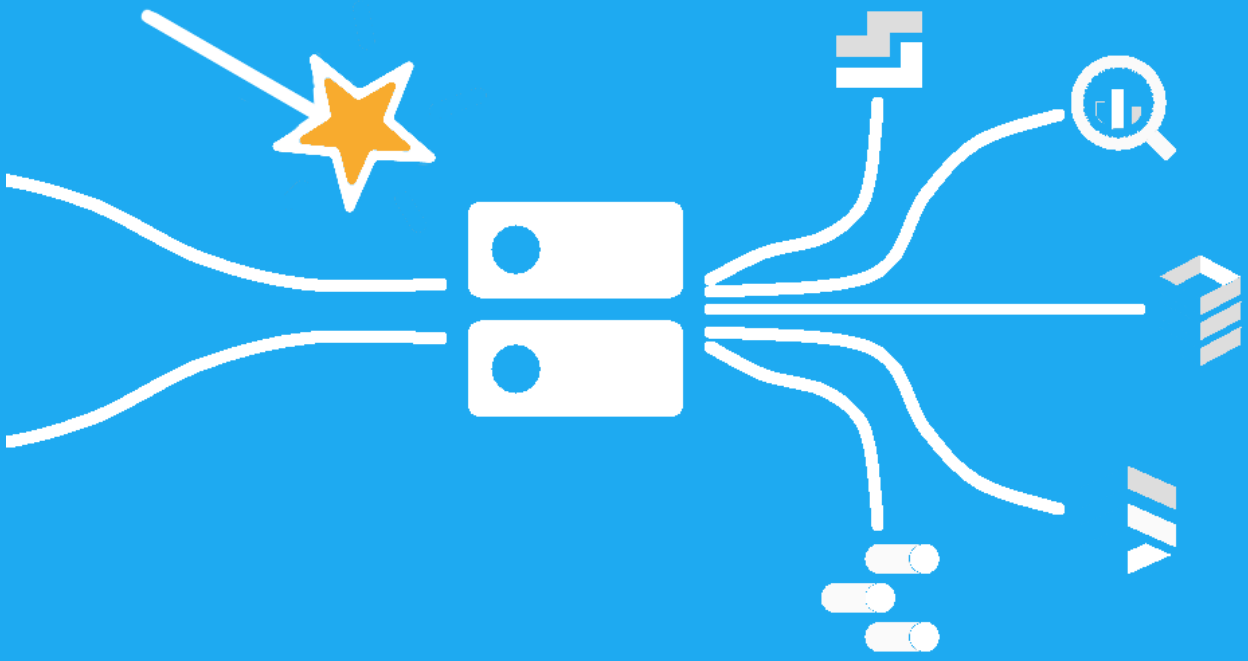
Das inoffizielle Handbuch *Leseprobe*

Google Tag Manager Server Templates

Vorlagen nutzen, verstehen und selbst entwickeln



Markus Baersch

2. Auflage | April 2022

Inklusive Bonuskapitel: Tag-Vorlage für client-side Google Tag Manager

Auszug / Leseprobe Das ganze E-Book: markus-baersch.de/gtm-server-templates-buch

Vorwort

ITP, ETP und viele andere Beschränkungen der Browser beim Tracking erschweren immer mehr das clientseitige Tracking. Der Einsatz des Serverside Google Tag Managers wird dabei häufig als Universal-Lösung gesehen. Doch der Serverside Google Tag Manager löst viele aber nicht alle Probleme. Viel wichtiger ist allerdings, das der erst kürzlich aus der Beta-Phase entlassene Serverside Google Tag Manager viel Potential hat, das aber derzeit erst durch die individuelle Entwicklung von Templates von Clients, Tags und Variablen vollumfänglich genutzt werden kann. Denn noch ist das Angebot an vordefinierten Tags, Clients und Variablen sehr begrenzt. Zwar wächst das Angebot in der Template Gallery stetig, doch ist die Fähigkeit zur Entwicklung eigener Templates ein wichtiger Schritt, wenn keine Abhängigkeit gegenüber Dritten entstehen soll.

Markus Baersch hat mit diesem Buch einen einfachen Einstieg in die Template-Entwicklung für den Serverside Google Tag Manager geschaffen, das sich auf notwendige theoretische Informationen und einen starken Praxisteil fokussiert. Es werden viele Themen für einen einfachen und schnellen Einstieg angesprochen, der in die Lage versetzt, sich detaillierter mit den Themen zu beschäftigen.

Marcus Stade

Vorwort zur zweiten Auflage

Es hat kein halbes Jahr gedauert, bis sich so wesentliche Neuerungen im Funktionsangebot des ssGTM ergeben haben, dass ein Update dieses E-Books erforderlich wurde. Die geschlossene Lücke betrifft einen der wesentlichen Unterschiede zwischen dieser Lösung zur Errichtung eines eigenen Tracking-Endpunkts und vielen anderen: Die Anreicherung von Daten am Server, bevor diese von den Tags an die jeweiligen Dienste weitergegeben werden können.

Dazu sind zwei wesentliche Dinge hinzugekommen:

1) Promises, die die Nutzung von APIs möglich machen und
2) Firestore als eine eigene Persistenz (jenseits von Big Query) zum schnellen Lesen und Schreiben eigener Daten

Diese Ergänzungen haben neben einem komplett neuen Kapitel zur Datenanreicherung auch an anderen Stellen zu kleinen Aktualisierungen und Ergänzungen geführt. So wurde in den Tipps z. B. der Abschnitt Rückmeldung vom Client via returnResponse() hinzugefügt.

Wer zudem die Beispielcodes aus der ersten Auflage mit der zweiten vergleicht, wird feststellen, dass die bisherigen Callback-Funktionen von API Aufrufen wie *sendHttpGet()* in bestehenden Codes *nicht* gegen Promises (Infos zu Promises siehe auch [38]) ausgetauscht wurden. Denn die Anpassung in den Google APIs bedeutet weder, dass die bisherigen Methoden nicht mehr funktionieren, noch bringt es in den konkreten Beispielen einen wirklichen Vorteil. Die neuen Beispiele im hinzugekommenen Kapitel zur Anreicherung von Daten demonstrieren an zentraler Stelle die o. G. Vorteile, so dass man nicht das ganze E-Book durcharbeiten muss, um an das hinzugekommene Wissen der zweiten Auflage zu kommen. Viel Spaß mit dem ssGTM und seinen neuen Möglichkeiten!

Markus Baersch, April 2022

Einführung

Serverside GTM ohne Templates ist wie Rennwagen ohne Rennstrecke

Seit es die serverseitige Variante des Google Tag Managers gibt, ist dessen Nutzen primär auf das Google Universum fokussiert: Vorgefertigte Clients konzentrieren sich ganz auf Google Analytics Formate und die sichere Auslieferung des clientseitigen Containers sowie der Trackingscripts. Tags senden Daten an Google Analytics oder Ads. Für alles andere gibt es eine rudimentäre Vorlage zum Senden von Requests. Das ist schon fast alles.

Um den serverseitigen Google Tag Manager (ab hier kurz: “ssGTM”) zur vollen Entfaltung zu bringen, ist je nach gedachtem Einsatz deutlich mehr nötig. Datensammlung in anderen Formaten, Weitergabe an andere Dienste als Google Analytics oder Ads: Das alles geht nur mit benutzerdefinierten Vorlagen. Diese muss man selbst erstellen oder auf durch Dritte bereitgestellte Templates für Clients, Tags oder Variablen zurückgreifen.

Die Community Gallery [1] des Google Tag Managers hält bereits einige Vorlagen bereit, die importiert und eingesetzt werden können. Diese lösen Aufgaben wie Anbindung von Systemen (z. B. ActiveCampaign, Klavyio, Mailchimp oder HubSpot), serverseitiges Tracking für Facebook und andere Tracking-Dienste.

Vorhandene Vorlagen für Variablen ermöglichen die Verarbeitung oder Anpassung von empfangenen Daten für die gezielte Weiterverarbeitung in Tags. Clients hingegen fehlen derzeit komplett und können nur manuell importiert werden, wenn anderenorts eine Vorlage existiert.

Wenngleich die eher dünne Versorgungslage mit der Zeit facettenreicher wird, braucht es für individuelle Anforderungen beim Empfang, der Verarbeitung und Weitergabe von eingehenden Daten meist speziell zugeschnittene Vorlagen.

