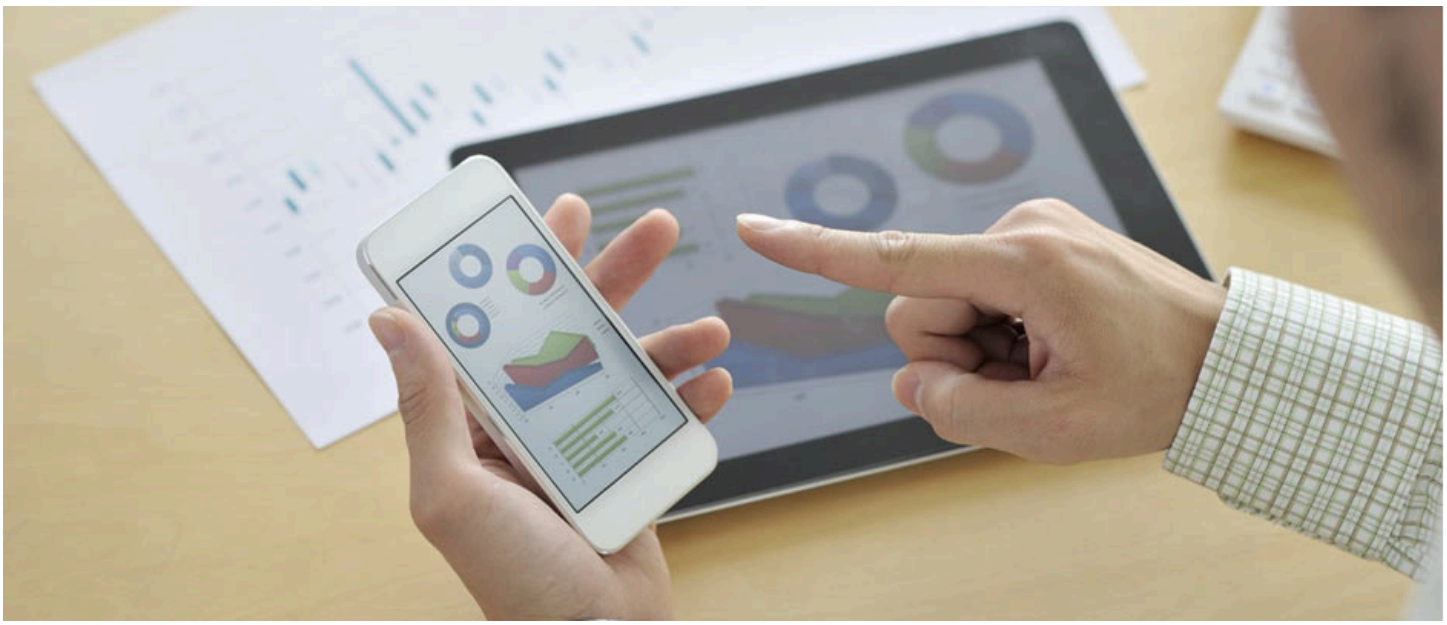
scientiamobile

"The first step in a great mobile experience"



SCIENTIAMOBILE WURFL.JS LITE GETTING STARTED GUIDE

Support

The ScientiaMobile Enterprise Support Portal is open to all WURFL users, both commercial license holders and evaluation users. It represents the combined knowledge base for the WURFL community. Commercial licensees are invited to post questions in the forum using the account to which their licenses are associated. This may mean faster handling of those posts by ScientiaMobile's personnel.

For commercial license holders, there are tiered support levels to address a variety of business support needs. After logging into your account, commercial licensees with support options can access the Enterprise Support portal to post tickets. These tickets will receive expedited attention.

To inquire about support plans, use our License Inquiry or our General Inquiry form.

Update Notifications

If you would like to be notified of our API updates, major data updates, and other technical changes, please subscribe to our ScientiaMobile Announcements list.

scientiamobile

www.scientiamobile.com

Tel +1.703.310.6650

E-mail: sales@scientiamobile.com

Copyright © 2026 ScientiaMobile, all rights reserved. WURFL Cloud, WURFL OnSite, WURFL and, InFuze WURFL InSight and respective logos are trademarks of ScientiaMobile. Apache is the trademark of the Apache Software Foundation. NGINX is the trademark of Nginx Software Inc. Varnish is the trademark of Varnish Software AB

ScientiaMobile WURFL.js Lite Getting Started Guide

Introduction

WURFL.js provides access to ScientiaMobile's cloud-based WURFL Device Description Repository (DDR). With a single JavaScript snippet, developers can detect a device and its capabilities in order to better control, optimize, and track the success of their website.

Only **WURFL.js Basic, Standard, and Pro** services will detect iPhone and iPad models. The free WURFL.js Lite will only return generic iPhone and iPad results.

Important Note: *More recent iPads will impersonate a desktop browser through the User-Agent string. WURFL.js Basic, Standard and Pro services will still be able to detect the device as an iPad (thanks to sophisticated additional logic), and return information about the specific model. The free version (WURFL.js Lite) will handle those devices as desktop browser*.

