It's time to ditch the client-side tag manager

Michel Bieze

Shell Recharge Solutions

Tag Managers are like closets

Image Source: pexels.com | Author: Ron Lach

Tag Managers are like closets

Image Source: pexels.com | Author: Ron Lach

Messy closet symptoms

✓ Lost overview

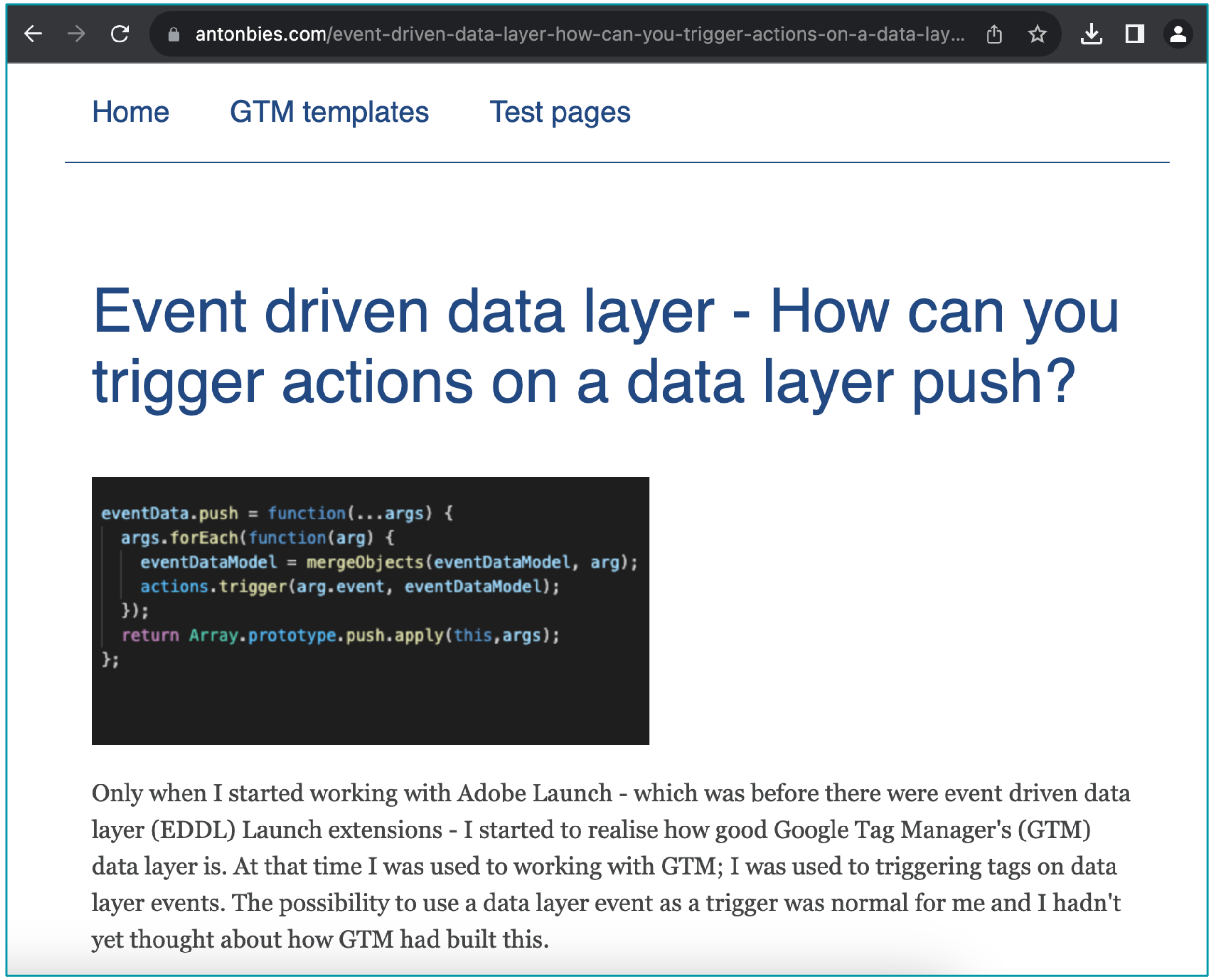
✓ Large container size

✓ Custom JavaScript tags

The opportunity

Tag Manager

The inspiration



antonbies.com/event-driven-data-layer-how-can-you-trigger-actions-on-a-data-lay...