Dieses Buch zeigt anhand konkreter Beispiele, wie eigene Templates für Tags, Clients und Variablen erstellt und eingesetzt werden können.

Über den Autor

Markus Baersch ist Geschäftsführer der gandke marketing & software gmbh (www.gandke.de) aus Mönchengladbach, die Kunden aus jedem Marktsegment bei der erfolgreichen Vermarktung und laufenden Optimierung ihrer Shops oder B2B-Websites unterstützt. Seit ca. 15 Jahren gehören hauptsächlich Digital Analytics, Tag Management und SEM zum beruflichen Alltag.

Zusammen mit Michael Janssen betreibt er seit 2016 den Web-Analytics Podcast "beyond pageviews" auf termfrequenz.de [2].

Konzeption und Implementierung von client- und serverseitigem Tracking und -tagging mit und ohne Google Tag Manager sind seit Jahren ein Schwerpunktthema - sowohl operativ als auch in Form von Schulungen, Workshops und Vorträgen. Seit der öffentlichen Verfügbarkeit des ssGTM als Beta im August 2020 ist dieser schnell die bevorzugte Plattform für komplexere Tracking-Setups geworden. Daraus sind einige Vorlagen für den client- und serverseitigen Tag Manager entstanden, die z. T. auch in der Community Gallery verfügbar sind.

Stand der Dinge, Änderungen und Ergänzungen

Dieses Buch ist in der ersten Auflage im Herbst 2021 entstanden; zufällig zeitgleich mit dem Ende der Beta-Phase. Der ssGTM war zu diesem Zeitpunkt keine anderthalb Jahre öffentlich zugänglich. Änderungen sind daher je nach Zeitpunkt des Lesens sehr wahrscheinlich.

Die zweite Auflage auf dem Stand vom April 2022 beinhaltet wesentliche Ergänzungen, die im Vorwort zur Auflage beschrieben sind.

Aktualisierungen zu den in diesem Buch vorgestellten Möglichkeiten und weitere Ressourcen rund um den Tag Manager finden sich auf der Website des Autors [3] und bei YouTube [4].

Custom Templates in Google Tag Manager

Benutzerdefinierte Vorlagen im Google Tag Manager (“GTM”) sind nichts grundlegend Neues. Das Konzept ist schon aus dem normalen clientseitigen Google Tag Manager bekannt und findet sich auch beim ssGTM sehr ähnlich wieder.

Ein Template dient dazu, dem GTM neue Elemente hinzuzufügen, die bei der Anlage eines Tags, einer Variable - oder beim ssGTM auch einem Client - zur Auswahl stehen.

Funktionsweise

Um ein eigenes Template zu erstellen, bietet der GTM einen eigenen Vorlageneditor an, in dem Templates erstellt, mit einer Oberfläche zur Definition von Parametern (“Feldern”) versehen, getestet und mit den erforderlichen Berechtigungen versehen werden können.

Zur Programmierung der Templates wird auf JavaScript in einer sogenannten “Sandbox” gesetzt. Dadurch wird der Code eines Templates in einem abgesicherten Modus zur Laufzeit des GTM betrieben, in dem ausschließlich Funktionen und Ressourcen genutzt werden können, die dort durch die Sandbox zur Verfügung gestellt werden. Diese Einschränkungen sollen sowohl für Sicherheit und Robustheit des Codes sorgen, als auch Transparenz darüber schaffen, welche Informationen z. B. ein Tag verarbeiten und wohin es Daten senden darf.

In diesen Aspekten unterscheiden sich Templates für den Betrieb im Browser nicht von denjenigen, die für den ssGTM entwickelt werden. Im Detail zeigen sich aber deutliche Unterschiede.

Darum sind eigene Templates am Server unerlässlich

Ein wesentlicher Punkt bzgl. Templates, der beim ssGTM anders ist als im Browser: Es gibt keine Alternativen. Während man im clientseitigen GTM das volle Potenzial von JavaScript in Form von *Benutzerdefinierten JavaScript*-Variablen und Scriptcode in *Benutzerdefinierten HTML*-Tags nutzen kann, ist die Situation am Server deutlich restriktiver. Alle gewünschten Verarbeitungsschritte beim Empfang von eingehenden Requests (in Clients) und deren Weitergabe durch Tags sind durch die bestehenden Eigenschaften dieser Elemente beschränkt. Dabei werden Daten aus dem Event-Modell des ssGTM oder eingehenden Request-Daten weitergegeben. Entweder bleiben die Daten dabei unverändert oder werden transformiert; z. B. durch Einsatz von Variablen.

Dieser Unterschied zwischen Server- und Web-GTM ist durchaus gravierend: Im Gegensatz zum Web-GTM kann der ssGTM nicht einfach JavaScript nutzen, um Transformationen zu ermöglichen. Es gibt nur ein begrenztes Angebot an APIs, die in Vorlagen dazu dienen, um

- Informationen zu empfangen (bspw. anfordern von Cookies aus dem Browser),
- zu transformieren und
- zu senden (Request senden, Cookie setzen etc.).

Viele in Templates für Web-Container verfügbare Standard-Funktionen der Sandbox sind daher auch am Server nicht nutzbar.

Das Schaubild aus der Einführung in die Funktionsweise des SSTM [5] zeigt die wesentliche Rolle von Clients und Tags im ssGTM in der Verarbeitungskette.

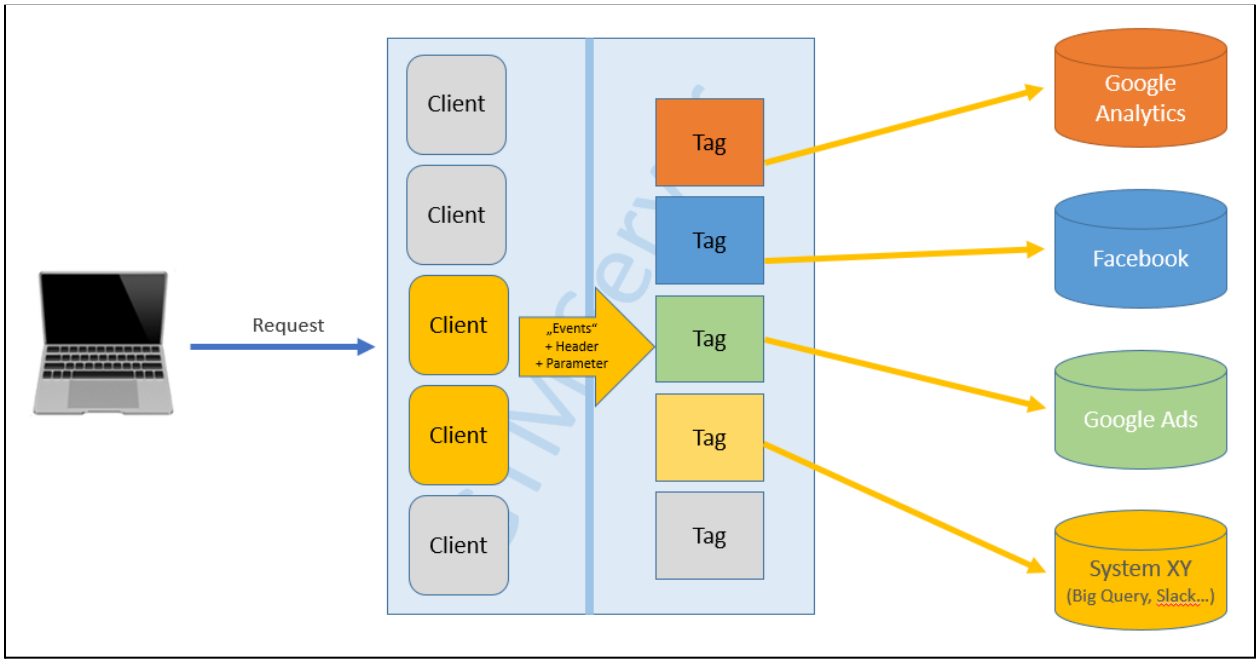


Abbildung 1: Funktionsweise des ssGTM

In einem Satz: Requests werden entgegengenommen, von *Clients* (die dieses Format verarbeiten können) in ein Standard-*Event-Modell* übertragen und auf der anderen Seite von *Tags* weiterverarbeitet, die die Daten im passenden Format an die Tracking-Dienste senden.

Sobald ein anderes Request-Format als “Analytics” (Universal Analytics, GA4 oder die entsprechenden Fassungen des Measurement Protocols) verarbeitet werden soll, ist ein dazu passender eigener Client erforderlich.

