GTM Glossar

Google Tag Manager — Begriffe & Definitionen

analytics.ewerkstatt.com
76 Einträge • alphabetisch sortiert

Die Welt des Trackings und insbesonders die Welt des Google Tag Managers steckt voller Begriffe, die man aus dem alltäglichen Leben heraus nicht kennt. Wer sich mit dem Google Tag Manager beschäftigt, der sollte sich dagegen mit diesen Begrifflichkeiten auseinandersetzen. Das Glossar soll Dich dabei unterstützen.

Alle Einträge stammen aus dem Online-Glossar auf **analytics.ewerkstatt.com** und sind hier alphabetisch geordnet. Die Online-Fassung wird laufend ergänzt.

Register

#

_fbp und _fbc

`_fbp` und `_fbc` sind die beiden First-Party-Cookies, die das Meta Pixel setzt. Sie liegen auf der eigenen Domain, nicht auf einer von Meta.

`_fbp` ist eine Browser-Kennung, die das Pixel beim ersten Laden erzeugt. Sie hat die Form `fb.1.<Zeitstempel>.<Zufallszahl>` und ordnet mehrere Besuche demselben Browser zu.

`_fbc` entsteht nur, wenn jemand über eine Meta-Anzeige kommt. Die Anzeige hängt den Parameter `fbclid` an die Ziel-URL, das Pixel schreibt ihn in das Cookie, ebenfalls mit dem Präfix `fb.1.` und einem Zeitstempel. Ohne dieses Cookie geht die Verbindung zwischen Klick und späterer Conversion verloren, sobald die URL beim ersten Seitenwechsel wegfällt.

Für die Conversions API sind beide Werte wichtig. Serverseitig gemeldete Ereignisse ohne `_fbp` und `_fbc` lassen sich deutlich schlechter zuordnen, und die Übereinstimmungsqualität im Events Manager fällt entsprechend aus.

Im Container lassen sich beide über eine Variable vom Typ „Eigenes Cookie“ auslesen und an ein serverseitiges System weiterreichen. Gesetzt werden sie erst nach der Einwilligung, weil das Pixel bis dahin nicht lädt. Ein leeres Cookie ist also kein Fehler, sondern der erwartete Zustand vor der Zustimmung.

A

Änderungsverlauf

Der **Änderungsverlauf** protokolliert, wer wann was im Container geändert hat. Zu finden ist er unter Verwaltung → Containeränderungsverlauf.

Erfasst werden alle Aktionen auf Objektebene: angelegt, geändert, gelöscht, jeweils mit Nutzerkonto und Zeitstempel. Der Verlauf reicht mehrere Monate zurück und lässt sich nach Nutzer und Objekttyp filtern.

Er beantwortet die Frage, die nach jedem unerklärlichen Datenbruch kommt: Wurde am fraglichen Tag etwas am Container geändert? Zeigt der Verlauf nichts, liegt die Ursache woanders, meist an der Website oder am Consent-Management-Tool.

Zurücksetzen lässt sich aus dem Änderungsverlauf nichts. Dafür gibt es die Versionen, in denen sich ein früherer Stand wiederherstellen lässt.

Arbeitsbereich

Ein **Arbeitsbereich** (Workspace) ist ein Entwurfsstand des Containers. Alle Änderungen an Tags, Triggern und Variablen landen zuerst dort und wirken sich auf die Website erst aus, wenn daraus eine Version erstellt und veröffentlicht wird.

Jeder Container hat einen Standard-Arbeitsbereich. In der kostenlosen Version sind bis zu drei gleichzeitig möglich, in Tag Manager 360 beliebig viele. Mehrere Arbeitsbereiche sind dafür gedacht, dass verschiedene Personen an verschiedenen Themen arbeiten, ohne sich gegenseitig zu veröffentlichen.

Wird in einem anderen Arbeitsbereich veröffentlicht, zeigt der eigene einen Hinweis auf Änderungen im Container und verlangt eine Aktualisierung. Betreffen die Änderungen dieselben Objekte, entsteht ein Konflikt, der von Hand aufgelöst werden muss. Bei größeren Umbauten ist es deshalb praktischer, nacheinander zu arbeiten.

Der Vorschaumodus zeigt immer den Stand des Arbeitsbereichs, in dem man gerade ist, nicht den veröffentlichten Stand.

Ausgelöste und nicht ausgelöste Tags

Im Vorschaumodus teilt der Reiter „Tags“ alle Tags des Containers in zwei Gruppen: **ausgelöst** und **nicht ausgelöst**, jeweils bezogen auf das links ausgewählte Ereignis.

Der Wert liegt in der zweiten Gruppe. Wer ein Tag dort anklickt, sieht, welche Bedingung nicht erfüllt war. Das ist der schnellste Weg zur Antwort auf die Frage, warum ein Tag nicht feuert.

Drei Gründe kommen in Frage. Der Trigger traf nicht zu, dann zeigt die Ansicht die fehlgeschlagene Bedingung mit dem tatsächlichen Wert. Ein Ausnahme-Trigger hat blockiert. Oder die zusätzliche Einwilligungsprüfung ist fehlgeschlagen, was als eigener Hinweis erscheint.

Ein ausgelöstes Tag bedeutet nur, dass der Container es ausgeführt hat. Ob der Empfänger die Daten auch angenommen hat, zeigt erst der Netzwerk-Tab oder die DebugView der Zielplattform.

Auslöseoptionen

Die **Auslöseoptionen** (Tag-Auslöseoptionen) begrenzen, wie oft ein Tag feuern darf. Sie stehen in den erweiterten Einstellungen und kennen drei Werte.

- *Unbegrenzt*: Das Tag feuert jedes Mal, wenn ein Trigger zutrifft. Standard bei neuen Tags.
- *Einmal pro Ereignis*: Das Tag feuert höchstens einmal je Ereignis in der Datenschicht, auch wenn mehrere Trigger darauf passen.
- *Einmal pro Seite*: Das Tag feuert höchstens einmal pro Seitenaufruf.

„Einmal pro Seite“ ist die richtige Wahl für Basis-Tags, die auf ein Ereignis des Consent-Management-Tools hören. Solche Ereignisse feuern erneut, wenn jemand seine Einstellungen ändert, und ohne Begrenzung entstünde dabei ein zweiter Seitenaufruf in GA4.

Bei Single-Page-Anwendungen ist zu beachten, dass „Seite“ den tatsächlichen Dokumentaufruf meint. Ein virtueller Seitenwechsel setzt den Zähler nicht zurück.

Die Auslöseoptionen sind kein Mittel gegen Doppelzählung durch zwei Trigger an einem Tag. Feuern beide beim selben Vorgang, aber bei verschiedenen Ereignissen, greift die Begrenzung „Einmal pro Ereignis“ nicht.

Ausnahme-Trigger

Ein **Ausnahme-Trigger** verhindert, dass ein Tag feuert. Er wird im Tag unter „Ausnahme“ eingetragen und sticht jeden auslösenden Trigger.

Technisch ist das ein gewöhnlicher Trigger. Ausnahme wird er erst dadurch, an welcher Stelle im Tag er steht. Derselbe Trigger kann bei einem Tag auslösen und bei einem anderen blockieren.

Übliche Anwendungen sind interner Traffic anhand eines Cookies oder der IP-Adresse, Testumgebungen anhand des Hostnamens und einzelne Seiten, die aus der Messung fallen sollen.

Nicht verwendet werden sollten Ausnahme-Trigger für die Einwilligung. Vor dem Consent Mode war ein blockierender Trigger mit dem Muster `.*` der übliche Weg, heute gehört das in die zusätzlichen Einwilligungsprüfungen. Beides parallel zu pflegen erzeugt eine Logik, die niemand mehr durchschaut, und führt zu Tags, die aus unklaren Gründen nicht feuern.

Im Vorschaumodus erscheint ein so blockiertes Tag unter „Nicht ausgelöste Tags“ mit dem Hinweis auf die zutreffende Ausnahme.

B

Benutzerdefinierte Variablen

Benutzerdefinierte Variablen legt man selbst an, im Gegensatz zu den integrierten Variablen, die nur ein- und ausgeschaltet werden. Sie stehen im Bereich Variablen unterhalb der integrierten.

Wichtige Typen

- Datenschicht-Variable: liest einen Wert aus der Datenschicht, der häufigste Typ überhaupt.
- Benutzerdefiniertes JavaScript: berechnet einen Wert in einer anonymen Funktion mit `return`.
- Konstante: ein fester Wert, typischerweise eine Mess- oder Conversion-ID.
- Nachschlagetabelle und Tabelle für reguläre Ausdrücke: bilden Eingabewerte auf Ausgabewerte ab.
- Eigenes Cookie, URL, DOM-Element, JavaScript-Variable, Zufälliges Element aus dem DOM.
- Google Tag-Einstellungen und Konfigurationseinstellungen: bündeln Parameter, die mehrere Tags gemeinsam nutzen.

Die Typen DOM-Element und JavaScript-Variable greifen auf die Seite selbst zu und sind fragil, weil sie von Markup oder globalen Objekten abhängen. Sie sind die letzte Wahl, wenn ein Wert nicht über die Datenschicht bereitgestellt werden kann.

Eine gebräuchliche Namenskonvention benennt den Typ mit: `DLV - ecommerce.value` für Datenschicht-Variablen, `CJS - item_ids` für eigenes JavaScript, `Const - GA4 Mess-ID` für Konstanten.

Benutzerdefinierte Vorlage

Eine **benutzerdefinierte Vorlage** (Custom Template) ist ein selbst gebauter Tag- oder Variablen-Typ. Angelegt wird sie im Bereich Vorlagen.

Eine Vorlage besteht aus zwei Teilen: der Oberfläche, in der die Eingabefelder definiert werden, und dem Code, der beim Auslösen läuft. Der Code ist kein normales JavaScript, sondern Sandboxed JavaScript, eine eingeschränkte Fassung ohne Zugriff auf `window`, `document` und die üblichen Browser-APIs. Alles, was die Vorlage tun darf, läuft über eingebaute Funktionen wie `injectScript`, `sendPixel`, `setCookie` oder `copyFromDataLayer`, und jede davon muss in den Berechtigungen freigeschaltet sein.

Der Aufwand lohnt sich, wenn dasselbe Stück Tracking-Code in vielen Containern gebraucht wird oder wenn Kolleginnen und Kollegen ein Tag pflegen sollen, ohne Code anzufassen. Für einen einmaligen Sonderfall ist Benutzerdefiniertes HTML der schnellere Weg.

Eigene Vorlagen lassen sich als Datei exportieren und in andere Container importieren oder in der Vorlagengalerie veröffentlichen.

Benutzerdefiniertes HTML

Benutzerdefiniertes HTML ist ein Tag-Typ, der beliebiges HTML und JavaScript auf der Seite ausführt. Er ist der Rettungsanker für alles, wofür es keine Vorlage gibt.

Der Code läuft direkt im Seitenkontext, ohne Sandbox. Damit hat er vollen Zugriff auf `window` und `document`, bringt aber auch alle Nachteile mit: keine eingebaute Einwilligungsprüfung, keine Berechtigungskontrolle, und ein Fehler im Code kann die Seite beeinträchtigen. Die Einwilligung muss hier über zusätzliche Einwilligungsprüfungen geregelt werden, sonst feuert das Tag uneingeschränkt.

Für Meta, LinkedIn, Clarity und Hotjar gibt es Vorlagen in der Vorlagengalerie, die diese Punkte abdecken. Benutzerdefiniertes HTML ist dort nur nötig, wenn die Vorlage etwas nicht kann, etwa eine Ereignis-ID zur Deduplizierung mitzugeben.

Die Option „Dokumentschreibvorgang unterstützen“ aktiviert `document.write` für alte Skripte. Sie bricht bei asynchron geladenen Seiten und sollte aus bleiben.

In einem Audit sind viele Custom-HTML-Tags ein Hinweis auf gewachsene Technik. Jedes einzelne ist Code, den niemand testet und der bei einem Browser-Update ausfallen kann.

Benutzerdefiniertes JavaScript

Eine Variable vom Typ **Benutzerdefiniertes JavaScript** berechnet ihren Wert selbst. Sie ist immer eine anonyme Funktion, die sofort aufgerufen wird und einen Wert zurückgibt:

```
function() {
  // read items from the data layer, return their IDs
  var items = {{DLV - ecommerce.items}} || [];
  return items.map(function(i) { return i.item_id; }).join(',');
}
```

Andere Variablen lassen sich als `{{Name}}` einsetzen, auch mitten im Code.

Regeln

Der Tag Manager wertet Variablen mehrfach pro Ereignis aus. Die Funktion darf deshalb keine Seiteneffekte haben: kein Push in die Datenschicht, kein Zähler, kein Request. Sie muss außerdem mit fehlenden Daten umgehen können und `undefined` zurückgeben, statt einen Fehler zu werfen, der das ganze Ereignis abbricht.

