Introduction to the React Connector

The Frosmo React Connector is a JavaScript module that allows you to seamlessly integrate your React application with the Frosmo Platform. The React Connector works through the `FrosmoPlacement` component, which is a wrapper for React components whose display rules and content you manage with the Frosmo Platform. Since `FrosmoPlacement` is part of the React DOM, this integration takes place safely inside the React rendering pipeline.

Each `FrosmoPlacement` component is linked to a placement and a modification defined in the Frosmo Platform, so the display rules you define for the placement and the modification will apply to anything inside the component. In addition, the child components will also have access to the modification content and event tracking functionality.

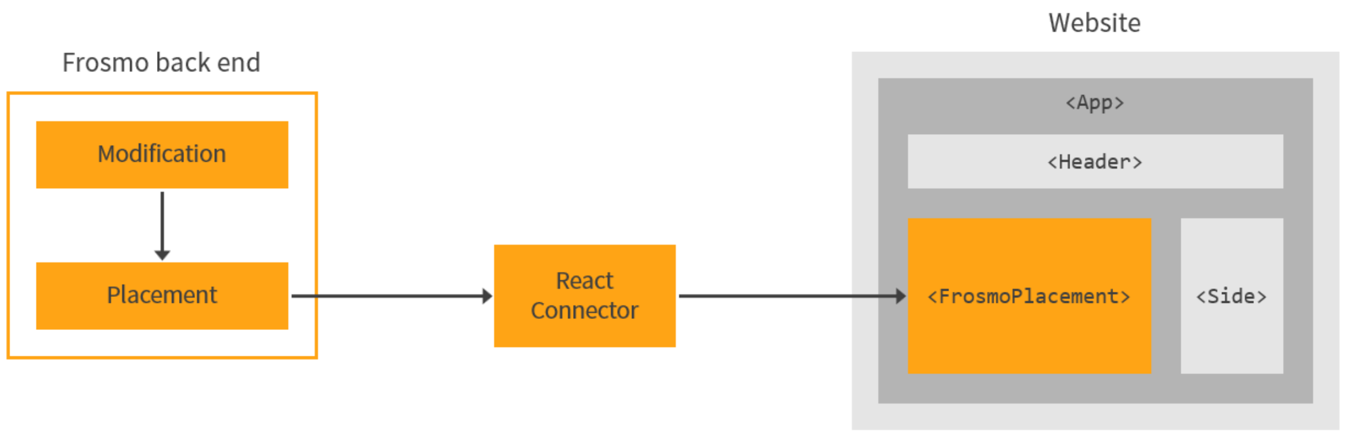


Figure: Connecting the Frosmo Platform to a React application

For more information, see:

- Why use the React Connector?
- How the React Connector works
- How FrosmoPlacement interacts with the Frosmo Platform

Why use the React Connector?

If you're running a React application with the Frosmo Platform, there are several benefits to using the React Connector:

Develop inside the React rendering pipeline

You have two basic options for using the Frosmo Platform to modify a web page built wholly or in part with React:

- **Modify the DOM after React has updated it.** This is the traditional approach, where you set a placement to target an HTML element through a CSS or jQuery selector. You work outside the React code, and your modifications therefore take effect outside the React rendering pipeline.
- **Modify the React DOM.** This is the React Connector approach, where you associate a placement with a React component rather than an HTML element. You effectively work the placements into your React code, and you therefore make modifications inside the React rendering pipeline.