Gleiches gilt für Tags, die i. d. R. für die Weitergabe an Tracking-Dienste wie Google Analytics, Facebook, Matomo oder andere Empfänger wie Big Query Datenbanken o. Ä. zuständig sind.

In der Praxis bedeutet dies: Selbst für Kleinigkeiten braucht man ein eigenes Template. Transformationen oder Ergänzungen von Daten für einen bestimmten Empfänger (in Form eines Tags) sind zum Glück mit Variablen - auch mittels bereits bestehenden Vorlagen - zu erledigen. Dazu muss aber das passende Tag erst einmal vorhanden sein.

Templates für Clients hingegen sind immer dann notwendig, wenn alternative Formen der Erhebung von Daten (i. d. R. im Browser) und Versand zum ssGTM in einem eigenen Format gewünscht sind. Das kann ein selbst definiertes Format sein oder Requests eines bestehenden Systems wie Piwik PRO, Matomo, Plausible oder andere Analytics-Lösungen, um im Webanalyse - Kontext zu bleiben.

Auch APIs von CRM Systemen, Marketing-Automation, Customer Data Platforms, Apps, Zugangskontrollsystemen oder beliebigen anderen potenziellen Datenquellen wie IoT Geräten können genutzt werden, um Daten an einen ssGTM zu senden und sie dort weiter zu verarbeiten.

Glücklicherweise sind bereits einige Vorlagen zu verschiedensten Zwecken vorhanden. Für einen ersten Schritt ist es daher stets eine gute Idee, sich in der *Community Template Gallery* umzusehen. Das kann entweder in einem “Gesamtkatalog” [1] im Web geschehen - oder direkt im GTM bei der Anlage von neuen Elementen.

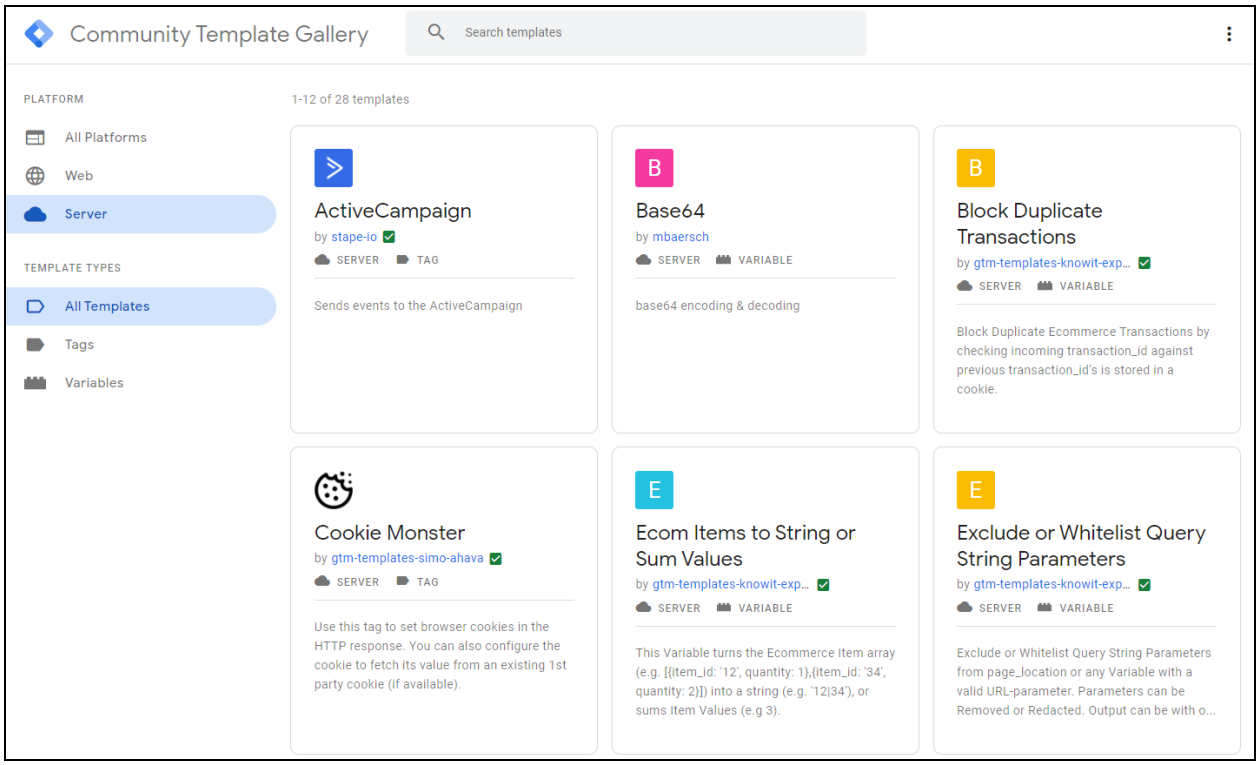


Abbildung 2: Community Template Gallery

Benutzerdefinierte Templates nutzen

Wird ein neues Element wie ein Tag oder eine Variable (Clients sind derzeit nicht in der Gallery vorhanden) im GTM angelegt, kann der Typ aus einer Liste ausgewählt werden. Oberhalb dieser Listen findet sich ein Link zur Auswahl verfügbarer Vorlagen aus der Community.

Import aus der Gallery

Es werden vorhandene Vorlagen aus der Gallery zum Import angeboten. Zu jedem Typ gibt es, neben Angaben zum Autor, Link zum Repository und der Autoren-Website etc., genaue Informationen darüber, was an Berechtigungen erforderlich ist.

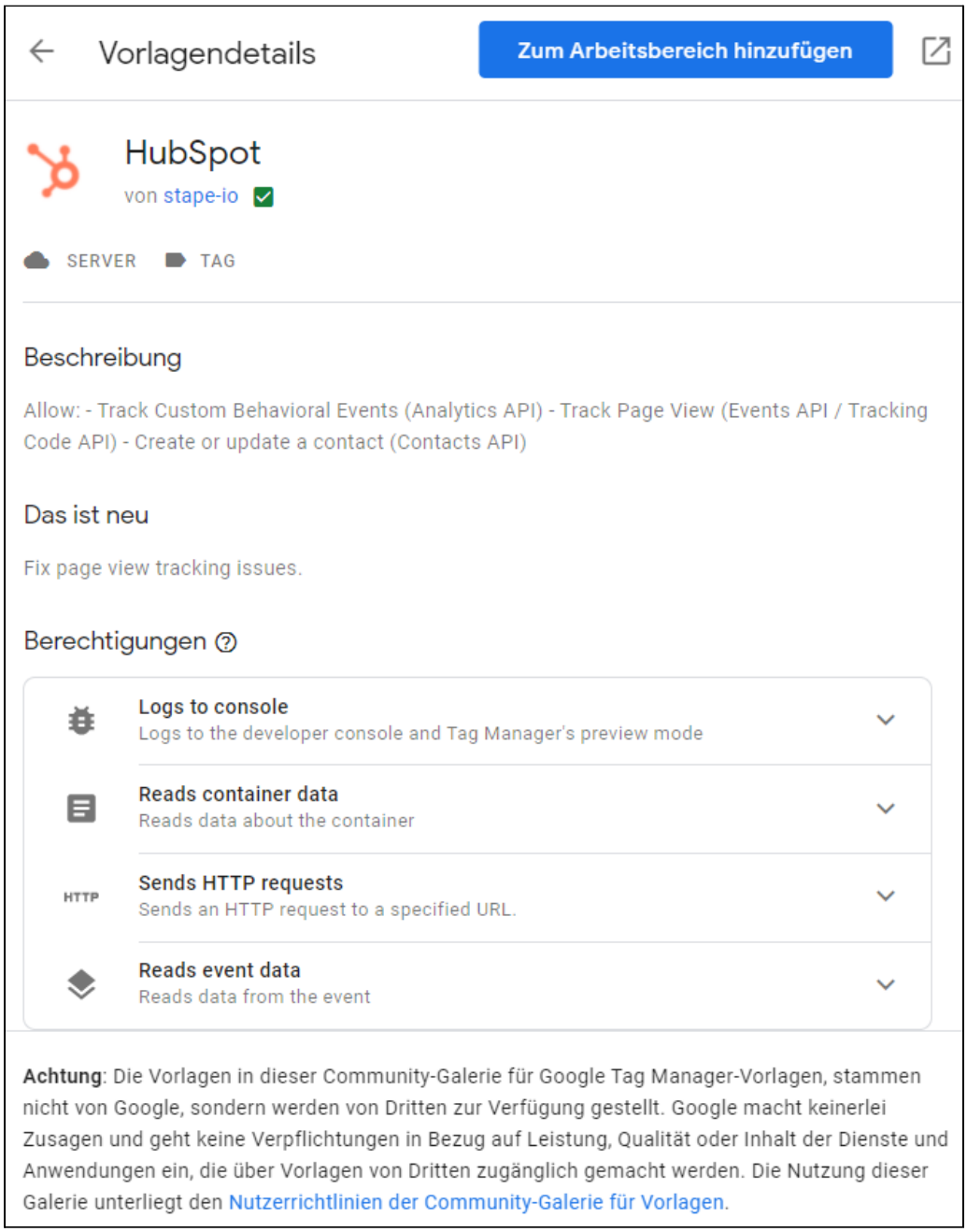


Abbildung 3: Details einer Vorlage aus der Community Template Gallery

Mit einem Klick kann die Vorlage importiert und dann im eigenen GTM Container genutzt werden.

