Answering the Top Questions People Asked About Server-Side Tagging on Google

So that you don't need to use ChatGPT!

INDEX

Google released the server-side Google Tag Manager (SGTM) public beta on August 12, 2020. In a nutshell, server-side tagging entails running a Google Tag Management container on the server.

Server-side tagging is not a new concept, but it has become more popular in recent years due to its benefits and advantages over traditional client-side tracking.

Server-side tracking has been used for many years in the form of server logs, which record basic information about each request made to a web server, such as the URL, IP address, user agent, and response code. However, server logs do not provide detailed information about user behavior, such as clicks, form submissions, and other interactions.

Server-side tagging as a more advanced form of tracking has been around for some time, but it has gained popularity in recent years due to the increasing demand for more accurate and reliable data, as well as the rise of privacy concerns related to client-side tracking.

Today, many businesses are adopting server-side tagging as a way to improve their data collection, analysis, and optimization efforts. By moving tracking and analytics to the server-side, businesses can reduce the reliance on client-side scripts, improve data accuracy and reliability, and comply with privacy regulations such as GDPR and CCPA.

But how does it solve some of the biggest problems faced by the digital advertising and marketing industry?

Let's find out what questions people are searching on Google?

Server-side tagging (SST) is an alternate technique for monitoring user activity and gathering statistics in order to improve a service by embedding tags, scripts, or snippets of code into a website or mobile application.

The fundamental concept is identical to that of client-side tracking, with one extension: there is a cloud server container in the loop. The data is delivered first to sGTM and then then to the analytics services you choose, as opposed to sending event data straight to analytics services.

This code is executed on the server rather than in the user's browser, which can minimize the amount of time it takes for the page to load while still protecting important information about the user.

It takes control away from third parties like Google, Facebook and gives it to the first party, which is you, the website owner or operator. SST does this by hosting both the website and its users' data on a single, secure central server controlled by the website owner.

The (server-side) container will function as a proxy, relaying information between browsers and devices and the real endpoints that receive the data. Thus, it serves as a protective data filter between your consumers and third-party suppliers seeking to monitor data.

Since it functions as an HTTP API endpoint, any client browser, device, or other source that is able to communicate over the HTTP protocol is able to send questions to the server-side container itself.

Workers (Docker images) referred to as Clients are set up inside the Server container to listen for these incoming HTTP requests, which they then parse into a common event structure.

The clients then run a virtual container with the event data object. The tags, triggers, and variables respond to the event push similarly to "normal" Google Tag Manager.

All of the aforementioned activities take place within the server-side tagging environment.

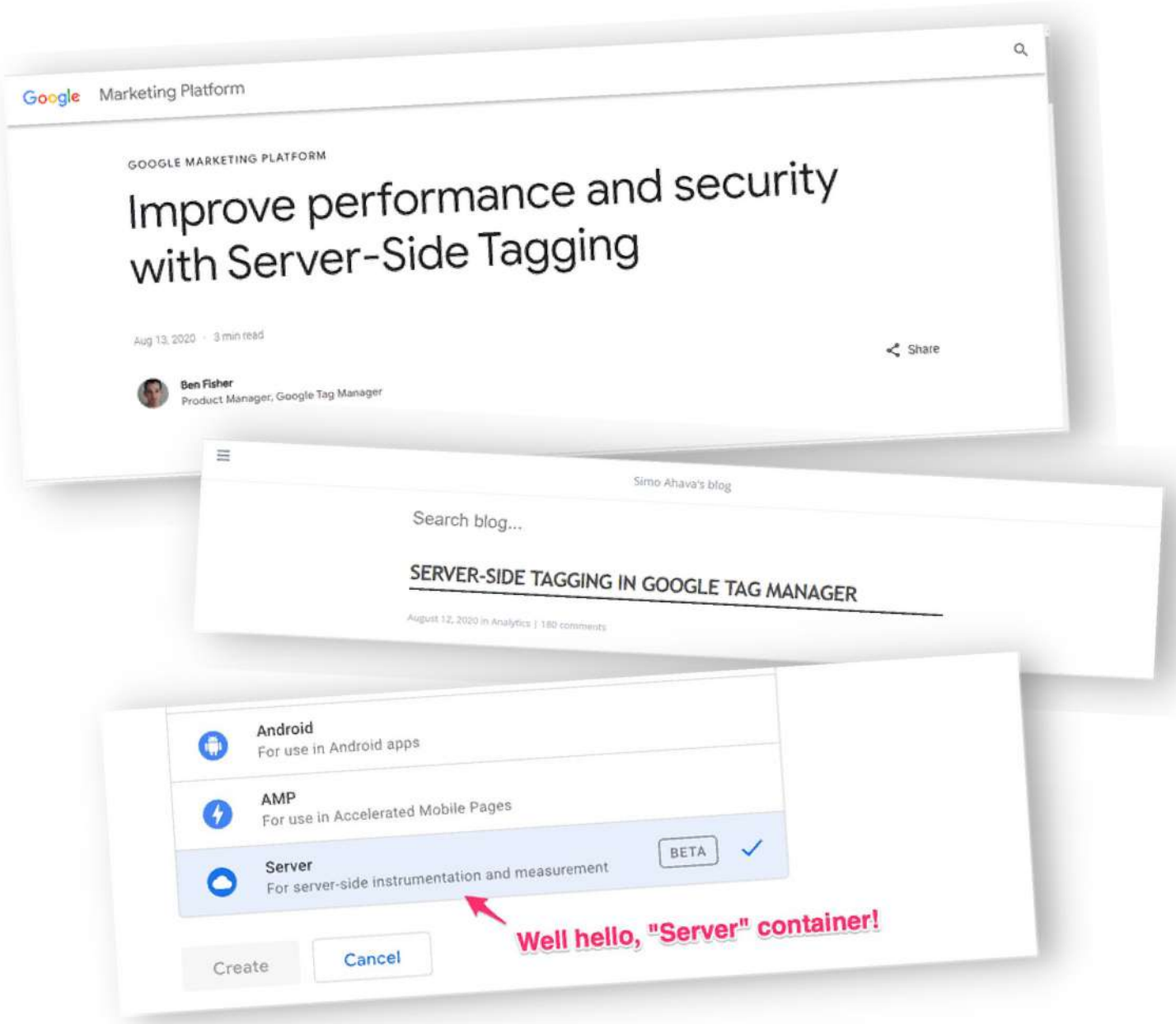
Note: *The browser or app sending the data will only know what's going on if the Client adds information to the HTTP response, which can be set up however the Client wants.*

Practically, this endpoint would be mapped with a custom subdomain that is part of the same domain hierarchy as the website that is submitting the queries.

This would be done in order to ensure that the correct information is returned and most ad blockers, browsers currently don't impose tag-based restrictions on first-party subdomains.