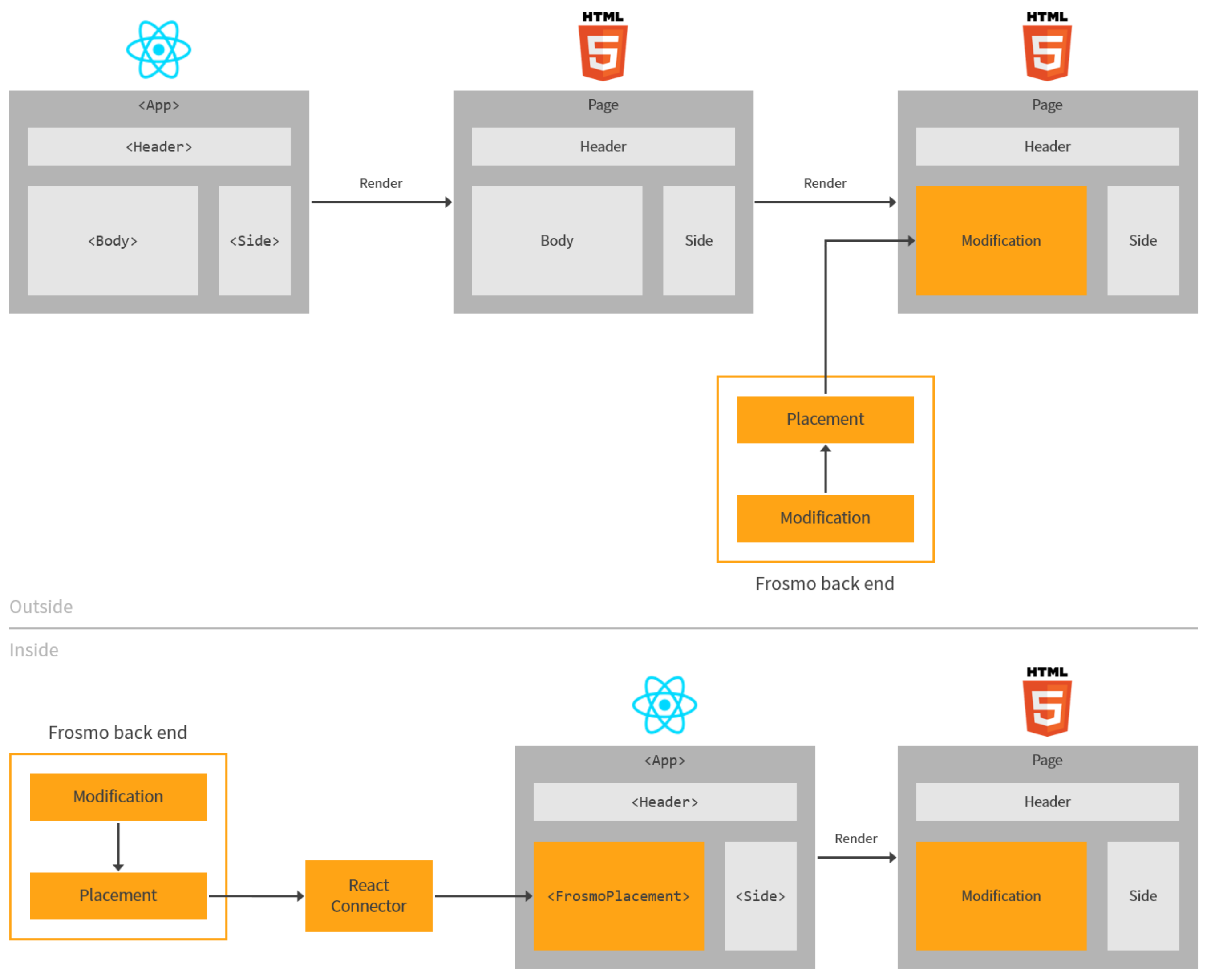


Figure: Developing outside vs. inside the React rendering pipeline

While the traditional approach can work just fine with React applications, it can also result in problems, such a flickering and missing elements. The React Connector approach, however, does not suffer from these problems, since you're modifying the web page before it gets rendered by React.

In short, if you're using the Frosmo Platform with a React application, and if you want to guarantee an optimal user experience for your visitors, use the React Connector when developing your application.

Keep your UI code in a single codebase

The `FrosmoPlacement` component allows you to create your Frosmo-managed web page elements directly in your React code. This means that you can define the structure and presentation of those elements in React, rather than in Frosmo modifications, keeping all your UI code in one codebase. You can use modification content purely for delivering element content, such as text and graphics. You can even omit modification content entirely and just use the modification for its display rules.