Ein weiterer Vorteil von Templates: Üblicherweise werden diese weiterentwickelt, wenn sich z. B. das Format des angebundenen Dienstes ändert. Stehen Aktualisierungen zur Verfügung, informiert der GTM aktiv darüber und ermöglicht den Import einer neuen Vorlage nach Kontrolle der Änderungen.

Wie das obige Beispiel zeigt, muss ein Tag zur Anbindung eines Dienstes nicht zwingend vom jeweiligen Betreiber selbst bereitgestellt werden. Vielmehr stammen die meisten Vorlagen derzeit aus der Praxis: Jemand wollte oder musste Dienst X (hier HubSpot) anbinden, hat dazu ein Template entwickelt und stellt dieses über die Template Gallery auch anderen Nutzern zur Verfügung.

Manueller Import

Selbst für Clients gibt es bereits einige Kandidaten. Diese liegen aber (noch) auf GitHub und anderen Plattformen bzw. werden ggf. vom Autor auf der eigenen Website angeboten. Sucht man bei GitHub nach "*tag manager client template*", finden sich zum Zeitpunkt der Erstellung dieses Buchs (Herbst 2021) 16 Repositories.

Auch als "Gist" (dem kleinen Bruder der Repositories) lassen sich einige weitere Templates für Clients dort finden; unter anderem den Client *200 Qapla'* [6] (mehr dazu gleich). Alle Templates bestehen im Kern aus einer einzigen Datei, die alle erforderlichen Angaben zum Template, den Code und Felddefinitionen, angelegte Tests etc. enthält. Diese heißt bei Standard-Vorlagen aus der Gallery stets *template.tpl*; kann aber wie im Beispiel von *200 Qapla'* auch anders heißen, solange die Erweiterung *.tpl* beibehalten wird.

Lädt man die Datei (z. B. über die Schaltfläche "*Raw*" über dem Code und "*Speichern unter...*" im Browser) auf den eigenen Rechner, kann das Template manuell im ssGTM importiert werden.

Dazu wird unter "Vorlagen" im ssGTM im entsprechenden Bereich (hier: Client-Vorlagen) eine neue Vorlage angelegt und über das "..." Menü oben links der Import der Vorlage gestartet.

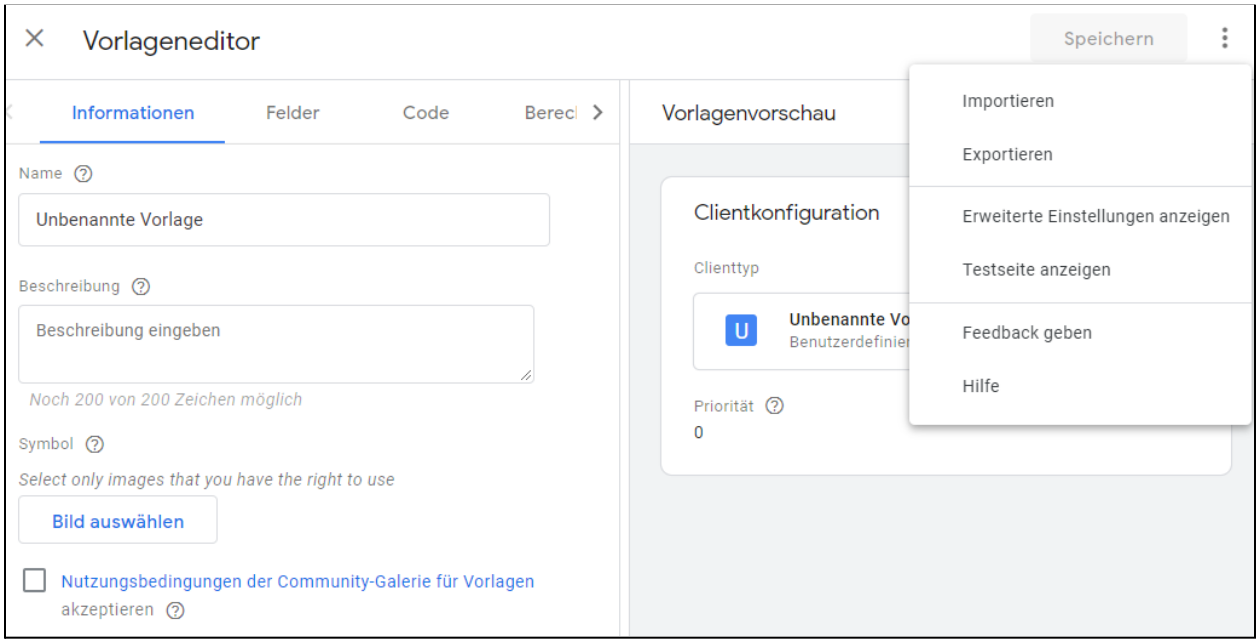
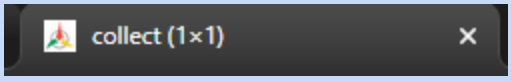


Abbildung 4: Import einer Vorlage aus einer Datei

Nach der Auswahl der Import-Datei werden alle Einstellungen eingelesen und die Informationen, Felder, Code etc. übernommen. Anschließend kann die Vorlage gespeichert und zur Anlage neuer Clients im Server-Container verwendet werden.

Hinweis: Dieser spezielle Client tut nichts Böses und ist nur für die GTM-Vorschau beim manuellen Absetzen von Test-Requests aus dem Browser relevant.

Zur Erklärung: Im Verlauf der Entwicklung der Vorlagen werden gelegentlich Testdaten erzeugt, in dem manuelle Anfragen an den Server gestellt werden. Jedes Mal, wenn dazu im Browser ein Request manuell aus der Adresszeile an den Tag Server abgesetzt wird, ruft dieser auch ein Icon (favicon.ico) ab. Diese Requests sieht man inzwischen zwar nicht mehr im Debugger, aber dennoch beantwortet der Client Abfragen des Icons mit einem Symbol: Dem Logo des Klingonischen Imperiums.



Das ist der Neigung des Autoren zu Star Trek und seinem Humor geschuldet. Den Client im eigenen ssGTM zu installieren, um erste Blicke in den Vorlageneditor zu werfen, sei explizit empfohlen ;)

Auf diese Weise können bestehende Vorlagen für Clients, Tags und Variablen in den eigenen Container übernommen und dort genutzt werden. Solange es für die eigenen Anforderungen also bereits fertige Lösungen gibt, muss kein eigener Code geschrieben werden. Individuelle Anforderungen hingegen brauchen üblicherweise ebenso individuelle Lösungen. Genau das wird nun anhand von Beispielen in den folgenden Kapiteln gezeigt.

Ziele / Beispielszenarien

Die nächsten Abschnitte befassen sich mit den Grundlagen und der konkreten Entwicklung von Templates für Tags, Clients und Variablen. Die Reihenfolge orientiert sich an den typischen Anforderungen, die beim Aufbau eines serverseitigen Taggings mit dem ssGTM entstehen.

Dabei sollen folgende Anforderungen abgedeckt werden, die ohne angepasste Vorlagen nicht umsetzbar sind:

Tag: Matomo-Anbindung

Matomo ist als First Party Webanalyse bei entsprechender Konfiguration in der Lage, auch ohne Cookies und ohne explizite Zustimmung betrieben zu werden [31]. Eine Einschätzung, die von vielen (aber nicht allen) Datenschützern geteilt, wird. Aus diesem Grund ist Matomo zur “Ersatz-Datensammlung” in Zeiten lückenhafter Datensammlung für Google Analytics aufgrund fehlender Zustimmung zum Tracking sehr beliebt.

