Adobe

DEVELOPERS LIVE

Adobe Client Data Layer: Track your visitor data

Jean-Christophe Kautzmann

Laurentiu Magureanu

Benedikt Wedenik

Who are we?

Jean-Christophe Kautzmann

Sr. Software Engineer
Adobe AEM Sites

Laurentiu Magureanu

Software Engineer
Adobe AEM CIF

Benedikt Wedenik

Senior Consultant
Cloud Expert

Agenda

- The Adobe Client Data Layer
- Integration with the AEM Core Components
- Integration with custom components
- Integration with Adobe Launch
- Q & A

The Adobe Client Data Layer

What is a data layer?

A data layer consists of a JavaScript client-side event-driven data store that can be used on web pages:

- to collect data about what the visitors experience on the web page;
- to communicate this data to digital analytics and reporting servers.

What does the Adobe Client Data Layer do?

The Adobe Client Data Layer is a JavaScript store for data and events happening on a page within the scope of a request. It provides an API to:

- Register data that is to be merged into the data layer state.
- Trigger events that relate to the data stored in the data layer.
- Get the current data layer state of all merged data.
- Register listeners that are called for specific events or data changes.

Getting the Data Layer ready

- Loading the data layer script:

```
<script src="adobe-client-data-layer.min.js" defer async></script>
```

- Declaring the adobeDataLayer array:

```
window.adobeDataLayer = window.adobeDataLayer || [];
```

adobeDataLayer.push()

- Pushing a Data Object:

```
window.adobeDataLayer.push({
    "page": {
        "title": "Getting Started"
    }
});
```

- Pushing an Event Object:

```
window.adobeDataLayer.push({
    "event": "show",
    "eventInfo": {
        "reference": "component.accordion-1-item-2"
    },
    "component": {
        "accordion-1": {
            "shownItems": [
                "component.accordion-1-item-1",
                "component.accordion-1-item-2"
            ]
        }
    }
});
```

adobeDataLayer.push()

- Pushing an Object to Delete Data:

```
window.adobeDataLayer.push({
    "component": {
        "map-1": null,
        "accordion-1": {
            "shownItems": [
                undefined,
                "component.accordion-1-item-2"
            ]
        }
    }
});
```

adobeDataLayer.push()

- Pushing a Function:

```
var myHandler = function(event) {
    console.log(event);
};
window.adobeDataLayer.push(function(dl) {
    dl.getState();
    dl.addEventListener("click", myHandler);
});
```

adobeDataLayer.getState()

- Getting the merged state:

```
window.adobeDataLayer.push(function(dl) {
    var state = dl.getState();
    console.log(state);
});
```

- Console output:

```
{
    "page": {
        "title": "Getting Started"
    },
    "component": {
        "hero-1": {
            "title": "Learn More",
            "link": "learn-more.html"
        },
        "accordion-1": {
            "title": "Step by step",
            "shownItems": [
                "component.accordion-1-item-2"
            ]
        },
        "accordion-1-item-1": {…},
        "accordion-1-item-2": {…}
    }
}
```

adobeDataLayer.getState()

- Getting the merged state of a specific part:

```
window.adobeDataLayer.push(function(dl) {
    var state = dl.getState("component.hero-1");
    console.log(state);
});
```

- Console output:

```
{
    "title": "Learn More",
    "link": "learn-more.html"
}
```

adobeDataLayer.addEventListener()

- Registering an Event Listener:

```
var myHandler = function(event) {
    console.log(event);
};
window.adobeDataLayer.push(function(dl) {
    dl.addEventListener(
        "adobeDataLayer:change",
        myHandler,
        {"path": "component.accordion-1"}
    );
});
```

