



A step-by-step guide to implementing Web SDK

By Meghan Powers

Senior Manager, Data Governance, Strategy & Analytics

<https://www.linkedin.com/in/meghanpowers1/>

May 2024





Table of contents

1. Background	-----	3
2. Before implementation	-----	6
3. During implementation	-----	8
4. After implementation	-----	10

1 Background

About me

My name is Meghan Powers, and I lead the **Digital Data Strategy team**, a team of half analysts and half developers, at CarMax. CarMax is the nation's largest used car retailer and has been named *Fortune Magazine's* "100 Best Companies to Work For" 20 years in a row. I'm proud to have been a part of CarMax's success since joining the company eight years ago.

I am honored to be serving my second year as an Adobe Analytics Champion. I'm also an Adobe Analytics Certified Expert Business Practitioner and have been working in this field for 15 years. In my free time I love playing softball, doing all the water sports, snowboarding, and traveling (up to 17 countries and 30 states). But mostly I enjoy spending time with my friends and family in Richmond, Virginia, and running after my 8-year-old daughter and 5-year-old son (and husband).

After leading my team through the migration to the Adobe Web SDK, I thought I'd take some time and share our experience working with the SDK, steps we took during each phase of the migration, and some key learnings and gotchas. While this will look different for each company, hopefully this can serve as a helpful blueprint and guide for teams just starting down this road.



Why you should implement the SDK in the first place?



Faster performance

Better page performance means better Core Web Vitals scores and higher SEO rankings.



Adobe-side processing

Processing for tags will be done on the Adobe side instead of the CarMax side. This lessens the weight of tagging on the page and reduces code.



Server-side 3P

Server-side third-party tags are on their way. With the Web SDK, we'll be able to implement them more easily and harness first-party data through Adobe Launch—and produce higher match rates for acquisition.



Cookieless world

The Web SDK provides an out-of-the-box solution for sending tags server side, which will enable us to run campaigns in a cookieless world, as well as have better matching rates in acquisition now.

Why you should implement the SDK in the first place

Here is what I mean by faster performance. We've seen huge leaps in all Core Web Vital metrics on the microsites where we've implemented the SDK, as well as better data quality (see graphic below).

We've also seen a huge increase in the amount of data we're able to send to some of our third-party acquisition campaigns via third-party server-side tags, which SDK enabled us to add.

SDK's have produced large Core Web Vital improvements

Metric	Improved by...
Avg Core Web Vitals Score - Performance	Over 25%
SEO Score	Over 5%
First Contentful Paint (s)	Over 20%
Speed Index (s)	Over 25%
Largest Contentful Paint (s)	Over 25%
Time to Interactive (s)	Over 20%
Total Blocking Time (ms)	Over 45%
Cumulative Layout Shift	Over 85%
3rd party code blocked the main thread for ____ ms*	Over 50%
Avg Adobe Tag Manager Weight (ms)	Over 50%

Background

Team members include myself as team lead, four developers, two analysts, and one QA. Carmax.com is made up of smaller microsites, each with its own codebase, each owned by a separate product team. We decided to implement the SDK one microsite at a time (since our team was also still responsible for doing all the “run the business” tagging work).



How we determined the order of microsite launches

1. What would make the biggest impact on SEO (Core Web Vitals)?
 - a. Pulled CWV baselines for all microsites*
 - b. Pulled data in workspace on which ones were biggest landing pages for external traffic
2. What would make the biggest impact on acquisition? (Because of server-side tags/programs like FB CAPI)
3. Considered size and complexity of each microsite
4. Took all those factors together and put them in rough order
5. Talked to first few teams and got buy-in
 - a. Let them know LoE for them (small) and migration
 - b. Showed them the benefits
 - c. Included actual data from other microsites we'd already converted
4. Pivoted a couple times from that order as needed

*How to pull Core Web Vitals:

1. Open dev tools
2. Install Lighthouse extension
3. Run mobile **and** desktop lighthouse reports
4. Take a snapshot each morning and each evening for both mobile and desktop for seven days (make sure you get each day of the week)
5. Use analysis workspace to determine the percentage of traffic from mobile versus. desktop for that page
6. Take the weighted average of mobile and desktop for all seven days
7. This is your baseline number for each metric

2 Before implementation

Before starting an SDK migration, it's important not just to understand the technical underpinnings of it, but also to connect with it—everyone does a better job when they know the “why” behind the “what.” Here are some steps I took with my team to accomplish that.

1. I assigned my team members different personas relevant to that microsite and had them shop the site as that persona so they would understand how our customers use those pages and identify with their pain points.
 - a. Example: “You are an expectant mother shopping for a bigger family vehicle”
2. We then had a meeting with our whole team and the product team who owns the microsite.
 - a. My team brought their questions and perspectives from the persona exercise.
 - b. The project manager talked to us about how customers use the pages and why it is important to carmax.com and to the company.
 - c. The analyst talked to us about what data they typically use to make decisions and various ways they use the data.
 - d. The developers talked to us about any gotchas with the various features and any unique aspects of their codebase.