Looking for more device and browser information? Upgrading will provide more than 20 additional device capabilities, including

- Browser and browser version
- Operating system and OS version
- Screen resolution height and width

In addition, you can customize your WURFL.js instance to include even more device capabilities useful for Advertising, Web Optimization, OTT Video, eCommerce, Gaming, and Security applications. Either **upgrade to a paid subscription** or **contact our sales team** to discuss your use case and customize a plan that fits your needs.

Quick Start

Just copy the WURFL.js Lite snippet and paste it into your HTML markup. WURFL will return a JSON object with 3 properties describing the device visiting your website. Use these WURFL device capabilities in your website optimization, analytics, or advertising.

For iPhone detection and 20 more device capabilities, you can easily upgrade to a **Basic, Standard, or Pro** account.

```
<script src="https://wurfl.io/wurfl.js"></script>
```

For the best detection experience, we recommend loading WURFL.js via the async method, as detailed in the `Async loading` section below.

The `WURFL` object is now available to use. WURFL.js will return the WURFL object with 3 device capabilities as properties. These WURFL properties are:

- WURFL.complete_device_name (**learn more**)
- WURFL.form_factor (**learn more**)
- WURFL.is_mobile (**learn more**)

You can test if a device is a smartphone with this code:

```
if (WURFL.is_mobile === true && WURFL.form_factor === "Smartphone") {
    // targetSmartPhoneDevices();
}
```

If you **upgrade** to a Basic, Standard, or Pro plan, full iPhone detection will let you test if a device is an Apple iPhone 14 Pro with code like this:

```
if (WURFL.complete_device_name === "Apple iPhone 14 Pro") {
    // targetAppleiPhone14Pro();
}
```

You can push device information into **Google Analytics using dimensions** with code like this:

```
/**
 * Example of pushing WURFL capabilities to google analytics
 */
ga('send', 'pageview', {
        'dimension1': WURFL.complete_device_name,
        'dimension2': WURFL.form_factor,
        'dimension3': WURFL.is_mobile,
})
```

Async loading

WURFL.js dispatches an event called `WurflJSDetectionComplete` to the `document` node when it's finished. This allows you to include it in `async` mode and avoid DOM-blocking content. We *do not recommend* that the script tag be placed in the document's head element.

```
<script src="https://wurfl.io/wurfl.js" async></script>
```

In Javascript you can listen for the event:

```
document.addEventListener("WurflJSDetectionComplete", function(){
    console.log("WURFL.js is finished:")
    console.log(window.WURFL)
});
```

Please note that the event listener must be defined **before** the WURFL.js script tag is included to avoid a race condition.

For example, the Google Analytics code in the previous example could be fired only after WURFL.js is finished:

```
document.addEventListener("WurflJSDetectionComplete", function(){
        ga('send', 'pageview', {
                'dimension1': WURFL.complete_device_name,
                'dimension2': WURFL.form_factor,
                'dimension3': WURFL.is_mobile,
        })
})
```

Loading WURFL.js programmatically

Some users may want to call WURFL.js programatically from within Javascript or a framework like React or Angular. In these cases, care should be taken to avoid creating a race condition between the execution of the WURFL.js script and the event handler `WurflJSDetectionComplete` firing, which may lead to unpredictable results. Here is an example snippet that makes use of WURFL.js' `WURFLPromises` feature and demonstrates loading WURFL.js via Javascript script-tag injection. This example also shows a method for WURFL.js users to receive faster and more reliable detection results for requests from browsers that participate in the User-Agent Client Hints initiative. More information regarding User-Agent Client Hints is available in the `Consideration for User-Agent Client Hints for WURFL.js Lite` section below.

```
const wjsPromise = new Promise((resolve, reject) => {
        const url = new URL('https://wurfl.io/wurfl.js')
        const runWjs = (src) => {
                const wjs = document.createElement('script')
                wjs.src = src
                wjs.async = true
                wjs.onload = () => resolve(window.WURFLPromises.complete)
                wjs.onerror = () => reject()
                document.head.appendChild(wjs)
        }

        if ('userAgentData' in navigator) {
                const hints = ['architecture', 'bitness', 'model', 'platformVersion',
'uaFullVersion', 'fullVersionList']
                navigator.userAgentData.getHighEntropyValues(hints).then((uach) => {
                        url.searchParams.set('uach', JSON.stringify(uach))
                        runWjs(url.toString())
                })
        } else {
                runWjs(url.toString())
        }
})

wjsPromise.then((res) => {
        // WURFL is ready to use
        console.log(res.WURFL)
}).catch(() => console.error(`Failed to load WURFL.js`))
```

Execution time limit

For certain devices WURFL.js will run a number of tests to identify the device. The tests are usually fast and accurate. However, if you want WURFL.js to spend more time to make the results even more reliable, you can allow more execution time. This is done by appending "?time_limit=" to the script reference. The below example will allow WURFL.js to execute for up to one second:

```
<script src="https://wurfl.io/wurfl.js?time_limit=1000"></script>
```