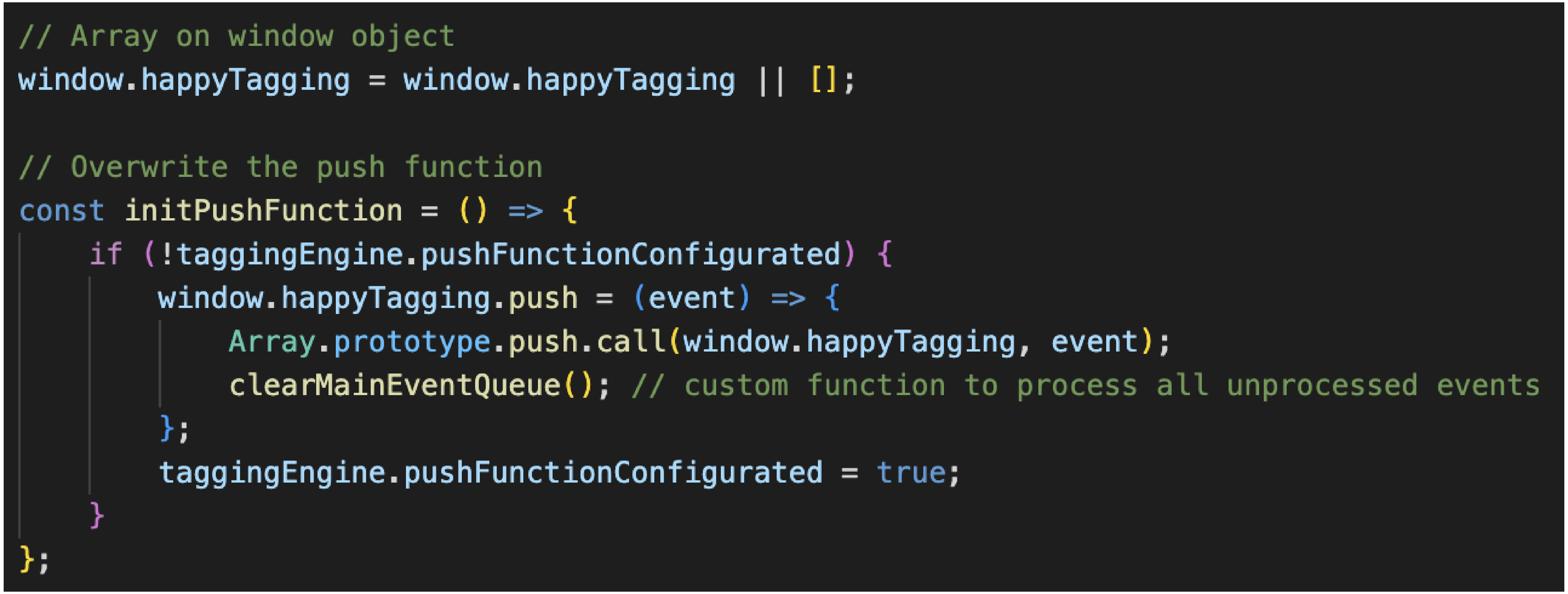
Event driven data layer - How can you trigger actions on a data layer push?

```
eventData.push = function(...args) {
  args.forEach(function(arg) {
    eventDataModel = mergeObjects(eventDataModel, arg);
    actions.trigger(arg.event, eventDataModel);
  });
  return Array.prototype.push.apply(this,args);
};
```

Only when I started working with Adobe Launch - which was before there were event driven data layer (EDDL) Launch extensions - I started to realise how good Google Tag Manager's (GTM) data layer is. At that time I was used to working with GTM; I was used to triggering tags on data layer events. The possibility to use a data layer event as a trigger was normal for me and I hadn't yet thought about how GTM had built this.