Next came the requirements gathering phase. This is my team's typical approach to documenting requirements, along with a sample from our Excel file. We find it's really helpful to include screenshots.


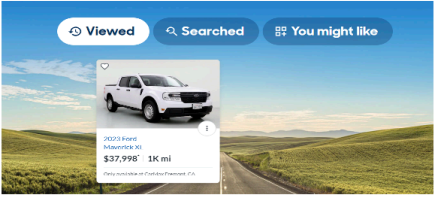
1. Partner with product team.
 - a. Get all pages, features, and experiences in scope (in codebase).
 - b. Get QA credentials, logins, account statuses, etc. to get into all experiences.
 - c. Ask their analyst how their current analytics are working, what KPIs are most important, any business questions they want to be able to answer that they can't now, etc.
2. Develop requirements (sample below).
3. Perform pre end-to-end testing** against those requirements.
4. Partner with product team to go through requirements to make sure it's comprehensive and tags will answer their business questions.
5. Walk our developers through the requirements and answer their questions.
6. Create (Leankit) cards.
 - a. One card for first-party page load
 - b. One card for third-party page load
 - c. One card for each CTA/click action
 - d. Developers create a tech spec for data layer using the same Excel requirements doc

**Why do we do end-to-end testing before implementation?

1. This was an important addition to our process that we started doing after the first one.
2. It helps determine what tagging is currently working as expected versus "broken"/unideal..
 - a. Tagging that is working can be "lifted and shifted"—less work on developers.
 - b. We evaluate each broken tag and assign one of these statuses:
 - i. Level 1: Must be fixed before SDK can go live
 - ii. Level 2: Not a blocker for the SDK to go live but should be fixed in a fast follow (and then prioritize cards within level 2)
 - iii. Level 3: Not a blocker for the SDK and not critical to fix right away—can be put in a backlog of "nice to have" tagging cleanup work (and then prioritize cards within level 3)



Requirements sample

Screenshot/Page Reference	Description	Event Type	Adobe tagging	Adobe tagging value/details
	Homepage	pageload	channel (eVar9) Subsection (prop1) Pagename (eVar1) General pageload requirements events	Homepage Homepage Homepage
	SHOP ALL CARS	click	Link Originating Page (prop3) Link Name (prop4) Link Position (prop5) Concatenate - link page link name link position (prop6) Prop 21 (Search initiation method) eVar 32 (Search initiation method) Timestamp - Time at Minute and 15 Minute Level (prop65)	Homepage Shop All Cars LYCG Homepage Shop All Cars LYCG Homepage Shop All Cars LYCG Homepage Shop All Cars LYCG hour:minute AM/PM (e.g. 8:45 PM)
	Viewed	click	Link Originating Page (prop3) Link Name (prop4) Link Position (prop5) Concatenate - link page link name link position (prop6) Timestamp - Time at Minute and 15 Minute Level (prop65)	Homepage Viewed Homepage Homepage Viewed Homepage hour:minute AM/PM (e.g. 8:45 PM)
	Searched	click	Link Originating Page (prop3) Link Name (prop4) Link Position (prop5) Concatenate - link page link name link position (prop6) Timestamp - Time at Minute and 15 Minute Level (prop65)	Homepage Searched Homepage Homepage Searched Homepage hour:minute AM/PM (e.g. 8:45 PM)
	"Hero Tile"	click	Link Originating Page (prop3) Link Name (prop4) Link Position (prop5) Concatenate - link page link name link position (prop6) Timestamp - Time at Minute and 15 Minute Level (prop65)	Homepage <link text (e. 2023 Ford Maverick XL)> Homepage Home Base Hero Tile Recommended Car Homepage <link text (e. 2023 Ford Maverick XL)> Homepage Home Base Hero Tile Recommended Car hour:minute AM/PM (e.g. 8:45 PM)

3 During implementation

Now you're ready to start building. These are the implementation steps we took. Unfortunately I am not a developer (trust me, no one wants me touching code), so this is more the high-level approach we took rather than the actual code you'll need.

1. Create separate Adobe Launch property—just for SDK.
2. Create a new TESTING REPORT SUITE and assign it to that new SDK Launch property.
3. Since it might take weeks or months, lift and shift rules from old Launch property to SDK property and then swap over all at once.
4. We divvied microsites up into pages, and then into sections within each page, and then individual CTAs within each section. We created one 'card'/task per CTA.
5. Refine work/card as close to development as possible so details are fresh.
6. QA each card (individual CTA).
7. Use new testing report suite to compare QA data in SDK to QA data in AppMeasurement.
 - a. Counts of events, props, eVars
 - b. Check main KPIs
3. Once QA passes,, sits in "Ready" status until we are ready to launch whole SDK at one time.
4. Once cards are all done, partner with product team to pick a launch date that works for both teams (no big tests about to go live, etc.).