ES6-Syntax akzeptiert der Container inzwischen. Meldet er beim Speichern oder im Vorschaumodus einen Kompilierfehler, hilft ein Umschreiben auf ES5.

Nicht zu verwechseln mit dem Tag-Typ Benutzerdefiniertes HTML, der Code ausführt, statt einen Wert zu liefern, und mit der benutzerdefinierten Vorlage, die in einer Sandbox läuft.

Berechtigungen

Berechtigungen im Google Tag Manager werden auf zwei Ebenen vergeben: für das Konto und für jeden einzelnen Container.

Auf Kontoebene gibt es nur zwei Stufen. „Nutzer“ darf das Konto sehen, „Administrator“ darf zusätzlich Nutzer verwalten und Container anlegen.

Containerebene

- *Kein Zugriff*: Der Container taucht in der Liste nicht auf.
- *Lesen*: Ansehen von Tags, Triggern und Variablen, keine Änderungen.
- *Bearbeiten*: Änderungen im Arbeitsbereich, aber kein Veröffentlichen.
- *Genehmigen*: zusätzlich Versionen erstellen.
- *Veröffentlichen*: alle Rechte, einschließlich Livestellung.

Die Stufe „Bearbeiten“ ist die natürliche Wahl für alle, die mitarbeiten, aber nicht allein live schalten sollen. In der Praxis wird sie selten genutzt, weil ohne Veröffentlichungsrecht jede Änderung eine zweite Person braucht.

Zugriff wird immer an ein Google-Konto vergeben, nie an eine Rolle oder ein Team. Beim Ausscheiden von Mitarbeitern oder beim Wechsel der Agentur gehört die Liste deshalb durchgesehen.

C

Consent Mode

Der **Consent Mode** (deutsch *Einwilligungsmodus*) übergibt an den Google Tag und an Google Analytics 4 die Information, ob die/der Besucher/in der Datenerhebung und der Auslieferung von Werbe-Cookies zugestimmt hat. Eingeführt 2020, aktuelle Version: **Consent Mode v2**, verpflichtend für Werbetreibende im Europäischen Wirtschaftsraum.

Übergeben werden vier Signale, jeweils mit dem Wert `granted` oder `denied`:

- `ad_storage` — Werbe-Cookies erlaubt?
- `ad_user_data` — Übermittlung von Nutzerdaten an Google für Werbung erlaubt?
- `ad_personalization` — Personalisierte Werbung erlaubt?
- `analytics_storage` — Analytics-Cookies erlaubt?

Bei fehlender Einwilligung sendet GA4 anonyme, **cookielose Pings**: keine Client-ID, keine User-ID, kein wiedererkennbarer Nutzerbezug. Fehlende Conversions und Nutzerzahlen werden durch **modellierte Conversions** bzw. modellierte Daten ergänzt — Google ergänzt also, was technisch nicht gemessen werden konnte.

Praktischer Hinweis: Der Consent Mode wird über ein Consent-Management-System (CMP) wie Cookiebot, Borlabs Cookie oder Complianz vermittelt. Ohne CMP-Integration ist Consent Mode v2 nicht datenschutzkonform einsetzbar.

Container

Der **Container** ist die Einheit, in der alle Tags, Trigger und Variablen einer Website liegen. Er wird über ein Code-Snippet eingebunden und über seine Container-ID im Format `GTM-XXXXXXX` identifiziert.

Ein GTM-Konto kann mehrere Container enthalten. Üblich ist einer pro Website. Mehrere Websites in einen Container zu legen spart Pflegeaufwand, macht aber jede Trigger-Bedingung vom Hostnamen abhängig und jede Änderung zu einem Risiko für alle beteiligten Sites.

Container-Typen

Beim Anlegen wird der Typ gewählt und bleibt danach fest: Web, iOS, Android, AMP oder Server. Der Server-Container gehört zum Server-Side Tagging und läuft in einer eigenen Umgebung, nicht im Browser.

Das Snippet

Der Web-Container besteht aus zwei Teilen: einem JavaScript-Block, der möglichst weit oben im `<head>` steht, und einem `<noscript>`-iframe direkt nach dem öffnenden `<body>`. Der zweite Teil greift nur bei deaktiviertem JavaScript und hat für GA4 keinen praktischen Nutzen.

In WordPress übernimmt das Einbinden meist ein Plugin, etwa GTM4WP oder ein Consent-Management-Tool wie Complianz oder Borlabs Cookie. Bindet zusätzlich noch das Theme den Container ein, lädt er doppelt, und jedes Tag feuert zweimal.

Import und Export

Ein Container lässt sich als JSON exportieren und in einen anderen importieren. Beim Import stehen „Zusammenführen“ und „Überschreiben“ zur Wahl. „Überschreiben“ verwirft den bestehenden Inhalt des Arbeitsbereichs und ist nur beim Neuaufsetzen sinnvoll.

Container-ID

Die **Container-ID** identifiziert einen Container eindeutig. Sie hat das Format `GTM-` gefolgt von sieben Zeichen und steht rechts oben in der Oberfläche.

Die ID ist öffentlich, sie steht im Quelltext jeder Seite, die den Container lädt. Ein Zugriff auf den Container ist damit nicht möglich, dafür braucht es eine Berechtigung. Sie lässt sich nicht ändern und nicht übertragen: Ein neu aufgebauter Container bekommt immer eine neue ID, und das Snippet auf der Website muss dann getauscht werden.

Nicht zu verwechseln mit der Mess-ID von Google Analytics 4 im Format `G-XXXXXXXXXX` und der Conversion-ID von Google Ads im Format `AW-XXXXXXXXX`. Alle drei begegnen einem im selben Container, gehören aber zu verschiedenen Systemen.

Innerhalb des Containers steht die eigene ID als integrierte Variable „Container-ID“ zur Verfügung.

Container-Snippet

Das **Container-Snippet** ist der Code, der den Container auf der Website lädt. Es besteht aus zwei Teilen.

Der erste ist ein JavaScript-Block, der möglichst weit oben in den `<head>` gehört. Er legt das Array `window.dataLayer` an, pusht das Ereignis `gtm.js` mit einem Zeitstempel und hängt ein Script-Tag auf `https://www.googletagmanager.com/gtm.js?id=GTM-XXXXXXX` in die Seite. Diese Datei enthält den kompletten veröffentlichten Container.

Der zweite Teil ist ein `<noscript>`-iframe direkt nach dem öffnenden `<body>`. Er greift nur bei deaktiviertem JavaScript und hat für GA4 keinen praktischen Nutzen, weil GA4 ohne JavaScript ohnehin nicht misst.

Einbindung in WordPress

In WordPress übernimmt das Einbinden üblicherweise ein Plugin: GTM4WP oder ein Consent-Management-Tool wie Complianz oder Borlabs Cookie, das den Container selbst lädt. Läuft beides parallel oder bindet zusätzlich das Theme den Code ein, lädt der Container doppelt und jedes Tag feuert zweimal. Zu prüfen ist das im Quelltext: Kommt `GTM-` mehr als einmal vor, liegt genau dieser Fehler vor.

Wird eine Umgebung verwendet, enthält das Snippet zusätzlich die Parameter `gtm_auth` und `gtm_preview`.

Conversion-Verknüpfung

Die **Conversion-Verknüpfung** (Conversion Linker) ist ein Tag-Typ, der Klick-Informationen aus Google-Anzeigen in ein First-Party-Cookie schreibt.

Der Hintergrund: Wer über eine Anzeige kommt, trägt Parameter wie `gclid` in der URL. Ohne dieses Tag geht die Information beim nächsten Seitenwechsel verloren, und eine Conversion, die erst zwei Klicks später passiert, lässt sich der Anzeige nicht mehr zuordnen.

Das Tag gehört auf alle Seiten, üblicherweise mit dem Trigger „Alle Seiten“, und muss vor den Conversion-Tags laufen. Konfigurieren lässt sich daran kaum etwas, außer der Unterstützung für weitere Domains bei Cross-Domain-Strecken.

Wie alle Google-Tags bringt es eine eingebaute Einwilligungsprüfung mit. Im Basic-Modus des Consent Mode bekommt es zusätzlich eine Prüfung auf `ad_storage`.

In Containern, die ausschließlich mit dem neuen Google-Tag arbeiten, ist die Funktion dort bereits enthalten. Ein separates Conversion-Verknüpfungs-Tag ist dann nicht mehr nötig, schadet aber auch nicht.

Conversions API

Die **Conversions API** (CAPI) ist Metas serverseitiger Weg, Ereignisse zu melden. Statt aus dem Browser gehen die Daten von einem Server an Meta, was sie unabhängig von Adblockern und von den Beschränkungen macht, denen Browser-Cookies unterliegen.

Gesendet wird ein Ereignis mit Zeitstempel, Ereignisname, der URL der auslösenden Seite, der Ereignisquelle und einem Block mit Nutzerdaten. Diese Nutzerdaten werden vor dem Versand gehasht, üblich sind E-Mail-Adresse und Telefonnummer, dazu die Cookie-Werte _fbp und _fbc. Je mehr davon mitkommt, desto besser die Zuordnung.

In WordPress übernimmt das meist ein Plugin. „Meta for WooCommerce“ sendet die Bestelldaten serverseitig, andere Lösungen hängen sich an die Bestellbestätigung. Ohne Plugin läuft es über einen eigenen Endpunkt oder über einen Server-Container.

Sobald CAPI und Pixel dasselbe Ereignis melden, wird die Deduplizierung zur Pflicht. Ohne sie zählt jeder Kauf zweimal.

Datenschutzrechtlich ändert der serverseitige Versand nichts an der Einwilligungspflicht. Wer ohne Zustimmung gehashte Kontaktdaten an Meta schickt, hat dasselbe Problem wie mit einem Pixel ohne Einwilligung, nur schlechter sichtbar. Die Prüfung muss dann im sendenden System stattfinden, nicht im Container.

Cross-Domain-Tracking

Cross-Domain-Tracking erlaubt es, dieselbe Person über mehrere Domains hinweg in Google Analytics 4 als **einen Nutzer mit einer Sitzung** zu erkennen. Typischer Fall: Ein Online-Shop läuft auf `shop.beispiel.de`, das Bezahlsystem auf `checkout.beispiel.de`. Ohne Cross-Domain-Tracking würde GA4 die zweite Domain als neuen Nutzer mit neuer Sitzung erfassen.

Technisch funktioniert es so: Sobald die/der Besucher/in einen Link auf eine andere konfigurierte Domain klickt, hängt der Google Tag automatisch einen Parameter `_gl` an die Ziel-URL an. Dieser enthält die Client-ID — die andere Domain übernimmt sie damit.

Eingerichtet wird Cross-Domain-Tracking pro Web-Datenstream unter **Mehr Tagging-Einstellungen → Domains konfigurieren**. Hier werden alle Domains aufgelistet, zwischen denen die Sitzung fortgesetzt werden soll. Wichtig: Auf **jeder** dieser Domains muss derselbe Google Tag (also dieselbe Mess-ID) eingebunden sein.

Hinweis zur Verweis-Ausschlussliste: Ohne Cross-Domain-Tracking taucht der Wechsel auf eine zweite Domain als „Verweis“ auf — die ursprüngliche Trafficquelle geht verloren. Cross-Domain-Tracking löst dieses Problem sauber.

D

dataLayer.push

Mit `dataLayer.push()` schreibt die Website Daten in die Datenschicht. Es ist der einzige vorgesehene Weg, den Google Tag Manager über etwas zu informieren.

```
window.dataLayer = window.dataLayer || [];
window.dataLayer.push({
  'event': 'add_to_cart',
  'ecommerce': { 'value': 49.9, 'currency': 'EUR' }
});
```

Enthält das Objekt den Schlüssel `event`, entsteht ein Ereignis, auf das ein Trigger für benutzerdefinierte Ereignisse hören kann. Ohne `event` werden nur Werte hinterlegt, die spätere

Ereignisse lesen können.

Zusammenführung

Der Tag Manager führt jeden Push mit dem bisherigen Zustand zusammen, und zwar rekursiv. Ein zweiter Push mit einem `ecommerce`-Objekt überschreibt also nicht das erste, sondern mischt sich hinein. Deshalb gehört vor jedes E-Commerce-Ereignis ein eigener Push:

```
window.dataLayer.push({ ecommerce: null });
```

Ohne diese Zeile tauchen Artikel aus einem früheren Ereignis in einem späteren wieder auf, was sich in GA4 als falsche Artikelzahlen zeigt.

Zeitpunkt

Das Array muss vor dem ersten Push existieren, daher die Zeile mit `|| []`. Werte, die schon beim Seitenaufruf gebraucht werden, gehören vor das Container-Snippet. Ein Push, der nach dem Feuern des Tags kommt, hilft diesem Tag nicht mehr.