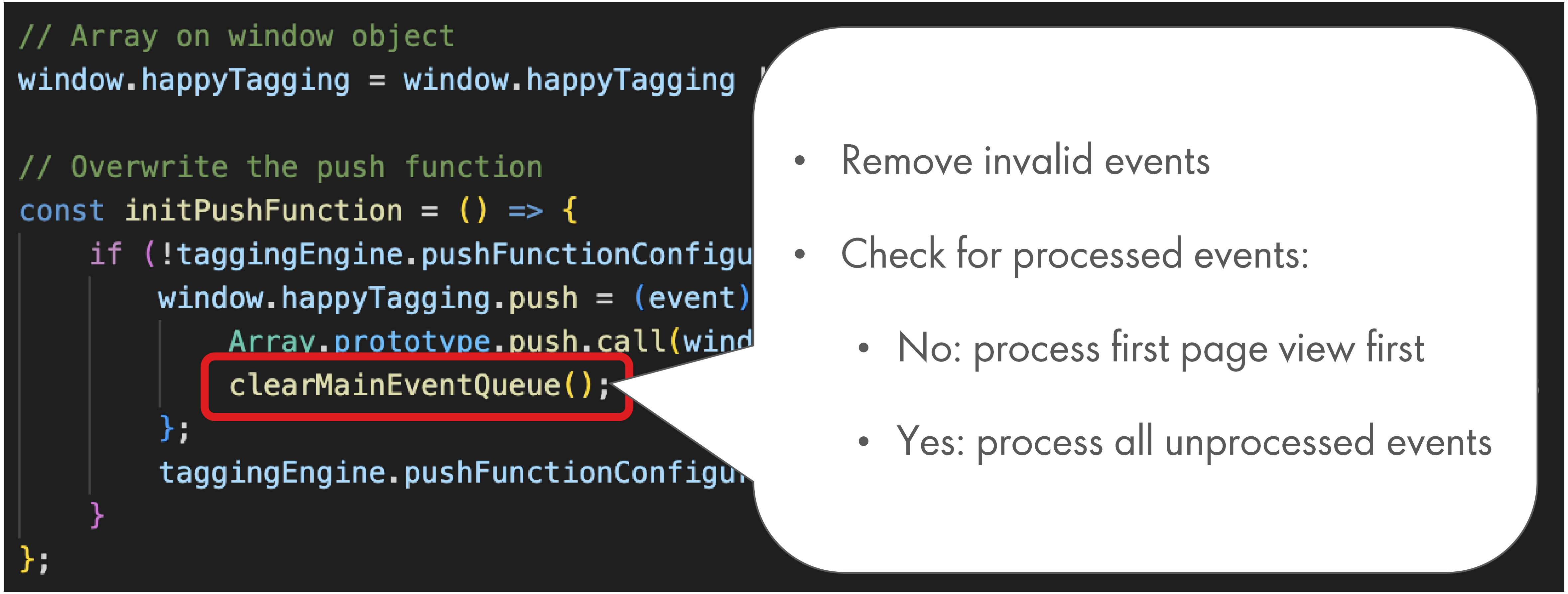
Source: https://antonbies.com/event-driven-data-layer-how-can-you-trigger-actions-on-a-data-layer-push/

Overwriting the push function



```
// Array on window object
window.happyTagging = window.happyTagging || [];

// Overwrite the push function
const initPushFunction = () => {
    if (!taggingEngine.pushFunctionConfigurated) {
        window.happyTagging.push = (event) => {
            Array.prototype.push.call(window.happyTagging, event);
            clearMainEventQueue(); // custom function to process all unprocessed events
        };
        taggingEngine.pushFunctionConfigurated = true;
    }
};
```

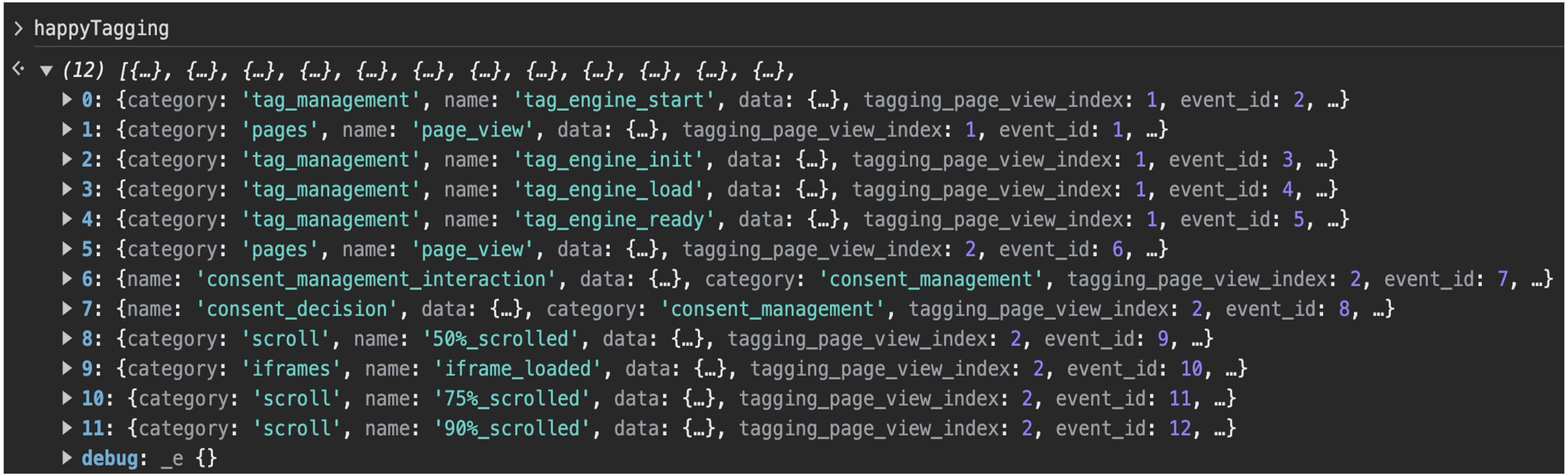
Clearing the event queue



```
// Array on window object
window.happyTagging = window.happyTagging

// Overwrite the push function
const initPushFunction = () => {
    if (!taggingEngine.pushFunctionConfigu
        window.happyTagging.push = (event)
            Array.prototype.push.call(wind
            clearMainEventQueue();
        };
        taggingEngine.pushFunctionConfigu
    }
};
```

