Module 3 – Post Implementation

This toolkit is designed for Master Architect Exam Aspirants. There are **three** Modules. Study Each module per week to stick to schedule. Technical Parts of applications are depicted in Videos, you can learn more about them from experience League. You can visit Get prep page to understand the contents and anticipate the learning journey.

This is Master Exam, Architect toolkit Module 3. This module contains 11 sections.

3.1 Implement Adobe Analytics

Adobe requires code on your site or app to send data to Adobe's data collection servers. The following steps indicate how a typical implementation works.

1. When a visitor comes to your site, a request is made to your web server.
2. Your site's web server sends the page code information, and the page displays in the browser.
3. The page loads, and the Analytics JavaScript code runs.
   The JavaScript code sends an image request to Adobe data collection servers. Page data that you defined in your implementation are sent as part of a query string in this image request.
4. Adobe returns a transparent pixel image.
5. Adobe servers store collected data in one or more *report suites*.
6. Report suite data populates the reports that you can access in a web browser.

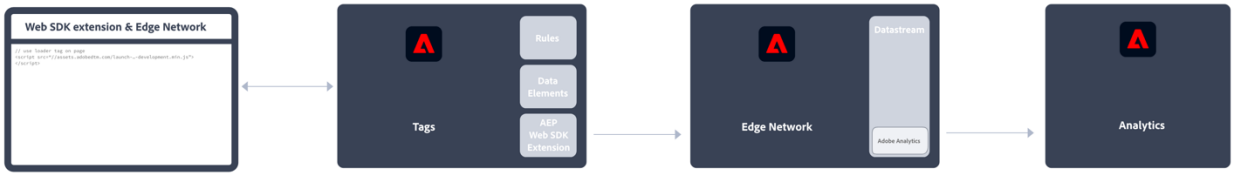
The JavaScript code execution occurs quickly and does not noticeably affect page load times. This approach allows you to count pages that were displayed when a visitor clicked **Reload** or **Back** to reach a page, because the JavaScript runs even when the page is retrieved from cache.

Adobe Analytics requires code within your website, mobile app, or other application to send data to data collection servers. There are several methods to implement this code, depending on platform and your organization's needs.

Website implementation methods

For your **website**, the following implementation methods are available:

- **Web SDK extension**: The standardized and recommended method to implement Adobe Analytics for new customers. Install the **Adobe Experience Platform Web SDK extension** in Adobe Experience Platform Data Collection **Tags**, use a loader tag on each page, and send data to Adobe Experience Platform **Edge Network** in a format convenient to your organization. The Edge Network forwards incoming data to Adobe Analytics in the correct format.



See How to implement Adobe Analytics using the Adobe Experience Platform Web SDK extension. for more information.

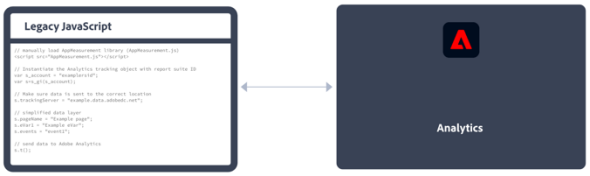
- **Web SDK**: You can manually load the Web SDK libraries on your site if you do not want to use Adobe Experience Platform Data Collection. Reference the Web SDK library (alloy.js) on each page, and send the desired tracking calls to the Adobe Experience Platform **Edge Network** in a format convenient to your organization. The Edge Network forwards incoming data to Adobe Analytics in the correct format.

See How to implement Adobe Analytics using the Adobe Experience Platform Web SDK for more information.

- **Analytics extension**: Install the **Adobe Analytics extension** in Adobe Experience Platform Data Collection **Tags**. Place a loader tag on each page, and use the Adobe Analytics extension to determine how each variable is defined. Use this implementation method if you do want the convenience of Tags, but not want to use the Edge Network infrastructure.

See How to implement Adobe Analytics using the Analytics extension for more information.

- **Legacy JavaScript**: The historical manual method to implement Adobe Analytics. Reference the AppMeasurement library (AppMeasurement.js) on each page and then outline variables and settings used in an implementation.



This implementation method can be useful for implementations using custom code and is still recommended when you (want to) use:

- activity map data,

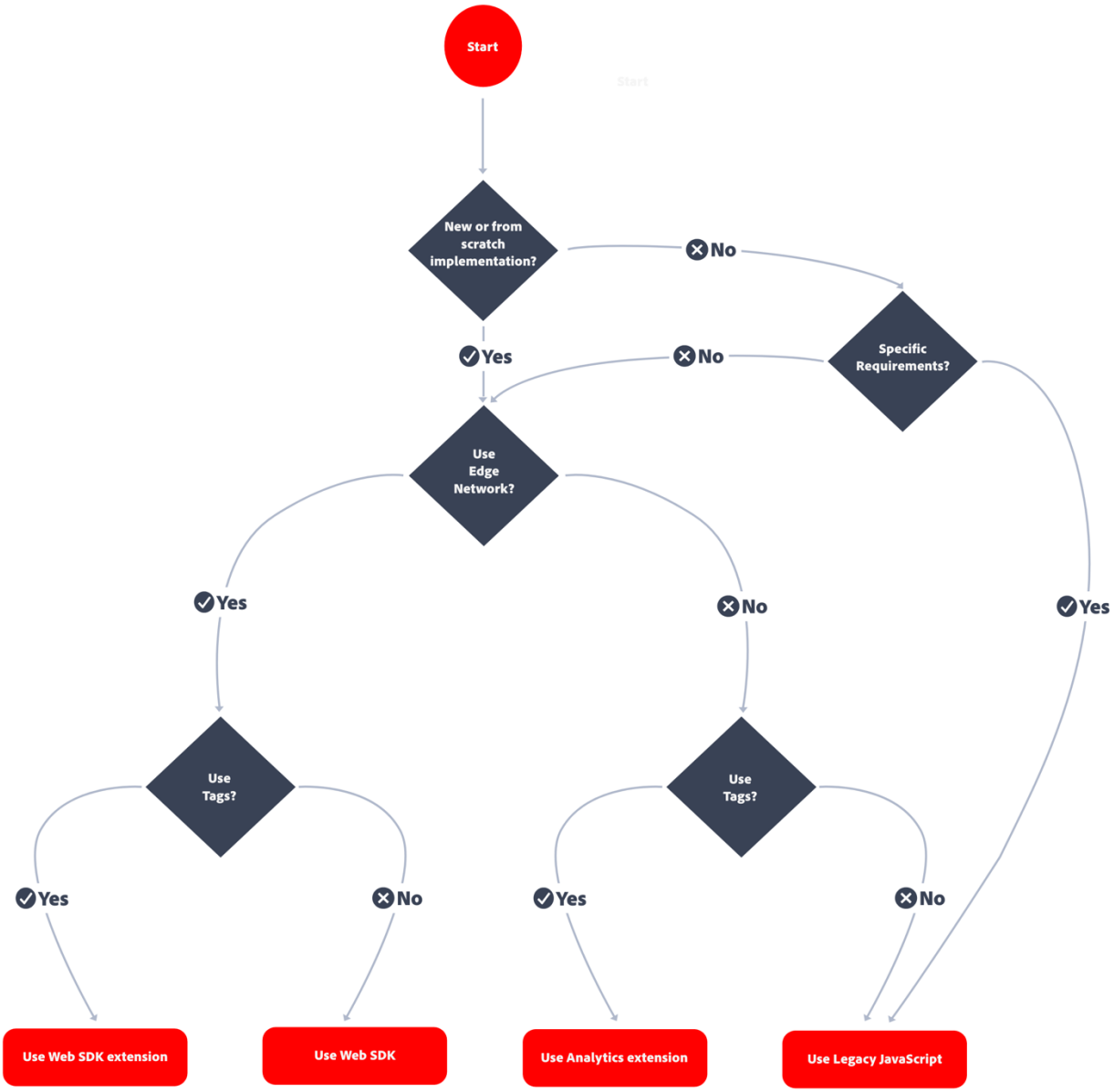
  INFO

  Using the latest Web SDK, Activity Map is supported. See Enable Activity Map for more information.

- streaming media measurement,
- livestream API or livestream triggers,
- AMP page tracking

See Implement Adobe Analytics with AppMeasurement for JavaScript for more information.

The following decision flow might help you select an implementation method:



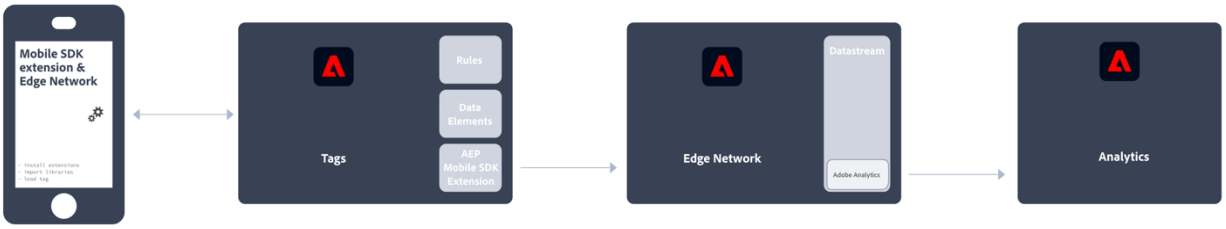
TIP

Please contact Adobe for advice and best practices on which implementation to choose based on your current situation.

Mobile app implementation methods

For your **mobile app**, the following implementation methods are available:

- **Mobile SDK extension**: The standardized and recommended method to implement Adobe Analytics in your mobile app. Use dedicated libraries to easily send data to Adobe from within your mobile app. Install the **Adobe Experience Platform Mobile SDK extension** in Adobe Experience Platform Data Collection **Tags** and implement the correct code in your app to import libraries, register extensions and load the tag configuration. This sends data to Adobe Experience Platform **Edge Network** in a format convenient to your organization. Experience Edge forwards incoming data to Adobe Analytics in the correct format.



  See Implement Adobe Analytics using the Adobe Experience Platform Mobile SDK for more information.

- **Analytics extension**: Install the **Adobe Analytics extension** in Adobe Experience Platform Data Collection **Tags**, and implement the correct code in your application to import libraries, register extensions and load the tag configuration. Use the Analytics extension to determine how each variable is defined. Use this implementation method if you do want the convenience of Adobe Experience Platform Data Collection, but not want to use Adobe's Experience Platform Edge network infrastructure.

  See Implement Adobe Analytics using the Analytics extension for more information.
- .

- # Variables, functions, methods, and plug-ins overview

Analytics provides a number of variables to collect Analytics data. Variables in this section are split into several sections:

- **Page variables** are values that are typically used directly in reporting. Common page variables include props, eVars, and events.
- **Config variables** are settings values that help make sure the correct data reaches Adobe. Common config variables include trackingServerSecure, charSet, and linkTrackVars. Config variables typically do not populate dimension items.
- **Functions and methods** are pieces of code that perform a specific task when referenced. Common functions include t(), tl(), and clearVars().

- **Configuration Variables**

Configuration variables control the way data is captured and processed in reporting. They do not typically contain dimension or metric values.

Setting configuration variables

In implementations using the Web SDK extension or Analytics extension, configuration variables are typically found in the extension's settings:

1. Log in to Adobe Experience Platform Data Collection using your AdobeID credentials.
2. Click the desired tag property.
3. Click the Extensions tab, then Click Configure under the extension.

In JavaScript implementations using AppMeasurement.js, configuration variables are typically set at the top of the JS file.

- **Prepare to implement Adobe Analytics**
- **Implement Adobe Analytics with Adobe Experience Platform Edge**

Adobe Experience Platform Edge allows you to send data destined to multiple products to a centralized location. Experience Edge forwards the appropriate information to the desired products. This concept allows you to consolidate implementation efforts, especially spanning multiple data solutions.

- **Implement Adobe Analytics using the Analytics extension**

Through the lifetime of Adobe Analytics, Adobe has offered several different methods to implement code on your site for data collection. Adobe's current recommended method is through tags in Adobe Experience Platform.

- **Implement Adobe Analytics with AppMeasurement for JavaScript**

  AppMeasurement for JavaScript has historically been a common method to implement Adobe Analytics. However, with increasing popularity of Tag Management Systems, using tags in Adobe Experience Platform is recommended.

- **Implement Analytics on other platforms**
- **Implement Analytics for Mobile Devices**

To implement Analytics for mobile devices, use the Adobe Experience Platform Mobile SDK.

- **Implementation use cases**
- **Validate your implementation**
- **FAQs about Analytics implementation**

Frequently asked questions about implementation, and links to more information.

- **Review your implementation**

3.2 Analytics Components Guide

Analytics components help you fine tune and empower your analysis of data. Components include:

- **Dimensions:** Reference for dimensions usable in Adobe Analytics.
- **Metrics:** Reference for metrics usable in Adobe Analytics.
- **Segmentation:** Focus on a subset of your data.
- **Calculated metrics:** Use simple formulas to combine metrics, or advanced functions for statistical analysis.
- **Virtual report suites:** Create a virtual silo of data based on a report suite. Allows the ability to cleanse or segment data for a better user experience. Some features can only be used in virtual report suites.
- **Cross-Device Analytics**: A special type of virtual report suite allows you do configure Cross-Device Analytics.
- **Alerts:** Receive notifications any time data goes above or below a threshold.
- **Classifications:** Reorganize and group dimensions to obtain additional insight.
- **Real-time reporting:** Get reports and trends the minute they are available.

