Web Embeds

Overview

If you want to embed (third-party) web content in an Electron `BrowserWindow`, there are three options available to you: `<iframe>` tags, `<webview>` tags, and `BrowserViews`. Each one offers slightly different functionality and is useful in different situations. To help you choose between these, this guide explains the differences and capabilities of each option.

Iframes

Iframes in Electron behave like iframes in regular browsers. An `<iframe>` element in your page can show external web pages, provided that their Content Security Policy allows it. To limit the number of capabilities of a site in an `<iframe>` tag, it is recommended to use the `sandbox` attribute and only allow the capabilities you want to support.

WebViews

> *Important Note: we do not recommend you to use WebViews, as this tag undergoes dramatic architectural changes that may affect stability of your application. Consider switching to alternatives, like `iframe` and Electron's `BrowserView`, or an architecture that avoids embedded content by design.*

WebViews are based on Chromium's WebViews and are not explicitly supported by Electron. We do not guarantee that the WebView API will remain available in future versions of Electron. To use `<webview>` tags, you will need to set `webviewTag` to `true` in the `webPreferences` of your `BrowserWindow`.

WebView is a custom element (`<webview>`) that will only work inside Electron. They are implemented as an "out-of-process iframe". This means that all communication with the `<webview>` is done asynchronously using IPC. The `<webview>` element has many custom methods and events, similar to `webContents`, that provide you with greater control over the content.

Compared to an `<iframe>`, `<webview>` tends to be slightly slower but offers much greater control in loading and communicating with the third-party content and handling various events.

BrowserViews

BrowserViews are not a part of the DOM - instead, they are created in and controlled by your Main process. They are simply another layer of web content on top of your existing window. This means that they are completely separate from your own `BrowserWindow` content and their position is not controlled by the DOM or CSS. Instead, it is controlled by setting the bounds in the Main process.

`BrowserViews` offer the greatest control over their contents, since they implement the `webContents` similarly to how the `BrowserWindow` does it. However, as `BrowserViews` are not a part of your DOM, but are rather overlaid on top of them, you will have to manage their position manually.