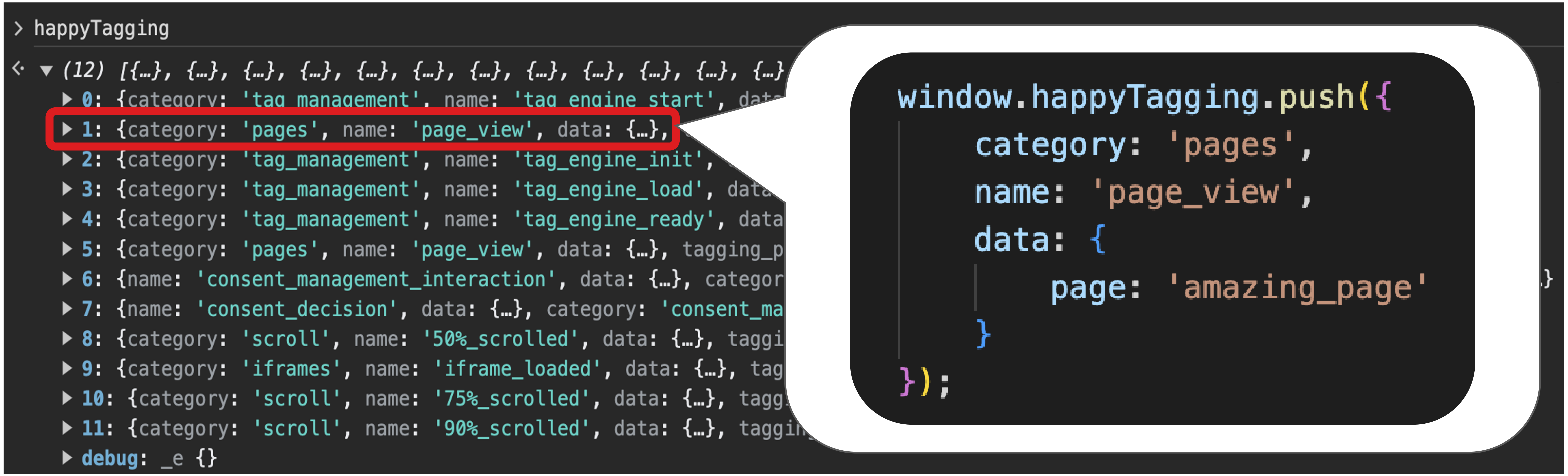
- Remove invalid events
- Check for processed events:
  - No: process first page view first
  - Yes: process all unprocessed events

The event queue



```
> happyTagging
<· ▼ (12) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
    ▶ 0: {category: 'tag_management', name: 'tag_engine_start', data: {…}, tagging_page_view_index: 1, event_id: 2, …}
    ▶ 1: {category: 'pages', name: 'page_view', data: {…}, tagging_page_view_index: 1, event_id: 1, …}
    ▶ 2: {category: 'tag_management', name: 'tag_engine_init', data: {…}, tagging_page_view_index: 1, event_id: 3, …}
    ▶ 3: {category: 'tag_management', name: 'tag_engine_load', data: {…}, tagging_page_view_index: 1, event_id: 4, …}
    ▶ 4: {category: 'tag_management', name: 'tag_engine_ready', data: {…}, tagging_page_view_index: 1, event_id: 5, …}
    ▶ 5: {category: 'pages', name: 'page_view', data: {…}, tagging_page_view_index: 2, event_id: 6, …}
    ▶ 6: {name: 'consent_management_interaction', data: {…}, category: 'consent_management', tagging_page_view_index: 2, event_id: 7, …}
    ▶ 7: {name: 'consent_decision', data: {…}, category: 'consent_management', tagging_page_view_index: 2, event_id: 8, …}
    ▶ 8: {category: 'scroll', name: '50%_scrolled', data: {…}, tagging_page_view_index: 2, event_id: 9, …}
    ▶ 9: {category: 'iframes', name: 'iframe_loaded', data: {…}, tagging_page_view_index: 2, event_id: 10, …}
    ▶ 10: {category: 'scroll', name: '75%_scrolled', data: {…}, tagging_page_view_index: 2, event_id: 11, …}
    ▶ 11: {category: 'scroll', name: '90%_scrolled', data: {…}, tagging_page_view_index: 2, event_id: 12, …}
    ▶ debug: _e {}
```