Content Security Policy

If your website uses a Content-Security-Policy (CSP), you will need to add exemptions for WURFL.js for the `script-src` and the `connect-src` directives like so:

```
Content-Security-Policy: script-src https://wurfl.io; connect-src https://wurfl.io
```

Consideration for User-Agent Client Hints for WURFL.js Lite

As part of the **User-Agent reduction/freeze**, User-Agents from Chrome on desktop and Android platforms no longer contain information accurate enough to be solely used for device detection. As a result, we encourage you to implement and start requesting User-Agent Client Hints from your clients. We also recommend that you delegate these Client Hints to WURFLjs for the best detection experience.

Please note that User-Agent Client Hints are only available in a secure context (`https`).

In order to opt in to receive User-Agent Client Hints, you will need to set your server's response headers to advertise support for them using the `Accept-CH` header. Here's an example:

```
Accept-CH: sec-ch-ua-platform-version,sec-ch-ua-full-version,sec-ch-ua-full-version-list,sec-ch-ua-model,sec-ch-ua-arch,sec-ch-ua-bitness,sec-ch-ua-wow64
```

Next, you will want to delegate these Client Hints to WURFLjs so that they can be used as a part of the detection process. This can be achieved by setting a permissions-policy header:

```
permissions-policy: ch-ua-platform-version=(self "https://wurfl.io"),ch-ua-full-version=(self "https://wurfl.io"),ch-ua-full-version-list=(self "https://wurfl.io"),ch-ua-model=(self "https://wurfl.io"),ch-ua-arch=(self "https://wurfl.io"),ch-ua-bitness=(self "https://wurfl.io"),ch-ua-wow64=(self "https://wurfl.io")
```

If you are using a platform/CMS that restricts your ability to set server response headers, it may be easier to use the `delegate-ch` meta tag instead. Using this meta tag, you can request User-Agent Client Hints on WURFL.js's behalf and delegate them in one go:

```
<meta http-equiv="delegate-ch" content="sec-ch-ua https://wurfl.io; sec-ch-ua-bitness https://wurfl.io; sec-ch-ua-arch https://wurfl.io; sec-ch-ua-model https://wurfl.io; sec-ch-ua-platform https://wurfl.io; sec-ch-ua-platform-version https://wurfl.io; sec-ch-ua-full-version https://wurfl.io; sec-ch-ua-full-version-list https://wurfl.io; sec-ch-ua-mobile https://wurfl.io">
```

Additional information on adding support for User-Agent Client Hints are available in our guide **here**.

Accessing a User-Agent Client Hints enriched result

As mentioned in the `Quick Start` section above, we recommend loading WURFL.js asynchronously and listening to the `WurflJSDetectionComplete` event, which is generated once the device detection process is complete. Depending on the complexity of the environment and the method in which WURFL.js receives the User-Agent Client Hint data, this process can take some time to complete. As a result, if you desire intermediate device detection

information while you wait for the full device detection process to complete, you can listen to the `WurflJSInitComplete` event.

The `WurflJSInitComplete` event is generated with the information that is readily available and provides reasonable defaults until WURFL.js is able to consolidate device information from various sources and generate the `WurflJSDetectionComplete` event.

```
document.addEventListener("WurflJSInitComplete", function(){
        console.log("WURFL.js is initialized and can provide intermediate device
information:")
        console.log(window.WURFL)
});
```

Just as with the `WurflJSDetectionComplete` event, the event listener for the `WurflJSInitComplete` event must be defined **before** the WURFL.js script tag is included.

Enrich Google Analytics 4 (GA4) with WURFL.js capabilities

A popular use case for WURFL.js is to enrich the reporting provided by Google Analytics. This allows you to add capability values returned by WURFL.js (like `marketing_name` or `release_msrp`) to your Google Analytics reports. Please note that the WURFL capabilities or device attributes that are available be used are dependent on your WURFLjs plan. If you require the use of capabilities that are not part of your plan, please **contact our sales team** to create a custom plan to fit your needs.

This section will outline two easy methods to integrate WURFL.js with Google Analytics 4 (GA4). One way is to use the Google Tag Manager (GTM) and the alternative is via the Google Tag (aka gtag.js or simply gtag).