- **Marketing channels:** Understand how visitors arrive to your site and determine which channels are most successful.

Dimensions

Dimensions are variables in Adobe Analytics that typically contain string values. Common dimensions include Page, Referring domain, or an eVar. In contrast, metrics contain numeric values that tie to a dimension. A basic report shows rows of string values (dimension), against a column of numeric values (metric).

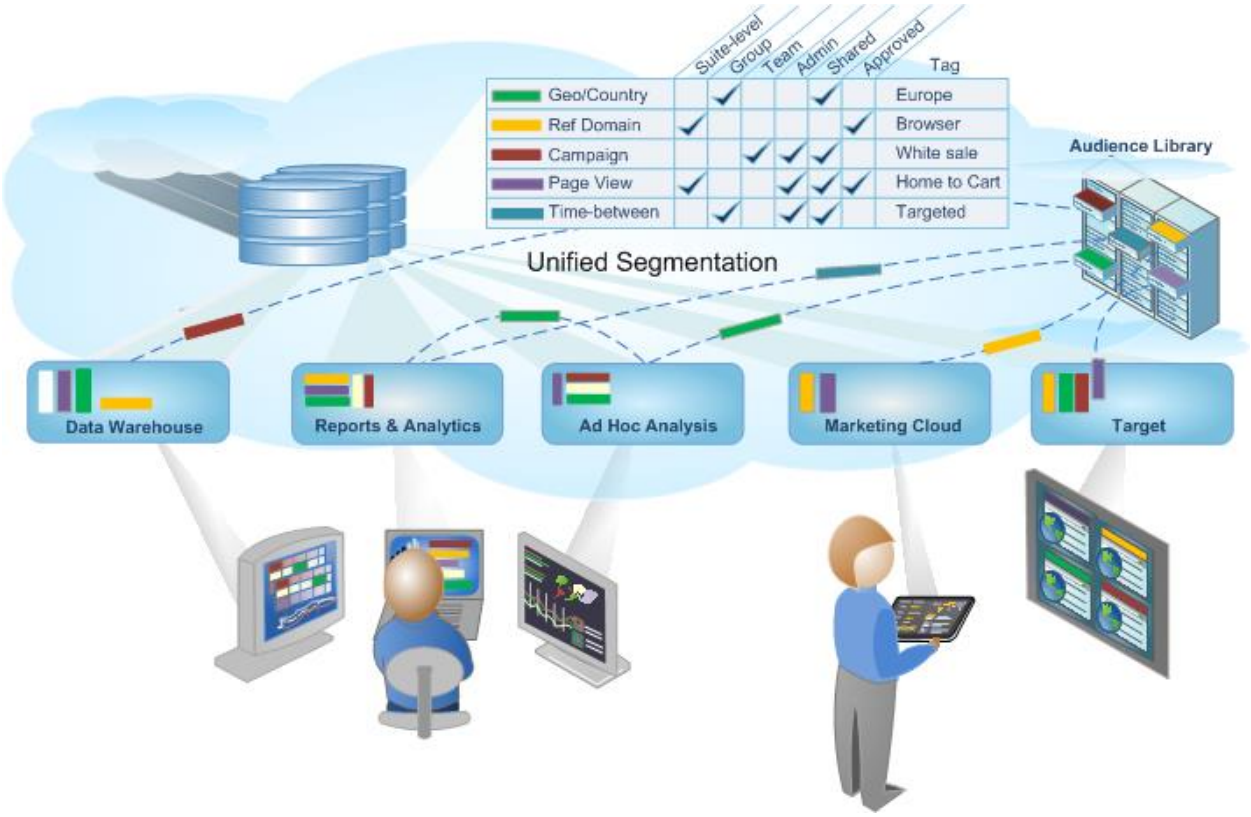
Metrics

Metrics allow you to quantify dimension items, such as to see which pages on your site have the most page views. You can also trend metrics over time, such as to see how many orders visitors make on your site each day. A basic report shows rows of string values (dimension), against a column of numeric values (metric).

Segmentation

Adobe Analytics lets you build, manage, share, and apply powerful, focused audience segments to your reports using Analytics capabilities, the Adobe Experience Cloud, Adobe Target, and other integrated Adobe products.

Analytics segmentation includes the Segment Builder to construct segments and run a pre-test, and the Segment Manager to collect, tag, approve, set security, and share segments across your organization.



Data Scientists and Marketing Analysts can employ, extend, and refine segments for analysis specific to his or her needs, and then save the segment for other users to extend, refine and save as a new segment to the library. Once set in motion, it's a cycle of designing and managing codified audience insights as a unified segment workflow.

- **Calculated Metrics**

Calculated and Advanced Calculated (or Derived) metrics are custom metrics that you can create from existing metrics.

- **Virtual Report Suites**

Virtual report suites segment your Adobe Analytics data so you can control access to each segment.

- **Cross-Device Analytics**
- **Alerts**
- **Classifications**
- **Locations**

The Locations manager allows you to create, edit, or delete locations.

- **Calendar Events**
- **Scheduled reports queue**
- **Real Time Reporting**
- **Marketing Channels**

3.3 Analytics Admin Guide

Adobe Analytics currently has two areas for administrators:

- Adobe Admin Console: Use this area for provisioning Experience Cloud tools, and managing user permissions. It is located at adminconsole.adobe.com.
- Analytics Admin Tools: Use this area for report suite and variable management. It can be accessed by clicking Admin in the top header of Adobe Analytics.

This guide covers:

- All tasks that are done in Analytics Admin Tools. This area includes setting up report suites, variables, classifications, or data governance. See Admin Tools for a list of report suite and company settings available.
- All Analytics-specific tasks that are done in the Adobe Admin Console. This area includes product provisioning and user permission management. See Adobe Analytics in the Adobe Admin Console for a list of actions that can be done in the Adobe Admin Console.

This guide does not cover many of the generic capabilities that the Adobe Admin Console offers. Instead, see Admin Console in the Enterprise user guide.

Key articles

- Analytics First Admin Starter Guide: If your organization is brand new to Adobe Analytics, follow this guide to start getting value out of Adobe Analytics.
- Report Suite Manager: The most commonly accessed admin tool in Analytics. The Report Suite Manager contains settings for variable management classification management, time zones, and more.

- **Analytics Release Notes**
- **Adobe admin console**
- **Analytics admin tools**
- **Admin API**

3.4 The Event Driven Layer

The Event-Driven Data Layer

By Jim Gordon | 04/23/2019

Recently, I wrote an appeal to Adobe suggesting they implement first-party support for an event-driven data layer in Adobe Launch. I *specifically* mentioned an "event-driven, asynchronous data layer". Let's just call it the Event-Driven Data Layer (EDDL) to keep it simple. Since the article's release, I've had a number of good conversations on this topic. The consensus *seems* to be that this is the right direction. For the purposes of keeping the article a manageable length, I didn't go into too much detail of what an EDDL might look like. The goal of this article is to define what it is and why it's a good thing.