Üblicherweise wird dazu im Client neben dem Google Analytics-Trackingcode auch ein weiterer Code für Matomo implementiert.

Im Beispiel wird - beschränkt auf Seitenaufrufe - die Weitergabe aller empfangenen Hits per Tag Template an Matomo umgesetzt. Das Ergebnis ist ein guter Startpunkt für einen eigene, vollständige Matomo-Anbindung und zeigt alle wesentlichen Konzepte zur Tag-Entwicklung auf.

Client: Eigenes Datenformat nutzen

Im Client-Beispiel wird ein eigenes Datenformat als Datenquelle für den ssGTM verwendet. Die Requests sollen entgegengenommen, in das Standard-Event-Modell übernommen und neben der Weitergabe an Google Analytics (Universal Analytics) speziell für die Nutzung in Matomo mittels des zuvor erstellten Tags optimiert werden.

Variable: Klick IDs extrahieren

Die Extraktion einer Klick ID aus der URL einer aufgerufenen Seite ist i. d. R. kein Hexenwerk. Am Server steht diese URL aber nur komplett als Feld im Event-Modell zur Verfügung. Die Extraktion einzelner Parameter daraus ist mit Bordmitteln nicht möglich. Zudem gibt es inzwischen mehr als einen Parameter, mit dem Google Ads eine Klick ID übergibt, was diese spezielle Aufgabe noch etwas komplexer macht.

Obschon das Problem mit existierenden Community Gallery Templates lösbar ist, wird im Beispiel eine auf diese konkrete Aufgabe spezialisierte Variable erstellt.

Daten anreichern

Mit der Möglichkeit des Abrufs von Daten von APIs oder dem Lesen und Schreiben von Informationen zur Weitergabe an Tags befasst sich ein eigenes Kapitel. Hier finden sich ebenso Beispielcodes zur Nutzung von Promises für Datenanreicherungs-Variablen und Firestore als Persistenz am Beispiel eines wechselnden Salts für Hash-Funktionen.

Bonus: Tag Template für client-side Google Tag Manager

Da sich Vorlagen für den Google Tag Manager für beide Varianten sehr ähnlich sind, wird im Bonuskapitel eine Tag-Vorlage erstellt, die im “normalen” Tag Manager im Browser dazu eingesetzt werden kann, Tracking-Requests in genau dem Format zu erzeugen, welches im Kapitel zum Client-Template eingesetzt wird. Damit kann über das eigene Format ein komplettes Tracking vom Browser über den ssGTM bis zu Facebook, Google Analytics oder andere Dienste aufgebaut werden.

Umfang: die berühmten 80%

Um den Prozess der Entwicklung zu demonstrieren und zugleich den Umfang dieses Buchs nicht zu sprengen, wird jedes Beispiel gewisse Abstriche im Konfigurations- und Funktionsumfang machen. Das Ziel ist nicht ein einsatzfertiges Template für Matomo oder eigene Request-Formate, sondern das Verständnis für die Vorgehensweise und wesentlichen Konzepte bei der Template-Entwicklung. Dazu reichen üblicherweise 80% des Funktionsumfangs, der vergleichbar schnell hergestellt werden kann. Am Beispiel des Tags: Seitenaufrufe absetzen muss für die Demonstration reichen. Sollen auch Events, E-Commerce & Co. verarbeiten werden, muss der Rest in Eigenregie nach gleichem Vorbild implementiert werden.

Als Startpunkt sind die Beispiele jedoch gut geeignet. Wer einen Dienst wie z. B. *Plausibe Analytics* [30] mit Daten füttern will, muss zwar andere Parameter auf anderem Weg zusammenstellen und versenden, das Prinzip bleibt aber gleich.

Wer sich die Mühe macht, die für eine typische Anwendung erforderliche Arbeit in ein eigenes Template zu stecken, sollte das Ergebnis unbedingt auch in die Community Gallery einstellen. Die Hürden sind gering und der Nutzen für andere, die das gleiche Problem am ssGTM lösen wollen, sehr hoch. Daher: Nur Mut! Dazu folgen am Ende dieses Buchs noch weitere aufmunternde Details.

Achtung: Dies ist kein JavaScript Kurs!

Die Entwicklung der Vorlagen aus den Beispielen wird zwar im Detail und mit kommentiertem Code gezeigt, dennoch sind ein Grundverständnis für die Funktionsweise des ssGTM und rudimentäre JavaScript-Kenntnisse erforderlich, um die Beispiele nachvollziehen zu können.

Bonus: “Luke Logbridge Pixel” Vorlage für den clientseitigen Tag Manager

In diesem Bonuskapitel wird auf Basis des theoretischen Tracking-Formats für den *Luke Logbridge* Client am Server die Grundlage für einen realen Einsatz auf einer Website gelegt.

Das ist mit dem gewonnenen Wissen über die Funktionsweise und Entwicklung von Templates am Server erfreulicherweise kaum mit großen Umstellungen verbunden. Denn fast alle Prinzipien, die die vorangegangenen Kapitel im Umfeld des serverseitigen Tag Managers zur Entwicklung eigener Vorlagen vorgestellt wurden, gelten auch im Browser.

Bei allen Unterschieden des Einsatzzwecks von GTM im Browser und am Server, geht es doch auf beiden Seiten um Tracking. Zudem gibt es viele Gemeinsamkeiten bei der Oberfläche für beide Container-Typen.

Die Gemeinsamkeiten beinhalten auch den Vorlageneditor und alles, was damit zusammenhängt und in den vorherigen Kapiteln beschrieben ist:

- JavaScript in der Sandbox zur Entwicklung
- APIs und Berechtigungen
- Felder zur Konfiguration

Die Unterschiede zwischen beiden Arten von Templates betreffen vor allem die “Lebensbedingungen” beider Container. Im ssGTM dreht sich alles um den Empfang von Tracking-Daten, deren Verarbeitung und Weitergabe. Im Ökosystem eines Servers.

Clientseitige Vorlagen existieren im Browser. Hier besteht Zugriff auf alle Informationen, die zur “Laufzeit” einer Website verfügbar sind. Dazu gehören DOM Eigenschaften, dataLayer, Cookies und Browser-seitiger Speicher wie localStorage etc..

Die Hauptaufgabe des GTM im Browser im Zusammenhang mit Tracking besteht aus der Sammlung der erforderlichen Informationen, die für das Tracking eines Ereignisses wie einem Seitenaufruf erforderlich sind. Dabei wird er i. d. R. durch den dataLayer unterstützt und mit Informationen versorgt, die für das Tracking nützlich sind.

Der Satz an zur Verfügung stehenden APIs unterscheidet sich daher vom Server, was durch die unterschiedlichen Aufgabenbereiche und Laufzeitbedingungen bedingt ist. Überschneidungen sind aber dennoch durchaus vorhanden. Genug jedenfalls, um sich an ein Tag Template für den Browser zu wagen.

Schritt 1: Testumgebung erstellen

Bitte nicht vergessen: Voraussetzung für alle folgenden Schritte ist ein clientseitiger "*Web*"-Container für den Google Tag Manager. Alles, was bisher in diesem Buch passiert ist, wurde in einem Container vom Typ "*Server*" umgesetzt.

Daher ab hier den Container wechseln und entweder einen bestehenden und auf einer realen Website eingebundenen Container wählen oder einen komplett neuen Container anlegen.

Im Fall eines bestehenden Containers sollte dringend ein neuer Workspace verwendet werden, in dem die Änderungen isoliert und ansatzweise sicher davor sind, versehentlich veröffentlicht zu werden. Wenn ein neuer Container verwendet wird, kann dieser mit einer Chrome-Erweiterung wie dem *GTM Helper* [26] in eine beliebige bestehende Website "injiziert" werden, um das Tag in der Vorschau des Containers zu erproben, ohne diesen auf der Website tatsächlich verbauen zu müssen.

Schritt 2: Neue Tag-Vorlage im Tag Manager anlegen

Im Web-Container gibt es ebenfalls einen Bereich "*Vorlagen*", in dem unter "*Tag-Vorlagen*" ein neuer Eintrag angelegt wird. Der eingeblendete Vorlageneditor entspricht 1:1 dem aus dem Server-Container. Wie dort fängt alles mit einem Namen an, der als Bezeichnung auf der Seite "*Informationen*" eingegeben wird.