- Console output:

```
{
    "component": {
        "accordion-1": {
            "title": "Step by step",
            "shownItems": [
                "component.accordion-1-item-1"
            ]
        },
        "accordion-1-item-1": {
            "title": "Step One",
            "parent": "component.accordion-1"
        },
        "accordion-1-item-2": {
            "title": "Step Two",
            "parent": "component.accordion-1"
        }
    }
}
{
    "event": "show",
    "eventInfo": {
        "reference": "component.accordion-1-item-2"
    },
    "component": {
        "accordion-1": {
            "shownItems": [
                "component.accordion-1-item-1",
                "component.accordion-1-item-2"
            ]
        }
    }
}
```

Adobe

adobeDataLayer.removeEventListener()

- Unregistering all listeners for the adobeDataLayer:change event:

```
window.adobeDataLayer.push(function(dl) {
    dl.removeEventListener("adobeDataLayer:change");
});
```

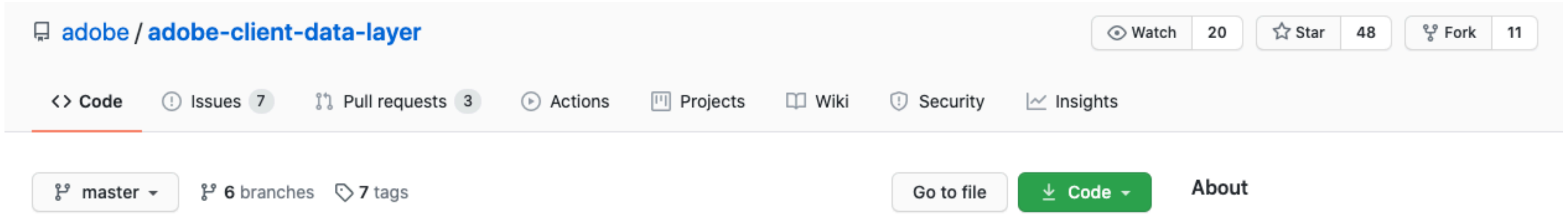
- Unregistering a specific listener for the click event:

```
window.adobeDataLayer.push(function(dl) {
    dl.removeEventListener("click", myHandler);
});
```

Contributions welcome!

Feel free to contribute to the ACDL project (questions, issues, PRs, feedback, …):

https://github.com/adobe/adobe-client-data-layer



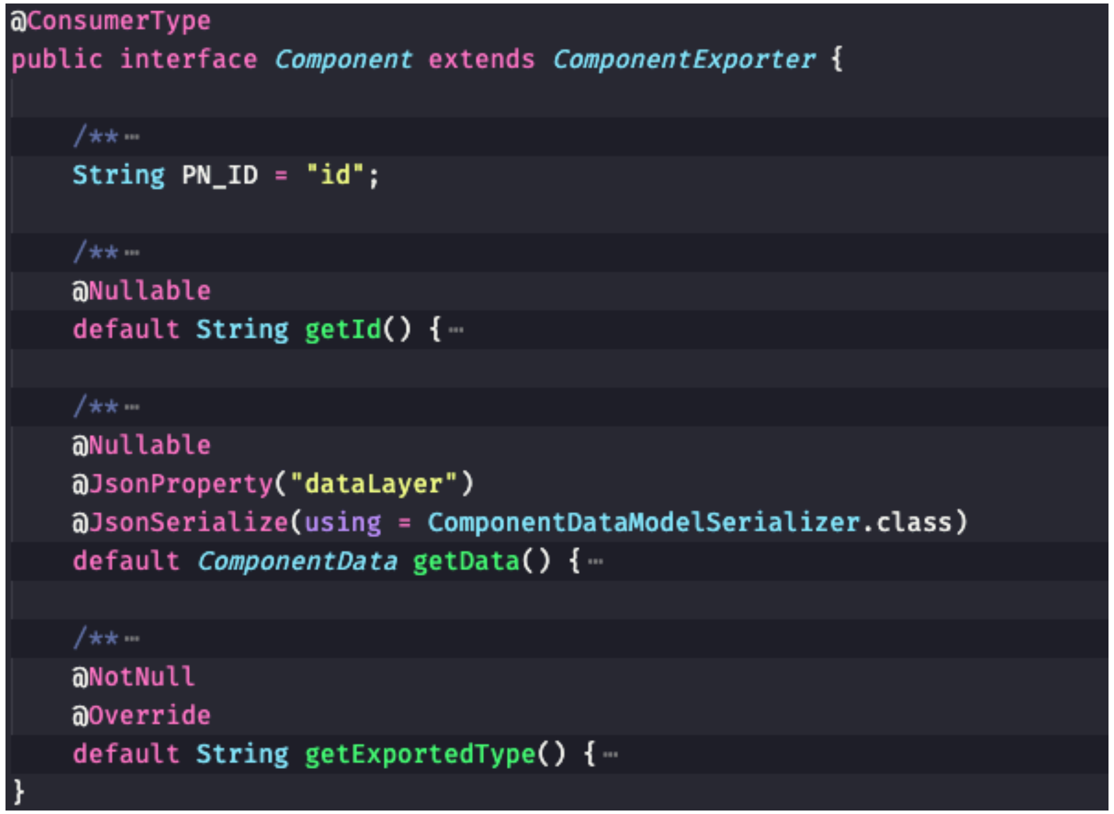
Integration with the AEM Core Components