Before we begin, let's be clear about one thing. There's an EDDL and a CEDDL.

CEDDL = Customer Experience Digital Data Layer. Legacy W3C.
EDDL = Event-Driven Data Layer. What this article is about.

I tried to find a less confusing abbreviation. Unfortunately, this one made the most sense. They're both types of data layers. A good way to remember the difference is the CEDDL is 25% more cumbersome to spell and to implement. That's being generous. The Event-Driven Data Layer describes what you're implementing: a data layer that is constructed and transmitted to your TMS by events.

Let's first walk through a concept that seems obvious but very few people care to think about. That is this: **in a TMS, every tag is triggered by a some event**. That means your pageview is triggered on an event. That event might be TMS Library Loaded (Page Top), DOM Ready, Window Loaded, or any other indication that the page has loaded. These are all events that happen on a page or screen. They are as much of an event as a click, mouseover, form submission, or anything else.
A pageview is a *tag (*think Adobe Analytics beacon*)*. Viewing a page is an *event*. A custom link is a tag. Clicking a button is an event. Logging in is an event. Submitting a form is an event. That event might also be associated with some tags. Make sense? This is important because **we need to abstract the tool from the data**. Yes, you might choose to load Adobe Analytics pageview code when the window loads… but you COULD also choose to load a pageview on a click. The EDDL thinks independent of tool-specific definitions. Let's dive into why this is the preferred method.

Table of Contents

- Related posts:

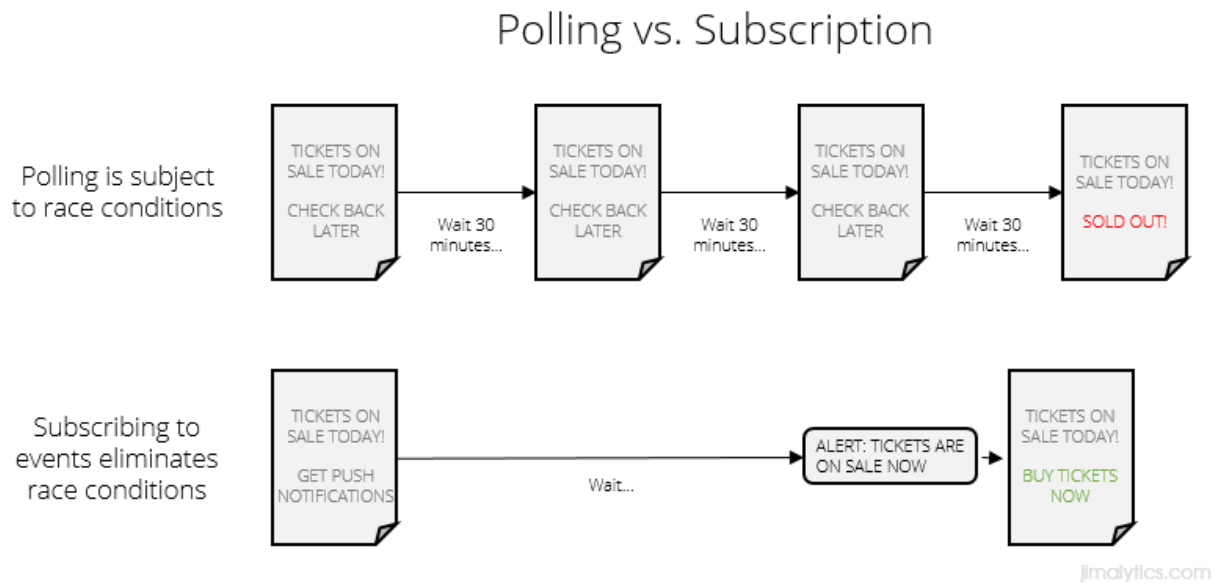
It's harder to screw up

That doesn't mean you *can't* screw it up. It's just harder to. You still need some code that sits in your header. The difference with the EDDL is:

1. It's less code
2. You drop it in once and never have to touch it again

Maybe that means you're simply declaring a variable (*var foo = [];*). Maybe it means you're dropping in a little more code (copy/paste exercise). There's no way around it – the variable or the function has to exist before you do anything with it. After that it's all gravy. Timing is a non-issue with an EDDL. That's because it proactively sends a message when a thing happens. Other data layer methodologies *poll* objects (like the Data Element Changed event). What does that mean?

Imagine you're waiting on popular concert tickets to go on sale. You know what *day* they go on sale, but not what *time*. You know they'll sell out fast, so you refresh Ticketmaster every half hour to see if their status changed. If you're checking (or *polling*) to see if they're for sale at 12:00 and they go on sale at 12:01, you might be out of luck. This is what's called a *race condition*. After the status of the tickets change, you're racing to see if you can get one before they're gone.