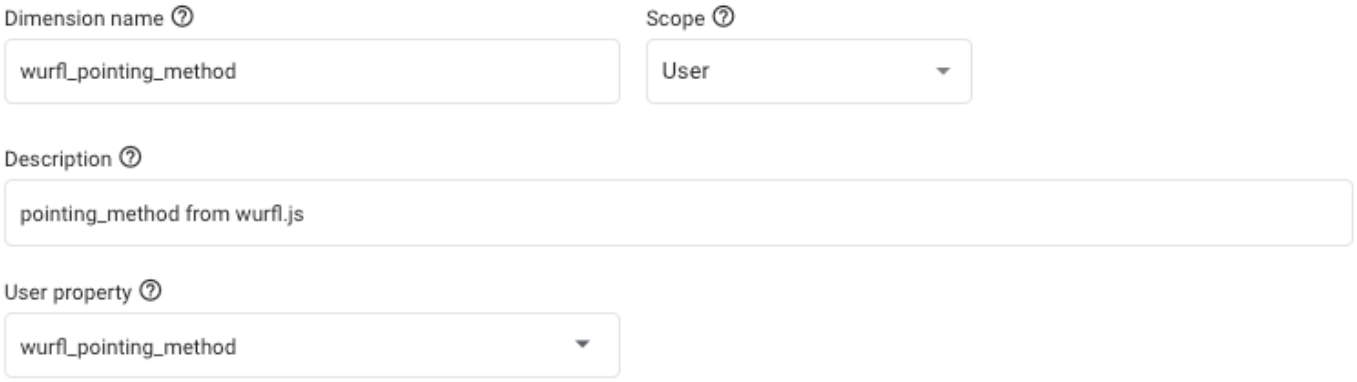
Regardless of the method in which you choose to feed WURFL.js capability data into Google Analytics, you will first need to define the set of custom dimensions that GA4 should make available in your reports.

Defining Custom Dimensions in Google Analytics 4 (GA4)

GA4 offers the option to create custom dimensions. These dimensions have a small set of characteristics that need to be populated initially. The most importantant of these fields is the `scope`. Since the device data returned from WURFL.js is mostly tied to the user, the `User scope` is recommended to be used. However, the `Event scope` can also be used if appropriate.

Custom dimensions, for all WURFL.js capabilities that you would like to track, must be individually created in GA4 like so:

1. In GA4 go to `Administration` -> `Custom Definitions` and add a custom dimension. A suggested naming convention for the dimension name is `wurfl_<capability_name>`.



2. Save and repeat this process to add a custom dimension for every WURFL.js capability that you wish to track in your Google Analytics reports.

Using Google Tag (gtag)

When you're using gtag directly on your website (i.e. not via GTM), a function called gtag() is exposed. This gtag() function can be used to populate the newly created custom dimentions in GA4 with the capability values from WURFL.js.

Hint: If your websites includes a script with this source `https://www.googletagmanager.com/gtag/js?id=G-XXXXXX`, you're using the gtag directly on your website.

1. Insert this code into your page's body element:

```
<script>
document.addEventListener("WurflJSDetectionComplete", function () {
        gtag('set', 'user_properties', {
                'wurfl_form_factor': WURFL.form_factor,
                'wurfl_pointing_method': WURFL.pointing_method,
                'wurfl_marketing_name': WURFL.marketing_name,
                'wurfl_model_name': WURFL.model_name
        });

});
</script>
```

Customize the `user_properties` object in the snippet above with the WURFL.js capabilities you need. Note that the key name must match the custom dimension names created in GA4 in the previous steps.

2. Include the WURFL.js script tag:

```
<script src="https://wjs.wurflcloud.com/wurfl.js" async></script>
```

Note: If you are using WURFL.js Lite, use this script tag instead:

```
<script src="https://wurfl.io/wurfl.js" async></script>
```

Please ensure that the script tag to include WURFL.js is placed after the event listener code from Step 1.

3. Once WURFL.js has completed loading, the `WurflJSDetectionComplete` event will fire and desired capabilities from the WURFL object can be pushed to Google Analytics. The integration is now functional.

Using Google Tag Manager (GTM)

If you're using Google Tag Manager (GTM) to include scripts, including the GA4 tag, you can also use GTM to manage WURFL.js' integration with GA4.