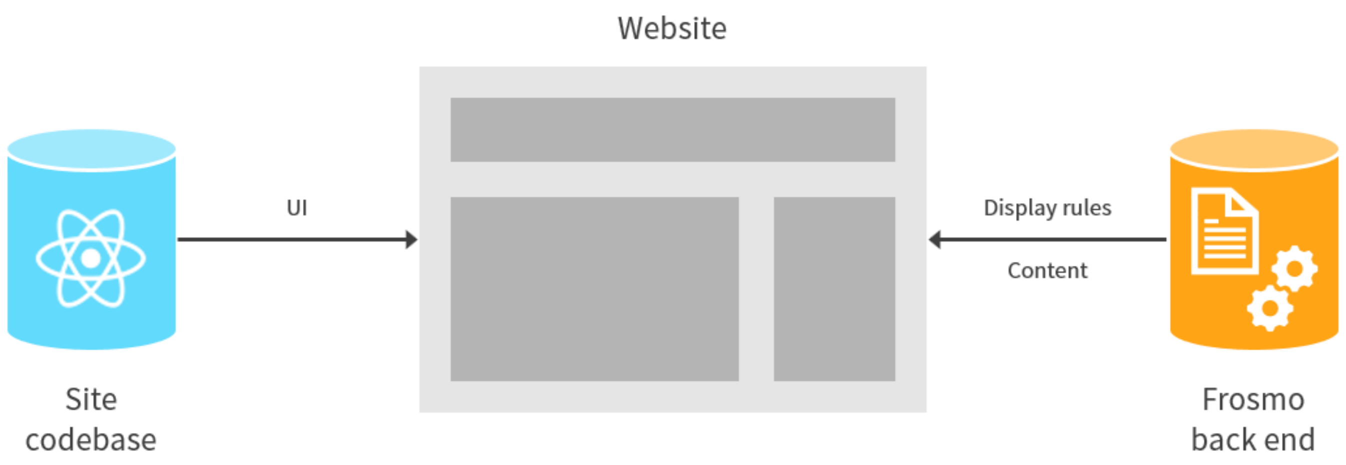


Figure: Developing in a single codebase

Separate content, presentation, and display rules

When you wrap a UI element inside a `FrosmoPlacement` component, you define the element's display rules in a Frosmo placement and modification, and its structure and presentation in your React code. The content of the element, in turn, can live in the modification, in your React code, or in a separate content management system (CMS). The degree of separation is entirely up to you.

