Architecture Document

Framework Agnostic Component

Initial Draft, Feb 2018

1. Introduction

1.1 Purpose

Framework agnostic component helps to build reusable HTML elements and use them to build performant, portable and maintainable apps.

1.2 Scope

The current solution demonstrates how framework agnostic component works with pure Java Scripts as well as with frameworks like React & Vue.js with the same look and feel & functionalities.

1.3 Overview

Various IT groups within one IT organization developed similar UI capabilities around multiple JavaScript frame works (viz. Angular, React or Vue.js etc.). This leads to issues like duplication of work, non-portability, lack of reusability and cost/maintenance overhead in the long run.

Framework agnostic component would address the problem of portability and reusability with the same functionality for any frameworks. This will help IT groups to standardize the components and migrate to a single Web UI.

1.4 Out of Scope

The current solution should work seamlessly with all the frameworks like Angular, React & Vue.js etc. However, for demonstration purpose we have showed integration with pure Java Script, React & Vue.js. The integration with other frameworks is not included as part of the illustration.

2. Architecture

1. To create a framework agnostic component, that are truly reusable and interoperable with all the benefits of the React and Vue.js ecosystems, Custom Elements have been created.

   The below diagram (Fig-1) demonstrates the high-level architecture.

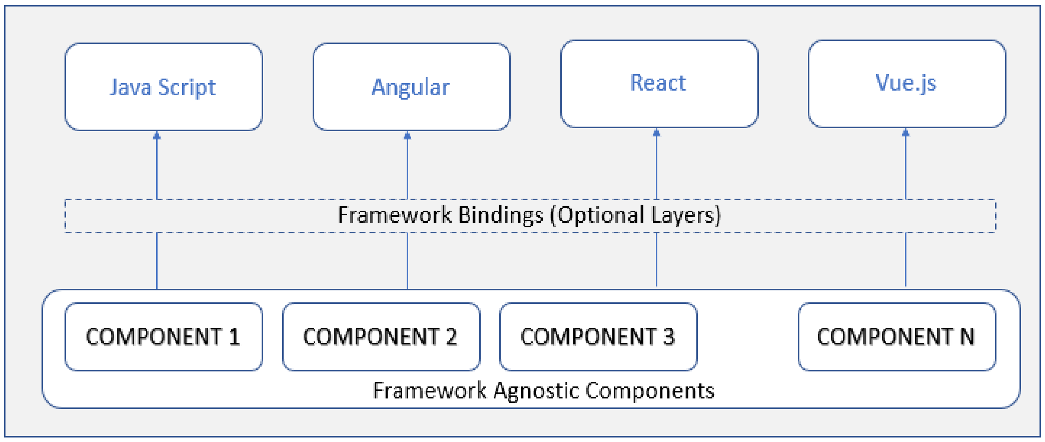

Fig-1: High-level architecture

2. The component comprises the CSS component (visual aspects) and a Web Component, based on the Lifecycle callbacks -

```
connectedCallback () {
// Called every time the element is inserted into the DOM
}

disconnectedCallback() {
// Called every time the element is removed from the DOM.
}

attributeChangedCallback(attrName, oldVal, newVal) {
// Called when an attribute was added, removed, or updated
}

adoptedCallback() {
        // Called if the element has been moved into a new document
}
```

3. The custom elements are written in pure JavaScripts and defined as a tag. That makes the component model without any framework lock-in. Hence, would increase the re-usability, consistency and reduce the manageability overhead.

3. Design

1. The Custom Component < **popup-info** /> (which shows up custom pop-up messages) defined inside the **PopUpInfo.js**. It is written is simple JavaScript extending HTMLElement and using connectedCallback() method. It has two portions – the styling/css part and the web part.
2. The component <popup-info /> is invoked by various callers like PopUp_Demo.html (Simple JavaScript), React_DEMO.html (based on React framework) and Vue_DEMO.html (based on Vue.js).