Along the way, I found it really important to keep stakeholders in the loop. In order to do that:

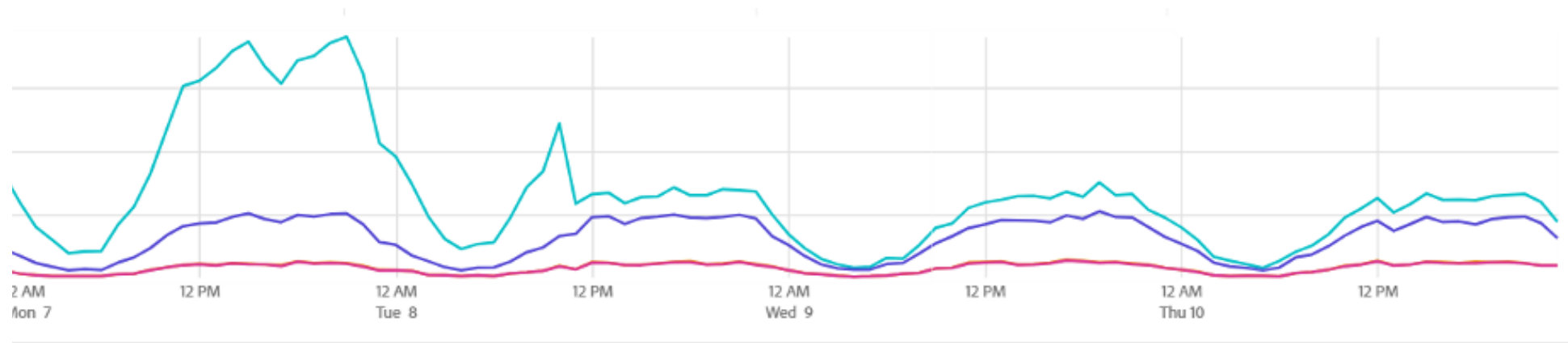
1. Establish a stakeholder group.
 - a. Make sure to include acquisition teams that are driving traffic to that page.
 - b. There will probably be some common stakeholders to keep informed across any microsites, but then some others may change depending on the microsite.
3. Email status updates throughout project.
4. One to two weeks ahead of launch date, email stakeholders to let them know about the date and make sure there are no conflicts.
 - a. Make sure the acquisition team is prepared to monitor third-party data.

Finally, it's time to launch to production. We found success by following these general steps:

1. Coordinate time and day with product team—they do their part then we do ours, same hour.
2. Establish KPIs ahead of time with our team and product team (PT) analyst.
 - a. Ahead of time, create a workspace with hour-by-hour measurement of KPIs.

3. Perform live QA with our devs and PT analyst of hourly workspace approximately three hours later.
 - a. Determine if anything dropped off in the hour of launch (see graph below).
 - b. If not, we're good to go—and keep monitoring.
 - c. If so, try and troubleshoot on the spot with devs.
 - d. If a quick fix, put in; If a quick fix, determine on the spot with PT analyst whether we can live with that bug or if need to revert while fixing..
 - e. If reverting,, just swap back to old Adobe Launch property.

Hour-by-hour KPI measurement in Adobe Analysis Workspace



4 After implementation

Hopefully all has gone well, and you have the Adobe Web SDK up and running in prod with no major issues. Take a minute and celebrate as a team because I know it wasn't easy to get to this point.

Post-launch to production

1. If everything was successful, send a follow-up email to the stakeholder group to let them know that the launch was successful.
2. Continue to monitor KPIs for another week and do spot checks of tags in production and other production data in Analysis Workspace.
3. Repeat steps to pull core web vitals numbers for seven days to get post-production data.
4. After a week, follow up with stakeholder group to let them know that all KPIs are still stable and to report out results of Core Web Vitals percentage change of pre-post results.
5. Celebrate as a team!

Learnings

No project would be complete if we didn't learn from mistakes and unforeseen circumstances. Here are some of the "gotchas" we discovered along our journey:

1. We noticed a bit of data clunkiness with half the site in SDK and the other half not.
 - a. When we turned on a new microsite, we saw an increase in internal URL marketing channel visits for those pages.
2. Pay attention to Adobe Target.
 - a. Make sure the Adobe Target extension is enabled and all check boxes are checked in applicable rules.
 - b. Partner closely with the product team to make sure you know what tests are running and have them test any active tests in the SDK environment.
 - c. If you can avoid launching to production during an active test, even better.
3. Don't forget about third-party tags, and make sure to migrate them too.
 - a. Also make sure to QA them. We took the extra step of having each partner test their own tags and getting their sign-off that they looked good.
4. Do pre- and post-end-to-end testing.
 - a. **Don't just rely on QA for each individual card/task/CTA.**
5. We initially built SDKs using processing rules but found that to be unstable, so we moved away from that approach and built everything in Adobe Launch.
6. We've seen some instability building the SDK on our old data layer, so we are now following Adobe's best practices and moving to an event-driven data layer—which is already working better on our newest SDK implementation.

I hope this guide was helpful, and I'd be interested in hearing best practices, tips and tricks, or other learnings or "gotchas" from anyone else who's implemented the SDK. You can find me on the [Adobe Experience League](#) or on [LinkedIn](#).



Adobe, the Adobe logo, and Experience League are either registered trademarks or trademarks of Adobe in the United States and/or other countries.

© 2024 Adobe. All rights reserved.