In WordPress gehört der Code je nach Fall ins Child-Theme, in ein Snippet-Plugin oder in ein Code-Element. PHP-Ausgaben werden dabei mit `wp_json_encode()` escaped.

Datenschicht

Die **Datenschicht** (dataLayer) ist die Schnittstelle zwischen der Website und dem Google Tag Manager. Technisch ist sie ein JavaScript-Array namens `window.dataLayer`, in das die Website Objekte schreibt.

Jedes Tracking beginnt mit der Frage, welche Daten dort landen. Was in der Datenschicht steht, kann der Container lesen. Was nicht drinsteht, muss er sich aus dem DOM zusammensuchen, und das bricht bei der nächsten Designänderung.

Die Datenschicht muss vor dem Container-Snippet angelegt werden, wenn Werte schon beim Seitenaufruf zur Verfügung stehen sollen:

```
window.dataLayer = window.dataLayer || [];
window.dataLayer.push({
  'page_type': 'product',
  'user_logged_in': true
});
```

Spätere Ereignisse kommen über dataLayer.push dazu. Der Tag Manager führt die Objekte zusammen, ein späterer Push überschreibt also nur die Schlüssel, die er selbst enthält. Bei verschachtelten Objekten wie `ecommerce` führt das dazu, dass Artikel aus einem früheren Ereignis hängen bleiben, weshalb vor jedem E-Commerce-Ereignis ein `{ecommerce: null}` gepusht gehört.

In WordPress befüllt meist ein Plugin die Datenschicht. GTM4WP liefert Seiten- und WooCommerce-Daten, Consent-Tools schreiben ihre eigenen Ereignisse hinein. Gelesen werden die Werte über Datenschicht-Variablen.

Datenschicht-Variable

Eine **Datenschicht-Variable** (Data Layer Variable, kurz DLV) liest einen Wert aus der Datenschicht. Sie ist der meistgenutzte Variablentyp im Google Tag Manager.

Konfiguriert wird nur der Name des Schlüssels. Verschachtelte Werte werden mit Punkt angesprochen, Array-Elemente über ihren Index:

```
ecommerce.value
ecommerce.items.0.item_id
user.company.name
```

Version 1 oder 2

Unter den erweiterten Einstellungen steht die Version der Datenschicht. Version 2 ist der Standard und der richtige Wert. Version 1 stammt aus der Zeit vor der Punktnotation und interpretiert einen Punkt im Namen als Teil des Schlüssels, nicht als Verschachtelung.

Persistenz

Ein einmal gepushter Wert bleibt für den Rest des Seitenaufrufs lesbar, auch bei späteren Ereignissen. Das ist praktisch, führt aber zu Fehlern, wenn ein Wert für ein Ereignis gilt und für das nächste nicht mehr. Bei Single-Page-Anwendungen bleibt die Datenschicht über den ganzen Seitenwechsel hinweg bestehen, dort müssen überholte Werte ausdrücklich auf `undefined` oder `null` gesetzt werden.

Findet die Variable ihren Schlüssel nicht, liefert sie `undefined`. Ein Standardwert lässt sich in der Variablen hinterlegen.

Debug-Modus

Der **Debug-Modus** ist der Zustand, in dem sich eine Seite befindet, während eine Debug-Sitzung des Vorschaumodus läuft.

Im Container steht er als integrierte Variable „Debug-Modus“ zur Verfügung, mit dem Wert `true` oder `false`. Damit lässt sich Test-Traffic von produktiven Tags fernhalten, indem ein Ausnahme-Trigger auf diese Variable prüft.

Davon zu unterscheiden ist der Debug-Modus von Google Analytics 4, der die DebugView in der Property speist. Er wird über den Parameter `debug_mode` im Tag oder über die Erweiterung „GA4 Debugger“ aktiviert. Läuft eine Debug-Sitzung des Tag Manager, schaltet GA4 diesen Modus automatisch mit ein.

Ereignisse im Debug-Modus zählen in GA4 trotzdem in die regulären Berichte. Wer Testdaten sauber halten will, arbeitet mit einer eigenen Property oder filtert internen Traffic über eine Regel in der Verwaltung.

Deduplizierung

Deduplizierung verhindert, dass dasselbe Ereignis doppelt zählt, wenn es auf zwei Wegen gemeldet wird. Bei Meta ist das der Normalfall, sobald neben dem Pixel auch die Conversions API läuft.

Der Mechanismus ist eine gemeinsame Ereignis-ID. Beide Wege senden dasselbe Ereignis mit demselben Namen und derselben ID, Meta erkennt das Paar und zählt es einmal. Im Pixel steht die ID im dritten Argument:

```
fbq('track', 'Purchase', {
  value: {{DLV - ecommerce.value}},
  currency: {{DLV - ecommerce.currency}}
}, { eventID: 'purchase_' + {{DLV - ecommerce.transaction_id}} });
```

Die ID muss auf beiden Seiten identisch gebildet werden. Die Bestellnummer ist dafür die naheliegende Grundlage, weil sie in Shop und Browser dieselbe ist. Ein Zufallswert oder ein Zeitstempel funktioniert nicht, weil Server und Browser dann verschiedene IDs erzeugen.

Meta gleicht nur innerhalb eines begrenzten Zeitfensters ab. Kommt das serverseitige Ereignis Stunden später, etwa aus einem nächtlichen Abgleich, greift die Deduplizierung nicht mehr.

Der häufigste Fehler in WooCommerce-Shops: Das Plugin „Meta for WooCommerce“ sendet sowohl serverseitig als auch ein eigenes Browser-Pixel. Kommt aus dem Container ein drittes `Purchase`-Ereignis dazu, hilft keine ID mehr. In dem Fall gehört das Pixel im Container weg.

E

Eingebaute Einwilligungsprüfung

Google-Tags bringen eine **eingebaute Einwilligungsprüfung** mit. Sie ist Teil der Tag-Vorlage und lässt sich weder abschalten noch konfigurieren. Betroffen sind unter anderem der Google-Tag, GA4-Ereignis-Tags, Google Ads-Conversion-Tracking, Remarketing und die Conversion-Verknüpfung.

Diese Tags feuern auch bei abgelehnter Einwilligung, verhalten sich dann aber anders: Sie setzen und lesen keine Cookies und senden nur einen reduzierten, cookielosen Ping. Aus diesen Pings modelliert Google Conversions. Das ist der Unterschied zwischen dem Verwerfen eines Tags und der eingebauten Prüfung, die das Tag ausführt und intern drosselt.

Welche Einwilligungstypen ein Tag eingebaut berücksichtigt, zeigt die Tag-Vorlage selbst an. Bei GA4 sind es `analytics_storage` und `ad_user_data`, bei Google Ads zusätzlich `ad_storage` und `ad_personalization`.

Basic oder Advanced

Ob die eingebaute Prüfung allein genügt, ist eine Entscheidung über die Betriebsart des Consent Mode. Im Advanced-Modus reicht sie, die Tags laden sofort und senden vor der Zustimmung cookielose Pings. Im Basic-Modus soll vor der Zustimmung überhaupt nichts an Google gehen, dafür braucht es zusätzlich eine zusätzliche Einwilligungsprüfung auf Tag-Ebene.

Die datenschutzrechtliche Bewertung des Advanced-Modus ist in Österreich und Deutschland umstritten, weil bereits ohne Einwilligung Daten wie IP-Adresse und User-Agent an Google übertragen werden. Die Entscheidung liegt beim Verantwortlichen der Website.

Tags ohne eingebaute Prüfung, also alle Nicht-Google-Tags und jedes „Benutzerdefinierte HTML“, feuern ohne zusätzliche Prüfung uneingeschränkt.

Einstellungen zur Nutzereinwilligung

Die **Einstellungen zur Nutzereinwilligung** sind ein Abschnitt in den erweiterten Einstellungen jedes Tags. In der deutschen Oberfläche steht dort bis heute der Zusatz „BETA“, obwohl die Funktion seit Jahren produktiv genutzt wird und die Grundlage für die Einwilligungsübersicht bildet.

Der Abschnitt heißt vollständig „Einstellungen zur Nutzereinwilligung (BETA) – Zusätzliche Einwilligungsprüfungen“ und bietet genau zwei Optionen:

- *Keine zusätzlichen Einwilligungsprüfungen erforderlich* (Standard bei neuen Tags)
- *Zusätzliche Einwilligung für das Auslösen des Tags erforderlich*, dazu die Liste der Einwilligungstypen, die alle vorliegen müssen

Entscheidend ist das Wort „zusätzlich“. Die Einstellung ergänzt die eingebaute Einwilligungsprüfung, die Google-Tags ohnehin mitbringen, sie ersetzt sie nicht. Bei einem GA4-Tag mit „Keine zusätzlichen Prüfungen“ regelt weiterhin der Consent Mode, ob Cookies gesetzt werden oder nur ein cookieloser Ping geht. Bei einem Tag ohne eingebaute Prüfung, etwa „Benutzerdefiniertes HTML“, ist „Keine zusätzlichen Prüfungen“ gleichbedeutend mit „feuert immer“.

Die Einstellung lässt sich für viele Tags gleichzeitig über die Einwilligungsübersicht setzen. Einzeln steht sie im Tag unter Erweiterte Einstellungen, unterhalb der Tag-Auslöseoptionen.

Einwilligungsinitialisierung

Die **Einwilligungsinitialisierung – Alle Seiten** ist ein Trigger, der vor allen anderen Triggern feuert. Er ist für genau eine Aufgabe gedacht: die Standardeinwilligung zu setzen, bevor irgendein anderes Tag läuft.

Die Reihenfolge im Container lautet: Einwilligungsinitialisierung, dann Initialisierung, dann Seitenaufruf, danach alle übrigen Ereignisse. Der Trigger ist in jedem Web-Container vorhanden und lässt sich nicht löschen.

Auf diesen Trigger gehört das Template des Consent-Management-Tools, etwa „Complianz – Consent Mode“, „Cookiebot CMP“ oder „CookieHub CMP“. Sonst nichts. Ein GA4-Tag oder ein Pixel an dieser Stelle läuft, bevor die Standardwerte gesetzt sind, und umgeht damit den gesamten Consent Mode.

Setzt das Consent-Tool die Standardwerte bereits per Inline-Skript im Seitenquelltext, bleibt der Trigger leer. Beide Wege gleichzeitig zu nutzen erzeugt widersprüchliche Zustände.

Einwilligungstypen

Einwilligungstypen sind die Kategorien, in denen der Consent Mode die Zustimmung verwaltet. Jeder Typ steht auf `granted` oder `denied`.

- `analytics_storage` – Speicher für Analysezwecke, betrifft GA4. CMP-Kategorie Statistik.
- `ad_storage` – Speicher für Werbezwecke, betrifft Google Ads und Floodlight. CMP-Kategorie Marketing.
- `ad_user_data` – Übermittlung von Nutzerdaten an Google für Werbezwecke, unter anderem für erweiterte Conversions.
- `ad_personalization` – personalisierte Werbung und Remarketing.
- `functionality_storage` – Speicher für Funktionen der Website.
- `personalization_storage` – Speicher für Personalisierung, etwa Empfehlungen.
- `security_storage` – sicherheitsbezogener Speicher, steht in der Praxis immer auf `granted`.

Die beiden Typen `ad_user_data` und `ad_personalization` kamen mit Consent Mode v2 dazu. Ohne sie liefern Google-Ads-Konten in der EU keine Remarketing-Zielgruppen mehr.

Im Google Tag Manager tauchen die Typen an zwei Stellen auf: als Werte in der Standardeinwilligung, die das Consent-Management-Tool setzt, und in den zusätzlichen Einwilligungsprüfungen eines Tags. Eigene Typen lassen sich nicht anlegen, die Liste ist von Google vorgegeben.

Im Netzwerk-Tab codiert der Parameter `gcs` den Status der beiden älteren Typen, `gcd` den aller vier Werbe- und Analysetypen. `gcs=G100` bedeutet, dass nichts erteilt wurde, `G111` steht für Analyse und Werbung erteilt.

Einwilligungsübersicht

Die **Einwilligungsübersicht** (Consent Overview) ist eine Ansicht im Container, die für jedes Tag zeigt, ob es vor dem Auslösen eine Einwilligung prüft. Sie liegt im Bereich Tags hinter dem Schild-Symbol rechts oben und muss vorher in den Container-Einstellungen aktiviert werden.

Die Tags werden in zwei Gruppen geteilt. Als „Einwilligung konfiguriert“ gelten Google-Tags, weil sie den Consent Mode eingebaut berücksichtigen, sowie alle Tags, bei denen unter „Zusätzliche Einwilligung erforderlich“ mindestens ein Einwilligungstyp hinterlegt ist. Alles andere landet unter „Einwilligung nicht konfiguriert“.