Adding events to the queue



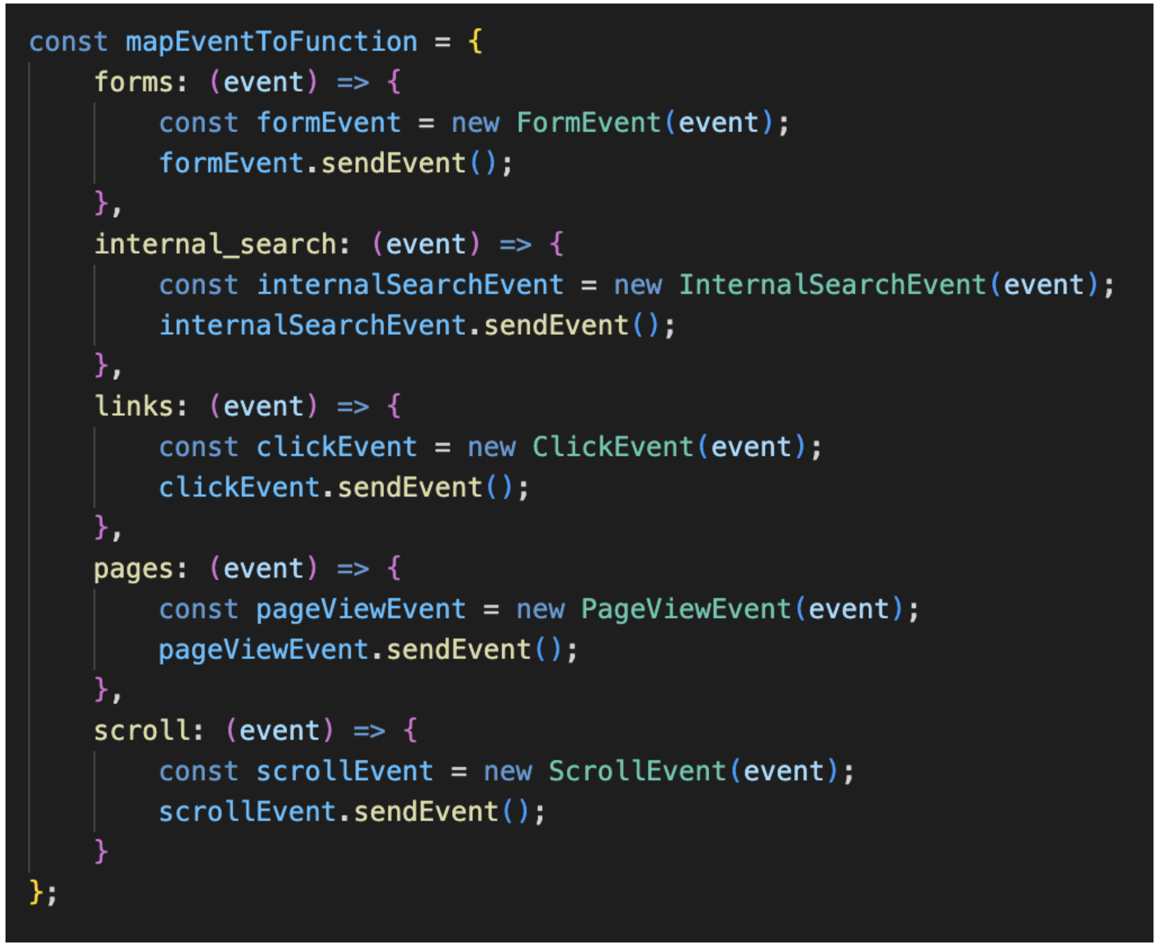
```
> happyTagging
<· ▼ (12) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}
   ▶ 0: {category: 'tag_management', name: 'tag_engine_start', data
   ▶ 1: {category: 'pages', name: 'page_view', data: {…},
   ▶ 2: {category: 'tag_management', name: 'tag_engine_init',
   ▶ 3: {category: 'tag_management', name: 'tag_engine_load', data
   ▶ 4: {category: 'tag_management', name: 'tag_engine_ready', data
   ▶ 5: {category: 'pages', name: 'page_view', data: {…}, tagging_p
   ▶ 6: {name: 'consent_management_interaction', data: {…}, categor
   ▶ 7: {name: 'consent_decision', data: {…}, category: 'consent_ma
   ▶ 8: {category: 'scroll', name: '50%_scrolled', data: {…}, taggi
   ▶ 9: {category: 'iframes', name: 'iframe_loaded', data: {…}, tag
   ▶ 10: {category: 'scroll', name: '75%_scrolled', data: {…}, tagg
   ▶ 11: {category: 'scroll', name: '90%_scrolled', data: {…}, taggin
   ▶ debug: _e {}
```

```
window.happyTagging.push({
    category: 'pages',
    name: 'page_view',
    data: {
        page: 'amazing_page'
    }
});
```