Da das neue Tag dazu dienen soll, Requests im "*Luke Logbridge*" Format an einen ssGTM zu senden, ist "*Luke Logbridge Pixel*" ein guter Name. Danach kann direkt auf den Reiter "*Code*" gewechselt werden, um eine erste Fassung des Templates zu erstellen und zu erproben.

Den Anfang machen - das ist nun schon fast Routine - eine Liste von APIs, die im Tag verwenden werden. Wie bei den Server-Templates reichen für eine erste Fassung wenige APIs, die im Verlauf des Ausbaus noch ergänzt werden.

```
const getUrl = require('getUrl');
const getReferrerUrl = require('getReferrerUrl');
const readTitle = require('readTitle');
const encodeUriComponent = require('encodeUriComponent');
const sendPixel = require('sendPixel');
```

Da die Namen der APIs sprechend sind, ist deren Zweck leicht abzulesen. Für viele Daten, die zur Erstellung eines Tracking-Hits benötigt werden, ist eine eigene API zuständig, die den Zugriff

auf die jeweilige Information erlaubt. So kann sehr granular bestimmt werden, welche Daten ein Tag im GTM lesen und verarbeiten kann - genau wie bei Server-Templates.

Aus der Tabelle der Parameter des Abschnitts "*Das Luke Logbridge Format*" werden z. B. die folgenden Informationen benötigt bzw. werden vom Client am Server verarbeitet. Dabei sind auch Werte, die in der Ergänzung für Google Analytics später hinzugefügt wurden.

- lc = "Location" / URL
- tl = Seitentitel
- rf = Referrer

Eben diese Daten werden über die ersten drei APIs eingelesen. Die vierte dient dazu, die Werte für die Verwendung als URL Parameter zu codieren (was auch schon am Server genutzt wurde). Die letzte API wird den Hit an den eigenen Endpunkt senden.

Unter Verwendung der Best Practices aus dem gleichnamigen Kapitel wird in der ersten Code-Fassung mittels einer *addParam()* betitelten Funktion aus den durch die o. a. APIs gewonnenen Informationen (und einigen Konstanten) ein erster Tracking-Hit aufgebaut und versendet.

```
const getUrl = require('getUrl');
const sendPixel = require('sendPixel');
const getTimestampMillis = require('getTimestampMillis');
const encodeUriComponent = require('encodeUriComponent');
const getReferrerUrl = require('getReferrerUrl');
const readTitle = require('readTitle');

var addParam = function(name, value) {
  if (value) return "&" + name + '=' +
  encodeUriComponent(value.toString()); else return "";
};
var url = data.endpointUrl ||
"https://gtm-xxxxx-yyyyy.uc.r.appspot.com/luke_logbridge";

//Parameter-String erstellen und GET Request senden
if (!url)
  data.gtmOnFailure();
else {
  var en = data.eventName || "PageView";
  var params = "?ev=" + en +
      addParam('tl', readTitle()) +
      addParam('rf', getReferrerUrl()) +
      addParam('lc', getUrl());

  sendPixel(url + params, data.gtmOnSuccess(), data.gtmOnFailure);
}
```

Da es noch keine Einstellungen gibt, die die URL des Endpunkts und den Event-Namen über das *data* Objekt definierbar machen, sind für beide Variablen Fallback-Werte definiert, die einen Test eines PageViews am eigenen Server-Endpunkt ermöglichen. Dass es noch keine Einstellungen gibt, stört die Ausführbarkeit des Codes nicht. Der Grund ist im Schritt 2 des Kapitels zum Variablen-Template erläutert.

Hinweis: Für den Test den Standard-Endpunkt als Fallback (rot markiert) für die *url* im Code unbedingt gegen die Adresse des eigenen Servers austauschen!

Gibt es keinen eigenen Endpunkt, kann ersatzweise die URL https://httpbin.org/anything genutzt werden. Dahinter steckt ein Endpunkt, der alle Request-Daten in der Antwort wieder zurücksendet. Er wurde schon im Kapitel zur serverseitigen Tag-Entwicklung eingesetzt.

Bevor der Code gespeichert werden kann, sind wieder Berechtigungen zu definieren, die die eingebundenen APIs betreffen. Dem Tag-Beispiel wird hier maximale Freiheit eingeräumt. Im Gegensatz zu den anderen Templates aus diesem Buch sind aber die vergebenden Berechtigungen mit "*Alle*" bzw. "*Beliebige HTTPS URLs*" tatsächlich passend für den Live-Betrieb. Denn es sollen die kompletten URLs der Seite und des Referrers genutzt werden und der Hit an jeden (später in den Einstellungen definierbaren) Endpunkt gesendet werden können.

Tag anlegen

Nach dem Speichern der Vorlage kann im Bereich "*Tags*" des GTM ein neues Tag unter Verwendung der Vorlage erzeugt werden. Als Trigger ist "*Alle Seiten*" für den Testbetrieb ausreichend.

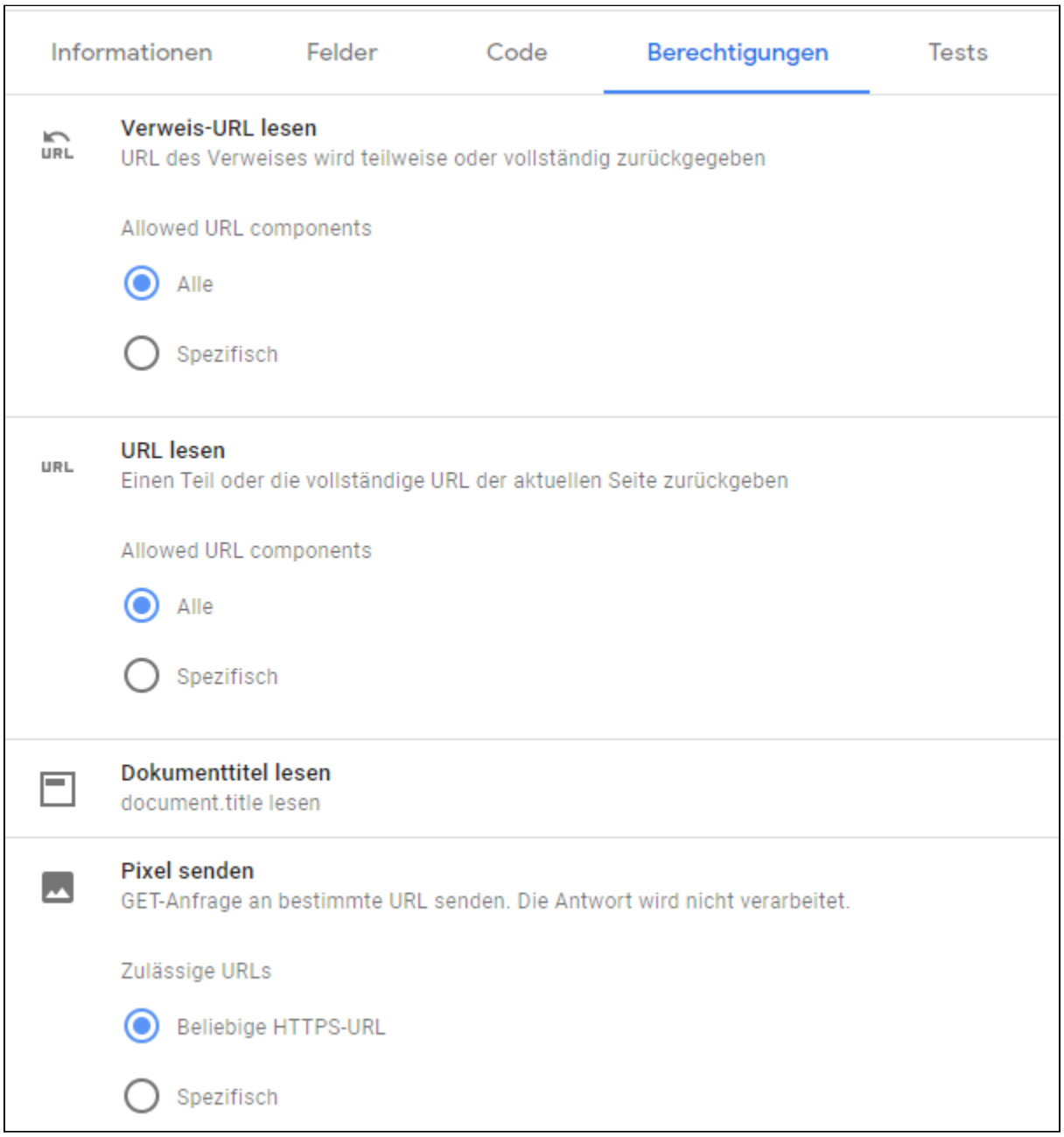


Abbildung 36: Maximale Freiheit für die APIs des Tags

Wird der Tag Manager nun in die Vorschau versetzt, zeigt der Debugger des Tag Assistant in einem separaten Reiter beim "*Container Loaded*" Event das Feuern des neuen Tags. Die Rückmeldung des Tags an den GTM sollte "*Succeeded*" sein.

