LEARN & MASTER GTM DATA LAYERS

2ND EDITION

WRITTEN BY

HIMANSHU SHARMA

FOUNDER OF OPTIMIZESMART.COM

OptimizeSmart.com

Google Tag Manager Data Layers

Written by Himanshu Sharma, Founder of Optimize Smart

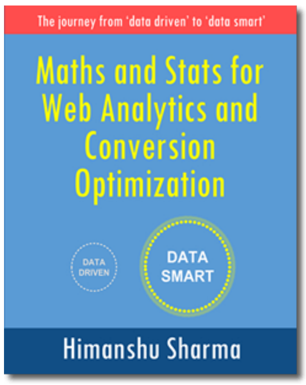
About the author

- Founder, OptimizeSmart.com
- Over 15 years of experience in digital analytics and marketing
- Author of four best-selling books on digital analytics and conversion optimization
- Nominated for Digital Analytics Association Awards for Excellence
- Runs one of the most popular blogs in the world on digital analytics
- Consultant to countless small and big businesses over the decade

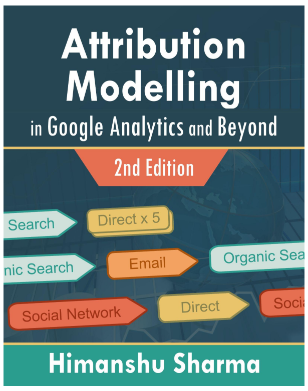
Website: www.optimizesmart.com

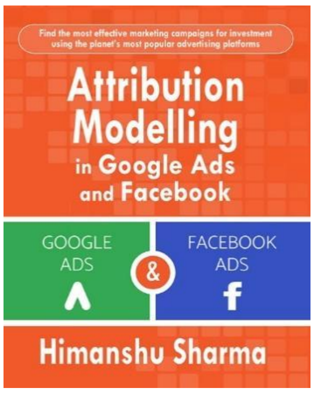
Linkedin: https://www.linkedin.com/in/analyticsnerd

Facebook: https://www.facebook.com/optimizesmart


The journey from 'data driven' to 'data smart'
Maths and Stats for Web Analytics and Conversion Optimization
DATA DRIVEN
DATA SMART
Himanshu Sharma

Master the Essentials of
Email Marketing Analytics
The journey from inbox placement to conversion
Himanshu Sharma





Following are our most downloaded ebooks for career advancement:

#1 **[Sales and ROI Accelerator (150+ pages)]**

WHAT'S INSIDE: My step-by-step blueprint for generating record sales and ROI by leveraging analytics data.

#2 **Set Up Your Google Analytics 4 (GA4) Account Correctly And Fast (70 pages)**

WHAT'S INSIDE: Learn to set up your GA4 account correctly and fast using this 62 points checklist.

FAQ: Do you show "How" to do each item on the checklist? If so, with screenshots?
Yes. There are links to the articles with detailed step by step instructions.

FAQ: Does this ebook cover GTM too?
Yes.

#3 Google Tag Manager Data Layers (100+ pages)

WHAT'S INSIDE: My step-by-step blueprint for getting started with data layers. Get the only ebook on GTM data layers ever published. Learn the JavaScript behind it.

#4 Learn to Read E-Commerce Reports in Google Analytics (100+ pages)

WHAT'S INSIDE: My step-by-step guide to reading both standard and enhanced e-commerce reports in Google Analytics. E-commerce reports are the most valuable reports in Google Analytics.

#5 Do you want better skills in digital analytics and marketing? If yes, then register for the free training:

Here’s what we’re going to cover…

1. Why digital analytics is the key to online business success.
2. The number 1 reason why most marketers are not able to scale their advertising and maximize sales.
3. Why Google and Facebook ads don’t work for most businesses & how to make them work.
4. Why you won’t get any competitive advantage in the marketplace just by knowing Google Analytics.
5. The number 1 reason why conversion optimization is not working for your business.
6. How to advertise on any marketing platform for FREE with an unlimited budget.
7. How to learn and master digital analytics and conversion optimization in record time.

Table of Contents

Get helpful tips on a daily basis

If you are the type of person who finds it helpful to receive short tips on building your website traffic, improving conversions, fixing attribution issues and learning about analytics in general, then follow me on LinkedIn. I post a few short tips each day.

Click here and follow me on LinkedIn

GTM is a really powerful tool and understanding the data layer is the key to getting the most out of Google Tag Manager.

However, in order to understand and use Data Layers, you would need to understand a little bit of JavaScript first.

Why JavaScript?

JavaScript is the most useful programming language for a web analyst.

All the measurement and marketing tags whether it is Google Analytics tracking code, ecommerce tracking code, event tracking code or Google Adwords conversion tracking code are written in JavaScript.

You need a good working knowledge of JavaScript in order to implement:

- Ecommerce tracking
- Event tracking
- Phone call tracking
- Scroll tracking

- Video tracking
- Social interactions tracking
- Enhanced ecommerce tracking
- Cross-domain tracking

…and just about any tracking in Google Analytics.

Google Analytics tracking code is a JavaScript snippet.

Google Tag Manager Data Layer is a JavaScript array.

Both ***analytics.js library*** and **gtag.js library** used by Google Analytics are JavaScript files. Without JavaScript, there would be no Google Analytics and Google Tag Manager would not be useful.

Role of JavaScript in a web document

The three most important scripting languages which are widely used to create a web document are HTML, CSS, and JavaScript.

HTML (or HyperText Markup Language) is used to develop text content of a web document.

CSS (or Cascading Stylesheet) is used to control the layout and formatting of the text developed through HTML.

JavaScript is used to control the behavior of HTML and CSS elements, users and web browsers.

These three scripting languages together create the modern web page which we see today.

How to add JavaScript to a web document

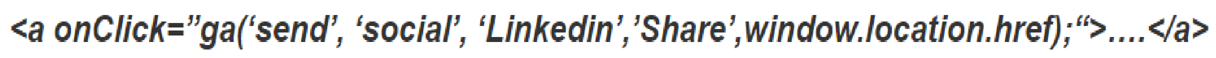
<u>There are three methods to add JavaScript code to an HTML document:</u>

1. Inline JavaScript
2. Embedded JavaScript
3. External JavaScript

#1 Inline JavaScript

Inline JavaScript is the JavaScript added to an HTML tag.

For example:

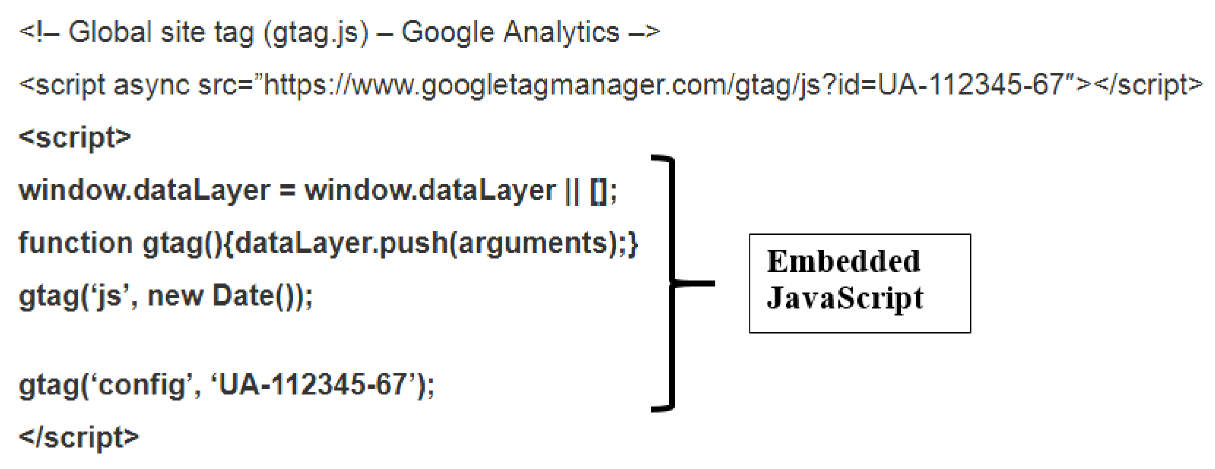


```
<a onClick=”ga(‘send’, ‘social’, ‘Linkedin’,’Share’,window.location.href);“>....</a>
```

#2 Embedded JavaScript

Embedded JavaScript is the JavaScript added to an HTML document using the <script> tag which doesn't contain the 'src' attribute.

For example:



```
<!– Global site tag (gtag.js) – Google Analytics –>
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-112345-67"></script>
<script>
window.dataLayer = window.dataLayer || [];
function gtag(){dataLayer.push(arguments);}
gtag('js', new Date());

gtag('config', 'UA-112345-67');
</script>
```

Embedded JavaScript

The bold text is the embedded JavaScript code.

As you can see, the Google Analytics tracking code that you copy-paste on each web page contains embedded JavaScript.

You need to use script tags in order to use embedded JavaScript.

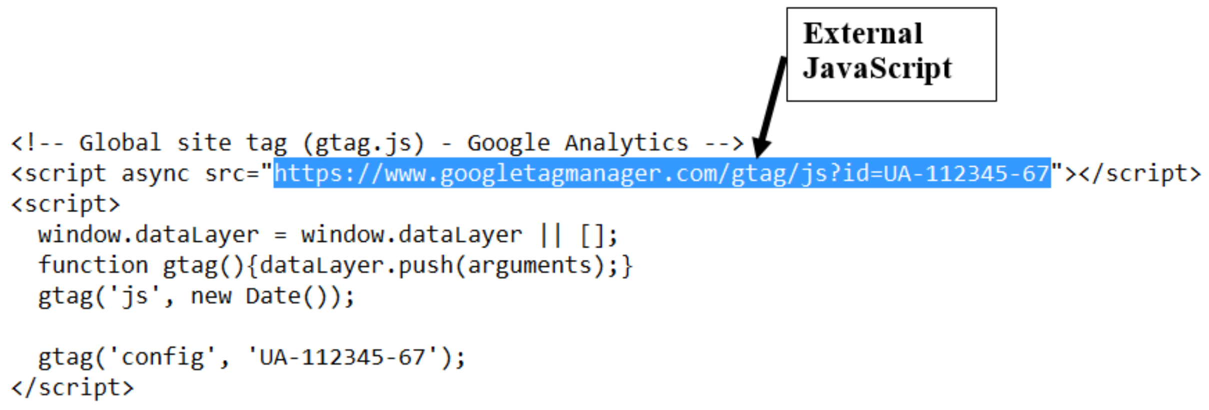
JavaScript code placed outside of the script tag will not execute.

Note: Every <script> tag must have the closing </script> tag and you can place script tags anywhere in the head or body section of an HTML document.

#3 External JavaScript

External JavaScript is the JavaScript added to an HTML document using the <script> tag which contains the 'src' attribute.

For example:



```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-112345-67"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'UA-112345-67');
</script>
```

Here all the JavaScript code is stored in an external JavaScript file and this file is called via the <script> tag.

External JavaScript is the recommended method to add JavaScript to an HTML document.

You should avoid using inline and embedded JavaScript wherever possible.

There are three big advantages of using external JavaScript:

#1 Your JavaScript code does not interfere with other JavaScript code added to the same page and thus enable unobtrusive scripting.

#2 It becomes easier to maintain, debug, understand and reuse JavaScript code

#3 Cached JavaScript files can speed up page loads.

Introduction to JavaScript statements

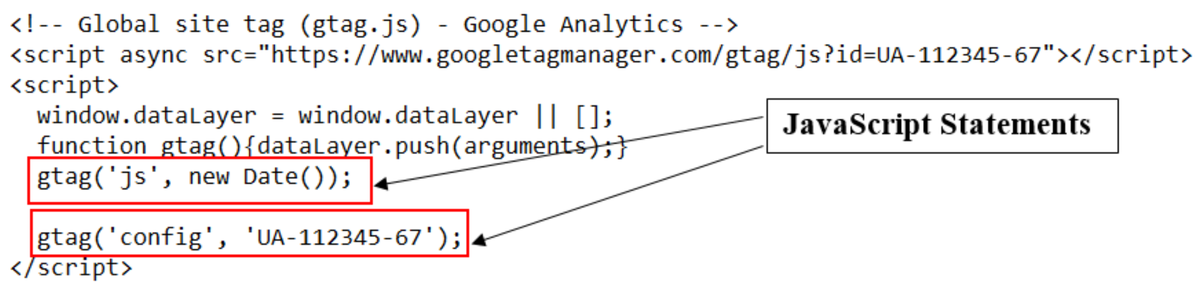
A JavaScript statement is an instruction which is executed by a web browser. Following is an example of JavaScript statement:

ga('create', 'UA-XXXX-Y', 'auto');

Another example of JavaScript statement:

ga('send', 'pageview');

The Google Analytics tracking code that you use is made up of bunch of JavaScript statements:



```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-112345-67"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'UA-112345-67');
</script>
```

Each JavaScript statement must end with a semicolon;

So following code is not valid:

gtag('js', new Date())

gtag('config', 'UA-112345-67')

This is the correct code:

gtag('js', new Date());

gtag('config', 'UA-112345-67');

This missing semicolon error may look trivial but it can easily create a hard to diagnose tracking issue. This is because it is not very easy to spot a tiny missing semicolon in a big piece of code.

Order of execution of JavaScript statements

JavaScript statements are executed in the order in which they are written.

By default, they are executed from top to bottom and from left to right.

The order of execution is as important as the code itself.

If you change the order of JavaScript statements, your code will start behaving differently or may stop working.

So following code is not valid:

```
gtag('config', 'UA-112345-67');

gtag('js', new Date());
```

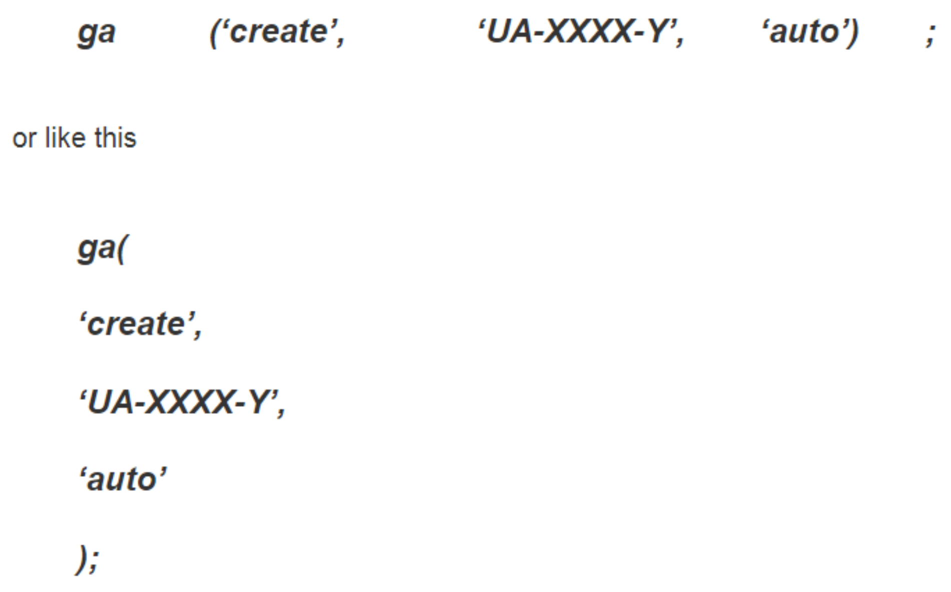
This is the correct code:

```
gtag('js', new Date());

gtag('config', 'UA-112345-67');
```

JavaScript and white spaces

JavaScript ignores multiple white spaces.

So the statement **ga('create', 'UA-XXXX-Y', 'auto');** can be written as:



```
ga       ('create',        'UA-XXXX-Y',      'auto')      ;
```

or like this

```
ga(
'create',
'UA-XXXX-Y',
'auto'
);
```

Similarly the statement **x=y+z;** can be written as **x = y + z ;**

It is a good practice to put spaces around operators (= + - * /).

Use extra white spaces to increase the readability of your JavaScript code.

In general, the use of extra white spaces does not create any issue in your code.

Introduction to comments in JavaScript

Comments are used to add a note to a JavaScript code and to stop a JavaScript statement from being executed.

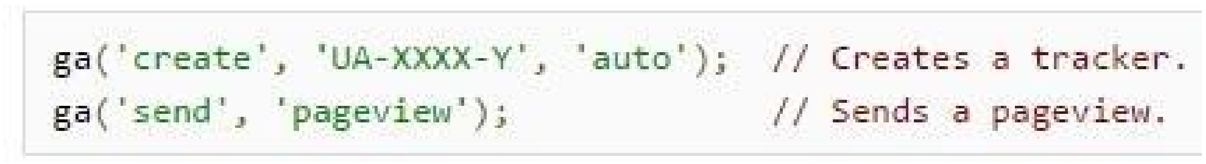
There are two types of comments in JavaScript:

#1 Single-line comments

#2 Multi-line comments

A single-line comment begins with two forward slashes (//) and runs to the end of the line.