The fact that the requests are handled in this manner causes them to be treated as having taken place in a first-party context. This has a variety of significant ramifications, one of which being the manner in which cookies can be read and written.

On August 12, 2020, Google unveiled the public beta of Google Tag Manager (SGTM), a server-based version of GTM.



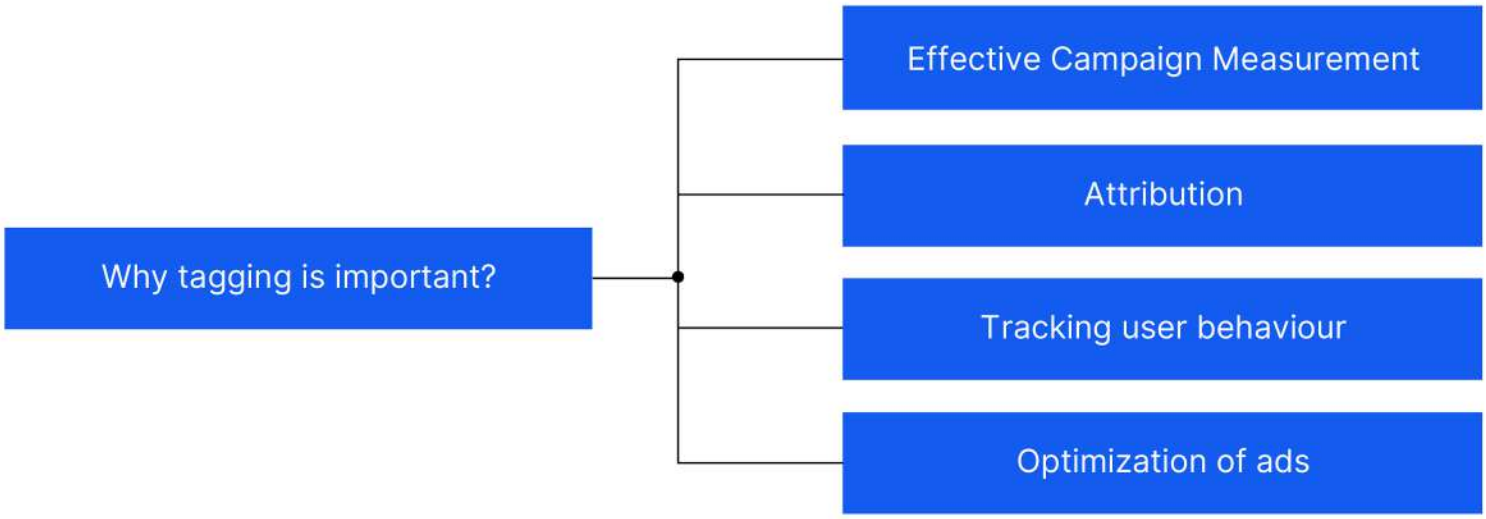
#3 What is tagging and why is it important?

Tagging in digital advertising refers to the **process of adding a piece of code or a "tag" to a website or advertisement in order to track user behavior and collect data about ad performance**. Tags are typically provided by third-party analytics or advertising platforms, and are used to collect information such as impressions, clicks, conversions, and other user interactions.

Tagging is important in digital advertising because it allows advertisers and marketers to measure the effectiveness of their campaigns, optimize ad performance, and make data-driven decisions about how to allocate their advertising budgets. By tracking user behavior and collecting data about ad performance, businesses can gain insights into user behavior and preferences, identify trends and patterns, and optimize their campaigns for better performance and ROI.

In addition, tagging is important for attribution, which refers to the process of assigning credit for a conversion or other user action to the appropriate ad or marketing channel. By using tags to track user behavior and conversions, businesses can accurately attribute conversions to specific ads or marketing channels, and adjust their campaigns accordingly.

Overall, tagging is a crucial component of digital advertising, as it enables businesses to measure and optimize their ad performance, make data-driven decisions, and maximize their ROI.



#4 What is the purpose of server-side tagging?

The purpose of server-side tagging is to track and analyze user behavior on a website or application by sending data from the server-side to a third-party analytics or marketing tool. Server-side tagging allows web developers to track user behavior and collect data without relying on client-side tracking scripts, which can improve the accuracy and reliability of the data collected.

Server-side tagging can be used to track a wide range of user interactions, such as clicks, form submissions, purchases, and other events. By collecting this data, businesses can gain insights into user behavior, identify trends, and optimize their website or application for better performance and user experience.

Server-side tagging can also improve website performance by reducing the amount of client-side code that needs to be loaded and executed on the user's device. This can lead to faster load times, better page responsiveness, and improved user engagement.

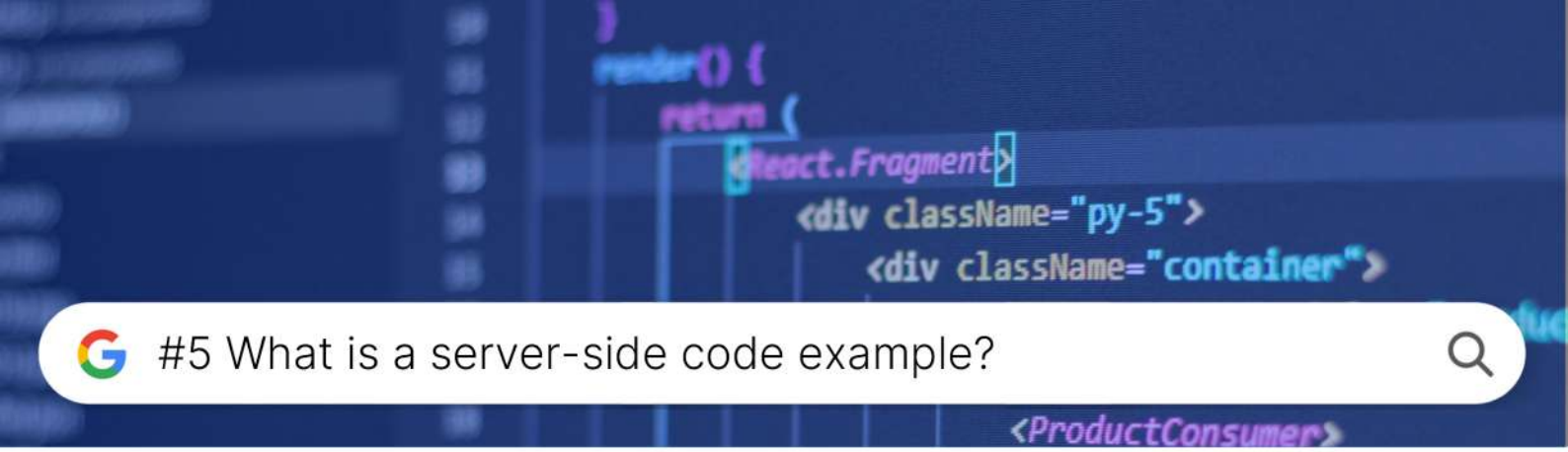
Overall, the purpose of server-side tagging is to enable businesses to collect accurate, reliable, and actionable data on user behavior, which can be used to drive business growth and improve the user experience.