Populating the data layer

- Each component generates its state as JSON
- The HTL scripts of the component renders that JSON as data-attribute
- A JS utility will capture all the generated data and push it to DL
- The same JS utility will add event handlers for click events
- Special components (e.g.: *Accordion*) will push custom events (show, hide)

Generating component data

- Component interface has a *getData* method

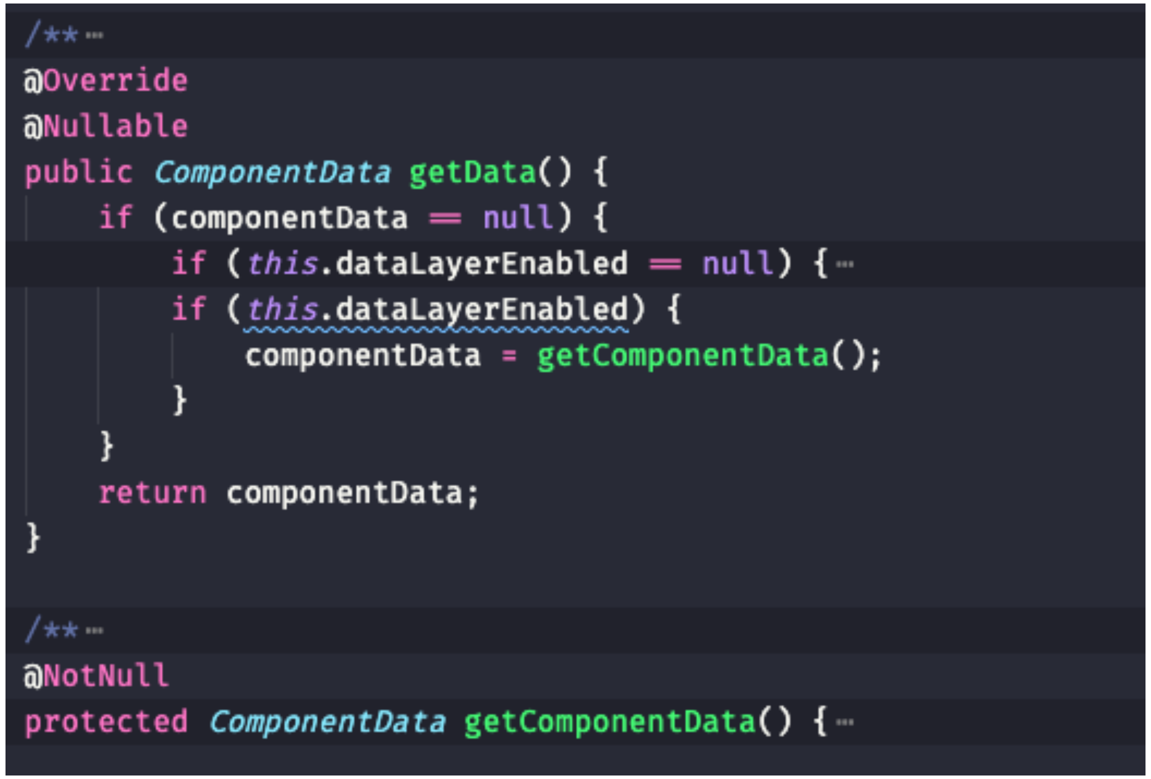


```
@ConsumerType
public interface Component extends ComponentExporter {

    /**…
    String PN_ID = "id";

    /**…
    @Nullable
    default String getId() {…

    /**…
    @Nullable
    @JsonProperty("dataLayer")
    @JsonSerialize(using = ComponentDataModelSerializer.class)
    default ComponentData getData() {…

    /**…
    @NotNull
    @Override
    default String getExportedType() {…
}
```