For example:



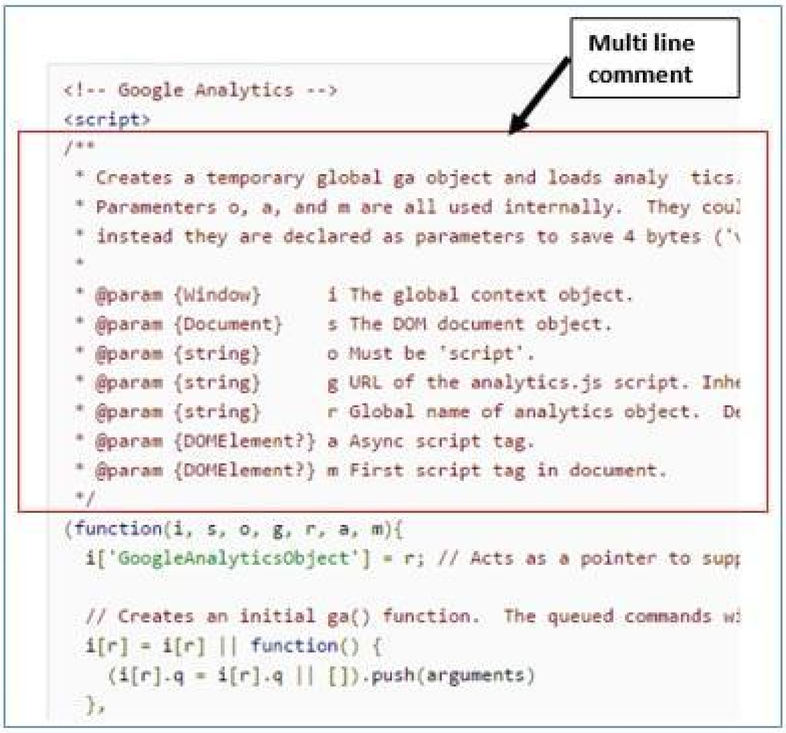
```
ga('create', 'UA-XXXX-Y', 'auto');  // Creates a tracker.
ga('send', 'pageview');             // Sends a pageview.
```

Here 'creates a tracker' and 'sends a pageview' are the single-line comments added to the Google Analytics tracking code.

Note: Any text between // and the end of the line is ignored by JavaScript i.e. it is not executed.

A multi-line comment starts with /* and ends with */ .

For example:



```
<!-- Google Analytics -->
<script>
/**
 * Creates a temporary global ga object and loads analy  tics.
 * Paramenters o, a, and m are all used internally.  They coul
 * instead they are declared as parameters to save 4 bytes ('\
 *
 * @param {Window}      i The global context object.
 * @param {Document}    s The DOM document object.
 * @param {string}      o Must be 'script'.
 * @param {string}      g URL of the analytics.js script. Inhe
 * @param {string}      r Global name of analytics object.  De
 * @param {DOMElement?} a Async script tag.
 * @param {DOMElement?} m First script tag in document.
 */
(function(i, s, o, g, r, a, m){
  i['GoogleAnalyticsObject'] = r; // Acts as a pointer to supp

  // Creates an initial ga() function.  The queued commands wi
  i[r] = i[r] || function() {
    (i[r].q = i[r].q || []).push(arguments)
  },
```

Note: Any statements between /* and */ will not be executed by JavaScript. The forward slash is used whether a comment is a single-line or multi-line comment.

Comments are commonly used to stop a statement from being executed during testing. For example following JavaScript statement won't execute as it has been turned into a comment:

// setTimeout("ga('send','event','Profitable Engagement','time on page more than 3 minutes')",180000);

Introduction to JavaScript variables

Variables are used to store values.

Creating a variable in JavaScript is called ***'declaring a variable'***.

You can declare a variable in JavaScript by using the keyword 'var'.

For example:

var _gaq; // create a variable and name it _gaq

Another example:

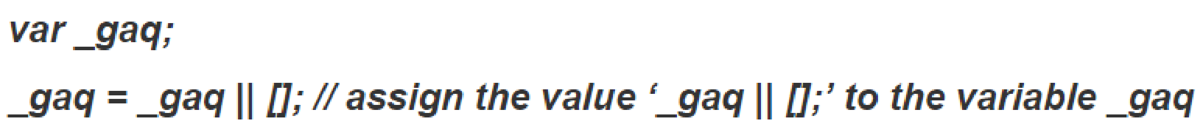
var pageTracker; // create a variable and name it pageTracker

Note: It is good programming practice to declare all variables at the beginning of a script/code.

The assignment operator '='

Use the assignment operator '=' to assign a value to a variable.

For example:



```
var _gaq;
_gaq = _gaq || []; // assign the value '_gaq || [];' to the variable _gaq
```

Another example:

```
var pageTracker;
pageTracker = _gat._getTracker(“UA-12345-1”); /* assign the value
_gat._getTracker(“UA-12345-1”) to the variable pageTracker */
```

You can also declare a variable and assign it a value at the same time.

For example:

```
var _gaq = _gaq || [];
```

How to use the value of a JavaScript variable

To use the value of a variable, just use the variable name without surrounding it with quotes (single or double quotes).

For example:

```
var videoName= ‘superman vs Batman’;
ga(‘send’, ‘event’, ‘videos’, ‘play’,videoName);
```

If you enclose a variable name in quotes (single or double quotes) then it will become a string value.

For example:

```
var videoName= ‘superman vs Batman’;
ga(‘send’, ‘event’, ‘videos’, ‘play’,’videoName’);
```

Here 'videoName' is a string value and not the variable we declared earlier.

So this event tracking code will pass 'videoName' as the event label instead of passing 'superman vs Batman' as the event label.

Rules for naming JavaScript variables

There are six characteristics of variable names in JavaScript:

#1 Variable names are case sensitive.

#2 Variable names can be alphanumeric.

#3 No spaces are allowed in a variable name.

#4 No special characters (other than $ and underscore) are allowed in a variable name.

#5 A variable name cannot begin with a number.

#6 A keyword cannot be used as a variable name.

#1 Variable names are case sensitive

For example, the following are all different variables:

```
var pageTracker;
var pagetracker;
```

var PageTracker;

var PAGETRACKER;

#2 Variable names can be alphanumeric

For example:

var alpha23;

#3 No spaces are allowed in a variable name

For example:

var first name = 'Himanshu'; // not valid

#4 No special characters (other than $ and underscore) are allowed in a variable name

For example:

var _gaq = "Peter"; // underscore is allowed in a variable name

var US$ = 1500; // dollar sign is allowed in variable name

var First-name= "anna"; // hyphen is not allowed in a variable name

#5 A variable name cannot begin with a number

For example:

var 23alpha; // not valid

#6 A keyword cannot be used as a variable name (more about keywords later)

Best practices for creating variable names in JavaScript

#1 Use lower-case for variable names wherever possible.

#2 Use either underscore to separate each word in variable name or use camel casing. For example:

var first_name = 'Himanshu';

var firstName= 'Himanshu';

Note: In a camel case we capitalize the first letter of each word except the first word.

Introduction to JavaScript keywords

A keyword (or identifier) is a reserved word that carries a special meaning.

Following are some examples of keywords in JavaScript:

1. var
2. function
3. return
4. for
5. true
6. false
7. null

8. undefined
9. if

So,

> **var function;** // not valid as 'function' is a keyword.
>
> **var positiveResult = true;** // valid
>
> **var negativeResult = false;** // valid

true and false are keywords in JavaScript. They are also **boolean values**.

So you don’t need to put any quote marks around them.

If you put quote marks around them then they will become string values.

For example:

ga('create', 'UA-XXXX-Y', {'allowLinker': true}); // valid

ga('create', 'UA-XXXX-Y', {'allowLinker': 'true'}); /* not valid because 'true' is a string value and allowLinker doesn't accept string */

Note: All JavaScript identifiers are case sensitive. So JavaScript does not interpret VAR or Var as the keyword var:

> **Var _gaq;** // not valid
>
> **VAr _gaq;** // not valid
>
> **VAR _gaq;** // not valid
>
> **var _gaq;** // valid

Data types in JavaScript

A JavaScript variable can store value of any data type. It can store:

1. Numeric value (whole number, decimal number)
2. String value
3. Boolean value
4. Undefined value
5. null value
6. Array
7. Multidimensional array
8. Object
9. Array of Objects
10. Other variables

For example:

Numeric values

```
var a = 3;
a = 3.14159265;
a = -3;
a = -3.14159265;
```

Strings

```
a ='himanshu';
a = "sharma";
```

Boolean values

```
a = true;
a = false;
```

Undefined value

```
a= undefined;
```

Null value

```
a = null;
```

Arrays

```
a = [1,2,3];
a= [[‘name’,’john’],[‘age’,20]];
```

Objects

```
a = {'age':24};
dataLayer = [{'name':’john’,’age’:23}, {'name':’ash’,’age’:42}];
```

Other variable

```
var half = 12;
var full = half * 2;
```

In JavaScript the datatype of a variable depends upon the value it has currently stored.

For example:

> **var _gaq;** // _gaq is undefined
>
> **_gaq = 5;** // Now _gaq has become a Number
>
> **_gaq= "John";** // Now _gaq has become a String
>
> **_gaq =true;** // Now _gaq has become a boolean
>
> **_gaq=[];** // Now _gaq has become an array
>
> **_gaq ={};** // Now _gaq has become an object

Because of this flexibility in data types, JavaScript programs are highly prone to errors.

Strings in JavaScript

A string is a series of characters of any length.

The string must be enclosed by single or double quote marks.

For example:

```
var name = 'Himanshu sharma';
var name2 = "John lewis";
var address = "28 station road";
var emailAddress= 'crazynitch[@]gmail[.]com';
```

Note: Any character available on a keyboard can be used in a string including white spaces.

You can use quotes inside a string, as long as they don't match the quotes surrounding the string.

Example-1:

```
var comment = "It's alright"; // valid
var wrestlerName = "Hunter 'Hearst' Helmsley"; // valid
ga('send', 'event', {
'eventCategory': 'Wrestling',
'eventAction': 'Heavy weight'',
'eventLabel': wrestlerName
});
```

Example-2:

```
ga('send', 'event', {
'eventCategory': 'Wrestling',
'eventAction': 'Heavy weight'',
'eventLabel': 'Hunter "Hearst" Helmsley' // valid
});
```

Example-3:

```
ga('send', 'event', {
'eventCategory': 'Wrestling',
'eventAction': 'Heavy weight'',
'eventLabel': 'Hunter 'Hearst' Helmsley' /* not valid because inner quotes
match with surrounding quotes */
});
```

Example-4:

```
var hhh = "Hunter "Hearst" Helmsley"; /* not valid because inner quotes
match with surrounding quotes */
```

If you want to use quotes inside a string which match the quotes surrounding the string then you need to use back slash \ (also known as escaping character).

For example:

```
var hhh = "Hunter \"Hearst\" Helmsley"; // valid
hhh = 'Hunter \'Hearst \' Helmsley'; // valid
```

If you enclose a number in quotes (either single or double quotes), it will be treated as string by JavaScript.

For example:

```
ga('send', 'event', 'videos', 'play', 'Superman', '40'); /* not valid because '40' is a string value and not number */
ga('send', 'event', 'videos', 'play', 'Superman', 40); // valid because 40 is a number
```

Needless to say, you need to be careful with how you use quotes in your JavaScript code.

You can make a perfectly valid number a string just by surrounding it with quotes.

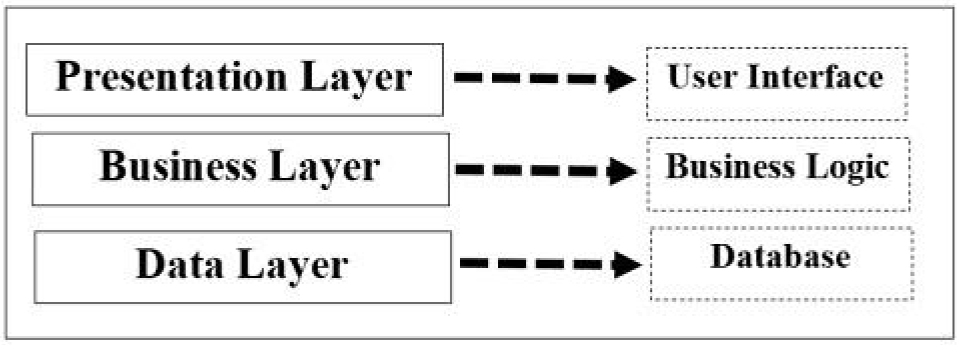
The three layers in programming

In the context of programming there are three different types of layers (also called 'logic'):

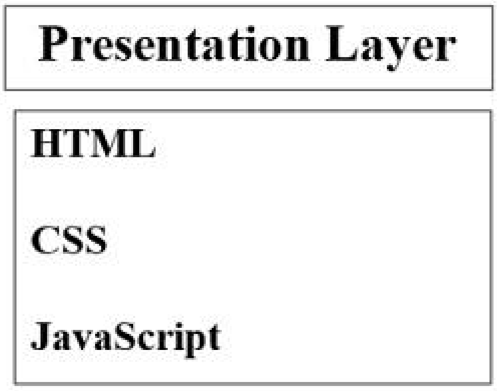
#1 Presentation Layer

#2 Business Layer

#3 Data Layer



#1 Presentation Layer (or Presentation Logic)



The presentation layer corresponds to the user interface. It is used to manage interaction with the users. The presentation layer is made up of one or more scripting languages. The three scripting languages which are most commonly used for creating a presentation layer are HTML, CSS, and JavaScript.

HTML (or HyperText Markup Language) is used to develop text content of a presentation layer.

CSS (or Cascading Stylesheet) is used to control the layout and formatting of the text developed through HTML.

JavaScript is used to control the behavior of HTML and CSS elements, users and web browsers.

When you interact with a web page, you are interacting with the presentation layer.

#2 Business Layer (or Business Logic)

The business layer corresponds to business rules and workflows that determine how data can be created, stored and changed. The business layer sits between the presentation layer and the data layer.

#3 Data Layer (or Data Logic)

The data layer corresponds to the data structure and database. The data layer should contain all the data you want to send to various analytics and/or marketing tags.

Advantage of using a data layer

In the context of Google Tag Manager, data layers are used to collect data from a website.

If you treat the Google Tag Manager tool like a car, then its engine is the ***container tag*** and its skin (look and feel) and controls are the user interface.

The container tag provides all the functionality needed for the GTM tool, to run and deploy tags on your website.

The user interface makes it easy for you, as an end-user, to control the container tag. Just like, when you drive a car, the car steering, makes it easy for you to control the car engine, make it, turn the car left or right.

When coders refer to GTM, they usually refer to the container tag.

When non-coders refer to GTM, they usually refer to the user interface.

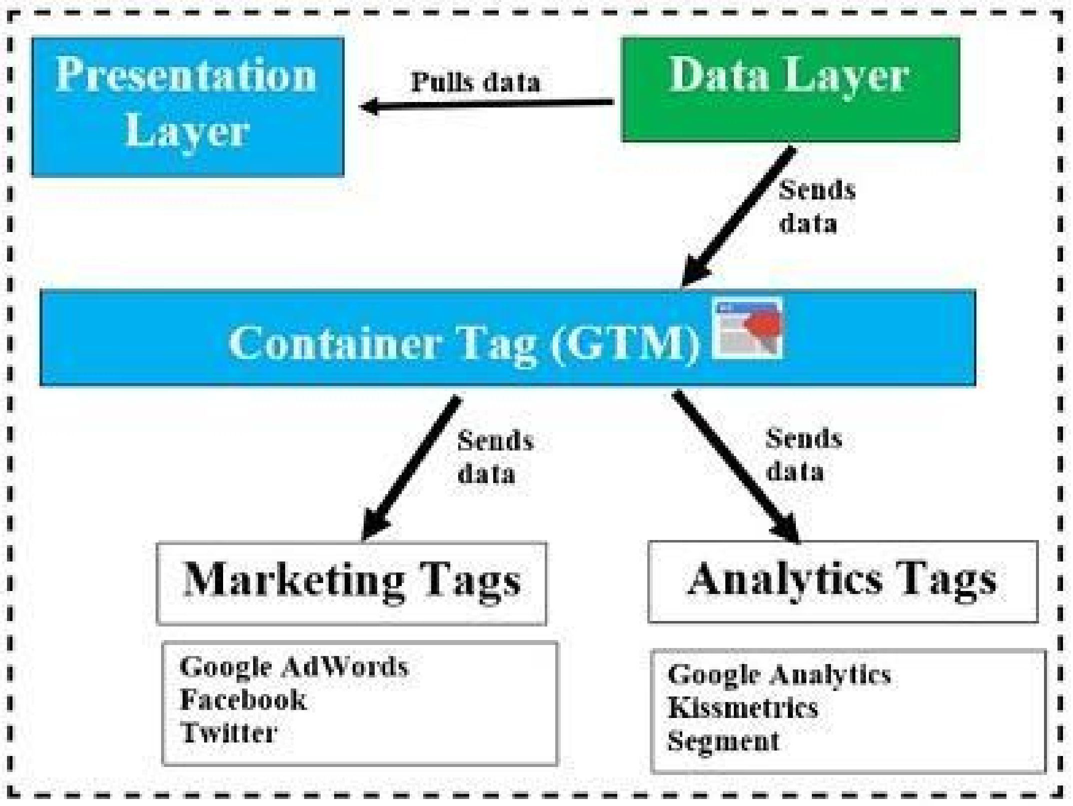
Thus depending upon the context, GTM can either mean the 'container tag' or the 'user interface'.

A container tag can be used to directly pull data from the HTML of a web page by traversing the **HTML DOM** (logical structure of an HTML document) using JavaScript.

However, when you use the container tag to pull data directly from the HTML of a web page and push it into your analytics and marketing tags, you create an unreliable tracking setup.