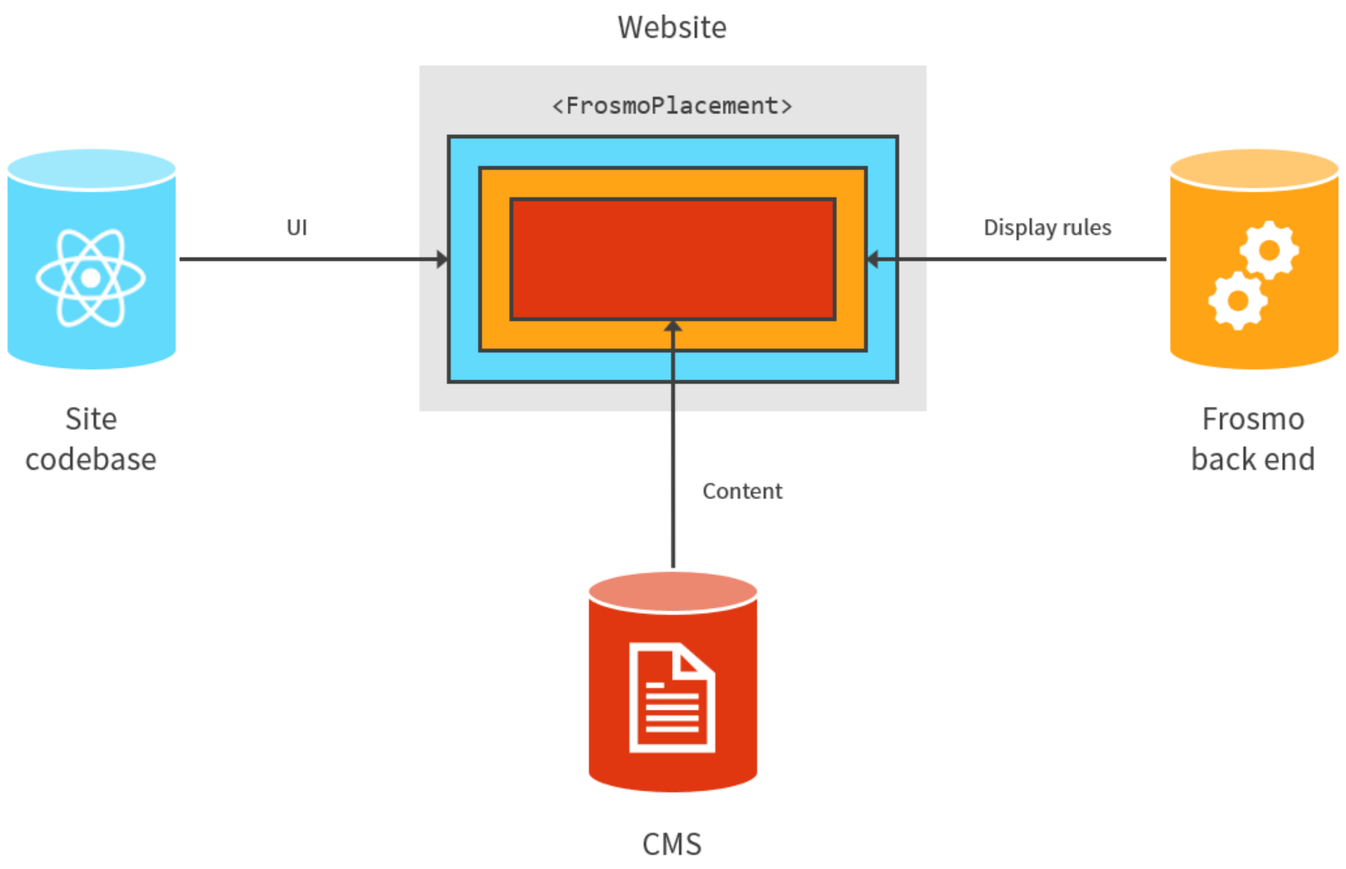
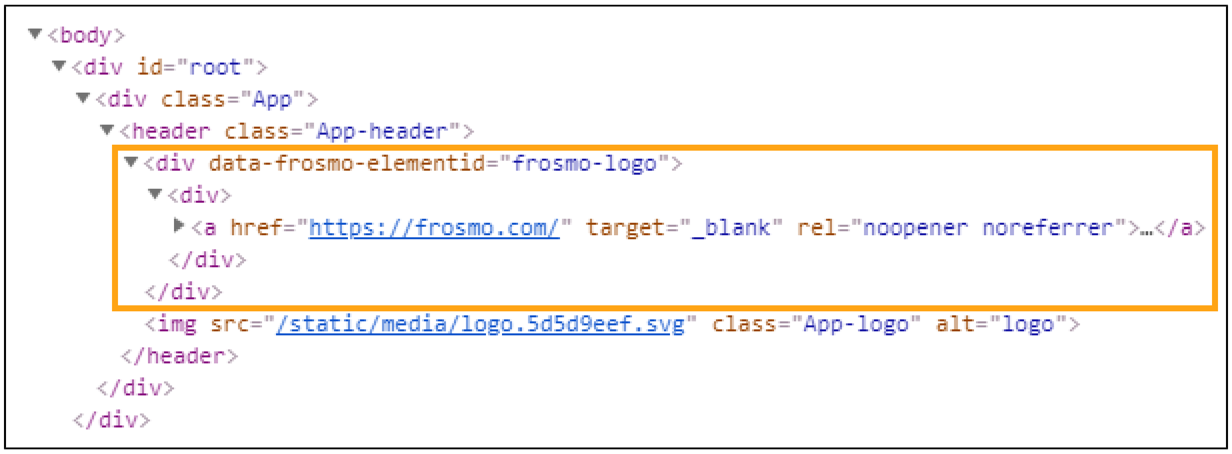


Figure: Separation of content, presentation, and display rules

How the React Connector works

The React Connector module provides the functionality necessary to integrate your React application with the Frosmo Platform. The main feature of the module is the `FrosmoPlacement` component, a reusable piece of React code that implements the connection between your application and the platform. In essence, the component takes on the role of a placement inside a React application: the component maps to a specific placement, which in turn links the component to the modification assigned to that placement. The component has the `frosmoModificationContext` prop, which provides access to the content and other features of the associated modification.