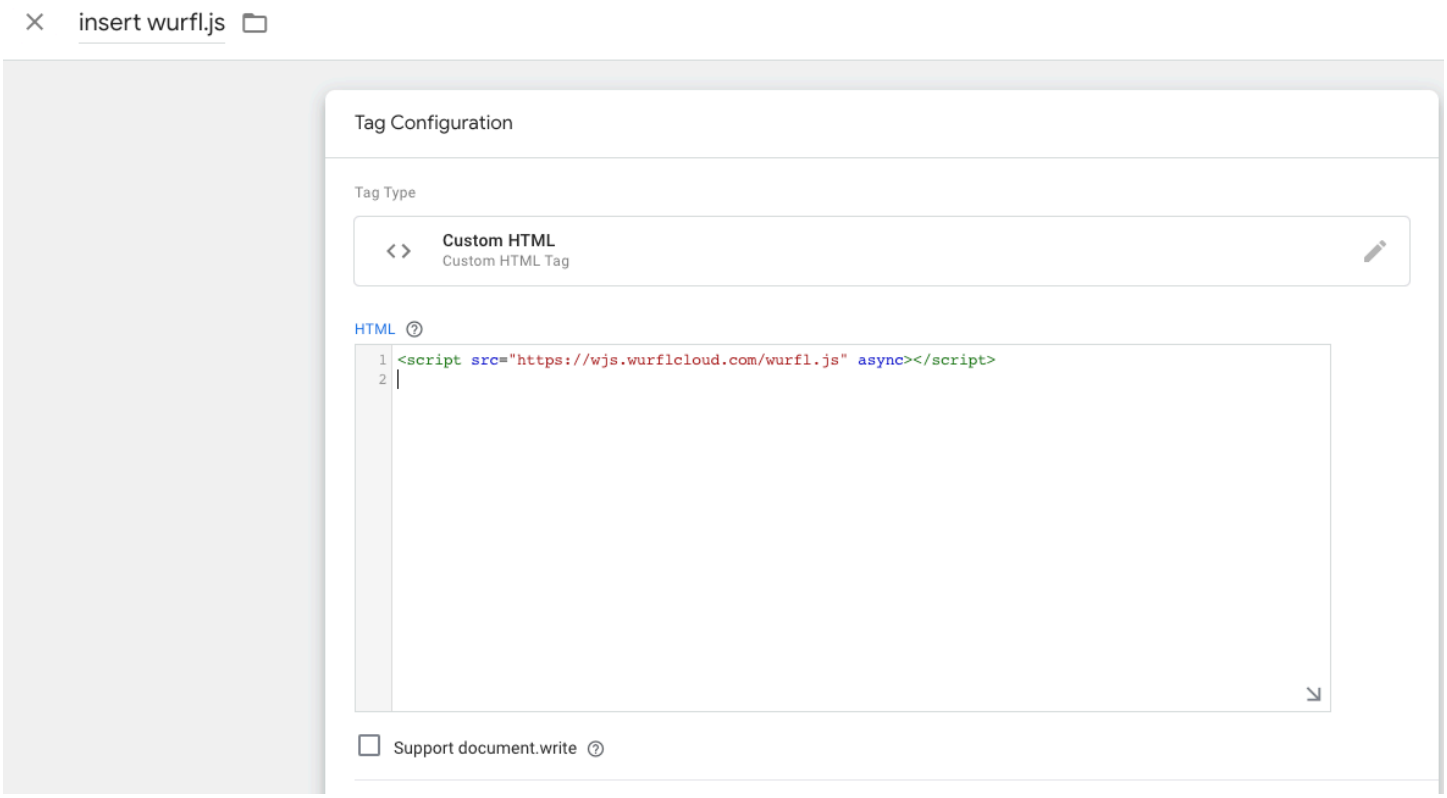
1. Assuming that you've added GA4 through GTM, the first step is to add a tag for WURFL.js:

a. Click `Tags` -> `new` and choose `Custom HTML`. Insert the following script tag:

```
<script src="https://wjs.wurflcloud.com/wurfl.js" async></script>
```

Note: If you are using WURFL.js Lite, use this script tag instead:

```
<script src="https://wurfl.io/wurfl.js" async></script>
```



b. Under `Triggering`, choose, `Initialization - All Pages`

c. Give the tag a descriptive name like "insert wurfl.js" and save

2. The next step is to create variables for each of the WURFL.js capabilities, whose values need to be sent to GA4. Note that the number and name of the variables created must match those defined in GA4 custom dimensions:

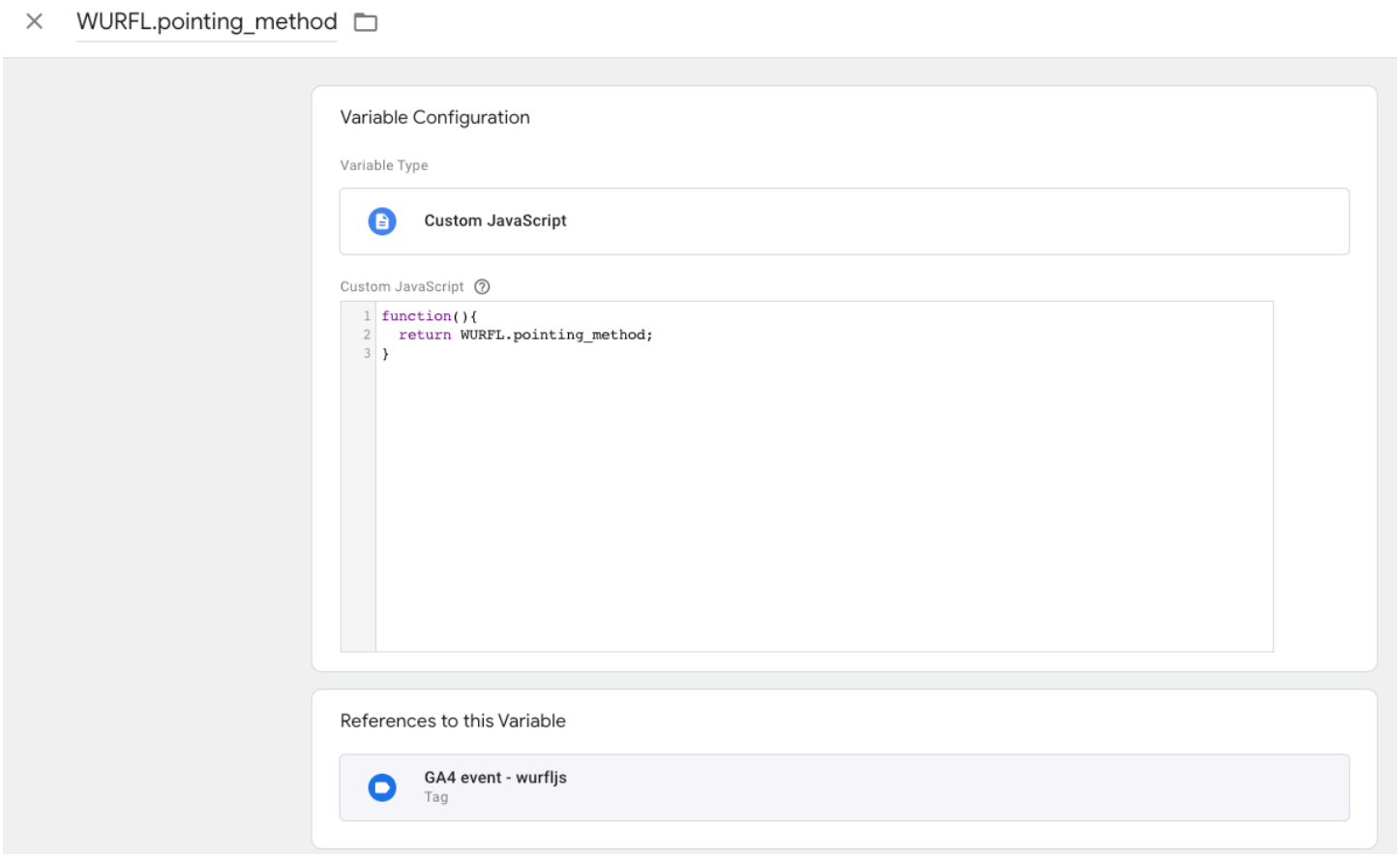
a. Go to `Variables`, in the `User-Defined Variables` section and click on `New`.

b. Choose `Custom Javascript` as the type.

c. In the text field some code is expected. Here is an example for the `pointing_method` capability:

```
function(){
  return WURFL.pointing_method;
}
```

Hint: Give the variable a name that is easily recognizable. For example, use the name of the WURFL.js capability. Here's a sample naming template: `WURFL.<capability_name>`

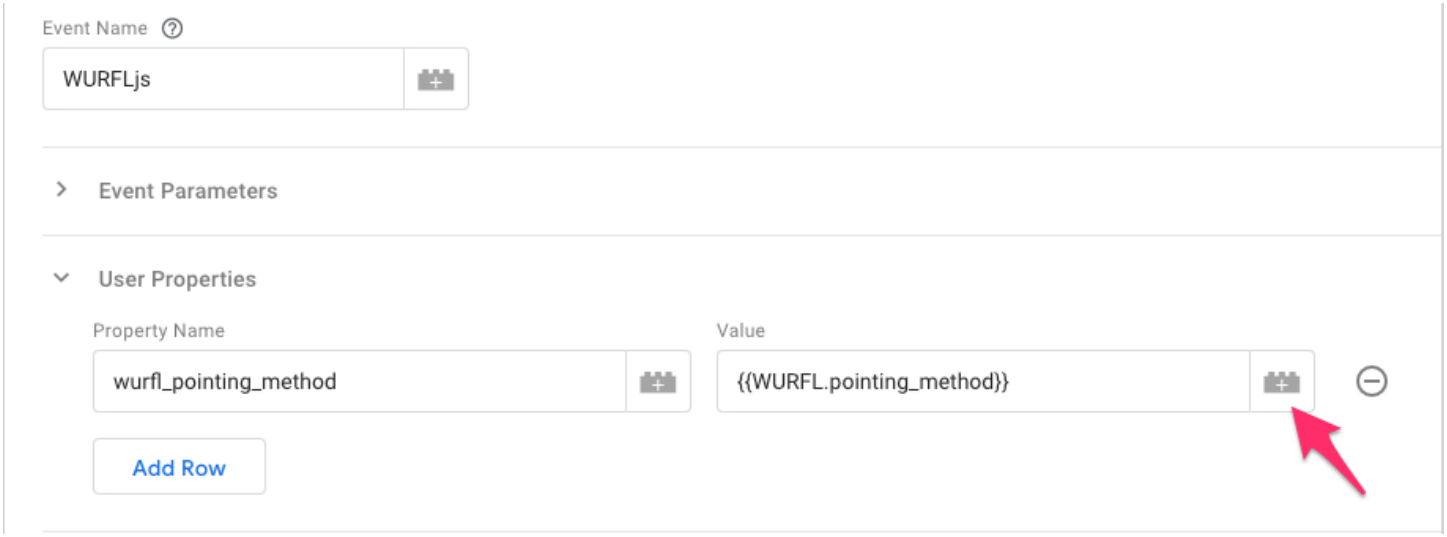


d. Save and repeat this process until you've defined all variables to match the custom dimensions from the `Defining Custom Dimensions in Google Analytics 4 (GA4)` section.