Mehrere Tags lassen sich in dieser Ansicht gemeinsam markieren und ihre Einwilligungseinstellungen in einem Schritt setzen. Das ist der schnellste Weg, einen gewachsenen Container nachzuziehen.

Was die Ansicht nicht prüft

Die Einwilligungsübersicht prüft nur, *ob* etwas hinterlegt ist, nicht ob es inhaltlich passt. Ein Meta-Pixel, das auf `analytics_storage` statt auf `ad_storage` wartet, erscheint als sauber konfiguriert. Ebenso wenig sagt sie etwas über die Standardwerte aus, die das Consent-Management-Tool setzt.

Stehen die in der EU nicht auf `denied`, ist das Ergebnis trotz grüner Übersicht falsch.

Den tatsächlichen Zustand zeigt der Vorschaumodus im Tab „Einwilligung“, jeweils vor und nach der Zustimmung.

Elementsichtbarkeit

Der Trigger **Elementsichtbarkeit** feuert, wenn ein bestimmtes Element im sichtbaren Bereich des Browserfensters erscheint. Intern nutzt er die Intersection Observer API.

Das Element wird über seine ID oder einen CSS-Selektor ausgewählt. Dazu kommen drei Einstellungen: der Mindestanteil des Elements, der sichtbar sein muss, eine Mindestdauer der Sichtbarkeit und die Frage, ob der Trigger einmal pro Seite, einmal pro Element oder bei jedem Erscheinen feuert.

Typische Anwendungen sind Werbebanner, Danke-Meldungen nach einer Formularübermittlung und Bereiche weit unten auf langen Seiten. Für Danke-Meldungen ist der Trigger oft der einzige Weg, wenn das Formular-Plugin kein eigenes Ereignis liefert und keine eigene Danke-Seite aufruft.

Die Option „DOM-Änderungen beobachten“ ist nötig, wenn das Element erst nachträglich in die Seite kommt, was bei AJAX-Formularen immer der Fall ist. Ohne diese Option sucht der Trigger nur einmal und findet nichts.

Wie alle Trigger, die sich auf Markup stützen, bricht auch dieser bei Änderungen am Frontend. Ein Ereignis in der Datenschicht ist die stabilere Lösung, wenn sie sich einrichten lässt.

Erweiterter Abgleich

Der **erweiterte Abgleich** (Advanced Matching) gibt dem Meta Pixel Kontaktdaten mit, damit Meta ein Ereignis einem Konto zuordnen kann. Übertragen werden unter anderem E-Mail-Adresse, Telefonnummer, Vor- und Nachname, Ort und Postleitzahl, jeweils gehasht.

Es ist das Gegenstück zu den vom Nutzer bereitgestellten Daten bei Google Ads und arbeitet nach demselben Prinzip.

Automatisch oder manuell

Der automatische Abgleich wird im Werbeaccount aktiviert. Das Pixel durchsucht dann selbständig Formularfelder der Seite nach passenden Werten. Das ist unzuverlässig und greift auf Felder zu, die niemand dafür vorgesehen hat.

Der manuelle Abgleich übergibt die Werte ausdrücklich beim Initialisieren des Pixels, üblicherweise aus der Datenschicht:

```
fbq('init', {{Const - Meta Pixel-ID}}, {
  em: {{DLV - user.email}},
  ph: {{DLV - user.phone}}
});
```

Das Pixel normalisiert und hasht die Werte vor dem Versand, im Klartext geht nichts raus. Trotzdem ist das eine Verarbeitung personenbezogener Daten, die ohne Einwilligung nicht stattfinden darf. Das Tag braucht dieselbe Einwilligungsprüfung wie jedes andere Marketing-Tag.

Wie gut der Abgleich funktioniert, zeigt der Events Manager als Bewertung der Übereinstimmungsqualität je Ereignis.

Events Manager

Der **Events Manager** ist die Oberfläche, in der Meta die Ereignisse einer Datenquelle verwaltet. Dort wird das Pixel angelegt, dort laufen auch die Ereignisse der Conversions API auf.

Für die Einrichtung sind drei Ansichten wichtig.

Testereignisse zeigt in Echtzeit, was von der eigenen Sitzung ankommt, mit allen Parametern. Das ist das Gegenstück zur DebugView von GA4 und der richtige Ort, um ein neues Tag zu prüfen. Aufgerufen wird die Seite mit einem Testcode, den der Events Manager vorgibt.

Diagnose sammelt Auffälligkeiten über alle Ereignisse: fehlende Pflichtparameter, ungültige Werte, Ereignisse ohne Zuordnung. Hier taucht auch auf, wenn Deduplizierung nicht greift.

Übersicht zeigt je Ereignis die Anzahl und die Bewertung der Übereinstimmungsqualität, die angibt, wie gut sich die Ereignisse Konten zuordnen lassen. Sie steigt mit den Daten aus dem erweiterten Abgleich.

Die Zahlen im Events Manager weichen regelmäßig von denen in Google Analytics 4 ab. Das ist normal: Andere Zuordnungslogik, anderes Einwilligungsverhalten, andere Modellierung. Ein Abgleich auf die Kommastelle ist nicht zu erwarten.

F

Formularübermittlung

Der Trigger **Formularübermittlung** feuert, wenn ein `<form>`-Element abgeschickt wird. Er hört auf das Submit-Ereignis des Browsers.

Genau darin liegt sein Problem. Die meisten Formulare auf modernen Websites schicken nichts mehr ab, sondern senden die Daten per AJAX und verhindern das Standardverhalten. Ninja Forms, Contact Form 7, Gravity Forms und die Bricks-Formulare arbeiten alle so. Der Trigger feuert dann nicht, und selbst wenn er feuert, sagt er nichts darüber, ob die Übermittlung erfolgreich war.

Der verlässliche Weg führt über ein eigenes Ereignis in der Datenschicht, das das Formular-Plugin nach der erfolgreichen Übermittlung auslöst. Viele Plugins bieten dafür einen Hook, GTM4WP bringt fertige Pushes für gängige Formulare mit.

Die Bedingungen des Triggers stützen sich auf die Formular-Variablen unter den integrierten Variablen, die erst aktiviert werden müssen. Die Option „Auf Tags warten“ verzögert die Übermittlung, damit Tags noch senden können, „Validierung prüfen“ unterdrückt das Feuern bei fehlerhaften Eingaben.

G

Google Tag

Der **Google Tag** (Skriptdatei `gtag.js`) ist das offizielle JavaScript-Snippet zur Anbindung von Google Analytics 4 und weiteren Google-Produkten direkt im Quellcode einer Website — ohne den Umweg über den Google Tag Manager.

Das Grundgerüst sieht so aus:

```
<script async src="https://www.googletagmanager.com/gtag/js?id=G-XXXXXXX">
<script>
window.dataLayer = window.dataLayer || [];
function gtag(){
dataLayer.push(arguments);
}
gtag('js', new Date());
gtag('config', 'G-XXXXXXX');
</script>
```

Eigene Ereignisse werden über `gtag('event', 'name', { parameter: 'wert' })` gesendet. Das ist ausreichend für einfache Tracking-Setups, die wenige zusätzliche Ereignisse brauchen.

Für komplexere Setups — mehrere Werbenetzwerke, viele benutzerdefinierte Ereignisse, Server-Side Tagging — empfiehlt sich der Google Tag Manager. Beides lässt sich auch kombinieren, sollte dann aber sauber dokumentiert werden, um Doppel-Tracking zu vermeiden.

Google Tag Manager

Der **Google Tag Manager** (kurz **GTM**) ist ein kostenloses Tool zum zentralen Verwalten von Tracking- und Marketing-Tags. Statt jeden einzelnen Tag — Google Analytics, Google Ads, Meta Pixel, LinkedIn Insight Tag etc. — direkt im Quellcode der Website einzubauen, wird nur ein einziger **GTM-Container** eingebunden. Alle weiteren Tags werden anschließend in der GTM-Oberfläche konfiguriert.

GTM kennt drei zentrale Bausteine:

- **Tag** — der Code, der ausgespielt wird (z. B. ein GA4-Ereignis, ein Meta-Pixel-Event).
- **Trigger** — die Bedingung, bei der ein Tag feuern soll (z. B. Seitenaufruf, Klick, Formularversand).
- **Variable** — dynamische Werte, die in Tags und Triggern verwendet werden (z. B. Seitentitel, Klick-URL, Cookie-Wert).

Vorteile gegenüber direkter Einbindung: schnelleres Ausrollen neuer Tags ohne Entwicklerzugriff, Versionierung, Vorschau-Modus zum Testen, und mit dem Server-Side Tagging auch eine deutlich verbesserte Trennung von First- und Third-Party-Welt.

GTM-Konto

Das **Konto** ist die oberste Ebene im Google Tag Manager. Darunter liegen die Container.

In Agenturen hält man es meist so: ein Konto pro Kunde, darin ein Container pro Website. Der Grund ist die Rechteverwaltung, die auf Kontoebene beginnt. Wer Zugriff auf das Konto hat, sieht alle darin liegenden Container.

Ein Konto gehört dem Unternehmen, nicht der Agentur. Wird es im Google-Konto der Agentur angelegt, muss spätestens beim Wechsel des Dienstleisters die Inhaberschaft übertragen werden. Sauberer ist, das Konto im Google-Konto des Kunden anzulegen und der Agentur Zugriff zu geben.

Konten lassen sich umbenennen, aber nicht zwischen Google-Konten verschieben. Einen Container kann man dagegen exportieren und in einem anderen Konto neu aufbauen, wobei Container-ID und Versionshistorie verloren gehen.

Initialisierung

Der Trigger **Initialisierung - Alle Seiten** feuert vor allen anderen Triggern außer der Einwilligungsinitialisierung. Er ist in jedem Web-Container vorhanden und lässt sich nicht löschen.

Die Reihenfolge lautet: Einwilligungsinitialisierung, Initialisierung, Seitenaufruf, danach alle weiteren Ereignisse.

Gedacht ist der Trigger für Tags, die vor allem anderen laufen müssen, etwa das Setzen eines Cookies oder einer Variablen, auf die spätere Tags angewiesen sind.

Ein Tag mit zusätzlicher Einwilligungsprüfung gehört nicht auf diesen Trigger. Zum Zeitpunkt der Initialisierung ist das Banner noch offen, das Tag wird verworfen und nicht nachgeholt, wenn die Zustimmung später kommt. Solche Tags hängt man an das Kategorie-Ereignis des

Consent-Management-Tools.

Integrierte Variablen

Integrierte Variablen liefert der Google Tag Manager fertig mit. Sie lassen sich nur ein- und ausschalten, nicht bearbeiten. Zu finden sind sie unter Variablen → Konfigurieren.

In einem neuen Web-Container ist nur ein kleiner Teil aktiv, im Wesentlichen die Seiten-Variablen. Alles übrige fehlt, bis man es einschaltet. Das ist eine häufige Fehlerquelle: Ein Trigger, der auf „Klick-Text“ prüft, während die Variable nicht aktiviert ist, feuert nie, und der Vorschaumodus zeigt den Wert schlicht als `undefined`.

Gruppen

- Seiten: Seiten-URL, Seitenhostname, Seitenpfad, Verweis-URL
- Dienstprogramme: Ereignis, Umgebungsname, Container-ID, Containerversionsnummer, Zufallszahl, HTML-ID, Debug-Modus
- Fehler: Fehlermeldung, Fehler-URL, Fehlerzeile
- Klicks: Klick-Element, Klick-Klassen, Klick-ID, Klick-Ziel, Klick-URL, Klick-Text
- Formulare: Formularelement, Formularklassen, Formular-ID, Formularziel, Formular-URL, Formulartext
- Verlauf: Neues Verlaufsfragment, Altes Verlaufsfragment, Verlaufsquelle
- Videos: Video-Anbieter, Video-Status, Video-URL, Video-Titel, Video-Dauer, aktuelle Videozeit, Videoprozentsatz, Video sichtbar
- Scrollen: Scrolltiefe-Schwellenwert, Scrolltiefe-Einheiten, Scrollrichtung
- Sichtbarkeit: Prozentsatz sichtbar, Dauer auf dem Bildschirm

Die Gruppen für Klicks, Formulare, Videos, Scrollen und Sichtbarkeit werden nur befüllt, wenn auch der passende Trigger-Typ aktiv ist. Ohne Klick-Trigger bleiben die Klick-Variablen leer, selbst wenn sie aktiviert sind.

Reichen die integrierten Variablen nicht, legt man eine benutzerdefinierte Variable an, meist eine Datenschicht-Variable oder eine Variable vom Typ Benutzerdefiniertes JavaScript.

J

JavaScript-Fehler

Der Trigger **JavaScript-Fehler** feuert, wenn auf der Seite ein unbehandelter Fehler auftritt. Er hängt sich an `window.onerror`.

Die Details stehen in den integrierten Variablen Fehlermeldung, Fehler-URL und Fehlerzeile, die vorher aktiviert werden müssen.