In a regular web application, a placement directly translates into a `<div>` element populated with the content of the associated modification. In a React application, however, a placement is more like a funnel that allows the application to populate its own rendered element, built using a `FrosmoPlacement` component, with the correct modification content. Like a placement, though, the component gets rendered as a `<div>` element on the page.



```
▼<body>
  ▼<div id="root">
    ▼<div class="App">
      ▼<header class="App-header">
        ▼<div data-frosmo-elementid="frosmo-logo">
          ▼<div>
            ▶<a href="https://frosmo.com/" target="_blank" rel="noopener noreferrer">…</a>
            </div>
          </div>
          <img src="/static/media/logo.5d5d9eef.svg" class="App-logo" alt="logo">
        </header>
      </div>
    </div>
```

Figure: FrosmoPlacement rendered in the page source code

The React Connector module additionally relies on the Frosmo SPA module, which manages the communication between the `FrosmoPlacement` components and the Frosmo Platform. When enabled, the Frosmo SPA module is included in the Frosmo custom script for your site.

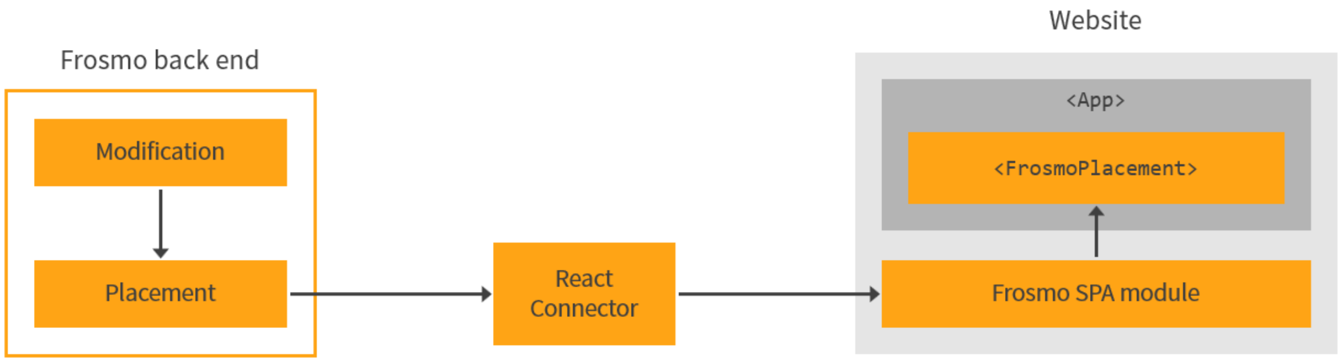


Figure: Communication between FrosmoPlacement and the Frosmo Platform

For more information about installing the React Connector module, see Installing the React Connector. To enable the Frosmo SPA module for your site, contact your Frosmo representative.

How FrosmoPlacement interacts with the Frosmo Platform

The `FrosmoPlacement` component interacts with the Frosmo Platform as follows:

When the platform finds and returns a modification

1. `FrosmoPlacement` is mounted.
2. `FrosmoPlacement` requests a modification for the placement by calling the `requestBySelector()` function of the Frosmo SPA module.
3. The Frosmo SPA module requests the modification from the Message API.
4. The Message API returns the correct modification for the placement based on the display rules defined for the placement and modification.
5. The Frosmo SPA module returns the modification as `Promise.<ModificationContext>`.
6. `FrosmoPlacement` updates its content based on the returned modification.

