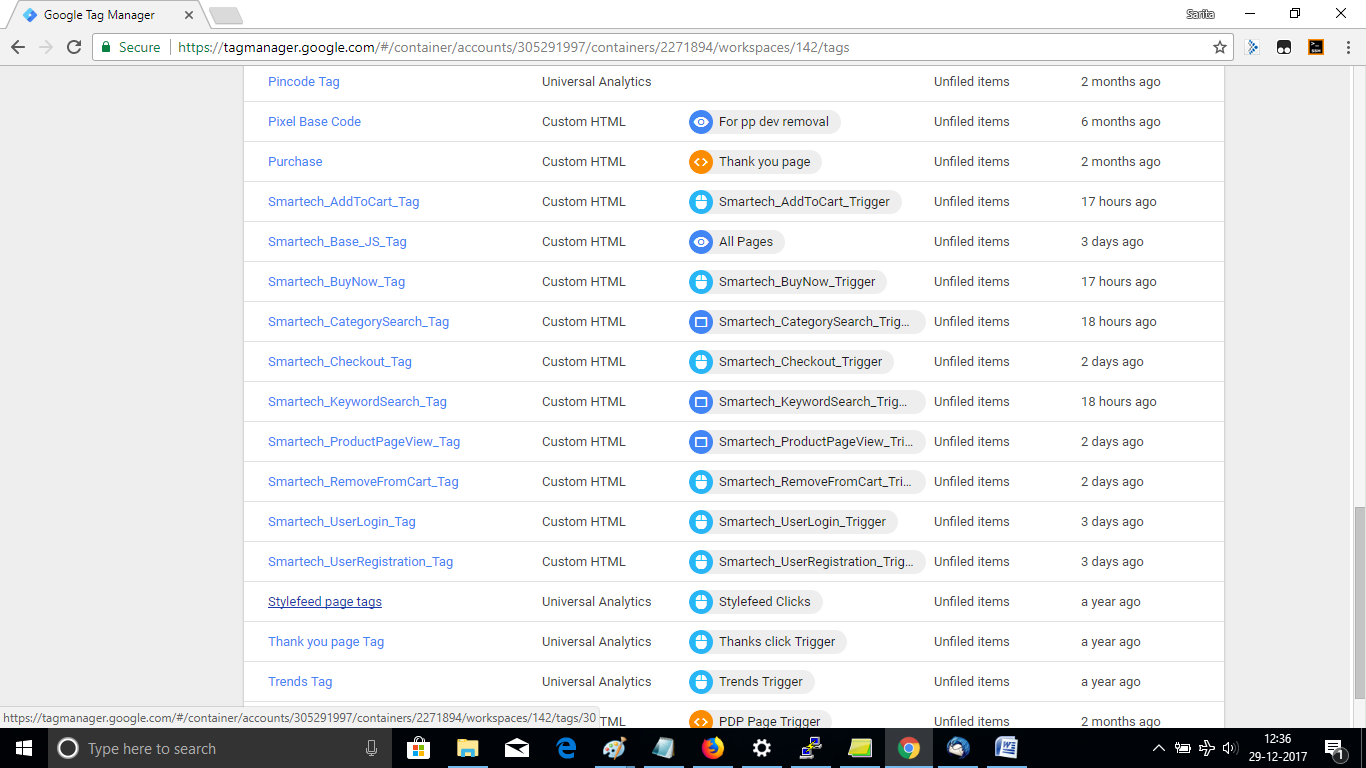


While importing the JSON file always select the active workspace and also select the Merge option, so that the setting add to the existing setup of the GTM (As showing the image in the next page).



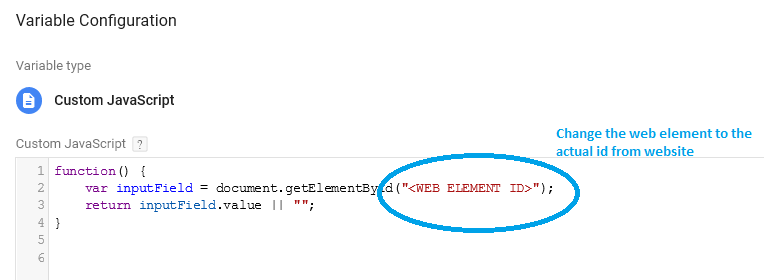
1. **The pre-defined variables, triggers and tags will be available in the container with prefix 'Smartech\_' for all the items.**

Post the settings are imported, navigate to the variable / trigger / tag or overview and locate for Smartech\_ keyword prefix and all the pre-defined settings will be visible in the container.