Adobe

Generating component data

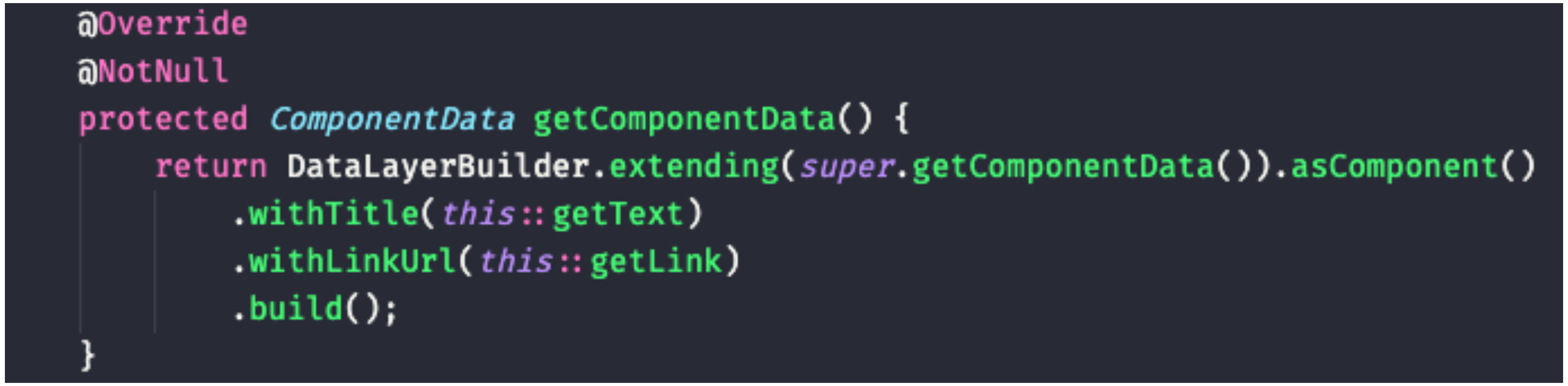
- The *AbstractComponent* implementation relies on calling *getComponentData*



```
/** ⋯
@Override
@Nullable
public ComponentData getData() {
    if (componentData == null) {
        if (this.dataLayerEnabled == null) {⋯
        if (this.dataLayerEnabled) {
            componentData = getComponentData();
        }
    }
    return componentData;
}

/** ⋯
@NotNull
protected ComponentData getComponentData() {⋯
```