Why server-side tagging?

User behavior tracking and data collection without client-side tracking scripts

Ability to track wide variety of data efficiently

Improved website performance due to reduction of client-side code



Here's an example of server-side tagging code in JavaScript using the Google Tag Manager (GTM) platform:

```
const { createServer } = require('http');
const { TagManager } = require('react-gtm-module');

const server = createServer((req, res) => {
 // Server-side rendering code here

 const tagManagerArgs = {
  gtmId: 'GTM-XXXXXX', // Replace with your own
GTM ID
  dataLayer: { // Add any custom data layer variables
   pageType: 'homepage',
   userId: '12345'
  },
  events: { // Add any custom events to trigger
   'some-event': 'triggered'
  }
 };

 // Server-side tagging code using the TagManager
module
 TagManager.initialize(tagManagerArgs);

 // Send server response
 res.writeHead(200, { 'Content-Type': 'text/html' });
 res.end(html);
});

server.listen(3000, () => {
 console.log('Server started on port 3000');
});
```

In this example, the **"createServer"** function from the Node.js **"http"** module is used to create a server that will render the website on the server-side. The react-gtm-module package is used to load and initialize the GTM container on the server-side.

The **"tagManagerArgs"** object contains the GTM ID, any custom data layer variables, and any custom events to trigger. The **"TagManager.initialize"** function is then called with the **"tagManagerArgs"** object to initialize the GTM container.

After the GTM container is initialized, the server sends the HTML response back to the client using the **"res.end"** method.

This is just an example and may not be suitable for all server-side tagging scenarios. It's important to consult the documentation for your specific tagging platform and programming language to ensure that you are using the correct syntax and methods.

#6 Why do I need server-side tracking?

There are several reasons why you may need server-side tracking for your website or mobile app:

- Increased data accuracy: Server-side tracking can provide more accurate data compared to client-side tracking, as it is not affected by ad-blockers or other client-side issues that can limit the accuracy of data being collected. This can help you make more informed decisions about your website or mobile app.

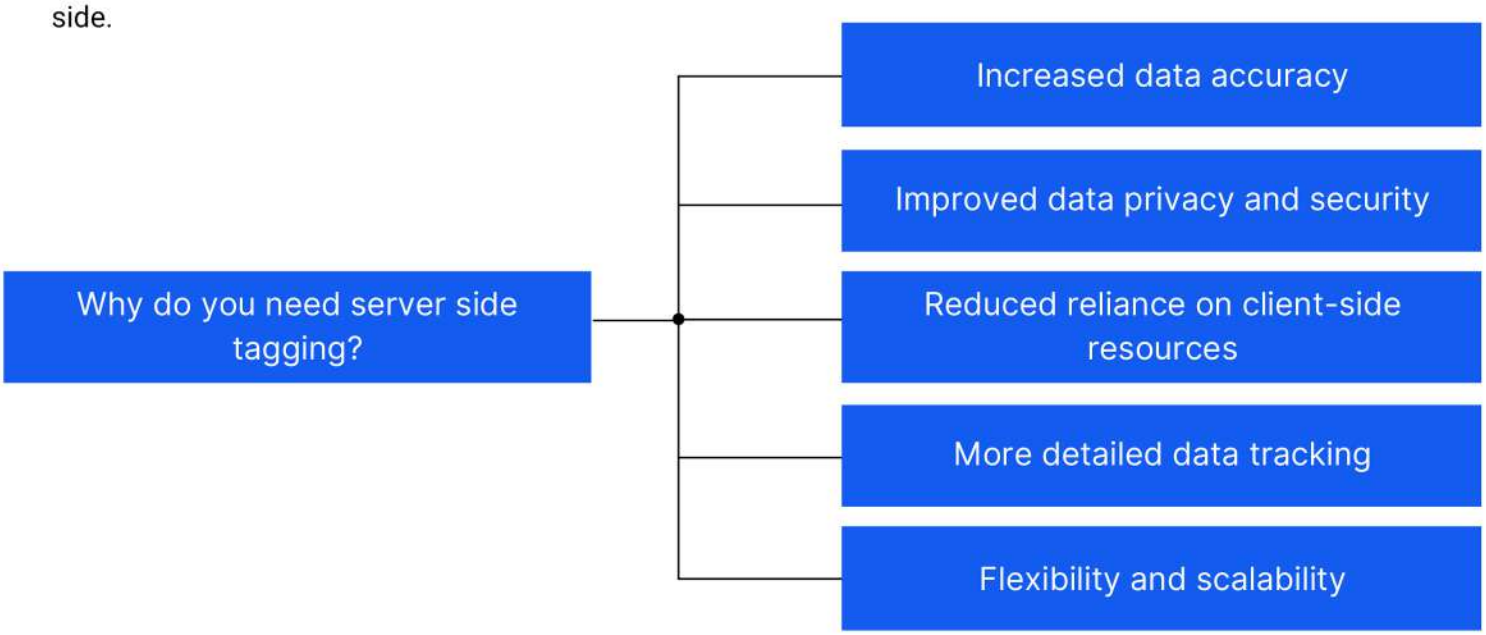
- Improved data privacy and security: Server-side tracking can help you maintain the privacy and security of your user's data, as it does not rely on client-side JavaScript code that can be intercepted by third-party scripts.

- Reduced reliance on client-side resources: Server-side tracking can help reduce the load on client-side resources, such as the user's browser or device, as tracking data is processed on the server-side.

- More detailed data tracking: Server-side tracking can allow you to capture more detailed data about user interactions with your website or mobile app, as it can capture data that is not available on the client-side, such as server-side events.

- Flexibility and scalability: Server-side tracking can provide more flexibility and scalability for your tracking needs, as it can be customized to suit your specific requirements and can handle larger amounts of data.

In summary, server-side tracking can provide more accurate data, improve data privacy and security, reduce reliance on client-side resources, allow for more detailed data tracking, and provide flexibility and scalability for your tracking needs.



#7 Should I use server-side tagging?

Here are some factors to consider:

- **Performance:** Server-side tagging can be a practical solution for website owners prioritizing speed and performance. Processing tags on the server allows to decrease the amount of JavaScript that must be loaded in the user's browser thereby improving the performance of your website.

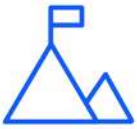
- **Data privacy:** Worried about data privacy and security? Server-side tagging may be the right solution for you because it can help reduce the risk of data breaches or intrusions. You can better manage what data is gathered and delivered to third-party suppliers by managing tag requests on the server side.