Damit lassen sich Frontend-Fehler als Ereignis an Google Analytics 4 schicken und dort auswerten. Als schneller Überblick ist das brauchbar, ein Fehler-Monitoring ersetzt es nicht: Es fehlen Stacktrace, Browserversion und die Zuordnung mehrerer Fehler zu einer Ursache.

Zwei Einschränkungen sind zu beachten. Fehler aus Skripten fremder Herkunft meldet der Browser aus Sicherheitsgründen nur als `Script error.` ohne Details. Und auf Seiten mit vielen Fehlern erzeugt der Trigger sehr schnell sehr viele Ereignisse, was in GA4 ins Limit für Ereignisse pro Sitzung läuft und andere Daten verdrängt.

K

Klick-Trigger

Der **Klick-Trigger** feuert bei einem Klick auf der Seite. Es gibt ihn in zwei Varianten.

„Nur Links“ reagiert auf Klicks auf `<a>`-Elemente und bringt eine Besonderheit mit: die Option „Auf Tags warten“. Damit verzögert der Container die Navigation um eine einstellbare Zeit, typischerweise 2000 Millisekunden, damit das Tag noch senden kann, bevor die Seite wechselt.

„Alle Elemente“ reagiert auf Klicks auf beliebige Elemente, also auch auf Buttons und `<div>`-Container. Diese Variante wartet nicht.

Die Bedingungen stützen sich auf die Klick-Variablen unter den integrierten Variablen: Klick-Text, Klick-Klassen, Klick-ID, Klick-URL und Klick-Element. Sind diese Variablen nicht aktiviert, bleiben die Bedingungen leer und der Trigger feuert nie.

Warum Klick-Trigger fragil sind

Bedingungen auf CSS-Klassen oder Linktexte brechen, sobald jemand das Design ändert oder eine Übersetzung einspielt. Der Klick landet außerdem oft auf einem verschachtelten Element, etwa dem `<span>` innerhalb des Buttons, sodass die erwarteten Klassen fehlen.

Stabiler ist ein eigenes Ereignis aus der Website in der Datenschicht. Klick-Trigger sind die Wahl, wenn sich an der Seite nichts ändern lässt.

Konstante

Eine **Konstante** ist eine Variable mit einem festen Wert. Sie ist der einfachste Variablentyp und besteht aus einem einzigen Eingabefeld.

Gedacht ist sie für Werte, die an mehreren Stellen im Container vorkommen und sich selten ändern: die Mess-ID von GA4, die Conversion-ID von Google Ads, eine Pixel-ID.

Der Nutzen zeigt sich beim Wechsel. Steht die Mess-ID direkt in fünf Tags, muss sie fünfmal getauscht werden, und der eine übersehene Eintrag schickt weiter Daten in die alte Property. Als Konstante ist es eine Stelle.

Werte, die sich pro Seitenaufruf ändern können, gehören nicht in eine Konstante. Währung, Versandkosten oder Bestellwert kommen aus der Datenschicht, auch wenn sie im Moment immer gleich aussehen.

L

li_fat_id

`li_fat_id` ist die Klick-Kennung von LinkedIn. Sie hängt als Parameter an der Ziel-URL einer Anzeige und wird vom Insight Tag in ein First-Party-Cookie gleichen Namens geschrieben.

Sie erfüllt dieselbe Aufgabe wie `gclid` bei Google Ads und `_fbc` bei Meta: den Klick auf die Anzeige mit einer späteren Conversion zu verbinden, auch wenn die URL beim ersten Seitenwechsel verloren geht.

Gebraucht wird der Wert vor allem serverseitig. Meldet ein System eine Conversion über die LinkedIn Conversions API, ist `li_fat_id` neben der gehashten E-Mail-Adresse das wichtigste Merkmal für die Zuordnung.

Im Container lässt sich der Wert auf zwei Wegen auslesen: aus der URL über eine Variable vom Typ URL mit der Komponente Abfrage, oder aus dem Cookie über eine Variable vom Typ „Eigenes Cookie“. Der zweite Weg ist der verlässlichere, weil er auch auf Folgeseiten funktioniert.

Wie bei Meta entsteht das Cookie erst nach der Einwilligung, weil das Insight Tag vorher nicht lädt. Wer die Kennung serverseitig braucht, muss den URL-Parameter deshalb gegebenenfalls schon vor dem Tag sichern.

LinkedIn Conversion

Eine **Conversion** in LinkedIn ist eine im Campaign Manager angelegte Regel, die zählt, wenn eine bestimmte Aktion auf der Website passiert. Jede Conversion bekommt eine eigene Conversion-ID.

Es gibt zwei Arten. Die seitenbasierte Conversion löst aus, sobald eine URL aufgerufen wird, die einer Regel entspricht, typischerweise eine Danke-Seite. Dafür ist im Container nichts weiter zu tun, das Insight Tag genügt.

Die ereignisspezifische Conversion wird dagegen ausdrücklich ausgelöst. Sie ist die richtige Wahl, wenn es keine eigene Danke-Seite gibt, etwa bei einem Formular, das die Bestätigung per AJAX einblendet:

```
<script>
  // LinkedIn event-specific conversion
  if (window.lintrk) {
    window.lintrk('track', { conversion_id: {{Const - LinkedIn Conv Lead}} });
  }
</script>
```

Das Tag braucht das Insight Tag als Setup-Tag über die Tag-Sequenzierung, sonst existiert lintrk zum Zeitpunkt des Aufrufs noch nicht. Die Abfrage auf `window.lintrk` fängt den Fall ab, dass die Einwilligung fehlt und das Basis-Tag deshalb gar nicht geladen hat.

Ausgelöst wird das Ganze über einen Trigger für benutzerdefinierte Ereignisse auf demselben Datenschicht-Ereignis, das auch GA4 und Google Ads bedient.

LinkedIn Conversions API

Die **LinkedIn Conversions API** meldet Conversions serverseitig, ohne das Insight Tag im Browser. Sie ist LinkedIns Gegenstück zu Metas Conversions API.

Der Reiz liegt bei LinkedIn weniger in der Umgehung von Adblockern als im B2B-Verkaufsprozess. Ein Abschluss passiert dort oft Wochen nach dem Klick und nicht auf der Website, sondern im CRM. Solche Conversions lassen sich nur serverseitig melden, entweder aus dem CRM heraus oder über einen Upload.

Zugeordnet wird über die Merkmale, die mitgeschickt werden: die gehashte E-Mail-Adresse, Angaben zur Person und Firma sowie die Klick-Kennung li_fat_id, sofern sie beim ersten Kontakt gesichert wurde. Ohne eines dieser Merkmale läuft die Conversion ins Leere.

Wird dieselbe Conversion zusätzlich über das Insight Tag gemeldet, ist Doppelzählung möglich. Eine Ereignis-ID wie bei Metas Deduplizierung gibt es hier nicht in vergleichbarer Form, deshalb ist es am einfachsten, pro Conversion-Regel einen Weg festzulegen und den anderen konsequent wegzulassen.

Auch hier gilt: Der serverseitige Versand ändert nichts an der Einwilligungspflicht. Die Prüfung muss im sendenden System stattfinden, der Container ist daran nicht beteiligt.

LinkedIn Insight Tag

Das **LinkedIn Insight Tag** ist LinkedIns Tracking-Code. Es misst Seitenaufrufe und Conversions, speist Retargeting-Zielgruppen und liefert im Campaign Manager Angaben zur Zusammensetzung der Website-Besucher nach Branche, Unternehmensgröße und Funktion.

Technisch lädt es `insight.min.js` von `snap.licdn.com` und legt die Funktion `lintrk` an. Zugeordnet wird es über die Partner-ID.

Einbindung über den Container

LinkedIn stellt eine eigene Vorlage in der Vorlagengalerie bereit, die dem Weg über Benutzerdefiniertes HTML vorzuziehen ist. Die Partner-ID kommt als Konstante hinein.

Das Basis-Tag gehört auf das Marketing-Ereignis des Consent-Management-Tools, mit der Auslöseoption „Einmal pro Seite“ und einer zusätzlichen Einwilligungsprüfung auf `ad_storage`. Wie das Meta Pixel hat es keine eingebaute Prüfung.

Ob es ankommt, zeigt der Campaign Manager unter Analysen beim Insight Tag. Der Status wechselt nach dem ersten Traffic auf „Aktiv“, was einige Stunden dauern kann.

Für B2B-Websites mit kleinen Fallzahlen ist zu beachten, dass LinkedIn demografische Auswertungen erst ab einer Mindestzahl an Besuchern ausweist. Bei wenig Traffic bleiben diese Berichte leer, obwohl das Tag korrekt läuft.

LinkedIn Partner-ID

Die **Partner-ID** identifiziert das Werbekonto, dem das LinkedIn Insight Tag seine Daten meldet. Sie ist eine sechs- bis siebenstellige Zahl und steht im Campaign Manager unter Analysen beim Insight Tag.

Im Code taucht sie an zwei Stellen auf: als `_linkedin_partner_id` und im Array `window._linkedin_data_partner_ids`. Das Array erlaubt mehrere IDs gleichzeitig, was gebraucht wird, wenn eine Website für mehrere Werbekonten misst, etwa bei getrennten Konten je Land.

Im Container gehört sie in eine Konstante, nicht direkt ins Tag. Dann ist sie beim Wechsel des Kontos an einer Stelle zu tauschen.

Nicht zu verwechseln mit der Conversion-ID, die zu einer einzelnen Conversion gehört, und nicht mit der Kampagnen- oder Konto-ID aus der URL des Campaign Manager.

---

lintrk

`lintrk` ist die globale JavaScript-Funktion, die das LinkedIn Insight Tag auf der Seite anlegt. Sie ist das Gegenstück zu `fbq` bei Meta und `gtag` bei Google.

Praktisch relevant ist ein einziger Aufruf, das Melden einer ereignisspezifischen Conversion:

```
window.lintrk('track', { conversion_id: 12345678 });
```

Die Funktion existiert erst, wenn das Basis-Tag geladen hat. Ein Aufruf davor bricht mit einem Fehler ab, der im Container nicht auffällt, weil das Tag als ausgelöst gilt. Deshalb gehört jeder Aufruf in eine Abfrage auf `window.lintrk`, und das Basis-Tag gehört als Setup-Tag über die Tag-Sequenzierung davor.

Dass `lintrk` fehlt, ist der Normalfall, solange keine Einwilligung für Marketing vorliegt. Im Vorschaumodus lässt sich das prüfen, indem man im Reiter „Einwilligung“ den Zustand vor und nach der Zustimmung vergleicht.

M

Measurement Protocol

Das **Measurement Protocol** ist eine HTTPS-Schnittstelle, über die Ereignisse direkt an Google Analytics 4 gesendet werden können — ohne Browser, ohne Google Tag, ohne JavaScript. Die Daten werden als JSON-Payload an einen Endpunkt geschickt.

Typische Einsatzfälle:

- **Offline-Conversions** — etwa Telefonate, Ladenbesuche oder Vertragsabschlüsse, die nicht im Browser stattfinden.
- **Server-zu-Server-Tracking** — Ereignisse aus dem Backend (etwa erfolgreiche Zahlungen) direkt an GA4 senden.
- **IoT- und Embedded-Geräte** — Smart Devices ohne Browser, die trotzdem Telemetrie erfassen sollen.
- **Ergänzung zum Server-Side Tagging**.

Aufruf-Skizze (vereinfacht):

```
POST https://www.google-analytics.com/mp/collect?measurement_id=G-XXX&api_secret=YYY
```

Im Body steht ein JSON mit `client_id` und einem Array von Events. Das `api_secret` wird in der Verwaltung des Datenstreams erzeugt und sollte serverseitig sicher verwahrt werden — niemals im Browser-Code ausspielen.

Meta Pixel

Das **Meta Pixel** ist der Tracking-Code von Facebook und Instagram. Es misst Seitenaufrufe und Aktionen auf der Website und meldet sie an den Werbeaccount, wo sie für Conversion-Messung und Remarketing genutzt werden.

Technisch legt es die globale Funktion `fbq` an und lädt `fbevents.js` von `connect.facebook.net`. Identifiziert wird der Account über die Pixel-ID, eine 15- bis 16-stellige Zahl.

Der Code im Original

Metas Anleitung liefert diesen Basis-Code. Er lädt das Pixel und meldet den ersten Seitenaufruf:

```
<!-- Facebook Pixel Code -->
<script>
!function(f,b,e,v,n,t,s)
{if(f.fbq)return;n=f.fbq=function(){n.callMethod?
n.callMethod.apply(n,arguments):n.queue.push(arguments)};
if(!f._fbq)f._fbq=n;n.push=n;n.loaded=!0;n.version='2.0';
n.queue=[];t=b.createElement(e);t.async=!0;
t.src=v;s=b.getElementsByTagName(e)[0];
s.parentNode.insertBefore(t,s)}(window, document,'script',
'https://connect.facebook.net/en_US/fbevents.js');
fbq('init', '111111111111111');
fbq('track', 'PageView');
</script>
<!-- End Facebook Pixel Code -->
```

Die Zahl in `fbq('init', '111111111111111');` ist ein Platzhalter. Dort steht die Pixel-ID, die Meta im Events Manager vergibt.

