**Dispatching Custom Events for Front End Analytics Application**

When we have a web analytics application that collects front end event data such as Tealium, Adobe Analytics and so on, we can dispatch an custom event whenever user does action on the page. For example, we can dispatch a custom event to indicate the user clicked certain button on the page.

Event can be emitted from the custom object that is exposed globally.

Then, the analytics application can listen to the event and do something like this.

<pre>

mysite.events.on('widget:opened', function() {

  // Whatever the analytics function provided by the vendor

  whatever.function.sendData('widget:opened');

});

</pre>

The easiest way to do this is to use node.js events. For example, we have a global namespace called mysite. We can add events object that is the EventEmitter.

Then, we can simply emit event when the element is clicked. This works on most browsers including IE11.

<pre>

import { EventEmitter } from 'events';

mysite.events = new EventEmitter();

document.targetElem.addEventListener('click', () => {

  mysite.events.emit('widget:opened');

})

</pre>

Because events are from node.js, you may need to use plugins in the bundler to use the module. For example, if you are using rollup, you need to use rollup-plugin-node-builtins.

The alternative is to dispatch a custom event. It is not as clean as emitting event, but it also works.

<pre>

const widgetOpenEvent = new CustomEvent('widget:opened')

document.targetElem.addEventListener('click', () => {

  document.dispatchEvent(widgetOpenEvent);

});

</pre>

Then, analytics code can attach an event listener to the document to listen to this custom event. Note that CutomEvent requires a polyfill if you need to make it work with IE11.