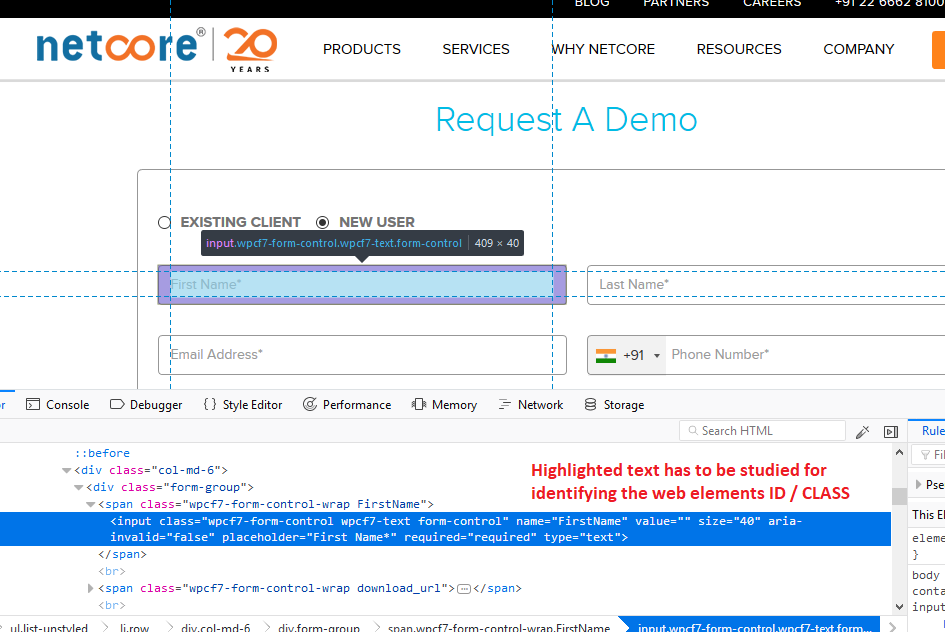


1. **In the variables editing will be required for replacing <WEB ELEMENT ID> with actual web element id for the desired payload parameters.**

All the variables comes with pre-defined custom JS code to identify the web element as given below with ID of the web element. We can modify to have getElementsByClassName as well.



In order to identify / locate the ID or CLASSNAME of the web element we can inspect the web element and find out as shown in the below image



We will need to edit the <WEB ELEMENT ID> with the actual web element id of the element picking from page source / inspect element.

For eg. assuming a web elements code to be

<input class=”MY TEST CLASS” name=”MY NAME” id=”MY ID” value=”” size=”40” type=”text>

Then the variable definition will be

*function() {*

*var InputField = document.getElementById(‘MY ID’);*

*return InputField.value || “”;*

*}*

In some of the cases where the web element does not have id base implementation and only uses class name and some other identifiers then we can still use the similar code and use getElementsByClassName and proceed with the GTM setting. Given below is a sample example.

<input class=”MY TEST CLASS” name=”MY NAME” value=”” size=”40” type=”text>

In this case the variable definition will be

*function() {*

*var InputField = document.getElementsByClassName(‘MY TEST CLASS’);*

*return InputField.value || “”;*

*}*

There are situations when the classname always is being used by multiple web element i.e multiple web elements of a form falls under same class. In such situations we go by the index number of the form fields to fetch the fields values. As an example given below is source code and corresponding GTM setting for variable.

<input class=”MY TEST CLASS” name=”MY NAME 0 ” value=”” size=”40” type=”text>

<input class=”MY TEST CLASS” name=”MY NAME 1” value=”” size=”40” type=”text>

<input class=”MY TEST CLASS” name=”MY NAME 2” value=”” size=”40” type=”text>

i. Capturing 1st field value

*function() {*

*var InputField = document.getElementsByClassName(‘MY TEST CLASS’);*

*return InputField[0].value || “”;*

*}*

ii. Capturing 3rd field value

*function() {*

*var InputField = document.getElementsByClassName((‘MY TEST CLASS’);*

*return InputField[2].value || “”;*

*}*

At times it does make sense to use Query Selector for the variable definition instead of the getting into index number of web elements and making mistakes.