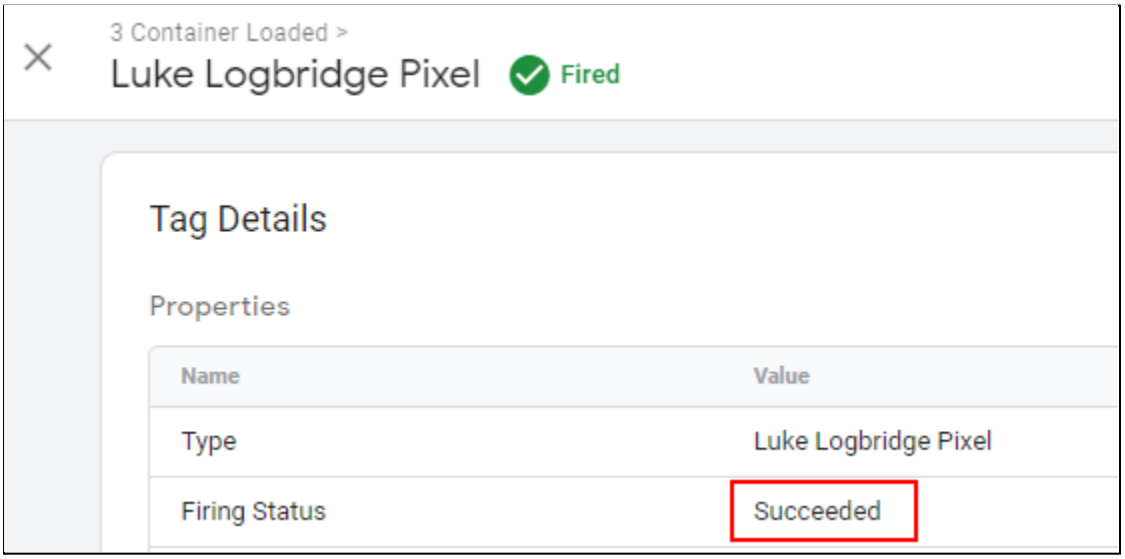


Abbildung 37: Erfolgsmeldung im Debugger

Diese Rückmeldung basiert auf dem Aufruf von `data.gtmOnSuccess()` im Erfolgsfall der `sendPixel()` Funktion am Ende des Codes. Das Pendant `data.gtmOnFailure()` wird entweder dann aufgerufen, wenn beim Versand etwas schiefgelaufen ist (durch die Zuweisung der Funktion beim Fehler-Callback) oder wenn keine *url* vorhanden ist (was faktisch nie eintreten sollte).

In der Dokumentation der clientseitigen APIs [28] ist diese Zuweisung ebenfalls erklärt. Zudem ist im Kapitel zum serverseitigen Tag das Prinzip der asynchronen Verarbeitung aller Funktionen beschrieben, die die Generierung von Anfragen und Antworten betreffen. Genauso wird auch hier beim Versand des ausgehenden Requests durch das Tag asynchron gearbeitet.

Mehr Rückmeldung ist im Debugger leider nicht zu sehen. Es sei denn, es ist etwas Unerwartetes während der Laufzeit geschehen. Evtl. Fehlermeldungen durch das Tag wären im Debugger unter *Errors* sichtbar.

Browser-Konsole nutzen

Damit kontrolliert werden kann, was mit dem Request geschehen ist, ist die Konsole des Browsers für Web-GTM Tags die bessere Wahl. Dort landet auch alles, was im Tag Template z. B. mittels *logToConsole()* protokolliert wird. Zur Erprobung kann an beliebiger Stelle innerhalb des Template-Codes die folgende Anweisung hinzugefügt werden.

```
require('logToConsole')('logToConsole - Meldungen landen auch
hier.');
```

Nach Speichern des Templates und Aktualisierung der GTM-Vorschau zeigt sich die Meldung im Browser, wenn die Konsole (in z. B. Chrome via STRG + UMSCHALT + J) geöffnet wird.

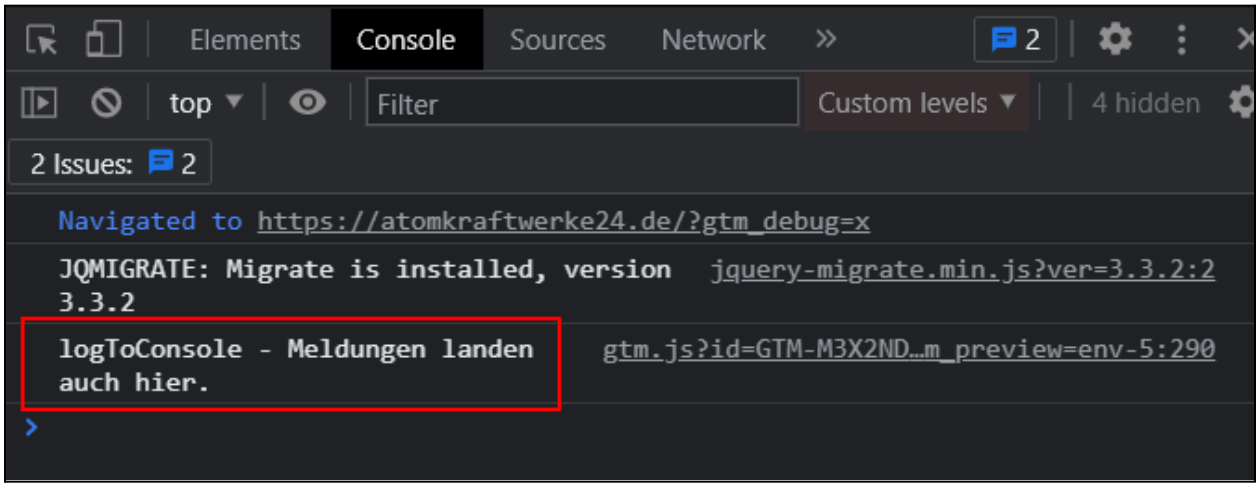


Abbildung 38: Meldungen in der Konsole des Browsers

Sieht man zudem eine Fehlermeldung, die den angegebenen Endpunkt für das Tag betrifft, ist der eigene ssGTM evtl. aktuell nicht erreichbar. Bei einem Server im Testbetrieb liegt das daran, dass er derzeit nicht in der Vorschau betrieben wird und daher “schläft”.

Unabhängig vom Ergebnis muss der Request sich aber beim Wechsel auf auf “Network”zeigen und in der Liste der Requests, wenn nach “*luke*” gesucht wird. Die Details der Fundstelle können durch Klick eingeblendet werden. Beim “*Request*” ganz unten findet sich eine Liste der übergebenen Parameter.