The problem with this approach is that the structure of HTML (**HTML DOM**) can change any time without your knowledge, as commercial websites update all the time and as a result, any tag can stop working any day, any time without any prior notice.

To fix the problem, we create and use data layers, which stores all the information we want to collect about a web page.



Once you set up a data layer, the container tag can be used to pull data from the data layer of a page instead of its HTML.

So no matter what happens to HTML DOM, the data layer will remain the same and your tags will continue to work (unless of course, someone/ something breaks the data layer itself).

When you can use DOM scraping instead of data layers?

Each tag management solution (TMS) provider, implement the data layer in a slightly different way. So most likely, you won't be able to use the same data layer with every TMS (if you use more than one TMS).

Data layers should be hard-coded on a website by your developer or IT team. So as long as you are using data layers, you may need to depend upon IT for its updates and maintenance.

***DOM scraping* is a technique used to pull data directly from the HTML elements (buttons, links, forms, etc) of a web page by using JavaScript and the knowledge of HTML DOM.**

Almost all TMS vendors provide the functionality of scraping the HTML DOM. DOM scraping comes handy when it is not possible to implement the required data layers in a reasonable amount of time, in order to set up certain tracking.

Through DOM scraping you can implement TMS solutions faster, as there is little to no dependence on IT and you have more control over the implementation.

However, the TMS solution implemented through DOM scraping is not a stable solution, as changes in the HTML DOM (like change in the ID of an HTML element) can quickly break the TMS solution.

So TMS solution deployment through DOM scraping is more of a quick fix when you lack development resources or time. It is not something, you can/should rely upon for a long period of time and/or where a lot of money is at stake (like setting up tracking for an airline website).

What information should you store in a data layer?

If you want to avoid the hassle of modifying data layers every time you deploy a new GTM solution then you should aim to hard-code one **universal data layer** on each web page of your website. Each universal data layer should include all the key attributes of the web page, on which it is embedded.

Your data layer should include all the key information your marketing and analytics tags currently need. It should also include all the key information you may need in the near future, for additional tracking.

This information could be:

#1 Page attributes

- page title
- page URL
- page category etc

#2 Product(s)' attributes - attributes of a product(s) listed on a web page like:

- product name
- product id

- product price
- product category
- product image URL
- product variant (color, size, dimension) selected
- product quantity selected
- product views
- product clicks (clicks on a product listing on a category page)
- product availability (available, out of stock)
- product coupon
- product coupon used

#3 Users' attributes

- User ID (login ID)
- Client ID
- User type (logged-in user, logged-out user, member, non-member, customer, non-customer, repeat customer, high-value repeat customer, etc)
- User's browsing behavior
- User's preferences
- User's purchase history
- Users' interactions (clicking a button, submitting forms, signups, purchase, watching videos, etc)
- Users' web browser
- Users' operating system
- Users' device (mobile, desktop, tablet)

Any information that might be useful either today or in the near future for additional tracking, should be included in the universal data layer.

What is a data layer in the context of Google Tag Manager?

A data layer is a JavaScript array of a single object which is used:

# To store all the key attributes of a web page like page title, page URL, page category, etc.

# To store all the key information your marketing and analytics tags currently need like user ID, client ID, user's preferences, users' purchase history, product ID, product price, etc.

# To store all the key information you may need in the near future, for additional tracking.

# To send information from a website to the GTM container tag.

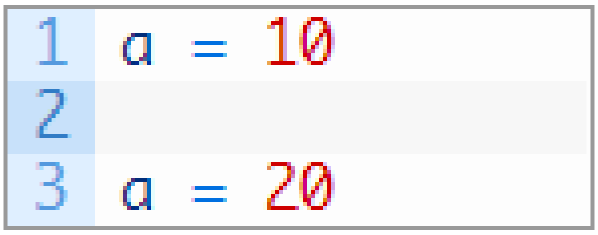
In order to understand the concept of a data layer, you would first need to understand JavaScript arrays and objects.

Introduction to Arrays and Variables

An array is a special variable that can store multiple elements, of the same or different data types, at a time.

Variable is a location in the computer memory which is used for storing data. A variable can store different data at different times.

For example, if 'a' is the name of a variable, then 'a' can be used to store a value like '10' and then at a different point in time, can be used to store another value like '20'.



```
a = 10

a = 20
```

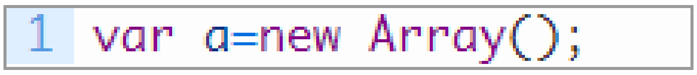
However, a variable can store only one value at a time. If you want a variable to store more than one value at a time, then you need to use a special variable called '**array**'.

An 'array' is made up of one or more elements. These elements can be strings (like 'hello'), numeric values (like 10), undefined values, boolean values (like true, false), other arrays or objects.

Creating an Array Variable via array function

An array can be created, by either using the **array function** or **array literal notation []**.

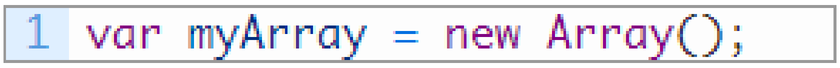
For example, here is how the empty array created through array function will look like in JavaScript:



```
var a=new Array();
```

Here 'a' is the name of the array and 'var' is a command which is used to create a variable in JavaScript. You can give any name to your array.

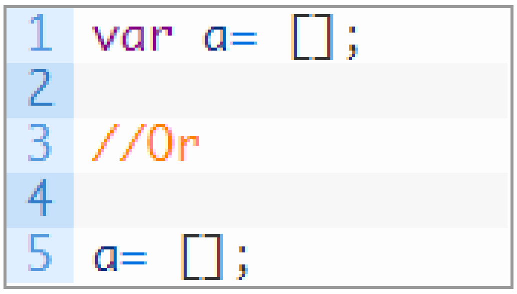
For example:



```
var myArray = new Array();
```

Creating an Array Variable via array literal notation

The array literal notation method is preferred, for creating an array. For example:



```
var a= [];

//Or

a= [];
```

Here a =[] is an **empty array**.

Following is an example of an array variable which contains one element of type number:

a = [10];

Following is an example of an array variable which contains two elements of type number:

a = [10, 20];

Following is an example of an array variable which contains one element of type string:

a = ["Hello"];

Following is an example of an array variable which contains three elements of type string:

a=["hi","bye","goodbye"];

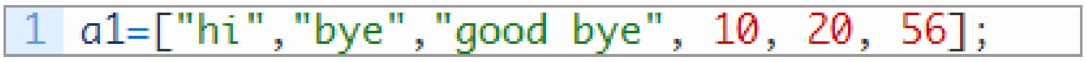
Following is an example of an array variable which contains one element of type number and one element of type string:

a = [100, "Hello"];

Arrays which contains elements of different data types

An array variable can be used to store elements of different data types.

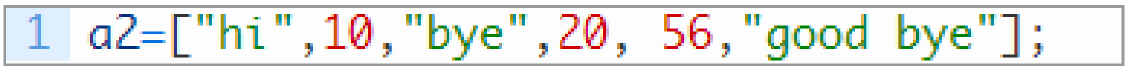
For example, you can create an array which contains both string and number values:



```
a1=["hi","bye","good bye", 10, 20, 56];
```

Here, the array variable named 'a1' is made up of six elements. Out of these six elements, the first three elements are of type string and the last three elements are of type 'number'.

You can also create an array, like the one below:



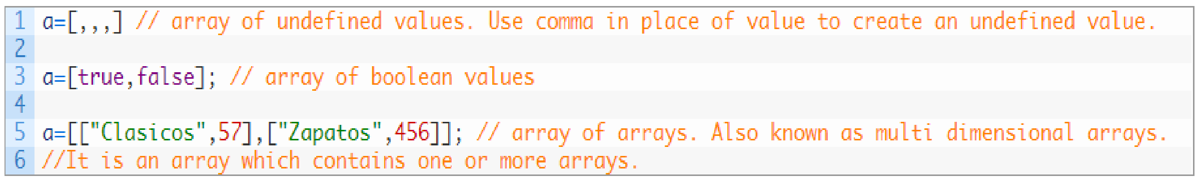
```
a2=["hi",10,"bye",20, 56,"good bye"];
```

Here, the array variable named 'a2' is made up of 6 elements.

Out of these six elements, three elements are of type string and the remaining three elements are of type 'number'.

Note: Even when both a1 and a2 contain the same elements, they are still different array variables, because the order of the elements is different.

Different Types of Array variables



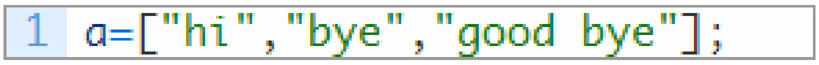
```
a=[,,,] // array of undefined values. Use comma in place of value to create an undefined value.

a=[true,false]; // array of boolean values

a=[["Clasicos",57],["Zapatos",456]]; // array of arrays. Also known as multi dimensional arrays.
//It is an array which contains one or more arrays.
```

Following are the different types of array variables:

- Array of Strings
- Array of numbers
- Array of undefined values
- Array of boolean values
- Array of objects
- Array of array

Introduction to an array of Strings

Following is an example of an array of strings:



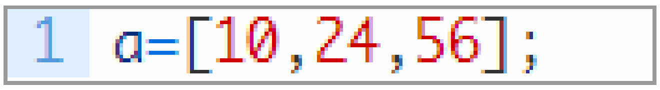
```
a=["hi","bye","good bye"];
```

Here the array variable named 'a' has got three elements named: 'hi', 'bye' and 'goodbye'.

All of these three elements are of type 'string' as they are string values. We call such arrays an **array of strings**.

Introduction to an array of numbers

Following is an example of an array of numbers:



```
a=[10,24,56];
```

Here, all of these three elements have got numeric values. We call such arrays an **array of numbers**.

Array of undefined values

Following is an example of an array of undefined values:

a=[,,,]; // array of undefined values.

We use a comma in place of value to create an undefined value.

Array of boolean values

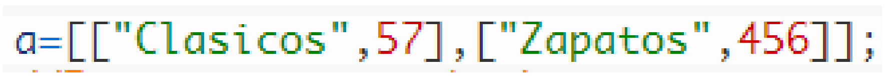
Following is an example of an array of boolean values:

a=[true,false]; // array of boolean values

Array of array (or Multi-Dimensional Array)

A multi-dimensional array is an array that contains one or more arrays.

For example:



```
a=[["Clasicos",57],["Zapatos",456]];
```

Here the array named ‘a’ contains the following two elements of type 'array':

["Clasicos",57]

["Zapatos",456]

Accessing the elements of an array

You can access an array element by referring to its index number. An Array index number starts with zero.

For example, consider the following array:

a=[10,24,56];

Here,

a[0] refers to the first element of the array i.e. 10

a[1] refers to the second element of the array i.e. 24

a[2] refers to the third element of the array i.e. 56

Introduction to Objects

Just like an array, an object is also a special variable that can store multiple elements, of the same or different data types, at a time.

However, unlike an array, an object's element/property is in the format ***name:value.***

Here, 'name' is the property name and 'value' is the property value.

Creating an object variable via object literal notation

The object literal notation is the preferred method, for creating an object. For example:

```
var a= {};


//Or


a= {};
```

Here a ={} is an **empty object**.

Note: You can give any name to the JavaScript object.

Following is an example of an object variable which contains one element:

```
        User = {"firstName":"John"};
```

Following is an example of an object which contains two elements:

```
        User = {"firstName":"John", "lastName":"Marshall"};
```

Following is an example of an object which contains four elements:

User = {"firstName":"John", "lastName":"Marshall", "age":85, "eyeColor":"blue"};

__**Following is the syntax to create a JavaScript object:**__

<object-name>={"property-name1":property-value1, "property-name2":property-value2, ..."property-nameN":property-valueN};

Here property-name can be any string or identifier.

An identifier is a reserved word, which carries a special meaning and can not be used as a variable name. For example 'event' is an identifier.

A property's value can be a:

1. string
2. numeric value
3. undefined value
4. boolean value
5. array
6. multi-dimensional array
7. object
8. an array of objects.

Improving the readability of an object

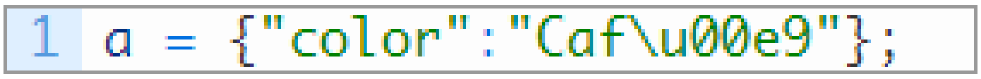
The following example code creates an object called "user" and adds 4 properties to it:

```
user={"firstName":"John", "lastName":"Marshall", "age":85, "eyeColor":"blue"};
```

White spaces and line breaks are not important in JavaScript. So to improve readability you can write the same 'user' object as:

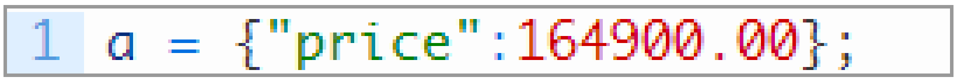
```
 user={

"firstName":"John",

"lastName":"Marshall",

"age":45,

"eyeColor":"blue"

};
```

Other examples of objects



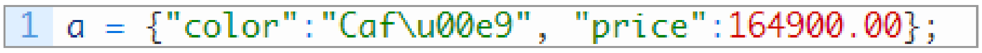
```
a = {"color":"Caf\u00e9"};
```

Here *'color*' is the name of the property of the object 'a' and '*Caf\u00e9*' is the value of the 'color' property which is a string.



```
a = {"price":164900.00};
```

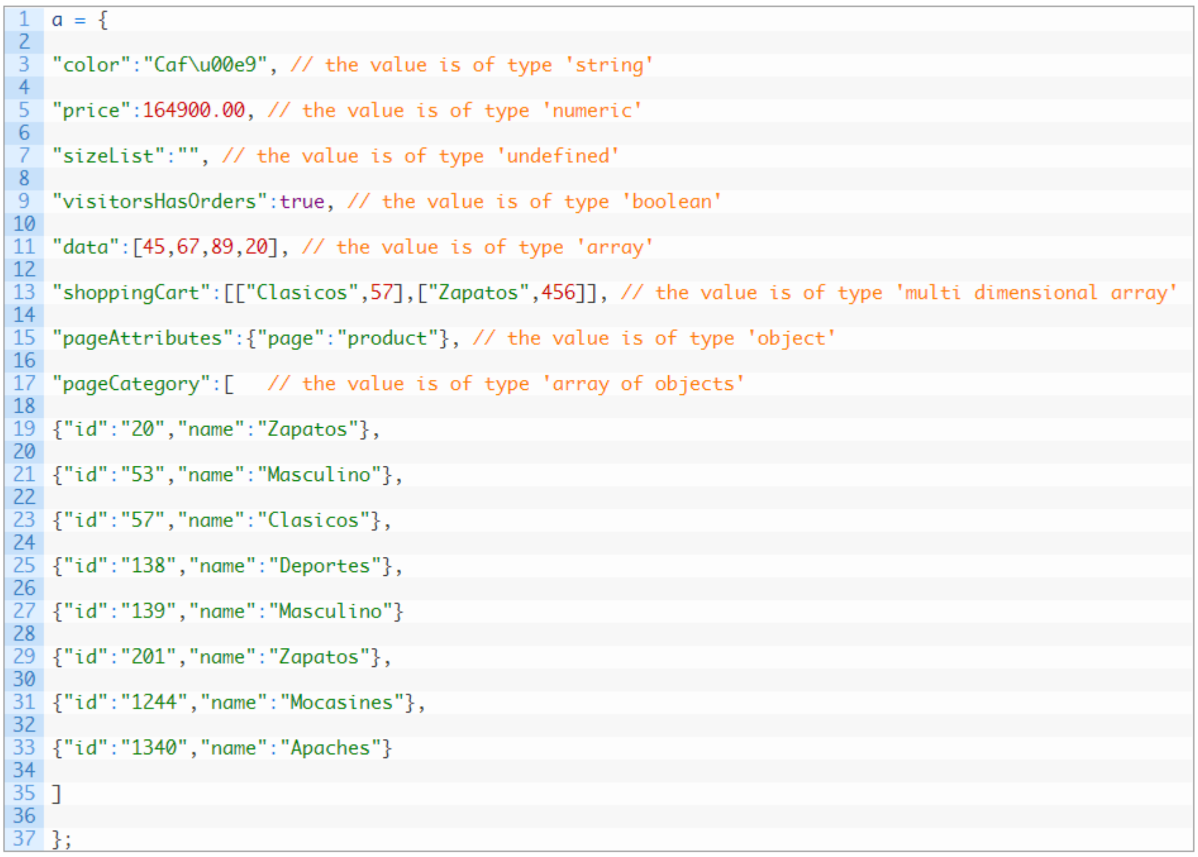
Here '*price*' is the name of the property of the object 'a' and '*164900.00*' is the value of the 'price' property which is a numerical value.



```
a = {"color":"Caf\u00e9", "price":164900.00};
```

Here the object 'a' has got two properties 'color' and 'price'.

Similarly, we can define an object with multiple properties:



```
a = {

"color":"Caf\u00e9", // the value is of type 'string'

"price":164900.00, // the value is of type 'numeric'

"sizeList":"", // the value is of type 'undefined'

"visitorsHasOrders":true, // the value is of type 'boolean'

"data":[45,67,89,20], // the value is of type 'array'

"shoppingCart":[["Clasicos",57],["Zapatos",456]], // the value is of type 'multi dimensional array'

"pageAttributes":{"page":"product"}, // the value is of type 'object'

"pageCategory":[   // the value is of type 'array of objects'

{"id":"20","name":"Zapatos"},

{"id":"53","name":"Masculino"},

{"id":"57","name":"Clasicos"},

{"id":"138","name":"Deportes"},

{"id":"139","name":"Masculino"}

{"id":"201","name":"Zapatos"},

{"id":"1244","name":"Mocasines"},

{"id":"1340","name":"Apaches"}

]

};
```

Accessing the elements of an object

You can access an object element/property by referring to its property name.