Generating component data

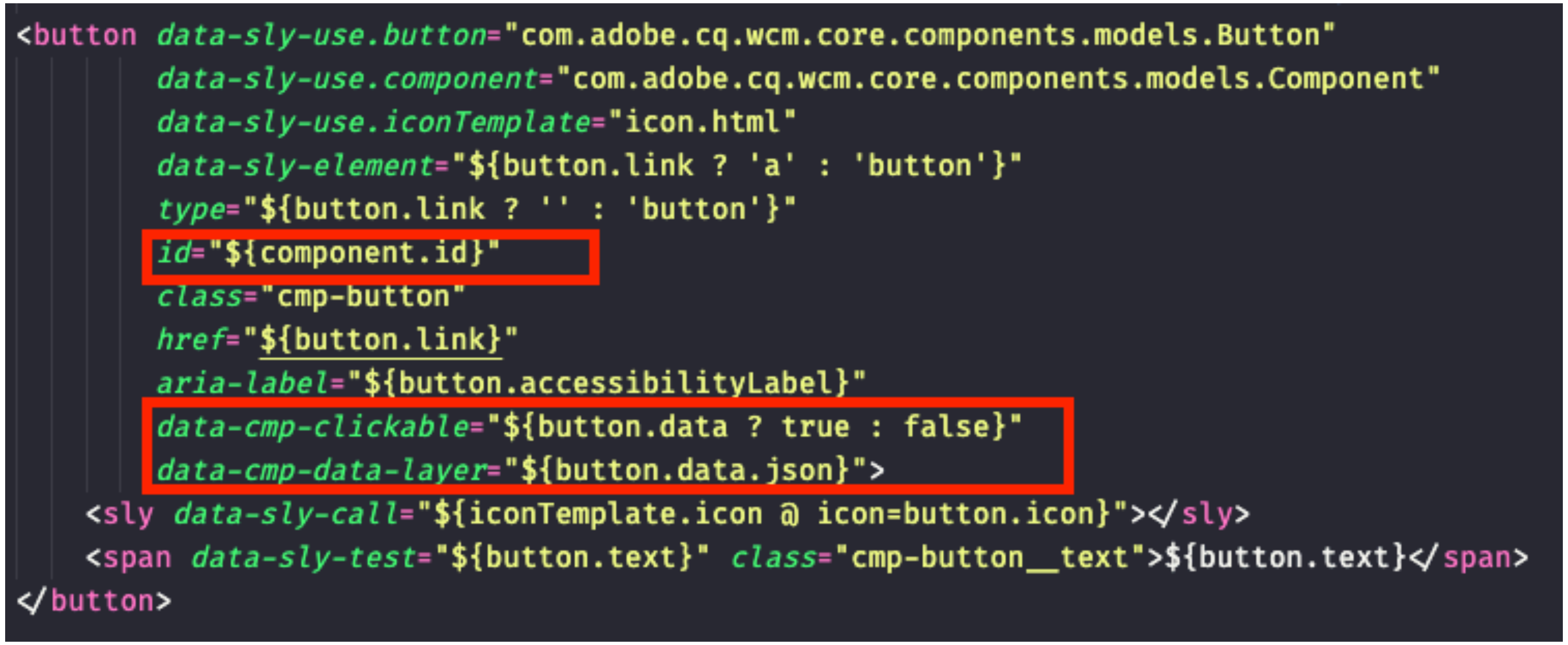
- Each component can override *getComponentData* to customize the output
- The customization is done using the *DataLayerBuilder* utilities



```
    @Override
    @NotNull
    protected ComponentData getComponentData() {
        return DataLayerBuilder.extending(super.getComponentData()).asComponent()
            .withTitle(this::getText)
            .withLinkUrl(this::getLink)
            .build();
    }
```

Generating component data

- The HTL script renders the *ID* and component data JSON
- If the component is clickable it renders a *data-cmp-clickable*



```
<button data-sly-use.button="com.adobe.cq.wcm.core.components.models.Button"
        data-sly-use.component="com.adobe.cq.wcm.core.components.models.Component"
        data-sly-use.iconTemplate="icon.html"
        data-sly-element="${button.link ? 'a' : 'button'}"
        type="${button.link ? '' : 'button'}"
        id="${component.id}"
        class="cmp-button"
        href="${button.link}"
        aria-label="${button.accessibilityLabel}"
        data-cmp-clickable="${button.data ? true : false}"
        data-cmp-data-layer="${button.data.json}">
    <sly data-sly-call="${iconTemplate.icon @ icon=button.icon}"></sly>
    <span data-sly-test="${button.text}" class="cmp-button__text">${button.text}</span>
</button>
```