Im Container gehört diese ID nicht direkt in den Code, sondern in eine Konstante, hier `Meta Pixel ID`. Die Zeile sieht dann so aus:

```
fbq('init', {{Meta Pixel ID}});
```

Entscheidend sind die fehlenden Anführungszeichen um die geschweiften Klammern. Der Container ersetzt den Platzhalter beim Kompilieren durch einen Funktionsaufruf, der den Wert typgerecht

liefert. Bei einer ID fällt das nicht auf, bei Zahlen und Arrays wie Bestellwert oder Artikelliste würden zusätzliche Anführungszeichen daraus Strings machen, und Meta verwirft sie stillschweigend.

Jedes weitere Ereignis ist ein eigenes Tag mit eigenem Trigger. Für einen Lead nach der Formularübermittlung genügt eine Zeile:

```
<script>
  fbq('track', 'Lead');
</script>
```

Dieses Tag braucht das Basis-Tag als Setup-Tag über die Tag-Sequenzierung. Ohne den Basis-Code existiert `fbq` noch nicht: Das Tag gilt im Container trotzdem als ausgelöst, bei Meta kommt aber nichts an.

Einbindung über den Container

Zwei Wege sind üblich. Die Vorlage „Facebook Pixel“ aus der Vorlagengalerie ist der bequemere: weniger Code, Einwilligungseinstellungen direkt am Tag. Sie wird von Meta allerdings nicht mehr gepflegt, weshalb neuere Funktionen fehlen können. Der zweite Weg ist Benutzerdefiniertes HTML.

Das Basis-Tag gehört auf das Marketing-Ereignis des Consent-Management-Tools, mit der Auslöseoption „Einmal pro Seite“ und einer zusätzlichen Einwilligungsprüfung auf `ad_storage`. Eine eingebaute Prüfung hat das Pixel nicht, ohne diese Einstellung feuert es uneingeschränkt.

Die Tags für die einzelnen Standard-Ereignisse bekommen das Basis-Tag als Setup-Tag über die Tag-Sequenzierung, sonst laufen sie ins Leere, wenn `fbq` noch nicht existiert.

Das `<noscript>`-Bild aus Metas Anleitung gehört nicht in den Container. Es umgeht jede Einwilligungsprüfung und bringt nichts, weil der Container ohne JavaScript ohnehin nicht läuft.

In WooCommerce-Shops bringt das Plugin „Meta for WooCommerce“ ein eigenes Pixel mit. Läuft es parallel zum Container, feuert alles doppelt.

Meta Standard-Ereignisse

Standard-Ereignisse sind die von Meta vordefinierten Ereignisnamen, die das Meta Pixel kennt. Nur sie lassen sich im Werbeaccount ohne Weiteres als Conversion-Ziel und für die Gebotsoptimierung auswählen.

Gesendet werden sie mit `fbq('track', '<Name>', {…})`. Die gebräuchlichsten sind `PageView`, `ViewContent`, `AddToCart`, `InitiateCheckout`, `AddPaymentInfo`, `Purchase`, `Lead`, `CompleteRegistration`, `Search` und `Contact`.

Zuordnung zu GA4

Wer die Datenschicht bereits für GA4 befüllt hat, kann dieselben Ereignisse nutzen. Die Namen unterscheiden sich:

- `view_item` wird zu `ViewContent`
- `add_to_cart` wird zu `AddToCart`
- `begin_checkout` wird zu `InitiateCheckout`
- `purchase` wird zu `Purchase`

Auch die Parameter heißen anders. Meta erwartet `content_ids`, `content_type`, `contents`, `value` und `currency`, während GA4 mit `items` arbeitet. Bei `Purchase` sind `value` und `currency` Pflicht, ohne sie ist die Conversion wertlos.

Werden GTM-Variablen in einem Custom-HTML-Tag eingesetzt, gehören sie ohne Anführungszeichen in den Code. Der Container ersetzt `{{…}}` beim Kompilieren durch einen Funktionsaufruf, der den Wert typgerecht liefert. Zusätzliche Quotes machen aus Zahlen und Arrays Strings, und Meta verwirft sie

dann stillschweigend.

Für alles, wofür es kein Standard-Ereignis gibt, existiert `fbq('trackCustom', …)`. Solche Ereignisse lassen sich im Werbeaccount nur über eine benutzerdefinierte Conversion weiterverwenden.

Microsoft Clarity

Microsoft Clarity zeichnet Sitzungen auf und erstellt Heatmaps. Es ist kostenlos und wird häufig neben Google Analytics 4 eingesetzt, weil es eine andere Frage beantwortet: nicht wie viele, sondern was die Leute auf der Seite tun.

Für die Einbindung gibt es eine Vorlage in der Vorlagengalerie. Konfiguriert wird nur die Projekt-ID. Wie bei allen Nicht-Google-Tags fehlt die eingebaute Einwilligungsprüfung, das Tag braucht also eine zusätzliche Prüfung, üblicherweise auf `analytics_storage`.

Datenschutzrechtlich ist Clarity heikler als eine reine Zahlenmessung, weil Sitzungsaufzeichnungen Eingaben und Bildschirminhalte erfassen können. Clarity maskiert Formularfelder standardmäßig, die Maskierung lässt sich aber lockern. Bei Seiten mit Kontakt-, Bestell- oder Gesundheitsdaten gehört die Einstellung geprüft, bevor das Tag live geht.

Clarity kann sich mit einer GA4-Property verbinden. Dann lassen sich Aufzeichnungen nach GA4-Zielgruppen filtern, was die Suche nach Sitzungen mit Abbruch im Checkout erheblich abkürzt.

N

Nachschlagetabelle

Eine **Nachschlagetabelle** (Lookup Table) bildet Eingabewerte auf Ausgabewerte ab. Man wählt eine Eingabevariable und trägt Paare ein: Ist der Wert X, gib Y zurück.

Ein häufiger Fall ist die Zuordnung mehrerer Domains zu ihren Mess-IDs, wenn ein Container mehrere Websites bedient. Eingabe ist dann die integrierte Variable „Seitenhostname“, Ausgabe die jeweilige Mess-ID.

Der Vergleich ist exakt und unterscheidet Groß- und Kleinschreibung. Geprüft wird von oben nach unten, die erste Übereinstimmung gewinnt. Trifft keine Zeile zu, liefert die Variable den Standardwert, sofern einer gesetzt ist, sonst `undefined`.

Sind Muster statt exakter Werte nötig, etwa alle URLs unterhalb eines Pfades, ist die Tabelle für reguläre Ausdrücke der richtige Typ.

O

Ordner

Ordner gliedern die Objekte eines Containers. Ein Tag, ein Trigger oder eine Variable kann in genau einem Ordner liegen.

Ordner haben keinerlei Wirkung auf die Ausführung. Sie sind reine Sortierung in der Oberfläche, vergleichbar mit Verzeichnissen im Dateisystem. Nicht zugeordnete Objekte landen unter „Nicht im Ordner“.

Sinnvoll ist eine Gliederung nach Plattform, etwa Google Analytics, Google Ads, Meta, LinkedIn, Consent und Utilities. Ab etwa dreißig Objekten wird das zur Pflicht, sonst findet niemand mehr, welches Tag wofür zuständig ist. Eine Alternative ist die Gliederung nach Kunde oder Website, wenn mehrere Sites in einem Container liegen.

Ordner allein ersetzen keine Namenskonvention. Erst beides zusammen macht einen gewachsenen Container wieder wartbar.

S

Scrolltiefe

Der Trigger **Scrolltiefe** feuert, wenn eine festgelegte Scrollposition erreicht wird. Die Schwellenwerte lassen sich in Prozent oder in Pixeln angeben, vertikal oder horizontal.

Üblich sind 25, 50, 75 und 100 Prozent. Welcher Wert ausgelöst hat, steht in den integrierten Variablen Scrolltiefe-Schwellenwert, Scrolltiefe-Einheiten und Scrollrichtung, die vorher aktiviert werden müssen.

Google Analytics 4 misst Scrolling bereits selbst, als Teil der optimierten Analyse, allerdings nur bei 90 Prozent und als Ereignis `scroll`. Ein eigener Trigger im Tag Manager lohnt sich nur, wenn feinere Stufen gebraucht werden. Beides parallel laufen zu lassen erzeugt doppelte Daten.

Bei Prozentangaben ist zu beachten, dass sich der Wert auf die Dokumenthöhe beim Auslösen bezieht. Lädt eine Seite Inhalte nach, verschiebt sich die Bezugsgröße, und 50 Prozent bedeuten nicht mehr dasselbe wie beim Laden.

Seitenaufruf-Trigger

Der **Seitenaufruf-Trigger** feuert beim Laden einer Seite. Er kennt drei Zeitpunkte, die sich darin unterscheiden, wie weit der Browser beim Aufbau der Seite ist.

- *Seitenaufruf* (`gtm.js`): so früh wie möglich, sobald der Container geladen ist. Das DOM ist noch unvollständig.
- *DOM bereit* (`gtm.dom`): das HTML ist geparst, Bilder fehlen eventuell noch. Der richtige Zeitpunkt für alles, was Elemente auf der Seite ausliest.
- *Fenster geladen* (`gtm.load`): alles ist geladen, einschließlich Bildern und externen Ressourcen.

Messende Tags gehören so früh wie möglich, damit sie auch bei schnellen Weiterklicks noch laufen. Tags, die Werte aus dem DOM brauchen, gehören auf „DOM bereit“.

Der Trigger lässt sich auf bestimmte Seiten einschränken, meist über die integrierte Variable „Seitenpfad“ oder „Seiten-URL“.

Bei Single-Page-Anwendungen feuert der Seitenaufruf-Trigger nur einmal, beim echten Dokumentaufruf. Virtuelle Seitenwechsel brauchen stattdessen einen Trigger auf Verlaufsänderung oder ein eigenes Ereignis aus der Anwendung.

Server-Side Tagging

Beim **Server-Side Tagging** werden Tracking-Daten nicht direkt vom Browser an Google, Meta, TikTok etc. gesendet, sondern an einen **eigenen Server-Container**. Dieser empfängt, verarbeitet und leitet die Daten an die Zielplattformen weiter — alles auf der eigenen Infrastruktur.

Vorteile

- **Performance:** Der Browser sendet nur einen einzigen Request — der Server übernimmt die Weiterleitung an viele Drittsysteme.
- **Datenschutz:** Daten lassen sich vor der Weiterleitung bereinigen, anonymisieren oder filtern.
- **Werbeblocker-Resistenz:** Anfragen laufen über die eigene Domain — werden seltener blockiert.
- **Datenqualität:** Längere Cookie-Lebensdauer durch Setzen serverseitiger First-Party-Cookies.

Nachteile

- **Einrichtungsaufwand:** Server-Container muss konfiguriert, gehostet und gewartet werden.
- **Kosten:** Hosting (Google Cloud Run, eigener VPS) und ggf. Custom-Domain.
- **Komplexität:** Mehr bewegliche Teile — Fehler sind schwerer zu diagnostizieren als beim Client-Side-Tagging.

Eingerichtet wird Server-Side Tagging über den Google Tag Manager in der Variante *Server-Container*. Für GA4 wird die `measurement_id` dann nicht mehr direkt an Google geschickt, sondern an die eigene Tagging-Domain.

Standardeinwilligung

Die **Standardeinwilligung** (default consent state) legt fest, welchen Status die Einwilligungstypen haben, bevor die Besucherin oder der Besucher überhaupt etwas angeklickt hat.

Gesetzt wird sie vor allen anderen Tags, entweder durch das Template des Consent-Management-Tools auf dem Trigger Einwilligungsinitialisierung – Alle Seiten oder durch ein Inline-Skript des Tools vor dem Container-Snippet. Beides gleichzeitig führt zu widersprüchlichen Werten und ist der häufigste Grund für unerklärliche Consent-Zustände.

Für die EU, den EWR, die Schweiz und Großbritannien gehören `ad_storage`, `analytics_storage`, `ad_user_data` und `ad_personalization` auf `denied`. Über die Regionseinstellung können andere Regionen abweichende Werte bekommen. Ein Template mit Consent Mode, dessen Standardwerte auf `granted` stehen, ist wirkungslos, auch wenn die Einwilligungsübersicht sauber aussieht.