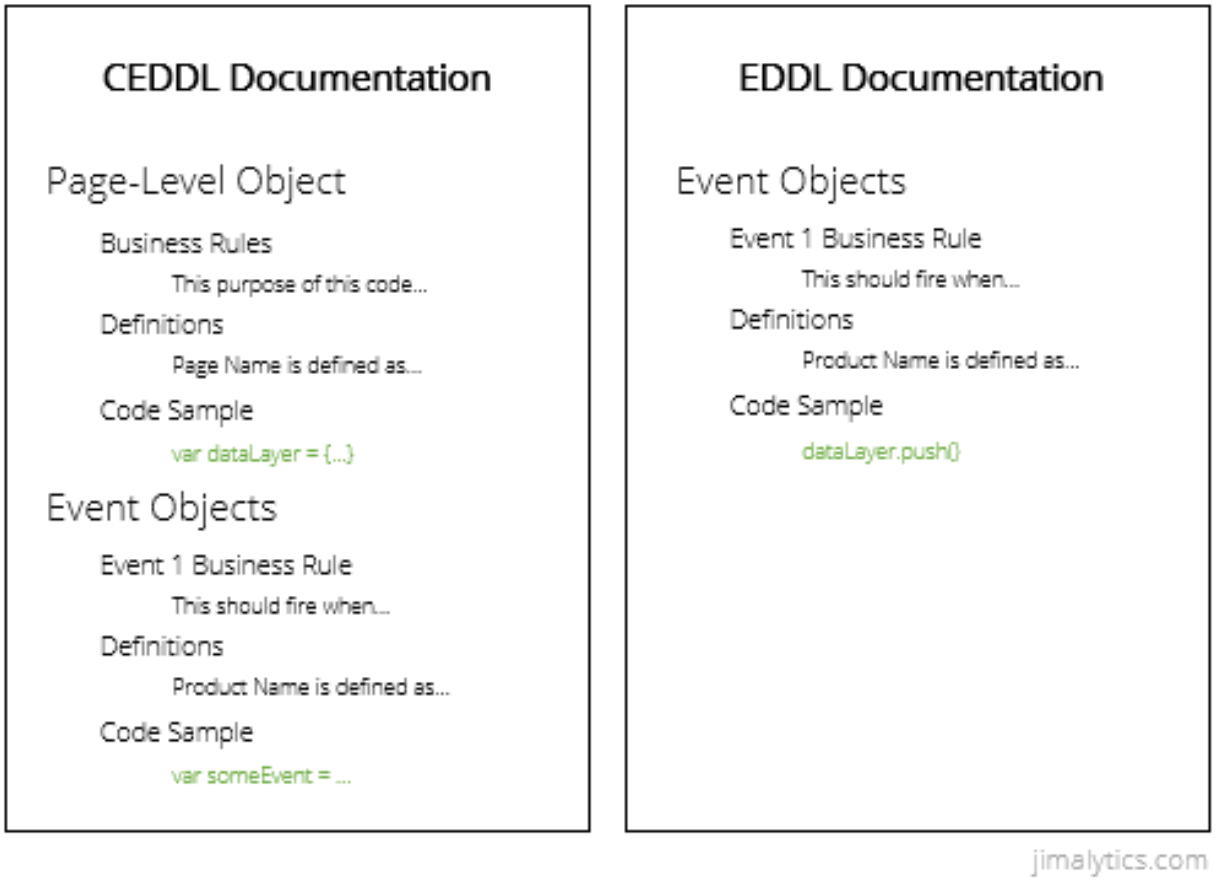


The same thing happens when you monitor data layers. If you go from Page A to Page B and you're monitoring the object, there's a very real chance you'll be on the next page before your TMS realizes anything happened. That seriously sucks. You know what would fix this and save a lot of time? *Subscribing* to push notifications. Proactively tell me when the tickets are on sale. Don't worry about loading the entire page object up-front. If we need to wait on servers, *let me know* when user information has propagated and then we'll trigger a pageview. The EDDL uses these push notifications so you don't have to worry about missing out on data.

It's easier to communicate

Prioritization of data layers is difficult when it *feels* like a lot of work and its value isn't immediately clear. Multiple code patterns paired with multiple sets of instructions *feel* like

more work than one. We're already taking focus away from the value of the data layer at this point.



The CEDDL requires teams to learn multiple concepts/patterns. There's a page object… and then there are events. While it might be subconscious, multiple concepts/patterns (even simple ones) requires switching mental gears. They're also both explained as though they're different things. Here's an example from Adobe where both the W3C and another methodology are recommended. If I'm not very technical or I'm new to analytics, this would make my head spin.

The EDDL is much simpler. You explain it once, use the same code pattern, and can be easily dropped into a template. Here's how the documentation might look:

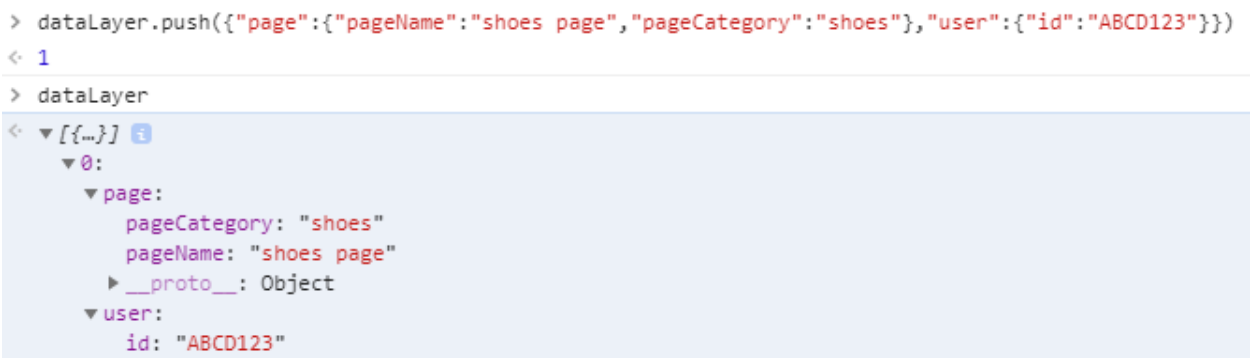
Data Layer Documentation
1.0: Page and User Information
Trigger: As soon as the information is available on each page loads or screen transition.
Code: *dataLayer.push({"event":"Page Loaded","page":{…}…});*

1.1: Email Submit
Trigger: When a user successfully submits an email address.
Code: *dataLayer.push({"event":"Email Submit","attributes":{…}…});*

The documentation is consistent. We aren't suggesting page stuff is different from event stuff. Remember that pageviews are triggered via events, too. You won't send a pageview until you have the data you need, either. You're not trying to figure out whether you can cram it into the header, before Page Bottom, ahead of DOM Ready, or before Window Loaded. We can just say: "*When the data is there, send the pageview.*" A lot of companies opted out of the W3C CEDDL for this reason alone.

It's just as comprehensive

You can literally create the W3C schema with it. An effective EDDL has some kind of computed state I can access that functions like any other JSON object. What does a computed state mean, exactly? In the context of data layers, it means the content that was passed into it is processed into some kind of comprehensive object. Let's pretend we're using *dataLayer.push() and* I'm pushing information into the data layer about the page and the user. This is a Single Page App, so we will want to dynamically replace the name of the page as the user navigates. Similar to Google Tag Manager, this pushes the page and user data into the *dataLayer* object (because *dataLayer* makes more sense than *digitalData*):

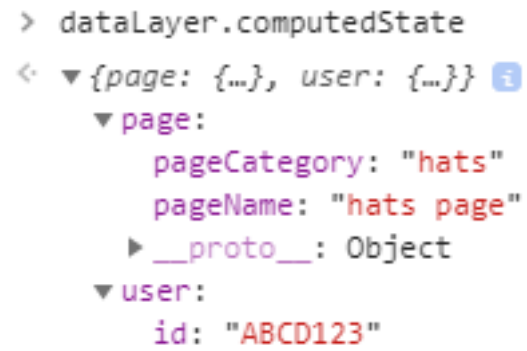


As a business user, it's a bit of a stretch to learn how arrays work. I just want to see what the data looks like when the page loads so I know what I can work with. That's where a computed state is useful. As a technical stakeholder, I can advise the business user to paste *dataLayer.computedState* into their console to see what data is available:

Looks like your average JSON object, right? Let's see what happens when we want to change a field.