3. Now, we need to create an event in GTM to send the WURFL.js data to GA4:

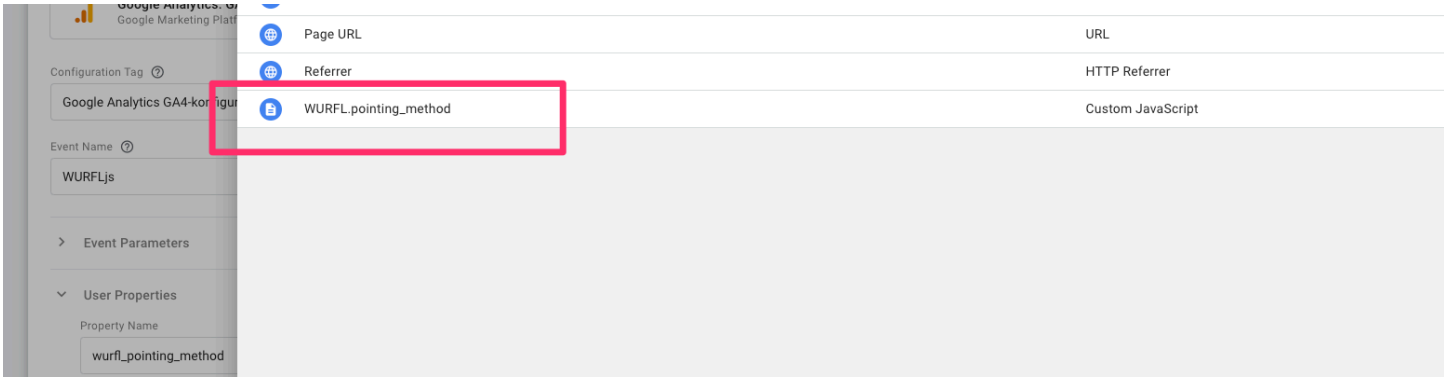
a. Click on `Tags` -> `new` and choose `Google Analytics: GA4 Event`

b. Next, expand the `User Properties` section and add rows for each WURFL.js capability to track. Note that the `Property Name` must match the names as defined in the custom dimensions, as done in the `Defining Custom Dimensions in Google Analytics 4 (GA4)` section.



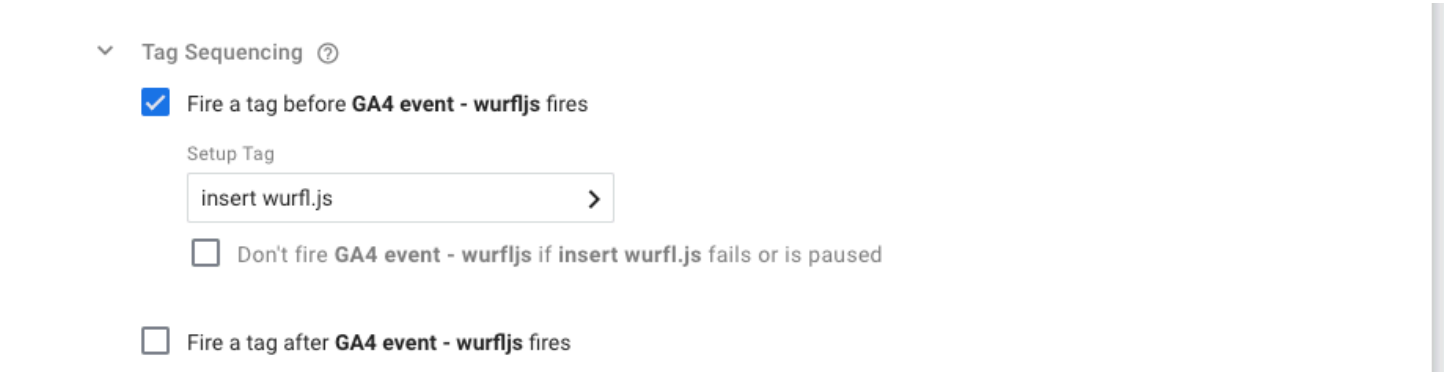
Each property is assigned its value from the variables created in the previously.