There are two methods to access object properties:

objectName.propertyName

or

objectName["propertyName"]

For example, consider the following object:

User = {"firstName":"John", "lastName":"Marshall"};

Here,

User.firstName refers to the first element of the object.

User.lastName refers to the second element of the object.

Similarly,

User["firstName"] refers to the first element of the object.

User ["lastName"] refers to the second element of the object.

Difference between Arrays and Objects

Following is an example of an array that has the following three properties/elements: "Amit", "Sinha" and 35:

a = ["Amit", "Sinha", 35];

Since we use numbers to access array elements, we can access the first property of the array 'a' by the notation:

a[0];

We can re-write the array 'a' as the following object:

b = {"FirstName": "Amit", "LastName":"Sinha", "Age": 35};

Here 'b' is an object that has the following three properties/elements:

1. "FirstName": "Amit"
2. "LastName":"Sinha"
3. "Age": 35

Since we use property names to access object elements, we can access the first property of the object 'b' by the notation:

b.FirstName;

So while both array and objects are special variables that can store multiple elements (of the same or different data types) they store data in a different format and are accessed differently.

In JavaScript, arrays use numbered indexes whereas objects use named indexes.

Array of objects

An array variable can also be used to store one or more objects.

Following is an example of an array variable which contains only one object:

```
a=[{"id":"20","name":"Zapatos"}];
```

Here the object is: {"id":"20","name":"Zapatos"}

Following is an example of an array variable which contains two objects:

```
a=[{"id":"20","name":"Zapatos"},{"id":"53","name":"Masculino"}];
```

Here the objects are:

- {"id":"20","name":"Zapatos"}
- {"id":"53","name":"Masculino"}

White spaces and line breaks are not important in JavaScript. So to improve readability you can write the array named 'a' as:

```
a=[

{"id":"20","name":"Zapatos"},

{"id":"53","name":"Masculino"},

];
```

Following is an example of an array variable which contains multiple elements of type 'object':

```
a=[

{"id":"20","name":"Zapatos"},

{"id":"53","name":"Masculino"},

{"id":"57","name":"Clasicos"},

{"id":"138","name":"Deportes"},

{"id":"139","name":"Masculino"}

{"id":"201","name":"Zapatos"},

{"id":"1244","name":"Mocasines"},

{"id":"1340","name":"Apaches"}

];
```

An array of all available data types

You can also create an array which contains elements of all available data types:

```
a=["size", 10, true,,["Clasicos",57],{"id":"20","name":"Zapatos"}];
```

Here,

"size" is an array element of type string.

10 is an array element of type number.

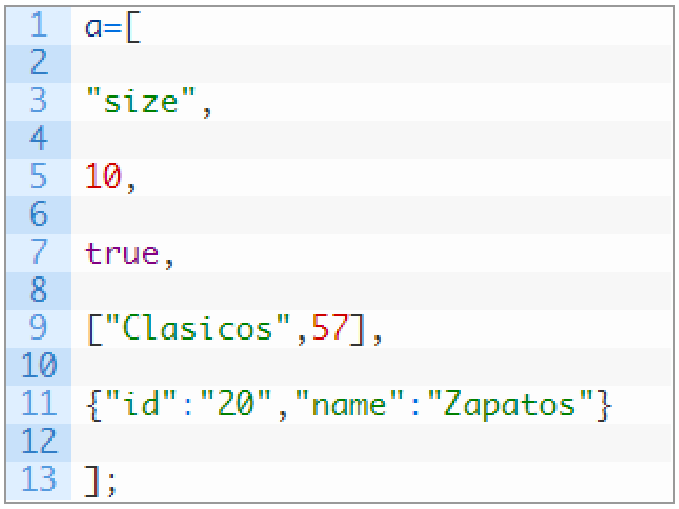
true is an array element of type boolean.

,, (empty value) is an array element of type undefined.

["Clasicos",57] is an array element of type array.

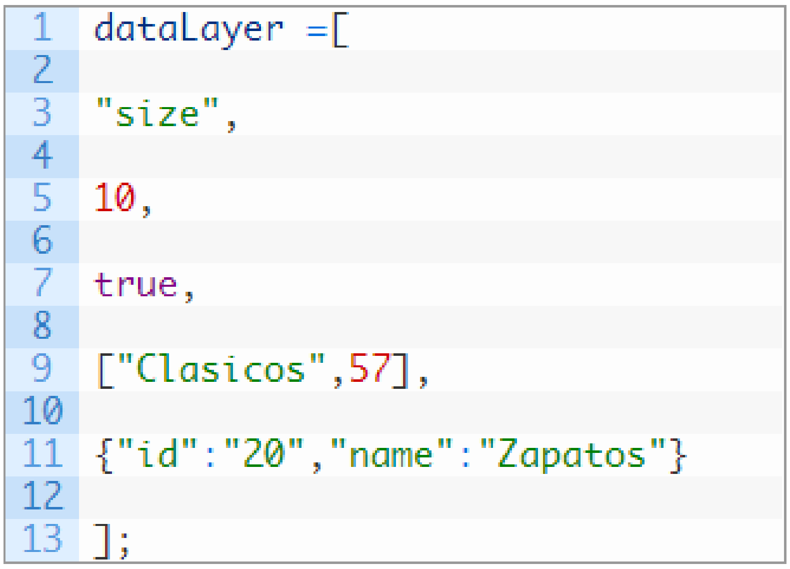
{"id":"20","name":"Zapatos"} is an array element of type object.

White spaces and line breaks are not important in JavaScript. So to improve readability you can write the same 'a' array as:



```
a=[

"size",

10,

true,

["Clasicos",57],

{"id":"20","name":"Zapatos"}

];
```

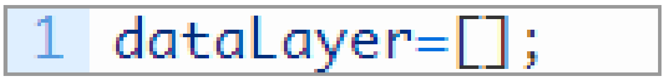
Since dataLayer is a JavaScript array, you can create a data layer like the one below:



```
dataLayer =[

"size",

10,

true,

["Clasicos",57],

{"id":"20","name":"Zapatos"}

];
```

Google Tag Manager uses the array variable name called 'dataLayer'

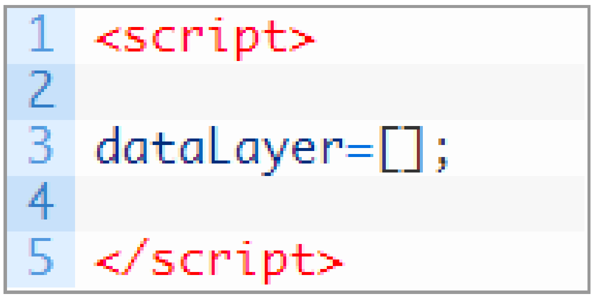
So if you want to create an empty data layer array, it would look like the one below:



```
dataLayer=[];
```

Since this line of code is a JavaScript statement, you need to enclose it within the <script> and </script> tags, so that you can embed the code into your HTML document.

So the final setup will look like the one below:



```
<script>

dataLayer=[];

</script>
```

Congratulations, you just created your first data layer.

Note: A data layer is also known as **dataLayer queue**, **dataLayer object,** and **data collection layer**.

Data layer is an array of a single object

In the context of Google Tag Manager, a data layer is usually an array of a single object:

dataLayer=[{}];

Since this line of code is a JavaScript statement, you need to enclose it within the <script> and </script> tags, so that you can embed the code into your HTML document.

So the final setup will look like the one below:

```
<script>
dataLayer=[{}];
</script>
```

<u>Following is an example of a data layer which contains one element:</u>

```
dataLayer = [{ 'pageCategory': 'Statistics' }];
```

<u>Following is an example of a data layer which contains two elements:</u>

```
dataLayer = [{ 'pageCategory': 'Statistics', 'visitorType': 'high-value' }];
```

<u>Following is an example of a data layer which contains more than two elements:</u>

```
 dataLayer=[ {

"color":"Caf\u00e9",

"price":164900.00,

"sizeList":"",

"visitorsHasOrders":true,

"data":[45,67,89,20],

"shoppingCart":[["Clasicos",57],["Zapatos",456]],

"pageAttributes":{"page":"product"},

"pageCategory":[

{"id":"20","name":"Zapatos"},

{"id":"53","name":"Masculino"},

{"id":"57","name":"Clasicos"},

{"id":"138","name":"Deportes"},

{"id":"139","name":"Masculino"}

{"id":"201","name":"Zapatos"},

{"id":"1244","name":"Mocasines"},

{"id":"1340","name":"Apaches"}

]

}];
```

Congratulations, you have just created your first **complex data layer.**

Quick Recap

Remember:

a=[]; // creates an array

a={}; // creates an object

a=[{}]; // creates an array of a single object

dataLayer in GTM is an array of a single object.

So,

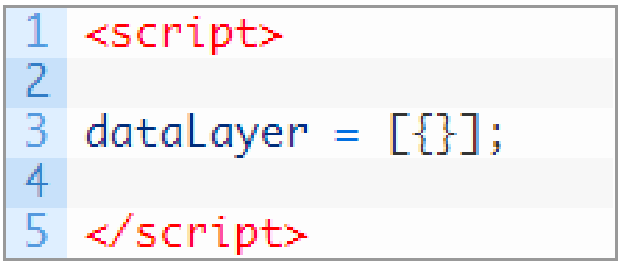
dataLayer = [{...}];

What are data layer variables?

These are the variables used in the data layer.

It is in the format: **'variable_name': 'variable_value'**.

<u>Following is an example of a data layer which contains no variable:</u>



```
<script>

dataLayer = [{}];

</script>
```

<u>Following is an example of a data layer which contains one variable:</u>

```
<script>

dataLayer = [{'pageCategory': 'Statistics'}];

</script>
```

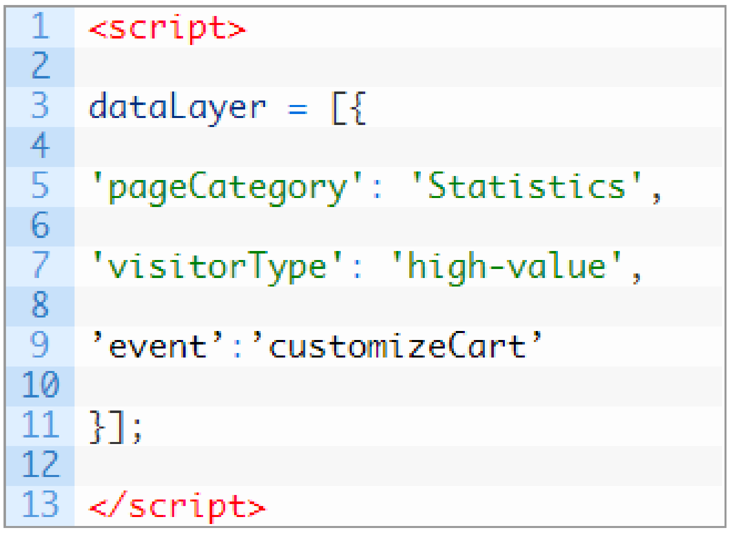
Following is an example of a data layer which contains two variables:

```
<script>

dataLayer = [{'pageCategory': 'Statistics', 'visitorType': 'high-value'}];

</script>
```

Since white spaces and line breaks are not important in JavaScript, so to improve readability you can re-write the data layer as:

```
<script>

dataLayer = [{

'pageCategory': 'Statistics',

'visitorType': 'high-value'

}];

</script>
```

Following is an example of a data layer which contains three variables:



```
<script>

dataLayer = [{

'pageCategory': 'Statistics',

'visitorType': 'high-value',

’event’:’customizeCart’

}];

</script>
```

Here, *PageCategory*, *visitorType,* and *event* are the names of the data layer variables.

The value of *PageCategory is ‘Statistics*’.

The value of *visitorType is ‘High Value*’.

The value of the *event* is *customizeCart*

Note: Data layer variables are also known as ***data layer messages***.

How you can initialize a data layer?

By setting up variables in your data layer.

Initializing a data layer with one variable:

```
<script>

dataLayer = [{'pageCategory': 'Statistics'}];

</script>
```

Initializing a data layer with two variables:

```
<script>

dataLayer = [{'pageCategory': 'Statistics', 'visitorType': 'high-value'}];

</script>
```

Initializing a data layer with three variables:

```
<script>

dataLayer = [{

'pageCategory': 'Statistics',

'visitorType': 'high-value',

’event’:’customizeCart’

}];

</script>
```

Appending/passing messages onto the data layer queue => Adding data layer variables to the data layer.

Updating data layer => adding/updating/removing data layer variables.

Pushing information into the data layer => adding data layer variables to the data layer.

How information is pushed into a data layer?

<u>There are two methods that can be used to push information into a data layer:</u>

#1 By hard coding the data layer variables in the data layer, preferably above the container tag.

#2 By dynamically adding objects to your data layer through the ‘push’ method of dataLayer object.

Use the first method, if you want to push information into a data layer on page load.

For example, for ecommerce tracking or enhanced ecommerce tracking we use the first method, as we want the ecommerce data to be available on page load.

Use the second method, if you want to push information into a data layer on any event other than the page load.

For example, if you want to push information into the data layer on click on ‘Add to Cart’ button then use the second method.

The ‘push’ method of dataLayer object

Through the ‘push’ method of dataLayer object, you can dynamically add object(s) to your data layer. The pushed object can contain one or more data layer variables.

Following is the syntax of the 'push' method:

```
dataLayer.push({'variable_name': 'variable_value'});
```

For example, let us suppose the following data layer is hard-coded on a web page:

```
<script>

dataLayer = [{'pageCategory': 'Statistics'}];

</script>
```

Now, this data layer already contains one variable.

If you used the following push method:

```
<script>

dataLayer.push({'visitorType': 'high-value'});

</script>
```

The hard coded data layer will now look like this:

```
<script>

dataLayer = [{'pageCategory': 'Statistics','visitorType': 'high-value'}];

</script>
```

You can also push/add multiple variables at once, into the data layer.

For example:

```
<script>

dataLayer.push({'pageCategory': 'Statistics','visitorType': 'high-value'});

</script>
```

Use 'window.dataLayer.push()'

This is the same as the *dataLayer.push()* method but with an added advantage.

'dataLayer' is a **global JavaScript array variable** and can thus be accessed by any function that can also access the window object.

The advantage of using the ‘window’ prefix is that you can avoid conflict with any local JavaScript array variable that uses the ‘dataLayer’ name.

So instead of using:

dataLayer.push({'variable_name': 'variable_value'});

Use

window.dataLayer.push({'variable_name': 'variable_value'});

Example-1:

```
<script>

window.dataLayer.push({'visitorType': 'high-value'});

</script>
```

Example-2:

```
<script>

window.dataLayer.push({'pageCategory': 'Statistics','visitorType':
'high-value'});

</script>
```

About ‘Scope’ in JavaScript

In JavaScript, there are two types of scope: **local scope** and **global scope**.

Scope determines the accessibility of JavaScript variables.

A variable declared within a JavaScript function is called a **local variable**. A local variable has got a local scope and can only be accessed within the function where it is declared.

A variable declared outside any function is called a **global variable**. A global variable has got a global scope and can be accessed by any function.

Note: In JavaScript, objects and functions are also called as variables.

Multiple initializations of a data layer leads to overwriting of the data layer

Let us suppose following data layer is hardcoded in the head section (<head>....</head> of a web page

```
<script>

dataLayer = [{'pageCategory': 'Statistics'}];

</script>
```

Let us suppose, you later, hardcoded second data layer somewhere in the body section (<body>....</body>) of the web page:

```
<script>

dataLayer = [{'visitorType': 'high-value'}];

</script>
```

Now what will happen on page load, is that only the data from the second data layer will be available to the GTM container tag. The second initialization of the data layer will overwrite the first one.

So if you want to access the ‘pageCategory’ data layer variable from within GTM, you won’t be able to. Because on page load, the data layer will look like this:

```
<script>

dataLayer = [{'visitorType': 'high-value'}];

</script>
```

Instead of:

```
<script>

dataLayer = [{

'pageCategory': 'Statistics',

'visitorType': 'high-value'
}];

</script>
```

In order to avoid this issue, **do not initialize a data layer more than once on the same webpage.**

If you want to push new information to the existing data layer then use the 'window.push()' method.

For example,

```
<script>

dataLayer = [{'pageCategory': 'Statistics'}];

</script>

.

.

<script>

window.dataLayer.push({'visitorType': 'high-value'});

</script>
```

Now the final data layer after the push will look like the one below:

```
<script>
```

```
    dataLayer = [{
    'pageCategory': 'Statistics',
    'visitorType': 'high-value'
    }];
</script>
```

Push information into a data layer only if the ‘dataLayer’ variable has already been declared

In order to avoid multiple initializations of a data layer and consequently overwriting the data layer with new information, always check whether the dataLayer variable has already been declared.