Side note: In hindsight, I probably should have named the *pageCategory* and *pageName* fields simply "*category*" and "*name*". I'll save naming convention recommendations for another post…

Here we're just wanting to change the *pageName* and *pageCategory* fields. You can see the shoes data is still in that array above the hats page data. However, since that was the last information passed into the data layer, the computed state should update to reflect those changes:



There you have it. I should have the ability to add data and clear out fields, as well. For those who aren't as technical, **please note that this computed state stuff is NOT functionality baked into *dataLayer.push()* by default**. I did some extra work to manually create the *computedState* object. This is for modeling purposes only. Also note that a key differentiation between this and GTM's data layer is a publicly exposed computed state. GTM does retain a computed state, but isn't (easily) accessible via your console.

With this functionality, the EDDL is more capable and accessible than any other client-side data layer technique.

Final Thoughts

One reason people don't implement data layers is because it's intimidating. You're part of a large organization with many stakeholders. Maintaining data layer standards takes work. You're right. Let's also acknowledge that maintaining *anything* is hard and requires a certain level of discipline. There's turnover on your team. Developers cycle in and out. Oh, by the way – you have to get this stuff prioritized, too!

The Event-Driven Data Layer is easier to document. It's easier to implement and not as vulnerable to timing issues. It's what a minority of sophisticated companies have already built for themselves (100 different ways) and what the majority needs to adopt. This data layer supports any schema you want. If you like how the W3C is structured, build it. If not, don't.

One critical piece you'll notice I didn't link you to some EDDL library. All of this information reflects how an EDDL *should* behave. There are many EDDLs out there and there won't likely be one single standard. However, Event-Driven Data Layers *will* eventually replace the CEDDL model. It makes more sense. I will commit to settling on a single recommendation in the coming months. There are a few examples out there:

1. Google Tag Manager
2. Data Layer Manager

If you have a public-facing event-driven data layer framework, add a comment or message me on Twitter and I would be happy to add it to this list. In the meantime, if you're working on a data layer – don't let the lack of a "standard" stop you from building one. If you want to use the CEDDL, go for it! *Having* a data layer is better than not having one.

3.5 Analytics Export Guide

This guide outlines ways to get data out of Adobe Analytics. It includes:

- **Data feeds:** Receive an hourly or daily export of raw data. Every row is an individual hit, and every column is a variable. Data feeds are typically sent to FTP sites.
- **Data Warehouse:** Use a request wizard to retrieve a spreadsheet output of data. Data Warehouse uses a different processing architecture to allow any number of rows and any number of unique values.
- **FTP and SFTP:** Best practices for using FTP and SFTP with Adobe

Follow the link above to view a tutorial video.

3.6 Adobe Target Business Practitioner Guide

Adobe Target is the Adobe Experience Cloud solution that provides everything you need to tailor and personalize your customers' experience so you can maximize revenue on your web and mobile sites, apps, social media, and other digital channels.

Last Updated: August 14, 2023 (See What Changed)

NOTE

In addition to this guide, the following Adobe Target guides are also available:

- *Adobe Target Developer Guide*
- *Adobe Target Tutorials*

For release information, see Target release notes (current) in this guide.

The following sections point you to useful links in this guide, arranged by intended audience based on typical job functions:

- All Target users
- Marketers
- Developers
- Target and Adobe Experience Cloud admins
- Analysts
- QA engineers

All Target users

Marketers, developers, administrators, analysts, and quality assurance engineers.

- Target release notes: Contains information about the current release, information about known issues that affect Target, a list of important changes to this documentation, and an archive of past release notes.
- Introduction to Target: Explains the core concepts of the Target solution.
- Integrate Target with the Adobe Experience Cloud: Explains how to integrate Target with other Experience Cloud solutions, including Analytics for Target (A4T), Experience Cloud Audiences, and Adobe Campaign.
- Adobe Target Tutorials: Provides tutorials and videos to help you get the most out of Target.
- Troubleshooting Target: Provides links to troubleshooting information contained in this guide, including information about the character limits and other limits (offer size, audiences, profiles, values, parameters, and so forth) that affect activities and other elements in Target.
- Target for mobile apps: Explains how Target can be used for mobile app optimization and personalization.
- Resources and contact information: Provides information about more resources to help you learn about Target features and how to contact Adobe should you need help.

Marketers

- Activities: Explains how to set up, manage, and QA Target activities.
- Audiences: Explains how to determine who sees content and experiences in targeted activities.
- Experiences and offers: Explains how to specify which content displays when a visitor meets the audience criteria for an activity.
- Recommendations: Explains how Recommendations activities automatically display products or content that might interest your customers based on previous user activity or other algorithms.

Developers

- Adobe Target Developer Guide: This portal provides resources and guides for Adobe Target developers, including API and SDK documentation to implement Target.

Target and Adobe Experience Cloud admins

- Administer Target: Explains how to add users and configure your Target account.

Analysts

- Audiences: Explains how to determine who sees content and experiences in targeted activities.
- Reports: Explains how to interpret the performance of your activities.

QA engineers

- Activities: Explains how to set up, manage, and QA Target activities.

3.7 Adobe Experience Cloud Identity Service

The Experience Cloud Identity Service enables the common identification framework for Experience Cloud Application and Services. It works by assigning a unique, persistent ID known as the Experience Cloud ID (ECID) to a site visitor.

Understanding the main entities of identity

To better understand how Adobe helps uniquely identify visitors and resolves identity information, reade the breakdown below:

- **Experience Cloud Identity Service**: The Experience Cloud Identity Service **is responsible for setting the Experience Cloud ID (ECID)**. For more information, read the Experience Cloud Identity Service overview.
- **Experience Cloud ID (ECID)**: The ECID is a shared identity namespace used across Adobe Experience Platform and Adobe Experience Cloud applications to identify people and devices. For more information on the ECID, read the ECID overview.
- **Experience Platform Identity Service**: The Experience Platform Identity Service provides you with a comprehensive view of your customers and their behavior by bridging identities across devices and systems. For more information, read Experience Platform Identity Service overview.

Getting Started

- Overview
- Requirements for the Experience Cloud Identity Service
- Standard Implementation with Platform tags

Experience Cloud ID Javascript Libraries

JavaScript for the Experience Cloud Identity Service is located at: https://github.com/Adobe-Marketing-Cloud/id-service/releases

New or Featured Items

- Opt-in service
- getVisitorValues
- FAQs
- idSyncContainerID

Release Notes

Version 4.4 July 17, 2019 release includes support for the SHA-256 hashing algorithm that allows you to pass in customer IDs or email addresses, and pass out hashed IDs.

Version 4.0 February 12, 2019 release includes the Opt-in service used to identify if you can place a cookie on a user's device or browser when visiting your site.

- See the latest Experience Cloud Release Notes for new features and fixes.
- See the Previous Release Notes section for older releases.