- **Tag management:** If you have a large website or app with many tags, server-side tagging can make it easier to manage and maintain tags. By centralizing all tags, you can more simply update and troubleshoot tags.

- **Technical expertise:** Server-side tagging requires more technical expertise than traditional client-side tagging, so if you don't have the resources or skills to manage server-side tagging, it may not be the best option for you unless you have Tagmate.

Overall, server-side tagging can offer benefits such as improved website performance, better data privacy and security, and easier tag management. However, it's important to weigh these benefits against your specific needs and goals before deciding whether or not to use server-side tagging.

Factors to consider before migrating to server-side tagging



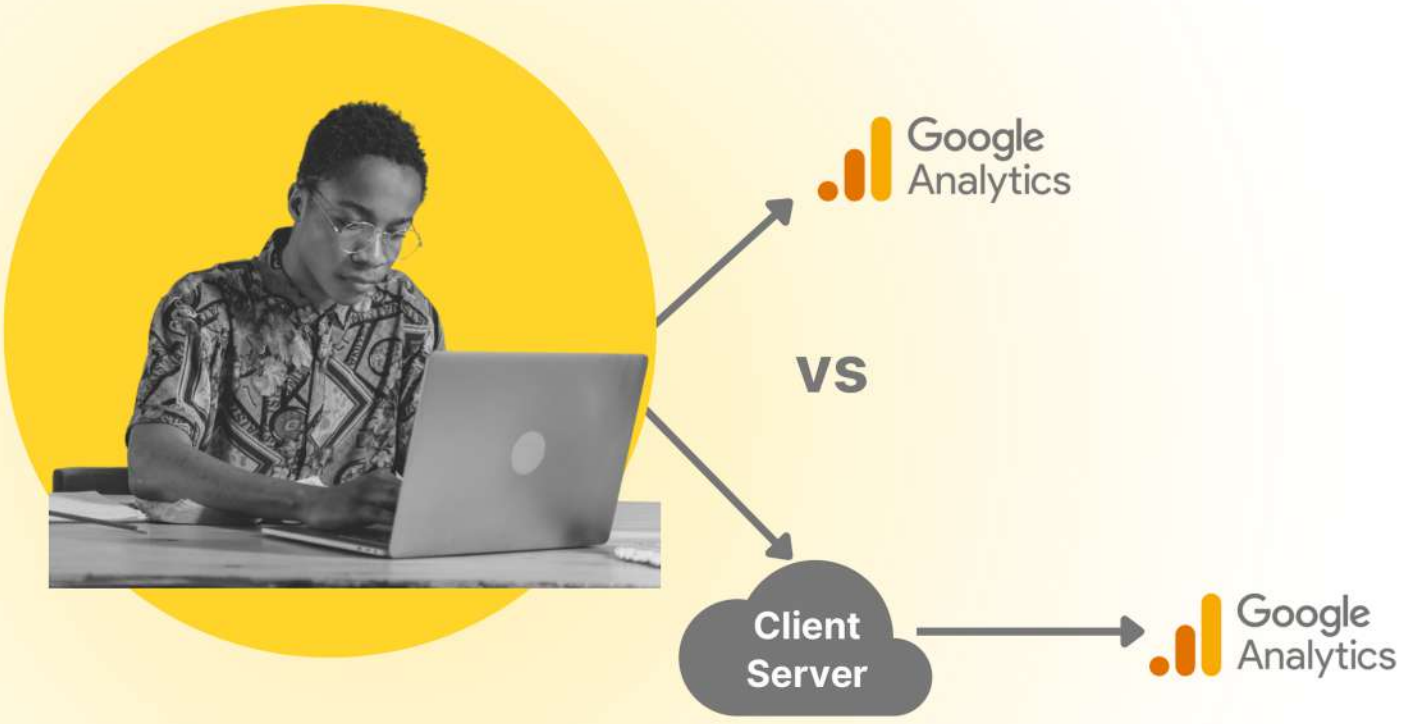
Website Performance

Data Privacy and Associated Risks

Tag Management

Requirement of Technical Expertise

Server-Side Tagging vs Client Side Tagging



#8 What is server-side tagging vs client-side tagging?

Server-side tagging and client-side tagging are two different methods of implementing digital analytics tracking on a website or mobile app.

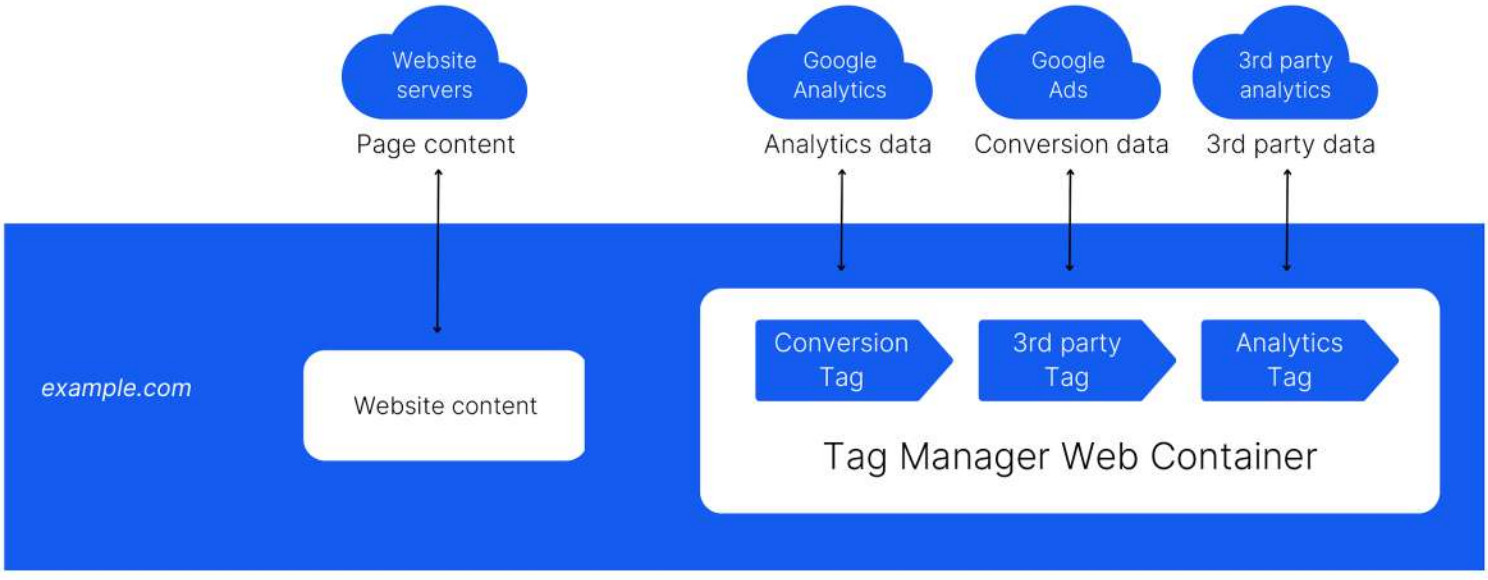
Client-side tagging involves placing a tracking code on the client-side of the website or mobile app, typically in the form of JavaScript code, that sends data to a third-party analytics platform when a user interacts with the website or app. The tracking code runs in the user's browser or device, and sends data to the analytics platform using client-side requests. Examples of popular client-side tagging solutions include Google Analytics and Adobe Analytics.