Generating component data

- The javascript utility will push component data

```
var components = document.querySelectorAll("[data-cmp-data-layer]");

components.forEach(function(component) {
  addComponentToDataLayer(component);
});

dataLayer.push({
  event: "cmp:loaded"
});
```

- The javascript utility registers click event handler

```
var clickableElements = document.querySelectorAll("[data-cmp-clickable]");

clickableElements.forEach(function(element) {
  attachClickEventListener(element);
});
```

Adobe

Enabling Data Layer integration

The Adobe Client Data Layer is disabled by default. To enable it:

- Create a sling config under `/conf/<my-site>/sling:configs/com.adobe.cq.wcm.core.components.internal.DataLayerConfig`.
- Add the enabled *boolean* property and set it to *true*.
- Add a *sling:configRef* property to the *jcr:content* node of your site.

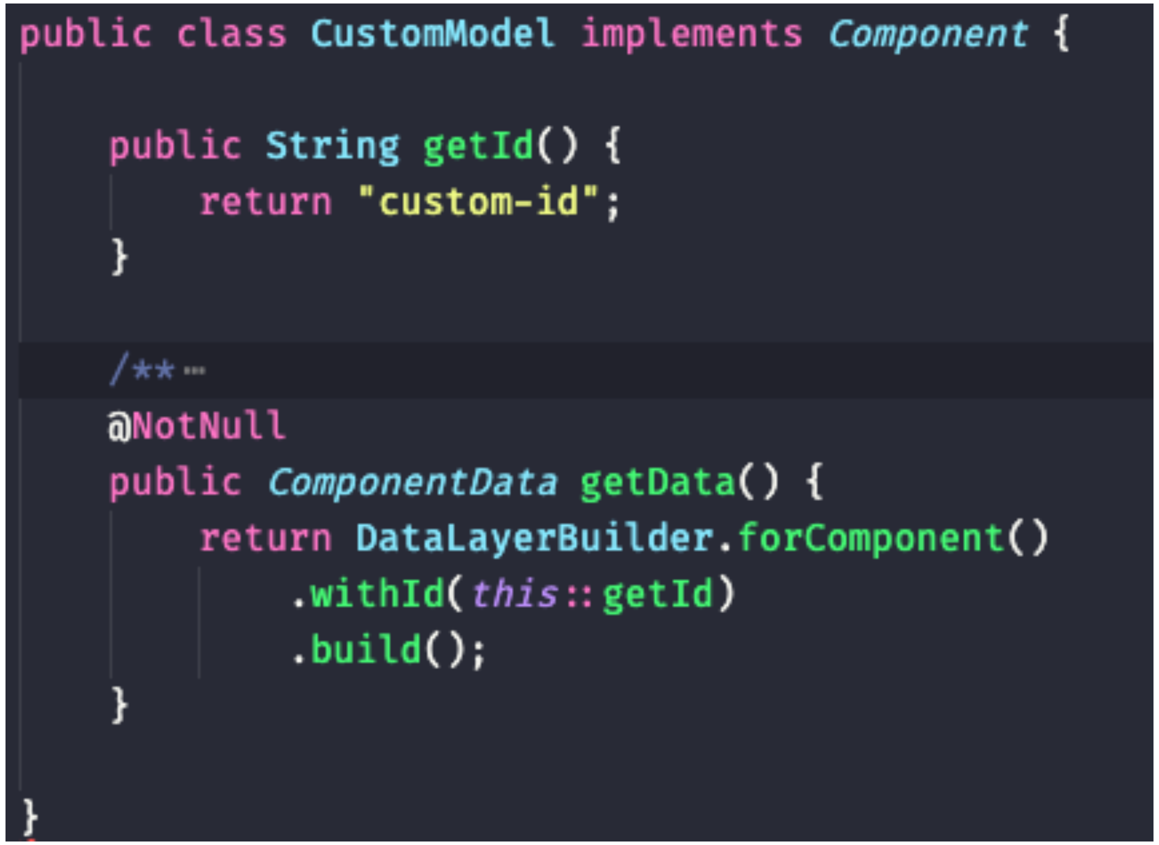
Integration with Custom Components

Using existing integration

To automatically add a custom component to the data layer:

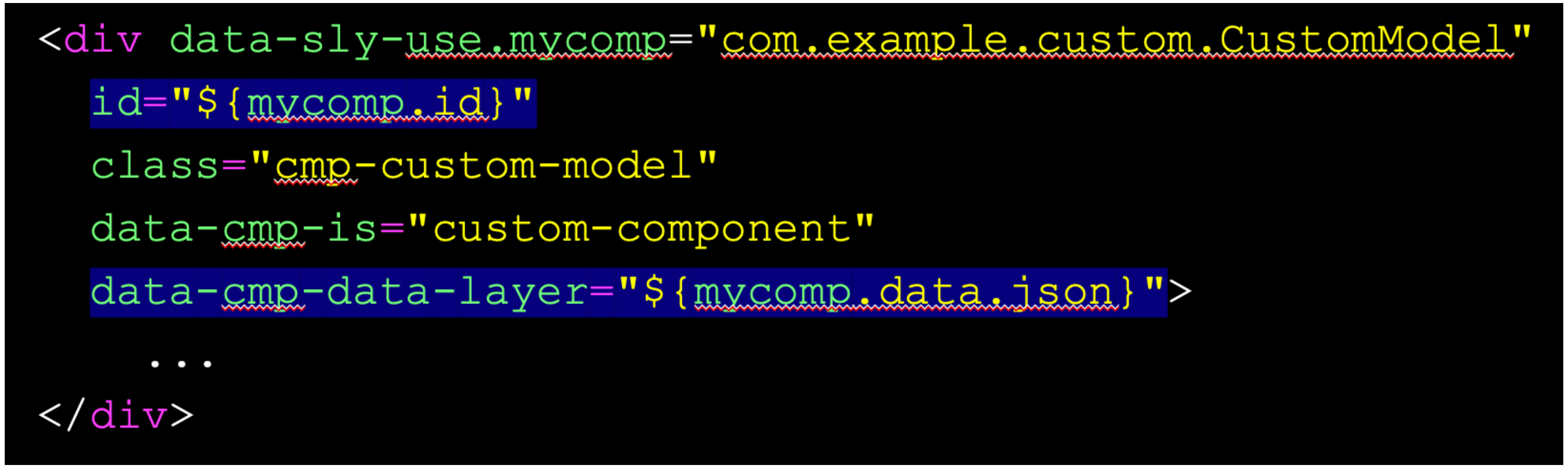
- Define the properties of the custom component model that needs to be tracked.
- Add a component *ID* to the the custom component HTL.
- Add the *data-cmp-data-layer* attribute to the custom component HTL.
- Generate the JSON data structure in your model using the *DataLayerBuilder* utility

Custom component model



```java
public class CustomModel implements Component {

    public String getId() {
        return "custom-id";
    }

    /**…
    @NotNull
    public ComponentData getData() {
        return DataLayerBuilder.forComponent()
            .withId(this::getId)
            .build();
    }

}
```

Custom HTL data generation



```
<div data-sly-use.mycomp="com.example.custom.CustomModel"
   id="${mycomp.id}"
   class="cmp-custom-model"
   data-cmp-is="custom-component"
   data-cmp-data-layer="${mycomp.data.json}">
     ...
</div>
```

Adobe

Data Layer Events for Custom Components

To leverage existing integration for events:

- In the custom component HTL add the *data-cmp-clickable* attribute to the element to be tracked.
- Make sure the component HTL has an *ID* on the top DOM element.

