# Other Integration Methods

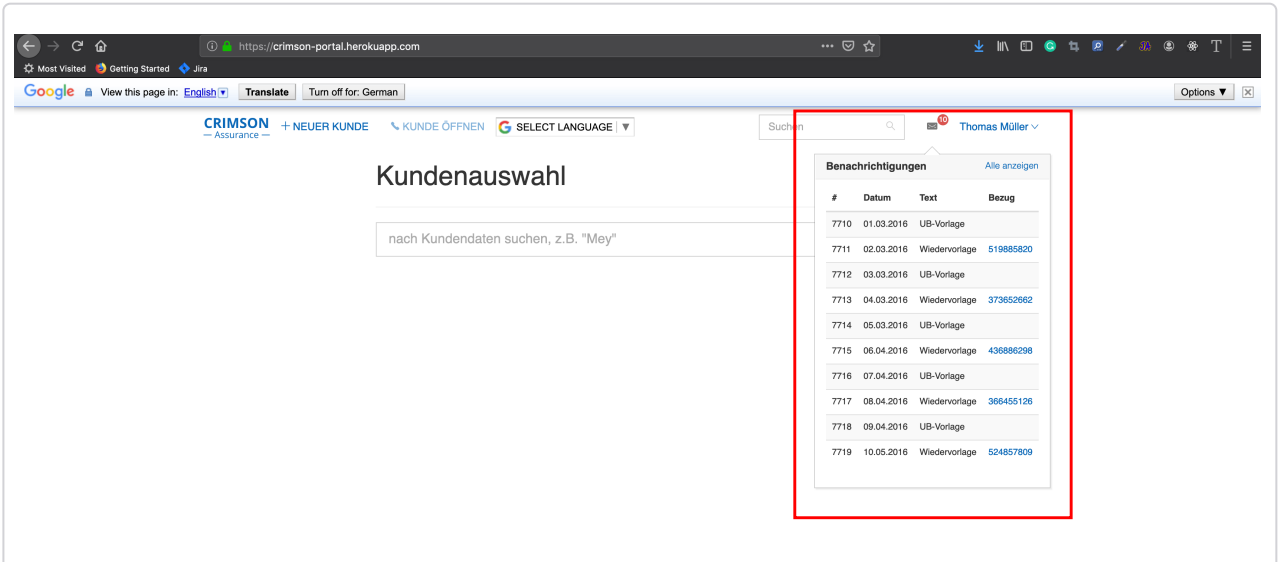
In this lesson, we'll continue looking at integration methods

**We'll cover the following**

- Integration with JavaScript
- Presentation logic in the postbox
- Assets with integrated HTML
- Resilience
- With and without JavaScript

# Integration with JavaScript #

In the example, the integration of the frontend is practically always done via links. However, the *postbox* displays an overview of the current messages in the *main* application.



The postbox application uses transclusion for integration with the main application

For this, a simple link is not enough. Still, this integration is also a link. A look into the HTML code shows:

```
<a href="https:&#x2F;&#x2F;crimson-postbox.herokuapp.com/m50000/messages"
  class="preview" data-preview="enabled"
  data-preview-title="Notifications"
  data-preview-selector="table.messages-overview"
  data-preview-error-msg="Postbox unreachable!"
  data-preview-count="tbody>tr" data-preview-window>
```

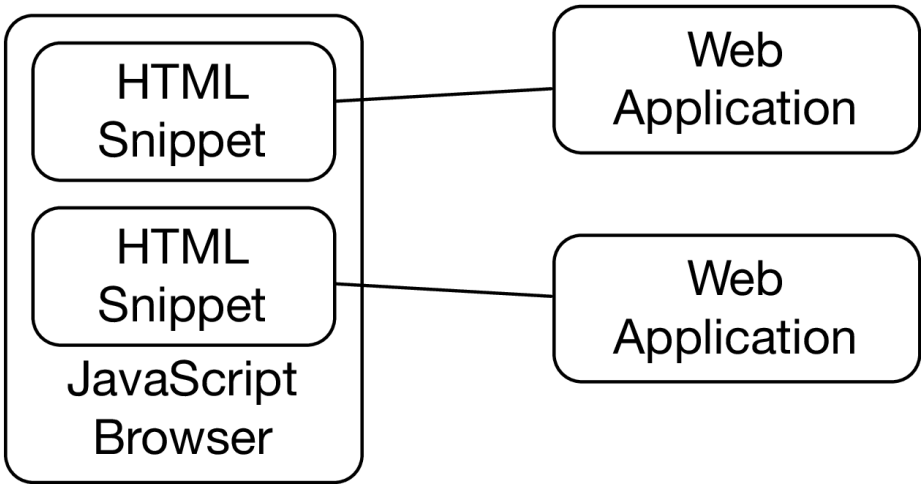
Actually, when running the application, the HTML code contains German expressions. They can be translated by Google Translate. However, the HTML code remains unchanged and still includes German expressions. For convenience, the English expressions are shown in the HTML code in the listing here.