On the other hand, server-side tagging involves sending data to an analytics platform from the server-side of the website or mobile app, typically using server-side requests, rather than relying on client-side JavaScript code. This means that the tracking code is executed on the server, rather than in the user's browser or device.

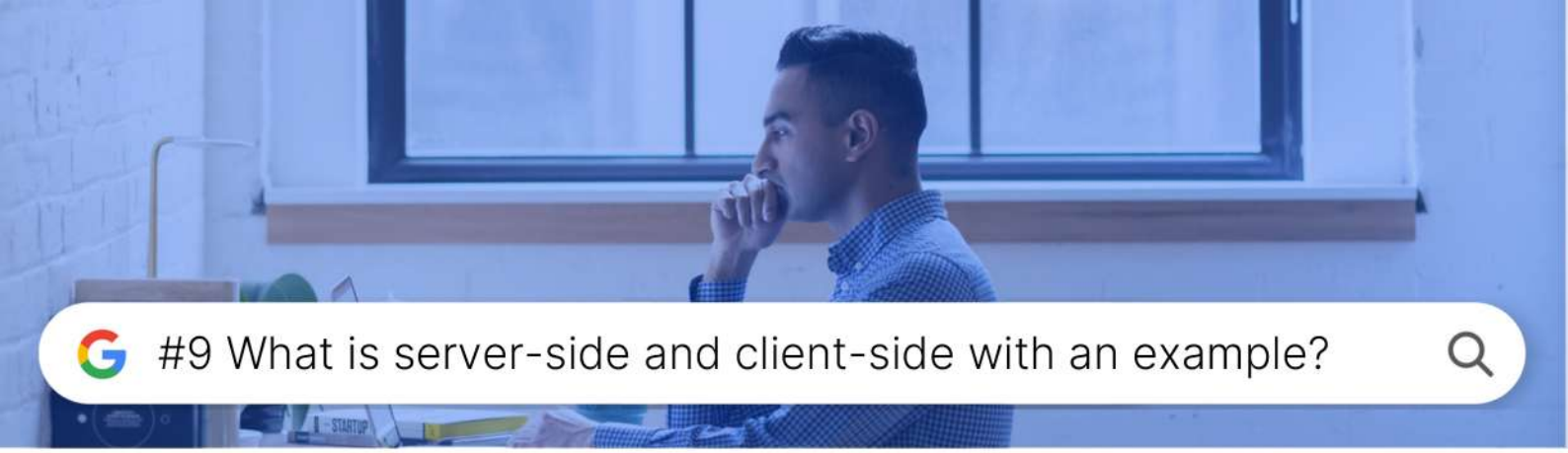
Examples of popular server-side tagging solutions include Snowplow and Segment.

The main difference between server-side tagging and client-side tagging is where the tracking code is executed, and how data is sent to the analytics platform. Server-side tagging can offer advantages such as improved data privacy and security, reduced reliance on client-side JavaScript, and the ability to send more detailed data to the analytics platform.

However, it can also be more complex to implement and maintain compared to client-side tagging, and may require more resources on the server-side. Client-side tagging is generally easier to implement and can provide real-time data tracking, but it can also be affected by ad blockers and other client-side issues that can limit the accuracy of the data being collected.



Client Side Tagging



#9 What is server-side and client-side with an example?

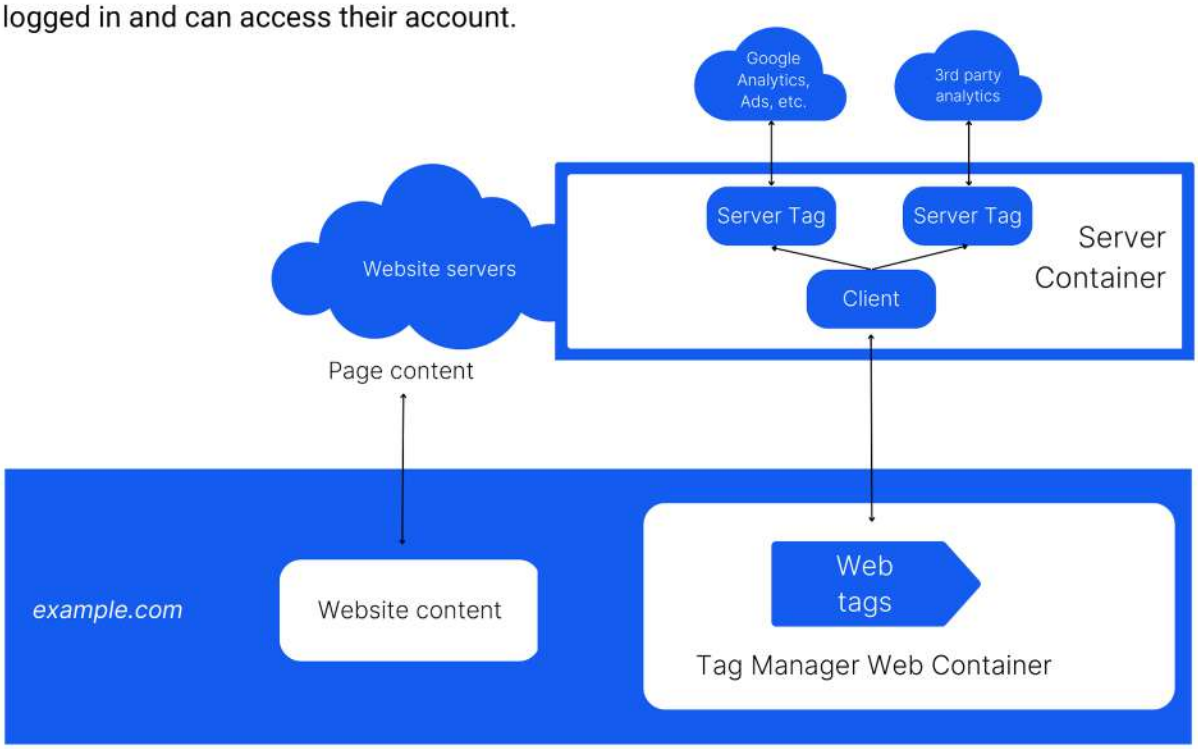
Server-side and client-side are two terms used in web development to describe where code is executed. Here's an example to illustrate the difference:

Let's say you have a website with a login form that asks for a username and password. When a user enters their username and password and clicks the login button, the website needs to check if the credentials are valid and allow access to the user's account.