Experience Cloud Resources

- Adobe Privacy Center
- Adobe Experience Cloud
- Adobe Training and Tutorials
- Product Documentation Home

3.8 Analytics Import Guide

This guide explains how to get data into Analytics. It includes:

- **Data Sources**: Upload a file to a designated Adobe FTP site. Adobe retrieves the file and includes the data in your report suite.
- **Bulk Data Insertion API** lets you upload server call data in batches of files instead of using client-side libraries such as AppMeasurement.
- **Data Insertion API**: Send data directly to Adobe's data collection through an API.
- **Adobe Exchange Marketplace**: Find current integrations available to enrich your Adobe Analytics report suite.

- **Data Sources**
- **Bulk Data Insertion API**
- **Data Insertion API**

- **Data Connectors**
- **Import Use Cases**

3.9 Analytics Tools Guide

The Analytics Tools Guide includes information about product features, use cases, task instructions, and best practices for the following Analytics tools:

Tool	Description
Analysis Workspace	Analysis Workspace is a flexible browser tool that allows you to quickly build analyses and share insights. Using the drag-and-drop interface, you can craft your analysis, add visualizations to bring data to life, curate a dataset, share and schedule projects with anyone in your organization.
Analytics dashboards	Analytics dashboards and their mobile scorecards allow executive users to view a broad rendering of important summary data quickly and easily on their own mobile devices. Curators add visualizations to mobile scorecard projects and share them with executives. Scorecards provide a way to target and measure KPIs and provide a clear indication of how well organizations are working to achieve their targets.
Activity Map	Activity Map is an Adobe Analytics application that is designed to rank link activity using visual overlays and provide a dashboard of real-time analytics to monitor audience engagement of your web pages. Activity Map lets you set up different views to visually identify the acceleration of customer activity, quantify marketing initiatives, and act on audience needs and behaviors."
Report Builder	Report Builder is an add-in for Microsoft Excel. Report Builder lets you build customized requests from Adobe Analytics data that are inserted into your Excel worksheets. Requests can dynamically reference cells within your worksheet, and you can update and customize how Report Builder presents the data.
Analytics APIs	Analytics APIs allow you to directly call Adobe's servers to perform almost any action that you can perform in the user interface. You can create reports to explore, get insights, or answer important questions about your data. You can also manage components of Adobe Analytics, such as the creation of segments or calculated metrics.
Reports & Analytics	Reports & Analytics is a tool with dozens of pre-built reports and visualizations. These are now available within Analysis Workspace. **Effective December 31, 2023, Adobe intends to discontinue Reports & Analytics and its accompanying reports and features. Adobe recommends using Analysis Workspace for most reporting needs.** For additional information, see Analysis Workspace.

- **Get started with Adobe Analytics**

- Landing page
- Analysis Workspace
- Report Builder
- Activity Map
- Reports and Analytics
- Adobe Analytics dashboards
- Reporting API
- Labs

3.10 Data Insertion Process

The Data Insertion API supports HTTP POST and HTTP GET for submitting data to Adobe Experience Cloud servers.

Note: The Analytics response to each data insertion includes a status message (SUCCESS or FAILURE).

HTTP POST

Use an HTTP POST to submit properly-formatted Data Insertion XML to the Data Insertion URL. The Data Insertion URL differs from the standard JavaScript data submission URL. Adobe ClientCare can provide the domain name of the Adobe data collection servers where you should send data. For example:

http://namespace.sc.omtrdc.net/b/ss//6
http://namespace.sc.omtrdc.net/b/ss//6
Note: The "6" code at the end of the URL indicates that the data submission requires XML processing.

Upon receipt, Adobe servers perform basic tag validation of the data insertion. If it encounters an error, Adobe returns a Failure response. If the data insertion is successful, Adobe queues the data insertion request for processing by the standard Analytics Data Processing Engine. The engine processes these requests in the same way it processes data collected via JavaScript.
When using HTTP POST with the Data Insertion API, consider the following:

- The Data Insertion API requires data in UTF-8 format. Specify the character encoding in the opening XML tag, as shown in XML Data Insertion Format.
- Replace Ampersand (&), greater-than (>), and less-than (<) symbols with their HTML equivalents when passing them into a Analytics variable. For example, submit <evar1>News & Sports <local> </evar1> as <evar1>News & Sports <local> </evar1>.
- To submit data over an encrypted connection, the application must be configured to support HTTPS POST commands. Some tools that let you do this include:
    - PHP versions 4.3.0 and higher support OpenSSL, and can be used to POST data through SSL port 443. You can see an example of this in Sample Data Insertion (PHP).
    - You can use cURL to send data over an SSL connection. For information about HTTPS POST in Java, see http://java.sun.com/developer/technicalArticles/Security/secureinternet/.
    - Microsoft .Net Framework version 1.1 supports the HTTP header: Expect: 100-Continue in HTTP POST requests, but Experience Cloud servers reject POST data sent with this type of request. To avoid this, set ServicePointManager.Expect100Continue = False. For more information, see http://msdn.microsoft.com/library/en-us/cpref/html/frlrfsystemnetservicepointmanagerclassexpect100continuetopic.asp .

HTTP GET

Use an HTTP GET to submit data to the Data Insertion URL in a query-string format that supports shortened variable names (for more information, see "Variables and Query String Parameters" in the *Analytics Implementation Guide*, available at **Help** > **Documentation** in the Experience Cloud.

HTTP GET reduces bandwidth needs by 30% - 40%, but Adobe data collection servers do not send response messages, so if the data insertion doesn't work as expected, you do not have that feedback for troubleshooting purposes. For more information, see the HTTP GET Sample.

The Data Insertion URL differs from the standard JavaScript data submission URL. The *<rsid>* in the URL is the report suite where you want to submit the data. Adobe ClientCare can provide the domain name of the Adobe data collection servers where you should send data. For example:

http://namespace.sc.omtrdc.net/b/ss/*<rsid\>*/0
http://namespace.sc.omtrdc.net/b/ss/*<rsid\>*/0
Note: The "0" code at the end of the URL indicates that the data submission requires JavaScript processing.

Upon receipt, Adobe servers perform basic tag validation of the data insertion. If it encounters an error, Adobe returns a Failure response. If the data insertion is successful, Adobe queues the data insertion request for processing by the standard Analytics Data Processing Engine. The engine processes these requests in the same way it processes data collected via JavaScript.
Note: The Data Insertion API requires data in UTF-8 format.

3.11 Implement the Experience Cloud in Websites with Tags Tutorial

Implement the Experience Cloud in Websites with Tags is the perfect starting point for front-end developers or technical marketers who want to learn how to implement the Adobe Experience Cloud solutions on their website.