c. Click the icon and select the variable that the property is for



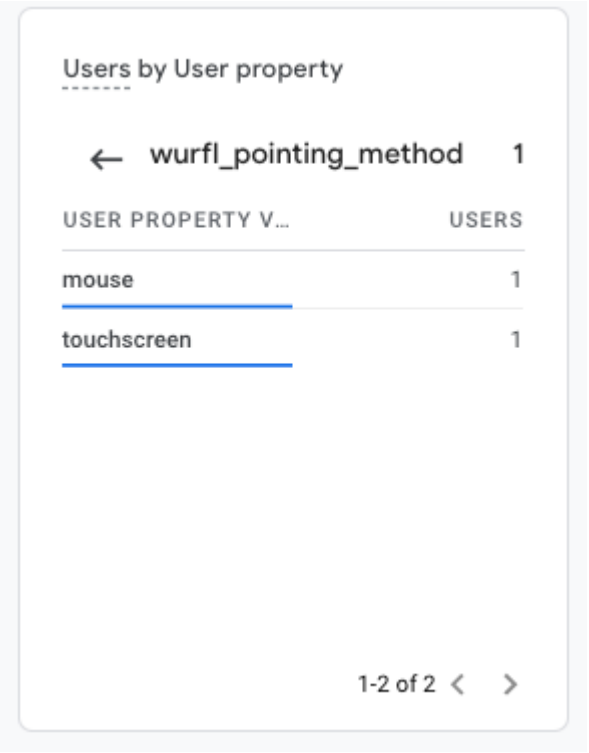
d. Repeat and add rows to the event until you have all the desired WURFL.js capabilities covered

e. Finally, make sure to sequence this event *after* the insertion of WURFL.js. Expand the `Tag Sequencing` section and select the WURFL.js insertion event to fire before this GA4 Event.

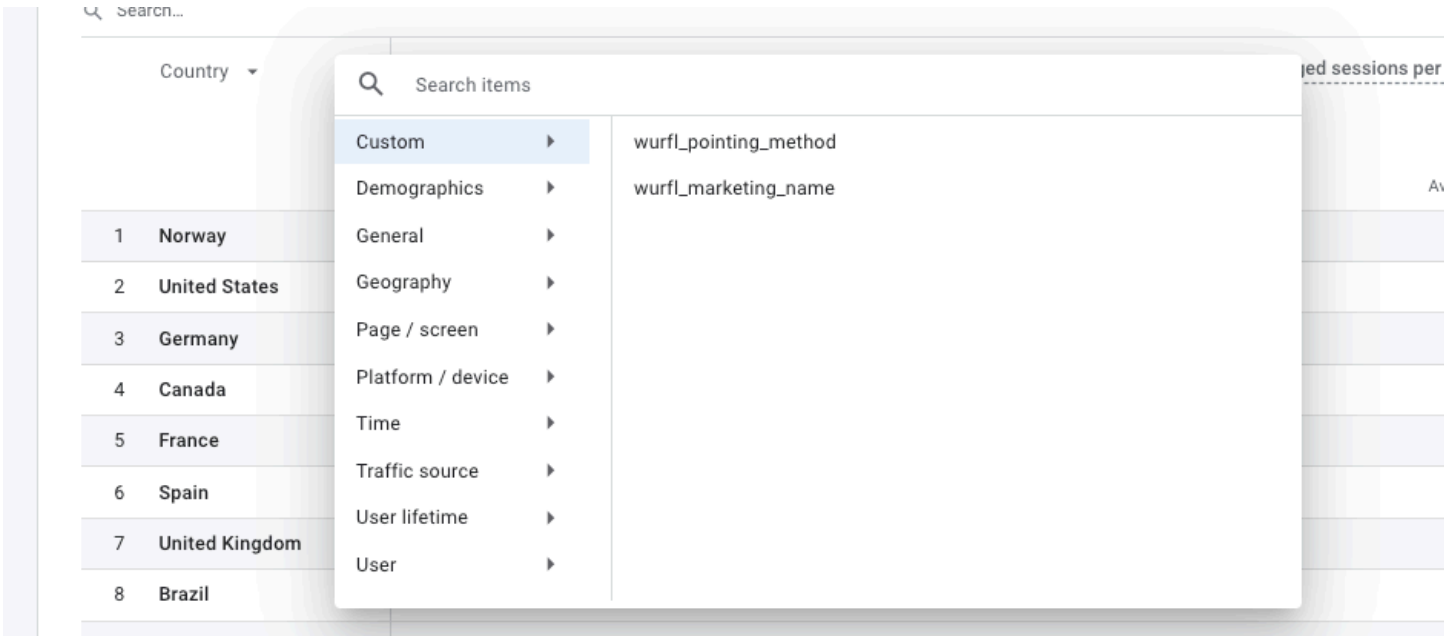


f. Save and then publish the configuration

The data is now flowing from WURFL.js via GTM to GA4. Please note that the GA4 custom dimensions will not have data usable for reporting in GA4 due to a 24 hour delay on the GA4 side. However, you can see the data flowing on the real time reports and even in the debug view, if you have that enabled.



After 24 hours the data will be available in the reports as `user properties`, for inclusion in any reports.



Terms of Service WURFL.js Lite

The WURFL.js Lite Terms of Service are available here.