Server-side example: In a server-side implementation, the login form is sent to the server, where the server-side code (e.g. PHP, Ruby, Python) processes the form data and verifies the user's credentials against a database. If the credentials are valid, the server sends back a response to the user's browser indicating that the user is logged in and can access their account.

Client-side example: In a client-side implementation, the login form is sent to the user's browser, and client-side code (e.g. JavaScript) processes the form data and validates the user's credentials locally on the browser. If the credentials are valid, the client-side code sends a request to the server to log in the user and display the appropriate content.

In summary, server-side refers to code that is executed on the server, while client-side refers to code that is executed on the user's browser. In the example above, server-side code is used to process the login form on the server, while client-side code is used to handle the form submission and display the results to the user.



Server Side Tagging

Tagmate

Web Tagging made lightning fast

Who said you need hours to set up web tags and marketing pixels?
Do it in a flash with Tagmate.
And save 25x time.

Products

- GA4 tag setup
- UA to GA4 tag migration
- Fb and Google Ads pixel setup
- Server side tagging for FCAPI and Google Ads

visit www.tagmate.app to simplify your tag management

#10 How do you tell if a rule is server-side or client side?

To determine if a rule is server-side or client-side, you need to understand where the rule is being executed.

A server-side rule is executed on the server, before the web page is sent to the user's browser. It can be identified by looking at the code that generates the web page or application, typically on the server-side. Server-side rules can be implemented using server-side scripting languages like PHP, Python, or Ruby.

A client-side rule, on the other hand, is executed on the user's browser or device, after the web page has been received by the browser. It can be identified by looking at the code that is executed by the browser, typically written in client-side scripting languages like JavaScript, HTML, or CSS.

Here are some ways to tell if a rule is server-side or client-side:

- Look at the code: If the code is written in a server-side language, like PHP or Ruby, it is likely a server-side rule.

If the code is written in a client-side language, like JavaScript or HTML, it is likely a client-side rule.

- Check where the rule is being executed: If the rule is being executed on the server, it is likely a server-side rule. If the rule is being executed in the user's browser or device, it is likely a client-side rule.

- Check the purpose of the rule: Server-side rules are typically used to process data, perform calculations, or retrieve information from a database. Client-side rules are typically used to modify the appearance or behavior of the web page or application, respond to user actions, or track user behavior.

In summary, to determine if a rule is server-side or client-side, you need to examine the code and understand where it is being executed, and for what purpose.

	Server Side Tagging	Client Side Tagging
Code	The code is written in PHP or Ruby.	The code is written in JavaScript or HTML.
Execution of rule	The rule is executed on the server.	The rule is executed on user's browser or device.
Purpose of rule	The purpose is to process data, perform calculations, or retrieve information from a database.	The purpose is to modify the appearance or behavior of the web page or app, respond to actions, or track user behavior.

Benefits of Server-Side Tagging

#11 Why is the server-side better for SEO?

Server-side tracking does not directly impact SEO, as search engines do not consider tracking methods when evaluating a website's ranking.

However, server-side tracking can indirectly improve your SEO by providing you with more accurate data on user behavior and website performance.

Server-side tracking can indirectly impact SEO in a few ways:

- **Site speed:** Server-side tracking can help improve site speed by reducing the amount of code that needs to be loaded on the client-side. Faster load times can improve user experience, which in turn can positively impact SEO.

- **Troubleshooting:** It can help to identify and resolve technical issues that may negatively impact a website's performance. For example, server-side tracking can help to detect server errors, page speed issues, and broken links, which can all affect a website's ranking on search engines.

- **Data accuracy:** Since it involves collecting data directly on the server that hosts a website, rather than through client-side scripts that run on a user's browser, it can provide more accurate data than client-side tracking, which can help identify SEO opportunities and optimize content accordingly.

This method can potentially provide more accurate data because it is not affected by ad blockers or privacy settings that may block or limit tracking cookies and scripts.

- **Security:** Server-side tracking can provide an extra layer of security by keeping sensitive data, such as user information and tracking data, on the server-side rather than on the client-side.

However, it's important to note that server-side tracking can also have potential drawbacks.

For instance, if the poorly implemented tracking code could potentially lead to slow down the website's performance, which could negatively impact SEO rankings.

Additionally, server-side tracking may not be able to capture certain types of data that are only available through client-side tracking, such as information about user clicks and mouse movements.

#12 Is server-side tracking better?

Whether server-side tracking is better than client-side tracking depends on your specific needs and the context of your website or mobile app. Here are some factors to consider when evaluating server-side tracking:

- **Data accuracy:** Server-side tracking can provide more accurate data than client-side tracking, as it is not subject to issues such as ad-blockers or other client-side limitations. This can be particularly important if you need highly accurate data for analytics or advertising purposes.
- **Security and privacy:** Server-side tracking can be more secure and private than client-side tracking, as it does not rely on client-side code that can be intercepted by third-party scripts. This can be particularly important if you are collecting sensitive user data.
- **Flexibility and customization:** Server-side tracking can be more flexible and customizable than client-side tracking, as you can control the logic of your tracking and have more control over the data you collect.
- **Development and maintenance:** Server-side tracking can be more complex to develop and maintain than client-side tracking, as it requires additional infrastructure and development resources.

Overall, server-side tracking can provide more accurate, secure, and customizable tracking compared to client-side tracking. However, it may also require additional development and maintenance resources, and may not be necessary for all websites or mobile apps. It's important to evaluate your specific needs and goals to determine whether server-side tracking is the right choice for your website or mobile app.

Factors to consider when evaluating server-side tagging

More accurate data with server-side tagging

Secure and private than client-side tracking

Flexibility and customization of logic is present

Relatively complex to develop and maintain

#13 Is server-side more secure?

In general, server-side code can be more secure than client-side code, due to the following reasons:

- **Reduced attack surface:** Server-side code is executed on the server, which is typically better protected and more difficult to access than client-side code running on the user's device. This reduces the attack surface that can be exploited by attackers.
- **Better control over user input:** With server-side code, the web application has more control over user input, and can validate and sanitize user input before it is processed. This can help prevent security vulnerabilities such as SQL injection, cross-site scripting (XSS), and other types of attacks.
- **Protection of sensitive data:** Server-side code can be used to protect sensitive data by storing it securely on the server, rather than transmitting it to the user's device where it may be vulnerable to theft or interception.
- **Easier updates and patches:** Server-side code can be updated and patched more easily than client-side code, which may require users to manually update their devices or browsers.

That being said, server-side code can still be vulnerable to security risks if it is not designed and implemented properly. Attackers can still find ways to exploit vulnerabilities in server-side code, such as through misconfigured servers, outdated software, weak passwords, and other security weaknesses.

Therefore, it is important to follow security best practices when designing and implementing server-side code, and to keep up-to-date with the latest security patches and updates.