The link contains **additional attributes**. They ensure that the information of the *postbox* is displayed in the current web page and define how exactly this happens. This information is interpreted by less than 60 lines of JavaScript code from the asset project (see https://github.com/ewolff/crimson-styleguide/tree/master/components/preview (https://github.com/ewolff/crimson-styleguide/tree/master/components/preview)). The code uses jQuery, so every application in the system now needs to use a compatible version of jQuery when using this code from the asset project.

However, this is not mandatory. Alternatively, any microservice that integrates the *postbox* with such a link can write its own code to interpret the link. After all, every microservice writes its own code to read data from JSON that other microservices provide. Therefore, code that integrates other projects' HTML should be fine, too.

This type of integration is called *transclusion* because it includes HTML

from more than one microservice in a web page. In this example, the transclusion is implemented with JavaScript code that integrates the different backends (see the drawing below). The JavaScript code runs in the browser, loads HTML fragments of the other web applications, and displays them in the current web page.




Integration with JavaScript

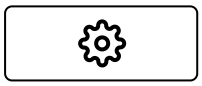
# Presentation logic in the postbox #

With transclusion, *the postbox* retains control over how messages are displayed, even if the messages are shown as previews in another service. This leads to a clean architecture.

The code for displaying the postbox is located in the postbox service, even if the postbox is displayed in another service. This shows how frontend integration can contribute to an elegant solution and architecture.

The approach of dynamically including content from other URLs is not only used for the *postbox*. For each customer, there is also an overview of the offers, applications, claims, and contracts. The links for this are located below the postbox icon and are repeated further down in the

inventory.



Just as with the postbox, this information is also referenced with links whose contents display on the web page. In this case, the URLs are in the same microservice but still provide a better modularization. It also shows that the code solves a general technical problem and is reusable beyond the integration between microservices.

# Assets with integrated HTML #

Transclusion embeds HTML fragments into other web pages. The HTML can require assets such as CSS or JavaScript. There are different approaches for ensuring that these assets are present.

- The HTML can be designed to not require any assets at all. Thus, no assets have to be shared between the microservices.
- The HTML uses only the assets from the shared asset library – in the example, `crimson-styleguide`. No special measures are necessary because the assets are present in all microservices. microservices.
- The HTML can also bring along or link to its own assets. However, in this case you have to be careful not to load the assets more than once if several contents are transcluded in one web page.

Till Schulte-Coerne has written a blog post (https://www.innoq.com/en/blog/transclusion/) about this topic.

# Resilience #

The application achieves a high degree of resilience and reliability through the consistent use of links. If one microservice fails, the other microservices continue to work; they can still display the links.

The JavaScript code contacts the *postbox* to transclude an overview of the messages on the web page. If this doesn't work, the code displays an exclamation point, but the application still works.

Thanks to JavaScript, loading the transcluded HTML is carried out in the background, so that the failure of one system does not affect the transclusion of the other content and high performance is achieved.

# With and without JavaScript #

The user can also use the applications if JavaScript is disabled. In this case, for example, the automatic completion of customer names does not work, nor does displaying the overview of messages in the *postbox*.

Nevertheless, the application remains usable. The start page is rendered completely as HTML on the server and does not require client-side templates. The *postbox* simply offers a link instead of the displayed overview. If the user clicks on this link, they get to the *postbox*.

1 Which of the following counts as transclusion?

A) A frontend reads data from two backends and creates a monolithic UI.

B) Accessing data from an app `x` in another app `y` via a link that takes you to `x`.

C) Accessing data from app `x` in app `y` via a link that displays data from `x` in `y`.

Submit Answer

Question 1 of 3
0 attempted

Reset Quiz

In the next lesson, we'll look at an app.

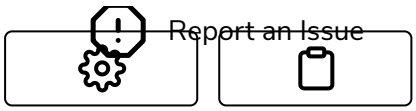
← Back

Integration in Assurance App

Next →

Example

Mark as Completed


Report an Issue