Zur Standardeinwilligung gehört der Parameter `wait_for_update`, üblich sind 500 Millisekunden. In dieser Zeit kann das Consent-Tool eine gespeicherte Entscheidung melden, bevor der erste Hit rausgeht. Ohne diese Wartezeit sendet ein Wiederkehrer den ersten Seitenaufruf mit `denied`, obwohl er längst zugestimmt hat.

Prüfen lässt sich das Ergebnis im Vorschaumodus im Tab „Einwilligung“: Dort steht eine Zeile für die Standardwerte und eine weitere für das Update nach dem Klick im Banner.

T

Tabelle für reguläre Ausdrücke

Die **Tabelle für reguläre Ausdrücke** (RegEx Table) funktioniert wie eine Nachschlagetabelle, vergleicht die Eingabe aber mit einem regulären Ausdruck statt mit einem festen Wert.

Die Zeilen werden von oben nach unten geprüft, die erste passende gewinnt. Die Reihenfolge ist damit Teil der Logik: Ein Muster wie `.*` gehört ans Ende, sonst greift es für alles.

Zwei Schalter stehen zur Wahl. „Groß-/Kleinschreibung ignorieren“ ist standardmäßig aktiv, „Vollständige Übereinstimmung“ verlangt, dass der Ausdruck den gesamten Eingabewert abdeckt statt nur einen Teil.

Typischer Einsatz ist die Gruppierung von URLs zu Seitentypen: `/produkt/` wird zu `product`, `/blog/` zu `article`, alles übrige zu `other`. Das Ergebnis geht dann als Parameter an GA4 und wird dort als benutzerdefinierte Dimension registriert.

Tag

Ein **Tag** ist im Google Tag Manager der Teil, der beim Auslösen tatsächlich etwas tut: ein Ereignis an Google Analytics 4 schicken, eine Google-Ads-Conversion melden, das Meta-Pixel laden. Variablen und Trigger bereiten nur vor, gesendet wird über Tags.

Jedes Tag hat einen Typ, eine Konfiguration und mindestens einen Trigger. Der Typ ist die Vorlage: „Google-Tag“, „Google Analytics: GA4-Ereignis“, „Google Ads-Conversion-Tracking“ und „Conversion-Verknüpfung“ sind eingebaut, weitere Vorlagen kommen aus der Vorlagengalerie. Gibt es für ein Werkzeug keine Vorlage, bleibt das Tag „Benutzerdefiniertes HTML“.

Auslöseoptionen

Unter „Erweiterte Einstellungen“ legt man fest, wie oft ein Tag feuern darf: einmal pro Ereignis, einmal pro Seite oder ungebremst. Dort stehen auch die Tag-Priorität, bei der die höhere Zahl früher feuert, und die Tag-Sequenzierung, mit der sich ein Tag vor oder nach einem anderen ausführen lässt.

Einwilligung

Jedes Tag hat eigene Einwilligungseinstellungen. Google-Tags prüfen die Einwilligung eingebaut über den Consent Mode. Für alle anderen wird unter „Zusätzliche Einwilligung erforderlich“ hinterlegt, welche Einwilligungstypen vorliegen müssen. Welche Tags das schon tun, zeigt die Einwilligungsübersicht.

Häufigster Fehler in gewachsenen Containern sind zwei Trigger an einem Tag, die beim selben Vorgang beide feuern können. Das Tag sendet dann doppelt, bei einem `purchase`-Ereignis fällt das erst im Umsatzbericht auf.

Tag Assistant

Der **Tag Assistant** ist Googles Werkzeug zum Prüfen von Tags. Er läuft unter `tagassistant.google.com` im Browser und ist zugleich die Oberfläche, die sich beim Vorschaumodus des Google Tag Manager öffnet.

Die frühere Chrome-Erweiterung „Tag Assistant Legacy“ wurde eingestellt. Wer sie noch installiert hat, sieht teils veraltete Bewertungen, insbesondere zu Universal Analytics.

Neben dem Container zeigt der Tag Assistant auch, welche anderen Google-Tags auf einer Seite laufen: Google-Tag, GA4, Google Ads. Das hilft bei der Frage, ob ein Tag doppelt eingebunden ist, etwa einmal über den Container und einmal direkt im Theme.

Für Seiten, die den Container über eine Umgebung laden, lässt sich die Debug-Sitzung direkt auf diese Umgebung richten.

Tag Manager API

Die **Tag Manager API** erlaubt den programmatischen Zugriff auf Konten, Container, Arbeitsbereiche, Tags, Trigger, Variablen und Versionen. Aktuell ist Version 2.

Sie lohnt sich, wenn dasselbe Setup in vielen Containern gebraucht wird, etwa beim Aufsetzen eines Standard-Trackings für neue Kunden, oder wenn Container aus einem anderen System heraus gepflegt werden sollen.

Die Authentifizierung läuft über OAuth 2.0 im Namen eines Nutzers, der Zugriff auf das jeweilige Konto hat. Ein Dienstkonto funktioniert nur, wenn es ausdrücklich als Nutzer im Konto eingetragen ist. Bei mehreren Kunden bedeutet das eine eigene Autorisierung je Konto.

Die Objekte werden immer in einem Arbeitsbereich angelegt und geändert, nie direkt live. Ein sinnvoller Schnitt für Automatisierungen ist deshalb: alles per API vorbereiten und eine Version erstellen, das Veröffentlichen aber als bewussten Schritt von Hand belassen. Der dafür nötige Berechtigungsbereich wird dann gar nicht erst angefordert.

Die API kennt Kontingente pro Minute und pro Tag. Wer viele Objekte anlegt, braucht ein Wiederholungsverhalten mit wachsenden Wartezeiten.

Tag-Priorität

Die **Tag-Priorität** bestimmt, in welcher Reihenfolge Tags gestartet werden, die beim selben Ereignis feuern. Sie steht in den erweiterten Einstellungen und ist eine ganze Zahl, auch negativ.

Die höhere Zahl startet früher. Ohne Angabe gilt 0, weshalb ein Tag mit Priorität 10 vor allen unkonfigurierten läuft.

Wichtig ist die Einschränkung: Die Priorität steuert nur den Start, nicht das Ende. Alle Tags werden asynchron ausgeführt, ein Tag mit höherer Priorität kann also nach einem später gestarteten fertig werden. Wer echtes Nacheinander braucht, etwa weil ein Tag auf den Basis-Code eines anderen angewiesen ist, nimmt die Tag-Sequenzierung.

In der Praxis wird die Priorität selten gebraucht. Ein sinnvoller Einsatz ist das Setzen einer Variablen oder eines Cookies vor allen messenden Tags.

Tag-Sequenzierung

Die **Tag-Sequenzierung** legt fest, dass ein Tag vor oder nach einem anderen ausgeführt wird. Sie steht in den erweiterten Einstellungen jedes Tags.

Zwei Felder stehen zur Wahl: ein Setup-Tag, das vorher läuft, und ein Cleanup-Tag, das danach läuft. Beide Tags brauchen keinen eigenen Trigger, sie werden über die Sequenz ausgelöst.

Der klassische Fall ist ein Pixel, das erst einen Basis-Code braucht. Das Ereignis-Tag bekommt das Basis-Tag als Setup-Tag, und die Reihenfolge stimmt unabhängig davon, was der Trigger sonst noch auslöst.

Zwei Schalter steuern das Verhalten bei Fehlern. Standardmäßig feuert das Haupt-Tag nicht, wenn das Setup-Tag scheitert. Umgekehrt lässt sich einstellen, dass das Cleanup-Tag entfällt, wenn das Haupt-Tag scheitert.

Sequenzierung ist keine Prioritätsregel. Sie erzwingt echtes Nacheinander, während die Tag-Priorität nur die Startreihenfolge beeinflusst und Tags trotzdem parallel laufen lässt.

Timer-Trigger

Der **Timer-Trigger** feuert in festen Abständen, solange jemand auf der Seite ist. Eingestellt werden das Intervall in Millisekunden und die maximale Anzahl an Wiederholungen.

Dazu kommt eine Bedingung, auf welchen Seiten der Timer überhaupt startet. Ohne diese Einschränkung läuft er auf jeder Seite.

Gedacht war der Trigger ursprünglich für die Messung der Verweildauer, weil Universal Analytics die Zeit auf der letzten Seite nicht messen konnte. In Google Analytics 4 ist das überholt: Die Engagement-Zeit wird dort ohnehin erfasst, und künstliche Ereignisse alle paar Sekunden verfälschen sie zusätzlich.

Sinnvoll bleibt der Timer für Sonderfälle, etwa um bei einem Video oder einer langen Anwendung in festen Abständen einen Lebenszeichen-Ping zu senden. In den meisten Containern, in denen er auftaucht, stammt er aus der Universal-Analytics-Zeit und kann weg.

Trigger

Ein **Trigger** bestimmt, wann ein Tag feuert. Jedes Tag braucht mindestens einen, ohne Trigger bleibt es untätig.

Technisch hört ein Trigger immer auf ein Ereignis in der Datenschicht und prüft zusätzlich Bedingungen. Auch was in der Oberfläche wie ein eigener Typ aussieht, ist intern ein Ereignis: Ein Klick-Trigger reagiert auf `gtm.click`, ein Seitenaufruf-Trigger je nach Zeitpunkt auf `gtm.js`, `gtm.dom` oder `gtm.load`.

Typen

- Seitenaufruf: Seitenaufruf, DOM bereit, Fenster geladen
- Klick: Nur Links, Alle Elemente
- Nutzerinteraktion: Elementsichtbarkeit, Formularübermittlung, Scrolltiefe, Timer, YouTube-Video
- Benutzerdefiniertes Ereignis
- Sonstige: Verlaufsänderung, JavaScript-Fehler, Trigger-Gruppe
- Initialisierung und Einwilligungsinitialisierung

In der Praxis ist das benutzerdefinierte Ereignis der wichtigste Typ. Die Website pusht ein eigenes Ereignis, der Trigger hört darauf. Das ist stabiler als Klick-Trigger, die sich auf CSS-Klassen oder Linktexte stützen und bei der nächsten Designänderung brechen.

Ausnahmen

Neben den auslösenden Triggern kennt jedes Tag Ausnahmen. Trifft eine Ausnahme zu, feuert das Tag nicht, auch wenn der auslösende Trigger passt. Üblich sind Ausnahmen für internen Traffic oder für Testumgebungen.

Trigger für benutzerdefinierte Ereignisse

Der **Trigger für benutzerdefinierte Ereignisse** hört auf einen Ereignisnamen in der Datenschicht. Er ist der wichtigste Trigger-Typ im Google Tag Manager.

Die Website pusht ein Ereignis, der Trigger reagiert darauf:

```
window.dataLayer.push({
  'event': 'lead_submitted',
  'form_id': 'kontakt'
});
```

Im Trigger steht dann `lead_submitted` als Ereignisname. Optional lässt sich der Name als regulärer Ausdruck auswerten, etwa `^cmplz_event_` für alle Kategorie-Ereignisse von Complianz.

Der Vorteil gegenüber Klick-Triggern ist die Stabilität. Das Ereignis ist ein vereinbarter Vertrag zwischen Website und Container, unabhängig von Markup und Design. Wird die Seite umgebaut, wandert der Push mit, und im Tag Manager ändert sich nichts.

Alle E-Commerce-Ereignisse von GA4 laufen über diesen Typ: `view_item`, `add_to_cart`, `begin_checkout`, `purchase`. In WordPress liefert GTM4WP diese Pushes für WooCommerce.

Der Ereignisname selbst steht als integrierte Variable „Ereignis“ zur Verfügung und lässt sich direkt an GA4 weitergeben.

Trigger-Gruppe

Eine **Trigger-Gruppe** feuert erst, wenn alle in ihr enthaltenen Trigger mindestens einmal ausgelöst haben. Sie ist eine Und-Verknüpfung über Ereignisse hinweg.

Das unterscheidet sie von den Bedingungen innerhalb eines Triggers. Diese prüfen mehrere Kriterien beim selben Ereignis, die Trigger-Gruppe dagegen sammelt über die Zeit: erst muss A passiert sein, dann B, in beliebiger Reihenfolge.

Ein Beispiel ist eine Messung, die erst zählen soll, wenn jemand sowohl ein Video gestartet als auch bis zum Ende der Seite gescrollt hat. Ein anderes ist das Warten auf zwei unabhängige Datenschicht-Ereignisse, bevor ein Tag alle nötigen Werte beisammen hat.

Die Gruppe feuert höchstens einmal pro Seite. Wiederholt sich einer der enthaltenen Trigger, löst das kein weiteres Ereignis aus.

Im Vorschaumodus erscheint die Gruppe als eigenes Ereignis in der Zeitleiste, sobald die letzte Bedingung erfüllt ist.

U

Umgebung

Eine **Umgebung** (Environment) ist ein eigener Ausspielkanal desselben Containers. Damit läuft auf der Staging-Site eine andere Version als auf der Live-Site, ohne dass ein zweiter Container nötig wäre.