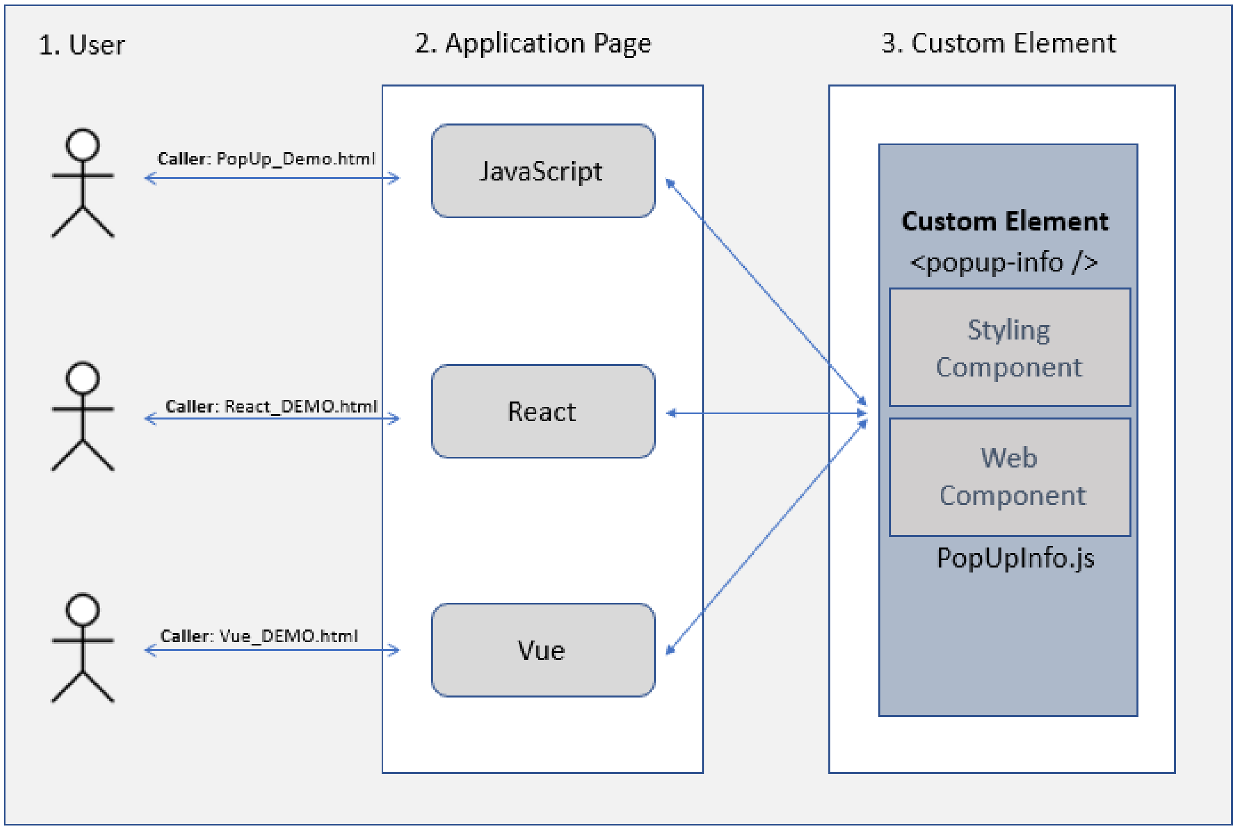

Fig-2: Sequence Diagram

3. The component acts as a custom tag and demonstrated exactly same behavior irrespective of the framework. Hence called as Framework Agnostic Component.

4. Conclusion

This document illustrates designing of a simple custom component which is framework agnostic.

It's an opportunity for large organization to consolidate their front-end UI into a pattern library (based on the framework agnostic components). This could be very useful as a company grows and splits into multiple teams.

Document Tracking

The following chart is used to log of all changes made to this document.

Version	Date of edit/change	Who made the edit/change	Description of edit/change
0.1	02/28/2018	Amitabha Sen Niyogi	Initial Draft