Event Tracking with Data Layer and Data Attributes

Abhishek Tiwari

Citation: *A. Tiwari*, "Event Tracking with Data Layer and Data Attributes", Abhishek Tiwari, 2014. doi:10.59350/27ntt-v1d68

Published on: January 03, 2014

Recently I wrote about a standard data layer for customer experience data. Extending our last discussion, in this post I am going to describe how you can use custom data attributes (`data-*`) and data layer for event tracking.

As an example and to demonstrate use of custom data attributes and data layer, we are going to use Google Analytics. But you can use similar approach with other tracking services such as Omniture SiteCatalyst, WebTrends, etc.

Custom Data Attributes

A custom data attribute is an attribute whose name starts with the string `data-`. The W3C specification for data-attributes suggests that:

> Custom data attributes are intended to store custom data private to the page or application, for which there are no more appropriate attributes or elements.

Custom data attributes are not specific to HTML5. All browsers including IE6/7 already let you use the `data-*` attribute. To access `data-*` attributes in JavaScript you can use `.getAttribute()` method. Otherwise in jQuery `.data()` method is available to access the `data-*` attributes.

Data Layer

As I mentioned in my recent post, a data layer has two primary data sources,

1. Data received from server-side (core data about page and user)
2. Data appended by tag layer or client-side JavaScript (cookies, events, page attributes, etc.)

Raw event tracking

Classic Google Analytics

Classic Google analytics (`ga.js`) uses event tracking method `_trackEvent()`. On event hit you have to call `_trackEvent()` method with parameters `category`, `action`, **`label`**, `value`, etc.

```
// label and value are optional
_trackEvent('category', 'action', 'label', 'value')
```

This example illustrates how you might use the event tracking method `_trackEvent()` to record user interaction with a video Play button on your page.

```
<a href="#" onClick="_gaq.push(['_trackEvent', 'Videos', 'Play', 'Gone
   With the Wind', 4]);">Play</a>
```

In this case, reports will display this event,

- *Videos* as event category
- *Play* as event action
- *Gone With the Wind* as event label

Universal Analytics

If you are using Universal Analytics (`analytics.js`) then on event hit you send event details using `ga` function.

To send an event, you pass the `ga` function the `send` command with the `event` hit type along with event details such as `category`, `action`, **`label`**, `value`, etc.

```
// label and value are optional
ga('send', 'event', 'category', 'action', 'label', `value`);
```

This is same as above but with explicit field names. ~~~ ga('send', { 'hitType': 'event', 'eventCategory': 'category', 'eventAction': 'action', 'eventLabel': 'label', 'eventValue': `value` }); ~~~

This shows how you can employ event tracking function `ga` with `send` command to record user interaction with a video Play button on your page.

```
<a href="#" onClick="ga('send', 'event', 'Videos', 'Play', 'Gone With the
    Wind', 4);">Play</a>
```

Event tracking using data=* attribute

A more cleaner way to accomplish above examples will be to use custom `data-*` attributes. In following example, we are using 4 custom data attributes,

- `data-ev` - event tracking tag name
- `data-category` - category of event
- `data-action` - type of interaction
- `data-`**`label`** - label for event

```
<a href="#" data-event="ev" data-category="Videos" data-action="Play" data
    -label="Gone With the Wind" data-value="4">Play</a>
```

In addition to above markup, we will also need a JavaScript snippet to read custom data attributes when event fires and send or push to tracking tag. Following JavaScript will be triggered when user clicks `Play` button and it will read custom data attributes for click element and push event data to both classic Google Analytics and Universal Analytics.

```
eventTracking = function () {
    $t = $('[data-event="ev"]');
    $t.click(function () {
        var evCat = $(this).data('category') ? $(this).data('category') :
            '',
            evAct = $(this).data('action') ? $(this).data('action') : '',
            evLab = $(this).data('label') ? $(this).data('label') : '',
            evVal = $(this).data('value') ? $(this).data('value') : '';
        try {
            _gaq.push(['_trackEvent', evCat, evAct, evLab, evVal]);
            ga('send', 'event', evCat, evAct, evLab, evVal);
        } catch (e) {
            console.log(e);
        }
    });
};
```

As you can see, custom data attributes make event tracking more re-usable. Now you can track events with multiple tags on same time, in this case Google Analytics and Universal Analytics.

Event tracking using data layer

Google Tag Manager

In order to trigger the event tag in Google Tag Manager(GTM), we just need to call the push function of the `dataLayer` object as described below:

```
<a href="#" onclick="
dataLayer.push({
    'event': 'myTrackEvent',
    'eventCategory': 'Videos',
    'eventAction': 'Play',
    'eventLabel': 'Gone With the Wind',
    'eventValue': '4.00'
});">Play</a>
```

Obviously your Google Tag Manager setup includes Google Analytics or Universal Analytics tag, and a custom event tag named `myTrackEvent` so that event data can be collected when event fires.

As you can see, markup is mixed with JavaScript code. So a better option will to use custom data attributes in markup.

```
<a href="#" data-event="ev" data-category="Videos" data-action="Play" data
   -label="Gone With the Wind" data-value="4">Play</a>
```

So how it works? Simply put, when user interacts with a page, event data held by custom data at-

tributes is appended to data layer using following JavaScript snippet. An event update in data layer is captured by tracking tags and pushed to data collection servers.

```
eventTracking = function () {
    $t = $('[data-event="ev"]');
    $t.click(function () {
        var evCat = $(this).data('category') ? $(this).data('category') :
            '',
            evAct = $(this).data('action') ? $(this).data('action') : '',
            evLab = $(this).data('label') ? $(this).data('label') : '',
            evVal = $(this).data('value') ? $(this).data('value') : '';
        try {
            dataLayer.push({;
                'event': 'myTrackEvent',
                'eventCategory': evCat,
                'eventAction': evAct,
                'eventLabel': evLab,
                'eventValue': evVal,
            });
        } catch (e) {
            console.log(e);
        }
    });
};
```

Customer Experience Digital Data Layer (CEDDL)

How events are described using CEDDL? Well, CEDDL provides an `event` object to collect information when user interact with page or page components.

```
digitalData.event[n].eventInfo = {
    eventName: "Playing video Gone With the Wind",
    eventAction: "Play",
    eventPoints: 4,
    type: "Videos",
};
```

In addition, you can also describe the a `component` object.

> Component object is intended to capture information about a content component included as part of a page, such as a video

```
digitalData.component[n].componentInfo = {
    componentID: "video123",
    componentName: "Gone With the Wind",
    description: "Gone With the Wind Video"
};
```

Any interaction with component — in this case playing video — will be captured by event object. Although not very clear in specification but I believe `componentID` is element id.

```
<a href="#" id="video123" .. .. >Play</a>
```