**Scenario:**

While working on a web application project, I got the requirement to display 3d charts in the browser. I was using Angularjs as my frontend so I searched for the charting tools that are AngularJs compatible, free, open-source and 3d. After exhaustive sorting and listing of various charting packages, I found ZingCharts, easier and suitable to implement. Zingcharts offered free support except for a small watermark. (I didn’t mind the watermark). ZingChart took a JSON as input to render a 3d chart as SVG output. It looked great on the browser.

Zingchart is a really useful library for Angularjs. It even offers an inbuilt functionality to download individual (or bulk charts using chartset) as pdf. However, when it comes to render and create a custom pdf (like adding data grid, other text) things become little messy. Yes, in addition to render charts in the browser, I was also having a requirement to show the same charts in custom pdf which will display chart as well as related data in grid.

I was using ItextSharp for rendering HTML pages to pdf. But this time the case was slightly different. The HTML tag “Zingchart” needed a script to render it to proper chart and that in no case was possible using itextsharp. However, I had an alternative to use C#’s inbuilt Data Visualization chart, recreate methods for the same and render the charts again completely. This alternative would not only make the code redundant, it would change the look and feel of browser’s chart with the one that in pdf. This seemed not an optimal solution.

This led to another set of research guided intensely by my CTO, for such tools which takes the rendered HTML of the browser as input and get me a pdf in output. The search ended when CTO suggested to use Node-Horseman repository ([link](https://github.com/johntitus/node-horseman)).

Node Horseman is a library that allows running phantomjs from the node. It has rich APIs that does several web scraping work including the one of my requirement.

I won’t dive deeper on how Node horseman works as the link is a self-explanatory resource.