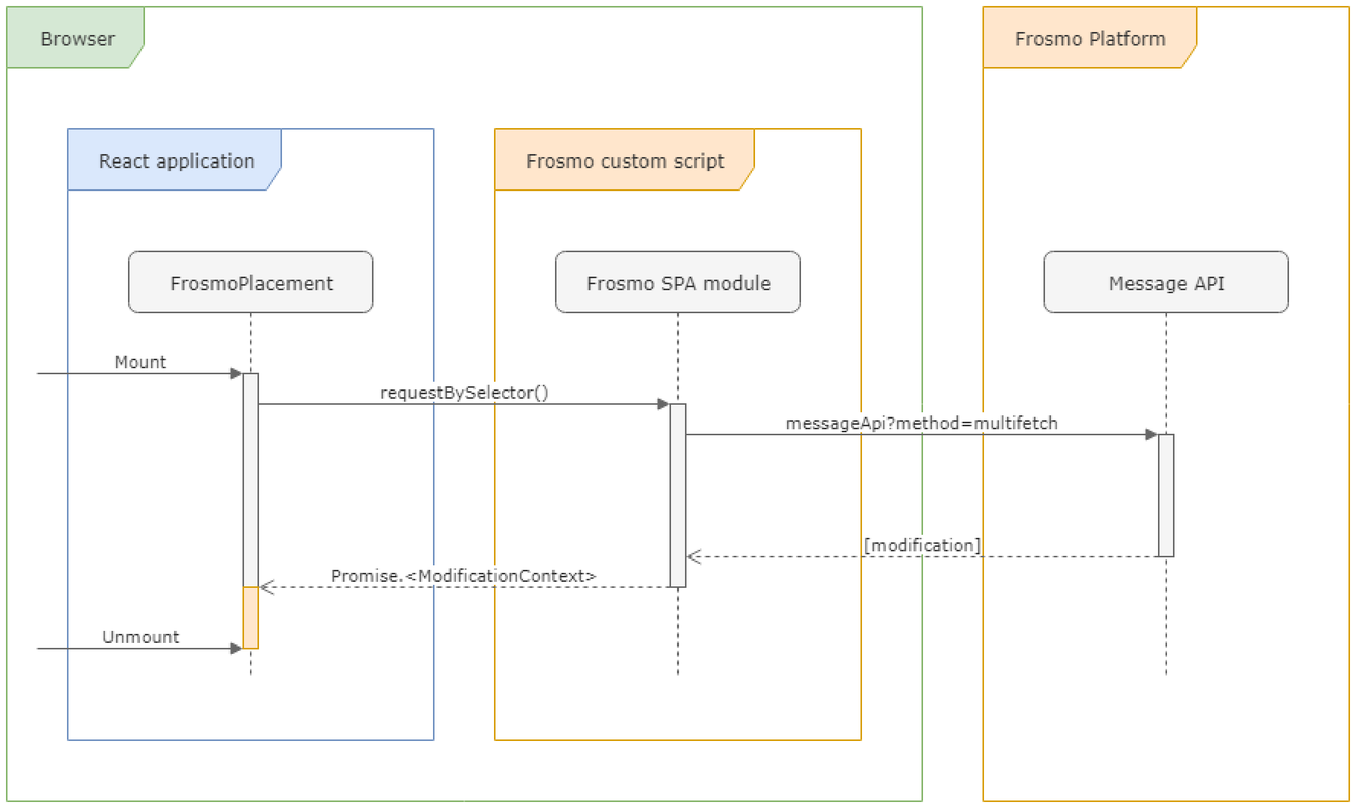


Figure: Interaction flow with the Frosmo Platform when the modification is found

When the platform cannot find a modification

1. `FrosmoPlacement` is mounted.
2. `FrosmoPlacement` requests a modification for the placement by calling the `requestBySelector()` function of the Frosmo SPA module.
3. The Frosmo SPA module requests the modification from the Message API.
4. The Message API returns an empty modification, meaning the API could not find a modification for the placement. Either the placement does not exist or the placement has no modifications assigned to it.
5. The Frosmo SPA module returns an error as `Promise.reject`.
6. `FrosmoPlacement` does not update its content.