Each lesson contains how-to exercises and foundational information to help you implement the Experience Cloud and understand its value. Demo sites are provided for you to complete the tutorial, so you can learn the underlying techniques in a safe environment. After completing this tutorial, you should be ready to start implementing all of your marketing solutions through tags on your own website.

INFO

This tutorial uses application-specific extensions and libraries (AppMeasurement.js for Adobe Analytics, at.js for Adobe Target). If you are looking to implement Adobe Experience Platform Web SDK, please see the Implement Adobe Experience Cloud with Web SDK tutorial.

After completing this you will be able to:

- Create a tag property
- Install a tag property on a website
- Add the following Adobe Experience Cloud solutions:
  - **Adobe Experience Platform Identity Service**
  - **Adobe Target**
  - **Adobe Analytics**
  - **Adobe Audience Manager**
- Create rules and data elements to send data to the Adobe solutions

- Validate the implementation using the Adobe Experience Cloud Debugger
- Publish changes through development, staging, and production environments

NOTE

Adobe Experience Platform Launch is being integrated into Adobe Experience Platform as a suite of data collection technologies. Several terminology changes have rolled out in the interface which you should be aware of while using this content:

- Platform Launch (Client Side) is now **tags**
- Platform Launch Server Side is now **event forwarding**
- Edge configurations are now **datastreams**

NOTE

Similar multi-solution tutorials are also available for Web SDK and Mobile SDK.

Prerequisites

In these lessons, it is assumed that you have an Adobe Id and the required permissions to complete the exercises. If not, you may need to reach out to your Experience Cloud Administrator to request access.

- For tags, you must have permission to Develop, Approve, Publish, Manage Extensions, and Manage Environments. For more information on tag user permissions, see the documentation.
- For Adobe Analytics, you must know your tracking server and which report suites you will use to complete this tutorial
- For Audience Manager, you must know your Audience Manager Subdomain (also known as the “Partner Name” “Partner ID,” or “Partner Subdomain”)

Also, it is assumed that you are familiar with front-end development languages like HTML and JavaScript. You do not need to be an expert in these languages to complete the lessons, but you will get more out of them if you can comfortably read and understand code.

About tags

The tags feature of Adobe Experience Platform is the next generation of website tag and mobile SDK management capabilities from Adobe. Tags gives customers a simple way to deploy and manage all of the analytics, marketing, and advertising solutions necessary to power relevant customer experiences. There is no additional charge for Tags. It is available for any Adobe Experience Cloud customer.

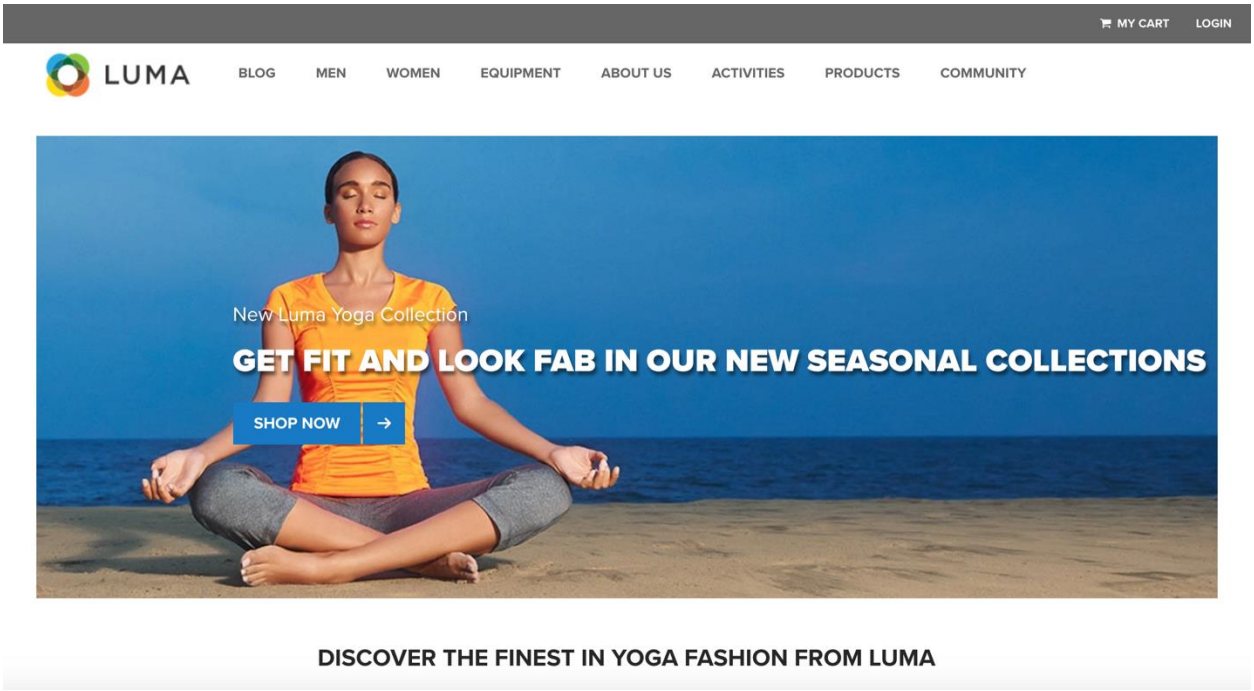
Tags for websites allows you to centrally manage all of the JavaScript related to analytics, marketing, and advertising solutions used on your desktop and mobile sites. For example, if you deploy Adobe Analytics, tags will manage the AppMeasurement JavaScript library, populate variables, and fire requests.

The content of your container is minified, including your custom code. Everything is modular. If you don't need an item, it is not included in your library. The result is an implementation that is fast and compact.

Tags is also a platform that allows third-party vendors to create extensions to make it easy to deploy their solutions through tags. An extension is a package of code (JavaScript, HTML, and CSS) that extends the tags interface and client functionality. You can think of tags as an operating system, and extensions are the apps you use to achieve your tasks.

About the Lessons

In these lessons, you will implement the Adobe Experience Cloud into a fake retail website called Luma. The Luma site has a rich data layer and functionality that will allow you to build a realistic implementation. You will build your own tag property, in your own Experience Cloud organization, and map it to our hosted Luma site using the Experience Cloud Debugger.



Get the Tools

1. Because you will be using some browser-specific extensions, we recommend completing the tutorial using the Chrome Web Browser
2. Add the Adobe Experience Cloud Debugger extension to your Chrome browser
3. Download the sample html page (right-click on this link and click "Save Link As")
4. Get a text editor in which you can make changes to the sample html page. (If you don't have one, we recommend trying Brackets)
5. Bookmark the Luma site

NOTE

You might find it easier to complete this tutorial with the Luma site open in Chrome, while you read this tutorial and complete the Data Collection interface steps in a different browser.

Let’s get started!