Reduced attack surface because of data being on owner's server

More control over user input and ability to process user input before it is processed

Protection of sensitive data due to its storage on a secured server

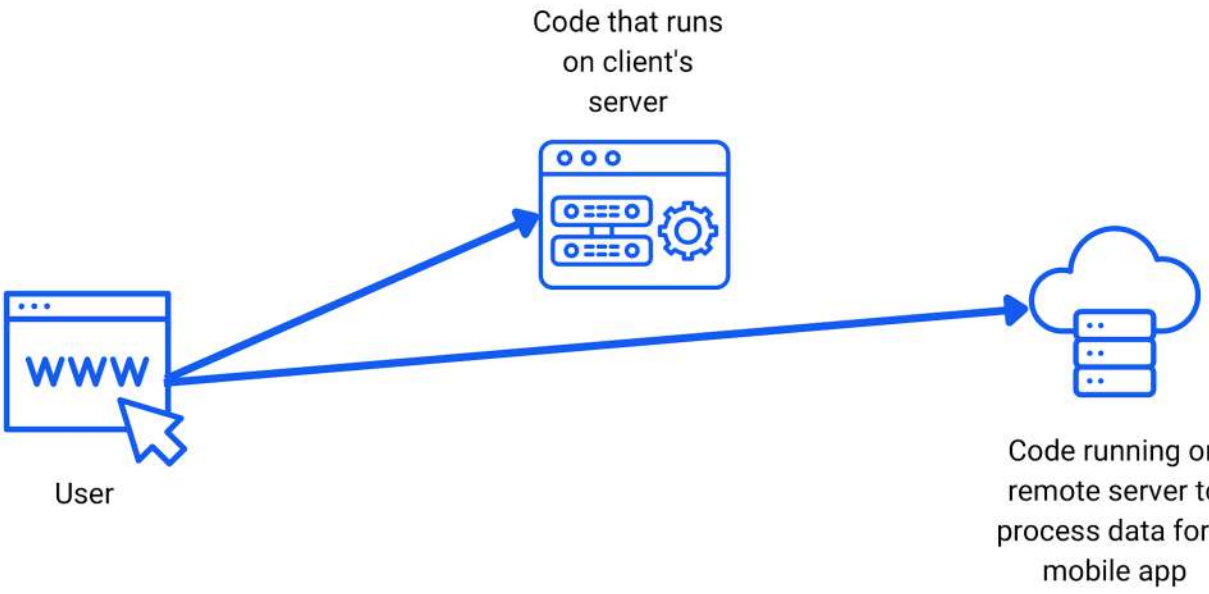
Easier updates and patches than client side code

Getting Technical On The Server-Side

Server-side code generally refers to code that runs on the server, as opposed to code that runs on the client (i.e. the user's browser). In web development, this typically refers to the backend of a website or web application.

The backend is responsible for managing data storage, processing requests from the client, and generating responses to be sent back to the client. This often involves using server-side programming languages and frameworks such as PHP, Python, Ruby on Rails, or Node.js.

So while server-side code is often associated with the backend of a web application, it's important to note that not all server-side code is necessarily backend code. For example, server-side code could also refer to code that runs on a remote server to process data for a mobile app or another type of application.



#15 How does S2S tracking work?

Server-to-server (S2S) tracking is a method of tracking user activity and events on a web application that is performed entirely on the server-side, without relying on client-side scripts or third-party cookies. Here's how S2S tracking typically works:

- **User activity:** A user interacts with a web application by visiting web pages, submitting forms, or performing other actions.
- **Server-side tracking code:** The web application includes a server-side tracking code that is executed on the server when the user performs an action. This tracking code is typically implemented in a server-side programming language such as PHP, Python, or Java.
- **Event data:** The tracking code captures information about the user's action, such as the page they visited, the form they submitted, or the button they clicked. This data is stored on the server as a log or database entry.
- **Reporting:** The stored event data can be used for reporting and analysis purposes, such as tracking user behavior, measuring website performance, or optimizing advertising campaigns.

S2S tracking can offer several benefits over traditional client-side tracking methods, including increased security and privacy, improved accuracy and reliability, and reduced dependence on client-side technologies.

However, S2S tracking can also be more complex to implement and maintain than client-side tracking, and may require specialized server-side programming skills and tools.

Here's a real-life example of how S2S tracking might be used in a web application:

Suppose that you operate an e-commerce website and want to track user purchases for marketing and analytics purposes. Instead of relying on client-side tracking scripts or third-party cookies, you decide to implement S2S tracking to capture user purchase data on the server.

Here's how the S2S tracking process might work:

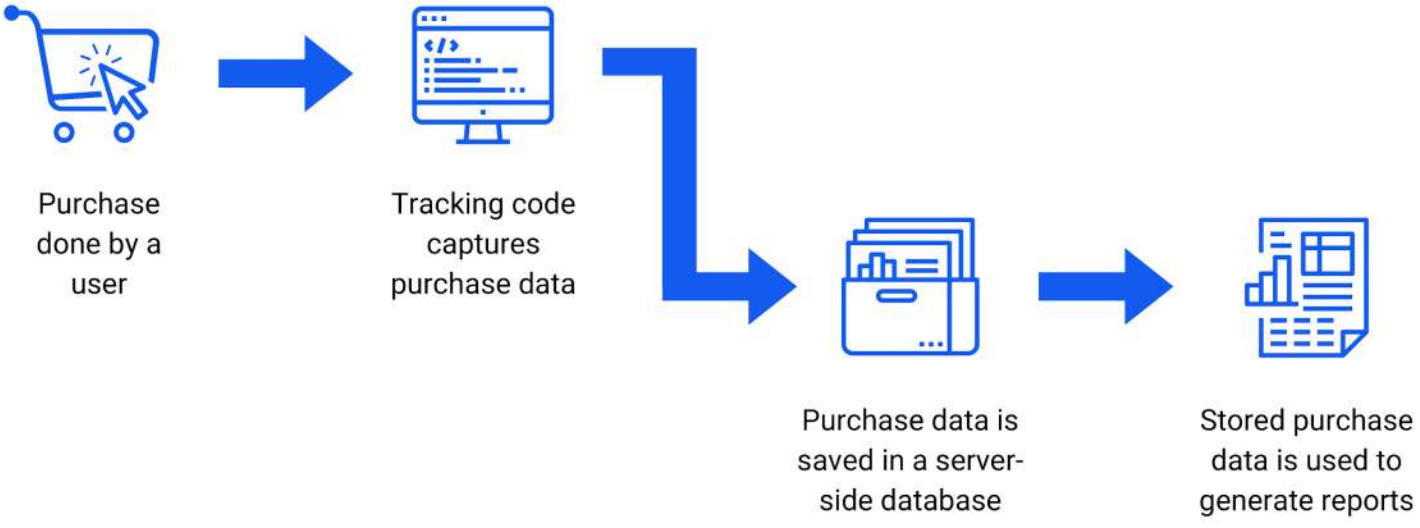
- **User purchases:** A user visits your e-commerce website and makes a purchase by adding items to their shopping cart and completing the checkout process.
- **Server-side tracking code:** As the user performs these actions, the web application's server-side tracking code captures data about the purchase, such as the products purchased, the purchase amount, and the user's contact information. This tracking code is executed on the server using a server-side programming language such as PHP or Python.