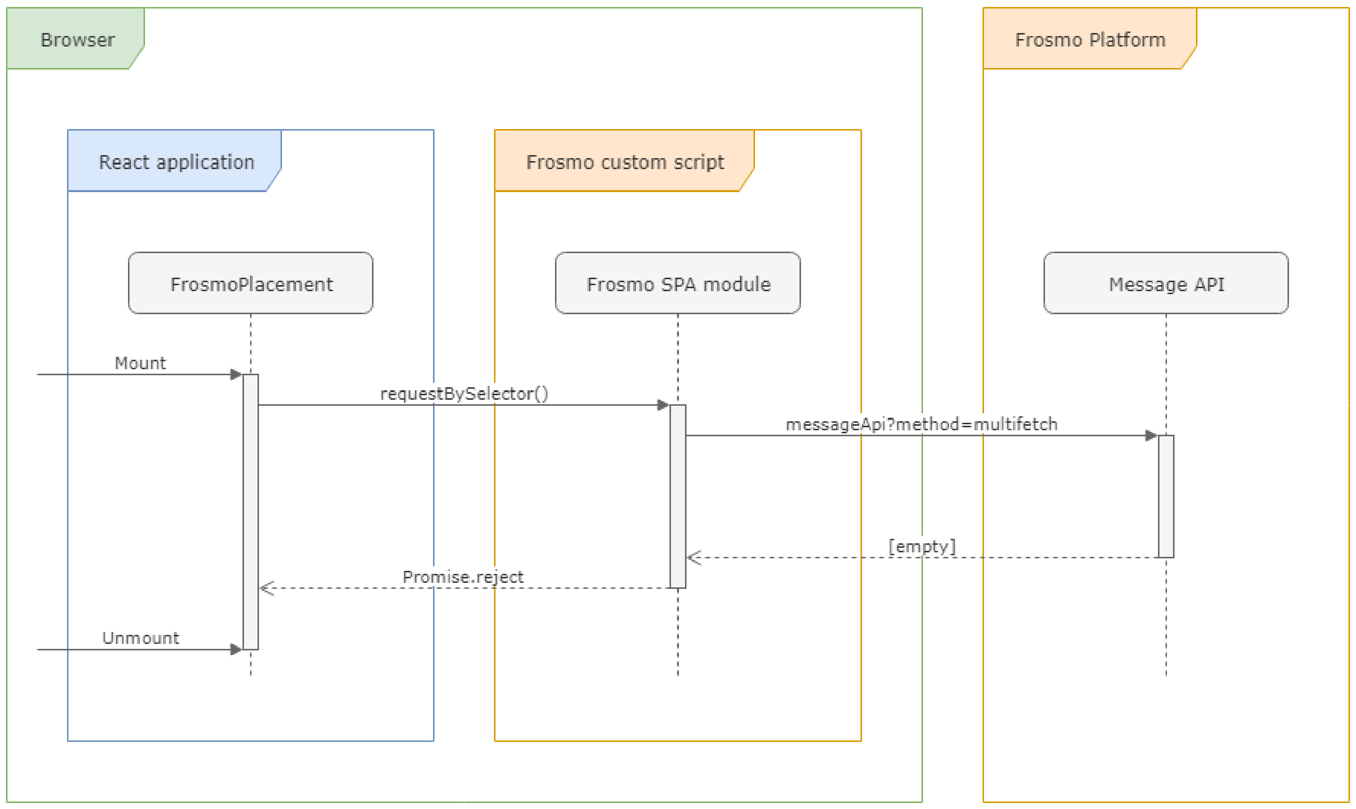


Figure: Interaction flow with the Frosmo Platform when the modification is not found

When the platform does not respond

1. `FrosmoPlacement` is mounted.
2. `FrosmoPlacement` requests a modification for the placement by calling the `requestBySelector()` function of the Frosmo SPA module.
3. The Frosmo SPA module requests the modification from the Message API.
4. The Message API does not respond.
5. The Frosmo SPA module timeouts and returns an error as `Promise.reject`.
6. `FrosmoPlacement` does not update its content.