Event processing

- Add: Event ID
- Add: Tagging Page View Index
- Enrich with page properties
- Dispatch JS event

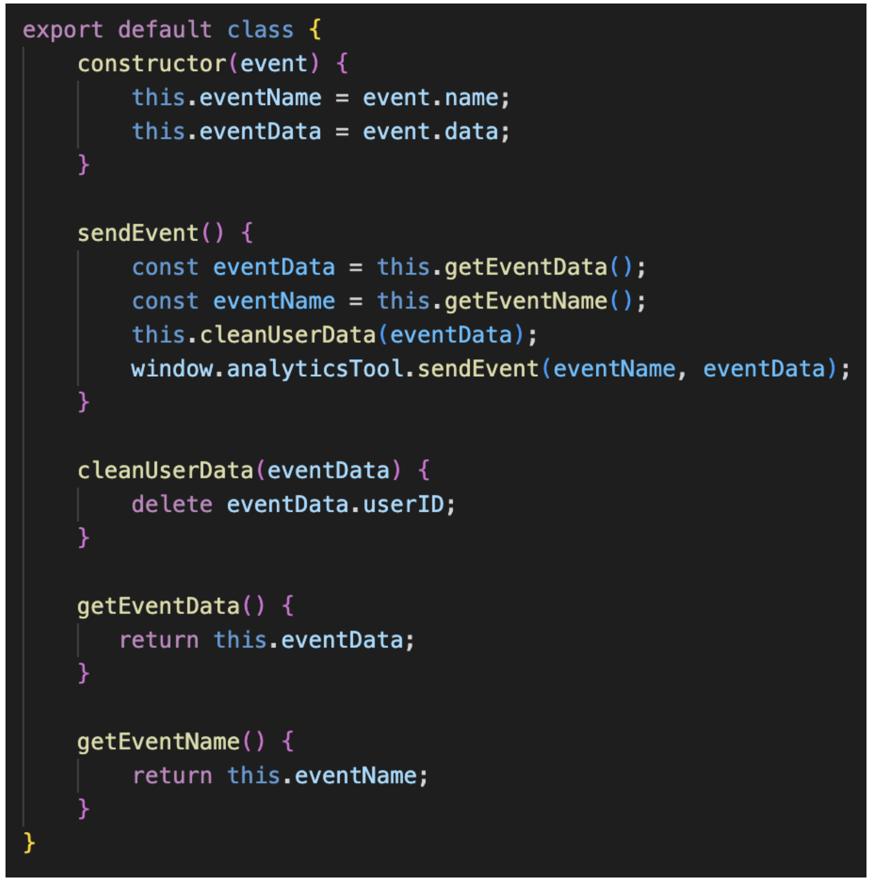
Lookup objects to keep things clear and functionally grouped...



```
const mapEventToFunction = {
    forms: (event) => {
        const formEvent = new FormEvent(event);
        formEvent.sendEvent();
    },
    internal_search: (event) => {
        const internalSearchEvent = new InternalSearchEvent(event);
        internalSearchEvent.sendEvent();
    },
    links: (event) => {
        const clickEvent = new ClickEvent(event);
        clickEvent.sendEvent();
    },
    pages: (event) => {
        const pageViewEvent = new PageViewEvent(event);
        pageViewEvent.sendEvent();
    },
    scroll: (event) => {
        const scrollEvent = new ScrollEvent(event);
        scrollEvent.sendEvent();
    }
};
```