Consider the below example

<input class=”MY TEST CLASS” name=”MY NAME” id=”MY ID” value=”” size=”40” type=”text>

*function() {*

*var InputField = document.querySelector(‘input[name=” MY NAME”]’);*

*return InputField.value || “”;*

*}*

<select class=”MY TEST CLASS” name=”list” id=”MY ID” value=”” size=”40” type=”text>

*function() {*

*var SelectField = document.querySelector(‘select[name=”list”]’);*

*return SelectField.value || “”;*

*}*

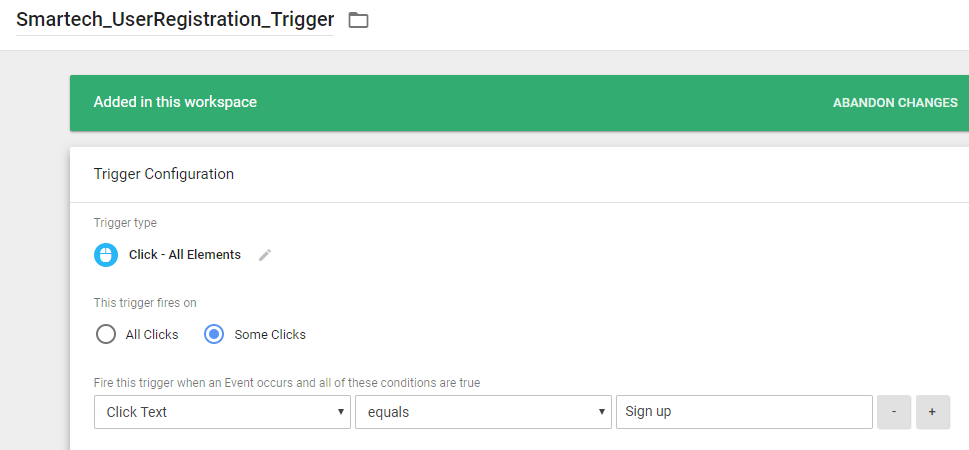
1. In the Triggers the clicks / page views must be set matching to the condition on as per the clients page details.

The triggers are created based on the industry specific recipe, but not configured to the website’s web element id (click id) or web element class (click class).

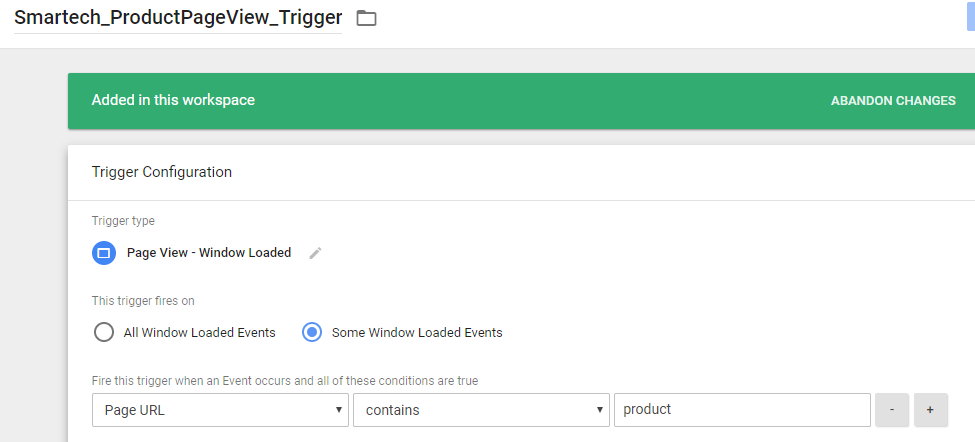
The change which needs to be done in the triggers is to locate the trigger corresponding to a user action and edit and add filter condition of Click ID / Click Classes etc whichever is feasible.

Given below are some examples of the various trigger possible.

1. Submit button click



1. Page / Window Loaded



1. In the Tags sections only the dispatch id's must be changes to the ones set on the Smartech panel.