If the dataLayer variable has already been declared, only then proceed to push new information. You can achieve this objective by using the following code:

```
    <script>
    window.dataLayer = window.dataLayer;
    window.dataLayer.push({'variable_name': 'variable_value'});
```

```
</script>
```

If the dataLayer variable has not already been declared then assign a new empty array variable to it and only then proceed to push new information.

You can achieve this objective by using the following code:

```
<script>

window.dataLayer = [];

window.dataLayer.push({'variable_name': 'variable_value'});

</script>
```

In order to achieve both of the aforementioned objectives at the same time, combine the code like the one below:

```
<script>

window.dataLayer = window.dataLayer || [];

window.dataLayer.push({'variable_name': 'variable_value'});

</script>
```

For example:

```
<script>

dataLayer = [{''pageCategory': 'Statistics'}];

</script>
```

```
.

.

<script>

window.dataLayer = window.dataLayer || [];

window.dataLayer.push({'visitorType': 'high-value'});

</script>
```

Now the final data layer, will look like the one below:

```
<script>

dataLayer = [{

'pageCategory': 'Statistics',

'visitorType': 'high-value'

}];

</script>
```

Overwriting the value of an existing data layer variable

By pushing a variable of the same name as an existing variable, but with a new value to the data layer.

Let us suppose this is your existing data layer:

```
<script>

dataLayer = [{'pageCategory': 'Statistics','visitorType': 'high-value'}];

</script>
```

Now if you used the push method, like the one below:

```
<script>

dataLayer.push({'pageCategory': 'Maths'});

</script>
```

The value of pageCategory data layer variable would be overwritten and now your data layer, would like the one below:

```
<script>

dataLayer = [{'pageCategory': 'Maths','visitorType': 'high-value'}];

</script>
```

Data layer for ecommerce tracking (ecommerce data layer variables)

The data layer which contains ecommerce data layer variables is called the **ecommerce data layer.**

Following are the examples of ecommerce data layer variables which are recognized by GTM:

- ***currencyCode***
- ***impressions***
- ***ecommerce***
- ***click***
- ***actionField***
- ***products***
- ***detail***
- ***add***
- ***remove***
- ***promoView***
- ***promotions***
- ***promoClick***
- ***checkout***
- ***checkout_option***
- ***purchase***
- ***refund***

- ***id***
- ***affiliation***
- ***revenue etc***

Note: All the ecommerce data layer variables are reserved words (identifiers). They all carry a special meaning and therefore should not be used as a regular variable name in data layers while using GTM.

We use ecommerce data layers when we need to install ecommerce tracking or enhanced ecommerce tracking code on a website via GTM.

The ecommerce data layer pulls ecommerce data from your shopping cart (via a server-side script) and then sends the data to GTM.

Following is an example of ecommerce data layer:

```
<script>
// Send transaction data with a pageview if available
// when the page loads. Otherwise, use an event when the transaction
// data becomes available.
dataLayer.push({
  'ecommerce': {
    'purchase': {
      'actionField': {
        'id': 'T800',                      // Transaction ID. Required for purchases and
refunds.
        'affiliation': 'OptimizeSmart',
        'revenue': '998.00',              // Total transaction value (incl. tax and
shipping)
        'tax':'199.6',
        'shipping': '1.9',
        'coupon': 'WINTER_SALE'
      },
      'products': [{                     // List of productFieldObjects.
        'name': '62 points GA Checklist',    // Name or ID is required.
        'id': '62',
        'price': '18.55',
        'brand': 'Optimize Smart',
        'category': 'eBook',
        'variant': 'Blue',
        'quantity': 1,
        'coupon': ''                      // Optional fields may be omitted or set to
empty string.
        },
        {
        'name': '9 points social media checklist',
        'id': '9',
        'price': '19.75',
        'brand': 'Optimize Smart',
        'category': 'eBook',
        'variant': 'Green',
        'quantity': 1
        }]
    }
  }
});
</script>
```

Note: All the ecommerce data layer variables are highlighted in the code in bold text.

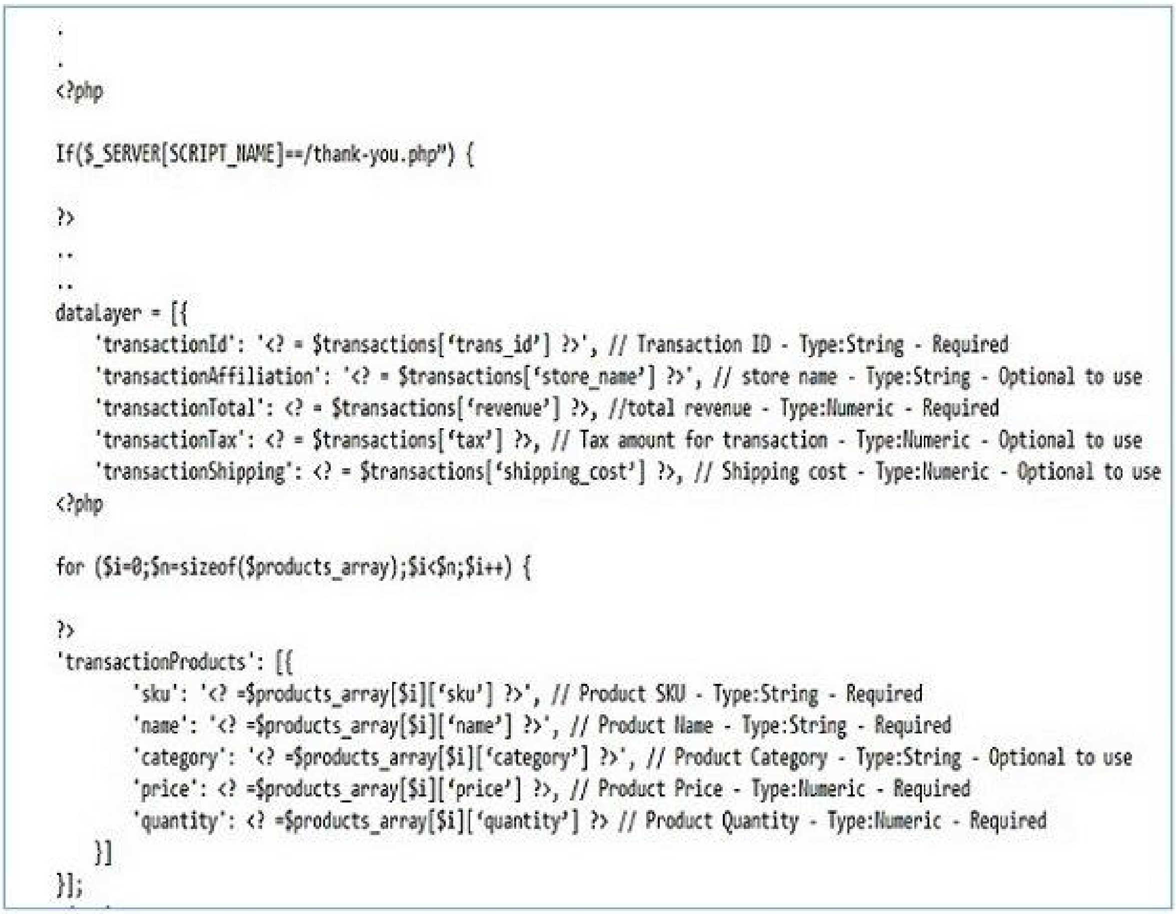
Ecommerce Data Layers contain Server Side code

The ecommerce data layer that you saw earlier and the ecommerce data layers that you see in the Google help documentation are all **example codes**.

You can't just copy-paste them on your website and then expect the ecommerce tracking to work.

To make your ecommerce data layers to actually work and collect ecommerce data, you would need to add a server-side script (like PHP) to it.

For example, here is how a data layer with a server-side script, may look like:



```
.
.
<?php

If($_SERVER[SCRIPT_NAME]==/thank-you.php") {

?>
..
..
dataLayer = [{
    'transactionId': '<? = $transactions[‘trans_id’] ?>', // Transaction ID - Type:String - Required
    'transactionAffiliation': '<? = $transactions[‘store_name’] ?>', // store name - Type:String - Optional to use
    'transactionTotal': <? = $transactions[‘revenue’] ?>, //total revenue - Type:Numeric - Required
    'transactionTax': <? = $transactions[‘tax’] ?>, // Tax amount for transaction - Type:Numeric - Optional to use
    'transactionShipping': <? = $transactions[‘shipping_cost’] ?>, // Shipping cost - Type:Numeric - Optional to use
<?php

for ($i=0;$n=sizeof($products_array);$i<$n;$i++) {

?>
'transactionProducts': [{
        'sku': '<? =$products_array[$i][‘sku’] ?>', // Product SKU - Type:String - Required
        'name': '<? =$products_array[$i][‘name’] ?>', // Product Name - Type:String - Required
        'category': '<? =$products_array[$i][‘category’] ?>', // Product Category - Type:String - Optional to use
        'price': <? =$products_array[$i][‘price’] ?>, // Product Price - Type:Numeric - Required
        'quantity': <? =$products_array[$i][‘quantity’] ?> // Product Quantity - Type:Numeric - Required
    }]
}];
```

This is what a hardcoded ecommerce data layer actually looks like. You would need the help of a web developer to create ecommerce data layers.

What should be the format of the information you push into a data layer?

The information that you push into a data layer, from either front-end or back-end must be in the format, GTM can understand.

For example, if you try to push **ecommerce data layer variables** into a data layer which is not recognized/recommended by GTM, then your ecommerce tracking won't work.

```
<script>
dataLayer = [{
.
.
'Transaction Id': '1234',
...
}]
}];
</script>
```

Here GTM won't be able to recognize 'Transaction Id' as **ecommerce data layer variable**. The correct data layer snippet would be:

```
<script>
dataLayer = [{
.
.
'id': '1234',
...
}]
}];
```

```
        </script>
```

Note: 'id' is an example of an identifier (reserved word). So it carries a special meaning and should not be used as a regular data layer variable name while using GTM.

Naming conventions for data layer variables

You should use a consistent naming convention for data layer variables.

If you use different variable names that refer to the same variable on different pages, GTM may not be able to fire tags in all the desired locations.

For example,

If you set up the page category for the 'signup' page by using the '*pageCategory*' data layer variable, then the page category for the product purchase page should also be set up by using the '*pageCategory*' data layer variable and not by using data layer variables like '*Pagecategory'* or '*PageCategory*'.

That is because **data layer variable names are case sensitive.**

So the following set up won't work the way, you think it should:

```
// signup page:

dataLayer.push({'pageCategory': 'members-signup'});

// Product purchase Page:

dataLayer.push({'PageCategory': 'product-purchase'});
```

But the following set up will work the way it should:

```
// signup page:

dataLayer.push({'pageCategory': 'members-signup'});

// Product purchase Page:

dataLayer.push({'pageCategory': 'product-purchase'});
```

Data layer Variable names should be enclosed in quotes

```
dataLayer.push({pageCategory: 'members-signup'});     // Won't work

dataLayer.push({'pageCategory': 'members-signup'});     // works
```

The name 'Data Layer' itself is case-sensitive

So, '*dataLayer*' is different from '*datalayer*', and using the latter won't work.

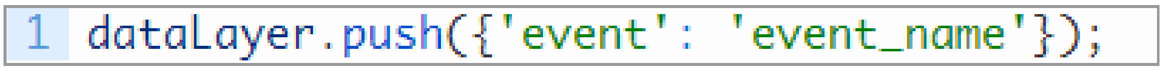
For example:

```
datalayer.push({'pagecategory': 'members-signup'});     // Won't work

dataLayer.push({'pageCategory': 'members-signup'});     // works
```

What is an 'event' data layer variable?

It is a special data layer variable that is used with JavaScript event listeners. It is used to fire a tag when a user interacts with webpage elements such as form, button, links, etc.

<u>The syntax for setting up an event:</u>



```
dataLayer.push({'event': 'event_name'});
```

Here ‘event’ and ‘event name’ are strings.

For example, the following code update the data layer’s ‘event’ variable when a user clicks on a button:

```
<a href="https://www.optimizesmart.com/download"
name="pdfDownload" onclick="window.dataLayer.push({'event':
'pdf-download'});" >PDF download</a>
```

<u>Other examples of special data layer variables</u>

- **hitType**
- **nonInteraction**

- **ecommerce** etc

FAQs

Q. Can each web page have its own unique data layer?

A. Yes, and it should.

Q. Do you need to put the same set of variables in the data layer on every page?

A. No.

Q. When a data layer is updated in case of a tracked event?

A. When the tracked event occurs like: mouse click, form submission, page load, etc.

What are dynamic data layer variables?

Dynamic data layer variables are data layer variables which are pushed dynamically (during run time) to a data layer.

The syntax for setting up dynamic data layer variables

```
dataLayer.push({'variable_name': 'variable_value'});
```

For example, let us suppose the following data layer is hard-coded on a web page:

```
<script>

dataLayer = [{'pageCategory': 'Statistics'}];

</script>
```

If you used the following push method:

```
<script>

dataLayer.push({'visitorType': 'high-value'});

</script>
```

The hard coded data layer will now look like the one below:

```
<script>

dataLayer = [{'pageCategory': 'Statistics','visitorType': 'high-value'}];

</script>
```

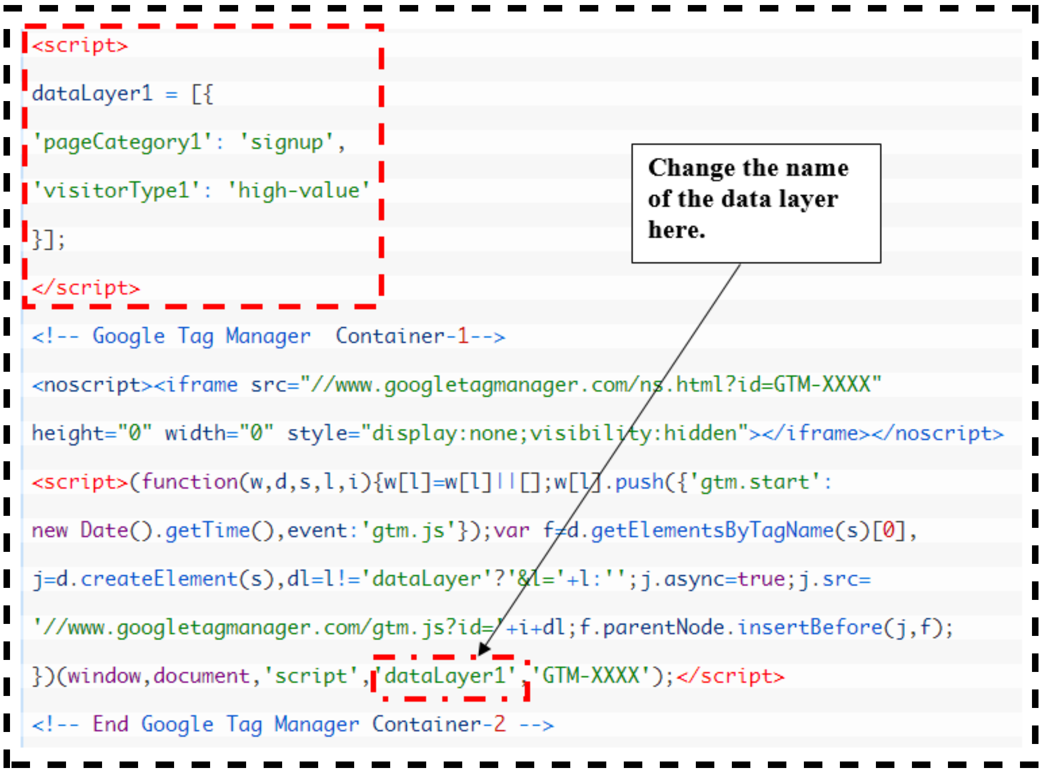
Here 'pageCategory' is a data layer variable whereas 'visitorType' is a dynamic data layer variable, as it has been pushed dynamically into the data layer during run time.

Note: Dynamic data layer variables are not hard-coded on a web page.

How you can rename a data layer (or data layer object)?