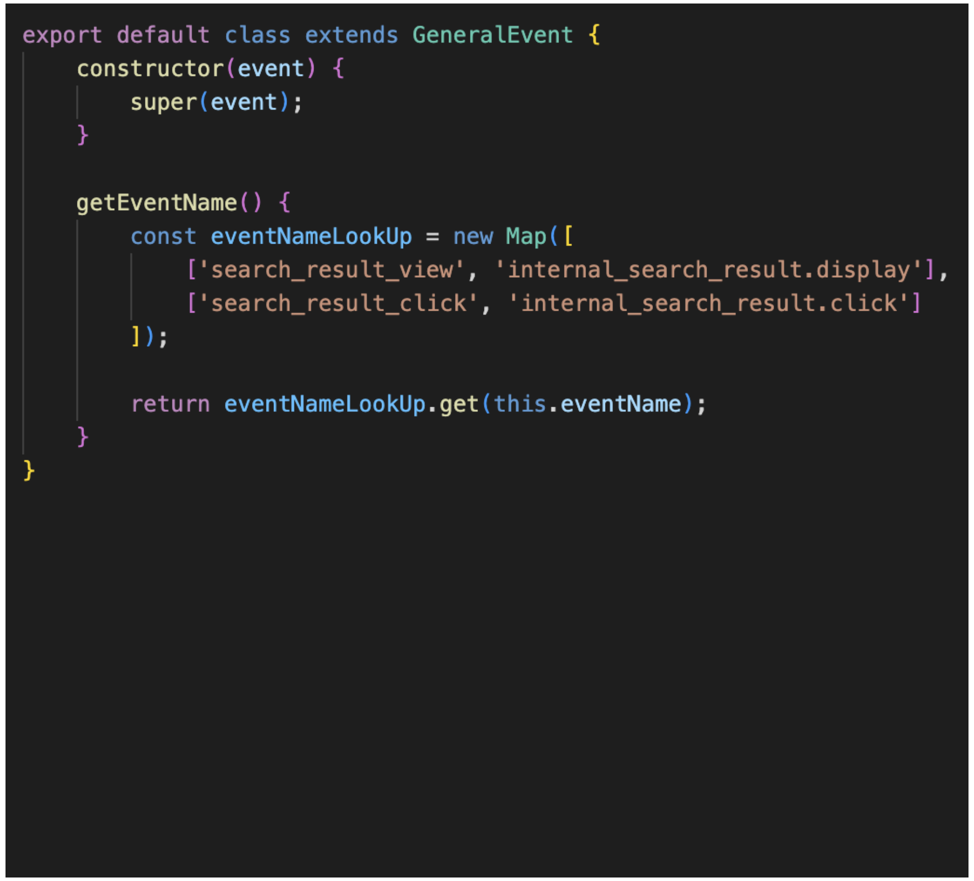
… and process events through class-based functions