Custom events (*show, hide, etc*) should be triggered by component *clientlibs*

Integration with Adobe Launch

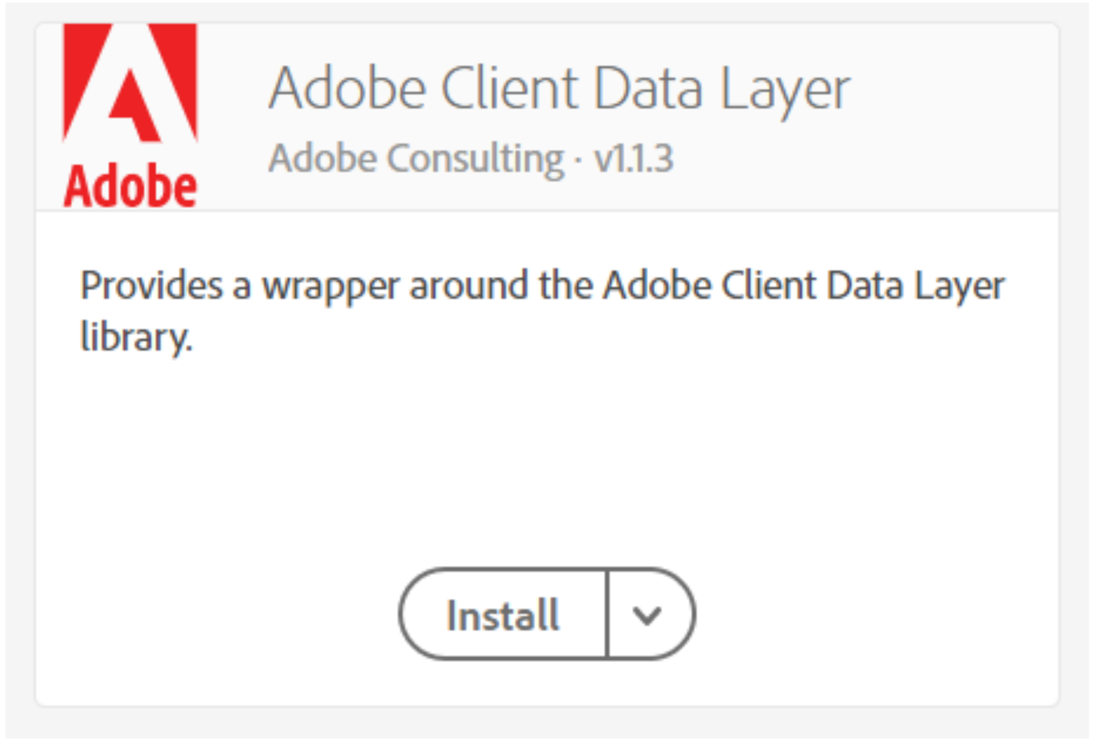
ACDL Launch Extension – Features

- Loading & injecting of the ACDL JavaScript library.
- Listening to Data Layer push events.
- Retrieving the Computed State of the Data Layer.
- Retrieving a specific element from the Data Layer by path.
- Pushing data & events into the adobeDataLayer object.
- Renaming and resetting / compacting the adobeDataLayer object.

Live Demo

ACDL Launch Extension – Installation

To install the Adobe Client Data Layer Extension, navigate to the Extension catalogue in Launch Extension and select the Adobe Client Data Layer.


Adobe Client Data Layer
Adobe Consulting · v1.1.3

Provides a wrapper around the Adobe Client Data Layer library.

Resources

- Adobe Client Data Layer:
  https://github.com/adobe/adobe-client-data-layer
- Integration with the Core Components:
  https://github.com/adobe/aem-core-wcm-components/blob/master/DATA_LAYER_INTEGRATION.md
- ACDL Launch Extension:
  https://exchange.adobe.com/experiencecloud.details.104231.adobe-client-data-layer.html
- Collect page data with Adobe Analytics:
  https://docs.adobe.com/content/help/en/experience-manager-learn/sites/integrations/analytics/collect-data-analytics.html

Q & A

Thank You!

Adobe