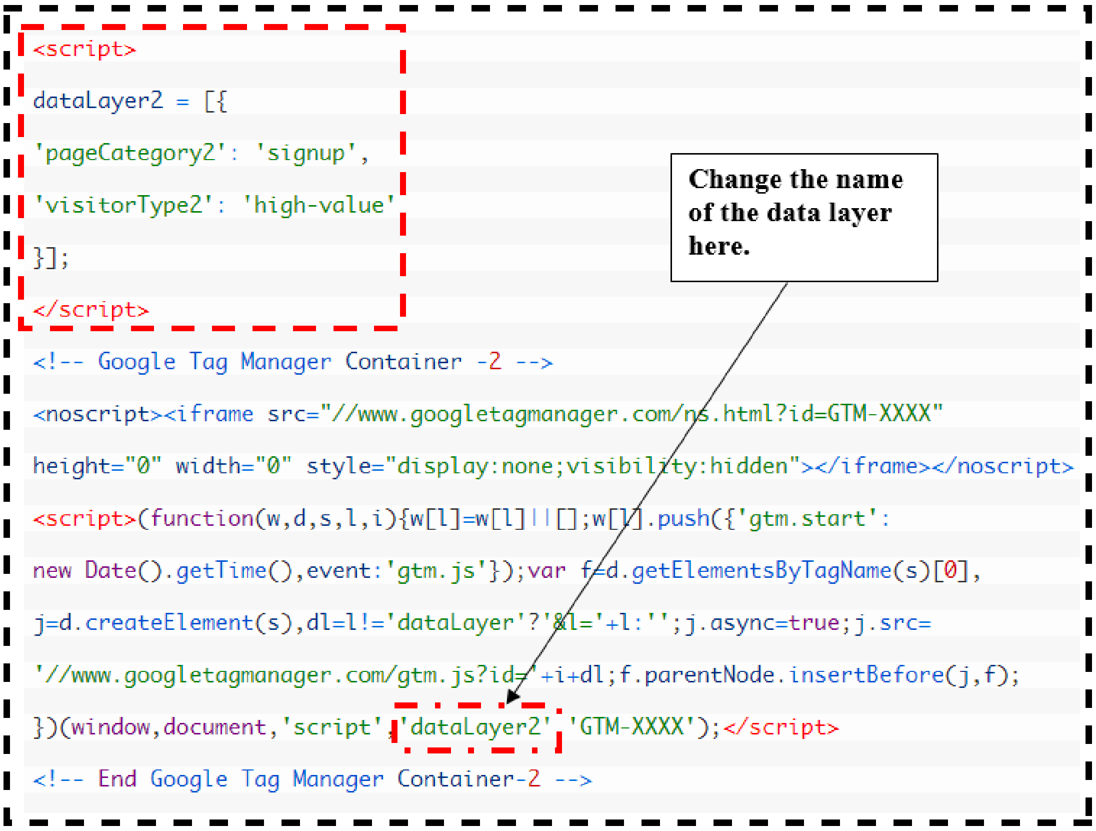
The default name of the data layer (or data layer object) is 'dataLayer'.

By making a small change (highlighted below in bold text) in your GTM container code snippet you can rename your data layer:



```
<script>
dataLayer1 = [{
'pageCategory1': 'signup',
'visitorType1': 'high-value'
}];
</script>
<!-- Google Tag Manager  Container-1-->
<noscript><iframe src="//www.googletagmanager.com/ns.html?id=GTM-XXXX"
height="0" width="0" style="display:none;visibility:hidden"></iframe></noscript>
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'//www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer1','GTM-XXXX');</script>
<!-- End Google Tag Manager Container-2 -->
```

Another example:



```
<script>
dataLayer2 = [{
'pageCategory2': 'signup',
'visitorType2': 'high-value'
}];
</script>
<!-- Google Tag Manager Container -2 -->
<noscript><iframe src="//www.googletagmanager.com/ns.html?id=GTM-XXXX"
height="0" width="0" style="display:none;visibility:hidden"></iframe></noscript>
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'//www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer2','GTM-XXXX');</script>
<!-- End Google Tag Manager Container-2 -->
```

Another example:

<!-- Google Tag Manager -->

<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':

new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],

j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=

'//www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);

})(window,document,'script','**addYourNewDataLayerNameHere**','GTM-XXXX');<

/script>

<!-- End Google Tag Manager -->

For example:

```
<body>
<script>
 SmartDataLayer = [{
'pageCategory': 'signup',
'visitorType': 'high-value'
}];
</script>
<!-- Google Tag Manager -->
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'//www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','SmartDataLayer','GTM-XXXX');</script>
<!-- End Google Tag Manager -->


<a href="https://www.optimizesmart.com/download" name="pdfDownload"
onclick="window.SmartDataLayer.push({'event': 'pdf-download'});" >PDF
download</a>
```

When do you need to rename your data layer?

You should rename your data layer when you are using multiple GTM container tags on the same web page and you don't want each GTM container tag to use the same data layer.

For example:

```
<body>
<script>
dataLayer1 = [{
'pageCategory1': 'signup',
'visitorType1': 'high-value'
}];
</script>

<!-- Google Tag Manager  Container-1-->
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'//www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer1','GTM-XXXX');</script>
<!-- End Google Tag Manager Container-2 -->

<!--=========================================================-->

<script>
dataLayer2 = [{
'pageCategory2': 'signup',
'visitorType2': 'high-value'
}];
</script>

<!-- Google Tag Manager Container -2 -->
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'//www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer2','GTM-XXXX');</script>
<!-- End Google Tag Manager Container-2 -->

<!--=========================================================-->

<script>
dataLayer3 = [{
'pageCategory3': 'signup',
'visitorType3': 'high-value'
}];
</script>

<!-- Google Tag Manager  Container-3 -->
<noscript><iframe src="//www.googletagmanager.com/ns.html?id=GTM-XXXX"
height="0" width="0"
style="display:none;visibility:hidden"></iframe></noscript>
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
```

```
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'//www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer3','GTM-XXXX');</script>
<!-- End Google Tag Manager Container-3 -->

<!--=========================================================-->


<a href="https://www.optimizesmart.com/download1" name="pdfDownload1"
onclick="window.dataLayer1.push({'event': 'pdf-download1'});" >PDF
download 1</a>

<a href="https://www.optimizesmart.com/download2" name="pdfDownload2"
onclick="window.dataLayer2.push({'event': 'pdf-download2'});" >PDF
download 2</a>

<a href="https://www.optimizesmart.com/download3" name="pdfDownload3"
onclick="window.dataLayer3.push({'event': 'pdf-download3'});" >PDF
download 3</a>
```

Note: By default, each GTM container tag uses a common data layer.

FAQ

Q. Do you need to be a web developer in order to use the data layer?

A. I am often asked this question. The answer is 'no', not for the very basic data layers. But you still need a basic knowledge of HTML, DOM, and JavaScript.

Q. When do you really need a web developer to write data layers for you?

A. When you have to create complex data layers, like the one which pulls ecommerce data from the back-end (server-side) and converts that data into a format, which GTM can understand.

What is a data layer snippet?

A data layer snippet is the data layer enclosed within <script> ….</script> tags.

Following is an example of a data layer snippet

```
<script>

dataLayer = [{'pageCategory': 'Statistics','visitorType': 'high-value'}];

</script>
```

What is a GTM container tag snippet?

It is the code that is used to create a new Google Tag Manager container tag.

A container tag holds all the marketing/analytics tags, triggers, and variables for your website. When you are installing Google Tag Manager on your website, you are basically installing a container tag.

Following is an example of container tag snippet:

```
<!-- Google Tag Manager -->
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'//www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-XXXX');</script>
<!-- End Google Tag Manager -->
```

Note: The container tag snippet should be placed on each and every web page of your website.

Where you should place the data layer snippet?

The data layer snippet should be placed above your GTM container tag snippet in the head section (<head>....</head>) of a web page:

For example:

```
<head>
.
.
.
<script>
  dataLayer = [{'pageCategory': 'Statistics','visitorType': 'high-value'}];
</script>

<!-- Google Tag Manager -->
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'//www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-XXXX');</script>
<!-- End Google Tag Manager -->
</head>

<body>
<!-- Google Tag Manager -->
<noscript><iframe src="//www.googletagmanager.com/ns.html?id=GTM-XXXX"
height="0" width="0"
style="display:none;visibility:hidden"></iframe></noscript>
<!-- End Google Tag Manager -->
```

Why the data layer snippet should be placed above the GTM container snippet?

If you place the data layer snippet below the container tag snippet then the data layer variables may not be available to the container tag.

For example, the following set up may not work:

```
<!-- Google Tag Manager -->

...

<!-- End Google Tag Manager -->

<script>

  dataLayer = [{'pageCategory': 'Statistics','visitorType': 'high-value'}];

</script>
```

Here the data layer variables 'pageCategory' and 'visitorType' may not be available to the GTM container, as the data layer code has been placed after the container snippet.

FAQs

<u>Q. When does Google Tag Manager function best?</u>

A. When used alongside a data layer. You should know this by now!

Q. Is it mandatory to declare the data layer while using GTM?

A. No. Explicitly declaring a data layer is optional.

Q. Can you use GTM without explicitly implementing the data layer?

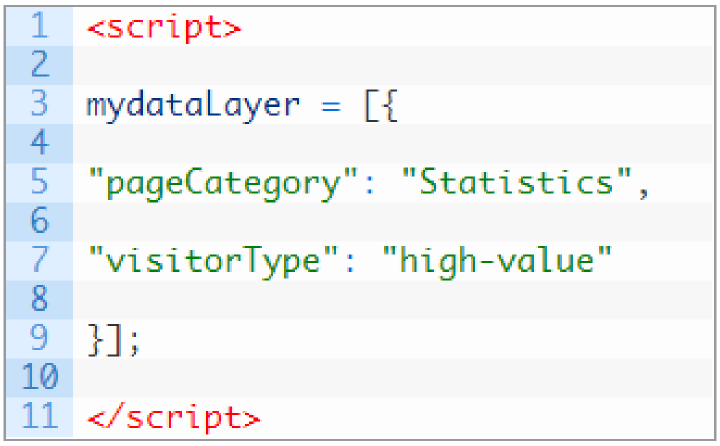
A. Yes. In that case, the GTM container code will automatically create its own empty data layer object for you.

Do you need a GTM container tag to implement a data layer?

No.

You can create a javascript array that stores all the information you want to collect about a page and use it as a data layer.

For example:



```
<script>

mydataLayer = [{

"pageCategory": "Statistics",

"visitorType": "high-value"

}];

</script>
```

FAQ

Q. How you can selectively fire tags on page load?

A. Through GTM triggers.

Q. What is the downside of not using the data layer?

A. You cannot use 'events' without data layer and above all your GTM implementation could be pretty shaky.

Q. How you can access values from a page without implementing the data layer?

A. By using GTM custom JavaScript variables.

How you can implement a data layer on a web page?

By adding following snippet of code above the container tag snippet in the head section <head>....</head> of your HTML document:

```
<script>
dataLayer = [];
</script>
```

For example, you might want to set data layer variables to indicate that the page is a 'Maths' page and that the visitor is a 'high-value' visitor.

To do this, you initialize the data layer as follows:

```
<script>
dataLayer = [{'pageCategory': 'Maths','visitorType': 'high-value'}];
</script>
```

FAQ

Q. What is the scope of data layer variables?

A. Data layer variables are page specific. That means they exist, as long as a user remains on the same page.

Q. How you can declare data layer variables that are relevant across pages?

A. By using the same data layer variable name on multiple pages of your website.

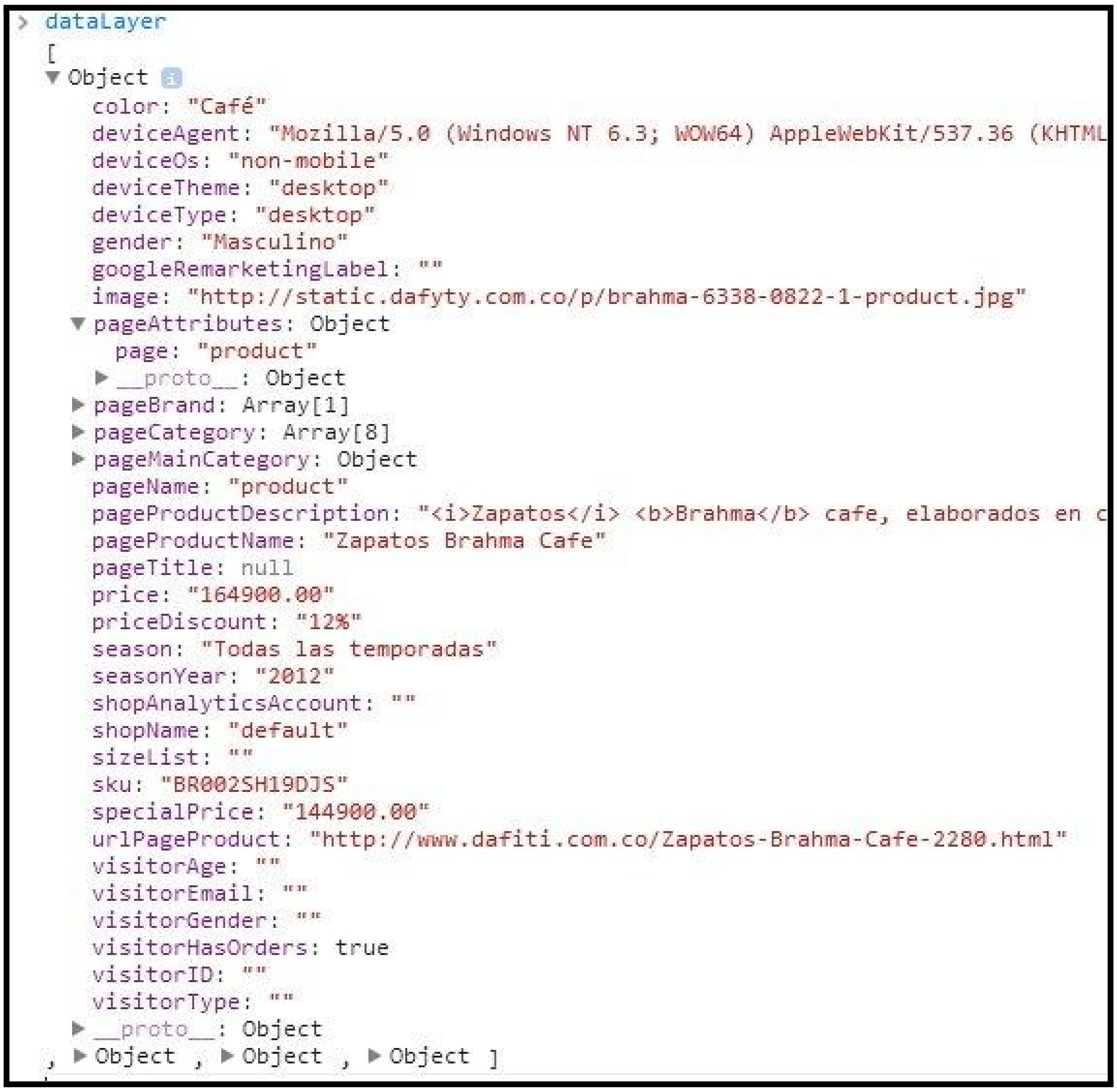
Following is a real-life example of a data layer being used on a website

```
<script type="text/javascript">
var dataLayer = [{
"color":"Caf\u00e9",
"deviceAgent":"Mozilla\/5.0 (Windows NT 6.3; WOW64) AppleWebKit\/537.36
(KHTML, like Gecko) Chrome\/35.0.1916.153 Safari\/537.36"
"deviceOs":"non-mobile",
"deviceTheme":"desktop",
"deviceType":"desktop",
"gender":"Masculino",
"googleRemarketingLabel":"",
"image":"http:\/\/static.dafyty.com.co\/p\/brahma-6338-0822-1-product.jpg",
"pageAttributes":{"page":"product"},
"pageBrand":[{"id":"2","name":"Brahma"}],
"pageCategory":[
{"id":"20","name":"Zapatos"},
{"id":"53","name":"Masculino"},
{"id":"57","name":"Clasicos"},
{"id":"138","name":"Deportes"},
{"id":"139","name":"Masculino"},
{"id":"201","name":"Zapatos"},
{"id":"1244","name":"Mocasines"},
{"id":"1340","name":"Apaches"}
],
"pageMainCategory":"id":"1340","name":"Apaches"},
"pageName":"product",
"pageProductDescription":"<i>Zapatos<\/i> <b>Brahma<\/b> cafe, elaborados
en cuero con costuras visibles en la zona delantera la cual es redonda. Forro en
cuero y suela en goma. Un calzado muy comodo que puedes combinar tanto
con tus atuendos formales como informales.",
"pageProductName":"Zapatos Brahma Cafe",
"pageTitle":null,
"price":"164900.00",
"priceDiscount":"12%",
"season":"Todas las temporadas",
"seasonYear":"2012",
"shopAnalyticsAccount":"",
"shopName":"default",
"sizeList":"",
"sku":"BR002SH19DJS",
"specialPrice":"144900.00",
"urlPageProduct":"http:\/\/www.dafiti.com.co\/Zapatos-Brahma-Cafe-280.htm
l",
}];
```

dataLayer = CookieTracking.changeGtm(dataLayer);

</script>

And here is how this data layer looks like in **Google developers tool's console tab**:



```
> dataLayer
  [
  ▼ Object i
      color: "Café"
      deviceAgent: "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML
      deviceOs: "non-mobile"
      deviceTheme: "desktop"
      deviceType: "desktop"
      gender: "Masculino"
      googleRemarketingLabel: ""
      image: "http://static.dafyty.com.co/p/brahma-6338-0822-1-product.jpg"
    ▼ pageAttributes: Object
        page: "product"
      ▶ __proto__: Object
    ▶ pageBrand: Array[1]
    ▶ pageCategory: Array[8]
    ▶ pageMainCategory: Object
      pageName: "product"
      pageProductDescription: "<i>Zapatos</i> <b>Brahma</b> cafe, elaborados en c
      pageProductName: "Zapatos Brahma Cafe"
      pageTitle: null
      price: "164900.00"
      priceDiscount: "12%"
      season: "Todas las temporadas"
      seasonYear: "2012"
      shopAnalyticsAccount: ""
      shopName: "default"
      sizeList: ""
      sku: "BR002SH19DJS"
      specialPrice: "144900.00"
      urlPageProduct: "http://www.dafiti.com.co/Zapatos-Brahma-Cafe-2280.html"
      visitorAge: ""
      visitorEmail: ""
      visitorGender: ""
      visitorHasOrders: true
      visitorID: ""
      visitorType: ""
    ▶ __proto__: Object
  , ▶ Object , ▶ Object , ▶ Object ]
```

If you look at this screenshot closely, you can see that all data layer variables are reported in **alphabetical order** in the developer's tool console tab.