Generic



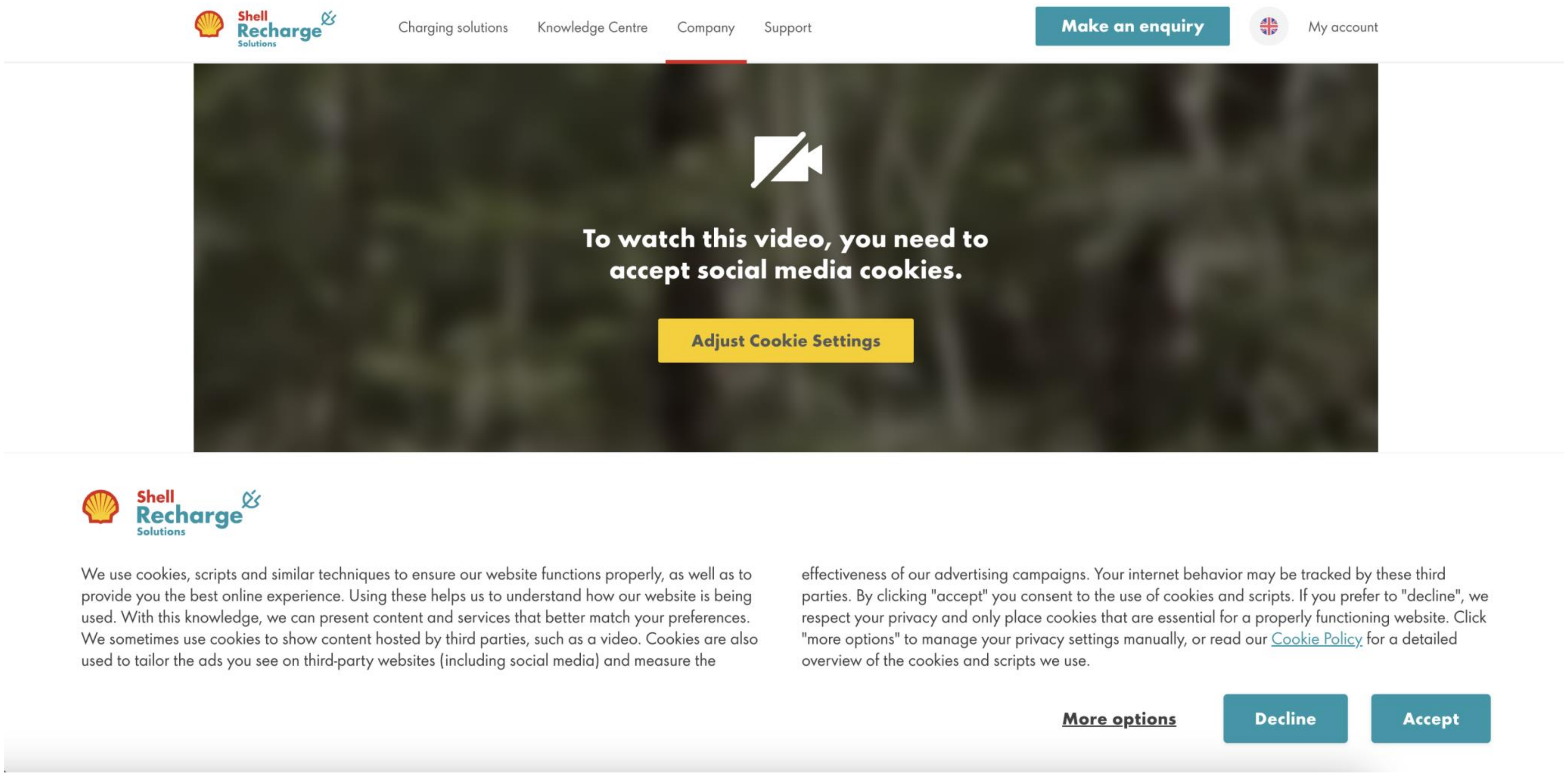
```
export default class {
    constructor(event) {
        this.eventName = event.name;
        this.eventData = event.data;
    }

    sendEvent() {
        const eventData = this.getEventData();
        const eventName = this.getEventName();
        this.cleanUserData(eventData);
        window.analyticsTool.sendEvent(eventName, eventData);
    }

    cleanUserData(eventData) {
        delete eventData.userID;
    }

    getEventData() {
       return this.eventData;
    }

    getEventName() {
        return this.eventName;
    }
}
```

Specific to internal search



```
export default class extends GeneralEvent {
    constructor(event) {
        super(event);
    }

    getEventName() {
        const eventNameLookUp = new Map([
            ['search_result_view', 'internal_search_result.display'],
            ['search_result_click', 'internal_search_result.click']
        ]);

        return eventNameLookUp.get(this.eventName);
    }
}
```

Easier integrations as triggers are not confined in the Tag Manager



We use cookies, scripts and similar techniques to ensure our website functions properly, as well as to provide you the best online experience. Using these helps us to understand how our website is being used. With this knowledge, we can present content and services that better match your preferences. We sometimes use cookies to show content hosted by third parties, such as a video. Cookies are also used to tailor the ads you see on third-party websites (including social media) and measure the effectiveness of our advertising campaigns. Your internet behavior may be tracked by these third parties. By clicking "accept" you consent to the use of cookies and scripts. If you prefer to "decline", we respect your privacy and only place cookies that are essential for a properly functioning website. Click "more options" to manage your privacy settings manually, or read our Cookie Policy for a detailed overview of the cookies and scripts we use.

Not all scripts need to be loaded …at the same time

- Only load desired scripts
- 2 requests to load all scripts:
  - Load base scripts
  - Load optional scripts

What do you need to get started

- ✓ Method to inject scripts
- ✓ CDN / file hosting
- ✓ Code Repository e.g., GitHub
  - ✓ Version management
  - ✓ Approval flows
  - ✓ Code transpilation & bundling
  - ✓ Dependency scanning

Pro’s and Con’s

Pro’s

- Vendor agnostic
- Scalable
- Tailored
- Integrate with test automation & dependency scanning

Con’s

- Tech & JS heavy
- Solution availability
- Maintenance
- Too complex for straightforward implementations

Thank you

Shell Recharge Solutions