#15 How does S2S tracking work?

- **Event data:** The tracking code stores this purchase data in a server-side database or log file. The data might include information such as the user's name, email address, purchase date and time, product names, product prices, and payment information.

- **Reporting:** The stored purchase data can be used to generate reports and analyses about user purchasing behavior, such as total sales revenue, top-selling products, average order value, and customer demographics. This information can be used to optimize the e-commerce website's marketing and advertising campaigns, as well as to identify opportunities for improving the user experience and increasing customer satisfaction.

By using S2S tracking, the e-commerce website can avoid relying on client-side technologies that may be blocked by users' browsers or subject to privacy concerns. Instead, the server-side tracking code captures purchase data directly on the server, providing a more reliable and secure method of tracking user behavior.



Flowchart of an example of S2S tracking process

#16 How does server-side authentication work?

Server-side authentication is the verification of a user's identification on the server-side of a web application. The following are the general activities involved in server-side authentication:

- **User authentication request:** When a user tries to access a secure page or API endpoint on a web application, they are usually asked to authenticate themselves by entering a username and password or other authentication credentials.

- **Server-side authentication process:** When the server receives user credentials, server-side authentication starts. After that, the server checks the credentials against the user's information it already has on file to make sure they're legitimate. This may include comparing the user's provided password to the saved password hash or comparing the user's credentials to an external authentication service like OAuth.

- **Authentication response:** If the user's credentials are correct, the server-side authentication procedure will normally create an authentication token or session ID, which will be kept on the server and connected with the user's identity. The server then returns the token or session ID to the user's browser.

- **Subsequent requests:** When the user makes subsequent requests to the web application, they typically include the authentication token or session ID in the request headers or cookies. The server then verifies the token or session ID to ensure that the user is still authenticated and authorized to access the requested resource.

If the user logs out of the web application, the server can invalidate the authentication token to prevent it from being used again. Similarly, if the user's authentication token is compromised or stolen, the server can invalidate the token to prevent unauthorized access to the user's account.

To avoid brute force attacks, server-side authentication may include extra security features such as encrypting the authentication credentials during transmission and implementing additional security checks such as rate limitation and IP blocking.

The details of server-side authentication can change depending on the web application and the specific authentication method used.

#17 What are disadvantages of server-side authentication?

Server-side authentication has several potential disadvantages that developers should consider when deciding whether to use it for their web application:

- **Increased Server Load:** With server-side authentication, every user authentication request requires the server to do more work. This can make the server work harder, especially for web apps with a lot of users or a lot of traffic. To fix this problem, developers may need to improve their code for server-side authentication or add more servers to handle the load.

- **Higher Latency:** Because server-side authentication requires a round-trip to the server, it can add more latency to the authentication process. This can be especially noticeable for people using the web app from far away or with slow internet connections.

- Server-side authentication can be harder to set up than client-side authentication, especially for developers who aren't familiar with server-side programming languages or authentication mechanisms. This can make it harder to make and keep the authentication code up-to-date over time.

- **Possible Security Flaws:** Authentication on the server side can be safer than authentication on the client side, but it is not immune to security flaws. For example, attackers might be able to bypass authentication or steal user credentials by taking advantage of flaws in the authentication code. Developers need to be careful about keeping their server-side authentication code secure and up-to-date with the latest security patches and best practices.

- **Problems with Compatibility:** Server-side authentication might not work with all web browsers and devices. This can make it harder for some people to use the web app, especially if they are using older or less common devices.

Why server side authentication can be a bad idea for you?

Increases the server load

Adds latency to the authentic process

Security flaws still persist

Might not be compatible with all browsers and devices

#19 What is server-side rendering and why do we need it?

Server-side rendering (SSR) is the process of rendering a web page on the server and sending the fully rendered page to the client's browser, instead of sending an empty HTML file that is then populated with content using client-side JavaScript. SSR can be used for both static and dynamic websites, and can be implemented using various server-side technologies such as Node.js, Ruby on Rails, and Django.

There are several reasons why we need server-side rendering:

- **Improved page load times:** By rendering the page on the server and sending a fully rendered page to the client's browser, SSR can improve page load times, as the client does not have to wait for JavaScript to download and execute before seeing the content.
- **Better SEO:** Search engines such as Google and Bing prefer sites with server-side rendered content, as it allows them to easily crawl and index the content, resulting in better search engine rankings.
- **Accessibility:** SSR can improve accessibility for users with disabilities, as it ensures that content is available and rendered properly even if the user has JavaScript disabled or is using a screen reader.
- **Performance:** SSR can improve performance by reducing the load on the client-side JavaScript engine, as less JavaScript code is required to render the page.

Overall, server-side rendering is a valuable technique for improving website performance, SEO, accessibility, and user experience. It allows websites to deliver fully rendered pages to clients quickly and efficiently, resulting in a better user experience and improved search engine rankings.

#20 When should I use server-side paging?

You should consider using server-side paging when you have a large amount of data that needs to be displayed on a web page, such as a table or a list, and you want to improve the performance and user experience of your application.

Server-side paging involves retrieving a subset of data from a database or other data source, based on a specific page size and page number, and returning only the data that is needed for the current page. This approach can improve the performance of your application by reducing the amount of data that needs to be transferred and processed by the client, and by reducing the load on the server.

Here are some situations where server-side paging can be particularly useful:

- **Large datasets:** If you have a large amount of data to display, server-side paging can help reduce the amount of data that needs to be loaded and processed by the client, improving performance and reducing server load.
- **Interactive UI:** If you have an interactive UI, such as a data grid or a list with sorting and filtering capabilities, server-side paging can allow you to load only the data that is needed for the current view, improving performance and reducing the load on the server.
- **Slow database queries:** If your database queries are slow or complex, server-side paging can help improve performance by reducing the amount of data that needs to be processed and returned by the database.

Overall, server-side paging can be a useful technique for improving the performance and user experience of your web application when working with large datasets, interactive UIs, or slow database queries.

Tagmate

Web Tagging made lightning fast

Who said you need hours to set up web tags and marketing pixels?
Do it in a flash with Tagmate.
And save 25x time.

Products

- GA4 tag setup
- UA to GA4 tag migration
- Fb and Google Ads pixel setup
- Server side tagging for FCAPI and Google Ads

visit www.tagmate.app to simplify your tag management