That's how you should write your data layer variables, in alphabetical order. This way the debugging of the data layer becomes easier.

Another thing worth noting is that the data layer above contains some dynamic data layer variables, near the end like *VisitorsAge*, *VisitorsEmail*, *VisitorsGender*, etc.

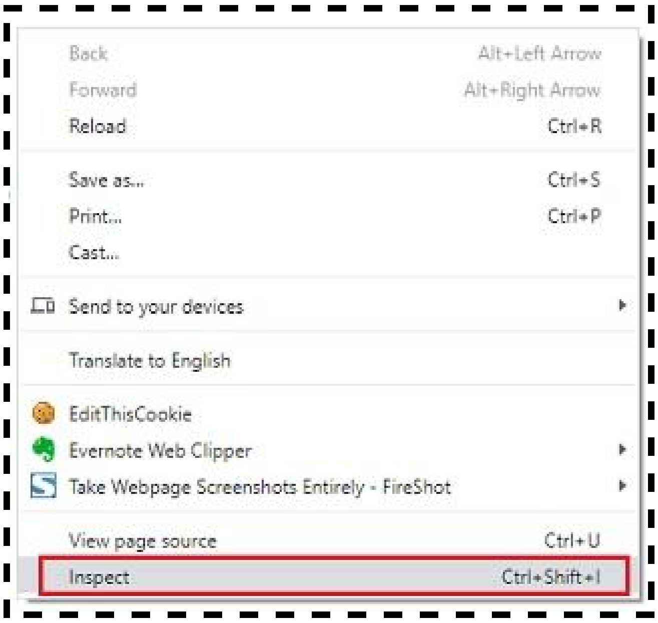
These data layer variables weren't hard-coded on the web page but were dynamically pushed into the data layer via the push method.

How to check data layer in Google Chrome Console?

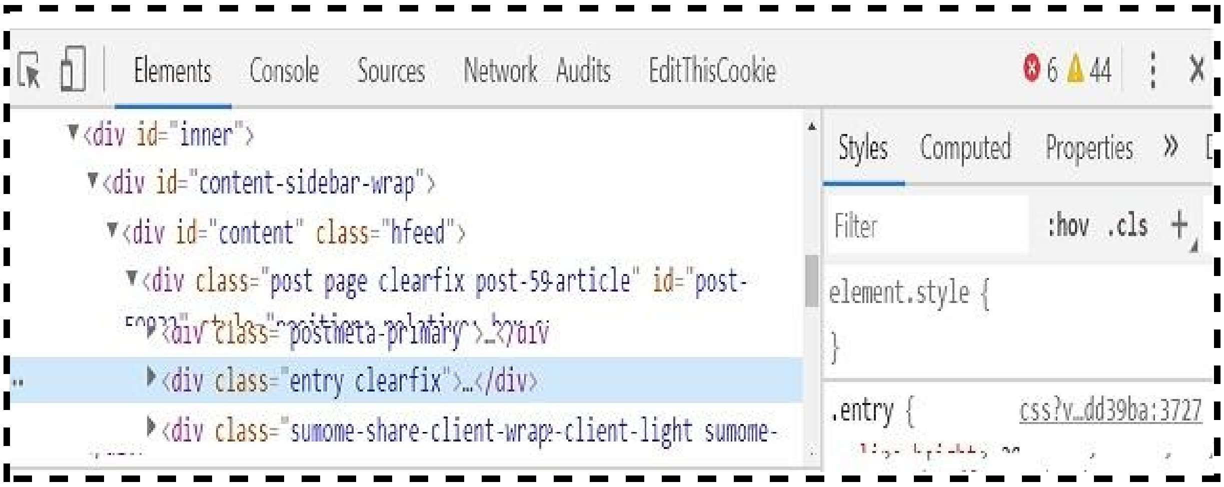
You can see the data layer on any web page via the 'Console' tab of the Google developers tool.

To do that, follow the steps below:

Step-1: In Google Chrome, right-click on a web page (the page must have Google Tag Manager installed) and then select '**Inspect**' from the drop-down menu:



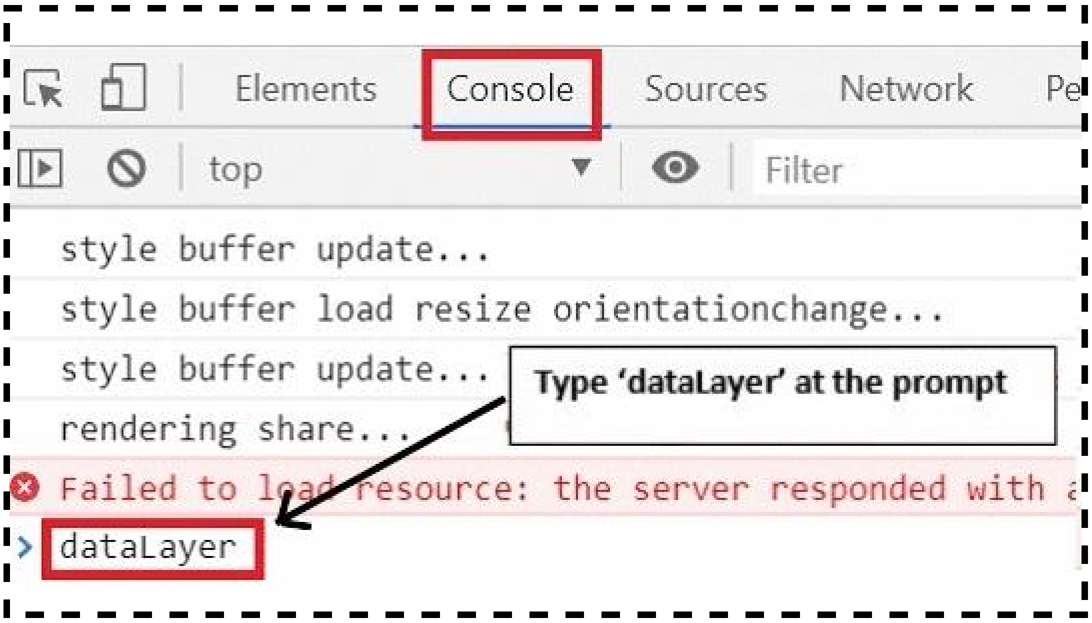
This should open up the developers' tool window at the bottom of your screen:



```
▼<div id="inner">
  ▼<div id="content-sidebar-wrap">
    ▼<div id="content" class="hfeed">
      ▼<div class="post page clearfix post-59 article" id="post-
        ▶<div class="postmeta-primary">…</div>
        ▶<div class="entry clearfix">…</div>
        ▶<div class="sumome-share-client-wrap-client-light sumome-
```

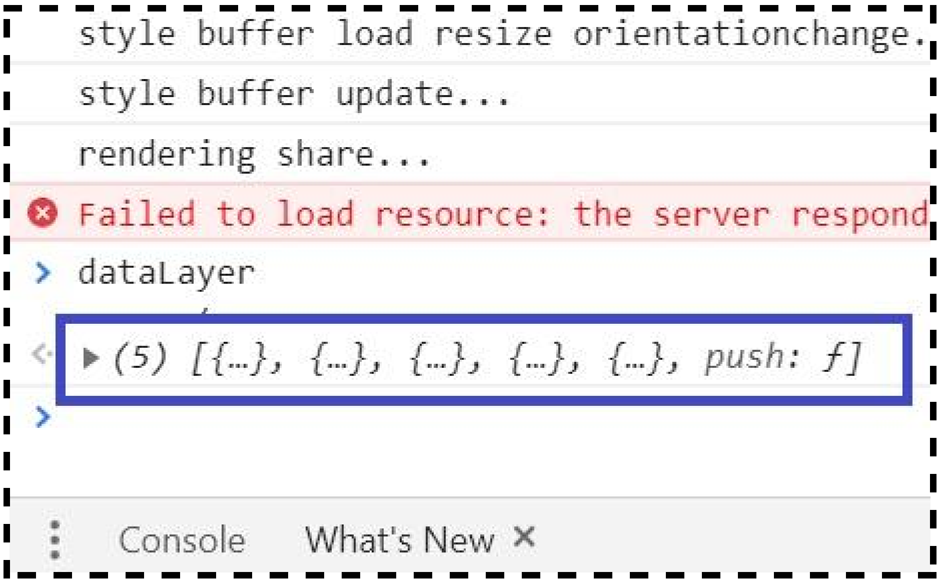
```
element.style {
}
.entry {                          css?v…dd39ba:3727
```

Step-2: Click on the '***Console***' tab at the top right of the tool window and then type 'dataLayer' at the command prompt:



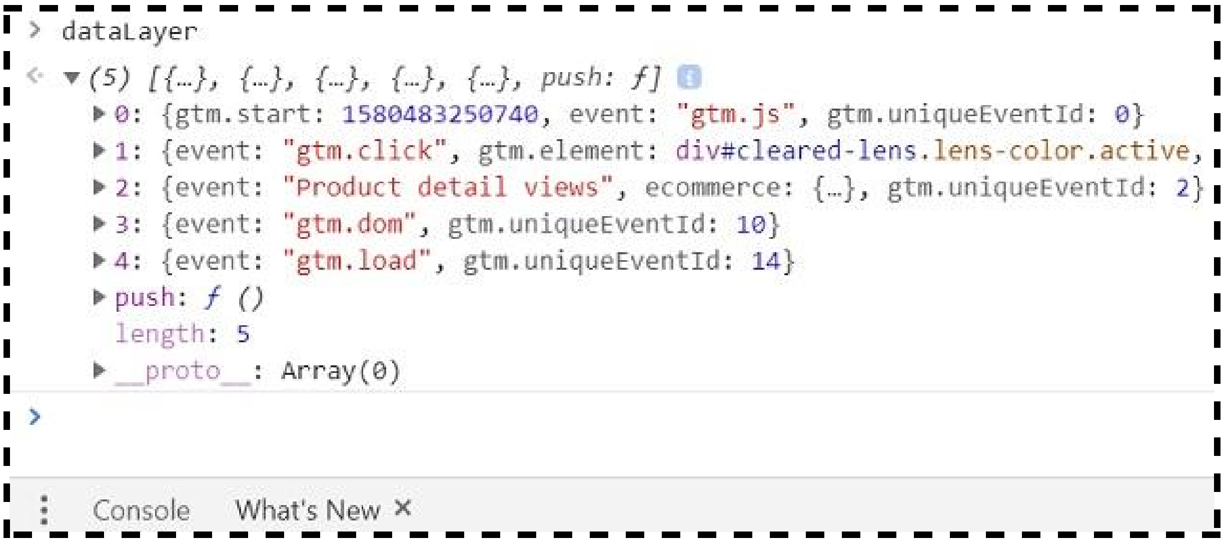
```
style buffer update...
style buffer load resize orientationchange...
style buffer update...
rendering share...
Failed to load resource: the server responded with a
> dataLayer
```

You should now see a line of code just below the dataLayer:



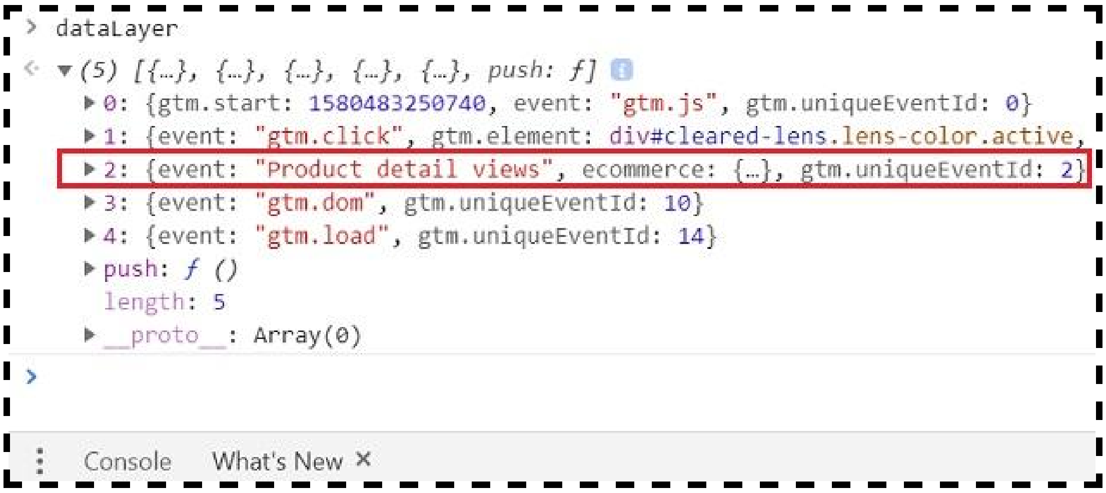
```
style buffer load resize orientationchange.
style buffer update...
rendering share...
Failed to load resource: the server respond
> dataLayer
< ▶ (5) [{…}, {…}, {…}, {…}, {…}, push: ƒ]
>
```

Click on this line of code. You should now see all the data layer variables:

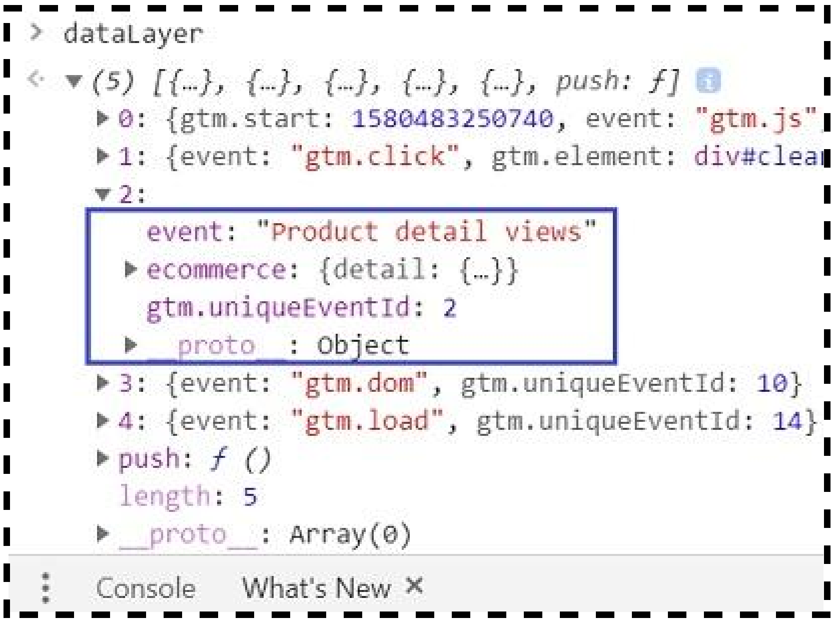


```
> dataLayer
< ▼(5) [{…}, {…}, {…}, {…}, {…}, push: ƒ]
   ▶0: {gtm.start: 1580483250740, event: "gtm.js", gtm.uniqueEventId: 0}
   ▶1: {event: "gtm.click", gtm.element: div#cleared-lens.lens-color.active,
   ▶2: {event: "Product detail views", ecommerce: {…}, gtm.uniqueEventId: 2}
   ▶3: {event: "gtm.dom", gtm.uniqueEventId: 10}
   ▶4: {event: "gtm.load", gtm.uniqueEventId: 14}
   ▶push: ƒ ()
    length: 5
   ▶__proto__: Array(0)
>
```

Click on a data layer variable to see more details about that variable:



```
> dataLayer
< ▼(5) [{…}, {…}, {…}, {…}, {…}, push: ƒ]
   ▶0: {gtm.start: 1580483250740, event: "gtm.js", gtm.uniqueEventId: 0}
   ▶1: {event: "gtm.click", gtm.element: div#cleared-lens.lens-color.active,
   ▶2: {event: "Product detail views", ecommerce: {…}, gtm.uniqueEventId: 2}
   ▶3: {event: "gtm.dom", gtm.uniqueEventId: 10}
   ▶4: {event: "gtm.load", gtm.uniqueEventId: 14}
   ▶push: ƒ ()
    length: 5
   ▶__proto__: Array(0)
>
```



```
> dataLayer
< ▼(5) [{…}, {…}, {…}, {…}, {…}, push: ƒ]
    ▶0: {gtm.start: 1580483250740, event: "gtm.js"
    ▶1: {event: "gtm.click", gtm.element: div#clea
    ▼2:
       event: "Product detail views"
      ▶ecommerce: {detail: {…}}
       gtm.uniqueEventId: 2
      ▶ __proto__: Object
    ▶3: {event: "gtm.dom", gtm.uniqueEventId: 10}
    ▶4: {event: "gtm.load", gtm.uniqueEventId: 14}
    ▶push: ƒ ()
     length: 5
    ▶__proto__: Array(0)
```

gtm.js, gtm.dom and gtm.load events

Every data layer gets the following three events from a container tag:

1. gtm.js
2. gtm.dom
3. gtm.load events

These events are fired in the following sequence: **gtm.js > gtm.dom > gtm.load** gtm.js is a JavaScript library file used by the container tag. If this library file doesn't load then your GTM won't work. Once gtm.js library file is loaded, the container tag passes the **event 'gtm.js'** to the data layer.