Jeder Container hat zwei eingebaute Umgebungen: „Live“ und „Latest“. Weitere lassen sich unter Verwaltung → Umgebungen anlegen, typischerweise eine für Staging.

Jede Umgebung hat ein eigenes Snippet mit den zusätzlichen Parametern `gtm_auth` und `gtm_preview`. Dieses Snippet muss auf der jeweiligen Website eingebaut sein, sonst greift die Zuordnung nicht. In WordPress nehmen Plugins wie GTM4WP dafür eigene Felder entgegen.

In der Praxis lohnen sich Umgebungen dort, wo ein echter Staging-Workflow existiert. Bei kleineren Websites ist der Aufwand höher als der Nutzen, dort genügt der Vorschaumodus.

Die integrierte Variable „Umgebungsname“ gibt den Namen der aktiven Umgebung zurück und eignet sich als Bedingung, um Test-Traffic von produktiven Tags fernzuhalten.

V

Variable

Eine **Variable** liefert einen Wert, den Tags und Trigger verwenden. Tags setzen sie ein, um Daten zu befüllen, Trigger, um Bedingungen zu prüfen.

Geschrieben wird eine Variable in der Oberfläche als `{{Name}}`. Dieser Platzhalter lässt sich in fast jedem Feld verwenden, auch mitten in einem Text.

Es gibt zwei Gruppen: integrierte Variablen, die der Tag Manager mitbringt und die nur aktiviert werden, und benutzerdefinierte Variablen, die man selbst anlegt.

Variablen werden bei jedem Ereignis neu ausgewertet, und zwar mehrfach. Eine Variable darf deshalb keine Seiteneffekte haben. Wer in einer Variablen vom Typ Benutzerdefiniertes JavaScript ein Ereignis in die Datenschicht pusht oder einen Zähler hochsetzt, bekommt unvorhersehbare Ergebnisse.

Findet eine Variable keinen Wert, gibt sie `undefined` zurück. Für diesen Fall lässt sich in den meisten Variablentypen ein Standardwert hinterlegen. Das ist sauberer, als in GA4 später mit fehlenden Parametern umzugehen.

Welchen Wert eine Variable zu einem bestimmten Ereignis hatte, zeigt der Vorschaumodus im Tab „Variablen“.

Verlaufsänderung

Der Trigger **Verlaufsänderung** (History Change) feuert, wenn sich die URL ändert, ohne dass die Seite neu geladen wird. Das passiert bei Sprungmarken und bei Single-Page-Anwendungen, die über die History API navigieren.

Er ist damit das Gegenstück zum Seitenaufruf-Trigger, der bei virtuellen Seitenwechseln nicht mehr feuert.

Die zugehörigen integrierten Variablen heißen Neues Verlaufsfragment, Altes Verlaufsfragment und Verlaufsquelle. Die Quelle gibt an, wodurch die Änderung ausgelöst wurde, etwa `pushState` oder `popstate`.

Eine häufige Falle sind Filter und Sprungmarken, die ebenfalls die URL ändern. Ohne Einschränkung feuert der Trigger dann auch dort und erzeugt Seitenaufrufe, die keine sind. Eine Bedingung auf den Pfad statt auf die vollständige URL hilft meistens.

Wo die Anwendung ein eigenes Ereignis in die Datenschicht pushen kann, ist das der sauberere Weg: Dort lässt sich der Zeitpunkt steuern, und Titel sowie Pfad der neuen Ansicht kommen gleich mit.

Veröffentlichen

Veröffentlichen schaltet eine Version des Containers live. Erst danach wirken Änderungen auf der Website, alles davor ist Entwurf.

Technisch wird beim Veröffentlichen die Datei `gtm.js` auf Googles Servern neu geschrieben. Da sie mit kurzer Cache-Zeit ausgeliefert wird, ist die Änderung innerhalb weniger Minuten bei allen Besuchern aktiv. Ein Deployment der Website ist dafür nicht nötig, und genau das ist der Grund, warum Tracking im Tag Manager und nicht im Theme liegt.

Vor dem Veröffentlichen gehört ein Durchgang im Vorschaumodus. Der Dialog zeigt außerdem, welche Objekte sich gegenüber der zuletzt veröffentlichten Version geändert haben.

Wird eine Umgebung genutzt, lässt sich eine Version zuerst dorthin veröffentlichen und erst nach der Abnahme in die Live-Umgebung. Ohne Umgebungen gibt es nur einen Zustand: veröffentlicht oder nicht.

Version

Eine **Version** ist ein eingefrorener Stand des Containers. Sie entsteht aus einem Arbeitsbereich und enthält alle Tags, Trigger, Variablen und Ordner zum Zeitpunkt der Erstellung.

Versionen werden fortlaufend nummeriert und können Name und Beschreibung tragen. Beides lohnt sich: Eine Liste aus „Version 34“, „Version 35“, „Version 36“ hilft ein halbes Jahr später niemandem, „GA4 purchase auf Datenschicht umgestellt“ dagegen schon.

Eine Version zu erstellen veröffentlicht sie noch nicht. Beides ist im selben Dialog möglich, aber getrennt: „Version erstellen“ speichert nur, erst „Version veröffentlichen“ schaltet sie live. Diese Trennung ist der Grund, warum sich Änderungen über die Tag Manager API automatisiert vorbereiten und trotzdem von Hand freigeben lassen.

Jede frühere Version lässt sich ansehen, mit der aktuellen vergleichen und wiederherstellen. Beim Wiederherstellen wird der Inhalt der alten Version in den Arbeitsbereich geladen; live geht er erst mit einer neuen Veröffentlichung. Das ist der schnellste Weg zurück, wenn eine Änderung das Tracking zerlegt hat.

Vom Nutzer bereitgestellte Daten

Vom Nutzer bereitgestellte Daten (User-Provided Data) ist ein Variablentyp, der Kontaktdaten für erweiterte Conversions in Google Ads bereitstellt: E-Mail-Adresse, Telefonnummer, Name, Adresse.

Die Daten werden vor dem Versand im Browser gehasht, an Google geht also kein Klartext. Google gleicht die Hashes mit angemeldeten Konten ab und ordnet so Conversions zu, die über Cookies nicht mehr zuzuordnen wären.

Automatisch oder manuell

Die Variable kennt zwei Modi. „Automatische Erfassung“ durchsucht die Seite nach Mustern, die nach einer E-Mail-Adresse aussehen. Das ist unzuverlässig und greift auf beliebige Felder im DOM zu, auch auf fremde. „Manuelle Konfiguration“ bindet jedes Feld einzeln an eine Variable, üblicherweise an Werte aus der Datenschicht. Der manuelle Weg ist der richtige.

Bei WooCommerce liefert GTM4WP die Bestelldaten in der Datenschicht, sodass sich die Felder direkt darauf binden lassen.

Die Übermittlung hängt am Einwilligungstyp `ad_user_data`. Ohne diese Zustimmung sendet das Tag keine Kontaktdaten, auch wenn sie konfiguriert sind. Ob die Zuordnung funktioniert, zeigt Google Ads erst nach einigen Tagen in der Diagnose der Conversion-Aktion.

Vorlagengalerie

Die **Vorlagengalerie** (Community Template Gallery) ist das Verzeichnis der von Drittanbietern bereitgestellten Tag- und Variablen-Vorlagen. Erreichbar ist sie über Vorlagen → Galerie durchsuchen.

Eine importierte Vorlage erscheint danach als eigener Tag-Typ im Container, mit Eingabefeldern statt Code. Meta, LinkedIn, Microsoft Clarity, Hotjar, Matomo und die meisten Consent-Management-Tools sind dort vertreten.

Der Vorteil gegenüber Benutzerdefiniertem HTML liegt in der eingebauten Einwilligungsprüfung und darin, dass die Vorlage in einer Sandbox läuft und nur deklarierte Berechtigungen bekommt. Welche das sind, zeigt der Container vor dem Hinzufügen an.

Vorlagen werden nicht automatisch aktualisiert. Erscheint eine neue Fassung, meldet der Container das im Bereich Vorlagen, das Einspielen bleibt ein bewusster Schritt.

Vorlagen aus der Galerie stammen von ihren jeweiligen Anbietern, nicht von Google. Vor dem Einsatz lohnt ein Blick auf Herausgeber und angeforderte Berechtigungen.

Vorschaumodus

Der **Vorschaumodus** zeigt, was der Container auf einer Seite tatsächlich tut. Gestartet wird er über die Schaltfläche „Vorschau“ rechts oben, die eine Debug-Sitzung eröffnet und die angegebene URL in einem neuen Tab öffnet.

Gezeigt wird der Stand des aktuellen Arbeitsbereichs, nicht der veröffentlichte. Genau dafür ist der Modus da: Änderungen prüfen, bevor sie live gehen.

Aufbau

Links steht die Zeitleiste aller Ereignisse in der Reihenfolge ihres Auftretens. Klickt man ein Ereignis an, zeigen die Reiter rechts den Zustand zu genau diesem Zeitpunkt: ausgelöste und nicht ausgelöste Tags, der Inhalt der Datenschicht, die Werte aller Variablen und der Einwilligungsstatus.

Der Reiter „Einwilligung“ ist bei Consent-Problemen die wichtigste Ansicht. Er zeigt eine Zeile mit den Standardwerten und eine weitere mit dem Update nach dem Klick im Banner.

Technisch setzt der Modus ein Cookie im Browser. Er wirkt deshalb nur in dieser Browsersitzung und auf der Domain, die den Container lädt. Bei Cross-Domain-Strecken bricht die Verbindung an der Domaingrenze ab, sofern die zweite Domain den Container nicht ebenfalls lädt.

Z

Zonen

Zonen (Zones) erlauben es, einen Container aus einem anderen heraus zu laden. Die Funktion gibt es nur in Tag Manager 360, der kostenpflichtigen Variante.

Gedacht sind sie für größere Organisationen, in denen ein zentraler Container die Grundlage bildet und einzelne Abteilungen oder Länder eigene Container bekommen, ohne Zugriff auf den zentralen zu haben.

Der einbindende Container legt fest, unter welchen Bedingungen die Zone geladen wird und welche Tag-Typen darin erlaubt sind. Damit lässt sich etwa verhindern, dass ein untergeordneter Container Benutzerdefiniertes HTML ausführt.

In kleineren Setups gibt es die Funktion nicht, und sie wird dort auch nicht gebraucht. Wer mehrere Websites mit einem Container bedient, arbeitet stattdessen mit Bedingungen auf den Hostnamen und einer Nachschlagetabelle für die IDs.

Zusätzliche Einwilligungsprüfungen

Mit **zusätzlichen Einwilligungsprüfungen** wird ein Tag an bestimmte Einwilligungstypen gebunden. Die Option steht im Tag unter Einstellungen zur Nutzereinwilligung und heißt dort „Zusätzliche Einwilligung für das Auslösen des Tags erforderlich“.

Eingetragen wird eine Liste von Typen, etwa `analytics_storage` für ein GA4-Tag oder `ad_storage` für eine Google-Ads-Conversion. Stehen mehrere Typen in der Liste, müssen alle auf `granted` stehen. Es ist eine Und-Verknüpfung, keine Oder-Verknüpfung.

Wie die Prüfung abläuft

Der Trigger feuert zuerst, danach prüft der Container den Einwilligungsstatus. Fehlt eine Zustimmung, wird das Tag verworfen. Im Vorschaumodus taucht es dann unter „Nicht ausgelöste Tags“ mit dem Hinweis auf die fehlende Einwilligung auf, nicht unter den blockierten Triggern.

Der wichtigste Fallstrick

Ein verworfenes Tag wird **nicht nachgeholt**, wenn die Einwilligung später auf derselben Seite erteilt wird. Das Tag hängt an seinem Trigger, und der ist bereits vorbei. Ein GA4-Konfigurationstag auf „Initialisierung – Alle Seiten“ mit Prüfung auf `analytics_storage` feuert beim ersten Seitenaufruf nie, weil das Banner zu diesem Zeitpunkt noch offen ist. Der erste Seitenaufruf geht damit verloren.

Der übliche Ausweg: Basis-Tags nicht auf die Initialisierung legen, sondern auf das Kategorie-Ereignis des Consent-Tools, etwa `cmplz_event_statistics` bei Complianz. Dieses Ereignis feuert sowohl beim Laden mit gespeicherter Einwilligung als auch im Moment der Zustimmung. Die Tag-Auslöseoption steht dabei auf „Einmal pro Seite“, damit eine spätere Änderung der Einstellungen keinen zweiten Seitenaufruf erzeugt.

Blockierende Trigger mit dem Muster `.*` zusätzlich zur Einwilligungsprüfung sind der Weg aus der Zeit vor dem Consent Mode. Beides parallel zu pflegen verdoppelt die Logik, ohne etwas zu gewinnen.