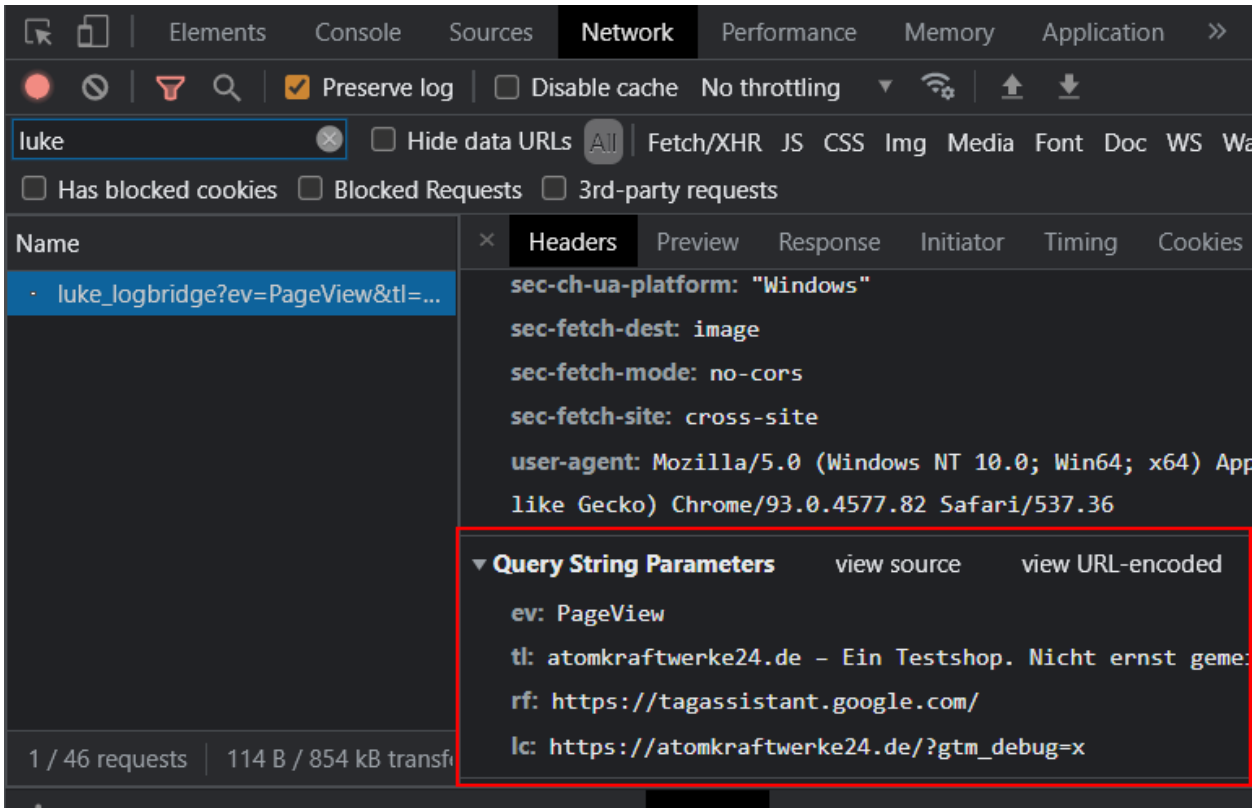


Abbildung 39: Erfolgreich versandter Tracking-Request an den eigenen ssGTM

Ebenso ist oberhalb der Liste zu sehen, was an Headern im Request versendet wurde… und was der ssGTM geantwortet hat.

Ergebnis am ssGTM

Viel interessanter ist zu sehen, was in der Vorschau des serverseitigen Containers vom selbst erstellten *Luke Logbridge* Client aus dem Request gemacht wurde. Der Client sollte den Request verstanden und in ein Event umgewandelt haben. Zu diesem Event werden auf dem Reiter *Event-Data* die vom Tag im Browser gesendeten und in das Event-Modell übersetzten Daten angezeigt.

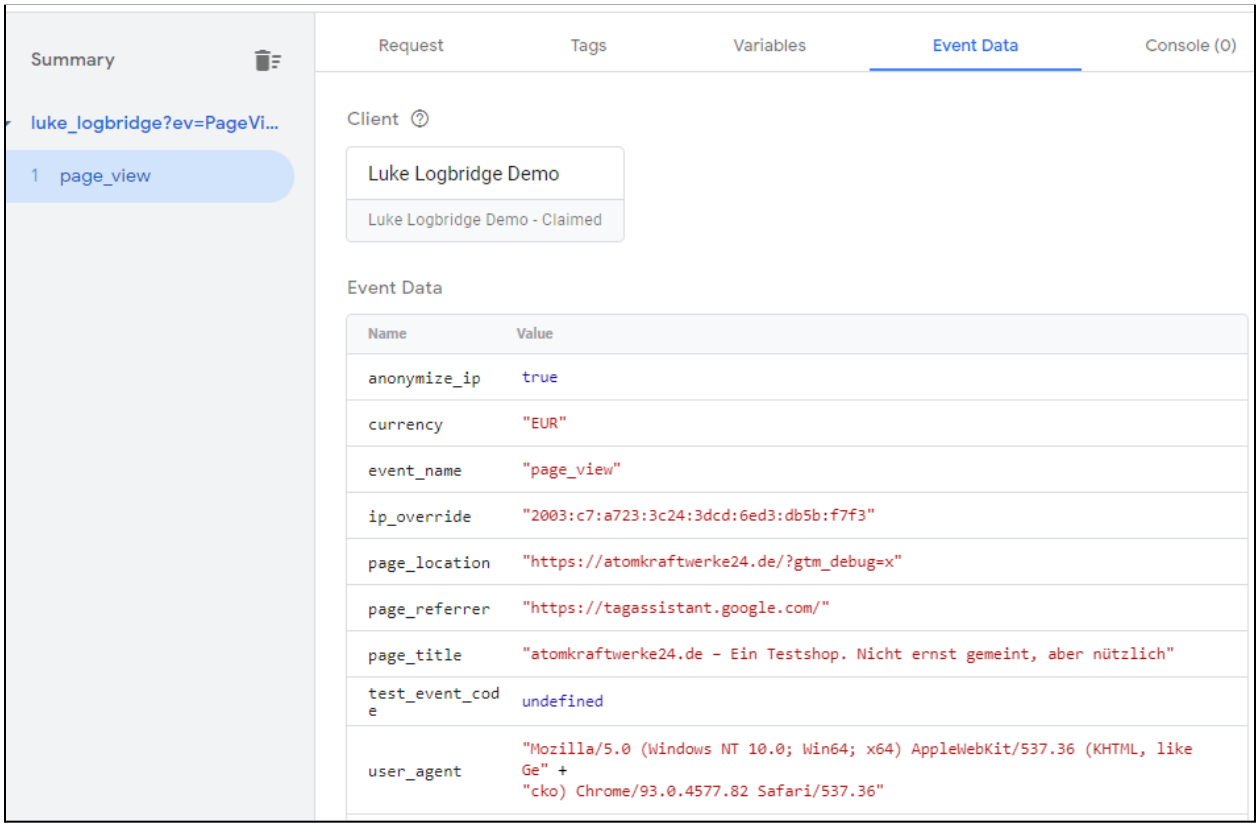


Abbildung 40: Ergebnis am ssGTM der bisher gesendeten Daten im Event-Modell

Damit wurden erstmals Daten im eigenen Format auf der Website erhoben, an den eigenen Server gesendet und dort verarbeitet. Wurden am Server noch Tags zur Übergabe an Facebook oder Analytics aus den vorherigen Kapiteln getriggert, haben die Daten den ssGTM auch zielgerichtet verlassen.

Glückwunsch! Der Tracking-Prozess ist erstmals auf Basis von komplett selbst erstellten Vorlagen durchlaufen. Das darf gefeiert werden!

Ergebnis bei Verwendung des Test-Endpunkts

Wurde kein eigener ssGTM für den Empfang verwendet, sondern der Test-Endpunkt unter https://httpbin.org/anything, ist eine Kontrolle im Network Tab der Konsole ebenfalls - bedingt - möglich. Dort sieht man sowohl den Statuscode 200 als auch die Details des ausgehenden Requests. Soweit gibt es keinen Unterschied zum oben beschriebenen Fall bei Nutzung eines ssGTM.

Um die Details der Antwort zu betrachten, kann der Eintrag mit dem Aufruf aus der Liste der Requests doppelt angeklickt werden. In einem neuen Tab wird die URL erneut abgerufen und zeigt die Antwort des Test-Servers. Ebenso wie beim Testbetrieb des serverseitigen Tags werden die Parameter separat im “Ergebnis-Objekt” unter `args` aufgelistet.


```
{
  "args": {
    "ev": "PageView",
    "lc": "https://atomkraftwerke24.de/?gtm_debug=x",
    "rf": "https://tagassistant.google.com/",
    "tl": "atomkraftwerke24.de \u2013 Ein Testshop"
  },
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
ge/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
...
}
```


Der Versand funktioniert also - nun fehlen noch die erforderlichen Einstellungen und weitere Daten, die von Tag-Vorlage verarbeitet werden sollen.

Schritt 3: Ausbau und Einstellungen

Um den Client am Server mit allen Daten zu bedienen, die dort vom derzeitigen Stand des Codes verarbeitet werden, fehlen der Tag-Vorlage noch einige Parameter. Diese sind...