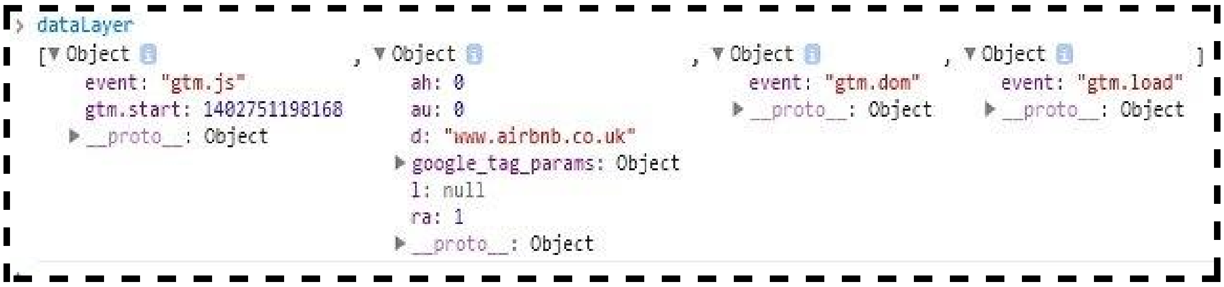
gtm.dom is a DOM file used by the container tag. If this file doesn't load then it means, DOM has not loaded. Once the DOM is loaded, the container tag passes the **event 'gtm.dom'** to the data layer.

gtm.load is a load file used by the container tag. If this file doesn't load then it means, window and all its contents are not loaded. Once the window and all its contents are fully loaded, the container tag passes the **event 'gtm.load'** to the data layer.

If your data layer doesn't get any/all of these events then it means something has gone wrong with your tag manager implementation. It could either be a broken container tag or a broken data layer or both.

If you have a hard-coded data layer on a page then the developers' tool window will show you at least four JavaScript objects.

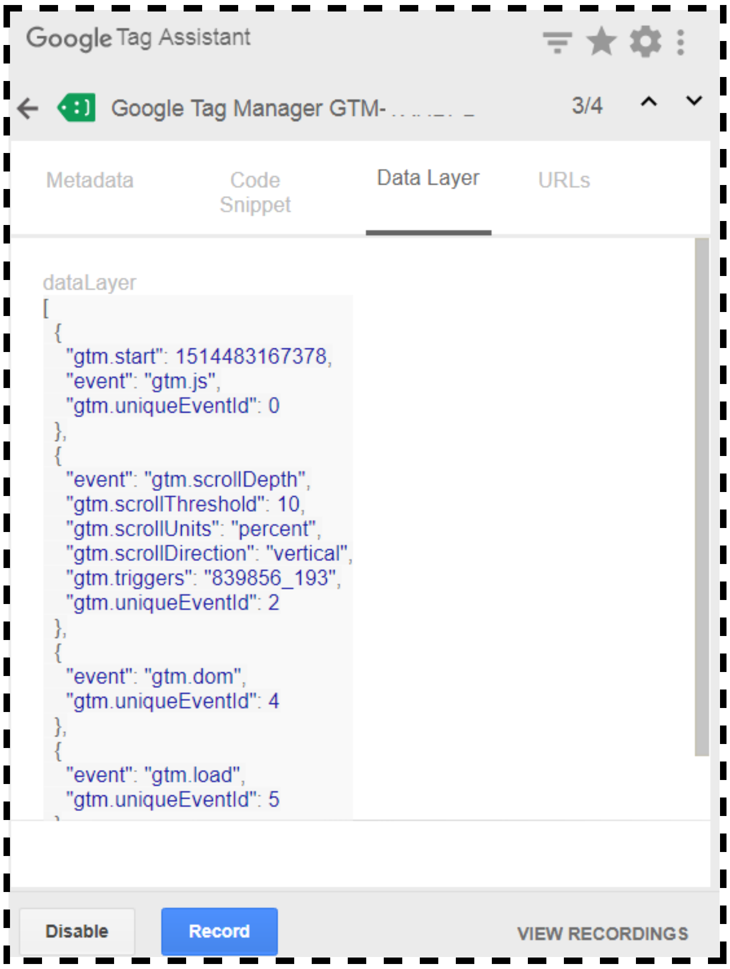
One JavaScript object would be your data layer and the other three will be the default JavaScript event objects:



```
> dataLayer
  [▼Object                           , ▼Object                           , ▼Object                , ▼Object                ]
      event: "gtm.js"                    ah: 0                               event: "gtm.dom"        event: "gtm.load"
      gtm.start: 1402751198168           au: 0                             ▶__proto__: Object      ▶__proto__: Object
    ▶__proto__: Object                   d: "www.airbnb.co.uk"
                                       ▶google_tag_params: Object
                                         l: null
                                         ra: 1
                                       ▶__proto__: Object
```

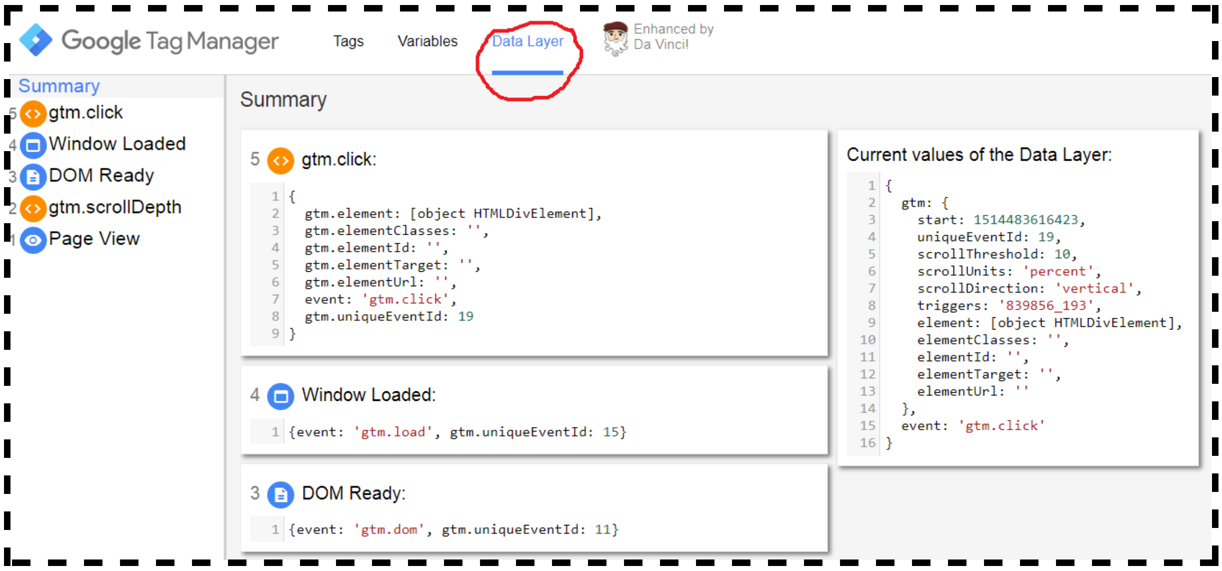
Other methods to check data layers on a web page

Another method to inspect a data layer is by using the Google Tag Assistant Chrome extension:



```
dataLayer
[
  {
    "gtm.start": 1514483167378,
    "event": "gtm.js",
    "gtm.uniqueEventId": 0
  },
  {
    "event": "gtm.scrollDepth",
    "gtm.scrollThreshold": 10,
    "gtm.scrollUnits": "percent",
    "gtm.scrollDirection": "vertical",
    "gtm.triggers": "839856_193",
    "gtm.uniqueEventId": 2
  },
  {
    "event": "gtm.dom",
    "gtm.uniqueEventId": 4
  },
  {
    "event": "gtm.load",
    "gtm.uniqueEventId": 5
```

The best method to inspect a data layer is by using the 'Data Layer' tab of your **GTM debug and preview mode**:



5 gtm.click:

```
{
  gtm.element: [object HTMLDivElement],
  gtm.elementClasses: '',
  gtm.elementId: '',
  gtm.elementTarget: '',
  gtm.elementUrl: '',
  event: 'gtm.click',
  gtm.uniqueEventId: 19
}
```

4 Window Loaded:

```
{event: 'gtm.load', gtm.uniqueEventId: 15}
```

3 DOM Ready:

```
{event: 'gtm.dom', gtm.uniqueEventId: 11}
```

Current values of the Data Layer:

```
{
  gtm: {
    start: 1514483616423,
    uniqueEventId: 19,
    scrollThreshold: 10,
    scrollUnits: 'percent',
    scrollDirection: 'vertical',
    triggers: '839856_193',
    element: [object HTMLDivElement],
    elementClasses: '',
    elementId: '',
    elementTarget: '',
    elementUrl: ''
  },
  event: 'gtm.click'
}
```

FAQ

Q. Can you use the same data layer with every tag management solution?

A. Each tag management solution (TMS) provider implements a data layer in a slightly different way. So most likely, you won't be able to use the same data layer with every (TMS) (if you use more than one TMS).

Q. Who should hard code data layers on a website: you or your web developer?

Data layers should be hardcoded on a website by your developer or IT team.

Q. Does the use of data layers make you independent from your developer or IT?

A. As long as you are using hardcoded data layers, you may need to depend upon IT for its updates and maintenance.

Q. Do you need to be a web developer in order to use a data layer?

A. The answer is "No", not for the very basic data layers. But you still need a basic knowledge of HTML, DOM, and JavaScript.

Q. When do you really need a web developer to write data layers for you?

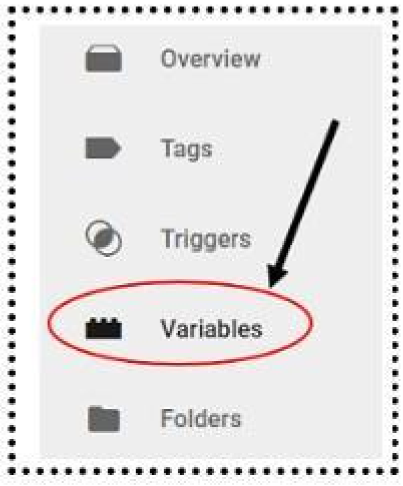
A. When you have to create complex data layers, like the one which pulls ecommerce data from the back-end (server-side) and converts that data into a format, which GTM can understand.

Data layer variable in Google Tag Manager

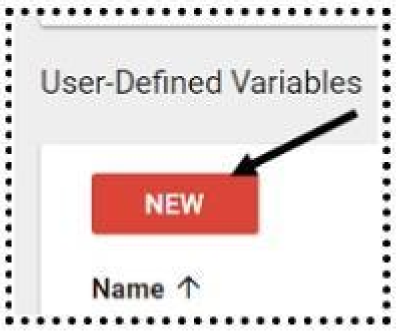
A data layer variable is a user-defined variable in GTM through which you can pull data from the data layer into Google Tag Manager.

Follow the steps below to create a data layer variable in GTM:

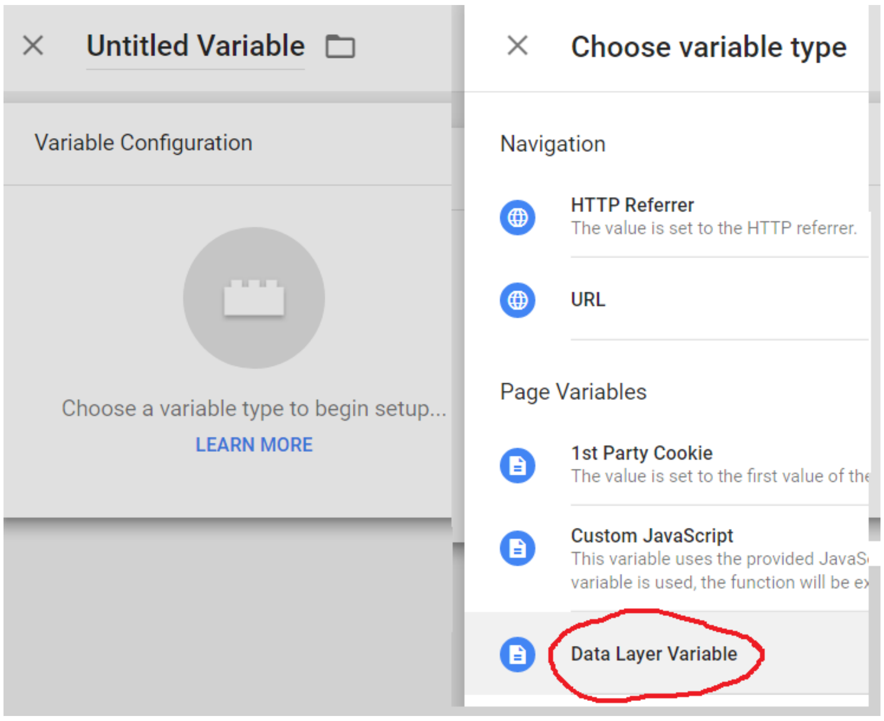
Step-1: Login to your GTM account and then click on the 'Variables' link in the left-hand side menu:



Step-2: Scroll down to '**User-Defined Variables**' section and then click on the '**New**' button:



Step-3: Select 'Data Layer Variable' from the '*Choose Variable Type*' menu



Untitled Variable

Variable Configuration

Choose a variable type to begin setup...

LEARN MORE

Choose variable type

Navigation

HTTP Referrer
The value is set to the HTTP referrer.

URL

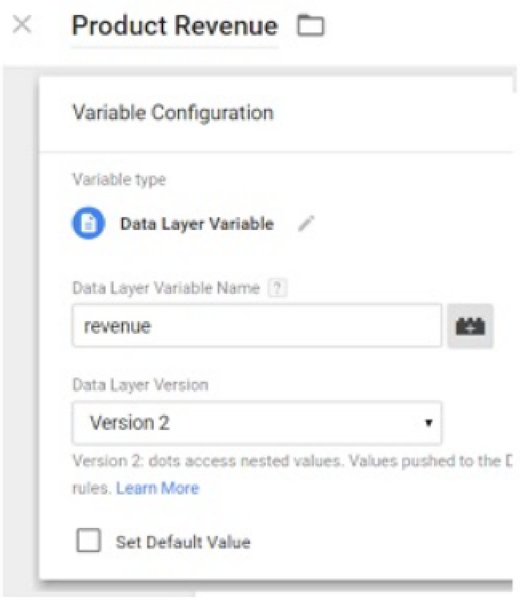
Page Variables

1st Party Cookie
The value is set to the first value of the

Custom JavaScript
This variable uses the provided JavaS
variable is used, the function will be ex

Data Layer Variable

Step-4: Enter a name for your variable and specify the Data Layer key (in our case ‘revenue’) whose value you want to retrieve:



Product Revenue

Variable Configuration

Variable type

Data Layer Variable

Data Layer Variable Name

revenue

Data Layer Version

Version 2

Version 2: dots access nested values. Values pushed to the D
rules. Learn More

Set Default Value

In order to understand how data layer variables can be used in real-life tracking situation, read this article: **Learn to track Website Sales in Facebook via Google Tag Manager**

You are doing Google Analytics all wrong. Here is why...

I have dealt with hundreds of Google Analytics accounts in my career.

I have seen many issues, from incorrect tracking code, selecting the wrong KPIs, to analyzing data without using custom reports or advanced segments.

But do you know the biggest issue of all in Google Analytics?....

It is the "misinterpretation of analytics data".

Many marketers make the mistake of crediting conversions to the wrong marketing channel.

And they seem to be making this mistake over and over again.

They give the credit for conversions to the last touchpoint (campaign, ad, search term...).

They can’t help themselves because they believe that the Google Analytics reports are ‘what you see is what you get’.

But they are actually ‘what you interpret is what you get’.

This has resulted in marketers making wrong business decisions and losing money.

All the data you see in Google Analytics reports today lies to you unless you know exactly how to interpret it correctly.

For example, let's talk about direct traffic.

All untagged or incorrectly tagged marketing campaigns, from display ads to emails, could be reported as direct traffic by Google.

Whenever a referrer is not passed, the traffic is reported as direct traffic by Google.

Mobile applications don’t send a referrer. Word/PDF documents don’t send a referrer.

‘302 redirects’ sometimes cause the referrer to be dropped. Sometimes browsers don’t pass the referrer.

During an HTTP to HTTPS redirect (or vice versa), the referrer is not passed because of security reasons.

All such traffic is reported as direct traffic by Google.

So on the surface, it may look like most people are visiting your website directly, but this is not usually the case.

But this analysis does not end here because you are still not looking at the complete picture.

People do not always access your website directly and then make a purchase straight away.

They are generally exposed to multiple marketing channels (display ads, social media, paid search, organic search, referral websites, email, etc.) before they access your website directly.

Before they make a purchase.

So if you are unaware of the role played by prior marketing channels, you will credit conversions to the wrong marketing channels.

Like in the present case, to direct traffic.

To get this type of understanding, you need to understand and implement web analytics.

But **you learn data analysis and data interpretation from web analytics and not from Google Analytics.**

The direction in which your analysis will move will determine the direction in which your marketing campaigns will move.

You get that direction from ‘web analytics’ and not from ‘Google Analytics’.

Web/digital analytics is not about Google Analytics (GA) or Google Tag Manager (GTM). It is about analyzing and interpreting data, setting up goals, strategies and KPIs.

It’s about creating a strategic roadmap for your business. That's why the knowledge of web/digital analytics is so important.

So, what I have done is put together some completely free training for you.

This training will teach you what digital analytics really is and how I have been able to leverage it to generate floods of new sales and customers.

I will also show you how you can copy what I have done to get similar results.

You can sign up for the free training here:

Reserve My Seat Now

I hope you find it helpful.

All the best,

Himanshu