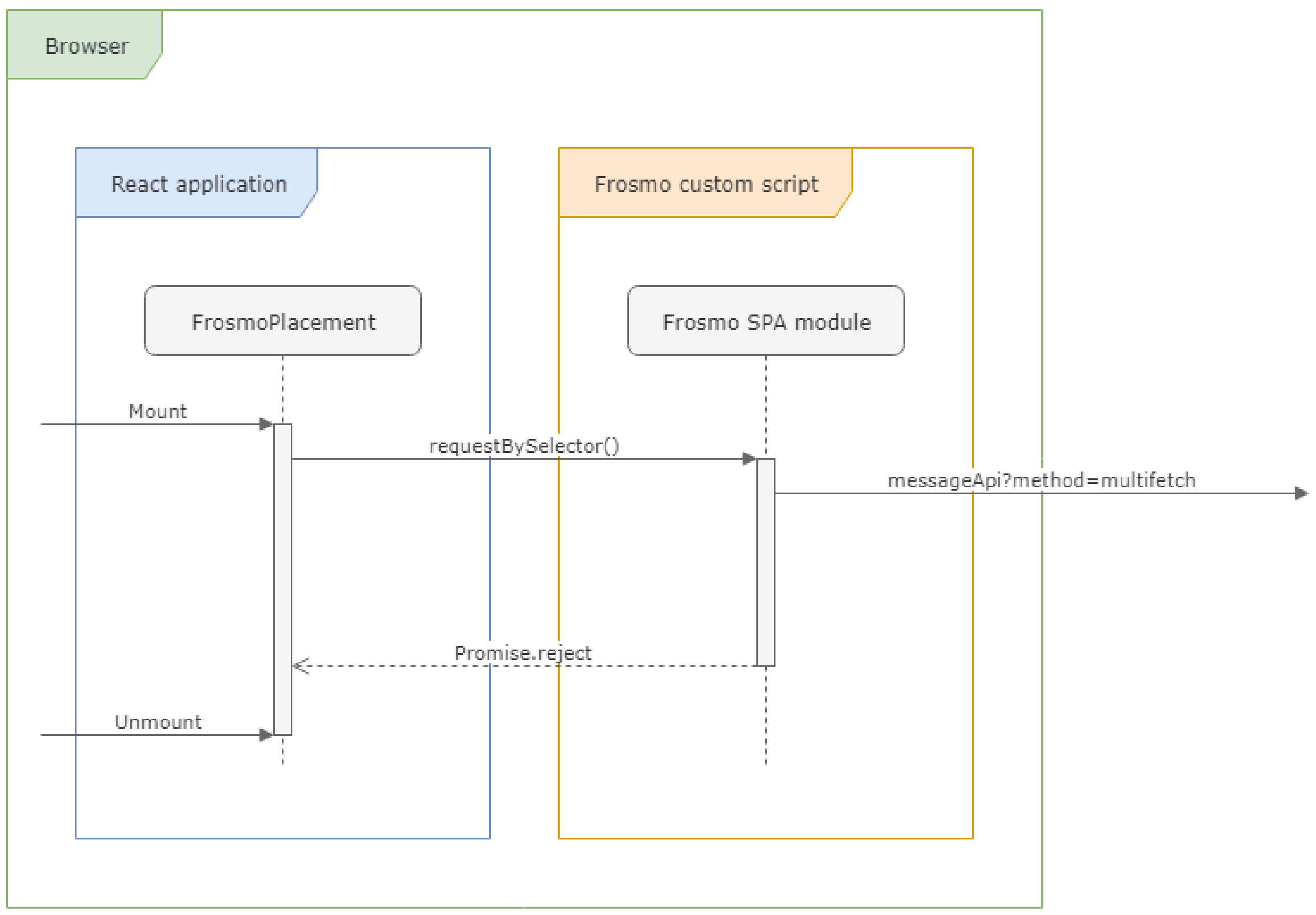


Figure: Interaction flow with the Frosmo Platform when the platform does not respond